UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA
INGENIERÍA ELÉCTRICA – INSTRUMENTACIÓN

SISTEMA DE ADQUISICIÓN CON SENSORES DE PRESIÓN PARA LA DETECCIÓN DE ALTERACIONES EN PISADAS DE NIÑOS

TESIS
QUE PARA OPTAR POR EL GRADO DE:
MAESTRO EN INGENIERÍA ELÉCTRICA

PRESENTA:
ING. ALBERTO RAMIRO GARIBAY MARTÍNEZ

TUTOR PRINCIPAL
DRA. ROSALBA CASTAÑEDA GUZMÁN, ICAT

COMITÉ TUTOR
DR. NASER QURESHI, ICAT
DR. ASUR GUADARRAMA SANTANA, ICAT
DR. SANTIAGO JESÚS PÉREZ RUIZ, ICAT
DR. JOSÉ LUIS BENITEZ BENITEZ, ICAT

CIUDAD DE MÉXICO, DICIEMBRE DE 2018

JURADO ASIGNADO:

Presidente:	DR. NASER QURESHI
Secretario:	DR. ASUR GUADARRAMA SANTANA
Vocal:	DRA. ROSALBA CASTAÑEDA GUZMÁN
1 er. Suplente:	DR. SANTIAGO JESÚS PÉREZ RUIZ
2 d o. Suplente:	DR. JOSÉ LUIS BENITEZ BENITEZ

Lugar o lugares donde se realizó la tesis: Ciudad Universitaria, CD. MX.

TUTOR DE TESIS:

DRA. ROSALBA CASTAÑEDA GUZMÁN

FIRMA

Agradecimientos

Se agradece a todos mis sinodales por su apoyo a lo largo de la realización del presente trabajo y durante el posgrado, de igual forma a toda la gente que estuvo detrás de todo este proceso animándome a terminar.

Investigación realizada gracias al programa UNAM-PAPIIT IG 100517.

Resumen

En este trabajo se presenta el desarrollo de una matriz de sensores de presión de tipo capacitivo donde el elemento activo es una hoja (aproximadamente 0.2 [mm]) de Tereftalato de polietileno (PET, por su sigla en inglés), para la detección de alteraciones o problemas ortopédicos en niños con problemas de sobrepeso. Al inicio de esta tesis se reporta una revisión de algunos de los conceptos básicos necesarios para entender el funcionamiento de estos sensores y su aplicación en el sensado de presión. Después, se describe una metodología de fabricación de los sensores. En la parte final de esta tesis se presenta el trabajo realizado para la caracterización de sensores fabricados en el ICAT y su validación para aplicaciones en un sistema de detección de problemas ortopédicos en niños con sobrepeso.

Existen diversos tipos de láminas de PET, desde las hojas de $1[\mu m]$ o menos hasta de $0.5[mm]$. Parte del proceso de fabricación de los sensores implica realizar un depósito de películas delgadas metálicas sobre la superficie de las hojas de PET, con el objetivo de construir estructuras de tipo capacitivo. Parte del trabajo fue realizar dichos depósitos en condiciones tales que las propiedades capacitivas del PET no se perdieran, una de las ventajas del PET a comparación de otros polímeros es que tiene un amplio margen entre la temperatura de transición ($70[°C]$]) y la de fusión ($255[°C]$), reblandeciéndose notablemente a partir de los $200[°C]$, lo que permitió realizar el depósito de películas delgadas mediante evaporación térmica.

Debido a que las hojas de PET utilizadas son flexibles, uno de los intereses en desarrollar una plantilla para el pie de un niño con películas de PET es integrar cuarenta y ocho de ellos en un arreglo bidimensional ergonómico para su uso en detección de problemas en niños con problemas de sobrepeso.

La validación experimental del desempeño de los sensores diseñados y fabricados en esta tesis se realizó mediante el registro de señales de presión generadas en diferentes materiales con geometrías simples. Las pruebas fueron orientadas a ilustrar el potencial de los sensores para la detección del nivel de presión. Como materiales de prueba se utilizaron diferentes cuerpos como botas y rollos de cinta adhesiva que se usaron para ejercer presión sobre los sensores. Se demostró la versatilidad del arreglo de sensores y su capacidad para obtener información particular y relevante de cada tipo de excitación.

El mérito de este trabajo reside en haber logrado desarrollar un prototipo de un sistema basado en sensores capacitivos de presión capaz de detectar problemas o anomalías en las pisadas de niños con problemas de obesidad, esto en conjunto con el diagnóstico de un médico experto en el tema, así como desarrollar una metodología propia para la fabricación de los sensores incluyendo el depósito de electrodos de plata sobre polímeros. Así mismo, este trabajo pretende ser una respuesta a la necesidad que existe en el ámbito académico

universitario de poder disponer de un entorno de diseño y prototipado de aplicaciones de medición, aunque podría ser para trabajos futuros una plataforma sólida para automatización de procesos y espacios físicos. La plataforma Open Hardware Arduino ha demostrado, ser una opción muy interesante para incluir en un Laboratorios de prototipado. Son muchas sus ventajas, entre las que destacamos su costo, su libre difusión y exención de costos de patentes por desarrollo, esto va de la mano con la gran comunidad de usuarios que existe la cual genera una cantidad de aplicaciones las cuales reducen el tiempo de desarrollo.

Esto, en combinación con el software de la empresa National Instruments: LabVIEW, fue ideal para la realización de este trabajo ya que el procesamiento de los datos adquiridos en el entorno de programación gráfica de LabVIEW Edición de Estudiante permitió reducir considerablemente el tiempo de desarrollo y el costo de la aplicación final.

Contenido

1.Introducción

Hoy en día el avance en la investigación de nuevos materiales, ha traído consigo una gran variedad de aplicaciones, por ejemplo, el campo de los sensores, debido que se busca sensores cuyas características posea mayor sensibilidad, dimensiones más pequeñas y sobre todo bajo costo. Unos de los sensores ampliamente utilizados, son los sensores capacitivos. En la presente tesis se estudian sensores de presión capacitivos con películas de PET enfocados a la detección de malformaciones y anomalías en las pisadas de niños con problemas de obesidad y sobre peso.

La configuración estructural de dichos sensores es tal que operan en modo capacitivo y con dimensiones espaciales adaptables a varias configuraciones. Para efectos prácticos se implementó un arreglo de veinte sensores los cuales se caracterizaron experimentalmente a fin de corroborar su rendimiento y estabilidad en la respuesta. A lo largo de ésta tesis se presentan los elementos que justifican el enfoque y permite visualizar el potencial de los sensores en términos de instrumentación de tecnología novedosa para la detección de malformaciones y anomalías en las pisadas de niños con problemas de obesidad y sobre peso. En particular se describen los elementos teóricos y técnicos necesarios para el diseño de un prototipo de sistema formado por arreglos matriciales de estos sensores para su uso en conjunto con un sistema de adquisición y procesamiento de señales con una interfaz gráfica con la finalidad de facilitar una interpretación más simple obtenida de los sensores que ayude a los médicos a dar diagnostico con más certeza al paciente para su uso en la medicina.

2.Fundamento Teórico

2.1. Capacitor

Los capacitores se consideran dispositivos básicos en el mundo de la electrónica. Para comprender el funcionamiento de los sensores capacitivos, es importante conocer y entender las propiedades físicas, geométricas y eléctricas, así como los principios básicos de funcionamiento de los capacitores.

Un capacitor es un dispositivo que consta de dos electrodos separados por un aislante. Los capacitores generalmente constan de dos placas conductoras separadas por un material no conductor conocido como dieléctrico (ε_r). La energía eléctrica o carga se almacena en las placas conductoras. El material dieléctrico puede ser aire, cerámica, polímero, o algún otro aislante [1].

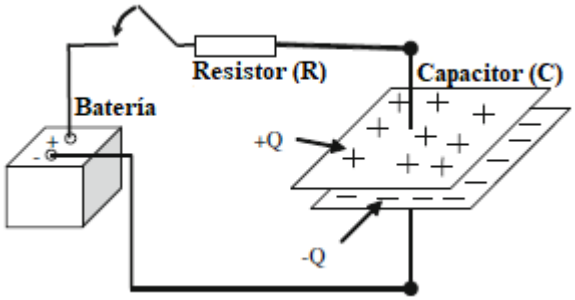


Figura 2.1. Circuito eléctrico de almacenamiento de carga eléctrica en un capacitor.

La figura 1 muestra un circuito elemental donde se carga el capacitor en el momento que se cierra el interruptor.

Al aplicarse un voltaje a través de las dos terminales del capacitor, las placas conductoras comenzarán a almacenar energía eléctrica hasta que la diferencia de potencial a lo largo del capacitor sea la misma que la de la fuente de voltaje. La carga eléctrica permanecerá en las placas incluso después de desconectar la fuente de voltaje a menos que otro componente consuma esta carga o que el capacitor pierda esta carga debido a una fuga, ya que ningún dieléctrico es un aislante perfecto. Los capacitores con pequeñas fugas pueden mantener su carga durante un considerable periodo de tiempo. La placa conectada con la terminal positiva almacena carga positiva ($+Q$) en su superficie y la placa conectada a la terminal negativa almacena carga negativa ($-Q$).

El tiempo que se requiere para que el capacitor se considere totalmente cargado es determinado por una *constante de tiempo* (τ). El valor de la constante de tiempo describe el

tiempo de carga del capacitor al 63% de su capacidad total. La constante de tiempo se mide en segundos y se pude definir como se muestra en la ecuación 1.1, donde, R es el resistor conectado en serie con el capacitor con capacitancia C [1].

$$\tau = RC \tag{2.1}$$

La capacitancia es la propiedad eléctrica de los capacitores. Es una medida de la cantidad de carga que el capacitor puede almacenar cuando este es sometido a una diferencia de potencial entre sus placas. La capacitancia se mide en Farad (F) y puede ser definida en unidades de $\left[\frac{coulomb}{volt}\right]$ o $\left[\frac{C}{V}\right]$ como se muestra en la ecuación 2.2.

$$C = \frac{Q}{V} \tag{2.2}$$

donde,

C es la capacitancia en $[F]$,

Q es la magnitud de la carga almacenada en cada placa en $[C]$,

V es el voltaje aplicado a las placas en $[V]$.

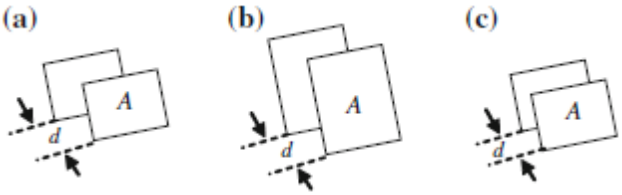


Figura 2.2. Factores que influyen en el valor de la capacitancia (a) normal. (b) Incremento en la superficie del área, incremento en la capacitancia. (c) Decremento en la distancia de separación de las placas, incremento en la capacitancia [1].

Un capacitor con una capacitancia de un farad puede almacenar un coulomb de carga cuando el voltaje a través de las terminales es de $1[V]$. Un campo eléctrico existirá entre las dos placas de un capacitor si el voltaje es aplicado a una de las placas. El campo eléctrico resultante se debe a la diferencia entre las cargas eléctricas almacenadas en la superficie de cada placa. La capacitancia describe los efectos en el campo eléctrico debido al espacio de separación entre las dos placas.

La capacitancia depende de la geometría de los conductores y no de la fuente externa de carga o de la diferencia de potencial que se aplique sobre el capacitor. El espacio entre las dos

placas del capacitor es cubierto o llenado por el material dieléctrico. En general, el valor de la capacitancia es determinado por el material dieléctrico, la distancia entre placas, y el área de cada placa (esto se muestra en la figura 2). La capacitancia de un capacitor se puede expresar en términos de su geometría y la constante del material dieléctrico como se muestra en la ecuación 2.3 [1].

$$C = \varepsilon_r \frac{\varepsilon_0}{d} \tag{2.3}$$

donde,

C es la capacitancia $[F]$,

ε_r es la permitividad estática relativa (constante dieléctrica) del material entre las placas,

ε_0 es la permitividad del vacío, que es igual a $8.854x10^{-12}\left[\frac{F}{m}\right]$,

A es el área de cada placa en $[m^2]$,

d es la distancia de separación de las dos placas en $[m]$

El fenómeno de la capacitancia se relaciona con el campo eléctrico entre dos placas del capacitor. La fuerza del campo eléctrico entre las dos placas decrementa a medida que la distancia de separación entre las dos placas aumenta. Una baja fuerza de campo eléctrico o una separación mayor entre placas bajará el valor de la capacitancia. Las placas conductoras con una mayor área de superficie son capaces de almacenar una mayor cantidad de carga; por lo tanto, se obtiene un mayor valor de capacitancia al tener una mayor área de superficie.

2.2. Constante Dieléctrica

La brecha entre las dos superficies de un capacitor se llena con un material no-conductor, como goma, vidrio o madera lo cual separa los dos electrodos del capacitor. Este material tiene una cierta constante dieléctrica. La constante dieléctrica es una medida de la influencia que tiene el campo eléctrico. La capacitancia de red incrementará o decrementará dependiendo del tipo de material dieléctrico. La permitividad se refiere a la habilidad de un material de transmitir un campo eléctrico. En los capacitores, el incremento en la permitividad permite almacenar la misma cantidad de carga con un campo eléctrico menor, lo que lleva a un incremento en la capacitancia.

De acuerdo con la ecuación 2.3, la capacitancia es proporcional a la cantidad de la constante dieléctrica. Mientras que la constante dieléctrica entre las placas del capacitor aumenta, la

capacitancia también crece proporcionalmente. La capacitancia se puede expresar en términos de la constante dieléctrica como se muestra en la ecuación 2.4 [1]:

$$C = \varepsilon_r \cdot C_0 \qquad (2.4)$$

donde,

C es la capacitancia $[F]$,

ε_r es la permitividad estática relativa (constante dieléctrica) del material entre las placas,

C_0 es la capacitancia en ausencia de la constante dieléctrica.

El valor de la constante dieléctrica depende del material. El aire, por ejemplo, tiene una constante dieléctrica nominal igual a 1.0.

2.3. Efecto Piezoeléctrico

Existen otro tipo de sensores ampliamente utilizados hoy en día, los cuales se basan en el efecto piezoeléctrico (PE). Se encontró que cuando un esfuerzo mecánico se aplicaba a estos cristales, se producía electricidad y el voltaje generado era proporcional al esfuerzo aplicado. Sin embargo, el efecto inverso fue descubierto más tarde por Gabriel Lippmann en 1881. Las primeras aplicaciones se hicieron durante la Primera Guerra Mundial, con transductores piezoeléctricos ultrasónicos. Actualmente, la piezoelectricidad se utiliza comúnmente en la vida cotidiana. Por ejemplo, en el sensor de las bolsas de aire de los automóviles donde el material detecta el cambio en la aceleración del automóvil y envía una señal eléctrica que activa la bolsa de aire [33].

2.31. Funcionamiento

En general, en un material piezoeléctrico, cuando una tensión mecánica se aplica, aparecen cargas eléctricas en sus dos lados opuestos y a la inversa, si un campo eléctrico se aplica a este material, se genera una tensión mecánica en éste. Estos son los efectos piezoeléctricos directo e inverso (ver figura 2.3). De acuerdo con estas propiedades, un elemento piezoeléctrico es un tipo de transductor electromecánico (convertidor de energía) [7].

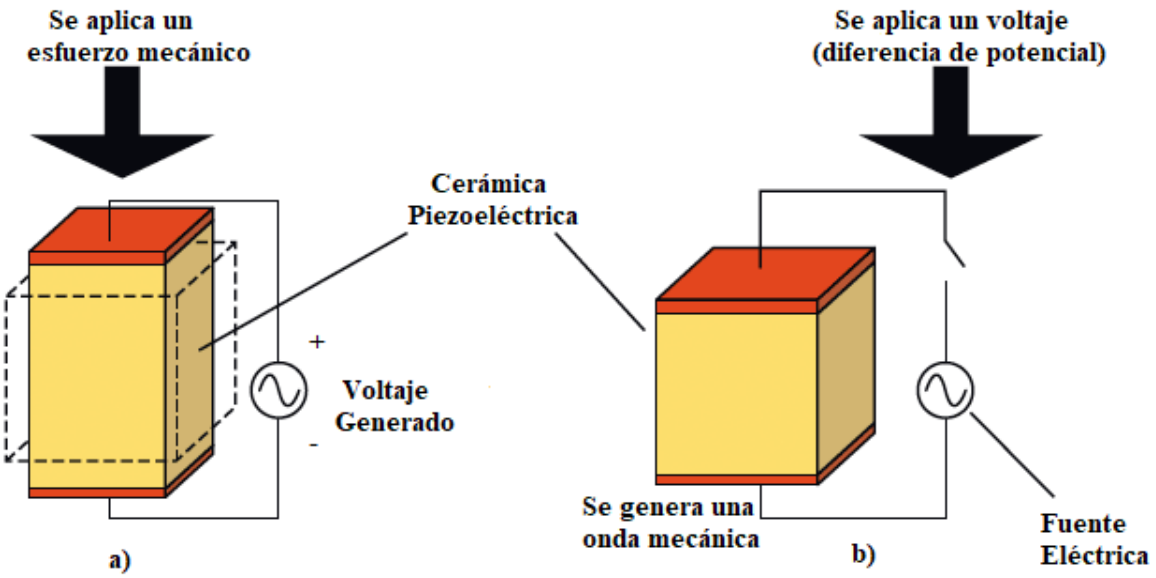


Figura 2.3. Diagrama de excitación del material piezoeléctrico: a) efecto directo y b) efecto inverso.

En el efecto directo se aplica una tensión mecánica y se genera una diferencia de potencial en la superficie del piezoeléctrico y en el caso inverso se aplica un voltaje y se genera ondas mecánicas [7].

La naturaleza de los materiales piezoeléctricos está íntimamente relacionada con la cantidad de dipolos eléctricos que se encuentran dentro de estos materiales. Estos dipolos pueden ser introducidos mediante iones en lugares de la red cristalina con entorno de carga asimétrica o por algunos grupos moleculares con propiedades eléctricas.

Los dipolos son pares de cargas eléctricas (carga positiva y carga negativa) separados espacialmente por una distancia pequeña ($\vec{d}$). Las cargas eléctricas se encuentran en las moléculas que forma el material y tienen enlaces covalentes (entra cargas positivas y negativas). Algunas moléculas forman dipolos permanentes por diferencia de electronegatividad entre sus átomos [7].

Estos dipolos tienden a tener la misma dirección cuando están uno junto a otro, y todos en conjunto forman regiones conocidas como Dominios de Weiss. Estos dominios normalmente tienen una orientación aleatoria, pero se pueden alinear usando un proceso de poleo, que es un proceso por el cual un fuerte campo eléctrico es aplicado a través del material. Sin embargo, no todos los materiales piezoeléctricos pueden ser polarizados.

La razón por la cual un material piezoeléctrico genera un voltaje es porque cuando se le aplica un esfuerzo mecánico, la estructura cristalina es perturbada y cambia la dirección de polarización $\vec{d}$ de los dipolos eléctricos. Dependiendo de la naturaleza del dipolo (si es introducido por grupos iónicos o moleculares), este cambio en la polarización puede ser

causado ya sea por una re-configuración de los iones dentro de la estructura cristalina o por una re-orientación de los grupos moleculares. Como consecuencia, mientras más grande sea el esfuerzo mecánico aplicado, mayor será el cambio en la polarización con lo cual se producirá mayor cantidad de electricidad [7].

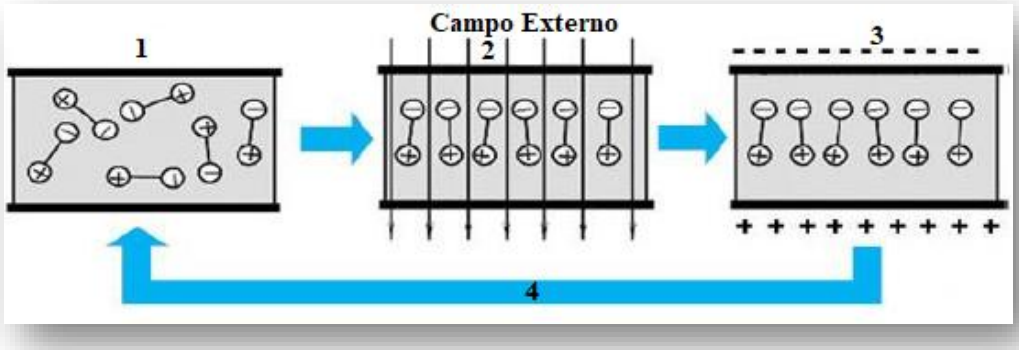


Figura 2.4. Orientación de los dipolos en un material piezoeléctrico [7].

1) En un material piezoeléctrico, los dipolos están parcialmente semi-orientados (polarizadas).
2) Si se produce una variación de presión y/o temperatura, los dipolos se orientan en una dirección dada.
3) Como resultado se generan regiones con carga eléctrica neta y se produce una diferencia de potencial entre los planos que delimita el dieléctrico.
4) En un tiempo posterior el campo externo desaparece y el medio regresa a su estado inicial por agitación térmica de las moléculas [7].

Una cerámica piezoeléctrica tradicional es una masa de cristales cerámicos de perovskita, cada uno contiene un pequeño ion metálico tetravalente, usualmente titanio o zirconio, en una red de iones metálicos divalentes más grandes, que normalmente son bario e iones O2. Bajo condiciones que confieren simetría tetragonal o romboédrica en los cristales, cada cristal tiene un momento dipolar.

El cambio en $\vec{d}$ aparece como un cambio de la densidad de carga superficial sobre las caras del cristal. Por ejemplo, al aplicar correctamente una fuerza de $2[kN]$ sobre un cubo de cuarzo de $1[cm^3]$ se puede producir un voltaje de $12500[V]$ [3].

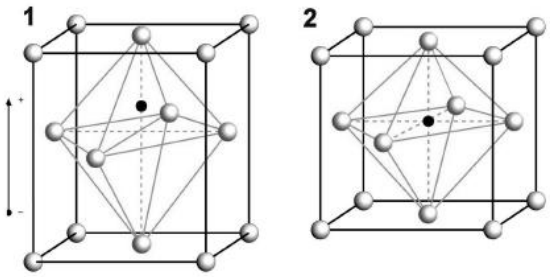


Figura 2.5. Estructura Perovskita de las cerámicas piezoeléctricas tipo PZT: 1) Debajo de la temperatura de Curie con dipolo $\vec{d}$. 2) Por encima de la temperatura de Curie sin dipolo [4].

Los materiales piezoeléctricos pueden ser manufacturados o creados por el hombre o se pueden ser encontrados en la naturaleza. El material piezoeléctrico más comúnmente encontrado de forma natural es el cuarzo, los materiales piezoeléctricos más eficientes son cerámicos y además fabricados por el hombre. Debido a que tienen una estructura cristalina compleja, el proceso con el que se fabrican es muy preciso y sigue pasos muy específicos.

2.4. Tereftalato de Polietileno

El tereftalato de polietileno comúnmente conocido como PET es un polímero lineal termoplástico obtenido por policondensación del ácido tereftálico $(C_6H_4(COOH)_2)$ adicionado con etilenglicol (CH_2OHCH_2OH) [5], y cuyas propiedades más importantes se muestran en la tabla 2.1.

Tabla 2.1. Propiedades Generales del PET [5].

Propiedades Generales del Tereftalato de Polietileno (PET)
Cristalinidad y transparencia, aunque admite cargas de colorantes
Buen comportamiento frente a esfuerzos permanentes
Alta resistencia al desgaste
Muy buen coeficiente de deslizamiento
Buena resistencia química
Buenas propiedades térmicas
Muy buena barrera a CO_2, aceptable barrera a O_2 y humedad
Compatible con otros materiales barrera que mejoran en su conjunto la calidad barrera de los envases y por lo tanto permiten su uso en mercados específicos
Alto grado de reciclabilidad
Aprobado para su uso en productos que deban estar en contacto con productos alimentarios
Alta rigidez y dureza
Altísima resistencia a los esfuerzos permanentes
Superficie barnizable
Gran indeformabilidad al calor
Muy buenas características eléctricas y dieléctricas
Alta resistencia a los agentes químicos y estabilidad a la intemperie
Propiedades ignifugas en los tipos aditivados
Alta resistencia al plegado y baja absorción de humedad que lo hacen muy adecuado para la fabricación de fibras
Procesable por soplado, inyección y extrusión.
Esterilizable por gamma y óxido de etileno
Liviano
Barrera contra gases

El PET es un material caracterizado por su ligereza, resistencia mecánica a la compresión y a las caídas, alto grado de transparencia y brillo, es una barrera contra gases, es 100% reciclable.

El PET presenta una estructura molecular con regularidad estructural necesaria para tener un potencial de cristalización. Aunque es un polímero polar, sus propiedades como aislante eléctrico a temperatura ambiente son buenas a altas frecuencias, debido a que el material, al estar por debajo de su temperatura de transición vítrea (Tg), tiene restricciones en la orientación de dipolos.

Tabla 2.2. Propiedades Físicas del PET [6].

Propiedades Físicas del PET	
Absorción de agua – Equilibrio (%)	< 0.7
Absorción de agua – en 24 horas (%)	0.1
Densidad $\left[{}^{g}/_{cm^3}\right]$	Amorfo: 1.3 – 1.4 Semicristalino: 1.45 – 1.51
Índice refractivo	1.58 – 1.64
Índice de oxigeno límite (%)	21
Inflamabilidad	Auto extinguible
Resistencia a los ultravioletas	Buena
Resistencia a la radiación	Buena

Tabla 2.3. Propiedades Mecánicas del PET [6].

Propiedades Mecánicas del PET	
Resistencia a la tracción hasta la deformación $[MPa]$	59
Resistencia a la tracción hasta la rotura $[MPa]$	No rompe
Alargamiento hasta la rotura	No rompe
Resistencia a la tracción $[MPa]$	190 - 160
Módulo de elasticidad en tracción $[MPa]$	2420
Resistencia a la flexión $[MPa]$	86
Resistencia al impacto Charpy	No rompe
Coeficiente de fricción	0.2 – 0.4
Dureza - Rockwell	M94 – 101
Dureza a la presión de la bola	117
Resistencia al impacto $[Jm^{-1}]$	13 - 35
Relación de Poisson	0.34 – 0.44 (orientado)

Tabla 2.4. Propiedades Térmicas del PET [6, 7].

Propiedades Térmicas del PET	
Calor especifico $[\boldsymbol{Jkg^{-1}K^{-1}}]$	1200 – 1350
Coeficiente de expansión térmica $\boldsymbol{x10^{-6}[K^{-1}]}$	20 – 80
Conductividad térmica $[\boldsymbol{Wm^{-1}K^{-1}}]$	0.15 – 0.4
Temperatura máxima de utilización $[°\boldsymbol{C}]$	115 – 170
Temperatura mínima de utilización $[°\boldsymbol{C}]$	-60 a -40
Temperatura máxima de utilización en continuo $[°\boldsymbol{C}]$	60
Temperatura reblandecimiento VICAT – (10N) $[°\boldsymbol{C}]$	79
Temperatura reblandecimiento VICAT – (50N) $[°\boldsymbol{C}]$	75
Temperatura reblandecimiento HDT A-1,8$[\boldsymbol{MPa}][°\boldsymbol{C}]$	69
Temperatura reblandecimiento HDT B-0,45$[\boldsymbol{MPa}][°\boldsymbol{C}]$	73
Coeficiente de expansión lineal $[°\boldsymbol{C^{-1}}]$	$< 6.10^{-5}$

Tabla 2.5. Propiedades Químicas del PET [6, 7].

Resistencia química	
Ácidos concentrados	Buena
Álcalis	Mala
Alcoholes	Buena
Grasas y aceites	Buena
Halógenos	Buena
Hidrocarburos Aromáticos	Aceptable
Aprobado para contacto con alimentos	Si

Tabla 2.6. Propiedades Ópticas del PET [6].

Propiedades ópticas del PET	
Transmisión de luz (%)	89
Refracción	1.576

Tabla 2.7. Propiedades Eléctricas del PET [6].

Propiedades eléctricas del PET	
Constante dieléctrica a 1 $[\boldsymbol{MHz}]$	3
Factor de disipación a 1 $[\boldsymbol{kHz}]$	0.002
Resistencia dieléctrica $[\boldsymbol{kVmm^{-1}}]$	17
Resistividad superficial $[\boldsymbol{\Omega/_{sq}}]$)	10^{13}
Resistividad de volumen $[\boldsymbol{\Omega cm}]$	$> 10^{14}$

Las aplicaciones eléctricas de los polímeros se utilizan ampliamente para desarrollar diversos tipos de películas y aplicaciones desde las películas ultradelgadas para capacitores de $1[\mu m]$ o menos hasta de $0.5[mm]$, utilizadas para aislamiento de motores. Debido a su alta resistencia dieléctrica y mecánica, el PET se utiliza como aislante de ranuras y fases en motores, condensadores, bobinas y transformadores. Así también ha servido para ser utilizado en la fabricación de conectores eléctricos de alta densidad, bloques terminales, circuitos integrados y partes electromecánicas.

2.5. Sensores capacitivos

El PET es un polímero relativamente barato y además es un polímero que se puede obtener de material reciclado. Hay que tener en cuenta que el PET no posee la propiedad de piroelectricidad ni Piezoelectricidad, pero, es un polímero polar, que le confiere a este material sea relativamente fácil de polarizar las cargas debido a efectos de ejercer presión sobre él.

Un sensor capacitivo, convierte los cambios de la capacitancia ya sea debido a algún movimiento, así como variaciones en la temperatura, etc. en una señal eléctrica [7]. De acuerdo a la ecuación (1), los sensores capacitivos realizan su función debido a la variación de al menos uno de los tres parámetros de un capacitor: distancia (d), área de las placas capacitivas (A), y la constante dieléctrica (ε_r)[8];

$$C = \varepsilon_r \frac{\varepsilon_0 A}{d} \tag{2.5}$$

donde,

C es la capacitancia $[F]$,

ε_r es la permitividad estática relativa (constante dieléctrica) del material entre las placas,

ε_0 es la permitividad del vacío, que es igual a $8.854x10^{-12}\left[\frac{F}{m}\right]$,

A es el área de cada placa en $[m^2]$,

d es la distancia de separación de las dos placas en $[m]$.

En la actualidad existe una gran variedad de sensores cuya base principal es el principio capacitivo descrito en la ecuación (1). Las funciones para las cuales se utilizan estos sensores van desde el sensado de humedad, a través del sensado de nivel, hasta el sensado de desplazamiento. El uso de los sensores basados en el efecto capacitivo tiene una gran variedad de aplicaciones industriales, incluyendo la industria automotriz. En la figura 1 se muestra un ejemplo de un sensor de proximidad capacitivo.

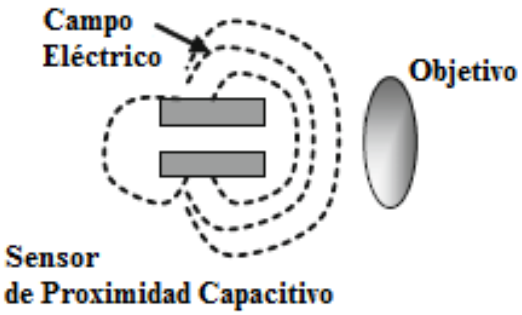


Figura 2.6. Sensor de proximidad basado en el efecto capacitivo.

El funcionamiento del sensor capacitivo mostrado en la figura 1 es tal que, al disminuir la distancia entre el sensor de proximidad y el objetivo, el efecto del campo eléctrico distribuido alrededor del capacitor experimenta un cambio, y esto es detectado por la unidad de control [9]. En nuestro caso lo que se buscará es una relación de la capacitancia en función de la presión.

2.6. Sensores polares y acondicionamiento

Como y se mencionó anteriormente, los sensores desarrollados tienen como material dieléctrico un polímero, el cual tiene cierta componente de polarización, esto se debe a que un polímero es una cadena de monómeros los cuales pueden estar orientados en diferentes direcciones, y presentan la propiedad de que pueden estar polarizados de fábrica (inicio). Dicha propiedad se debe a que son

elaborados por extrusión, lo que implica calentar el polímero y posteriormente estirar el producto, debido a esto muchas de las cadenas de monómeros se alinean y esto genera la pre polarización del material.

Debido a que el polímero esta formado por miles de filamentos, donde cada filamento puede tener al final un átomo del cual no se sabe en que dirección se va a polarizar y tomando en cuenta que por el método de fabricación ya hay muchos átomos alineados, existe una relación entre el área del sensor y la sensibilidad del mismo (lo cual no sucede con los materiales piezoeléctricos).

Ya que el tiempo para el desarrollo del sistema fue limitado ya no se realizó la caracterización de la matriz de sensores, lo cual queda como trabajo que se puede realizar a futuro para mejorar este prototipo. Es importante mencionar que sería conveniente realizar una etapa de pre acondicionamiento donde se sometan los sensores a una diferencia de potencial para ver el efecto que tiene en la polarización del material y como se relaciona con la sensibilidad.

2.7. Diseño y fabricación de los sensores

Cuando se diseñan sensores capacitivos, la geometría de estos juega un papel importante dado que la capacitancia está en función de la separación entre electrodos y del área de éstos. Los electrodos planos con geometrías cuadradas y cilíndricas son los más estudiados y los más utilizados en la práctica. Sin embargo, en un arreglo de estos sensores es importante considerar la rugosidad, efectos de borde en las placas paralelas porque esto suele inducir errores en la medición de la capacitancia real de cada sensor. En este capítulo siguiente se describe el diseño, método de fabricación de los sensores capacitivos, además el control y limitación en el depósito de material conductor sobre la película de PET.

2.7.1. Necesidades y limitaciones del diseño de los sensores

De estudios previos, se sabe que la geometría del área de sensado adecuado para cada sensor es la geometría circular debido a que la capacitancia parásita generada por el sensor contiguo es menor comparado con la geometría cuadrada. Adicionalmente es conveniente estimar la separación mínima entre dos sensores para evitar el llamado traslape electrónico, mejor conocido del inglés como *crosstalk* [2].

3. Sistema de Instrumentación

En la actualidad es difícil visualizar la vida sin una computadora personal (PC), desde el hogar, la oficina o la industria. En los último veinte años se ha adoptado rápidamente el uso de estos dispositivos los cuales se han integrado fuertemente en la vida del hombre, debido a esto se provocó una revolución en la instrumentación clásica o tradicional para la realización de sistemas de prueba, medición y automatización. Debido a la omnipresencia de la PC surgió el concepto de instrumentación virtual, el cual ofrece muchos beneficios a ingenieros y científicos que requieren tener buena productividad, exactitud y rendimiento en sus aplicaciones.

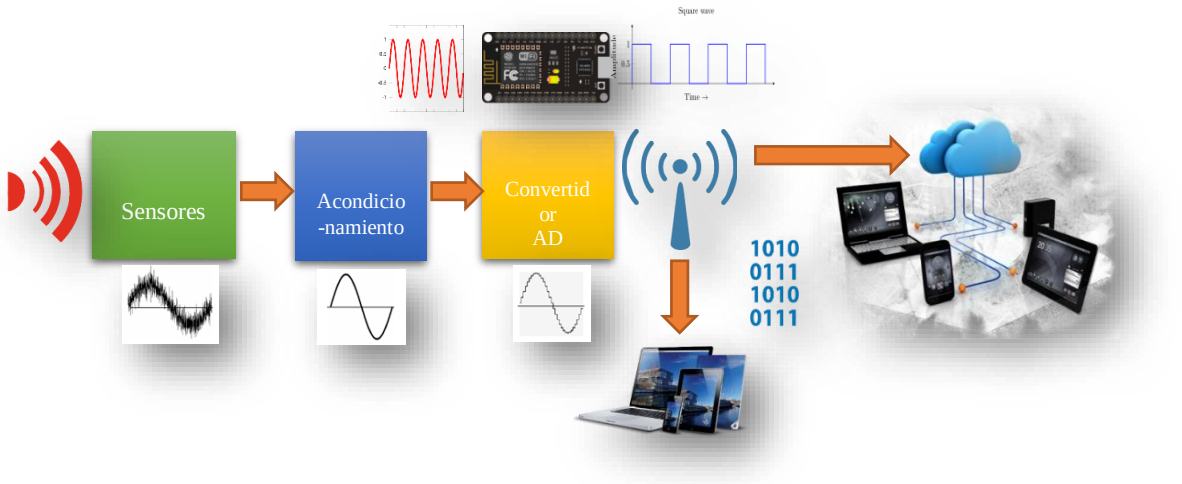


Figura 3.1. Diagrama de un Sistema de Instrumentación.

Un instrumento virtual, consiste de una computadora industrial estándar o PC, equipada con un poderoso software de aplicación y hardware (tarjetas de adquisición), que en conjunto realizan funciones de instrumentos de medición tradicionales.

Los instrumentos virtuales marcan la transición fundamental entre los sistemas de instrumentación clásicos centrados en hardware a nuevos sistemas centrados en software, los cuales explotan el poder de procesamiento, almacenamiento, el despliegue de la información y la capacidad de conexión (protocolos de comunicación) de las computadoras de escritorio.

En la actualidad, los sistemas de monitoreo y control de procesos se realizan bajo esquemas de instrumentación virtual debido al constante avance tecnológico que experimentamos. Los sistemas de instrumentación virtual son ampliamente utilizados por su bajo costo, gran flexibilidad y reconfigurabilidad, así como por su alto rendimiento y ahorro considerable en tiempo de desarrollo; razón por la cual constituyen una evolución natural respecto a los sistemas tradicionales de instrumentación [8].

3.1. Adquisición de Datos

La adquisición de datos (data acquisition o DAQ) es un proceso mediante el cual se miden parámetros físicos reales de variables como voltaje, corriente, temperatura, flujo, nivel, posición, etc. Dicha información se ingresa a la computadora para su posterior análisis y procesamiento con el fin de obtener una salida de información; la cual puede ser almacenada, desplegada en pantalla o enviada a un sitio remoto utilizando algún método de transmisión inalámbrica [8].

Para realizar el proceso de adquisición de datos se requiere el uso de un sensor o transductor (o un conjunto de ellos como es en este caso), el cual proporciona señales eléctricas proporcionales a la magnitud física de la variable a medir. Dicho sensor se puede conectar a la computadora utilizando tarjetas electrónicas para adquirir señales, o bien, puede provenir de sitios remotos si la adquisición se realiza mediante dispositivos de conexión inalámbrica tales como Bluetooth, WiFi o ZigBee. Además, también se pueden tener señales que provengan de puertos o buses de comunicación asociados a la computadora como son el puerto serial, el puerto USB o interfaces PCIX [8, 9].

Figura 3.2. Tarjetas de adquisición y generación de datos.

En general, la adquisición de datos se puede realizar tanto para ingresar señales a la computadora como para extraer señales de la misma; aunque el término “adquisición” implica solamente adquirir o ingresar datos. El método básico para realizar la manipulación de los datos incluye el uso de tarjetas de adquisición (DAQ), las cuales son dispositivos que se encargan de adquirir señales físicas provenientes de los sensores y de generar señales físicas provenientes de datos procesados por la computadora. Las tarjetas de adquisición son

la base de la instrumentación virtual, ya que mediante su uso se realiza la comunicación de parámetros físicos del mundo real con la computadora lo cual permite interactuar físicamente con señales tanto virtuales como reales.

Figura 3.3. Esquema básico de la adquisición de datos.

Las tarjetas de adquisición pueden adquirir o generar señales de acuerdo a su funcionalidad y características comerciales. En general, se pueden configurar para su uso en dos modos diferentes, los cuales a su vez se pueden subdividir en tres modos:

Tabla 3.1. Modos comunes de operación en las tarjetas de adquisición de datos.

Adquisición de Señales	Generación de Señales
Analógicas	Analógicas
Digitales	Digitales
Contadores/Temporizadores	Contadores/Temporizadores

En estos seis casos diferentes se pueden manipular señales tanto analógicas como digitales provenientes de sensores que midan algún parámetro físico o generar señales físicas procesadas por la computadora. El uso de las tarjetas de adquisición involucra el uso de convertidores analógico-digital (ADCs) y convertidores digital-analógico (DACs) para la conversión de las señales de analógica a digital y de digital a analógica, respectivamente, ya que las señales procesadas por la computadora son señales digitales y se requiere del proceso de conversión. Además, se tienen procesos de acondicionamiento de las señales para su manipulación correcta, los cuales involucran métodos de amplificación, filtrado, aislamiento y/o linealización para evitar efectos negativos como el ruido, sobrecargas, aliasing, no linealidad y otros [10].

3.2. Lenguajes de programación

Hoy en día se tienen varios lenguajes de programación los cuales pueden utilizarse para desarrollar aplicaciones de instrumentación virtual en diferentes áreas de estudio. Estos lenguajes tienen en común el hecho de que se basan en conjuntos de instrucciones de texto creando líneas de código. Como ejemplos de estos lenguajes se tienen: C/C++, C#, Java, Phyton, por mencionar los más utilizados. Dichos lenguajes ofrecen diferentes ventajas y desventajas entre sí, las cuales permiten el desarrollo de interfaces virtuales aplicables a la instrumentación [8].

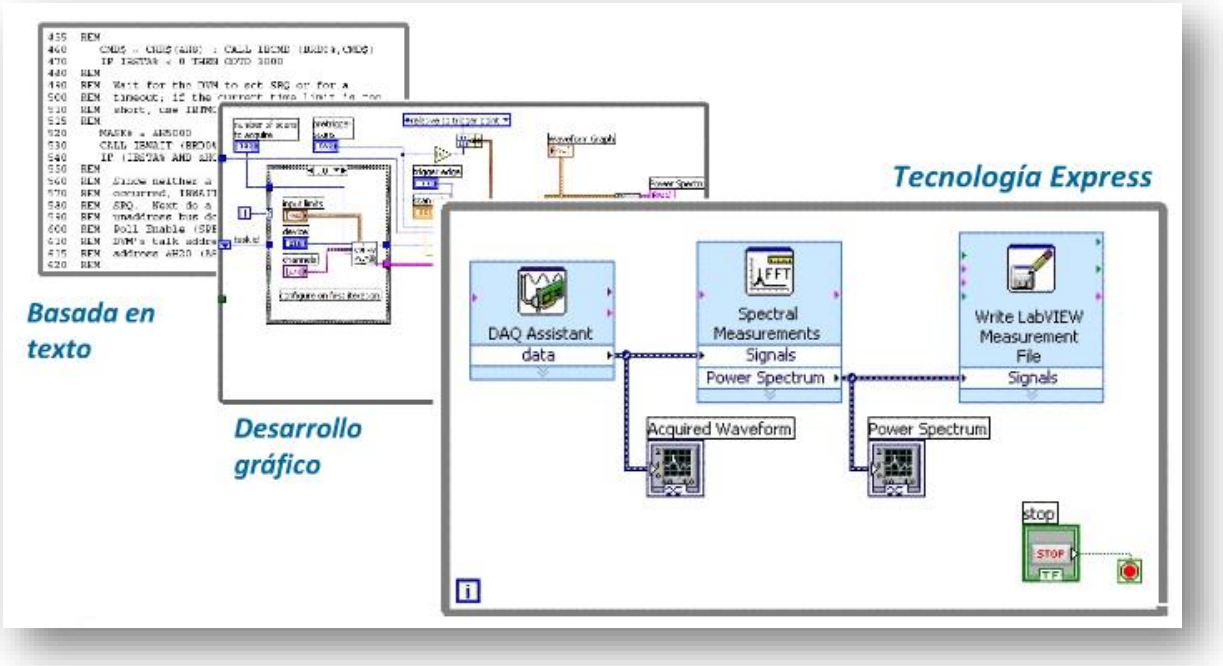


Figura 3.4. Evolución en la programación.

Sin embargo, la instrumentación virtual se basa en la interacción del usuario con interfaces computacionales gráficas para el control y monitoreo de sistemas físicos, por lo cual los lenguajes gráficos ofrecen mayores ventajas respecto a los lenguajes tradicionales basados en texto. El lenguaje gráfico —también llamado lenguaje G— más utilizado para desarrollar aplicaciones de instrumentación virtual es el LabVIEW® (Laboratory Virtual Instrument Engineering Workbench) desarrollado por la empresa National Instruments en 1986, el cual elimina múltiples detalles sintácticos asociados con los lenguajes basados en texto, ya que se trata de un modelo de programación gráfica con el cual se tienen diferentes ventajas en relación a los lenguajes mencionados anteriormente [11].

Los códigos gráficos incluyen una interfaz de usuario completamente gráfica y un código fuente basado en el uso de bloques de conexión interconectados mediante cables. La creación de los lenguajes de programación gráfica, y su inherente evolución, ha permitido el desarrollo de múltiples protocolos e interfaces de comunicación creados con el objetivo de abarcar una amplia gama de aplicaciones industriales programables en lenguaje gráfico, lo cual ha constituido la base de la instrumentación virtual [10].

En la actualidad existen diferentes protocolos de comunicación utilizados para transmitir y recibir datos de múltiples dispositivos. En el ámbito de la instrumentación virtual se encuentra un conjunto de protocolos e interfaces de comunicación aplicables a la transferencia de datos entre la computadora con la aplicación virtual ejecutándose y los periféricos externos.

3.3. Universal Serial Bus

3.3.1. Introducción

Las siglas USB provienen del inglés: Universal Serial Bus (Bus Serie Universal), por lo que como su nombre indica, se trata de un sistema de comunicación entre dispositivos electrónicos-informáticos que sólo transmite una unidad de información a la vez. El bus USB se compacta en un cable de cuatro hilos, dos para datos y dos para alimentación. Esto supone un gran ahorro, tanto de espacio como de material. De acuerdo a estos parámetros, una de las principales ventajas que se obtiene de USB es precisamente su diseño, una fuente desventaja es que no es robusto.

Introducido y estandarizado por un grupo de compañías (Compaq, DEC, IBM, Intel, Microsoft, NEC, HP, Lucent, Philips y Nortel) en 1995. La idea fundamental fue la de reemplazar la gran cantidad de conectores disponibles en las computadoras personales simplificando la conexión y configuración de dispositivos logrando grandes anchos de banda [12].

En resumen, existen cuatro versiones de USB, las cuales se muestran en la tabla 3.2.

Tabla 3.2. Versiones más comunes de USB [12].

Versión	Fecha de Lanzamiento	Velocidad de Transferencia
USB 1.0	Enero 1996	1.5 [Mb/s]
USB 1.1	Septiembre 1998	12 [Mb/s]
USB 2.0	Abril 2000	480 [Mb/s]
USB 3.0	Noviembre 2008	5 [Gb/s]

El USB organiza el bus en una estructura de árbol descendente, con múltiples dispositivos conectados a un mismo bus, en la que los dispositivos conocidos como concentradores (hubs), encaminan las señales desde un dispositivo al anfitrión (host) o viceversa.

Primero está el controlador del bus, este es la interfaz entre el bus USB y el bus de la computadora, a él se conectan los dispositivos USB. A un concentrador se puede conectar uno o más dispositivos, que a su vez pueden ser otros concentradores, así tenemos varios dispositivos conectados a un sólo controlador; como máximo 127 [14].

Es conveniente resaltar que todos los dispositivos deben seguir reglas de comportamiento básicas. Por tanto, todos los dispositivos se configuran de la misma forma, y es mucho más fácil gestionar los recursos que proveen; sin embargo, esto no significa que todos los dispositivos son iguales, sino, que todos tienen un sistema de configuración idéntico.

Para proteger sus identidades, existe una clasificación estandarizada, (gestionada por el controlador) y en función de esa clasificación, los dispositivos se manejan de una forma u otra, siempre cumpliendo los estándares, permitiendo entre otras cosas, una simplificación en la gestión de los dispositivos, ya que un mismo controlador sirve para varios dispositivos de diferentes tipos, aparte de poder tener un número casi ilimitado de dispositivos idénticos en un mismo sistema (siempre se pueden añadir nuevos controladores) [13] [14].

Los sistemas se pueden clasificar según su direccionalidad y momento en el que se efectúa la transmisión en los diferentes tipos mostrados en la tabla 3.3.

Tabla 3.3. Clasificación por direccionalidad de la comunicación USB [12].

Direccionalidad	Descripción
Símplex	En este modo solo es posible la transmisión en un sentido, del terminal que origina la información hacia el que la recibe y procesa. Un ejemplo claro de este tipo son las emisoras de radiodifusión.
Semidúplex (half -dúplex)	Permite la transmisión en ambos sentidos de manera alterna. Un ejemplo de este tipo son las transmisiones efectuadas por radioaficionados.
Dúplex (full – dúplex)	Consiste en la transmisión en ambos sentidos de manera simultánea. Esta forma de trabajo es la más eficiente. Un ejemplo son las comunicaciones telefónicas.

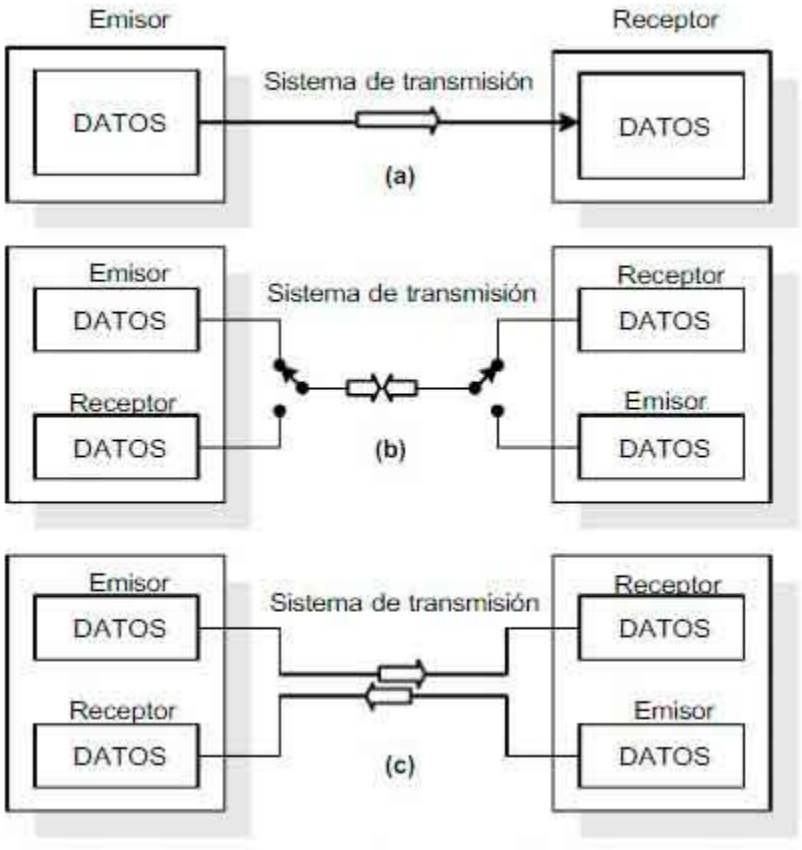


Figura 3.5. Sistemas de transmisión: simplex (a), semiduplex (b) y dúplex (c).

En el presente trabajo la versión de USB utilizada con las tarjetas de adquisición es la 2.0, el cual tiene cuatro líneas y el modo de transferencia de datos es “Semiduplex” [13].

3.3.2. Funciones

Dentro de la terminología USB, la computadora personal (PC) que soporta este tipo de bus, se denomina anfitrión USB (USB Host); mientras que por su parte y dentro de la misma terminología, todo dispositivo conectado al anfitrión, se denomina periférico.

A diferencia de los dispositivos y los concentradores, existe tan solo un anfitrión dentro del sistema USB, que como ya se mencionó es la computadora personal, particularmente una porción del mismo denominado controlador USB del anfitrión. Este tiene la misión de hacer de interfaz entre la computadora y los diferentes dispositivos.

Existen algunas particularidades respecto a este controlador. Su implementación es una combinación de hardware y software todo en uno. Puede proveer de uno o dos puntos de conexión iniciales, denominados concentradores raíz, a partir de los cuales y de forma ramificada se conectan los periféricos.

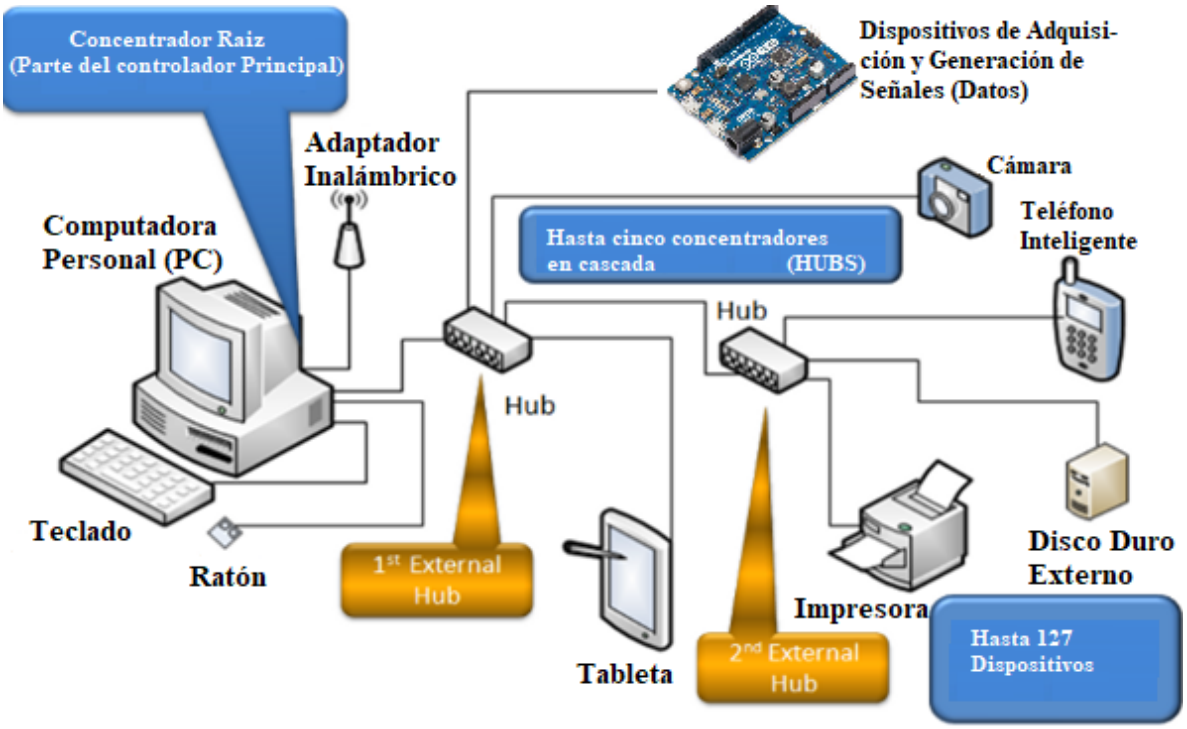


Figura 3.6. Topología USB.

El anfitrión USB trabaja con los diferentes dispositivos valiéndose del controlado, compuesto por una parte de hardware y otra de software, de esta forma conjunta, el anfitrión es responsable a nivel de hardware, de los siguientes aspectos dentro del sistema USB:

Tabla 3.4. Funciones del anfitrión USB en la mayoría de los sistemas que lo incluyen [14].

Funciones del anfitrión USB
Detectar tanto la conexión de nuevos dispositivos USB al sistema como la remoción de aquellos ya conectados, configurarlos y ponerlos a disposición del usuario.
Administrar y controlar el flujo de datos entre el anfitrión y los dispositivos USB.
Administrar y regular los flujos de control entre el host y los dispositivos USB.
Recolectar y resumir estadísticas de actividad y estado de los elementos del sistema.
Proveer de una cantidad limitada de energía eléctrica para aquellos dispositivos que pueden abastecerse con tan solo la energía proveniente de la computadora.

Quizá una de las mayores ventajas para los equipos actuales es el hecho de que el puerto USB solamente necesita una interrupción (IRQ) y una dirección de memoria y todos los dispositivos conectados a él, solamente necesitan una etiqueta (ID) para su identificación dentro de la cadena de 127 dispositivos, sin necesitar más recursos [14].

El manejo de los dispositivos USB se hace por software, concretamente por el propio sistema operativo, por lo que los dispositivos USB son más fáciles de fabricar y por tanto más baratos. Además, USB es una tecnología abierta por la que no hay que pagar derechos, lo que siempre abarata los costos de fabricación.

Cada dispositivo puede tener un cable de hasta 5 [m] de longitud. Además, conectándolos en cadena, el último dispositivo puede estar a 635 [m] de la computadora. Cada dispositivo puede funcionar como concentrador, es decir, incluir uno o más conectores USB, de modo que podamos conectar un dispositivo a otro en cadena [14].

Uno de los inconvenientes del puerto USB es que, actualmente, la mayoría suministra únicamente 900 [mA] para los dispositivos conectados. Esto se puede solucionar adquiriendo un concentrador USB con toma de alimentación eléctrica externa.

3.3.3. Arquitectura General

Como se mencionó anteriormente, el USB está dado esencialmente por un cable especialmente diseñado para la transmisión de datos entre la computadora y diferentes periféricos, que pueden acceder simultáneamente al mismo con el fin de recibir o transmitir datos. Todos los dispositivos conectados acceden al canal o medio para transmitir sus datos de acuerdo a las normas de administración del host regido por un protocolo que consecutivamente va dando la posibilidad de transmitir a cada periférico.

La arquitectura del bus garantiza la posibilidad de que los periféricos sean conectados y desconectados del host mientras este y otros periféricos están operando normalmente, característica a la que se denomina "*conectar y usar*" (plug and play), sin perjuicio para ningún dispositivo en funcionamiento.

Todos los dispositivos USB responden también a un mismo patrón estandarizado, que más allá de las características propias de cada fabricante, comprende los mismos elementos funcionales los cuales son:

Transreceptor

El cable USB está compuesto por solo cuatro cables: Vbus (Voltaje d e polarización) , D+, D- y GND (Tierra). La información y los datos se mueven por los cables D+ y D-, con las velocidades antes mencionadas (tabla 3.2.). Fabricado dentro del mismo chip controlador de periférico, y puede verse como la interfaz misma de un dispositivo externo contra el resto del sistema.

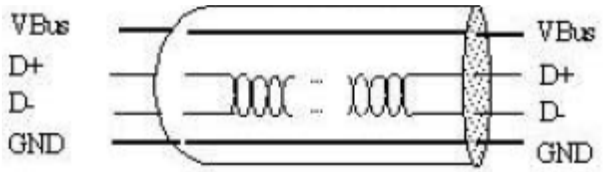


Figura 3.7. Cable USB [17].

Motor de Interfaz Serie (Serial Interface Engine)

Tiene la función de seriar y agrupar las transmisiones, además maneja los protocolos de comunicación, las secuencias de paquetes, el control de verificación por redundancia cíclica (CRC) y la codificación NRZI (No Retorno al Cero Invertido).

Unidad de interfaz de función

Este elemento administra los datos que son transmitidos y recibidos por el cable USB. Se basa y apoya en el contenido y estado de los FIFOs (First Input First Output). Monitorea los estados de las transacciones, los buffers FIFO, y solicita atención para diversas acciones a través de interrupciones contra el CPU del host.

FIFOs (primero en entrar – primero en salir)

El controlador 8x930Ax, tiene un total de 8 buffers tipo FIFO, cuatro de ellos destinados a la transmisión y cuatro destinados a la recepción de datos. Tanto para la transmisión como para la recepción, los buffers soportan cuatro tareas o funciones, numeradas de 0 a 3. La función 0 tiene reservado en el buffer en espacio de 16 [B], y se dedica a almacenar información de control relacionada a las transferencias. La función 1 es configurable para disponer de más de 1025 [B], y finalmente las funciones 2 y 3 disponen cada una de 16 [B]. Estas tres últimas funciones se emplean para el control de interrupciones y transmisiones [13, 14].

Es importante destacar que el controlador del periférico es totalmente programable, empleando el conjunto de instrucciones MCS51 o MSC251, ambos productos de Intel que ha de ser más de interés de las empresas fabricantes de dispositivos externos USB [13, 14].

Lo diferentes tipos de enchufes y puertos USB disponibles actualmente se muestran en la figura 3.8.

Figura 3.8. Puertos y conectores USB [15].

El conector que utilizan las tarjetas de desarrollo (Arduino Due) utilizadas en este trabajo es el USB 2.0 Mico Tipo B. Los pines del conector USB 2.0 Tipo A que es el que comúnmente tienen las computadoras personales se muestran en la figura 3.9.

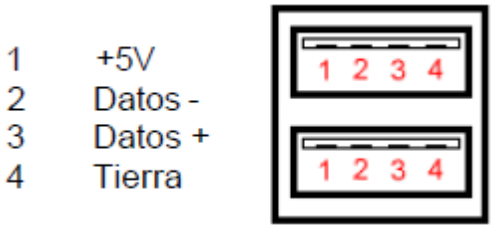


Figura 3.9. Pines del conector USB [16].

3.3.4. Protocolo y transmisión de datos

Toda transferencia de datos o transacción que emplee el bus, involucra al menos tres paquetes de datos. Cada transacción se da cuando el controlador de host decide qué dispositivo hará uso del bus, para ello envía un paquete al dispositivo específico. Cada uno de los mismos tiene un número de identificación, otorgado por Controlador de Host cuando la computadora inicia o bien, cuando un dispositivo nuevo es conectado al sistema. De esta forma, cada uno de los periféricos puede determinar si un paquete de datos es o no para sí [18].

Técnicamente este paquete de datos se denomina "*Paquete Ficha*" (Token Packet). Una vez que el periférico afectado recibe el permiso de transmitir, arranca la comunicación y sus tareas específicas; el mismo informará al host con otro paquete que ya no tiene más datos que enviar y el proceso continuará con el siguiente dispositivo. Este protocolo tiene un sistema muy eficiente de recuperación de errores, empleando uno de los modelos más seguros como es el CRC (Código de Redundancia Cíclica) [18]. Y puede estar implementado al nivel de software y/o hardware de manera configurable. De hecho, si el control es al nivel de hardware, no vale la pena activar el control por software, ya que sería duplicar tareas innecesariamente.

Transmisión Asincrónica

En la transmisión serial la información generada en el transmisor es recuperada en la misma forma en el receptor, para lo cual es necesario ajustar adecuadamente un sincronismo entre ambos extremos de la comunicación. Para ello, tanto el receptor como el transmisor deben disponer de relojes que funcionen a la misma frecuencia y posibilite una transmisión exitosa. Como respuesta a este problema surgió la transmisión asincrónica, empleada masivamente años atrás para la comunicación entre los equipos servidores conocidos como hosts y sus terminales. En este modelo cabe entender que ambos equipos poseen relojes funcionando a la misma frecuencia, por lo cual, cuando uno de ellos desea transmitir, prepara un grupo de bits encabezados por un bit conocido como de arranque, un conjunto de 7 u 8 bits de datos, un bit de paridad (para control de errores), y uno o dos bits de parada. El primero de los bits enviados anuncia al receptor la llegada de los siguientes, y la recepción de los mismos es efectuada. El receptor conocer perfectamente cuántos bits le llegarán, y da por recibida la información cuando verifica la llegada de los bits de parada. El esquema de los datos se muestra en la figura siguiente.

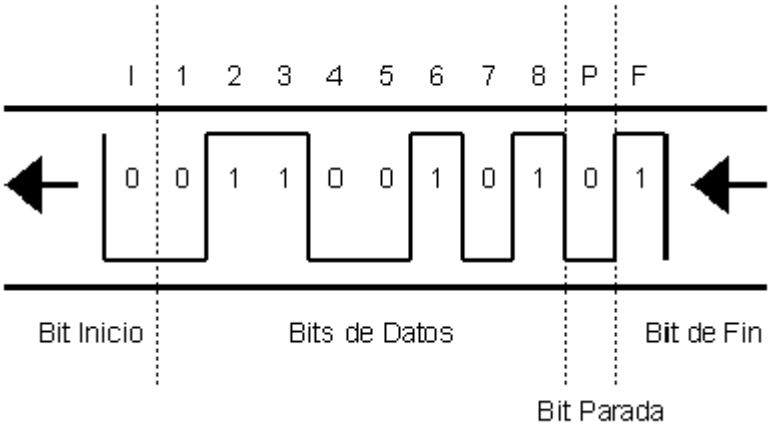


Figura 3.10. Esquema de la transmisión asincrónica [17].

Se denomina transmisión asincrónica no porque no exista ningún tipo de sincronismo, sino porque el sincronismo no se halla en la señal misma, más bien son los equipos mismos los que poseen relojes que posibilitan la sincronización. La sincronía o asincronía siempre se comprende a partir de la señal, no de los equipos de transmisión o recepción.

Transmisión sincrónica

En este tipo de transmisión, el sincronismo viaja en la misma señal, de esta forma la transmisión puede alcanzar distancias mucho mayores como también un mejor aprovechamiento de canal. En la transmisión asincrónica, los grupos de datos están compuestos por generalmente 10 bits, de los cuales 4 son de control. Evidentemente el rendimiento no es el mejor. En cambio, en la transmisión sincrónica, los grupos de datos o paquetes están compuestos por 128 [b], 1024 [b] o más, dependiendo de la calidad del canal de comunicaciones.

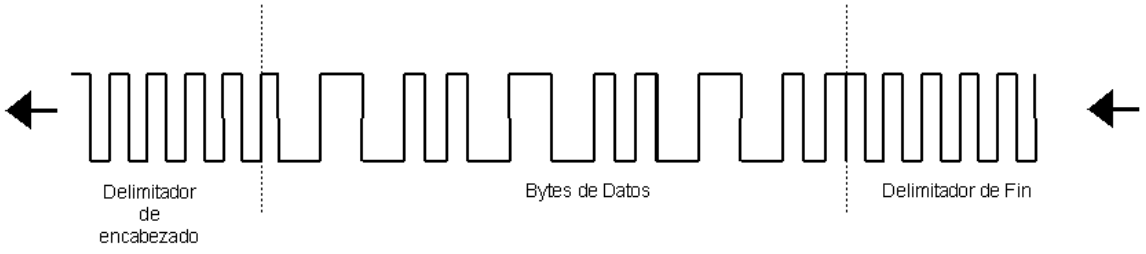


Figura 3.11. Esquema de la transmisión sincrónica [17].

Las transmisiones sincrónicas ocupan en la actualidad gran parte del mundo de las comunicaciones seriales, especialmente las que emplean el canal telefónico.

Transmisión isocrónicas

Inicialmente vale la pena aclarar el origen de este término tan extraño, ISO (algún) CRONOS (tiempo). La transmisión isocrónica ha sido desarrollada especialmente para satisfacer las demandas de la transmisión multimedial por redes, esto es integrar dentro de una misma transmisión, información de voz, video, texto e imágenes. La transmisión isocrónica es una forma de transmisión de datos en la cual los caracteres individuales están solamente separados por un número entero de intervalos, medidos a partir de la duración de los bits. Contrasta con la transmisión asincrónica en la cual los caracteres pueden estar separados por intervalos aleatorios. La transferencia isocrónica provee comunicación continua y periódica entre el host y el dispositivo, con el fin de mover información relevante a un cierto momento. La transmisión isocrónica se encarga de mover información relevante a algún tipo de transmisión, particularmente audio y video.

Transmisión Bulk

La transmisión Bulk, es una comunicación no periódica, explosiva típicamente empleada por transferencias que requieren usar todo el ancho de banda disponible o en su defecto son demoradas hasta que el ancho de banda completo esté disponible. Esto implica particularmente movimientos de imágenes o video, donde se requiere de gran potencial de transferencia en poco tiempo.

Transmisiones de control

Es un tipo de comunicación exclusivamente entre el host y el dispositivo que permite configurar este último, sus paquetes de datos son de 8, 16, 32 o 64 bytes, dependiendo de la velocidad del dispositivo que se pretende controlar.

Transmisiones de interrupción

Este tipo de comunicación está disponible para aquellos dispositivos que demandan mover muy poca información y poco frecuentemente. Tiene la particularidad de ser unidireccional, es decir del dispositivo al host, notificando de algún evento o solicitando alguna información. Su paquete de datos tiene las mismas dimensiones que el de las transmisiones de control.

3.4. Arduino

Arduino es un proyecto nacido en el año 2005 con la idea de desarrollar una placa de hardware libre integrada con un microcontrolador y una interfaz para programarlo. Está diseñado y construido para que sea muy fácil utilizarlo y en él se pueden desarrollar proyectos multidisciplinarios. La placa se puede comprar a un precio accesible o bien cada persona puede armarlas libremente es sus casas, dado que toda la información del esquema del circuito o el PCB son de licencia libre y se encuentran en la red (Internet) muy fácilmente [34].

El Hardware es una placa que tiene un microcontrolador Atmel y varios puertos de entrada/salida digitales y analógicos, existen complementos conocidos como "*shields*", estos complementos brindan una mayor autonomía al programador y usuario de la placa lo cual hace de Arduino un entorno muy amigable y simple de trabajar.

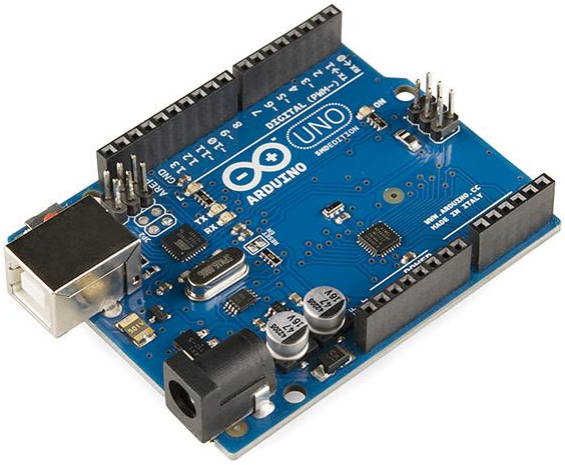


Figura 3.12. Arduino UNO [34]

A su vez, es una plataforma que cuenta con una gran cantidad de sensores y periféricos desarrollados y pensados para ser utilizados en Arduino, esto facilita mucho la integración de la placa con el mundo que lo rodea, pudiendo de esta manera desarrollar prácticamente cualquier cosa sin incurrir en un alto costo y perder mucho tiempo en el diseño, debido a esto es ampliamente utilizado en muchas instituciones educativas a nivel internacional.
En general cualquier placa o tarjeta de desarrollo Arduino se compone de las partes mostradas en la tabla 3.5.

Tabla 3.5. Partes generales de las tarjetas de desarrollo Arduino Due [34].

Parte	Descripción
Entradas/Salidas	Existen varios modelos distintos de la placa Arduino, pero todas tienen en común una cantidad definida de periféricos de entrada y salida, según la placa tendrá más o menos puertos de entrada y salida.
Alimentación	La mayoría de las placas de Arduino se alimentan con 5 [V] aunque algunas lo pueden hacer con tensiones de hasta 12 [V].
Comunicación	Existen algunos puertos destinados a la comunicación de la placa con algún otro dispositivo, estos son los pines RX y TX, mediante estos puertos es que se establece la comunicación como por ejemplo con un display LCD.
Complementos	Los complementos o Shields son de gran ayuda para los programadores ya que reducen el tiempo de desarrollo y entregan un producto solido que podemos utilizar fácilmente.

Debido a que Arduino es una plataforma electrónica de open-hardware y open-source, flexible y fácil de usar para la creación de prototipos, es una de las razones principales por la cual se utilizó para la realización de este proyecto. Puede ser usado tanto para desarrollar objetos interactivos autónomos, como para ser conectado a una computadora personal y comunicarse con varios tipos de software (como por ejemplo Flash, Matlab, Python) en este caso LabVIEW.

Su hardware libre está basado en una sencilla placa con un microcontrolador y un entorno de desarrollo que implementa el lenguaje de programación Processing/Wiring. Al ser open-hardware, tanto su diseño como su distribución es libre: es decir, puede utilizarse libremente para el desarrollo de cualquier tipo de proyecto sin haber adquirido ninguna licencia.

El entorno de desarrollo integrado libre puede ser descargado de forma gratuita. Los ficheros de diseño de referencia (CAD) están disponibles bajo una licencia abierta, por lo que cada uno es libre de adaptarlos a sus necesidades [35].

3.4.1. Arduino Due

Arduino Due es la primera tarjeta de desarrollo construida con un microcontrolador de 32 bit CortexM3 ARM el cual puede ser programado mediante el IDE de Arduino. posee 54 pines digitales de entrada y salida (de los cuales 12 pueden ser usados como salidas PWM), 12 entradas análogas, 2 salidas análogas, 4 UART (transmisor/receptor universal asíncrono del inglés: universally asynchronous receiver/transmitter), cristal oscilador de 84 [MHz], una conexión compatible con USB-OTG, 2 TWI, Jack de poder, conexión JTAG, botón reset y

un botón borrar. También tiene otras funcionalidades como audio, DMA y una librería experimental para multitareas [19].

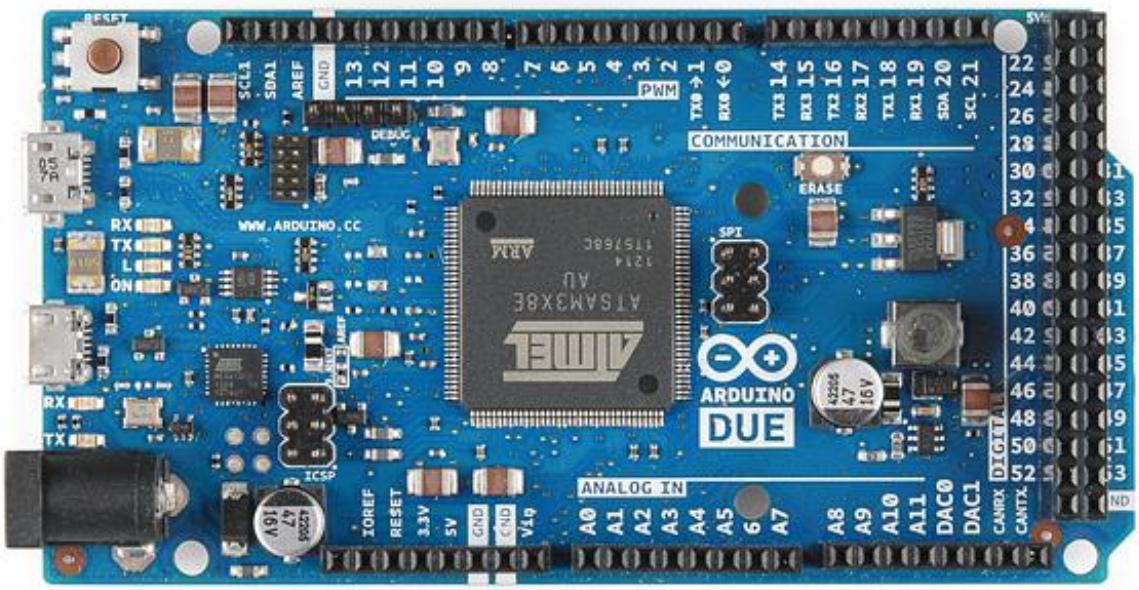


Figura 3.13. Tarjeta de desarrollo Arduino Due [19, 20].

Para realizar una compilación de código de programación y que sea compatible con esta tarjeta, se requiere una versión superior al IDE de Arduino 1.5.0.

De acuerdo a las limitaciones de voltaje de sistema impuestas por el Atmel SAM3X8E, la mayoría de los Shield de Arduino que funcionan con 5 [V] no funcionarán correctamente con la tarjeta Arduino Due. A diferencia de otras tarjetas, Arduino Due trabaja con 3,3 [V], pudiendo tolerar un voltaje máximo en sus pines I/O de 3.3 [V]. Las características generales de esta tarjeta se muestran en la tabla 3.5 [19]:

Tabla 3.6. Características generales de la tarjeta de desarrollo Arduino Due.

Arduino Due	
Microcontrolador	AT91SAM3X8E
Voltaje de operación	3.3 [V]
Voltaje recomendado de entrada	7 - 12 [V]
Pines de entrada y salida digitales	54 pines I/O (12 PWM)
Pines de entrada análogos	12
Pines de salida análogos	2
Corriente de salida total en los pines I/O	130 [mA]
Corriente DC máxima en el pin de 3.3 [V]	800 [mA]
Corriente DC máxima en el pin de 5V	800 [mA]
Memoria Flash	512 [KB]
SRAM	96 [kB]
Frecuencia de reloj	84 [MHZ]

3.4.2. Arduino y el Puerto Serie

Generalmente, todas las placas Arduino disponen al menos de una unidad UART. Las placas Arduino UNO y Mini Pro disponen de una unidad UART que operan a nivel TTL 0[V] / 5[V], por lo que son directamente compatibles con la conexión USB. Por su parte, Arduino Mega y Arduino Due disponen de 4 unidades UART TTL 0[V] / 5[V] [22].

Los puertos serie están físicamente unidos a distintos pines de la placa Arduino. Lógicamente, mientras usamos los puertos de serie no podemos usar como entradas o salidas digitales los pines asociados con el puerto serie en uso.

En Arduino UNO y Mini Pro los pines empleados son 0 (RX) y 1 (TX). En el caso de nuestra placa, el Arduino Due, cuenta con cuatro puertos de serie, el puerto serie 0 está conectado a los pines 0 (RX) y 1 (TX), el puerto serie 1 a los pines 19 (RX) y 18 (TX) el puerto serie 2 a los pines 17 (RX) y 16 (TX), y el puerto serie 3 a los pines 15 (RX) y 14 (TX) [22].

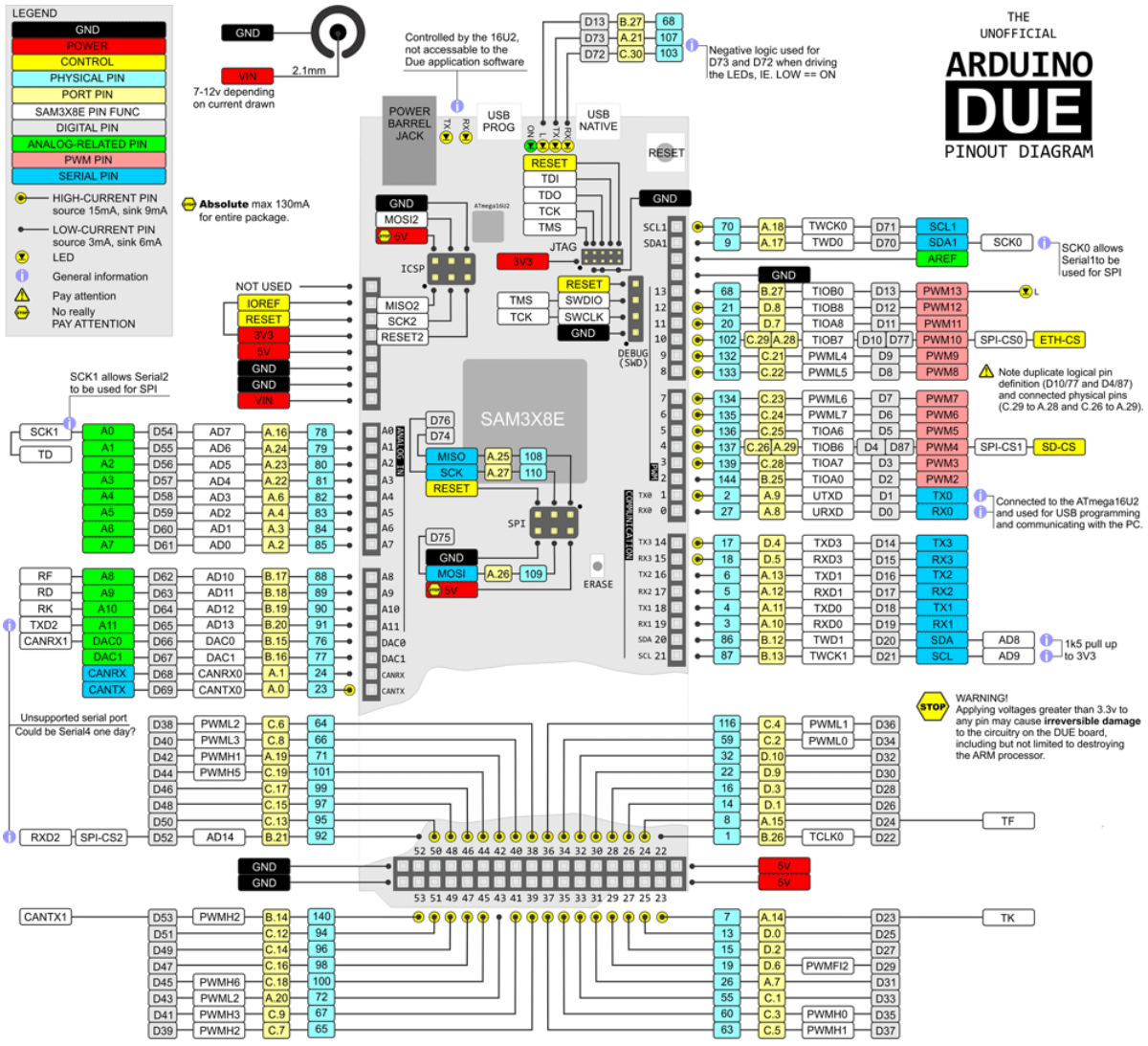


Figura 3.14. Diagrama de Pines para la tarjeta de desarrollo Arduino Due [25].

Muchos modelos de placas Arduino disponen de un conector USB o Micro USB conectado a uno de los puertos de serie, lo que simplifica el proceso de conexión con una computadora. Sin embargo, algunas placas, como por ejemplo la Mini Pro, prescinden de este conector por lo que la única forma de conectarse a las mismas es directamente a través de los pines correspondientes. La interfaz USB de la computadora nos permite tener un puerto serie virtual que usamos para conectar con Arduino. Además de comunicación, el USB se ocupó como una interfaz de alimentación. Al conectar las tarjetas Arduino a la computadora mediante un cable USB, la computadora instala un puerto serie virtual (COM) desde el cual se accede al puerto serie de la tarjeta Arduino. Por este motivo es necesario instalar en el ordenador los drivers de interfaz USB de los microcontroladores de Atmel, esto se hace automáticamente al instalar el IDE de Arduino [22].

3.5. Entorno de desarrollo integrado para Arduino

Dado que el Arduino es una pequeña computadora que ejecuta una serie de códigos que previamente le hemos introducido, se requiere de un programa para poder meter estos códigos a la propia placa. Este programa se llama IDE, que significa "Integrated Development Environment" ("Entorno de Desarrollo Integrado"). Este IDE se instala en nuestra PC, es un entorno muy sencillo de usar y en él se escribe el programa que la placa de desarrollo Arduino ejecutará. Una vez escrito, lo cargaremos a través del USB y Arduino comenzará a trabajar de forma autónoma [21].

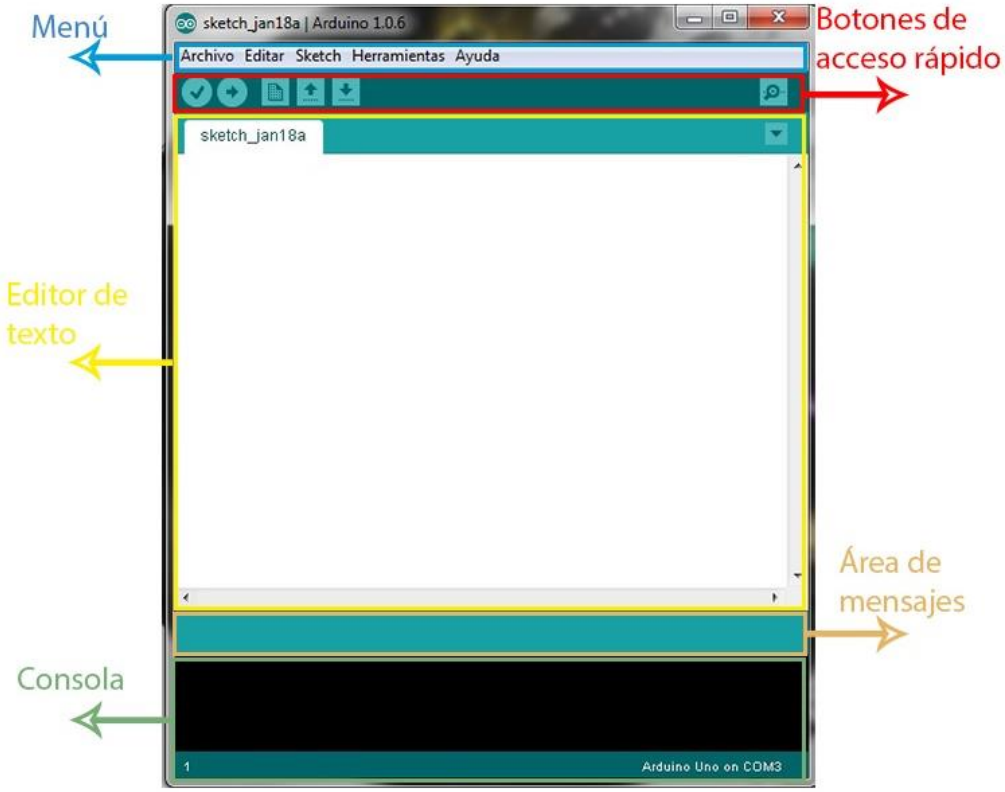


Figura 3.15. Partes del ambiente de desarrollo de Arduino [21].

El lenguaje estándar para programar las tarjetas de desarrollo Arduino es C++, aunque es posible programarlo en otros lenguajes. No es un C++ puro, sino que es una adaptación que proveniente de avr-libc que provee de una librería de C de alta calidad para usar con GCC en los microcontroladores AVR de Atmel y muchas funciones específicas para los MCU AVR de Atmel [22].

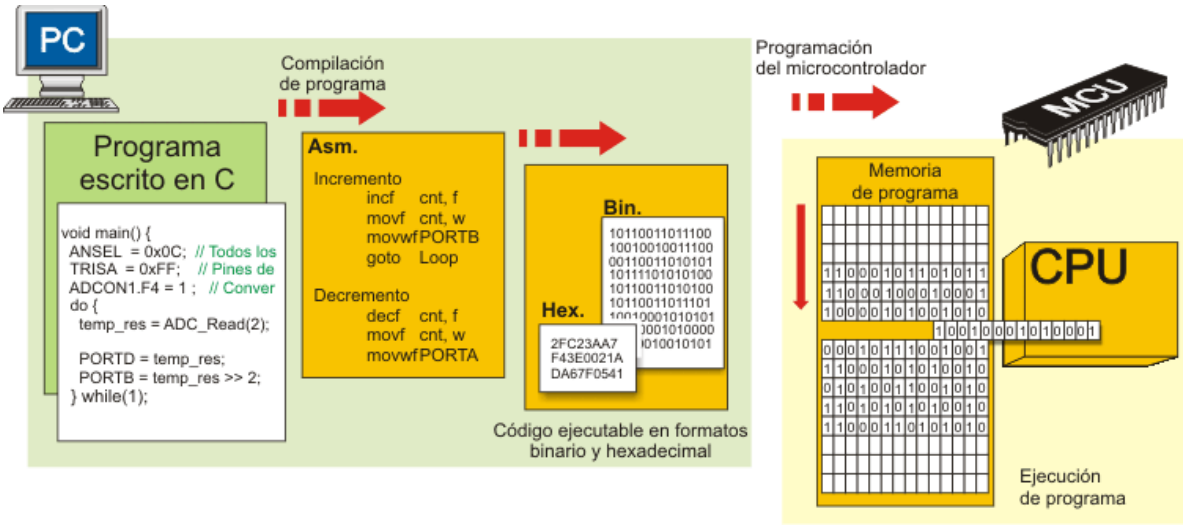


Figura 3.16. Esquema básico de programación del microcontrolador [23].

La figura anterior es un ejemplo general de lo que sucede durante la compilación de programa de un lenguaje de programación de alto nivel a bajo nivel [23].

avr-binutils, avr-gcc y avr-libc son las herramientas necesarias para programar los microcontroladores AVR de Atmel. GCC es un conjunto de compiladores que se considera el estándar para los Sistemas Operativos derivados de UNIX y requiere de un conjunto de aplicaciones conocidas como binutils que son unas herramientas de programación para la manipulación de código de objeto. Cuando GCC está construido para ejecutarse en un sistema como Linux, Windows o mac OS y generar código para un microcontrolador AVR, entonces se denomina avr-gcc. avr-gcc es el compilador que usa el IDE de arduino para convertir el sketch en C++ a un fichero binario (.hex) que es el que se carga en la flash del MCU y que ejecuta [22].

Arduino provee librerías que facilitan la programación del microcontrolador. El gran éxito de Arduino en parte se debe a que permite programar un microcontrolador sin tener que saber todo lo anterior y da herramientas sencillas y específicas para programar los microcontroladores que suministra en sus placas.

El lenguaje de programación usado por Arduino está basado en Processing, el cual, es un lenguaje de programación y entorno de desarrollo integrado de código abierto basado en Java, de fácil utilización, y que sirve como medio para la enseñanza y producción de proyectos multimedia e interactivos de diseño digital [22, 24].

4.Metodología y Desarrollo Experimental

4.1. Diseño de la plantilla

La plantilla para medir la pisada en niños, se diseñó con recomendaciones de médicos especialistas. El diseño contemplo la pisada de niños entre 6 y12 años, es decir en tallas de 13 a 23 cm de largo y un ancho de 9 cm, con la intensión de detectar al menos 12 puntos de presión que los especialistas consideran importantes para el diagnóstico, con lo cual poder determinar los problemas en la pisada. Así, el área total se calculó en un rectángulo de 26 x 11 [cm] en donde se pudieran acomodar 48 sensores en una matriz de 8x6. Además, este diseño puede emplearse tanto para pie izquierdo como pie derecho, aunque para una medición completa tendrán que utilizarse dos plantillas simultáneamente. En este trabajo solo se plantea la elaboración de una plantilla (una matriz de 26 x 11 [cm]) y se espera que el sistema sea suficientemente flexible para, adicionar la segunda plantilla, duplicando el sistema.

4.2. Construcción de la matriz de sensores capacitivos

Para construir el sistema capacitivo se empleó una lámina de PET de 28x14 [cm], en la que se depositaron mediante el método de evaporación térmica, películas de plata con una geometría circular, con diámetro de 1 [cm]. Para hacer la evaporación previamente se elaboró una máscara con la configuración deseada la cual se muestra en la figura 4.1.

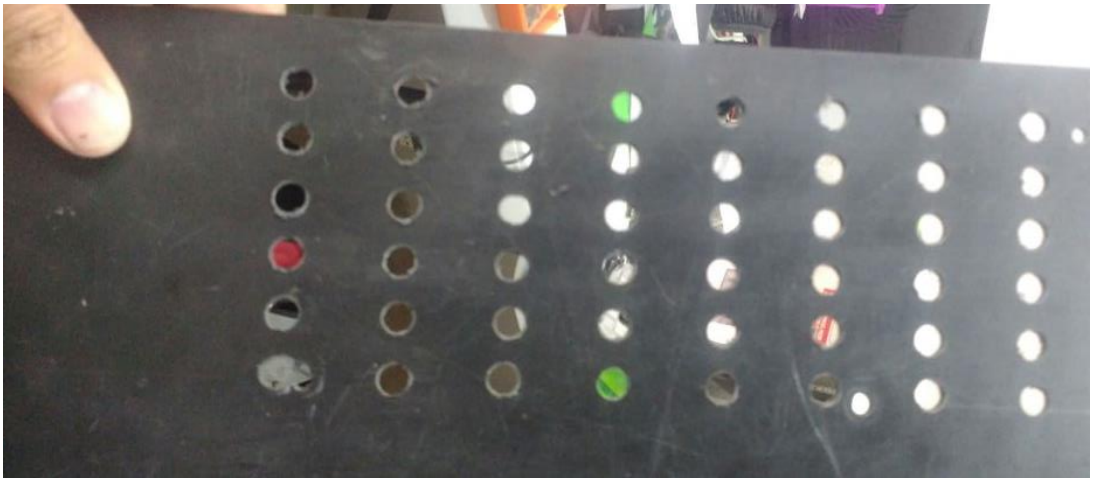


Figura 4.1. Máscara de plástico empleada para evaporar los electrodos de plata.

4.2. Fabricación de Electrodos por Evaporación Térmica

Como se mencionó anteriormente se fabricó la matriz de sensores capacitivos utilizando evaporación térmica, la cual consiste en el calentamiento por efecto Joule de un crisol de tungsteno hasta la evaporación del material que se pretende depositar, en este caso la plata. Para esto se utilizó la evaporadora que se muestra en la figura 4.2.

Figura 4.2. Evaporadora.

El vapor del material se condensa en forma de película delgada sobre la superficie de la máscara de plástico, en los lugares destinados para los electrodos,

Las condiciones de trabajo para la evaporación fueron las siguientes:

Tabla 4.1. Parámetros de configuración para la fabricación de los electrodos de plata mediante evaporación térmica.

Variable	Valor
Presión	$8x10^{-6}[Torr]$
Voltaje	$75[VCA]$
Corriente	$3{,}5[A]$

Con esto se obtuvo que el grosor de la película de plata fue de 600 $[\eta m]$, y finalmente la matriz de sensores capacitivos mostrada en la figura 4.3.

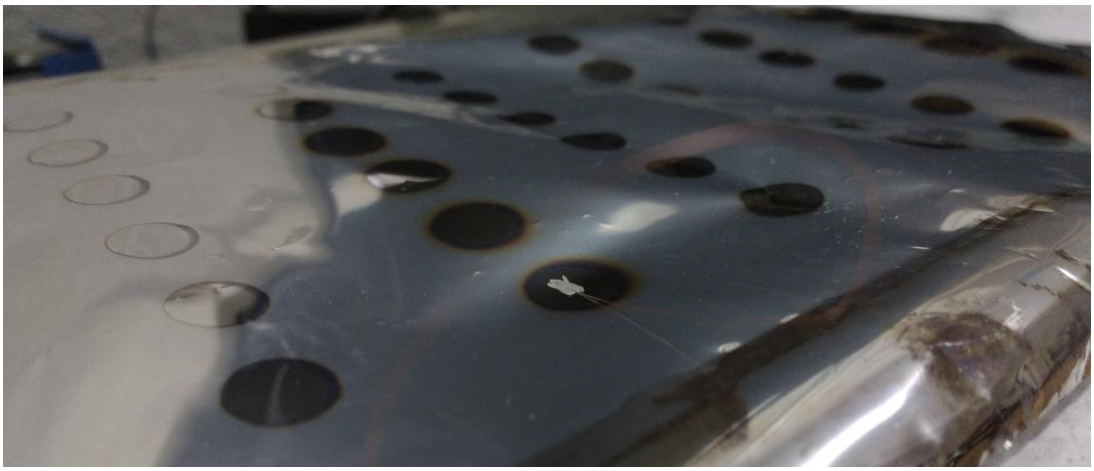


Figura 4.3. Matriz de sensores capacitivos.

Posteriormente se tuvo que utilizar alambre de cobre para llevar las señales de los electrodos al sistema de adquisición de datos, el diámetro del alambre utilizado esta entre los 80 y 100 $[\mu m]$ y el largo de 20 [cm], los cuales se fijaron a los electrodos mediante el uso de pintura de plata y del otro lado del alambre se soldó un pin de cobre.

Se optó por trabajar únicamente con 20 sensores debido a la dificultad y el tiempo que se requería para fijar los cables de conexión a cada uno de los sensores, esto derivado de las dimensiones del alambre de cobre, el cual tiene un recubrimiento que fue necesario remover para cada uno de los extremos de conexión, posteriormente se tuvo que fijar un extremo a uno de los electrodos, utilizando pintura de plata. Por otro lado, se tuvo soldar un pin de cobre el cual era necesario para tener acceso al conector de la tarjeta de adquisición de datos.

El esquemático de conexión entre los sensores y los dispositivos de adquisición de datos se muestra en la figura 4.4.

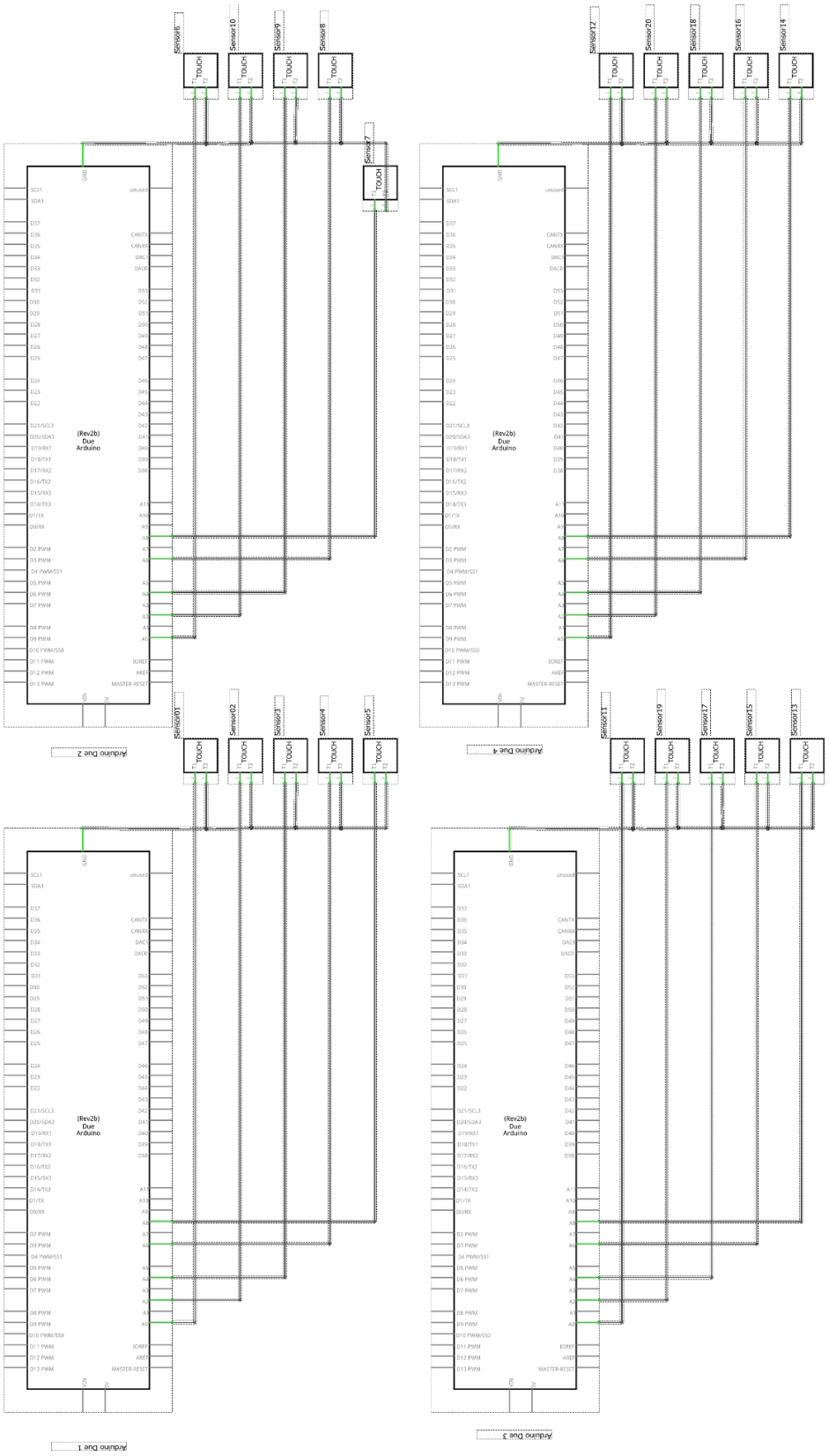


Figura 4.4. Esquemático de conexiones entre los sensores y las tarjetas de adquisición de datos.

El diagrama de conexiones del sistema de adquisición se muestra en la figura 4.5.

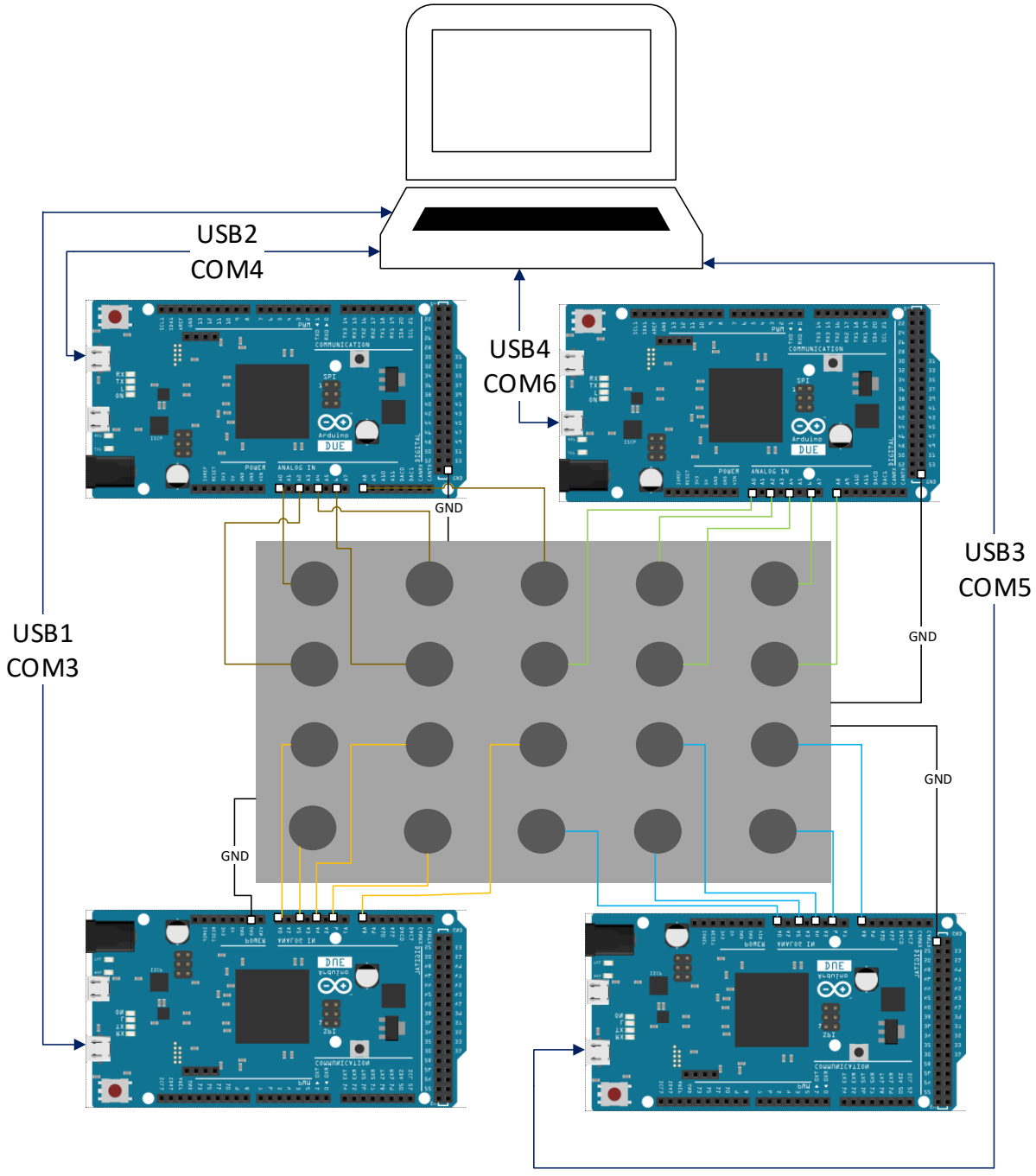


Figura 4.5. Diagrama de conexiones del sistema de instrumentación.

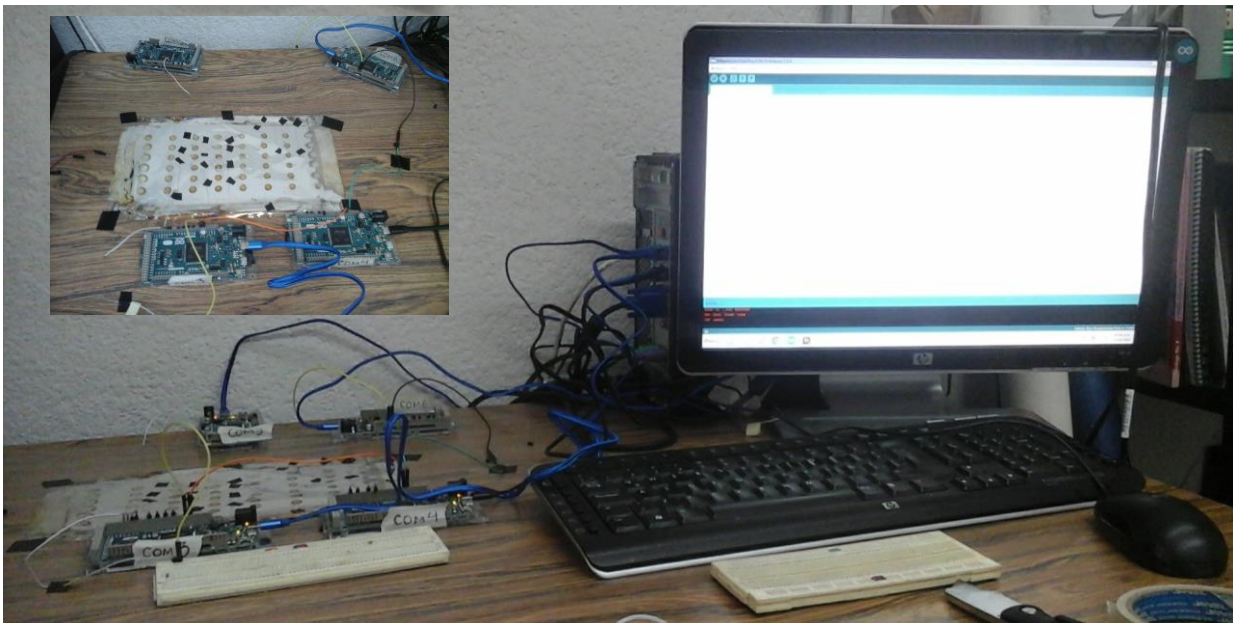


Figura 4.6. Fotografía del prototipo desarrollado.

En la figura 4.6 se puede observar el sistema de adquisición de datos completo, el cual incluye la matriz de sensores capacitivos conectados a las tarjetas de adquisición, que a su vez se encuentran conectadas a la computadora.

4.3. Requerimientos de software

El software desarrollado en este trabajo en multiplataforma, ya que es compatible con diferentes sistemas operativos.

Tabla 4.2. Sistemas operativos compatibles con el software desarrollado.

Sistema Operativo	Versión
Microsoft Windows	7 o superior
Mac OS X	Lion (10.7) o superior
Linux	Debian 9 o superior

En la tabla 4.3 se muestra el software utilizado para el desarrollo del proyecto.

Tabla 4.3. Software de desarrollo utilizado para la implementación de las aplicaciones del proyecto.

Software de Desarrollo	Versión
Arduino IDE	1.8.4
LabVIEW	2016
Visa	16.0

LabVIEW se utilizó para llevar a cabo gran parte de la programación de este proyecto, en particular la interfaz gráfica, en la figura 24 se muestra la ventana principal.

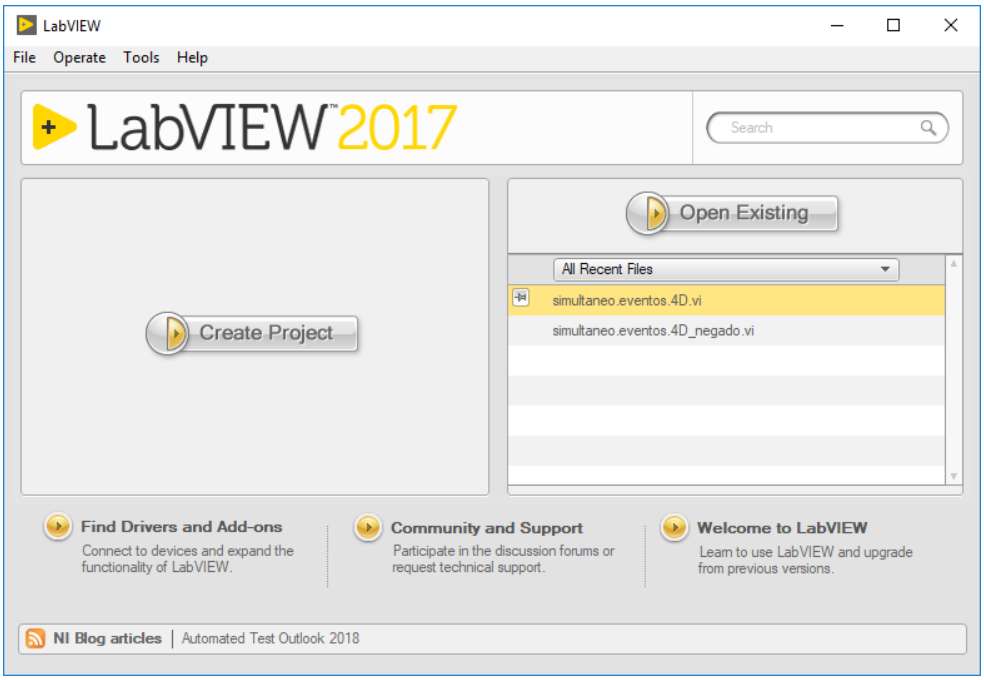


Figura 4.7. LabVIEW 2017.

En conjunto con LabVIEW se utilizó VISA, el cual, es un software de interfaz de E/S Universal (las siglas provienen del inglés: Virtual Instrument Software Architecture), básicamente es un estándar para configurar, programar y depurar sistemas de instrumentación que comprenden interfaces GPIB, VXI, PXI, serial (RS232/RS485), Ethernet/LXI y/o interfaces USB. NI-VISA proporciona la interfaz de programación entre el hardware y los

entornos de desarrollo de aplicaciones como LabVIEW, LabWindows™/CVI y Measurement Studio para Microsoft Visual Studio [26].

Las licencias de desarrollo para NI-VISA están incluidas con los productos de National Instruments como entornos de desarrollo y hardware GPIB. Las licencias para despliegue de NI-VISA están incluidas en sistemas desplegados que tienen hardware de National Instruments, entonos de desarrollo de National Instruments y aplicaciones escritas usando entonos de desarrollo de NI [26].

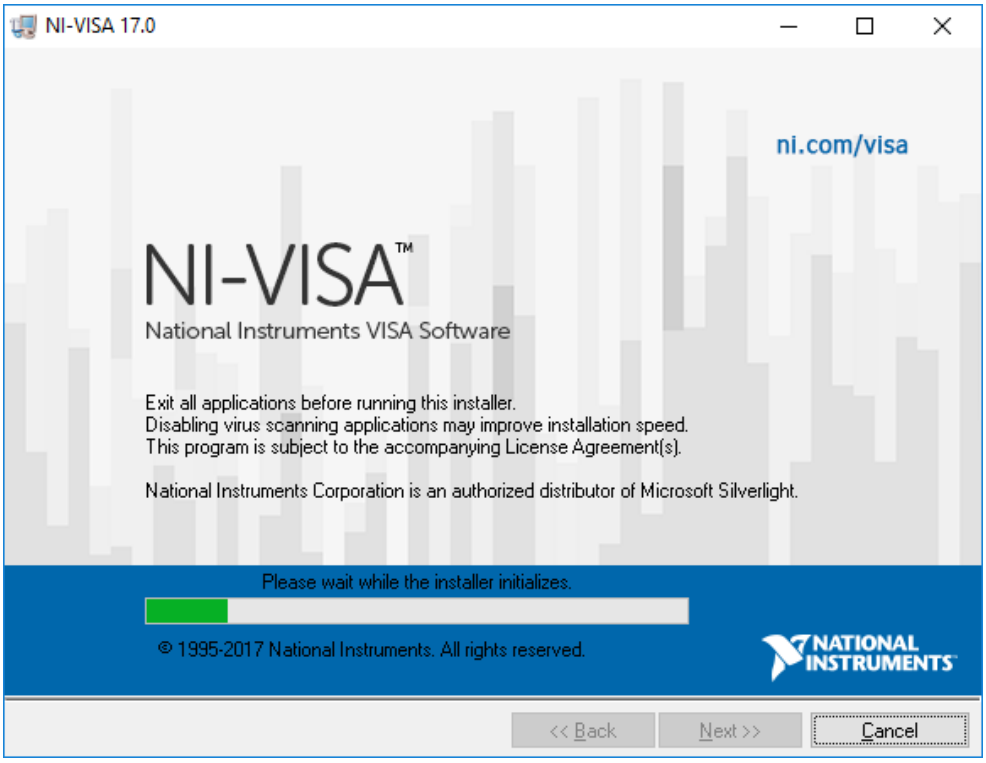


Figura 4.8. Instalación de VISA 17.0.

En resumen, VISA se utilizó en la programación del controlador que gestionó la comunicación entre las tarjetas de desarrollo de Arduino y LabVIEW.

4.4. Desarrollo y programación de la aplicación en Arduino

4.4.1. Configuración y programación del microcontrolador

El algoritmo de programación empleado para la adquisición de los datos provenientes de los sensores capacitivos, así como el envío de dicha información a la computadora para su posterior procesamiento se muestra en la figura 4.9.

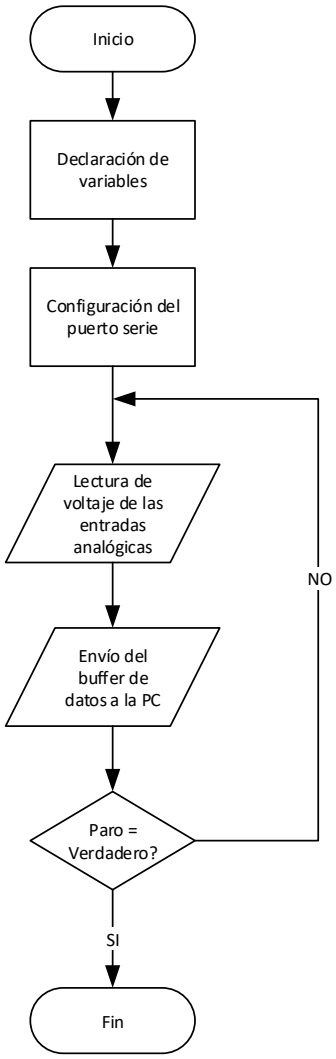


Figura 4.9. Algoritmo de programación de la tarjeta de desarrollo Arduino Due.

La programación e implementación del algoritmo previamente mostrado en la figura 4.7. se explicará más adelante. Primero se explicará cómo se realizó la configuración del IDE de Arduino para poder programar las tarjetas de desarrollo que realizaron la adquisición de las señales provenientes de los sensores capacitivos de presión.

Inicialmente se tuvieron que instalar los controladores correspondientes para el sistema operativo mediante el gestor de tarjetas lo cual permitió el uso de las tarjetas de desarrollo Arduino Due.

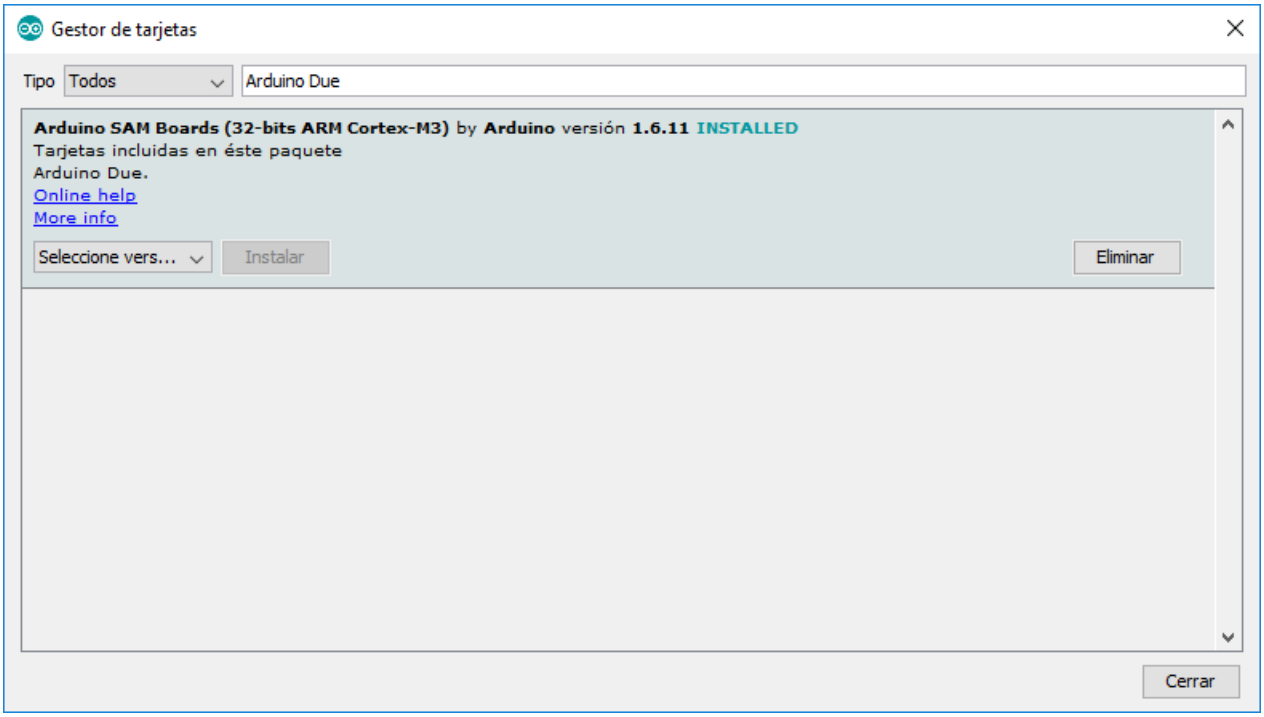


Figura 4.10. Gestor de tarjetas del IDE de Arduino.

Posteriormente se verificó que las cuatro tarjetas estén instaladas y reconocidas de forma correcta por el sistema operativo mediante el administrador de dispositivos de Windows.

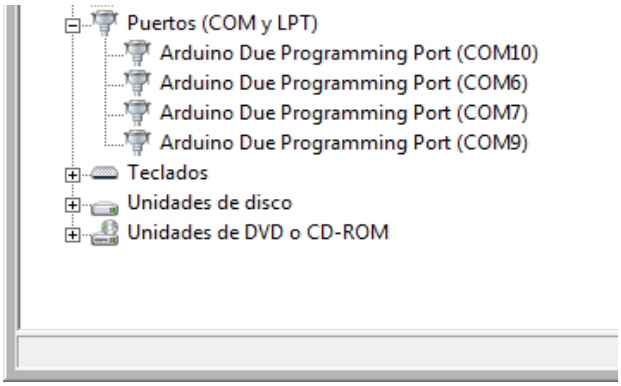


Figura 4.11. Administrador de dispositivos de Windows.

Después de haber verificado la correcta instalación de las tarjetas de adquisición de datos se procedió a realizar la programación de cada una de las tarjetas.

Inicialmente se declararon doce variables numéricas de tipo entero (int), en las tarjetas Arduino Due una variable de este tipo (int) almacena un valor de 32 [b] (4 [B]) [27]. Esto produce un rango de valores que van de -2147483648 a 2147483647 (el valor mínimo es de -2^{31} y el máximo de $2^{31}-1$). Esto con el fin de almacenar temporalmente cada una de las doce señales analógicas de entrada con las que cuenta la tarjeta (ver figura 4.10).

```
int ai0 = 0; //Se declara e inicia de la varable que almacenara el voltaje leido por el pin 0
int ai1 = 0; //Se declara e inicia de la varable que almacenara el voltaje leido por el pin 1
int ai2 = 0; //Se declara e inicia de la varable que almacenara el voltaje leido por el pin 2
int ai3 = 0; //Se declara e inicia de la varable que almacenara el voltaje leido por el pin 3
int ai4 = 0; //Se declara e inicia de la varable que almacenara el voltaje leido por el pin 4
int ai5 = 0; //Se declara e inicia de la varable que almacenara el voltaje leido por el pin 5
int ai6 = 0; //Se declara e inicia de la varable que almacenara el voltaje leido por el pin 6
int ai7 = 0; //Se declara e inicia de la varable que almacenara el voltaje leido por el pin 7
int ai8 = 0; //Se declara e inicia de la varable que almacenara el voltaje leido por el pin 8
int ai9 = 0; //Se declara e inicia de la varable que almacenara el voltaje leido por el pin 9
int ai10 = 0; //Se declara e inicia de la varable que almacenara el voltaje leido por el pin 10
int ai11 = 0; //Se declara e inicia de la varable que almacenara el voltaje leido por el pin 11
```

Figura 4.12. Declaración de las variables de entrada.

Posteriormente, se declaró una variable de tipo carácter (char), en este caso se ocupó una variable de este tipo para enviar los valores de voltaje mediante un buffer de 48 caracteres el cual contenía la información de las doce entradas analógicas de voltaje (cada uno de estos caracteres ocupa 1 byte de memoria) esto con el fin de enviarlo vía serie a la computadora.

```
char buffer[48]; //buffer de datos que contiene las 12 entradas analógicas
                 //este buffer de caracteres se envia por el puerto serie
```

Figura 4.13. Declaración de la variable de salida.

En la siguiente etapa se declararon las funciones (también conocidas como procedimientos o sub-rutinas) las cuales son un fragmento de código con una etiqueta o nombre las cuales pueden ser utilizadas en cualquier parte de la rutina principal (figura 4.12).

```
void setup() {
  Serial.begin(9600);
  analogReadResolution(12);
              }
```

Figura 4.14. Declaración de funciones para la rutina de adquisición de datos.

Donde, la función "*Serial.begin()*", establece la velocidad de datos en bits por segundo (baudio) para la transmisión serie. Para comunicarse con computadoras las velocidades admitidas son comúnmente 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, o 115200 [28]. Sin embargo, se pueden seleccionar otras velocidades. En este caso se utilizó una velocidad de datos de 9600 [b/s].

Se puede agregar un segundo argumento (opcional) el cual configura el tipo de dato, paridad y los bits de paro, si no se agrega dicho argumento, por defecto se toman 8 bits de datos, sin paridad y un bit de paro. El cual es suficiente para el envío del buffer de datos que contiene las 12 señales adquiridas.

La segunda función "*analogReadResolution()*", es una extensión de la interfaz de programación de aplicaciones (abreviada como API del inglés: Application Programming Interface) analógica para el Arduino Due. La cual establece el tamaño (en bits) del valor que regresa la función "*analogRead()*" [28]. El valor por defecto es de 10 [bit] ($2^{10} = 1024$, regresa valores entre 0 - 1023) para garantizar compatibilidad con tarjetas basadas en microcontroladores AVR.

En el presente trabajo, las tarjetas utilizadas (Arduino Due) tienen convertidores analógico-digital (ADC) con la capacidad de trabajar con 12 [bit] de resolución en cada una de las entradas analógicas, debido a esto el valor de voltaje adquirido por cada una de las entradas utilizando la función "*analogRead()*" esta entre 0 y 4095 ($2^{12} = 4096$).

Posteriormente, en la rutina principal se realiza la lectura de cada una de las entradas analógicas (12) y se crea el buffer de datos que se envía por el puerto serie con un tiempo de ciclo de 150 [ms] (ver figura 4.13).

```
void loop() {
ai0=analogRead(0);
ai1=analogRead(1);
ai2=analogRead(2);
ai3=analogRead(3);
ai4=analogRead(4);
ai5=analogRead(5);
ai6=analogRead(6);
ai7=analogRead(7);
ai8=analogRead(8);
ai9=analogRead(9);
ai10=analogRead(10);
ai11=analogRead(11);
sprintf(buffer, "%04i\t%04i\t%04i\t%04i\t%04i\t%04i\t%04i\t%04i\t%04i\t%04i\t%04i\t%04i\t\n", ai0,ai1,ai2,ai3,ai4,ai5,ai6,ai7,ai8,ai9,ai10,ai11);
Serial.print(buffer); //Envio de lo datos por el puerto serie
delay(150); //Tiempo de ciclo en milisegundos
          }
```

Figura 4.15. Rutina principal adquisición de datos.

Donde, la función "analogRead()" lee el valor del pin analógico especificado (en total 12 por cada dispositivo). Como se mencionó anteriormente, la resolución del CAD es de 12 [bit]. Esto significa que mapeara voltajes entre 0 y 3.3 [V] a valores entero entre 0 y 4095. Esto produce una resolución entre unidad de lectura de:

$$ResL = \frac{V_{in}max}{2^{Res_{CAD}-1}} = \frac{3.3[V]}{4095} = 0{,}000806[V] \qquad (4.1)$$

Donde:

$ResL$ es la resolución de lectura para cada una de las entradas analógicas en [V].

$V_{in}max$ es el voltaje máximo que puede adquirir nuestro dispositivo en [V].

Res_{CAD} es la resolución de convertidor analógico digital en [bit].

4.5. Programación de la interfaz gráfica

El desarrollo de la aplicación principal consta de cuatro partes: Inicialización, Comunicación, Procesamiento y Presentación (Despliegue).

4.5.1. Inicialización

La aplicación principal comienza con inicialización de los puertos de comunicación con las tarjetas de adquisición, así como de todas las variables del sistema de procesamiento y despliegue e interpretación de los datos. Inicialmente, se abren simultáneamente cada uno de los cuatro puertos de comunicación entre la computadora y las tarjetas de desarrollo, posteriormente se hace una lectura de los datos presentes en el puerto, e inmediatamente se cierra el puerto para garantizar que al iniciar la rutina principal de adquisición no haya información presente en el puerto con lo cual se garantiza la sincronización entre las tarjetas de adquisición y la computadora.

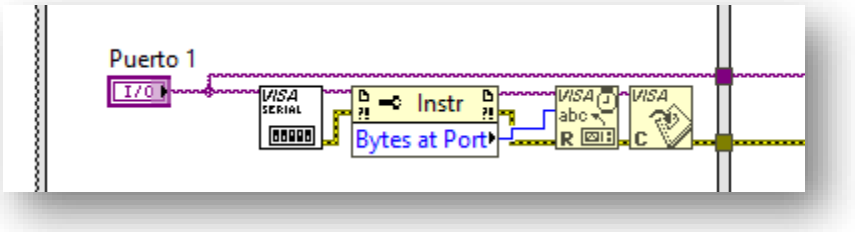


Figura 4.16. Inicialización del puerto de comunicación.

Después de realizar la inicialización de las variables del programa comienza la rutina principal, la cual consta de cuatro estructuras de repetición (while loop) temporizadas las cuales se encargan de realizar tres tareas: comunicación, procesamiento y presentación de los datos. Estas tres operaciones se realizan en sincronía, cada una de ellas tiene diferente prioridad y diferente tiempo de ciclo.

Tabla 4.4. Prioridades y tiempo de ciclo.

Nombre de Proceso (Tarea)	Prioridad	Tiempo de Ciclo [ms]
Comunicación	500	250
Procesamiento	300	500
Presentación	100	500

4.5.2. Sincronización

Para realizar la sincronización, prioridad y el tiempo de ciclo de las estructuras se utilizó la función "*Synchronize Timed Structure Starts.vi*" en conjunto con las estructuras temporizadas.

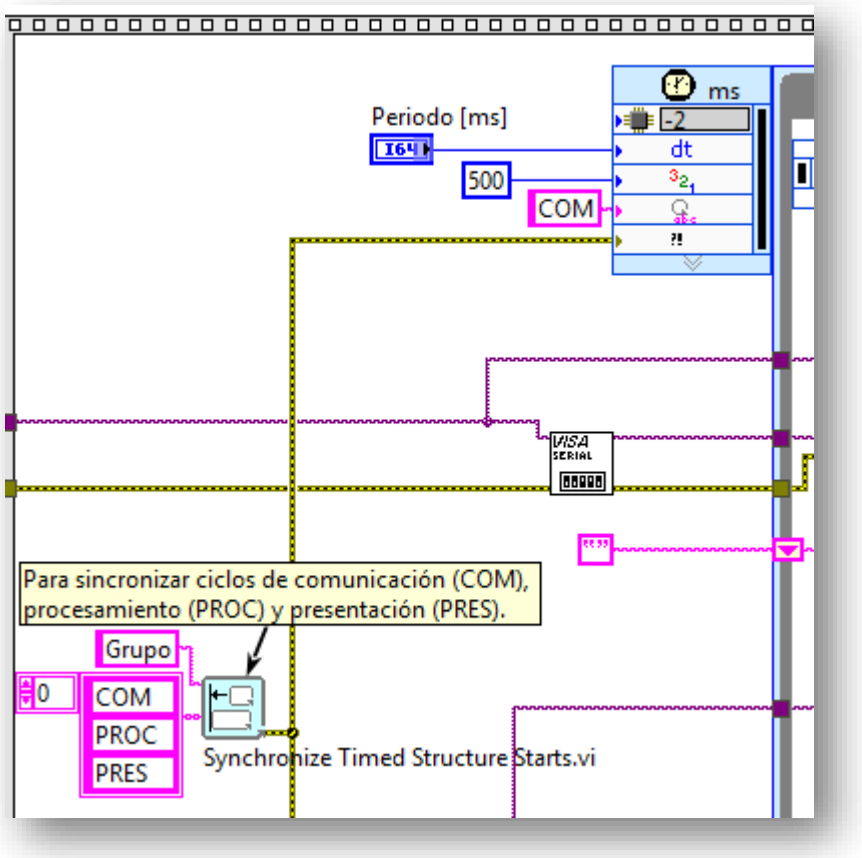


Figura 4.17. Sincronización de las estructuras de la rutina principal.

Como se puede observar en la tabla, la prioridad más alta (500) la tiene la estructura de comunicación, después la estructura de procesamiento (300) y la más baja la estructura de presentación de datos (100). Esto se programó de esta forma, debido a que se requiere garantizar que siempre haya datos disponibles, ya que, sin esto la etapa de procesamiento no tendría datos que procesar. De igual forma si no hay datos en la etapa de procesamiento, no se podría realizar el despliegue o presentación de la información.

Esto está relacionado, de igual forma, con el tiempo de ciclo de cada una de las estructuras. El menor tiempo de ciclo (200 [ms]) lo tiene la estructura de comunicación, el cual está en sincronía con el tiempo de ciclo de la tarjeta de adquisición de datos. Esto para garantizar

que el buffer de datos provenientes de la tarjeta se lea y descargue en tiempo para evitar que dicho buffer llegue incompleto o que se sature.

Tabla 4.5. Relación de sincronía entre la computadora y las tarjetas de adquisición.

Relación de Sincronía	Estado del Buffer	Resultado
$\boldsymbol{\tau_{AD} < \tau_C}$	Incompleto	No hay resultados que presentar ya que los datos llegan incompletos.
$\boldsymbol{\tau_{AD} > \tau_C}$	Saturado	Retraso incremental en la presentación de los datos, se presentan datos fuera de tiempo.
$\boldsymbol{\tau_{AD} = \tau_C}$	Completo	El buffer se lee y despliega en tiempo y forma, no hay perdida de información.

Donde,

τ_{AD} – Tiempo de ciclo en la tarjeta de desarrollo [ms].

τ_C – Tiempo de ciclo de la rutina de comunicación entre la computadora y la tarjeta de desarrollo [ms].

τ_P - Tiempo de ciclo de la rutina de procesamiento de datos [ms].

τ_D – Tiempo de ciclo de la rutina de despliegue (presentación) de los datos [ms].

En todos los casos se debe de cumplir $\tau_C < \tau_P < \tau_D$ para garantizar que haya datos que desplegar.

4.5.3. Comunicación

En la etapa de comunicación LabVIEW debe tener acceso a cada uno de los puertos seriales, para esto se debe iniciar una sesión VISA. Para esto se utiliza con la función "*VISA Configure Serial Port VI*" para cada puerto serie utilizado.

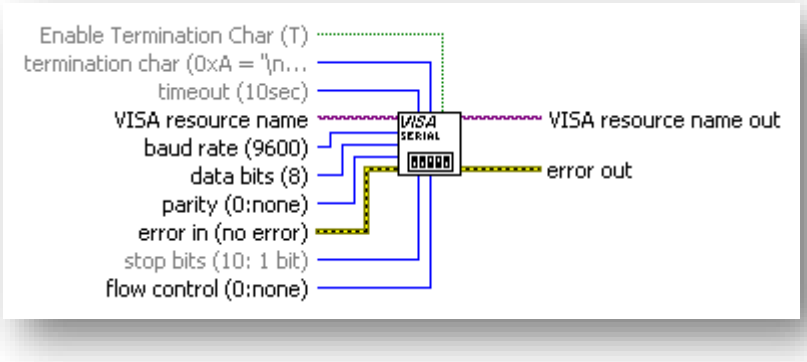


Figura 4.18. Configuración del puerto serie mediante VISA en LabVIEW.

Se debe iniciar la comunicación serie en el puerto especificado usando la terminal "***VISA resource name***", así mismo, se establece la velocidad de transmisión serial (tasa de baudios o baudaje) el cuál es el número de unidades de señal por segundo [29]. Un baudio en este caso contiene 8 [b]. La tasa de baudios seleccionada fue de 9600 [baud] lo que equivale a transmitir 9600 señales o caracteres de 8 [b] por segundo.

Otro parámetro que se configuró para hacer la correcta lectura del buffer de datos fue el "*termination char*" (carácter de terminación) el cual se estableció (en la instrucción que envía los datos desde la tarjeta de adquisición a la computadora) como una nueva línea (linefeed) para especificar en donde termina la cadena de caracteres que se están enviando/recibiendo, la operación de lectura finaliza cuando la computadora lee el carácter de terminación desde el dispositivo serial. El equivalente en hexadecimal del carácter de terminación linefeed (\n) utilizado es 0xA y tiene un peso de 2 [B] [30].

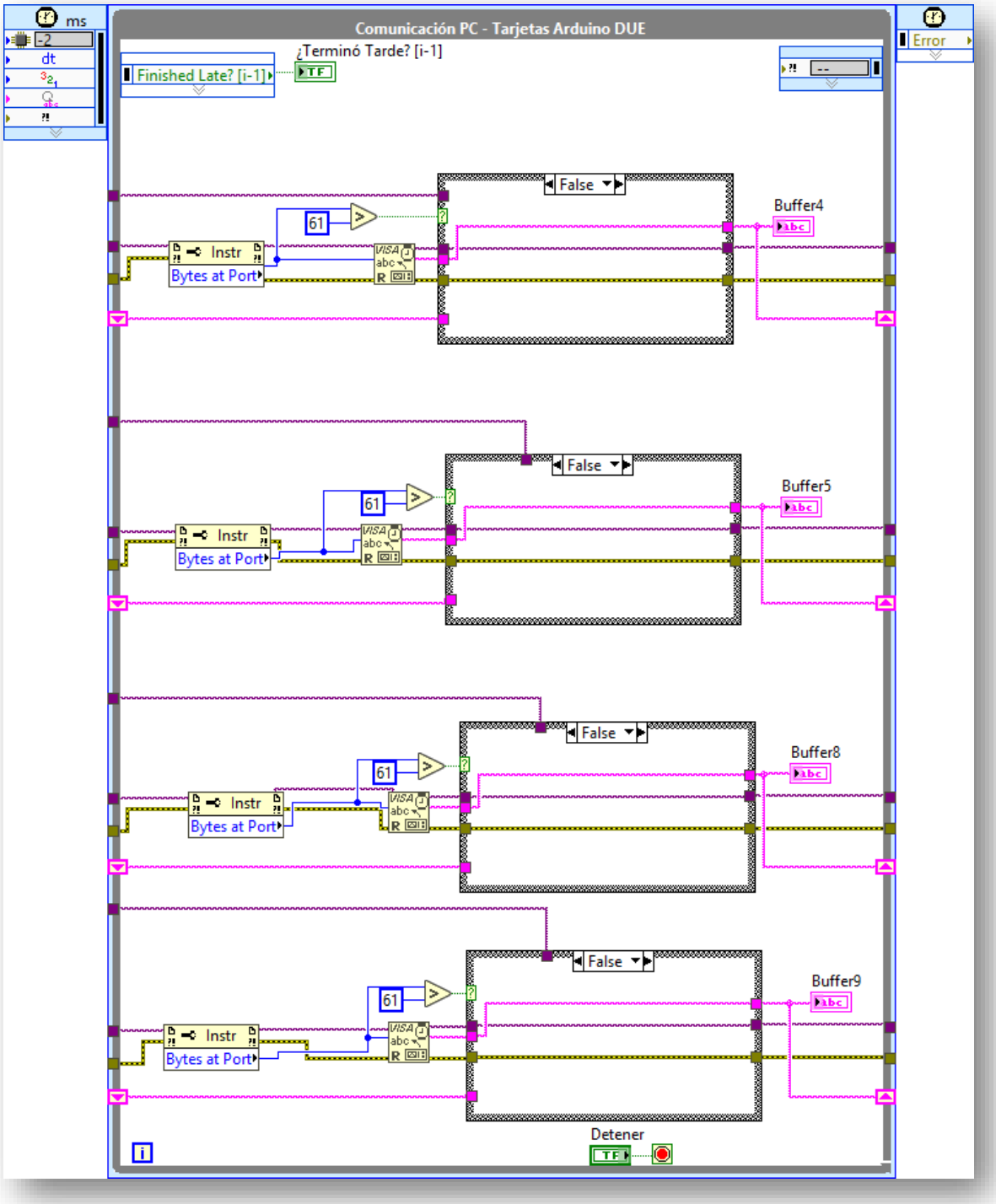


Figura 4.19. Ciclo de comunicación entre la computadora y las tarjetas de desarrollo.

Dentro del ciclo de comunicación se cuenta la cantidad de bits que hay presente en ese momento en cada uno de los cuatro puertos.

$$n_B = 4[B]n_{EA} + (n_{EA}[B] - 1[B]) + nB_{tc}[B] \quad (4.2)$$

Donde,

n_B – Es el número de bytes transmitidos por el puerto serie.

n_{EA} – Es el número de estradas analógicas del dispositivo, el cual es de 12.

nB_{tc} – Es el número de bytes que ocupa el carácter de terminación utilizado, el cual para una nueva línea es 2 [B].

Por lo tanto,

$$n_B = 4 * 12 + (12 - 1) + 2 = 61[B]$$

Entonces, se espera que lleguen 61 [B] por cada puerto.

Si el buffer es mayor a 61 [B] se limpia, si es menor se deja pasar el último buffer completo que se recibió, y si es igual se lee el nuevo buffer.

4.5.4. Procesamiento y despliegue de datos

Para poder procesar los datos fue necesario llevar los 4 buffers al cilclo de procesamiento. La forma de llevar estos datos a otro ciclo corriendo en paralelo fue mediante el uso de variables locales.

Figura 4.20. Variable local

Cabe mencionar que, hasta esta etapa del programa, los datos provenientes de cada uno de los sensores no se encuentran en un formato numérico ni tampoco en valores de voltaje nominal. Como se mencionó anteriormente los valores adquiridos hasta el momento varían en un rango de 0 a 4096 y se encuentran en formato ASCII en una sola cadena de caracteres.

Por lo tanto, para hacer la conversión se realizó lo siguiente:

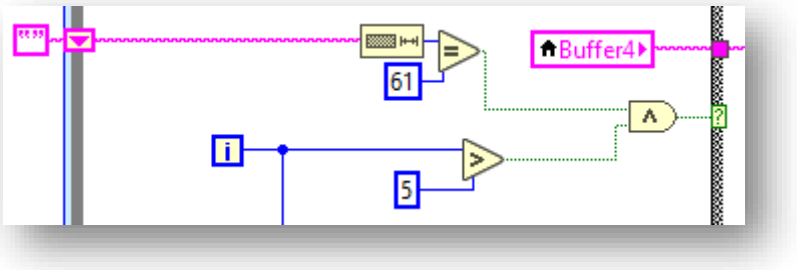


Figura 4.21. Rutina para asegurar que se procese el buffer esperado.

Inicialmente verificamos que el tamaño de la cadena que contiene el buffer de datos sea de 61 caracteres, como se mencionó anteriormente, se espera un buffer de 61 [B] o 61 caracteres, ya que cada carácter ocupa 1[B] de información. Simultáneamente, esperamos 5 iteraciones de 500 [ms] cada una para empezar a procesar los datos, con esto damos tiempo a la rutina de comunicación que este en sincronía con las tarjetas de adquisición y se garantiza que no haya perdida de información.

Ya que se tiene el buffer completo, se separan cada una de las entradas, aún en formato ASCII, para esto se debe tomar en cuenta cual es el formato de origen. Como se mencionó anteriormente, los datos vienen en el siguiente formato:

$$\%d\backslash t\%d\backslash t\%d\backslash t\%d\backslash t\%d\backslash t\%d\backslash t\%d\backslash t\%d\backslash t\%d\backslash t\%d\backslash t\%d\backslash n \qquad (4.3)$$

Donde,

$\%d$ – representa una variable o número entero (int, long o short).

$\backslash t$ – Representa una constante de tabulación.

$\backslash n$ - Representa una constante de salto de línea.

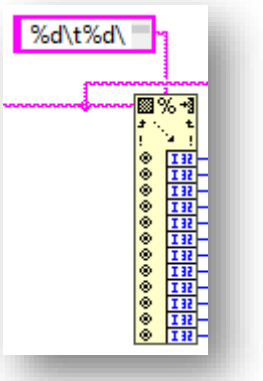


Figura 4.22. Función de conversión y separación del buffer.

Como se muestra en la figura se utiliza el formato antes mencionado en conjunto con la función "*Scan From String VI*" para separar y convertir los datos a un formato numérico entero.

Para realizar la conversión a valores de voltaje nominal se requiere el uso de la siguiente ecuación:

$$V_n = \frac{V_{EAD} V_{Dmax}}{2^{RCAD}} \qquad (4.4)$$

Donde,

V_n – Voltaje Nominal

V_{EAD} – Voltaje presente en la entrada analógica del dispositivo de adquisición no escalado.

V_{Dmax} – Voltaje máximo que puede medir el dispositivo de adquisición.

$RCAD$ – Resolución del convertidor analógico digital del dispositivo de adquisición.

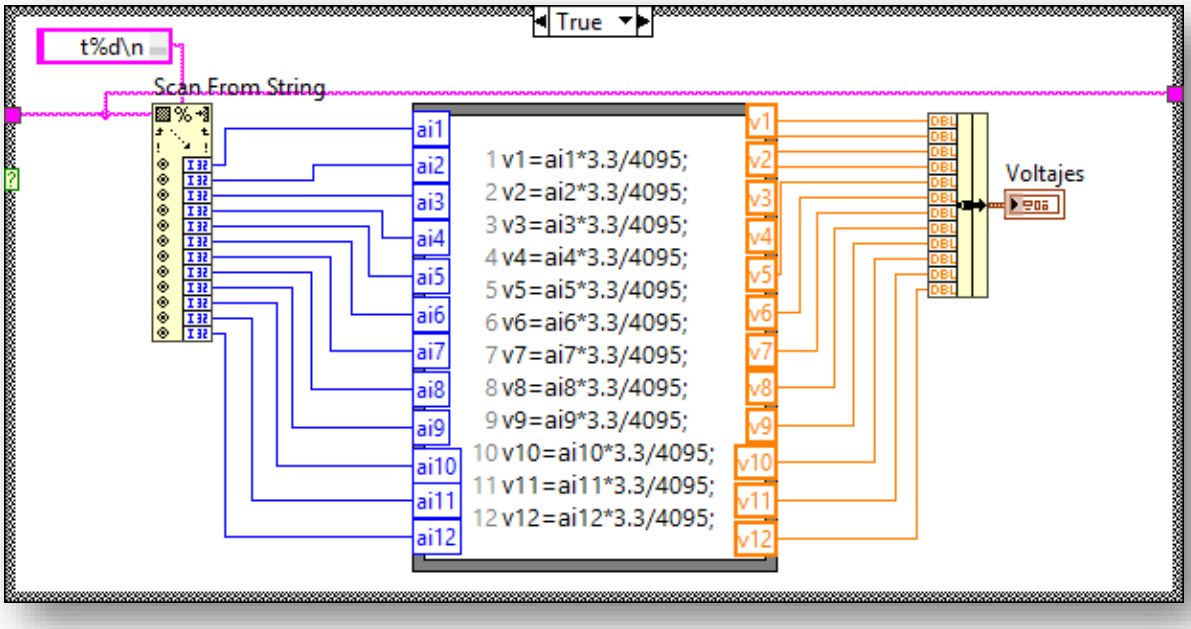


Figura 4.23. Rutina de conversión a voltajes nominales.

En la figura se muestra la implementación de la ecuación anterior en el programa, la cual se aplica para todas las entradas analógicas de los cuatro dispositivos de adquisición de datos. Con esto se tiene el voltaje nominal en [V] el cual varía en un rango de 0 a 3.3[V].

4.5.5. Interfaz con el usuario

En esta parte se asignó un color para indicar la variación del voltaje en cada uno de los sensores. Tomando el rango de voltaje medible con las tarjetas de adquisición, se realizó una asignación de color en formato RGB [31] (del inglés: Red, Green, Blue, en español, Rojo, Verde, Azul) utilizando únicamente el color rojo donde 0 [V] corresponde al color negro (RGB: 0,0,0) y 3.3 [V] al color rojo (RGB: 255,0,0) [32]. Se realizó un sub programa para realizar dicha conversión para cada dispositivo, el código se muestra en la figura 4.21.

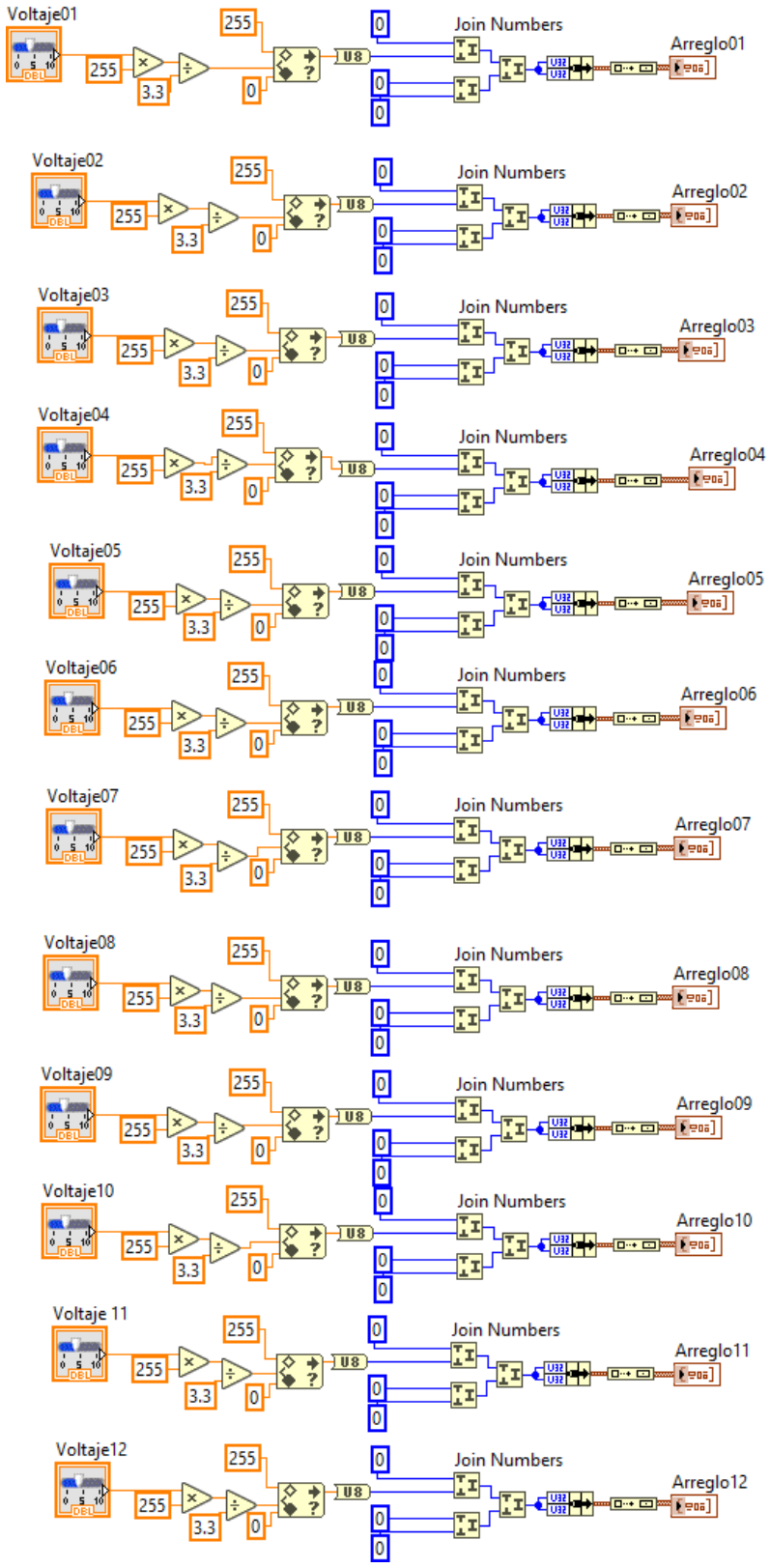


Figura 4.24. Rutina de conversión de voltaje nominal a color en formato RGB.

El subVI (sub programa) tiene la siguiente representación en el diagrama a bloques.

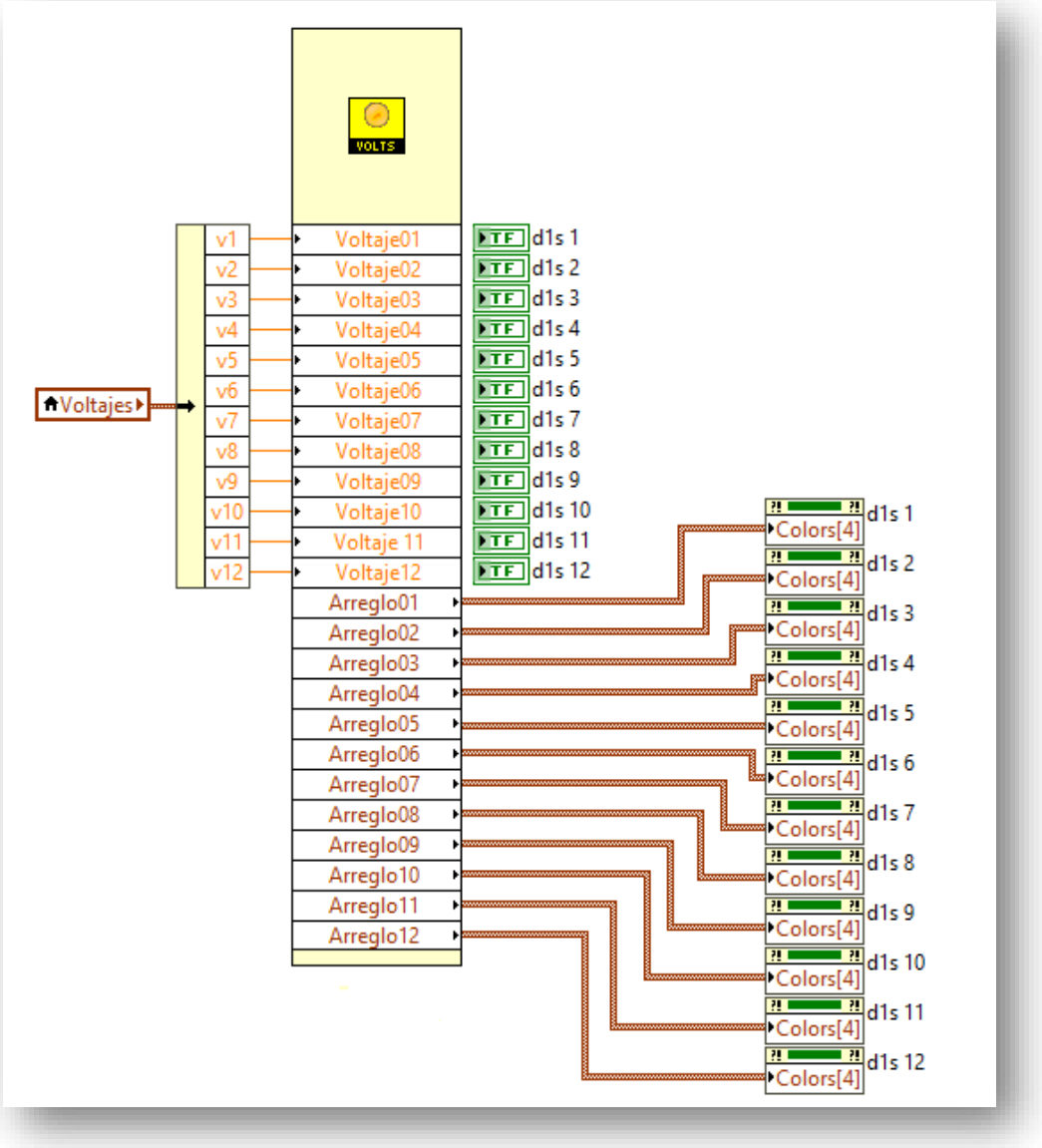


Figura 4.25. SubVI de conversión de voltaje a color.

En conjunto con los "*property nodes*" (nodos de propiedad) asociados a cada uno de las variables booleanas que representan los sensores, se realizó la programación del cambio de color dependiendo del voltaje adquirido proveniente de los sensores capacitivos.

A continuación, se muestra la interfaz gráfica utilizada para experimentación y obtención de resultados utilizada por el desarrollador en la figura 4.23.

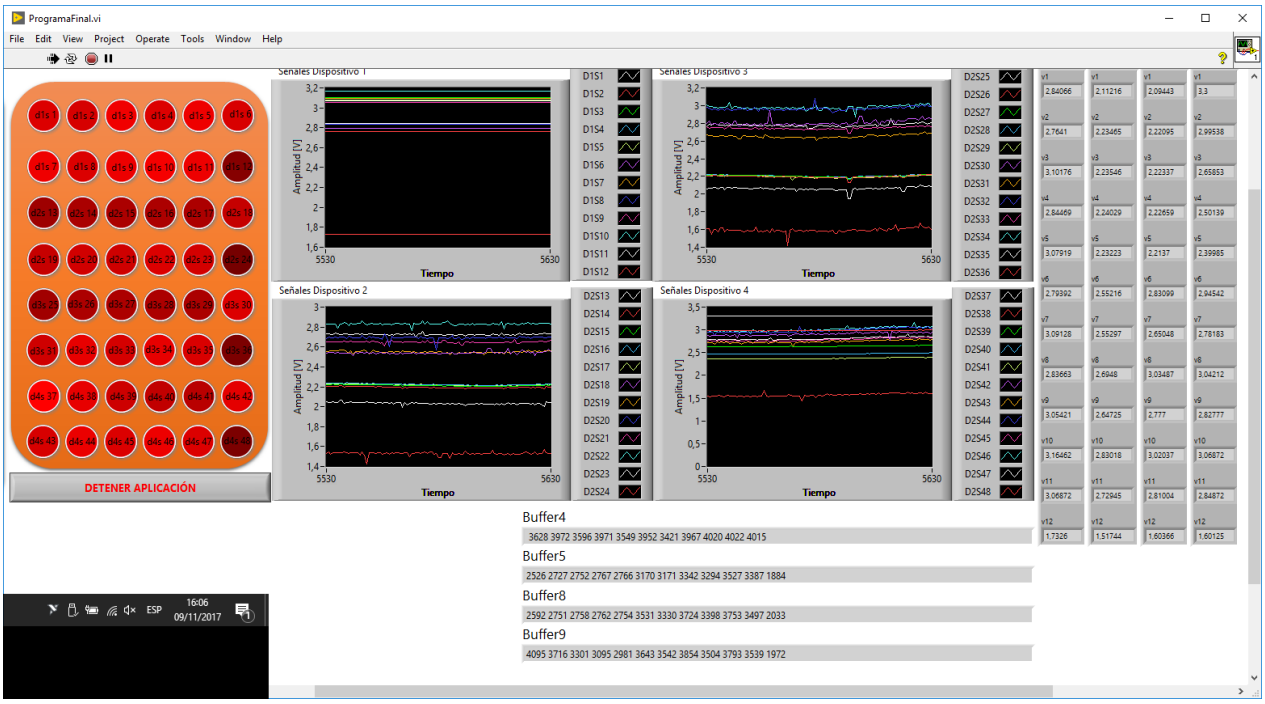


Figura 4.26. Interfaz utilizada para la obtención de resultados finales y experimentación.

Finalmente se muestra la interfaz con el usuario final, dicha interfaz está diseñada para que un usuario sin conocimientos de programación en este caso un médico pueda interpretar la diferencia de presión en base al cambio de color en cada uno de los círculos que representan los sensores que sirven como plantilla para detectar problemas de obesidad en niños.

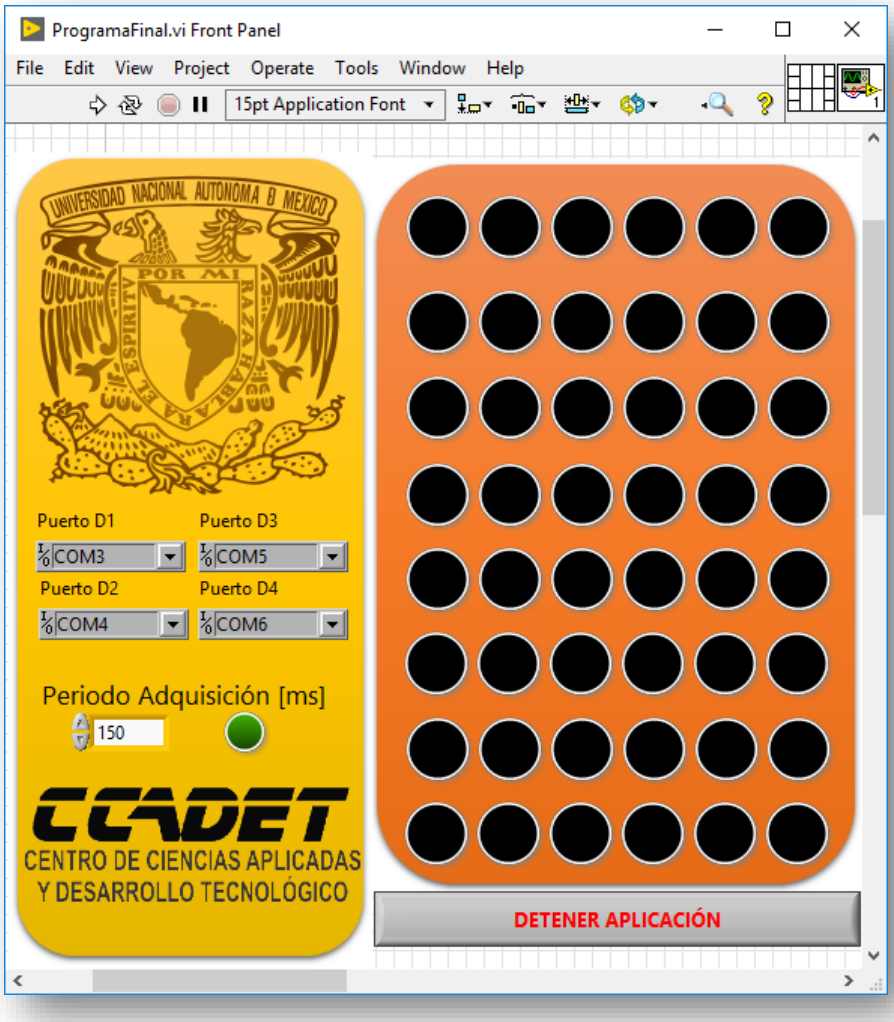


Figura 4.27. Interfaz con el usuario final.

La idea fundamental es que en base al cambio de presión en cada uno de los sensores el programa haga una conversión de voltaje a un color que varía en diferentes tonalidades de rojo, siendo el rojo 255.0.0 en RGB el nivel máximo de voltaje equivalente a 3.3 [V], y el valor mínimo de presión que corresponde a un valor RGB de 0.0.0 que es el color negro.

5.Pruebas experimentales y resultados

En esta parte se presentan los resultados experimentales de los sensores fabricados en el laboratorio. Para llevar a cabo los experimentos se trabajó únicamente con una matriz de 5x4 sensores y un arreglo de indicadores gráficos, donde cada uno de ellos se encuentra asociado a una tarjeta de adquisición, por lo tanto, es capaz de desplegar el voltaje de 12 sensores por cada indicador. La cual se muestra en la figura 5.1.

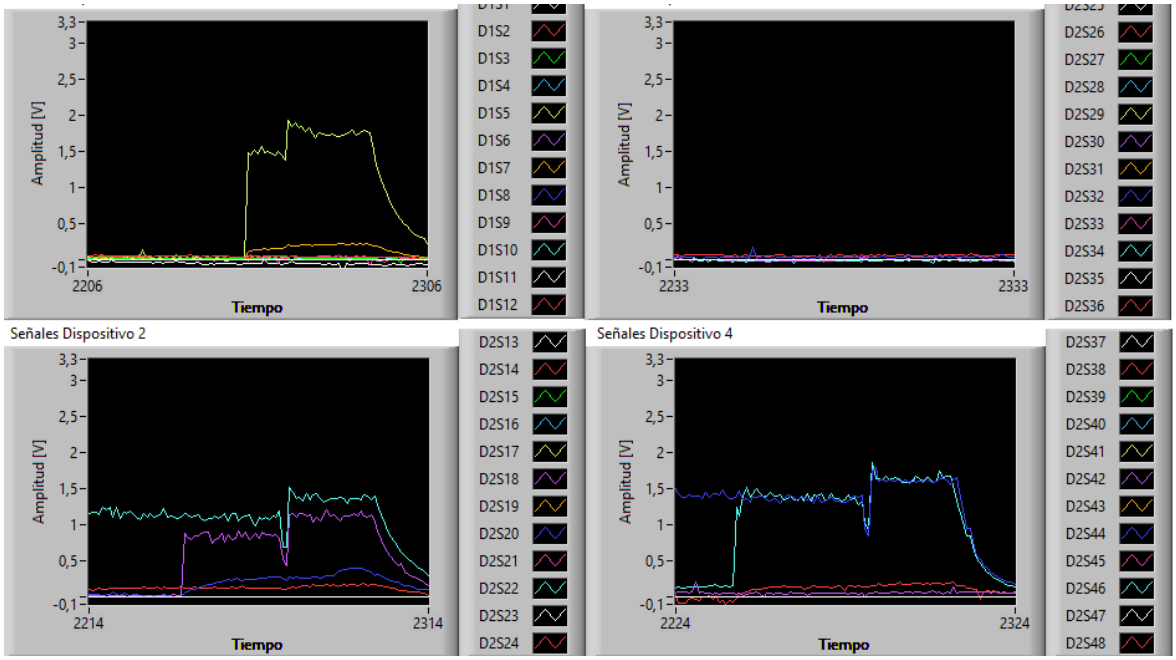


Figura 5.1. Gráficas de voltaje contra tiempo en LabVIEW de los cuatro dispositivos de adquisición de datos con las 48 entradas de voltaje adquiriendo de forma simultanea.

Para esta parte se seleccionó un sensor al azar en cada dispositivo y se ejerció presión con el dedo de la mano sobre dichos sensores durante cierto tiempo cada prueba tuvo una duración entre 40 y 60 [s] aproximadamente. Mediante una rutina en LabVIEW se midió y almacenó en la computadora la amplitud de voltaje medida en cada uno de estos cuatro sensores, con esto se generó un archivo de texto con los datos adquiridos en cada sensor. Se tomó una muestra cada 100 [ms].

Posteriormente se utilizó Python para reproducir y visualizar los datos obtenidos en una gráfica de una forma similar a Matlab [34].

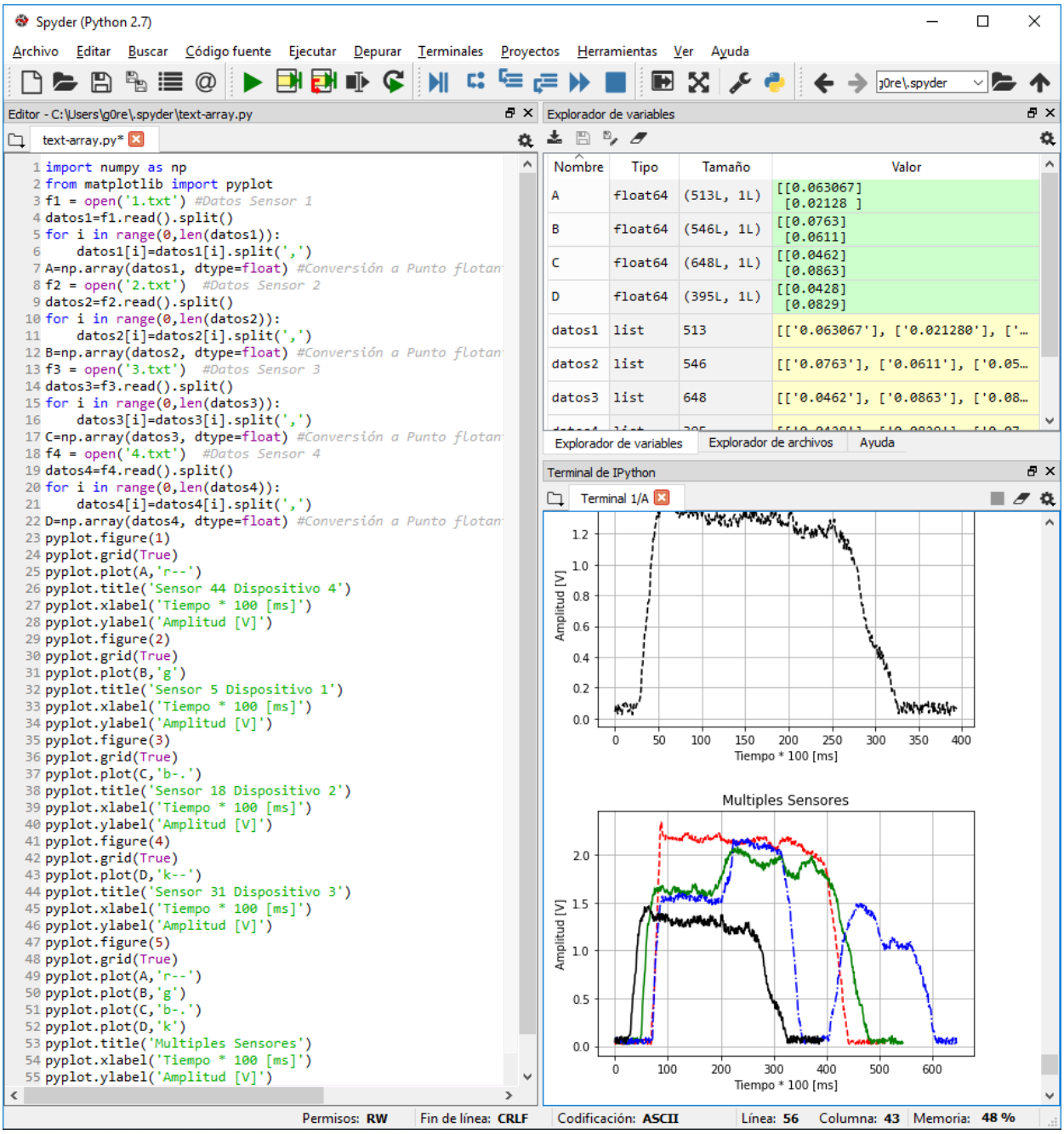


Figura 5.2. Código en Python (Spyder [35]) utilizado para graficar las señales eléctricas de voltaje provenientes de los sensores.

A continuación, se muestran las señales eléctricas en respuesta a la excitación generada por la presión ejercida sobre los sensores. En la figura se muestra la respuesta en voltaje contra tiempo.

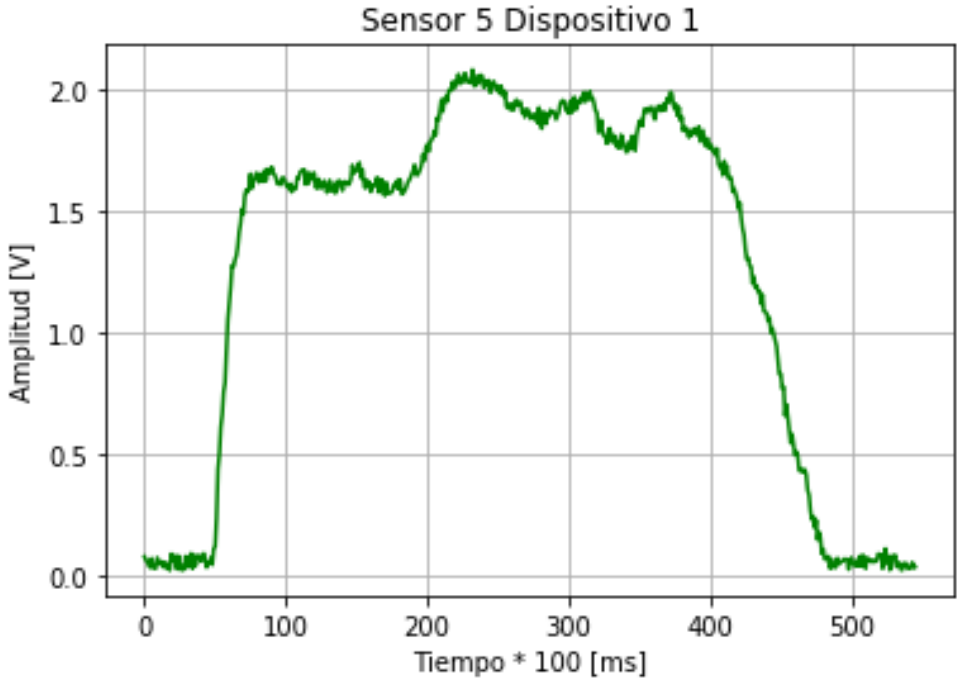


Figura 5.3. Datos adquiridos para uno de los sensores conectado al dispositivo número 1.

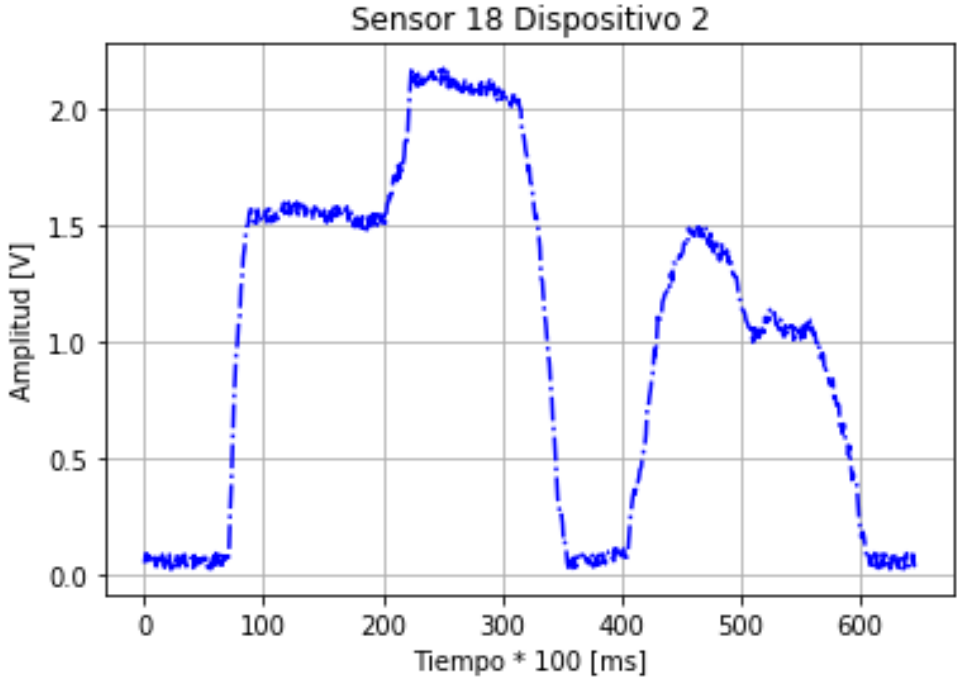


Figura 5.4. Datos adquiridos para uno de los sensores conectado al dispositivo número 2.

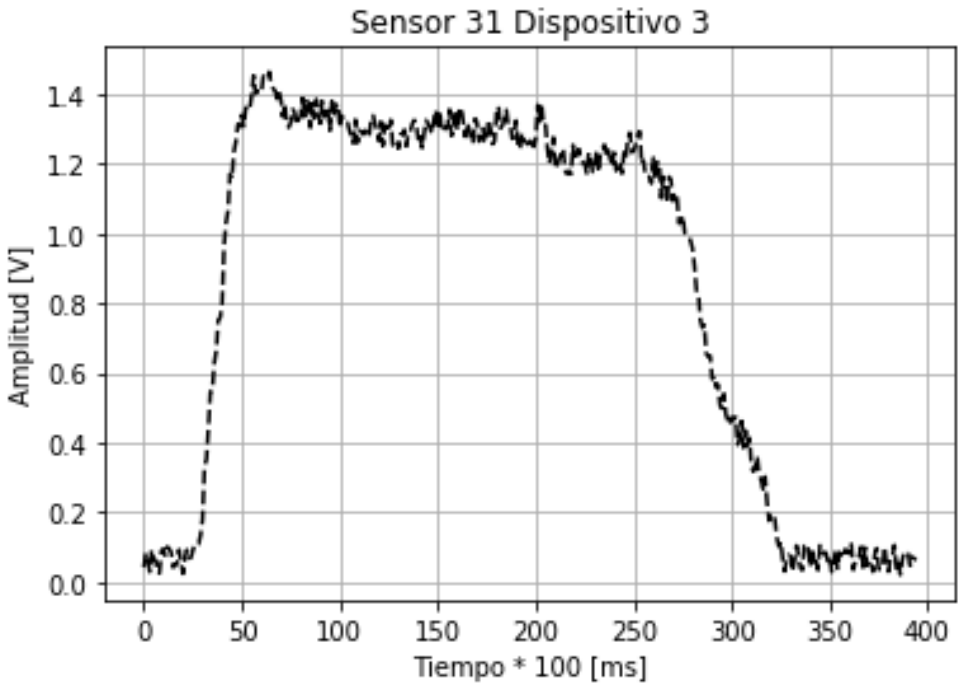


Figura 5.5. Datos adquiridos para uno de los sensores conectado al dispositivo número 3.

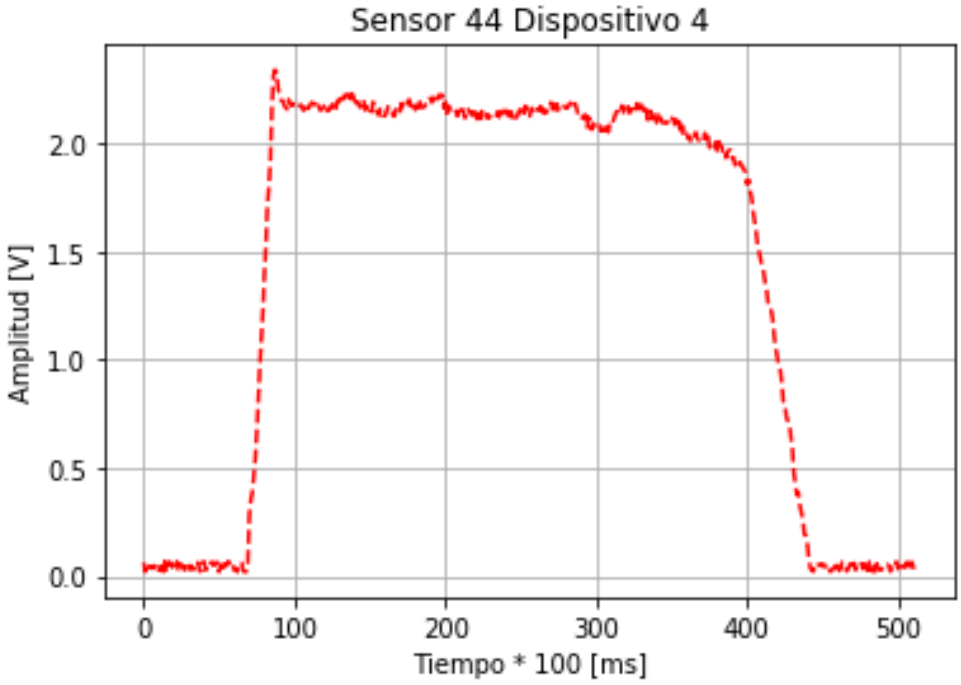


Figura 5.6. Datos adquiridos para uno de los sensores conectado al dispositivo número 4.

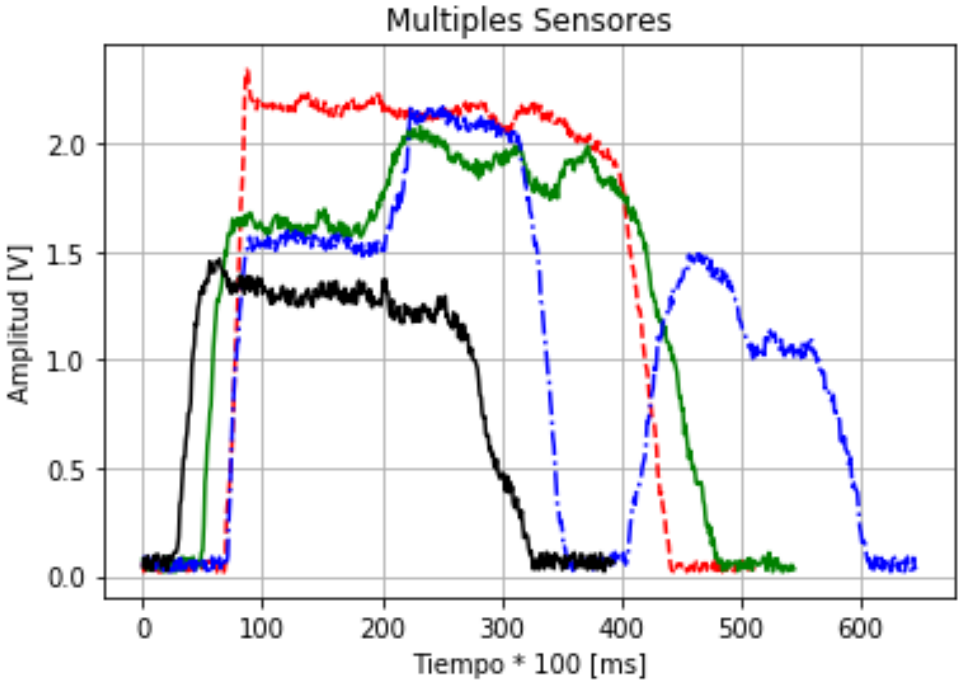


Figura 5.7. Comparación entre las señales adquiridas con 4 diferentes sensores.

En todas las gráficas se muestra el voltaje generado en cada sensor contra el tiempo donde cada señal se desplaza en el tiempo en intervalos de 0.1 [s].

Se puede observar de la figura 5.7 que las amplitudes máximas en la respuesta de los sensores difieren en un 2% aproximadamente, a diferencia del sensor conectado al dispositivo número 3 (negro), esto debido a que la fuerza con la que se excitó al sensor fue menor a comparación de los otros experimentos, ya que la mayoría de los sensores en las diferentes pruebas realizadas tuvieron una respuesta casi idéntica.

Otro aspecto importante es el ruido presente en cada una de las señales, esto puede deberse a la forma de conectar los sensores a cada entrada de los dispositivos, ya que se tuvo que utilizar pintura de plata para unir los electrodos a los alambres de cobre de 0.1 [mm] de diámetro para conectarlos a los dispositivos de adquisición. En el otro extremo del alambre se tuvo que soldar un pin de cobre de 1 [mm] de diámetro para insertarlo en la terminal de las tarjetas.

Hay que enfatizar que este trabajo consistió en desarrollar un prototipo de una matriz de sensores capacitivos, debido a esto falta trabajo a futuro por desarrollar como lo sería un circuito de acoplamiento o filtrado para eliminar el ruido presente en los sensores y mayor longitud del bus de conexión.

De igual forma, como trabajo futuro se puede diseñar e implementar un bus de datos más robusto el cual tenga conectores flexibles, en conjunto con una plantilla con una cubierta que evite el desgaste debido contacto con los electrodos.

Se piensa en la manufactura de una cubierta basada en algún polímero que sea flexible y robusta a la vez, esto pensando evitar el desprendimiento de los conectores del bus de datos a los electrodos, así como el desgaste de los electrodos causado por el contacto directo.

Estas mejoras harían que este sistema sea utilizable en una clínica u hospital por el médico experto en el tema.

6. Conclusiones

En este trabajo se desarrolló un sistema de adquisición de datos con una interfaz gráfica para una matriz de sensores capacitivos para la identificación de problemas en niños con problemas de obesidad y sobre peso. Se adquirió conocimiento en técnicas especializadas de manufactura de sensores capacitivos y del depósito de películas delgadas metálicas en polímeros, en condiciones de alta temperatura.

Se logró la manufactura de 48 sensores. Los resultados presentados tienen el propósito de establecer la prueba de concepto sobre la implementación de la tecnología asociada. Con ese objetivo se muestra el desempeño de sensores individuales y en conjunto trabajando de la mano con el software. A pesar de que lo que se desarrolló en el presente trabajo es un prototipo, esto deja en claro, es posible realizar su implementación en arreglos más complejos con mejores tipos de acondicionamiento en una plantilla más robusta que son parte del trabajo futuro.

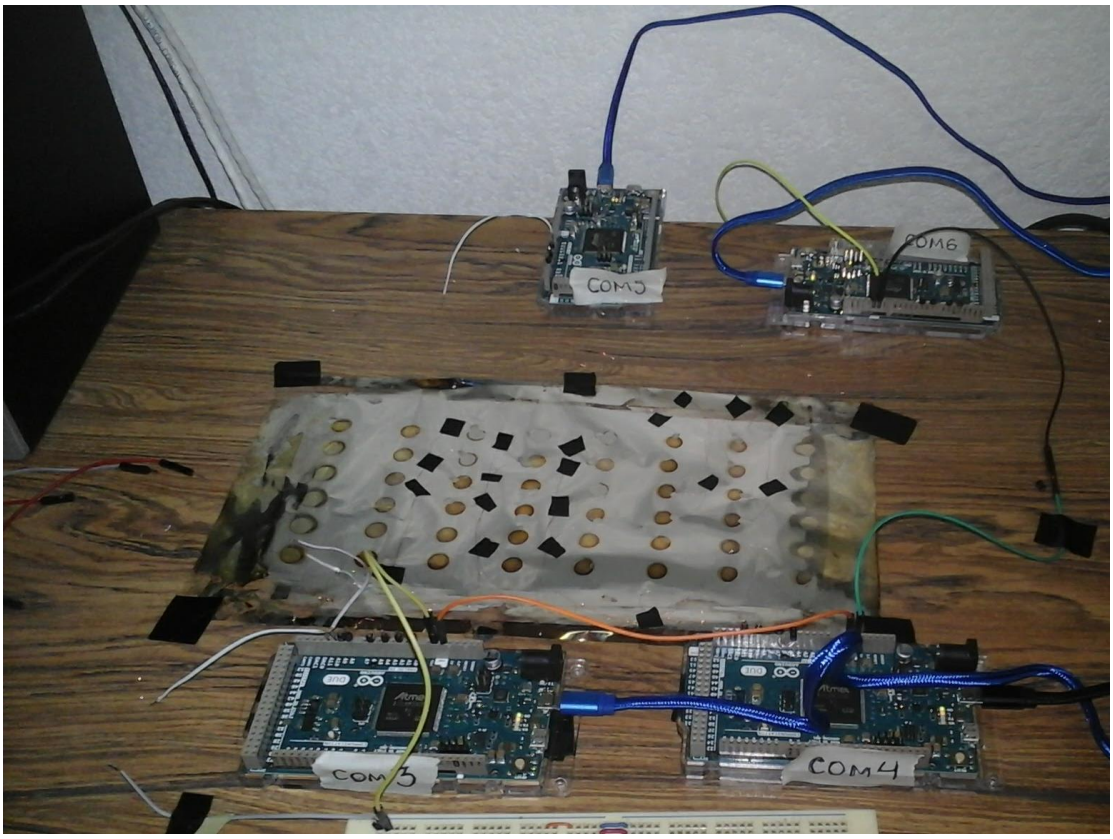


Figura 6.1. Prototipo de la matriz de sensores polares tipo capacitivos en conjunto con las tarjetas de adquisición de datos.

Se logró realizar un diseño original de sensores polares tipo capacitivo. Los cuales son muy económicos, compactos y relativamente fáciles de construir.

Se implementó de la adquisición de datos utilizando Arduino, donde su ambiente de desarrollo es código abierto y, por lo tanto, se pueden acceder al código de las librerías, se pueden modificar y no se requiere ninguna licencia para utilizarlo. De igual forma, es fácil de programar ya que cuenta con una gran cantidad de funciones para realizar todo tipo de tareas dentro de sus diferentes librerías, debido a esto, gran porcentaje del código más complejo que se requiere, como por ejemplo para establecer la comunicación serie, configurar una entrada analógica, se encuentran en dichas librerías. Otra gran ventaja es que no se necesita ninguna tarjeta de programación como sucede en la mayoría de las placas de desarrollo, Arduino cuenta con un software conocido como bootloader [36] que viene cargado en el microprocesador que permite la auto programación y evita la necesidad de contar con una tarjeta programadora externa para cargar instrucciones al microcontrolador.

Relativo al costo de las tarjetas de desarrollo, se puede conseguir una por menos de 14 dólares, que es un precio muy económico comparado con otras tarjetas que intentan cumplir los mismos requisitos. Evidentemente existen diferentes modelos y alternativas, el costo puede variar, pero no demasiado.

La metodología de programación y el desarrollo de las aplicaciones fue relativamente rápida y robusta, aplicable a prácticamente cualquier computadora portátil de uso de oficina, versátil ya que es modificable y ajustable a las diferentes arquitecturas y sistemas operativos. A su vez, el costo de software fue bastante económico debido al uso de una versión de LabVIEW estudiantil para la interfaz gráfica.

Se realizó una aplicación de bajo costo y con poco tiempo de desarrollo. Arduino podría ser la mejor opción disponible cuando lo que se quiere desarrollar no está destinado a la venta y no se dispone de mucho tiempo para el desarrollo de la aplicación. A consecuencia de que la programación no se realiza en lenguaje ensamblador directamente, el precio a pagar por el uso de las librerías es un retraso en la ejecución de las instrucciones, algunos cuantos microsegundos que en el caso de esta aplicación son irrelevantes, pero significativos para otras aplicaciones de adquisición de datos o control en tiempo real.

7.Referencias

[1] E. Terzic, A Neural Network Approach to Fluid Quantity, Hardcover: Springer, 2012.

[2] A. Ledoux, Theory of Piezoelectric Materials and Their Applications in Civil Engineering, Massachussets: MIT, 2011.

[3] A. H. A. Pereira, Cerámicas piezoeléctricas: funcionamiento y propiedades, São Carlos: ATCP Engenharia Física, 2010.

[4] J. C. Gallardo, Diseño de un compactador de PET de uso ligero, Culhuacan: IPN, 2008.

[5] E. Rélot, Estudio de mezclas y copolímeros de PET/PEN, Barcelona: eeigm, 2005.

[6] I. J. S.A., «Industrias JQ,» Plasticos de ingeniería, 20 julio 2016. [En línea]. Available: http://www.jq.com.ar/imagenes/productos/pet/dtecnicos/dtecnicos.htm. [Último acceso: 20 septiembre 2017].

[7] B. Reyes, Sensores piezoeléctricos de tipo capacitivo para aplicaciones en tecnología de imágenes fotoacústicas, México: UNAM, 2015.

[8] A. Pineda, Instrumentación Virtual. Fundamentos de programación gráfica con LabVIEW, México: Instituto Tenológico y de Estudios Superiores de Monterrey, 2011.

[9] N. Instruments, «National Instruments,» National Instruments, 10 Agosto 2017. [En línea]. Available: http://www.ni.com/data-acquisition/why-choose/esa/. [Último acceso: 2017 Septiembre 20].

[10] D. Ashlock, «Guía de acondicionamiento de señales para ingenieros,» National Instruments, Austin, 2015.

[11] N. Instruments, «LabVIEW,» National Instruments, 1 agosto 2017. [En línea]. Available: http://www.ni.com/es-mx/shop/labview/buy-labview.html. [Último acceso: 30 septiembre 2017].

[12] P. Martín, «USB,» fi.uba.ar, Argentina, 2009.

[13] A. López, «Puertos USB - Bus Serie Universal y descripción de la norma IEEE 1394,» *Sistemas de regulación y control automáticos,* vol. sv, nº sn, p. 18, 2003.

[14] S. Torres, «Universal Serial Bus,» iuma.ulpgc.es, 10 agosto 2010. [En línea]. Available: http://www.iuma.ulpgc.es/~avega/int_equipos/trab9899/usb_1/index.html. [Último acceso: 8 septiembre 2017].

[15] proyectosfie.com, «Universal Serial Bus, The Easy Way to Plug & Play,» 11 octubre 2002. [En línea]. Available: http://proyectosfie.com/html/documentos/usb/Completo%20Manual%20USB.pdf. [Último acceso: 11 octubre 2017].

[16] «¿Qué es un Micro USB?,» CurioSoando, 3 julio 2014. [En línea]. Available: https://curiosoando.com/que-es-un-micro-usb. [Último acceso: 6 octubre 2017].

[17] A. Aguirre, «Interfaz USB genérica para comunicación con dispositivos electrónicos,» 28 junio 2015. [En línea]. Available: https://www.fing.edu.uy/inco/grupos/mina/pGrado/pgusb/Docs/Presentacion_Avance.pdf. [Último acceso: 11 octubre 2017].

[18] E. López, «INGENIERIA EN MICROCONTROLADORES, Protocolo USB (UNIVERSAL SERIAL BUS),» micro, México, 2015.

[19] Arduino, «Arduino.cl,» Arduino, 1 enero 2017. [En línea]. Available: http://arduino.cl/arduino-due/. [Último acceso: 6 octubre 2017].

[20] J. Graff, «La Evolución de LabVIEW: Décadas de Desarrollo,» *Instrumentation Newsletter,* vol. 21, nº 2, pp. 3-6, 2009.

[21] anónimo, «Aprendiendo a manejar Arduino en profundidad,» Aprendiendo Arduino, 26 marzo 2015. [En línea]. Available: https://aprendiendoarduino.wordpress.com/2015/03/26/lenguaje-de-programacion-de-arduino-estructura-de-un-programa/. [Último acceso: 1 septiembre 2017].

[22] anónimo, «Arduino Forum,» Arduino, 13 noviembre 2012. [En línea]. Available: https://forum.arduino.cc/index.php?topic=132130.0. [Último acceso: 2 septiembre 2017].

[23] F. Martínez, «OpenWebinars,» OpenWebinars, 3 febrero 2015. [En línea]. Available: https://openwebinars.net/blog/tutorial-arduino-ide-arduino/. [Último acceso: 1 septiembre 2017].

[24] M. Verle, MICROCONTROLADORES PIC – PROGRAMACIÓN EN C CON EJEMPLOS, Belgrado: MikroElektronika, 2009.

[25] D. Mattana, «LENGUAJE DE PROGRAMACION C++,» Enjoy Life, 9 septiembre 2014. [En línea]. Available: http://enjoylife.com.ar/novedades/lenguaje-de-programacion-c/. [Último acceso: 1 septiembre 2017].

[26] N. Instruments, «National Instruments VISA,» National Instruments , 1 enero 2014. [En línea]. Available: https://www.ni.com/visa/. [Último acceso: 2 septiembre 2017].

[27] Anónimo, «Arduino,» Arduino, 1 enero 2017. [En línea]. Available: https://www.arduino.cc/reference/en/language/variables/data-types/int/. [Último acceso: 16 octubre 2017].

[28] J. D. Warren, Arduino Robotics, New York: Springer, 2011.

[29] M. Otis, VISA and Serial Communication, Texas: Measurement and Automation Experiments, 2011.

[30] anónimo, «Stack Overflow,» Stack Overflow, 26 mayo 2010. [En línea]. Available: https://stackoverflow.com/questions/2915785/how-many-bytes-is-n-r. [Último acceso: 3 octubre 2017].

[31] M. Kieran, Understanding Desktop Color, Berkley: Peachpit Press, 1994.

[32] National Instruments, «How Does a LabVIEW Color Box Interpret a Numeric Value for Color?,» National Instruments, 18 abril 2014. [En línea]. Available: http://digital.ni.com/public.nsf/allkb/B61D723CC7B7BE4E862568F8004C3ACB. [Último acceso: 24 octubre 2017].

[33] A. Zamora, Diseño, construcción y caracterización de un transductor ultrasónico angulado para aplicaciones médicas, México D.F.: FACULTAD DE INGENIERÍA, UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO , 2011.

[34] ETOOLS, «QUE ES Y PARA QUE SIRVE ARDUINO,» 1 10 2016. [En línea]. Available: http://www.electrontools.com/Home/WP/2016/04/20/que-es-y-para-que-sirve-arduino/. [Último acceso: 4 01 2018].

[35] I. ELECTRONICA, «INTEK ELECTRONICA,» 2009. [En línea]. Available: http://www.intekelectronica.com.ar/productos/arduino/. [Último acceso: 1 3 2018].

[36] Arduino, «Bootloader Development,» Arduino, 2018. [En línea]. Available: https://www.arduino.cc/en/Hacking/Bootloader. [Último acceso: 6 3 2018].

8.Anexos

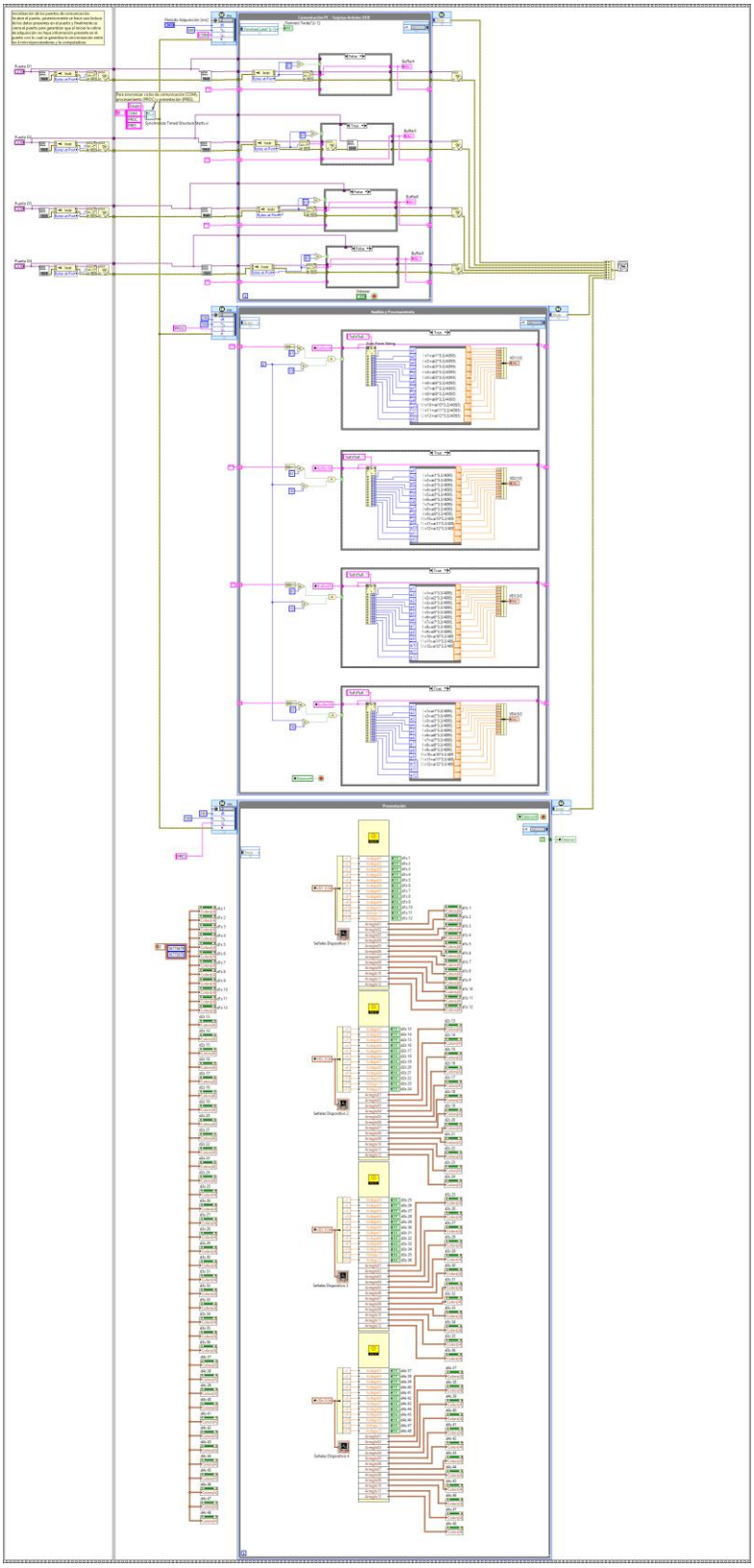
8.1. Código en Arduino

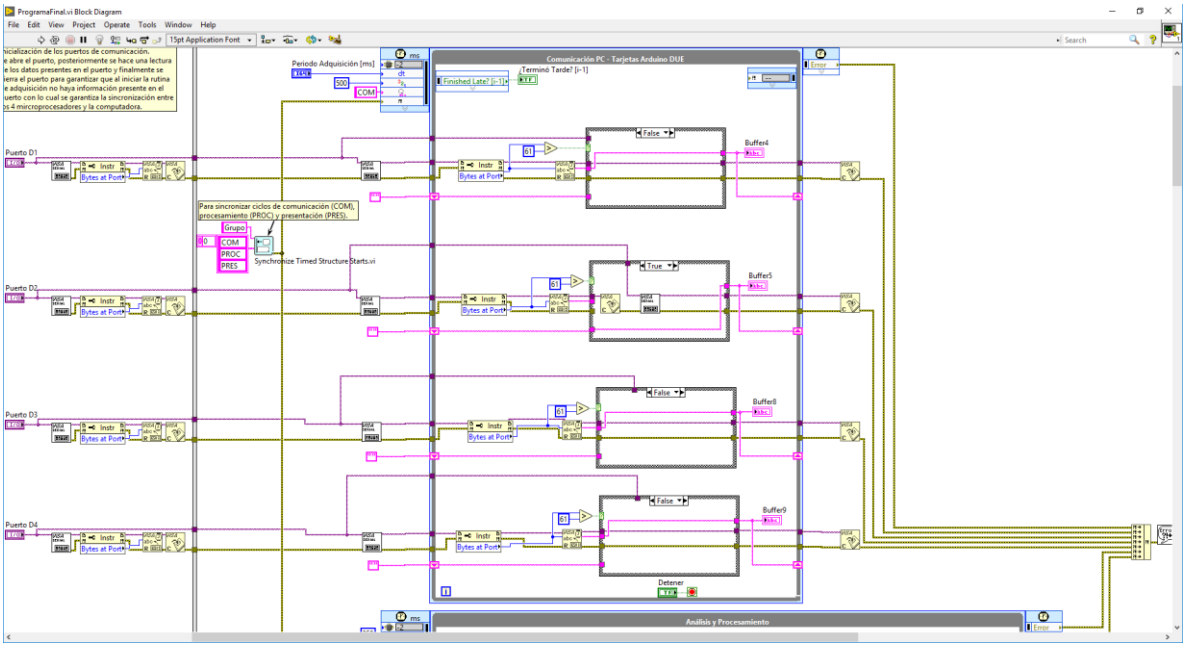
Programa para la programación del microcontrolador de la tarjeta de desarrollo Arduino.

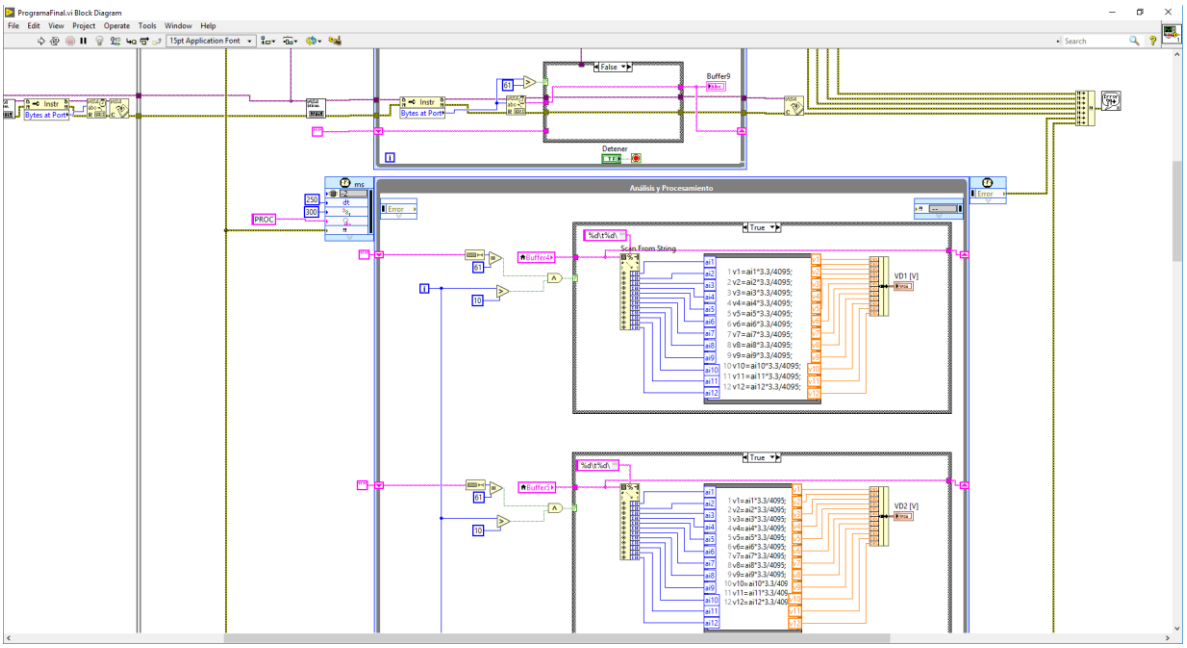
```
//Declaración de variables
int ai0 = 0; //Se declara e inicia de la varable que almacenara el voltaje leido por el pin 0
int ai1 = 0; //Se declara e inicia de la varable que almacenara el voltaje leido por el pin 1
int ai2 = 0; //Se declara e inicia de la varable que almacenara el voltaje leido por el pin 2
int ai3 = 0; //Se declara e inicia de la varable que almacenara el voltaje leido por el pin 3
int ai4 = 0; //Se declara e inicia de la varable que almacenara el voltaje leido por el pin 4
int ai5 = 0; //Se declara e inicia de la varable que almacenara el voltaje leido por el pin 5
int ai6 = 0; //Se declara e inicia de la varable que almacenara el voltaje leido por el pin 6
int ai7 = 0; //Se declara e inicia de la varable que almacenara el voltaje leido por el pin 7
int ai8 = 0; //Se declara e inicia de la varable que almacenara el voltaje leido por el pin 8
int ai9 = 0; //Se declara e inicia de la varable que almacenara el voltaje leido por el pin 9
int ai10 = 0; //Se declara e inicia de la varable que almacenara el voltaje leido por el pin 10
int ai11 = 0; //Se declara e inicia de la varable que almacenara el voltaje leido por el pin 11

char buffer[48]; //buffer de datos que contiene las 12 entradas analógicas
                 //este buffer de caracteres se envia por el puerto serie

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

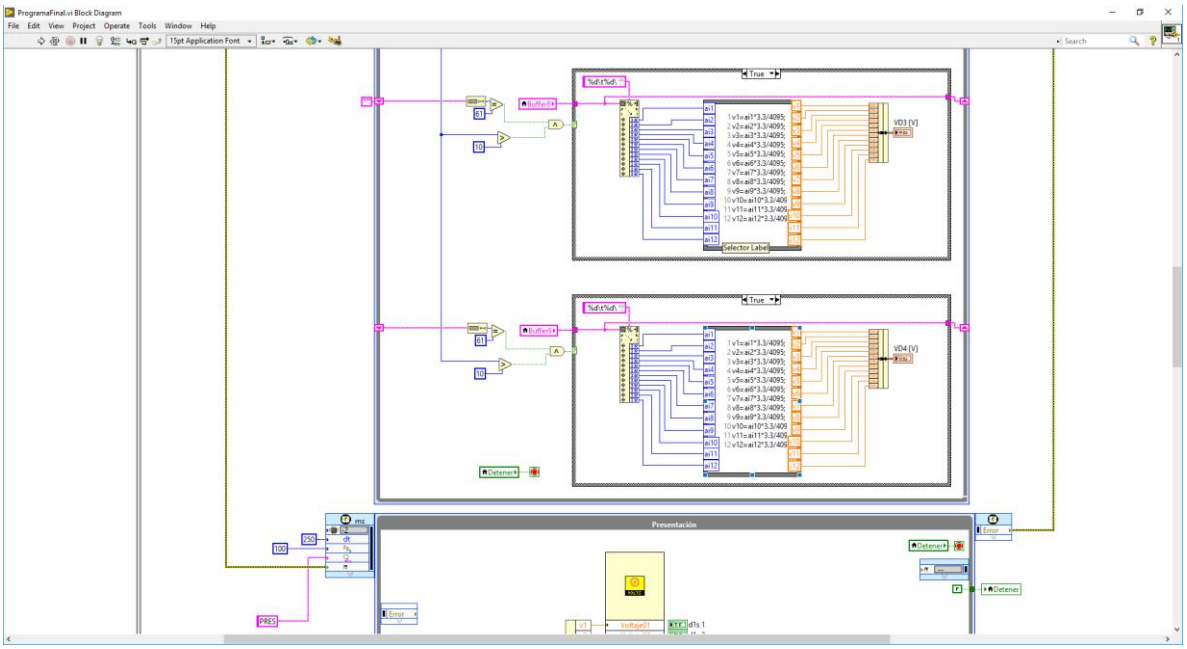
```
  analogReadResolution(12);
        }

void loop() {
ai0=analogRead(0);
ai1=analogRead(1);
ai2=analogRead(2);
ai3=analogRead(3);
ai4=analogRead(4);
ai5=analogRead(5);
ai6=analogRead(6);
ai7=analogRead(7);
ai8=analogRead(8);
ai9=analogRead(9);
ai10=analogRead(10);
ai11=analogRead(11);
sprintf(buffer,
"%04i\t%04i\t%04i\t%04i\t%04i\t%04i\t%04i\t%04i\t%04i\t%04i\t%04i\t%04i\t\n",
ai0,ai1,ai2,ai3,ai4,ai5,ai6,ai7,ai8,ai9,ai10,ai11); //04 - 4 digitos, i - enteros, \n - salto de linea
Serial.print(buffer); //Envio de lo datos por el puerto serie
delay(250); //Tiempo de ciclo en milisegundos
        }
```

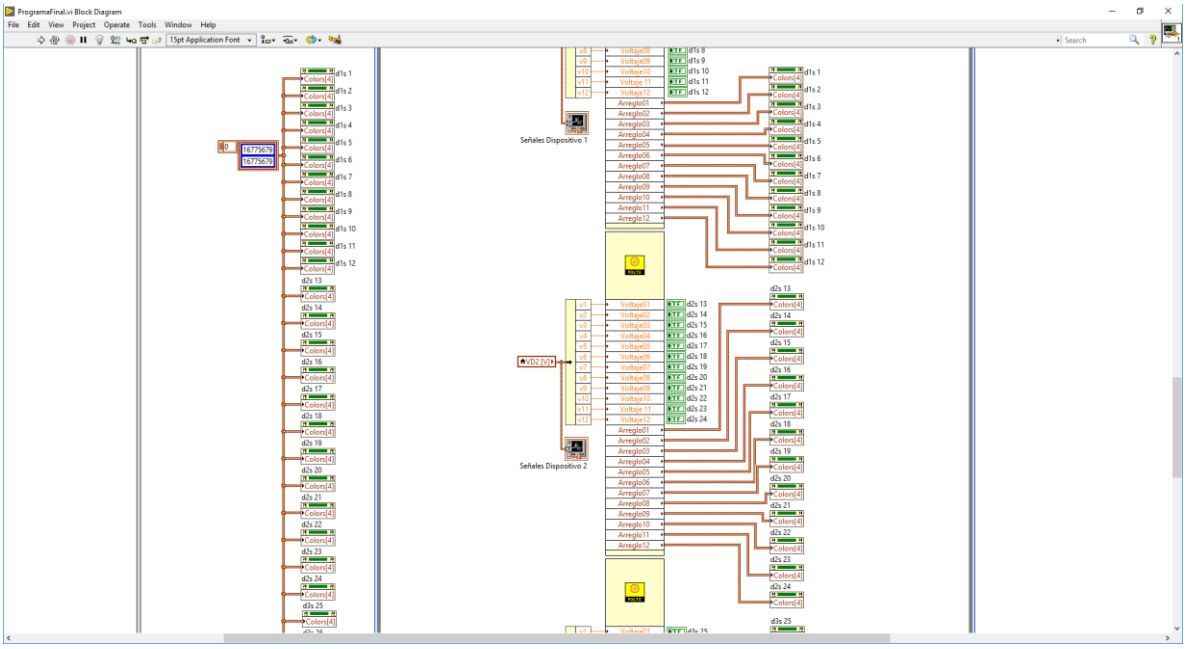
8.2. Código en LabVIEW

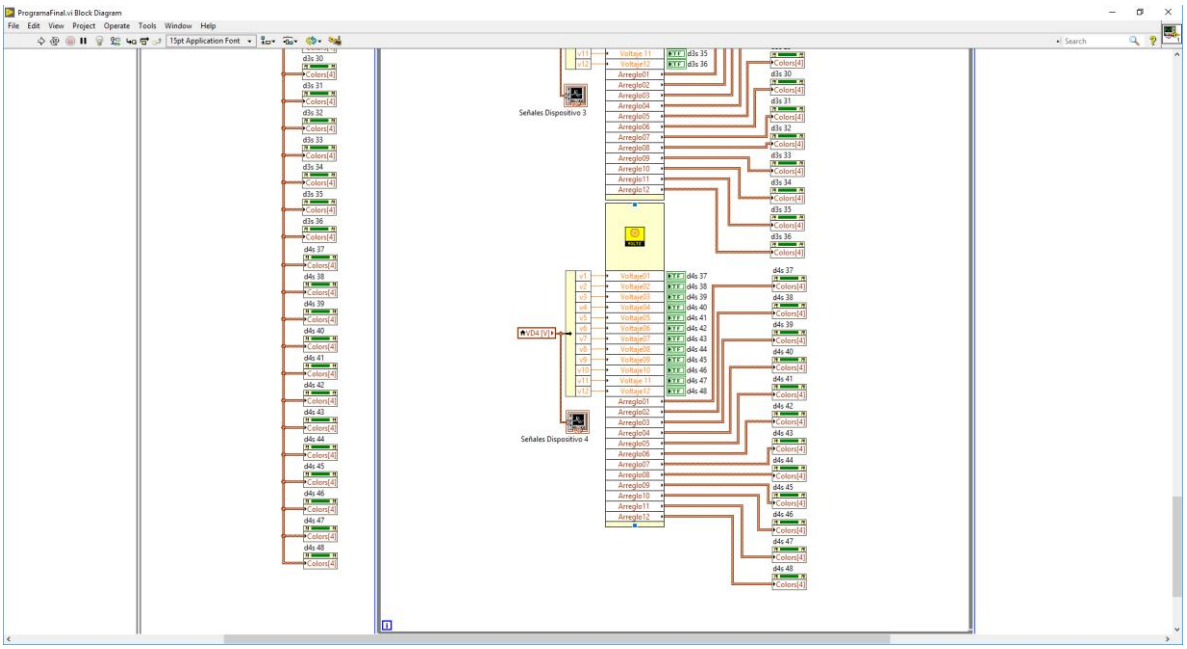



ProgramaFinal.vi Block Diagram
File Edit View Project Operate Tools Window Help
Comunicación PC - Tarjetas Arduino DUE
Periodo Adquisición [ms]
Terminó Tarde? [i-1]
Finished Late? [i-1]
Puerto D1
Puerto D2
Puerto D3
Puerto D4
Buffer4
Buffer5
Buffer8
Buffer9
Detener
Análisis y Procesamiento


ProgramaFinal.vi Block Diagram
File Edit View Project Operate Tools Window Help
Buffer9
Detener
Análisis y Procesamiento
PROC
Scan From String
VD1 [V]
VD2 [V]


ProgramaFinal.vi Block Diagram
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
%d\t%d\t
True
Buffer[0]
Buffer[1]
VD3 [V]
VD4 [V]
Selector Label
Detener
ms
dt
Presentación
Error
PRES


ProgramaFinal.vi Block Diagram
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
Señales Dispositivo 1
Señales Dispositivo 2
VD2 [V]
Colors[4]


ProgramaFinal.vi Block Diagram
File Edit View Project Operate Tools Window Help
Señales Dispositivo 3
Señales Dispositivo 4

8.3. Código en Python

Programa en Python utilizado para realizar la conversón de los datos adquiridos en formato txt a una variable de tipo arreglo lo cual permitió reproducir los experimentos realizados.

```
import numpy as np

from matplotlib import pyplot

f1 = open('1.txt') #Datos Sensor 1

datos1=f1.read().split()

for i in range(0,len(datos1)):

   datos1[i]=datos1[i].split(',')

A=np.array(datos1, dtype=float) #Conversión a Punto flotante

f2 = open('2.txt')  #Datos Sensor 2

datos2=f2.read().split()

for i in range(0,len(datos2)):

   datos2[i]=datos2[i].split(',')

B=np.array(datos2, dtype=float) #Conversión a Punto flotante

f3 = open('3.txt')  #Datos Sensor 3

datos3=f3.read().split()

for i in range(0,len(datos3)):

   datos3[i]=datos3[i].split(',')

C=np.array(datos3, dtype=float) #Conversión a Punto flotante

f4 = open('4.txt')  #Datos Sensor 4

datos4=f4.read().split()

for i in range(0,len(datos4)):
```

```
    datos4[i]=datos4[i].split(',')
D=np.array(datos4, dtype=float) #Conversión a Punto flotante
pyplot.figure(1)
pyplot.grid(True)
pyplot.plot(A,'r--')
pyplot.title('Sensor 44 Dispositivo 4')
pyplot.xlabel('Tiempo * 100 [ms]')
pyplot.ylabel('Amplitud [V]')
pyplot.figure(2)
pyplot.grid(True)
pyplot.plot(B,'g')
pyplot.title('Sensor 5 Dispositivo 1')
pyplot.xlabel('Tiempo * 100 [ms]')
pyplot.ylabel('Amplitud [V]')
pyplot.figure(3)
pyplot.grid(True)
pyplot.plot(C,'b-.')
pyplot.title('Sensor 18 Dispositivo 2')
pyplot.xlabel('Tiempo * 100 [ms]')
pyplot.ylabel('Amplitud [V]')
pyplot.figure(4)
pyplot.grid(True)
pyplot.plot(D,'k--')
```

```
pyplot.title('Sensor 31 Dispositivo 3')

pyplot.xlabel('Tiempo * 100 [ms]')

pyplot.ylabel('Amplitud [V]')

pyplot.figure(5)

pyplot.grid(True)

pyplot.plot(A,'r--')

pyplot.plot(B,'g')

pyplot.plot(C,'b-.')

pyplot.plot(D,'k')

pyplot.title('Multiples Sensores')

pyplot.xlabel('Tiempo * 100 [ms]')

pyplot.ylabel('Amplitud [V]')

#pyplot.subplots_adjust(left=None, bottom=0.1, right=None, top=0.9, wspace=None, hspace=0.7)
```