UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIO SUPERIORES ARAGÓN

SISTEMA PARA EL MONITOREO DE VARIABLES CLIMÁTICAS AL INTERIOR DE UN INVERNADERO

TESIS

Que para obtener el título de:

INGENIERO ELÉCTRICO ELECTRÓNICO

Presenta:

JOSÉ SALVADOR RESENDIZ RODRIGUEZ

Director de tesis:

DR. ISMAEL DÍAZ RANGEL

Ciudad **Nezahualcóyotl, Estado de México, 2022**

Agradecimientos

Principalmente a mis padres: Consuelo Rodríguez Cisneros y Salvador Resendiz Corona, por brindarme todo el apoyo económico y moral para poder seguir adelante a pesar de todas las dificultades, también por haberme enseñado la disciplina necesaria para poder llegar hasta este punto.

Agradezco también a todos mis profesores durante la carrera por haberme brindado todo su conocimiento, especialmente al Dr. Ismael Díaz Rangel, por haberme brindado la ayuda en mi desarrollo académico, personal y por todo el asesoramiento necesario para solucionar todos desafíos a lo largo del desarrollo de este proyecto, así como al Mtro. Alejandro Andrés Serapio Carmona y al Ing. Juan Manuel Hernández Contreras, por todas las asesorías y consejos.

Ofrezco una especial mención y agradecimiento a todos los amigos que hice durante mi estadía en la FES Aragón, por todo el asesoramiento y apoyo que compartimos mutuamente.

Y, por último, agradezco a la máxima casa de estudios Universidad Nacional Autónoma de México, por haberme proporcionado la oportunidad y todos los medios necesarios para mi desarrollo profesional y académico.

Contenido

1. Introducción

La historia de la civilización humana está directamente relacionada con el desarrollo de la agricultura, las primeras civilizaciones que dejaron de ser nómadas para convertirse en sedentarias tuvieron cercanía con grandes ríos y lagos, lo cual marco el paso de la recolección de frutos y semillas a las prácticas agrícolas. Civilizaciones como las mesoamericanas son ejemplos del esplendor cultural, social y económico de grupos humanos, que se debió en gran medida a su éxito en la producción y comercialización de alimentos [1].

En la actualidad, a la par que se desarrollan y crecen las civilizaciones, es necesario mejorar las herramientas y técnicas de cultivo, para poder optimizar la producción y cubrir la demanda, principalmente para el consumo humano.

Hoy en día, existen alternativas para mejorar la producción de alimentos tales como los invernaderos y los cultivos transgénicos, cada una de estas opciones tiene ventajas y desventajas en su implementación y uso.

Los alimentos transgénicos son organismos que poseen en su composición uno o varios genes diferentes de los que se les atribuyen en un principio. Mediante técnicas de biotecnología, se pueden utilizar genes extraídos de seres vivos, modificados en laboratorios y reintroducidos en el mismo u otro organismo. Técnicamente se conocen como Organismos Modificados Genéticamente (OMG) y su objetivo es dotar a estos organismos de cualidades especiales de las que carecerían. De este modo, las plantas transgénicas pueden sobrevivir a plagas, soportar mejor las sequías, o resistir el efecto de algunos herbicidas.

Sus detractores, apuntan a que el uso de estos productos se ha generalizado en muy poco tiempo sin que se pueda comprobar si los alimentos transgénicos tienen consecuencias a largo plazo. Juan Felipe Carrasco, ingeniero agrónomo que encabezó en 2010 una campaña de Greenpeace contra los transgénicos, es uno de los muchos opositores a este tipo de alimentos debido al impacto medioambiental y la pérdida de biodiversidad que suponen [2].

Por otra parte, una de las herramientas ampliamente utilizada en el sector agropecuario, son los invernaderos, un invernadero es una estructura metálica o de plástico cubierta por materiales translúcidos para conseguir la máxima luminosidad en el interior. Dentro del invernadero se obtienen condiciones artificiales (microclima) que genera en las plantas una

mayor productividad con un mínimo costo y en menos tiempo. Resguarda a las plantas o cultivos que están en su interior y los protege de daños ambientales como heladas, fuertes vientos, granizo y plagas de insectos. Por tanto, en un invernadero, se puede cultivar en cualquier época del año y de esta forma ser más productivos [3].

Una de las desventajas de un invernadero, es el costo inicial de su implementación; sin embargo, no requieren un estudio a mediano, corto y largo plazo de impacto en la salud de las personas, debido a que no se altera de ninguna manera las propiedades naturales del cultivo, para optimizar los cultivos. Se han desarrollado propuestas tecnológicas que consisten en el monitoreo y automatización de algunos procesos como el control de variables climáticas; sin embargo, el costo de su implementación es elevado y puede no resultar rentable para invernaderos pequeños.

Este trabajo consiste en el desarrollo de una propuesta para el monitoreo de algunas variables importantes para cultivo en invernaderos, y que tiene como propósito tener un costo de implementación que pueda estar al alcance de agricultores con invernaderos pequeños. El sistema propuesto, permitirá al agricultor conocer el estado de la humedad en sustratos, humedad, temperatura y nivel de luz en el ambiente, proporcionándole información que facilite la corrección de las variables, manteniéndolas en un rango de valores óptimo, de esta forma se podrá mejorar la producción.

1.1 Objetivo general

Desarrollar un sistema que realice el monitoreo de la humedad, temperatura e iluminación de un invernadero.

1.2 Objetivos particulares

- Diseñar módulos de adquisición para la medición de humedad en sustratos, humedad, temperatura e iluminación en el ambiente.
- Realizar el diseño de la etapa de adquisición de señales que permita la expansión de más variables a monitorear.
- Desarrollar una interfaz gráfica en una computadora personal, que reciba la información de los módulos y que posibilite observar las mediciones.
- Implementar una interfaz de hardware para permitir la comunicación entre módulos de adquisición y la computadora personal con un alcance de hasta 50 metros.

- Crear una base de datos que almacene las mediciones, y que se conecte con la interfaz de usuario, para poder observar la evolución de variables en el transcurso de tiempo sobre un intervalo seleccionado.
- Incluir en la interfaz gráfica una sección para la configuración del envío de alertas vía GSM, cuándo alguna variable este fuera del rango definido por el agricultor.
- Realizar PCB para los diversos módulos, así como carcasas protectoras.

1.3 Actividades

- Investigar principales tipos y características de invernaderos.
- Conocer condiciones para cultivo de jitomate en invernadero.
- Estudiar tipos de sensores de humedad de suelo.
- Caracterizar sensores de humedad del suelo.
- Identificar tipos de sensores para medir variables climáticas.
- Investigar lenguajes de programación para el uso de sensores.
- Implementar sistema de alertas SMS.
- Diseñar PBC maestra con reguladores de voltaje, sistema GSM y 3 módulos de digitalización independientes.
- Diseñar PCB para montaje de sensores.
- Conectar PCB con base de datos.
- Crear una interfaz gráfica de usuario para observar el estado de las variables y establecer ajustes del sistema.
- Crear base de datos para almacenar las lecturas de sensores.
- Crear algoritmos y códigos para captura de datos en cotas de tiempo definidas.
- Diseñar modelo 3D para alojar componentes electrónicos del sistema.

1.4 Justificación

En la actualidad existen personas que poseen pequeños invernaderos que no cuentan con soporte tecnológico, principalmente porque los sistemas que se encuentran en el mercado son muy costosos, ya que están enfocados a invernaderos de grandes dimensiones, y en la mayoría de los casos las empresas dedicadas a la venta de estos sistemas no cuentan con alternativas para los pequeños productores. Tal es el caso de los invernaderos de la FES Aragón; por lo que se tiene como propósito contribuir en optimizar sus cultivos mediante la implementación de sistemas de control de variables. Esto será llevado a cabo en varias etapas, las cuales en una definición inicial son:

1. Desarrollo de módulos de adquisición de las variables indicadas por el responsable de los invernaderos de la Facultad como más significativas: humedad, temperatura e iluminación en el ambiente, así como la humedad en el sustrato.

2. Diseño y programación de una interfaz gráfica de usuario sobre una PC para observar las mediciones de las variables al momento.

3. Implementación de una base de datos enlazada a la interfaz gráfica de usuario, para el almacenamiento histórico de mediciones y su visualización gráfica en intervalos temporales seleccionable.

4. Incorporación de un módulo GSM para el envío de alertas al teléfono del agricultor cuándo las mediciones estén fuera de un rango establecido.

5. Aislamiento de módulos para protección de componentes electrónicos.

6. Pruebas del sistema en el invernadero (los puntos anteriores consideran solo pruebas de laboratorio), y realización de correcciones y ajustes.

7. Automatización de riego, ventilación e iluminación.

Para este trabajo se tiene el propósito de desarrollar los primeros cuatro puntos mencionados.

El proyecto además del beneficio de mejorar la producción, contribuirá a la formación de recursos humanos, ya que se requiere una colaboración entre alumnos y profesores de las carreras de Ingeniería Eléctrica Electrónica y de Planificación para el Desarrollo Agropecuario, lo cual ayuda al desarrollo de aplicación del conocimiento, prestación de servicio social, y trabajos de titulación entre otros.

1.5 Organización del trabajo

Capítulo 2. Se describen las características y los tipos de invernaderos existentes, condiciones climáticas para el cultivo de jitomate en invernadero, plataformas para el desarrollo de software y los sensores disponibles para la obtención de variables climáticas.

Capítulo 3. En este capítulo se describen los sensores seleccionados que conformaran el sistema, el diseño electrónico, los algoritmos y programación del sistema, base de datos, así como la elaboración de la interfaz gráfica de usuario.

Capítulo 4. En este capítulo se describen las pruebas llevadas a cabo de distintos diseños electrónicos, pruebas para la caracterización de los sensores analógicos, y sus respectivos resultados.

2.Marco teórico

2.1 Principales tipos de invernaderos

Un invernadero es una construcción agrícola de estructura metálica o plástica, usada para el cultivo y/o protección de plantas, con cubierta de película plástica traslucida que no permite el paso de la lluvia al interior y que tiene por objetivo reproducir o simular las condiciones climáticas más adecuadas para el crecimiento y desarrollo de las plantas cultivadas establecidas en su interior [4].

2.1.1 Invernadero Túnel

Está compuesto por uno o varios módulos con una serie de arcos fabricados con tubos cilíndricos galvanizados, los cuales no precisan de zapatas de hormigón, lo que posibilita su traslado y fácil instalación. Su forma permite alojar un volumen mayor de aire en su interior y proporciona resistencia a lluvia (Ilustración 2-1) [5].

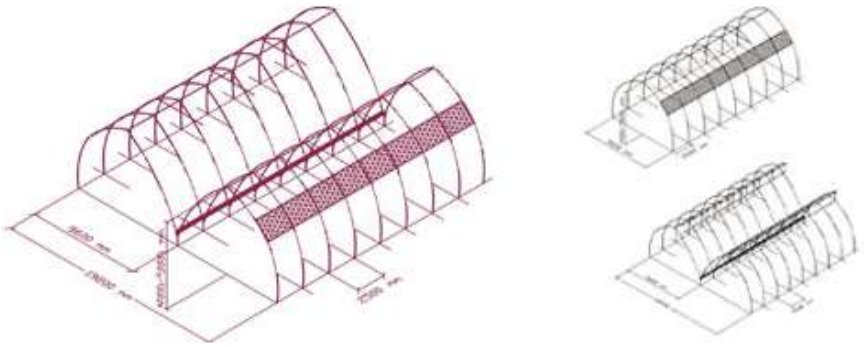


Ilustración 2-1 Invernadero tipo túnel.

Ventajas:

- Se trata de un tipo de invernadero barato y sencillo.
- Buen reparto de la luminosidad en el interior del invernadero.
- Reduce considerablemente el problema de la condensación y el goteo del agua en los cultivos debido a la cubierta curva, la cual favorece la evacuación hacia las paredes del agua proveniente de la condensación en la cubierta plástica.
- Montaje rápido y sin soldaduras.
- Facilita las operaciones agrícolas con maquinaria.

Características estándares:

- Ancho: 8- 9.60 m.
- Altura al cenit: 4 - 5 m.

- Distancia entre arcos: 2,50 m. (externas).
- Bastidores de refuerzo perimetrales.

2.1.2 Invernadero Capilla

El invernadero tipo capilla o también denominado multicapilla, se caracteriza por la forma de su cubierta formado por arcos curvos semicirculares y por su estructura totalmente metálica. Las diferentes partes se unen con grapas, tuercas y tornillos, por lo que no es necesario soldar (Ilustración 2-2) [6].

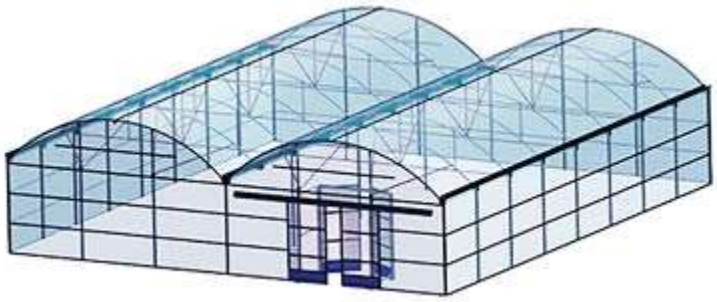


Ilustración 2-2 Invernadero tipo capilla.

Ventajas:

- Pocos obstáculos en su estructura.
- Buena ventilación.
- Permite la instalación de ventilación cenital, así como ventilación perimetral.
- Buen reparto de la luminosidad en el interior del invernadero.
- Fácil instalación.

Características estándares:

- Ancho: 8 m - 9.60 m.
- Altura bajo canal: 4 m - 5 m – 5.50 m.
- Altura al zenit: 5.80 m - 6.30 m - 6.80 m.
- Separación entre pilares: 5 m (interior) – 2,50 m (exterior).

2.1.3 Invernaderos Góticos

El tipo de Invernadero Gótico se diferencia del tipo capilla en el diseño de los arcos, siendo estos de tipo ojival, permite albergar un mayor volumen de aire, proporcionando un mejor microclima e iluminación interior. Está diseñado para adaptarse a todo tipo de cultivos, particularmente a cultivos suspendidos y su construcción está orientada a climas extremos (Ilustración 2-3) [7].

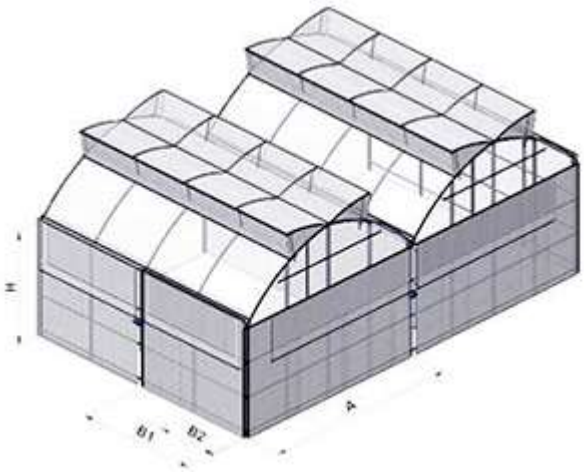


Ilustración 2-3 Invernadero tipo gótico.

Ventajas:

- Alta duración y resistencia a la corrosión.
- Eficacia de fijación del plástico de cubierta (buena hermeticidad).
- Montaje rápido y sin soldaduras.
- Mayor distancia de la ventilación al cultivo.
- Fácil deslizamiento de la condensación.
- Mayor ventilación.
- Permite realizar labores agrícolas mecanizadas en su interior.

Características estándares:

- Ancho: 8 m - 9.60 m dependiendo del modelo.
- Altura bajo canal: 4 - 4,50 – 5 m.
- Altura al cenit: entre 6 y 7,40 metros dependiendo del modelo.
- Distancia entre pilares: 5 m internas – 2,50 m externas.

2.1.4 Invernadero Tropical o asimétrico

Se denominan Invernaderos Tropicales porque su uso está muy extendido en estas regiones, y Asimétrico, porque, a diferencia de los invernaderos tipo capilla y góticos, su geometría es asimétrica, siendo uno de los lados de la cubierta más inclinado que el otro.

La inclinación de la cubierta se estudia en función de la incidencia perpendicular sobre la misma de la luz al medio día solar, durante el invierno, con el objetivo de aprovechar al máximo la radiación solar incidente.

La ventilación de este invernadero suele ser fija y es resuelta a través de las aperturas localizadas en el centro de cada uno de los arcos estructurales que corren a lo largo de todo

el techo. Las aperturas permiten ventilación natural y la salida de aire caliente (Ilustración 2-4) [8].

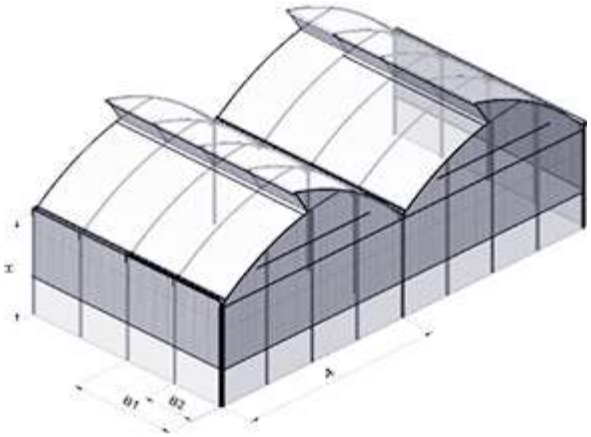


Ilustración 2-4 Invernadero asimétrico.

Ventajas:

- Buen aprovechamiento de la luz en época invernal.
- Elevada inercia térmica debido a su gran volumen unitario.
- Buena ventilación debido a su elevada altura.
- Permite la instalación de ventilación cenital a sotavento.
- Resistencia a fuertes vientos.
- Montaje rápido y sin soldaduras.

Características estándares:

- Ancho: 9.60 m.
- Altura del pilar: 6,4 - 6,90 - 7,40 m.
- Altura debajo canal: 4 - 4,50 - 5 m.
- Distancia entre pilares: 4 - 5 m. (internos). 2 – 2,50 m. (externos).
- Bastidores de refuerzo perimetrales.

2.2 Cultivo de jitomate en invernadero

El tomate (rojo) Solanumlycopersicum, pertenece a la familia Solanaceae. Es una planta herbácea anual, bianual, de origen centro y sudamericano. Actualmente es cultivado para consumo fresco e industrializado.

El tomate rojo es una hortaliza que presenta una alta diversidad genética, existiendo innumerables variedades con distinto aspecto, color y sabor, además de una demanda que aumenta continuamente y, con ella su producción y comercialización. No obstante, este

incremento de la producción obedece más bien a un mayor rendimiento que a un crecimiento en la superficie cultivada.

Estos rendimientos superiores a su vez, son producto de la incorporación de altas tecnologías de cultivo, que permiten el manejo de los factores ambientales (climáticos) y recursos naturales (agua, suelo, fertilizantes) conjuntamente al manejo y prácticas adecuadas del cultivo. Esto permite la oferta de tomate durante todo el año [9].

2.2.1 Condiciones de suelo

La rusticidad de la planta de tomate permite que sea poco exigente a las condiciones de suelo. Sin embargo, debe tener un buen drenaje, de aquí la importancia de un suelo con alto contenido de materia orgánica. En suelos arcillosos y arenosos, se desarrolla con un mínimo de 40 cm de profundidad.

En cuanto al pH de suelo, el óptimo debe oscilar entre 6 y 6.5 para que la planta se desarrolle y disponga de nutrientes adecuadamente [9]. El pH es la medida del grado de acidez o alcalinidad de una sustancia o una solución. El pH se mide en una escala de 0 a 14. En esta escala, un valor pH de 7 es neutro, esto significa que la sustancia o solución no es ácida ni alcalina. Un valor pH de menos de 7 significa que es más ácida y un valor del pH de más de 7 significa que es más alcalina [10].

Los suelos pueden ser desde ligeramente ácidos hasta ligera a medianamente alcalinos, es posible encontrar cultivos de tomate establecidos en suelos que presentan pH 8, siendo un factor posible de manejar, ya que el tomate es la especie cultivada en invernadero que mejor tolera las condiciones de pH. Situación similar respecto a la salinidad, tanto del suelo como del agua de riego, incluso en suelos enarenados, sobre presentar conductividades superiores 3dS/m (conductividad eléctrica, deciSiemens por metro) técnica que reduce la evapotranspiración (perdida de humedad por evaporación) al disminuir el movimiento del agua por capilaridad [9].

Para cultivos en condiciones de ambiente controlado, es muy importante el tipo de sustrato que se utilice. Un sustrato puede definirse como cualquier tipo de material en el que se depositan semillas o raíces para posibilitar su desarrollo. Los sustratos sin suelo, conocidos como artificiales, poseen algunos requerimientos básicos para ser viables: estar libres de

patógenos, poseer buenas cualidades de aireación y drenaje, y una capacidad de retención de agua suficiente para prevenir una sequedad excesiva [11].

Algunos de los sustratos más comunes en México son:

- Fibra de coco**;** formada por cascara de coco molida. Este es un de los sustratos más ampliamente utilizados, que se producen en varias zonas de México. La fibra de coco tiene propiedades que previene la desintegración típica de otro sustrato, por lo cual es posible utilizarlo durante varios años.
- La Composta puede considerarse como un sustrato de relativamente bajo costo y con alto grado de sustentabilidad formado por desechos orgánicos locales. Producida de manera correcta, la composta contiene gran cantidad de microorganismos benéficos y en consecuencia puede reducir el uso de fertilizantes.
- La Vermiculita**,** que es un mineral natural que se expande el ser calentado. Posee una estructura de placas cóncavas que le permite retener grandes volúmenes de agua, así como nutrientes con carga positiva tales como potasio, magnesio y calcio. Suele emplearse en mesclas, y no por sí solo en general, pero también pueden utilizarse para propagar semilla.
- La Perlita es un cristal volcánico amorfo que, al igual que la vermiculita, se expande al ser calentado, lo cual es parte del proceso de preparación para su uso como sustrato de cultivo. Posee alta permeabilidad y baja capacidad de retención de agua; razón por la cual es usado como aditivo en otros sustratos.
- La Turba o "Peat moss". Es un producto que incrementa la capacidad de retención de agua y nutrientes del suelo. Tiene un pH acido, en un rango de 3.8 a 4.3. Uno de los mayores beneficios de este sustrato es que no solo previene el lixiviado de nutrientes excesivo, sino que libera lentamente el cultivo. Su mayor desventaja es que repele agua, por lo que es difícil mantenerlo en nivel adecuado de hidratación.

En México la fibra de coco ha ido sustituyendo los sustratos tradicionales compuestos a base de turba. Esto ha sido así porque la fibra de coco ofrece una mayor precocidad para plantas sanas, tiene un gran poder de retención (tanto minerales como agua).

Uno de los principales problemas que presenta la fibra de coco es su contenido en sales, pues el cultivo del cocotero se hace en zonas costeras, azotadas por vientos salinos, brisas y suelos más o menos salinos. Al final, esas fibras del coco contienen una gran cantidad de sales que podrían fácilmente pasar al cultivo. La solución consiste en "lavar" la fibra de coco antes y después del proceso de triturado.

Estas son algunas de las principales características genéricas, físicas y químicas, de los sustratos a base de fibra de coco [11]:

- pH: 5,5 a 6,5.
- Conductividad eléctrica media: < 0.8 mS/cm (mili Siemens por centímetro).
- Porosidad total, o "% de aireación": 10 a 40 %.
- Capacidad de retención de agua a 1/3 de bar: 25 a 50 %.
- C/N (Relación carbono orgánico a Nitrógeno total) 80:1.

2.2.2 Condiciones climáticas

Aunque se produce en una amplia gama de condiciones de clima y suelo, el tomate prospera mejor en climas secos con temperaturas moderadas (Tabla 2-1). Su rusticidad asociada a nuevas variedades permite su cultivo en condiciones adversas. No obstante, el tomate es una especie de estación cálida, su temperatura óptima de desarrollo varía entre 18 y 30°C, por ello, el cultivo al aire libre se realiza en climas templados. Temperaturas extremas pueden ocasionar diversos trastornos, ya sea en la maduración, precocidad o color. Temperaturas bajo 10°C afectan la formación de flores y temperaturas mayores a 35°C pueden afectar la fructificación. Asimismo, la temperatura nocturna puede ser determinante en la producción, ya que, cuando es inferior a 10°C originaría problemas en el desarrollo de la planta y frutos, provocando deformidades [9].

Tabla 2-1 Temperaturas para el cultivo de tomate

Temperatura (°C)	Consecuencia
-2	Se hiela la planta
10 a 12	Detiene su desarrollo
18 - 25	Desarrollo normal de la planta
21 - 24	Mayor desarrollo de la planta
25 - 30	Germinación optima
15 - 22	Maduración
Temperaturas optimas (diurna)	
23 - 26	Desarrollo
23 - 26	Floración
Temperaturas optimas (nocturna)	

13 – 16	Desarrollo
15 - 18	Floración

No obstante, se debe considerar que los valores de temperaturas por sí solos son referenciales, puesto que su interacción con otros factores repercute mayormente. Por ejemplo, la combinación de altas temperaturas con humedad baja, puede generar aborto floral y baja viabilidad del polen, el desarrollo del tomate requiere que ésta oscile entre 60% y 70%, considerando que humedades relativas muy elevadas favorecen el desarrollo de enfermedades fungosas y bacterianas, además, dificultan la fecundación, debido a que el polen se compacta abortando parte de las flores [9]. En términos generales las necesidades óptimas para el desarrollo y producción del cultivo de jitomate son: temperatura en el día de 23 a 25°C en la noche de 15 a 17°C. la máxima es de 30°C, la mínima 8°C, humedad relativa 60%, máxima 70%, mínima 50%, luz 100%, 12 horas de luz y 12 horas de oscuridad, excelente ventilación para tener suficiente oxígeno y bióxido de carbono y humedad del suelo por debajo de la capacidad de campo [12].

2.3 Sensores eléctricos

En Física, se llaman magnitudes a aquellas propiedades que pueden medirse y expresar su resultado mediante un número y una unidad. Son magnitudes: la longitud, la masa, el volumen, el voltaje, la temperatura, la intensidad luminosa, etc. [13] por ejemplo, en una lectura de 12 °C, la magnitud es la temperatura, el numero 12 corresponde a la cantidad de temperatura presente en la medición, y los grados Celsius (°C) es la unidad utilizada para representar la cantidad temperatura en comparación a otros parámetros físicos, en el caso de los grados Celsius, se compara la escala de temperatura respecto al punto de congelamiento del agua a nivel del mar.

El término sensor se refiere a un elemento de medición que detecta la magnitud de un parámetro físico, (temperatura, humedad, presión, voltaje, etc.) y lo cambia por una señal que puede interpretar el sistema. Al elemento activo de un sensor se le conoce comúnmente como transductor [14], el transductor es un dispositivo que transforma el efecto de una magnitud física, en otro tipo de señal, normalmente eléctrica [15].

En los sensores eléctricos, la magnitud física obtenida por el transductor, es comúnmente un nivel de voltaje, el cual corresponde al estado en el que se encuentra la magnitud física que se desea medir. De esta manera es posible establecer modelos matemáticos que caractericen el nivel de voltaje obtenido por el sensor y asignarle un estado definido de la magnitud física original y de esta forma, conocer el estado en el que se encuentra la magnitud física.

Para que se pueda establecer una comunicación entre el sistema electrónico y lo sensores, el sensor debe de proporcionar la señal en un formato que el sistema electrónico pueda interpretar, en los sistemas electrónicos existen dos tipos de señales:

- Las señales analógicas son percibidas en el ambiente y se transforman en señales eléctricas mediante un transductor, para su tratamiento electrónico [16]. Las señales analógicas eléctricas varían de forma continua en el tiempo, entre un intervalo de valores correspondientes a niveles de voltaje (Amplitud), es decir, que a cada valor en el tiempo le corresponderá un valor de voltaje. (Ilustración 2-5).
  Las señales analógicas presentan varias desventajas, como la atenuación de las señales debido a la resistencia eléctrica de los conductores por las que se transmiten, esto sucede a largas distancias y provoca lecturas erróneas, otra de las desventajas, es que son vulnerables a ruidos naturales o artificiales, es decir, que la señal puede sufrir alteraciones debido a la interferencia con otros sistemas electrónicos, como las telecomunicaciones o pueden verse afectadas por las condiciones climáticas a las que están expuestas, esto provoca una cambio en el estado de la señal y por consecuencia una lectura errónea.

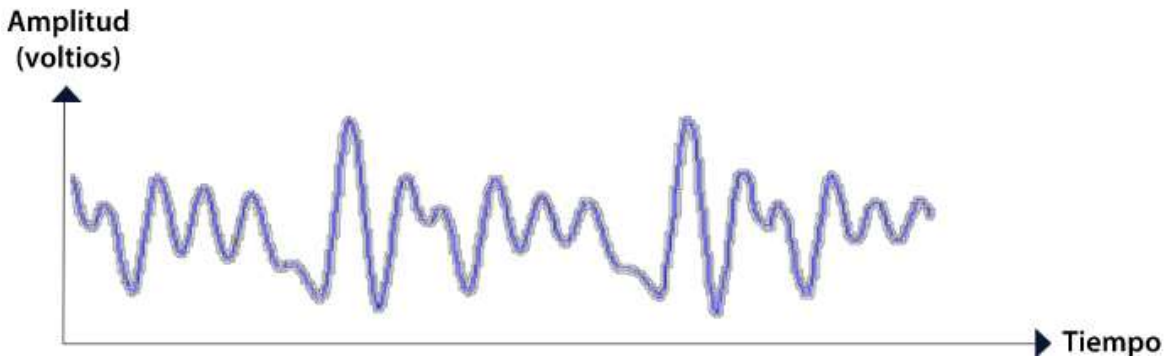


Ilustración 2-5 Señal analógica.

- Las señales digitales son variables eléctricas con dos niveles bien diferenciados que se alternan en el tiempo transmitiendo información según un protocolo previamente acordado. Cada nivel eléctrico representa Uno de dos símbolos: 0 o 1, nivel alto o nivel bajo, etc [17] (Ilustración 2-6).
  Las señales digitales presentan varias ventajas frente a las analógicas: un de las ventajas, es que, en ciertos protocolos de comunicación, se pueden alcanzar distancias mayores sin que la señal se atenúe y existan lecturas erróneas, existe la posibilidad de

corregir errores de las lecturas obtenidas y son menos vulnerables al ruido artificial y al ruido natural.

Existe una amplia variedad de protocolos para la transmisión de datos utilizando señales digitales, en las cuales se profundizará más adelante.

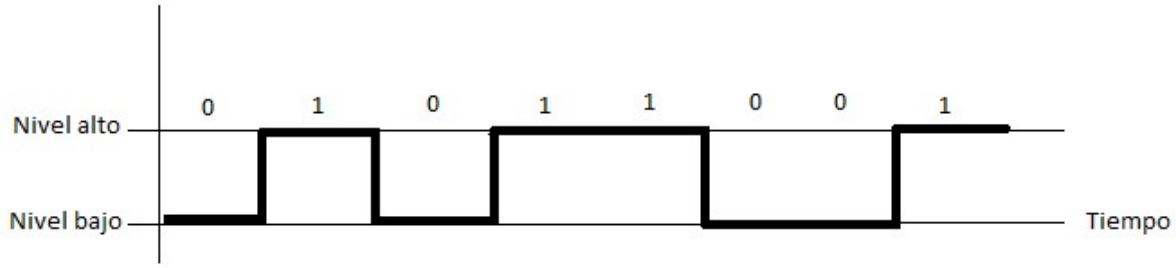


Ilustración 2-6 Señal digital.

2.3.1 Sensores de humedad de suelo

Los sensores de humedad de suelo se encargan de enviar un nivel de voltaje al sistema electrónico, correspondiente a un nivel determinado de humedad en el suelo. Esto se consigue principalmente empleando dos tipos de sensores:

- Sensores conductivos. Este tipo de sensores utiliza la conductividad eléctrica del suelo con el que este en contacto para variar una corriente eléctrica, en función de la cantidad de humedad y salinidad presente en el suelo, se tendrá una mayor o menor conductividad eléctrica, lo cual produce una mayor o menor corriente eléctrica respectivamente, a mayor corriente eléctrica (mayor humedad), menor será el voltaje presente en el transductor, y a menor corriente eléctrica (menor humedad), mayor será el voltaje presente en el transductor. De esta manera se puede tomar una muestra de voltaje, el cual puede ser interpretado por el sistema electrónico.
- Sensores capacitivos. Los capacitores o condensadores son elementos eléctricos que pueden almacenar y liberar energía eléctrica basándose en fenómenos relacionados con campos eléctricos.

  Los sensores capacitivos utilizan un capacitor, este capacitor se pone en contacto directo con el suelo para poder variar la capacitancia, en función de la cantidad de humedad presente en el suelo se tendrá una mayor o menor capacitancia, a mayor humedad presente en el suelo, menor será la capacitancia presente en el transductor, y a menor humedad presente en el suelo, mayor será la capacitancia presente en el transductor. De esta forma se puede tomar una muestra de capacitancia y se le asigna un valor determinado de voltaje, el cual puede ser interpretado por el sistema electrónico.

2.3.1.1 Higrómetro de suelo FC-28

Un higrómetro de suelo FC-28 (Ilustración 2-7) es un sensor que mide la humedad del suelo. Son ampliamente utilizados en sistemas automáticos de riego para detectar cuando es necesario activar el sistema de bombeo.

El FC-28 es un sensor sencillo que mide la humedad del suelo por la variación de su conductividad. No tiene la precisión suficiente para realizar una medición absoluta de la humedad del suelo. El FC-28 se distribuye con una placa de medición estándar que permite obtener la medición como valor analógico o como una salida digital, activada cuando la humedad supera un cierto umbral, este umbral de sensibilidad puede ser ajustado mediante una pequeña resistencia variable instalada en la placa electrónica [18]. La señal que entrega este sensor es de tipo analógica, tiene un precio actual de $20 MXN.

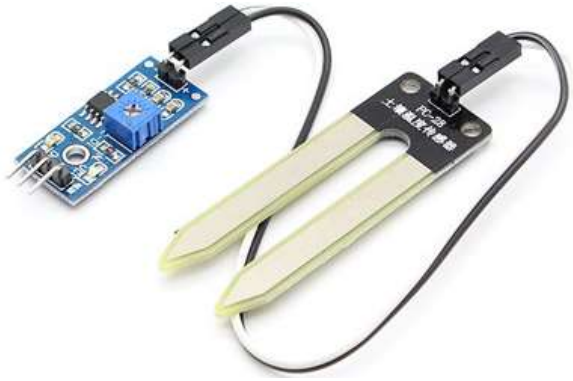


Ilustración 2-7 Sensor FC-28.

2.3.1.2 Sensor de humedad capacitivo

Este sensor de humedad, es un sensor que emplea un transductor capacitivo para medir el nivel de humedad presente en el suelo, está construido en materiales resistentes a la corrosión entregando una mayor durabilidad, la señal de salida tiene una mayor estabilidad debido a que no depende de la conductividad presente en el suelo para tomar las mediciones, como si lo hacen los sensores conductivos (Ilustración 2-8). La señal que entrega este sensor es de tipo analógica, tiene un precio actual de $31.16 MXN [19].

Ilustración 2-8 Sensor de humedad capacitivo.

2.3.1.3 Sensor VH400

El VH400 es un sensor electrónico profesional de humedad del suelo. Es tan sensible que puede medir la humedad en las manos al contacto con la cuchilla.

Hecho de plástico ABS duradero y fibra de vidrio, el sensor de humedad VH400 es absolutamente resistente al agua y resistente a la corrosión, listo para aplicaciones de alto estrés. Se puede enterrar a cualquier profundidad en el suelo, o se puede insertar sobre el suelo o en el suelo de plantas en macetas.

Muchos otros sensores basados especialmente en la conductividad o resistividad, no son efectivos, porque las sales y los fertilizantes interfieren con sus lecturas. El sensor de humedad VH400 mide la constante dieléctrica del suelo, lo cual lo hace insensible a la salinidad del suelo (Ilustración 2-9). La salida que entrega este sensor es de tipo analógica, tiene un precio actual de $39.95 USD [20].

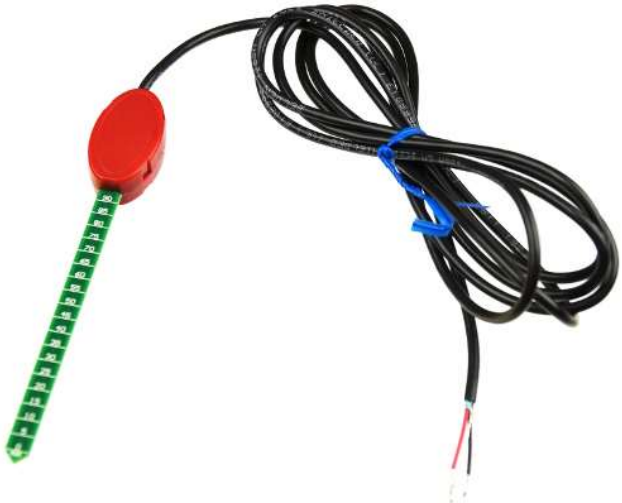


Ilustración 2-9 Sensor VH400.

2.3.2 Sensores de variables climáticas

Como ya hemos visto anteriormente, las principales variables climáticas que afectan el óptimo desarrollo del cultivo de tomate son: temperatura ambiente, humedad ambiente y luminiscencia. Existe una amplia variedad de sensores que cumplen con el propósito de medir estar variables, algunos proporcionan señales digitales o señales analógicas, cada Uno con un precio de mercado en función de sus materiales de construcción, durabilidad y fiabilidad.

2.3.2.1 Sensor DHT11

El sensor DHT11 es un sensor digital de temperatura y humedad relativa, de bajo costo y fácil uso. Integra un sensor capacitivo de humedad y un termistor para medir la temperatura en el aire circundante, y muestra los datos mediante una señal digital en el pin de datos (no posee salida analógica). Utilizado en aplicaciones académicas relacionadas al control automático de temperatura, aire acondicionado, monitoreo ambiental en agricultura y más.

El sensor DHT11 (Ilustración 2-10) tiene soporte para las plataformas Arduino/Raspberry Pi/Nodemcu, utilizar estas plataformas es muy sencillo tanto a nivel de software como hardware. A nivel de software se dispone de bibliotecas para Arduino con soporte para el protocolo OneWire. En cuanto al hardware, solo es necesario conectar el pin VCC de alimentación a 3-5V, el pin GND a Tierra (0V) y el pin de datos a un pin digital de la plataforma que se desee emplear. El protocolo de comunicación entre el sensor y el microcontrolador emplea un único hilo o cable, la distancia máxima recomendable de longitud de cable es de 20m.

En comparación con el DHT22, este sensor es menos preciso, menos exacto y funciona en un rango más pequeño de temperatura y humedad, pero su empaque es más pequeño y de menor costo. Sus características técnicas son [21]:

- Voltaje de operación: 3v a 5v DC.
- Rango de medición de temperatura: 0 a 50°C.
- Precisión de medición de temperatura: ±2.0°C.
- Resolución de temperatura: 0.1°C.
- Rango de medición de humedad: 20% a 90% HR.
- Precisión de medición de humedad: 5%.
- Resolución de humedad: 1% HR
- Tiempo de medición: 1s.
- Tipo de señal: digital, usa el protocolo OneWire.
- Material de construcción: plástico celeste.

El DHT11 tiene un precio actual de $34 MXN [19].

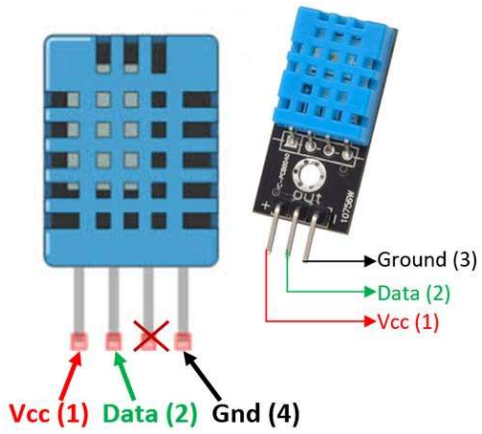


Ilustración 2-10 Distribución electrónica DHT11.

2.3.2.2 Sensor DHT22

El DHT22 (AM2302) es un sensor digital de temperatura y humedad relativa de buen rendimiento y bajo costo. Integra un sensor capacitivo de humedad y un termistor para medir el aire circundante, y muestra los datos mediante una señal digital en el pin de datos (no posee salida analógica). Utilizado en aplicaciones de control automático de temperatura, aire acondicionado, monitoreo ambiental en agricultura y más.

El sensor DHT22 (Ilustración 2-11) tiene soporte para las plataformas Arduino/Raspberry Pi/Nodemcu, utilizar estas plataformas es muy sencillo tanto a nivel de software como hardware. A nivel de software se dispone de bibliotecas para Arduino con soporte para el protocolo OneWire. En cuanto al hardware, solo es necesario conectar el pin VCC de alimentación 3 a 6v, el pin GND a Tierra (0V) y el pin de datos a un pin digital de la plataforma de desarrollo que se desee emplear. El protocolo de comunicación entre el sensor y el microcontrolador emplea un único hilo o cable, la distancia máxima recomendable de longitud de cable es de 20m.

El DHT22 presenta mejores prestaciones respecto al sensor DHT11, como mejor resolución, mayor precisión y un empaque más robusto. Sus características técnicas son [21]:

- Voltaje de operación: 3v a 6v DC.
- Rango de medición de temperatura: -40 a 80°C.

- Resolución de temperatura: 0.1°C.
- Rango de medición de humedad: 0 a 100% HR.
- Resolución de humedad: 0.1%HR.
- Tiempo de medición: 2s.
- Tipo de señal: digital, utiliza el protocolo OneWire.
- Material de construcción: carcasa plástica blanca.

El DHT22 tiene un precio actual de $89 MXN [19].

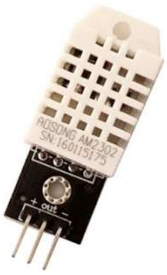


Ilustración 2-11 Sensor DHT22.

2.3.2.3 Sensor LM35

El LM35 es un sensor de temperatura (Ilustración 2-12) de buenas prestaciones a un bajo precio. Posee un rango de trabajo desde -55ºC hasta 150ªC. Su salida es de tipo analógica y lineal con una pendiente de 10mV/ºC (mili volt por gado Celsius). El sensor es calibrado de fábrica a una precisión de 0.5ºC.

Es un sensor ampliamente utilizado por su fácil uso y variadas aplicaciones. No necesita de ninguna configuración adicional para ser utilizado. Se alimenta directamente con una fuente de 5V y entrega una salida analógica entre 0V a 1.5V. Este voltaje analógico puede ser leído por el ADC de un microcontrolador como PIC o Arduino. Entre sus aplicaciones podemos encontrar termómetros, termostatos, sistemas de monitoreo y más. Sus características técnicas son [22]:

- Voltaje de operación: 4v a 30v (5v recomendado).
- Rango de trabajo: -55 a +150°C.
- Precisión de medición: ±0.5°C (solo aplica en el rango de -10 a +85°C).
- Tipo de señal: analógica.
- Curva característica: lineal.

Debido a que este sensor tiene una salida de tipo analógica, se debe obtener la función matemática que describa su comportamiento, para de esta forma convertir la señal analógica de voltaje entregada por el sensor, en una medición de temperatura expresada en grados Celsius. Este sensor tiene un precio actual de $27.5 MXN [19].

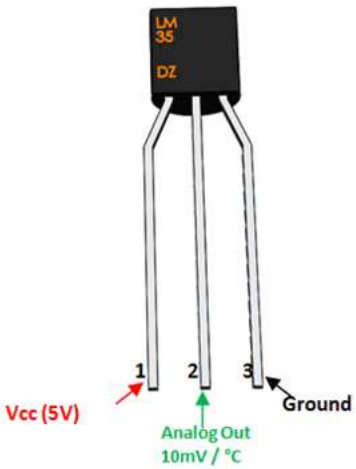


Ilustración 2-12 Sensor LM35.

2.3.2.4 Sensor DS18B20-1

El sensor de temperatura DS18B20 (Ilustración 2-13) es un dispositivo que se comunica de forma digital. Cuenta con tres terminales: Vcc, GND y pin Data. Este sensor utiliza comunicación por protocolo serial digital OneWire. Esté protocolo de comunicación permite enviar y recibir datos utilizando un solo cable. A diferencia de otros, que utilizan dos o más líneas de comunicación digital [23]. cuenta con soporte para las plataformas Arduino y Nodemcu. Dispone de bibliotecas compatibles con Arduino IDE, para la obtención de mediciones.

Sus características técnicas son:

- Rango de operación: -50 a 125°C.
- Precisión: ±0.5°C.
- Tipo de señal: digital con protocolo OneWire.
- Material de construcción: plástico y acero inoxidable.

Este sensor tiene un precio actual de $48 MXN [19].

Ilustración 2-13 Sensor DS18B20.

2.3.2.4 Sensor BME280

El sensor BME280 (Ilustración 2-14) integra en un solo dispositivo sensores de presión atmosférica, temperatura y humedad relativa, con gran precisión, bajo consumo energético y un formato ultra compacto. Basado en tecnología BOSCH piezo-resistiva con gran robustez EMC, alta precisión y linealidad, así como con estabilidad a largo plazo. Se conecta directamente a un microcontrolador a través de los protocolos digitales I2C o SPI. En cuando al sensor de humedad relativa presenta un desempeño sobresaliente comparado a los sensores DHT22 o DHT21.

Este tipo de sensores pueden ser utilizados para calcular la altitud con gran precisión (barómetro), por lo que es un sensor muy utilizado en sistemas de Autopiloto para Drones, entregando medidas de altitud con una precisión de hasta 1m. Otras aplicaciones son: Monitoreo de clima, Internet de las Cosas, Monitor de salud/fitness, Automatización del hogar o Domótica y Aire acondicionado. Características técnicas [24]:

- Voltaje de operación: 1.8v a 3.3v DC (sin placa de circuito impreso).
- Tipo de señal: digital, usa los protocolos I^2C o SPI.

- Rango de presión: 300 a 1100hPa
- Resolución: 0.16Pa.
- Rango de temperatura: -40 a 85°C.
- Resolución de temperatura: 0.01°C.
- Precisión de temperatura: 1°C.
- Rango de humedad relativa: 0 a 100%HR.
- Precisión de humedad: ±3%.
- Rango de altura medible: 0 a 9100m.
- Ultra bajo consumo de energía.
- Tiempo de medición: 0.0063s.

Este sensor tiene soporte para las plataformas Nodemcu y Arduino, cuenta con bibliotecas compatibles con Arduino IDE. Tiene un precio actual de $75 MXN [19].

Ilustración 2-14 Sensor BME280.

2.3.2.5 Sensor BH1750

El módulo BH1750 es un sensor de iluminación digital (Ilustración 2-15) para medición de flujo luminoso (iluminancia) de la empresa Rohm Semiconductor. Entrega una salida digital con el protocolo I2C. Su desempeño es mejor al de un Foto-Resistor (LDR), pues no es necesario realizar conversiones de voltaje para obtener datos interpretables.

El BH1750 entrega la intensidad luminosa directamente en unidades Lux (lx). El lux es la unidad derivada del Sistema Internacional de Unidades para la iluminancia o nivel de iluminación. Equivale a un lumen /m². Se usa en la fotometría como medida de la luminancia, tomando en cuenta las diferentes longitudes de onda según la función de luminosidad, un modelo estándar de la sensibilidad a la luz del ojo humano. Sus características técnicas son [25]:

- Voltaje de operación: 3v a 5v DC.
- Tipo de señal: digital con protocolo I²C.
- Respuesta espectral similar al ojo humano.
- Rango de medición: 1 a 65535Lx.

- Baja dependencia de la fuente de luz (halógeno, LED, incandescente o luz solar) para tomar mediciones.

Tiene un precio actual de $47 MXN [19].

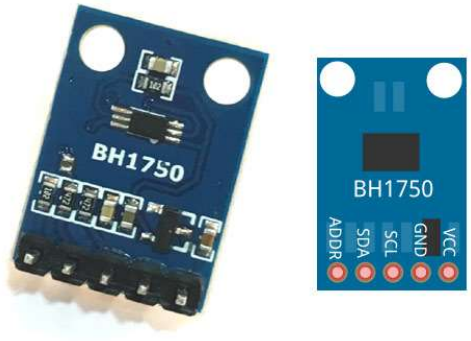


Ilustración 2-15 Sensor BH1750.

2.3.2.6 Sensor TEMT6000

TEMT6000 es un fototransistor de silicio con un encapsulado transparente en miniatura para montaje en superficie en una placa de circuito impreso (Ilustración 2-16). El dispositivo es sensible al espectro visible (luz visible). Sus especificaciones técnicas son [26]:

- Receptividad adaptada al ojo humano.
- Angulo de medición: ± 60°.
- Pequeñas dimensiones.
- Voltaje de trabajo: 3v a5v.
- Tipo de señal: analógica.

Este sensor entrega una señal de tipo analógica, por lo que es necesario obtener su curva característica para poder transformar los niveles de voltaje entregados por el sensor, en una medición de luminiscencia expresada en luxes. Tiene un precio actual de $55 MXN [19].

Ilustración 2-16 Sensor TEMT6000.

2.4 Arduino

Arduino es una plataforma de desarrollo basada en software y una placa electrónica de hardware libre que incorpora un microcontrolador reprogramable [27].

Esta plataforma de desarrollo ofrece varias ventajas respecto a otros sistemas otros sistemas:

- Económico**.** Las placas de Arduino son relativamente económicas en comparación con otras plataformas que utilizan microcontroladores. La última versión más cara de los módulos Arduino puede ser ensamblada a mano, e incluso los módulos pre ensamblados de Arduino tienen un consto menor a $50 USD.
- Multiplataforma. El software de Arduino (IDE) tiene soporte en los sistemas operativos Windows, Macintosh OSX y linux. La mayoria de los sistemas con microcontroladores, estan limitados a Windows.
- Entorno de programación sencillo. El software de Arduino (IDE), es fácil de usar para principiantes, pero lo suficientemente flexible para los usuarios avanzados.
- Software de código abierto y extensible. El software de Arduino está publicado como una herramienta de código abierto, tiene extensiones disponibles por programadores experimentados. El lenguaje puede ser expandido a través de bibliotecas C++, y las personas que quieren entender detalles técnicos pueden saltar del entorno Arduino al lenguaje de programación AVR C en el cual está basado, se puede agregar directamente código AVR-C dentro de la programación Arduino, si así se desea.
- Hardware de código abierto y extensible. Los planes de las placas Arduino se publican bajo una licencia Creative Commons, por lo que los diseñadores de circuitos experimentados pueden hacer su propia versión del módulo, extenderlo y mejorarlo. Incluso los usuarios relativamente inexpertos pueden construir la versión del módulo de prueba para comprender cómo funciona y ahorrar dinero [28]**.**

A nivel de software, la plataforma se llama Arduino IDE, con esta plataforma podemos editar y diseñar el código de programación que se encarga de dar instrucciones específicas a las placas de desarrollo y a los dispositivos externos requeridos, tanto las placas de desarrollo como los dispositivos externos (sensores o actuadores) se consideran hardware.

2.4.1 Arduino IDE

Un entorno de desarrollo integrado (IDE) es un sistema de software para el diseño de aplicaciones que combina herramientas del desarrollador comunes en una sola interfaz gráfica de usuario (GUI). Generalmente, un IDE cuenta con las siguientes características:

- Editor de código fuente. Es un editor de texto que ayuda a escribir el código de software con funciones como el resaltado de la sintaxis con indicaciones visuales, el relleno automático especifico del lenguaje y la comprobación de errores a medida que se escribe el código.
- Automatización de compilación local. Son herramientas que automatizan tareas sencillas y repetibles como parte de la creación de una compilación local del software

para su uso por parte del desarrollador, como la compilación del código fuente en un código binario y la ejecución de pruebas automatizadas.

- Depurador. Es un programa que sirve para probar otros programas y mostrar la ubicación de un error en el código original de forma gráfica [29].

El software Arduino de código abierto (IDE) (Ilustración 2.17) hace que sea fácil escribir código y subirlo a la placa. Se ejecuta en Windows, Mac OS X y Linux. El entorno está escrito en Java que es otro software de código abierto. Este software se puede usar con cualquier placa Arduino [30].

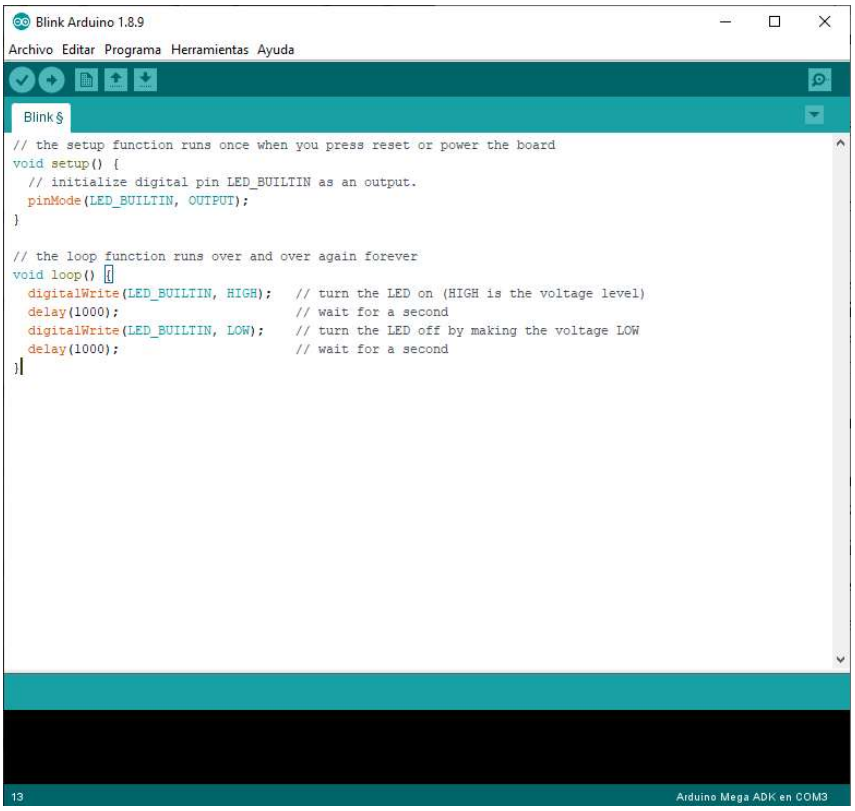


Ilustración 2-17 IDE Arduino.

2.4.2 Programación en Arduino

Los lenguajes de programación como el de Arduino, contienen herramientas tales como estructuras de control, funciones, variables y bibliotecas con las que el desarrollador dispone para codificar, cada una de estas herramientas sirven a un propósito en específico y haciendo uso de una o más de estas, se puede lograr una infinidad de algoritmos y códigos para cada

propósito en específico. A continuación, se hace mención de las funciones necesarias para la elaboración de este proyecto.

2.4.3 Estructuras de control

Las estructuras se pueden definir como un conjunto de instrucciones que realizan una tarea en específico, en el lenguaje de programación de Arduino existen estructuras como, por ejemplo: la estructura for, estructura while, estructura if, estructura if/else, etc.

2.4.3.1 Estructura if

Esta estructura también se le conoce como condicional, la función principal de esta estructura es la de ejecutar o no ciertas instrucciones basándose en si se cumple una condición o no. Para utilizar esta estructura se utiliza la siguiente sintaxis: *if (condición) {instrucciones en caso de cumplirse la condición}*. en esta estructura, en caso de que la condición no se cumpla, el código ignorara todas las instrucciones que contenga la estructura.

2.4.3.2 Estructura if/else

Esta estructura es prácticamente igual a la estructura if, con la única diferencia de que en esta estructura contiene instrucciones en caso de que se cumpla la condición y también en caso de que no se cumpla la condición. Para utilizar esta estructura se utiliza la siguiente sintaxis: *if (condición) {instrucciones en caso de cumplirse la condición} else {instrucciones en caso de no cumplirse la condición}*

2.4.3.3 Estructura for

Esta estructura es utilizada para crear bucles controlados, mediante una variable de control que el desarrollador puede manipular, se puede controlar el número de iteraciones que tiene el ciclo. Esta estructura necesita tres parámetros: variable de control, condición e incremento. La variable de control es la que lleva el conteo del número de veces que se ha repetido el ciclo, esta variable es indispensable para evitar que el ciclo se repita indefinidamente.

La condición, es la que determina cuando se termina el bucle, esta condición se puede establecer al igual que en el ciclo if, la diferencia radica en que en la estructura for, la condición es la que determina cuando se debe detener el ciclo, mientras que en la estructura if, esta condición se utiliza para determinar si se ejecuta o no lo que está dentro de la estructura.

El incremento, es un numero el cual se suma a la variable de control una vez que se ha completado un ciclo, este incremento puede ser cualquier número según convenga al desarrollador, normalmente este incremento es de 1.

Para poder utilizar esta estructura es necesario utilizar la siguiente sintaxis: *for (variable de control; condición; incremento)* {instrucciones a repetir}

2.4.3.4 Estructura while

Esta estructura, al igual que la estructura for sirve para crear ciclos controlados, la diferencia radica en la forma en la que mantienen bajo control el ciclo, en la estructura while el ciclo se repite mientras se cumpla una condición, una vez que la condición deja de cumplirse, el ciclo termina. Para utilizar esta estructura es necesario utilizar la siguiente sintaxis: *while (condición) {instrucciones a repetir mientras se cumple la condición}.*

2.4.3.5 Void setup ()

Esta estructura es utilizada en Arduino para establecer parámetros iniciales que el código necesita para funcionar correctamente, esta estructura solo se ejecuta una sola vez cada que el microcontrolador se pone en funcionamiento, en esta estructura es muy común encontrar parámetros como: tasa de baudios para comunicación serial, iniciación de protocolos de comunicación, configuración inicial para sensores, memorias, displays, etc.

2.4.3.6 Void loop ()

Esta estructura dentro del lenguaje de programación de Arduino que se ejecuta una vez que ya se ha terminado de ejecutar la estructura void setup, esta estructura se ejecuta constantemente siempre y cuando el microcontrolador que contenga el código se encuentre energizado y ningún factor externo o alguna línea de código se lo impida. Esta estructura contiene las instrucciones necesarias para que el microcontrolador cumpla con las tareas que se le asignen. En esta estructura se aloja la parte funcional en la mayoría de los códigos programados en Arduino.

2.4.4 Bibliotecas

Son paquetes de instrucciones y funciones adicionales a las que cuenta el lenguaje de programación. En el caso de Arduino IDE, al ser un software de código abierto las bibliotecas que se pueden utilizar, son creadas por usuarios de la plataforma y distribuidas de forma

gratuita. Para poder hacer uso de estas bibliotecas, es necesario contar con los archivos de la biblioteca, y se tiene que mandar "llamar" a la misma para hacer uso de las instrucciones y paquetes con las que cuenta.

Existe una gran cantidad de bibliotecas con funciones diferentes. Para el desarrollo de este proyecto se hace uso de más de una a la vez, las cuales se describen a continuación.

2.4.4.1 Wire.h

Esta biblioteca se utiliza para hacer uso de la comunicación serial mediante el protocolo I^2C, este protocolo de comunicación es utilizado por una gran variedad de dispositivos electrónicos como sensores, pantallas, memorias flash, digitalizadores, etc. en el caso de este proyecto, este protocolo es utilizado por algunos modelos de sensores de humedad, luz solar y digitalizadores. Esta biblioteca no es necesario descargarla externamente ya que está incluida en el Arduino IDE, solo es necesario "llamarla" con la línea de código: *#include<wire.h>*

2.4.4.2 ErriezBH1750.h

Esta biblioteca es necesaria para poder hacer uso del sensor de luz BH1750, contiene varias funciones para configurar parámetros, establecer resolución de mediciones, seleccionar dirección la dirección I^2C a la que está conectado el sensor, obtener lecturas de más de un sensor conectado, etc. esta biblioteca se puede descargar gratuitamente desde el portal de desarrollo GitHub, para llamarse al IDE de Arduino se utiliza la línea de código: *#include<ErriezBH1750.h>*

2.4.4.3Adafruit_BME280.h

Esta biblioteca es necesaria para poder hacer uso del sensor BME280 el cual se encarga de obtener lecturas de humedad relativa, temperatura ambiente, presión atmosférica y altitud, aunque las últimas no son necesarias para este proyecto, si se hace uso de la medición de temperatura ambiente y la humedad relativa. Esta biblioteca permite obtener lecturas del sensor, seleccionar la dirección I^2C a la que está conectado el sensor, obtener lecturas de más de un sensor conectado, obtener lecturas del sensor utilizando el protocolo de comunicación serial SPI, etc. esta biblioteca se puede descargar gratuitamente del portal de

desarrollo GitHub, para llamarse al IDE de Arduino se utiliza la línea de código: *#include<Adafruit_BME280.h>*

2.4.4.4 DHT.h

Esta biblioteca es necesaria para hacer uso del sensor DHT 11 y DHT 22, este sensor es una opción para obtener lecturas de humedad relativa y temperatura ambiente, esta biblioteca permite obtener lecturas de temperatura ambiente y humedad relativa, seleccionar entre el modelo DHT 11 y DHT 22, utilizar el protocolo de comunicación serial OneWire, obtener datos de más de un sensor conectado. Esta biblioteca se puede descargar gratuitamente del portal de desarrollo GitHub, para llamarla al IDE de Arduino se utiliza la línea de código: *#include "DHT.h"*

2.4.4.5 Adafruit_ADS1015.h

Esta biblioteca es necesaria para hacer uso del digitalizador ADS1015 y ADS1115, con este digitalizador podemos ampliar las entradas analógicas de los microcontroladores. Esta biblioteca permite seleccionar entre los modelos ADS1015 y ADS115, seleccionar la dirección I2C a la que se encuentre conectado, obtener múltiples lecturas de más de un digitalizador conectado, hacer un comparador con las lecturas obtenidas por el digitalizador, etc. esta biblioteca se puede descargar gratuitamente del portal de desarrollo GitHub, para llamarla al IDE de Arduino se utiliza con la línea de código: *#include <Adafruit_ADS1015.h>*

2.4.4.6 Separador.h

Esta biblioteca es utilizada para separar partes de una cadena de caracteres, esto es muy útil cuando se utilizan cadenas de varios valores concatenados y separarlos posteriormente para almacenarlos donde sea necesario. Esta biblioteca fue creada por un canal de YouTube llamado "loticos" y se puede descargar gratuitamente desde el enlace que se proporciona en su video: "EXTRAER PARAMETROS SEPARADOS POR COMAS EN ARDUINO". Para llamar esta biblioteca al IDE de Arduino, se utiliza la línea de código: *#include<Separador.h>*

2.4.4.7 Sim8001.h

Esta es la biblioteca es utilizada para controlar el módulo SIM 8001, con este módulo SIM es posible enviar y recibir mensajes GSM desde un microcontrolador sin necesidad de utilizar un teléfono móvil, esta biblioteca se puede descargar gratuitamente desde el portal de desarrollo GitHub. Para llamar esta biblioteca al IDE de Arduino se utiliza la línea de código: *#include<sim8001.h>*

2.4.5 Variables

En todos los lenguajes de programación las variables son una herramienta fundamental para el desarrollo de programas, su función principal es almacenar datos de forma temporal que pueden ser modificados en cualquier momento a conveniencia del desarrollador. Las variables se clasifican dependiendo de los valores que pueden almacenar, entre las más básicas podemos encontrar las variables: int, String, char, float, boolean, etc. Las variables pueden ser de dos tipos, variables locales y variables globales, se le denominan variables locales a las variables que se utilizan únicamente en las estructuras void setup o void loop, una variable declarada en la estructura void setup no puede ser utilizada en la estructura void loop y viceversa, cabe señalar que las variables si puedes ser utilizadas en estructuras de control como if, while o for al mismo tiempo, siempre y cuando estas estructuras se encuentren dentro del void loop o void setup, si una variable es declarada dentro del void setup y se utiliza en una estructura if, también es posible utilizarla en una estructura for, siempre y cuando ambas estructuras se encuentren dentro del void setup, si la variable está declarada dentro del void setup y se utilizar dentro de una estructura if y se quiere utilizar dentro de una estructura for que se encuentra dentro del void loop, se producirá un error en la compilación del código ya que esta variable, as estar declarada dentro del void setup, es una variable local y no puede ser utilizada por el void loop. Las variables globales por el contrario se pueden utilizar en multiples estructuras del codigo, para poder utilizar una variable tanto en el void loop como en el void setup, se tiene que declarar fuera de cualquier estructura, es decir, al inicio del programa donde se declararn las librerias para que se inicialicen antes de llegar a las estructuras que las necesiten.

2.5 Tarjetas de desarrollo

Una tarjeta de desarrollo es una placa de circuito impreso (PCB) que contiene la configuración necesaria para hacer funcionar un microcontrolador.

Un microcontrolador es un circuito integrado que en su interior contiene una unidad central de procesamiento (CPU), unidades de memoria (RAM y ROM), puertos de entrada y salida y periféricos. Estas partes están interconectadas dentro del microcontrolador, y en conjunto forman lo que se le conoce como microcomputadora. Se puede decir con toda propiedad que un microcontrolador es una microcomputadora completa encapsulada en un circuito integrado (Ilustración 2-18). Toda microcomputadora requiere de un programa para que realice una función específica. Este se almacena normalmente en la memoria ROM. No está de más mencionar que sin un programa, los microcontroladores carecen de utilidad.

El propósito fundamental de los microcontroladores es el de leer y ejecutar los programas que el usuario le escribe, es por esto que la programación es una actividad básica e indispensable cuando se diseñan circuitos y sistemas que los incluyan. El carácter programable de los microcontroladores simplifica el diseño de circuitos electrónicos. Permiten modularidad y flexibilidad, ya que un mismo circuito se puede utilizar para que realice diferentes funciones con solo cambiar el programa del microcontrolador [31].

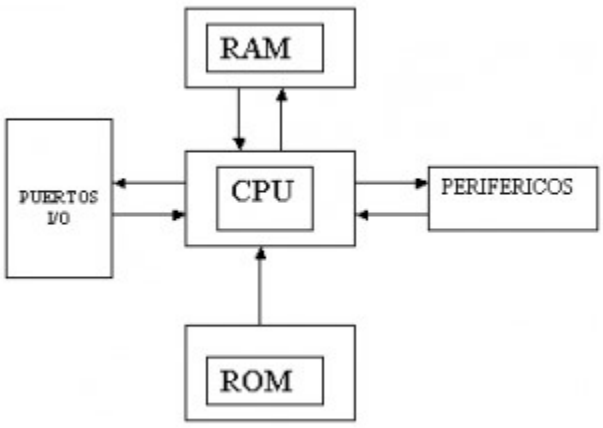


Ilustración 2-18 Esquema de un microcontrolador.

Una tarjeta de desarrollo, contiene conectores para comunicar dispositivos externos con los puertos del microcontrolador. Algunas placas contienen dispositivos adicionales integrados para ofrecer más opciones de desarrollo requeridas en proyectos más específicos. Existe una

gran variedad de tarjetas de desarrollo, cada una tiene características diferentes como en tamaño, costo, capacidad de procesamiento, dispositivos integrados, numero de conectores, etc., se deben tomar en cuenta las características del proyecto a desarrollar para poder elegir la tarjeta que más se adapte a las necesidades del proyecto.

2.5.1 Arduino Uno

La tarjeta Arduino Uno, es una tarjeta basada en un microcontrolador Atmega328 (Ilustración 2-19). Tiene 14 pines de entrada/salida digital (de los cuales 4 pueden ser utilizados para salidas PWM), 6 entradas análogas, un resonador cerámico de 16 MHz, un conector para USB tipo hembra, un Jack para fuente de Poder, un conector ICSP y un botón reset. Sus características técnicas son [32]:

- Microcontrolador: ATmega328p.
- Voltaje operativo: 5v.
- Voltaje de entrada (recomendado): 7 a 12v DC.
- Pines de entrada y salida digitales: 14 (de los cuales 6 son salidas PWM).
- Pines de entrada analógicas: 6.
- Memoria flash: 32KB (de los cuales 0.5KB son usados por el bootloader).
- SRAM: 2KB.
- EEPROM: 1KB.
- Velocidad de reloj: 16MHz.

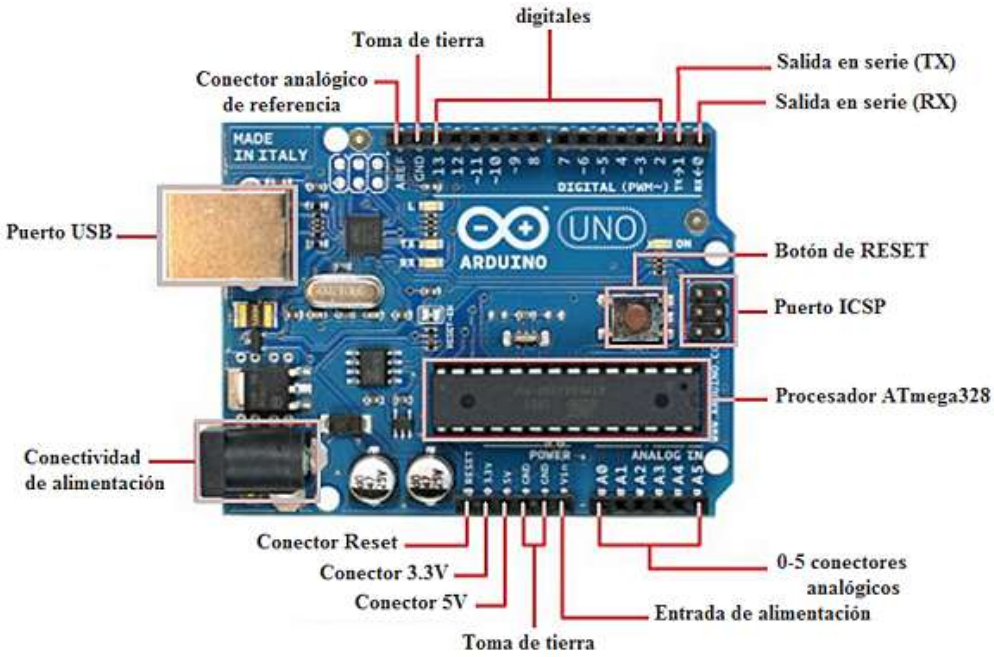


Ilustración 2-19 Arduino Uno.

2.5.2 Arduino Nano

Arduino Nano es una placa de desarrollo de tamaño compacto (Ilustración 2-20), basada en el microcontrolador ATmega328P. Tiene 14 pines de entrada/salida digital (de los cuales 6 pueden ser usando con PWM), 6 entradas analógicas, un cristal de 16Mhz, conexión Mini-USB, terminales para conexión ICSP y un botón de reseteo

Posee las mismas capacidades que un Arduino Uno, tanto en potencia del microcontrolador como en conectividad, solo se ve recortado en su conector USB y en el conector jack de alimentación [33]. características técnicas:

- Microcontrolador: Atmega328p.
- Voltaje operativo: 5v.
- Voltaje de entrada (recomendado): 5 a 12v DC.
- Pines de entrada y salida digitales: 14 (de los cuales 6 son salidas PWM).
- Pines de entrada analógicas: 6.
- Memoria flash: 32KB (de los cuales 0.5KB son usados por el bootloader).
- SRAM: 2KB.
- EEPROM: 1KB.
- Velocidad de reloj: 16MHz.

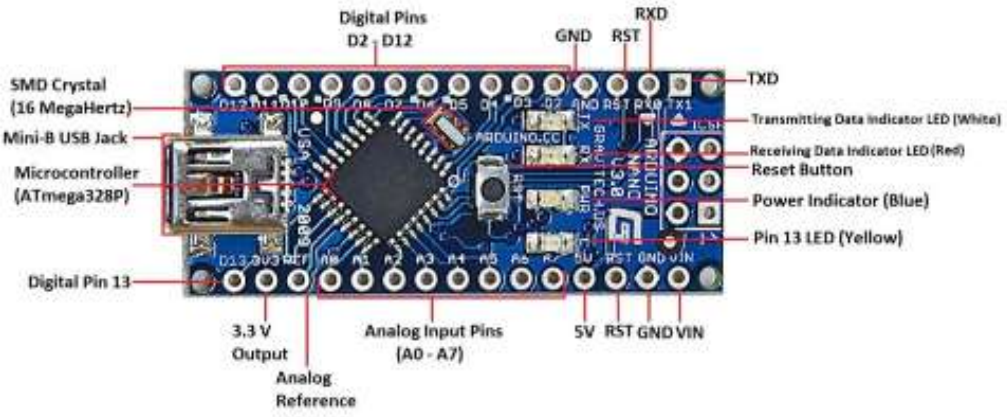


Ilustración 2-20 Arduino Nano.

2.5.3 Arduino Mega

Arduino Mega (ilustración 2-21) es una tarjeta de desarrollo de Hardware libre construida con el microcontrolador Atmega 2560. Forma parte del proyecto Arduino que involucra una comunidad internacional dedicada al diseño y manufactura de placas de desarrollo de Hardware.

El Arduino Mega tiene 54 pines de entrada/salida, de los cuales exactamente 14 de ellos pueden ser utilizados como salidas de PWM (Modulación por ancho de pulso), cuenta con otras 16 entradas analógicas y 4 UARTs (puertos serial).

En cuanto a la velocidad del microcontrolador podemos decir que cuenta con un Cristal de 16MHz y una memoria Flash de 256K. Maneja un rango de voltaje de entrada de entre 7 y 12 volt, se recomienda una tensión de entrada de 9 Volt. Características técnicas [34]:

- Microcontrolador: Atmega2560.
- Voltaje operativo: 5v.
- Voltaje de entrada: 7 a 12v.
- Voltaje de entrada (limite): mínimo 6v, máximo 20v.
- Pines digitales de entrada y salida: 54 (de los cuales 14 proveen salida PWM).
- Pines analógicos de entrada: 16.
- Memoria flash: 256KB (de los cuales 8KB son utilizados para el bootloader).
- SRAM: 8KB.
- EEPROM: 4KB.
- Velocidad de reloj: 16MHz.

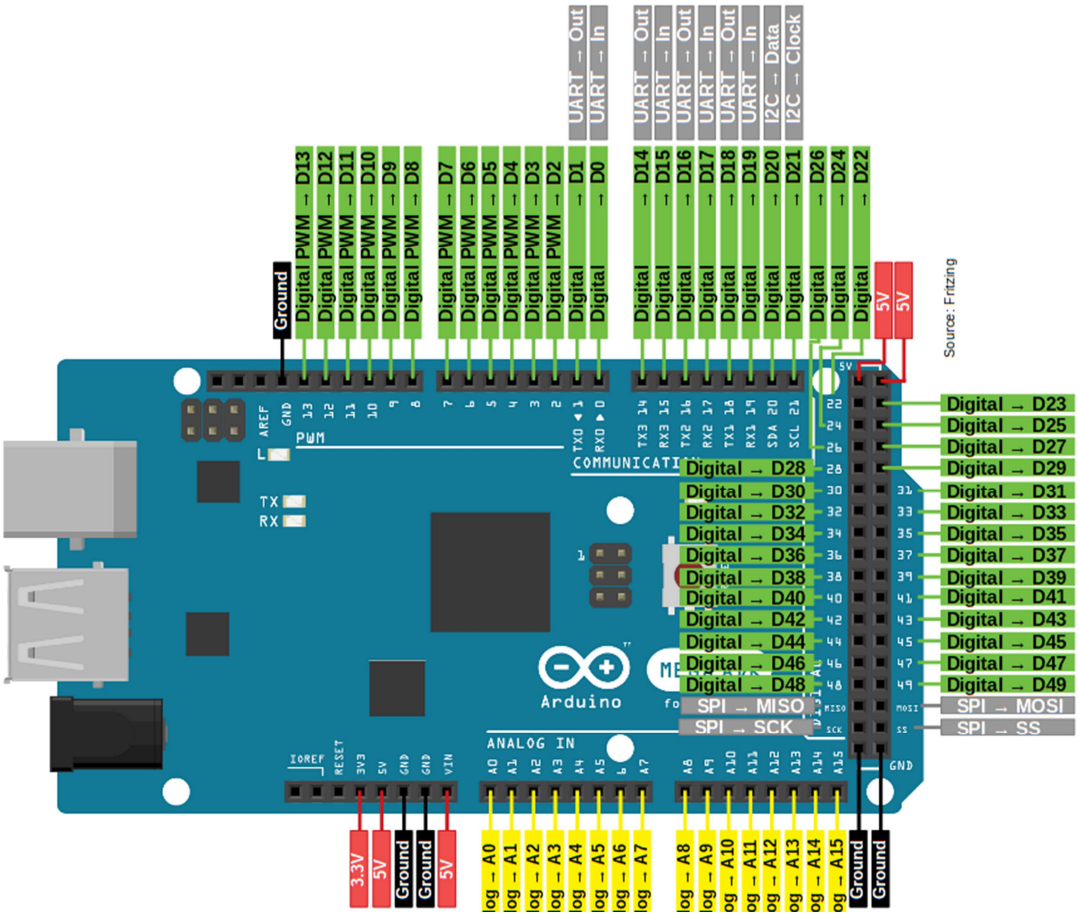


Ilustración 2-21 Arduino Mega.

2.5.4 NodeMCU ESP32

El Módulo Wifi + Bluetooth ESP32 (Ilustración 2-22) incorpora el potente módulo ESP-WROOM-32 que integra Wi-Fi, Bluetooth y Bluetooth BLE, ideal para desarrollar productos de internet de las cosas. La integración de Bluetooth LE y Wi-Fi permite una amplia gama de aplicaciones, el uso de Wi-Fi permite una comunicación de mediano alcance y conectarse a una red LAN y/o a través de un Router con conexión a Internet, mientras que el Bluetooth nos permite conectarse directamente a otro dispositivo como un celular.

ESP32 integra un amplio conjunto de periféricos como sensores táctiles capacitivos, sensores Hall, amplificadores de bajo nivel de ruido, interfaz para SD, Ethernet, SPI, UART, I2S. e I2C. sus principales características son [35]:

- Voltaje operativo: 2.2 a 3.6v
- Procesador: Tensilica Xtensa 32-bit LX6.
- ROM: 448KB.
- SRAM: 520KB.
- SRAM en RTC: 16KB.
- Frecuencia de reloj: 240MHz.
- Wi-Fi: 802.11 b/g/n/e/I (802.11n @ 2.4 GHz hasta 150 Mbit/s).
- Bluetooth: v4.2 BR/EDR and Bluetooth Low Energy (BLE).
- Memoria flash: 4MB.
- Pines digitales: 24 (algunos solo de entrada).
- Pines Analógicos: 2 (algunos pines soportan un amplificador de ganancia programable).
- UART: 2.
- Seguridad: Estandares IEEE 802.11 incluyendo WFA, WPA/WPA2 and WAPI, 1024-bit OTP, up to 768-bit for customers y Aceleración criptográfica por hardware: AES, HASH (SHA-2), RSA, ECC, RNG.

Ilustración 2-22 NodeMCU ESP32.

2.6 LabVIEW

LabVIEW (acrónimo de Laboratory Virtual Instrument Engineering Workbench), LabVIEW (Ilustración 2-23) es un software de ingeniería de sistemas que requiere pruebas, medidas y control con acceso rápido a hardware e información de datos. LabVIEW ofrece un enfoque de programación gráfica que le ayuda a visualizar cada aspecto de su aplicación, incluyendo configuración de hardware, datos de medidas y depuración. Esta visualización hace que sea más fácil integrar hardware de medidas de cualquier proveedor, representar una lógica compleja en el diagrama, desarrollar algoritmos de análisis de datos y diseñar interfaces de usuario personalizadas [36]. Este software fue desarrollado por National Instruments, el enfoque de National Instruments basado en plataforma, combina software y hardware modular para ayudarle a resolver retos de ingeniería complejos. Los productos de National Instruments son usados en una gran variedad de industrias, las más destacadas son: semiconductor, automotriz, aeroespacial y defensa, electrónica, energía, académica e investigación, sistemas inalámbricos, maquinaria industrial.

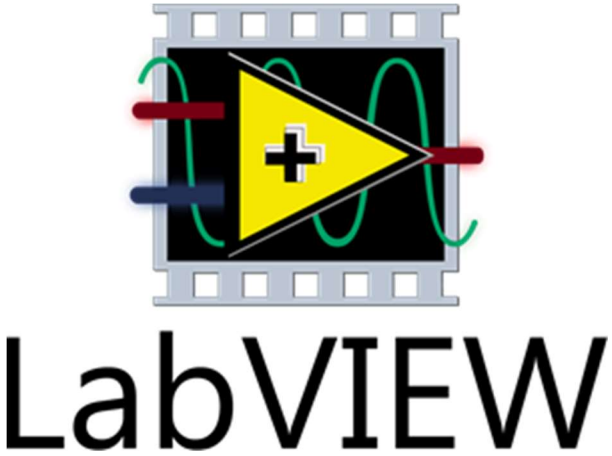


Ilustración 2-23 LabVIEW.

2.6.1 Panel frontal de LabVIEW

En este apartado se pueden observar indicadores tales como gráficos de señales o datos, tablas de datos, señalizadores, mensajes, etc., que proporcionan información de manera gráfica, también se pueden encontrar varios controles con los que el usuario puede interactuar para hacer una tarea específica o para establecer una configuración determinada (Ilustración 2-24).

Dentro de los indicadores necesarios para la realización de este sistema se pueden encontrar: *waveform chart, combo box, button, LED, tab control, waveform graph, time stamp, bolean button, visa resource name.*

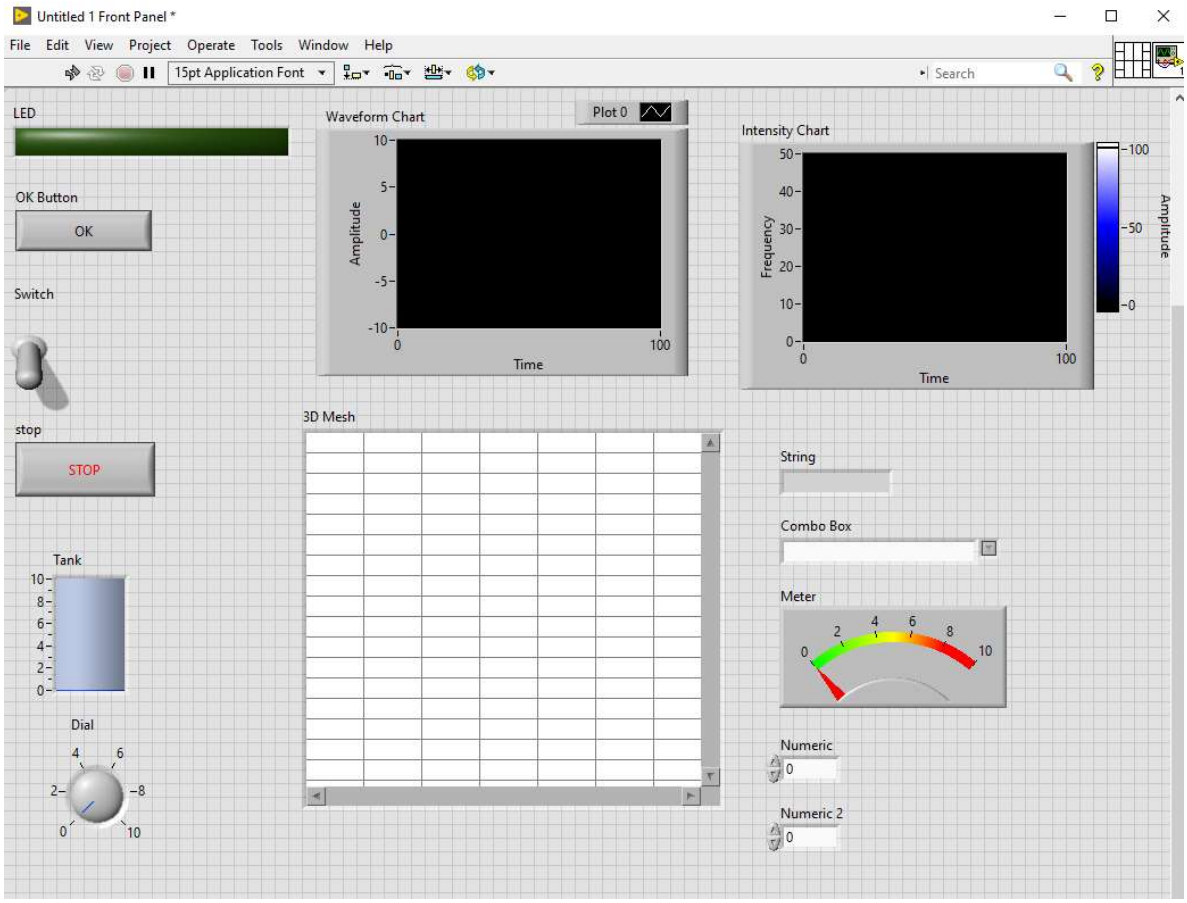


Ilustración 2-24 Panel frontal LabVIEW.

2.6.1.1 Tab control

Este más que un indicador, es un contenedor en el cual se pueden acomodar varios indicadores en un área determinada, la principal función de este contenedor es proporcionar un orden y clasificar los indicadores en diferentes áreas.

2.6.1.2 Combo box

Esta función es un menú desplegable en el que se pueden programar varias opciones y que el usuario final puede seleccionar fácilmente.

2.6.1.3 Waveform chart

Este es un indicador numérico que dibuja una gráfica con los valores que se le asignen, tiene 2 ejes, el eje x que corresponde a la amplitud de los valores que se le vayan introduciendo y el eje y, que corresponde al tiempo. Este indicador solo es capaz de graficar un valor a la vez, no puede graficar múltiples valores al mismo tiempo. Este indicador es muy útil para graficar los valores obtenidos por un sensor o cualquier otro dispositivo.

2.6.1.4 Waveform graph

Es un indicador que gráfica valores numéricos en dos ejes, el eje x y el eje y, la diferencia con el waveform chart es que este indicador si es capaz de graficar más de un valor al mismo tiempo, lo cual es muy útil para graficar valores numéricos que se encuentren almacenados.

2.6.1.5 Table

Es una tabla de celdas similar a Excel donde ponemos tener n cantidad de filas, así como n cantidad de columnas, puede almacenar datos números y texto. Este indicador es muy utilizado para la implementación de bases de datos.

2.6.1.6 Time stamp control

Es un calendario desplegable con el cual el usuario puede seleccionar distintas fechas sin necesidad de introducirlas manualmente, esta herramienta es muy útil para buscar datos almacenados por fechas.

2.6.1.7 Button

Es un botón con el que el usuario puede interactuar para activar la función o funciones que tenga programadas, funciona con la lógica *normalmente abierto*, es decir, que su valor se mantiene en 0, hasta que se presiona su valor cambia a 1 e inmediatamente regresa su valor inicial de 0.

2.6.1.8 Bolean button

Es un botón con el que el usuario puede interactuar para activar la función o funciones que tenga programadas, a diferencia del botón con lógica *normalmente abierto*, este botón mantiene el estado que se le asigne, cuando se inicia el programa tomara el valor inicial de 0, si se presiona tomara el valor de 1 y mantendrá ese valor hasta que se vuelva a presionar y volverá a su valor original de 0.

2.6.1.9 LED

Es un indicador lumínico con el que el usuario no puede interactuar, pero es muy útil para indicar el estado de algunos parámetros del sistema, por ejemplo: si alguna función esta activada o desactivada, si algo en específico está funcionando correctamente o si algo está fallando, puede tomar 2 colores diferentes dependiendo del propósito que se le asigne.

2.6.1.10 Visa resource name

Es un menú desplegable que contiene los puertos seriales disponible que se encuentren activos y conectados a la computadora sobre la cual se esté ejecutando la interfaz, mediante este menú se puede seleccionar el puerto al que esté conectado el hardware para conectar la interfaz gráfica con la obtención de datos. Para poder funcionar se necesita de un complemento para LabVIEW llamado *NI visa*.

2.6.2 Diagrama de bloques de LabVIEW

En este apartado se programa todo el funcionamiento del sistema, esta programación se hace a través de bloques, los cuales son elementos con una función específica y que en conjunto con otros bloques dan funcionalidad al sistema completo. en este apartado se utilizan estructuras de programación, funciones matemáticas, constantes, cables de conexión, comparadores, etc., con los cuales se manipulan todos los datos para darle funcionalidad a todo el sistema. Una vez concluida la aplicación, el usuario final no tiene acceso a este apartado (ilustración 2-25).

Dentro de las principales herramientas que se pueden encontrar en este apartado se encuentran: *while loop, case structure, visa configure serial port, visa read, scan from String, visa write, visa close, time wait, number to fractional String, bundle, number to decimal String, concatenate String, wave form chart, time stamp control, time stamp constant, combo box, string constant, numeric control, for loop, wave form graph, convert to dynamic data, property node, visa resource name, string to path, feedback node, etc.*

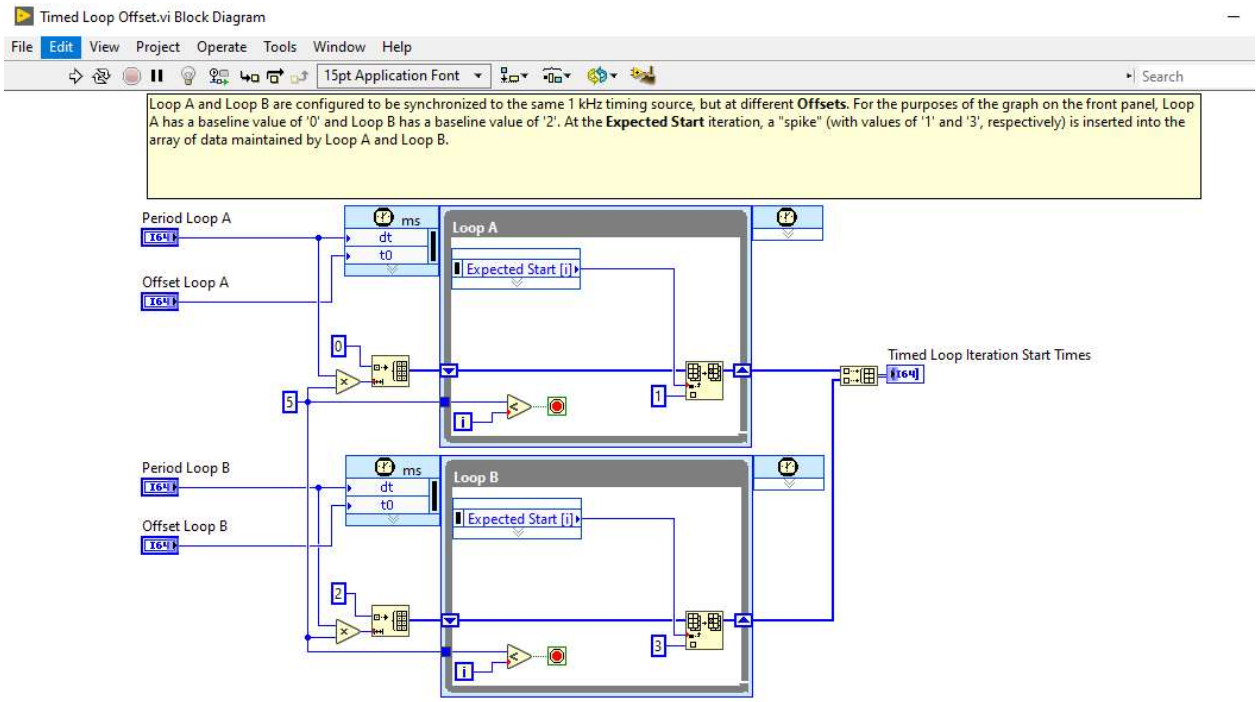


Ilustración 2-25 Diagrama de bloques LabVIEW

2.6.2.1 While loop

Es una estructura de control que produce un bucle, este bucle siempre va a ejecutar las funciones que se encuentren dentro de la estructura mientas la señal de paro se mantenga en 0, para detener este bucle, la señal de paro se debe establecer en 1, esto se puede lograr con un botón, por ejemplo, aunque se pueden utilizar múltiples métodos para lograr este propósito.

2.6.2.2 For loop

Es una estructura de control que produce un bucle controlado, el número de veces que se repite el bucle se puede controlar según sea conveniente para el usuario, esta estructura cuenta con 2 parámetros de control, "*N*" que corresponde al número de veces que se va a repetir el bucle, *"i"* que corresponde al incremento que se le aplica a *"N"* tras cada iteración.

2.6.2.3 Case structure

Es una estructura que permite ejecutar una de varias funciones, según se le indique. Esta estructura cuenta con 1 parámetro de control llamado *case selector,* que es el encargado de seleccionar el número de función deseada, este selector puede ser cambiado por el usuario

desde el panel frontal o se pueden emplear una gran cantidad de métodos más complejos para realizar esta función.

2.6.2.4 Visa resource name

Este bloque representa al menú *visa resource name* que se encuentra en el panel frontal, no cuenta con ningún parámetro de configuración, solo cuenta con una terminal de salida por la cual envía la información que obtiene del puerto serial seleccionado

2.6.2.5 Bytes at port

Mediante este bloque se pueden obtener los parámetros de comunicación serial, cuenta con más de 10 terminales de conexión mediante las cuales se puede obtener datos y parámetros del puerto serial seleccionado por el menú *visa resource name*, a pesar de la gran cantidad de terminales con las que cuenta, las más importantes y las más usadas son: *bytes at port, baud rate, allow transmit, parity. Bytes at port* arroja el número de bytes que se encuentran en el puerto serial al momento de la comunicación, *baud rate* indica la tasa de baudios a la que está trabajando el puerto serial seleccionado, *allow transmit* solo puede arrojar 2 valores posibles, cuando el valor es 0 indica que no se ha podido establecer comunicación con el puerto serial seleccionado y cuando el valor es 1 indica que si ha conexión con el puerto serial seleccionado, *parity* indica los bits de paridad, si se usa paridad par o paridad impar, no es común utilizar bits de paridad y no se encuentran muchos sistemas que los utilicen, por lo que en la realización de este proyecto no es un parámetro utilizado.

2.6.2.6 Visa read

Este bloque permite leer un numero especifico de datos provenientes del puerto serial seleccionado y lo envía en una cadena en formato String cuenta con 7 terminales de conexión de las cuales 4 son reservadas para la comunicación con otros bloques que funcionan con comunicación serial, específicamente *visa resource name* es utilizado por este bloque para tener comunicación con el puerto serial, *visa resource name out* permite a otro bloque que funcione con comunicación serial conectarse a este bloque y conectare al mismo puerto serial, *error in* y *error out* son utilizados para saber si se ha producido algún error en la comunicación serial. Del resto de terminales *byte count* es utilizado para saber el número de bytes que va a leer del puerto serial, este dato se puede introducir de forma manual o también lo puede proporcionar el bloque *property node,* la terminal *read buffer,* es la salida de datos

obtenida del puerto serial, estos datos lo proporciona en formato String y por último, la terminal *retun count,* proporciona el número de datos leídos, esta terminal no es utilizada comúnmente, aunque puede ser útil en algunas aplicaciones.

2.6.2.7 Visa configure serial port

Este bloque proporciona la configuración del puerto serial al resto de bloques que obtienen o envían datos al puerto serie, cuenta con 12 terminales de conexión, de las cuales la mayoría no se utilizan salvo en aplicaciones que así lo requieran, las más importantes son: *visa resource name, visa resource name out, error in, error out, baud rate,* de estas terminales, las primeras cuatro tienen la misma función en todos los bloques, por lo que su función es la misma que en el bloque visa read y también es la misma que en el resto de bloques que utilizan comunicación serie. Y por último la terminal *baud rate,* establece la tasa de baudios de la comunicación serie, este dato es proporcionado por el bloque *property node*.

2.6.2.8 Scan from String

Este bloque permite tomar un trozo de la cadena de datos proporcionada por el bloque *serial read,* esto se hace para separar los datos que se encuentran unidos en la cadena y enviarlas individualmente a una salida, una vez que ya obtuvo el trozo de la cadena que se indicó, envía el resto de la cadena por una terminal de salida a la que se pueden conectar otros bloques. Cuenta con n cantidad de terminales, el número de terminales, la cantidad de terminales depende de la cantidad de salidas que se asignen. Dentro de las terminales más importantes se encuentran: *input string, format String, remaining string.* La terminal "*input string"* es por donde ingresa la cadena de datos obtenida por el bloque *serial read,* la terminal *format string* es muy importante ya que de esta terminal depende que tipo de valores se van a obtener en cada una de las salidas individuales, recordemos que a este bloque la cadena de datos ingresa en formato *string,* es decir, es texto y para convertir este formato de texto a un formato número, es necesario indicárselo. existe una gran variedad de formatos que soporta este bloque, dentro de los cuales, los más importantes y los más utilizados son: valor numérico entero (*%d*) y formato numérico con punto decimal o número con punto flotante (*%f*), y por último la terminal *remaining string* se encarga de enviar lo que sobra de la cadena de datos después de que este bloque haya tomado el número valores indicados.

2.6.2.9 Visa write

Al contrario del bloque *visa read,* este bloque permite enviar una cadena de datos en formato string al puerto serie. Cuenta con 5 terminales de conexión, las cuales son: *visa resource name, visa resource name out, error in, error out, write buffer.* De las cuales las primeras 4 terminales tienen las mismas funciones en todos los bloques que utilicen la comunicación con el puerto serie. Por último, la terminal *write buffer* es la entrada de datos a enviar al puerto serie.

2.6.2.10 Visa close

Este bloque termina la comunicación con el puerto serie, este bloque es utilizado para cerrar de forma adecuada la comunicación sin producir errores de sincronización, solo cuenta con 3 terminales de comunicación: *visa resource name, error in, error out,* la función de estas terminales es la misma que en los demás bloques mencionados anteriormente.

2.6.2.11 Time wait

Este bloque es un temporizador que detiene el flujo del programa por Unos breves instantes de tiempo que se miden en milisegundos, al poner este bloque dentro de un ciclo while loop, por ejemplo, lo que se consigue es que entre cada iteración del bucle haya un espacio de tiempo que se le indique mediante este bloque. Cuenta con solo una terminal de conexión mediante la cual se introduce un valor numérico que indica el tiempo de retardo.

2.6.2.12 Number to fractional String

Este bloque permite convertir un valor numérico con punto decimal a formato string, es decir en texto ASCCI, esta función es indispensable si se necesitan enviar valores numéricos mediante el puerto serial, ya que el bloque *visa write* que permite realizar esta función, solo admite datos en formato string.

2.6.2.13 Number to decimal String

Este bloque permite convertir un valor numérico entero a formato string, es decir en texto ASCCI, esta función es indispensable si se necesitan enviar valores numéricos mediante el puerto serial, ya que el bloque *visa write* que permite realizar esta función, solo admite datos en formato string.

2.6.2.14 Bundle

Este bloque permite unir varios datos con diferentes formatos en una sola cadena sin necesidad de alterar el formato original de los datos, el número de entradas depende del número de datos que se necesiten agrupar y solo cuenta con una salida mediante la cual envía los datos agrupados.

2.6.2.15 Concatenate

Este bloque permite unir varios datos en formato string en una sola cadena, este bloque solo admite datos en formato string, es decir que solo admite texto.

2.6.2.16 Numeric comparison

Este no es un bloque, es una agrupación de funciones que permiten hacer comparaciones numéricas, en este apartado podemos encontrar funciones como: *mayor que, menor que, mayor o igual a, menor o igual a, igual a, mayor a 0, menor a 0, compuerta AND, compuerta OR, compuerta NOT, compuerta XOR, etc.*

2.6.2.17 Time stamp control

Este es un bloque que representa el *time stamp control* que se encuentra en el panel frontal, este bloque no cuenta con ningún parámetro de control o de configuración, solo cuenta una terminal de conexión por el cual salen los datos introducidos por el usuario desde el panel frontal.

2.6.2.18 Time stamp constant

Este bloque permite al desarrollador fijar una fecha que no se puede modificar desde el panel frontal.

2.6.2.19 Button

Este es un bloque que representa el *button* que se encuentra en el panel frontal, este bloque no cuenta con ningún parámetro de control o de configuración, solo cuenta una terminal de conexión de salida que proporciona datos acerca del estado del botón, si está presionado el dato que arroja es *true* y si no está presionado el dato que arroja es *false.*

2.6.2.20 Constant string

Este bloque permite al desarrollador fijar cadenas de texto que no se pueden modificar desde el panel frontal.

2.6.3 Paleta de controles de LabVIEW

Esta es una ventana emergente en la que se pueden elegir todos los elementos que el desarrollador tiene disponibles para construir el sistema. Esta ventana emergente es diferente para el diagrama de bloques y para el panel frontal. En cuanto al diagrama de bloques, esta ventana incluye elementos de programación (Ilustración 2-26) como: estructuras de programación, funciones matemáticas, elementos para manipular datos y señales, temporizadores, etc. y en cuanto al panel frontal, esta ventana incluye todos los elementos con los que el usuario final puede interactuar (ilustración 2-27) como: señalizadores, controles numéricos, botones, tablas de datos, etc.

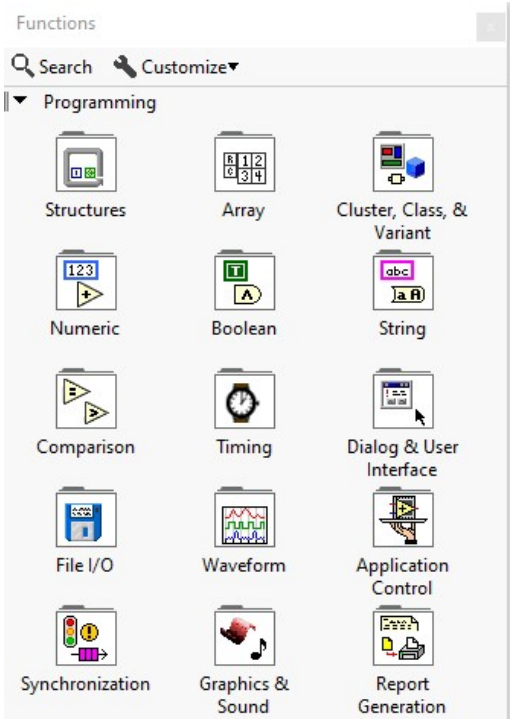


Ilustración 2-26 Controles para diagrama de bloques.

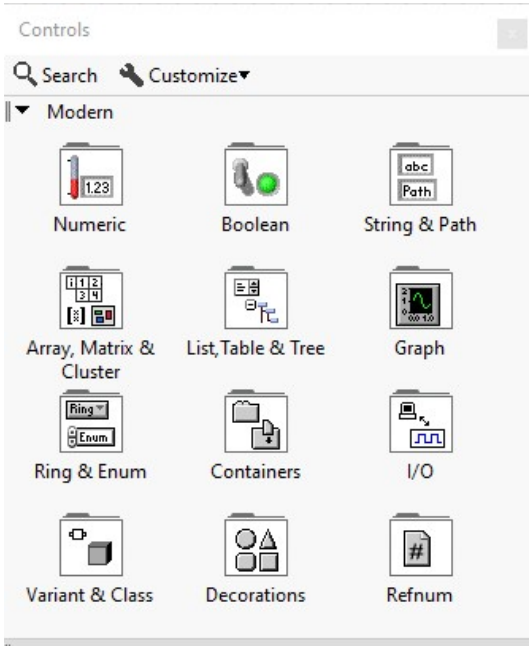


Ilustración 2-27 Controles para el panel frontal.

2.6.4 Complementos para LabVIEW

Además del software stock proporcionado National Instruments, se pueden adquirir complementos que proporcionan herramientas adicionales que permiten la implementación de datos más complejos, muchos de estos complementos proporcionan la compatibilidad con diferentes sistemas de software de terceros, para la realización de este proyecto, se necesitan los complementos *NI visa* y *Database connectivity toolkit.*

2.6.5 NI visa

Este es un complemento gratuito que se puede descargar desde la página principal de National Instruments, este complemento es necesario para poder obtener y enviar datos por el puerto serial.

2.6.6 Database connectivity toolkit

Este complemento no es gratuito, la función principal de este complemento es permitir la conexión entre bases de datos y LabVIEW.

Este complemento agrega nuevos bloques de función, dentro de los principales se encuentran: *Database open conection, Database close conection, Database insert data, Database execute query, Database fetch recordset data, Database variant to data.*

2.6.6.1 Database open connection

Este bloque permite establecer conexión con un archivo de Microsoft Access para crear una base de datos, este bloque cuenta con 8 terminales de conexión: *connection information, conection reference, error in, error out, User ID, password, connection information, connection timeout, prompt.* Mediante la terminal *connection information* se le indica la ruta en la que se encuentra guardo el archivo de Microsoft Access en el que se van a guardar los datos, mediante la terminal *connection reference* se pude conectar a otros bloques para proporcionarles la ruta donde se encuentra alojada la base de datos, las terminales *error in* y *error out,* permiten indicar si no se ha producido algún error de comunicación con la base de datos, las terminales *User ID* y *password* permiten restringir el acceso a solo los usuarios que dispongan de una contraseña que se establece por el desarrollador.

2.6.6.2 Database insert data

Este bloque permite escribir datos en el archivo Access, para poder hacerlo dispone de 10 terminales, aunque esencialmente solo se necesita de 7 de ellas: *connection information, connection reference, error in, error out, table, créate table, data.* Las primeras 4 terminales tienen la misma función en todos los bloques de esta categoría, es decir, que su función es la misma que la descrita en el bloque *Database open connection,* mediante la terminal *table* se le indica el nombre que se le asigna a la tabla en la que va a almacenar los datos, la terminal *créate table* solo admite 2 valores: verdadero y falso, cuando el valor es verdadero permite crear una tabla con el nombre que se le indique en la terminal *table* y cuando el valor es falso no permite crear ninguna tabla, la terminal *data* permite introducir los datos que se van a almacenar en la base de datos, deben de estar todos agrupados, no es posible introducirlos individualmente.

2.6.6.3 Database execute query

Este bloque permite ejecutar códigos SQL dentro de la base de datos, esto es muy útil en sistemas en donde se necesite interactuar con los datos almacenados en la base de datos, ya sea mediante consultas, ordenar datos, copiar datos, etc.

Este bloque pertenece a la categoría de herramientas avanzadas y su funcionamiento es muy complejo ya que entra directamente con la gestión SQL para bases de datos

2.6.6.4 Database fetch recordset data

Este bloque permite recuperar datos almacenados en la base de datos y convierte cada dato al formato que tenía originalmente e LabVIEW antes de ser almacenado en la base de datos. Cuenta con 5 terminales: *recordset reference, recordset out, error in, error out, recordset data.* La terminal *recordset reference* es la entrada de datos provenientes del bloque *Database execute query* una vez que ya realizo la ejecución del comando SQL que se le haya signado, las terminales *error in* y *error out* tienen las mismas funciones descritas en anteriores bloques, por último, la terminal *recordset data* contiene los datos recuperados de la base de datos.

2.6.6.5 Database variant to data

Este bloque permite convertir un arreglo de datos recuperado de una base de datos en un formato especifico. Cuenta con 5 terminales de conexión: *error in, error out, type, Database variant, data.* Las terminales *error in* y *error out* conservan la función anteriormente descrita en otros bloques, la terminal *Database variant* es la entrada de datos al bloque, la terminal *type* permite especificar el formato al cual se van a convertir los datos y por último la terminal *data* es la salida de datos convertidos al formato que se le haya especificado

2.7 SQL

En la actualidad el uso de las bases de datos se puede encontrar en casi cualquier parte de la vida cotidiana, ya que son muy útiles para poder almacenar datos históricos, datos estadísticos, datos económicos, datos personales, etc. aunque en la actualidad la mayoría de las bases de datos se encuentran digitalizadas y almacenadas en servicios de nube, el concepto de bases de datos ya se utilizaban antes de la aparición de este tipo de servicios y tecnologías aunque no se les denominaba de esta forma, los ejemplos más comunes son las bibliotecas, cinetecas, fonotecas, etc. que se caracterizan por ser lugares físicos en lo que se almacenan diversos tipos de datos para poder preservarlos y consultarlos cuando sea necesario. Con la aparición de las bases de datos digitales se hizo necesario la creación de herramientas para poder interactuar con los datos almacenados, una de las herramientas más utilizadas para este propósito es el lenguaje SQL.

SQL es un acrónimo en inglés para Structured Query Language. Un Lenguaje de Consulta Estructurado. Un tipo de lenguaje de programación que permite manipular y descargar datos de una base de datos. Tiene capacidad de hacer cálculos avanzados y álgebra. Es utilizado en la mayoría de empresas que almacenan datos en una base de datos. Ha sido y sigue siendo el lenguaje de programación más usado para bases de datos relacionales. [37]

Esta herramienta es muy compleja y se le puede dar una gran cantidad de usos en combinación con otros sistemas, por lo que no se profundizara en todas las funciones de este lenguaje, solo en las más básicas las cuales son:

- SELECT: permite seleccionar los datos a consultar
- WHERE: permite seleccionar que filtro aplicar a los datos que se van a consultar
- INSERT: permite insertar datos en la base de datos
- DELETE: permite eliminar datos de la base de datos
- UPDATE: permite actualizar datos

2.8 Protocolos de comunicación digital

Un protocolo, se define como las reglas para la transmisión de la información entre dos puntos. Un protocolo de comunicación de datos, es un conjunto de reglas que gobierna el intercambio ordenado de datos dentro de la red. Los elementos básicos de un protocolo de comunicaciones son: un conjunto de símbolos llamados conjunto de caracteres, un conjunto de reglas para la secuencia y sincronización de los mensajes construidos a partir del conjunto de caracteres y los procedimientos para determinar cuándo ha ocurrido un error en la transmisión y como corregir el error. Para que exista comunicación en ambos puntos al extremo de un canal se deben emplear la misma configuración de protocolos [38].

En las comunicaciones digitales los caracteres que se transmiten se denominan bits, un bit es la unidad más básica de información en los sistemas digitales, estos caracteres solo puede adoptar 2 valores: 1 lógico y 0 lógico, estos valores se obtienen mediante un nivel de voltaje, los niveles de voltaje correspondientes para cada valor lógico (Uno o cero lógicos) varían dependiendo del protocolo de comunicación, pero comúnmente el Uno lógico corresponde a un nivel de voltaje de 5v y el cero lógico corresponde a un nivel de voltaje de 0v. estos bits se transmiten a través de un canal físico que comúnmente se le llama hilo, línea, canal, pero es un cable eléctrico.

Existen varios tipos de comunicaciones digitales: alámbricas e inalámbricas, para el desarrollo de este proyecto solo se utilizarán las comunicaciones alámbricas, dentro de las cuales existen dos variantes: la comunicación digital paralela y la comunicación digital serie, cada una de las opciones tiene ventajas y desventajas, para poder elegir la más adecuada, es necesario saber las características de cada sistema y en base a esa información, elegir la que más se adecue a las necesidades de cada sistema.

2.8.1 Comunicación digital paralela

Este tipo de comunicación fue la primera en utilizarse en los sistemas digitales, debido a que es sencilla y la transmisión de datos es muy rápida. En este tipo de comunicación se utiliza un hilo para transmitir cada bit, es decir, si queremos transmitir un conjunto de datos de un byte (1 byte es igual a 8 bits) se necesitan 8 cables para poder transmitir la información, (Ilustración 2-28) si queremos transmitir un bus de datos de un nibble (1 nibble es igual a 4 bits) se necesitan 4 cables para poder transmitir la información.

La principal desventaja de este tipo de comunicación, es que se necesita un número elevado de cables para poder transmitir los datos, lo cual provoca que el sistema tenga que ser más robusto y se eleve la cantidad de materiales para poder fabricarlos. En la actualidad los dispositivos digitales tienden a ser de tamaño más compacto de lo que eran las generaciones anteriores, esta característica es inviable utilizando comunicación paralela debido a que los dispositivos actuales procesan una gran cantidad de datos, y para realizar todas las comunicaciones con los distintos componentes de un dispositivo, se necesitaría una gran cantidad de cables de transmisión y recepción de datos para que la comunicación paralela funcione.

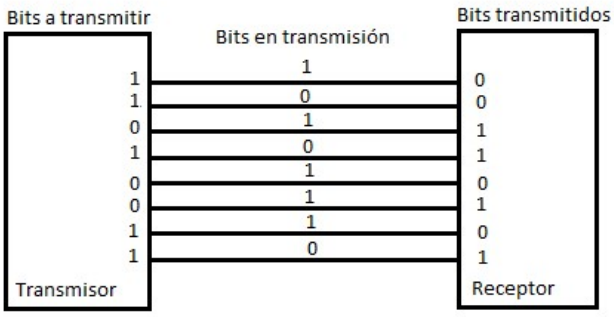


Ilustración 2-28 Comunicación digital paralela.

2.8.2 Comunicación digital serial

En las comunicaciones seriales, a diferencia de las comunicaciones paralelas, todos los bits de información se transmiten a través de un solo hilo, todos los bits pasan por el mismo hilo, (Ilustración 2-29) lo cual tiene la ventaja de que se pueden transmitir grandes cantidades de bits sin la necesidad de tener un número elevado de hilos, se reduce el tamaño de los dispositivos y reducen los materiales utilizados en la fabricación.

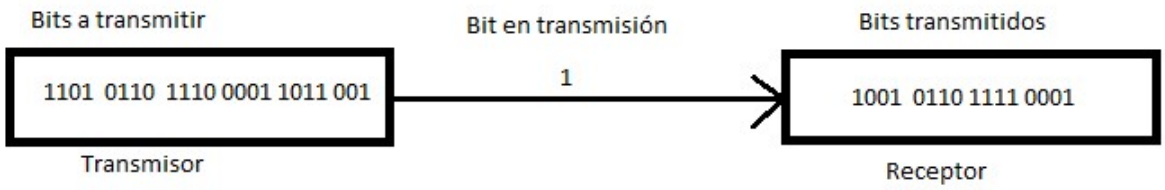


Ilustración 2-29 Comunicación digital serial.

Este tipo de transmisión presenta varias desventajas, una de las desventajas, es la velocidad de transmisión, ya que realiza de forma más lenta respecto a la comunicación paralela, debido a que en la comunicación serial solo existe un único canal de transferencia y miles de bits deben utilizar el mismo canal, y en la comunicación paralela existen múltiples canales que se pueden utilizar para realizar la transferencia, actualmente este problema se minimiza utilizando altas velocidades de transmisión, otra de las desventajas, es la complejidad de los sistemas, en este tipo de comunicaciones es más complicado que la transferencia de bits se realice de manera ordenada, debido a que las transferencias se realiza de manera muy rápida y se transmiten miles de bits en intervalos muy cortos de tiempo, si no se logra una sincronización de los procesos, la transferencia se realizará de manera errónea alterando la información transmitida, para corregir este problema se utilizan varias técnicas para sincronizar al transmisor y al receptor.

Los protocolos de comunicación serial se dividen en dos grupos: protocolos seriales síncronos y protocolos seriales asíncronos.

2.8.2.1 Protocolos síncronos

Este tipo de protocolos utiliza un señalizador que indica al transmisor cuando es momento adecuado para transmitir cada Uno de los bits, de igual forma pasa con el receptor. A este señalizador se le conoce como reloj, el reloj es una sucesión de Unos y ceros o lo que es lo mismo, una sucesión de pulsos o de niveles altos y niveles bajos. El reloj necesita un hilo exclusivo diferente del hilo de datos, se identifica como CLK, Clock, SCL, etc (Ilustración 2-30).

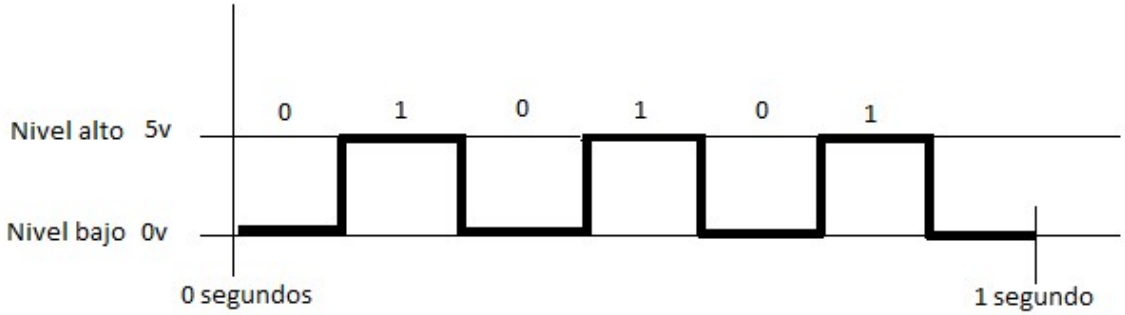


Ilustración 2-30 Reloj de sincronización.

Una de las características principales del reloj, es la frecuencia de operación, se mide en Hertz (Hz) y es la cantidad de pulsos que se producen en un segundo, por lo que, a mayor frecuencia, mayor es el número de pulsos producidos en un segundo. Por ejemplo, si se tiene un reloj con una frecuencia de 10Hz, significa que en un segundo el reloj produce 10 pulsos, si se quiere calcular el tiempo en el que se produce un pulso completo, se puede emplear la siguiente ecuación 2.1.

$$t = \frac{1}{F} \qquad \text{Ec 2.1}$$

En donde "t" es el tiempo medido en segundos que tarda el reloj en producir un pulso y "F" es la frecuencia medida en hertz. Haciendo el cálculo para 10Hz se obtiene el resultado de 0.1 segundos.

Los dispositivos actuales emplean frecuencias de varios MHz (mega hertz), incluso algunos dispositivos emplean frecuencias de GHz (giga hertz).

Para que un pulso se pueda usar para señalizar el inicio y el fin de una transferencia, cada pulso cuenta con dos flancos, un flanco de subida y un flanco de bajada, un flanco es la transición que hay cuando se presenta un cambio de nivel lógico, el flanco de subida es

cuando se pasa de un nivel bajo a un nivel alto y el flanco de bajada se produce cuando se pasa de un nivel alto a un nivel bajo, estos cambios de nivel son empleados por el transmisor y receptor para identificar el inicio y el fin de la transferencia. Se pueden emplear tanto los flancos de subida como los de bajada según sea conveniente para el sistema (ilustración 2-31).

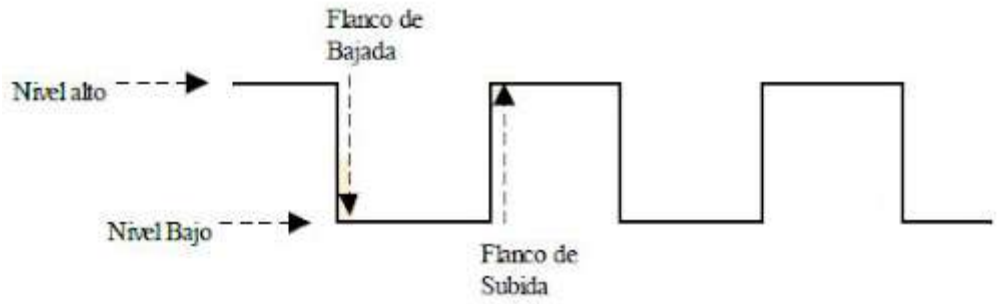


Ilustración 2-31 Flancos de un pulso digital.

Ya que la transmisión de cada bit depende directamente de la señal de reloj, se puede decir que la cantidad de pulsos proporcionados por el reloj establece la velocidad de transmisión de datos, es decir, que cuanto mayor sea la frecuencia de reloj, más bits de datos se pueden transmitir en un segundo, lo cual produce que la transferencia de una gran cantidad de bits se realice en un menor tiempo. (Ilustración 2-32)

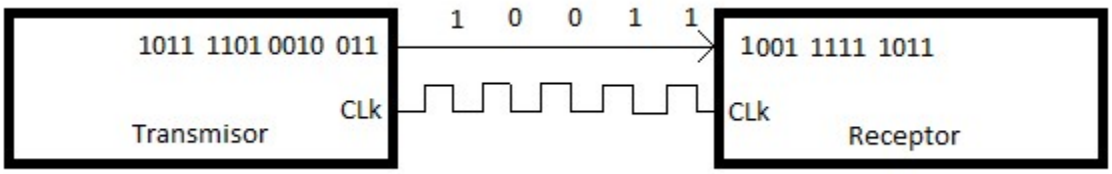


Ilustración 2-32 Transmisión serial síncrona.

2.8.1.2 Protocolos asíncronos.

En los protocolos asíncronos, no es necesario utilizar una señal de reloj para sincronizar al transmisor y al receptor, por lo que se requieren un menor número de cables que en los protocolos síncronos. En este tipo de protocolos, el receptor está activo en todo momento, esperando a que se presenten las instrucciones necesarias en el canal de transmisión, para poder iniciar la comunicación, en este caso no existe ningún reloj que indique cuando iniciar y cuando finalizar la comunicación, por lo que cada transferencia necesita un bit de inicio y

un bit de paro. Para que se pueda establecer una comunicación exitosa, el transmisor debe indicar una serie de instrucciones que el receptor pueda interpretar, y de esta forma identificar cuáles son los bits que corresponden a información, a la estructura que forman los datos e instrucciones se le denomina trama.

Para garantizar que la transmisión de los datos se realice sin errores, el transmisor y el receptor deben de trabajar a las mismas velocidades, es decir, que el transmisor no envíe datos más rápido o más lento de lo que el receptor puede recibir y que el receptor no reciba datos más rápido o más lento de lo que el transmisor puede enviar, a esta velocidad se le denomina tasa de baudios (baud rate).

La tasa de baudios índica cuantos bits se pueden transmitir en un segundo, se mide en bits por segundo (bps), establece la velocidad con la que el transmisor envía datos y la velocidad con la que el receptor recibe datos, la tasa de baudios debe de ser la misma para el transmisor y el receptor de lo contrario la información presentara errores, la tasa de baudios está estandarizada, es decir, que los dispositivos operan a velocidades que ya están predeterminadas, las velocidades más comunes son: 1200bps, 2400bps, 4800bps, 9600bps, 19200bps, 38400bps, 74880bps, 115200bps. Para calcular el tiempo que tarda en transmitirse un bit podemos utilizar la ecuación 2.2.

$$\boldsymbol{t = \frac{1}{bps}} \qquad \text{Ec 2.2}$$

En donde “t” es el tiempo en segundos que tarda en transmitirse un bit y “bps” es la tasa de baudios. Haciendo el cálculo de tiempo para una tasa de baudios de 9600bps se obtiene el resultado de 0.00010516 segundos.

Para hacer funcionar a un protocolo de comunicación serial asíncrona, se necesita una conexión eléctrica muy simple respecto a los protocolos seriales síncronos. (Ilustración 2-33)

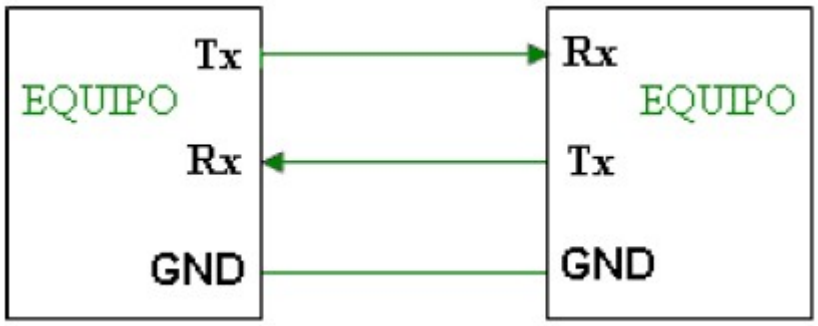


Ilustración 2-33 Conexión eléctrica para protocolos asíncronos.

Cada dispositivo que utilice una comunicación asíncrona necesita 2 terminales de conexión comúnmente identificadas como Tx y Rx, aunque puede adoptar diferentes nombres dependiendo el protocolo. Como se puede observar en la ilustración 2-35, la terminal Tx es la salida de datos del dispositivo, esta terminal debe de estar conectada a la terminal Rx del segundo dispositivo, la terminal Rx es la encargada de recibir los datos provenientes de otro dispositivo. Es importante que las terminales Rx estén conectadas a las terminales Tx y viceversa, si conecta una terminal Tx con otra terminal Tx o una terminal Rx con otra terminal Rx, no se podrá establecer la comunicación entre los dispositivos y puede producir daños en los componentes conectados al sistema. La terminal GND, no se utiliza para la comunicación entre los dispositivos, esta terminal forma parte del circuito electrónico.

2.9 Protocolo I2C

El protocolo I2C, es un protocolo de comunicación digital serial síncrono, necesita de 2 cables de comunicación, los cuales se identifican como SDA (Serial Data) y SLC (Serial Clock), SDA corresponde al canal o hilo por el que se envían todos los datos a transmitir y SCL corresponde al reloj de sincronización.

Para que este protocolo pueda funcionar, se necesita un dispositivo maestro (Master), que es el encargado de dar todas las instrucciones de lectura o escritura y puede enviar y recibir datos. También se necesita de Uno o varios dispositivos esclavos (Slave), estos dispositivos reciben instrucciones provenientes del dispositivo maestro para ejecutar una acción específica, pueden enviar y recibir datos.

La principal ventaja de este protocolo es la implementación de múltiples dispositivos utilizando únicamente 2 cables de comunicación, todos los dispositivos comparten el mismo

canal de transmisión y el mismo reloj de sincronización (ilustración 2-34). Muchos de los dispositivos que existen en el mercado, utilizan este protocolo de comunicación, es compatible con muchas de las tarjetas de desarrollo, como las tarjetas Arduino o las tarjetas ESP. Existe una amplia variedad de dispositivos que utilizan este protocolo de comunicación, dentro de los que podemos encontrar: sensores, memorias, microcontroladores, digitalizadores, displays, etc.

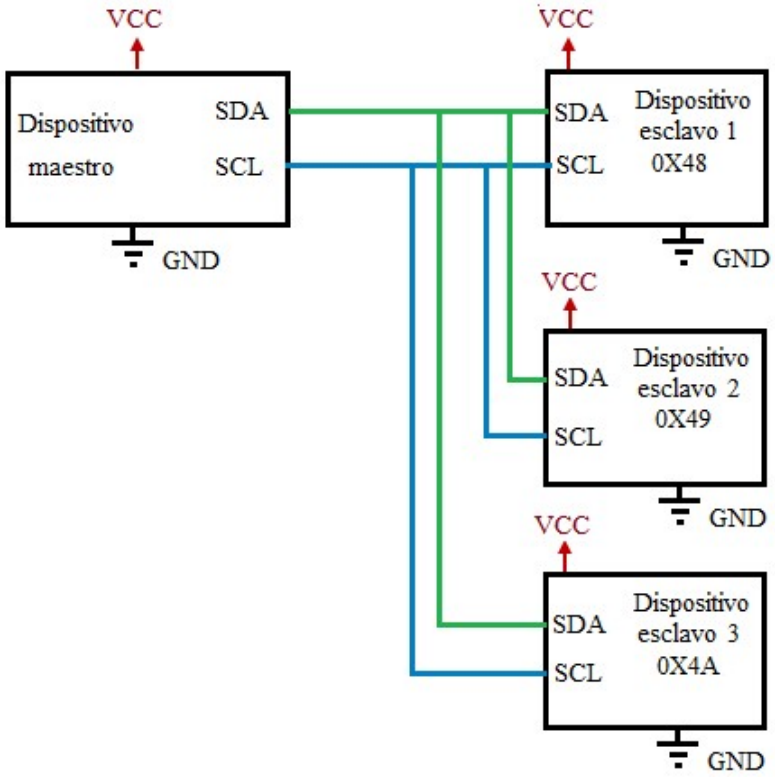


Ilustración 2-34 Conexión para el protocolo I2C.

Como se puede observar en la ilustración 2-34, todos los dispositivos, tanto como el dispositivo maestro y como los dispositivos esclavos, tienen sus terminales de SDA y de SCL conectadas entre sí, esto tiene la ventaja de poder implementar múltiples dispositivos con un reducido número de cables de conexión.

Para que la comunicación en este protocolo se realice de forma ordenada y correcta, el dispositivo maestro, que es el encargado de controlar el tráfico de datos en el canal de transmisión, tiene que saber que dispositivos están conectados y que dirección tiene cada Uno de los dispositivos, en la ilustración 2-34, se puede observar que todos los dispositivos esclavos tienen una “dirección” que en realidad es un numero en formato hexadecimal, para

el dispositivo 1 la dirección es 0X48, para el dispositivos 2 la dirección es 0X49 y para el dispositivo 3 la dirección es 0X4A, esta es una dirección de 7 bits en formato hexadecimal y sirve para que el dispositivo maestro pueda enviar instrucciones al dispositivo esclavo y pueda transmitir o recibir datos, se puede decir que la dirección es el nombre del dispositivo esclavo y mediante este mismo, el dispositivo maestro se puede comunicar con los dispositivos esclavos de forma individual. Esta dirección debe ser única para cada dispositivo, no puede haber más de un dispositivo con la misma dirección, de lo contrario, el sistema presentara errores. La dirección de cada dispositivo puede estar establecida por el fabricante del dispositivo o en algunos casos, se puede modificar manualmente haciendo las conexiones eléctricas que el fabricante indique.

El proceso de comunicación en el bus I2C (ilustración 2-35) [39]:

- El maestro comienza la comunicación enviando un patrón llamado “start condition”. Esto alerta a los dispositivos esclavos, poniéndolos a la espera de una transacción.
- El maestro se dirige al dispositivo con el que quiere hablar, enviando un byte que contiene los siete bits (A7-A1) que componen la dirección del dispositivo esclavo con el que se quiere comunicar, y el octavo bit (A0) menos significativo se corresponde con la operación deseada (R/W), lectura = 1 (recibir del esclavo) y escritura = 0 (enviar al esclavo).
- La dirección enviada es comparada por cada esclavo del bus con su propia dirección, si ambas coinciden, el esclavo se considera direccionado como esclavo-transmisor o esclavo-receptor dependiendo del bit R/W.
- Cada byte leído/escrito por el maestro debe ser obligatoriamente reconocido por un bit de ACK por el dispositivo maestro/esclavo.
- Cuando la comunicación finaliza, el maestro transmite una “stop condition” para dejar libre el bus.

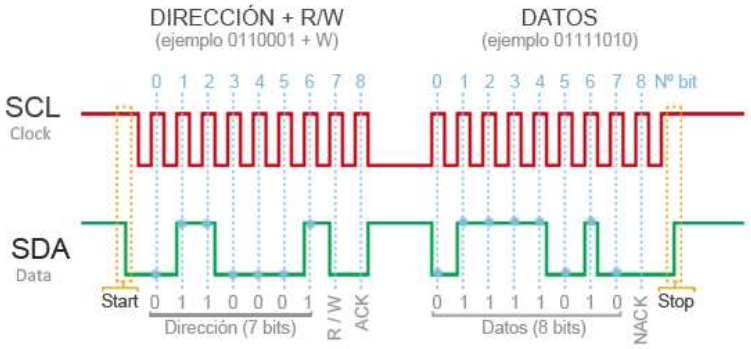


Ilustración 2-35 Proceso de Comunicación I^2C.

El dispositivo maestro solo puede comunicarse con Uno de los dispositivos esclavos a la vez, el orden en el que se comunica con los dispositivos esclavos, es según lo indique el dispositivo maestro, no hay un orden estricto a seguir.

Aunque el protocolo I2C es uno de los protocolos más utilizados, presenta algunas desventajas respecto a otros protocolos similares, una de las principales desventajas, es que su rango de alcance es de unos pocos metros, si se intenta utilizar este protocolo a distancias muy grandes, la comunicación se vuelve inestable o en la mayoría de los casos, no es posible establecer comunicación entre dispositivos maestro y esclavos, esto se debe principalmente a que los cables conductores que se utilizan para hacer la conexión, tienen una resistencia eléctrica que está presente de forma natural en el metal del que está formado el cable, esta resistencia o impedancia eléctrica, tiene un comportamiento inestable con las altas frecuencias de reloj que maneja el protocolo, produciendo que la señal eléctrica se atenué o se encuentre con una alta impedancia, bloqueando la corriente eléctrica, otra de las desventajas que presenta este protocolo, es la velocidad de transmisión, como se trata de una comunicación serial síncrona, la velocidad depende directamente de la frecuencia que el reloj (SLC) pueda alcanzar, la velocidad es de 100 kb/s en el modo estándar, aunque también permite velocidades de 3.4 Mb/s [39]. se puede considerar lenta respecto a otros protocolos similares, en este protocolo los dispositivos maestro y esclavos, no pueden enviar y recibir datos al mismo tiempo, cada una de estas transferencias se hace de forma individual y utilizan el mismo canal de comunicación, es decir, que utiliza una comunicación Half-duplex (ilustración 2-36).

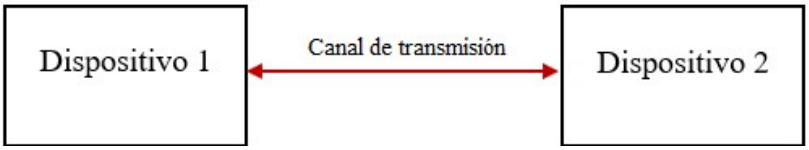


Ilustración 2-36 Comunicación Half-duplex.

La comunicación Half-duplex tiene la ventaja de utilizar solamente un canal de transmisión para enviar y recibir datos, el inconveniente con esta configuración, es que los dispositivos no pueden enviar y recibir datos al mismo tiempo, las transferencias se realizan una a la vez (ilustración 2-37), no en todos los sistemas es posible o conveniente utilizar la comunicación Half-duplex, en sistemas en los que ambos dispositivos se tengan que comunicar simultáneamente, este tipo de comunicación no es útil.

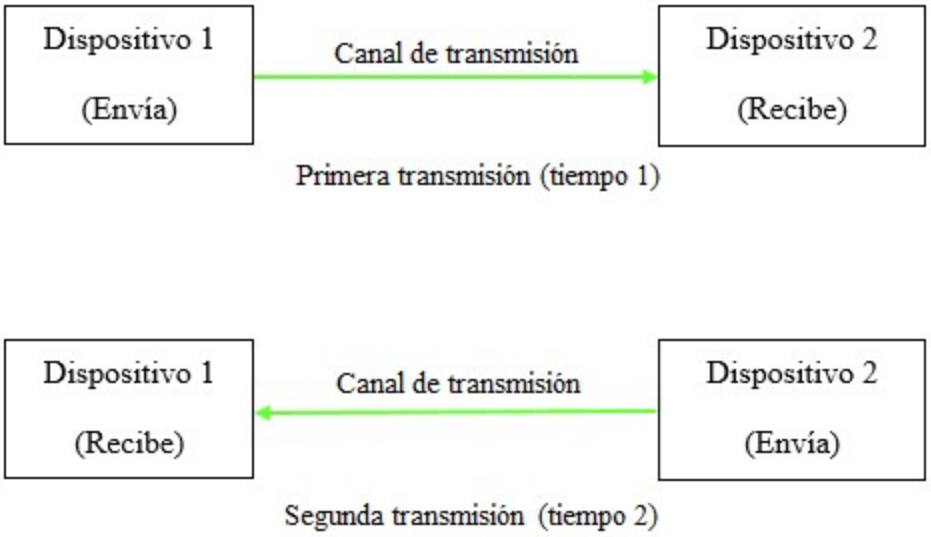


Ilustración 2-37 Transmisión con Half-duplex.

2.10 Protocolo SPI

El protocolo SPI (Serial Peripheral Interface), es un protocolo de comunicación serial síncrono, ampliamente utilizado para comunicar múltiples dispositivos dentro de un sistema, para funcionar, se necesitan mínimo 2 dispositivos, el dispositivo maestro (Master) y Uno o varios dispositivos esclavos (Slave), este protocolo necesita 2 cables para transmisión de datos y un hilo para el reloj de sincronización, se pueden identificar como: MOSI (Master Output Slave Input), MISO (Master Input Slave Output) y SCLK (Clock), además, cada Uno de los dispositivos esclavos necesita una terminal para que el dispositivo maestro pueda activarlo, se puede identificar como SS o CS (Chip Select). Todos los dispositivos dentro del sistema, tanto como los dispositivos esclavos, como el dispositivo maestro, comparten las terminales: MISO, MOSI y SCLK, excepto la terminal SS, la cual es única para cada Uno de los dispositivos esclavos, en la ilustración 2-38 se muestran las conexiones eléctricas necesarias para el funcionamiento de este protocolo.

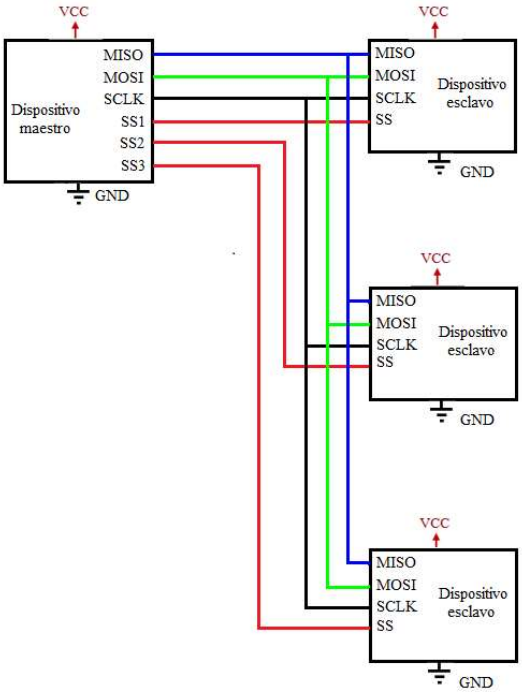


Ilustración 2-38 Conexión para el protocolo SPI.

En el protocolo SPI, el dispositivo maestro, es el encargado de dar todas las instrucciones y de controlar el tráfico de información en los canales de transmisión, como se puede observar en la ilustración 2-38, este protocolo tiene un número elevado de cables respecto a la comunicación I2C, el número de cables necesarios para hacer funcionar un sistema empleando el protocolo SPI, depende directamente de la cantidad de dispositivos que contenga todo el sistema, cada dispositivo esclavo necesita de una terminal independiente SS o CS, esta terminal es necesaria para cada dispositivo esclavo, debido a que, mediante esta terminal, el dispositivo maestro puede activar o desactivar cada Uno de los dispositivos esclavos, como el dispositivo maestro solo puede comunicarse con Uno de los dispositivos esclavos a la vez, es necesario que se active Uno de los dispositivos esclavos y se desactive al resto. Esta característica puede representar una desventaja para algunos sistemas en los que se tenga un número elevado de dispositivos, sin embargo, la principal ventaja de este sistema, es la velocidad de transmisión, en sistemas con Arduino se pude alcanzar hasta 8MHz de frecuencia de reloj, además de que, a diferencia del protocolo I2C, en el protocolo SPI, tanto el dispositivo maestro, como el dispositivo esclavo, si pueden enviar y recibir datos de forma simultánea, es decir, que utiliza la comunicación Full-duplex (Ilustración 2-39).

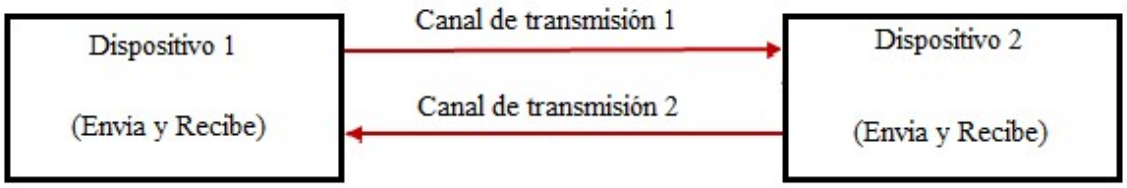


Ilustración 2-39 Comunicación Full-duplex.

En los sistemas Full-duplex, cada dispositivo tiene un canal exclusivo de transmisión y recepción de datos, por lo que las transferencias se realizan de forma más rápida, en comparación con los sistemas Half-duplex, en los cuales se necesita hacer dos transferencias diferentes.

2.11 UART TTL

UART (Universal Asynchronous Reciver Transmiter), es el hardware encargado de enviar datos de un dispositivo a otro utilizando comunicación serial asíncrona, este hardware se puede encontrar incluido en múltiples dispositivos, por ejemplo, en los microprocesadores que se encuentran en las placas de desarrollo, comúnmente la UART se utiliza para conectar módulos de expansión a las placas de desarrollo como módulos inalámbricos bluetooth o distintos tipos de sensores, aunque no es su única aplicación, se puede utilizar para múltiples tareas que necesite el usuario.

TTL (Transistor-Transistor Logic), es un estándar ampliamente utilizado y Uno de los más básicos, se puede encontrar en múltiples dispositivos electrónicos digitales, este tipo de tecnologías, indica varias características de construcción y de operación de los dispositivos construidos bajo este estándar, como temperaturas de operación, tipo de transistores con los que está construido el circuito integrado, entre otras cosas. Una de las principales características que indica este estándar, son los niveles de voltaje con los que se trabaja, para el estándar TTL, el 1 lógico o estado alto corresponde a un voltaje positivo de 5v o 3.3v y el 0 lógico corresponde a un voltaje de 0v, estos niveles voltaje tiene una tolerancia de error, es decir, no siempre se puede obtener 0v o 5v exactos, por lo que se deben de tomar en cuenta estos niveles de tolerancia, si estos voltajes se encuentran en un nivel fuera de las tolerancias establecidas, entonces se puede decir que se encuentra en un nivel de incertidumbre, el cual

no corresponde a 0 lógico ni a 1 lógico, si esto se presenta el sistema no podrá reconocer el bit y puede asignarlo de forma aleatoria o no asignarle ningún valor. (Tabla 2-2)

Tabla 2-2 Niveles lógicos TTL.

Niveles lógicos en TTL	**Niveles de voltaje**
Nivel bajo (0 lógico)	0v a 0.8v
Nivel alto (1 lógico)	2v a 5v
Nivel de incertidumbre o indeterminado	>0.8v o <2v

Debido a que la UART utiliza una comunicación serial asíncrona, es necesario contar con las líneas de transmisión de datos Tx y Rx, la conexión electrónica es la que se puede observar en la ilustración 2-33 vista en el titulo *0 Protocolos asíncronos*. Utiliza también una tasa de baudios o baud rate, la cual debe ser la misma ambos dispositivos.

Además de que el transmisor y el receptor operen bajo la misma tasa de baudios, la trama (estructura de bits) debe de contener una serie de bits que tiene funciones específicas para poder realizar una comunicación sin errores: bits de datos, bits de sincronización, bits de paridad.

Los bits de datos, son los bits que contienen la información útil, en otras palabras, es la información que se va a transmitir de un dispositivo a otro, están agrupados en paquetes de varios bits, no existe un numero definido de cuantos bits contiene cada paquete, pero generalmente se utiliza el tamaño de byte (8bits), el tamaño de cada paquete debe de ser establecido desde el principio para el transmisor y para el receptor, el tamaño de los paquetes no puede variar, si se establece en byte, todos los paquetes deben de ser del mismo tamaño.

Los bits de sincronización se pueden encontrar al principio y al final de la trama, y se encargan de indicarle al transmisor la entrada de datos provenientes del transmisor. Los bits de sincronización son: bit de inicio y bit de parada, el bit de inicio, indica al receptor que el transmisor ha enviado datos y es momento de recibirlos, solo se emplea un bit de inicio, y el bit de parada, indica al receptor que el envío de datos ha finalizado, se pueden llegar a emplear 2 bits como bits de parada.

Bits de paridad, los bits de paridad se utilizan para comprobar si no se han presentado errores durante la comunicación, existen dos variantes de bits de paridad: paridad par o paridad

impar, la paridad par se obtiene sumando la cantidad de los niveles altos (1 lógico) de los bits de datos, si la cantidad de estados altos da como resultado un numero par, el bit de paridad tomara el valor de 0 lógico o estado abajo, y si la suma de estados altos da como resultado un número impar, el bit de paridad tomara el valor de 1 lógico (Tabla 2-3).

Sí se establece una paridad par y la cantidad de niveles altos no da como resultado un numero par, el sistema detecta que se ha producido un error, lo mismo se aplica cuando se establece una paridad impar, si el sistema detecta que la suma de estados altos no da como resultado un número impar, el sistema detecta que se ha producido un error.

No es necesario hacer uso de los bits de paridad para poder transmitir, y tampoco es muy común utilizarlos, debido a que ralentiza la transmisión de datos.

Tabla 2-3 Bits de paridad.

Bits de datos	**Cantidad de "1"**	**Tipo de número**	**Valor que toma el bit de paridad**
10011101	5	Impar	1
11011101	6	Par	0

Para ejemplificar la transmisión de datos mediante UART TTL, se enviará el byte de datos: "01011001", sin hacer uso de los bits de paridad (Ilustración 2-40).

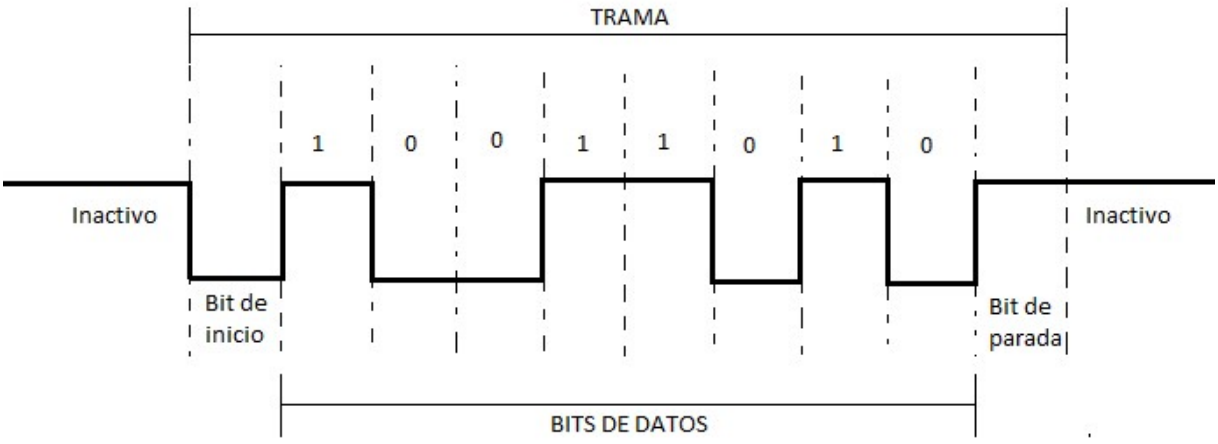


Ilustración 2-40 Transmisión de datos mediante UART.

Como se puede observar la ilustración 2-40, cuando el transmisor no está enviando información, el canal de transmisión se mantiene en estado alto o en 1 lógico, el bit de inicio siempre se presenta como un 0 lógico o un nivel bajo, cuando se presenta el bit de inicio el receptor comienza la recepción de datos, en el ejemplo de la ilustración 2-40, los bits a transmitir son: 01011001 y se observa que la transmisión inicia con el ultimo bit, esto se debe a que la transmisión en este tipo de comunicación, inicia con el bit menos significativo. Todo conjunto de bits sin importar el tamaño tiene un bit más significativo y un bit menos significativo, el bit más significativo es el que tiene mayor valor y el bit menos significativo es el que tiene menor valor, por ejemplo, en la cifra "235", el número más significativo es el número 2, debido a que su valor es de 200 y el numero menos significativo es el 5, debido a que su valor solo es 5, el mismo principio se aplica en los números binarios.

2.12 Protocolo RS232

RS232 es un protocolo de comunicación serial asíncrono, enviar datos mediante el protocolo RS232 presenta varias ventajas que solamente el uso de la UART, entre las principales ventajas se encuentra el rango de alcance, la distancia a la que se puede enviar datos utilizando el protocolo RS232 es mayor que el de la UART, esto se debe entre otras cosas, a los niveles lógicos que utiliza el protocolo RS232 (Tabla 2-4).

Tabla 2-4 Niveles lógicos en RS232.

Niveles lógicos en RS232	**Niveles de voltaje**
Nivel bajo (0 lógico)	3v a 15v
Nivel alto (1 lógico)	-15v a -3v
Nivel de incertidumbre o indeterminado	>-3v o <3v

Es posible convertir una transmisión de datos de UART TTL a RS232 usando un circuito integrado con la matrícula MAX232 (ilustración 2-41).

Este circuito integrado era ampliamente utilizado para establecer comunicación entre microcontroladores y computadoras. En anteriores generaciones de computadoras, era posible encontrar un puerto físico que utilizaba el protocolo RS232, hoy en día, este puerto ha ido desapareciendo de las computadoras, debido principalmente a la extensa compatibilidad de los dispositivos USB que soportan velocidades mayores y son más fiables,

aunque aún se utiliza en algunos sistemas industriales, para comunicar microcontroladores, entre otras aplicaciones. Una de las principales desventajas del protocolo RS232, es que no admite conectar múltiples dispositivos, la transmisión solo se puede hacer entre 2 dispositivos.

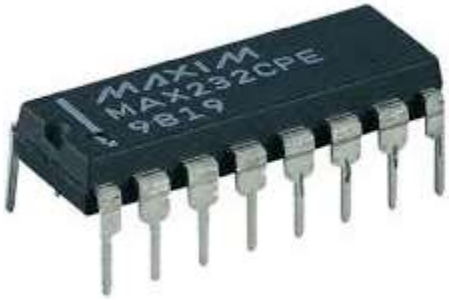


Ilustración 2-41 MAX232.

El circuito integrado MAX232, necesita una fuente de voltaje de corriente directa de 5v, que pueda suministrar una corriente eléctrica de 8 mA a 10 mA y trabaja a una temperatura mínima de -40°C y una máxima de 85°C, [40] soporta las tasas de baudios más comunes utilizadas por los microcontroladores, aunque a mayor tasa de baudios, menor es la distancia que puede alcanzar sin que la señal se atenúe.

Para poder convertir una señal proveniente de una UART TTL a una señal RS232, es necesario realizar las conexiones eléctricas que se muestra en la ilustración 2-42.

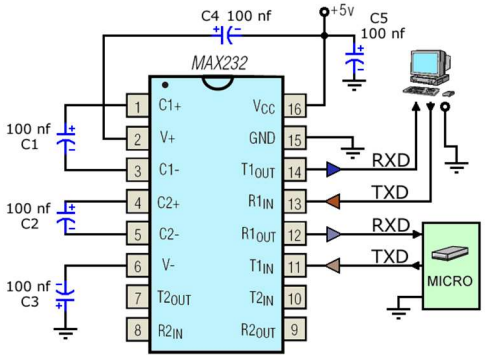


Ilustración 2-42 Conexión para MAX232.

2.13 Protocolo RS485

Cuando se necesita trasmitir a largas distancias o con velocidades mayores a las del protocolo RS232, RS485 es la solución. Utilizando enlaces con RS485 no hay limitación a conectar tan solo dos dispositivos, dependiendo de la distancia, la velocidad de transmisión y los circuitos integrados que se utilicen, se pueden conectar hasta 32 nodos con un simple par de cables.

Este protocolo presenta varias ventajas respecto a RS232 [41]:

- Bajo costo. Los circuitos integrados para transmitir y recibir son baratos y solo requieren una fuente de +5v para poder generar una diferencia mínima de 1.5v entre las salidas diferenciales. En contraste con RS232 que en algunos casos requiere de fuentes dobles para alimentar algunos circuitos integrados, o en su defecto arreglos de capacitores.
- Capacidad de interconexión. RS485 es una interface multi-enlace con la capacidad de poder tener múltiples transmisores y receptores. Con una alta impedancia receptora, los enlaces con RS485 pueden llegar a tener a lo máximo hasta 256 nodos.
- Longitud de enlace. En un enlace RS485 puede tener hasta 4000 pies de longitud (1.219Km), comparado con RS232 que tiene Unos límites típicos de 50 a 100 pies (30.5m).
- Rapidez. La tasa de baudios puede llegar a ser tan alta como 10Mbps.

Por todas las ventajas anteriormente descritas, el protocolo RS485 es ampliamente utilizada en la industria, se puede encontrar en sistemas automatizados, líneas de ensamble, sistemas de monitoreo, etc.

La razón por la que el protocolo puede transmitir a distancias largas, es debido a la técnica que utiliza para asignar los niveles lógicos. A diferencia de los protocolos RS232, SPI, I2C, incluso la UART TTL, que asignan los niveles lógicos utilizando un nivel de voltaje, este nivel de voltaje se encuentra entre la línea de datos y GND o tierra eléctrica. El protocolo RS485 no asigna niveles de voltaje, para poder asignar los niveles lógicos, se utilizan 2 líneas, a estas líneas se les llama A y B, la forma en la que se asignan los niveles lógicos utilizando estas dos líneas, es mediante la diferencia de voltaje que existe de una línea respecto a la otra, nunca respecto a GND (Tabla 2-5).

Tabla 2-5 Niveles lógicos en RS485.

Niveles lógicos en RS485	Líneas A y B
Nivel bajo (0 lógico)	Cuando A es positivo respecto a B
Nivel alto (1 lógico)	Cuando A es negativo respecto a B

Este sistema tiene una ventaja muy grande respecto a los protocolos que utilizan niveles de voltaje respecto a GND y es la razón principal por la que es más utilizado en un ambiente industrial y los demás protocolos no, la ventaja es que soporta muy bien los ambientes llenos de ruido artificial y ruido natural, esto se debe a que, como los niveles lógicos se asignan dependiendo las diferencias entre A y B, cuando el ruido altera las líneas de transmisión, este las altera de igual forma y proporción, por lo que no se produce ninguna diferencia entre ambas líneas, por lo tanto, no se produce ninguna alteración en la información transmitida (Ilustración 2-43).

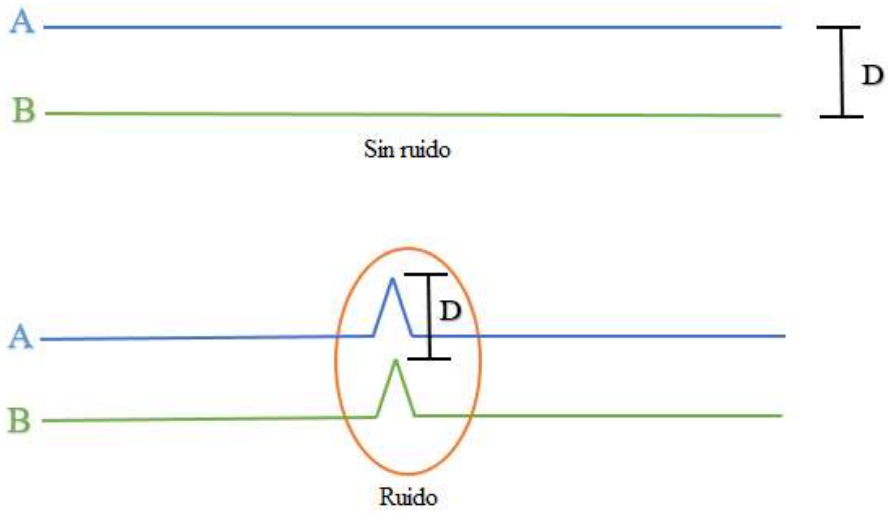


Ilustración 2-43 Ruido en RS485.

Es posible convertir una señal proveniente de una UART TTL a una señal de protocolo RS485, también se puede convertir una señal RS232 a una señal de protocolo RS485. Para poder convertir una señal de UART TTL a RS485, se puede utilizar el circuito integrado MAX485 (Ilustración 2-44).

Ilustración 2-44 MAX485.

El circuito integrado MAX485 presenta una configuración muy sencilla y fácil para poder convertir una señal TTL a una señal RS485, con este circuito integrado se puede llegar a tener una comunicación Half-Duplex, puede alcanzar una tasa de baudios de 2.5Mbps, necesita una fuente de alimentación de voltaje de 5 Vdc y como máximo 12 Vdc [42], no necesita ningún arreglo de componentes adicional para poder funcionar de forma correcta, aunque se puede configurar algún arreglo de acopladores ópticos de señal para reducir riesgos de corto circuito y proteger a la UART de corrientes inversas o corto circuitos. La disposición de las terminales de este circuito integrado se describe en la ilustración 2-45.

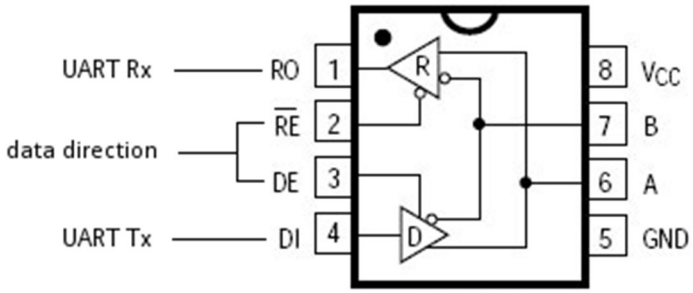


Ilustración 2-45 Terminales de MAX485.

La forma de conectar este circuito integrado es muy simple, la terminal 1, se conecta a la terminal Rx de la UART, la terminal 4, se conectar a la terminal Tx de la UART, la terminal 8, se conecta a la fuente de voltaje de 5 Vdc, las terminales 6 y 7, son las encargadas de enviar y recibir señales RS485, la terminal 5 se conecta a GND del sistema electrónico y por ultimo las terminales 2 y 3, son las encargadas de activar el modo emisor y el modo receptor, dependiendo cuál de los dos modos este activo, el circuito integrado envía o recibe datos (Tabla 2-6).

Tabla 2-6 Data direction MAX485.

RE	DE	Acción que realiza
Cuando la terminal se conecta a 5 Vdc	Cuando la terminal se conecta a 5 Vdc	Se activa el modo emisor
Cuando la terminal se conecta a GND	Cuando la terminal se conecta a GND	Se activa el modo receptor

Para convertir una señal de UART TTL a una señal de RS485, se necesita la conexión electrónica que se describe en la ilustración 2-46. Con esta conexión, se establece una comunicación Simplex, es decir, que los datos solo se pueden transmitir en una sola dirección, de un dispositivo 1 a un dispositivo 2, nunca del dispositivo 2 al dispositivo 1. También es posible lograr comunicaciones Half-duplex, incluso Full-duplex, aunque se necesita hacer uso de otros dispositivos electrónicos adicionales para lograrlo.

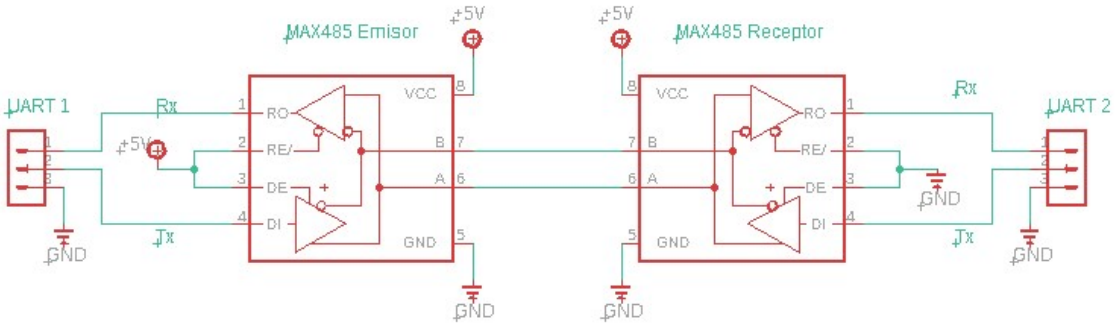


Ilustración 2-46 Conexión para comunicación simplex usando MAX485.

2.14 Conversión analógica-digital (ADC)

La digitalización es un proceso al cual se somete información en formato analógico para convertir dicha información a un conjunto de bits que representan a esta información y que puede ser manipulada o modificada por dispositivos digitales como pueden ser computadoras, Smartphones, tarjetas de desarrollo, etc. Dentro de las principales razones por las cuales se opta por la digitalización de la información, se pueden encontrar: el almacenamiento de datos en la nube o en discos duros, respaldos de seguridad, conservación de información atreves del tiempo, transmisión de información a largas distancias, estudio de datos estadísticos, etc.

Casi cualquier tipo de dato que se encuentra en el entorno en el que vivimos se puede digitalizar mediante el uso de un transductor, por ejemplo, los colores en una fotografía o documento, una canción, las imágenes que componen una película, los latidos del corazón, la temperatura del cuerpo humano, o en el caso de este proyecto, la humedad presente en el suelo utilizado para el cultivo de alimentos.

Al dispositivo encargado de realizar la conversión de analógico-digital, se le conoce como ADC (Analog to Digital Converter), este elemento tiene dos parámetros para realizar la conversión de la señal analógica: voltaje de referencia (Vref) y bits de resolución. El voltaje de referencia se establece dependiendo del voltaje máximo que puede tomar la señal analógica, por ejemplo, para el sensor FC-28 el voltaje máximo de la señal analógica es de 5v y 3v para el sensor capacitivo y el sensor VH400. El segundo parámetro son los bits de resolución, que representa la cantidad de fracciones en las cuales se puede dividir la señal analógica, es importante mencionar que la cantidad de bits de resolución es un parámetro que ya está establecido por el fabricante del ADC. De los parámetros mencionados anteriormente depende la precisión de la muestra, por ejemplo, para un voltaje de referencia de 5v y 10 bits de resolución (que es la resolución del ADC del Arduino Uno), se calculará la precisión de la muestra, para realizar este cálculo lo primero es realizar la conversión de los bits de resolución a decimal, para realizar esta conversión se realiza la siguiente operación: $\boldsymbol{Resolución\ en\ decimal = 2^{número\ de\ bits} = 2^{10} = 1024}$, al resultado obtenido se le debe de restar 1 ya que cuando se habla de bits el 0 también se toma en cuenta, por lo que la resolución real es de 1023, una vez realizado este cálculo, se sustituyen los datos en la ecuación 2.1.

$$\boldsymbol{Precisión = \frac{voltaje\ de\ referencia}{resolución\ en\ decimal}} \qquad \text{Ec 2.1}$$

Realizando el cálculo se obtiene el resultado de 4.89mv de precisión, esta precisión es aceptable siempre y cuando en la señal analógica no existan voltajes menores a 4.89 mV o que no sean relevantes, ya que al estar fuera del rango de precisión del ADC, estos datos se perderán durante el proceso de digitalización, en caso de necesitar una mayor precisión, se puede elegir un ADC que trabaje con mayor resolución, o bien disminuyendo el voltaje de referencia, teniendo siempre en cuenta que el voltaje de referencia no debe ser menor al voltaje máximo de la señal analógica que se va a digitalizar.

2.14.1 Digitalizador ADS1115

El ADS1115 es un conversor analógico digital con 16 bits de resolución, con un encapsulado ultra pequeño, cuenta con 4 ADC independientes, se puede alimentar con voltajes desde los 2 Vdc hasta los 5 Vdc (mismo que utiliza como voltaje de referencia), también permite la función de multiplexor de señales, así como la función de comparador de señales, se puede encontrar en el mercado en su versión pre ensamblada en un PCB que facilita su uso para tarjetas de desarrollo (Ilustración 2-47), utiliza el protocolo de comunicación digital I2C para el cual ofrece 4 direcciones I2C diferentes, por lo que se puede tener 4 de estos módulos funcionando simultáneamente (Ilustración 2-48).

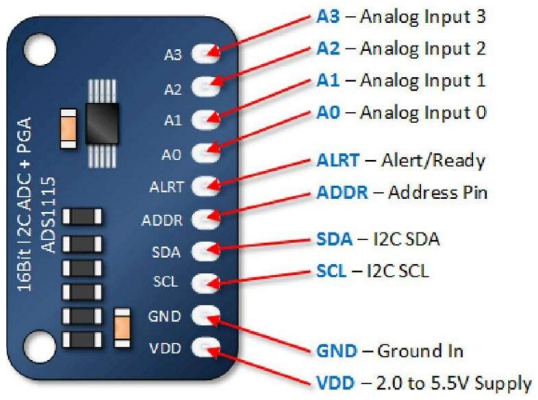


Ilustración 2-47 ADS115.

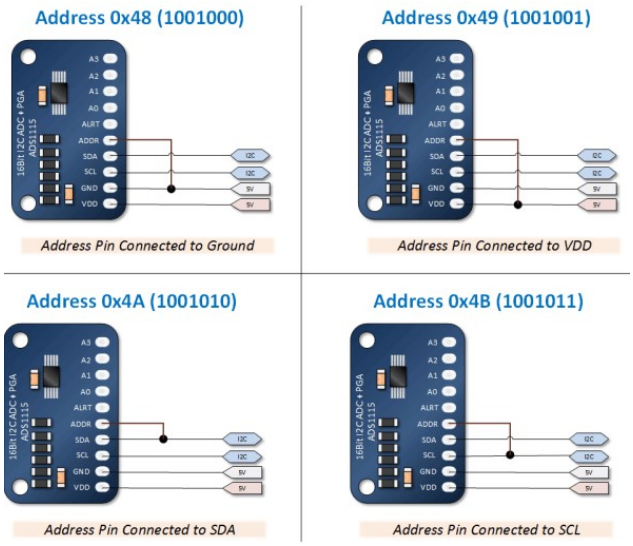


Ilustración 2-48 Direcciones I2C para ADS1115

2.15 SIM800L

El módulo SIM800L (Ilustración 2-49) es un dispositivo quad-band GSM/GPRS, trabaja en las frecuencias GSM850MHz, EGSM900MHz, DCS1800MHz y PCS1900MHz. Este módulo de telefonía celular permite añadir voz, texto, datos y GSM en un pequeño paquete, esta versión cuenta con un conector uFL que permite conectar una antena externa para mejorar la captación de cobertura móvil. Utiliza el mismo chip SIM800L que el módulo FONA de Adafruit, por lo que se pueden utilizar las mismas bibliotecas.

Por sí solo, este módulo no puede hacer nada. Se requiere un microcontrolador para controlarlo por ejemplo un Arduino, pero cualquier microcontrolador que opere de 3v a 5v con una UART puede enviar y recibir comandos a través de los pines RX/TX. También necesita de una tarjeta SIM 2G [43].

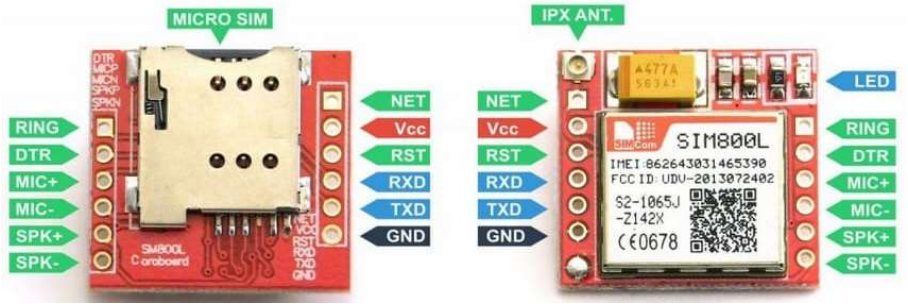


Ilustración 2-49 SIM800L.

3. Desarrollo

Para la elaboración de este proyecto, se tuvo un acercamiento con el responsable de los invernaderos del campus universitario F.E.S Aragón, aunque estos están enfocados principalmente para el uso académico de la carrera de Planificación para el Desarrollo Agropecuario, se tuvo asesoría para saber cuáles son las variables de importancia para mejorar el cultivo en un invernadero, y a partir de ahí se desarrolló esta propuesta que en esta etapa considera el monitoreo de la humedad, temperatura y luz ambiental, así como la humedad en el sustrato.

La propuesta consiste en tres módulos de adquisición de señales, que contienen los sensores y microcontrol para el monitoreo de variables; se realizaron diversas pruebas para lograr un correcto funcionamiento; también se desarrolló un módulo receptor de datos, así como de la programación de una interfaz gráfica de usuario, que permite al agricultor observar los valores de las mediciones, establecer niveles no deseados que activan una alerta que llega al teléfono del agricultor vía GSM; también incluye una bitácora con el histórico de las mediciones para dar la posibilidad de análisis.

Además de lo mencionado, los módulos de adquisición cuentan con puertos para aumentar la cantidad de sensores con propósito de expansión de monitoreo.

La idea de tres módulos tiene como finalidad el monitoreo en diversas zonas del invernadero, con el afán de poder controlar de mejor manera las condiciones de los cultivos.

En esta oportunidad se decidió trabajar de manera alámbrica y con transmisión de datos en formato digital dada su mejor capacidad para ser “inmune” al ruido, en comparación a las señales analógicas, cuando estas son transmitidas a mayores distancias.

3.1 Diagrama de bloques

En la ilustración 3-1, se muestra el diagrama de bloques que presenta los elementos principales del sistema. A grandes rasgos, se puede dividir en tres etapas: la etapa uno es la adquisición de datos, esta sección se conforma por los sensores, los módulos 1, 2 y 3, la etapa dos se encarga de la gestión de todos los datos (módulo maestro), esta sección está conformada por el convertidor RS485-UART, un Arduino Mega y el dispositivo SIM800L y

por último la etapa tres es la encargada del almacenamiento de datos y la parte gráfica con la que puede interactuar el agricultor, esta etapa está formada por la interfaz gráfica y la base de datos.

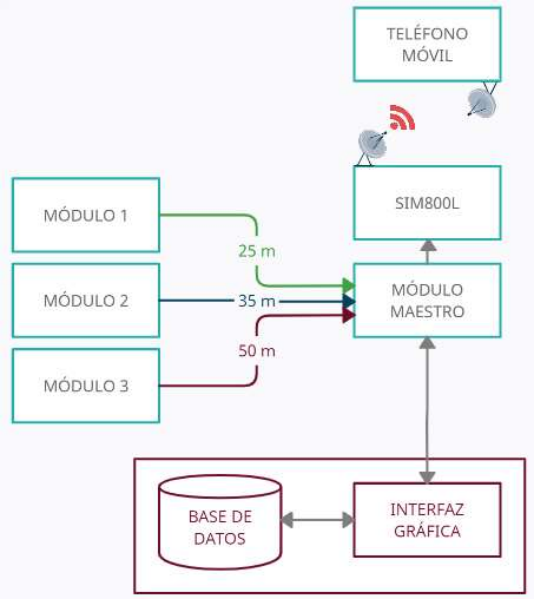


Ilustración 3-1 Diagrama de bloques para sistema de monitoreo.

3.1.1 Módulos 1, 2 y 3

Estos módulos son los encargados de controlar las lecturas de cada uno de los sensores que estén conectados al módulo, una vez que obtienen todas las lecturas de cada uno de los sensores del módulo, envía todos estos datos al módulo maestro, cada módulo debe de ser capaz de obtener datos en 1 segundo o menos. Cada módulo se debe colocar en una zona distinta del invernadero, por lo que se encontraran a diferentes distancias del módulo maestro.

Están constituidos por un microcontrolador Atmega328p, mismo que equipan las tarjetas de desarrollo Arduino Uno, por lo que se tienen las mismas prestaciones que una tarjeta Arduino Uno, también incluyen reguladores de voltaje para alimentar a los sensores que se encuentren conectados al módulo, además también cuenta con el convertidor UART_RS485 el cual se encarga de recibir una cadena de datos que contiene todas las lecturas de cada uno de los sensores, estos datos provienen de la UART del microcontrolador Atmega382P, se convierten a un formato RS485, esto se hace para que la señal no se pierda a lo largo del cable de transmisión. Para realizar la conversión, se empleó el circuito integrado MAX485 en configuración simplex, esta configuración es suficiente para este sistema, no es necesaria

la configuración Half-duplex o full-duplex ya que los datos siempre viajan en una sola dirección. Todo este arreglo se encuentra en el mismo PCB que los módulos 1, 2 y 3, pero su funcionamiento no forma parte de los módulos 1, 2 y 3, solo están alojados en el mismo espacio.

3.1.2 Módulo maestro

El principal componente, es un Arduino Mega que puede comunicarse con los módulos 1, 2 y 3, con la interfaz gráfica y con el módulo SIM800L. Recibe todos los datos de la etapa de adquisición del sistema, una vez que tiene todos los datos, los compara con el rango optimo, estos rangos son establecidos por el agricultor a través de la interfaz gráfica, si alguna de las lecturas esta fuera de rango, crea una instrucción que es enviada al módulo SIM800L para notificar directamente al agricultor sobre la situación dentro del invernadero. Una vez que ya se realizó la obtención de datos y la comparación con los rangos, los datos se agrupan en un solo paquete de datos y se envía directamente a la interfaz gráfica. Además, este módulo contiene un convertidor RS485-UART el cual recibe la cadena de datos que contiene las lecturas de los sensores en formato RS485, su función es convertir los datos de formato RS485 a niveles TTL, esto es debido a que el microcontrolador que utiliza el módulo maestro no tiene soporte para señales RS485 y para realizar esta conversión se hace uso del circuito integrado MAX485 en configuración simplex. Por último, en este módulo también aloja el módulo SIM800L encargado del envío de alertas GSM.

3.1.3 Interfaz gráfica y base de datos

La interfaz gráfica es un entorno mediante el cual el agricultor puede interactuar o modificar algunas configuraciones de forma gráfica, sin necesidad de hacer uso de lenguajes de programación, todo se realiza de forma sencilla e intuitiva. La interfaz gráfica es la encargada de mostrar en pantalla el estado actual de cada una de las variables del sistema y de establecer los parámetros deseados de las variables, también permite consultar datos históricos indicado la variable y un rango de fechas, estos datos históricos, son almacenados dentro de una base de datos, la cual se comunica únicamente con la interfaz gráfica, recibe datos de la interfaz y los almacena en unas tablas que ordena los datos por fechas y, cuando sea necesario consultar los datos almacenados, los recupera conforme las fechas en las que se le indica y las envía a la interfaz gráfica. Tanto la interfaz gráfica como la base de datos, no son componentes físicos

como sí lo son los módulos 1, 2 y 3 o el módulo maestro, por el contrario, estos dos elementos son software que se encuentra instalado en una computadora bajo el sistema operativo Windows 10.

3.2 Adquisición de datos (módulos 1, 2 y 3)

Esta etapa del proyecto es la encargada de controlar los sensores que toman las lecturas de las variables climáticas dentro del invernadero, así como del envió de datos en formato RS485.

3.3.1 Diagrama esquemático para módulos 1, 2 y 3

Los módulos 1, 2 y 3, tienen el mismo diagrama esquemático que se muestra en la ilustración 3-2, este se realizó en el software Autodesk Eagle. Es necesario aislar lo mejor posible de las condiciones climáticas que se encuentran dentro del invernadero al microcontrolador Atmega328p, al circuito integrado MAX485 y a los diversos componentes electrónicos que hacen funcionar a estos dos últimos, esto se debe a que los sensores necesitan estar en contacto directo con el ambiente para poder tomar lecturas de las variables climáticas, pero los circuitos integrados son muy sensibles a las condiciones climáticas, por lo que se deben separar, de lo contrario, se pueden producir daños físicos en los componentes, tales como oxidación en los pines de conexión o incluso cortos circuitos debido a la acumulación de humedad en la placa de circuito impreso (PCB). La forma de aislar estas dos partes, es colocándolas en PCB separados y conectadas eléctricamente por cable, el cual debe de tener una longitud máxima de 1m. De esta forma podemos proteger el PCB en el que están soldados los circuitos integrados y dejar en contacto directo con el ambiente a los sensores.

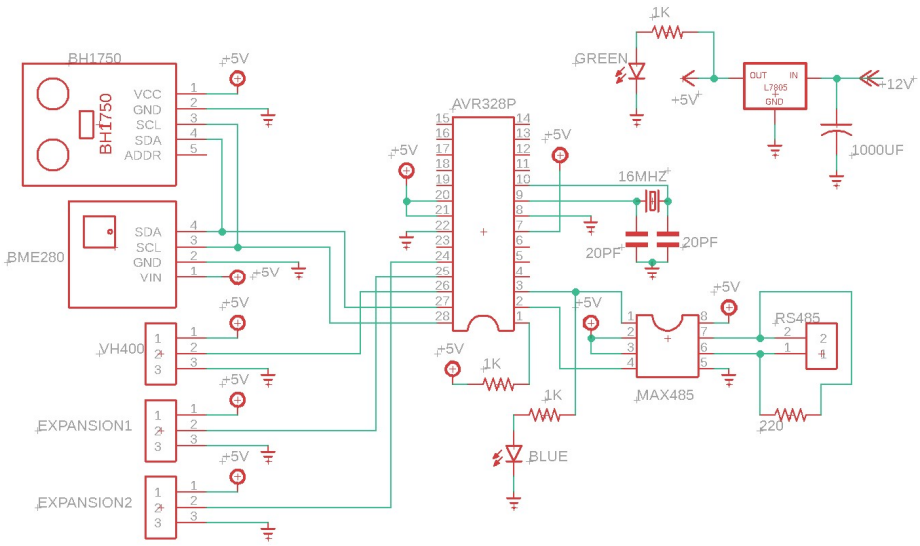


Ilustración 3-2 Etapa de adquisición (diagrama esquemático).

Como se muestra en la ilustración 3-2, los sensores que se utilizaron para este proyecto son: BH1750 para medir el nivel de luz presente en el ambiente, BME280 para medir la temperatura y humedad en el ambiente y el sensor VH400 para medir humedad en sustrato. El sensor BH1750 se decidió utilizar debido a que no es necesario realizar ningún tipo de experimentación para obtener el modelo matemático que describa su comportamiento, además de que es un sensor fácil de remplazar y se puede considerar de bajo costo, bajo estos mismos criterios se decidió utilizar el sensor BME280, por último, el sensor VH400 fue elegido de entre las demás opciones debido a los resultados obtenidos de las pruebas de humedad realizadas para obtener su modelo matemático. Estos sensores son controlados por un microcontrolador Atmega 328P, este microcontrolador es el mismo que utiliza la tarjeta de desarrollo Arduino Uno, por lo que se puede programar mediante el IDE de Arduino, además de que es compatible con todas las bibliotecas disponibles para arduino, dentro de las cuales se pueden encontrar las que utilizan los sensores BME280 y BH1750.Para poder alimentar todos estos dispositivos electrónicos, se necesita de una fuente de voltaje de 5 Vdc, uno de los problemas que se presentaron usando una fuente de 5 Vdc directamente, es que debido a estos módulos están a una distancia considerable del módulo maestro que proporciona este voltaje, se producían variaciones de voltaje a lo largo del cable utilizado, esto puede ocasionar que los sensores no reciban el suficiente voltaje que necesitan para su correcto funcionamiento, para solucionar este problema, se decidió utilizar una fuente de voltaje de 12 Vdc a lo largo de los cables que conectan a los tres módulos con el módulo

maestro y colocar un regulador de 5 Vdc para alimentar a los sensores y al microcontrolador, de esta forma si se presenta a una caída de voltaje ligera, no afecta a la alimentación de los sensores y el microcontrolador. Para poder enviar datos hasta el módulo maestro sin que la información se pierda a lo largo de los cables de mayor longitud, se hizo uso del protocolo de comunicación serial asíncrona RS485, para poder convertir los datos que salen del puerto serie del microcontrolador Atmega 328P, se empleó el circuito integrado MAX485, mismo que se puede observar en la ilustración 3-2. Por último, se utilizaron dos diodos LED, uno de color verde que sirve para indicar que el módulo se encuentra energizado y, otro de color azul que permite visualizar cuando el microcontrolador está enviando datos mediante el puerto serie.

3.3.2 PCB para etapa 1

Los módulos 1, 2 y 3, tienen el mismo diseño de PCB que se muestra en la ilustración 3-3. Este diseño pertenece a la placa en la que se encuentran colocados los circuitos integrados que se deben aislar lo mayor posible de las condiciones climáticas del invernadero para prolongar su vida útil.

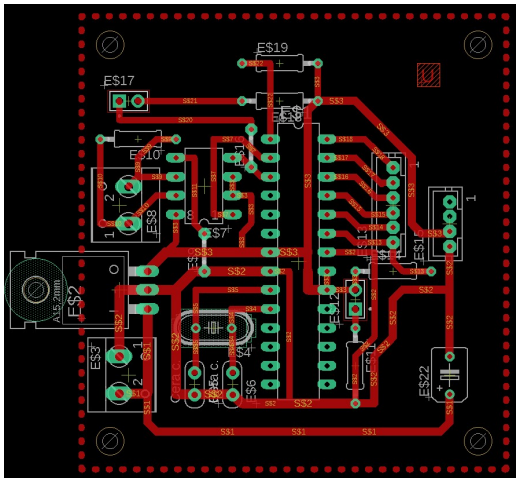


Ilustración 3-3 PCB para montaje de circuitos integrados (etapa de adquisición).

En la ilustración 3-4, se muestra el diseño de PCB en la que se encuentran alojados los sensores. Esta placa no aloja ningún componente electrónico esencial para el funcionamiento del sistema, su única función es conectar los sensores con el PCB de la ilustración 3-3.

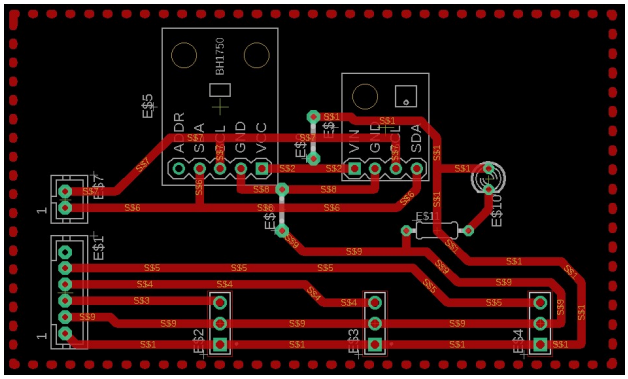


Ilustración 3-4 PCB para montaje de sensores (etapa de adquisición).

3.3.3 Diagrama de flujo para etapa de adquisición

El diagrama de flujo que aparece en la ilustración 3-5, pertenecen al algoritmo de programación para el microcontrolador Atmega328p de esta etapa, su función principal es enviar instrucciones a los sensores para que estos puedan obtener lecturas de las variables climáticas, también se encarga de recibir los datos de los sensores y formar una sola cadena de datos la cual se envía al módulo maestro.

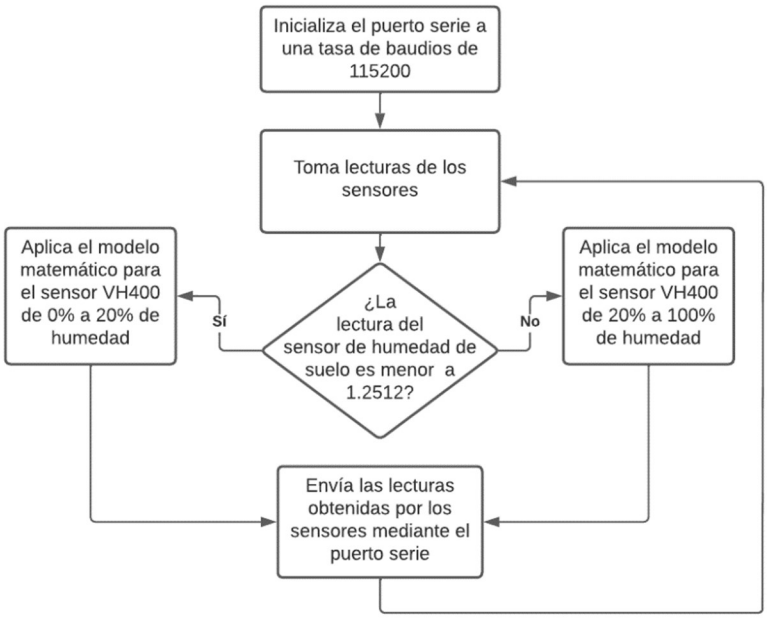


Ilustración 3-5 Diagrama de flujo etapa de adquisición.

3.3.4 Código fuente para la etapa de adquisición

En este apartado se muestra el código fuente programado en el microcontrolador Atmega328p de esta etapa.

```
void setup() {
   Serial.begin(115200);
```

Ilustración 3-6 Inicializa puerto serie a 115200 bps.

Como se está utilizando el mismo microcontrolador que en la tarjeta de desarrollo Arduino Uno, se utilizó también el Arduino IDE para realizar el código de programación, como se muestra en el diagrama de flujo, el primer paso fue inicializar el puerto serie a 115200 bps, esto se logra con la línea de código que se muestra en la ilustración 3-7, esta línea de código está dentro de la estructura "void setup ()", ya que al ser un parámetro inicial se debe colocar en esta posición de lo contario no será posible utilizar el puerto serie. Siguiendo el diagrama de flujo, el siguiente paso realizado fue, tomar lecturas de los sensores, para esto se emplearon las bibliotecas para los sensores BME280 y el sensor BH1750 (Ilustración 3-7), puesto que estas se pueden obtener de forma gratuita y son compatibles con el IDE de Arduino.

```
void loop() {
  uint16_t lux = sensor.read();
  int vh400 = analogRead(A0)*0.004887585;
  float EXP1 = analogRead(A1);
  float EXP2 = analogRead(A2);
  float Hum = bme.readHumidity();
  float Tem = bme.readTemperature();
```

Ilustración 3-7 Toma lecturas de los sensores.

Para obtener la lectura de del sensor VH400, solohizo falta leer al pin analógico al que el sensor se encuentre conectado, en este caso, se conectó al pin A0, una vez obtenido el valor digitalizado, se multiplica por la constante "0.004887585" que permite convertir el valor digitalizado a una lectura de voltaje, el mismo procedimiento se aplico para los puertos de expansión 1 y 2.El siguiente paso fue comparar la lectura obtenida por el sensor de humedad de sustrato con el valor "1.2512", esta comparación permite saber cuál modelo matemático se le aplicará a la lectura obtenida para convertirla en una medición de humedad, para realizar esta comparación se utilizó un "if" tal y como se muestra en la ilustración 3-8.

```
if(vh400<=1.2511){
  Hsuelo=((0.1782*vh400)-0.0339)*100;
}else{
  Hsuelo=((0.4422*vh400)-0.3703)*100;
}
```

Ilustración 3-8 Comparación de la lectura del sensor de humedad de sustrato.

Una vez que se ha realizado la comparación, se dcide cuál de los dos modelos matemáticos obtenidos de las pruebas de humedad se le aplica a la lectura del sensor VH400.

Por último, se envían las lecturas obtenidas por los sensores, para ejecutar esta tarea, lo mejor es enviar todos los datos en una cadena en formato de texto, ya que, si se enviaran en formato numérico, se dificultaría más porque sería enviar de uno en uno. Por tanto, lo primero es convertir los datos de formato numérico a formato texto, para convertir de "float" a "String", se hace uso de la función "dtostrf", el problema es que, esta función solo es capaz de convertir un dato a la vez, esto se pudo resolver almacenando las lecturas en un arreglo de datos "float" y utilizando un ciclo "for" para convertir los datos uno por uno y concatenarlos en una variable auxiliar.

```
float luz = lux;
lecturas [0] = Hsuelo;
lecturas [1] = luz/1000;
lecturas [2] = EXP1;
lecturas [3] = EXP2;
lecturas [4] = Tem;
lecturas [5] = Hum;
for (i = 0; i <= 5; i++) {
  float valor = lecturas [i];
  dtostrf(valor, 4, 2, conversion);
  datos += conversion;
  if (i<=4){
    datos += " ";
  }else{
      datos += " #";
  }
}
Serial.println(datos);
datos = "";
delay(100);
}
```

Ilustración 3-9 Envía las lecturas obtenidas mediante el puerto serie.

Es importante mencionar que, para distinguir un dato de otro, en la cadena de texto cada dato está separado por un espacio; además, al final de la cadena de texto, se agregó un carácter especial "#", el cual sirve para indicarle al microcontrolador del módulo maestro, hasta qué punto termina una cadena de datos e inicia una nueva. S i se visualizaran los datos que se envían por el puerto utilizando el monitor serie del IDE de Arduino, se debe observar una cadena de datos como la que se muestra en la ilustración 3-10.

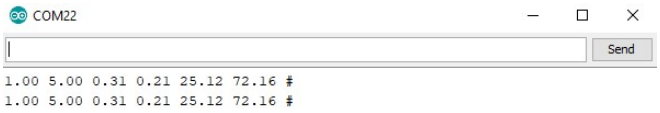


Ilustración 3-10 Trama de datos de los módulos 1, 2 y 3.

3.3.5 Modelo 3D para los módulos 1, 2 y 3

Para que los sensores puedan tomar lecturas, es necesario que se encuentren en contacto directo con las condiciones climáticas que se encuentran dentro del invernadero, por lo que los microcontroladores y todos los componentes electrónicos que conforman esta etapa también se ven expuestas a las condiciones ambientales que se generan dentro del invernadero, lo cual los expone a un ambiente húmedo, esto favorece la oxidación de las partes metálicas tales como pines de conexión, cables o pistas de cobre, además de favorecer la acumulación de suciedad producto del polvo o por insectos que se encuentren dentro. Para poder disminuir estos efectos, fue necesario que los componentes electrónicos que no necesiten estar en contacto directo con las condiciones climáticas del invernadero, se encuentren aislados dentro de un contenedor que los proteja.

El diseño del contenedor para alojar los componentes electrónicos más críticos, se realizó con el uso de impresión 3D, para el diseño, se utilizó el software SolidWorks, el diseño consta de dos partes: contenedor y la tapa del contenedor, estos dos elementos se diseñaron con base a las dimensiones de la PCB que se alojara en su interior, además de agregar elementos que faciliten su instalación dentro del invernadero.

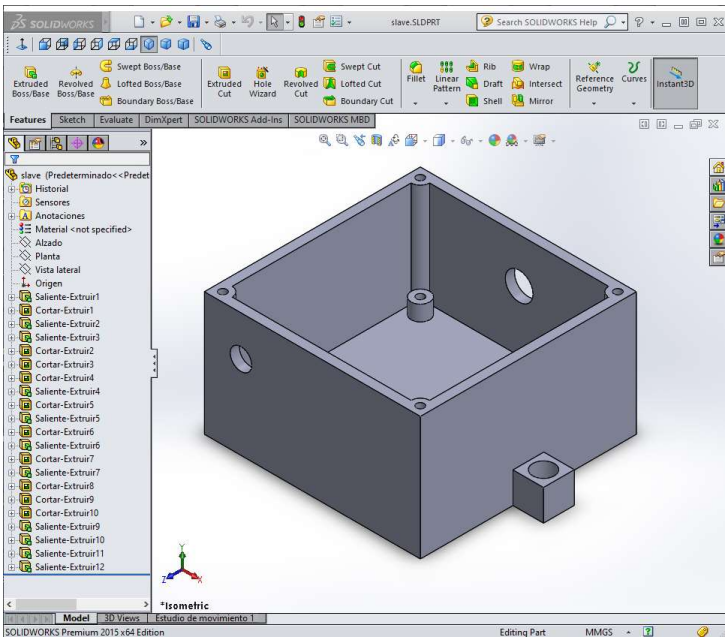


Ilustración 3-11 Modelo 3D para el contenedor de los módulos 1, 2 y 3.

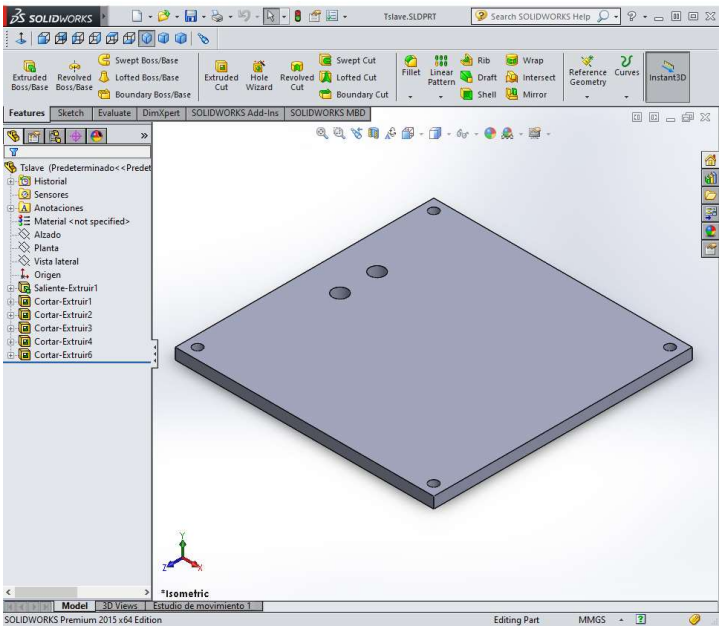


Ilustración 3-12 Modelo 3D de la tapa del contenedor de los módulos 1, 2 y 3.

3.4 Gestión de datos (módulo maestro)

Esta etapa está conformada por una tarjeta de desarrollo Arduino Mega, que equipa el microcontrolador Atmega2560, se eligió esta placa de desarrollo debido a que posee cuatro puertos seriales (UART) físicos y esto facilita mucho el código de programación para la recepción de datos, además dispone de un mayor número de terminales que perite una futura

versión. Este microcontrolador es el encargado de recibir las tres cadenas de datos provenientes de la etapa de adquisición de datos, una vez que ha recibido exitosamente las tres cadenas de datos, las une en una sola y la envía a la interfaz gráfica de usuario mediante el puerto USB que equipa la tarjeta de desarrollo Arduino Mega, por último, este microcontrolador también es el encargado de realizar las comparaciones necesarias para determinar si se envía una alerta GSM o no. Además de la tarjeta Arduino mega, esta etapa también se compone de tres circuitos integrados MAX485, que son los encargados de convertir el protocolo RS485 proveniente de la etapa 1, en niveles TTL para poder efectuar la comunicación entre los microcontroladores de la etapa de adquisición y el microcontrolador Atmega2560, además esta etapa se compone del módulo SIM800L que es el encargado de enviar las alertas GSM al agricultor en caso de que algún parámetro se encuentre en un estado crítico.

3.4.1 Diagrama eléctrico para módulo maestro

En la ilustración 3-13 se puede observar el diagrama esquemático para el módulo maestro, en este diagrama se incluye una tarjeta de desarrollo Arduino Mega el cual se encarga de controlar a los demás dispositivos, también se incluyen tres circuitos integrados MAX485 que sirven para convertir la información de formato RS485 a niveles TTL para que el Arduino Mega pueda recibir la información que ellos módulos 1, 2 y 3 le envían, se incluyó un puerto que permite conectar una fuente de voltaje de 12 Vdc el cual sirve para alimentar tanto a los dispositivos que conforman el módulo maestro así como también a los módulos 1, 2 y 3, aunque antes alimentar a los dispositivos de este módulo, se debe disminuir a un voltaje de 5 Vdc, para reducir el voltaje se hace uso de un regulador LM7805, todas las conexiones del sistema, están protegidas mediante un fusible de 1A que permite desconectar el sistema en caso de que se produzca algún corto circuito en las conexiones eléctricas, por último, también se incluyó un módulo SIM800L que permite el envío de alertas vía GSM.

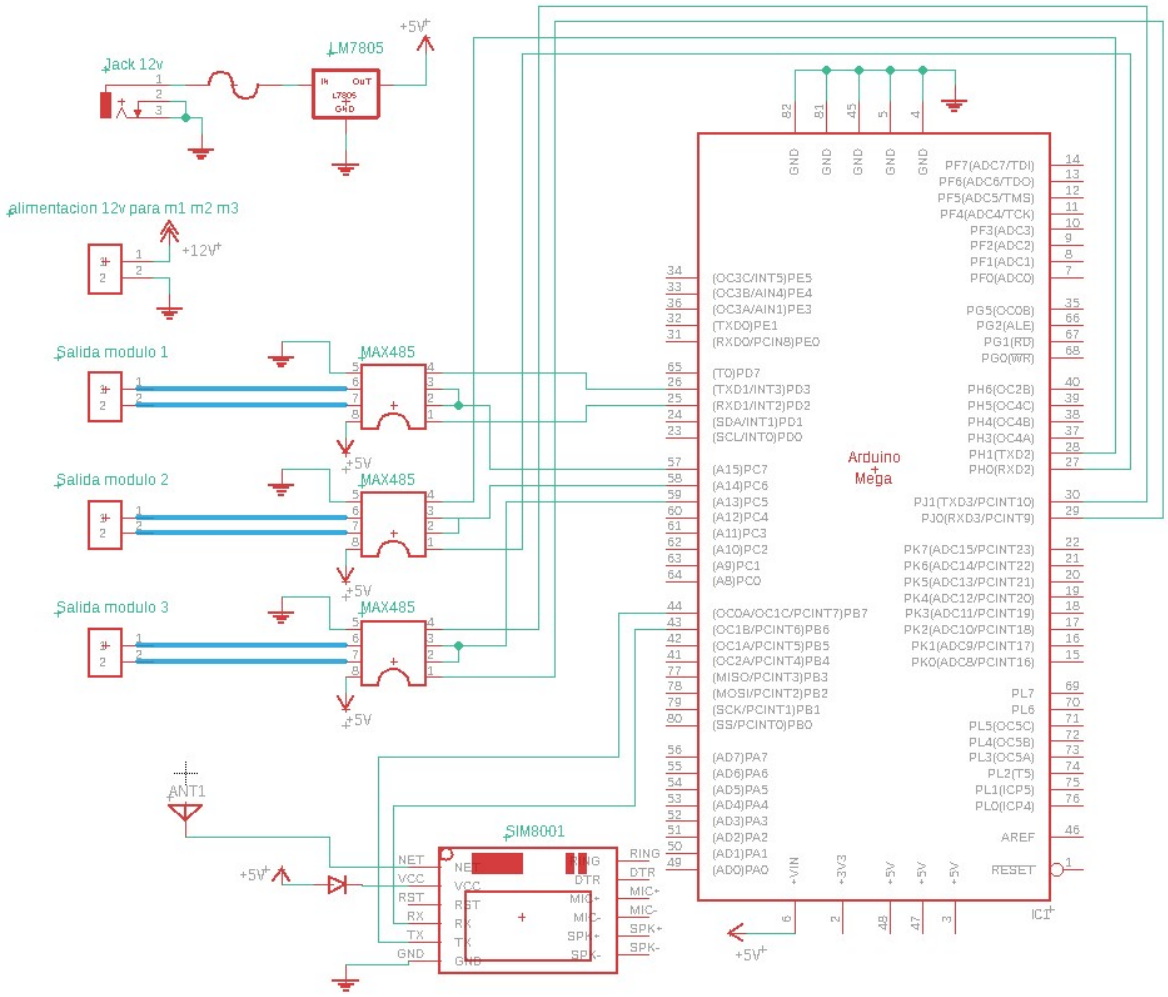


Ilustración 3-13 Diagrama esquemático para el módulo maestro.

3.4.2 PCB para módulo maestro

En la ilustración 3-14 se muestra el circuito impreso para el módulo maestro, este circuito impreso se realizó mediante el software de diseño Autodesk Eagle.

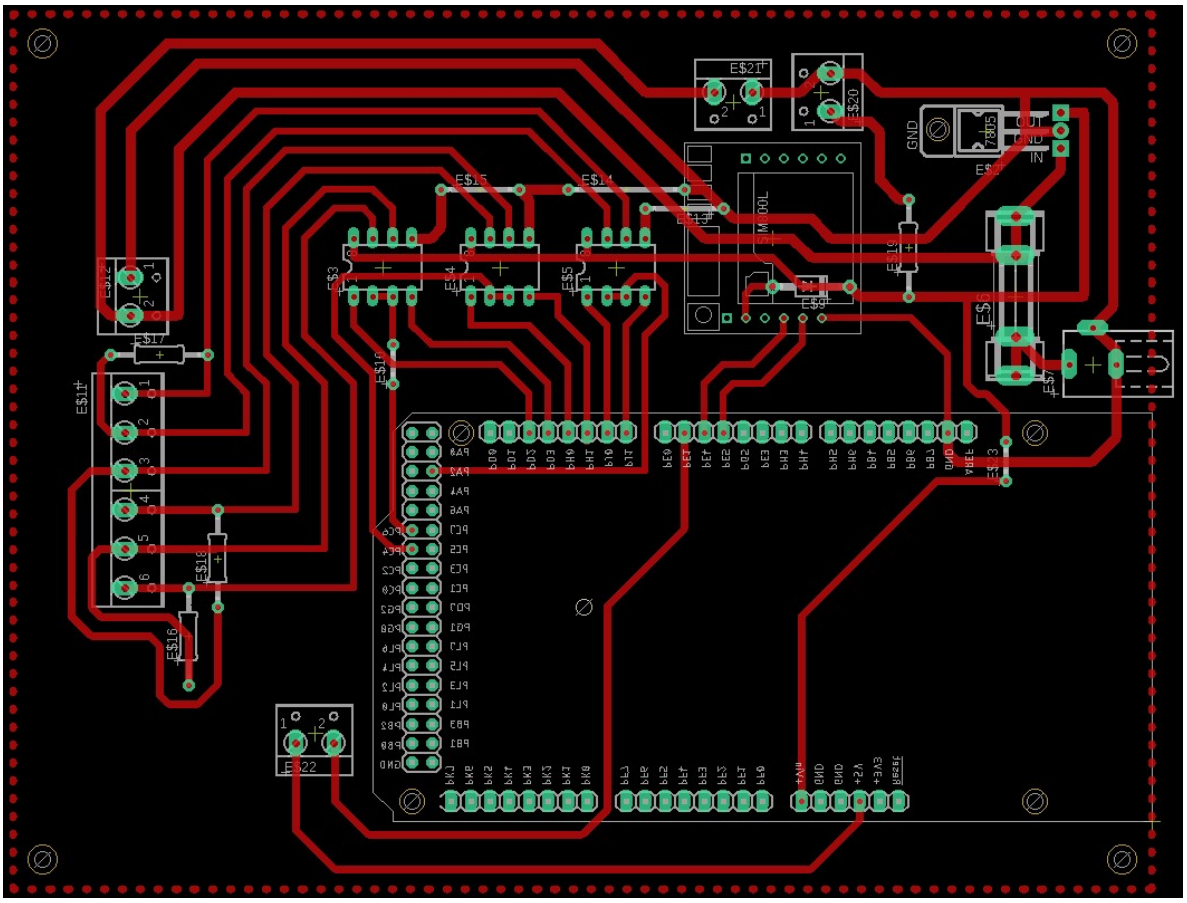


Ilustración 3-14 PCB para módulo maestro.

3.4.3 Diagrama de flujo para módulo maestro

El diagrama de flujo que aparece en la ilustración 3-15, pertenece al algoritmo de programación con el cual funciona el microcontrolador Atmega2560, su función principal es recibir todas las lecturas de los sensores, unirlas todas en una sola cadena de datos y enviarla hacia la interfaz gráfica, además de comparar las lecturas obtenidas con los rangos establecidos por el agricultor y enviar alertas GSM en caso de ser necesario.

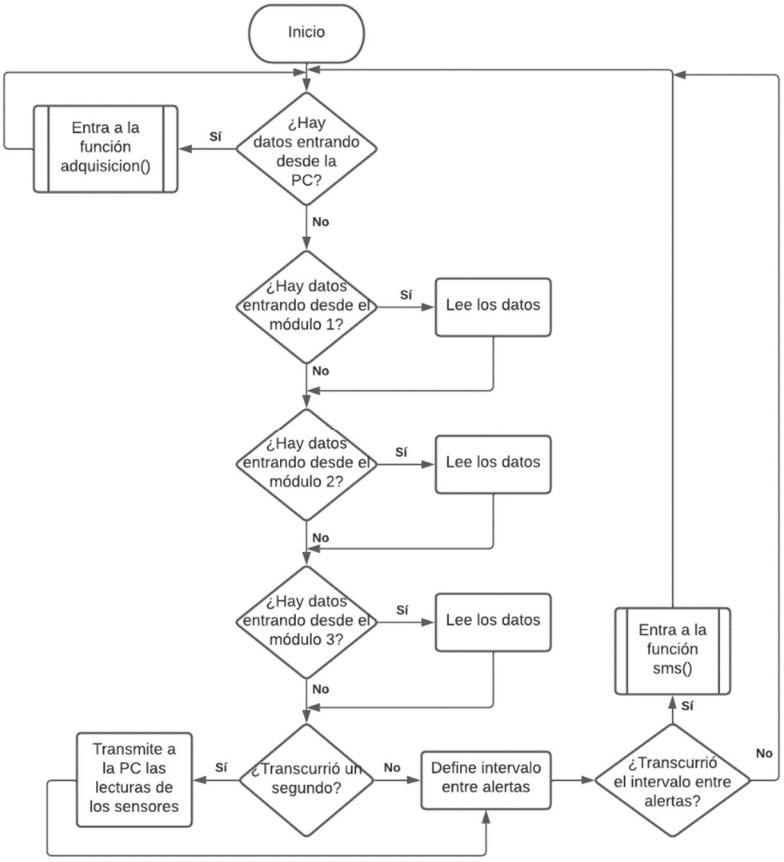


Ilustración 3-15 diagrama de flujo para módulo maestro.

En esta parte del diagrama de flujo, la principal función es la del control de temporizadores que permiten la establecer la frecuencia de muestreo, así como el intervalo de tiempo que debe de transcurrir en entre la evaluación de los valores actuales de las lecturas de los sensores y los umbrales mínimos y máximos para el envío de alertas vía GSM.

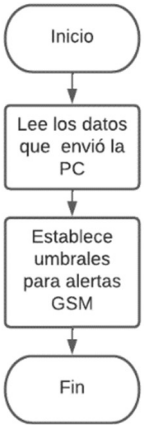


Ilustración 3-16 Función "adquisicion ()".

Esta parte del diagrama de flujo solo tiene la función de recibir los datos que la interfaz gráfica le envíe al módulo maestro para establecer umbrales mínimos y máximos para el envío de alertas vía GSM.

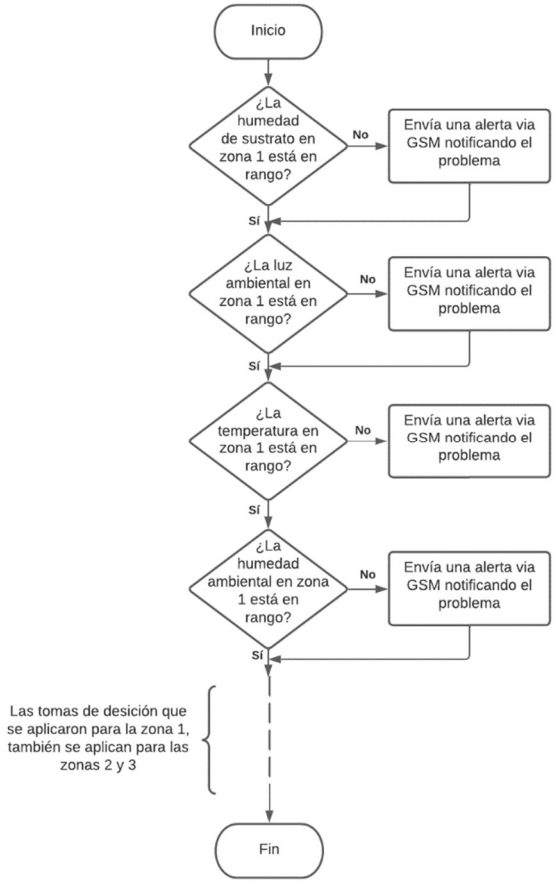


Ilustración 3-17 Función "sms ()".

Esta última parte del diagrama de flujo se ejecuta cada determinado intervalo de tiempo, el cual por default es de 15 min, pero que el agricultor puede modificar desde la interfaz gráfica. Cuando esta parte del diagrama de flujo se ejecuta, se comparan las lecturas actuales de los sensores con los umbrales que el agricultor haya establecido.

3.4.4 Código fuente para módulo maestro

En este apartado se muestra el código fuente programado en el microcontrolador Atmega 2560. Siguiendo el diagrama de flujo, lo primero fue establecer la tasa de baudios para los 4 puertos serie del Atmega 2560, ya que este microcontrolador es compatible con la Arduino IDE, el código se realizó utilizando esta herramienta.

```
Serial.begin(250000);
Serial1.begin(115200);
Serial2.begin(115200);
Serial3.begin(115200);
```

Ilustración 3-18 Establece la tasa de baudios para los puertos serie.

Para establecer la tasa de baudios de los cuatro puertos serie, se utilizaron las líneas de código que se muestran en la ilustración 3-18.

El siguiente paso es, desactivar las alertas GSM, para poder desactivar las alertas de alguna variable, fue necesario forzar estos rangos de tal forma que las lecturas proporcionadas por los sensores nunca se encuentren fuera este rango, por ejemplo, el rango de lecturas que puede proporcionar el sensor de BME280 para humedad ambiental, es de 0 como mínimo y como máximo de 100, al forzar el rango de comparación de -5 a 3000, la lectura proporcionada por el sensor nunca se va a encontrar fuera de rango, por lo que nunca se enviara la alerta, este mismo principio se aplica con las demás variables.

```
HSmin=-5;
Lmin=-5;
Tmin=-100;
HAmin=-5;
HSmax=3000;
Lmax=3000;
Tmax=3000;
HAmax=3000;
Minutos = 15;
```

Ilustración 3-19 Desactiva las alertas GSM.

Se puede observar que a una variable llamada “Minutos” se le asigna un valor de 15, esta variable sirve para determinar la cantidad de minutos que el microcontrolador debe esperar antes de evaluar si alguna variable dentro del invernadero está fuera del rango deseado, esto se decidió de esta forma, ya que si se realizara esta evaluación con la misma frecuencia con la que se envían los datos a la interfaz gráfica (1 segundo), en caso de que una o más variables se encontraran fuera de rango, se enviaría varios mensajes notificando este mismo problema y sería un desperdicio de tiempo aire.

El siguiente paso fue, consultar si hay datos entrando por el puerto serie 0, esta consulta se logró con la línea de código que se muestra en la ilustración 3-20.

```
void loop() {
   if (Serial.available()){
```

Ilustración 3-20 Consulta de datos para el puerto serie 0.

Ya que la interfaz gráfica se comunica con el módulo maestro mediante el puerto serie 0, con esta consulta se puede determinar si la interfaz gráfica está enviado datos y, como los únicos datos que la interfaz gráfica puede enviar al módulo maestro son aquellos utilizados para establecer rangos deseados para el envío de alertas, cuando esta condición se cumpla se manda llamar a una subrutina llamada "adquisición ()" que permite recibir estos datos y almacenarlos en sus respectivas variables (Ilustración 3-21).

```
void adquisicion(){
  String recibidos = Serial.readString();
  String lec1 = s.separa(recibidos, ',', 0);
 HSmin = lec1.toInt();
  String lec2 = s.separa(recibidos, ',', 1);
 HSmax = lec2.toInt();
  String lec3 = s.separa(recibidos, ',', 2);
 Lmin = lec3.toInt();
  String lec4 = s.separa(recibidos, ',', 3);
 Lmax = lec4.toInt();
  String lec5 = s.separa(recibidos, ',', 4);
 Tmin = lec5.toInt();
  String lec6 = s.separa(recibidos, ',', 5);
 Tmax = lec6.toInt();
  String lec7 = s.separa(recibidos, ',', 6);
 HAmin = lec7.toInt();
  String lec8 = s.separa(recibidos, ',', 7);
 HAmax = lec8.toInt();
  String lec9 = s.separa(recibidos, ',', 8);
 Minutos = lec9.toInt();
}
```

Ilustración 3-21 Subrutina de adquisición de datos de la interfaz gráfica.

Como los datos que envía la interfaz gráfica están todos unidos en una cadena en formato texto, es necesario separar todos los datos y convertirlo nuevamente a un formato numérico, para realizar estas acciones se hace uso de la biblioteca "Separador.h" y de la función "toInt ()".

En caso de que la condición de la ilustración 3-20, no se cumpla, el microcontrolador procede a recibir las cadenas de datos que los módulos 1, 2 y 3 estén enviando, para que el microcontrolador pueda distinguir hasta donde termina una cadena de datos y empieza una nueva, se utiliza el carácter "#" que, el microcontrolador de los módulos de la etapa de adquisición tiene que colocar al final de cada cadena de datos (Ilustración 3-22).

```
} else{
   while (Serial1.available()){
     str = Serial1.readStringUntil('#');
   }
   while (Serial2.available()){
     str1 = Serial2.readStringUntil('#');
   }
   while (Serial3.available()){
     str2 = Serial3.readStringUntil('#');
   }
```

Ilustración 3-22 Recibe los datos de los puertos 1, 2 y 3.

```
void loop() {
   if (Serial.available()){
    adquisicion();
   } else{
      while (Serial1.available()){
        str = Serial1.readStringUntil('#');
      }
      while (Serial2.available()){
        str1 = Serial2.readStringUntil('#');
      }
      while (Serial3.available()){
        str2 = Serial3.readStringUntil('#');
      }
```

Ilustración 3-23 Envía datos a la interfza gráfica.

Habiendo recibido las cadenas de datos, el siguiente paso es enviarlas a la interfaz gráfica mediante el puerto serie 0, pero antes, es necesario unir estas tres cadenas de datos en una sola, para esto solo hace falta concatenar las tres cadenas y almacenarlas en una variable tipo "String" que servirá para almacenar toda esta cadena de datos de forma temporal. Y posteriormente enviarla cada segundo.

```
   if(millis() > Timer + 1000){
      Timer = millis();
      Datos="";
      Datos += str;
      Datos += str1;
      Datos += str2;
      Serial.println(Datos);
      str ="0.0 0.0 0.0 0.0 0.0 0.0 ";
      str1="0.0 0.0 0.0 0.0 0.0 0.0 ";
      str2="0.0 0.0 0.0 0.0 0.0 0.0 ";
    }
```

Ilustración 3-24 Envía datos hacia la interfaz gráfica.

Es importante mencionar que en este caso no se utilizó la función “delay ()” para regular el envío de datos a la interfaz gráfica, ya que esta función produce retrasos en todo el código y como se utilizan distintos temporizadores, todos se verían afectaos por la función “delay ()”.

Por último, se limpia la variable auxiliar de datos y se le asignan valores en 0 a las variables que contienen a las cadenas de datos de los módulos de adquisición, esto se hace por si en algún momento alguno de los módulos se llega a dañar o desconectar, se rellene la cadena de datos con valores en 0 y no produzca errores en la interfaz gráfica.

Siguiendo el diagrama de flujo, el siguiente paso es evaluar si alguna de las variables del invernadero se encuentra fuera del rango deseado, pero primero hay que recordar que esta evaluación es realiza cada cierta cantidad de minutos que el agricultor haya establecido, por lo que primero, es necesario esperar esta cantidad de tiempo, nuevamente, para realizar este temporizador no se puede hacer uso de la función “delay ()” así que se utiliza un temporizador igual al utilizado en el paso anterior (Ilustración 3-25).

```
        Conversion = Minutos * 60000;
        if (millis() > TimAlert + Conversion){
            TimAlert = millis();
            sms();
        }
      }
}
```

Ilustración 3-25 Temporizador para el envío de alertas.

En caso de que haya cumplido el tiempo que marca el temporizador, se manda llamar a la subrutina llamada “sms ()” la cual se encarga de comparar el rango optimo establecido por el agricultor con el estado actual de las variables dentro del invernadero (Ilustración 3-26).

```
void sms(){
   String Dat0 = s.separa(Datos, ' ', 0);
   if (Dat0.toInt()<HSmin || Dat0.toInt()>HSmax){
        Sim800l.sendSms("+525618747131","ATENCION, LA HUMEDAD EN EL SUSTRATO DE ZONA 1 ESTA FUERA DEL RANGO ESTABLECIDO");
       }
   String Dat1 = s.separa(Datos, ' ', 1);
   if (Dat1.toInt()<Lmin || Dat1.toInt()>Lmax){
        Sim800l.sendSms("+525618747131","ATENCION, LA LUZ AMBIENTAL DE ZONA 1 ESTA FUERA DEL RANGO ESTABLECIDO");
       }
```

Ilustración 3-26 Envió de alertas vía GSM.

Para comparar el rango con el estado actual de las variables, se toma una muestra de la cadena de datos más reciente que haya obtenido el microcontrolador y, como esta cadena de datos está en formato texto y los datos esta unidos, primero es necesario separar cada dato y

convertirlo a un formato numérico, nuevamente se hace uso de la biblioteca “Separador.h” y de la función “toInt ()”.

3.4.5 Modelo 3D para módulo maestro

Para evitar que los componentes electrónicos que componen esta etapa se deterioren muy rápido, fue necesario alojar en un contenedor todos los componentes electrónicos, aunque estos componentes no están en contacto directo con las condiciones ambientales del invernadero, no es necesario tener un sistema tan hermético como en la etapa de adquisición, pero si es necesario proteger estos componentes.

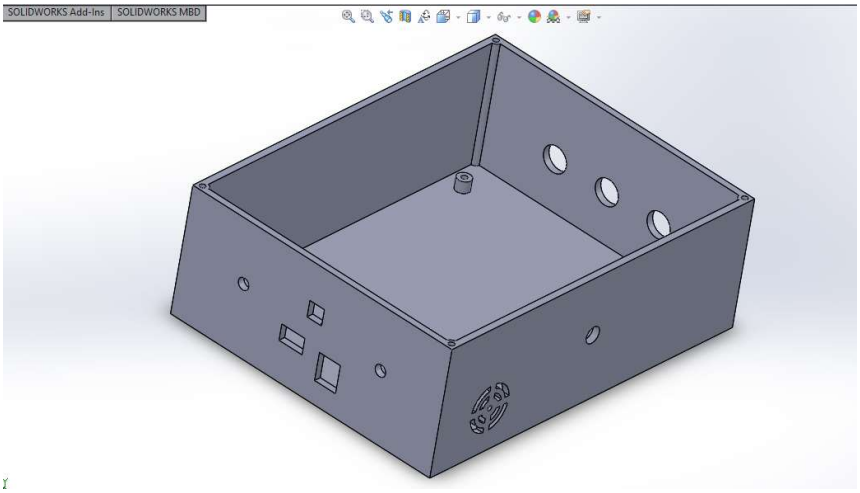


Ilustración 3-27 Modelo 3D para contenedor del módulo maestro.

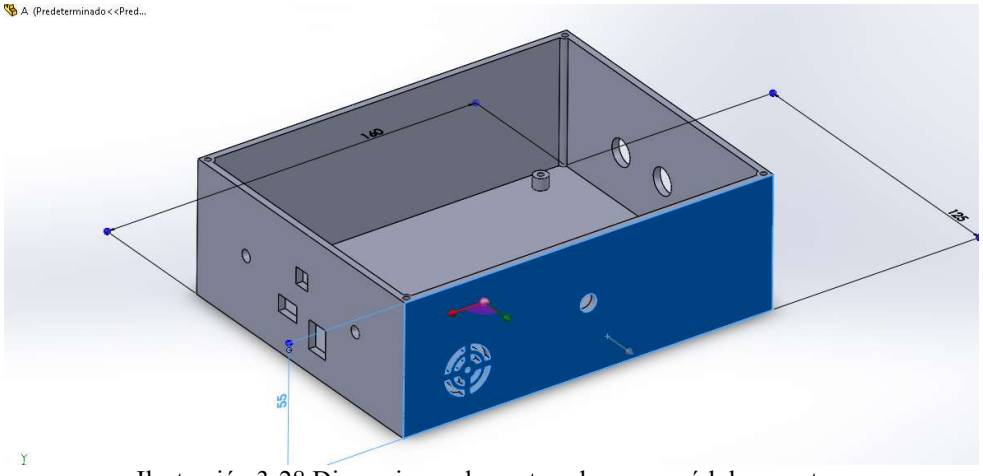


Ilustración 3-28 Dimensiones de contenedor para módulo maestro.

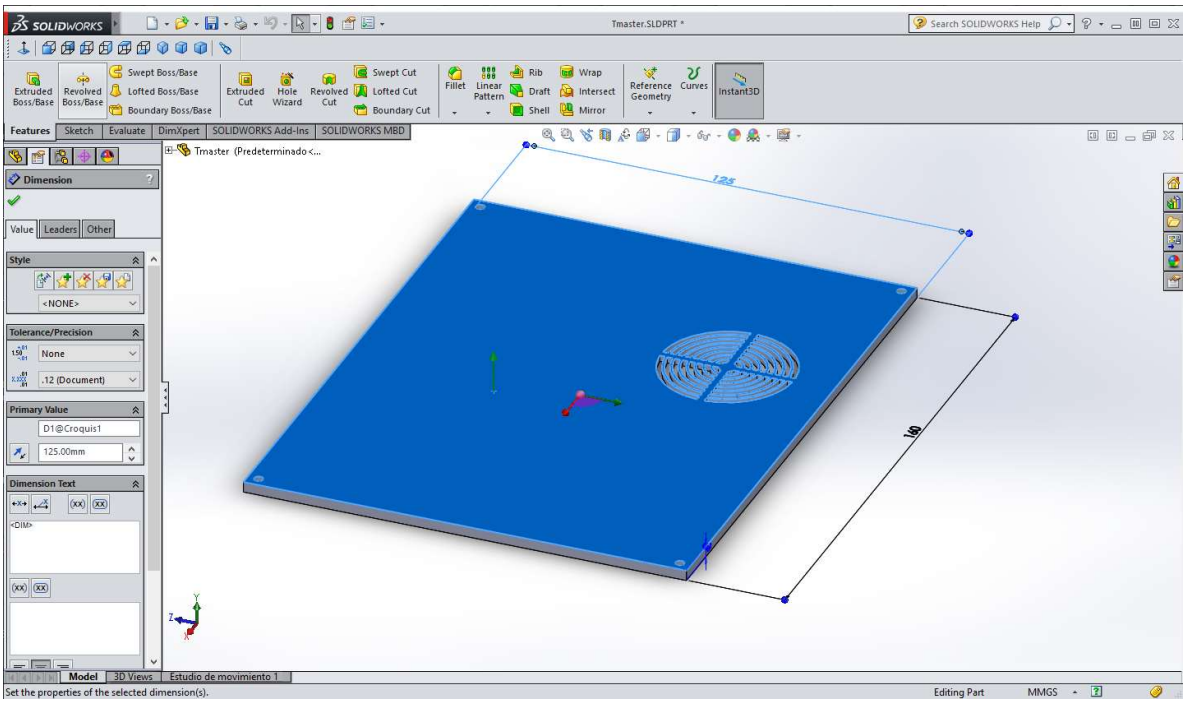


Ilustración 3-29 Tapa de contenedor para contenedor del módulo maestro.

3.5 Interfaz gráfica de usuario y base de datos

Esta etapa se dividió en 4 partes principales: adquisición y envío de datos, creación de base de datos, registro en base de datos, algoritmo para consulta de históricos.

Esta etapa se diseñó en el software de desarrollo LabVIEW, el cual funciona sobre una computadora y un sistema operativo, específicamente se utilizó Windows 10, la versión de LabVIEW sobre la cual fue desarrollada la interfaz gráfica, es la versión 2017 de 32 bits. Para el desarrollo de la interfaz gráfica, fue necesario contar con dos paquetes de expansión adicionales para LabVIEW, los cuales son: NIVisa y Database connectivity toolkit versión 2017.

3.5.1 Diseño del entorno gráfico

Esta parte de la interfaz tiene dos funciones principales, la adquisición de datos, la cual se encarga de obtener los datos provenientes del módulo maestro y graficarlos para que el agricultor pueda observarlos de una forma sencilla, la segunda función que tiene esta parte de la interfaz, es la de establecer los rangos aceptables de las variables para el envío de alertas GSM.

El primer paso fue crear un archivo nuevo de LabVIEW, esto se logra ejecutando el programa LabVIEW y creando un proyecto en blanco como se muestra en la ilustración 3-30.

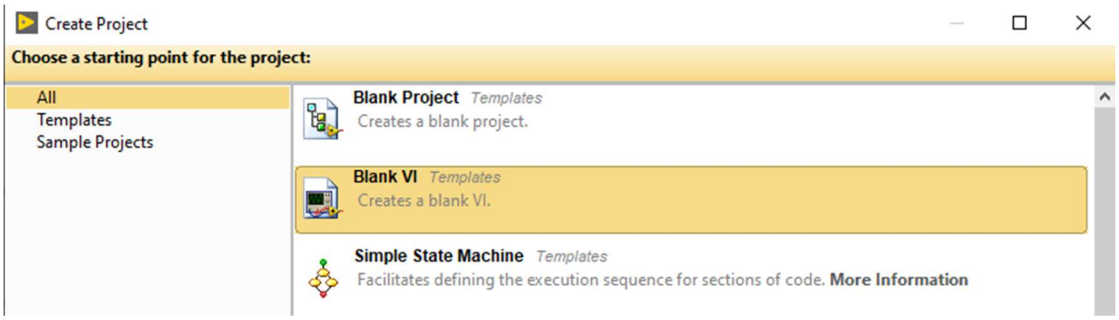


Ilustración 3-30 nuevo archivo LabVIEW.

Una vez creado el proyecto en blanco, se abren dos ventanas, una es el panel frontal y la segunda es la ventana de diagrama de bloques (ilustración 3-31).

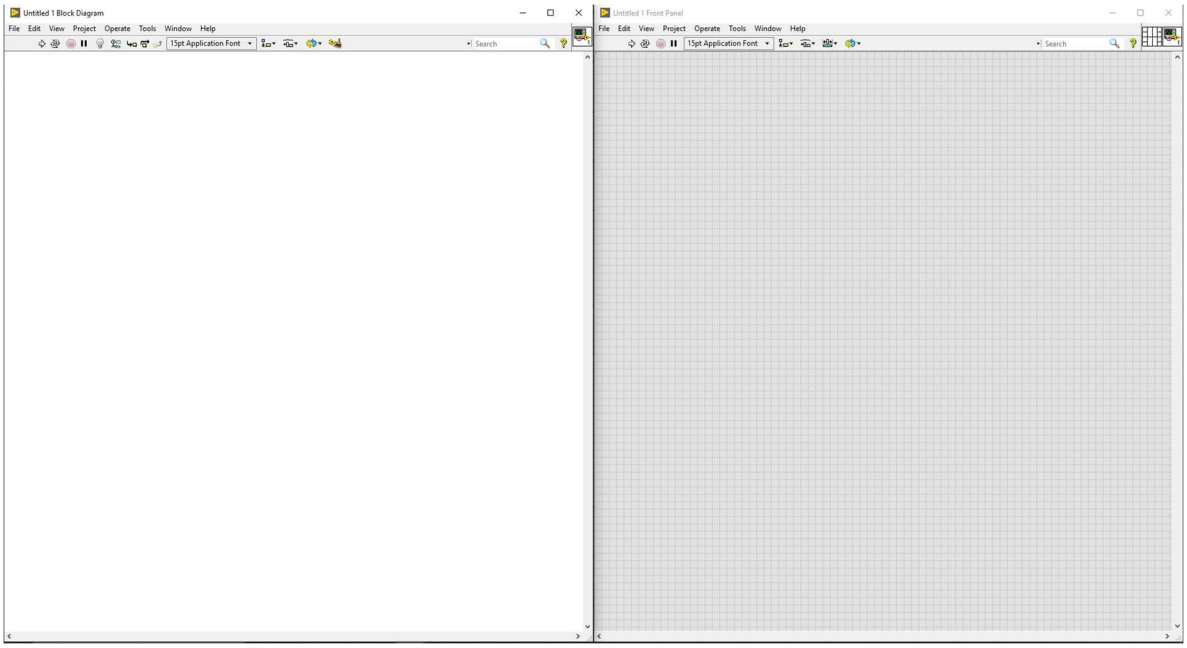


Ilustración 3-31 Panel frontal y diagrama de bloques.

En el panel frontal se diseñó la parte visual del sistema. Para mayor comodidad del agricultor, se separaron por secciones cada uno de los apartados con los que cuenta el sistema, primero fue necesario desplegar la paleta de controles, para realizar esta acción, solo es necesario dar un clic derecho en cualquier parte en blanco del panel frontal, lo siguiente es seleccionar el apartado con el nombre “containers” y seleccionar el que dice “tab control”, por último, se coloca en cualquier parte del panel frontal, en este caso se colocó en el centro del espacio de

trabajo ya que este tablero es el que contendrá todos los indicadores del sistema (Ilustración 3-32).

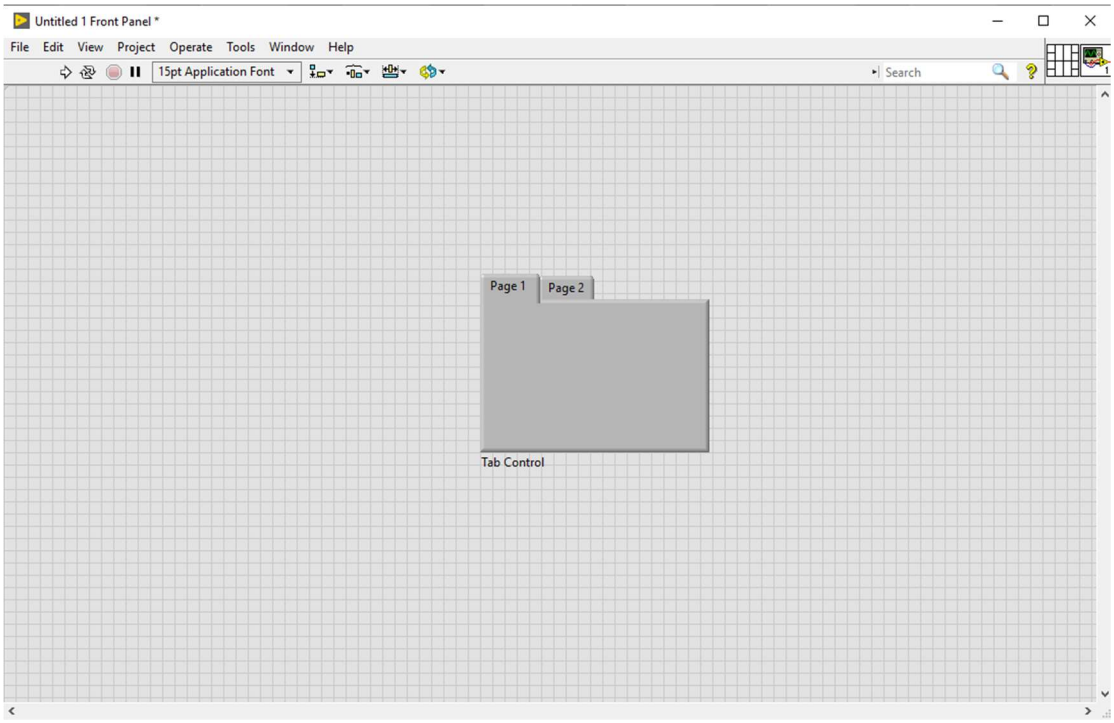


Ilustración 3-32 Tab control.

El siguiente paso realizado fue, modificar el tamaño del contenedor que se colocó, de tal forma que tenga el suficiente espacio para alojar todos los indicadores gráficos que utiliza el sistema, esto se logra dando clic izquierdo en una esquina del contenedor y manteniendo el clic para redimensionar el tamaño del contenedor tanto como sea conveniente. Como se puede observar en la ilustración 3-31, el contenedor solo contaba con dos apartados, llamados "page 1" y "page 2", y este sistema cuenta con cinco apartados diferentes, para poder añadir más apartados al contenedor solo es necesario dar clic derecho en la pestaña con el nombre "page 2", al hacer esto se desplegará un menú, dentro de dicho menú hay que seleccionar la opción que dice "add page after", de esta forma se pueden agregar nuevos apartados (Ilustración 3-33).

Ilustración 3-33 Agregar apartados al tab control.

Una vez agregados los cinco apartados, se les cambió el nombre según fuera la función que realizan (Ilustración 3-34) los nombres para cada apartado son los siguientes: “AJUSTES”, “ZONA 1”, “ZONA 2”, “ZONA 3” y “CONSULTAS”.

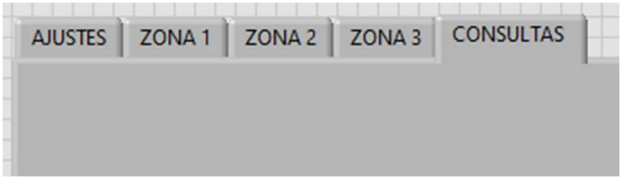


Ilustración 3-34 Apartados del entorno gráfico.

3.5.1.1 Apartado de zonas 1, 2 y 3

Ya que están nombrados cada uno de los apartados, lo siguiente fue agregar los controles e indicadores gráficos. Primero se agregaron los indicadores de los apartados zona 1, 2 y 3, mismos que proporcionan las lecturas actuales de los sensores y están separados por zonas, para agregar estos indicadores, hay que seleccionar en la paleta de herramientas la opción con el nombre de “graph” y seleccionar “waveform chart”, se colocan seis de estos indicadores gráficos y se acomodan en orden (Ilustración 3-35), el mismo procedimiento se realiza en las zonas 2 y 3. Una vez colocados los indicadores, solo resta modificar el nombre de cada uno según los datos que muestra, para realizar esto solo es necesario dar doble clic sobre el texto “amplitud” y escribir el nombre correspondiente.

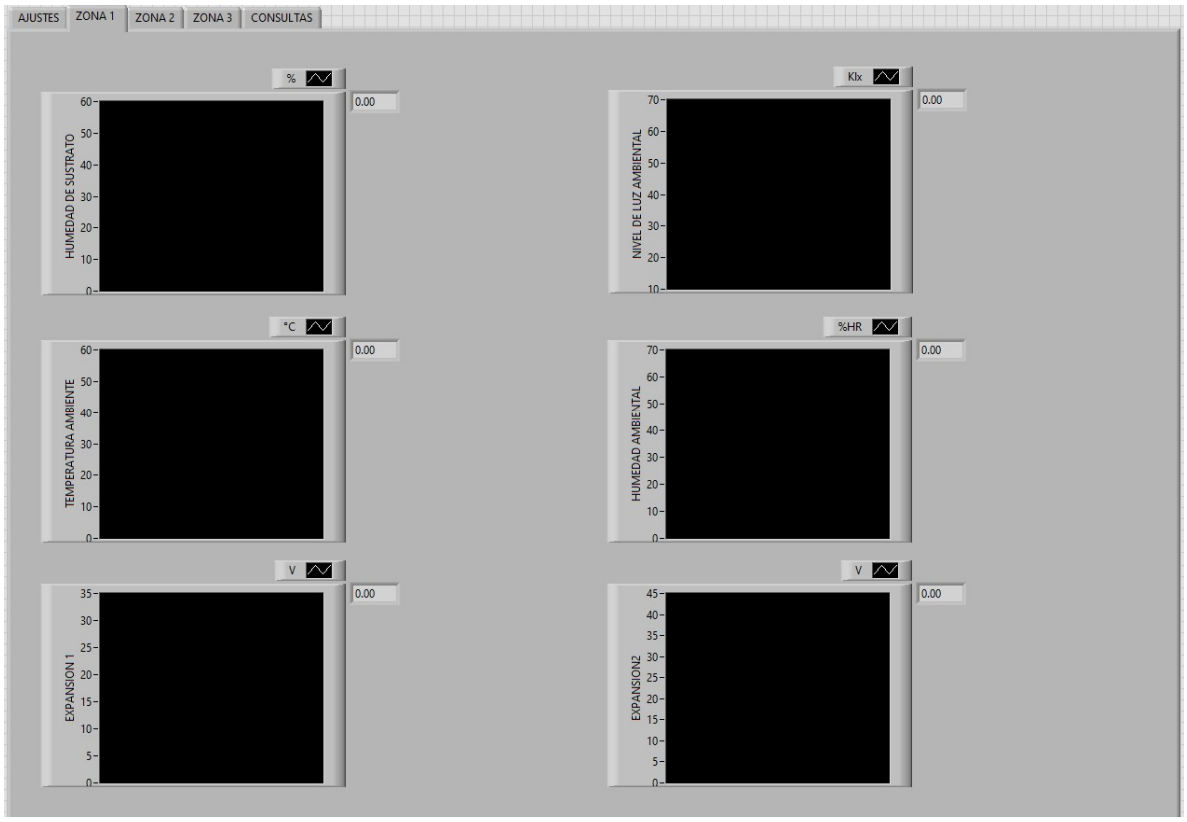


Ilustración 3-35 Apartados "zona 1, 2 y 3".

3.5.1.2 Apartado de ajustes

Este apartado contiene controles numéricos que el agricultor puede modificar para establecer rangos de lecturas óptimas para el envío de alertas GSM.

La primera consideración que se tomo fue agregar algún tipo de señalizador que sirva para que el agricultor pueda identificar si las alertas están activadas o desactivadas, el indicador utilizado es un LED, este se encuentra en la paleta de controles, en el apartado “Boolean” y tiene como nombre “Square LED”, una vez colocado, se debe redimensionar de tal forma que tenga el tamaño suficiente para alojar los controles numéricos y los dos botones que controlan las alertas. Una vez redimensionado, se agrega un contenedor que aloje los controles para este apartado del sistema, las dimensiones de este contenedor deben ser inferiores que las del LED colocado anteriormente, de tal forma que al colocar el contenedor dentro del LED se forme una especie de margen (Ilustración 3-36), este contenedor se encuentra en la paleta de controles, en el apartado “Decorations”, el contenedor utilizado tiene el nombre de “Raised Beveled Box”.

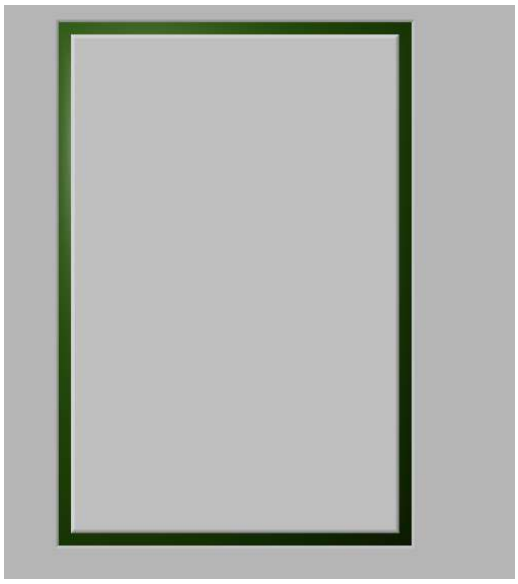


Ilustración 3-36 Indicador lumínico para los ajustes de alertas GSM.

Se agregaron nueve controles numéricos, que se pueden encontrar en la paleta de controles en el apartado “numeric” y tienen como nombre “numeric control” (Ilustración 3-37).

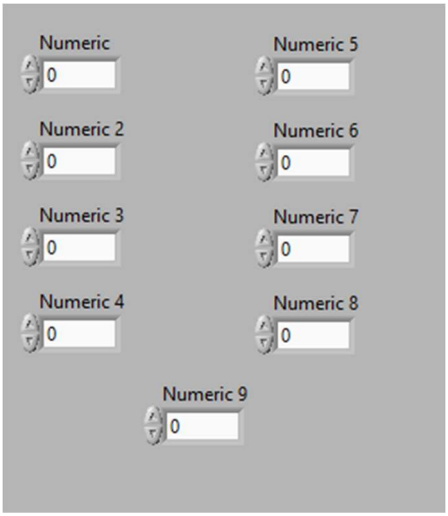


Ilustración 3-37 Controles numéricos para la asignación de rangos.

Habiendo colocado los controles numéricos, se le asignó un nombre a cada uno de los controles según sea su función, para cambiar el nombre solo basta con dar doble clic izquierdo sobre el nombre y modificarlo (Ilustración 3-38).

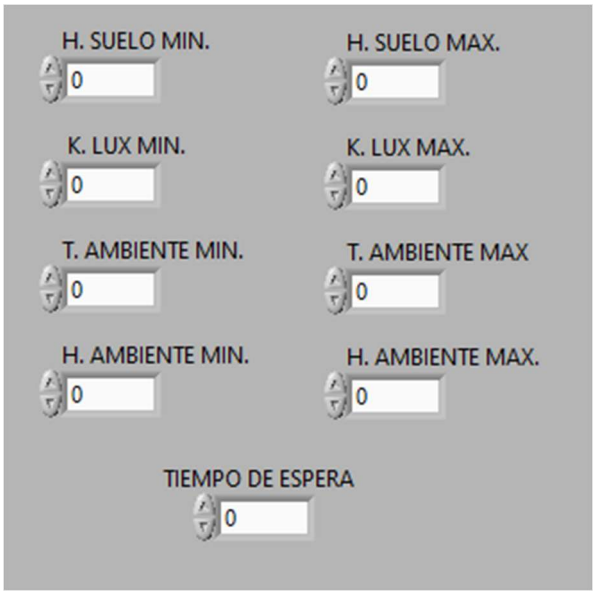


Ilustración 3-38 Asignación de nombre a los controles numéricos.

Fue necesario delimitar los valores que puede modificar el agricultor, ya que estos controles numéricos pueden adoptar valores negativos, así como valores mayores a los 1000, en este sistema no es necesario utilizar los valores negativos ya que ninguna lectura de ningún sensor proporciona valores tan grandes por lo no es necesario tener un rango tan amplio, se establece el valor más bajo en 0 y el valor más alto en 100. Para poder limitar los valores que pueden adoptar los controles numéricos, se da clic derecho sobre el control numérico, se desplegara un menú de opciones, se selecciona la opción “data entry…”, al seleccionar esta opción se abre una ventana emergente en la que se pueden modificar los parámetros del control numérico, la única opción que se debe modificar es la de “Data entry…” tal y como se observa en la ilustración 3-38.

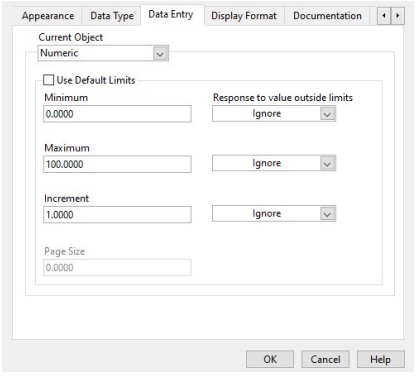


Ilustración 3-39 Delimitación de valores para controles numéricos.

Habiendo realizado la delimitación de valores de los controles numéricos, se colocaron dos botones: uno que sirve para activar las alertas y otro para desactivar las alertas, los botones se encuentran en la paleta de controles, en el apartado “Boolean” y tienen como nombre “OK button”. Teniendo los dos botones colocados, se les cambio el nombre para identificar la función que tiene cada uno, esto se logra dando doble clic sobre el nombre y remplazándolo por la función que realiza el botón. Hasta el momento, el apartado de “ajustes” de debe ver como se muestra en la ilustración 3-40.

Ilustración 3-40 Botones para activar y desactivar alertas.

Por último, se colocan todos estos controles dentro del espacio que se formó previamente con el LED y el contenedor (Ilustración 3-41).

Ilustración 3-41 Ajustes para alertas GSM.

Lo siguiente realizado en el apartado gráfico de “ajustes”, fue colocar un menú desplegable donde el agricultor pueda seleccionar el puerto de la computadora en el cual está conectado el módulo maestro del sistema. Esto se logra con el paquete de expansión NI Visa, en el apartado de diagrama de bloques, se despliega la paleta de controles dando clic derecho en cualquier parte vacía y se abrirá una pestaña emergente con todas las herramientas que se pueden utilizar en el diagrama de bloques, una vez abierta la paleta de controles, en la parte inferior de la ventana se encuentra un submenú con el nombre “Instrument I/O”, se selecciona la opción “Serial” y se selecciona el bloque “Configure port”, ya que está colocado el bloque “VISA configure port” (Ilustración 3-42), se puede observar que cuenta con varias terminales de conexión, se coloca el cursor en la terminal superior izquierda y se da clic derecho, se abrirá un menú en el cual se selecciona la opción “créate” y se selecciona la opción “control”, al realizar este procedimiento se creara un nuevo bloque que se une automáticamente con la terminal que se seleccionó previamente (Ilustración 3-43).

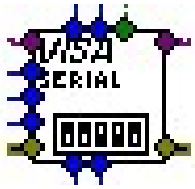


Ilustración 3-42 Visa configure port.

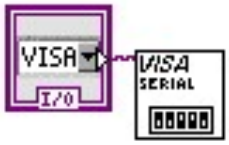


Ilustración 3-43 Visa resource name.

Habiendo realizado el paso anterior, en el panel frontal se creará automáticamente el menú desplegable anteriormente mencionado (Ilustración 3-44), con este menú es posible elegir el puerto COM de la computadora al que está conectado el módulo maestro, es importante mencionar que la computadora debe contar con los drivers necesarios para el dispositivo serie que se esté utilizando, en este caso, el Arduino Mega del módulo maestro utiliza el dispositivo CH340, si no se tienen los drivers necesarios, el dispositivo no aparecerá dentro de este menú.

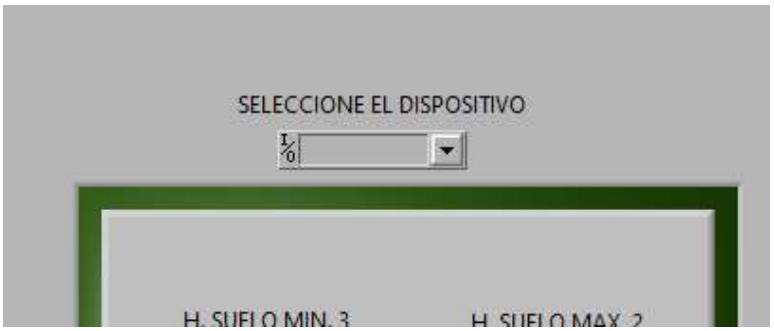


Ilustración 3-44 Selector de puerto serie.

A pesar de ya tener colocados todos los controles para el apartado de “ajustes”, aún quedaba bastante espacio en blanco disponible, para poder ocupar ese espacio en blanco y dar una mejor apariencia a este apartado, se colocó una imagen decorativa, esto se logra copiando una imagen con las dimensiones necesarias para aprovechar lo mejor posible el espacio y pegarla directamente en el espacio que se desea cubrir, tal y como se muestra en la Ilustración 3-45.

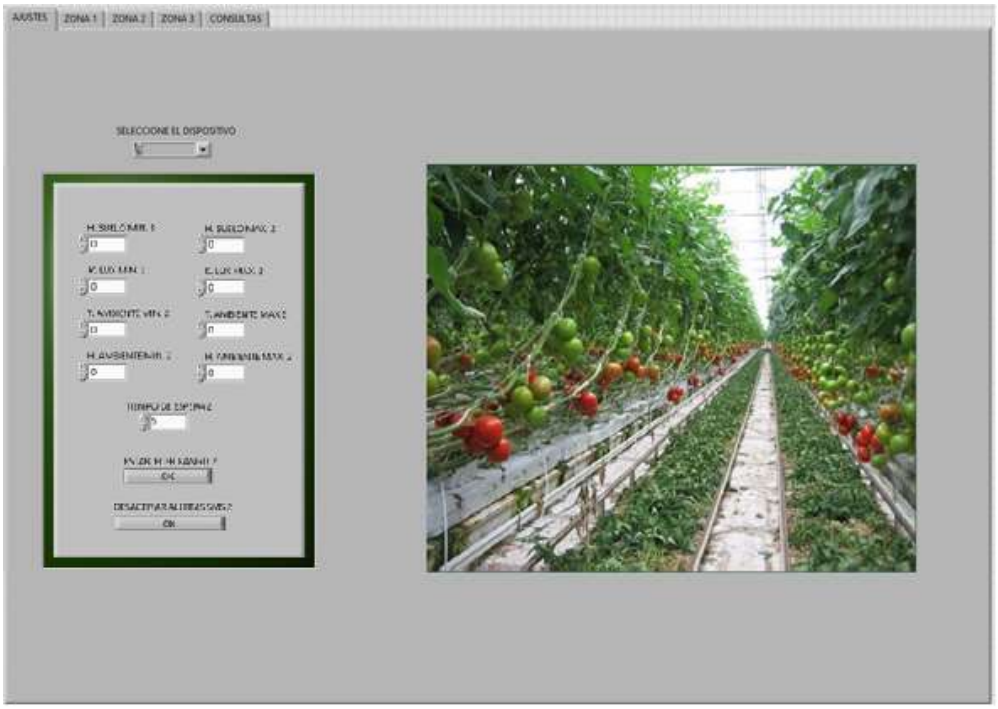


Ilustración 3-45 Apartado de ajustes con decoración.

3.5.1.3 Apartado de consultas

En el apartado de “consultas” se encuentran todo lo relacionado con la base de datos con la que cuenta el sistema, en este apartado se pueden consultar los datos históricos obtenidos de todos los sensores con los que cuenta el sistema, estos datos están ordenados por fechas y el

agricultor puede consultar los datos de cada sensor por un rango de dos fechas, la fecha inicial que indica al sistema desde donde se van a consultar los y una fecha final que es el límite de fechas en las que el sistema debe buscar.

Para empezar, se crearon dos calendarios desplegables mediante los cuales se puedan seleccionar las fechas inicial y final sin necesidad de escribirla manualmente. Para poder agregar estos calendarios, aunque primero fue necesario agregar en el diagrama de bloques, dos bloques “format date/time string” Uno para la fecha inicial y otro para le fecha final, mismo que se encuentra en la paleta de controles del diagrama de bloques, en la opción “timing” (Ilustración 3-47).

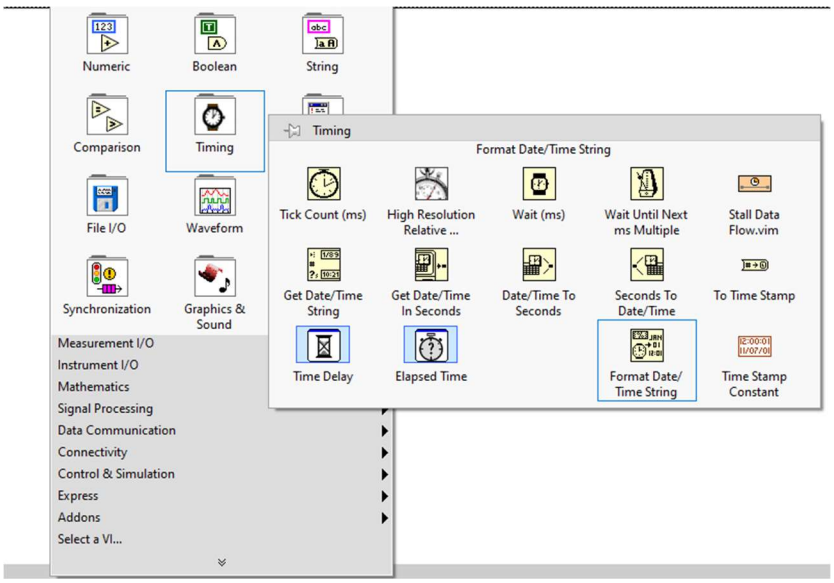


Ilustración 3-46 Format date/time string.

Con los bloques ya colocados, se puede observar que cuenta con varias terminales de conexión (Ilustración 3-46), en la terminal central izquierda (time stamp) se da clic derecho y en la opción “create” se selecciona “control”, esto hará que se genere un bloque adicional conectado al bloque “date format/time string” (Ilustración 3-47).

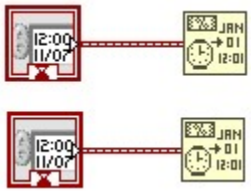


Ilustración 3-47 Time stamp control.

Los bloques que se colocaron al realizar el paso anterior, generan automáticamente en el panel frontal los calendarios anteriormente mencionados, solo fue cuestión de acomodarlos en apartado de “consultas” y renombrarlos (Ilustración 3-48).

Ilustración 3-48 Calendarios desplegable.

Una vez que ya se tenían los calendarios desplegables, el siguiente paso es crear un menú desplegable que permita seleccionar la variable que se va a consultar, para crear este menú, en la paleta de controles del panel frontal, en la opción “string & path” se selecciona la opción “combo box” se coloca dentro del apartado “consultas” y se cambia el nombre (Ilustración 3-49).

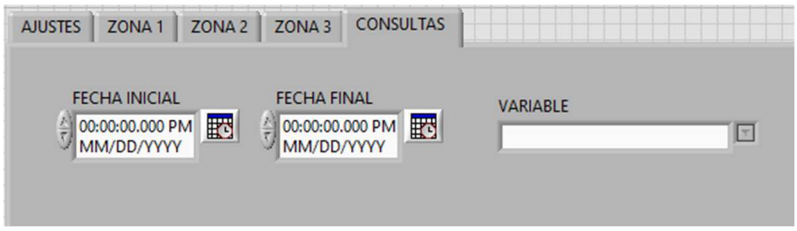


Ilustración 3-49 Menú de variables.

Se agregaron el nombre de las variables al menú, para lograr esto, se da clic derecho sobre el “combo box” que se acaba de colocar y se debe seleccionar la opción “Edit ítems…” al seleccionar esta opción, se abrirá una ventana emergente en la cual se pueden añadir valores (Ilustración 3-50).

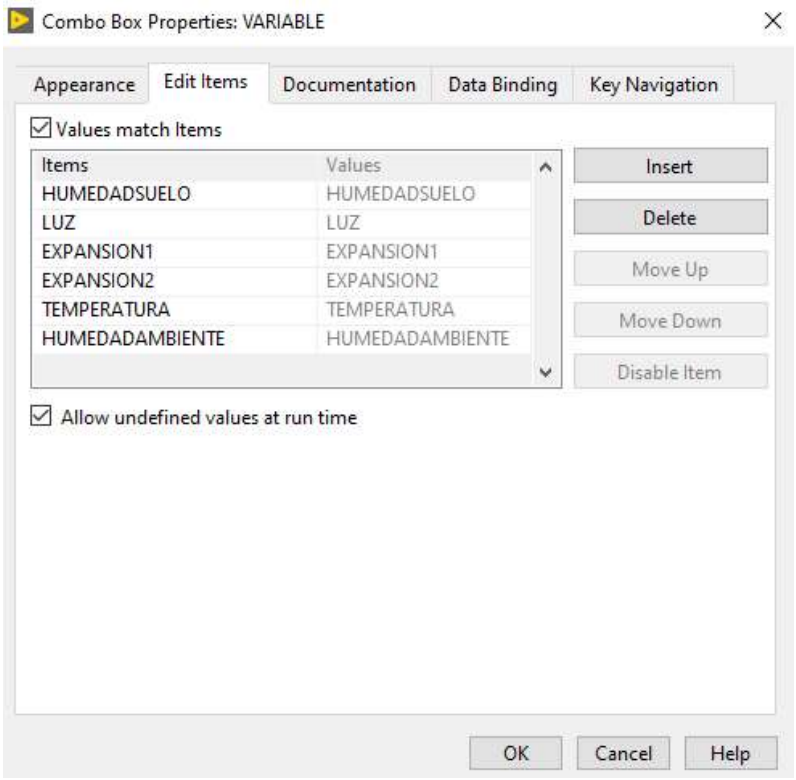


Ilustración 3-50 Añadir nombres de variables al menú.

Los últimos controles que se agregaron al apartado de consultas, fueron dos botones uno para limpiar los datos que se hayan consultado anteriormente y otro más para realizar la consulta (Ilustración 3-51).

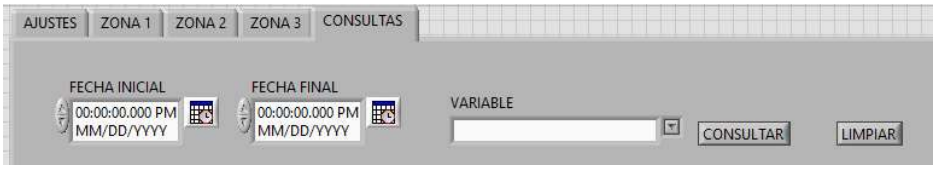


Ilustración 3-51 Controles del apartado "consultas".

Por último, se agregaron tres indicadores gráficos en donde se graficarán los valores obtenidos de la consulta, para agregarlo, en la paleta de controles del panel frontal, hay que seleccionar la opción “graph” y seleccionar “waveform graph”, por último, solo resta colocarla y redimensionarla según sea conveniente (Ilustración 3-52).

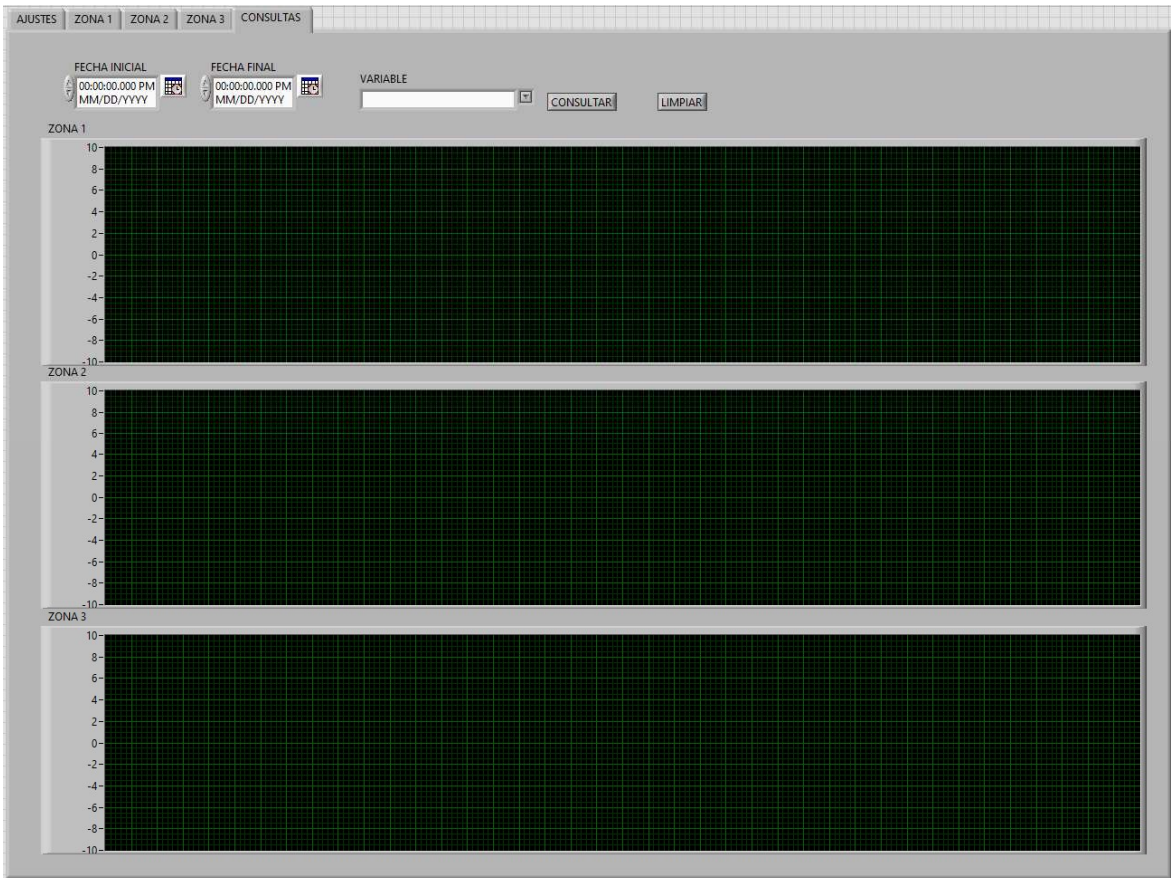


Ilustración 3-52 Apartado "Consultas".

3.5.2 Programación de la interfaz gráfica

Es importante mencionar que cada bloque, indicador o control que se coloque en el panel frontal, crea automáticamente un bloque en el diagrama de bloques, hasta este punto, se han colocado todos los indicadores y controles necesarios para crear el entorno gráfico con el cual el agricultor puede interactuar, por lo que todos estos indicadores y controles se encuentran también en el diagrama de bloques de LabVIEW (Ilustración 3-53), por esta razón es importante ordenar y separar estos bloques a fin de facilitar el desarrollo de la programación.

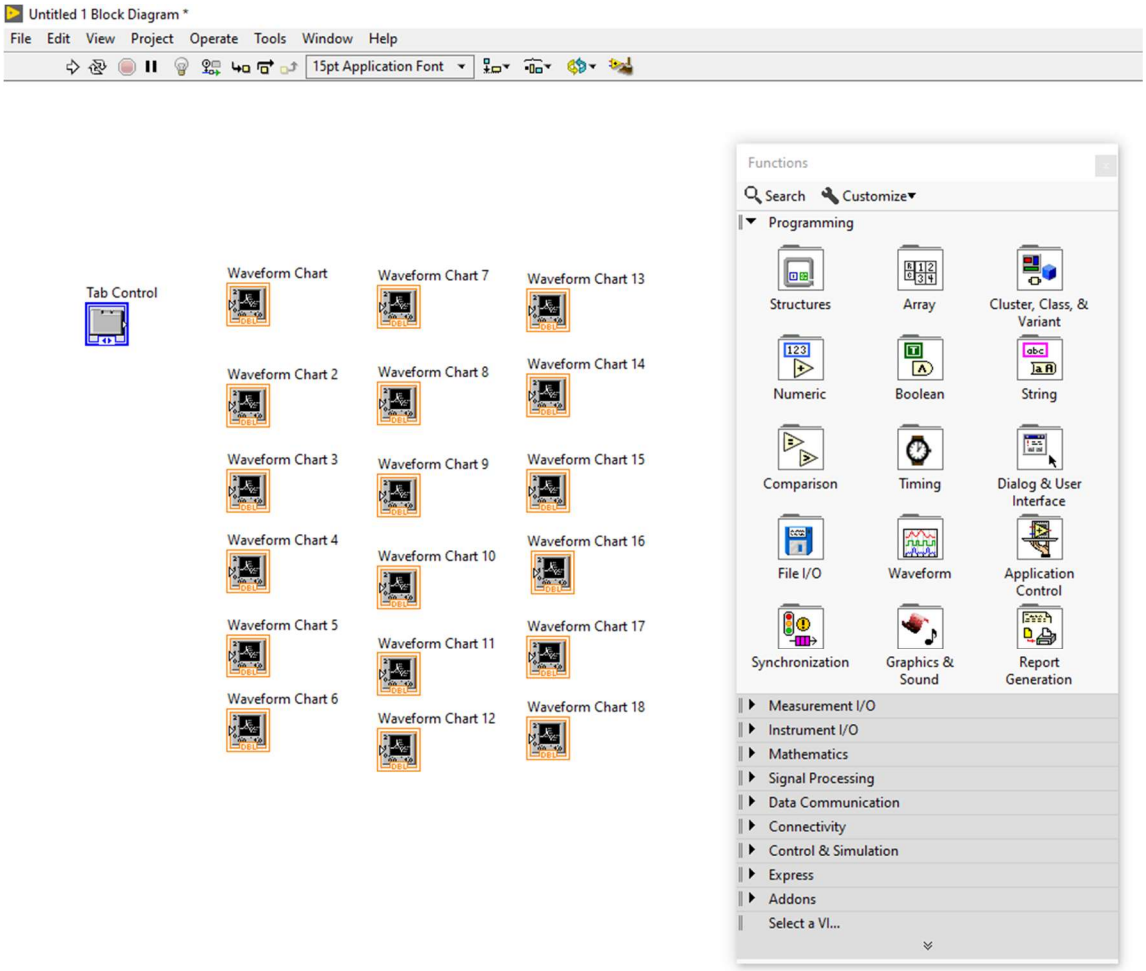


Ilustración 3-53 Bloques del entorno gráfico.

En la ilustración 3-53 se pueden observar algunos bloques pertenecientes a los indicadores y controles que se añadieron en el panel frontal, el problema es que cuando se agrega indicadores o controles en el panel frontal, en la mayoría de las ocasiones se generan en lugares aleatorios del diagrama de bloques, por lo que es recomendable ordenarlos y separarlos por grupos.

Para la programación de la interfaz, lo primero fue crear una estructura que permita repetir todas las funciones constantemente, la cual se identifica como “while loop”, esta estructura se encuentra en la paleta de controles del diagrama de bloques en el menú “structures”, al colocar esta estructura es necesario hacerlo con un espacio lo suficientemente amplio para

poder alojar todos los bloques que componen el sistema, aunque se puede modificar el tamaño de la estructura después de colocarlo si así es necesario (Ilustración 3-54).

Ilustración 3-54 Estructura "while loop" principal.

Como se puede observar en la ilustración 3-53, dentro de la estructura “while loop”, se generaron dos pequeños bloques, Uno con una letra “i” en color azul y otro con un círculo rojo dentro de un cuadrado verde, el bloque con la letra “i” es un contador que sirve para indicar el número de veces que se repite el ciclo while loop, para el diseño de la interfaz gráfica no se utiliza. El otro bloque con un círculo rojo, permite detener el ciclo while loop cuando sea necesario, en este caso, como el ciclo se va a repetir indefinidamente, se tiene que indicar que no debe detenerse en ningún momento el ciclo, para lograr esto, en la única terminal de conexión que tiene este pequeño bloque, se da clic derecho, se selecciona “create” y se selecciona “constant” (Ilustración 3-55), al realizar este paso automáticamente se generara otro pequeño bloque con una letra “F” que significa “false”, de esta forma el ciclo “while loop” se repite constantemente.

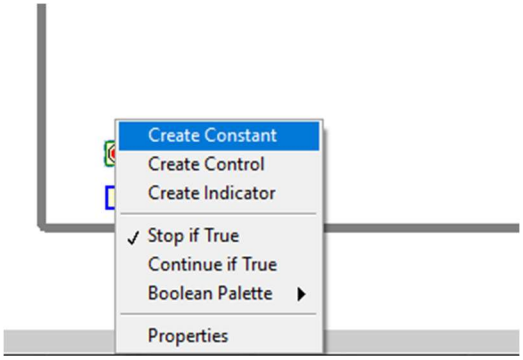


Ilustración 3-55 Configuración para el ciclo while loop.

3.5.2.1 Comunicación con módulo maestro

Una vez que se realizaron las configuraciones anteriores para el ciclo “while loop”, lo siguiente fue colocar todos los bloques que se tengan que ejecutar constantemente dentro del ciclo “while loop” (Ilustración 3-56). Estos bloques se fueron colocando según se fueran utilizando, esto para facilitar la búsqueda de los bloques y mantener todo en orden.

Ilustración 3-56 Indicadores de zonas 1, 2 y 3 en ciclo while loop.

Para poder mostrar las lecturas de los sensores en los indicadores, lo principal fue crear la adquisicion de datos, para esto se utilizó el paquete de expansion NI Visa de LabVIEW, para facilitar el trabajo se quitaron las etiquetas de los nombres de los bloques de los indicadores, esto se logra dando clic derecho sobre el bloque, se selecciona la opcion “visible items” y se desmarca la opcion “label” (Ilustración 3-57).

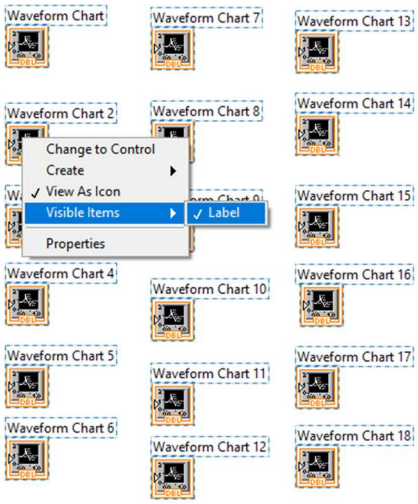


Ilustración 3-57 Quitar etiquetas de los bloques.

Para evitar que los indicadores sigan graficando datos aun cuando no hay datos entrantes del módulo maestro, se utilizó una estructura “case”, a esta estructura se le puede asignar dos o más casos, se encuentra en la paleta de controles del diagrama de bloques en el menú “structures”, se debe colocar esta estructura con el espacio suficiente para alojar en su interior todos los bloques de los indicadores gráficos (Ilustración 3-58).

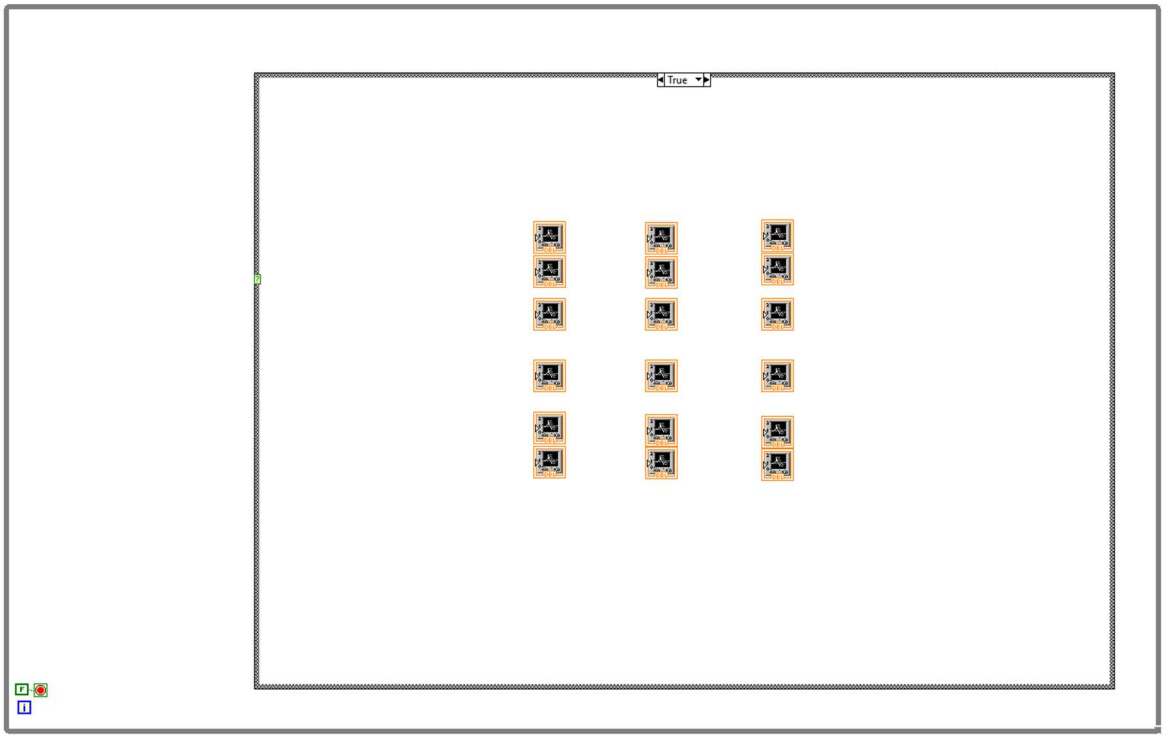


Ilustración 3-58 Case structure para adquisición de datos.

Hay que recordar que, los datos que se envían del módulo maestro a la interfaz gráfica están unidos todos en una cadena de datos en formato de texto, los indicadores gráficos que de las zonas 1, 2 y 3 solo admiten datos en formato numérico, por lo que es necesario separarlos y convertirlos a formato numérico, el bloque que permite realizar esta acción se encuentra en la paleta de bloques del diagrama de bloques, en el menú “string” el bloque tiene el nombre de “scan from string” (Ilustración 3-59).

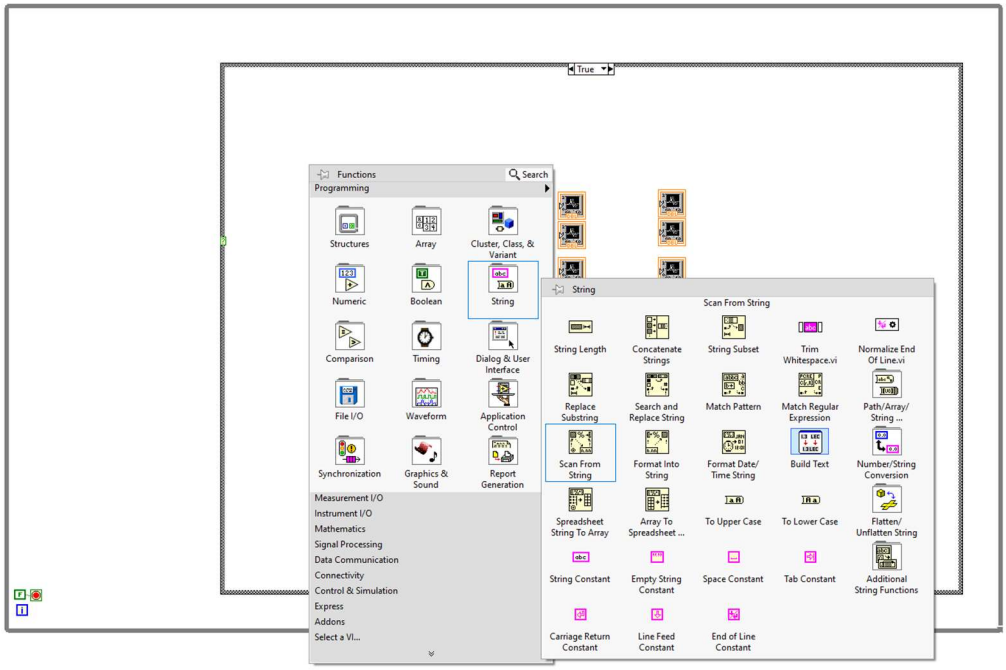


Ilustración 3-59 Separación y conversión de datos provenientes del módulo maestro.

Se colocaron tres de estos bloques, uno para cada zona, en la ilustración 3-60, se muestran las conexiones necesarias entre los bloques “scan from string” y los indicadores gráficos. A los bloques “scan from string” es necesario indicarles, a qué tipo de formato se va a convertir los datos que separen esto se logra agregando en la terminal superior del bloque el texto: “%f”, hay que agregar Uno para cada dato, con este texto se le indica al bloque que, los datos tienen que ser convertidos a formato numérico “float” (Ilustración 3-61).

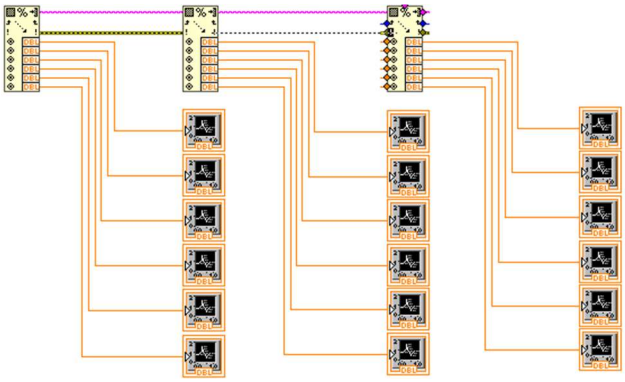


Ilustración 3-60 Asignación de datos a indicadores gráficos.

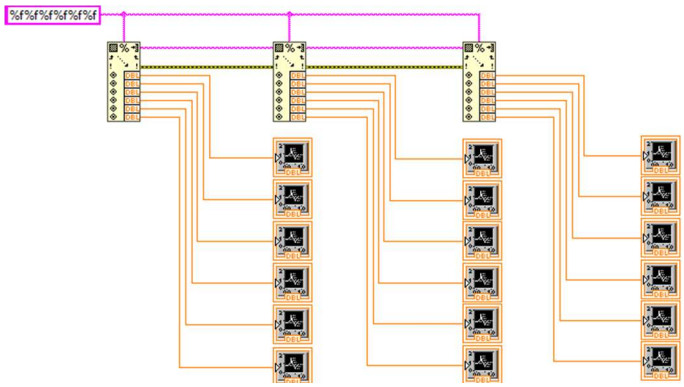


Ilustración 3-61 Asignación de formato numérico a datos extraídos de la cadena.

Hasta este punto, los indicadores ya tienen asignados un valor especifico de la cadena de datos, el problema es que los bloques “scan from string” no tienen una cadena de datos de la cual extraer datos, por lo que el siguiente paso, es leer el puerto seleccionado e indícale que lea la cadena de datos entrante, para realizar esta tarea se utiliza el bloque llamado “VISA read” este bloque pertenece al paquete de expansión NI Visa (Ilustración 3-62).

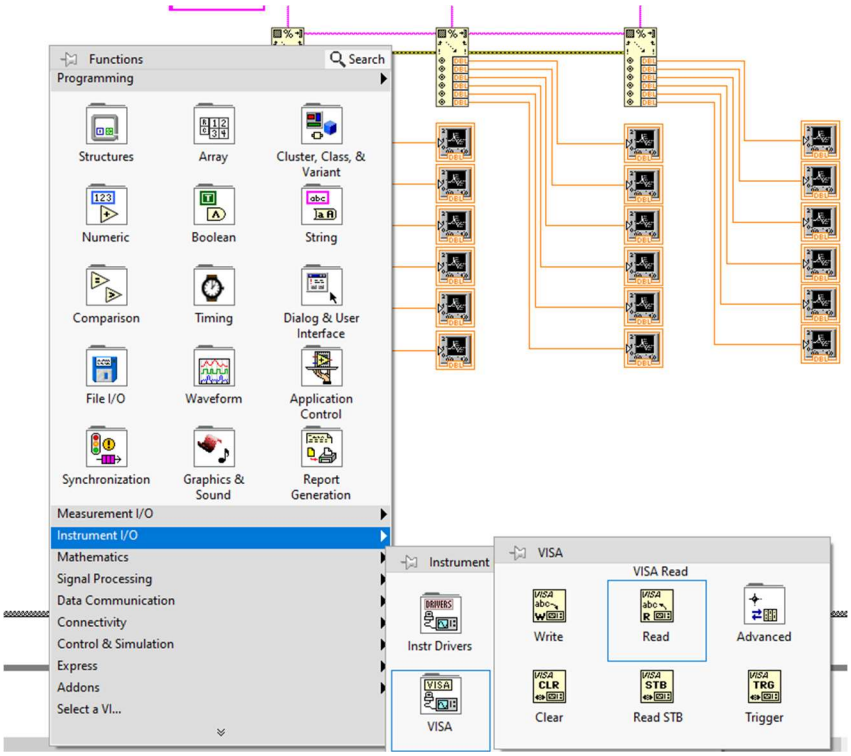


Ilustración 3-62 Visa read.

El bloque “visa read” sirve para leer la cadena de datos del puerto, pero aún falta configurar la comunicación con el puerto, lo primero que se debe hacer es configurar la tasa de baudios a la cual va a trabajar la comunicación con el módulo maestro, esta velocidad se determinó cuando se programó el microcontrolador Atmega 2560, la tasa de baudios que se determinó,

fue de 250000bps, por lo que la velocidad con la que se configuró el puerto de la interfaz gráfica fue la misma tasa de baudios, para establecer esta configuración se retomaron los bloques que se generaron al crear el menú desplegable donde el agricultor puede seleccionar el puerto al que está conectado el hardware, en el bloque "visa configure serial port", en la segunda terminal del lado izquierdo, se debe crear una constante numérica con un valor de 250000 (Ilustración 3-63).

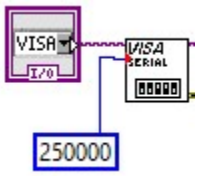


Ilustración 3-63 Tasa de baudios para el puerto serie.

Una vez que se configuró la tasa de baudios a la que va a trabajar la interfaz gráfica, lo siguiente fue agregar un bloque que permitiera activar la estructura "case" en donde están alojados los indicadores de las zonas 1, 2 y 3, hay que recordar que la estructura "case" se agregó para evitar que los indicadores sigan graficando datos incluso cuando no hay datos provenientes del módulo maestro, por lo que se necesitaba agregar un bloque que indicara cada que hay datos entrando por el puerto, dicho bloque se llama "bytes at port", este bloque contiene muchas funciones, pero la que es útil para este sistema es la que indica el número de bytes que hay en el puerto, en caso de que no haya datos provenientes del módulo maestro, los bytes en el puerto serán cero y en caso de que si haya datos provenientes del módulo maestro los bytes en el puerto serán mayor a cero. Este bloque se encuentra en la paleta de controles del diagrama de bloques en el submenú "Instrument I/O" (Ilustración 3-64).

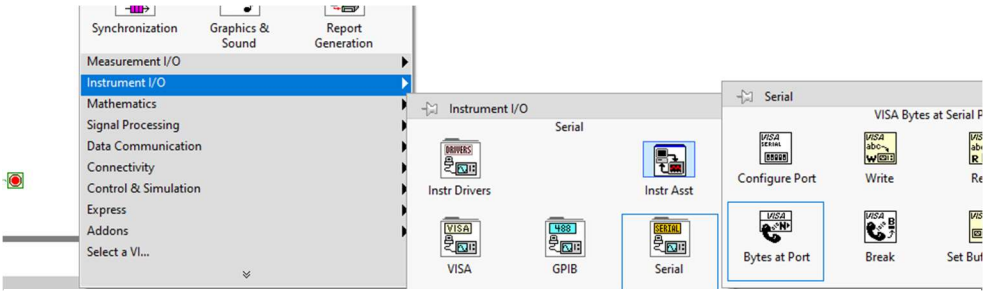


Ilustración 3-64 Bytes at port.

Como ya se mencionó anteriormente, el bloque “bytes at port” solo proporciona el número de bytes que hay en el puerto al momento de la comunicación, pero este dato no sirve para activar y desactivar la estructura “case”, para realizar esta acción se agregó un bloque de comparación, que proporcione como resultado “verdadero” si el número de bytes en el puerto es mayor a cero y “falso” si el número de bytes en el puerto es menor o igual a cero, esta función se puede encontrar en la paleta de controles del diagrama de bloques, en el menú “Comparison” (Ilustración 3-65).

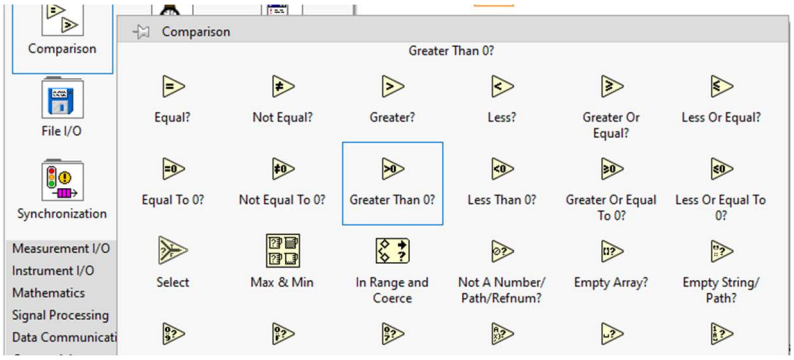


Ilustración 3-65 Greater than 0?.

Ya que se colocaron los bloques anteriores, solo resta hacer las conexiones que se pueden observar en la ilustración 3-66, con esto se concluyó la configuración necesaria para el puerto serie.

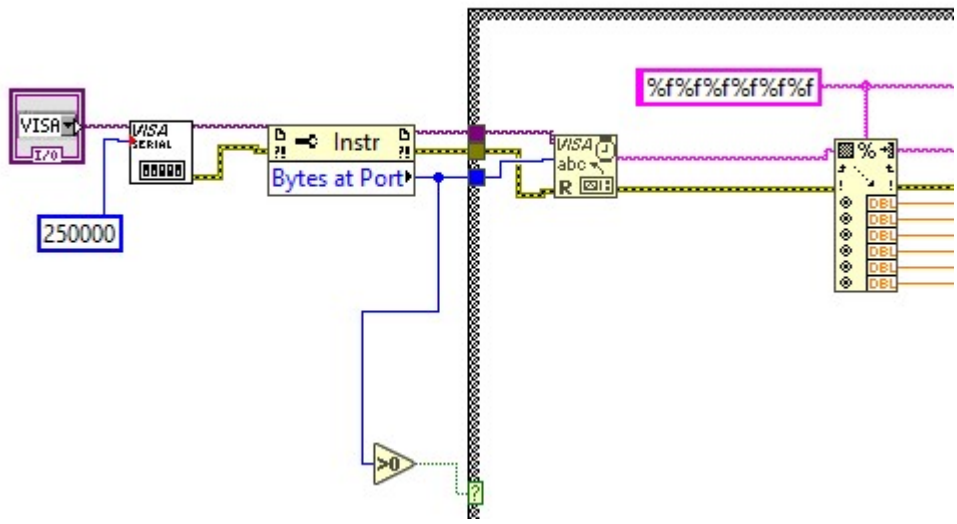


Ilustración 3-66 Conexiones para el puerto serie.

Hasta el momento, la interfaz gráfica solo es capaz de recibir datos, pero también debe ser capaz de enviar datos, específicamente debe de enviar los rangos para el envío de alertas GSM. Esta función de la interfaz tiene dos posibilidades: desactivar las alertas o activar las alertas, primero se inicia con la función de desactivar las alertas, ya que es más sencillo y no necesita de tantos bloques. El primer paso es identificar el bloque del botón que previamente se colocó en el panel frontal y que realiza la función "desactivar alertas", con este botón se puede controlar una estructura "case" con dos casos: Uno cuando el botón se presiona y se envía la orden de desactivar las alertas y otro cuando el botón no es presionado y no se envía ninguna instrucción. Se colocó la estructura "case" que permite desactivar las alertas, una vez colocada la estructura "case" se conecta el botón de "desactivar alertas" al control de la estructura, dentro de esta estructura se colocó un bloque que permita enviar datos por el puerto serie, el bloque que permite realizar esta acción, se llama "VISA Write" y se encuentra en el mismo apartado que el bloque "VISA Read". Ya que este bloque solo admite datos en formato de texto, es necesario escribir todos los datos en una sola cadena, en formato de texto, separando cada dato con una coma, el módulo maestro es la encargada de separar y convertir cada Uno de los datos que se envíen desde la interfaz gráfica, una vez ejecutados los pasos anteriores, solo basta realizar las conexiones que se observan en la ilustración 3-67.

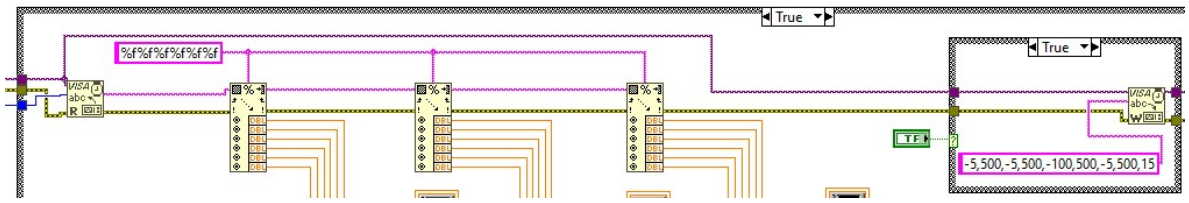


Ilustración 3-67 Función "desactivar alertas GSM".

Como se puede observar en la ilustración 3-67, en la estructura "case" que controla la función de desactivar las alertas GSM, está colocado un bloque "VISA Write" y a este bloque se encuentra conectado el siguiente texto: "-5,500,-5,500,-100,500,-5,500,15", esto se debe a que la forma en la que se desactivan las alertas en este sistema es, forzando un rango de valores los cuales los sensores no son capaces de proporcionar, por ejemplo, el sensor de humedad ambiente solo puede proporcionar valores de 0 a 100, para desactivar las alertas de este sensor, lo que se hace es establecer un rango de -5 a 500, de esta forma el sensor nunca podrá hacer que la lectura se salga de ese rango y de esta forma nunca se enviara la alerta, el

mismo principio se aplica para todos los demás sensores. Los datos en la cadena anteriormente mencionada están ordenados de la siguiente manera: el primer par de datos (-5,500) está destinados al sensor de humedad de suelo, es segundo par de datos (-5,500) está destinado al sensor de luz ambiental, el tercer par de datos (-100,500) está destinado al sensor de temperatura ambiente, el cuarto par de datos (-5,500) está destinado al sensor de humedad ambiental y por último, el dato 15, está destinado al tiempo de espera en el cual el sistema hace la comparación de los rangos y el estado actual de las variables.

Para activar las alertas GSM, el sistema necesita que el agricultor indique los rangos con los cuales se va a comparar el estado actual de las variables, el primer paso, es colocar nuevamente una estructura "case" que va a ser activada con el botón "establecer rango" que previamente se colocó en el panel frontal, una vez colocada la estructura "case", el siguiente paso es colocar dentro de la estructura un bloque "VISA Write", de igual forma se colocan los bloques correspondientes a los controles numéricos que se colocaron previamente en el panel frontal en el apartado "AJUSTES" (Ilustración 3-68), es importante mencionar que los controles numéricos deben esta colocados en el siguiente orden: humedad de suelo mínima, humedad de suelo máxima, luz ambiental mínima, luz ambiental máxima, temperatura ambiente mínima, temperatura ambiente máxima, humedad ambiente mínima, humedad ambiente máxima y tiempo de espera, en caso de no ser colocados en este orden el envío de alertas será erróneo, ya que el microcontrolador de la interfaz gráfica ya tiene programado este orden.

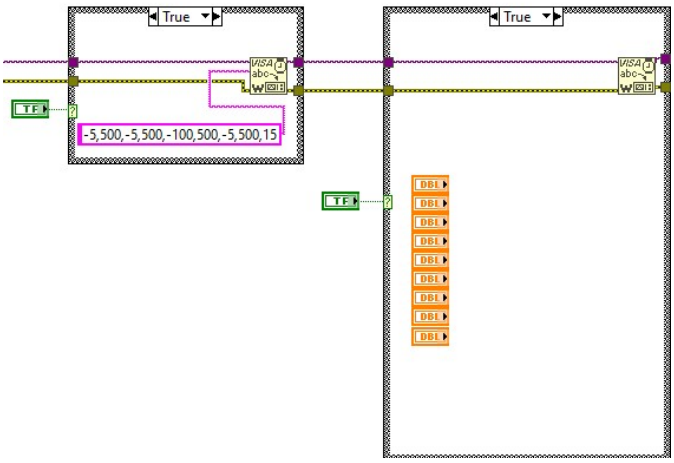


Ilustración 3-68 Estructura "case" para la función "establecer rango".

Debido a que el bloque “VISA Write” solo admite datos en formato de texto, es necesario convertir los datos numéricos a formato de texto, el bloque que permite realizar esta función se llama “Number to decimal string”, se localiza en la paleta de controles del diagrama de bloques, en el menú “string”, se colocó un bloque para cada control numérico, se conecta la única terminal con la que cuentan los controles numéricos a la terminal superior izquierda de los bloques “Number to decimal string” (Ilustración 3-69).

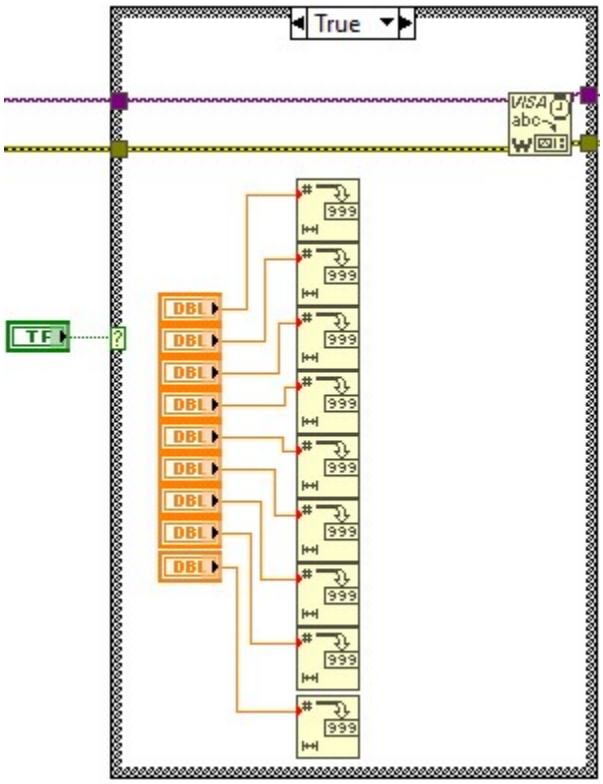


Ilustración 3-69 Conversión de datos numéricos a texto.

En este punto, ya se tienen los datos numéricos provenientes de los controles numéricos convertidos a texto, pero aún hay que unirlos en una sola cadena para que posteriormente el microcontrolador del módulo maestro los pueda recibir para posteriormente separarlos y convertirlos nuevamente a formato numérico. Es importante mencionar que los datos en la cadena deben de estar separados por una coma para que el microcontrolador del módulo maestro sea capaz de identificarlos individualmente. Para realizar este paso se colocó un bloque “Concatenate string” este bloque se encuentra en la paleta de controles del diagrama de bloques, en el menú “string”, este bloque inicialmente solo cuenta con dos terminales de

conexión, aunque se pueden agregar más terminales con el mouse de la computadora. Se deben agregar quince terminales más, además de las dos iniciales (Ilustración 3-70).

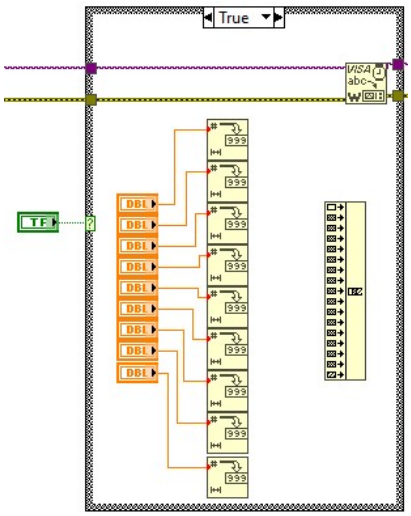


Ilustración 3-70 "Concatenate string" para la función “establecer rango”.

Ya que se tiene colocado el bloque “Concatenate string” con el número de terminales mencionado anteriormente, se hacen las conexiones que se muestran en la ilustración 3-71.

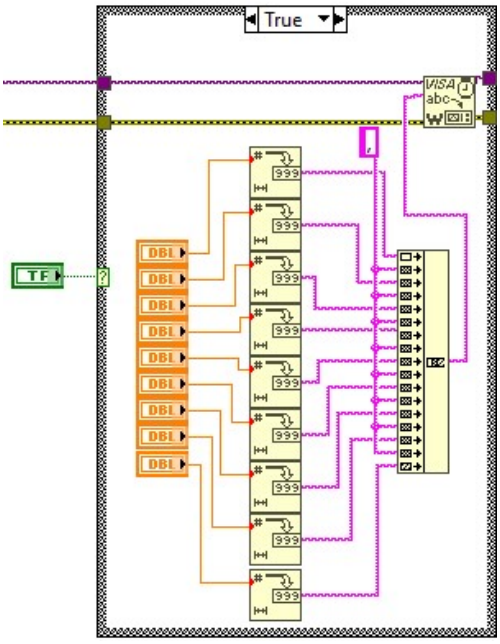


Ilustración 3-71 Función "establecer rango".

Una vez realizados todos pasos anteriores, se agregó un bloque “VISA close” (Ilustración 3-72) y las conexiones para los casos “false” de las estructuras “case” de las funciones “Establecer rango” y “Desactivar alertas”, así como un reloj flip flop que sirve para encender

o apagar el LED que indica si las alertas están activadas o no, si el botón “activar alertas” ha sido presionado, el LED se encenderá, y si el botón “desactivar alertas” se presiona, el LED se apagará. Para realizar este paso, se deben realizar las conexiones que aparecen en la ilustración 3-73.

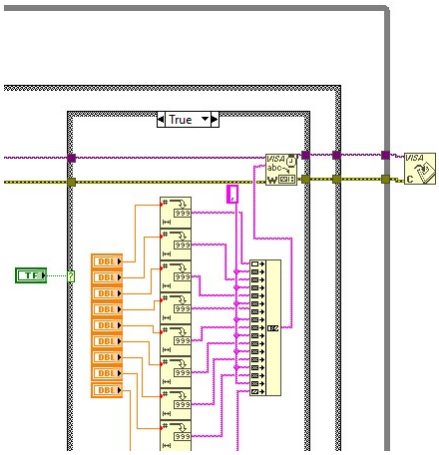


Ilustración 3-72 VISA close.

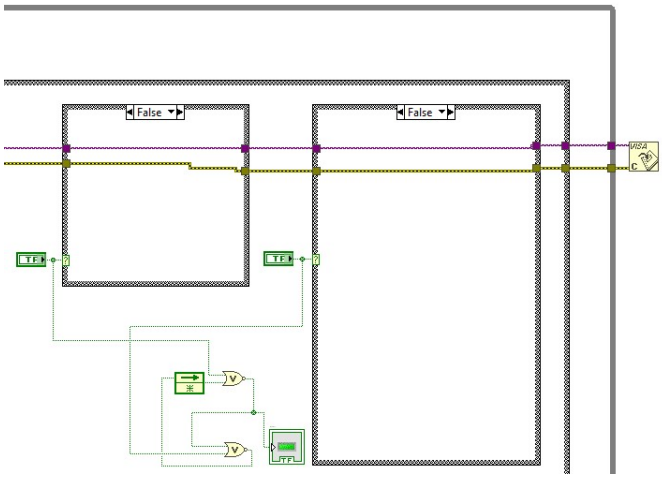


Ilustración 3-73 Conexiones para los casos "false" y el LED indicador de alertas.

También se realizaron las conexiones para la estructura “case” que controla a los indicadores gráficos de las zonas 1, 2 y 3 (Ilustración 3-74).

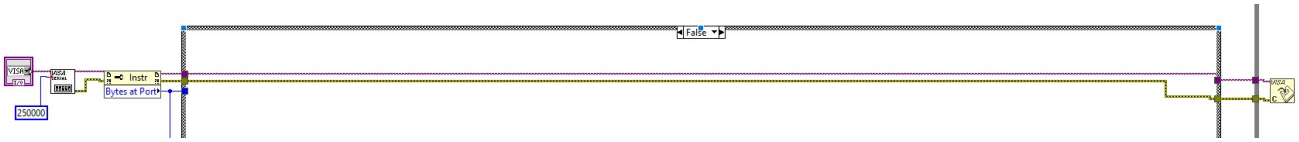


Ilustración 3-74 Conexiones para el caso "false" de la estructura “case” principal.

Con este último paso se finalizó la parte del intercambio de datos con el módulo maestro, aunque hasta este punto el sistema ya era capaz de realizar estas funciones, aun no era posible ejecutarlo ya que todavía era necesario configurar todos los demás bloques que se crearon durante la realización del apartado gráfico, por lo que antes de utilizarlo, se debía terminar todo el diseño de la interfaz gráfica.

3.5.2.2 Diseño de la base de datos

La base de datos que almacenara parte de las lecturas obtenidas por los sensores está diseñada en el software Microsoft Access, la razón por la que se eligió este software, es debido a que está optimizada para la creación de bases de datos, es compatible con SQL, además de que es un entorno más sencillo y común respecto a otros softwares similares.

Para empezar, se creó el diseño de las tablas donde se alojarán los datos, para realizar este paso, lo primero fue crear un documento de Microsoft Access, este se debe alojar estratégicamente para que la ruta del archivo en la computadora sea la misma, tanto si se ocupa en una computadora como si se ocupa en otra diferente, para esto el archivo se debe alojar en una ruta que se sea igual en todas las computadoras que utilicen el sistema operativo Windows, esta ruta se encuentra dentro del disco local C, la cual se crea automáticamente durante la instalación del sistema operativo y es igual en todas las computadoras que utilicen este sistema operativo, sabiendo lo anterior mencionado, se decidió alojar los archivos en esta ruta en la que se va a quedar permanentemente, primero hay que localizar el “disco local C” o “Local disk C”, este se encuentra en la pestaña “Este equipo” o “This PC” (Ilustración 3-74).

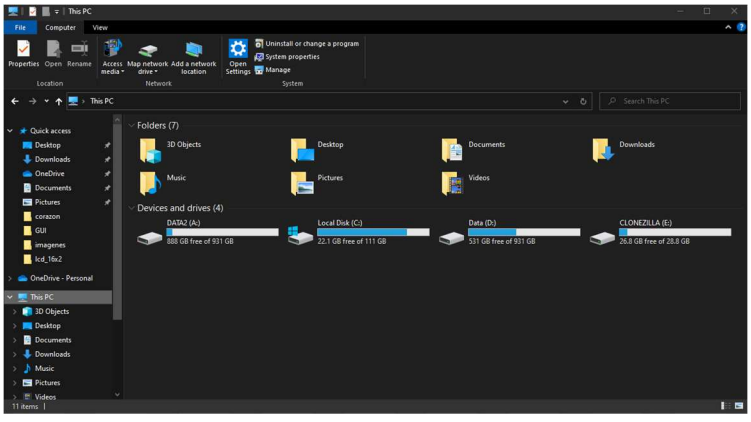


Ilustración 3-75 Este equipo.

En la ventana “Este quipo” se pueden encontrar todos los dispositivos de almacenamiento masivo que se encuentren conectados a la computadora, el número de discos que aparecen en esta ventana es variable para todas las computadoras ya que todas tienen diferentes dispositivos conectados, pero en todas debe de aparecer el disco local C, que es en donde se almacena el sistema operativo, así como los programas instalados, dentro del disco local C se encuentran varias carpetas (Ilustración 3-76) que contienen archivos críticos tanto del sistema operativo como de los programas instalados, por lo que no se deben modificar ni alterar los archivos de ninguna de estas carpetas, en lugar de agregar archivos a las carpetas ya existentes, se creó una carpeta nueva en la que se alojaran los archivos de la base de datos, como se puede observar en la ilustración 3-77, se le asignó el nombre de “Data”, el nombre que se le asigne a la carpeta no es relevante, aunque no debe de contener ningún carácter especial o espacios, esto para evitar que se produzcan errores con la comunicación entre LabVIEW y el archivo Access.

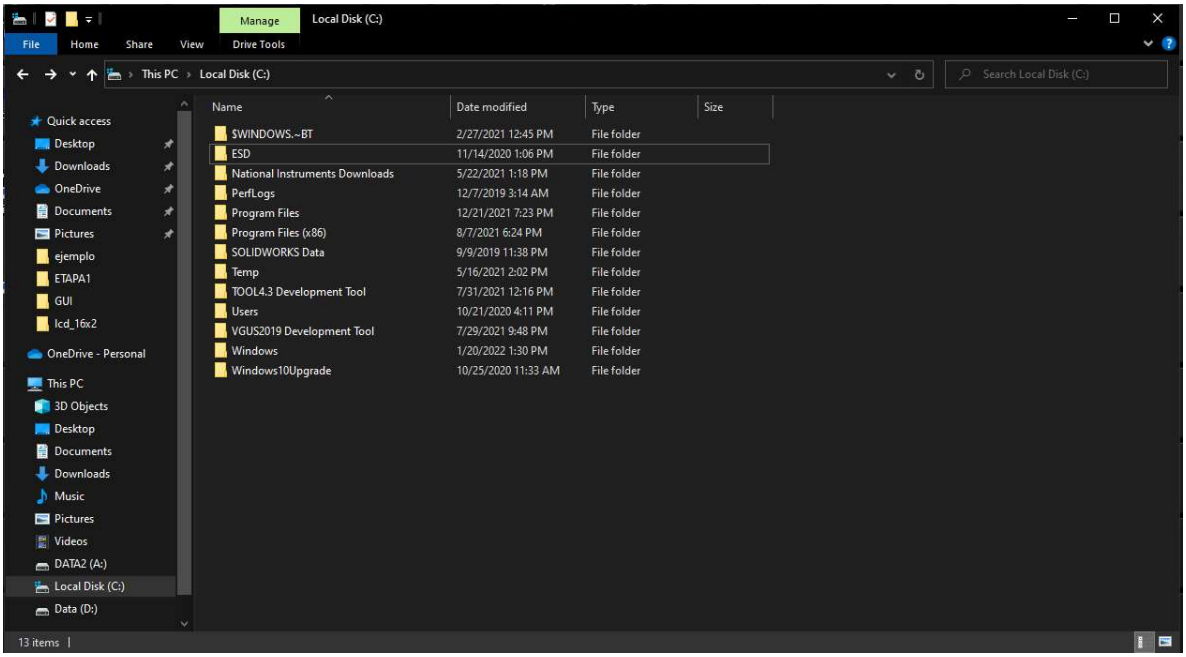


Ilustración 3-76 Disco local C.

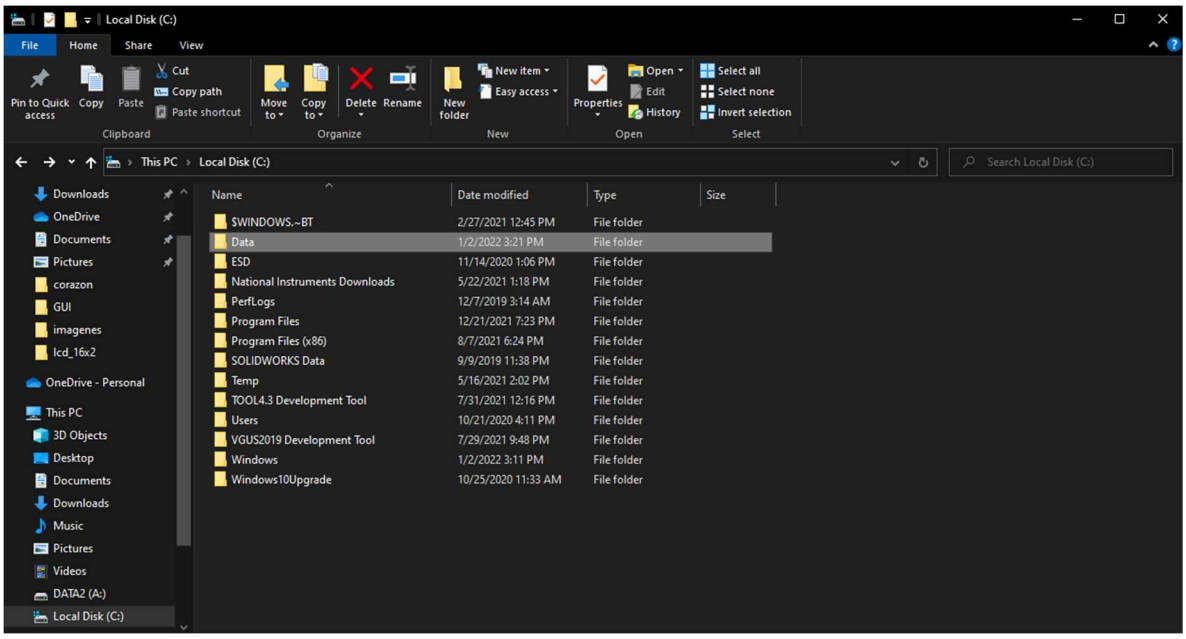


Ilustración 3-77 carpeta "Data" para alojar base de datos.

Dentro de la carpeta “Data” que se creó en el paso anterior, es donde se decidió que se alojen todos los archivos para la base de datos. Primero se ejecuta la aplicación Access, en la cual aparecerá una ventana con varias opciones (Ilustración 3-78) de las cueles, se selecciona la opción “Base de datos del escritorio en blanco”.

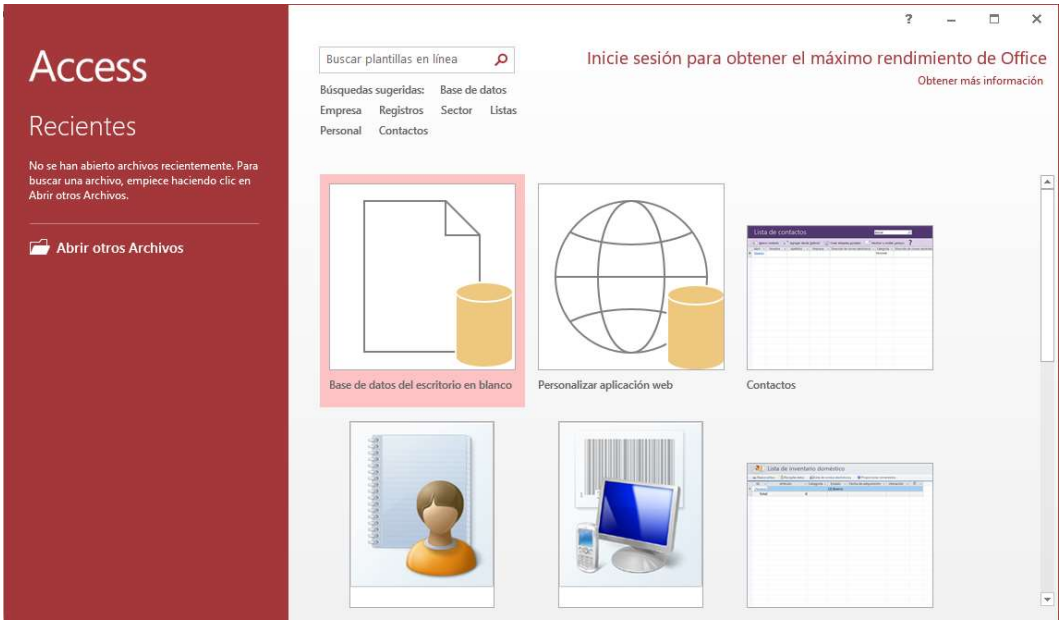


Ilustración 3-78 Microsoft Access.

Al seleccionar la opción antes mencionada, se abrirá una nueva ventana, que permite modificar el nombre y seleccionar la ruta en la cual se guardará el archivo (Ilustración 3-79).

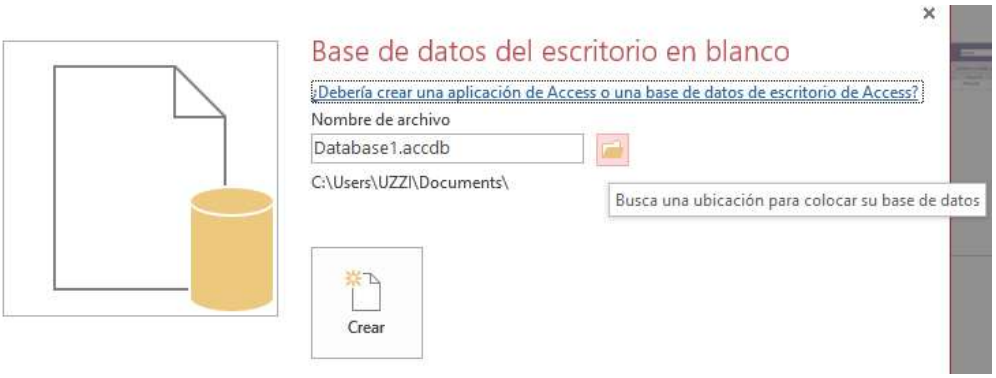


Ilustración 3-79 Crear archivo Access.

Al presionar el icono de la carpeta que se encuentra al lado de la casilla donde se puede modificar el nombre, se abrirá nuevamente una ventana diferente, en la que se debe de buscar la carpeta “Data” (Ilustración 3-80).

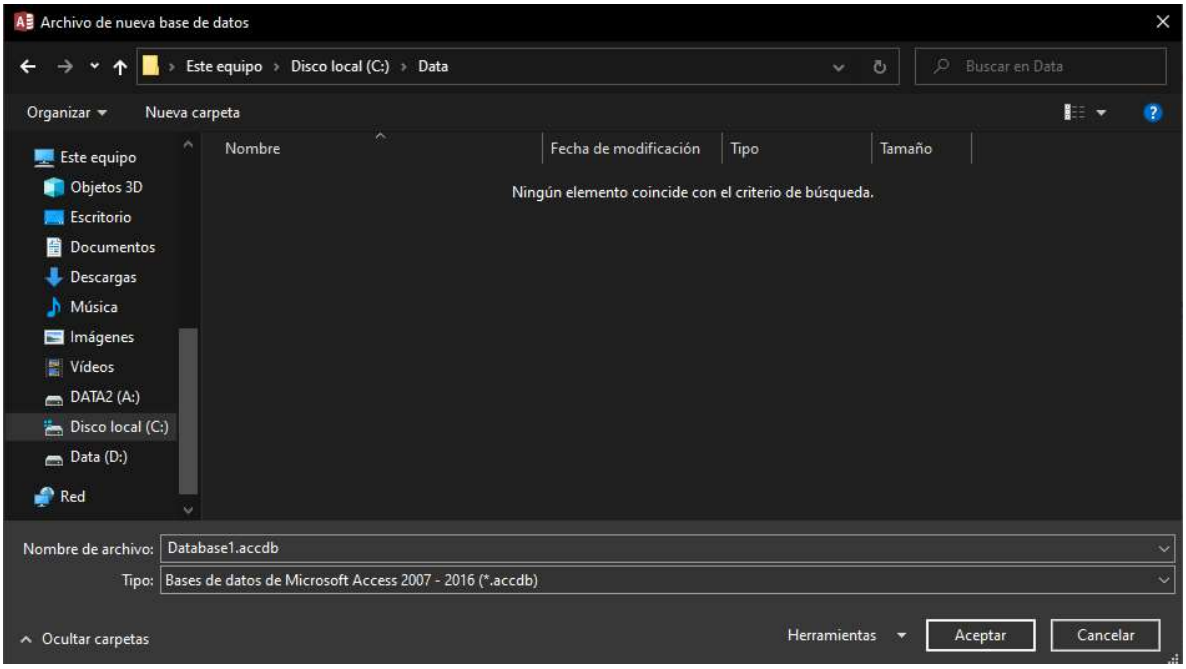


Ilustración 3-80 C:\Data.

En esta ventana se puede cambiar el nombre del archivo, el cual no es relevante siempre y cuando no contenga caracteres especiales ni espacios, en este caso, el nombre asignado fue: “Database”. En la opción “Tipo” se encuentra un menú desplegable (Ilustración 3-81) que permite modificar la versión de Access que tendrá el nuevo archivo, es importante seleccionar la versión “Bases de datos de Microsoft Access (formato 2002-2003) (*.mdb)”, esta modificación es necesaria para evitar problemas de compatibilidad.

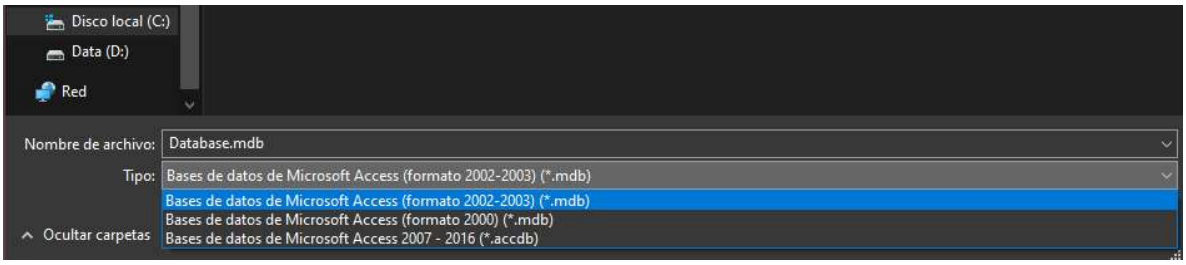


Ilustración 3-81 Cambio de formato 2002-2003 (.mdb)

Una vez que se haya modificado el nombre y el formato anteriormente mencionado, se guardan los cambios realizados dando clic en el botón "Aceptar", al realizar esta acción se regresara a la ventana que aparece en la ilustración 3-78, habiendo regresado a esta ventana, se procede a crear el archivo Access dando clic en el botón "Crear".

El siguiente realizado fue, crear las tablas que contendrán los datos obtenidos por los sensores, para realizar este paso, se debe abrir el archivo que se acaba de crear, habiendo abierto el archivo, aparecerá automáticamente una tabla en blanco (Ilustración 3-82).

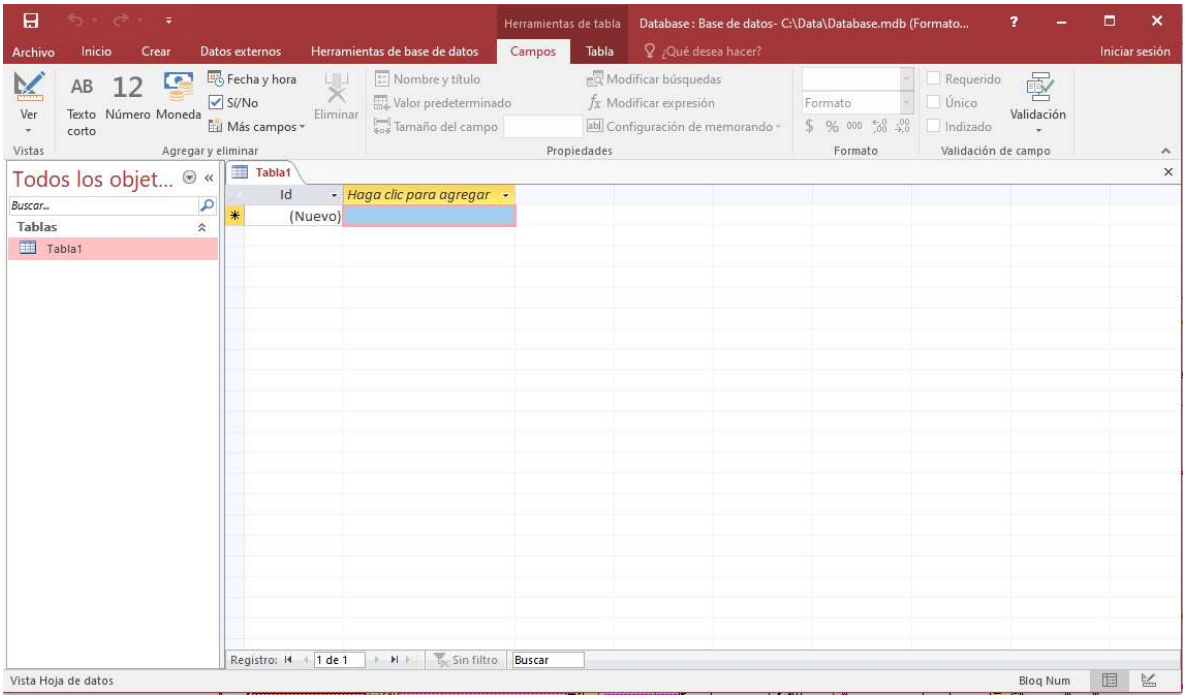


Ilustración 3-82 Tabla en blanco.

Cada columna tiene un formato de datos que almacena, según sea el formato es el tipo de datos que puede almacenar, a la primera columna, se le cambió el nombre que tiene por defecto de "id" a "FECHA" y se le asignó el formato "Fecha/Hora", para asignar el formato de una comuna, se abre la pestaña "Campos" que se encuentra en la barra de herramientas de

la parte superior y dentro de la pestaña se selecciona el menú desplegable con el texto "Autonumeracion" y se cambia por "Fecha/Hora" (Ilustración 3-83).

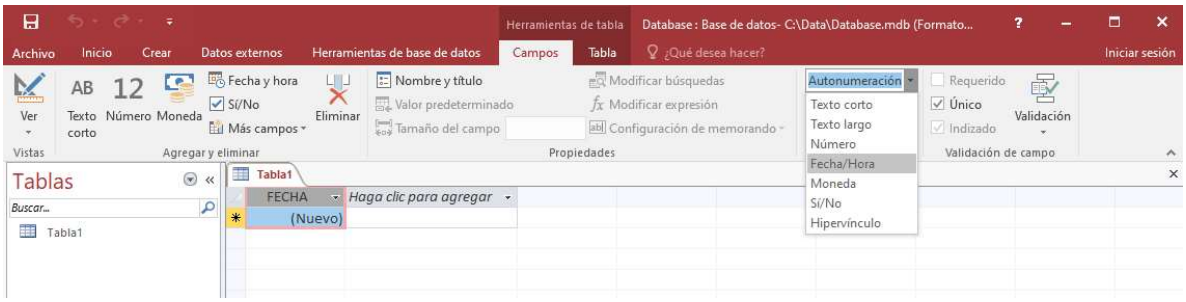


Ilustración 3-83 Cambiar formato de datos a columna.

Para agregar una columna nueva, se da clic sobre el texto "Haga clic para agregar", sabiendo esto se agregaron las siguientes columnas con los formatos especificados: "HUMEDADSUELO", "LUZ", "EXPANSION1", "EXPANSION2", "TEMPERATURA" y, por último, la columna "HUMEDADAMBIENTE", a todas estas columnas se les asignó el formato "Número". Para guardar los cambios realizados en el archivo se utiliza la combinación de teclas "CTRL + G", al realizar esta acción, se abrirá una venta en donde se puede modificar el nombre con el que se guardara la tabla, el nombre que se le asigno a la tabla es "MÓDULO1" (Ilustración 3-84).

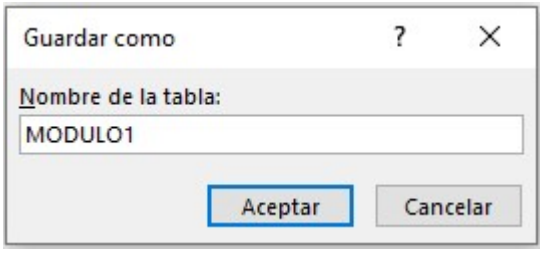


Ilustración 3-84 Renombrar tabla.

La tabla "MÓDULO1", es la encargada de almacenar los datos obtenidos por los sensores que se encuentran en la zona 1, por lo que también fue necesario crear las tablas MÓDULO2 y MÓDULO3 que almacenaran los datos obtenidos por los sensores que se encuentran en las zonas 2 y 3 respectivamente. Para agregar una nueva tabla, se selecciona la pestaña "Crear" que se encuentra en la barra de herramientas de la parte superior, dentro de la pestaña, se selecciona la opción "tabla" (Ilustración 3-85) y automáticamente se agregara una nueva tabla. Una vez creadas las tablas MÓDULO2 y MÓDULO3, es necesario agregar las mismas columnas con los mismos formatos que para la tabla MÓDULO1.

Ilustración 3-85 Crear nueva tabla.

Hasta este punto, las tablas ya estaban listas para almacenar los datos en sus respectivas celdas (Ilustración 3-86), aunque aún era necesario crear la conexión entre LabVIEW y el archivo Access.

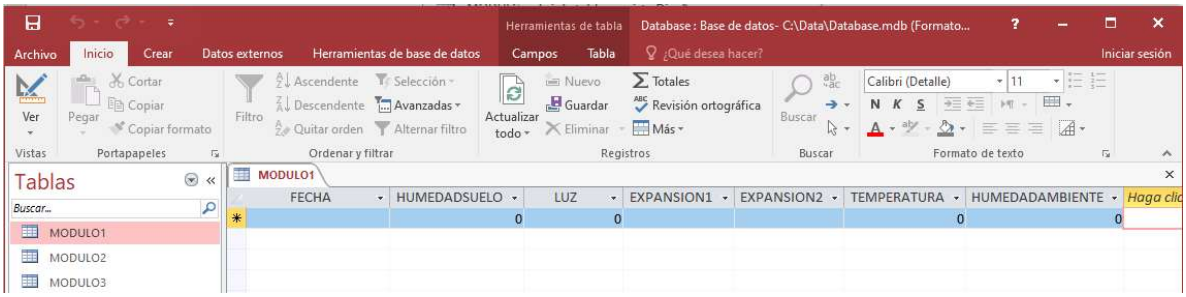


Ilustración 3-86 Tablas para zonas 1, 2 y 3.

Para crear esta conexión, lo primero es abrir el menú “Tools” que se encuentra en la barra de herramientas del diagrama de bloques de LabVIEW y se selecciona la opción “Create Data Link…” (Ilustración 3-87).

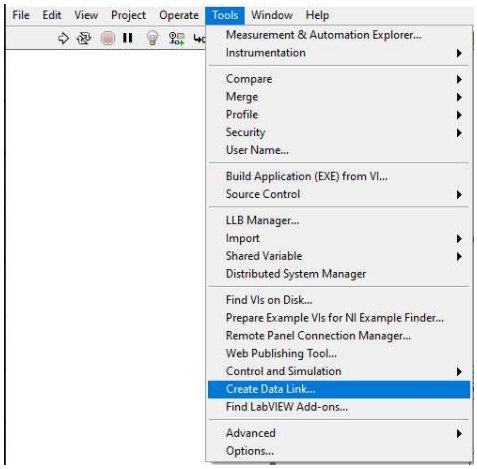


Ilustración 3-87 "Create Data Link".

Al seleccionar esta opción, se abrirá una ventana emergente (Ilustración 3-88), en esta ventana se debe seleccionar la opción “Microsoft Jet 4.0 OLED DB Provider”.

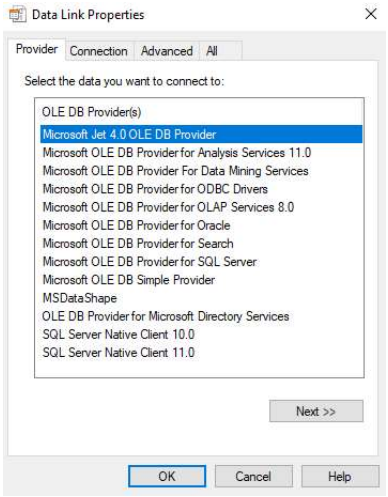


Ilustración 3-88 OLE DB Provider(s).

Una vez seleccionada la opción mencionada, se da clic en el botón “Next”, se abrirá la pestaña “Connection” en donde se debe seleccionar el archivo Access que se creó anteriormente (Ilustración 3-89), al dar clic en el icono “…”, se abrirá una nueva ventana emergente donde se puede buscar el archivo Access, es importante recordar que el archivo se encuentra en la carpeta “Data” dentro del disco local C (Ilustración 3-90).

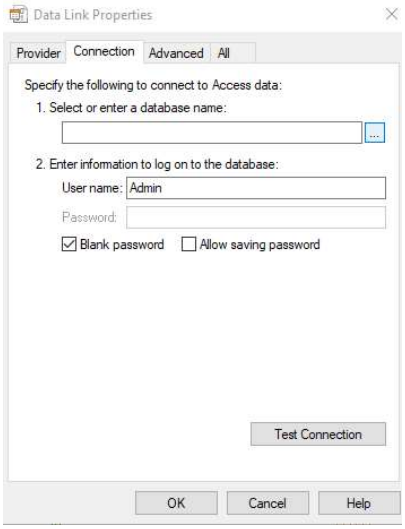


Ilustración 3-89 Conexión con el archivo Access.

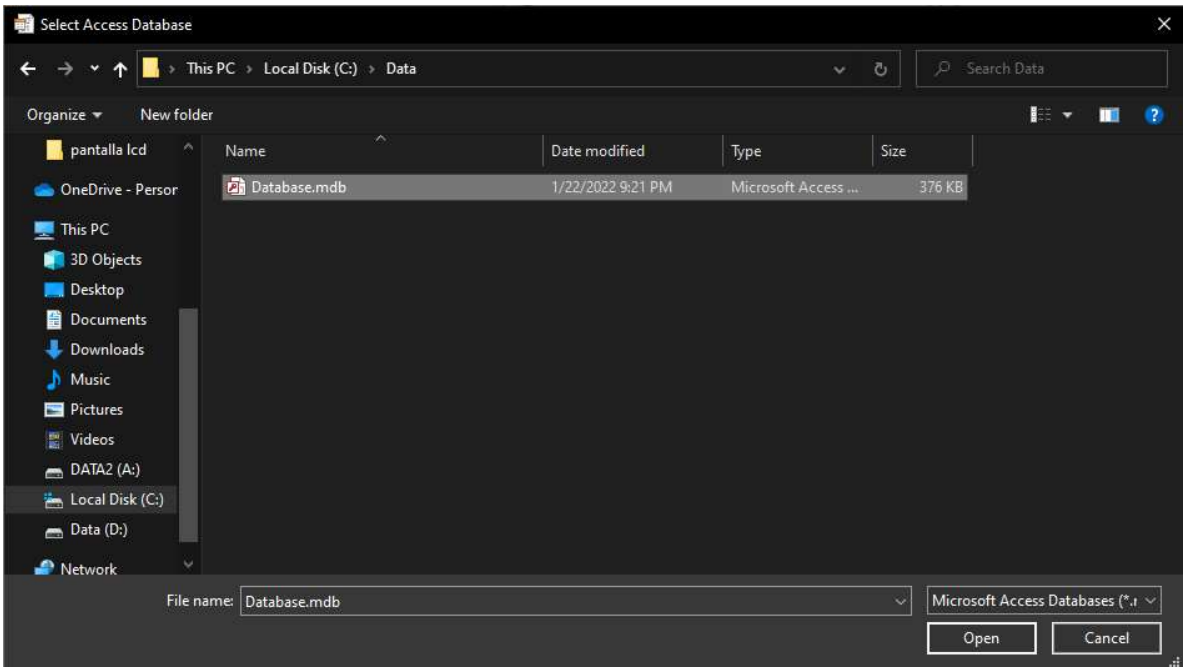


Ilustración 3-90 Dastabase.mdb.

Habiendo localizado el archivo “Database.mdb” que se creó en los pasos anteriores, se da clic en el botón “Open”, al realizar esta acción se regresara a la ventana que aparece en la ilustración 3-89, con la ruta del archivo en la casilla correspondiente. El siguiente paso realizado fue corroborar si hay conexión entre LabVIEW y el archivo Access, para realizar este paso, se da clic en el botón “Test Connection”, si no existen problemas en la conexión deberá aparecer el mensaje de la ilustración 3-91, de lo contrario es necesario revisar si re realizaron correctamente todos los pasos anteriores.

Ilustración 3-91 Prueba exitosa de conexión entre LabVIEW y Access.

Por último, se da clic en el botón de la ventana emergente de la ilustración 3-91 y también se da clic en el botón de la ventana de la ilustración 3-89, al realizar este último paso, se abrirá nuevamente una ventana emergente que permite asignar nombre y seleccionar la carpeta donde se guardará el archivo que realiza la conexión entre LabVIEW y Access (Ilustración

3-92), este archivo también se debe de guardar en la carpeta “Data” que se encuentra alojada en el disco local C, por temas de practicidad al archivo también se le asignó el nombre de “Data”, aunque se le puede asignar cualquier otro nombre siempre y cuando no contenga caracteres especiales ni espacios.

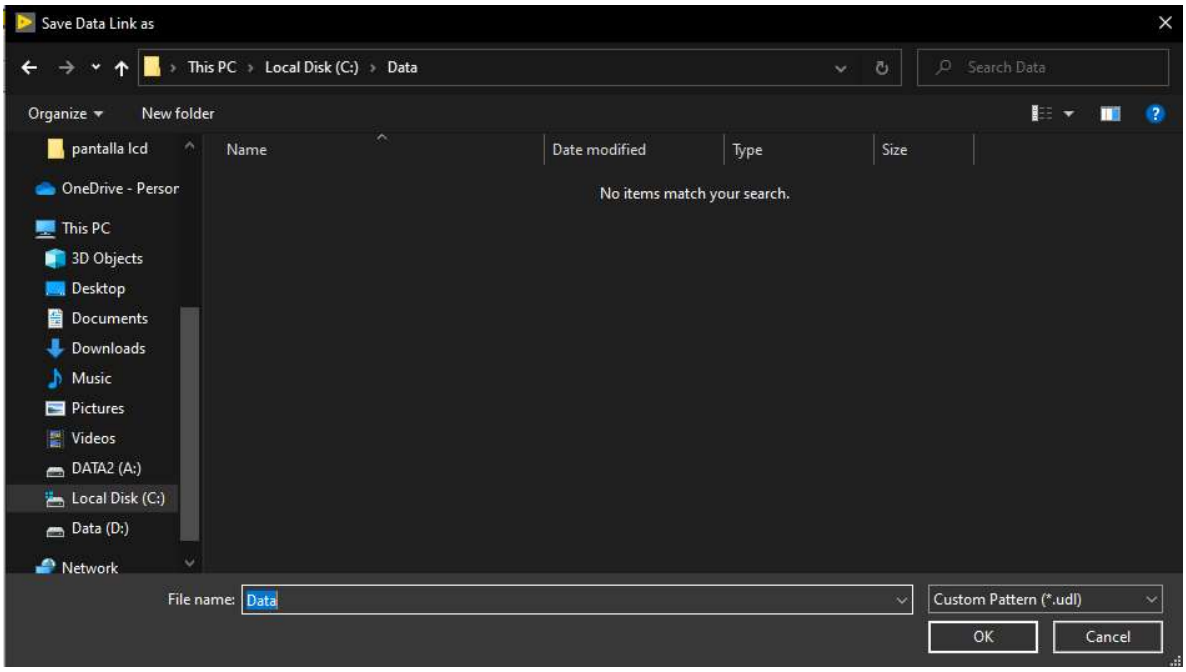


Ilustración 3-92 Data.udl.

Solo resta dar clic en el botón “OK” para finalizar este proceso, tras esto se debe abrir una ventana emergente que indica que el archivo se creó exitosamente, además, que indica la ruta en la que se encuentra almacenado el archivo (Ilustración 3-93), en este caso la ruta es la siguiente: “C:\Data\Data.udl”. esta ruta es muy importante ya se utilizó para durante la realización de la interfaz gráfica.

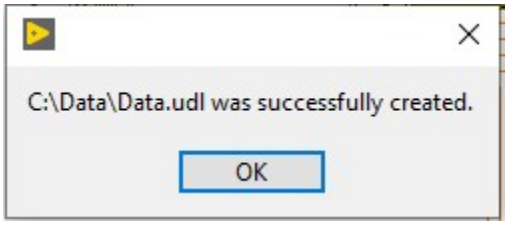


Ilustración 3-93 ruta del archivo udl.

Si se revisa la carpeta “Data” que se creó dentro del disco local C, se puede observar que se creó otro archivo además del archivo con extensión “.udl”, este archivo es el que permite

realizar la conexión entre LabVIEW y Access, mismo que no se debe renombrar, ni cambiarlo de carpeta o alterarlo de alguna forma

3.5.2.3 Almacenamiento de datos en base de datos

Realizados todos los pasos anteriores, se creó la configuración de bloques para registrar lecturas en la base de datos. Para empezar, se coloca el bloque "DB Tools Open Connection", este bloque se encuentra en la paleta de controles del panel frontal en el submenú "Connectivity" (Ilustración 3-94).

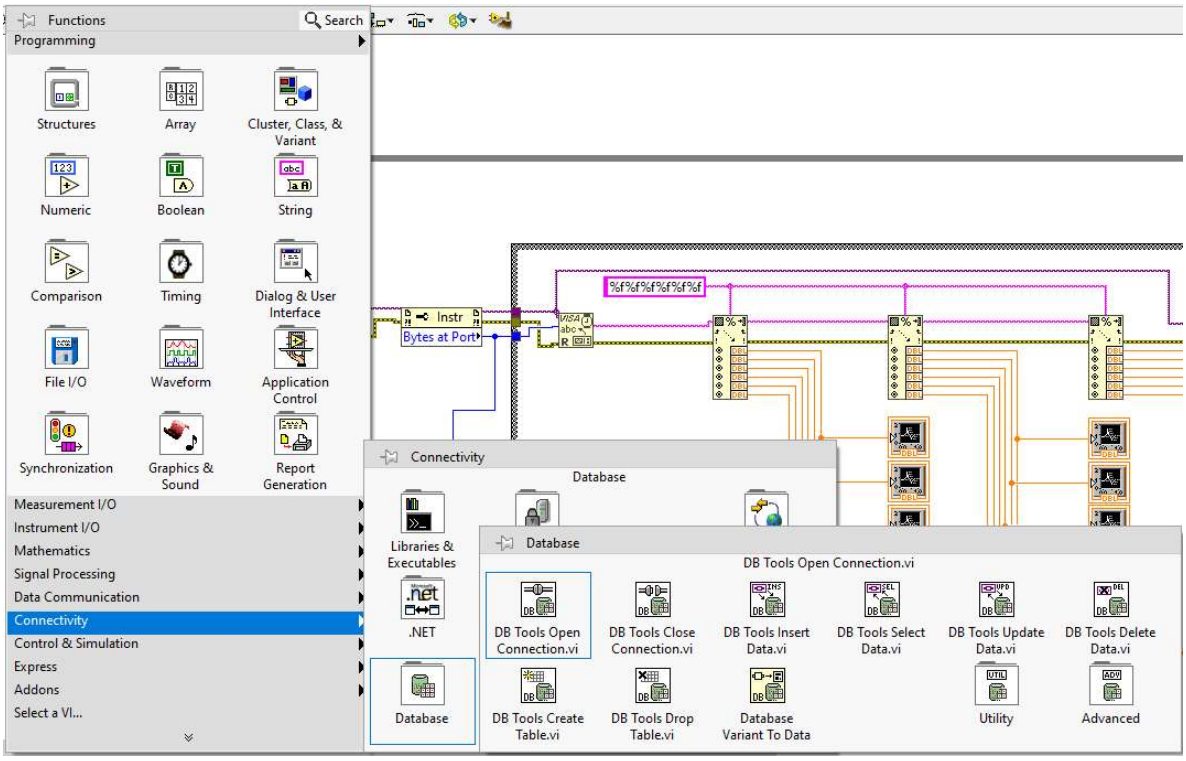


Ilustración 3-94 Submenú Connectivity.

Además del bloque "DB Tools Open Connection", se necita del bloque "String to Path", este bloque se encuentra en la paleta de controles del diagrama de bloques, en el apartado "string" (Ilustración 3-95).

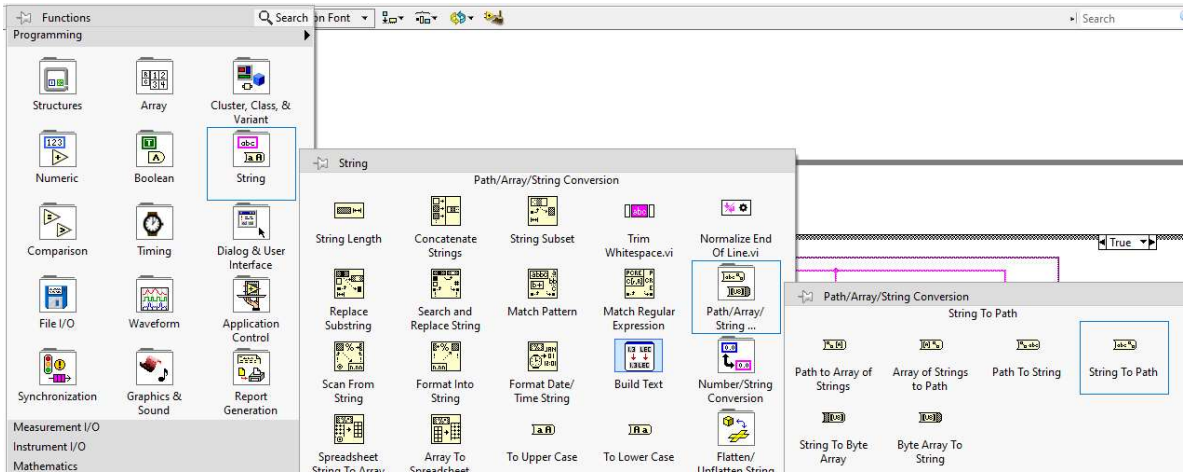


Ilustración 3-95 String to Path.

Habiendo colocado los bloques anteriormente mencionados, el siguiente paso era indicarle al bloque “DB Tools Open Connection” la ruta en la que se encuentra almacenado el archivo que establece la conexión entre LabVIEW y el archivo Access, hay que recordar que la ruta en la que se encuentra almacenado el archivo es la siguiente: “C:\Data\Data.udl”. Para realizar este paso, se realizaron las conexiones que se muestran en la ilustración 3-96.

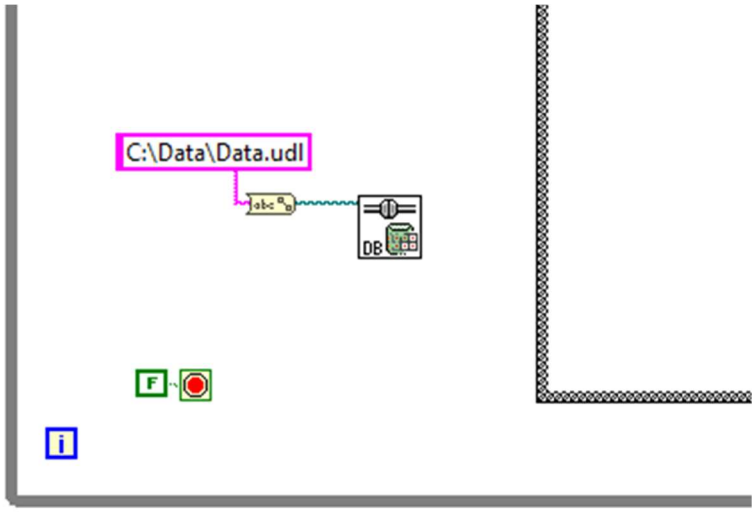


Ilustración 3-96 Especificar la ruta de la base de datos.

Para enviar las lecturas a la base de datos, lo primero es unir los datos en una cadena, pero cada dato debe de conservar el formato que ya tiene, el bloque que permite realizar esta acción se llama “Bundle”, este bloque se encuentra en la paleta de controles del diagrama de bloques, en el apartado “Cluster, Class & Variant” (Ilustración 3-97).

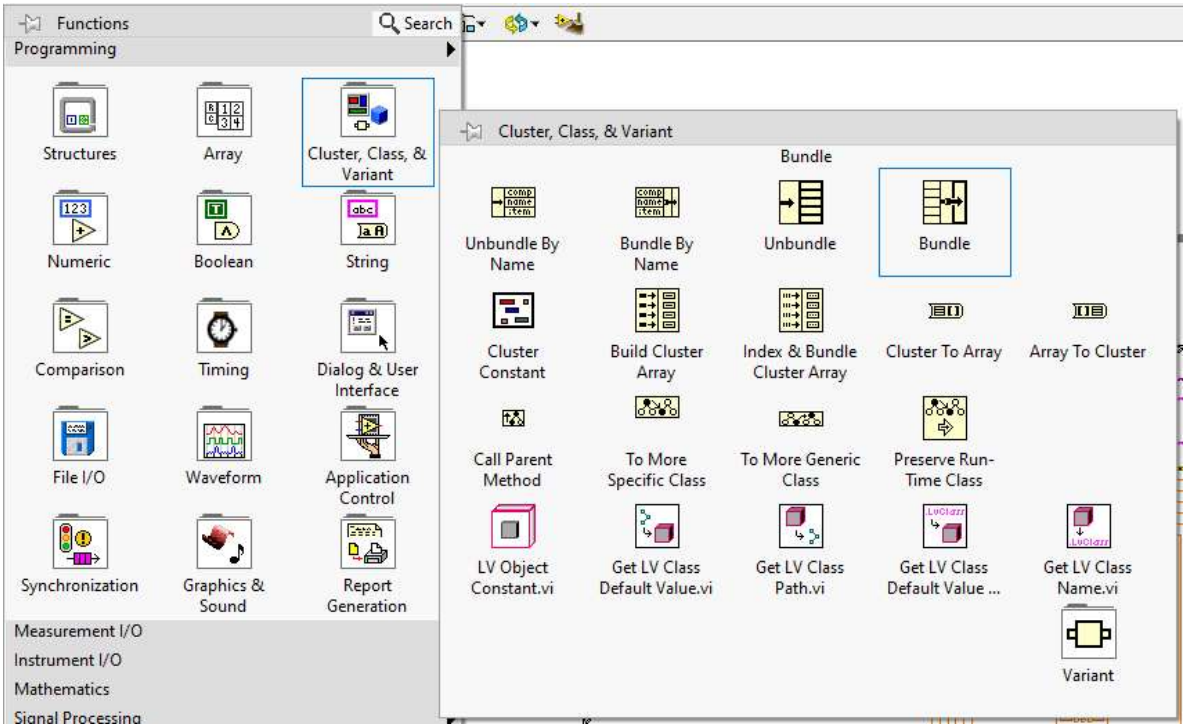


Ilustración 3-97 Cluster, Class & Variant.

se agregaron tres bloques “Bundle”, uno para cada grupo de datos correspondiente a cada una de las zonas del invernadero (Ilustración 3-98).

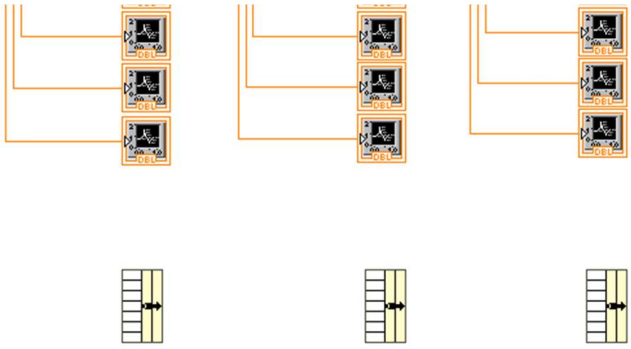


Ilustración 3-98 Bundle.

En la primera casilla de los bloques “Bundle” debe conectar la hora y la fecha en tiempo real, para obtener estos datos se debe colocar un bloque llamado “Get Date/Time in seconds”, mismo que se encuentra en la paleta de controles del diagrama de bloques en el apartado “Timing” (Ilustración 3-99).

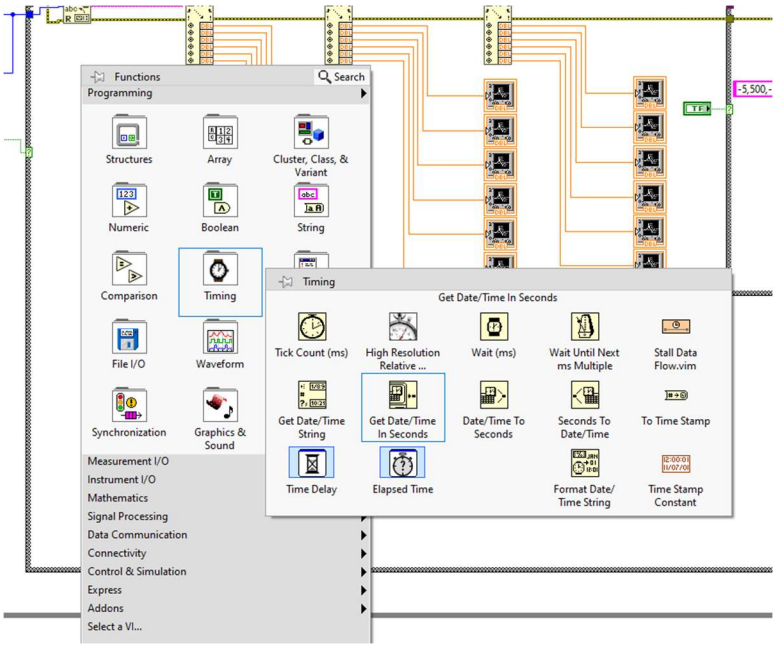


Ilustración 3-99 Submenu “Timing”.

Una vez colocado este bloque, se conecta la única terminal que tiene el bloque, a la primera casilla del bloque “Bundle”, además de los datos de los indicadores gráficos (Ilustración 3-100).

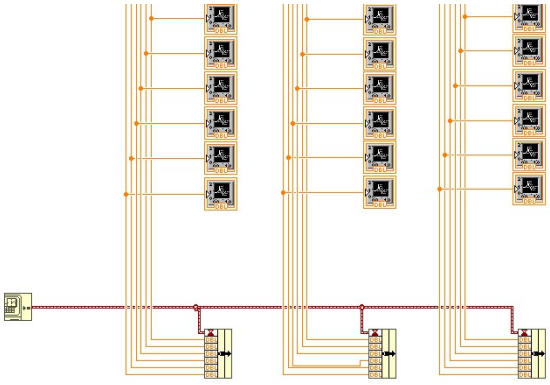


Ilustración 3-100 Conexión para los bloques "Bundle".

Para enviar la cadena de datos que se ha creado en el paso anterior a la base de datos, se colocaron tres bloques “DB Tools insert data”, uno para cada cadena de datos correspondiente a cada zona del invernadero, este bloque se encuentra en el submenú “Connectivity” que se muestra en la ilustración 3-93 y, se realizan las conexiones que se observan en la ilustración 3-101.

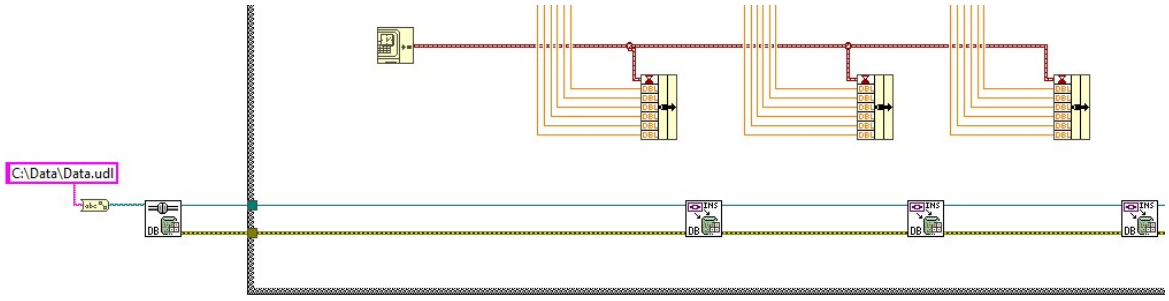


Ilustración 3-101 Bloques "Insert" para la base de datos.

A los bloques “DB Tools insert data” se les conecta la salida de los bloques “Bundle” que contiene la cadena de datos de las tres zonas del invernadero y, se les debe indicar cuál es el nombre de la tabla en la que se almacenaran los datos (Ilustración 3-102).

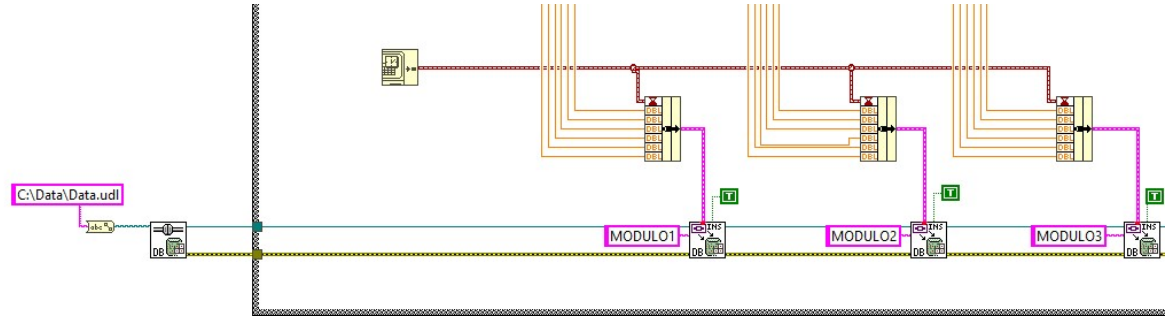


Ilustración 3-102 Conexiones para bloques “Insert”.

Se agregó un bloque “DB Tools close conection” fuera del ciclo “while loop” principal, mismo que se encuentra en el submenu que se muestra en la ilustración 3-93. Una vez colocado el bloque, se realizaron las conexiones que aparen en la isluatrcion 3-103.

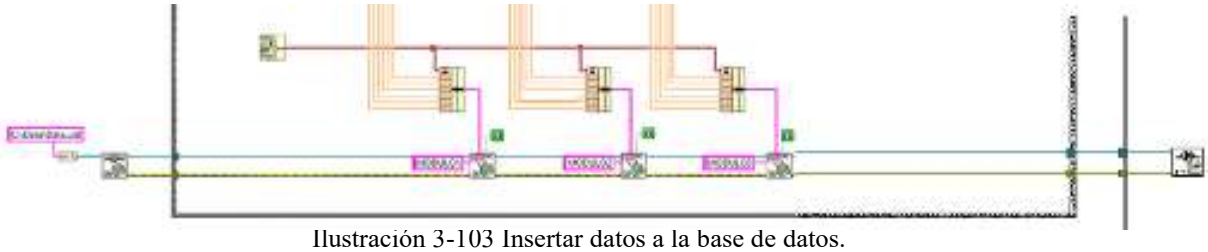


Ilustración 3-103 Insertar datos a la base de datos.

También se realizaron las conexiones en el caso “false” de la estructura “case” principal (Ilustración 3-104).

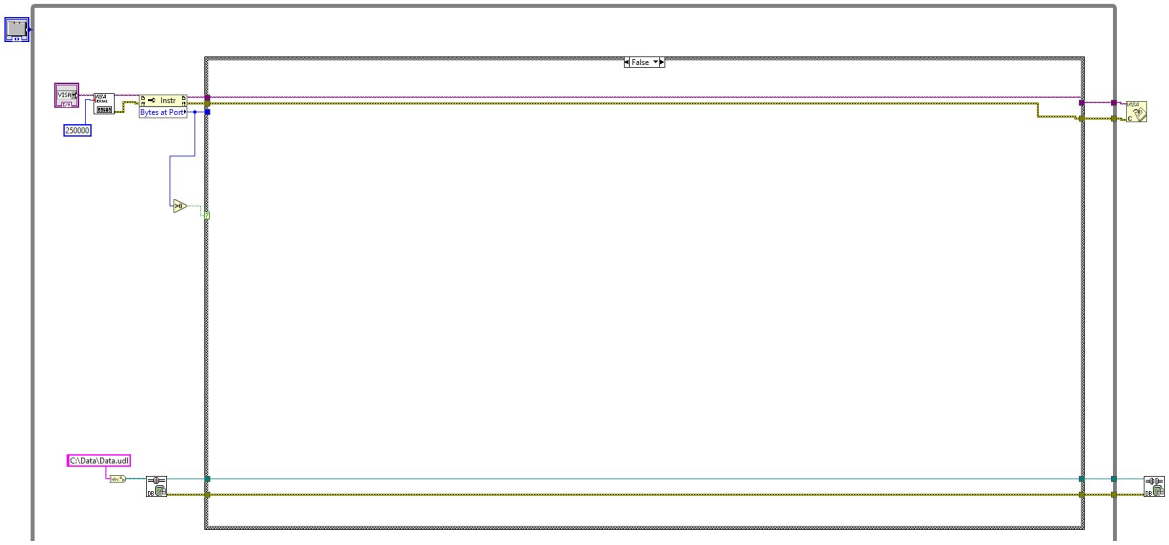


Ilustración 3-104 Conexiones para el caso "false" de la estructura "case" principal.

Por último, se tuvo que crear un temporizador que permitiera registrar las lecturas en la base de datos cada cierta cantidad de minutos, ya que el sistema recibe datos cada segundo y, registrar esta cantidad de datos ocuparía una cantidad considerable de espacio en la computadora al año, si bien la cantidad de espacio que se ocupa no es relevante para una computadora, el principal problema radica en que, al ser una cantidad muy grande de datos, el sistema puede alentarse al procesar esa cantidad de información, además de que los archivos con extensión ".mdb" no deben sobrepasar los 2 GB ya que de ser así, el archivo se corromperá, la solución que se le dio a este problema fue, registrar datos cada cinco minutos y de esta forma se reduce la cantidad de espacio que utiliza la base de datos y también se reduce la cantidad de datos que la interfaz gráfica tiene que procesar.

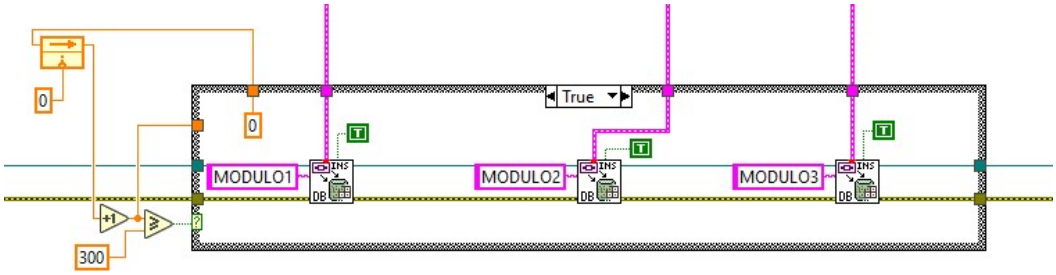


Ilustración 3-105 Temporizador para base de datos, parte 1.

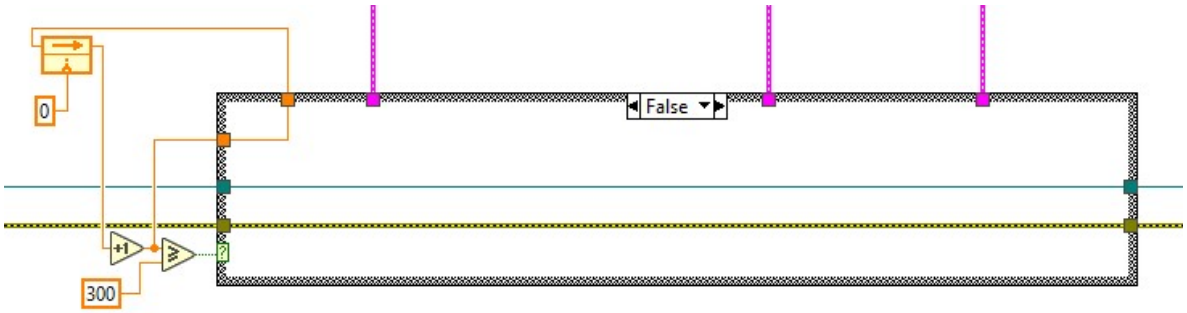


Ilustración 3-106 Temporizador para base de datos, parte 2.

3.5.2.4 Algoritmo para consultas de datos históricos

Hasta este punto el sistema es capaz de obtener y almacenar datos provenientes del módulo maestro, pero aún se debe crear un algoritmo que permita consultar los datos almacenados. Este algoritmo hace uso de una función “Select” del lenguaje de programación SQL, por lo que se debe crear las instrucciones necesarias para realizar los comandos que generan dichas consultas. Para crear este algoritmo se hace uso de los bloques creados durante la realización del apartado “Consultas” (Ilustración 3-107).

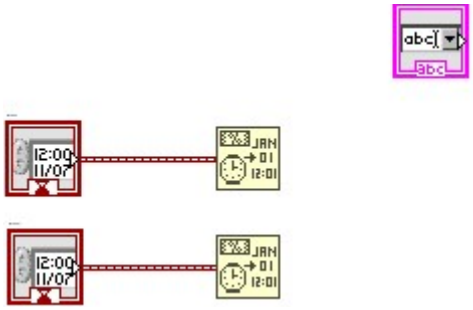


Ilustración 3-107 Bloques de control para algoritmo de "consultas".

El primer paso, es agregar a los bloques “Format date/Time string” el siguiente formato de fecha “%m/%d/%y”, este formato indica al bloque, que la fecha tendrá la disposición: “mes, día, año”, este formato el que utiliza SQL y si se utiliza algún otro formato, no se podrá generar la consulta.

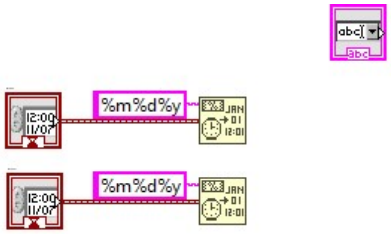


Ilustración 3-108 Asigna formato de fecha.

Se agregó un bloque “Concatenate”, que permite concatenar las cadenas de texto que conforman el algoritmo SQL.

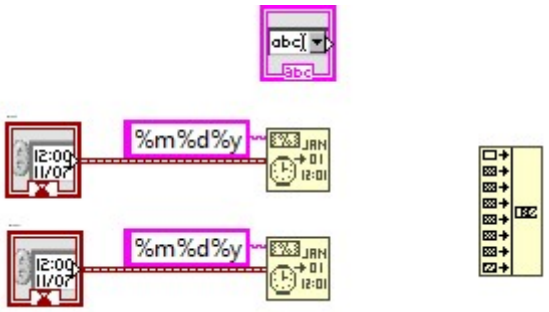


Ilustración 3-109 Algoritmo para generar consultas, parte 1.

Consecuente a esto, se agregó la primera parte del comando que compone al algoritmo, esta parte del algoritmo permite seleccionar los datos que se van a consultar, Uno de los datos consultados es la fecha, el segundo dato, lo selecciona el agricultor desde el menú desplegable de variables (Ilustración 3-110).

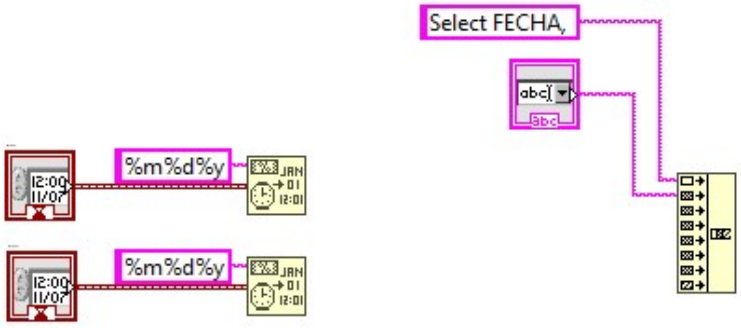


Ilustración 3-110 Algoritmo para generar consultas, parte 2.

En la siguiente parte del algoritmo se indica el nombre de la tabla en el que están alojados los datos a consultar (Ilustración 3-111).

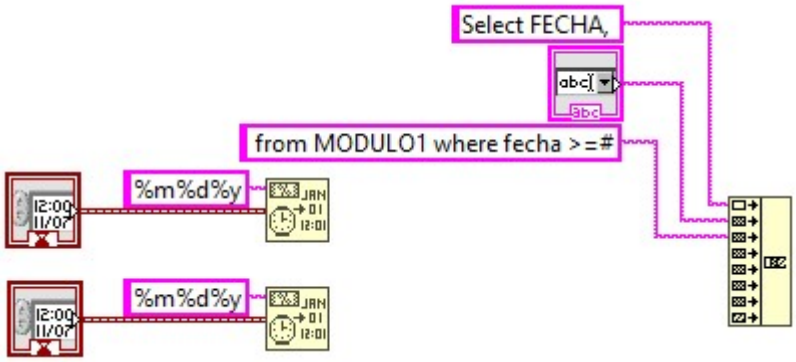


Ilustración 3-111 Algoritmo para generar consultas, parte 3.

Para indicar la fecha inicial desde la cual se van a consultar los datos se realizó la conexión que aparece en la ilustración 3-112.

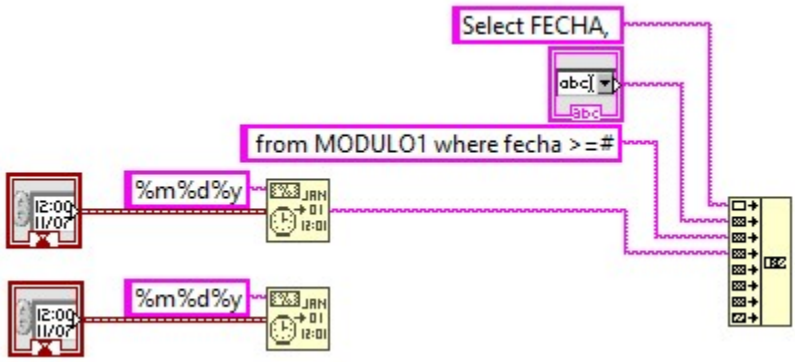


Ilustración 3-112 Algoritmo para generar consultas, parte 4.

Con las siguientes dos conexiones, se indica al algoritmo que consulte los datos solicitados entre la fecha inicial y la fecha límite que indique el agricultor (Ilustración 3-113).

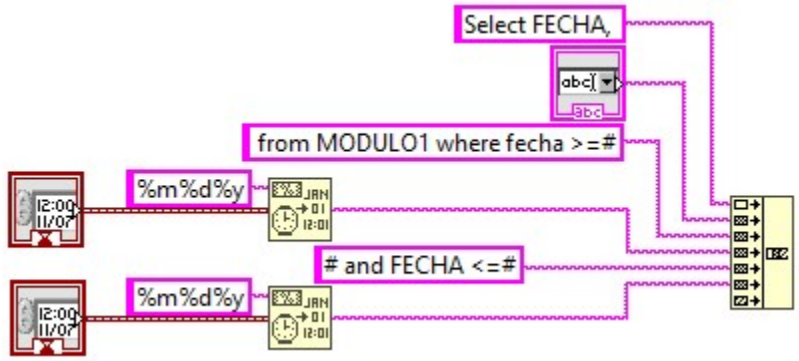


Ilustración 3-113 Algoritmo para generar consultas, parte 5

Por último, con la siguiente conexión se indica que, los datos consultados se deben ordenar por fecha de menor a mayor (Ilustración 3-114).

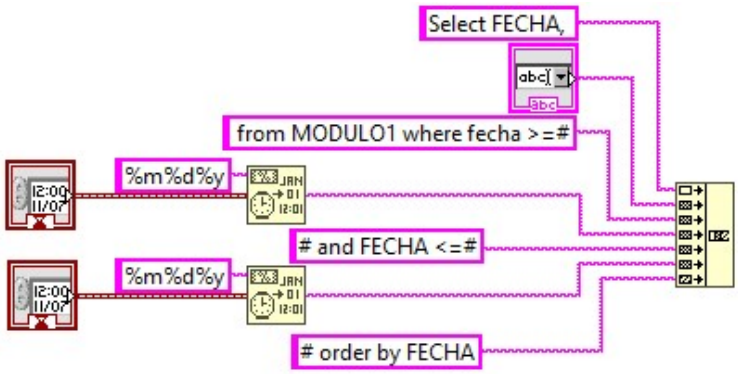


Ilustración 3-114 Algoritmo para generar consultas, parte 6.

Con el algoritmo que anterior, solo puede consultar datos de la tabla con el nombre “MÓDULO1”, es decir, datos de la zona 1, para poder consultar datos de las zonas 2 y 3, se ocupa el mismo algoritmo, pero se debe modificar el nombre de la tabla (Ilustración 3-115).

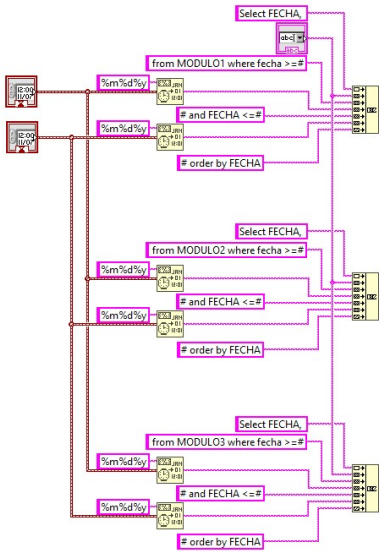


Ilustración 3-115 Algoritmo para generar consultas de todas las zonas.

Para agregar funcionalidad a los botones “Consultar” y “limpiar”, se colocaron los calendarios dentro de una estructura “case” y, al control de esta estructura se le conectó un reloj flip flop utilizando los botones anteriormente mencionados (Ilustración 3-116).

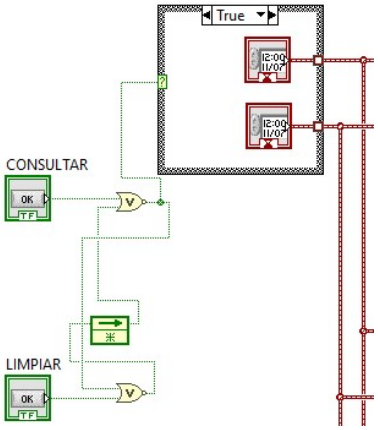


Ilustración 3-116 Función para el botón "Consultar".

Con la conexión anteriormente mostrada, se le agregó funcionalidad al botón” Consultar”, pero aún faltaba agregarle funcionalidad al botón “Limpiar”, para lograr esto, en el caso “False” de la estructura “case”, se agregan calendarios fijos con todos los parámetro en 0 (Ilustración 3-117), al hacer esto, se provoca que cuando se presione el botón “Limpiar” la

estructura “case” active el caso “False” y el algoritmo genere una consulta con las fechas inicial y final en 0 y, al no haber datos en este rango de fechas, automáticamente todos los datos que se pudieron haber consultado anteriormente se eliminan de las gráficas.

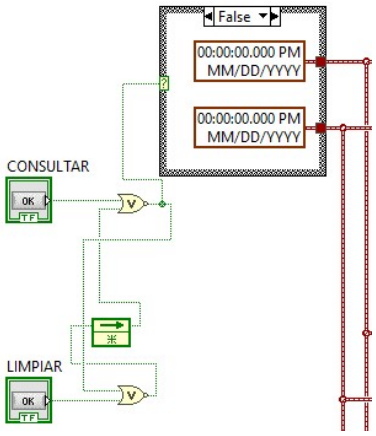


Ilustración 3-117 Función para el botón "Limpiar".

Para extraer todos los datos que el algoritmo indique y posteriormente mostrarlos en las gráficas que se crearon durante la realización del apartado gráfico de “Consultas”, se agregaron cuatro bloques (Ilustración 3-118), los cuales tienen como nombres: “DB Tools execute query”, “DB Tools fetch recordset data”, “DB Tools free object” y “Database variant to data”. los primeros tres bloques se pueden encontrar en el submenú “Connectivity” dentro del apartado “Database” y, el último de estos bloques se encuentra dentro del apartado “Advanced” del submenú “Connectivity”.

Ilustración 3-118 Extracción de información de la base de datos, parte 1.

Se colocaron tres grupos de los bloques anteriormente mencionados, un grupo por cada tabla de la base de datos. Una vez colocados los bloques mencionados, se colocó dos estructuras “For”, una dentro de otra, esta configuración de estructuras es importante para poder mostrar los datos consultados en las gráficas del apartado “Consultas”, dentro estas estructuras se debe colocar el bloque “Database variant to data” y realizar las conexiones que se muestran en la ilustración 3-119.

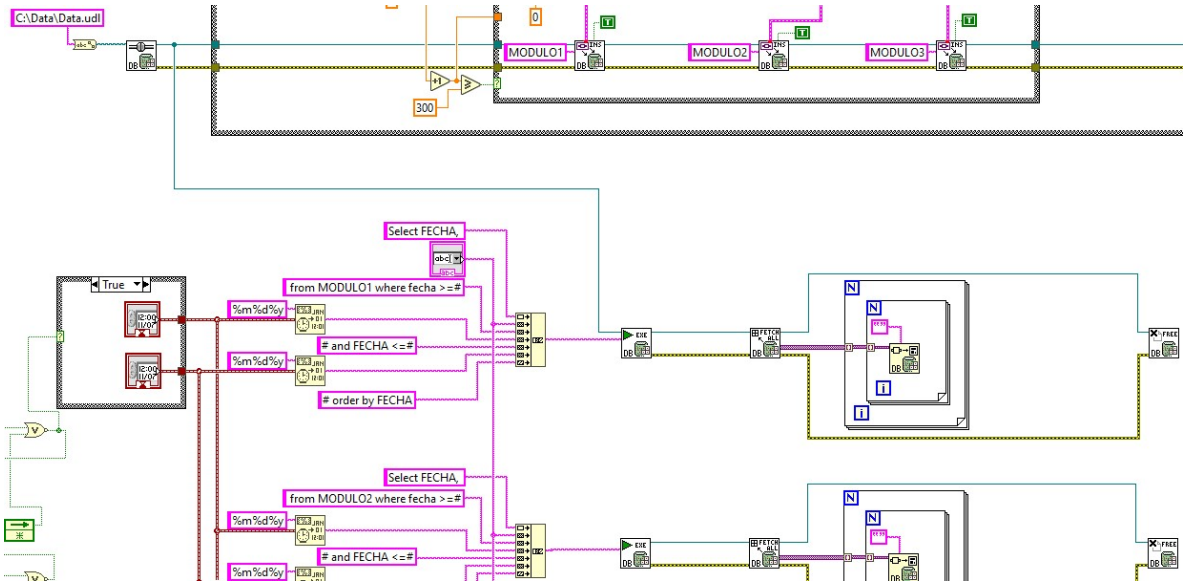
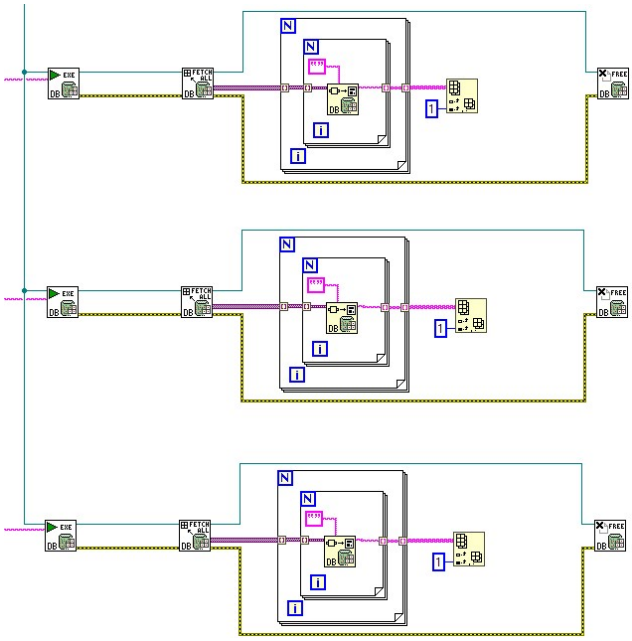


Ilustración 3-119 Extracción de información de la base de datos, parte 2.

Para mostrar los datos consultados en las gráficas que se colocaron durante la realización del apartado “Consultas”, es necesario indicar cual columna de datos se va a graficar, esto se logra con un bloque “Index array”, este bloque se encuentra en el apartado “Array” de la paleta de controles, en este caso, la columna que se va a graficar, es la columna 1.



. Ilustración 3-120 Extracción de información de la base de datos, parte 3.

para poder graficar los datos obtenidos de la consulta, se encontró con el inconveniente de que los datos obtenidos están en formato texto, por lo que primero se necesita convertir los datos a formato numérico, e indicar al bloque “Waveform graph” de qué forma se van a graficar los datos, esto se logra con los bloques “Decimal string to Number” y “Convert to dynamic data” respectivamente. El bloque “Decimal string to Number” se encuentra en el apartado “String” y el bloque “Convert to dynamic data” se encuentra en el submenú “Express” dentro del apartado “Sig manip”, al colocar este último bloque, se abrirá una ventana emergente en la cual se debe elegir la opción mostrada en la ilustración 3-121.

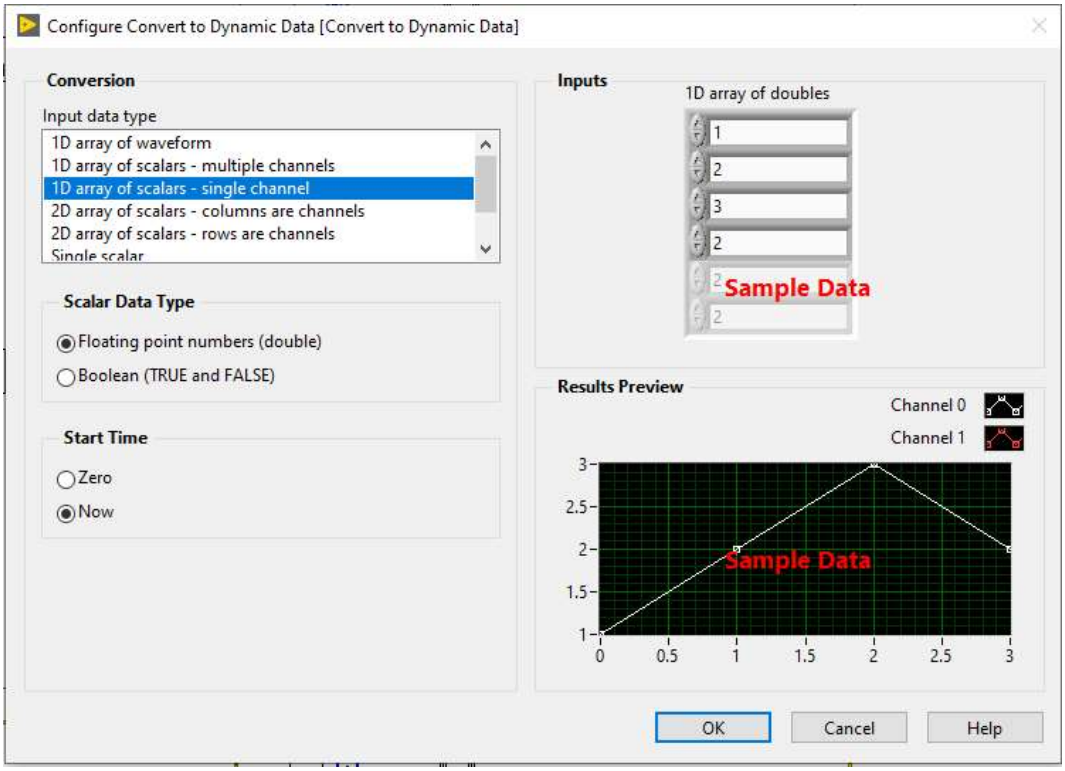


Ilustración 3-121 Configuración para el bloque "Convert to dynamic data".

Se agregaron tres de cada uno de los bloques mencionados anteriormente uno para cada zona del invernadero y, por último, se realizaron las conexiones que se muestran en la ilustración 3-122.

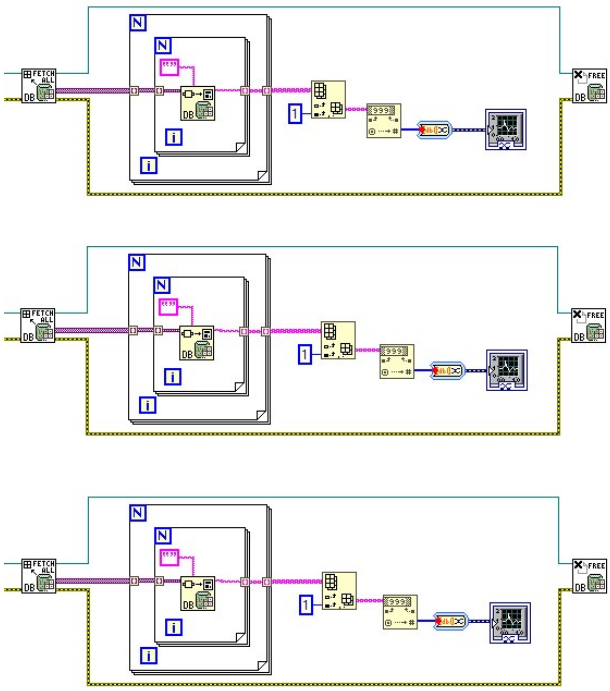


Ilustración 3-123 Mostrar datos consultados en gráficas, parte 1.

Consecuente a esto, se agregaron los bloques “Merge errors” y “DB tolos close connection”, el primero bloque se encuentra en el apartado “Dialog & user interface” y el último bloque se encentra en el submenú “Connectivity”, este bloque se debe agregar fuera del ciclo “while loop” principal, una vez colocados los bloques mencionados, se realizaron las conexiones que se muestran en la ilustración 3-124.

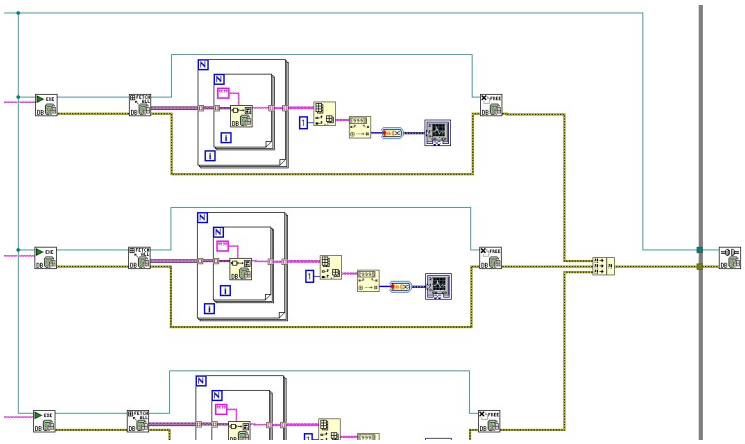


Ilustración 3-124 Mostrar datos consultados en gráficas, parte 2.

Para finalizar la programación del apartado gráfico, se agregó un bloque “Wait (ms)” y a la única terminal de conexión que tiene, se le conectó un numero constante con el valor 1000 (Ilustración 3-125), el cual permite establecer la frecuencia de muestreo de un segundo.

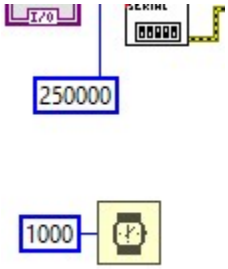


Ilustración 3-125 Frecuencia de muestreo.

3.5.3 Obtención del archivo ejecutable (.exe)

Lo que permite este archivo, es que el agricultor ya no tenga acceso al diagrama de bloques ni pueda modificar ningún parámetro del panel frontal, solamente tiene disponibles las funciones necesarias para que la interfaz gráfica funcione correctamente, el objetivo de este paso es que el agricultor no modifique accidentalmente alguna función o algún parámetro que sea necesario para el correcto funcionamiento del sistema. Los pasos que se siguieron para la obtención de este archivo se describen a continuación; en l la pestaña “Tools” se seleccionó la opción “Build Aplication (EXE) from VI…” (Ilustración 3-126).

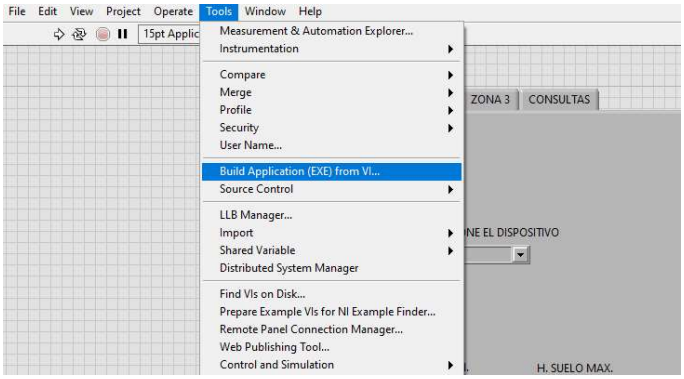


Ilustración 3-126 Buid Aplication (EXE) from VI....

Al seleccionar la opción antes mencionada, se abrirá una ventana emergente que permite seleccionar la ruta en la cual se guardara el archivo (Ilustración 3-127), en este caso el archivo se alojó en la carpeta “Data” que se creó durante la realización de la base de datos.

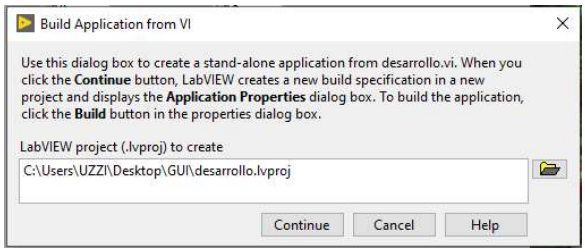


Ilustración 3-127 Seleccionar ruta para el archivo ejecutable.

Para poder seleccionar la carpeta en la que se alojará el archivo con extensión ". Ivproj", se da clic en el icono de la carpeta, al realizar esta acción se abrirá una ventana emergente que permite modificar el nombre del archivo, así como facilitar la búsqueda de la carpeta "Data" (Ilustración 3-128).

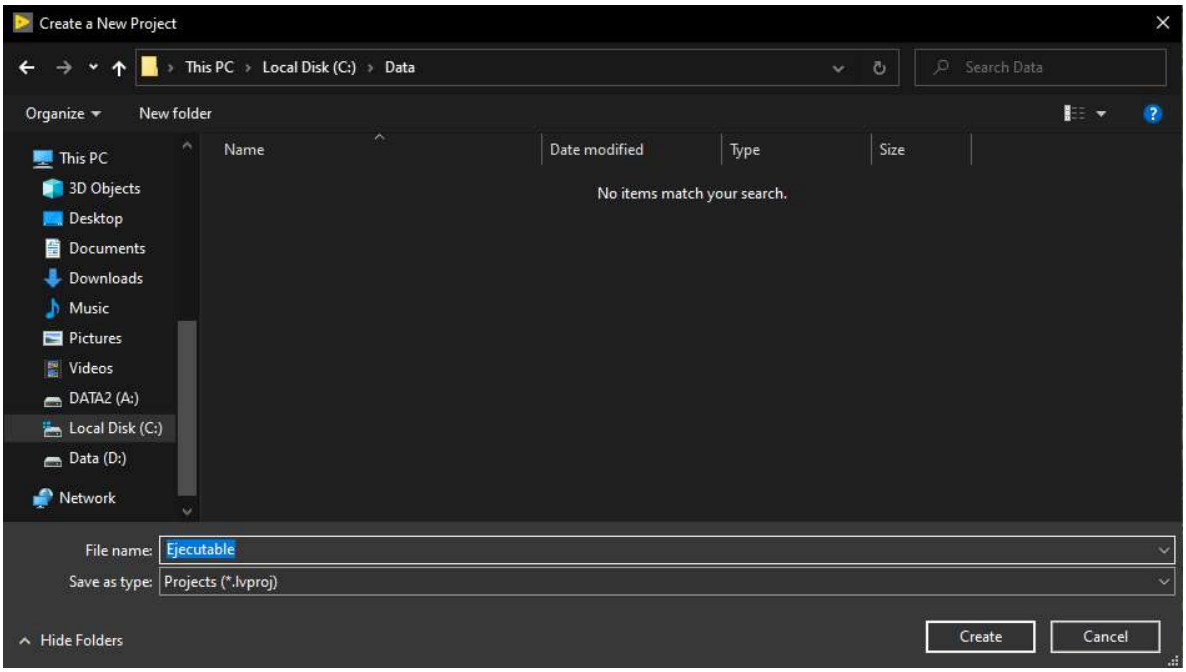


Ilustración 3-128 Asignación de nombre al archivo con extensión ". Ivproj".

En este caso, al archivo se le asignó el nombre de "Ejecutable" aunque pudo ser cualquier otro, una vez seleccionada la carpeta "Data" y asignado el nombre, se da clic en el nombre crear, al realizar esta acción se regresará a la ventana emergente de la ilustración 3-127, con el nombre y la ruta modificados se da clic en el botón "Continue". Al realizar esta acción se abrirán dos ventanas emergentes que se muestran en las ilustraciones 3-129 y 3-130.

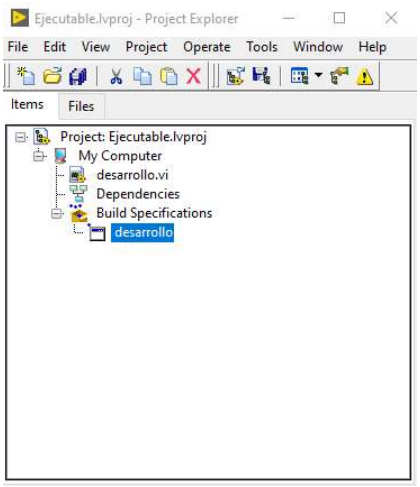


Ilustración 3-129 Explorador de archivos del proyecto.

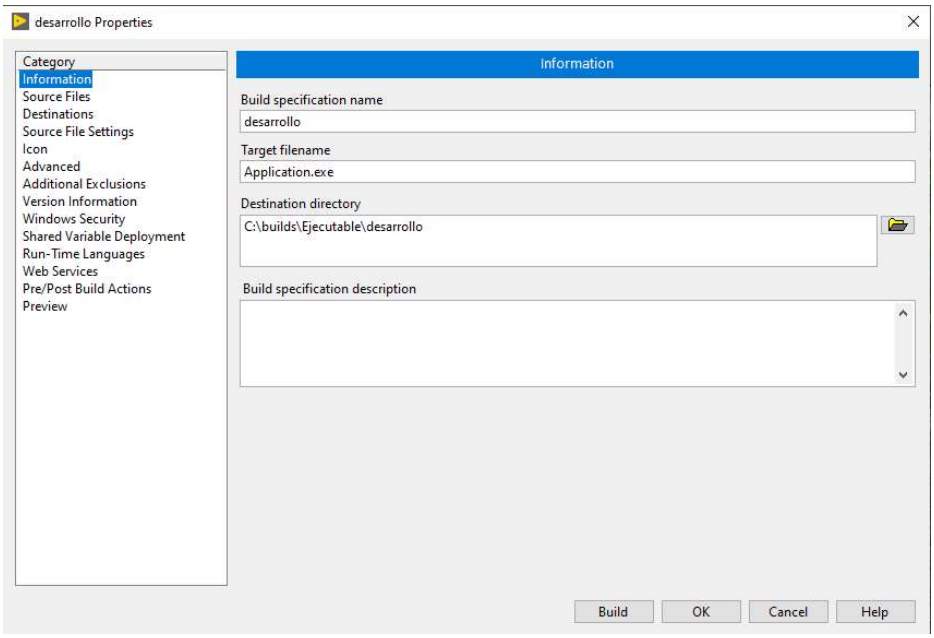


Ilustración 3-130 Propiedades del proyecto.

La ilustración 3-130 muestra la ventana en la que se pueden modificar las propiedades que tendrá el archivo, en este caso solo se modificó la carpeta que alojará el archivo con extensión ".exe", aunque también se puede modificar el icono con el que aparecerá el archivo ejecutable en la computadora. Para poder seleccionar la carpeta "Data", se da clic en el botón que tiene el icono de una carpeta, al realizar esta acción se abrirá una ventana que permite crear una carpeta nueva o utilizar la carpeta abierta (Ilustración 3-131).

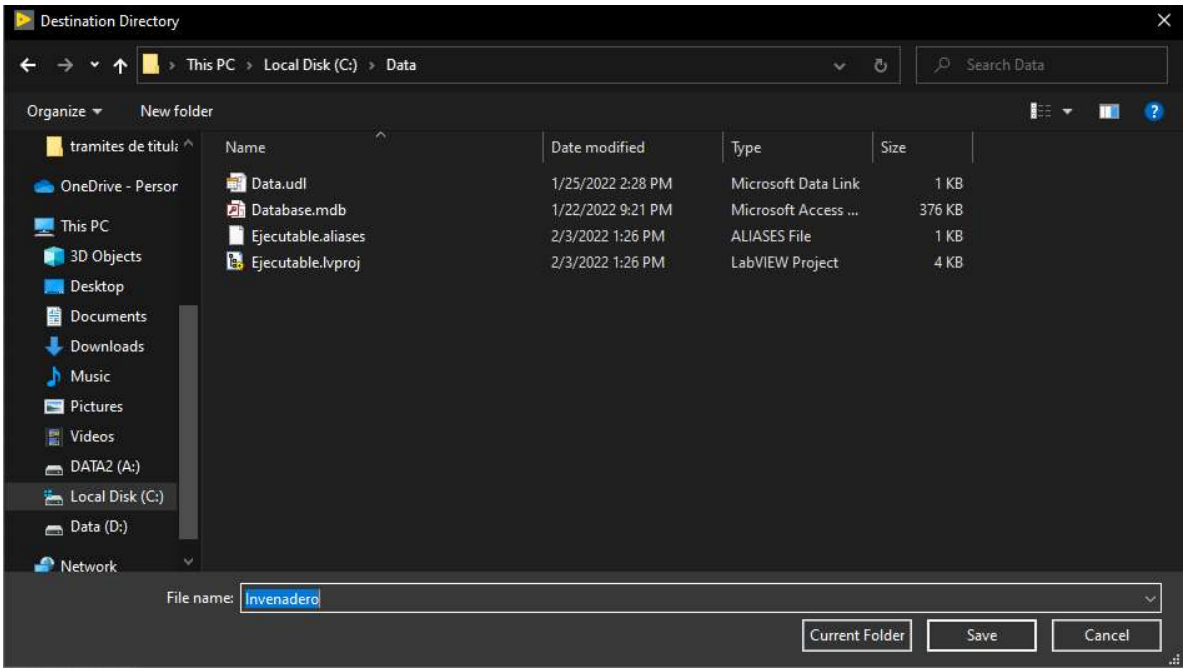


Ilustración 3-131 Asignación de nombre para el archivo con extensión ".exe".

Una vez abierta la carpeta “Data”, se da clic en el botón “Current Folder”, al realizar esta acción nuevamente se abrirá la ventana que aparece en la ilustración 3-130 en la cual se debe de modificar el nombre que tendrá el archivo ejecutable, para asignar el nombre se debe modificar el texto “Application.exe” por el nombre que tendrá el archivo, en este caso el nombre que se le dio al archivo es “Invernadero”, por último se da clic en el botón “Build”, al dar clic en el botón, se cerrará la ventana de la ilustración 3-130 y se abrirá una nueva ventana emergente en la que aparecerá una barra que indica el proceso de creación del archivo ejecutable (Ilustración 3-132).

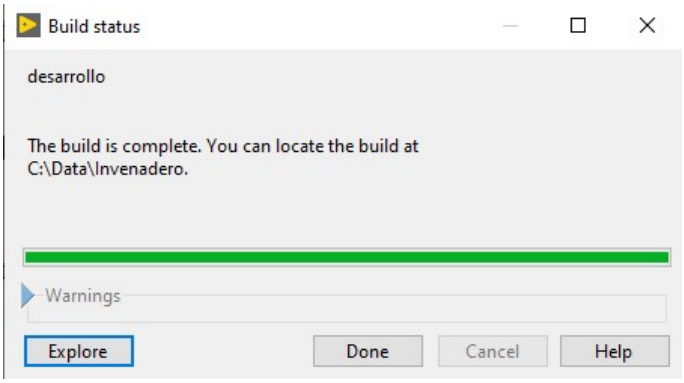


Ilustración 3-132 Estatus del proceso de creación del archivo ejecutable.

Una vez que haya finalizado el proceso, se puede dar clic en el botón “Explore” que abrirá la carpeta en la que se ha creado el archivo (Ilustración 3-133), en este caso es la carpeta “Data”.

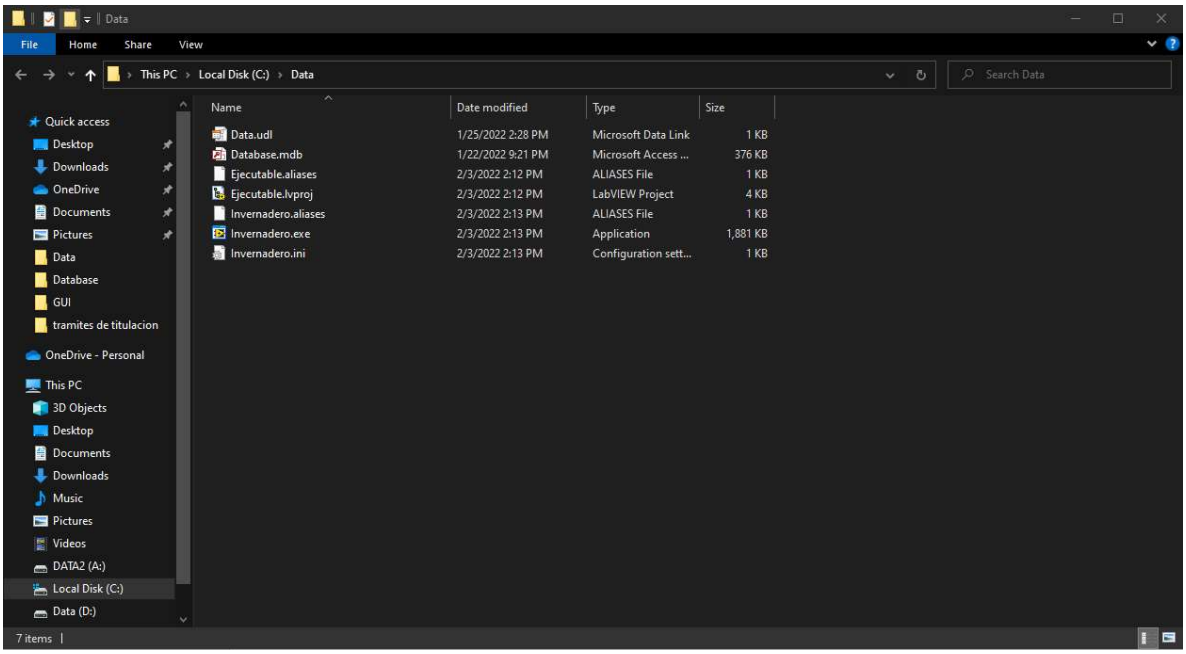


Ilustración 3-133 Archivos de la interfaz gráfica de usuario.

Para finalizar con el proceso, se da clic derecho sobre el archivo con el nombre “Invernadero.exe” y se selecciona la opción “Send to Desktop (create shortchut)” para crear un acceso directo desde el escritorio y es más fácil para el agricultor abrir la interfaz gráfica.

Al dar doble clic sobre el acceso directo que se acaba de crear, se abrirá la interfaz gráfica sin el diagrama de bloques y sin la posibilidad de modificar parámetros del panel frontal (Ilustración 3-134).

Ilustración 3-134 Interfaz gráfica de usuario.

4. Pruebas y resultados

Para poder asegurar que las lecturas obtenidas por los sensores de humedad sean lo más fiables posibles sin verse alterados por la cantidad de minerales que contenga el sustrato o el agua que se utilice para el riego, fue necesario realizar varias pruebas de laboratorio, a fin de obtener un modelo matemático que permita convertir el nivel de voltaje proporcionado por el sensor, en una lectura del porcentaje de humedad presente en el sustrato. También fue necesario realizar varias pruebas para poder enviar datos a largas distancias sin que se vean alterados o la comunicación sea inestable y, por último, se realizaron pruebas de conexión entre el módulo maestro y la etapa 3 para que las lecturas obtenidas por los sensores se muestren lo más pronto posible en la interfaz gráfica, así como el almacenamiento de estas lecturas en la base de datos.

4.1 Pruebas de comunicación entre módulo maestro y etapa 3

El objetivo principal de estas pruebas es ver si es posible comunicar el microcontrolador que forma al módulo maestro, con el software LabVIEW bajo el cual está diseñada la interfaz gráfica, de ser posible comunicarlos, se debe de obtener el método más rápido y más sencillo.

Para la primera propuesta, se tomó como ejemplo la lectura digitalizada de un potenciómetro conectada a una placa Arduino, la cual enviara los datos obtenidos atreves del puerto USB de la placa, las conexiones realizadas para esta prueba se muestran en la ilustración 4-1.

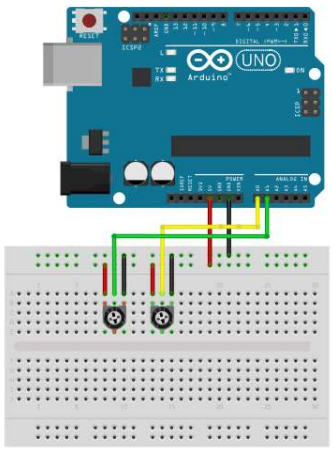


Ilustración 4-1 Conexión para prueba 1 de comunicación entre etapa 2 y 3.

El código necesario para poder realizar esta prueba se muestra en la ilustración 4-2, el cual consiste únicamente en el uso de dos puertos ADC de los 6 con los que dispone la tarjeta de Arduino Uno, estas lecturas se envían hacia la computadora que contiene la interfaz gráfica atreves del puerto serie.

```
void setup() {
  Serial.begin(9600);

}

void loop() {
  int lectural = analogRead(A0);
  int lectura2 = analogRead(A1);
  Serial.print(lectural);
  Serial.print(" ");
  Serial.println(lectura2);

}
```

Ilustración 4-2 Código para prueba 1 de comunicación entre etapa 2 y 3.

Cómo se puede observar en la ilustración 4-2, las dos lecturas se envían por el puerto serie y están separados por un espacio, es necesario dejar este espacio entre datos para que el programa de LabVIEW pueda identificar los valores de forma individual.

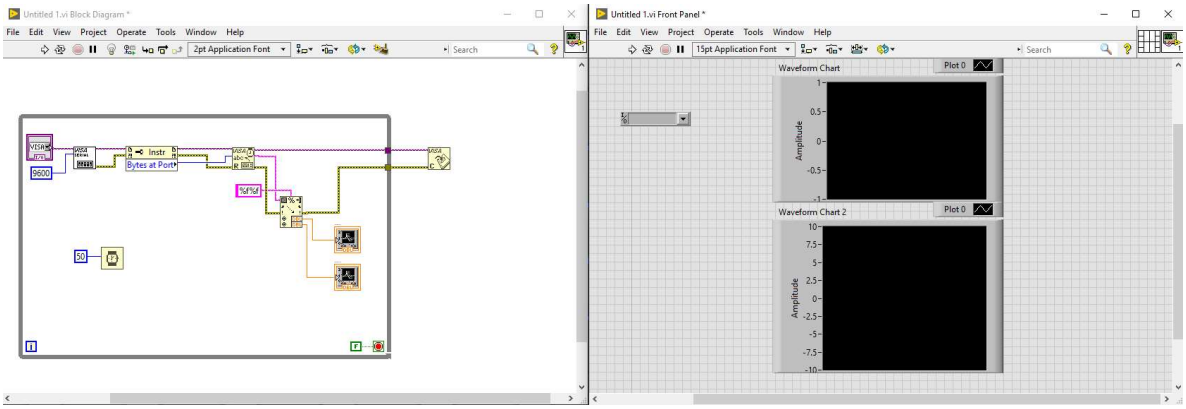


Ilustración 4-3 Prototipo 1 de programa adquisición de datos en LabVIEW.

El programa que se muestra en la ilustración 4-3, pertenece al primer prototipo de interfaz gráfica para la adquisición de datos, el problema con este prototipo es la sincronización entre el envío de datos del Arduino con el flujo de programa de LabVIEW, esto provoca que, aun cuando no hay datos en el puerto serie, el indicador gráfico sigue mostrando valores, lo cual produce que se muestre una la lectura errónea, ya que como no dispone de datos entrantes

este gráfica el valor por default el cual es 0. Este problema se puede corregir agregando una estructura “case” que permita mostrar datos en el indicador gráfico solamente cuando haya datos en el puerto serie, en la ilustración 4-4, se muestra la corrección que se tuvo que aplicar para corregir el problema anteriormente mencionado.

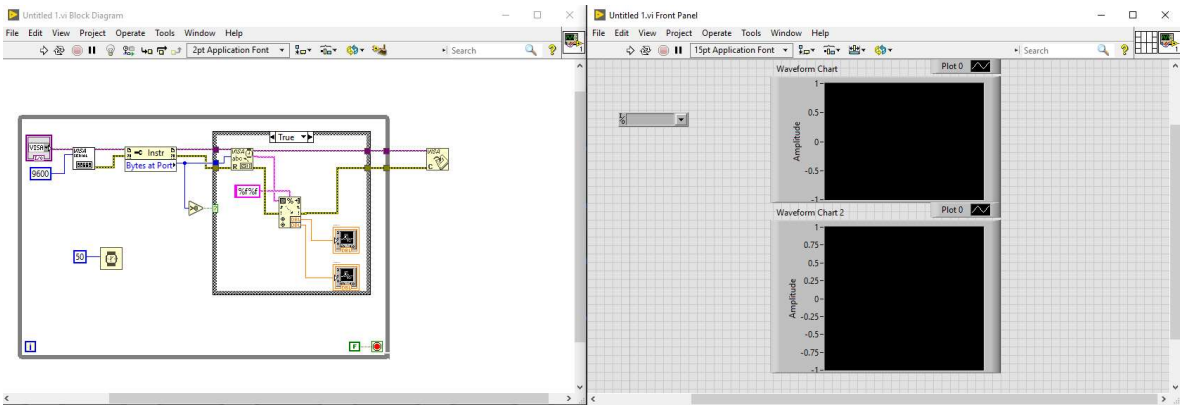


Ilustración 4-4 Prototipo 2 de programa adquisición de datos en LabVIEW.

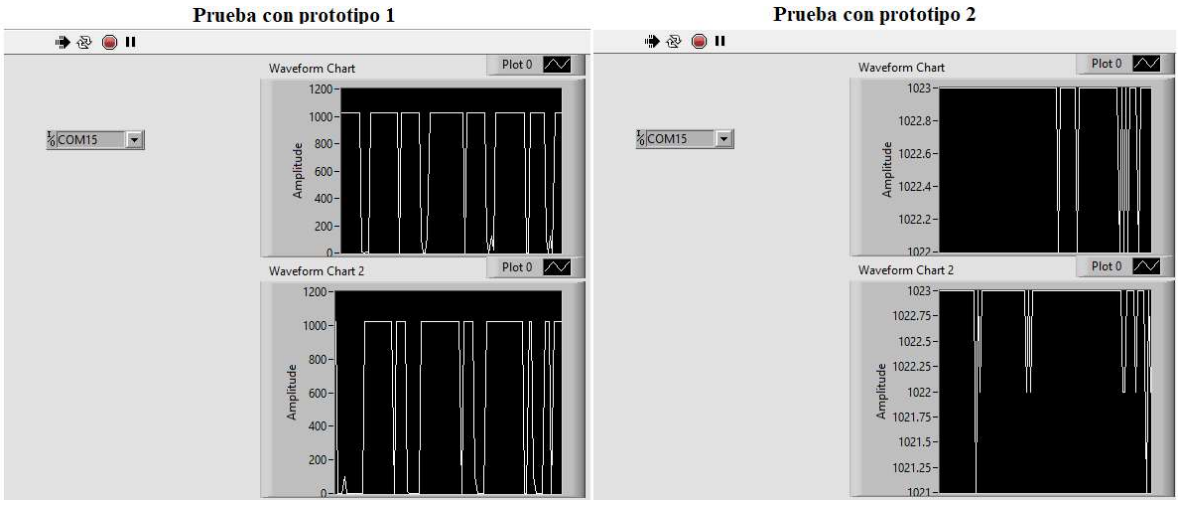


Ilustración 4-5 Resultados de los prototipos 1 y 2.

Como se puede observar en la ilustración 4-5, en los resultados que se obtuvieron del prototipo 1 las lecturas oscilan entre los valores 1024 y 0, cuando el indicador gráfico marca el valor 0 es debido a que en ese momento no había datos en el puerto serie por lo que el indicador gráfica el valor predeterminado en el caso de que no haya nada que graficar, mientras que en los resultados de la prueba con el prototipo 2, se puede observar que los valores oscilan entre 1024 y 1020, este es un comportamiento normal ya que las variaciones

que presenta la lectura corresponden a las ligeras vacaciones de voltaje que presenta la fuente de alimentación que se utilizó para alimentar los potenciómetros utilizados para esta prueba.

La siguiente prueba consiste en enviar datos desde LabVIEW hacia el Arduino, esto se utilizará para que el agricultor pueda establecer los rangos óptimos para el envío de alertas GSM. Para esta prueba se utilizará un Arduino Uno con una pequeño LCD 16x2 (Ilustración 4-6).

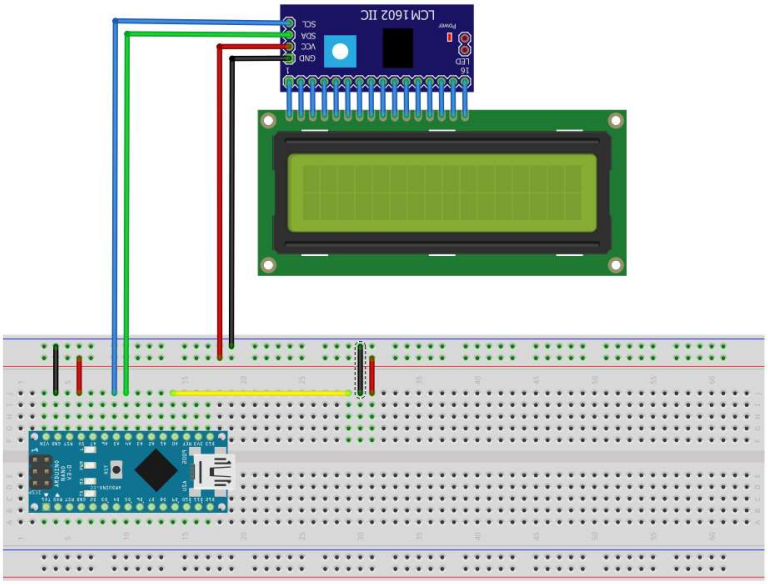


Ilustración 4-6 Arduino con LCD 16x2.

Con el circuito anterior se busca mostrar los datos recibidos por el Arduino Uno en el LCD, estos datos se enviarán desde LabVIEW. La finalidad de esta prueba es comprobar si se pueden recibir datos de forma correcta y estable desde LabVIEW.

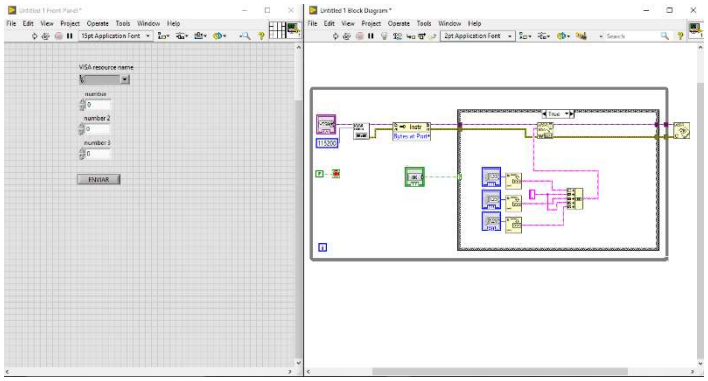


Ilustración 4-7 Interfaz de LabVIEW para prueba 2.

La interfaz mostrada en la ilustración 4-7, tiene la función de enviar tres números al Arduino Uno en formato de texto separados con una coma.

```
LiquidCrystal_I2C lcd(0x27,16,2);
#include <Separador.h>
Separador s;

void setup(){
  Serial.begin(115200);
  lcd.init();
  lcd.backlight();
  lcd.clear();
}
void loop(){
  if(Serial.available()){
      lcd.clear();
      String recibidos=Serial.readString();
      lcd.print(recibidos);
      String dato1=s.separa(recibidos, ',', 0);
      lcd.setCursor(0,1);
      lcd.print(dato1);
      String dato2=s.separa(recibidos, ',', 1);
      lcd.setCursor(3,1);
      lcd.print(dato2);
      String dato3=s.separa(recibidos, ',', 2);
      lcd.setCursor(6,1);
      lcd.print(dato3);
  }

}
```

Ilustración 4-8 Código de Arduino Uno para prueba 2 de comunicación.

Una vez que el Arduino Uno reciba los datos, debe mostrar la cadena de texto recibida, así como los números de forma independiente en formato numérico, para lograr esto se utilizó el código mostrado en la ilustración 4-8.

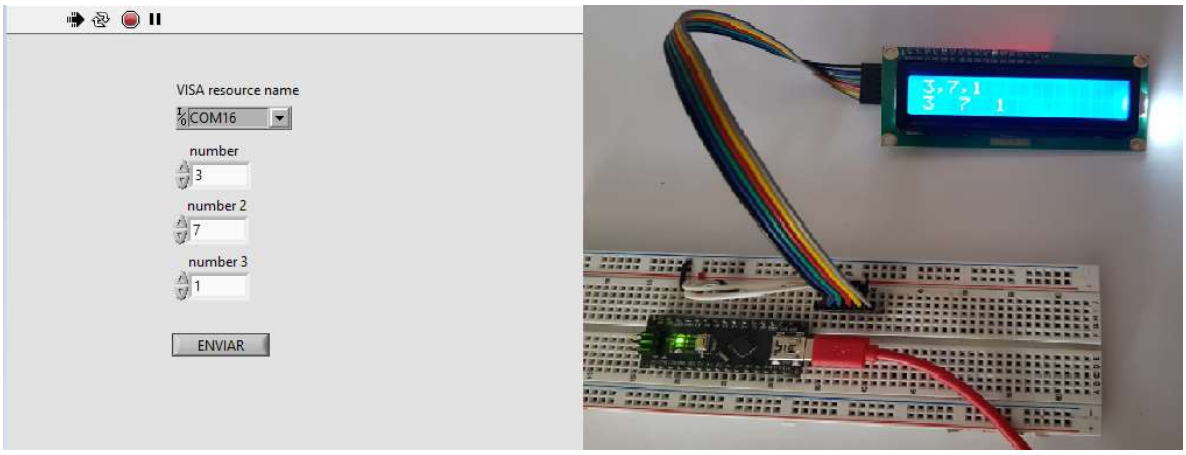


Ilustración 4-9 Resultados de la prueba 2 de comunicación.

En la ilustración 4-9 se observan los resultados de la prueba realizada, en esta prueba se enviaron los números 3, 7 y 1 desde la interfaz de LabVIEW, mismos que se muestran en el

LCD conectado al Arduino Uno, por lo que se puede decir que la prueba se realizó de forma exitosa.

4.2 Pruebas para los sensores de humedad de suelo

Como ya se mencionó en capítulos anteriores, los sensores para medir la humedad de suelo, proporcionan una señal analógica, por lo que es necesario convertir el valor analógico proporcionado por el sensor, en un porcentaje de humedad presente en el sustrato, donde el 100% corresponde a la saturación de humedad en el sustrato y 0% corresponde a un sustrato completamente seco. El hecho de que se agreguen sales y minerales al agua empleada para el riego de los cultivos dificulta realizar una medición fiable de humedad, ya que los métodos que los sensores emplean para realizar estas mediciones pueden verse afectados por la alteración de la conductividad del agua que producen estas sales y minerales.

El primer paso para poder realizar las pruebas con los sensores de humedad de suelo, es tomar muestras de agua que se utiliza para el riego de los cultivos, una vez conseguido estas muestras, se les realiza una prueba de conductividad, además se toma una muestra de agua destilada que, al tener muy pocas sales y minerales, es menos conductiva y permite observar como la conductividad del agua afecta a las lecturas proporcionadas por los sensores.

Ilustración 4-10 Prueba de conductividad para agua destilada (8uS/cm).

Ilustración 4-11 Prueba de conductividad para la muestra de agua 1 (2027uS/cm).

Ilustración 4-12 Prueba de conductividad para la muestra de agua 2 (3008uS/cm).

Una vez conociendo la conductividad de las muestras de agua tomadas de los contenedores utilizados para el riego de los cultivos, se deben etiquetar para evitar confundir las muestras (Ilustracion 4-13).

Ilustración 4-13 Muestras de agua.

También es necesario tomar muestras del sustrato utilizado para los cultivos, en este caso el sustrato utilizado es la fibra de coco. Las muestras de sustrato son tres: sustrato nuevo, sustrato de un uso y sustrato de desecho, para el caso del sustrato nuevo, como no se ha utilizado para cultivo, no tiene sales remanentes ni minerales de riegos anteriores, por otra parte, el sustrato de un uso ya se utilizó para el cultivo, por lo que ya cuenta con sales y minerales remanentes, y por último, se tiene el sustrato para desecho, que ya no se puede utilizar para cultivar nuevamente debido a las sales y minerales remanentes que ha acumulado de las veces que se ha utilizado, con estas muestras se pretende conocer cómo afectan a las lecturas de los sensores las sales y minerales tanto del agua utilizada como del propio sustrato, es importante saber este dato ya que se debe conocer hasta qué punto vale la pena comprar un sensor de mayor costo como lo es el VH400.

Ilustración 4-14 Muestras de sustratos.

La primera prueba se realizó con el sustrato nuevo, lo primero es asegurar que el sustrato este completamente seco, para lograr esto se utilizó un deshidratador de alimentos y una malla de tela que permita contener el sustrato y a su vez permita la evaporación de la humedad (Ilustración 4-15).

Ilustración 4-15 Deshidratación del sustrato.

El sustrato se somete a un proceso de deshidratación durante 8 horas a una temperatura de 60°C (Ilustración 4-16).

Ilustración 4-16 Proceso de deshidratación del sustrato.

Una vez terminado el proceso de deshidratación del sustrato, se debe tomar una muestra del sustrato seco en la cual se realizarán las pruebas de los sensores, pero antes de este paso, se necesita de un recipiente en el cual se alojará la muestra del sustrato, mismo que se debe de pesar con ayuda de una báscula (Ilustración 4-17).

Ilustración 4-17 Peso de recipiente 47g.

Restando el peso del recipiente, se toma una muestra de aproximadamente 150g de sustrato seco, el peso de la muestra se puede variar, en este caso con la cantidad de sustrato indicada es suficiente para realizar estas pruebas (Ilustración 4-18).

Ilustración 4-18 Peso de la muestra de sustrato.

Teniendo la muestra de sustrato, se debe agregar agua a fin de conocer la cantidad necesaria para saturar la muestra, el punto de saturación se refiere a cuando el sustrato ya no puede absorber más humedad y se comienza a encharcar, en el caso de esta muestra, el punto de saturación se alcanzó aproximadamente a los 720g de agua, o lo que es lo mismo 720 ml de agua (Ilustración 4-19), es importante recordar que un mililitro de agua pura es equivalente a un gramo, si bien el agua que se utilizó para saturar la muestra, no es agua 100% pura, las

variaciones entre el peso de un mililitro del agua utilizada no son relevantes para los resultados obtenidos.

Ilustración 4-19 Peso del agua para saturación de la muestra.

De los datos obtenidos en los pasos anteriores, solo son relevantes: el peso de la muestra del sustrato (150g) y los mililitros de agua necesarios para saturar la muestra de sustrato (720ml), este último dato se debe dividir entre la cantidad de mediciones que se desean tomar con los sensores, si se hacen pocas mediciones el procedimiento será más rápido, pero el resultado es más inexacto y, si se toman muchas muestras, el procedimiento será más lento pero el resultado será más exacto, en este caso se realizaran 20 lecturas, realizando la operación 720ml entre 20, da como resultado 36ml, por lo que cada medición se realizara con 36ml de diferencia. Para realizar estas mediciones, con ayuda de una báscula se toma una muestra de 150g de sustrato completamente seco, después con ayuda de una jeringa se agregan 36ml de agua destilada (Ilustración 4-20) y se mueve la muestra de sustrato a fin de esparcir uniformente el agua agregada.

Ilustración 4-20 Irrigación controlada de la muestra.

Por último, se deben tomar mediciones con los sensores de humedad de suelo, para hacer uso de los sensores y visualizar las mediciones, se utilizó un Arduino Uno con un LCD 16x2 que servirá para mostrar el voltaje proporcionado por los sensores (Ilustración 4-6).

```
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2);

void setup(){
  lcd.init();
  lcd.backlight();
  lcd.clear();
}
void loop(){
  int medicion = analogRead(A0);
  lcd.print(medicion*0.004887585);
  delay(200);
}
```

Ilustración 4-21 Código para del microcontrolador del circuito auxiliar

Las mediciones se deben realizar sensor por sensor, sin hacer uso de varios sensores a la vez, esto para evitar que puedan llegar a interferir entre sí (ilustraciones 4-22, 4-23 y 4-24).

Ilustración 4-22 Prueba con higrómetro FC-28.

Ilustración 4-23 Prueba con sensor capacitivo.

Ilustración 4-24 Prueba con sensor VH400.

Estas mediciones se deben de realizar con todas las muestras de sustrato que se tomaron del invernadero, así como con todas las muestras de agua obtenidas de los depósitos del invernadero, además de realizar las pruebas con agua destilada para poder conocer que tanto afecta la conductividad del agua a las lecturas de los sensores. Es importante mencionar que el objetivo principal de estas pruebas es poder obtener un modelo matemático que describa el comportamiento de los sensores lo más preciso posible además de conocer cuáles son los sensores que si se pueden utilizar y cuáles no, y hasta qué punto son fiables.

4.2.1 Resultados de las pruebas para los sensores de humedad de suelo

Los resultados obtenidos de las mediciones para el sustrato nuevo se muestran las tablas 4-1, 4-2 y 4-3.

Tabla 4-1 Sustrato nuevo con agua destilada.

Agua destilada (ml)	Sensor FC-28 (v)	Sensor capacitivo (v)	Sensor VH400
0	5	2.85	0.11
36	4.89	2.63	0.47
72	4.26	2.54	0.67
108	3.92	2.43	0.89
144	3.26	2.34	1.07
180	3.17	2.23	1.16
216	3.03	2.11	1.2
252	2.86	2	1.25
288	2.6	1.92	1.3
324	2.3	1.87	1.38
360	2.15	1.78	1.53
396	2.08	1.63	1.62
432	1.94	1.54	1.73
468	1.73	1.44	1.84
504	1.4	1.3	1.95
540	1.26	1.21	2.1
476	1.19	1.2	2.35
612	0.99	1.2	2.55
648	0.85	1.2	2.7
684	0.74	1.2	2.81
720	0.73	1.2	2.99

Las gráficas del comportamiento que presentan los sensores con agua destilada y sustrato nuevo se muestran en las ilustraciones 4-25, 4-26 y 4-27.

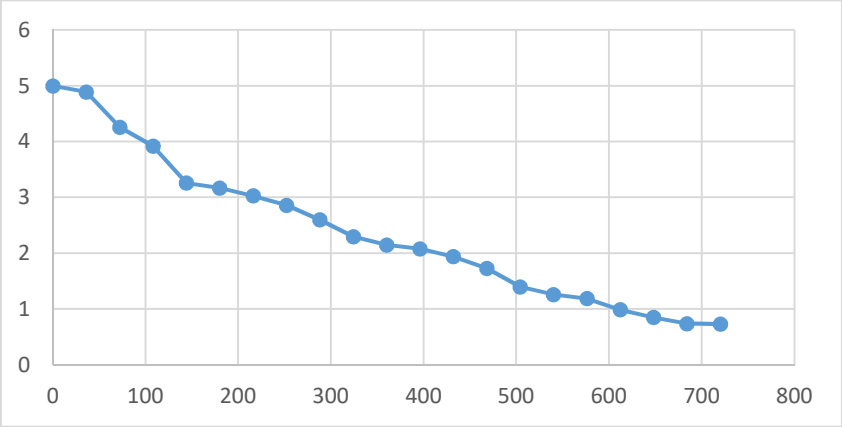


Ilustración 4-25 Gráfica para el sensor FC-28 con sustrato nuevo y agua destilada.

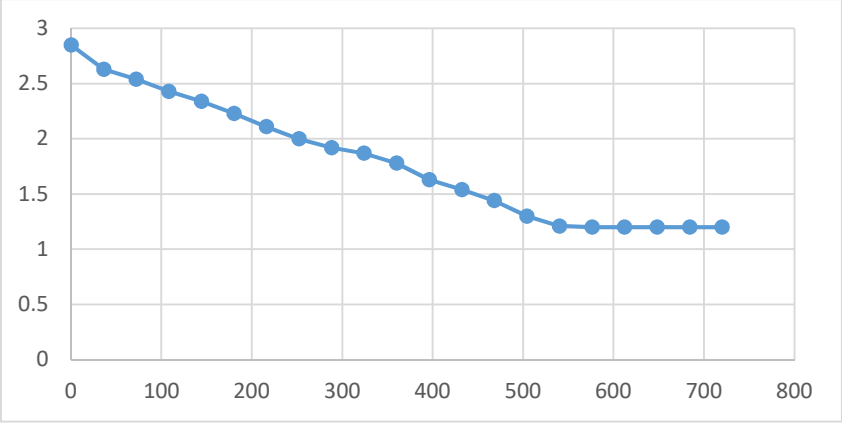


Ilustración 4-26 Gráfica para el sensor capacitivo con sustrato nuevo y agua destilada.

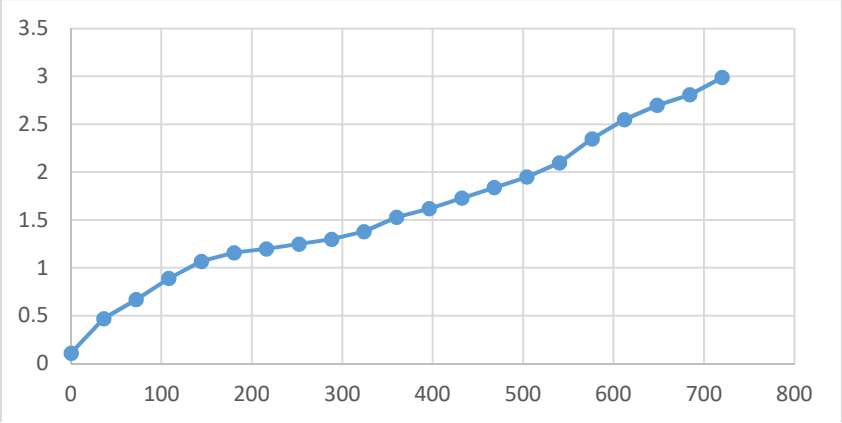


Ilustración 4-27 Gráfica para el sensor VH400 con sustrato nuevo y agua destilada.

Tabla 4-2 Sustrato nuevo con agua conductiva 2027uS/cm.

Agua conductiva 2027uS/cm	Sensor FC-28	Sensor capacitivo	Sensor VH400
0	5	2.85	0.11
36	4.57	2.55	0.48
72	3.7	2.42	0.62
108	3.35	2.33	0.83
144	3.04	2.25	1
180	2.85	2.17	1.17
216	2.55	2.1	1.23
252	2.33	2	1.37
288	2.17	1.93	1.54
324	2	1.82	1.62
360	1.85	1.7	1.75
396	1.72	1.58	1.85
432	1.53	1.43	2
468	1.37	1.3	2.18
504	1.27	1.21	2.28
540	1.19	1.2	2.37
576	1.12	1.2	2.46
612	1	1.2	2.54
648	0.87	1.2	2.63
684	0.78	1.2	2.83
720	0.7	1.2	3.02

Las gráficas del comportamiento que presentan los sensores con agua conductiva 2027uS/cm y sustrato nuevo se muestran en las ilustraciones 4-28, 4-29 y 4-30.

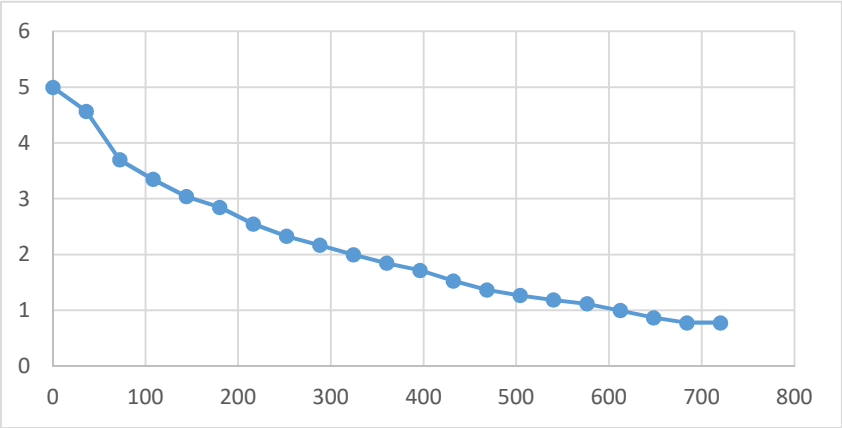
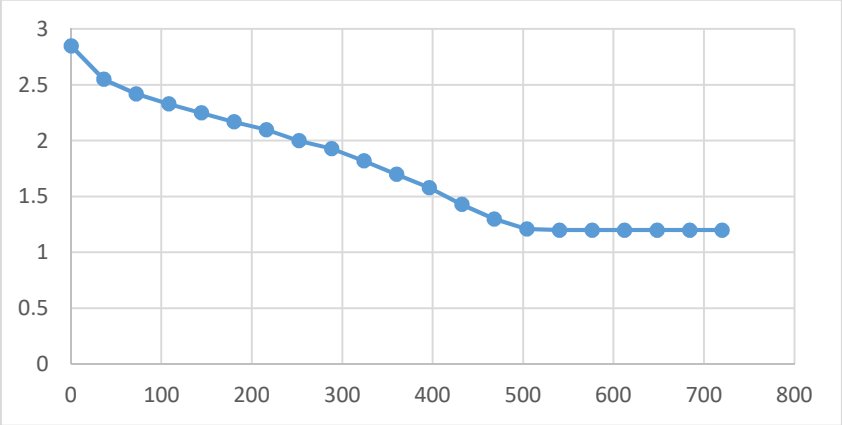


Ilustración 4-28 Gráfica para el sensor FC-28 con sustrato nuevo y agua conductiva 2027uS/cm.



.

Ilustración 4-29 Gráfica para el sensor capacitivo con sustrato nuevo y agua conductiva 2027uS/cm.

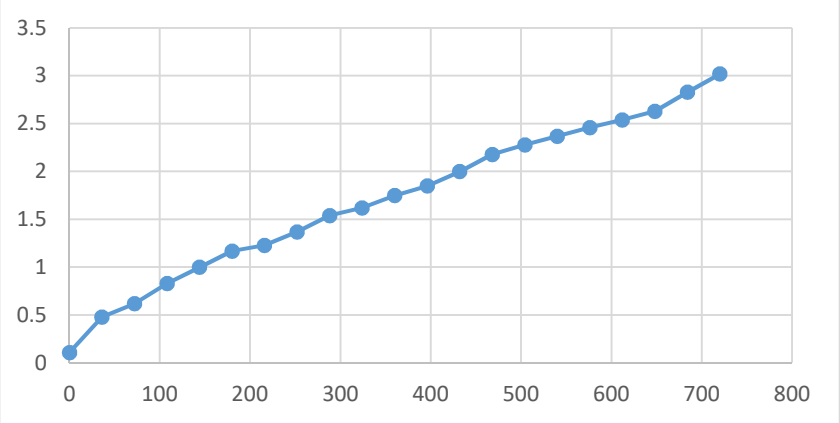


Ilustración 4-30 Gráfica para el sensor VH400 con sustrato nuevo y agua conductiva 2027uS/cm.

Tabla 4-3 Sustrato nuevo con agua conductiva 3008uS/cm.

Agua conductiva 3008uS/cm	Sensor Fc-28	Sensor conductivo	Sensor VH400
0	5	2.85	0.11
36	4.54	2.53	0.44
72	4.1	2.35	0.83
108	3.52	2.28	1.09
144	3.24	2.2	1.18
180	2.78	2.18	1.27
216	2.5	2.09	1.41
252	2.23	2	1.58
288	1.96	1.83	1.69
324	1.75	1.71	1.87
360	1.61	1.52	1.95
396	1.48	1.48	2.1
432	1.35	1.38	2.23
468	1.14	1.29	2.29
504	1.09	1.21	2.36
540	1.07	1.2	2.43
576	0.98	1.2	2.53
612	0.78	1.2	2.64
648	0.7	1.2	2.76
684	0.51	1.2	2.88
720	0.44	1.2	3

Las gráficas del comportamiento que presentan los sensores con agua conductiva 3008uS/cm y sustrato nuevo se muestran en las ilustraciones 4-31, 4-32 y 4-33.

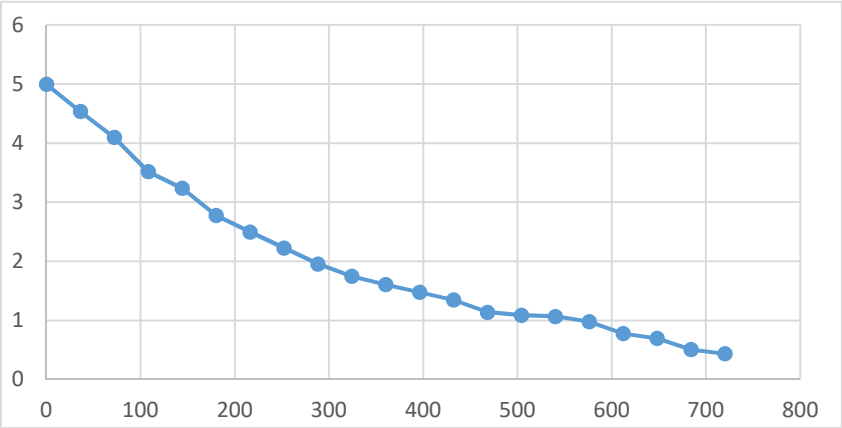


Ilustración 4-31 Gráfica para el sensor FC-28 con sustrato nuevo y agua conductiva 3008uS/cm.

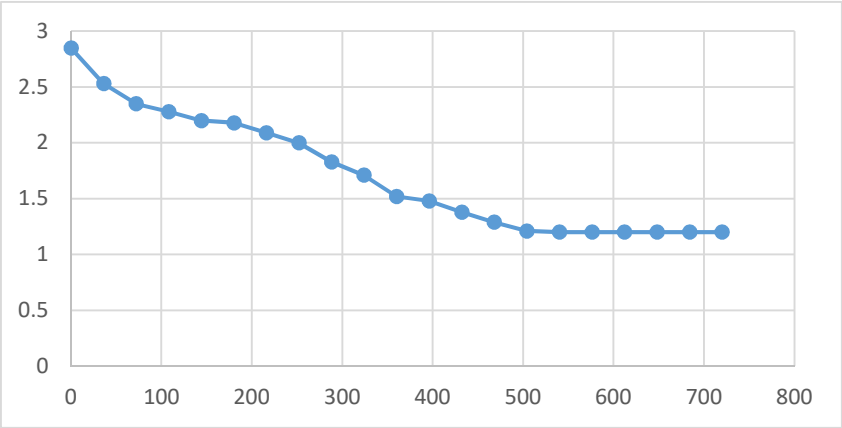


Ilustración 4-32 Gráfica para el sensor capacitivo con sustrato nuevo y agua conductiva 3008uS/cm.

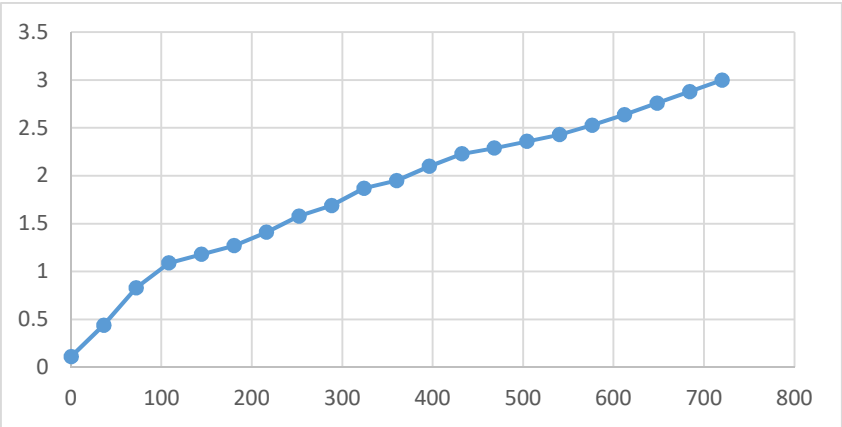


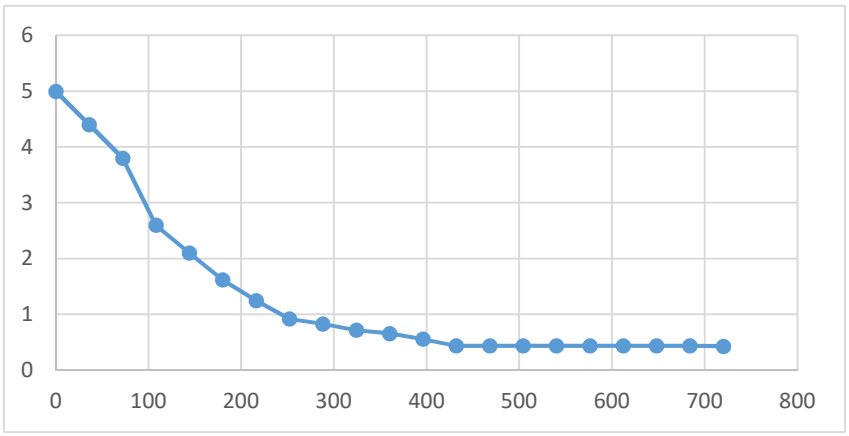
Ilustración 4-33 Gráfica para el sensor VH400 con sustrato nuevo y agua conductiva 3008uS/cm.

Los resultados obtenidos de las mediciones para el sustrato de un uso se muestran las tablas 4-4, 4-5 y 4-6.

Tabla 4-4 Sustrato de un uso con agua destilada.

Agua destilada	Sensor Fc-28	Sensor conductivo	Sensor VH400
0	5	2.85	0.13
36	4.4	2.68	0.42
72	3.8	2.33	0.83
108	2.6	2.18	1.15
144	2.1	2.08	1.38
180	1.62	1.92	1.53
216	1.25	1.84	1.63
252	0.92	1.76	1.77
288	0.83	1.69	1.86
324	0.72	1.56	2
360	0.66	1.47	2.13
396	0.56	1.38	2.21
432	0.44	1.26	2.3
468	0.44	1.26	2.39
504	0.44	1.2	2.53
540	0.44	1.2	2.69
576	0.44	1.2	2.78
612	0.44	1.2	2.87
648	0.44	1.2	2.91
684	0.44	1.2	3
720	0.44	1.2	3.04

Las gráficas del comportamiento que presentan los sensores con agua destilada y sustrato de un uso se muestran en las ilustraciones 4-34, 4-35 y 4-36.



· Ilustración 4-34 Gráfica para el sensor FC-28 con sustrato de un uso y agua destilada.

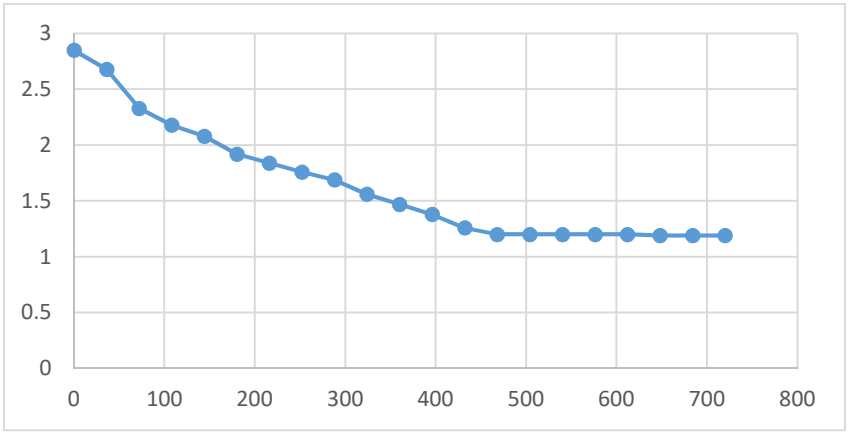


Ilustración 4-35 Gráfica para el sensor capacitivo con sustrato de un uso y agua destilada.

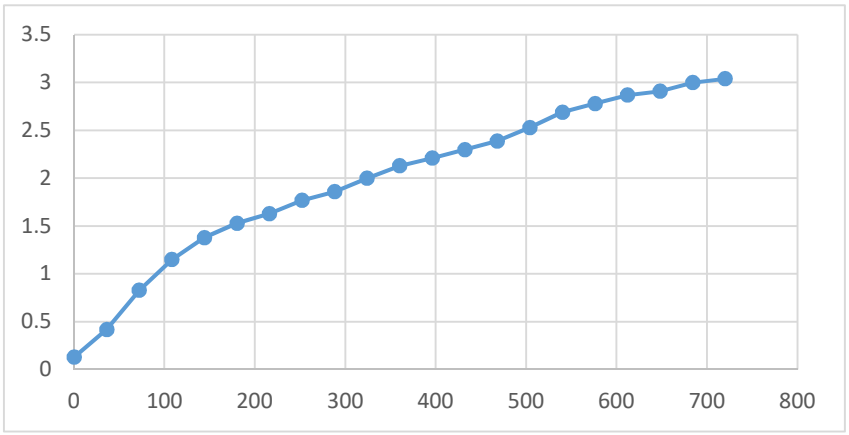


Ilustración 4-36 Gráfica para el sensor VH400 con sustrato de un uso y agua destilada.

Tabla 4-5 Sustrato de un uso con agua conductiva 2027uS/cm.

Agua conductiva 2027uS/cm	Sensor Fc-28	Sensor conductivo	Sensor VH400
0	5	2.85	0.16
36	4.6	2.46	0.65
72	2.87	2.32	1.01
108	2.1	2.22	1.12
144	1.45	2.14	1.33
180	1.08	2.04	1.47
216	0.95	1.95	1.63
252	0.87	1.85	1.72
288	0.76	1.76	1.81
324	0.64	1.68	1.93
360	0.58	1.57	2.03
396	0.47	1.42	2.12
432	0.45	1.36	2.23
468	0.44	1.26	2.34
504	0.44	1.2	2.42
540	0.44	1.2	2.63
576	0.44	1.2	2.74
612	0.44	1.2	2.86
648	0.44	1.2	2.98
684	0.44	1.2	3.02
720	0.44	1.2	3.04

Las gráficas del comportamiento que presentan los sensores con agua conductiva 2027uS/cm y sustrato de un uso se muestran en las ilustraciones 4-37, 4-38 y 4-39.

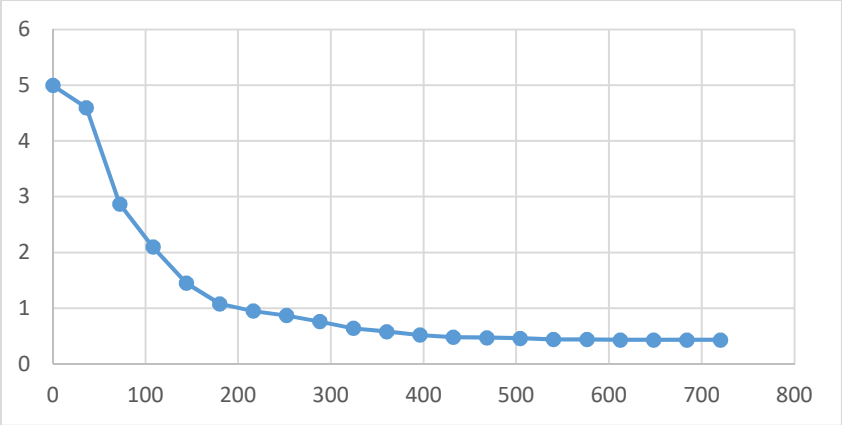


Ilustración 4-37 Gráfica para el sensor FC-28 con sustrato de un uso y agua conductiva 2027uS/cm.

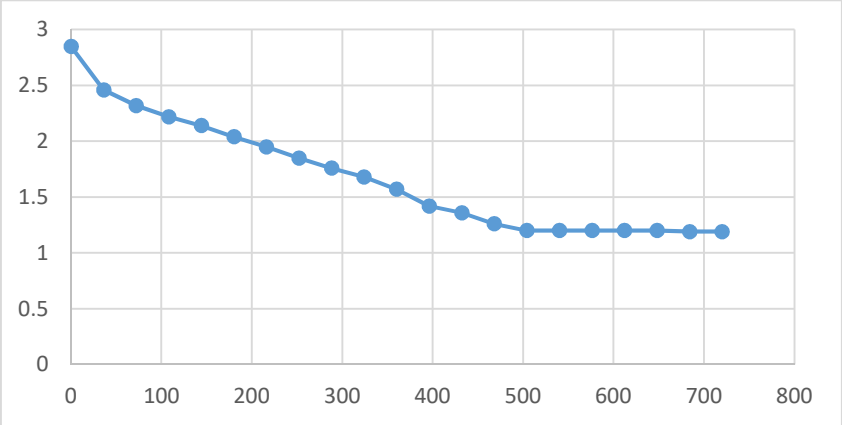


Ilustración 4-38 Gráfica para el sensor capacitivo con sustrato de un uso y agua conductiva 2027uS/cm.

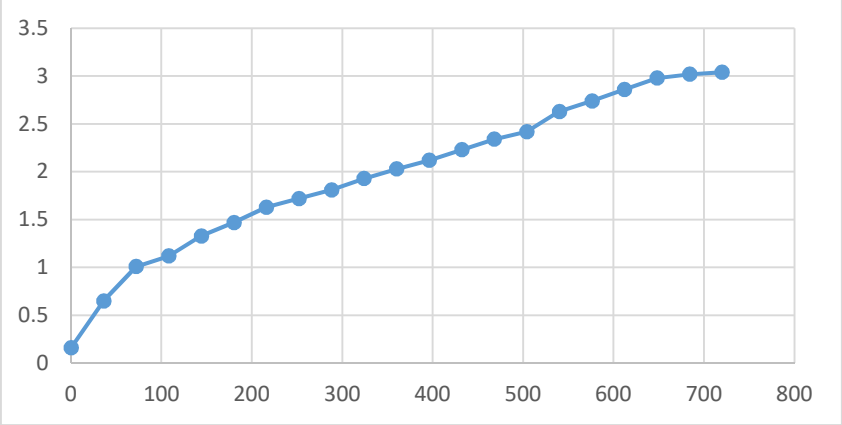


Ilustración 4-39 Gráfica para el sensor VH400 con sustrato de un uso y agua destilada 2027uS/cm.

Tabla 4-6 Sustrato de un uso con agua conductiva 3008uS/cm.

Agua conductiva 3008uS/cm	Sensor Fc-28	Sensor conductivo	Sensor VH400
0	5	2.85	0.16
36	4.2	2.48	0.63
72	2.78	2.31	0.88
108	1.42	2.23	1.13
144	0.93	2.11	1.36
180	0.82	2.01	1.47
216	0.73	1.93	1.6
252	0.62	1.85	1.73
288	0.49	1.7	1.83
324	0.44	1.61	1.91
360	0.44	1.5	2.03
396	0.44	1.41	2.14
432	0.44	1.26	2.23
468	0.44	1.2	2.31
504	0.44	1.2	2.44
540	0.44	1.2	2.56
576	0.44	1.2	2.66
612	0.44	1.2	2.74
648	0.44	1.2	2.85
684	0.44	1.21	2.97
720	0.44	1.2	3.04

Las gráficas del comportamiento que presentan los sensores con agua conductiva 3008uS/cm y sustrato de un uso se muestran en las ilustraciones 4-40, 4-41 y 4-42.

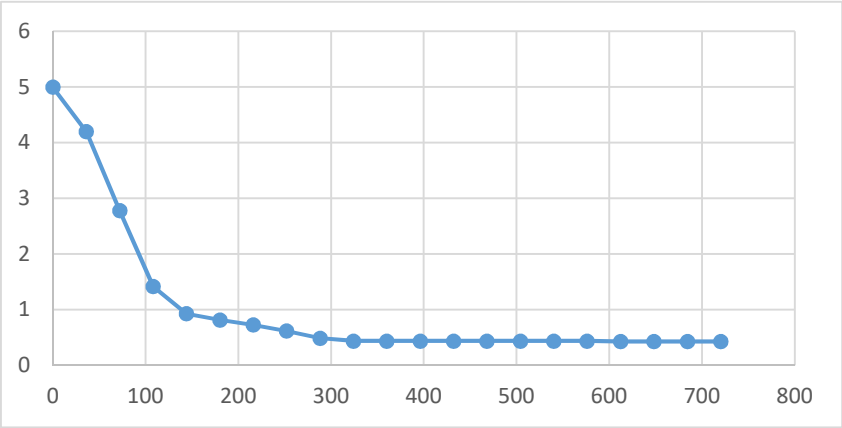


Ilustración 4-40 Gráfica para el sensor FC-28 con sustrato de un uso y agua conductiva 3008uS/cm.

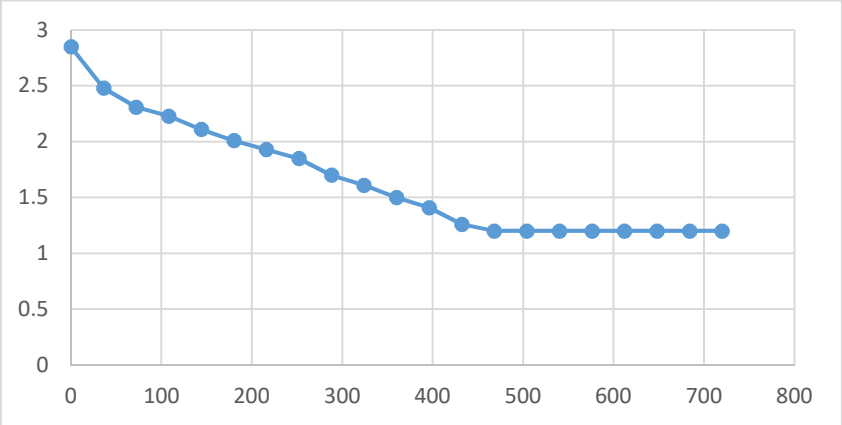


Ilustración 4-41 Gráfica para el sensor capacitivo con sustrato de un uso y agua conductiva 3008uS/cm.

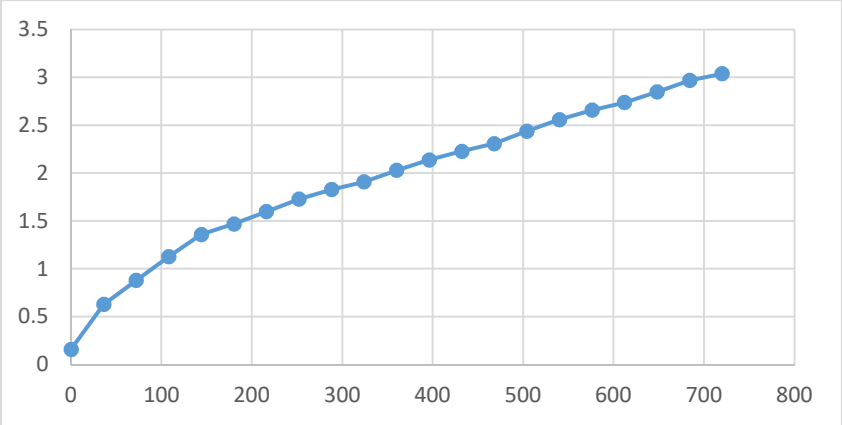


Ilustración 4-42 Gráfica para el sensor VH400 con sustrato de un uso y agua conductiva 3008uS/cm.

Los resultados obtenidos de las mediciones para el sustrato de desecho se muestran las tablas 4-7, 4-8 y 4-9.

Tabla 4-7Sustrato para desecho y agua destilada.

Agua destilada	Sensor Fc-28	Sensor conductivo	Sensor VH400
0	5	2.85	0.15
36	3.45	2.47	0.63
72	2	2.33	0.85
108	1.6	2.2	1.11
144	1.3	2.11	1.37
180	0.98	2.03	1.58
216	0.74	1.95	1.7
252	0.53	1.86	1.89
288	0.54	1.78	2.03
324	0.44	1.54	2.16
360	0.44	1.46	2.27
396	0.44	1.33	2.39
432	0.44	1.23	2.49
468	0.44	1.2	2.58
504	0.44	1.2	2.66
540	0.44	1.2	2.76
576	0.44	1.2	2.89
612	0.44	1.2	2.97
648	0.44	1.2	3.02
684	0.44	1.2	3.04
720	0.43	1.2	3.04

Las gráficas del comportamiento que presentan los sensores con agua destilada y sustrato para desecho se muestran en las ilustraciones 4-43, 4-4 y 4-45.

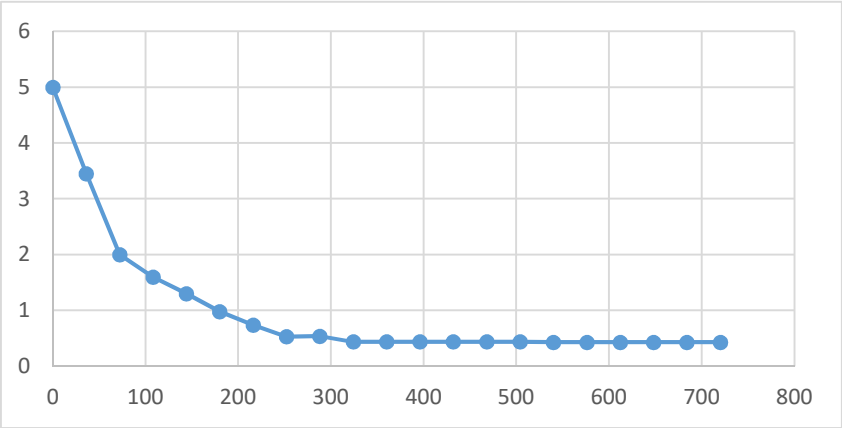


Ilustración 4-43 Gráfica para el sensor FC-28 con sustrato para desecho y agua destilada.

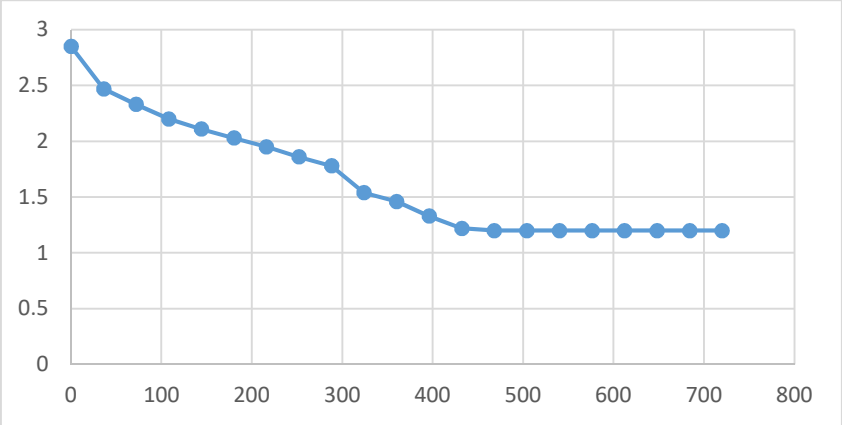


Ilustración 4-44 Gráfica para el sensor capacitivo con sustrato para desecho y agua destilada.

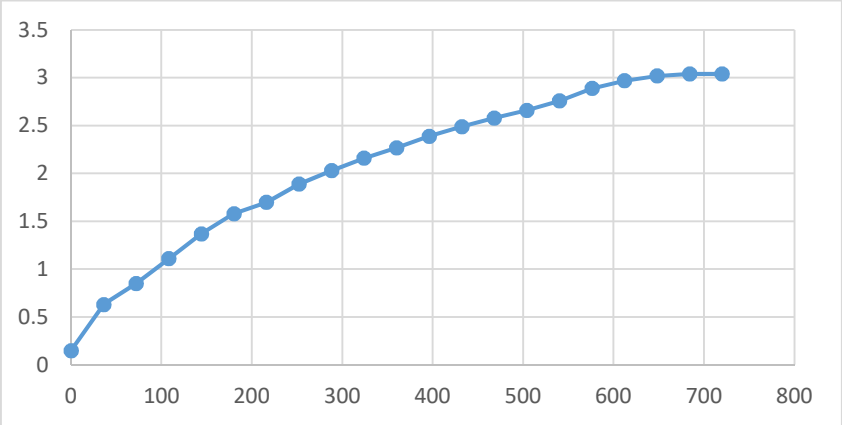


Ilustración 4-45 Gráfica para el sensor VH400 con sustrato para desecho y agua destilada.

Tabla 4-8 Sustrato para desecho con agua conductiva 2027uS/cm.

Agua conductiva 2027uS/cm	Sensor Fc-28	Sensor conductivo	Sensor VH400
0	5	2.85	0.14
36	3.45	2.5	0.43
72	2.54	2.42	0.88
108	1.7	2.31	1.16
144	1.53	2.24	1.34
180	1.45	2.11	1.42
216	1.05	1.96	1.53
252	0.84	1.84	1.65
288	0.71	1.71	1.74
324	0.53	1.57	1.89
360	0.44	1.43	1.99
396	0.44	1.31	2.14
432	0.44	1.25	2.29
468	0.44	1.2	2.46
504	0.44	1.2	2.58
540	0.44	1.2	2.66
576	0.44	1.2	2..73
612	0.44	1.2	2.82
648	0.44	1.2	2.94
684	0.44	1.2	3.02
720	0.44	1.2	3.04

Las gráficas del comportamiento que presentan los sensores con agua conductiva 2027uS/cm y sustrato para desecho se muestran en las ilustraciones 4-46, 4-47 y 4-48.

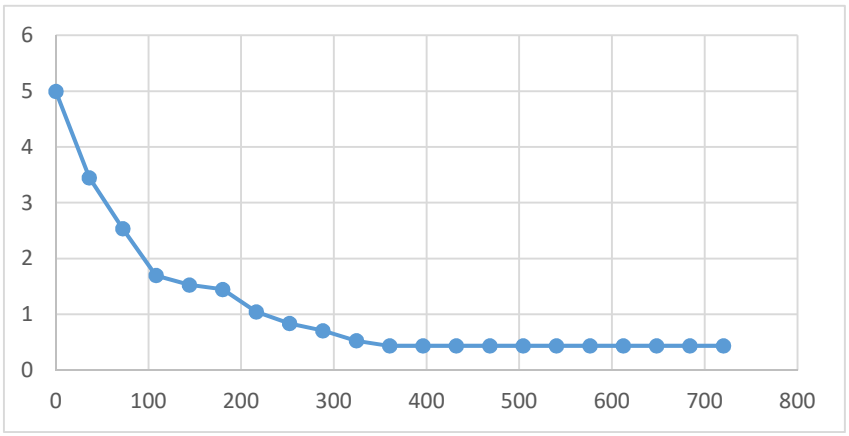


Ilustración 4-46 Gráfica para el sensor FC-28 con sustrato para desecho y agua conductiva 2027uS/cm.

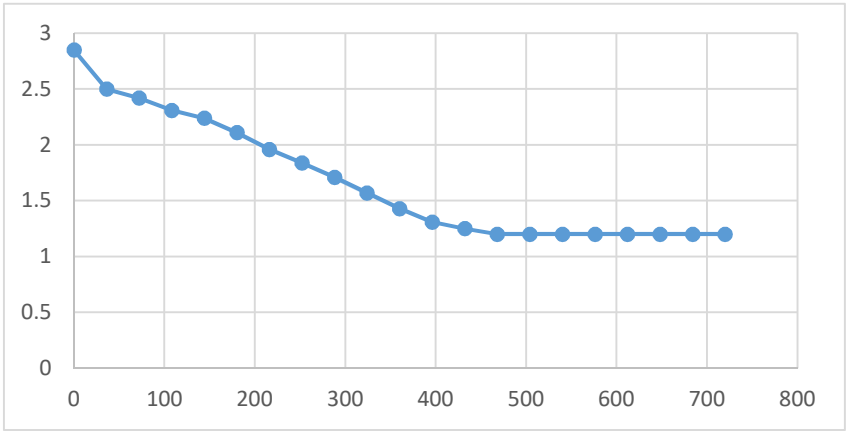


Ilustración 4-47 Gráfica para el sensor capacitivo con sustrato para desecho y agua conductiva 2027uS/cm.

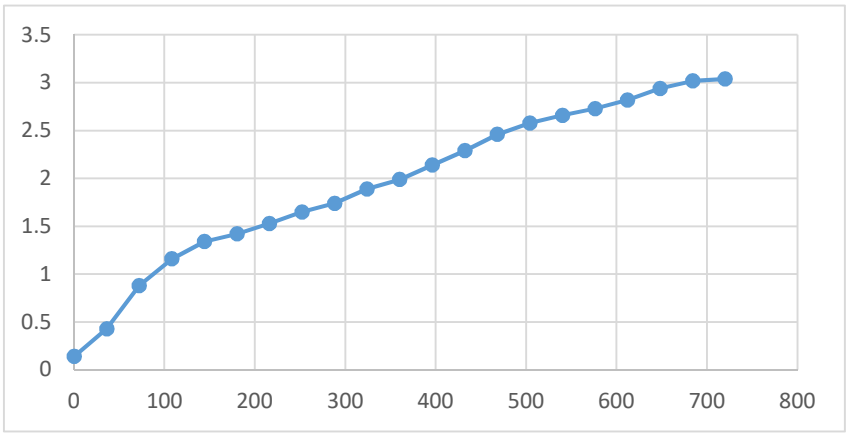


Ilustración 4-48 Gráfica para el sensor VH400 con sustrato para desecho y agua conductiva 2027uS/cm.

Tabla 4-9 Sustrato para desecho con agua conductiva 3008uS/cm.

Agua conductiva 3008uS/cm	Sensor Fc-28	Sensor conductivo	Sensor VH400
0	5	2.85	0.16
36	3.7	2.57	0.46
72	2	2.43	0.75
108	1.6	2.31	0.93
144	1.3	2.19	1.23
180	1.1	2	1.39
216	0.88	1.85	1.54
252	0.67	1.75	1.72
288	0.47	1.68	1.87
324	0.44	1.54	1.99
360	0.44	1.47	2.16
396	0.44	1.36	2.34
432	0.44	1.27	2.43
468	0.44	1.2	2.63
504	0.44	1.2	2.72
540	0.44	1.2	2.81
576	0.44	1.2	2.9
612	0.44	1.2	2.99
648	0.44	1.2	3.02
684	0.44	1.2	3.04
720	0.44	1.2	3.04

Las gráficas del comportamiento que presentan los sensores con agua conductiva 3008uS/cm y sustrato para desecho se muestran en las ilustraciones 4-49, 4-50 y 4-51.

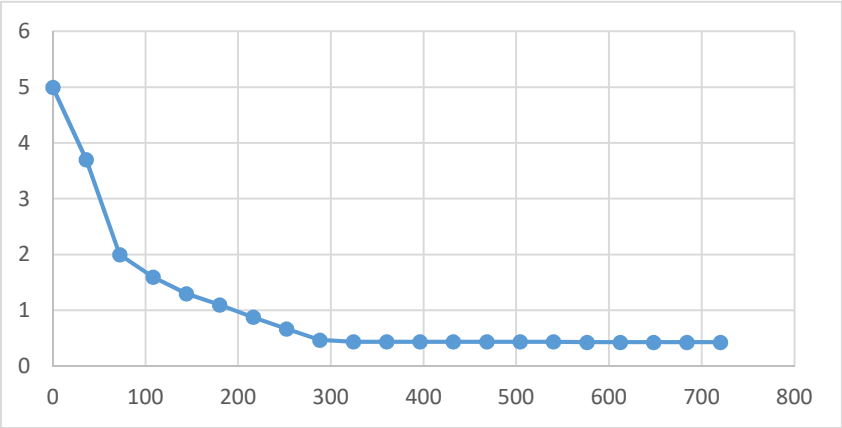


Ilustración 4-49 Gráfica para el sensor FC-28 con sustrato para desecho y agua conductiva 3008uS/cm.

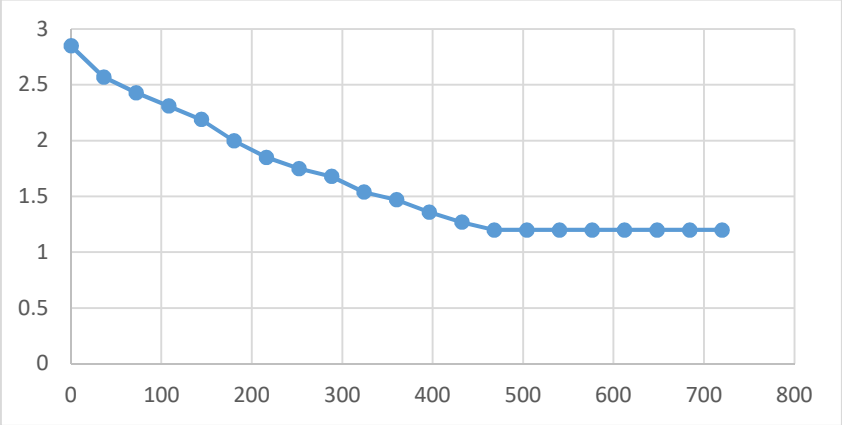


Ilustración 4-50 Gráfica para el sensor capacitivo con sustrato para desecho y agua conductiva 3008uS/cm.

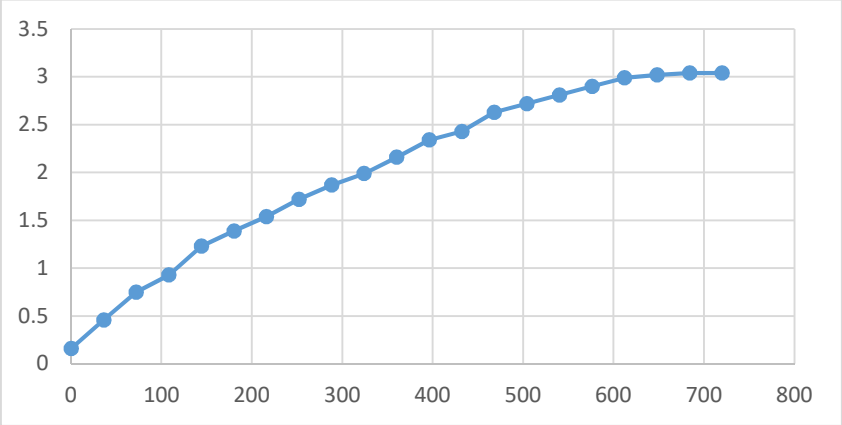


Ilustración 4-51 Gráfica para el sensor VH400 con sustrato para desecho y agua conductiva 3008uS/cm.

4.2.2 Conclusiones de las pruebas realizadas

Como se puede observar en las gráficas de comportamiento de los sensores, ninguno de los tres sensores tiene un comportamiento completamente lineal, sino que, tienen variaciones y curvas que no permiten realizar una ecuación lineal que describa completamente su

funcionamiento, por lo que se tienen que realizar aproximaciones para mantener el modelo matemático lo más sencillo posible, esto debido a que es muy complicado obtener un modelo matemático con estas características además de que los lenguajes de programación utilizados para programar los microcontroladores no cuentan con algunas funciones matemáticas necesarias para realizar este tipo de modelos matemáticos.

4.2.2.1 Observaciones para el sensor FC-28

Este sensor es el más barato de los 3 sensores que se probaron, como se puede observar en las gráficas de comportamiento para este sensor, es muy sensible a los cambios de conductividad en el agua, esto se hace muy evidente si se observan las gráficas de las pruebas realizadas con el sustrato nuevo, en las que tiene un comportamiento más o menos lineal, pero en cuanto se empieza a incrementar la conductividad del agua, pierde completamente el comportamiento lineal y empieza a presentar un comportamiento exponencial que no puede proporcionar la precisión adecuada para mantener las lecturas de humedad de suelo fiables, por esta razón no es posible utilizar este sensor para este proyecto, sin embargo podría utilizarse en sistemas en la que la conductividad del agua se mantenga muy baja.

4.2.2.2 Observaciones para el sensor capacitivo

Este sensor es más caro que el sensor FC-28, aunque se podría considerar de bajo costo respecto al sensor VH400. El principal problema de este sensor es el rango real de voltaje, ya que según las especificaciones técnicas del sensor, la señal de salida varía dentro del rango entre 3v a 0v, sin embargo, como se puede observar en las gráficas de comportamiento de este sensor, el rango real de voltaje es de 2.85v a 1.2v, lo cual limita bastante el rango de medición de este sensor, para descartar que fuera problema del sensor utilizado para realizar las pruebas, se probaron 3 sensores de este tipo, los resultados que se obtuvieron tuvieron unas ligeras variaciones, pero en ninguno de los tres casos se logró obtener una medición cercana a 0v, además de que el circuito eléctrico de este sensor se encuentra expuesto sin ningún tipo de protección, lo cual lo hace muy vulnerable a oxidación prematura o cortocircuitos debido a la acumulación de humedad, por lo que es necesario agregar una protección externa para el circuito eléctrico del sensor. Respecto a la respuesta del sensor en función de la conductividad del agua, se puede observar que mantiene cierta linealidad en las lecturas y no es tan vulnerable a la conductividad como el sensor FC-28, sin embargo, el

rango de voltaje no permite realizar mediciones de humedad de más del 70%, por lo que se puede decir que, utilizando este sensor, se puede garantizar la fiabilidad de las mediciones, aunque si se podría utilizar en aplicaciones donde la humedad se mantenga por debajo del 70% de humedad en el sustrato.

4.2.2.3 Observaciones para el sensor VH400

Este sensor es considerablemente más caro que las otras dos opciones analizadas, por lo que evidentemente es de mejor calidad, la especificación técnica de este sensor indica que el rango de voltaje para la señal de salida es de 0v a 3v, en la práctica, el rango de voltaje para este sensor es de 0.11v a 3.04v lo que es un rango muy similar al que especifica el fabricante, de igual forma que con el sensor capacitivo, para descartar que este rango fuera exclusivo del sensor utilizado, también se analizaron otros dos sensores del mismo modelo y en los tres sensores se mantuvo un rango de voltajes muy similar. Como se puede observar en las gráficas de comportamiento para este sensor, no es completamente lineal si no que tiene un comportamiento ligeramente logarítmico, sin embargo, es muy ligero este comportamiento, por lo que sí es posible hacer un modelo matemático lineal que se aproxime al comportamiento de este sensor, de las tres opciones analizadas, este sensor es el que mejor resultado dio en las pruebas realizadas.

4.3 Obtención del modelo matemático para los sensores de humedad de suelo

Como ya se mencionó anteriormente, el sensor FC-28, no se puede utilizar en este proyecto debido a que es muy susceptible a la conductividad del agua, podría utilizarse el sensor capacitivo, pero no es recomendable debido a que solo es efectivo para medir humedad menor al 70%, así que solo se obtendrá el modelo matemático para el sensor VH400 que es el que se utilizará para este proyecto.

El primer paso para obtener el modelo matemático es, sacar un promedio de todas las lecturas obtenidas de las pruebas a fin de obtener una sola tabla que relacione todas las lecturas, en la cual también se incluirá el porcentaje de humedad (Tabla 4-10).

Tabla 4-10 promedio obtenido para el sensor VH400.

Cantidad de agua (ml)	Promedio obtenido (v)	% de humedad
0	0.1366	0
36	0.5122	5
72	0.8133	10
108	1.0455	15
144	1.2511	20
180	1.3544	25
216	1.4966	30
252	1.6311	35
288	1.7411	40
324	1.8622	45
360	1.9822	50
396	2.1011	55
432	2.2144	60
468	2.3355	65
504	2.4377	70
540	2.5566	75
576	2.6711	80
612	2.7755	85
648	2.8677	90
684	2.9566	95
720	3.0277	100

Para ver el comportamiento del promedio obtenido, se debe graficar el porcentaje de humedad en función del promedio de las lecturas obtenidas (Ilustración 4-52).

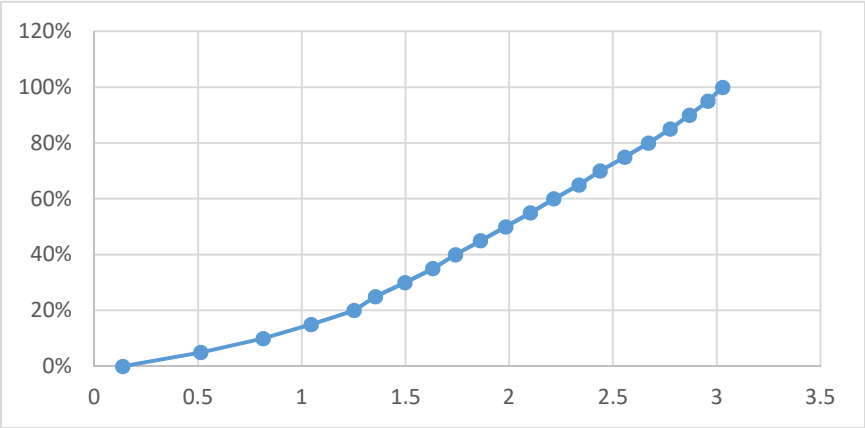


Ilustración 4-52 Porcentaje de humedad en función del promedio calculado.

Como se puede observar en la gráfica anterior, existen dos tendencias claras en el promedio de las lecturas proporcionadas por el sensor VH400, la primera se puede observar del 0% al 20% (Ilustración 4-53) y la segunda del 20% hasta el 100% (Ilustración 4-54) por lo que se pueden obtener 2 ecuaciones diferentes y utilizar una ecuación u otra según sea el rango al que pertenezca la lectura obtenida por el sensor. Utilizando Office Excel se realizan las gráficas anteriormente mencionadas y se les añade una línea de tendencia con su respectiva ecuación de la recta, que son las que se utilizaran como modelo matemático.

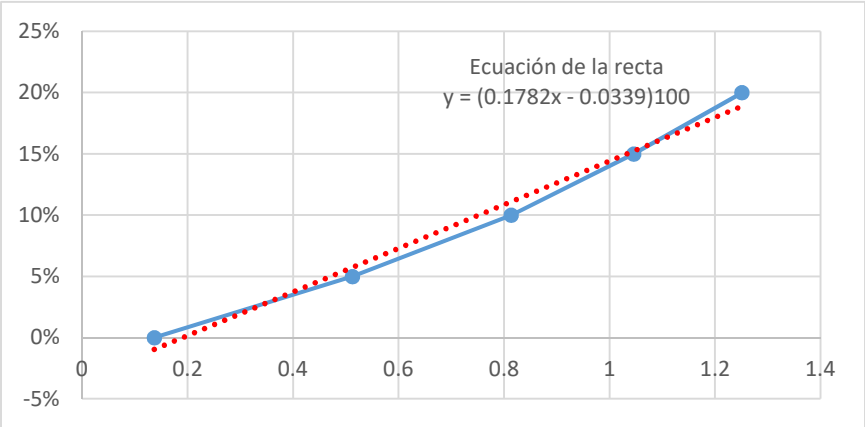


Ilustración 4-53 Gráfica para rango de 0% a 20%.

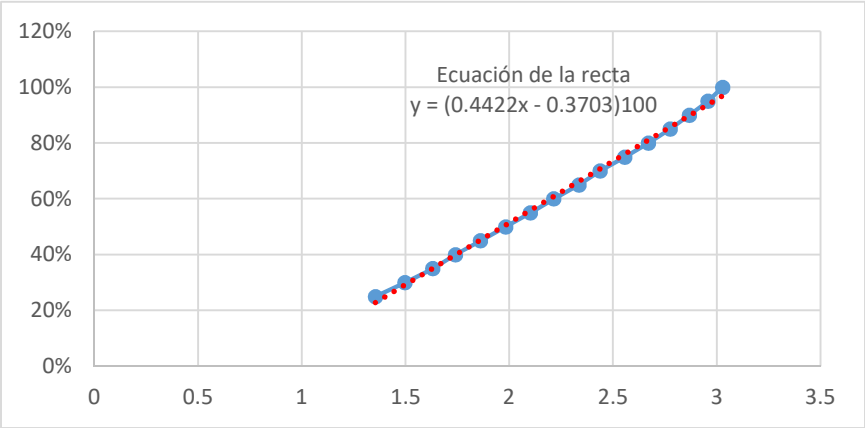


Ilustración 4-54 Gráfica para el rango de 20% a 100%.

La ecuación que describe el comportamiento del sensor de 0% al 20% es:

$$\%\boldsymbol{de\ humedad} = (\mathbf{0.1782}(\boldsymbol{lectura\ obtenida}) - \mathbf{0.0339})\mathbf{100} \qquad \text{Ec 4.1}$$

La ecuación que describe el comportamiento del sensor de 20% al 100% es:

$$\%\boldsymbol{de\ humedad} = (\mathbf{0.4422}(\boldsymbol{lectura\ obtenida}) - \mathbf{0.3703})\mathbf{100} \qquad \text{Ec 4.2}$$

Para poder ver con que precisión pueden aproximarse las dos ecuaciones que mostraron anteriormente al comportamiento del sensor, se realizará un cálculo aleatorio, por ejemplo, sustituyendo lectura de 0.8133v, que según la tabla 4-10 corresponde al 10% de humedad, en la ecuación 4.1, se obtiene el resultado de 11.10%, el cual es muy similar al original de 10%, y si se sustituye el valor 2.5566, que según la tabla 4-10 corresponde al 75% de humedad, en la ecuación 4.2,se obtiene el resultado de 76.02%, el cual es muy similar al original de 75%.

4.4 Comunicación entre etapa de adquisición y módulo maestro

Esta parte del proyecto es crucial para el buen funcionamiento del mismo ya que se debe de asegurar que la etapa de adquisición y el módulo maestro tenga una comunicación ininterrumpida para que las lecturas presentadas en la interfaz gráfica sean las mismas que las presentes dentro del invernadero en ese momento, además de que debe de soportar tanto ruido natural como ruido artificial y mantener la información sin alteraciones a lo largo del recorrido.

4.4.1 Versión 1

Para la primera versión de este proyecto se utilizó la configuración mostrada en el diagrama de bloques mostradas en la ilustración 4-55.

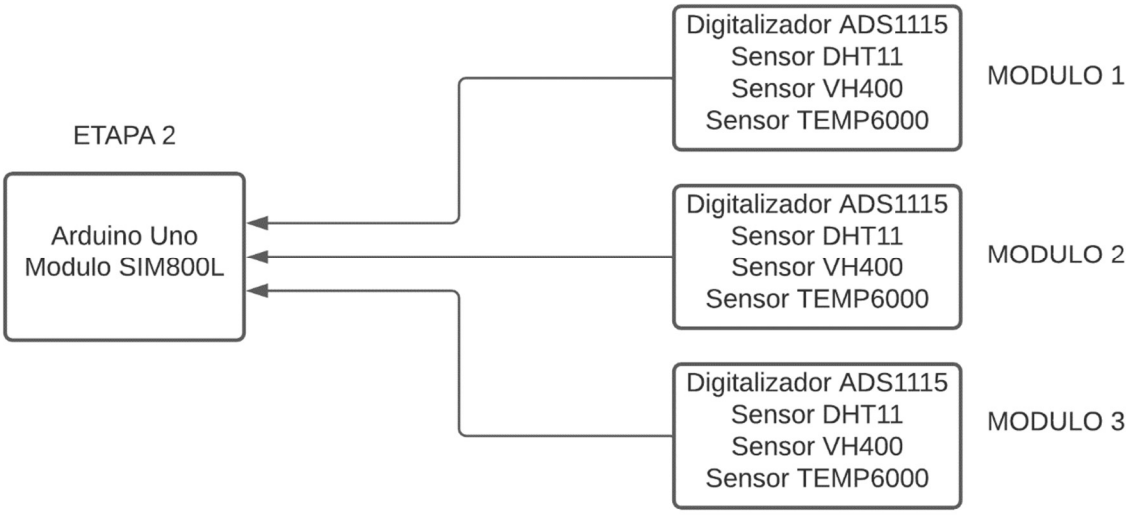


Ilustración 4-55 Diagrama de bloques para la versión 1.

Para esta versión solo se utilizó un Arduino Uno, mismo que controla todos los sensores de los tres módulos además de ser el encargado de enviar los datos a la interfaz gráfica y enviar los mensajes GSM. Esta versión está muy limitada en los sensores que puede utilizar ya que para los sensores BH1750 y BME280, solo se pueden conectar dos sensores simultáneamente con el mismo microcontrolador utilizando el protocolo I2C, por lo que para esta versión no es posible hacer uso de estos sensores y, como la mayoría de los sensores utilizados en esta versión son analógicos, se hace uso del convertidor analógico digital ADS1115 que también utiliza el protocolo I2C, pero para este dispositivo si se pueden utilizar tres dispositivos simultáneamente con el mismo microcontrolador.

En conclusión, esta versión está muy limitada en cuanto a los sensores que se pueden utilizar, pero el mayor problema que presenta, es la distancia máxima de comunicación entre el módulo maestro y los digitalizadores ADS1115 que componen a la etapa de adquisición, ya que al utilizar el protocolo I2C, la distancia de transmisión máxima que se pudo lograr fue de 1.5m aproximadamente y, hay que recordar que para el módulo 1 se ocupa una distancia de 25m, para el módulo 2 se necesita una distancia de 35m y para el módulo 3 se necesita una distancia de 50m, por lo que con esta versión no es posible cumplir con los objetivos de este proyecto.

4.4.2 Versión 2

En esta versión se buscó darle solución al principal problema de la versión 1, la cual es la distancia de comunicación entre la etapa de adquisición y el módulo maestro. El principal impedimento por el cual no se pueden alcanzar distancias mayores a los 2m, es debido a que se hace uso del protocolo de comunicación I2C, la solución más rápida, es utilizar otro protocolo de comunicación, en esta versión se utilizó el protocolo RS232, como se mencionó en el capítulo 2, este protocolo utiliza una comunicación asíncrona, por lo que es necesario utilizar un microcontrolador que pueda convertir la información del protocolo I2C al protocolo RS232, este microcontrolador se decidió que fuera el Atmega328p, debido a que se puede utilizar el IDE de Arduino para programarlo y es compatible con todas las bibliotecas de Arduino, sin embargo, este microcontrolador no tiene soporte para el protocolo RS232 por lo tanto es necesario hacer uso de un dispositivo adicional que convierta los datos de la UART del Atmega328p al protocolo RS232, este dispositivo es el circuito integrado MAX232. Como en esta versión cada módulo tiene su propio microcontrolador, es posible utilizar una mayor variedad de sensores como el BH1750 y el BME280, además, como el microcontrolador utilizado ya cuenta con ADC ya no es necesario utilizar el digitalizador ADS1115.

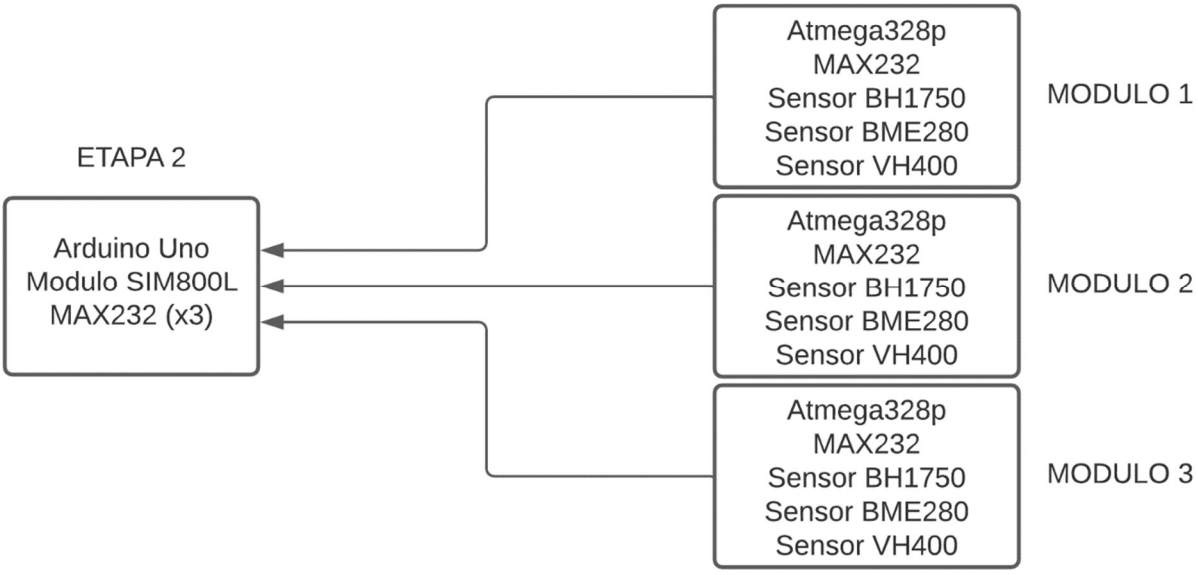


Ilustración 4-56 Diagrama de bloques para la versión 2.

En esta versión surgieron dos problemas, el primero es tarjeta de desarrollo Arduino Uno que se utilizó para el módulo maestro, esta solo dispone de una UART y para utilizar la

comunicación RS232 con tres módulos diferentes se necesita de tres UART además de una UART adicional para la comunicación con la interfaz gráfica, por lo que, al no disponer de estas, es necesario crearlas virtualmente con ayuda de la biblioteca *software serial,* el problema radica en que aunque sean creadas de forma virtual, se necesitan dos pines físicos del microcontrolador, una terminal para el TX y otra para el RX, esto provoca que se necesiten seis terminales del microcontrolador para la comunicaciones con los tres módulos de la etapa de adquisición, dos terminales para la comunicación con la interfaz gráfica y dos terminales más para la comunicación con el módulo SIM800L, esto da un total de 10 terminales digitales, el microcontrolador Atmega328p solo dispone de catorce terminales digitales, aunque aún sobran cuatro terminales esto puede limitar futuras funciones debido al número tan reducido de terminales.

El segundo problema que presenta esta versión, es la inestabilidad de la comunicación entre la etapa de adquisición y el módulo maestro, si bien con el uso del protocolo RS232 se pudo logar la comunicación entre estas dos etapas, esta era muy inestable y vulnerable al ruido natural y artificial, por lo tanto, este diseño no puede garantizar la comunicación entre estas dos etapas.

En conclusión, con esta versión se logró solucionar el problema de la limitación de los sensores que se podían utilizar en la versión 1 y también soluciono parcialmente el problema con la comunicación, aunque aún no es posible cumplir los objetivos propuestos para este proyecto.

4.4.3 Versión 3

Esta versión se centró principalmente en solucionar el problema aun existente de la comunicación entre la etapa de adquisición y el módulo maestro, para esto se analizaron algunos de los protocolos de comunicación más utilizados en la industria, dentro de los cuales sobresale el protocolo RS485 por su gran capacidad de soportar comunicaciones a largas distancias, además de que soporta muy bien los ambientes con ruido artificial y natural, por estas mismas características, este protocolo es ampliamente utilizado actualmente.

Para solucionar el problema de la versión 2 de la cantidad limitada de terminales digitales en el módulo maestro se decidió cambiar la tarjeta Arduino Uno por una tarjeta Arduino Mega, la tarjeta de desarrollo Arduino Mega, cuenta con 4 UART físicas por lo que ya no es

necesario crearlas virtualmente, además de que dispone de una cantidad muy superior de terminales digitales.

Para utilizar el protocolo RS485 es necesario convertir la información de la UART de los microcontroladores Atmega328p de la etapa de adquisición, ya que estos microcontroladores no ofrecen soporte para este protocolo, esta conversión se puede realizar utilizando el circuito integrado MAX485, con este circuito integrado los 3 módulos que componen la etapa de adquisición, pueden enviar datos utilizando el protocolo RS485, pero el Arduino Mega del módulo maestro tampoco tiene soporte para el protocolo RS485, por lo que ahora es necesario convertir los datos en formato RS485, en un formato que pueda ser interpretado por la UART del Arduino Mega, esta conversión también se puede realizar con el circuito integrado MAX485.

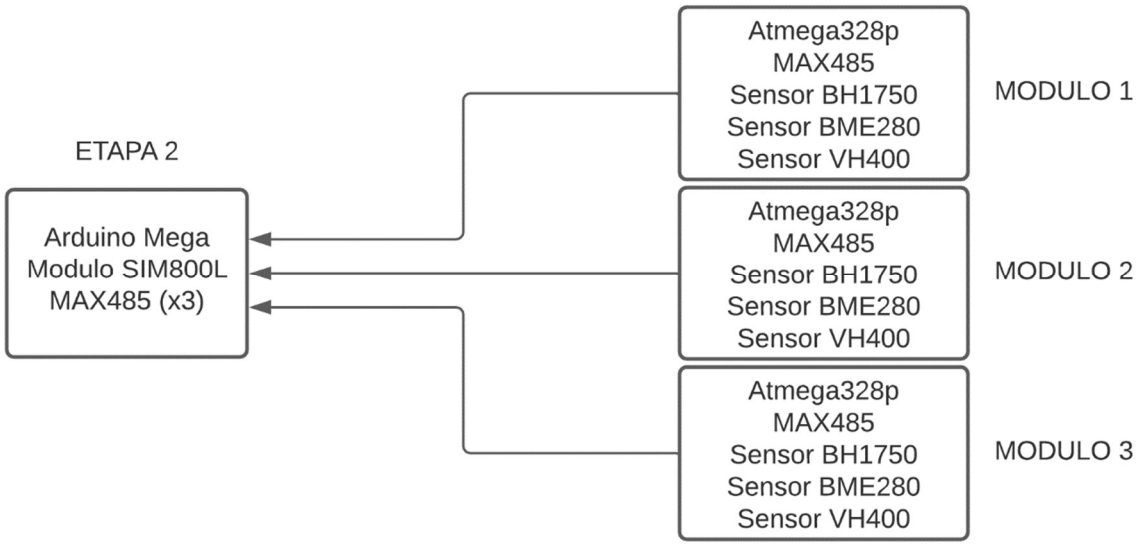


Ilustración 4-57 Diagrama de bloques para la versión 3.

Con esta versión fue posible establecer una comunicación estable entre la etapa de adquisición y el módulo maestro, fue posible alcanzar las distancias establecidas para cubrir las zonas del invernadero, además de tener la posibilidad de utilizar una mayor variedad de sensores, no solo los vistos en el capítulo 2, si no cualquier sensor que sea compatible con Arduino.

4.5 Pruebas de comunicación entre la etapa de adquisición y el módulo maestro

Como de las versiones anteriormente mencionadas solamente la versión 3 cumple con los requisitos necesario para mantener un óptimo funcionamiento del sistema, en todas las pruebas se utilizó esta versión.

El principal objetivo es poder enviar datos desde un Arduino Uno y recibirlo en un Arduino Mega utilizando el protocolo RS485.

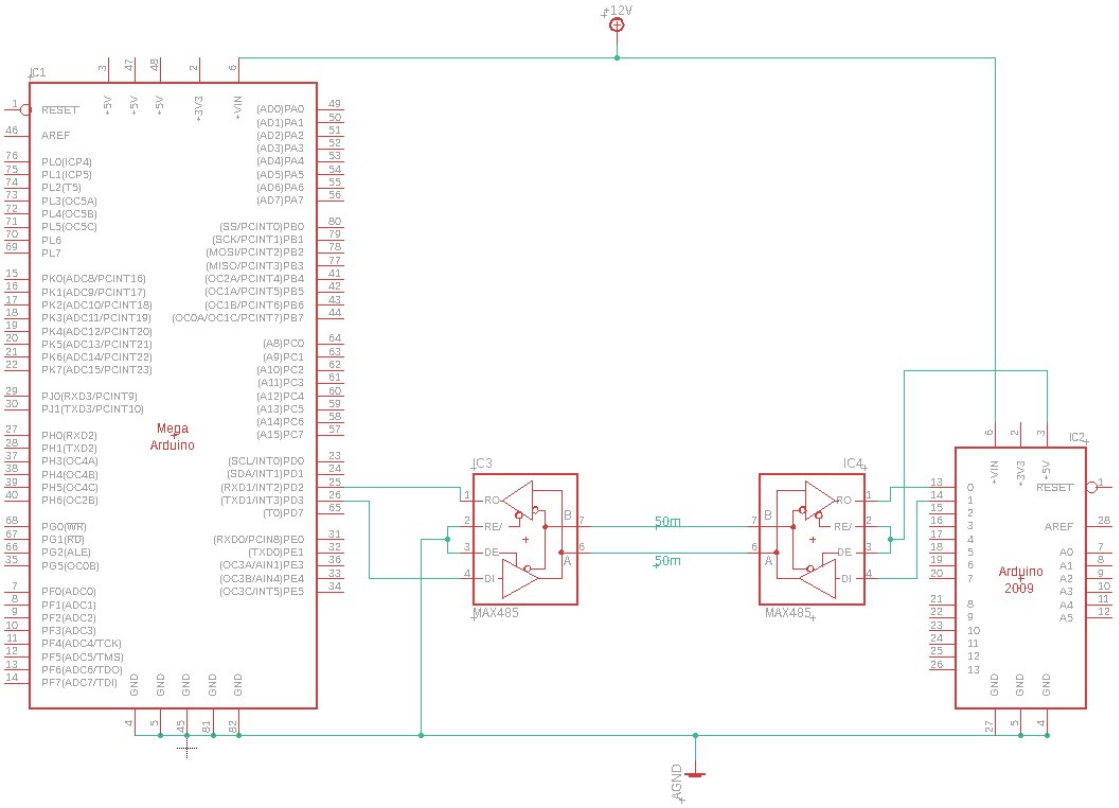


Ilustración 4-58 Circuito para pruebas de comunicación.

Para saber si el circuito propuesto en la ilustración 4-58 es útil para enviar datos a largas distancias, se utilizó el cable más largo utilizado en este proyecto (50m), para alcanzar esta distancia se utilizó cable de cobre UTP categoría 5, este tipo de cable es mayormente utilizado para redes de internet, por lo que está diseñado para trabajar con comunicaciones digitales.

Lo primero es enviar un dato cualquiera, esto solo es para verificar que realmente se pueda establecer una comunicación entre las dos tarjetas de desarrollo.

```
void setup() {
  Serial.begin(115200);

}

void loop() {
  Serial.println("Dato #");
  delay(200);

}
```

Ilustración 4-59 Código para la prueba 1 de comunicación con el Arduino Uno.

Para esta prueba el Arduino Uno tiene que enviar al Arduino Mega la cadena de texto “Dato #”, la razón por la cual se envía una cadena de texto y no un dato numérico, es porque es más fácil enviar varios datos en una cadena con formato de texto, mientras que para enviar varios datos en formato numérico se deben enviar uno a la vez, lo cual hace más complejo su envío y recepción. En la ilustración 4-59 se puede observar que la cadena de texto tiene un carácter especial (#), este carácter se utiliza como marcador para indicarle al Arduino Mega donde termina la cadena de texto.

```
void setup() {
  Serial.begin(115200);
  Serial1.begin(115200)

}

void loop() {
  while (Serial1.available()){
    String datoRecibido = readStringUntil('#');
  }
  Serial.println(datoRecibido);
  delay(100);
}
```

Ilustración 4-60 Código para la prueba 1 con Arduino Mega.

la única función del Arduino Mega para esta prueba es recibir por el puerto serie 1 el dato que le ha enviado el Arduino Uno y posteriormente imprimirlo utilizando el monitor serie del IDE de Arduino.

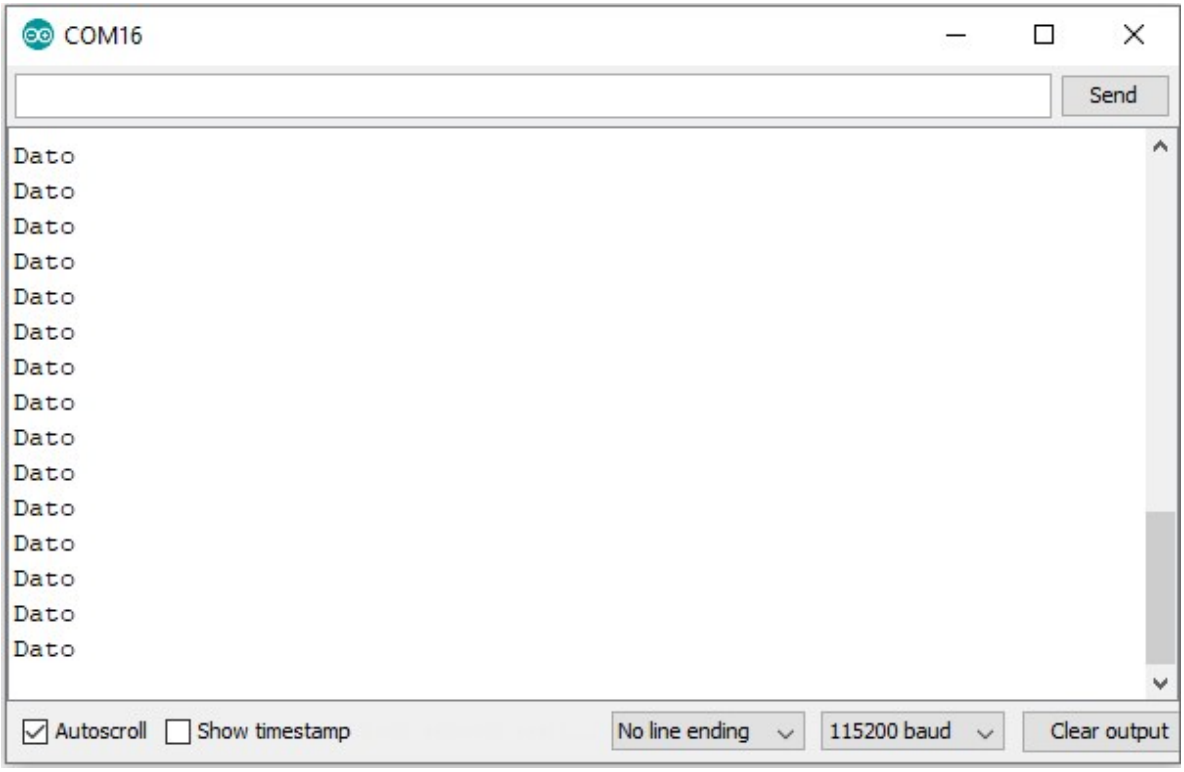


Ilustración 4-61 Resultado de la prueba 1 de comunicación.

Como se puede observar en la ilustración 4-61, si se pudo establecer la comunicación utilizando el protocolo RS485, lo cual permite cubrir las zonas más alejadas del invernadero. Una peculiaridad del resultado obtenido, es que, como se puede observar, el carácter "#" ya no aparece, esto es debido a que solo se utiliza como marcador para que el Arduino Mega lea los datos hasta que aparezca ese carácter, dejándolo fuera de la cadena de datos.

La siguiente prueba es, realizar la misma prueba anterior, pero ahora utilizando las tres medidas de cable y simulando el envío de lecturas de los sensores.

```
 int i;
 float cad [5];
 char buff[15];
 String valueString = "";

void setup() {
   Serial.begin(115200);
}

void loop() {
 float lux = 100;
  int HSuelo = analogRead(A0);
  int EXP1 = analogRead(A1);
  int EXP2 = analogRead(A2);
  float Hum = 100;
  float Tem = 100;


     cad [0] = HSuelo;*0.004887585;
     cad [1] = lux;
     cad [2] = EXP1*0.004887585;
     cad [3] = EXP2*0.004887585;
     cad [4] = Tem;
     cad [5] = Hum;
  for (i = 0; i <= 1; i++) {
    float val = cad [i];
    dtostrf(val, 4, 2, buff);
    valueString += buff;
    if (i<=4){
      valueString += " ";
    }else{
        valueString += " #";
    }
  }
  Serial.println(valueString);
  valueString = "";
  delay(100);
}
```

Ilustración 4-62 Código para la prueba 2 con Arduino Uno.

En esta prueba se utiliza un arreglo de datos en formato numérico que posteriormente se convierten a formato de texto y cada dato se separa por un espacio, al final de la cadena de texto se agrega el caracter “#” que servirá para indicarle al Arduino Mega en donde termina la cadena de texto.

```
void setup() {
  Serial.begin(115200);
  Serial1.begin(115200);
  Serial2.begin(115200);
  Serial3.begin(115200);

}

void loop() {
      while (Serial1.available()){
          String str = Serial1.readStringUntil('#');
      }
      while (Serial2.available()){
          String str1 = Serial2.readStringUntil('#');
      }
      String resultado =  str += str1;
      while (Serial3.available()){
          String str2 = Serial3.readStringUntil('#');
      }
      resultado += str2;
      Serial.println(resultado);
      resultado="";
      delay(100);

}
```

Ilustración 4-63 Código para la prueba 2 con Arduino Mega.

En esta prueba, el Arduino Mega recibe los datos de los tres Arduino Uno que simulan a los tres módulos del sistema, posteriormente debe unir todos estos datos en una sola cadena de texto y enviarlas por el puerto serie 0, para visualizar los datos se puede utilizar el monitor serie del Arduino IDE.

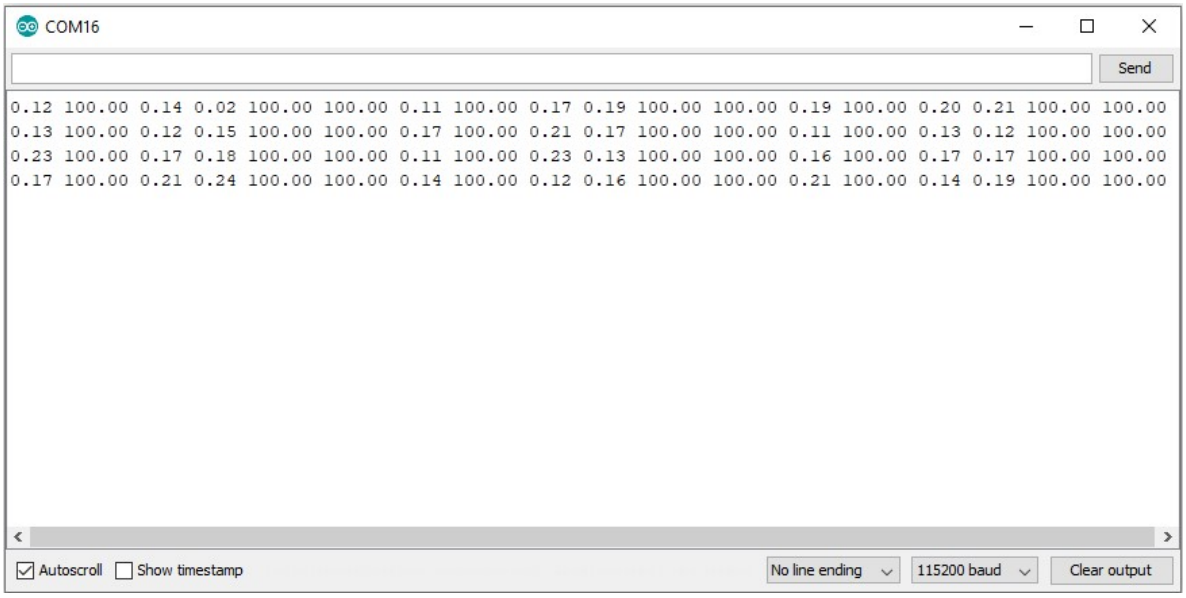


Ilustración 4-64 Resultado de la prueba 2 de comunicación.

Como se puede observar en la ilustración 4-64, se pudo lograr el objetivo de establecer una comunicación estable entre la etapa de adquisición y el módulo maestro a la distancia necesaria para cubrir las 3 zonas del invernadero, mediante el uso del protocolo RS485, con esta prueba se dan por concluidas las pruebas de comunicación.

4.6 Pruebas con el módulo SIM800L

Este módulo es el principal encargado del envió de alertas vía GSM, aunque se necesita de un microcontrolador que le indique las funciones que tiene que realizar, la prueba que se realizó con este módulo es muy sencilla, pero se debe verificar que el módulo cumple con el objetivo establecido de enviar las alertas justo en el momento en que sea necesario.

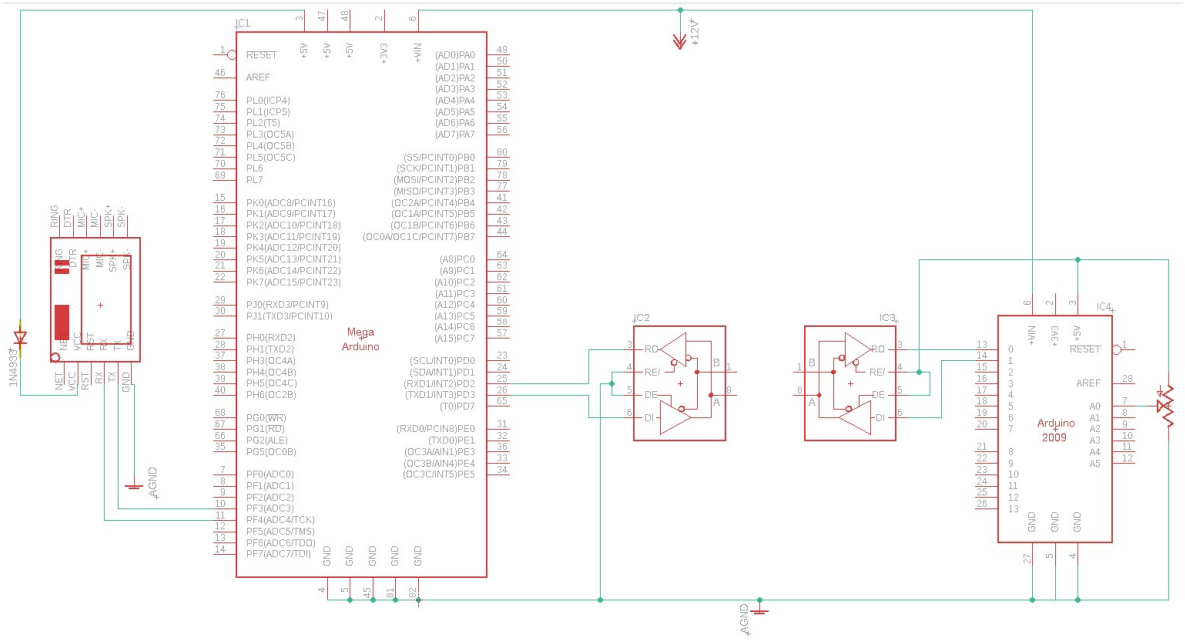


Ilustración 4-65 Circuito para la prueba con módulo SIM800L.

Para realizar esta prueba, se utiliza el circuito de la ilustración 4-65, el Arduino Uno tomará las lecturas de voltaje un potenciómetro las cuales enviará al Arduino Mega y este evaluará dichas lecturas, para enviar una alerta GSM cada que el valor de voltaje del potenciómetro este fuera de un rango que se establecerá en el código.

```
void setup(){
  Serial.begin(115200);

}
void loop(){
  char buff[15];
  float voltaje=analogRead(A0)*0.004887585;
  dtostrf(voltaje, 4, 2, buff);
  String dato = buff;
  dato = dato += "#";
  Serial.println(dato);
  delay(500);

}
```

Ilustración 4-66 Código de Arduino Uno para prueba con el módulo SIM800L.

En la ilustración 4-66 se muestra el código para el Arduino Uno que se utilizó para esta prueba, la función del código consta en tomar muestras de voltaje de un potenciómetro para posteriormente enviarlas hacia el Arduino Mega, como las lecturas obtenidas están en formato numérico, antes de enviarlas, se convierten a formato de texto para que sea más sencillo el proceso de recepción y manipulación de los datos para el Arduino Mega.

```
#include <Sim800l.h>
#include <SoftwareSerial.h>
Sim800l Sim800l;

void setup(){
  Serial1.begin(115200);
  Sim800l.begin();
}

void loop(){
  String recibido="";
  while (Serial1.available()){
    recibido = Serial1.readStringUntil('#');
  }
  float dato = recibido.toFloat();

  if(dato<=1 || dato>=4){
    Sim800l.sendSms("5618747131","PRUEBA EXITOSA");
  }
  delay(5000);
}
```

Ilustración 4-67 Código de Arduino Mega para prueba con el módulo SIM800L.

Una vez que el Arduino Uno haya enviado los datos, el Arduino Mega debe recibirlos y posteriormente volver a convertirlos en formato numérico para poder evaluarlos en un rango

establecido, si la lectura se encuentra dentro del rango establecido, no se enviara ninguna alerta, pero si la lectura se encuentra fuera del rango, se enviara una alerta vía GSM notificando el problema. Variando el potenciómetro conectado al Arduino Uno, se varía el valor de la lectura tomada, si esta lectura se encuentra dentro del rango entre 1v y 4v no se enviará ningún mensaje, pero si la lectura se encuentra fuera de este rango, se enviará el mensaje anteriormente mencionado, en el caso de esta prueba, el menaje que se enviara es: “PRUEBA EXITOSA”,

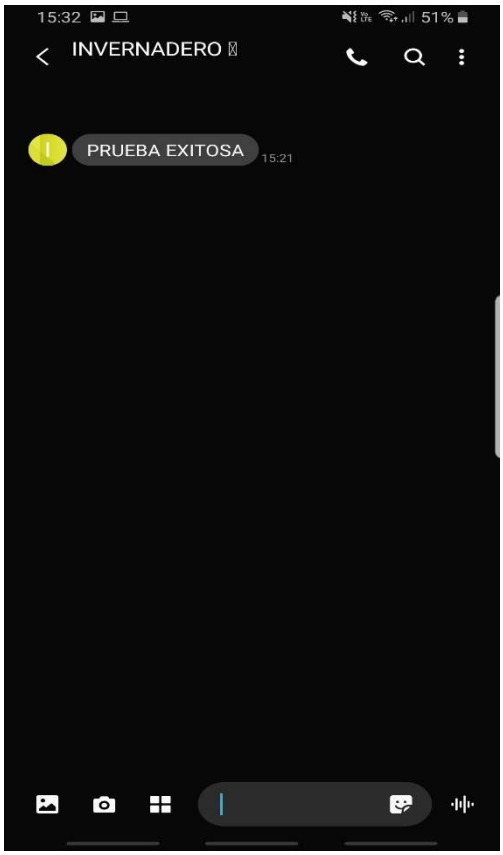


Ilustración 4-68 Resultado de la prueba con el módulo SIM800L.

Como se observa en la ilustración 4-68, la prueba anteriormente descrita se realizó exitosamente, aunque, es importante mencionar que el módulo SIM800L utiliza redes 2G y, es muy probable que en un futuro cercano deje de funcionar este tipo de tecnología debido a que actualmente se utilizan las redes 4G LTE y 5G, por lo que las redes 2G se puede decir que están obsoletas.

4.7 Pruebas para la base de datos

Para la base de datos, hay dos principales inconvenientes, el primero es referido al almacenamiento que se necesita para almacenar todos los datos, ya que, si bien una computadora actual dispone de bastante espacio de almacenamiento, el ocupar mucho de este este espacio disponible en una base de datos no es lo más óptimo, y el segundo inconveniente se refiere a la gran cantidad de datos que la interfaz gráfica tiene que procesar y esto puede provocar que el sistema se vuelva más lento o inestable incluso puede dejar de funcionar. Para ver hasta qué punto es relevante el primer inconveniente, lo primero es saber cuánto espacio en memoria ocupa la base datos almacenando cada segundo, que es la frecuencia de muestreo.

Para poder realizar la prueba anteriormente descrita, lo primero es diseñar la base de datos, LabVIEW se puede comunicar con archivos Microsoft Access con extensión ".mdb".

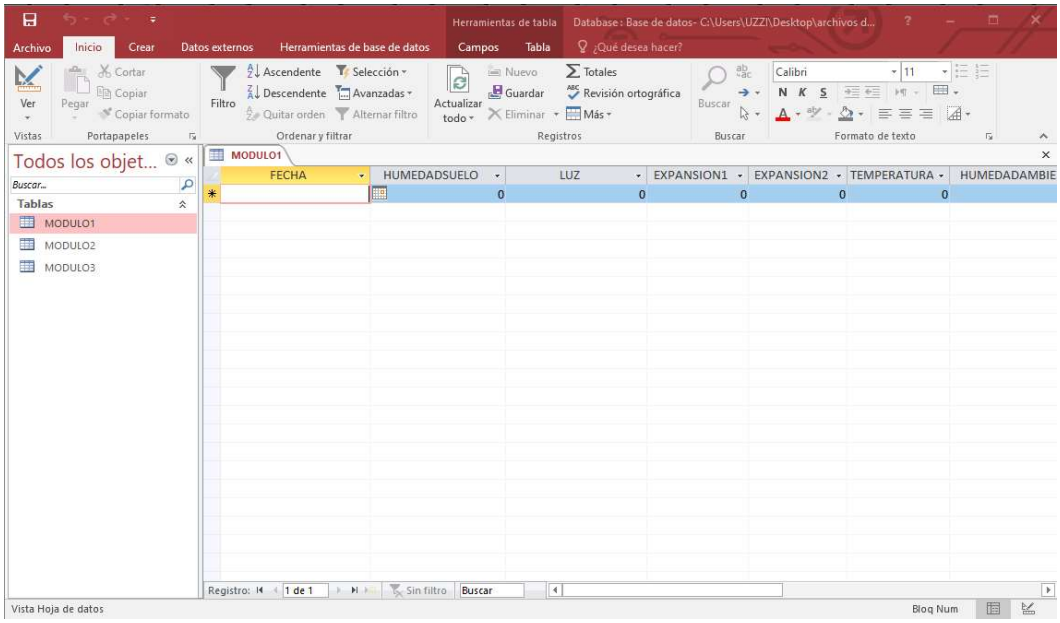


Ilustración 4-69 Tablas para base de datos.

En la ilustración 4-69, se muestran las tablas que contiene la base de datos, con los nombres reales de las columnas que tendrá la base de datos, el siguiente paso es insertar datos en las tablas, para poder insertar estos datos, se hace uso del pequeño programa de LabVIEW mostrado en la ilustración 4-67.

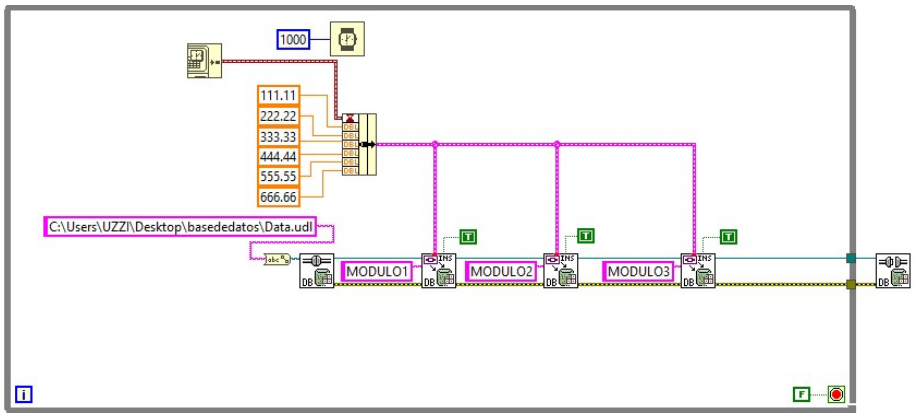


Ilustración 4-70 Programa auxiliar 1 para pruebas con la base de datos.

Con el programa mostrado anteriormente, se insertan datos en las tablas cada segundo durante un día entero, y se puede calcular cuánto espacio de almacenamiento ocupara la base de datos en un año.

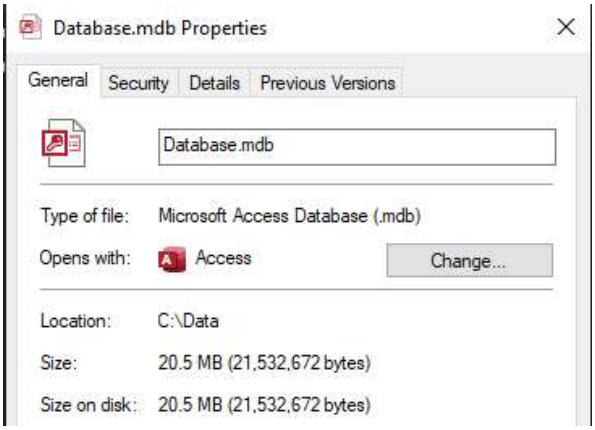


Ilustración 4-71 Tamaño de la base de datos durante un día de uso insertando datos cada segundo.

Tras un día de almacenar datos cada segundo, se puede observar que el espacio de almacenamiento total que ocupa la base de datos es de 20.5 MB, si este tamaño se multiplica por los 365 días del año, se obtiene el resultado de 7.3 GB, esto representa un grave problema para el funcionamiento de la base de datos, si bien para una computadora 7.3 GB no es un espacio relevante, el principal problema está en que los archivos con extensión ".mdb" no puede superar los 2GB de tamaño ya que, de ser así, el archivo se corromperá y se perderán los datos almacenados. Para solucionar este problema se creó un temporizador que permitiera controlar cada cuanto se insertan lecturas a la base de datos, no es posible utilizar otro "delay" con el bloque "wait (ms)" porque si se coloca otro de estos bloques se afectara también al

muestreo principal de lecturas. Existen muchas formas de formar este temporizador, pero la que se utilizó se muestra en las ilustraciones 4-72 y 4-73.

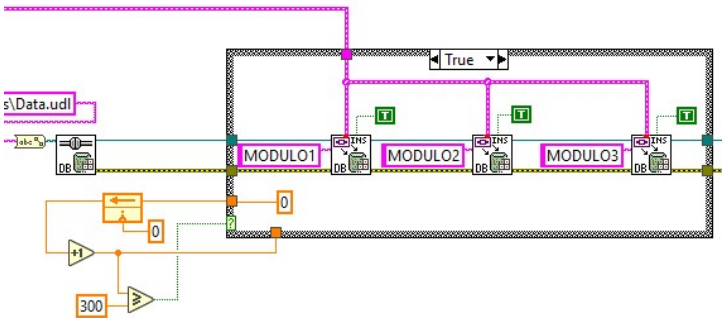


Ilustración 4-72 temporizador para la base de datos, parte 1.

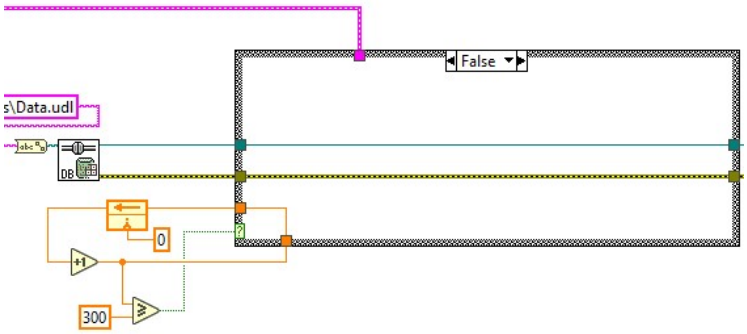


Ilustración 4-73 Temporizador para la base de datos, parte 2.

El funcionamiento del temporizador es muy sencillo, consta de un contador de segundos, sabiendo que cada minuto se compone de 60 segundos, se puede obtener el número de segundos que se deben contar antes de que se inserten datos a la base de datos, en las ilustraciones anteriores se observa que este número es “300”, este número se obtuvo tras convertir cinco minutos a segundos, si el número de veces que se ha repetido el ciclo “while” es inferior a 300, se activara el caso “False” de la estructura “case”, en este caso no se realiza ninguna acción salvo seguir incrementando el contador, si el contador es igual a 300 se activara el caso “True” de la estructura “case”, en este caso sí se insertan datos en la base de datos y también se reinicia el contador en 0.

Se volvió a realizar la prueba de almacenamiento realizada anteriormente, pero esta vez, se realizó durante un día completo e insertando datos cada 5 minutos y el resultado que se obtuvo se muestra en la ilustración 4-74.

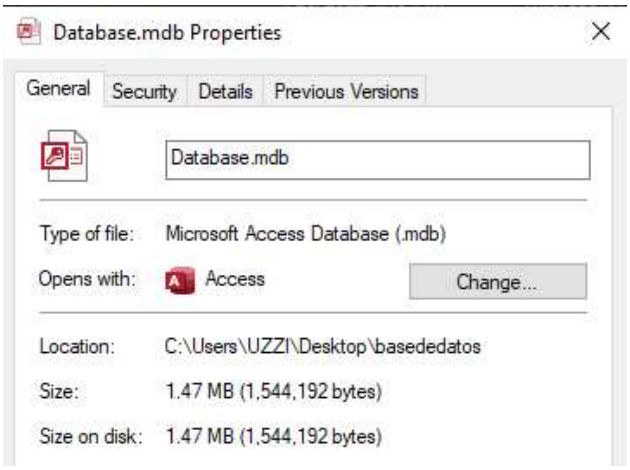


Ilustración 4-74 Tamaño de la base de datos durante un día de uso insertando datos cada cinco minutos.

Tras un día completo almacenando datos cada 5 minutos la base de datos ocupa un espacio de 1.47 MB, si se multiplica este valor por los 365 días del año se obtiene un resultado de 0.523 GB, habiendo obtenido este resultado se puede deducir que introduciendo datos cada 5 min, la base de datos se puede utilizar poco menos de 4 años sin necesidad de cambiar el archivo, aunque esto es considerando que el sistema estará funcionando las 24 horas del día durante los 365 días del año, si se utiliza el sistema con menor frecuencia, el archivo puede durar incluso más tiempo.

Aunque el archivo Access que contiene la base de datos tenga un tiempo determinado de vida útil, esto no significa que una vez llegando al límite de 2 GB el sistema ya no pueda utilizar ninguna base de datos, solo significa que se tiene que remplazar el archivo Access por otro archivo igual pero en blanco, solo basta con colocar el archivo nuevo en la carpeta “Data” y retirar el archivo viejo, este procedimiento es muy sencillo y ya no es necesario realizar todo el diseño de las tablas del archivo Access, ya que este archivo en blanco se proporcionará junto con la interfaz gráfica.

Conclusiones y trabajo futuro

Se diseñaron tres módulos idénticos para medir la humedad en sustratos, humedad, temperatura e iluminación en el ambiente, los cuales adicionalmente tienen puertos para poder agregar un par de sensores adicionales, todo es gestionado por un microcontrolador y transmitido de manera alámbrica.

Para la implementación de la interfaz gráfica, se trabajó con LabVIEW aprovechando que se cuenta con una licencia de uso en el grupo IDEA, además de que el tipo de programación en dicho entorno, facilita el diseño de interfaces y gestión de información. En la interfaz, el agricultor puede observar las mediciones de los tres módulos, que podrá ubicar en diversas zonas del invernadero, y conocer las lecturas de las variables requeridas.

Para poder alcanzar las zonas más alejadas del invernadero, se realizaron pruebas con varios protocolos de comunicación, dentro de los cuales, el que mejores resultados presentó para este sistema fue el RS485; con los dispositivos implementados, se logró comunicación a distancias muy superiores a los 50 metros de manera confiable.

Mediante el uso del software Access y un paquete de expansión para LabVIEW, fue posible crear una base de datos que registra información cada cinco minutos, para que posteriormente el agricultor pueda consultarlos en un intervalo que el misma defina, y de esta forma observar la evolución de las variables y contrastar esa información con el rendimiento de su invernadero, y con ello, corregir los parámetros que considere convenientes.

Se logró incorporar la función de envío de alertas vía GSM con el uso del módulo SIM800L, que tiene un muy bajo precio y es fácil de utilizar con Arduino, aunque este dispositivo utiliza redes 2G y en poco tiempo pueden llegar a dejar de funcionar, en el mercado existen módulos muy similares que funcionan con redes 4G o superiores, como es el caso del módulo IOT GA6.

Por último, se construyeron PCB mediante el uso de la técnica de planchado, aunque estas no tienen un acabado profesional, si son funcionales, además de la posibilidad de adquirir los servicios de empresas que se dediquen a realizar PCB con una calidad superior.

Con todo lo mencionado anteriormente, se puede llegar a la conclusión que el objetivo de desarrollar un sistema que realice el monitoreo de la humedad, temperatura e iluminación de un invernadero, fue cumplido en su totalidad.

Como trabajo futuro, lo principal es hacer funcionar este sistema en un invernadero real para poder comprobar su funcionamiento y fiabilidad, en caso de que la fiabilidad sea lo suficientemente buena, se puede proseguir al desarrollo de las etapas 5, 6 y 7 descritas en el capítulo 1.

Existen varias mejoras que se poder realizar a este sistema, una de ellas es, reemplazar la conexión alámbrica de los módulos 1, 2 y 3 y sustituirla con una conexión inalámbrica con el que no se necesite invertir en una infraestructura de cableado.

Una vez realizadas las pruebas en un entorno real, se puede tener retroalimentación para mejorar la interfaz, así como la presentación de la información histórica, a su vez se pretende agregar algunas opciones del envío de alertas, tal como el poder establecer desde la interfaz un número telefónico y el soporte con WhatsApp.

Otra de las mejoras que se podrían realizar es, diseñar la base de datos mediante un software o lenguaje de programación que no necesite de una licencia de pago, tal como puede ser Python, Java o el uso de otro tipo de hardware como las pantallas táctiles HMI.

Referencias

[1] R. A. Luis, «CIENCIORAMA,» 2014. [En línea]. Available: www.cienciorama.unam.mx/a/pdf323_cienciorama.pdf.

[2] F. J. y. D. Daniel, «Muy Interesante,» [En línea]. Available: https://www.muyinteresante.es/innovacion/articulo/ique-son-los-alimentos-transgenicos. [Último acceso: 8 Julio 2019].

[3] «MSC Invernaderos,» 10 Mayo 2017. [En línea]. Available: https://grupomsc.com/blog/invernadero/que-es-y-como-funciona-un-invernadero. [Último acceso: 5 Julio 2019].

[4] «Hydro Environment,» [En línea]. Available: https://www.hydroenv.com.mx/catalogo/index.php?main_page=page&id=44. [Último acceso: 1 Noviembre 2019].

[5] «Novagric,» [En línea]. Available: https://www.novagric.com/es/venta-invernaderos-novedades/tipos-de-invernaderos/invernadero-tunel. [Último acceso: 1 Noviembre 2019].

[6] «NOVAGRIC,» [En línea]. Available: https://www.novagric.com/es/venta-invernaderos-novedades/tipos-de-invernaderos/invernadero-capilla. [Último acceso: 4 Noviembre 2019].

[7] «NOVAGRIC,» [En línea]. Available: https://www.novagric.com/es/venta-invernaderos-novedades/tipos-de-invernaderos/invernaderos-goticos. [Último acceso: 4 Noviembre 2019].

[8] «NOVAGRIC,» [En línea]. Available: https://www.novagric.com/es/venta-invernaderos-novedades/tipos-de-invernaderos/invernaderos-asimetricos. [Último acceso: 26 Noviembre 2019].

[9] M. A. C. I. Agrónomo, Manual de cultivo del tomate bajo invernadero, Santiago,Chile: Instituto de desarrollo agropecuario , 2017.

[10] NIH, «Instituto nacional de cancer,» [En línea]. Available: https://www.cancer.gov/espanol/publicaciones/diccionario/def/ph. [Último acceso: 12 04 2020].

[11] West Analítica y Servicios S.A. de C.V., «Sustratos en fibra de coco,» Agricultura Razonada, Guadalajara, Mexico.

[12] secretaria de desarrollo agropecuario, «Instituto de Investigación y Capacitación Agropecuaria, Acuícola y Forestal,» [En línea]. Available: http://icamex.edomex.gob.mx/jitomate. [Último acceso: 04 04 2020].

[13] Gobieerno de España, «Ministerio de educacion,» [En línea]. Available: http://newton.cnice.mec.es/materiales_didacticos/magnitudes/magnitudes.html. [Último acceso: 11 04 2020].

[14] «Mecatronica LATAM,» [En línea]. Available: https://www.mecatronicalatam.com/es/tutoriales/sensores/. [Último acceso: 04 04 2020].

[15] Real academia española, «diccionario de la lengua española,» [En línea]. Available: https://dle.rae.es/transductor. [Último acceso: 05 04 2020].

[16] UNAM, «SUAyED,» [En línea]. Available: https://programas.cuaed.unam.mx/repositorio/moodle/pluginfile.php/824/mod_resource/content/5/contenido/index.html. [Último acceso: 11 04 2020].

[17] F. Miyara, Conversores D/A Y A/D, Rosario, Argentina, 2004.

[18] L. Llamas, 19 01 2016. [En línea]. Available: https://www.luisllamas.es/arduino-humedad-suelo-fc-28/. [Último acceso: 05 04 2020].

[19] «Mercado libre,» [En línea]. Available: https://www.mercadolibre.com.mx/. [Último acceso: 12 04 2020].

[20] Vegetronix, «vegetronix,» [En línea]. Available: https://www.vegetronix.com/Products/VH400/. [Último acceso: 11 04 2020].

[21] «naylampmechatronics,» [En línea]. Available: https://naylampmechatronics.com/sensores-temperatura-y-humedad/57-sensor-de-temperatura-y-humedad-relativa-dht11.html. [Último acceso: 12 04 2020].

[22] «naylampmechatronics,» [En línea]. Available: https://naylampmechatronics.com/sensores-temperatura-y-humedad/234-sensor-de-temperatura-analogico-lm35.html. [Último acceso: 12 04 2020].

[23] «HETPRO,» [En línea]. Available: https://hetpro-store.com/TUTORIALES/sensor-de-temperatura-ds18b20/. [Último acceso: 12 04 2020].

[24] «naylampmechatronics,» [En línea]. Available: https://naylampmechatronics.com/sensores-posicion-inerciales-gps/357-sensor-de-presiontemperatura-humedad-bme280.html. [Último acceso: 12 04 2020].

[25] «naylampmechatronics,» [En línea]. Available: https://naylampmechatronics.com/sensores-luz-y-sonido/76-modulo-sensor-de-luz-digital-bh1750.html. [Último acceso: 12 04 2020].

[26] VISHAY, [En línea]. Available: https://www.sparkfun.com/datasheets/Sensors/Imaging/TEMT6000.pdf. [Último acceso: 12 04 2020].

[27] «Arduino,» [En línea]. Available: http://www.arduino.com. [Último acceso: 13 04 2020].

[28] Arduino, «Arduino,» [En línea]. Available: https://www.arduino.cc/en/Guide/Introduction. [Último acceso: 13 04 2020].

[29] «RedHat,» [En línea]. Available: https://www.redhat.com/es/topics/middleware/what-is-ide. [Último acceso: 13 04 2020].

[30] «Arduino,» [En línea]. Available: https://www.arduino.cc/en/main/software. [Último acceso: 13 04 2020].

[31] «Electronica estudio,» [En línea]. Available: https://www.electronicaestudio.com/que-es-un-microcontrolador/. [Último acceso: 13 04 2020].

[32] «PLUSELECTRIC,» [En línea]. Available: https://pluselectric.wordpress.com/2014/09/21/arduino-uno-especificaciones-y-caracteristicas/. [Último acceso: 13 04 2020].

[33] «Arduino,» [En línea]. Available: https://arduino.cl/arduino-nano/. [Último acceso: 14 04 2020].

[34] C. Veloso, «Etools,» 19 06 2018. [En línea]. Available: https://www.electrontools.com/Home/WP/2018/06/19/arduino-mega-2560-caracteristicas/. [Último acceso: 14 04 2020].

[35] «Ferretronica,» [En línea]. Available: https://ferretronica.com/products/modulo-wifi-bluetooth-esp32. [Último acceso: 14 04 2020].

[36] National Instruments, «LabVIEW,» [En línea]. Available: https://www.ni.com/es-mx/shop/labview.html. [Último acceso: 14 04 2020].

[37] Datademia, «Datademia,» [En línea]. Available: https://datademia.es/blog/que-es-sql. [Último acceso: 10 marzo 2021].

[38] J. L. Cedillo Méndez, F. Rafael Esteban y L. O. Ramírez Salas Linares, 08 05 2012. [En línea]. Available: http://132.248.52.100:8080/xmlui/handle/132.248.52.100/734. [Último acceso: 2020 04 29].

[39] @jcrespom, «Aprendiendo Arduino,» 17 Octubre 2018. [En línea]. Available: https://aprendiendoarduino.wordpress.com/category/i2c/#:~:text=I2C%20es%20un%20bus%20de,velocidades%20de%203.4%20Mbit%2Fs.. [Último acceso: 11 Septiembre 2020].

[40] Texas Instruments, «All Datasheet,» Noviembre 2014. [En línea]. Available: https://www.ti.com/lit/ds/symlink/max232.pdf?ts=1605496091217. [Último acceso: 16 Noviembre 2020].

[41] I. E. L. Pérez, «Ingeniería en microcontroladores,» [En línea]. Available: http://www.electronica60norte.com/mwfls/pdf/rs-485.pdf. [Último acceso: 10 Septiembre 2020].

[42] Maxim products inc., 2014. [En línea]. [Último acceso: 05 Octubre 2020].

[43] Eneka, «Eneka lider en electronica,» [En línea]. Available: https://www.eneka.com.uy/robotica/modulos-comunicacion/m%C3%B3dulo-gsm-gprs-sim900-7477-detail.html. [Último acceso: 13 Febrero 2021].

ANEXO

Código para módulos 1, 2 y 3

```
#include<Wire.h>

#include <ErriezBH1750.h>

#include <Adafruit_BME280.h>

Adafruit_BME280 bme;

BH1750 sensor(LOW);

int i;

float lecturas[5];

char conversion[15];

String datos="";

float Hsuelo=0;

void setup() {

  Serial.begin(115200);

  Wire.begin();

  sensor.begin(ModeContinuous, ResolutionMid);

  sensor.startConversion();

  bme.begin(0x76);

}

void loop() {

 uint16_t lux = sensor.read();

 int vh400 = analogRead(A0)*0.004887585;

 float EXP1 = analogRead(A1)*0.004887585;

 float EXP2 = analogRead(A2)*0.004887585;
```

```
float Hum = bme.readHumidity();
float Tem = bme.readTemperature();
if(vh400<1.2512){
  Hsuelo=((0.1782*vh400)-0.0339)*100;
        }else{
  Hsuelo=((0.4422*vh400)-0.3703)*100;
}
float luz = lux;
lecturas [0] = Hsuelo;
lecturas [1] = luz/1000;
lecturas [2] = EXP1;
lecturas [3] = EXP2;
lecturas [4] = Tem;
lecturas [5] = Hum;
for (i = 0; i <= 5; i++) {
  float valor = lecturas [i];
  dtostrf(valor, 4, 2, conversion);
  datos += conversion;
  if (i<=4){
    datos += " ";
  }else{
     datos += " #";
          }
}
```

```
  Serial.println(datos);
  datos = "";
  delay(100);
}
```

Código para el módulo maestro

```
#include <Separador.h>
#include <Sim800l.h>
#include <SoftwareSerial.h>
Sim800l Sim800l;
Separador s;
unsigned long Timer,TimAlert,Conversion;
int HSmin,Lmin,Tmin,HAmin,HSmax,Lmax,Tmax,HAmax,Minutos;
char* text;
char* number;
String Datos;
String str="";
String str1="";
String str2="";
void setup() {
   Serial.begin(250000);
   Serial1.begin(115200);
   Serial2.begin(115200);
   Serial3.begin(115200);
   Sim800l.begin();
```

```
  HSmin=-5;
  Lmin=-5;
  Tmin=-100;
  HAmin=-5;
  HSmax=3000;
  Lmax=3000;
  Tmax=3000;
  HAmax=3000;
  Minutos = 15;
}
void loop() {
  if (Serial.available()){
   adquisicion();
  } else{
    while (Serial1.available()){
      str = Serial1.readStringUntil('#');
    }
    while (Serial2.available()){
      str1 = Serial2.readStringUntil('#');
    }
    while (Serial3.available()){
      str2 = Serial3.readStringUntil('#');
    }
```

```
  if(millis() > Timer + 1000){
    Timer = millis();
    Datos="";
    Datos += str;
    Datos += str1;
    Datos += str2;
    Serial.println(Datos);
    str ="0.0 0.0 0.0 0.0 0.0 0.0 ";
    str1="0.0 0.0 0.0 0.0 0.0 0.0 ";
    str2="0.0 0.0 0.0 0.0 0.0 0.0 ";
   }
   Conversion = Minutos * 60000;
   if (millis() > TimAlert + Conversion){
      TimAlert = millis();
      sms();
   }
  }
}
void adquisicion(){
  String recibidos = Serial.readString();
  String lec1 = s.separa(recibidos, ',', 0);
 HSmin = lec1.toInt();
  String lec2 = s.separa(recibidos, ',', 1);
 HSmax = lec2.toInt();
```

```
 String lec3 = s.separa(recibidos, ',', 2);
 Lmin = lec3.toInt();
 String lec4 = s.separa(recibidos, ',', 3);
 Lmax = lec4.toInt();
 String lec5 = s.separa(recibidos, ',', 4);
 Tmin = lec5.toInt();
 String lec6 = s.separa(recibidos, ',', 5);
 Tmax = lec6.toInt();
 String lec7 = s.separa(recibidos, ',', 6);
 HAmin = lec7.toInt();
 String lec8 = s.separa(recibidos, ',', 7);
 HAmax = lec8.toInt();
 String lec9 = s.separa(recibidos, ',', 8);
 Minutos = lec9.toInt();
}
  void sms(){
    String Dat0 = s.separa(Datos, ' ', 0);
    if (Dat0.toInt()<HSmin || Dat0.toInt()>HSmax){
        Sim800l.sendSms("+525618747131","ATENCION, LA HUMEDAD EN EL
SUSTRATO DE ZONA 1 ESTA FUERA DEL RANGO ESTABLECIDO");
        }
    String Dat1 = s.separa(Datos, ' ', 1);
    if (Dat1.toInt()<Lmin || Dat1.toInt()>Lmax){
        Sim800l.sendSms("+525618747131","ATENCION, LA LUZ AMBIENTAL DE
ZONA 1 ESTA FUERA DEL RANGO ESTABLECIDO");
```

```
    }
  String EXP11 = s.separa(Datos, ' ', 2);
  String EXP12 = s.separa(Datos, ' ', 3);
  String Dat2 = s.separa(Datos, ' ', 4);
  if (Dat2.toInt()<Tmin || Dat2.toInt()>Tmax){
    Sim800l.sendSms("+525535301951","ATENCION, LA TEMPERATIRA AMBIENTE DE ZONA 1 ESTA FUERA DEL RANGO ESTABLECIDO");
    }
  String Dat3 = s.separa(Datos, ' ', 5);
  if (Dat3.toInt()<HAmin || Dat3.toInt()>HAmax){
    Sim800l.sendSms("+525535301951","ATENCION, LA HUMEDAD AMBIENTAL DE ZONA 1 ESTA FUERA DEL RANGO ESTABLECIDO");
    }
  String Dat4 = s.separa(Datos, ' ', 6);
  if (Dat4.toInt()<HSmin || Dat4.toInt()>HSmax){
     Sim800l.sendSms("+525535301951","ATENCION, LA HUMEDAD EN EL SUSTRATO DE ZONA 2 ESTA FUERA DEL RANGO ESTABLECIDO");
    }
  String Dat5 = s.separa(Datos, ' ', 7);
  if (Dat5.toInt()<Lmin || Dat5.toInt()>Lmax){
    Sim800l.sendSms("+525535301951","ATENCION, LA LUZ AMBIENTAL DE ZONA 2 ESTA FUERA DEL RANGO ESTABLECIDO");
    }
  String EXP21 = s.separa(Datos, ' ', 8);
  String EXP22 = s.separa(Datos, ' ', 9);
```

```
String Dat6 = s.separa(Datos, ' ', 10);
if (Dat6.toInt()<Tmin || Dat6.toInt()>Tmax){
  Sim800l.sendSms("+525535301951","ATENCION, LA TEMPERATURA AMBIENTE DE ZONA 2 ESTA FUERA DEL RANGO ESTABLECIDO");
  }
String Dat7 = s.separa(Datos, ' ', 11);
if (Dat7.toInt()<HAmin || Dat7.toInt()>HAmax){
  Sim800l.sendSms("+525535301951","ATENCION, LA HUMEDAD AMBIENTAL DE ZONA 2 ESTA FUERA DEL RANGO ESTABLECIDO");
  }
String Dat8 = s.separa(Datos, ' ', 12);
if (Dat8.toInt()<HSmin || Dat8.toInt()>HSmax){
   Sim800l.sendSms("+525535301951","ATENCION, LA HUMEDAD EN EL SUSTRATO DE ZONA 3 ESTA FUERA DEL RANGO ESTABLECIDO");
   }
String Dat9 = s.separa(Datos, ' ', 13);
if (Dat9.toInt()<Lmin || Dat9.toInt()>Lmax){
  Sim800l.sendSms("+525535301951","ATENCION, LA LUZ AMBIENTAL DE ZONA 3 ESTA FUERA DEL RANGO ESTABLECIDO");
  }
String EXP31 = s.separa(Datos, ' ', 14);
String EXP32 = s.separa(Datos, ' ', 15);
String Dat10 = s.separa(Datos, ' ', 16);
if (Dat10.toInt()<Tmin || Dat10.toInt()>Tmax){
```

```
      Sim800l.sendSms("+525535301951","ATENCION, LA TEMPERATURA
AMBIENTE DE ZONA 3 ESTA FUERA DEL RANGO ESTABLECIDO");
      }
    String Dat11 = s.separa(Datos, ' ', 17);
    if (Dat11.toInt()<HAmin || Dat11.toInt()>HAmax){
      Sim800l.sendSms("+525535301951","ATENCION, LA HUMEDAD AMBIENTAL
DE ZONA 3 ESTA FUERA DEL RANGO ESTABLECIDO");
      }
 }
```