UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

“Desarrollo de un dosificador automático de croquetas bajo paradigma IoT”

TESIS

Para obtener el título de:

Ingeniero Eléctrico Electrónico

Presenta:

Ángel Amezquita Flores

Asesor de tesis:

Dr. Ismael Díaz Rangel

Ciudad Nezahualcóyotl, Estado de México, 2023

AGRADECIMIENTOS

La presente tesis se concluyó gracias a las valiosas opiniones, sugerencias, y la asistencia brindada por mis compañeros y profesores de grupo IDEA de la Facultad de Estudios Superiores Aragón. Agradezco, especialmente al Dr. Ismael Díaz Rangel el tiempo y la dedicación entregados durante la asesoría de este proyecto, así como, su paciencia y enseñanzas. Resulta importante también reconocer la labor que hace la Universidad Nacional Autónoma de Mexico por proveernos de una educación pública, gratuita, laica y de calidad, y por otorgar recursos para el desarrollo de este trabajo. También, quiero expresar mi gratitud a mis familiares y amigos que me ayudaron directamente en el proyecto, económicamente o, no menos importante, con su amistad. Finalmente, estoy muy agradecido con mi madre Leticia Flores López, mi padre Armando Amezquita Toledo, q. e. p. d., y mi hermana Ana Lucía Amezquita Flores por todo el apoyo y cariño que me brindan y que me motiva a ser un mejor profesional.

ÍNDICE

CAPÍTULO 1. INTRODUCCIÓN

Los perros están presentes en la vida de la mayoría de los mexicanos, tan solo el 69 % de los hogares cuenta con algún tipo de mascota, lo que da un total acumulado de 80 millones de mascotas, de las cuales 43.8 millones son perros (Instituto Nacional de Estadística y Geografía (INEGI), 2021).

Al momento de adquirir o adoptar un perro los dueños también adquieren una serie de cuidados y atenciones que se deben de tomar en cuenta. De estos cuidados uno de los más importantes es una correcta alimentación. Para mantener la salud del perro es importante que consuma una dieta equilibrada, que satisfaga sus necesidades energéticas y proporcione los nutrientes adecuados. El alimento seco facilita esta tarea, ya que en el mercado se puede encontrar distintas variedades adaptadas a cada etapa de crecimiento y a las diferentes necesidades del perro (Gracia García, 2018), porque con este tipo de alimento el dueño solo se debe de ocupar de dar la ración y el tipo de alimento sugeridos por un médico veterinario. Aunque, el dueño aún tiene el compromiso de aprender a dar a la mascota el alimento en los horarios y las cantidades que requiere, por ejemplo: cuando se elige el régimen de alimentación por porciones controladas, es preferible que la ración de alimento sea dividida en dos o más comidas (Case, Daristotle, Hayek, & Raasch, 2011) por día.

En este trabajo tiene como objetivo: diseñar un dispensador, configurable mediante una aplicación web y móvil, bajo paradigma IoT (Intenet of Things). El dispensador tiene el propósito de entregar el alimento a la hora y ración seleccionadas por el usuario. Para ello se desarrollaron una aplicación web y móvil que permiten al usuario: configurar las raciones de alimento, los horarios de entrega y visualizar el nivel de alimento en el depósito. Debido a que el dispositivo se conecta a internet, tiene la ventaja de que puede ser configurado desde cualquier parte del mundo con conexión a la red. También, se aprovecharon las siguientes características que ofrece la plataforma IoT: permite la actualización del firmware del microcontrolador en línea mediante conexión wifi, monitorear las conexiones del dispensador y enviar notificaciones a la aplicación móvil.

1.1 Objetivo general

Desarrollar un dosificador automático de croquetas bajo paradigma de internet de las cosas.

1.2 Objetivos particulares

- Capacidad de establecimiento de horarios de comida.
- Configuración de porciones de alimento.
- Medición de nivel de alimento disponible.
- Visualizador de estatus de configuraciones accesible en el sistema.
- Interfaz de comunicación vía aplicación móvil.
- Sistema de dosificación a prueba de atascos.

1.3 Actividades

- Analizar la alimentación de los perros.
- Investigar métodos para la medición de nivel en un recipiente.
- Conocer visualizadores de texto.
- Identificar alternativas para desarrollo IoT.
- Revisar métodos para la dosificación de alimentos.
- Estudiar motores de corriente directa.
- Examinar drivers de motores.
- Desarrollar una interfaz de operación sobre una aplicación web y sobre una aplicación móvil
- Crear y codificar algoritmos de programación para el sistema.
- Diseñar un prototipo del dosificador.
- Crear tarjetas PCB.

1.4 Justificación

En México la mayoría de los hogares con mascota optan por adoptar o adquirir un perro, con ello los dueños adquieren la obligación de garantizar el bienestar de sus mascotas y, por lo tanto, les deben proveer de una alimentación balanceada y nutritiva. Esto se puede lograr con distintos tipos de dietas, de las que los alimentos comerciales se han popularizado como base nutricional de perros y gatos. En los Estados Unidos la mayoría de los consumidores prefiere adquirir alimento seco (Case, Daristotle, Hayek, & Raasch, 2011), esto es porque: son fáciles de almacenar; son más baratos, comparados con los alimentos húmedos y semihúmedos, y existe una amplia variedad disponible en el mercado para distintos tamaños, edades y necesidades específicas.

Cuando el dueño opta por alimentar al perro con un régimen de porciones controladas tiene la obligación de proveerle al perro el alimento en las cantidades y horarios establecidos. Hay casos en los que el propietario no puede cumplir con el régimen de alimentación, por ejemplo, cuando se encuentra fuera de casa; por este motivo, en grupo IDEA se desarrolló un sistema automático dispensador de croquetas para mascotas configurable mediante aplicación Android que auxilia al dueño en la entrega de alimento en los horarios y cantidades configuradas. Este trabajo es una actualización tecnológica del sistema mencionado, en el que: se actualizó la tecnología de comunicación inalámbrica, de Bluetooth a wifi; se diseñó un prototipo del dispensador buscando

optimizar el espacio del contenedor y hacerlo más atractivo para el usuario; se incorporó el sensado de nivel, y se usó una plataforma IoT para la creación de una aplicación, visualmente, más atractiva, intuitiva y fácil de usar. Además, de utilizar otras características que ofrece la plataforma como: la actualización del firmware del microcontrolador *Over The Air* (mediante la comunicación con la plataforma IoT y sin la necesidad de cables); las notificaciones en la aplicación móvil, para notificar cuando el nivel de alimento sea demasiado bajo, y el monitoreo de las conexiones realizadas por el sistema.

1.5 Antecedentes

En el mercado existe una amplia oferta de dispensadores automáticos de alimento, por mencionar algunos, tenemos los siguientes:

- **PAPIFEED CAM1080 Automatic Pet Feeder** (figura 1.1a) que anuncia: una capacidad de 6 L, entrega hasta 30 comidas al día, es configurable mediante aplicación móvil, entrega 20 g de alimento por porción, utiliza conexión wifi y tiene un precio de $1244.59. El producto también menciona que utiliza cuatro baterías AA que sirven como respaldo energético. El fabricante recomienda utilizar alimento que tenga un tamaño entre los 5 y 10 mm.
- **Dispensador Wi-Fi de alimento para mascotas con cámara UHD 2K y grabador de voz Steren** (figura 1.1b) que ofrece: una capacidad de 4 L; dosificar de 1 hasta 10 porciones de 10 g cada una; se conecta a la red mediante conexión wifi; cuenta con una cámara Full HD incorporada y detección de movimiento, para monitorear la actividad de la mascota, y utiliza 3 baterías tipo D como respaldo de energía. El fabricante recomienda que sea rellenado con alimento seco que tenga un tamaño menor a 12 mm. Este dispensador tiene un precio en el mercado, al momento de escribir este trabajo, de $2990.00.

(a) ***(b)***

Figura 1.1. Dispensadores de alimento disponibles en el mercado

En el año 2019 en el grupo IDEA de la FES Aragón se diseñó un sistema automático dispensador de croquetas para mascotas configurable mediante aplicación Android (figura 1.2). Este sistema está conformado de una interfaz desarrollada en Android en la que se configuraban las raciones y horarios en los que se sirve el alimento. Se envía la información de las configuraciones mediante Bluetooth al sistema embebido para su posterior almacenamiento y procesamiento (Ramos Tellez, 2019). La figura 1.3 muestra el diagrama de bloques de ese sistema.

Figura 1.2. Prototipo del dispensador desarrollado en el 2019

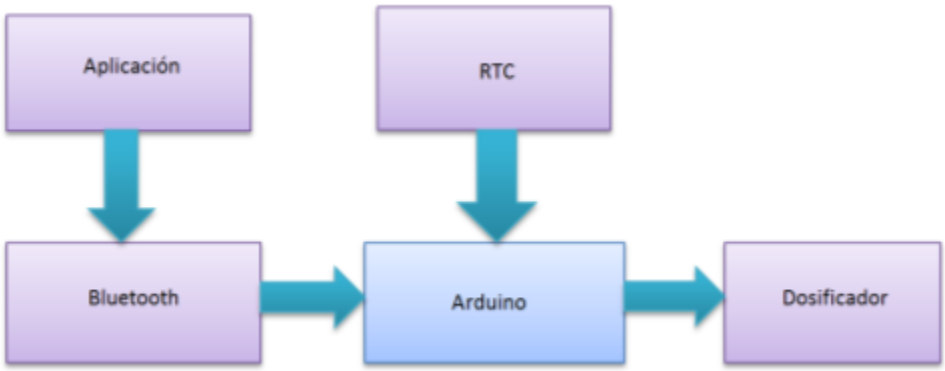


Figura 1.3. Diagrama a bloques del dispensador desarrollado en el 2019

CAPÍTULO 2. MARCO TEÓRICO

2.1 La alimentación de los perros

Diferentes estudios demuestran que a partir del comportamiento ingestivo natural, las necesidades nutritivas, el metabolismo y la anatomía podemos clasificar a los perros (familia Canidae) como omnívoros (Elices, 2010, p. 2).

En la naturaleza los animales se alimentaban de comida cruda sin procesar. Ellos no solo consumen lo que es para los humanos, alimentos aceptables son: carne, hígado, riñones, entre otros. También se alimentan de lo que consideramos inaceptable: huesos, sesos, piel y menudencias. Esto les permite obtener una dieta más balanceada.

Actualmente existen dos tipos de dietas para perros: caseras y comerciales (Agar & Hough, 2001, pág. 25).

2.1.1 Dieta casera

Agar y Hough (2001) mencionan que se debe de considerar lo siguiente para la preparación de alimentos en casa. La cocción altera los alimentos. Los cereales y vegetales se vuelven más fáciles de digerir porque los almidones se separan, pero algunos nutrientes son destruidos por el calor, y el sobrecalentamiento de las proteínas puede provocar reacciones químicas que las vuelve menos digeribles. Tomando esto en cuenta, la comida casera solo debe ser ligeramente cocida y el agua usada en la cocción debe mantenerse y ser mezclada con ella para conservar los nutrientes. No se debe de añadir sal. La comida casera debe de ser lo más variada posible para asegurar una adecuada nutrición.

También consideran que es posible proveer una dieta casera balanceada, pero requiere un esfuerzo considerable, puede ser cara, y produce olores fuertes. Por estas y otras razones, la mayoría de los dueños prefiere optar por los alimentos comerciales (p. 26).

La

Tabla 2.1 (Agar & Hough, 2001, pág. 27) sugiere una variedad de alimentos que se pueden usar para preparar una comida casera.

Tabla 2.1 Fuentes de nutrientes en las dietas caseras

Proteína
Carne – retirar la grasa (puede ser demasiada)
Pescado
Huevos y queso
Leche en polvo (pequeñas cantidades)
Legumbres – frijoles y chicharos (sabor poco agradable y tienden a causar flatulencias)

Grasas y aceites

Origen animal o vegetal (usar esporádicamente, denso contenido energético, un alto contenido de grasas puede reducir el consumo de otros nutrientes)

Carbohidratos

Pan y papas (también contiene pequeñas cantidades de proteínas)

Arroz y pasta (se digieren fácilmente)

Vitaminas y minerales

Vegetales verdes y raíces son una fuente barata de vitaminas, minerales y fibra, pero tienen un sabor poco agradable, y no son aceptados por todos los perros

La variedad en la dieta asegura un suministro adecuado de la mayoría de las vitaminas y minerales; usar suplementos si la variedad es insuficiente

2.1.2 Tipos de alimentos comerciales

Como afirman Agar y Hugh (2001) los alimentos comerciales pueden ser completos o complementarios:

- Los alimentos completos, como el nombre lo sugiere, son aquellos que pueden proveer una dieta completa, balanceada y adecuada por si solos.
- Los alimentos complementarios no son balaceados y requieren de otros alimentos para conformar una dieta balanceada y adecuada (p. 27).

Estos a su vez se pueden dividir en tres categorías de acuerdo con su contenido de humedad (alimentos húmedos, secos y semihúmedos) (Case, Daristotle, Hayek, & Raasch, 2011):

- Los alimentos húmedos (figura 2.1) contienen >60% de humedad (Elices, 2010). Están disponibles en variedad de presentaciones, de las que la más común es la enlatada (Agar & Hough, 2001). Se encuentran en la categoría de alimentos completos. Tienen alta palatabilidad y mayor densidad energética, lo que con un mal empleo puede conllevar sobrealimentación y obesidad. Precio (coste/caloría) elevado. Requieren de refrigeración después de abiertos y por lo tanto existe una alta posibilidad de desperdicio (Elices, 2010).

(a) *(b)*

Figura 2.1. Alimentos húmedos

- Los alimentos semihúmedos (figura 2.2) son del 15 al 35% humedad. Tienen una densidad energética intermedia entre los alimentos secos y húmedos (Elices, 2010). Están cubiertos de jarabe de maíz u otro tipo de jarabe que, combinado con el agua, previene que el alimento se seque y evita el crecimiento de microorganismos. Pueden permanecer bastante tiempo almacenados, no requieren refrigeración y se pueden surtir libremente. Son menos costosos que los alimentos húmedos y contienen variedad de ingredientes. La cubierta de jarabe tiende a incrementar el contenido energético y los hace también incompatibles con animales diabéticos (Agar & Hough, 2001).

Figura 2.2. Alimentos semihúmedos

- Los alimentos secos contienen del 3 al 12% de humedad (Elices, 2010). Dependen de su bajo contenido de humedad para su preservación. Son empaquetados en bolsas o en cajas de cartón. Se dividen en dos grupos: comidas y galletas. Pueden ser completos o complementarios (Agar & Hough, 2001).
  Existen varios métodos de fabricación (ilustrados en la figura 2.3): extrusionados (parte superior), granulados (parte inferior derecha) y copos (parte inferior izquierda) (Elices, 2010).
  En (Agar & Hough, 2001) se plantea que el ligero efecto abrasivo de los alimentos extrusionados en los dientes previene la acumulación de placa.
  Los alimentos secos son menos costosos que los alimentos húmedos y semihúmedos. Requieren un menor espacio para su almacenamiento y no necesitan refrigeración, pero pueden tener una menor palatabilidad que los húmedos y semihúmedos (p. 29).

Figura 2.3. Distintos métodos de fabricación de los alimentos secos

2.2 Sensores de nivel

Según Corona Ramírez et al. (2014) un transductor es un dispositivo que convierte una variable física en otra de dominio diferente.

Afirman que el sensor no solo convierte la señal, sino que además la salida del sensor será un dato útil para un sistema de medición. De este modo, un sensor se define como un dispositivo de entrada que provee una salida manipulable de la variable medida.

Webster (1998) define nivel como la altura que alcanza un líquido o materia en un tanque o depósito. Generalmente la posición de la superficie es comparada con un plano de referencia, usualmente el fondo del tanque.

Si la superficie del producto no es plana (por ejemplo, espuma, turbulencia, o materia en forma de granos gruesos) el nivel es definido como el promedio de la altura dentro de un área.

Existen varios métodos para medir el nivel de materia, en la presente sección se presentarán algunos de ellos.

2.2.1 Sensor ultrasónico

Ultrasonido es una onda acústica con una frecuencia mayor que el rango audible del ser humano, que es de 20 kHz. Puede estar dentro del rango audible de algunos animales, como perros, murciélagos, o delfines (Webster, 1998).

Los sensores ultrasónicos son utilizados en sistemas de medición no invasivos para determinar la distancia del emisor a un objeto dado. Una de las principales ventajas de este tipo de sensado es que, al ser una medición no invasiva, es decir, que no requiere contacto alguno para realizar la medida, la variedad de objetos que es posible medir es muy amplia. La señal ultrasónica se puede generar mediante diferentes técnicas, como electromagnéticas, ópticas, capacitivas y piezoeléctricas; de todas, esta última es una de las más utilizadas debido a su alta efectividad en comparación con las anteriores (Corona Ramírez, Abarca Jiménez, & Mares Carreño, 2014).

El principio de funcionamiento para el uso del ultrasonido como una herramienta de medición, es la técnica de tiempo de vuelo. El método de eco de pulso es un ejemplo. En el método de eco de pulso, un pulso ultrasónico es transmitido en un medio. Cuando el pulso alcanza otro medio, es total o parcialmente reflejado, y se mide la diferencia de tiempo de la emisión a la detección del pulso reflejado. Está diferencia de tiempo depende de la distancia y de la velocidad del sonido.

Sabiendo que el sonido viaja a una velocidad conocida *c*, y con el tiempo *t* que pasa entre la señal de salida y el eco que regresa se puede calcular la distancia *d* al objeto que ocasiona el eco (Webster, 1998).

$$d = \frac{ct}{2} \tag{2.1}$$

La figura 2.4 muestra un sistema simple de un sensor ultrasónico.

El oscilador genera una señal eléctrica con una frecuencia típica de 40 kHz. Esta señal eléctrica es transformada a vibraciones mecánicas de la misma frecuencia en el transmisor. Estas vibraciones generan ondas de sonido que son reflejadas por el objeto. El sonido reflejado causa una señal eléctrica en el receptor (Webster, 1998).

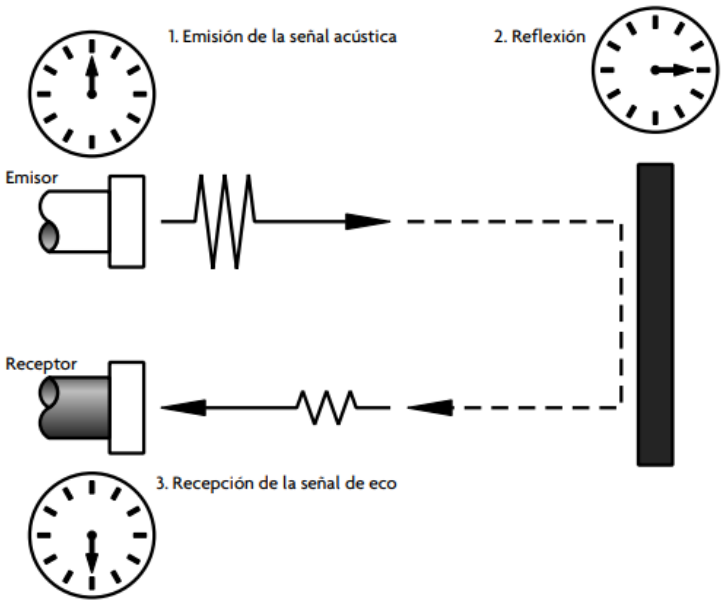


Figura 2.4. Medición de un sensor ultrasónico

El HC-SR04 (figura 2.5) es un sensor ultrasónico que mide la distancia a un objeto, con el método de eco de pulso, emitiendo una señal ultrasónica de 40 kHz en el transmisor. El módulo incluye transmisores ultrasónicos, receptor y circuito de control (Corona Ramírez, Abarca Jiménez, & Mares Carreño, 2014).

La descripción de los pines del sensor es la siguiente:

- VCC: Alimentación de 5V
- Trig: Señal de entrada que habilita una medición del sensor.
- Echo: Señal de salida que emite un pulso en alto con una duración correspondiente al tiempo que le toma a la señal ultrasónica salir del TX y llegar al RX.

Figura 2.5. Módulo HC-SR04

Para hacer una medición con el módulo HC-SR04 primero se tiene que enviar una señal en alto a través del pin Trig con una duración de al menos 10 µs, como se muestra en la figura 2.6. El trasmisor TX envía ocho pulsos cuadrados a una frecuencia de 40 kHz. Cuando se transmite la señal ultrasónica, el pin Echo envía una señal en alto con una duración que depende del tiempo que tarde la señal en llegar al receptor RX. Si el tiempo en alto de Echo es de 38 ms indica que el receptor nunca recibió alguna señal (Corona Ramírez, Abarca Jiménez, & Mares Carreño, 2014).

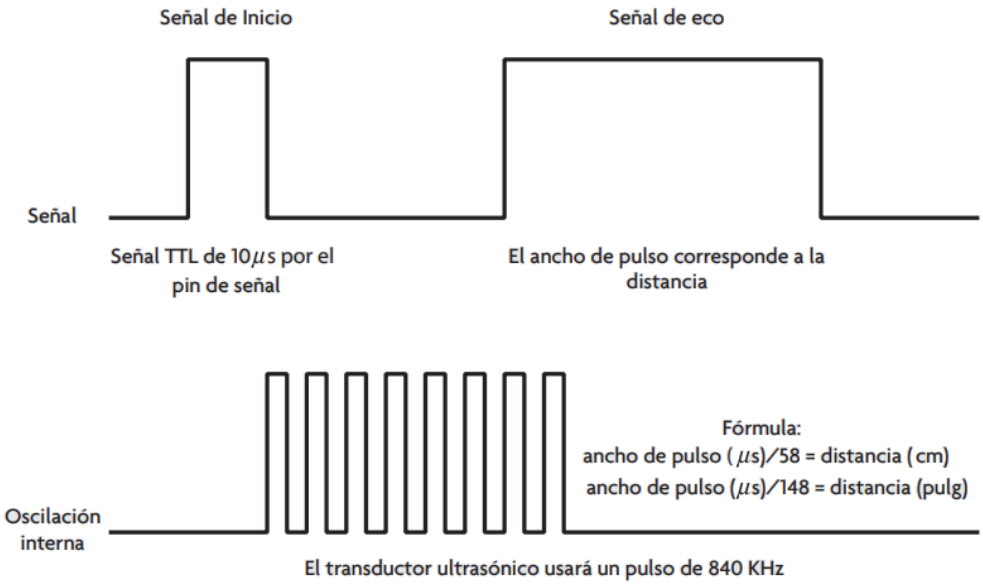


Figura 2.6. Diagrama de tiempos HC-SR04

2.2.2 Sensor láser

En el mercado podemos encontrar dos tipos principales de sensores láser de distancia: los de medición por triangulación (corto alcance y mayor precisión) y los de medición por tiempo de vuelo (largo alcance).

Los sensores láser de medición por triangulación (figura se conforman de los siguientes componentes (Leuze electronic, 2021):

- Transmisor: láser.
- Receptor.
- Visualizador OLED.
- Panel de control con botones de control.
- LED de estado.
- Conexión para conectar al control: conector M12.

Figura 2.7. Sensor láser de medición por triangulación

1. Carcasa.
2. LED de estado.
3. Botones de control.
4. Visualizador.
5. Transmisor.
6. Receptor.
7. Conexión.

Operan bajo el principio de medición de triangulación (figura 2.8), mediante el cual se determina la distancia de un objeto con el ángulo de incidencia de la luz reflejada del objeto (Leuze electronic, 2021).

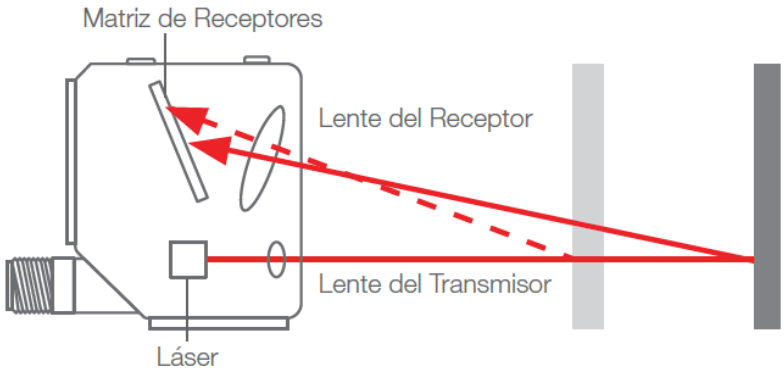


Figura 2.8. Principio de medición por triangulación

Las principales ventajas de este método de medición son (Leuze electronic, 2021):

- Cortos tiempos de respuesta y, debido a esto, altas velocidades de medición.
- Alta precisión.

El principio de funcionamiento de los sensores laser de tiempo de vuelo (figura 2.9) es el mismo que el de sus homólogos ultrasónicos, pero en vez de transmitir y recibir señales de ultrasonido para calcular la distancia a medir, lo hace con señales de luz.

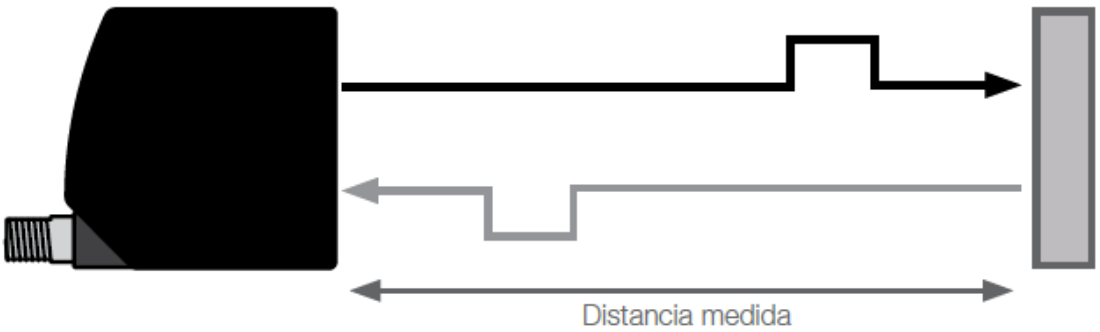


Figura 2.9. Sensor laser de tiempo de vuelo

2.2.3 Sensor infrarrojo

Los sensores infrarrojos de proximidad tienen dos elementos principales: un emisor de luz infrarroja y un elemento fotosensible, conocido como receptor. Por lo general los encapsulados están en un solo dispositivo como lo ilustra la figura

Comúnmente se utiliza un LED infrarrojo como elemento emisor, y un fotodiodo o fototransistor, para detectar la presencia o ausencia del haz de luz emitido. La figura 2.10 muestra el esquema básico de un sensor infrarrojo.

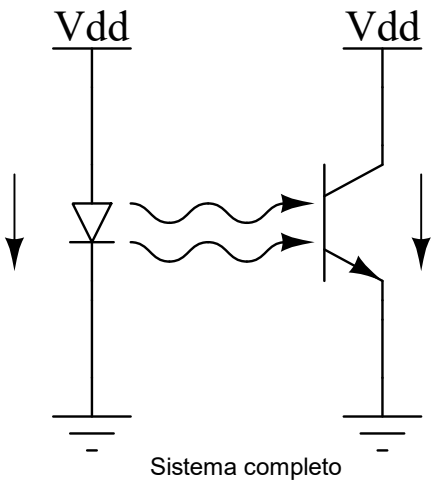


Figura 2.10. Esquema de un sensor infrarrojo

Las configuraciones típicas del fototransistor, al igual que en el caso de un transistor, son de emisor común o de colector común.

La configuración de emisor común (véase figura 2.11 a) consiste en generar un cambio en la señal eléctrica de estado alto a estado bajo. El voltaje de salida se mide en el colector. Este circuito básico tiene la función de actuar como un amplificador, ya que amplifica la corriente generada en la base debido a la presencia de luz.

La configuración de colector común (véase figura 2.11 b) funciona de forma opuesta a la de emisor común, ya que cambia el estado de la señal de estado bajo a estado alto en presencia de luz en la base. El voltaje de salida se mide en el emisor.

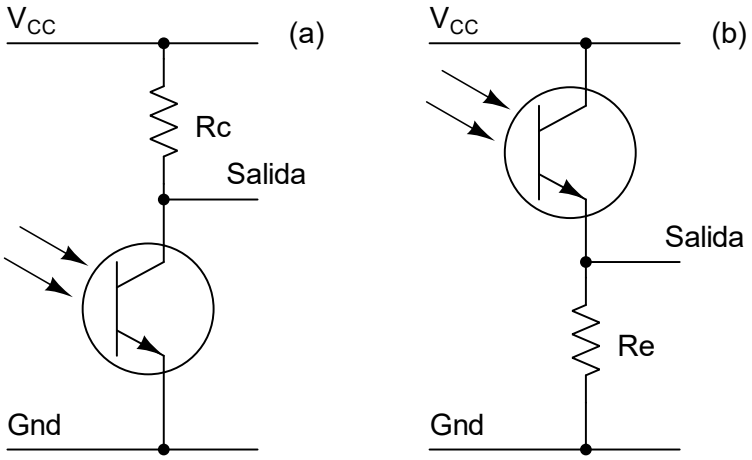


Figura 2.11. (a) Configuración de emisor común (b) Configuración de colector común

El fototransistor se puede utilizar en tipo *switch* o en modo activo. La configuración en modo de *switch* es ampliamente utilizada para detectar la presencia de un objeto o en *encoders*. Esta forma de operación consiste en llevar a corte o a saturación el transistor, con lo que se genera una señal digital que determina la presencia o ausencia del objeto. Para lograr esta señal, la fuente de luz debe estar presente, a fin de lograr la saturación del transistor, o estar obstruida por completo para llevar el transistor a corte. Por otra parte, la configuración en modo activo, también conocida como configuración lineal, consiste en medir el cambio de voltaje correspondiente a la intensidad de luz percibida por el elemento fotosensible. El modo de operación del fototransistor es manipulado cambiando el valor de la resistencia de carga. Para determinar en qué modo se desea utilizar al transistor, se debe elegir una entre las dos condiciones siguientes:

$$modo\ activo: \quad V_{CC} > R_L \times I_C$$

$$modo\ switch: \quad V_{CC} < R_L \times I_c$$

donde:

R_L : resistencia de carga

I_C : corriente de colector

V_{CC}: voltaje de alimentación

Existen distintas configuraciones para aprovechar la luz emitida por el LED (figura 2.12). En el inciso a) se muestra la configuración de retrorreflector, en esta configuración el objeto refleja el haz de la fuente de luz y produce un cambio en la intensidad de la señal que genera el receptor, de esta forma se detecta la presencia de un objeto. Esta configuración se caracteriza por tener un objeto auxiliar donde se refleja la fuente de luz. En b) se muestra la configuración de haz fijo, en general utilizada en detectores de presencia. En c) se muestra la configuración de foco fijo, en la que la intensidad de la señal registrada en el receptor depende de la proximidad del objeto al emisor. Esta configuración se utiliza para medidores de distancia y presencia (Corona Ramírez, Abarca Jiménez, & Mares Carreño, 2014).

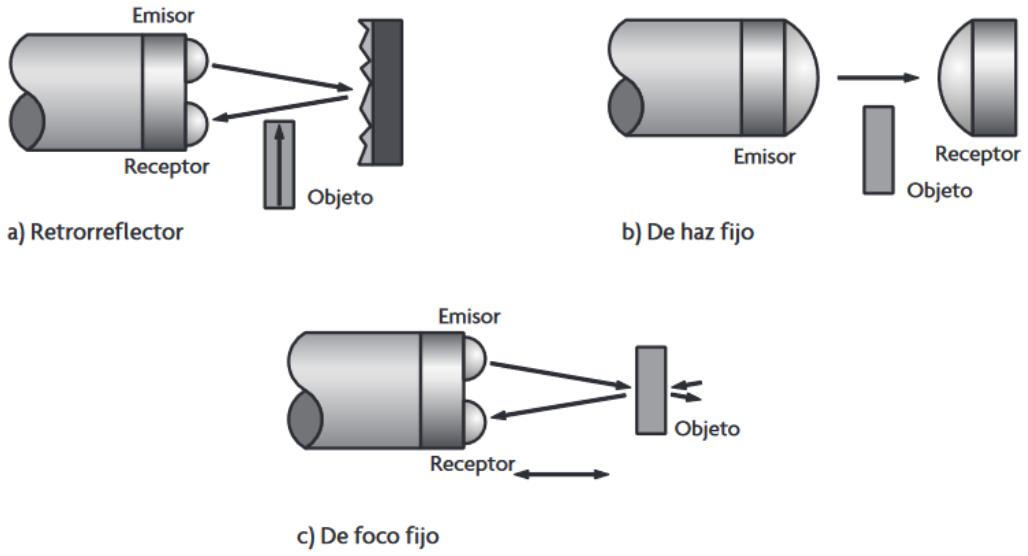


Figura 2.12. Configuraciones para el par emisor receptor

2.3 Visualizadores de texto

Un visualizador es un dispositivo con forma de pantalla destinado a la representación visual de información (REAL ACADEMIA ESPAÑOLA, s.f., definición 2).

La mayoría de los visualizadores de texto se pueden clasificar como reflectivos, transmisivos, o emisivos. Los visualizadores reflectivos dependen de la luz reflejada delante del visualizador. Los transmisivos dejan pasar o bloquean la luz para formar una imagen. Los emisivos emiten luz propia (Teel, 2021).

2.3.1 LCD (Liquid Crystal Display)

Los LCD son usados en relojes digitales, teléfonos celulares, laptops, en algunos televisores y otros sistemas electrónicos. Su principal ventaja sobre otras tecnologías de visualización es que son más ligeros y delgados, y consumen mucha menos energía para operar. La tecnología de los LCD depende de unas propiedades ópticas y eléctricas de un tipo de materiales conocidos como cristales líquidos (Ulaby, Maharbiz, & Furse, 2018).

Es uno de los tipos de visualizadores más utilizados en proyectos de electrónica, por su bajo costo y gran variedad que hay en el mercado.

Entre los diversos tipos de LCD que encontramos en el mercado están los alfanuméricos (figura 2.13). Este visualizador está conformado por una o más filas de celdas con caracteres. Cada celda consiste en un array de tamaño fijo que puede mostrar caracteres o símbolos predefinidos, o definidos por el usuario (Teel, 2021).

Figura 2.13. LCD alfanumérico

2.4 Servicios IoT

IoT, o internet de las cosas, se refiere a la ampliación de la conectividad de red y la capacidad de cómputo a objetos, dispositivos, sensores y elementos que habitualmente no se consideran computadoras (Rose, Eldridge, & Chapin, 2015).

Existen varios modelos de comunicación de la internet de las cosas, la información a continuación está relacionada con en el modelo de comunicación de dispositivo a la nube. En un modelo de comunicación de dispositivo a la nube, el dispositivo de la IoT se conecta directamente a un servicio en la nube, como por ejemplo un proveedor de servicios de aplicaciones (o plataforma IoT) para intercambiar datos y controlar el tráfico de mensajes. Este enfoque suele aprovechar los mecanismos de comunicación existentes (por ejemplo, las conexiones wifi o Ethernet cableadas tradicionales) para establecer una conexión entre el dispositivo y la red IP, que luego se conecta con el servicio en la nube (Rose, Eldridge, & Chapin, 2015).

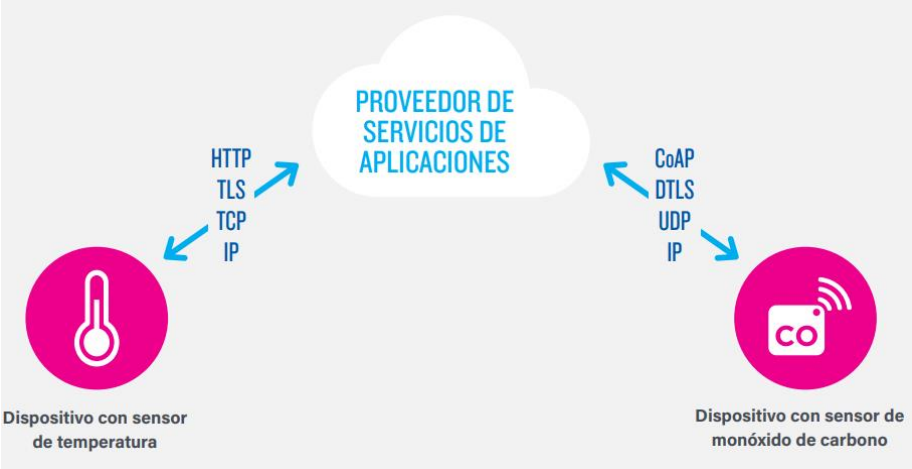


Figura 2.14. Modelo de comunicación de dispositivo a la nube

Junta de Andalucía (2021) define una plataforma de IoT (figura 2.15) como una arquitectura multicapa, que puede conllevar diferentes tecnologías, y que permite la gestión, el aprovisionamiento directo y la automatización de los dispositivos conectados dentro del campo de actuación de Internet de las Cosas. Menciona que las plataformas IoT ofrecen las capas de arquitectura y las tecnologías necesarias para adquirir, tratar, almacenar, analizar y ofrecer datos recogidos de sensores y dispositivos en general.

Las capacidades más importantes que aporta una plataforma IoT son:

- Desarrollo de aplicaciones.
- Gestión de datos a escala.
- Analítica de datos.
- Interoperabilidad e integración.
- Gestión de dispositivos.
- Seguridad.

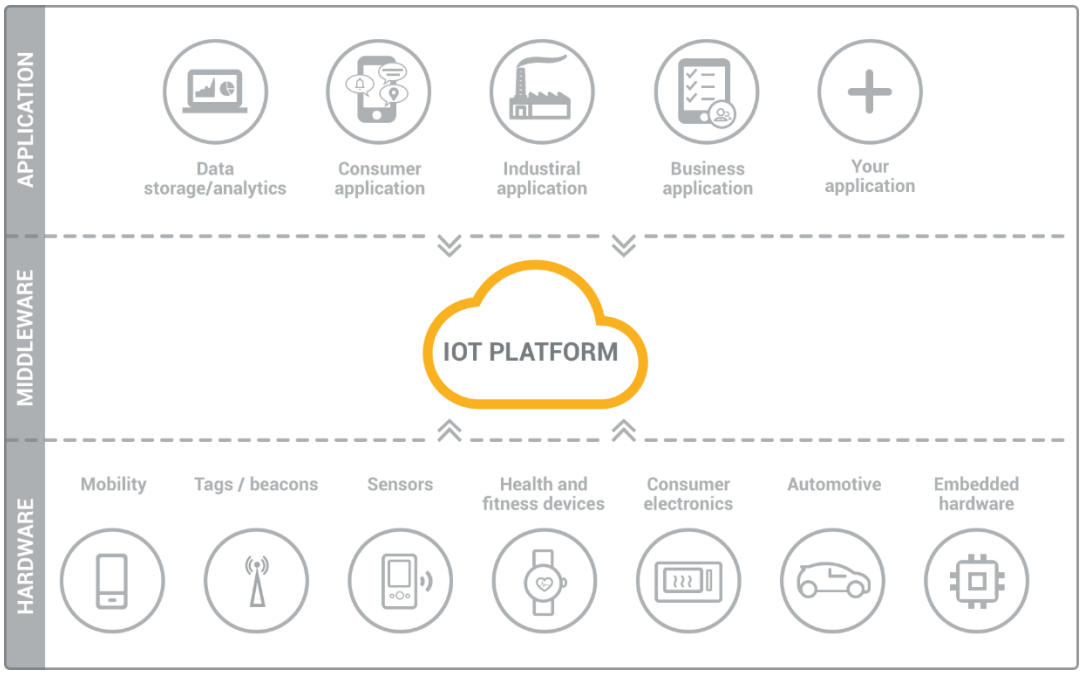


Figura 2.15. Plataforma IoT

Actualmente en el mercado existen una amplia variedad de plataformas IoT, con diferentes capacidades, entre las cuales se encuentran las siguientes.

2.4.1 Arduino IoT

Söderby (2023) menciona que Arduino IoT Cloud es una plataforma online que facilita la creación, despliegue y monitorización de proyectos de la internet de las cosas.

La describe como una platforma que permite crear proyectos IoT a cualquier persona, con una interfaz facil de usar, y una solución todo en uno para configuración, escritura de código, cargar código y vizualización.

Enumera las siguientes capacidades de Arduino IoT Cloud:

- Monitoreo de datos
- Sincronización de variables: la sincronización de varibles permite sincronizar variables entre dispositivos, con lo que se habilita la comunicación entre dispositivos con menos código.
- Programador: programa trabajos para que se activen o desactiven durante un período de tiempo específico (segundos, minutos, horas).
- Cargas Over-The-Air (OTA): carga código a dispositivos que no están conectados a la computadora del desarrollador.
- Webhooks: integra el proyecto con otros servicios, como IFTTT.
- Soporte de Amazon Alexa: hace posible que el proyecto sea controlado por voz con Amazon Alexa.
- Intercambio del Dashboard: comparte los datos con personas de distintas partes del mundo.

2.4.2 Cayenne

myDevices, inc. (2023b) afirma que Cayenne es la primer plataforma, para proyectos IoT, de arrastrar y soltar en el mundo que permite a los desarrolladores, diseñadores e ingenieros crear rápidamente prototipos y compartir sus proyectos de dispositivos conectados. Cayenne se diseñó para ayudar a los usuarios a crear prototipos de Internet de las cosas y luego llevarlos a producción.

Estas son las características, de la plataforma, que lista myDevices, inc. (2023a):

- Panel: el panel es la pantalla principal donde se puede configurar, personalizar, monitorear, administrar y controlar los dispositivos conectados.
- Visualización: almacena datos históricos que permiten ver patrones significativos de comportamiento para ayudar a comprender y guiar mejoras en los proyectos de IoT y los dispositivos y sensores conectados.
- Intercambio: comparte proyectos con otras personas.
- Planificación: crea eventos programados para las placas de desarrollo, sensores y actuadores conectados.
- Alertas: habilita la recepción de notificaciones cuando un dispositivo o sensor ha alcanzado un estado determinado.
- Seguimiento de dispositivos: muestra la ubicación actual y pasada de los dispositivos conectados.
- Código personalizado: se puede escribir código personalizado para el uso de dispositivos que, actualmente, no soporta Cayenne, o para un mayor control sobre el comportamiento de los dispositivos.

2.4.3 AWS IoT

AWS IoT proporciona servicios en la nube que conectan los dispositivos IoT a otros dispositivos y servicios en la nube de AWS. Si los dispositivos se pueden conectar a AWS IoT, AWS IoT puede conectarlos a los servicios en la nube que proporciona AWS, como lo ilustra la figura 2.16 (Amazon Web Services, Inc., 2023a).

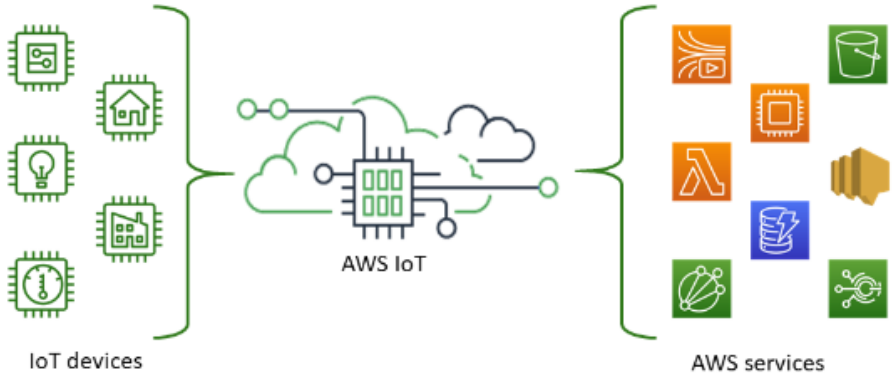


Figura 2.16. AWS IoT

AWS IoT ofrece los servicios mostrados en la figura 2.17. Los servicios tienen tres categorías:

- Software de dispositivo: brindan soporte a los dispositivos de IoT.
- Servicios de control: administran los dispositivos de la solución IoT.
- Servicios de datos: analizan los datos de los dispositivos de la solución IoT (Amazon Web Services, Inc., 2023c).

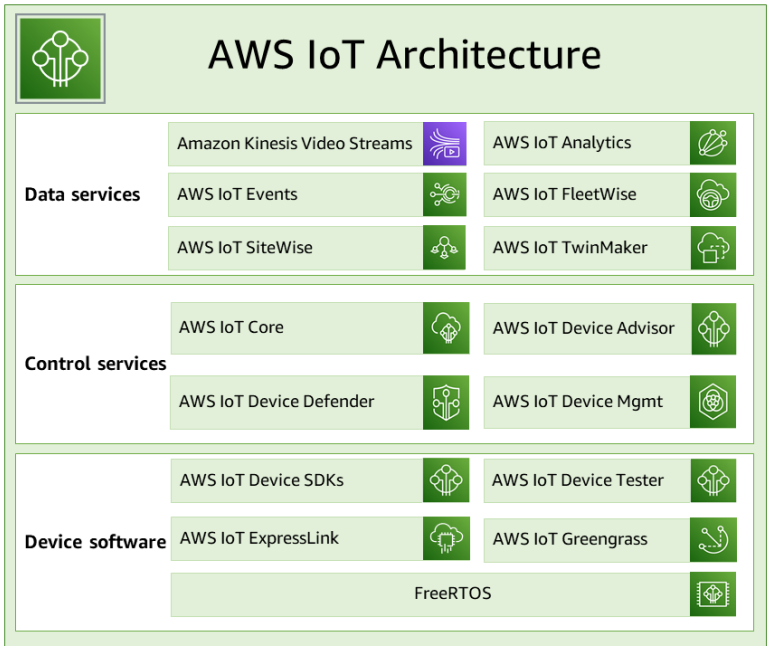


Figura 2.17. Servicios de AWS IoT

2.4.4 Blynk

Blynk es una compañía que ofrece infraestructura para el internet de las cosas (cita). Es pionera en ofrecer la capacidad de construir aplicaciones IoT sin código.

Blynk provee el software necesario para la creación de prototipos, la implementación y la gestión remota de dispositivos electrónicos a cualquier escala.

Ya sea proyectos personales de IoT o millones de productos comerciales conectados, permite a los usuarios conectar su hardware a la nube y crear aplicaciones iOS, Android y web, analizar datos históricos y en tiempo real de dispositivos, controlarlos de forma remota desde cualquier lugar, recibir notificaciones importantes, y mucho más (cita).

La plataforma cuenta con los siguientes componentes:

- **Blynk.Console** (figura 2.17): es una aplicación web con una variedad de funciones que se adaptan a las necesidades de los usuarios. Sus funcionalidades más importantes son:
    1. Configuración de dispositivos conectados a la plataforma.
    2. Gestión de dispositivos, datos, usuarios, organizaciones y ubicaciones.
    3. Monitoreo y control remoto de los dispositivos.

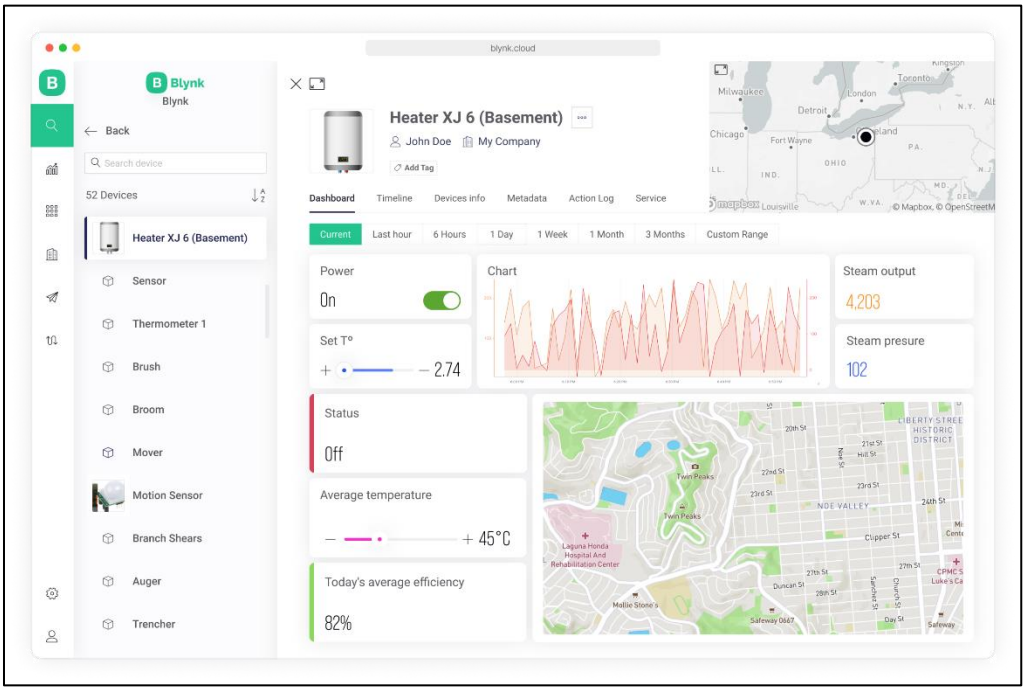


Figura 2.18. Blynk.Console

- **Blynk.Apps** (figura 2.19): es una aplicación móvil (disponible para los sistemas operativos iOS y Android) que provee estas funciones:
    1. Monitoreo y control remoto de los dispositivos, conectados, que trabajan con la plataforma Blynk.
    2. Configuración de la interfaz móvil de usuario durante la etapa de creación de prototipos y producción.
    3. Automatización de operaciones de los dispositivos conectados.

Figura 2.19. Blynk.Apps

- **Blynk.Edgent y librería Blynk**: es una solución diseñada para simplificar la conexión de los dispositivos soportados a la plataforma Blynk, proveyendo acceso a todas sus características avanzadas sin escribir mucho código.
  Las principales características de Blynk.Edgent son:
  1. Solicitud y provisionamiento de wifi para los dispositivos (pone a los dispositivos en línea y los autentica con un usuario).
  2. Gestión de conectividad a redes wifi, móviles y Ethernet.
  3. Intercambio de información entre el dispositivo y la nube.
  4. Integración de API con Blynk.Apps y Blynk.Cloud.
  5. Actualización del firmware Over-the-air en modelos de hardware seleccionados.

  La librería de Blynk está escrita en el lenguaje C++, portable y fácil de usar, que viene preconfigurada para trabajar con cientos de modelos de placas de desarrollo. Implementa un protocolo de conexión de transmisión, lo que permite una comunicación bidireccional y de baja latencia.

  La figura 2.20 muestra un programa que utiliza Blynk.Edgent.

main.cpp

```
// Fill-in information from your Blynk Template here
#define BLYNK_TEMPLATE_ID              ""
#define BLYNK_TEMPLATE_NAME            ""
#define BLYNK_FIRMWARE_VERSION         "0.1.0"
#define APP_DEBUG

#include "BlynkEdgent.h"

void setup()
{
  delay(100);
  BlynkEdgent.begin();
}

void loop() {
  BlynkEdgent.run();
}
```

Figura 2.20. Blynk.Edgent

- **Blynk.Cloud**: es la infraestructura de servidores de Blynk, es el núcleo de la plataforma que une a sus distintos componentes.
  Permite configurar el acceso de los usuarios a los dispositivos y datos, mediante la definición de roles y permisos.

2.5 Dosificadores de alimentos

Los dosificadores son dispositivos usados para regular el despacho de un producto (Garcia Torres, 2006). La mayoría de los dosificadores se conforman de tres partes:

- tolva de almacenamiento;
- sistema dosificador, y
- boquilla o tubo de descarga.

Los dosificadores se pueden clasificar como: de sólidos secos y sólidos en polvo, de líquido, y de gas (Garcia Torres, 2006). Aquí solo se analizarán algunos de los que dosifican sólidos secos y sólidos en polvo.

Los dosificadores de solidos secos y sólidos en polvo se pueden clasificar, a su vez, en:

- rotatorios;
- vibratorios, y
- de válvula

2.5.1 Dosificadores rotatorios

"Los alimentadores rotatorios consisten de un rotor y un motor. El polvo que se encuentra en la tolva de almacenamiento fluye por gravedad hacia el rotor y se descarga a través del giro del rotor" (Bazán Ramírez, 2010).

- **Dosificadores de tornillo**. Como lo menciona Bazán Ramírez (2010) están compuestos de un tornillo, una cubierta en U cilíndrica, y un motor (figura 2.21). Cuando gira el tornillo las partículas son forzadas a moverse desde la tolva hacia el final del alimentador. La dosificación de la sustancia se controla directamente mediante el regulamiento de la velocidad del motor.

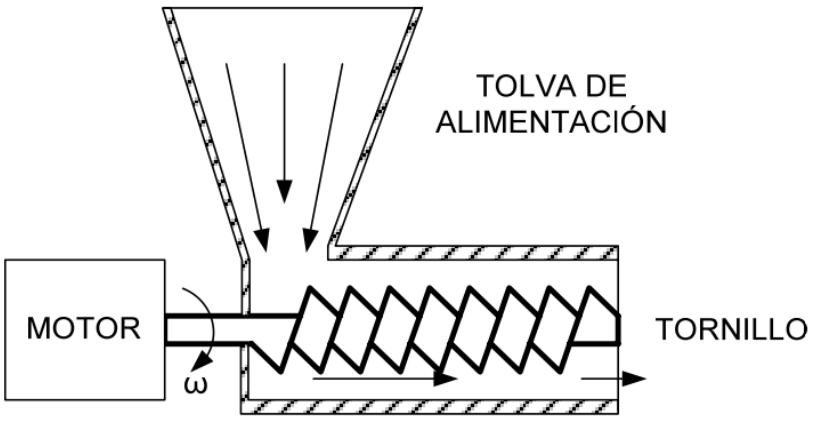


Figura 2.21. Dosificador de tornillo

- **Dosificador de plato.** El dosificador de plato (figura 2.22), dice Bazán Ramírez (2010), consta de un plato giratorio, una paleta estática y un motor. Añade que la velocidad de dosificación es regulada por la posición de la paleta, o por la velocidad angular del plato.

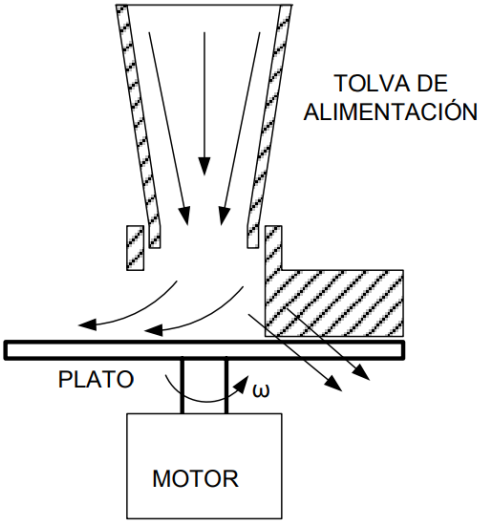


Figura 2.22. Dosificador de plato

- **Dosificador de banda.** Consisten en una banda sin fin (figura 2.23) que arrastra el polvo desde la tolva hasta el final de la longitud de la banda. La velocidad de dosificación se ajusta por la velocidad lineal de la banda, o la apertura de la compuerta de la tolva, pero lo más recomendable es variar la velocidad de la banda (Bazán Ramírez, 2010).

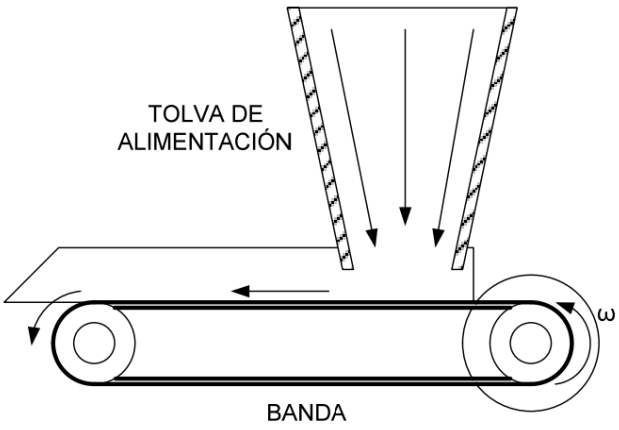


Figura 2.23. Dosificador de banda

- **Dosificador de cono.** Es usado para proveer un flujo libre constante en cascada de materiales granulados, o pellets, y algunos polvos. Está formado de un cono horizontal rotatorio (figura 2.24), con una tolva integrada al diámetro mayor del cono para el abastecimiento del material, así la descarga se realiza por el diámetro menor del cono (Bazán Ramírez, 2010).

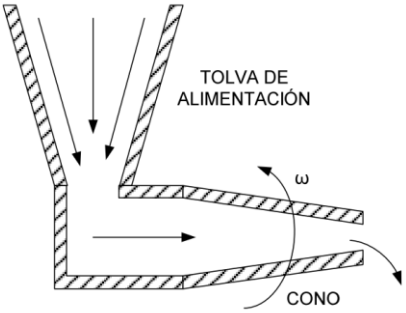


Figura 2.24. Dosificador de cono

2.5.3 Dosificadores vibratorios

Se de definen como aquellos que "utilizan un control electromecánico, o electromagnético, para generar la vibración que transporte de manera suave el polvo por un canal inclinado (figura 2.25). La vibración se selecciona lo más cercano posible a la frecuencia de resonancia" (Bazán Ramírez, 2010).

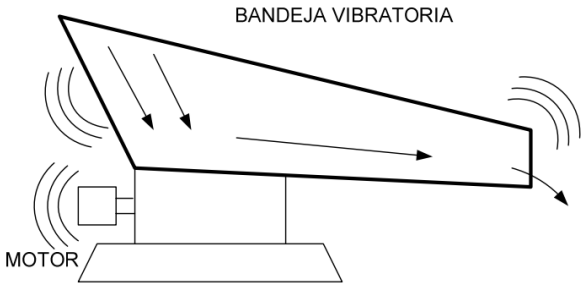


Figura 2.25. Dosificador vibratorio

2.5.4 Dosificadores de válvula

Se utilizan para el control de flujo libre de polvo. Se utilizan normalmente válvulas de paso (figura 2.26), compuertas de corte, válvulas de aleta, interruptores vibratorios o cerradura de tolvas para regular la magnitud de flujo (Bazán Ramírez, 2010).

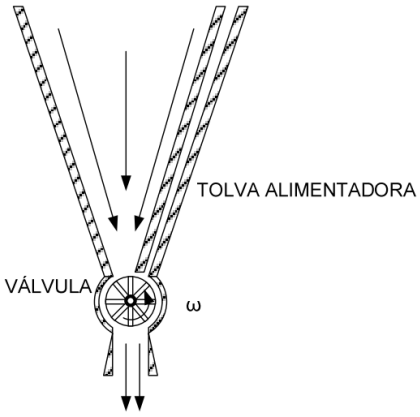


Figura 2.26. Dosificador de válvula

2.6 Motores de CD

Los motores de CD son máquinas que convierten energía eléctrica de cd en energía mecánica (Chapman). En las siguientes secciones se estudiarán algunos de los motores de CD que existen.

2.6.1 Servomotores de corriente directa

Es uno de los actuadores más utilizados en la integración de sistemas. En su interior contiene un motor con un reductor de velocidad y multiplicador de fuerza, y un circuito de control. En la mayoría de estos actuadores, el ángulo de giro es de 180°, pero puede ser modificado para tener un giro de 360°. La figura 2.27 muestra los componentes de un servomotor.

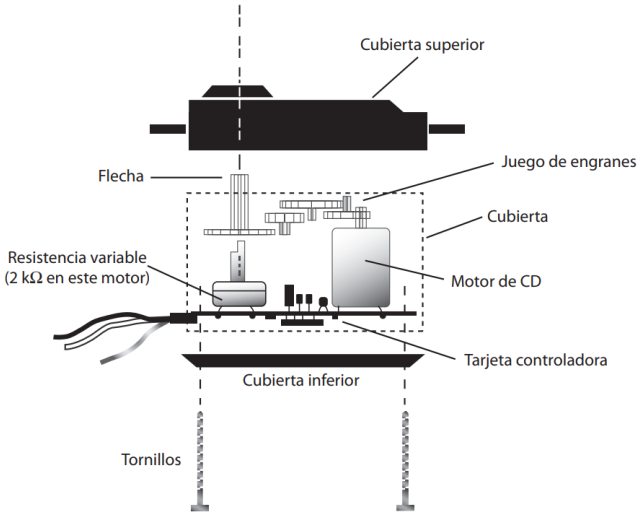


Figura 2.27. Componentes de un servomotor

Para el control del servomotor debe de aplicarse un pulso de duración y frecuencia específico. Por lo general, los servomotores disponen de tres cables, dos para la alimentación Vcc y GND (4.8 a 6 V), y un tercero para aplicar una secuencia de pulsos de control. Dependiendo del ancho de pulso, el circuito de control diferencial lleva al servomotor a la posición indicada. En la figura 2.28 se ilustran algunos ejemplos del posicionamiento del servomotor dependiendo del ancho de pulso.

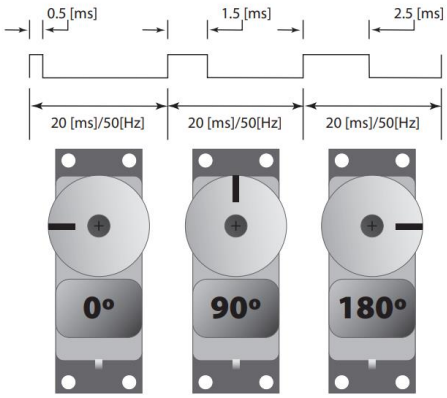


Figura 2.28. Posiciones del eje de un servomotor

Una de las técnicas más empleadas para el control de servomotores es la modulación por la por ancho de pulso, PWM (Pulse Width Modulation). Consiste en generar una onda cuadrada en la que se varía el tiempo que el pulso está a nivel alto, manteniendo el mismo periodo, con el objetivo de modificar la posición del eje según se desee.

El servomotor tiene márgenes de operación que se corresponden con el ancho del pulso máximo y mínimo. Los valores más generales se corresponden con pulsos de entre 1 ms y 2 ms de anchura, que sitúan al motor en ambos extremos (0º y 180º). Es importante hacer notar que valor de 1.5 ms indica la posición central o neutra (90º), mientras que otros valores del pulso lo sitúan en posiciones extremas.

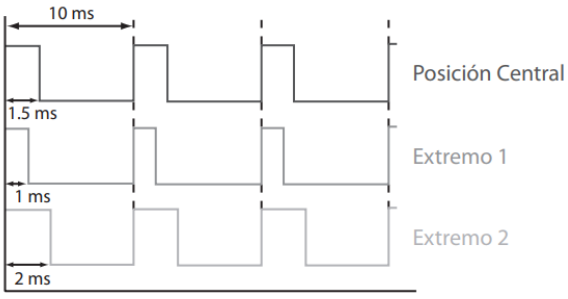


Figura 2.29. Tren de pulsos para el control de un servomotor

Para que un servomotor se mantenga en la misma posición es necesario suministrar en forma continua el pulso correspondiente. De este modo, si existe alguna fuerza externa que lo obligue a abandonar esta posición, el esquema de control corregirá el error (Corona Ramírez, Abarca Jiménez, & Mares Carreño, 2014).

2.6.2 Motores a pasos

Los motores a pasos (figura 2.30) son motores sin escobillas controlados digitalmente que giran un numero especifico de grados (pasos) cada que un pulso de reloj es aplicado a un circuito convertidor especial que es usado para controlar el motor. La cantidad de grados por paso (resolución) para un determinado motor a pasos puede ser tan pequeña como 0.72° por paso o hasta 90° por paso.

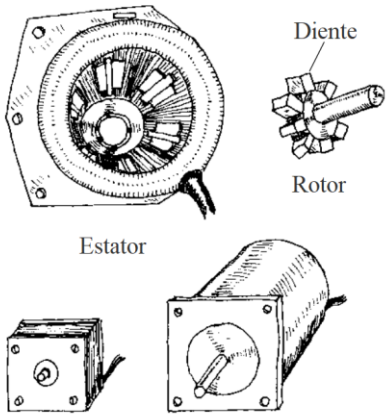


Figura 2.30 Motores a pasos

A diferencia de los servomotores, el eje de los motores a pasos puede girar hasta 360° y se puede hacer girar de forma continua como un motor de corriente directa (pero con una velocidad máxima menor) con la ayuda del circuito de control digital adecuado. Los motores a pasos proveen un torque mayor a bajas velocidades, haciéndolos ideales para aplicaciones donde se necesita una baja velocidad y control de alta precisión. Son utilizados, por ejemplo, en impresoras para controlar el ingreso del papel y para mover telescopios.

Para darnos una idea básica del funcionamiento del motor a pasos, observe la figura 2.31. Se muestra un motor a pasos de 15° por paso de reluctancia variable. La parte fija del motor, llamada estator, tiene ocho polos separados 45°. La parte móvil del motor, llamada rotor, está hecha de un material ferromagnético (un material que es atraído por campos magnéticos), tiene seis dientes separados 60° entre ellos. Para hacer girar el motor un paso, se aplica corriente, al mismo tiempo, a través de dos polos opuestos, o bobinas pares. La corriente aplicada ocasiona que el par opuesto de polos se magnetice. Esto también causa que los dientes del rotor se alineen con los polos como lo muestra la figura 2.30. Para hacer que el rotor gire 15° en dirección de las manecillas del reloj, desde esta posición, se remueve la corriente en el par de bobinas 1 y es transmitida por el par 2. Para hacer girar el eje del motor otros 15° en dirección de las manecillas del reloj, desde esta posición, se remueve la corriente en el par de bobinas 2 y es transmitida por el par 3. El proceso continúa de esta manera. Para hacer girar el rotor en sentido contrario de las manecillas del reloj, la secuencia descrita anteriormente es revertida (Scherz, 2000).

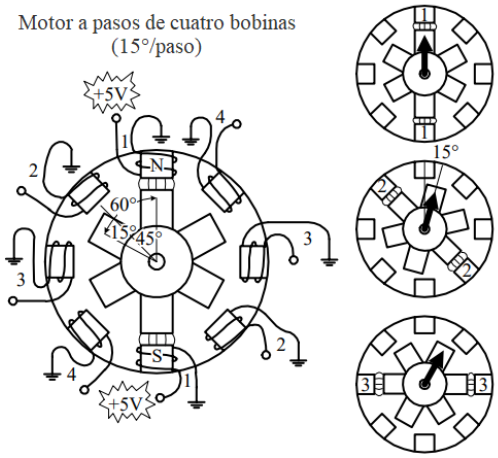


Figura 2.31. Funcionamiento del motor a pasos de reluctancia variable

2.6.3 Motorreductor de corriente continua

Un motorreductor es una máquina que combina un reductor de velocidad y un motor de corriente continua. La adición del reductor de velocidad a el motor disminuye la velocidad y aumenta el torque en la salida. Los parámetros más importantes en lo que corresponde a motorreductores son: velocidad angular (rpm), torque (Nm) y eficiencia (%). Para seleccionar el motor más conveniente para nuestra aplicación, se debe de calcular la carga, velocidad y el torque necesario.

Figura 2.32. Motor con engranes reductores.

Algunas ventajas de los motorreductores de CC son:

- Incremento de torque. El mecanismo del reductor de velocidad incrementa el torque en la salida, lo que permite mover mayores cargas que requieren más torque del que el motor solo puede generar.
- Disminución de la velocidad. El mecanismo del reductor de velocidad también reduce la velocidad angular de la flecha del motor. Esto puede ser una ventaja en aplicaciones donde las altas velocidades no son requeridas o deseables.
- Transmisión de potencia más eficiente. El mecanismo del motorreductor puede transmitir potencia más eficientemente que un motor de transmisión directa.
- Tamaño y diseño compactos.
- Mayor flexibilidad. La capacidad de ajustar el reductor de velocidad para proveer de las condiciones de operación deseadas permite un mayor control sobre el par y la velocidad de salida (ISL Products International, 2023).

CAPÍTULO 3. DESARROLLO EXPERIMENTAL

El propósito de este trabajo es crear un dispositivo dispensador de croquetas que pueda ser configurado por el usuario desde una aplicación móvil y/o web. La tarea principal del dispensador es: entregar alimento en las cantidades y horarios que el usuario seleccione en la aplicación. Además, muestra al usuario el nivel de alimento disponible en el contenedor, y manda una notificación cuando este sea bajo.

El dispensador muestra en un visualizador la hora actual y las configuraciones que tiene guardadas. Tiene un botón de *reset*, para restablecer el sistema, y un botón de *boot*, para restablecer las credenciales del Access Point inalámbrico al que se conecta el dispensador. Para establecer la comunicación del dispositivo con los servidores, se proporcionan las credenciales, mediante la aplicación móvil, del Access Point con acceso a internet al que se va a conectar el dispositivo. Una vez establecida la conexión, el usuario puede seleccionar la hora de entrega y la cantidad de alimento a servir por medio de la aplicación móvil y/o web, se envía la información al dispensador por medio de la plataforma Blynk IoT, y se guarda en el sistema. Para activar la entrega de alimento, continuamente se consulta al reloj de tiempo real del sistema (RTC) con los horarios definidos por el usuario, de haber coincidencia se activa el dosificador hasta suministrar la cantidad de alimento correspondiente a ese horario, esto siempre y cuando exista alimento disponible en el dispensador. Las aplicaciones permiten la configuración de hasta tres horarios y sus respectivas raciones.

Para cuantificar la cantidad de alimento, se incorporaron cuatro sensores infrarrojos, que de acuerdo a su lectura, se establece nivel alto, medio, bajo y no disponible, aunque en este último, por la ubicación del sensor, existe aún algo de alimento. Esta información es presentada en la aplicación para que el usuario conozca el estatus de disponibilidad, cuando ya se encuentra en no disponible, el usuario recibe una notificación de dicha condición.

En este capítulo se explicarán las principales partes del sistema y el proceso llevado a cabo para su implementación y puesta en marcha. Se mostrará el proceso diseño del circuito del sistema y todo lo que conlleva: el planteamiento del esquema electrónico, el diseño y fabricación de la placa de circuito impreso, y el montaje de los componentes. También, se describirá el diagrama de flujo implementado y se expondrán las partes más importantes de su programación. Se presentará el diseño de la aplicación móvil y web en la plataforma Blynk IoT, y las configuraciones realizadas. Por último, se mostrará el procedimiento del diseño y montaje del prototipo del dispensador, y las partes necesarias para su correcto funcionamiento.

3.1 Diagrama de bloques

En la figura 3.1 se muestra un diagrama de bloques de las etapas del sistema dispensador de alimento y sus componentes.

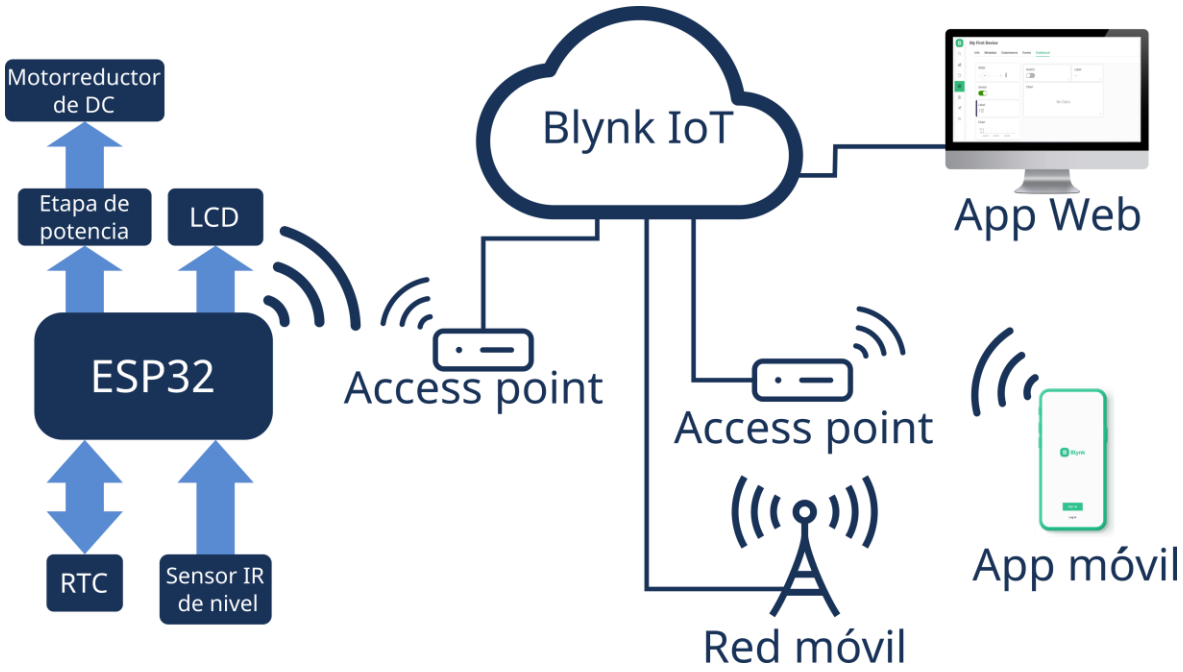


Figura 3.1. Diagrama de bloques del dispensador de alimentos

Los principales componentes del sistema son:

- **LCD**. El dispensador cuenta con un visualizador LCD alfanumérico de 20x4 caracteres en el que se muestra: la hora actual, las horas a las que se servirán las raciones de alimento y el tamaño de la ración.
- **Botones de *boot* y *reset.*** Tiene un botón de *reset* para reiniciar el sistema y un botón de *boot* para actualizar las credenciales de la red wifi.
- **Módulo sensor infrarrojo.** Se utilizó un módulo sensor de presencia infrarrojo para el sensado del nivel de alimento, que está conformado por cuatro sensores infrarrojos colocados a distintos niveles del contenedor de croquetas para detectar su presencia. El módulo tiene un circuito comparador de voltaje LM399, que convierte las señales que entregan los sensores a voltajes estables de 0 o 3.3 V, y un potenciómetro en cada canal para ajustar la sensibilidad de los sensores.
- **Módulo RTC.** Se trata del DS3231, se optó por un RTC externo y no por el que está integrado el microcontrolador, para que el sistema tenga una mayor autonomía.
- **Etapa de potencia.** Es el circuito que se usó para la activación del motorreductor, se utilizó un relé de 5 V, al que se le tuvo que amplificar el voltaje de salida del microcontrolador, debido a que este da un voltaje de 3.3 V, que es insuficiente para accionar los contactos del relé.

- **Motorreductor de CD.** Es un motorreductor de 12 V de corriente directa, con una velocidad angular de 10 RPM. Este motor tiene el torque (70 kg/cm) para accionar el tornillo sin fin del dispensador sin que ocurran atascos.
- **Blynk IoT.** Es la plataforma de internet de las cosas utilizada, está plataforma nos permite diseñar aplicaciones móviles y web; además, de proporcionarnos la infraestructura de servidores para la implementación del sistema. Otra capacidad que se aprovechó es la gestión de la conexión del dispositivo que ofrece Blynk mediante Blynk.Edgent. Y otras características más que serán explicadas en la sección 3.3.
- **Aplicación móvil.** Es la interfaz en la que se selecciona la cantidad de alimento que debe entregar el dispensador y los horarios de entrega, muestra la cantidad de alimento disponible y muestra una notificación cuando el nivel sea bajo.
- **Aplicación web.** Permite la configuración de las raciones y muestra la cantidad de alimento disponible.
- **Etapa de control.** Se utilizó un microcontrolador ESP32 para la etapa de control y comunicación del sistema. Las razones por las que fue elegido este microcontrolador son: que incorpora comunicación wifi y es compatible con la plataforma Blynk IoT.

3.2 Diagrama esquemático

En esta sección se presenta el diagrama de conexiones general (figura 3.2) del sistema.

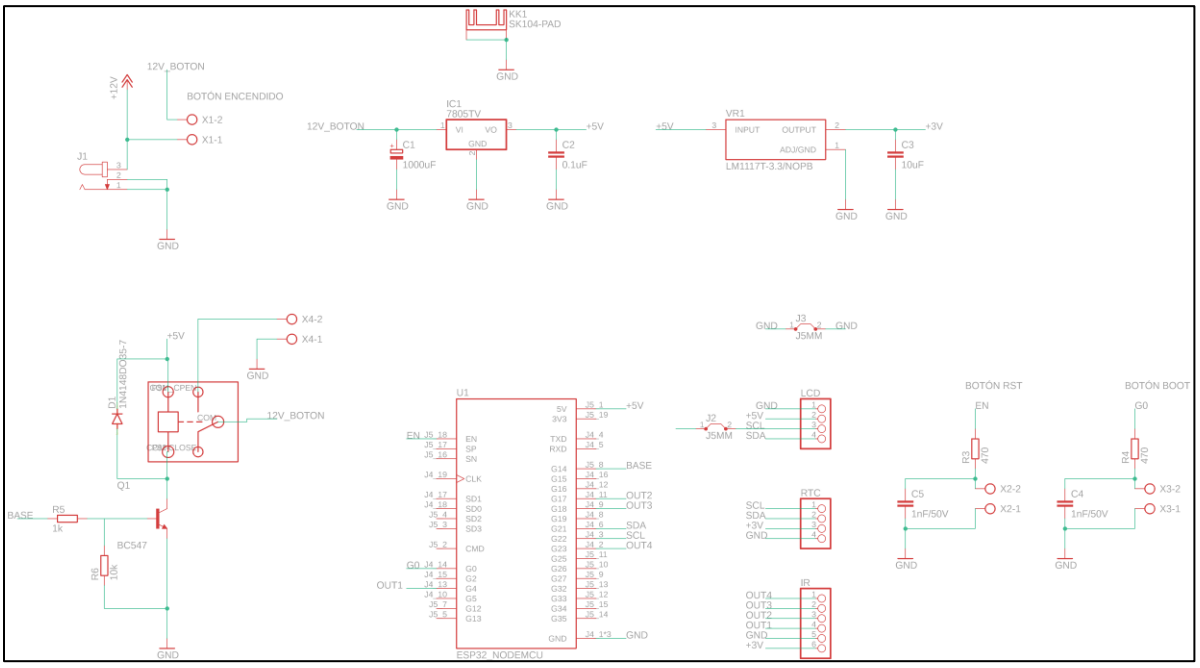


Figura 3.2. Diagrama esquemático del sistema

A continuación se explicará el diagrama del circuito diseñado, para ello se mostraran segmentos de bloques funcionales para una mayor claridad. En la figura 3.3 se muestra el circuito usado para energizar el sistema, está compuesto por un conector DC jack hembra de 2.5 mm, al que se le conecta un adaptador AC/DC de 12 V y 2 A. También, tiene una bornera en donde se conecta el *switch* de encendido. Se utilizó un adaptador de 12 V para alimentar al motorreductor.

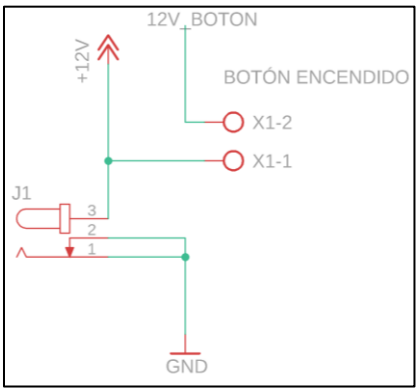


Figura 3.3. Circuito del conector para la alimentación y del botón de encendido

La placa de desarrollo del microcontrolador ESP32 utilizada tiene un regulador de voltaje AMS1117 que soporta 12 V a la entrada y entrega 3.3 V a la salida con los que alimenta al microcontrolador. A pesar de lo anterior, se optó por emplear un regulador LM7805 (parte izquierda de la figura 3.4) para reducir el voltaje de 12 a 5 V, y así evitar el sobrecalentamiento por la disipación de calor en el encapsulado más pequeño del regulador de la placa. Los 5 V también se usan para alimentar al LCD, y para activar al relevador.

Además se adicionó el regulador LM1117 (parte derecha de la figura 3.4) que reduce el voltaje de 5 V a 3.3 V. Se evadio usar el de la placa de desarrollo, para evitar su sobrecalentamiento. Este regulador alimenta a los módulos RTC y los sensores de presencia infrarrojos.

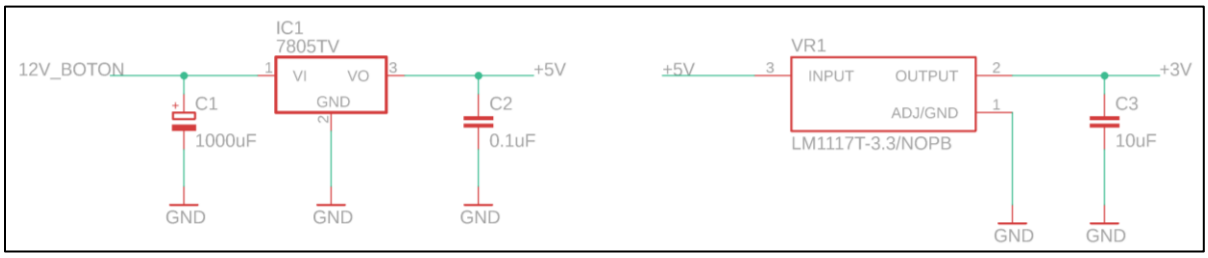


Figura 3.4. Circuito regulador de voltaje de 12 V a 5 V (izquierda) y de 5 a 3.3 V (derecha)

Los módulos RTC y del visualizador LCD se comunican con el microcontrolador mediante el protocolo I2C; por lo tanto, ambos se conectaron a los pines G21 y G22 del microcontrolador, que corresponden a los pines de SDA (Serial Data) y SCL (Serial Clock) respectivamente. Para energizar el microcontrolador se conectó el pin 5V a la salida del regulador LM7805. Se conectaron las cuatro salidas del módulo sensor infrarrojo (OUT1, OUT2, OUT3 y OUT4) a los pines de propósito general (G4, G17, G18 y G23) para la entrada de los valores digitales de presencia o no presencia de alimento. El pin G14 es la salida que va conectada a la base del transistor del circuito para la activación del relé. El pin EN va conectado al botón de *reset*, el cual reinicia al microcontrolador cuando es presionado. El pin G0 se conecta al botón de *boot*, si se mantiene presionado diez o más segundos se borra la configuración de red (SSID y contraseña).

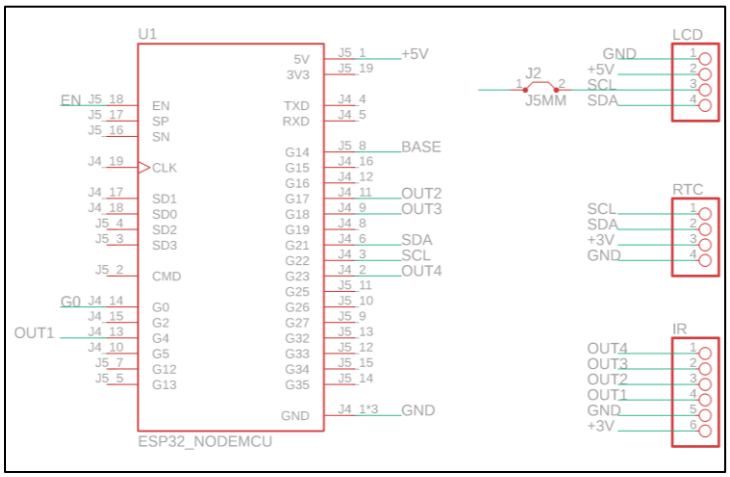


Figura 3.5. Circuito del microcontrolador y pines para la conexión de los módulos de sensores infrarrojos, reloj de tiempo real y visualizador LCD.

El microcontrolador provee un voltaje de 3.3 V en un GPIO (pin de entrada y salida de proposito general , por sus siglas en inglés) cuando se configura como salida. El relevador utilizado acciona su contactor cuando se conecta un voltaje de 5 V en la bobina del electroiman. Por lo expuesto anteriormente es necesario amplificar el voltaje del GPIO. En el circuito de la figura 3.6 se uso un transistor BC547 como conmutador para alimentar la bobina con 5 V cada que se aplique un voltaje de 3.3 V en el nodo BASE. Se conectó un 1N4148 como diodo de retorno para evitar los picos de voltaje que ocurren cuando se interrumpe la alimentación de la bobina del relé.

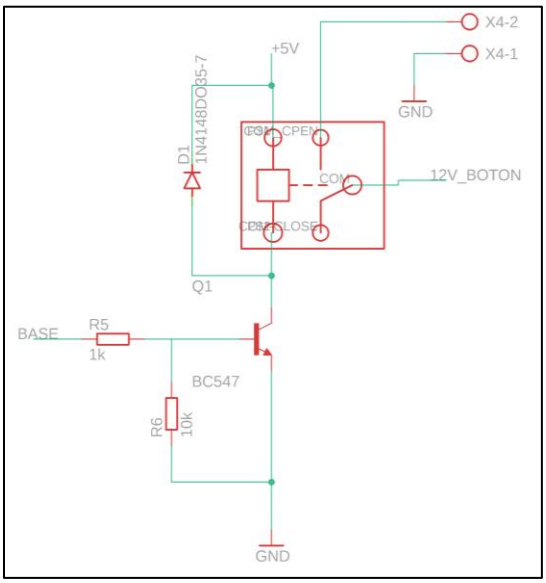


Figura 3.6. Circuito amplificador para relé de 5V

Para evitar el "rebote" en los botones de *reset* y *boot*, se utilizó un capacitor (figura 3.7), conectado en paralelo a los botones, para filtrar las transiciones indeseadas que existen cuando se presiona el botón.

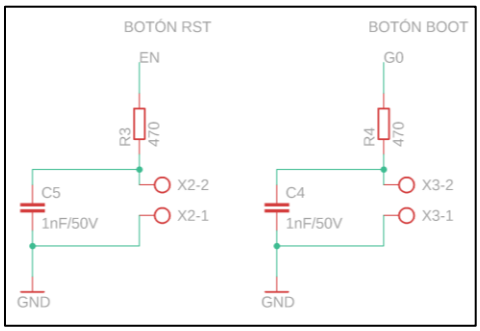


Figura 3.7. Circuito para la conexión de los botones de boot y reset

3.3 Diagrama de flujo y programación

El microcontrolador utilizado cuenta con dos procesadores que pueden trabajar en paralelo, los cuales fueron utilizados, por lo que en el diagrama de flujo (figura 3.8) se muestran dos procesos trabajando simultáneamente.

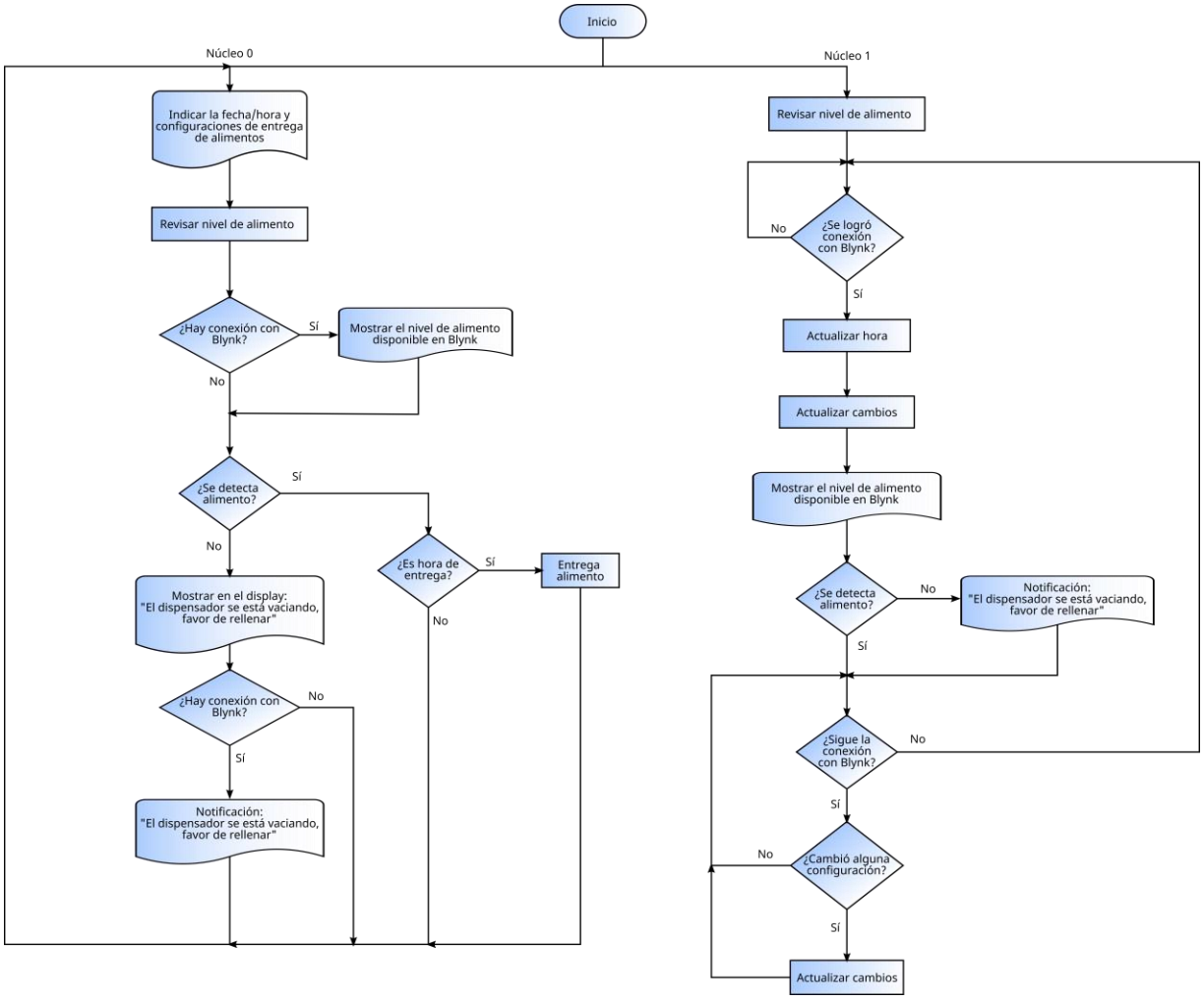


Figura 3.8. Diagrama de flujo

Para la explicación se inicia indicando las acciones realizadas por el núcleo 1. Lo primero que realiza el sistema es el sensado del nivel de alimento y se invoca al BlynkEdgent, como lo muestran las líneas de código 150 y 154.

```
150     nivel = (!digitalRead(PIN_IR1) + !digitalRead(PIN_IR2) +
!digitalRead(PIN_IR3) + !digitalRead(PIN_IR4)) * 25;
151
152     if(xBinarySemaphore != NULL)
153     {
154         BlynkEdgent.begin();
```

Con ello, un núcleo intentará comunicarse con la plataforma de Blynk, de no lograrse lo seguirá intentando reiteradamente, de lograse se hará una lectura de la hora proporcionada por la plataforma de Blynk, estos datos se cargan al RTC del sistema, asegurando con ello mantener esta información de manera precisa. En las líneas de código de la 107 a la 110 se muestra cómo se actualiza la hora del RTC. Se renuevan los valores de las variables asociadas de hora de entrega y raciones con los últimos valores que tengan guardados los servidores de Blynk, esto se hace para recibir las configuraciones, aunque se pierda conexión. Gracias a que se hizo un primer sensado del nivel de alimento, es posible enviar este a la plataforma de Blynk para que sea mostrado en la aplicación móvil y web. En caso de que no se detecte alimento, le comunica a la plataforma para que envíe una notificación a la aplicación móvil.

```
107 BLYNK_WRITE(InternalPinRTC) {   //revisa el valor de InternalPinRTC
108     time_t t = param.asLong();      //guarda el valor en la variable t
109
110     RTC.adjust(DateTime(t));    //configura la hora en el RTC
```

Estas acciones solo ocurren una vez establecida la conexión, posteriormente se continúa verificando, y si el usuario hizo algún cambio de configuración (hora y cantidad de alimento), esta información es leída y las variables asociadas son actualizadas. La siguiente porción de código ilustra cómo se actualizan.

```
53 BLYNK_WRITE(V0)      //obtiene el valor de los sliders(número de vueltas)
54 {
55     int vueltas = param.asInt();
56     EEPROM.write(1, vueltas);  //los guarda en la memoria flash de la esp32
57     EEPROM.commit();
58 }

74 BLYNK_WRITE(V9)      //obtiene el valor de la hora ingresada por el usuario
75 {
76     int hora, minuto;
77     String HoraEntrega1 = param.asString(); //hora en segundos a partir de
las 00:00
78     long HoraEntrega = HoraEntrega1.toInt();
79     segundos_a_horas(&hora, &minuto, HoraEntrega);//conversión de segundos
a horas
80     EEPROM.write(2, hora);
81     EEPROM.write(3, minuto);          //guarda la configuración en la
memoria flash de la esp32
82     EEPROM.commit();
83 }
```

En el núcleo 0 el flujo del programa empieza mostrando en el LCD la hora actual, provista por el RTC, y las configuraciones de entrega que tiene guardado el microcontrolador en la memoria flash (líneas de la 195 a la 204).

```
195         DateTime now = RTC.now();
196         lcd.setCursor(6, 0);
197         lcd.printf("%02d:%02d:%02d", now.hour(), now.minute(),
now.second());
198
199         lcd.setCursor(0, 1);
200         lcd.printf("P1: %2d Vts %02d:%02d Hrs", EEPROM.read(1),
EEPROM.read(2), EEPROM.read(3));
201         lcd.setCursor(0, 2);
202         lcd.printf("P2: %2d Vts %02d:%02d Hrs", EEPROM.read(5),
EEPROM.read(6), EEPROM.read(7));
203         lcd.setCursor(0, 3);
204         lcd.printf("P3: %2d Vts %02d:%02d Hrs", EEPROM.read(9),
EEPROM.read(10), EEPROM.read(11));
```

Se revisa el nivel de alimento que hay en el depósito (líneas de la 273 a la 275), si existe conexión con los servidores de Blynk, se actualiza el nivel mostrado en la aplicación móvil y web (líneas de la 278 a la 281). Si el nivel de alimento es menor al 25%, entonces, se muestra una alerta en el visualizador LCD (líneas de la 208 a la 223) y, si hay conexión con Blynk, también se le avisa para que envíe una alerta (líneas de la 294 a la 300).

```
273           nivel   =   (!digitalRead(PIN_IR1)   +   !digitalRead(PIN_IR2)   +
!digitalRead(PIN_IR3) + !digitalRead(PIN_IR4)) * 25; //sensado del nivel de
alimento
274
275     presenciaActual = (nivel > 0) ? 1 : 0;    //bandera que indica la
presencia de alimento
276
277     if (temp != nivel)                        //solo se actualiza cuando el
nivel actual es distinto al anterior
278     {
279         Blynk.virtualWrite(15, nivel);        //envía el nivel de alimento a
Blynk
280         temp = nivel;
281     }

208         while (presenciaActual == 0)
209         {
210             lcd.clear();
211             lcd.setCursor(0, 0);
212             lcd.print("El dispensador se");
213             lcd.setCursor(0,1);
214             lcd.print("esta vaciando,");
215             lcd.setCursor(0, 2);
216             lcd.print("favor de rellenar");
217             lcd.noBacklight();
218             vTaskDelay(pdMS_TO_TICKS(1000));
219             lcd.backlight();
220             vTaskDelay(pdMS_TO_TICKS(1000));
```

```
221             notificacion();
222             lcd.clear();
223         }
```

```
294       if((presenciaActual==0) && (presenciaAnterior==1))
295       {
296         Blynk.virtualWrite(1, 1);  //notificar sin alimento
297         Blynk.virtualWrite(1, 0);    //restablece el valor de la variable
usada para notificar
298       }
299       presenciaAnterior = presenciaActual;
300     }
```

Si el nivel es mayor al 25% se comparar la hora de entrega con las horas de entrega que ha configurado el usuario, si son iguales se activa el motor el tiempo necesario para despachar la cantidad de alimento que el usuario haya seleccionado. Como lo ilustran las siguientes líneas de código.

```
221         if((EEPROM.read(0) == 1 && now.hour() == EEPROM.read(2) &&
now.minute() == EEPROM.read(3) && ((now.second() == 0) || (now.second() ==
1))) && (presenciaActual == 1))
222         {
223             lcd.clear();
224             digitalWrite(PIN_MOTOR, HIGH);
225             lcd.setCursor(1, 0);
226             lcd.print("Entregando alimento");
227             Vuelta(EEPROM.read(1));
228             digitalWrite(PIN_MOTOR, LOW);
229             lcd.clear();
230         }
```

3.4 Diseño y fabricación de la PCB

Se diseñó la placa de circuito impreso con el software Eagle (figura 3.9). Se utilizó una placa de una capa de cobre de 10 x 10 cm. Se seleccionaron pistas más anchas en las partes que, se consideró, conducen más corriente, por ejemplo, en la alimentación de los reguladores, sensor infrarrojo, microcontrolador, RTC y en los contactores del relé. Se configuró una separación entre pistas que facilitara su fabricación en un *router* CNC. Por medio de la herramienta de polígono, se usó el área de cobre que hay entre las pistas como nodo común (GND) para conectar a los componentes que lo requieren.

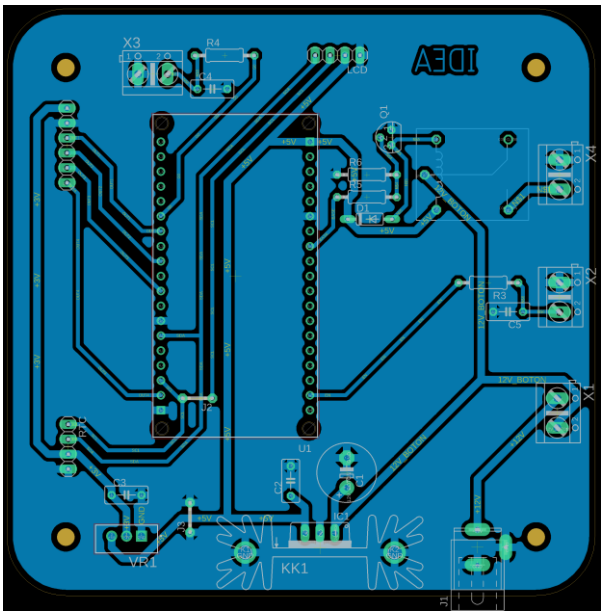


Figura 3.9. Diseño de la placa en el software Eagle

Para la fabricación de la placa se usó un *router CNC*, por lo que se tuvo que usar el software FlatCAM para convertir los archivos Gerber a archivos G-Code, que son los que se utilizan para darle instrucciones de corte, y Excellon, para las perforaciones. Se generaron archivos G-Code para el corte de las pistas y de la placa (figura 3.10). Para las perforaciones (figura 3.11) se generaron dos archivos Excellon, uno para hacer perforaciones de 0.8 mm y otro para perforaciones de 1.0 mm.

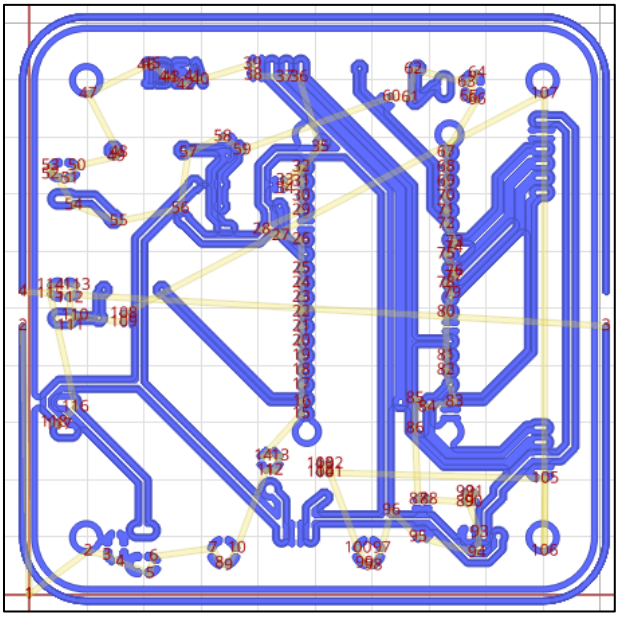


Figura 3.10. Gráfico generado por FlatCAM que ilustra en color azul los cortes del router CNC

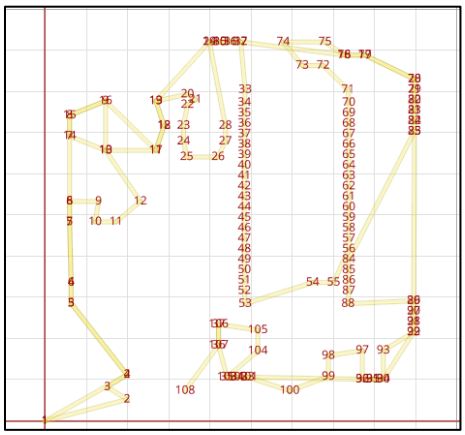


Figura 3.11. Gráfico generado por FlatCAM que ilustra las perforaciones que va a hacer el router

Una vez el *router* terminó su trabajo se procedió a soldar los componentes (figura 3.12). La figura 3.13 muestra la placa terminada.

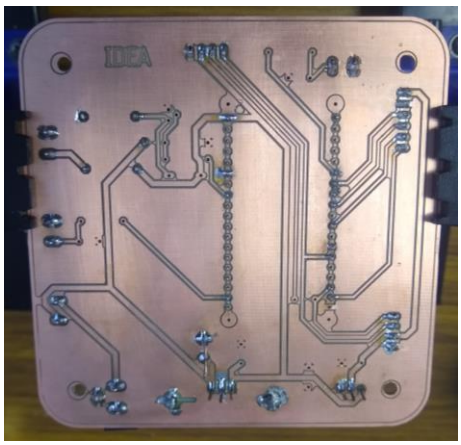


Figura 3.12. Soldado de los componentes a la placa

Figura 3.13. Circuito terminado en funcionamiento

3.5 Caja para circuito

Para contener el circuito impreso y los demás componentes, se diseñó e imprimió una caja en 3D. El proveedor de software Autodesk nos proporciona la facilidad de crear un modelo 3D, en el programa Fusion 360, de una placa de circuito impreso diseñada en Eagle. Esta característica nos permite dedicarle menos tiempo al modelado porque podemos añadir el modelo de la placa en otro y trabajar con base en ella.

En la figura 3.14 se muestra la caja diseñada para acomodar el cableado, la placa de circuito impreso, botones y módulos. El grosor de la caja es de 2 mm para obtener un mejor resultado en la impresora 3D; se agregaron rendijas para mejorar la circulación de aire en el circuito; se modelaron perforaciones para colocar los botones, para el paso del cableado de alimentación y de los sensores, y para los tornillos con los que se fijó la tapa. En la tapa (derecha de la figura 3.14) se hicieron perforaciones para el visualizador y el interruptor de encendido. La figura 3.15 es una fotografía del resultado final de la caja con sus componentes ordenados.

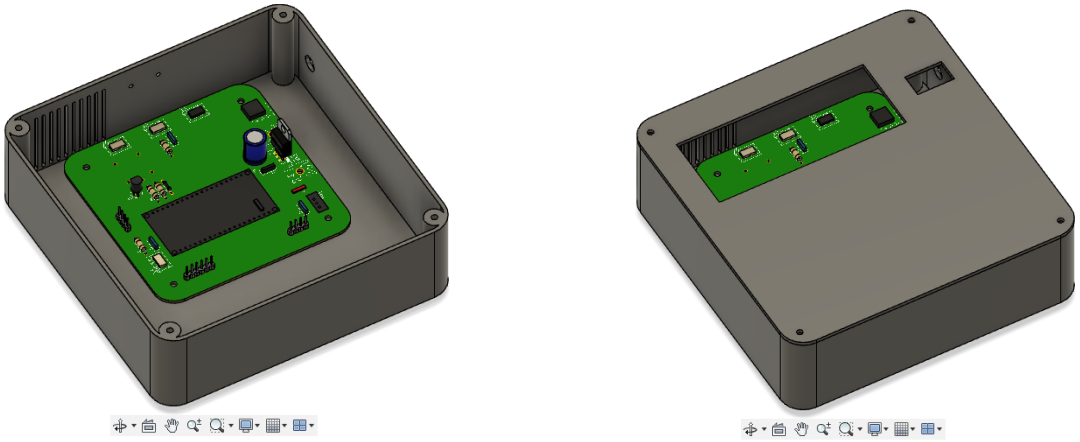


Figura 3.14. Caja para contener el circuito

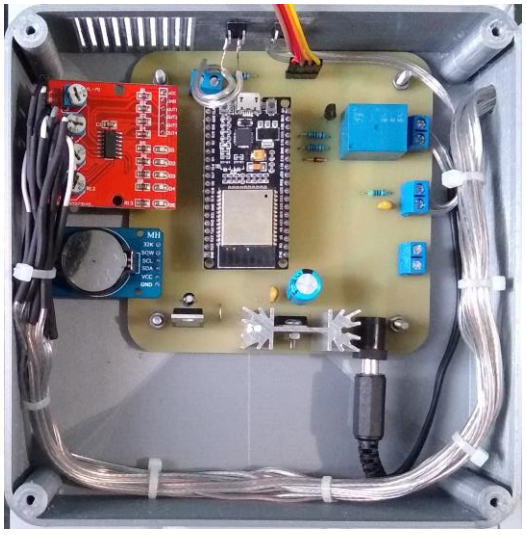


Figura 3.15. Caja con el circuito montado

3.6 Dispensador

El diseño del dispensador se basó en uno del tipo de tornillo sin fin. El prototipo se hizo con acrílico transparente por su: alta resistencia al impacto, facilidad de mecanización y moldeo, y fácil limpieza. Otra razón por la que se eligió este material es para apreciar mejor su funcionamiento.

En la parte inferior se colocó el tornillo sin fin mediante el cual se entrega el alimento, y su mecanismo de accionamiento. Para evitar que el alimento se salga cuando no es necesario, se levantó el tubo de la parte inferior, ligeramente de enfrente, como se muestra en la derecha de la figura 3.16.

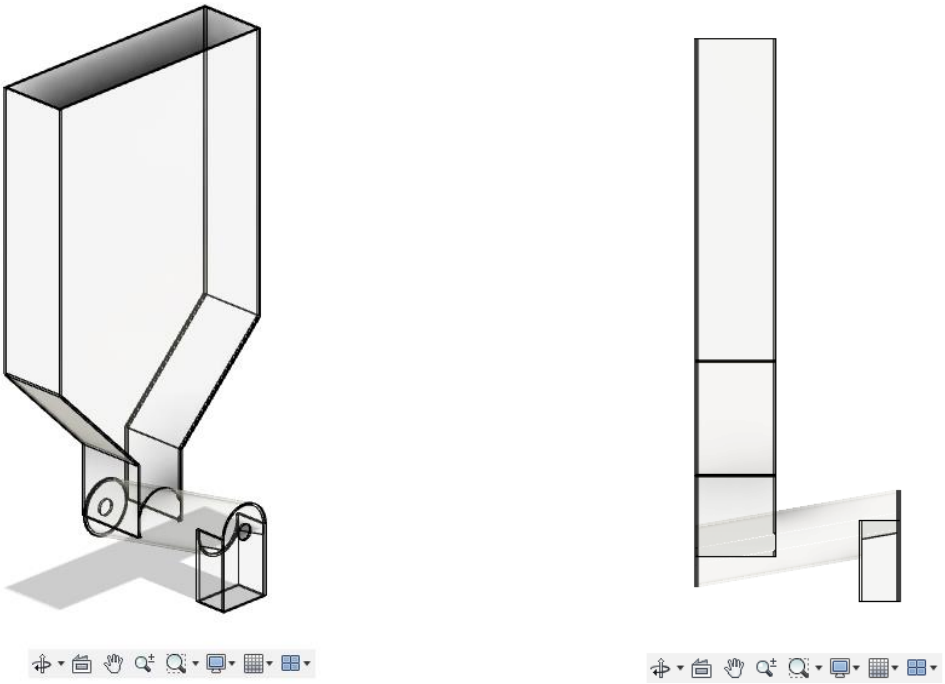


Figura 3.16. Modelo 3D del dispensador

Se compró un tornillo sin fin y se adaptó al tamaño del dispensador. Para accionar el tornillo se utilizó un motorreductor de 10 RPM y 12 V DC, el cual fue acoplado al tornillo con un cople nema de aluminio, mostrado en la figura 3.17.

Figura 3.17. Mecanismo para el accionamiento del tornillo sin fin. Desde la derecha hacia la izquierda: motorreductor, cople y tornillo sin fin.

Se colocaron los sensores infrarrojos a un nivel que corresponden aproximadamente al 25%, 50%, 75% y 100% del volumen del dispensador.

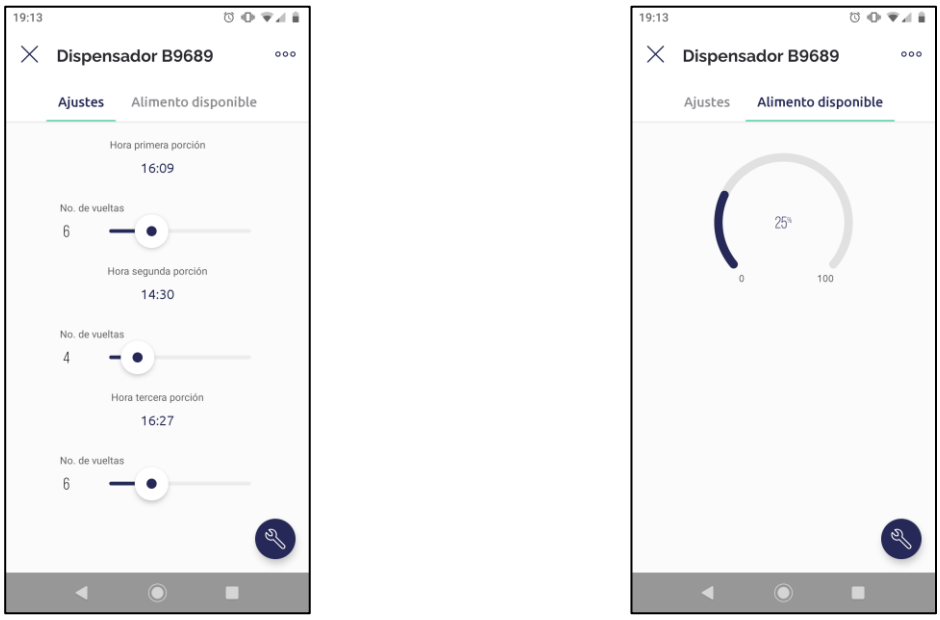
Se acomodó el cableado del circuito, se fijó el motorreductor a la base y la caja del circuito al dispensador. La figura 3.18 muestra el prototipo del dispensador terminado y en funcionamiento.

Figura 3.18. Prototipo terminado y en funcionamiento

3.7 Aplicación móvil y web

La plataforma Blynk IoT nos permite hacer aplicaciones sin código arrastrando y soltando *widgets* con las características que deseemos implementar.

En el caso de la aplicación móvil (figura 3.19), se usaron tres *widgets* para ingresar horas (figura 3.19a), en los que se introducirán los horarios de entrega. Se añadieron tres controles deslizantes, para establecer la cantidad de alimento, que nos permiten seleccionar hasta 20 vueltas. Y se agregó un indicador para visualizar la cantidad aproximada de alimento disponible (figura 3.19b).



(a) ***(b)***

Figura 3.19. Aplicación móvil

Para configurar la aplicación web se usó Blynk.Console. En el Web Dashboard (figura 3.20) se colocaron controles para ajustar la cantidad de alimento y un indicador para el nivel de alimento.

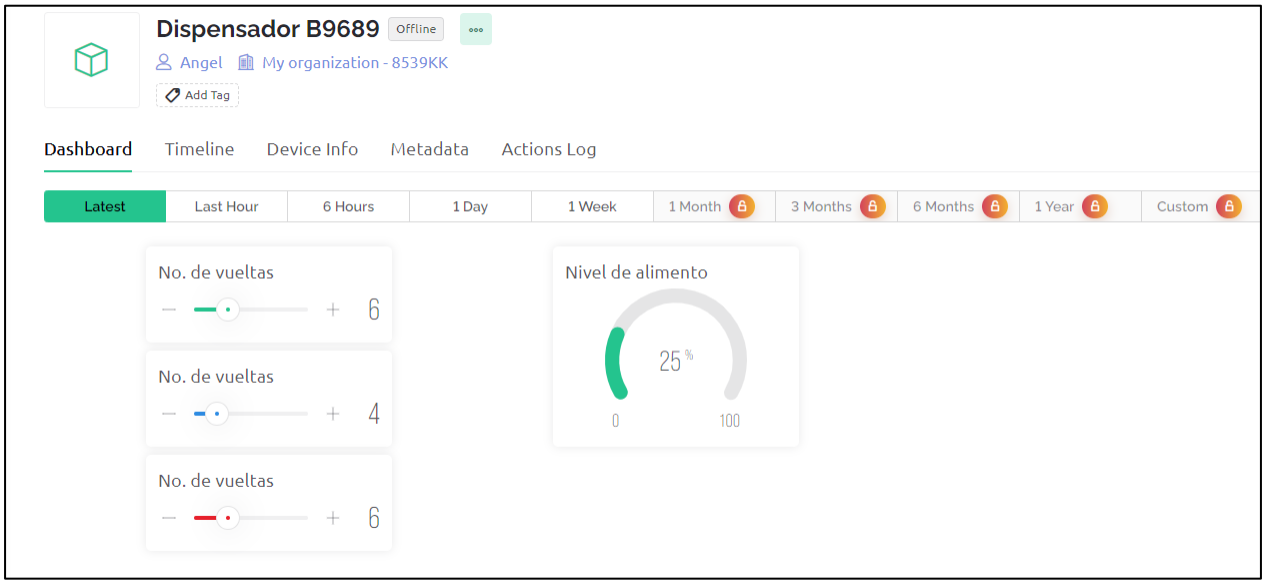


Figura 3.20. Aplicación web para la configuración del dispensador

Para entender cómo se implementó la notificación en la aplicación móvil, es necesario introducir el concepto de pin virtual. Un pin virtual se encarga del intercambio de datos entre el hardware (microcontrolador) y la aplicación de Blynk. El microcontrolador se programó para que establezca el valor del pin V1 como 1 cuando el dispensador este vacío y 0 en caso contrario. Las siguientes líneas de código muestran las funciones que actualiza el valor del pin.

```
296         Blynk.virtualWrite(1, 1);       //notificar sin alimento
297         Blynk.virtualWrite(1, 0);      //restablece el valor de la variable
usada para notificar
```

Se configuró el pin V1 de Blynk como una condición de automatización (hacer una o varias acciones en función de una condición), para que cuando su valor sea 1 (dispensador vacío) se notifique en la app móvil. En la figura 3.21 se muestran las configuraciones del pin V1 y en la figura 3.22 las de la automatización.

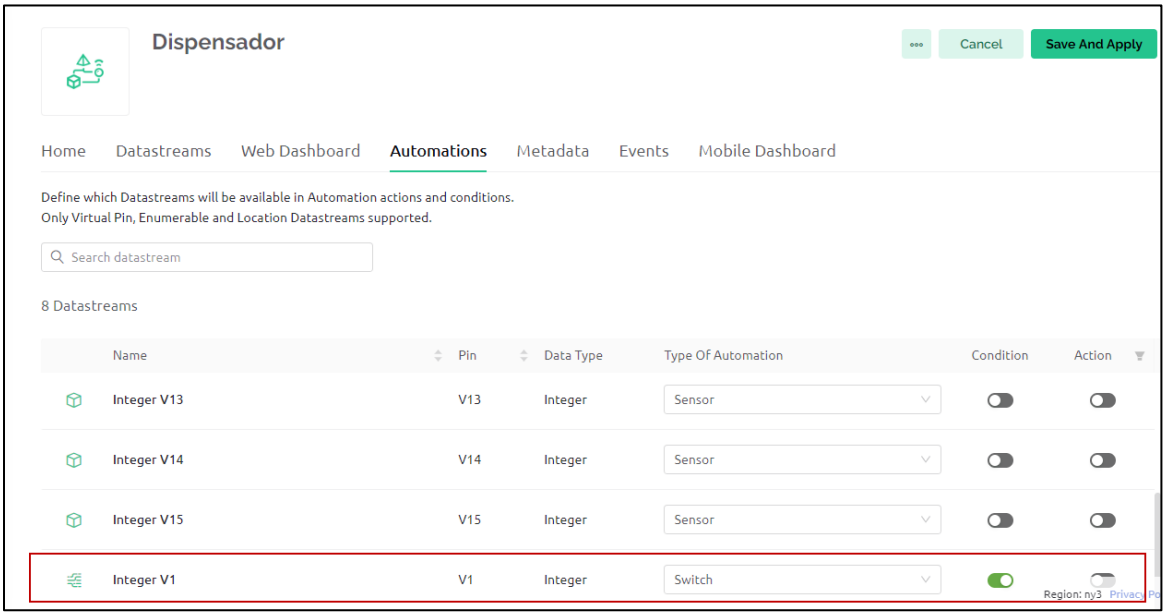


Figura 3.21. Configuración del pin V1 como condición de automatización

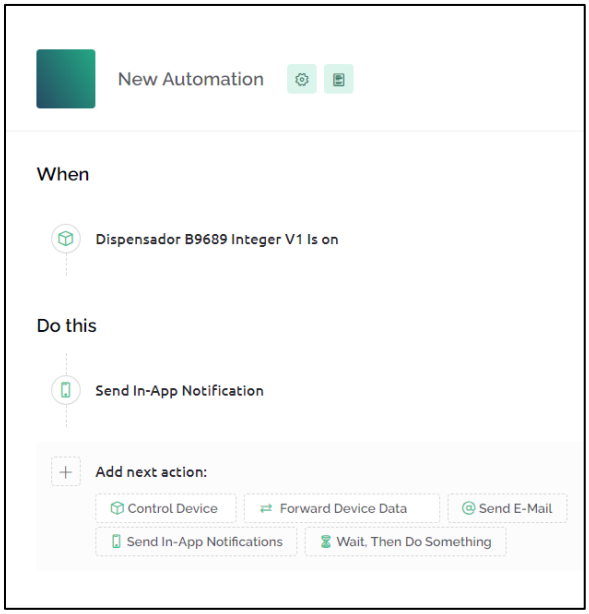
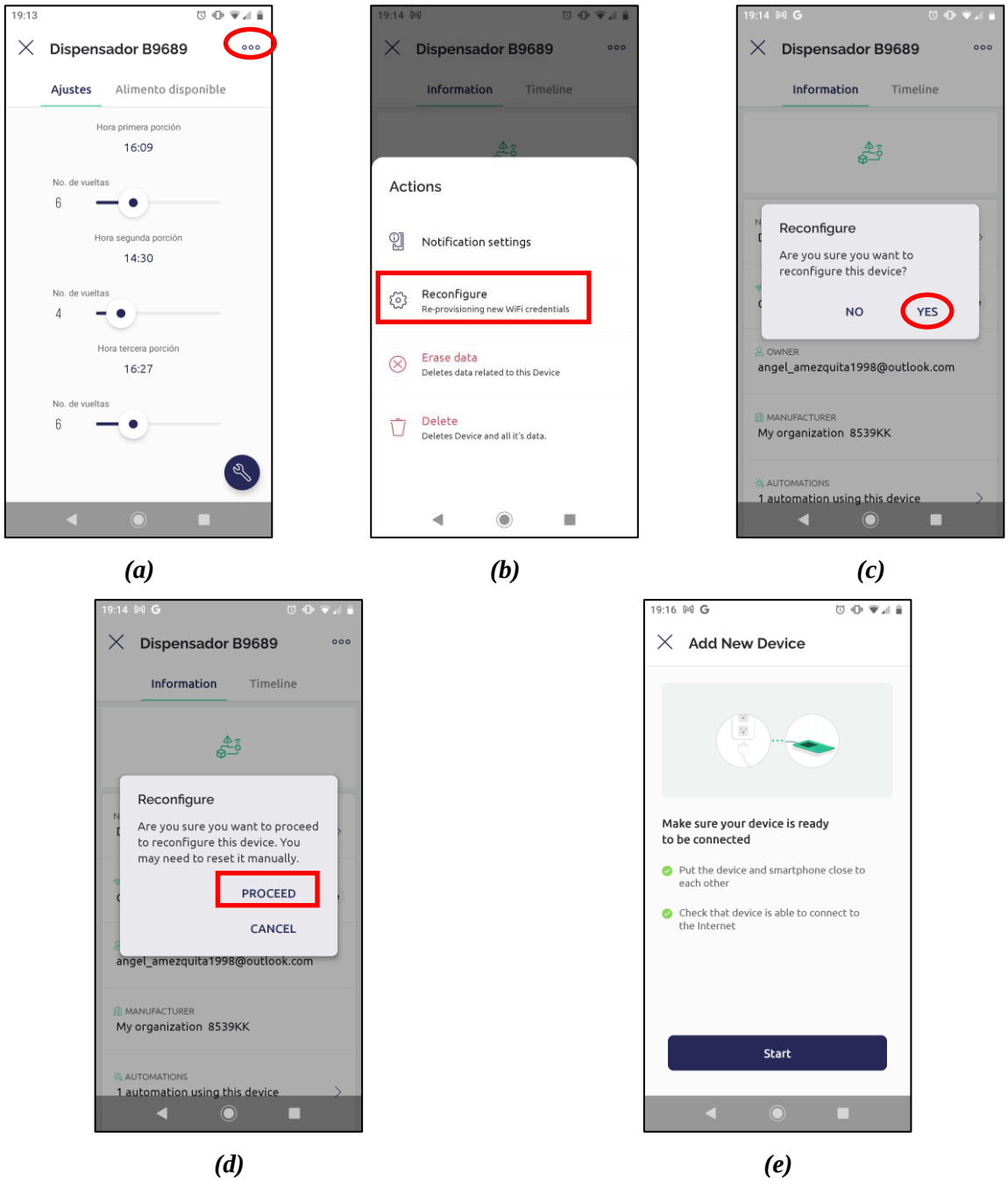


Figura 3.22 Automatización que envía la notificación de nivel bajo

Blynk.Edgent nos proporciona la capacidad de reconfigurar las credenciales de la red a la que se conecte el microcontrolador. Para ello: presionamos el botón de *boot* en el dispensador durante 10 o más segundos, en la aplicación móvil, tocamos los tres puntos de la parte superior derecha (figura 3.23a), repetimos el paso anterior y tocamos en *reconfigure* (figura 3.23b), una vez hecho esto seleccionamos que estamos seguros (figura 3.23c) y *proceed* (figura 3.23d), a partir de ahí seguimos las instrucciones de la aplicación (figura 3.23e).



(a) ***(b)*** ***(c)***

(d) ***(e)***

Figura 3.23. Pasos para reconfigurar las credenciales de la red

CAPÍTULO 4. PRUEBAS Y RESULTADOS

En este capítulo se presentarán las pruebas realizadas durante el desarrollo de este proyecto para cumplir los objetivos planteados. Se pormenorizarán las pruebas realizadas a varios sensores de nivel para ver cuál era el más adecuado para el proyecto, los sensores probados son los siguientes: sensor infrarrojo de presencia infrarrojo E18-D80K, sensor ultrasónico HC-SR04 y un sensor infrarrojo de presencia de cuatro canales. Se explicarán los atributos por los que se eligió un modelo de dispensador, de entre varios que se analizaron. Y se enseñará cómo se resolvieron los comportamientos imprevistos que se presentaron.

4.1 Sensor infrarrojo E18-D80K

Para ver si el sensor era capaz de detectar alimento a cierta distancia, se armó el circuito de la figura 4.1 y se programó la placa de desarrollo con las líneas de código que se muestran a continuación.

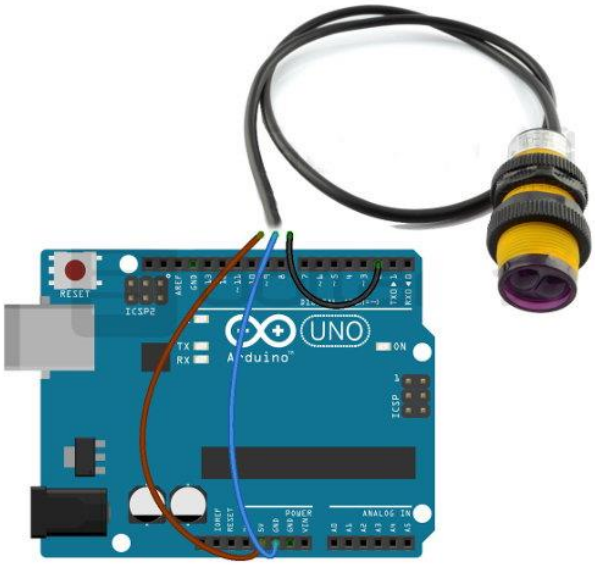


Figura 4.1. Circuito para pruebas con el sensor E18-D80K

```
// Programa para probar el sensor IR E18

void setup() {
  Serial.begin(9600);
  pinMode(2, INPUT);
}

void loop() {
  if(digitalRead(2) == LOW){
    Serial.println("Objeto detectado");
  }
  else{
    Serial.println("Objeto no detectado");
  }
}

```

El programa muestra en el monitor serial el mensaje “Objeto detectado” cuando el sensor detecta un objeto dentro del rango ajustado y, “Objeto no detectado” en el caso contrario.

Para comprobar si el sensor detectaba la presencia de alimento correctamente, se acercó a un recipiente con alimento seco para perros.

En la prueba se observó que el sensor detectaba la presencia de alimento, siempre y cuando estuviera dentro del rango de distancia ajustado.

4.2 Sensor ultrasónico HC-SR04

Para verificar si el sensor ultrasónico es útil para medir el nivel de alimento, se realizó la prueba descrita a continuación. Primero, se armó el circuito de la figura 4.2 y se cargó la placa de desarrollo con el siguiente programa de Arduino.

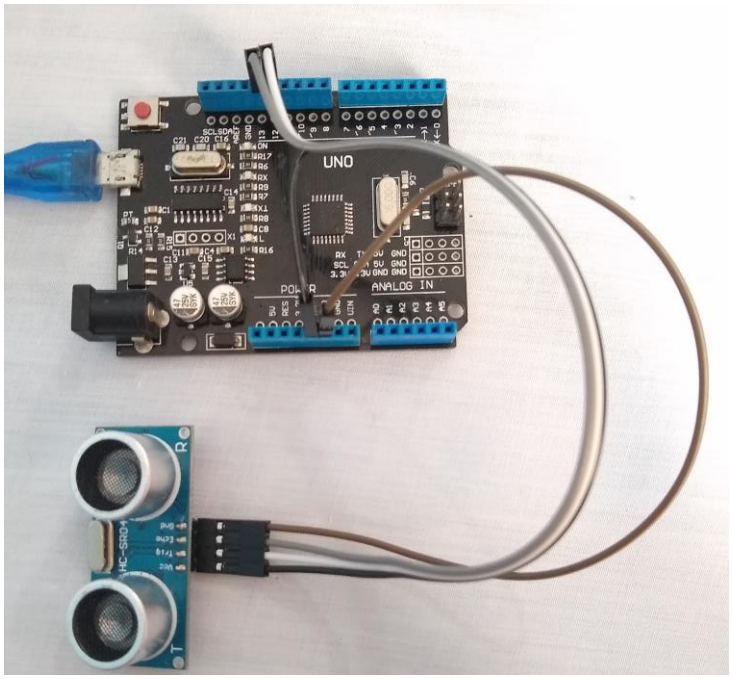


Figura 4.2. Circuito para pruebas con el sensor ultrasónico

```
#define echo 13
#define trig 12

unsigned long start, finish, time;
float dist;
void setup(){
  Serial.begin(9600);
  pinMode(echo, INPUT);
  pinMode(trig, OUTPUT);
}

void loop(){
  digitalWrite(trig, HIGH);
  delayMicroseconds(15);
```

```
15  digitalWrite(trig, LOW);
16
17  while(digitalRead(echo) == LOW);
18  start = micros();
19  while(digitalRead(echo) == HIGH);
20  finish = micros();
21
22  time = finish - start;
23
24  if (time > 25000)
25    time = 0;
26
27  dist = 0.01715*time;
28
29  Serial.print("Distancia: ");
30  Serial.print(dist);
31  Serial.println(" cm");
32
33  delay(15);
34
35}
```

El programa se encarga de: enviar un pulso en alto de 15 µs al pin Trig del sensor para inicializar la medición del sensor; espera a que se emita la señal de 40 kHz y el pin de Echo se coloque en nivel alto; espera a que el pin Echo tenga un nivel bajo, y cuenta el tiempo en el que estuvo en nivel alto; calcula la distancia en centímetros con la siguiente ecuación:

$$d = \frac{ct}{2}$$

si la velocidad del sonido a través del aire, *c*, es de 343 m/s obtenemos que:

$$d = 171.5\, m/s \times t$$
$$d = 1750\, cm/s \times t(\mu s) \times 10^{-6}$$
$$= 0.01715 \times t(\mu s) cm$$

e imprime el resultado en el monitor serial.

Se colocó el sensor apuntando hacia un recipiente con alimento para medir la distancia entre ellos, y se observó lo siguiente: el sensor era capaz de medir correctamente la distancia cuando el nivel de alimento estaba distribuido uniformemente; pero cuando existían irregularidades en el nivel, no era capaz de medir la distancia. Debido a que la homogeneidad del nivel de croquetas en el dispensador depende de la forma en que el usuario lo rellene o de cómo se valla vaciando, se llegó a la conclusión de que no era apto para esta aplicación.

4.3 Sensor infrarrojo de cuatro canales

Para probar la efectividad del sensor para detectar la presencia de alimento se armó el circuito de la figura 4.3 y se programó la placa de desarrollo con el siguiente código de Arduino.

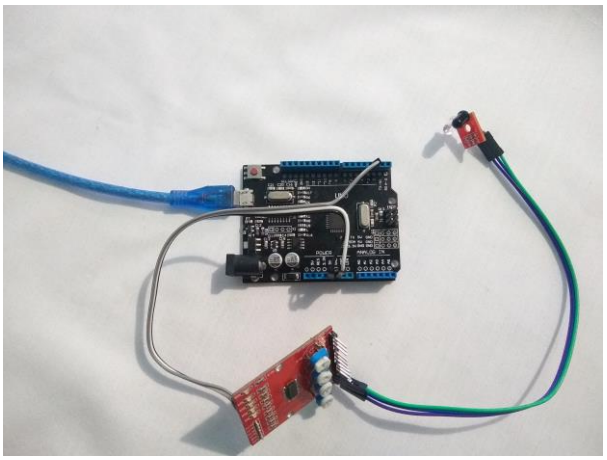


Figura 4.3. Circuito para pruebas con el sensor infrarrojo de 4 canales

```
// Programa para probar el sensor IR de cuatro canales

void setup() {
  Serial.begin(9600);
  pinMode(2, INPUT);
}

void loop() {
  if(digitalRead(2) == LOW){
    Serial.println("Objeto detectado");
  }
  else{
    Serial.println("Objeto no detectado");
  }
}

```

Se perforó una lámina de acrílico de 3mm de espesor y se introdujo uno de los sensores (figura 4.4). Esto se hizo con la finalidad de observar si detectaba la presencia de la lámina, pero esta no afectó el funcionamiento del sensor, porque la sensibilidad del fototransistor que implementa es menor en ángulos de incidencia más próximos a los 90°.

Figura 4.4. Sensor montado en la lámina de acrílico de 3 mm

Se aproximó la lámina de acrílico con el sensor a un recipiente con alimento, como lo ilustra la figura 4.5, y se notó que el sensor detectó el alimento acertadamente.

Figura 4.5. Prueba de detección de alimento con el sensor infrarrojo de cuatro canales

4.4 Dispensadores

Antes de fabricar el dispensador se hicieron pruebas con dos modelos, con el objetivo de examinar si estos modelos cumplían con las siguientes características:

- impedían el paso de alimento al momento de ser rellenados y cuando no se requería, y
- no se atascaban con el alimento.

Estos dos modelos se hicieron con láminas de cartón y se muestran en las figuras 4.6 y 4.7. Ambos modelos están basados en el dosificador de tornillo. En las figuras 4.6 y 4.7 se muestran los modelos examinados. En ellas se observan la vista lateral (figura 4.6a y 4.7a), la vista frontal (figura 4.6b y 4.7c) y a tres cuartos (figura 4.7b).

La prueba consistió en colocar el tornillo sin fin, rellenar el dispensador de alimento y observar si se derramaba. Después, se giraba el tornillo y verificaba que no se atascara.

El resultado fue que el dispensador de la figura 4.6 impedía mejor el paso de alimento, al momento de rellenar, que el de la figura 4.7, y que ambos funcionaban sin atascarse.

(a) ***(b)***

Figura 4.6. Tolva de cartón para pruebas

(a) ***(b)*** ***(c)***

Figura 4.7. Tolva de cartón para pruebas

4.5 Filtro motorreductor

Durante las primeras pruebas que se hicieron con el circuito, en una placa de pruebas, se observó que se reiniciaba cuando se activaba el motorreductor, se analizó la señal de voltaje en los cables de conexión del motorreductor en un osciloscopio y se vio que existía ruido, causado por las escobillas del motor, este se atenuó soldando un capacitor de 10 µF (que sirve como filtro paso bajas o de desacople) en las terminales del motorreductor como lo muestra la figura 4.8.

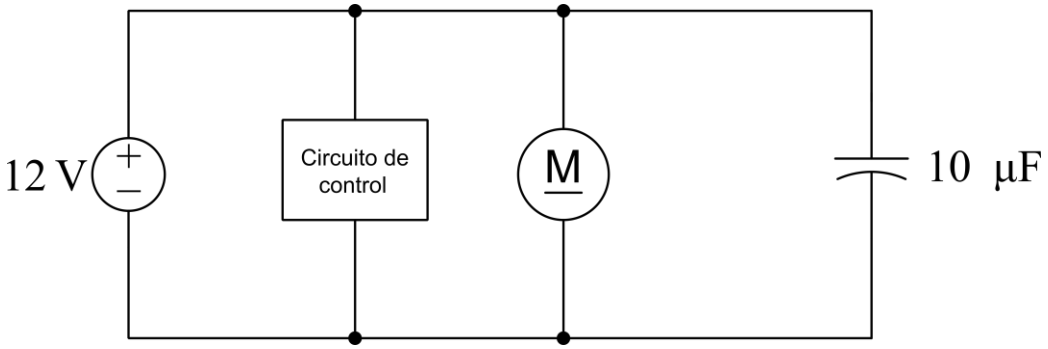


Figura 4.8. Filtro paso bajas para el motorreductor

4.5 Disipador para regulador de voltaje LM7805

También, se notó que el regulador de voltaje aumentaba demasiado su temperatura, mediante la ecuación 4.1 se obtuvo que la potencia disipada del regulador es de 1.68 W.

$$Potencia\ disipada = (V_{entrada} - V_{salida}) \times (I_{salida}) \quad (4.1)$$

$$Potencia\ disipada = (12 - 5)V \times (0.240)A = 1.68\ W$$

Con la ecuación 4.2 y la resistencia térmica de junción-ambiente (θ_{JA}), obtenida de la hoja de datos del regulador, se calculó la temperatura aproximada de junción (arriba de la temperatura ambiente).

$$\Delta T = P \times \theta_{JA} \quad (4.2)$$

$$\Delta T = 1.68W \times 50°\,C/W = 84°C$$

Con lo que con una temperatura de 20 °C la junción alcanzaría los 104 °C.

Por lo anterior se usó un disipador de calor, con este se lograría reducir la temperatura aproximadamente 54.6 °C como se muestra en la ecuación 4.3. Para ello se obtuvo la resistencia térmica de encapsulado-ambiente del disipador, interpolando linealmente los valores de resistencia térmica más cercanos a la potencia calculada en la ecuación 4.1.

$$\Delta T = P \times \theta_{JA}$$

$$\theta_{JA} = \theta_{JC} + \theta_{CA}$$

$$\theta_{JA} = 5\,°\,C/W + 12.5\,°\,C/W = 17.5\,°\,C/W$$

$$\Delta T = 1.68W \times 17.5\,°C/W = 29.4\,°C \quad (4.3)$$

4.6 Pruebas de funcionamiento del prototipo

Para comprobar el correcto funcionamiento del prototipo, se rellenó el prototipo con alimento (marca Ganador para perros adultos de razas medianas y grandes) a su máxima capacidad (5 kg), se configuró para que entregara 10 vueltas, se pesó el alimento y se obtuvieron los datos mostrados en la tabla 4.1.

Tabla 4.1 Datos obtenidos con 5 kg de alimento en el dispensador

Gramos de croquetas entregados en diez vueltas del tornillo	573	574	594	578	589	587	593	587	591	575
Media aritmética	585.1111111									
Desviación estándar	8.2388									

Para determinar si la cantidad de alimento entregada variaba considerablemente cuando el dispensador tiene menos alimento, se repitió la prueba anterior con la diferencia que en esta se rellenó el dispensador con 1.5 kg de alimento, y se obtuvieron las cantidades mostradas en la tabla 4.2.

Tabla 4.2 Datos obtenidos con 1.5 kg de alimento en el dispensador

Gramos de croquetas entregados en diez vueltas del tornillo	547	575	579	599	590	578	597	599	583	578
Media aritmética	582.5									
Desviación estándar	15.5653									

Los promedios solo tienen 3 g de diferencia y, por lo tanto, se puede concluir que la cantidad de alimento entregada no varía considerablemente con distintos niveles de alimento en el dispensador.

Para determinar los gramos de alimento entregados por cada vuelta del tornillo, se pesó el alimento entregado por 5, 10, 15, y 20 vueltas de tornillo de tres diferentes marcas de alimento. Las pruebas se realizaron con 1.5 kg de alimento en el dispensador. Para asegurar una correcta medición del peso, es necesario que el dispensador entregue unas vueltas de alimento, esto se hace para que el alimento se distribuya en la parte inferior del dispensador. En las tablas 4.3, 4.4 y 4.5 se muestran los datos obtenidos.

Tabla 4.3 Ganador adulto razas medianas y grandes

No. de vueltas	1				
Gramos	60	65	57	61	53
				Media aritmética	59.2
				Desviación estándar	4.4944
No. de vueltas	5				
Gramos	291	293	302	292	297
				Media aritmética	295
				Desviación estándar	4.5277
No. de vueltas	10				
Gramos	579	605	589	599	587
				Media aritmética	591.8
				Desviación estándar	10.2567
No. de vueltas	15				
Gramos	873	880	884	882	893
				Media aritmética	882.4
				Desviación estándar	7.2319
No. de vueltas	20				
Gramos	1178	1181	1197	1183	1197
				Media aritmética	1187.2
				Desviación estándar	9.1214

Tabla 4.4 NUPEC adulto raza mediana y grande

No. de vueltas	1				
Gramos	54	43	49	50	46
			Media aritmética		48.4
			Desviación estándar		4.1593
No. de vueltas	5				
Gramos	248	242	243	232	244
			Media aritmética		241.8
			Desviación estándar		5.9330
No. de vueltas	10				
Gramos	480	480	474	477	493
			Media aritmética		480.8
			Desviación estándar		7.2595
No. de vueltas	15				
Gramos	696	718	721	714	719
			Media aritmética		713.6
			Desviación estándar		10.1637
No. de vueltas	20				
Gramos	951	948	955	940	961
			Media aritmética		951
			Desviación estándar		7.8422

Tabla 4.5 PURINA Dog Chow adultos minis y pequeños

No. de vueltas	1				
Gramos	48	55	55	61	49
			Media aritmética		53.6
			Desviación estándar		5.2726
No. de vueltas	5				
Gramos	294	282	275	291	275
			Media aritmética		283.4
			Desviación estándar		8.8487
No. de vueltas	10				
Gramos	580	575	593	575	576
			Media aritmética		579.8
			Desviación estándar		7.6616
No. de vueltas	15				
Gramos	863	874	874	874	873
			Media aritmética		871.6
			Desviación estándar		4.8270
No. de vueltas	20				
Gramos	1186	1176	1201	1176	1185
			Media aritmética		1184.8
			Desviación estándar		10.2323

Para obtener la equivalencia de gramos por vuelta, se dividieron los promedios de los pesos (obtenidos para cada número de vueltas analizado) entre el número de vueltas y se calculó el promedio. Esto se hizo por cada una de las marcas de alimento. Las tablas 4.6, 4.7 y 4.8 muestran la relación obtenida de gramos de alimento por vuelta.

Tabla 4.6 Gramos por vuelta de Ganador adulto razas medianas y grandes

Numero de vueltas	*Media aritmética del peso entregado*	*Gramos por vuelta*
1	59.2	59.2
5	295	59
10	591.8	59.18
15	882.4	58.8266667
20	1187.2	59.36
	Promedio de gramos por vuelta	59.1133333

Tabla 4.7 Gramos por vuelta de NUPEC adulto raza mediana y grande

Numero de vueltas	*Media aritmética del peso entregado*	*Gramos por vuelta*
1	48.4	48.4
5	241.8	48.36
10	480.8	48.08
15	713.6	47.57333333
20	951	47.55
	Promedio de gramos por vuelta	47.99266667

Tabla 4.8 Gramos por vuelta PURINA Dog Chow adultos minis y pequeños

Numero de vueltas	*Media aritmética del peso entregado*	*Gramos por vuelta*
1	53.6	53.6
5	283.4	56.68
10	579.8	57.98
15	871.6	58.10666667
20	1184.8	59.24
	Promedio de gramos por vuelta	57.12133333

Con lo anterior se puede sugerir al usuario, que para obtener la relación de gramos por vuelta (de una marca de alimento que no se encuentre en las tablas 4.6, 4.7 y 4.8), siga los siguientes pasos:

- programe el dispensador para que entregue 3 vueltas, esto con la finalidad de que, con el movimiento del tornillo, el alimento se distribuya en la parte inferior;
- mida la cantidad de alimento que entrega el dispensador en 10 vueltas, y
- la divida entre 10.

También, es importante señalar que en las pruebas realizadas nunca se atascó el mecanismo, esto es porque el torque generado por el motorreductor es lo bastante elevado como para provocar que las croquetas que se comienzan a atascar se rompan con el tornillo.

CONCLUSIONES

Para dotar con la capacidad de establecer horarios de comida, se desarrolló una aplicación web y móvil, donde el usuario puede definir tres horarios, y a cada uno de ellos indicarle la cantidad de alimento a proporcionar, si el usuario solo quiere alimentar dos veces al día a su mascota, solo debe indicar cantidad "0" en uno de los horarios, para que la comida sea entregada en el horario, se incorporó un RTC al sistema, el cual cuenta con su propia batería de larga duración (hasta 5 años). Si es necesario reemplazar la batería, para actualizarle la hora, ya que inicia a las 12:00 del año 2000, solo se debe reiniciar al sistema y que se logre una conexión a Blynk, ya que incorpora una rutina inicial de actualización de hora sobre el RTC.

Las aplicaciones web y móvil también se planearon para que el usuario pueda configurar las porciones de alimento a entregar. El número de vueltas que se puede configurar es suficiente para entregar raciones que cubran las necesidades de alimento seco de cualquier tipo de perro. El dispensador es capaz de entregar una cantidad de alimento proporcional al número de vueltas que el usuario elija, solo es necesario consultar la relación de gramos por número de vueltas realizada en este trabajo, o hacer la propia con la marca de alimento que se vaya a usar.

Las aplicaciones se diseñaron para mostrar el nivel aproximado de alimento disponible en el dispensador. Para ello se utilizó un sensor de presencia infrarrojo de cuatro canales y se le asignó un porcentaje de alimento disponible a cada uno de los canales. Las ventajas que tiene este método para medir el nivel son: que son baratos, que, como están colocados, son fáciles de limpiar, y detectan acertadamente la presencia de alimento; pero tienen la desventaja de que son poco exactos y que para aumentar su exactitud se deben de instalar más sensores. Para disminuir el tráfico de datos hacia los servidores de Blynk IoT se programó el microcontrolador para que solo envíe información cuando exista un cambio en alguno de los sensores. También, se le incorporó a la aplicación móvil la capacidad de notificar cuando el nivel de alimento sea menor al 25 %.

Para que el usuario pueda visualizar las configuraciones sin tener que acceder a la aplicación, se implementó un visualizador LCD en el que se muestran: la hora actual (proporcionada por el RTC), las cantidades de alimento y las horas de entrega establecidas. El visualizador además muestra un mensaje cuando el nivel del dispensador es menor a aproximadamente el 25% del dispensador.

Se utilizaron las librerías que ofrece Blynk IoT para que la configuración o reconfiguración de las credenciales, del punto de acceso inalámbrico, sea intuitiva. La comunicación del sistema depende de la disponibilidad de los servidores de Blynk IoT, sin embargo, debido a que se utilizó una funcionalidad que actualiza la última modificación realizada a la aplicación, se garantiza que el sistema obtenga las últimas configuraciones (de entrega de alimento) de la aplicación, sin que el usuario tenga que reestablecerlos. Ya sea que el dispensador se apagué o que el usuario configure nuevas entregas de alimento cuando no tenga conexión a internet.

Se utilizó una funcionalidad con la que el sistema obtiene las ultimas configuraciones (de entrega de alimento) guardadas en los servidores cada que establece conexión con ellos, esto es para los casos en los que el dispensador se apagué, o no tenga conexión a internet cuando y se reconfiguren nuevas entregas de alimento.

Se examinaron dos modelos de dispensador y se eligió el que impedía mejor la salida de alimento al momento de rellenar. Anqué, al final se le hicieron modificaciones al prototipo, para facilitar su fabricación en acrílico y reducir la capacidad de alimento, porque era excesiva. Durante las pruebas de funcionamiento realizadas al prototipo, no se presenció ningún atasco que detenga totalmente o afecte la velocidad angular del tornillo, el torque del motorreductor es lo suficientemente elevado para hacer funcionar correctamente el tornillo, y para destrozar, con la presión del tornillo, las croquetas que se atasquen.

Con todo lo expuesto se puede concluir que el dispensador cumple con los objetivos planteados, pero esto no impide plantear las siguientes mejoras, o seguir las siguientes recomendaciones para obtener un mejor funcionamiento. Si el usuario lo requiere puede usar un SAI (sistema de alimentación ininterrumpida) para que el sistema tenga una mayor autonomía eléctrica. El dispensador puede ser desarrollado a mayor profundidad por un grupo multidisciplinario, para mejorar el atractivo visual del producto, el funcionamiento del mecanismo del dispensador, y mejorar la funcionalidad de las aplicaciones web y móvil. Para no depender de la disponibilidad o de las tarifas de una plataforma IoT de terceros, se puede examinar la posibilidad de desarrollar una propia. Se pueden analizar más opciones para la medición del nivel de alimento, por ejemplo, probando sensores de distancia láser, galgas extensiométricas, etc. Y, como ultima sugerencia, se puede estudiar que tan viable es su uso con alimento de otras especies.

REFERENCIAS

Agar, S., & Hough, S. (2001). *Small animal nutrition.* Butterworth-Heinemann.

Amazon Web Services, Inc. (2023a). *¿Qué es AWS IoT?* Recuperado el 13 de junio de 2023, de AWS IoT Core Guía para desarrolladores: https://docs.aws.amazon.com/es_es/iot/latest/developerguide/what-is-aws-iot.html

Amazon Web Services, Inc. (2023b). *¿Qué es IoT?* Recuperado el 7 de junio de 2023, de AWS: https://aws.amazon.com/es/what-is/iot/

Amazon Web Services, Inc. (2023c). *Cómo funciona AWS IoT*. Recuperado el 13 de junio de 2023, de AWS IoT Core Guía para desarrolladores: https://docs.aws.amazon.com/es_es/iot/latest/developerguide/aws-iot-how-it-works.html

Bazán Ramírez, M. A. (2010). *SISTEMA MILIDOSIFICADOR DE POLVOS: DISEÑO Y CONSTRUCCIÓN*. México: UNAM.

Case, L. P., Daristotle, L., Hayek, M. G., & Raasch, M. F. (2011). *Canine and feline nutrition : a resource for companion animal professionals.* Maryland Heights: Mosby Elsevier.

Corona Ramírez, L. G., Abarca Jiménez, G. S., & Mares Carreño, J. (2014). *Sensores y actuadores Aplicaciones con Arduino®.* GRUPO EDITORIAL PATRIA, S.A. DE C.V.

Elices, R. (2010). *Atlas de nutrición y alimentación práctica en perros y gatos.* Navarra: Grupo Asís Biomedia S.L.

Garcia Torres, E. M. (2006). *Diseño y construcción de un prototipo con sistemas SCADA aplicado al control del micro clima y dosificación del producto almacenado en silos.* Quito: Universidad Politécnica Salesiana.

Gracia García, O. (24 de marzo de 2018). *Cuidados básicos para los animales de compañía (perros y gatos)*. (Madrid Salud) Recuperado el 24 de enero de 2023, de https://madridsalud.es/cuidados-basicos-para-los-animales-de-compania-perros-y-gatos/

Instituto Nacional de Estadística y Geografía (INEGI). (2021). *PRESENTA INEGI RESULTADOS DE LA PRIMERA ENCUESTA NACIONAL DE BIENESTAR AUTORREPORTADO (ENBIARE) 2021*.

ISL Products International. (2023). *DC MOTOR & SMALL GEAR MOTORS – BASICS*. Recuperado el 26 de junio de 2023, de ISL Products International Ltd.: https://islproducts.com/design-note/dc-motor-dc-gear-motor-basics/

Junta de Andalucía. (2021). *Estado del arte y tendencias internet de las cosas e internet de todo.* Unión Europea Fondo Europeo de Desarrollo Regional.

Leuze electronic. (2021). *Original operating instructions ODS 9 Laser distance sensor.* Leuze electronic GmbH & Co.

Moya Bejarano, C. P. (2016). *DISEÑO DE UN DOSIFICADOR SEMIAUTOMÁTICO POR TORNILLO SIN FIN PARA UCHU JACU EN LA ORGANIZACIÓ NUNOPAC DE LA PARROQUIA AYORA DEL CANTÓN CAYAMBE.* Ibarra: Universidad Técnica del Norte.

myDevices, inc. (2023a). *Features*. Recuperado el 13 de junio de 2023, de Cayenne Docs: https://developers.mydevices.com/cayenne/docs/features/#features

myDevices, inc. (2023b). *Introduction*. Recuperado el 13 de junio de 2023, de Cayenne Docs: https://developers.mydevices.com/cayenne/docs/intro/

Ramos Tellez, R. Y. (2019). *SISTEMA AUTOMÁTICO DISPENSADOR DE CROQUETAS PARA MASCOTAS CONFIGURABLE DESDE APLICACIÓN ANDROID.* Nezahualcoyotl, Estado de México: FES Aragón UNAM.

REAL ACADEMIA ESPAÑOLA. (s.f.). *Diccionario de la lengua española, 23.ª ed.* Recuperado el 23 de mayo de 2023, de https://dle.rae.es

Rose, K., Eldridge, S., & Chapin, L. (2015). *La internet de las cosas- una breve reseña.* Internet Society.

Scherz, P. (2000). *Practical Electronics for Inventors.* McGraw-Hill.

Serrano, A. (24 de mayo de 2023). *Sensores Láser*. Obtenido de elion: https://www.elion.es/productos/sensores/medicion/

Söderby, K. (16 de mayo de 2023). *Getting Started With the Arduino IoT Cloud*. Recuperado el 13 de junio de 2023, de Arduino DOCS: https://docs.arduino.cc/arduino-cloud/getting-started/iot-cloud-getting-started

Teel, J. (28 de julio de 2021). *Introduction to Embedded Electronic Displays*. Recuperado el 23 de mayo de 2023, de Predictable Designs: https://predictabledesigns.com/introduction-embedded-electronic-displays/

Ulaby, F. T., Maharbiz, M. M., & Furse, C. M. (2018). *CIRCUIT ANALYSIS AND DESIGN.* Michigan Publishing.

Webster, J. G. (1998). *Measurement, Instrumentation, and Sensors Handbook.* CRC Press.

FUENTES DE IMÁGENES

Figura 1.1.a Dispensadores de alimento disponibles en el mercado, obtenido de: https://www.amazon.com.mx/PAPIFEED-Comedero-autom%C3%A1tico-inoxidable-desmontable/dp/B09TKH9Q31/ref=d_bmx_dp_v169bo3n_sccl_3_4/134-2115513-5815308?pd_rd_w=kK8WJ&content-id=amzn1.sym.407cc1e5-b3e8-4667-bdde-c659627405ac&pf_rd_p=407cc1e5-b3e8-4667-bdde-c659627405ac&pf_rd_r=052HDDARPNS7M7PABYB8&pd_rd_wg=TGlUG&pd_rd_r=d0b6db91-5843-4f75-bbcb-c8dc999ca74a&pd_rd_i=B09TKH9Q31&th=1, recuperado el: 30 de julio de 2023.

Figura 1.1.b Dispensadores de alimento disponibles en el mercado, obtenido de: https://www.steren.com.mx/dispensador-wi-fi-de-alimento-para-mascotas-con-camara-full-hd-y-grabador-de-voz.html, recuperado el: 30 de julio de 2023.

Figura 1.2. Prototipo del dispensador desarrollado en el 2019, obtenido de: Ramos Tellez, R. Y. (2019). SISTEMA AUTOMÁTICO DISPENSADOR DE CROQUETAS PARA MASCOTAS CONFIGURABLE DESDE APLICACIÓN ANDROID, recuperado el: 30 de julio de 2023.

Figura 1.3. Diagrama a bloques del dispensador desarrollado en el 2019, obtenido de: Ramos Tellez, R. Y. (2019). SISTEMA AUTOMÁTICO DISPENSADOR DE CROQUETAS PARA MASCOTAS CONFIGURABLE DESDE APLICACIÓN ANDROID, recuperado el: 30 de julio de 2023.

Figura 2.1.a Alimentos húmedos, obtenido de: https://estacionmascota.com/inicio/105-172-royal-canin-wet-all-dogs-puppy-lata.html, recuperado el: 5 de mayo de 2023.

Figura 2.1.b Alimentos húmedos, obtenido de: https://puppis.blog/articulo/alimentos-humedos-y-otros-trucos-de-hidratacion-para-el-verano, recuperado el: 5 de mayo de 2023.

Figura 2.2. Alimentos semihúmedos, obtenido de: Elices, R. (2010). Atlas de nutrición y alimentación práctica en perros y gatos, recuperado el: 5 de mayo de 2023.

Figura 2.3. Alimentos secos, obtenido de: Elices, R. (2010). Atlas de nutrición y alimentación práctica en perros y gatos, recuperado el: 5 de mayo de 2023.

Figura 2.4. Medición de un sensor ultrasónico, obtenido de: Corona Ramírez, L. G., Abarca Jiménez, G. S., & Mares Carreño, J. (2014). Sensores y actuadores Aplicaciones con Arduino®, recuperado el: 5 de mayo de 2023.

Figura 2.5. Módulo HC-SR04, obtenido de: https://gruposmj.com/tienda/ols/products/xn-sensor-de-distancia-ultrasnico-modelo-hc-sr04-5le, recuperado el: 5 de mayo de 2023.

Figura 2.6. Diagrama de tiempos HC-SR04, obtenido de: Corona Ramírez, L. G., Abarca Jiménez, G. S., & Mares Carreño, J. (2014). Sensores y actuadores Aplicaciones con Arduino®, recuperado el: 5 de mayo de 2023.

Figura 2.7. Sensor láser de medición por triangulación, obtenido de: Leuze electronic. (2021). Original operating instructions ODS 9 Laser distance sensor, recuperado el: 5 de mayo de 2023.

Figura 2.25. Dosificador vibratorio, obtenido de: http://www.ptolomeo.unam.mx:8080/xmlui/bitstream/handle/132.248.52.100/3875/bazanramirez.pdf?sequence=1&isAllowed=y, recuperado el: 20 de junio de 2023.

Figura 2.26. Dosificador de válvula, obtenido de: http://www.ptolomeo.unam.mx:8080/xmlui/bitstream/handle/132.248.52.100/3875/bazanramirez.pdf?sequence=1&isAllowed=y, recuperado el: 20 de junio de 2023.

Figura 2.27. Componentes de un servomotor, obtenido de: Corona Ramírez, L. G., Abarca Jiménez, G. S., & Mares Carreño, J. (2014). Sensores y actuadores Aplicaciones con Arduino® , recuperado el: 26 de junio de 2023.

Figura 2.28. Posiciones del eje de un servomotor, obtenido de: Corona Ramírez, L. G., Abarca Jiménez, G. S., & Mares Carreño, J. (2014). Sensores y actuadores Aplicaciones con Arduino® , recuperado el: 26 de junio de 2023.

Figura 2.29. Tren de pulsos para el control de un servomotor, obtenido de: Corona Ramírez, L. G., Abarca Jiménez, G. S., & Mares Carreño, J. (2014). Sensores y actuadores Aplicaciones con Arduino® , recuperado el: 26 de junio de 2023.

Figura 2.30 Motores a pasos, obtenido de: Scherz, P. (2000). Practical Electronics for Inventors , recuperado el: 26 de junio de 2023.

Figura 2.31. Funcionamiento del motor a pasos de reluctancia variable, obtenido de: Scherz, P. (2000). Practical Electronics for Inventors , recuperado el: 26 de junio de 2023.

Figura 2.32. Motor con engranes reductores, obtenido de: https://islproducts.com/design-note/dc-motor-dc-gear-motor-basics/, recuperado el: 26 de junio de 2023.

Figura 4.1. Circuito para pruebas con el sensor E18-D80K, obtenido de: https://solectroshop.com/es/content/82-como-detectar-objetos-con-sensor-de-proximidad-infrarrojo-e18-d80nk-en-arduino, recuperado el: 20 de julio de 2023.

Nota: las imágenes no indicadas son de autoría propia.

ANEXO

Código C/C++

```
#include <Arduino.h>
#include <RTClib.h>
#include <EEPROM.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <time.h>


#define PIN_MOTOR 14
#define PIN_IR1 4
#define PIN_IR2 18
#define PIN_IR3 17
#define PIN_IR4 23
#define LCD_ADDR 0x27
#define LCD_COLS 20
#define LCD_ROWS 4
#define EEPROM_SIZE 20
#define ESP_INTR_FLAG_DEFAULT 0 //ISR configuration flag
#define Vuelta(vueltas) vTaskDelay(vueltas * (pdMS_TO_TICKS(5757)))
//delay número de vueltas

// Fill-in information from your Blynk Template here
#define BLYNK_TEMPLATE_ID "TMPLMKw3G3BC"
#define BLYNK_DEVICE_NAME "Dispensador"

#define BLYNK_FIRMWARE_VERSION        "0.1.0"

#define BLYNK_PRINT Serial
//#define BLYNK_DEBUG

#define APP_DEBUG

// Uncomment your board, or configure a custom board in Settings.h
//#define USE_WROVER_BOARD
//#define USE_TTGO_T7
//#define USE_ESP32C3_DEV_MODULE
//#define USE_ESP32S2_DEV_KIT
#include <BlynkEdgent.h>

LiquidCrystal_I2C lcd(LCD_ADDR, LCD_COLS, LCD_ROWS);
RTC_DS1307 RTC;
byte temp = 0;
byte nivel;
bool presenciaActual = 0;          //indica la presencia de alimento
bool presenciaAnterior = 0;
unsigned long previousTime = 0;

void blynkTask(void *);
void display(void *);
```

```
49 void segundos_a_horas(int *, int *, long);
50 void sensor_nivel(void);
51 void notificacion(void);
52
53 BLYNK_WRITE(V0)                          //obtiene el valor de los
sliders(número de vueltas)
54 {
55     int vueltas = param.asInt();
56     EEPROM.write(1, vueltas);          //los guarda en la memoria flash del
esp32
57     EEPROM.commit();
58 }
59
60 BLYNK_WRITE(V13)
61 {
62     int vueltas = param.asInt();
63     EEPROM.write(5, vueltas);
64     EEPROM.commit();
65 }
66
67 BLYNK_WRITE(V14)
68 {
69     int vueltas = param.asInt();
70     EEPROM.write(9, vueltas);
71     EEPROM.commit();
72 }
73
74 BLYNK_WRITE(V9)
75 {
76     int hora, minuto;
77     String HoraEntrega1 = param.asString();     //Hora en segundos a partir
de las 00:00
78     long HoraEntrega = HoraEntrega1.toInt();
79     segundos_a_horas(&hora, &minuto, HoraEntrega);
80     EEPROM.write(2, hora);
81     EEPROM.write(3, minuto);
82     EEPROM.commit();
83 }
84
85 BLYNK_WRITE(V10)
86 {
87     int hora, minuto;
88     String HoraEntrega1 = param.asString();     //Hora en segundos a partir
de las 00:00
89     long HoraEntrega = HoraEntrega1.toInt();
90     segundos_a_horas(&hora, &minuto, HoraEntrega);
91     EEPROM.write(6, hora);
92     EEPROM.write(7, minuto);
93     EEPROM.commit();
94 }
95
96 BLYNK_WRITE(V11)
97 {
98     int hora, minuto;
```

```
99      String HoraEntrega1 = param.asString();     //Hora en segundos a partir
de las 00:00
100     long HoraEntrega = HoraEntrega1.toInt();
101     segundos_a_horas(&hora, &minuto, HoraEntrega);
102     EEPROM.write(10, hora);
103     EEPROM.write(11, minuto);
104     EEPROM.commit();
105}
106
107BLYNK_WRITE(InternalPinRTC) {   //check the value of InternalPinRTC
108     time_t t = param.asLong();      //store time in t variable
109
110     RTC.adjust(DateTime(t));    //configura la hora en el RTC
111}
112
113BLYNK_CONNECTED()
114{
115     Blynk.sendInternal("rtc", "sync"); //request current local time for
device
116     Blynk.syncAll();
117     Blynk.virtualWrite(15, nivel);
118
119     if (nivel==0)
120     {
121         Blynk.virtualWrite(1, 1);  //notificar sin alimento
122         Blynk.virtualWrite(1, 0);  //resetea el pin de notificación
123         presenciaActual=0;
124         presenciaAnterior=0;
125     }
126     else
127     {
128         presenciaActual=1;
129         presenciaAnterior=1;
130     }
131
132}
133
134void setup() {
135     Serial.begin(115200);
136     Wire.begin();
137     RTC.begin();
138     lcd.init();
139     lcd.clear();
140     lcd.backlight();
141     EEPROM.begin(EEPROM_SIZE);
142     pinMode(PIN_MOTOR, OUTPUT);
143     digitalWrite(PIN_MOTOR, LOW);
144
145     pinMode(PIN_IR1, INPUT);
146     pinMode(PIN_IR2, INPUT);
147     pinMode(PIN_IR3, INPUT);
148     pinMode(PIN_IR4, INPUT);
149
```

```
150     nivel = (!digitalRead(PIN_IR1) + !digitalRead(PIN_IR2) +
!digitalRead(PIN_IR3) + !digitalRead(PIN_IR4)) * 25;
151
152     if(xBinarySemaphore != NULL)
153     {
154         BlynkEdgent.begin();
155
156         xTaskCreatePinnedToCore(
157             vHandlerTask,
158             "Handler",
159             7 * configMINIMAL_STACK_SIZE,
160             NULL,
161             3,
162             NULL,
163             1);
164
165         xTaskCreatePinnedToCore(
166             blynkTask,
167             "blynk run",
168             8 * configMINIMAL_STACK_SIZE,
169             NULL,
170             1,
171             NULL,
172             1);
173
174
175         xTaskCreatePinnedToCore(
176             display,
177             "display",
178             7 * configMINIMAL_STACK_SIZE,
179             NULL,
180             1,
181             NULL,
182             0);
183     }
184
185     vTaskDelete(NULL);
186 }
187
188void loop() {
189}
190
191void display(void *pvParameters)
192{
193     for(;;)
194     {
195         DateTime now = RTC.now();
196         lcd.setCursor(6, 0);
197         lcd.printf("%02d:%02d:%02d", now.hour(), now.minute(),
now.second());
198
199         lcd.setCursor(0, 1);
200         lcd.printf("P1: %2d Vts %02d:%02d Hrs", EEPROM.read(1),
EEPROM.read(2), EEPROM.read(3));
```

```
201         lcd.setCursor(0, 2);
202         lcd.printf("P2: %2d Vts %02d:%02d Hrs", EEPROM.read(5),
EEPROM.read(6), EEPROM.read(7));
203         lcd.setCursor(0, 3);
204         lcd.printf("P3: %2d Vts %02d:%02d Hrs", EEPROM.read(9),
EEPROM.read(10), EEPROM.read(11));
205
206         notificacion();     //para actualizar la presencia anterior
207
208         while (presenciaActual == 0)
209         {
210             lcd.clear();
211             lcd.setCursor(0, 0);
212             lcd.print("El dispensador se");
213             lcd.setCursor(0,1);
214             lcd.print("esta vaciando,");
215             lcd.setCursor(0, 2);
216             lcd.print("favor de rellenar");
217             lcd.noBacklight();
218             vTaskDelay(pdMS_TO_TICKS(1000));
219             lcd.backlight();
220             vTaskDelay(pdMS_TO_TICKS(1000));
221             notificacion();
222             lcd.clear();
223         }
224
225         if((EEPROM.read(0) == 1 && now.hour() == EEPROM.read(2) &&
now.minute() == EEPROM.read(3) && ((now.second() == 0) || (now.second() ==
1))) && (presenciaActual == 1))
226         {
227             lcd.clear();
228             digitalWrite(PIN_MOTOR, HIGH);
229             lcd.setCursor(1, 0);
230             lcd.print("Entregando alimento");
231             Vuelta(EEPROM.read(1));
232             digitalWrite(PIN_MOTOR, LOW);
233             lcd.clear();
234         }
235

236         if((EEPROM.read(4) == 2 && now.hour() == EEPROM.read(6) &&
now.minute() == EEPROM.read(7) && ((now.second() == 0) || (now.second() ==
1))) && (presenciaActual == 1))
237         {
238             lcd.clear();
239             digitalWrite(PIN_MOTOR, HIGH);
240             lcd.setCursor(1, 0);
241             lcd.print("Entregando alimento");
242             Vuelta(EEPROM.read(5));
243             digitalWrite(PIN_MOTOR, LOW);
244             lcd.clear();
245         }
246
```

```
247         if((EEPROM.read(8) == 3 && now.hour() == EEPROM.read(10) &&
now.minute() == EEPROM.read(11) && ((now.second() == 0) || (now.second() ==
1))) && (presenciaActual == 1))
248         {
249             lcd.clear();
250             digitalWrite(PIN_MOTOR, HIGH);
251             lcd.setCursor(1, 0);
252             lcd.print("Entregando alimento");
253             Vuelta(EEPROM.read(9));
254             digitalWrite(PIN_MOTOR, LOW);
255             lcd.clear();
256         }
257
258     }
259}
260
261void blynkTask(void *pvPfarameters)
262{
263     for(;;)
264     {
265         BlynkEdgent.run();
266     }
267}
268
269void segundos_a_horas(int *hora, int *minuto, long segundos)
270{
271     *hora = segundos / 3600;
272     *minuto = (segundos / 60) % 60;
273}
274
275void sensor_nivel()
276{
277     nivel = (!digitalRead(PIN_IR1) + !digitalRead(PIN_IR2) +
!digitalRead(PIN_IR3) + !digitalRead(PIN_IR4)) * 25;
278
279     presenciaActual = (nivel > 0) ? 1 : 0;  //bandera que indica la
presencia de alimento
280
281     if (temp != nivel)                      //solo se actualiza cuando el
nivel actual es distinto al anterior
282     {
283         Blynk.virtualWrite(15, nivel);
284         temp = nivel;
285     }
286}
287
288void notificacion()
289{
290     unsigned long currentTime;
291     int interval = 9999;
292     currentTime = pdTICKS_TO_MS(xTaskGetTickCount());
293     sensor_nivel();
294
295     if(currentTime - previousTime > interval)
```

```
296     {
297       previousTime = currentTime;
298       if((presenciaActual==0) && (presenciaAnterior==1))
299      {
300          Blynk.virtualWrite(1, 1);  //notificar sin alimento
301        Blynk.virtualWrite(1, 0);
302       }
303       presenciaAnterior = presenciaActual;
304     }
305}
```

Plano del dispensador