FACULTAD DE INGENIERÍA Y CIENCIAS AGROPECUARIAS

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA ELECTRÓNICO PARA EL CONTROL Y REPOSICIÓN DEL INVENTARIO DE PRODUCTOS REFRIGERADOS EN EL HOGAR

AUTOR

Cristian Guillermo Játiva Mafla

AÑO

2017

FACULTAD DE INGENIERÍA Y CIENCIAS AGROPECUARIAS

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA ELECTRÓNICO PARA EL CONTROL Y REPOSICIÓN DEL INVENTARIO DE PRODUCTOS REFRIGERADOS EN EL HOGAR

Trabajo de Titulación presentado en conformidad con los requisitos establecidos para optar por el título de Ingeniero en Electrónica y Redes de Información

Profesor guía

Msc. Fausto Francisco Charro Simbaña

Autor

Cristian Guillermo Játiva Mafla

Año

2017

DECLARACIÓN PROFESOR GUÍA

Declaro haber dirigido este trabajo a través de reuniones periódicas con el estudiante, orientando sus conocimientos y competencias para un eficiente desarrollo del tema escogido y dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación

_____

Fausto Francisco Charro Simbaña

Master Universitario en Ingeniería de Sistemas Electrónicos

CI: 1712038734

DECLARACIÓN DOCENTE CORRECTOR

Declaro haber dirigido este trabajo a través de reuniones periódicas con el estudiante, orientando sus conocimientos y competencias para un eficiente desarrollo del tema escogido y dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación

---

David Fernando Pozo Espín

Master Universitario en Automática y Robótica

CI: 1717340143

DECLARACIÓN DE AUTORÍA DEL ESTUDIANTE

Declaro que este trabajo es original, de mi autoría, que se han citado las fuentes correspondientes y que en su ejecución se respetaron las disposiciones legales que protegen los derechos de autor vigentes.

_____

Cristian Guillermo Játiva Mafla

CI: 1723641872

AGRADECIMIENTOS

El agradecimiento para todos los docentes que con sus consejos y sabiduría me han inculcado valores y enseñanzas tan útiles la vida profesional y personal.

En especial el agradecimiento para mis docentes Francisco Charro y David Pozo ya que su ayuda fue clave en la realización de este proyecto.

DEDICATORIA

A Dios por cada día brindarme una oportunidad de tener salud, vida y felicidad.

A mi novia que con su apoyo en mi vida es y será la persona con la cual estaré bien.

A mi tía y abuelita que en vida compartieron momentos y enseñanzas que las recuerdo con cariño. Dios las tenga en su gloria.

Y, a mi familia entera, a mis padres que están conmigo en los momentos más importantes de mi vida que con su paciencia, esfuerzo y cariño han sido mi inspiración a lo largo de esta carrera.

RESUMEN

En la actualidad el día a día de las personas cada vez es más ajetreado y una necesidad que toda familia tiene es la alimentación, viéndose afectado el cliente por cuestiones de tiempo y estrés que podría ser causado por la ida y vuelta al supermercado, este prototipo ha sido propuesto como alternativa para facilitar al cliente la adquisición de su despensa.

En el siguiente trabajo de titulación se plantea el diseño e implementación de un sistema electrónico para el control y reposición del inventario de productos refrigerados en el hogar con la finalidad de automatizar el inventario existente en el refrigerador y reportar la carencia de uno de los productos.

Para controlar los datos enviados desde el prototipo se instalaron tres tipos de sensores que son : sensores de movimiento , peso y distancia , el sensor de movimiento detecta la presencia de huevos en una cubeta y permite saber cuándo uno de estos se retiró , el sensor de peso sirve para medir si un determinado producto (carne o queso) se encuentra con un peso menor al programado , y el sensor de distancia desempeña la función de medir la cantidad de agua (o cualquier otro producto líquido) e informar si la cantidad se encuentra por debajo de lo programado, también se informará si los productos que están en el sensor de peso y de distancia se encuentran ausentes durante un tiempo específico . En todos los casos cuando los sensores detectan escases o nivel bajo de algún producto envía una notificación vía correo electrónico al proveedor. Así mismo el prototipo cuenta con botones los cuales cumplen la función de notificar al proveedor de manera manual. El prototipo está desarrollado en el entorno Arduino Uno, el cual interpreta los datos enviados por los sensores y los módulos Xbee que comunican los datos enviados desde el refrigerador inalámbricamente hacia un host para su análisis y envió de correos al proveedor.

La implementación de este proyecto se realizará en una maqueta simulando a una refrigeradora con sus debidos compartimientos para la ubicación estratégica de los módulos electrónicos, y ubicación de productos.

ABSTRACT

Today the day to day of people is becoming more and more busy and a need that every family has is food, being affected the customer due to time and stress issues that could be caused by the round trip to the supermarket this prototype has Been proposed as an alternative for customer's ease.

In the following titling work, the design and implementation of an electronic system for the control and replacement of the inventory of refrigerated products in the home is proposed in order to automate the existing inventory in the refrigerator and report the lack of one of the products.

In order to control the data sent from the refrigerator three types of sensors were implemented which are: movement sensors, weight and distance, the motion sensor will be used to detect if one of these eggs was removed, the weight sensor It will be used to measure if a certain product (meat or cheese) is less than the programmed weight and in the case of complying said will be notified, and the distance sensor plays the role of measuring the amount of water (or any other Liquid product) and inform if the quantity is below the programmed, also the prototype will have buttons which when pressed the notification will be sent to the food supplier informing that the same has been finished, also will have a known electronic plate Such as Arduino Uno which will interpret data sent by sensors and Xbee modules which will communicate the data sent from the refrigerator wirelessly to a host for analysis and sent mails in the event of a product failure.

The implementation of this project will be done in a model simulating a refrigerator with its proper compartments for the location of electronic modules, and location of products which should be strategically located so that the sensors can the data mentioned above.

ÍNDICE

ÍNDICE DE FIGURAS

1. Capítulo I. Introducción

La constante rutina diaria de las personas en estos últimos tiempos se ha visto afectada debido a que las familias requieren más ingresos a sus hogares lo cual los ha llevado a trabajar inclusive mucho más de 8 horas diarias , impidiéndoles realizar algunas obligaciones necesarias entre las cuales podemos resaltar la falta de tiempo para comprar alimentos al supermercado debido al tiempo empleado en las actividades diarias , la distancia del supermercado al hogar , la congestión de tránsito que puedan existir en las ciudades, el estrés que podría generar al ver en el supermercado colas en las cajas para pagar las compras, entre otras.

Por tal razón la innovación tecnológica esta cada día en constante desarrollo para poder facilitar las actividades cotidianas de las personas y porque no de todos los seres vivos.

La propuesta es la creación del prototipo para que sirva de alternativa para solucionar los problemas mencionados anteriormente con la puesta en práctica de un concepto nuevo e innovador como lo es el Internet de las cosas (IoT) la cual es que todos los objetos de nuestra vida diaria estén interconectados entre ellos y ser notificado el estado de los mismos vía Internet para saber si están en correcto funcionamiento o si necesitan mantenimiento o ser reemplazados sin la necesidad de que un ser humano este en constante vigilancia del mismo ya que existirán sensores que serán capaces de realizar esta labor por nosotros.

1.1. Antecedentes

Dentro de las necesidades de cualquier persona los electrodomésticos vienen a ser indispensables debido a la ayuda que estos nos brindan.

En la actualidad el mercado de los electrodomésticos (como todos los mercados) actualizarán sus funcionalidades para ser denominados “electrodomésticos inteligentes” y estarían conectados a la red (Internet de las

Cosas) comunicándose con varios dispositivos alrededor con la opción de poder intercambiar información y a su vez que los mismos electrodomésticos pudieran ser controlados remotamente o mediante una aplicación Web.

Gracias a la renovación de los electrodomésticos se puede obtener ventajas como la automatización, en particular de los frigoríficos que se encuentran en la mayoría de despensas o supermercados.

1.2. Alcance

El alcance del proyecto contempla que el prototipo será capaz de interactuar directamente vía correo electrónico con el proveedor de alimentos para la notificación de la carencia de determinado producto, cada evento que se realice será también almacenado en una base de datos.

Para este proyecto se contemplan cuatro productos de consumo en el hogar, ubicados en una maqueta que simule la distribución física de los alimentos, los mismos que deben estar ubicados específicamente dentro de un recipiente, y mediante sensores de presencia, volumen y peso programados con un tiempo de espera para el retorno del recipiente, permite controlar el stock de un producto.

El prototipo cuenta también con botones de pedido directo que enviará la notificación al proveedor de alimentos.

Al contar con notificaciones vía correo electrónico es necesario que el host tenga una conexión a Internet estable mientras el prototipo esté en funcionamiento.

1.3. Justificación

El proyecto busca una interacción entre el cliente y el proveedor que aproveche las nuevas tecnologías, introduciendo el concepto de Internet de las Cosas en asuntos cotidianos como la adquisición de productos para el hogar. Este

servicio reduce el tiempo que el cliente consume cuando va al supermercado y podría ser aprovechado en otras actividades.

En el desarrollo del proyecto se pondrán en práctica lo aprendido en las algunas materias de la malla curricular de la carrera como: programación estructurada, microcontroladores, sistemas de comunicación radiantes, tecnologías inalámbricas, entre otras.

1.4. Objetivo general

- Diseñar e implementar un prototipo que permita controlar el stock de los productos, y gestionar su reposición cuando se hayan terminado mediante la notificación a un proveedor de alimentos vía correo electrónico.

1.5. Objetivos específicos

- Diseñar e implementar el circuito electrónico en una maqueta que simule la distribución física de los alimentos y sus contenedores.
- Diseñar un sistema de notificación mediante correo electrónico entre el consumidor y el proveedor de alimentos.
- Crear una base de datos para almacenar los eventos del prototipo.
- Analizar el funcionamiento y adaptación de los sensores a utilizarse en el prototipo.
- Documentar los resultados del proyecto.

2. Capítulo II. Marco teórico

El proyecto de titulación consta con las siguientes implementaciones tecnológicas tanto en hardware como en software, los elementos usados son:

HARDWARE

- 1 sensor ultrasónico.
- 2 sensores PIR (Sensor pasivo infrarrojo).
- 1 celda de carga
- 1 módulo HX711
- 1 Arduino Uno
- 2 módulos Xbee (Emisor y Receptor).
- 3 pulsadores

SOFTWARE

- XCTU (Configuración de módulos Xbee).
- Arduino Software.
- Processing
- Proteus
- phpMyAdmin

2.1. Domótica

La domótica es la integración de la tecnología para automatizar las acciones que se realizan en los hogares.

Para ello los hogares (y también edificios) deben tener instalados varios componentes electrónicos, como: los sensores, actuadores, equipos de comunicación, central domótica.

La central domótica es dónde se controlan los elementos de un hogar inteligente que puede ser mediante una computadora, o un dispositivo inteligente (Tablet, sistemas embebidos, smartphones), a su vez deben contar con el software que gestionará los eventos que se desean realizar dentro del hogar.

Los sensores se encargan de obtener información del entorno en el que se los instala, el actuador al momento de recibir una instrucción es capaz de activar un dispositivo, por ejemplo: en un parqueadero inteligente al momento de pulsar un botón se abre automáticamente la entrada para parquear el auto.

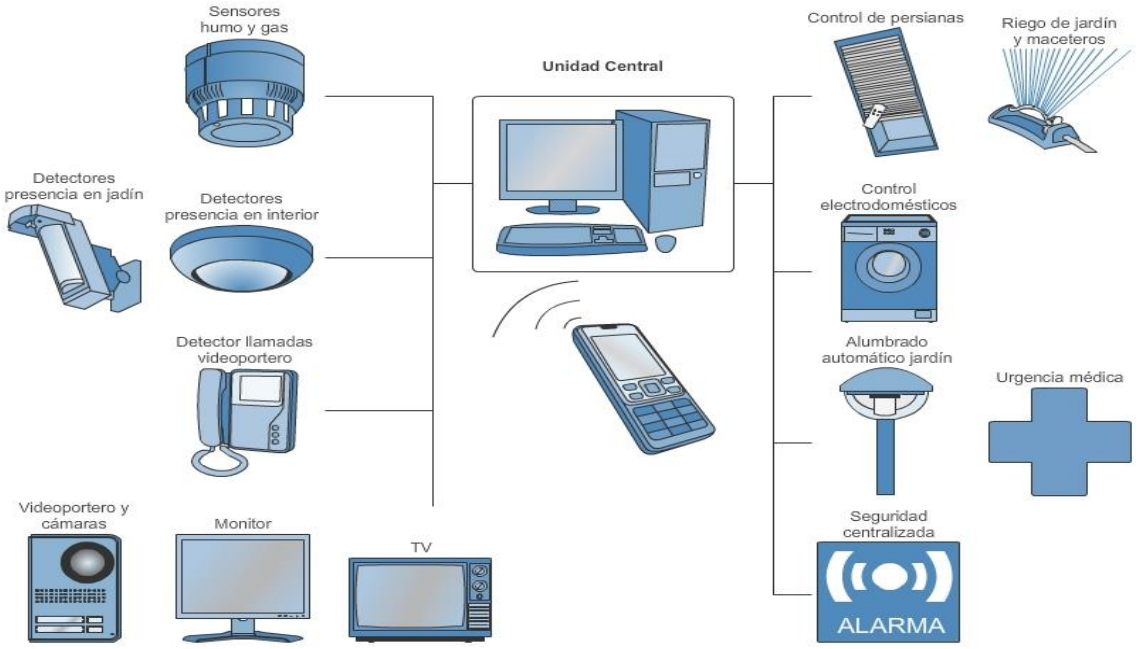

Figura 1. Elementos de un hogar domótico
Tomado de: iReformas (2015)

La domótica tiene un largo catálogo de aplicaciones para el hogar y las más sobresalientes se las detalla a continuación:

- **Seguridad**

La implementación de tecnología en el hogar y edificios facilita el tema de seguridad, mediante la instalación de cámaras IP, sensores de proximidad, detectores de humo, sensores biométricos, para el monitoreo y control de quien accede a nuestras instalaciones.

- **Entretenimiento**

El objetivo del entretenimiento es que el usuario acceda a las funciones de los dispositivos electrónicos a cualquier hora mediante una conexión a Internet, mediante televisores, equipos de sonido, consolas de videojuegos, etc.

- **Bienestar**

La posibilidad de controlar determinado actuador para la automatización desde un dispositivo inteligente, como encender las luces del hogar remotamente.

- **Ahorro de Energía**

Es posible limitar el consumo de energía eléctrica, telefónica, internet, programando un tope de consumo.

- **Comunicaciones**

Para el intercambio de datos entre los sensores y actuadores, la comunicación es fundamental, y mediante un dispositivo final como una laptop o un Smartphone podemos interactuar remotamente con los dispositivos instalados y monitorear su correcto funcionamiento.

2.2. Sensores

Sensores se denominan a los dispositivos que son capaces de transformar una señal física en una señal eléctrica.

Los sensores son muy usados en automatización debido a que la información tomada se la puede cuantificar, y con esa información interpretarla en diferentes tipos de aplicaciones electrónicas, mecánicas o industriales.

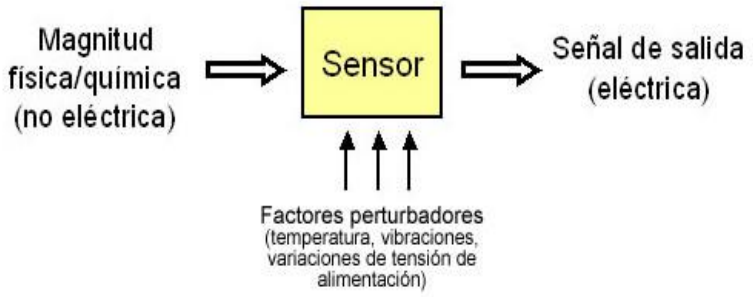

Figura 2. Funcionamiento de un sensor

Tomado de: Del Ángel (2015)

Existen diversos tipos de sensores que son:

- Mecánicos. - Actúan a los estímulos físicos que pueden ser: vibraciones, presión, velocidad, fuerza.
- Térmicos. - Son capaces de cuantificar datos de temperatura.
- Eléctricos. - Permiten medir las cantidades eléctricas como corriente, voltaje, resistividad.
- Magnéticos. – Sensores capaces de medir campos y flujos magnéticos.

2.3. Sensores Ultrasónicos

Los sensores ultrasónicos cumplen la función de medir distancias al detectar ondas sonoras a su alrededor, la distancia a medir puede variar según el tipo de sensor que van desde poder detectar objetos a centímetros de distancia hasta metros.

El funcionamiento del sensor ultrasónico consiste en producir un sonido y calcular cuánto tiempo ese sonido producido tarda en regresar.

La ecuación que determina la distancia del objeto al sensor es la siguiente:

$$d = \frac{1}{2} V * t \qquad \text{(Ecuación 1)}$$

Las variables de la ecuación descrita son:

d: Distancia total medida

V: Velocidad ultrasonido en el aire = 340m/s

t: tiempo transcurrido desde la emisión y recepción del pulso

Sensor Ultrasónico HC-SR04

Figura 3. Sensor ultrasónico HC-SR04
Tomado de: Sparkfun (s.f.)

El sensor HC-SR04 es ampliamente usado en aplicaciones y proyectos relacionados con electrónica debido a su compatibilidad con microcontroladores, Ras Berry Pi y Arduino.

Cuenta con 4 pines que son:

Vcc: Entrada a 5 volts

GND: Conexión a tierra

Trig: Envía señal acústica (ultrasónica)

Echo: Recibe la señal acústica enviada por Trig.

Básicamente el sensor al momento de detectar la proximidad de algún objeto el pin de Trig emite una señal de 10µs, que consta de 8 pulsos cada uno de ellos con 40 kHz, al momento de detectar presencia todo ese pulso regresa al sensor y será la señal recibida por el pin Echo, esta señal recibida es una señal eléctrica.

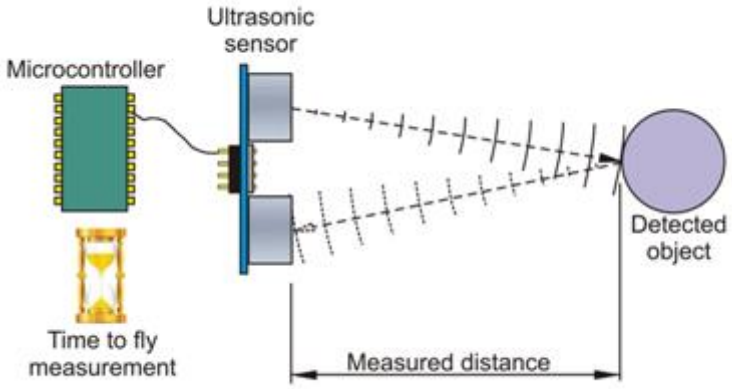

Figura 4. Funcionamiento de un sensor ultrasónico HC-SR04

Tomado de: Techmake Electronics. (s.f.)

La distancia expresada en cm será expresada por la siguiente ecuación:

$$Distancia_{cm} = \frac{Duración\ del\ pulso\ (\mu s)}{58}$$

(Ecuación 2)

Distancia: Distancia Total entre el objeto a medir y el sensor

Duración del pulso: Tiempo en el que el pulso retorna al sensor, por el pin Echo.

Las características generales de este sensor son:

Tabla 1.

Características Sensor HC-SR04

Descripción	Característica
Alimentación	5v
Frecuencia	40 KHz
Distancia Máxima	250 cm
Angulo de vista	< =15 grados
Temperatura Operativa	0 – 70°C

Adaptado de: Pérez (s.f.)

2.4. Sensores PIR (Sensor Pasivo Infrarrojo)

El sensor de movimiento o sensor PIR (Passive Infra red) es un dispositivo electrónico que es capaz de registrar movimiento en su campo de acción, el sensor detecta el calor emitido por el objeto al cual se pone en contacto.

Como respuesta al movimiento de cierto objeto en su campo de acción su nivel lógico puede estar en dos valores (0-1).

Es capaz de detectar el movimiento de objetos a una distancia máxima de 5 metros. Debido a su versatilidad, costo y dimensiones son usados para sistemas de detección de intrusos, iluminación, automatización, robótica y domótica.

Las características de este sensor son:

- Voltaje de Alimentación: 5 V
- Conexión a Tierra (GND)
- Módulos de calibración
- Temperatura Operación: -15°C a 70°C
- Lentes Fresnel para una mejor línea de vista infrarroja
- Amplificador incorporado para la detección de radiación emitida, provocando cambios en el voltaje.
- Salida para conexión con microcontroladores.

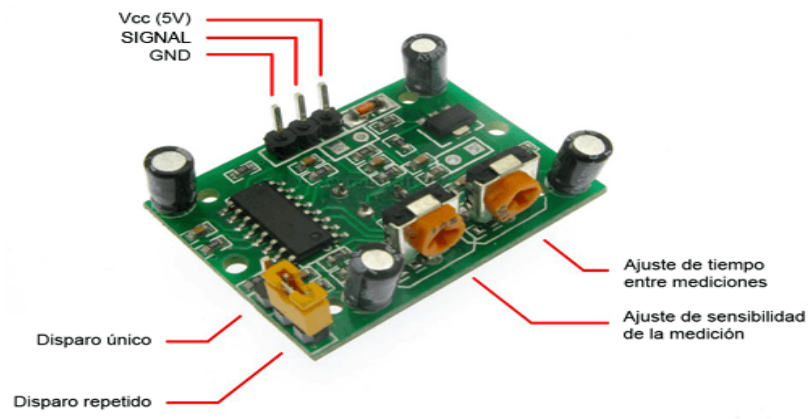

Figura 5. Esquema Eléctrico Sensor Pasivo Infrarrojo

Tomado de: Llamas (2015)

Cuando las señales infrarrojas alrededor del sensor empiezan a variar, el amplificador se activa para indicar que ha detectado movimiento a su alrededor, aumentando el voltaje que depende de la radiación producida por el objeto detectado. La salida va a permanecer activada por un periodo de tiempo determinado (dependiendo la configuración de los módulos de calibración del sensor) con lo cual el micro controlador conectado a la salida digital del sensor PIR interpreta esa información determinando si existe movimiento.

La longitud de onda irradiada por la radiación infrarroja va a ser mayor que la luz visible que permitirá que sea detectado el calor generado por la radiación infrarroja producida por los objetos.

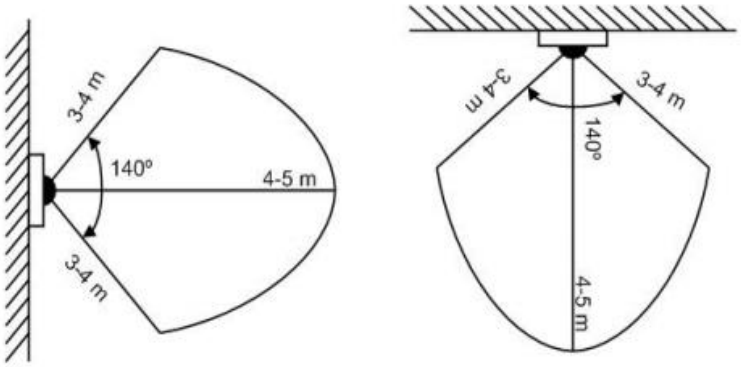

Figura 6. Alcance de un sensor PIR

Tomado de: Vega, Salgado, Lagos, Tapia, y Sánchez (2014)

2.5. Celda de carga

Las celdas de carga o galgas extensiométricas son dispositivos comúnmente usados para transformar la fuerza o peso en una señal eléctrica analógica. Internamente cuentan con las denominadas galgas que al detectar una fuerza ejercida estas comienzan a deformarse obteniendo el valor de la fuerza o el peso en esa celda.

Figura 7. Celda de carga

Tomado de: Logicbus. (s.f.)

2.5.1. Funcionamiento de las celdas de carga

Una celda de carga se encuentra compuesta en su interior de 4 galgas extensiométricas y están interconectadas una a otra en forma de puente, este circuito es mejor conocido como el puente Wheatstone, el principio de funcionamiento del puente de Wheatstone se basa en deformar las galgas al detectar un peso, el peso ejercido en esa celda de carga será convertido a señales eléctricas.

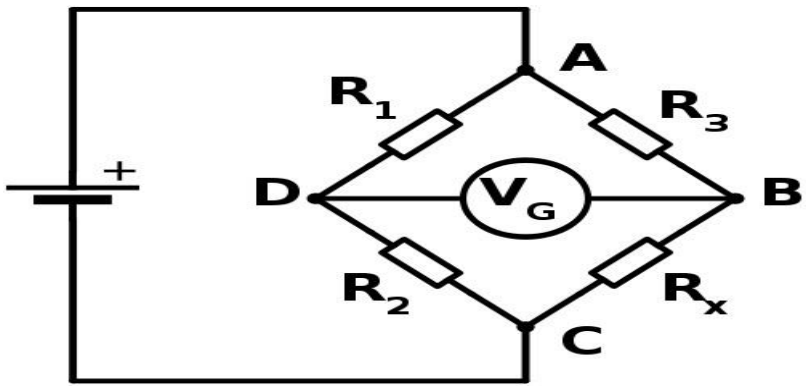

Figura 8. Puente Wheatstone

Tomado de: Casares (2016)

2.5.2. Tipos de Celdas de Carga

2.5.2.1. Celdas de carga con un solo punto

El uso de este tipo de celdas es utilizado para un cierto rango de peso que van desde objetos con un peso de 50 gramos hasta un máximo de 50 kg. Puede estar construido de aluminio, acero inoxidable o hierro.

Consta de dos agujeros en donde se puede ajustar tornillos para mantener su base fija y no causar problemas al momento de la lectura del peso colocado.

Consta de cuatro cables que van a ir conectados a un módulo HX 711 para ser interpretados los datos tomados por la celda de carga en un microcontrolador.

Figura 9. Celda de carga de un solo punto
Tomado de: Logicbus. (s.f.)

2.5.2.2. Celdas de carga de botón.

La forma de este tipo de celda es una circunferencia con un botón en el centro donde el peso será colocado, están fabricadas de acero inoxidable y en su interior cuentan con galgas de hoja de metal, son utilizadas en su mayoría en el sector industrial.

Figura 10. Celdas de carga tipo botón

Tomado de: Logicbus. (s.f.)

2.5.2.3. Celdas de carga tipo S.

Usadas en el sector industrial con el objetivo de detectar tensión y compresión.

Figura 11. Celdas de carga tipo S

Tomado de: Logicbus. (s.f.)

2.5.3. Calibración

Con la siguiente fórmula se obtiene el peso o fuerza aplicada a la celda de carga:

$$\text{Peso o Fuerza esperada} = \frac{\text{Valor de lectura}}{\text{Medida Real (Kg, lbs, etc)}} \qquad \text{(Ecuación 3)}$$

Donde “Valor de lectura” es el valor que se obtiene al poner el objeto en la celda de carga y “Medida real” es el peso exacto del objeto que puede ser expresado en diferentes magnitudes.

Tabla 2.

Especificaciones de las celdas de carga

Especificaciones de la Celda de Carga		
Capacidad Nominal de la Celda de Carga	50, 100, 200, 300, 500, 1,000, 2,000, 3,000, 5,000, 10,000 (lb)	25, 75, 100, 200, 500, 1,000, 2,000, 5,000 (kg)
Salidas Nominales	2.0 ± 0.2 mV/V	2.0 ± 0.2 mV/V
Balance en Cero	± 1.0% de C.N.	± 1.0% de C.N.
Error Combinado Debido a la No Linealidad y a la Histéresis	0.03% de C.N.	0.03% de C.N.
No Repetibilidad	0.01% de C.N.	0.01% de C.N.
Compensación de Temperatura	-10° a +40° C +14° a +104° F	-10° a +40° C +14° a +104° F
Resistencia en los Terminales	Entrada: 350Ω Señal: 350Ω ±3Ω	Entrada: 350Ω Señal: 350Ω ±3Ω
Voltaje de Excitación	20 VCD máximo	20 VCD máximo
Resistencia del Aislamiento	5 GigaΩ mínimo a 50 VCD	5 GigaΩ mínimo a 50 VCD
% de Cargas Máximas de C.N.	Sobrecarga Segura: 150 Sobrecarga Final: 300 Carga Lateral Segura: 100	Sobrecarga Segura: 150 Sobrecarga Final: 300 Carga Lateral Segura: 100

Tomado de: METTLER TOLEDO, (2001)

2.6. Módulo HX711

El módulo HX711 es un dispositivo analógico / digital, es la interface entre el microcontrolador y la celda de carga, es capaz de interpretar los datos obtenidos de la celda de carga, para así obtener un valor de fuerza o peso aplicado en la celda.

El módulo consta de 10 pines de ellos 6 pines están a un extremo y van conectados con la celda de carga y 4 pines que están al otro extremo se deben conectar con el microcontrolador.

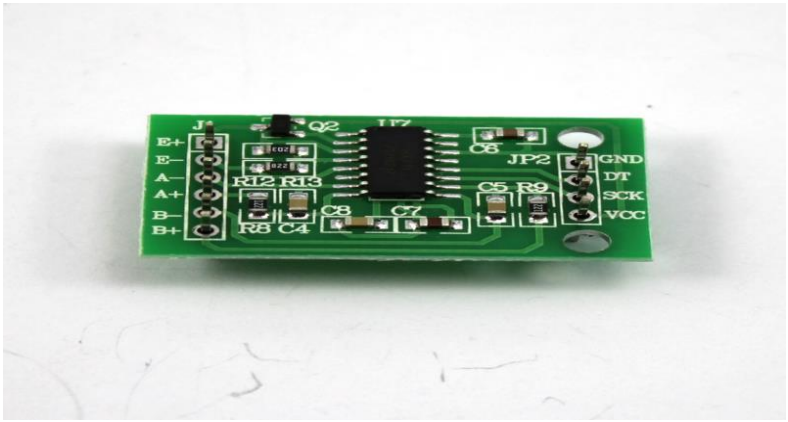

Figura 12. Módulo HX711

Tomado de: Sparkfun Load Cell. (s.f.)

Cuando se conecta el módulo HX711 a un microcontrolador se debe ajustar los valores de lectura para así poder obtener datos más precisos, de acuerdo a la escala medida por la celda de carga (Usar la Ecuación 3), la comunicación del HX711 con el microcontrolador se realiza mediante la salida de datos (DT) y reloj (SCK).

Tabla 3.

Conexiones de Celda de carga a Módulo HX711

Celda De Carga	Módulo HX711
Cable Rojo	Pin E+
Cable Negro	Pin E-
Cable Verde	Pin A-
Cable Blanco	Pin A+

Tomado de: Naylamp Mechatronics (2015)

Tabla 4.

Conexiones de Módulo HX711 a Arduino

Módulo HX711	Arduino UNO, MEGA, NANO
Pin GND	Pin GND
Pin DT	Pin A1
Pin SCK	Pin A0
Pin VCC	Pin 5V

Tomado de: Naylamp Mechatronics (2015)

2.7. Comunicaciones Inalámbricas

Como bien su nombre lo menciona, las comunicaciones inalámbricas se las realiza sin ningún medio físico como los cables, el medio de transmisión de esta tecnología es el aire mediante ondas electromagnéticas dentro de una banda específica dependiendo del uso que se quiere ofrecer para la transmisión de datos.

Para la conexión a una red inalámbrica el dispositivo deberá contar con una antena para poder establecer comunicación y además se deben analizar aspectos importantes tales como potencia, alcance o efectividad, todo lo mencionado dependerá de la Zona de Fresnel, que consiste en que las ondas electromagnéticas viajen sin ningún tipo de obstáculo en su camino que pueden ser: montañas, edificios, árboles, etc.

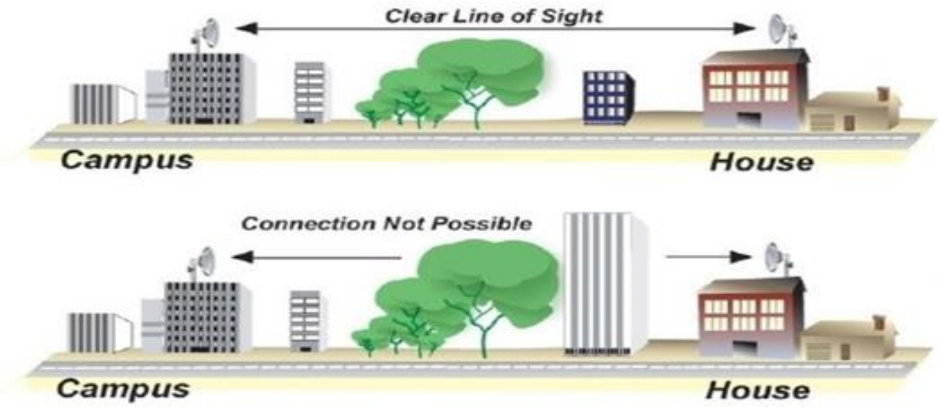

Figura 13. Zona de Fresnel

Tomado de: Santos (2015)

Para determinar si el enlace es apto se debe determinar la primera Zona de Fresnel se debe mantener despejado el 60%, y se obtiene de la siguiente fórmula:

. Fórmula de la primera Zona de Fresnel

$$r = 17.32 * \sqrt{(\frac{d}{4f})}$$ (Ecuación 4)

Donde las variables representan:

d: Distancia entre los enlaces en kilómetros

f: Frecuencia expresada en GHz.

r: Altura de la primera zona de Fresnel.

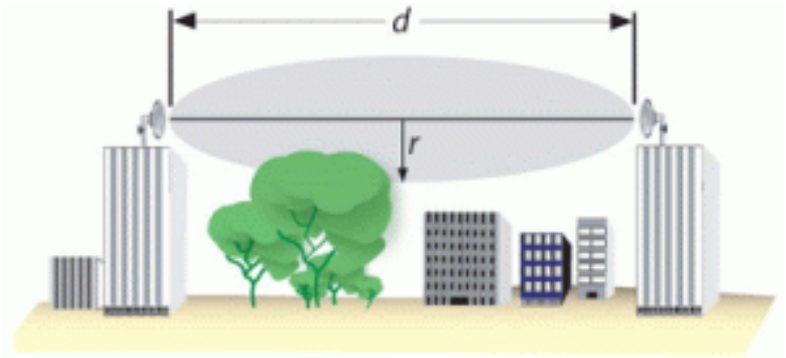

Figura 14. Primera Zona de Fresnel
Tomado de: Molina (s.f.)

Tipos de Comunicaciones Inalámbricas

De acuerdo al área de cobertura se clasifican a las comunicaciones inalámbricas de la siguiente manera:

- WPAN (Wireless Personal Area Network). - El alcance de esta red va a ser corto, utilizado frecuentemente en domótica. Ejemplos de este tipo de red tenemos una red ZigBee o Bluetooth.

- WLAN (Wireless Local Area Network). - Las redes de área local mediante comunicación inalámbrica tiene un aumento a la distancia con respecto a las WPAN, utilizada para interconectar dispositivos finales mediante un Access Point. Tecnologías como la 802.11b, 802.11a, 802.11g, 802.11n son utilizadas para WLAN.

- WMAN (Wireless Metropolitan Area Network). - Para las redes metropolitanas, al ser una red con demandas de velocidades de transmisión superiores se usa el protocolo IEEE 802.16 parecido al Wi-Fi con la diferencia que aumenta el ancho de banda y la cobertura. Ejemplo: WimaX

- WWAN (Wireless Wide Area Network). - Las redes de área extendida para las comunicaciones inalámbricas son usadas principalmente por la telefonía celular y satelital.

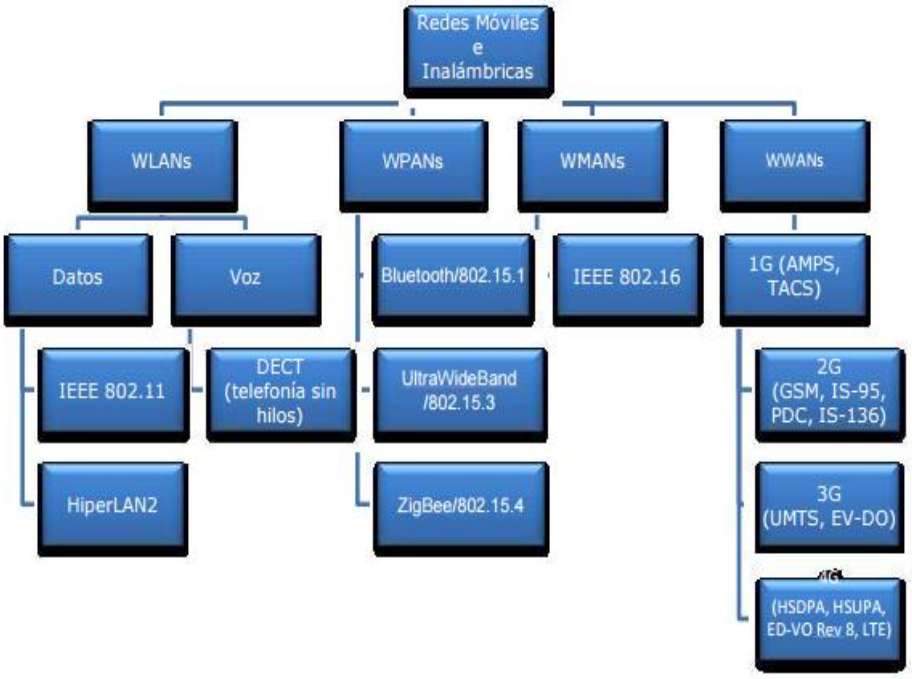

Figura 15. Clasificación de las redes móviles e Inalámbricas
Tomado de: Villapol (2010)

2.8. ZigBee

El protocolo Zigbee fue publicado en mayo del 2003 por la IEEE como complemento a redes Bluetooth y Wi-Fi, se necesitaba una tecnología inalámbrica que brinde confiabilidad con tasas de transmisión menor a Wi-Fi y Bluetooth, dado que la técnica de modulación es la Direct Sequence Spread Spectrum la velocidad de transmisión para ZigBee es de 250 kb/s.

Tabla 5.

Comparativas entre Wi-Fi, Bluetooth y ZigBee

Prestaciones	ZigBee 802.15.4	Bluetooth 802.15.1	WiFi 802.11
Consumo de corriente (mA)	30	65-170	350
Capacidad de red (nodos)	65000	30	7
Vida útil de la batería (días)	> 365	7	1
Velocidad de Transmisión RF (Kbps)	250	1000-3000	54000
Potencia de transmisión (mW)	1-2	1-100	40-200
Frecuencia de radio (GHz)	0.868; 0.915; 2.4	2.4	2.4
Rango de trabajo (metros)	1-100	1-100	30-100

Tomado de: Valdés, Pérez, Arias (2013).

Las frecuencias en las que ZigBee opera pueden ser 3 que son:

- Banda de 868 MHz (Europa)
- Banda de 915 MHz (Norteamérica)
- Banda de 2.4 GHz (Resto del mundo)

2.9. Módulos Xbee

Los módulos XBee son dispositivos que son comercializados por Digi Internacional que se componen de un transmisor/receptor (dependiendo de cómo se los configure) de ZigBee y un microprocesador, que tiene como objetivo intercambiar información de forma inalámbrica con un rango de alcance máximo que depende de la versión Xbee a utilizar y con una latencia baja.

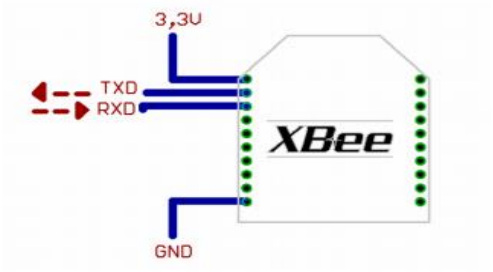

Figura 16. Pines Alimentación Xbee

Tomado de: Compututorials (2011)

Existen dos tipos de módulos Xbee que son la serie 1 y serie 2, la principal diferencia entre estos tipos de módulos es su compatibilidad, no es posible tratar de enviar datos de un módulo Xbee serie 1 a un Xbee serie 2 debido a que ambos usan un distinto microcontrolador y trabajan con distintos protocolos.

La Serie 1 de Xbee trabaja con el protocolo 802.15.4 y un micro controlador Freescale con lo que es posible ser usado en redes punto a punto y también en redes punto – multipunto.

La Serie 2 de Xbee es usada en redes MESH, su principio de funcionamiento consiste en tener varios nodos dentro de la red, lo cual al fallar un nodo el funcionamiento de la red no será afectado. El protocolo usado es el 802.11 y un micro controlador Ember, ideal para aplicaciones que usan redes Mesh y repetidores. Ambas versiones de Xbee son capaces de funcionar a temperaturas de -40°C a 85°C.

a)

b)

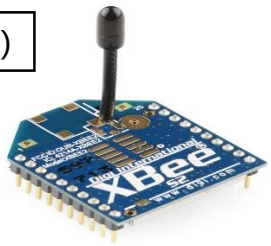

Figura 17. Módulos Xbee

a) Xbee Serie 1

b) Xbee Serie 2

Tomado de: Digi Internacional (s.f.)

Tabla 6.

Comparativa Módulos Xbee Serie 1 y Serie 2

	Serie 1	Serie 2
Alcance típico (interiores)	30 metros	40 metros
Mejor alcance (en línea recta)	100 metros	120 metros
Corriente de Transmisión/Recepción	45/50 mA	40/40 mA
Firmware	802.15.4 punto a punto	ZB Zig Bee mesh
Entradas/Salidas Digitales	8 I/O + 1 de solo entrada	11
Pines de entrada análogos	7	4
Pines análogos (PWM) de salida	2	Ninguno
Topologías punto a punto y estrella	Si	Si
Topologías Mesh y Cluster Tree	No	Si
Requiere nodo coordinador	No	Si
Configuración Punto a Punto	No	Más involucrado

Tomado de: Compututorials (2011)

2.10. Arduino Uno

Arduino UNO es una placa creada por la empresa del mismo nombre, dicha placa integra un microcontrolador Atmega 328, una entrada para fuentes de alimentación externa, la entrada USB para la programación del microcontrolador, también cuenta con 14 pines digitales y 6 pines analógicos.

El voltaje de entrada que se recomienda usar es de 7 a 12 V, y puede ser mediante fuentes de alimentación externas o el puerto USB de la placa.

2.10.1. Entradas y Salidas Digitales

Figura 18. Arduino UNO entradas /salidas digitales

Tomado de: Electronics Hacking (2016)

Los 14 pines digitales pueden ser utilizados como entradas o salidas. Es posible conectar diversos dispositivos para su control, como el caso de sensores digitales, cada pin digital de Arduino me proporciona un corriente de 40mA y una resistencia de 20 KΩ a 50 KΩ.

Además, algunos pines tienen funciones especializadas como:

- Pin 0 y 1. – El pin 0 se encarga de la recepción y el pin 1 para la transmisión de datos seriales
- Pines 2 y 3.- Pines para configurados para interrupciones externas.
- Pin 3,5,6,9,10,11.- Pines capaces de soportar modulación por ancho de pulsos, útiles para modificar voltajes de salida en un periodo de tiempo.
- Pin 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Pines para la comunicación SPI (Serial Peripherical Interface).
- Pin 13.- Pin de pruebas, está conectado un led que indica un valor alto o bajo.

2.10.2. Entradas / Salidas analógicas

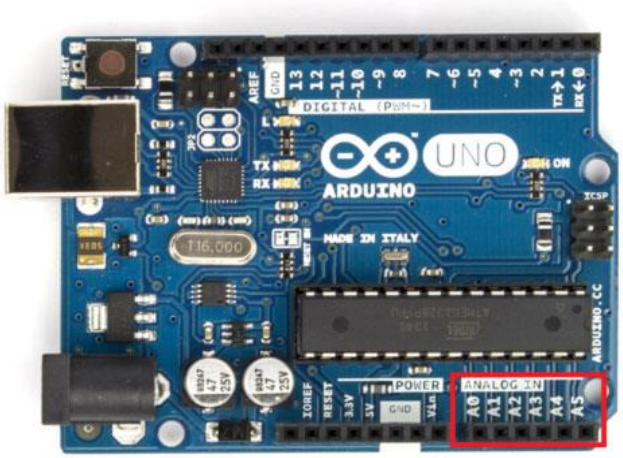

Figura 19. Arduino UNO Entradas/Salidas Analógicas

Tomado de: Fábrica Digital (s.f.)

Las entradas analógicas en Arduino son 6, nombradas A0, A1, A2, A3, A4, A5, cada pin ofrece 1024 estados o 10 bits. La tensión es de 5 voltios, pero es posible cambiar dicha tensión con la función analogReference ().

Ejemplo: Tensión de 5 V: analogReference (1023)

Tensión de 2.5v: analogReference (511)

Tabla 7.

Especificaciones técnicas Arduino Uno

Microcontrolador	Atmega328
Voltaje de operación	5V
Voltaje de entrada (Recomendado)	7 – 12V
Voltaje de entrada (Límite)	6 – 20V
Pines para entrada- salida digital.	14 (6 pueden usarse como salida de PWM)
Pines de entrada analógica.	6
Corriente continua por pin IO	40 mA
Corriente continua en el pin 3.3V	50 mA
Memoria Flash	32 KB (0,5 KB ocupados por el bootloader)
SRAM	2 KB
EEPROM	1 KB
Frecuencia de reloj	16 MHz

Tomado de: Gobierno de Canarias (2016)

2.11. Xctu

X-CTU es un programa para los sistemas operativos de Windows, Linux y Mac que brinda la posibilidad de configurar los dispositivos Xbee.

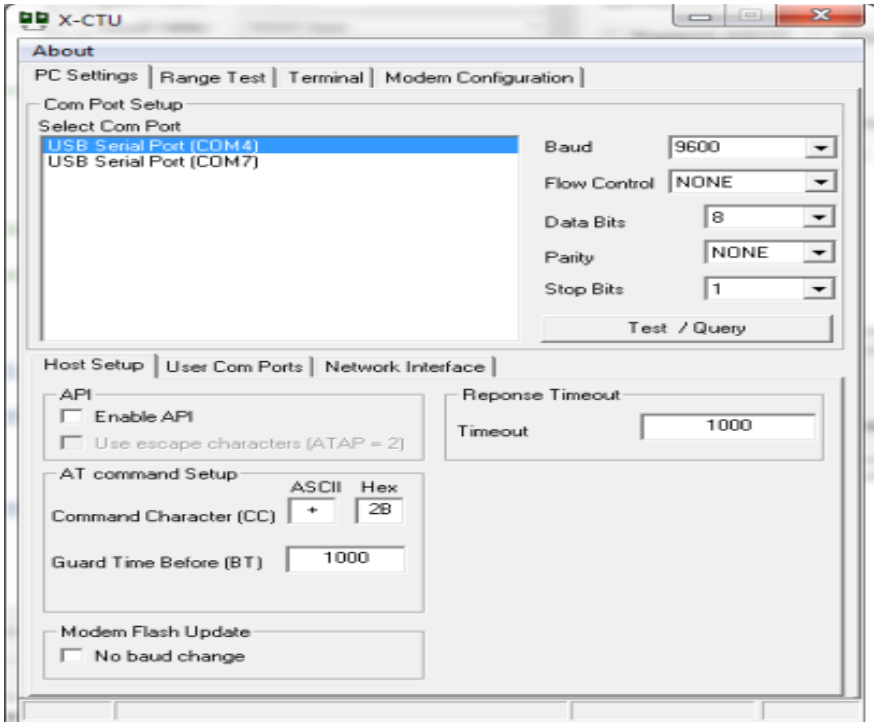

Figura 20. Pantalla Principal XCTU.

Las opciones del programa son:

Tabla 8.

Opciones software XCTU

PC Settings	Permite seleccionar el puerto y la configuración del mismo
Range Test	Permite desarrollar un test de rango entre dos dispositivos
Terminal	Permite acceso al puerto COM del computador con un programa de emulación terminal. También desde esta pantalla se puede acceder al firmware de los módulos mediante comandos AT. La lista de comandos AT se encuentra en el manual del producto
Modem Configuration	Permite programar los parámetros de los radios mediante una interfaz gráfica de usuario. Además permite cambiar la versión del firmware del radio

Tomado de: UTN. (s.f.)

2.11.1. Pc Settings

Esta opción presenta tres opciones de configuración que son:

2.11.1.1. Com Port Setup

El puerto COM se puede configurar los valores del módulo Xbee una vez que el dispositivo sea reconocido por la computadora, los datos con la posibilidad de configurar y modificar son:

- Velocidades de transmisión
- Control de flujo
- Bits enviados (4,5,6,7 hasta 8 bit de datos)
- Paridad
- Bits de Parada

La opción de Test /Query es capaz de comprobar si la configuración del módulo Xbee fue correcta como muestra la Figura 21

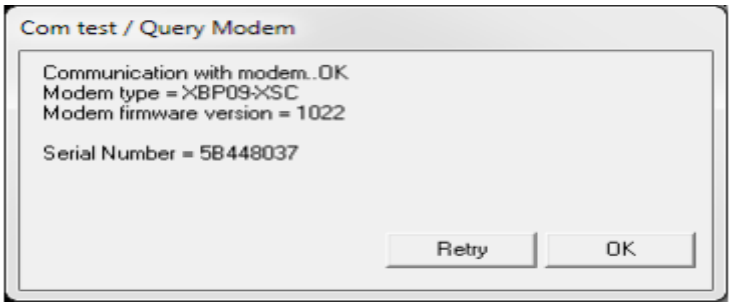

Figura 21. Mensaje exitoso de conexión al modem (Xbee)

2.11.1.2. Host Setup

Mediante esta opción se configura el firmware de los módulos Xbee, incluye el modo de comandos AT y el modo API

2.11.1.3. User COM Port

Permite agregar o eliminar puertos COM, dichos puertos son temporales ya que al cerrar el programa la configuración de los puertos desaparece.

2.11.2. Range Test

Mediante esta opción se comprueba el estado del enlace de radiofrecuencia,

mediante el envío y recepción de un paquete de datos en un tiempo determinado.

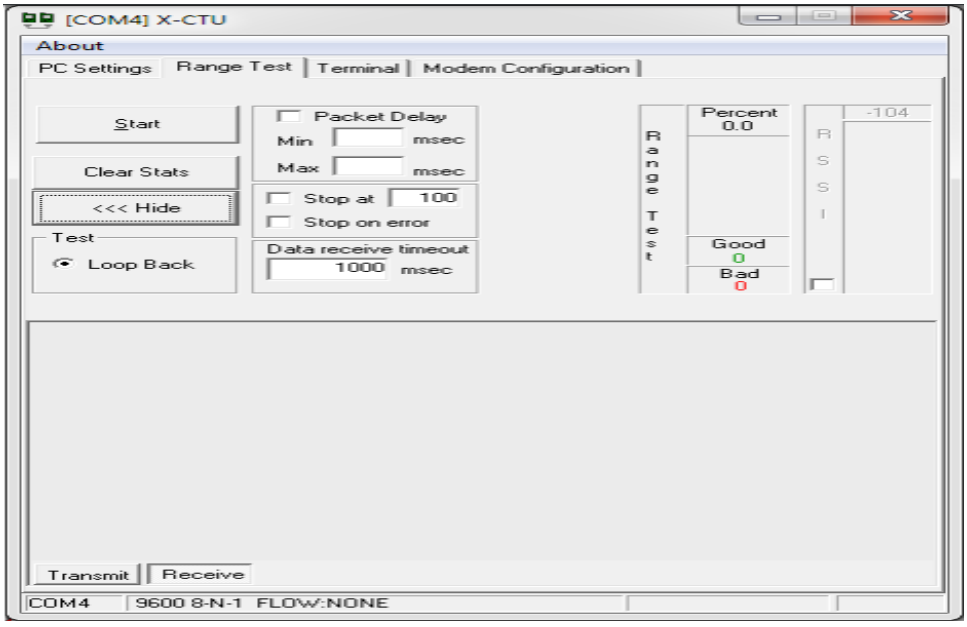

Figura 22. Pantalla XCTU Range Test

2.11.3. Terminal

Encontramos tres funciones básicas:

- Emulador de Terminal.
- Assemble Packet. - Envía y recibe datos programados por el usuario
- Show/Hide Hex.- Envía y recibe datos en formato ASCII y hexadecimal.

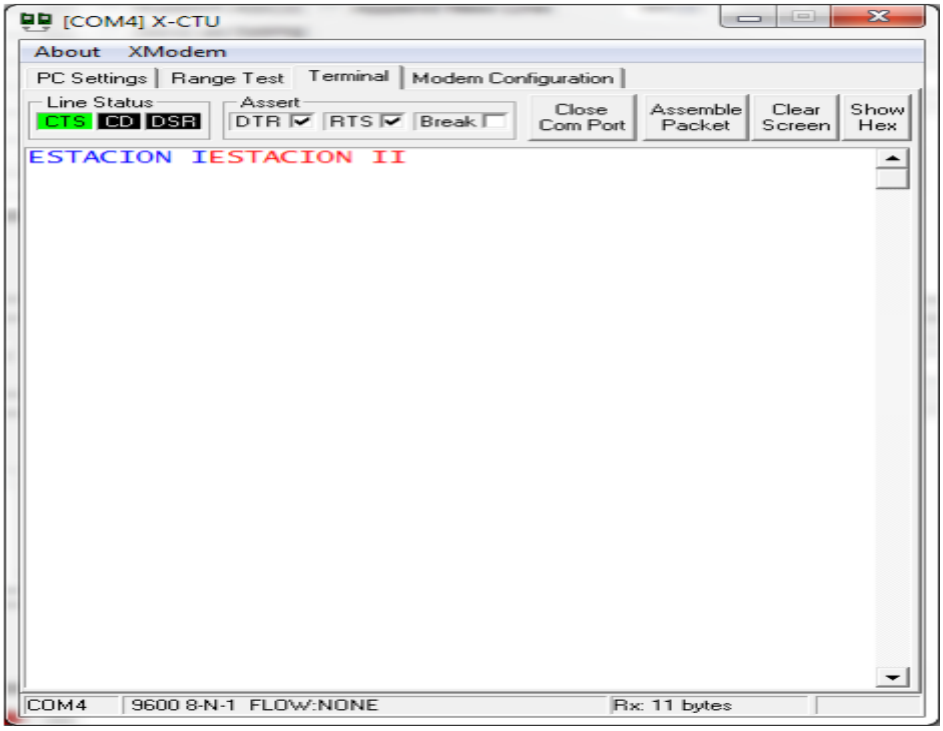

Figura 23. Pantalla Software XCTU – Opción Terminal

2.11.4. Modem Configuration

Sus principales funciones son:

- Proporcionar una interfaz gráfica al usuario para la configuración de los módulos Xbee
- Modificar y visualizar la información del microcontrolador instalado en el módulo conectado.
- Almacenamiento de perfiles de configuración de los módulos.

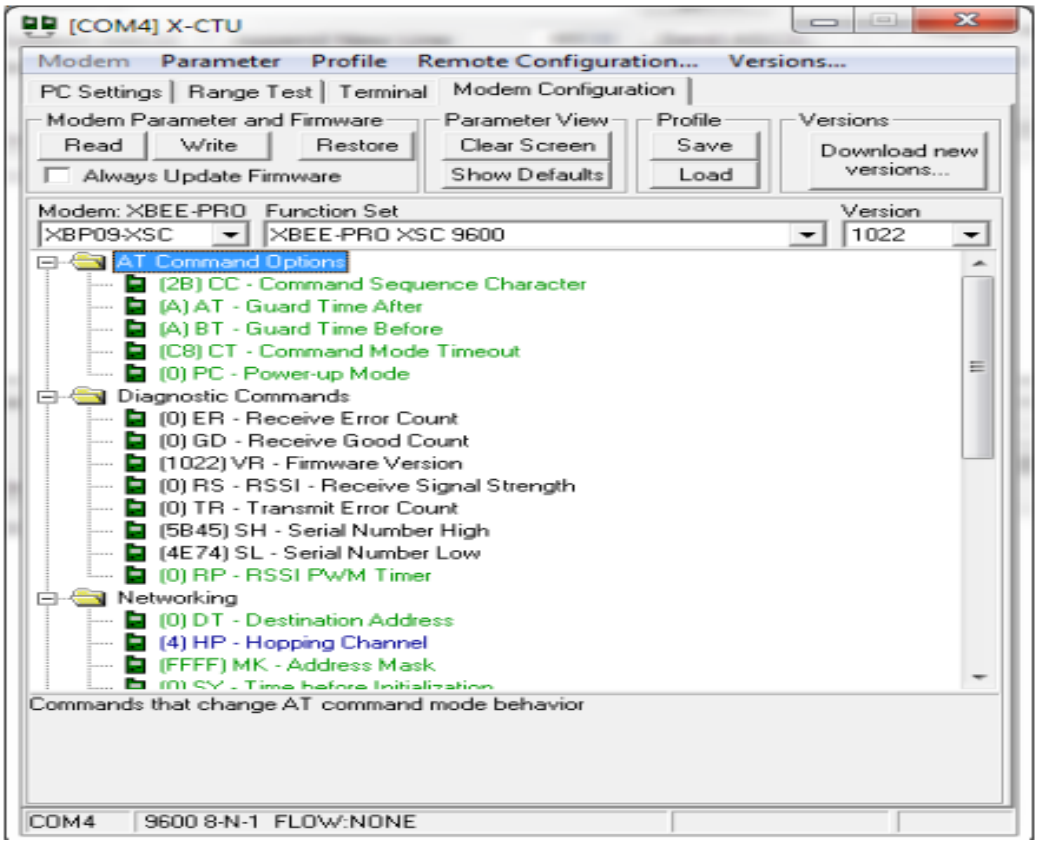

Figura 24. Ventana con opciones de configuración del firmware

2.12. Arduino IDE

Arduino, al ser una placa programable, cuenta con un software con el mismo nombre, con la finalidad de crear programas dependiendo de las necesidades de un proyecto.

La compatibilidad de este software no se limita a Windows únicamente, también puede ser instalado en computadoras con el sistema operativo de Linux y Mac.

La programación dentro del IDE puede ser sencilla de usar ya que se basa en C++, y una vez compilado el programa, deberá ser cargado a la placa a través del puerto USB instalado en la misma.

Figura 25. Pantalla principal de Arduino

2.12.1. Entorno de programación y configuración

La pantalla principal de IDE cuenta con la siguiente interfaz, como se muestra en la Figura 26.

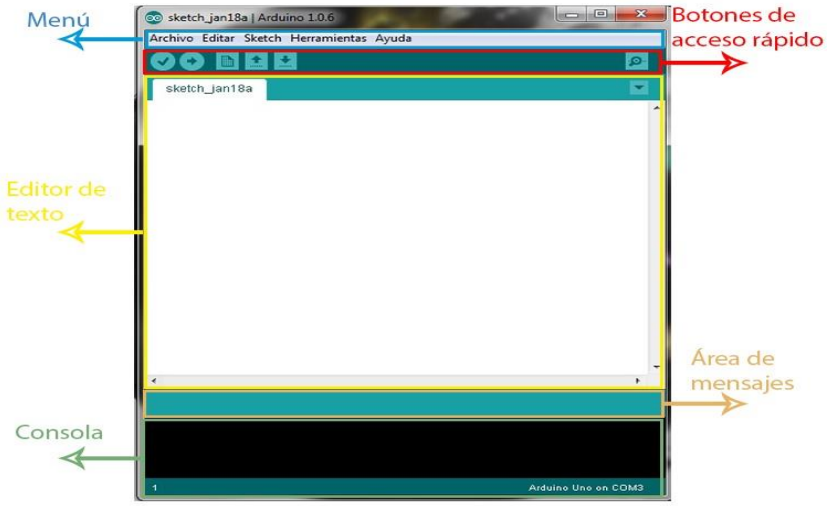

Figura 26. Zonas del IDE Arduino

Tomado de: Martínez (2015)

En la sección de botones de acceso rápido consta con los siguientes iconos:

-  Compila el programa, verificando errores de programación.
-  Carga el programa hacia la placa de Arduino.
-  Nuevo programa.
-  Abrir programa.
-  Guardar programa
-  Visualiza el monitor Serial, se establece comunicación con la placa de Arduino, en las que se puede observar las instrucciones programadas, siempre que tengamos el USB conectado.

En la sección denominada Editor de texto, se tienen que escribir las líneas de código que serán interpretados por la placa Arduino.

El área de consola y mensajes se muestra la información de los eventos realizados en el área de Editor de texto y de botones de acceso rápido, permitiendo obtener información como, por ejemplo, la compilación exitosa o fallida del código, detallando el estado actual del evento a realizar.

La comunicación entre el IDE y la placa de Arduino es fundamental para la carga de los programas hacia la placa, para ello, y al tener Arduino variedad de modelos, se debe escoger con cuál de todos vamos a trabajar, la ruta es zona de menú > Herramientas > Tarjeta como lo muestra la Figura 27.

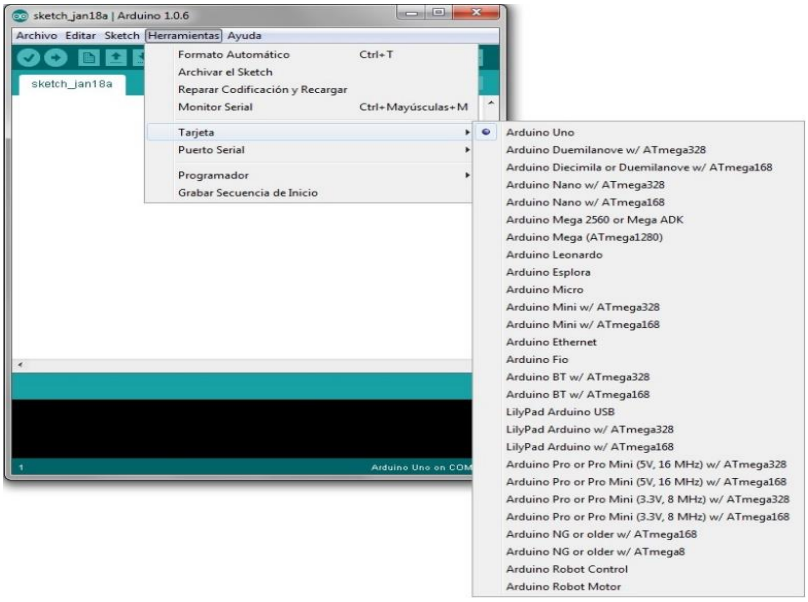

Figura 27. Selección del tipo de tarjeta Arduino

Tomado de: Martínez (2015).

Adicional es necesario elegir el puerto COM de la computadora en donde se conectó la placa de Arduino, está ubicada en la ruta zona de menú > Herramientas > Puerto Serial

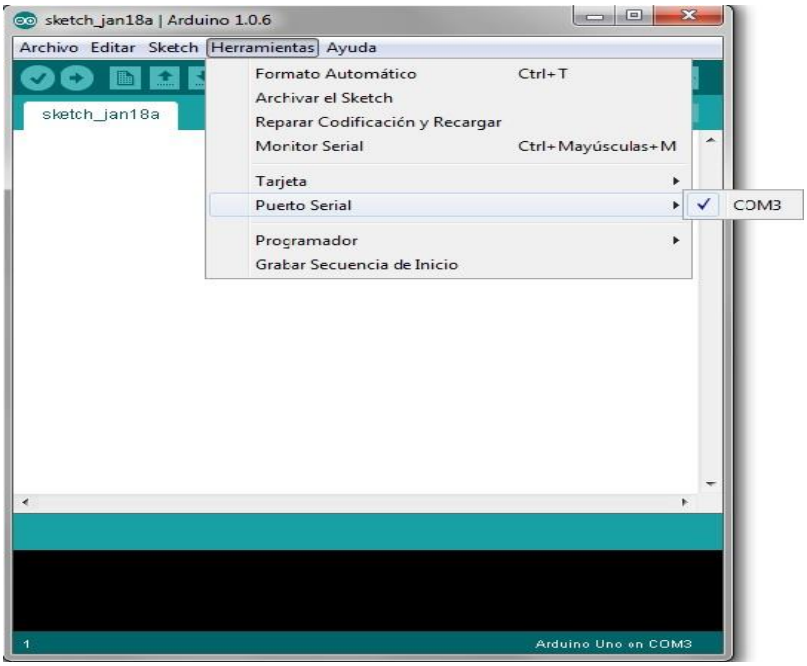

Figura 28. Selección del puerto conectado

Tomado de: Martínez (2015).

2.12.2. Lenguaje de Programación Arduino

Arduino se programa con un lenguaje de programación de medio nivel propio que es similar a C++.

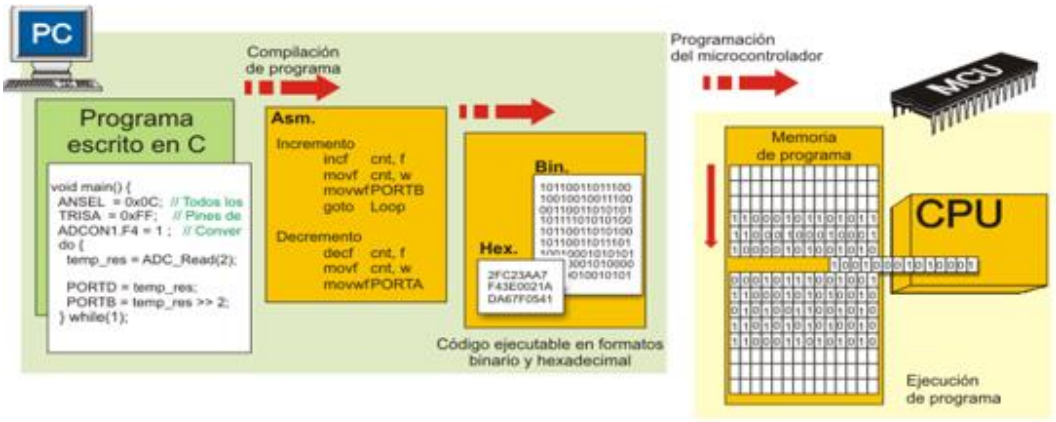

Figura 29. Proceso de Programación y Compilación de un microcontrolador
Tomado de: Crespo (2017)

La estructura básica de un programa realizado en IDE consta de dos partes importantes que son:

- Setup (). - Declaración de variables a usar en todo el programa
- Loop (). - Dentro de la función de loop se escriben todas las instrucciones que ejecutará el programa.

2.12.2.1. Uso de Variables

Las variables son el tipo de dato que se asignan a un valor escogido (String, float, char), deben ser declaradas en Setup (), pero también pueden ser declaradas en Loop (), la diferencia radica que una variable declarada en Setup () es una variable global, mientras que al ser declaradas en loop () son variables locales.

2.12.2.2. Tipos de Datos

Los tipos de datos compatibles con IDE son:

- Array. - Almacena objetos de un mismo tipo.
- Byte. - Variable asignable para valores numéricos de 8 bits.
- Int. - Variable que se utiliza para almacenar valores enteros positivos y negativos de 16 bits.
- Long. - Variable que se utiliza para almacenar valores enteros positivos y negativos de 32 bits.
- Float. - Almacena números decimales de 32 bits.

Tabla 9.

Rangos de tipos de datos

Tipo de Dato	Tamaño	Rango
Byte	8 bits	0 al 255
Int	16 bits	32767 al -32.768
Long	32 bits	-2147483648 a 2147483647.
Float	32 bits	3.4028235E +38 a +38-3.4028235E.

2.12.2.3. Operadores condicionales

Los operadores condicionales tienen como objetivo que para a una variable o constante se le debe asignar su comportamiento bajo cierta condición que usualmente son:

- If
- If – else
- For
- While
- Do- while

2.12.2.4. Entradas y salidas analógicas y digitales

Mediante la programación de las entradas y salidas digitales / analógicas asignamos determinada instrucción a un pin de Arduino. Dentro de las usadas son:

- pinMode (pin, mode). - Configuración que usualmente creada en modo global (Setup ()) para asignar a determinado pin la opción de entrada o salida.
- digitalRead(pin). - Lectura de un valor digital, que retorna un valor HIGH o LOW
- digitalWrite (pin, value). - Determina a un pin la opción de niveles HIGH y LOW.
- analogRead(pin). - Lectura de valores analógicos de 10 bits.
- analogWrite (pin, value). - Escritura de un valor analógico.

2.12.2.5. Funciones de tiempo

Las funciones de tiempo son programadas bajo variables con unidades de tiempo las que son:

- Milis (). - Variable que toma valores de long unsigned, capaz de medir los microsegundos que Arduino se encuentra funcionando.
- Delay (). - Tiempo asignable en microsegundos donde programa puede permanecer en pausa.

2.12.2.6. Puertos Serie

- Serial.begin(). - Se realiza mediante código la apertura del puerto serial y la velocidad de transmisión.
- Serial.print(). - Visualiza los datos por consola.
- Serial.println(). - Se visualiza los datos por consola y con un salto de línea.
- Serial.read(). - Instrucción de lectura de datos en el puerto serie.

2.13. Processing

Processing es un software libre y de código abierto creado por Ben Fry y Casey Reas en el año 2001, su lenguaje de programación lleva por nombre “Proccesing” que es basado en el lenguaje JAVA pudiendo usar todas sus funciones para proyectos enfocados principalmente a entornos gráficos.

El software es compatible con: Windows, Linux y Mac OS X que consta de un editor de texto para la escritura del código, los programas realizados en Processing son denominados *“sketch”*. También cuenta con la consola para la observación de errores de programación, el área de mensajes, el botón de ejecución y el de stop y el área para la barra del menú.

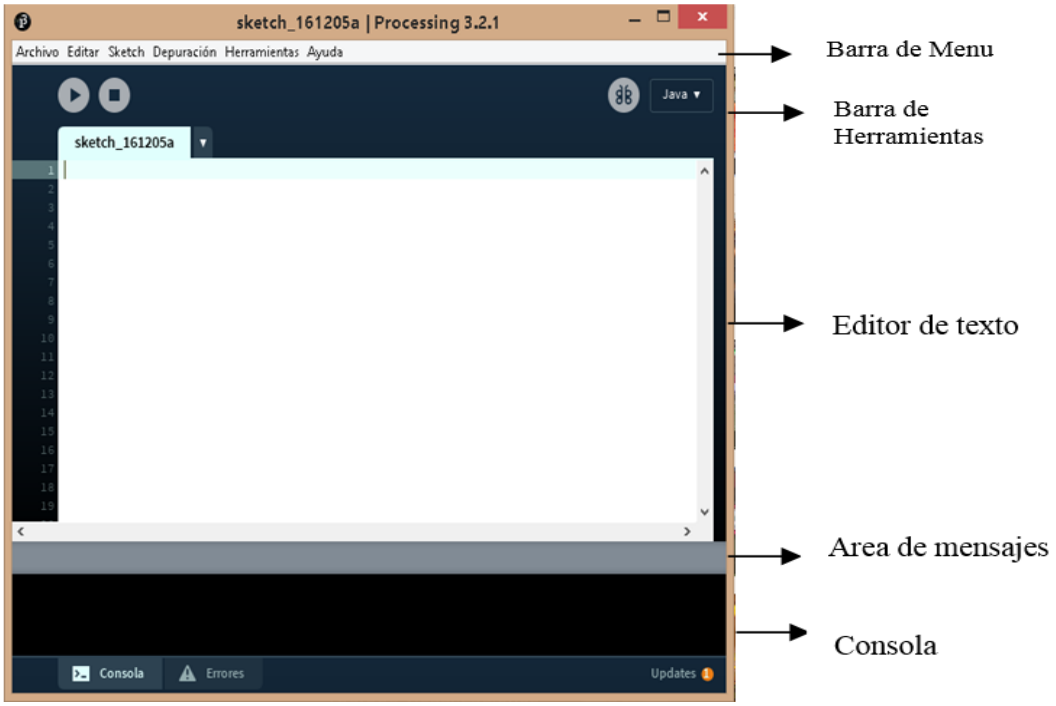

Figura 30. Processing: Pantalla principal

En la figura 30 se aprecia la interfaz donde se escribirá el código de programación que consta de un editor de texto, y la barra de herramientas para compilar y ejecutar los “sketches”.

Las aplicaciones que crea Processing son:

- Standalone. - Se ejecutan como aplicaciones propias de un Sistema Operativo.

- Applets. - Mediante la creación de applets podremos ejecutar la aplicación en una página web.

Processing al ser un lenguaje de programación basado en Java puede juntar sus características de programación con las de Java y usar las librerías exclusivas o clases para desarrollar aplicaciones más complejas, también se pueden realizar aplicaciones para dispositivos móviles.

2.13.1. Entorno de Programación Proccesing

Es nombrado como PDE (Processing Development Enviroment), es basado en Java y permite que la escritura del código sea intuitiva. Sus principales opciones que se pueden visualizar son:

 Da inicio a la aplicación

 Finaliza la aplicación

Editar. - Opción para modificar texto.

Export. – Exporta la aplicación a un ejecutable. Se puede escoger para cual Sistema Operativo se desea exportar.

Sketch. - Incluye librerías al proyecto.

2.13.2. Lenguaje de Programación Processing

Al ser enfocado para el uso de diseñadores, cuenta con características para el uso a nivel visual, con la posibilidad de programar figuras interactivas en 2D, 3D, figuras geométricas, entre otras. Las funciones más importantes a usarse son:

- size (x, y). - Crea una ventana con las dimensiones (en pixeles) especificadas por el programador.
- background. - Agrega un color de fondo al sketch.

- point (x, y). - Dibuja un punto en las coordenadas especificadas de X, Y.
- text (x, y). - Escribe una cadena de texto.
- line (x1, y1, x2, y2). - Dibuja una línea donde los primeros dos valores (x1, y1) es el inicio del trazo y (x2, y2) es el final.
- triangle (x1, y1, x2, y2, x3, y3). - Para función triangle se debe especificar 3 valores para el trazado del triángulo que será dibujado con una figura de relleno.
- quad (x1, y1, x2, y2, x3, y3, x4, y4). - Función para el dibujo de cuadrados, rectángulos, figuras geométricas de 4 lados.
- ellipse (x, y, width, high). - Dibuja una elipse en la cual los valores de X Y se especifica el centro de la elipse, el valor width es el ancho y high el alto de la elipse.
- bezier (x1, y1, cx1, cy1, cx2, cy2, x2, y2). - Función con cuatro pares de parámetros que dibuja líneas curvas.
- fill (valor numérico). - Agrega un relleno a las figuras geométricas.
- stoke (valor numérico). - Dibuja un contorno en la figura.
- stokeWeight (valor númerico). - Especifica el grosor de una línea.
- beginShape (). - Inicio de una figura a realizar con vértices.
- endShape (). - Punto final de la figura a realizarse con vértices
- vertex (x, y). - Agrega un vértice.
- Rotate (ángulo en radianes). - Gira la figura a determinado ángulo.

2.14. Proteus

Software creado por la compañía Labcenter Electronics Ltd, que cuenta con dos aplicaciones denominadas Ares e Isis para el diseño, implementación y simulación de circuitos electrónicos digitales y analógicos.

2.14.1. Isis

Isis es la aplicación para ejecutar gran cantidad simulaciones, tiene una amplia gama de componentes electrónicos en su librería que consta con más de 35000 componentes (PIC,s , leds , diodos , amplificadores , etc.) y también

cuenta con librerías de fabricantes como Philips , National Semiconductor , para tener valores estimados durante la simulación.

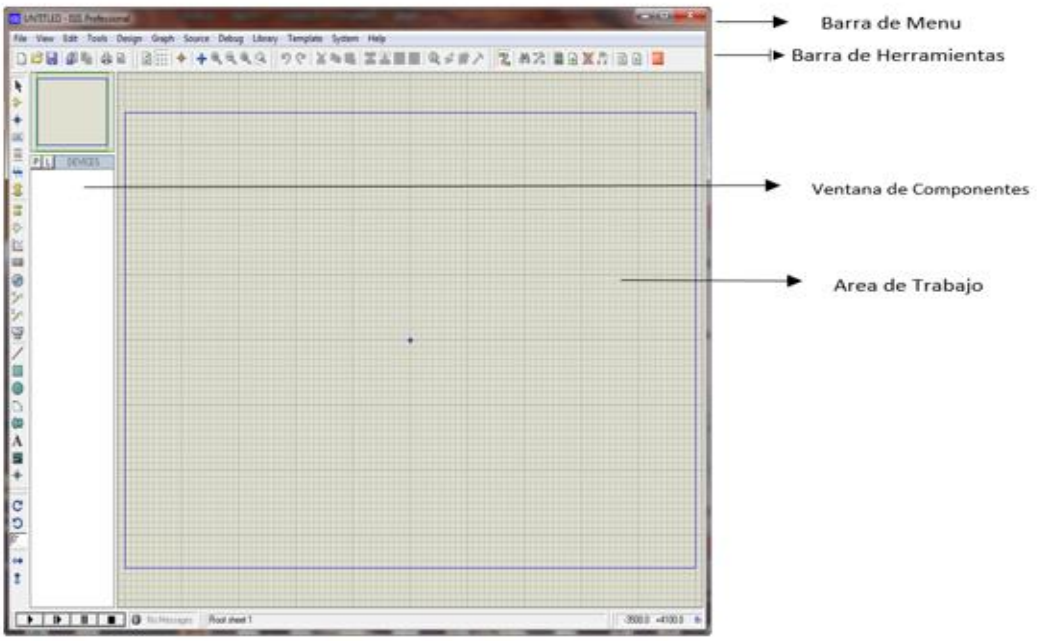

Figura 31. ISIS, pantalla principal

Una vez en el software para elegir elementos electrónicos en la zona de ventana de componentes se da clic el icono  .

Figura 32. Ventana de Componentes de ISIS

Los elementos seleccionados se los debe arrastrar hasta la zona de trabajo y unir los terminales a los cuales deberán conectarse.

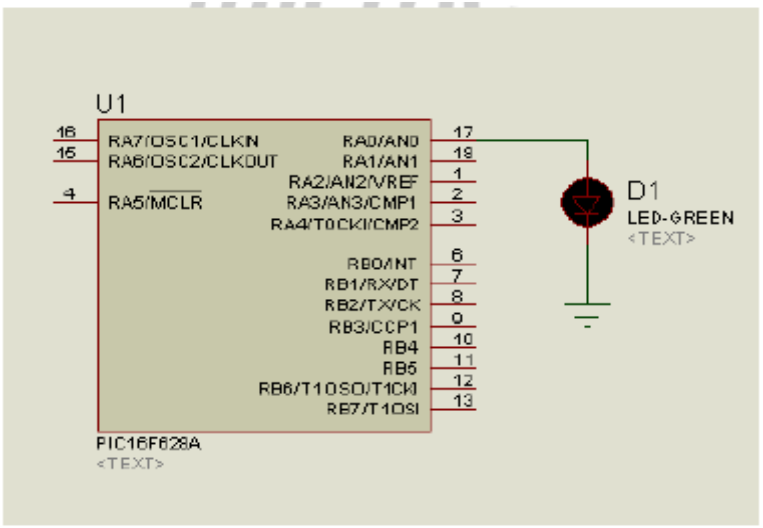

Figura 33. Modelo de conexión en ISIS

En el caso de simular microcontroladores, el programa para ser cargado en el PIC deberá compilarse en Assembly y obtener un archivo. hex (Ej: simulacion.hex) y en la aplicación de Isis dar doble clic en el microcontrolador que presentará la siguiente pantalla:

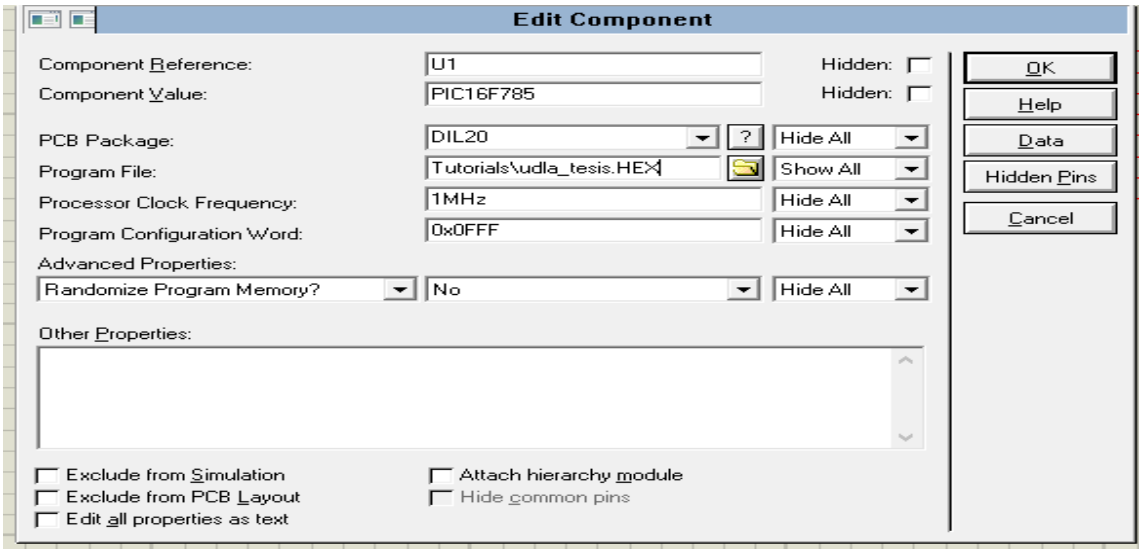

Figura 34. Configuración de Componentes

En la opción Program File se debe buscar la ruta donde se guardó el archivo .hex.

2.14.2. Ares

Area Professional Layout Design es el software que realiza el esquema del circuito para este ser impreso en la placa (baquelita), puede generar automáticamente el posicionamiento de los elementos en la placa con las rutas de conexión.

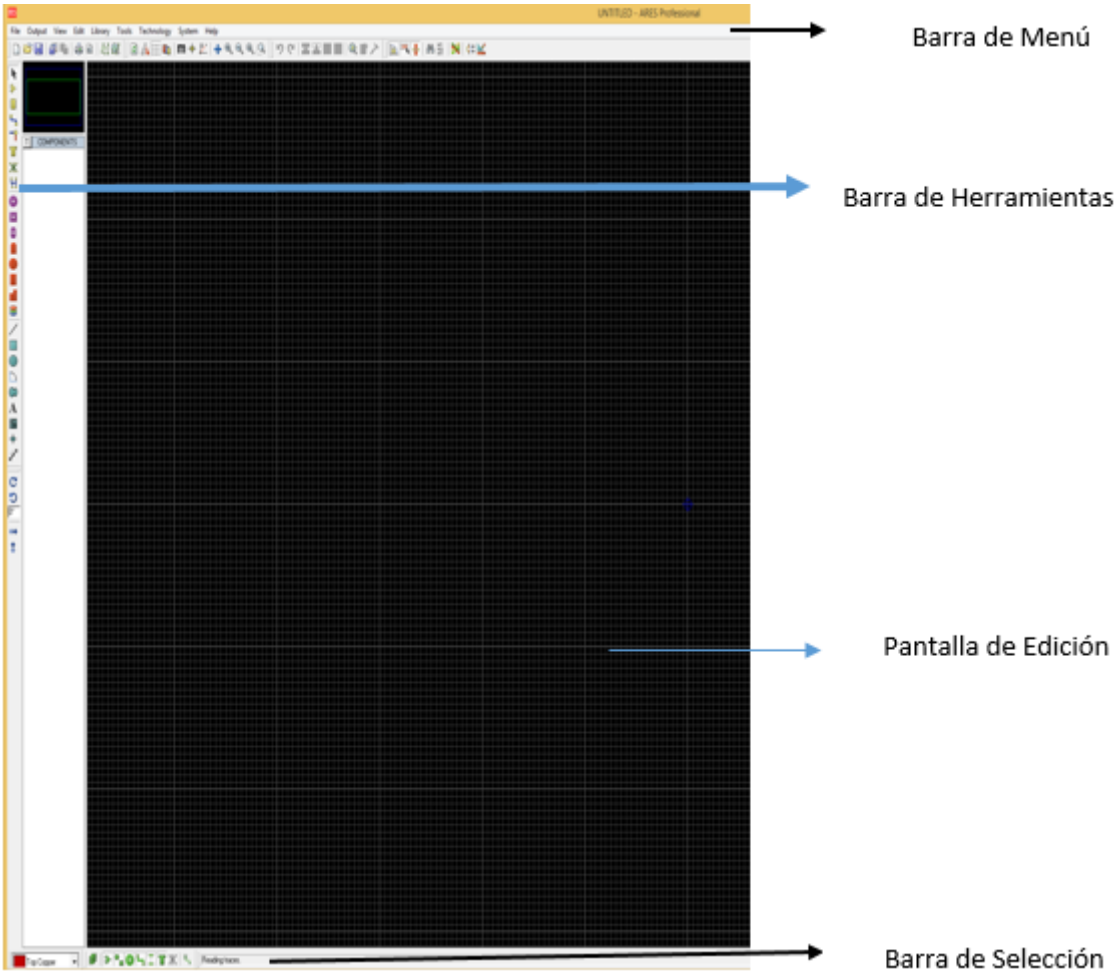

Figura 35. Pantalla principal de Ares

La interfaz gráfica del software consta de las opciones vistas en la Figura 35

Pantalla de Edición. - Lugar en donde se realiza el esquema de circuito para la creación de las pistas.

Barra de Estado. - Refleja la ubicación donde se encuentra el cursor, y el estado de compilación del circuito.

Barra de Selección. - Permite seleccionar los elementos que vamos a usar y deberán ser ubicados en la pantalla de edición

Barra de Herramientas. - Tenemos las opciones de la librería de los elementos para ubicarlos en la pantalla de edición, creación del perímetro del circuito, escribir leyendas, etc.

Creación de las rutas automáticamente.

Para la creación automática de las rutas se debe ubicar en la pantalla de edición todos los elementos y conectarlos tal como en la Figura 33.

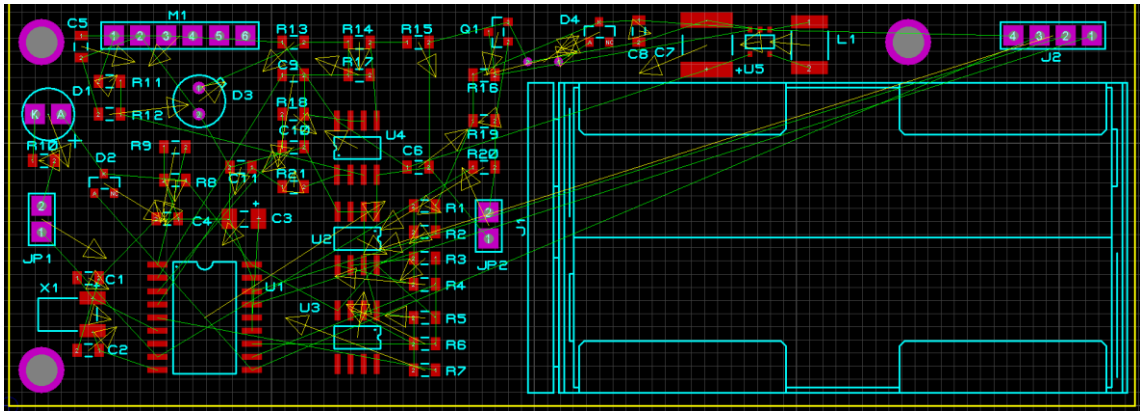

Figura 36. Circuito sin proceso de ruteo automático

Una vez conectados los elementos, para la creación de las rutas se debe pulsar en la barra del menú el siguiente icono  que desplegará las opciones de las rutas, como el ancho de las pistas, si el circuito impreso es de una o más capas, etc.

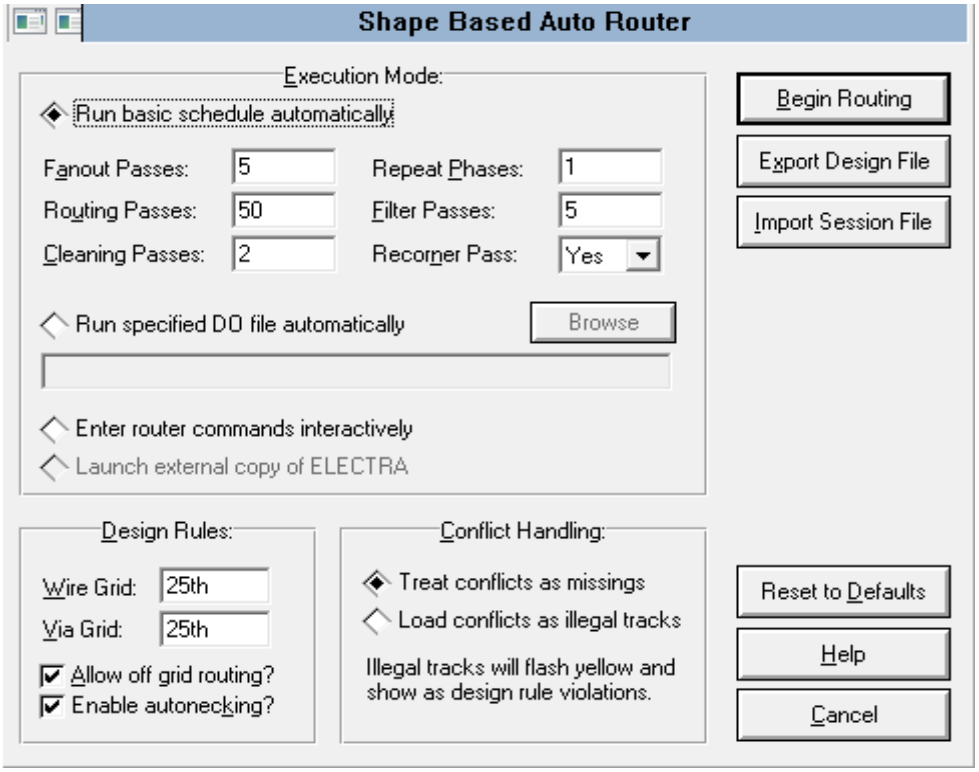

Figura 37. Opciones de Ruteo en Ares

Si el proceso de ruteo de las conexiones esta correcto en la barra de estado deberá mostrarse un mensaje indicando que todas las rutas han sido completadas.

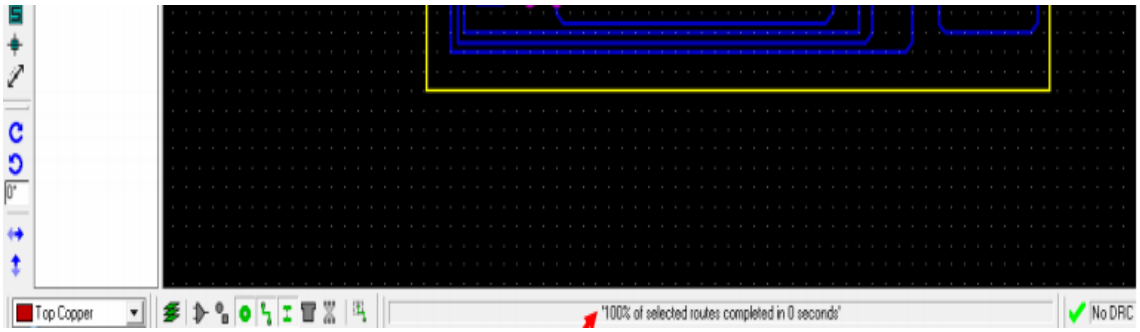

Figura 38. Proceso de ruteo completado

2.15. phpMyAdmin

Es un servidor de bases de datos desarrollados en PHP para controlar una o varias bases de datos vía Internet.

Algunas de las funcionalidades de phpMyAdmin son:

- Crear y modificar bases de datos
- Realizar sentencias SQL (Structure Query Language) para crear, eliminar y alterar registros.
- Exportar e importar bases de datos.
- Reparar y optimizar tablas, bases de datos, campos
- GUI intuitiva

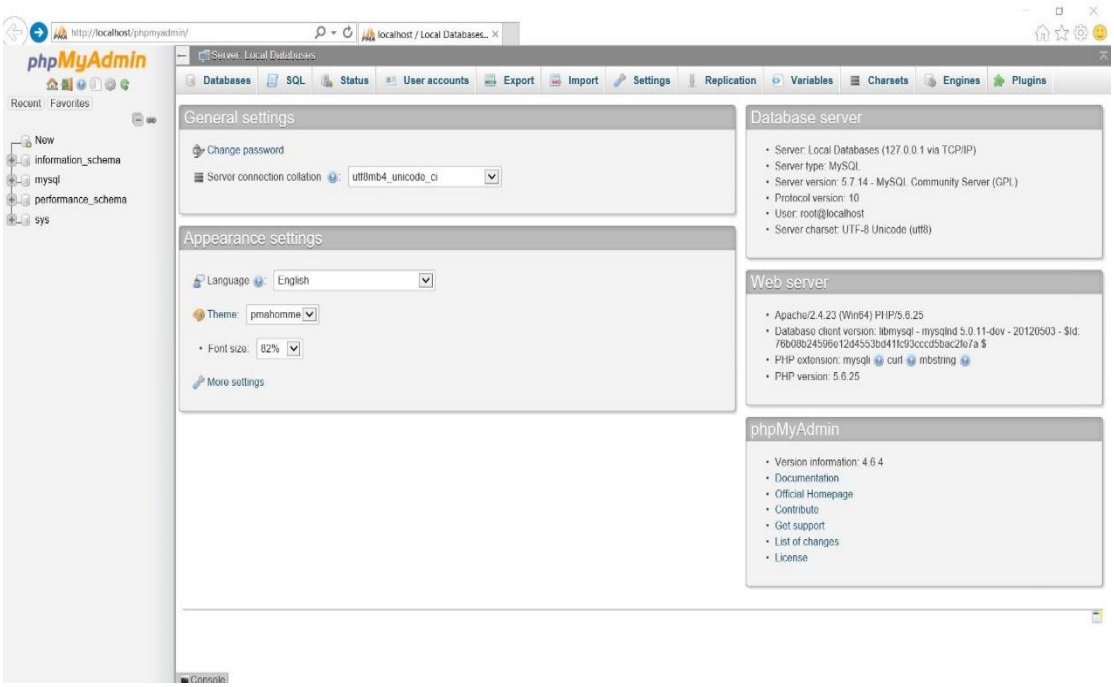

Figura 39. Pantalla de inicio de phpMyAdmin

Una vez en la página de inicio en la sección izquierda se muestran las bases de datos creadas, los otros partes de la página de inicio son:

Configuraciones Generales. - Cambio de contraseñas, conjunto de la conexión con el servidor.

Configuración de apariencia. - Selección de idioma, temas y tamaño de fuentes

Servidor de Bases de datos y Servidor Web. - Información detallada del servidor phpMyAdmin y del Servidor Web (IP, Versión, Tipo de Servidor, Conjunto de Caracteres del Servidor.

3. Capítulo III. Desarrollo e Implementación

Este capítulo tiene por objetivo detallar el paso a paso de la elaboración del prototipo, su proceso de funcionamiento está conformado por:

- Lectura de Sensores y Pulsadores
- Interpretación de datos en Arduino Uno
- Envío y recepción de datos mediante los módulos Xbee
- Interpretación de datos en Processing
- Envío de Notificaciones,logs y almacenamiento en la base de datos.

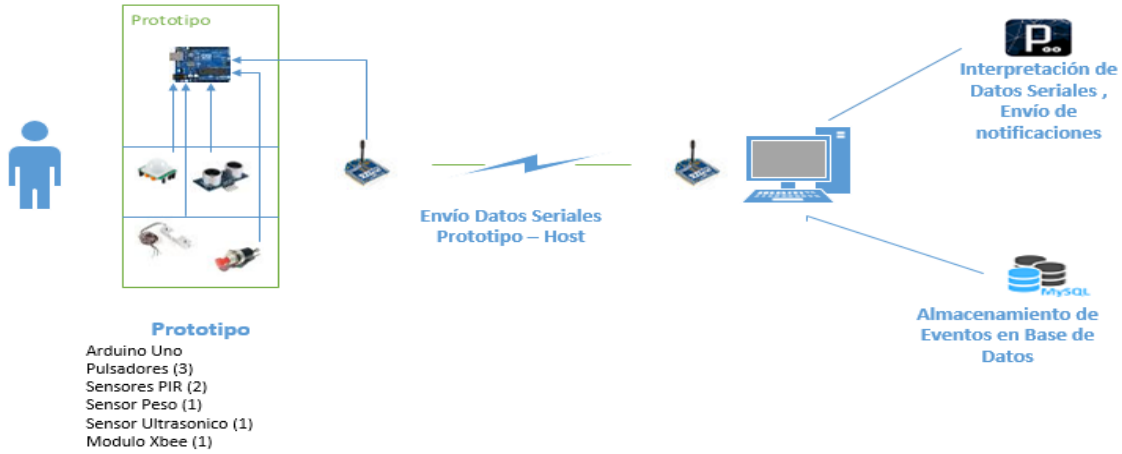

Figura 40. Diagrama de bloques de la implementación del proyecto

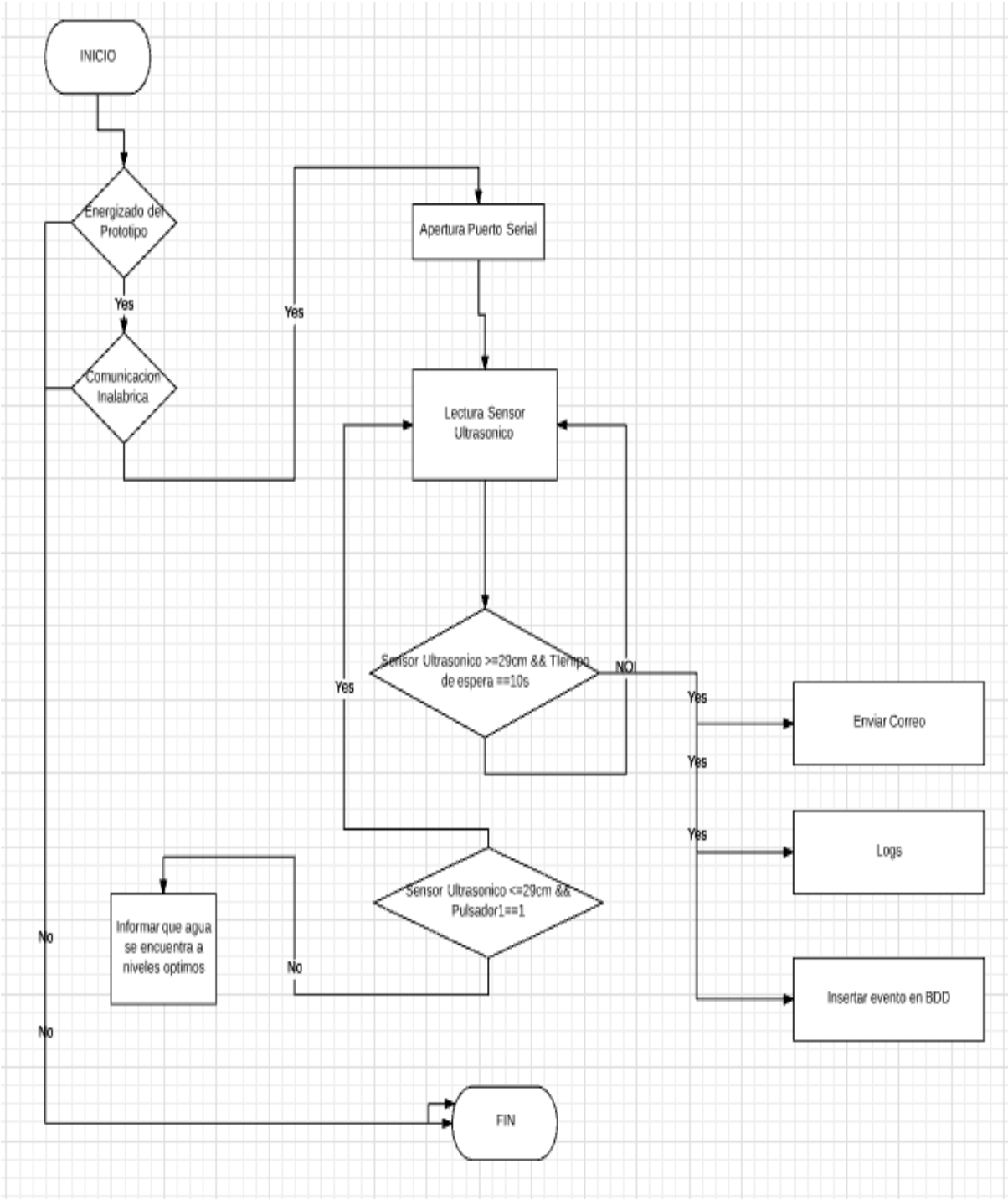

Figura 41. Diagrama de Flujo para el sensor ultrasónico

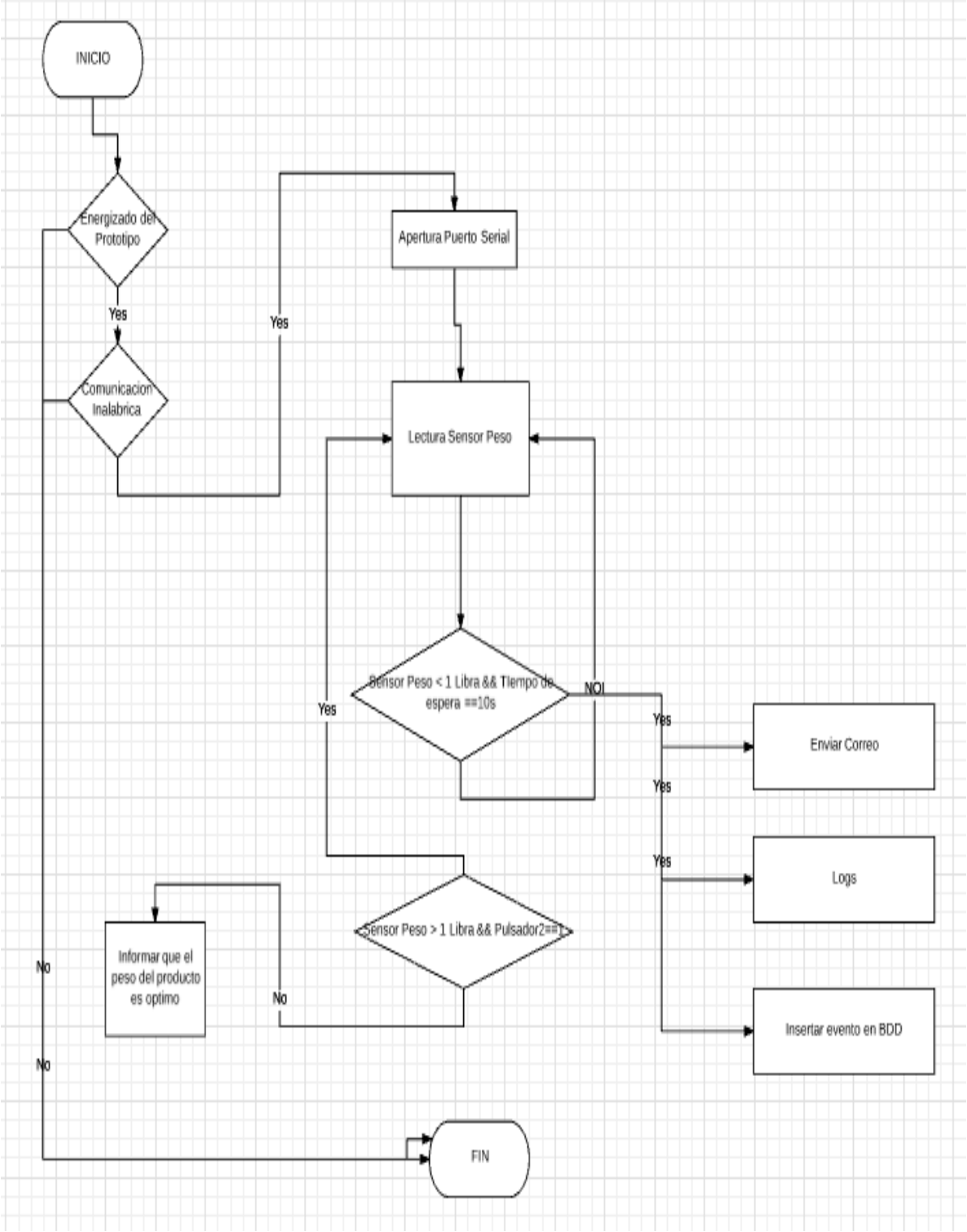

Figura 42. Diagrama de Flujo sensor Peso

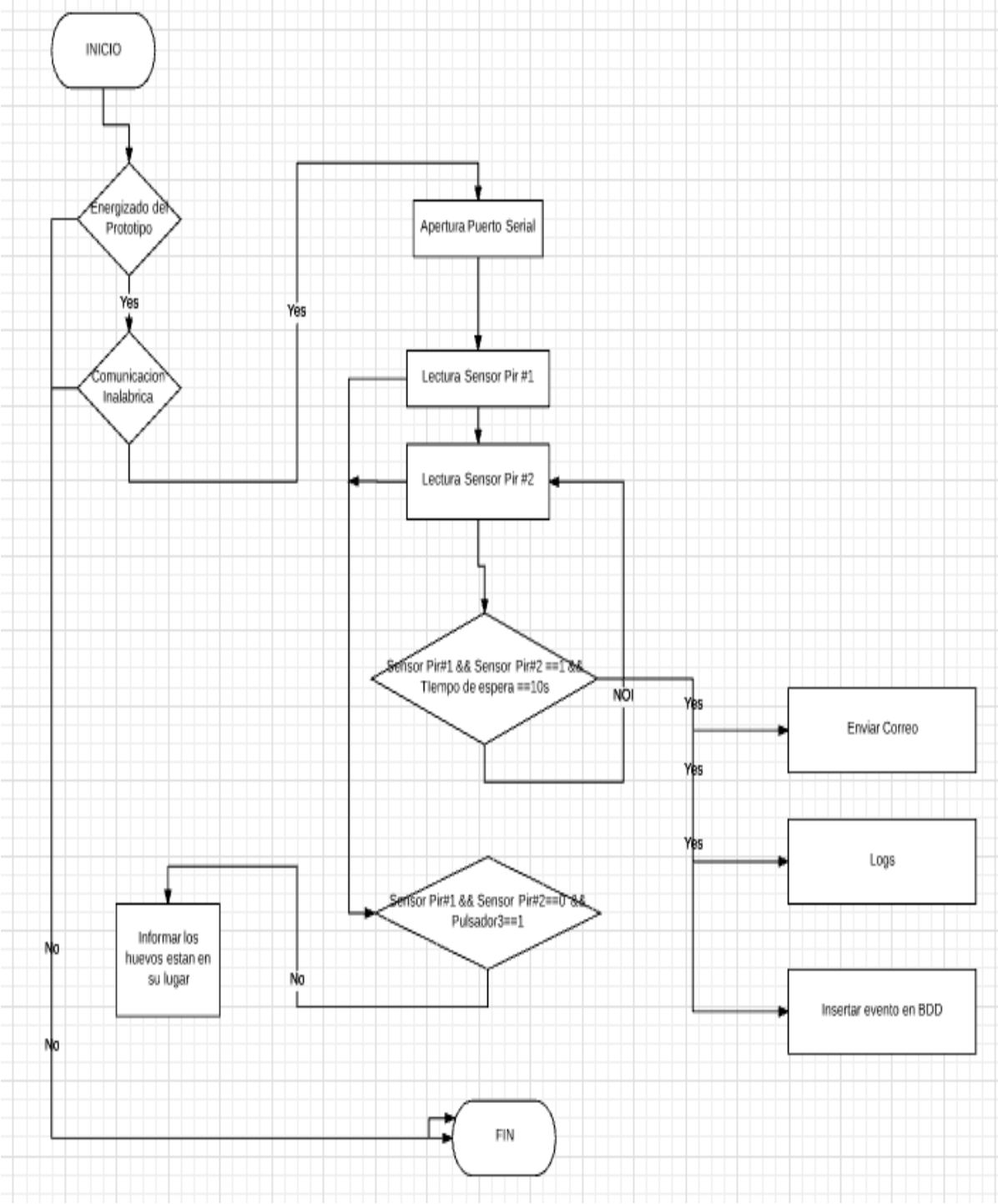

Figura 43. Diagrama de Flujo Sensores Infrarrojos (Presencia)

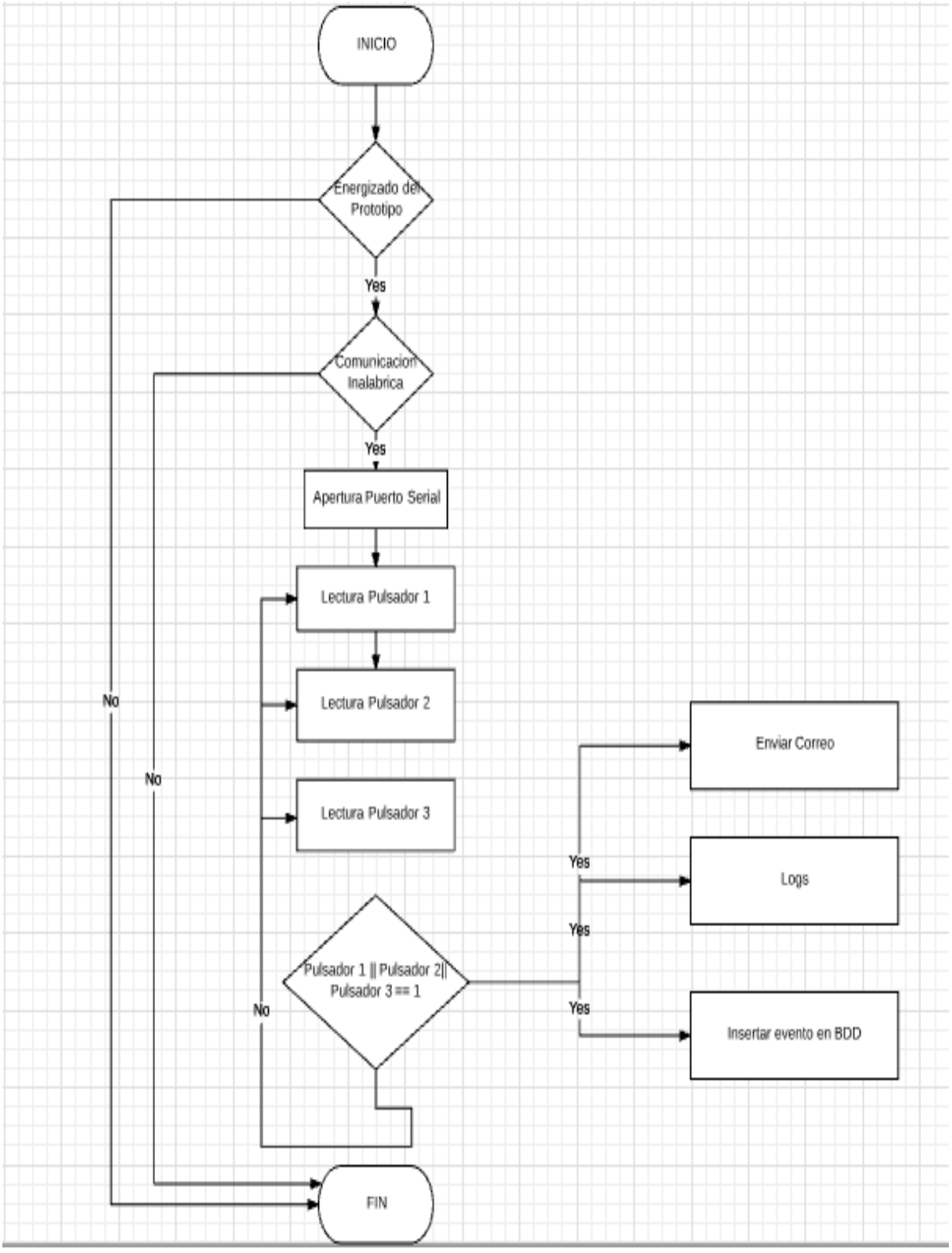

Figura 44. Diagrama de Flujo Pulsadores

3.1. Lectura de Sensores y Pulsadores

Los sensores instalados son:

3.1.1. Sensor de Peso

El sensor permitirá controlar el peso, con fines demostrativos, se colocó carne en su debido recipiente y con un peso inicial de 1 Kg (2,2 libras), al tener un peso menor a 1 libra y si el recipiente se encuentra fuera de su posición por un tiempo mayor a 10 segundos (tiempo de demostración) se enviará la notificación al proveedor.

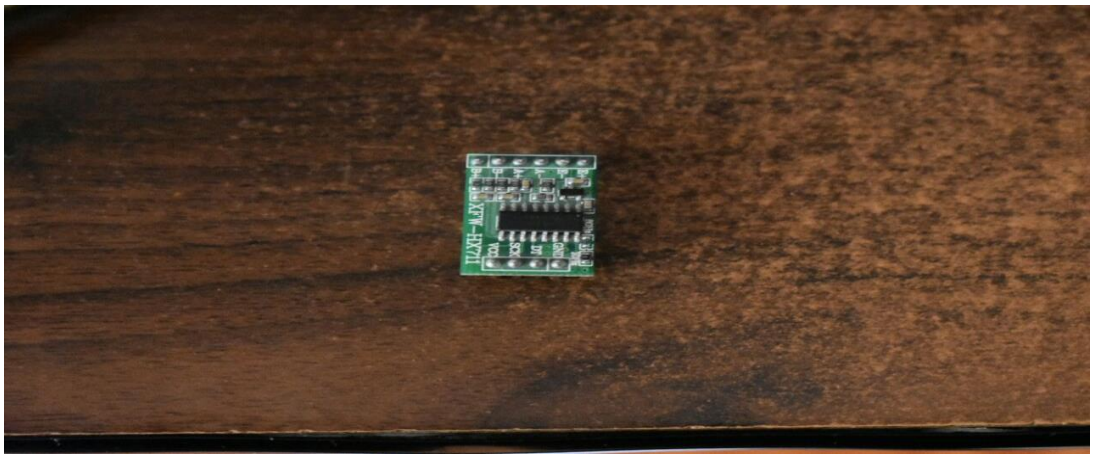

Figura 45. Módulo HX711

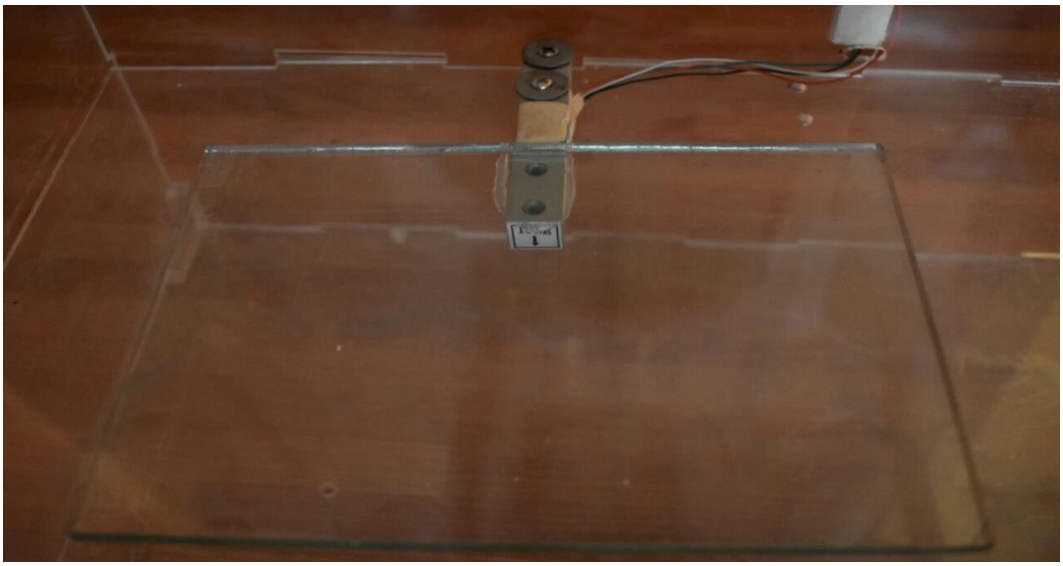

Figura 46. Celda de carga de 10 kg

3.1.2. Sensor Ultrasónico

Es el encargado de medir el nivel de agua o cualquier líquido en un recipiente, si el líquido está a una distancia mayor a 29 cm, será notificado.

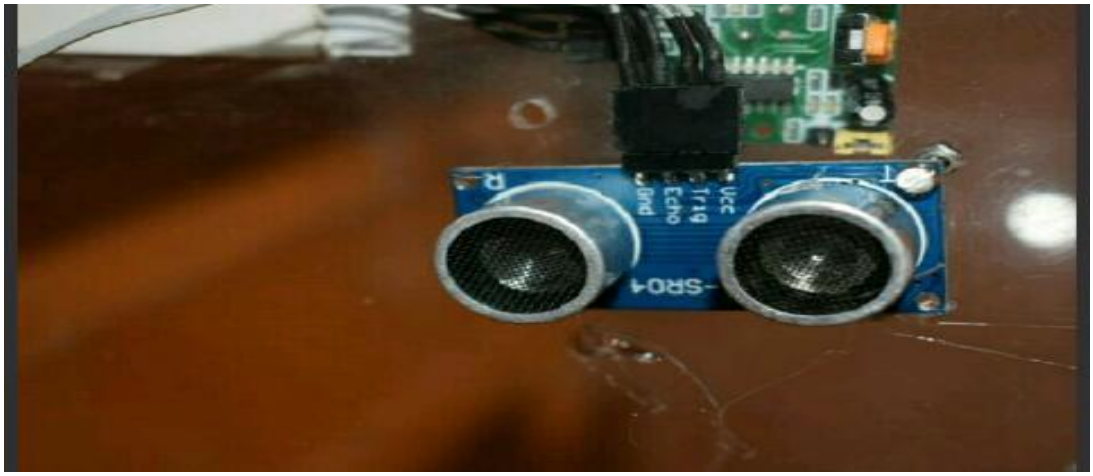

Figura 47. Sensor ultrasónico

3.1.3. Sensores de Presencia (PIR)

Están instalados dos (por motivos de demostración) y se encargan de ver la presencia de huevos en el prototipo.

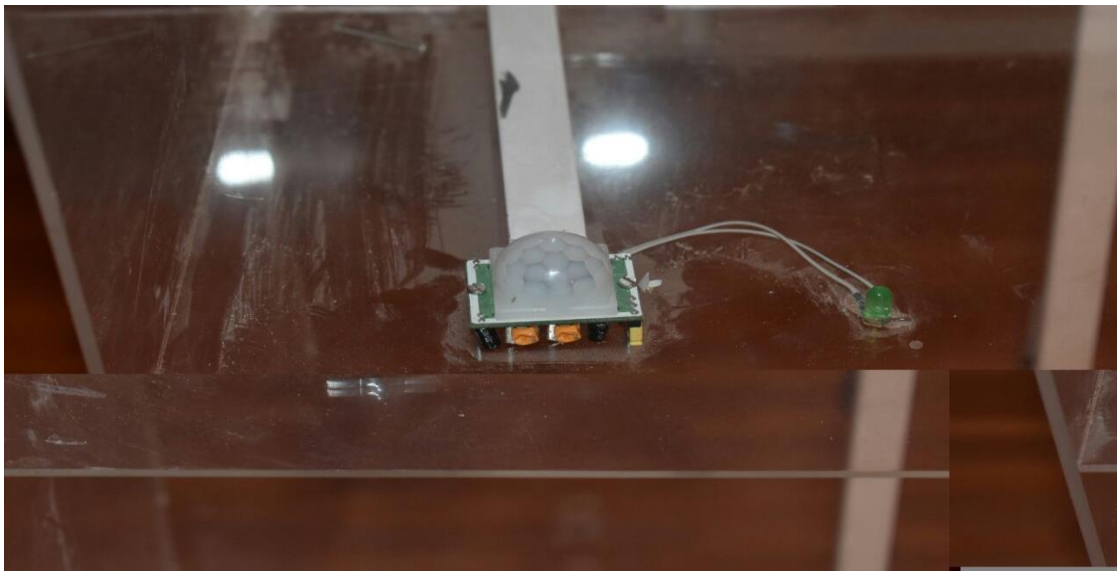

Figura 48. Sensor de presencia infrarroja

3.1.4. Pulsadores

Es posible enviar una notificación directamente sin la necesidad de ser monitoreadas por los sensores, manteniendo presionado máximo 5 segundos el pulsador instalado para cada producto.

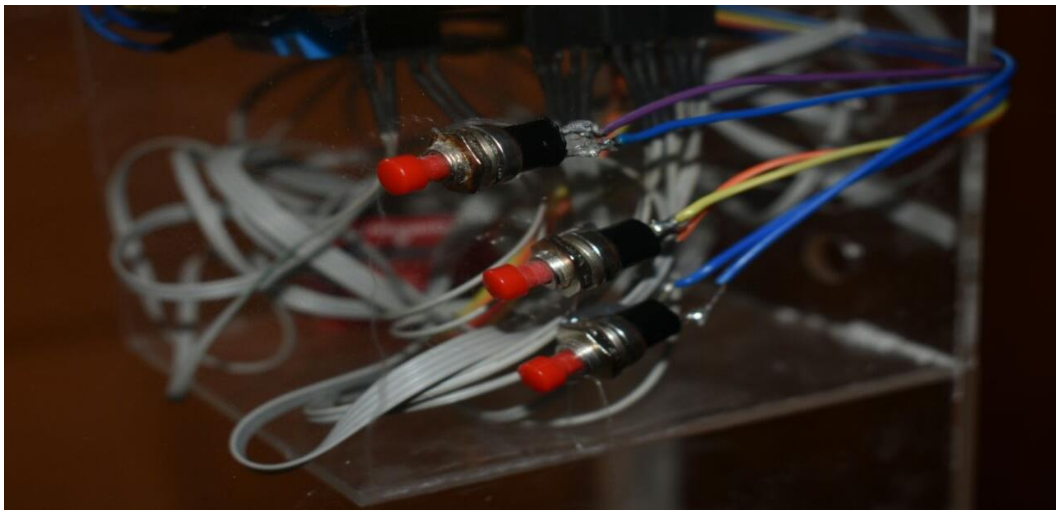

Figura 49. Pulsadores

3.1.5. Proceso de lectura de datos

Es importante antes de implementar el circuito, programar los sensores a usarse, para ello se usó Arduino tanto en la parte de hardware como software. En el anexo A se adjunta el código de programación de Arduino.

3.2. Interpretación de datos en Arduino Uno

La placa de Arduino Uno está programada en los pines digitales 0 y 1 para el módulo Xbee emisor, los pines digitales 2 y 3 se conectan los sensores de presencia, los pines 4 y 5 estarán conectados a las entradas del sensor ultrasónico (Trig y Echo), los pines 6, 7, 8 son para los pulsadores, y en los pines analógicos A0 y A1 serán para la conexión del módulo HX711 que interpretará el peso colocado en la celda de carga.

Figura 50. Arduino Uno

Mediante el monitor Serial del software de Arduino se observa la lectura de los diferentes sensores instalados con el siguiente orden:

Tabla 10.

Lectura de datos programados

Dato 1	Lectura de pulsador N°1
Dato 2	Lectura de pulsador N°2
Dato 3	Lectura de pulsador N°3
Dato 4	Lectura sensor PIR N°1
Dato 5	Lectura sensor PIR N°2
Dato 6	Lectura sensor ultrasónico
Dato 7	Lectura sensor de peso

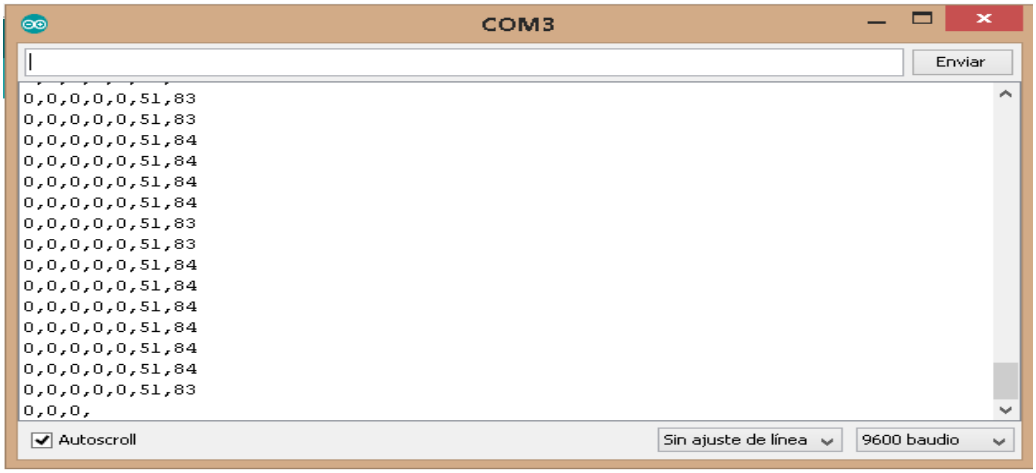

Figura 51. Lectura serial de sensores en Arduino IDE

Para el caso de la celda de carga se necesita calibrar su escala colocando el producto en la balanza elaborada, y se obtuvo un valor de lectura aproximado de 453998,6

Usando la Ecuación *3* y reemplazando los valores obtenidos para obtener el peso en libras se tiene como resultado:

$$Peso\ o\ Fuerza\ esperada\ = \frac{453998{,}6}{2.2}$$

$$Peso\ o\ Fuerza\ esperada\ = 206363$$

Dicho valor debe especificarse en el código de programación para una lectura real del valor de peso.

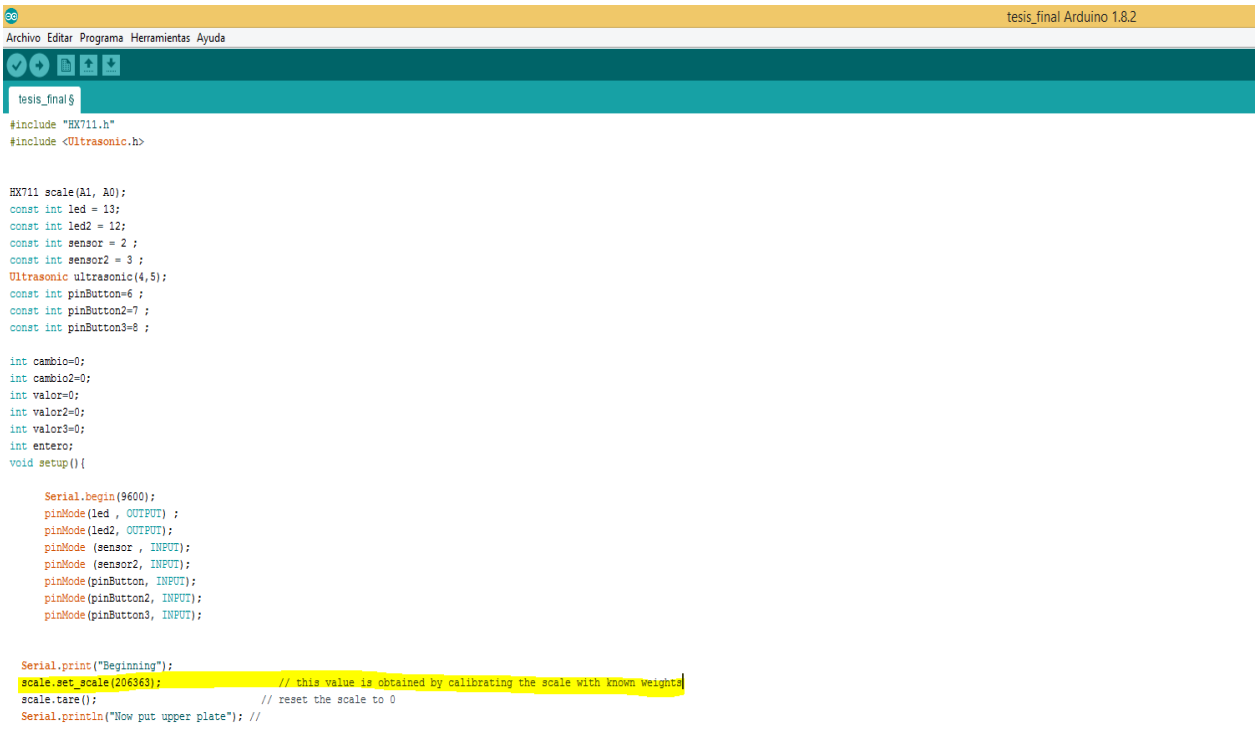

Figura 52. Especificación de escala para el sensor de peso en el código de programación Arduino

3.3. Módulo Xbee Emisor y Receptor

Sera el intérprete que enviará los datos de la lectura de los sensores inalámbricamente al módulo Xbee Receptor.

El módulo Xbee receptor estará conectado a un host recibiendo los datos enviados para su interpretación.

Configuración de módulos XBEE

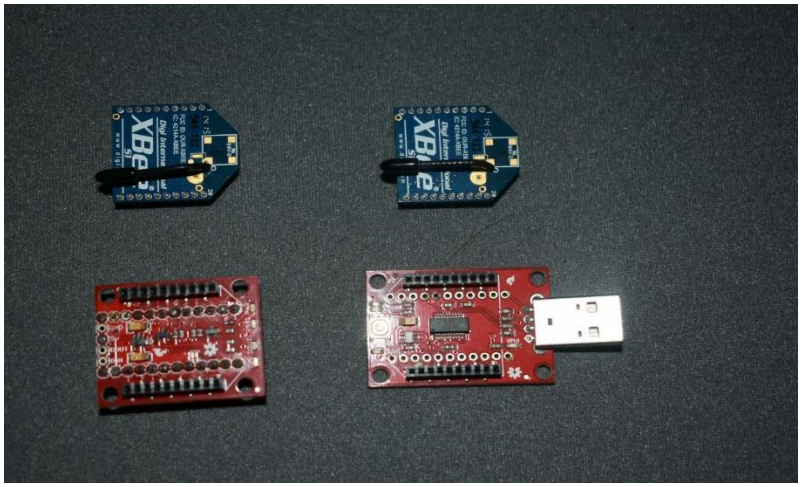

Figura 53. Módulos Xbee Emisor y Receptor

Para la configuración de los módulos Xbee se requiere del número de serie especificado en la parte posterior que son:

Tabla 11.

Números de serie Xbee a ser usados

Módulo Xbee Emisor	0013A20040D5A896
Módulo Xbee Receptor	0013A20040D5A89D

Para que el módulo Xbee en el host sea reconocido se debe instalar el driver que se encuentra en la página oficial del fabricante Digi: https://www.digi.com/support/productdetail?pid=3130

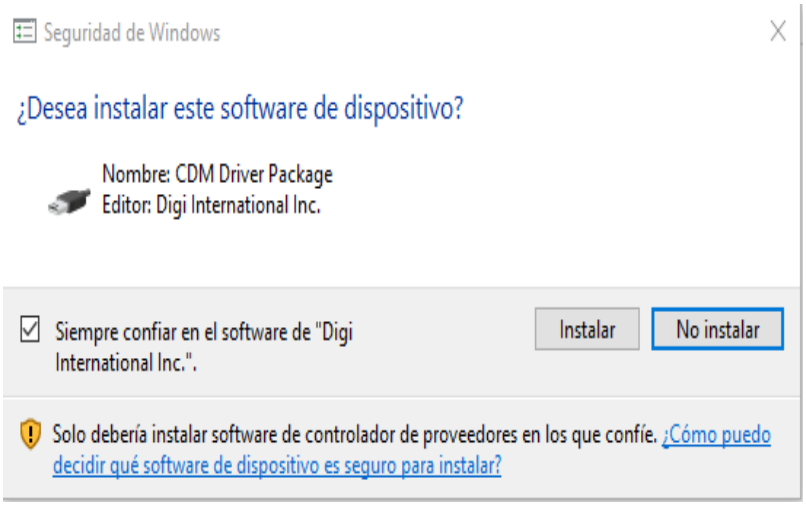

Figura 54. Instalación del Driver Xbee

El siguiente paso es configurar en el software de configuración XCTU conectando ambos módulos (emisor y receptor) al Xbee Dongle en un puerto USB.

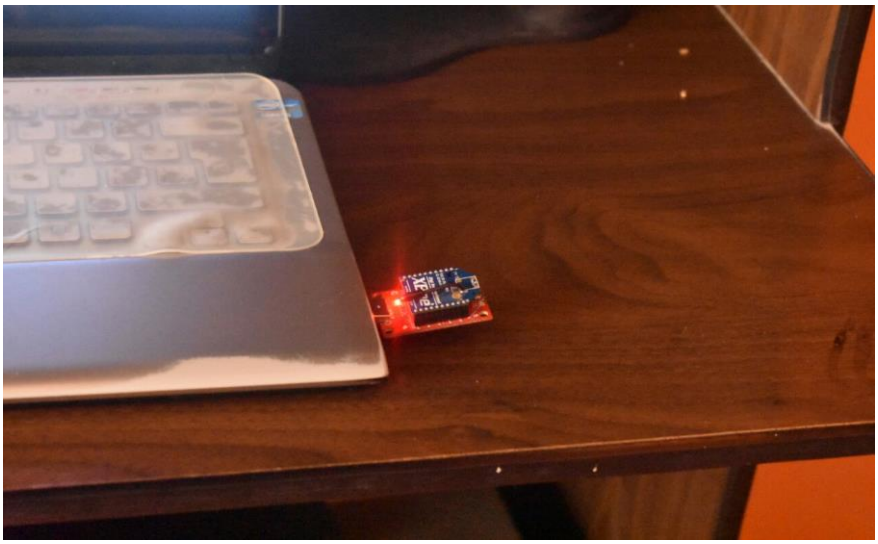

Figura 55. Conexión a la PC para configuración de los módulos Xbee vía USB

En el software XCTU se requiere especificar el puerto al que está conectado el módulo y el número de signos por segundo que es 9600 baudios.

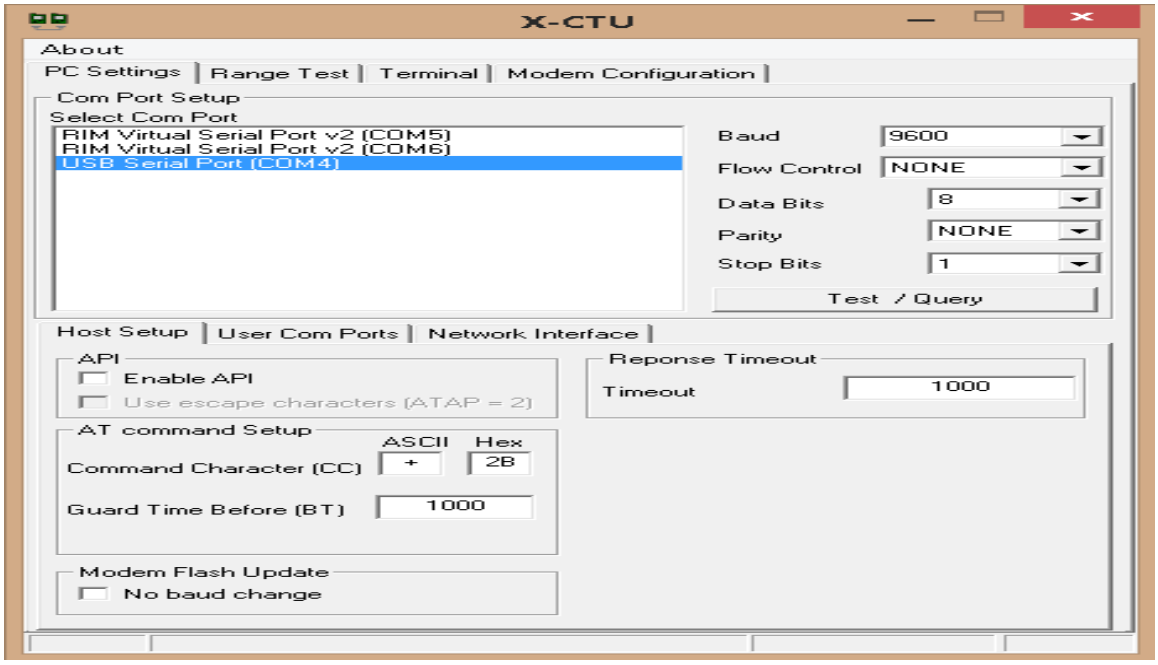

Figura 56. Configuración de Puerto

Para configuración de maestro y esclavo (Emisor - Receptor) se debe seleccionar la pestaña de Modem Configuration, una vez allí seleccionar Read para el reconocimiento del módulo Xbee.

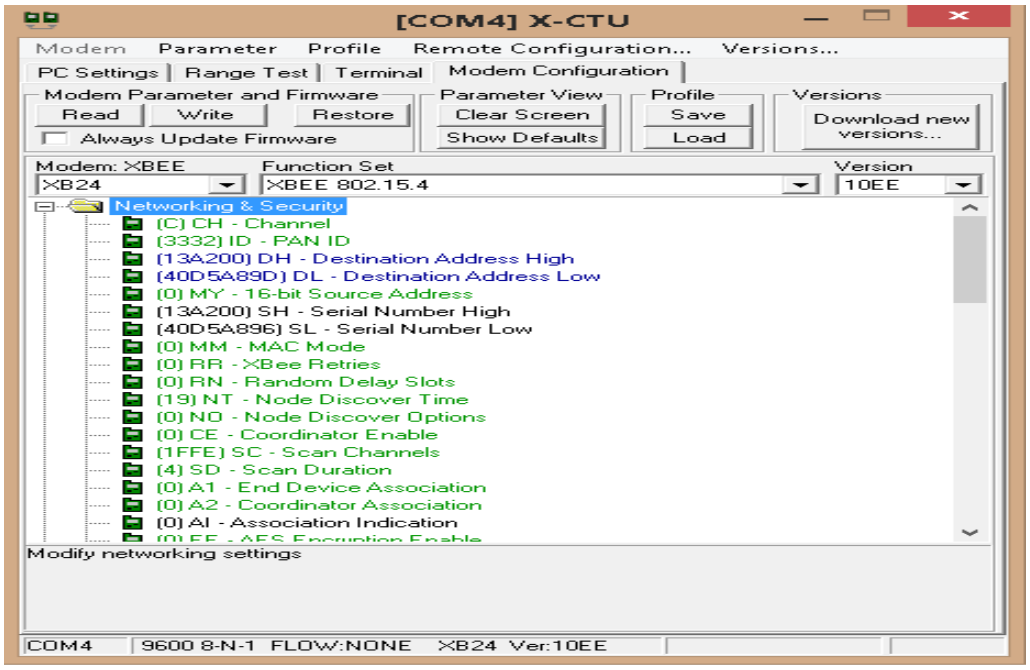

Figura 57. Parámetros de configuración Xbee

Dentro de la opción Networking & Security iremos a DH – Destination Address High, y digitar los 7 primeros números de serie del módulo receptor (Tabla 11), si estuviera conectado el módulo receptor se digitaría el número de serie del módulo emisor y en DL – Destination Address Low se digitarán los siguientes 7 números para terminar la configuración.

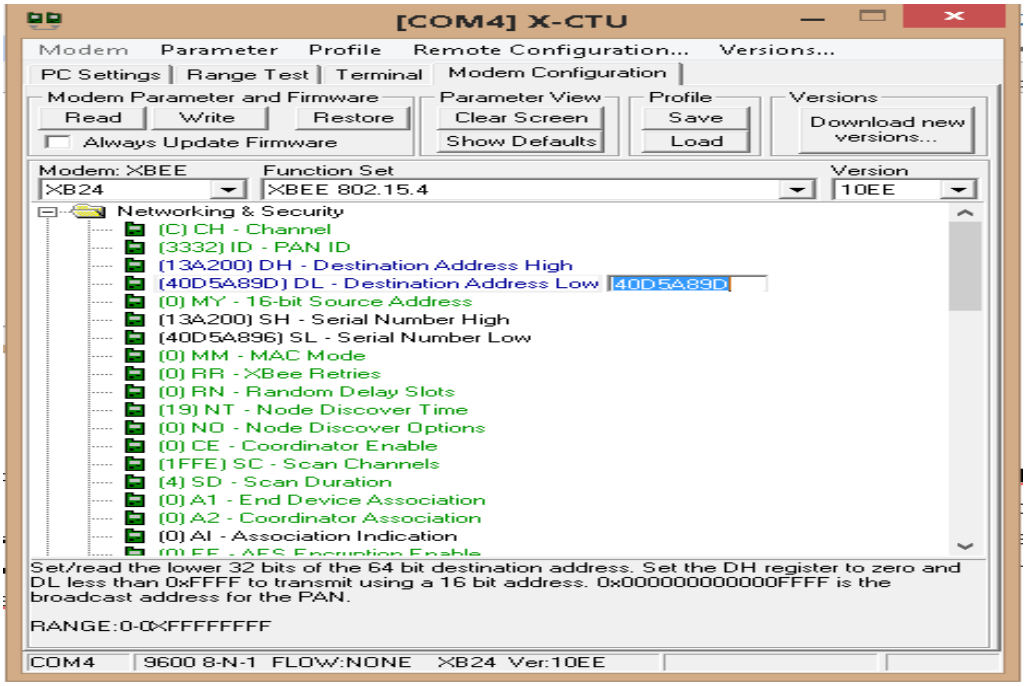

Figura 58. Configuración finalizada módulo Xbee Emisor

El mismo procedimiento se debe realizar para la configuración del módulo receptor y comprobamos él envió de datos abriendo el puerto serial del host.

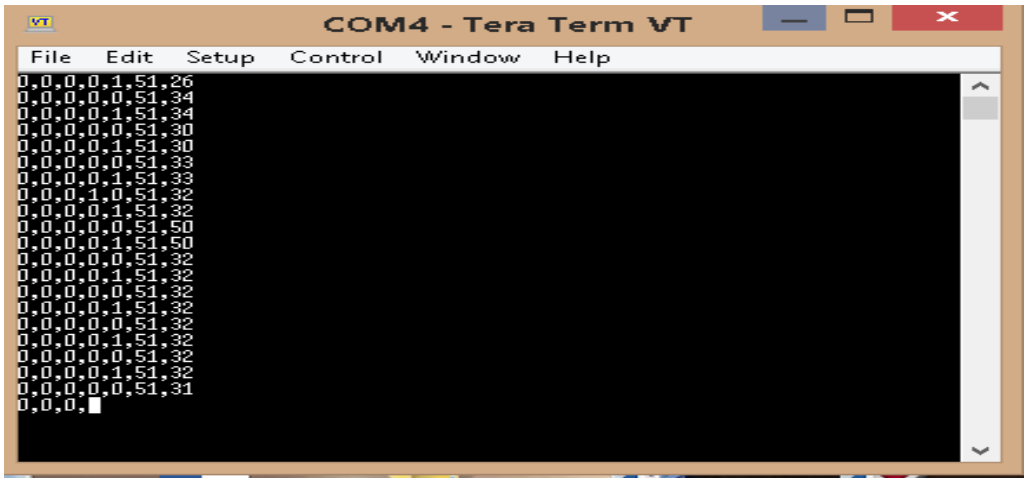

Figura 59. Comprobación de envío de datos por el puerto serial

3.4. Interpretación de datos en Processing

En un host instalado el aplicativo realizado en Processing y conectado vía USB el Módulo Xbee receptor, los datos serán recibidos de forma serial, en el código de programación están las condiciones bajo las cuales será notificado el usuario si determinado producto escasea, también se creó para la visualización de datos una interfaz que mostrará los datos de una forma más entendible para el usuario.

Además, en el icono de lupa se pueden observar los últimos registros tomados que el usuario realizó y se puede salir de la aplicación pulsando en el icono de salir. El código de programación será especificado en el anexo B.

Figura 60. Interfaz gráfica diseñada en Processing

3.5. Notificaciones, Logs y Almacenamiento en Base de Datos

3.5.1. Notificaciones.

La notificación de carencia de producto al usuario será mediante correo electrónico (creado en Gmail).

Para él envió de correos se creó dos cuentas de correo electrónico las para simular las cuentas del cliente y proveedor, las misma se las especificó de la siguiente manera:

Cliente:

información.faltante.udla@gmail.com

Proveedor:

supermercado.udla@gmail.com

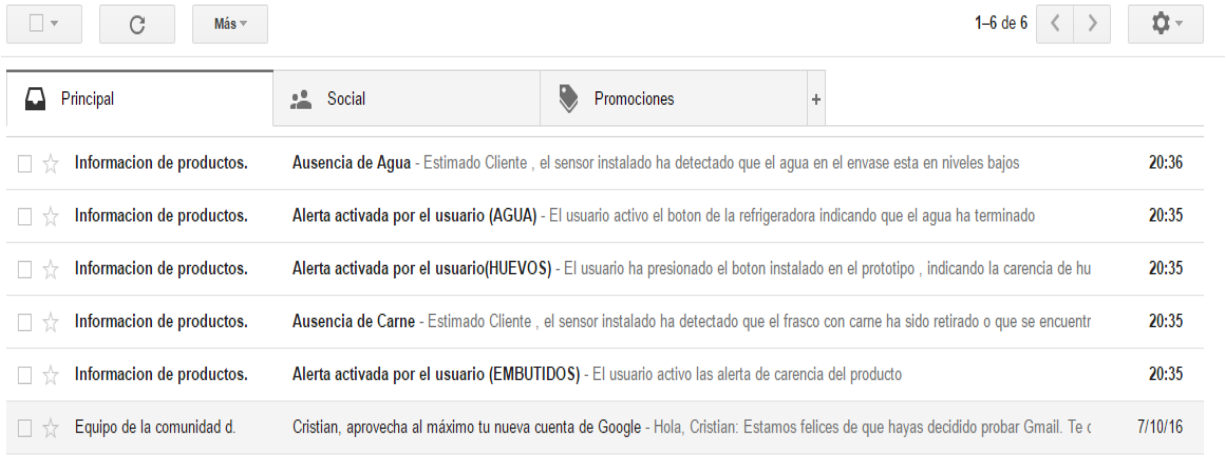

Figura 61. Notificación de productos faltantes vía correo electrónico

3.5.2. Logs

Los logs son datos almacenados en ficheros, para este proyecto los datos se los guardará en un archivo .txt, la extensión del archivo puede cambiar acorde con las necesidades que se requieran mediante código en Processing.

El archivo de texto plano que estará ubicado en la carpeta del proyecto, el archivo de texto plano guardará datos indicando el día, fecha y hora en que los sensores y pulsadores detectaron que el producto escasea o fue retirado de su lugar.

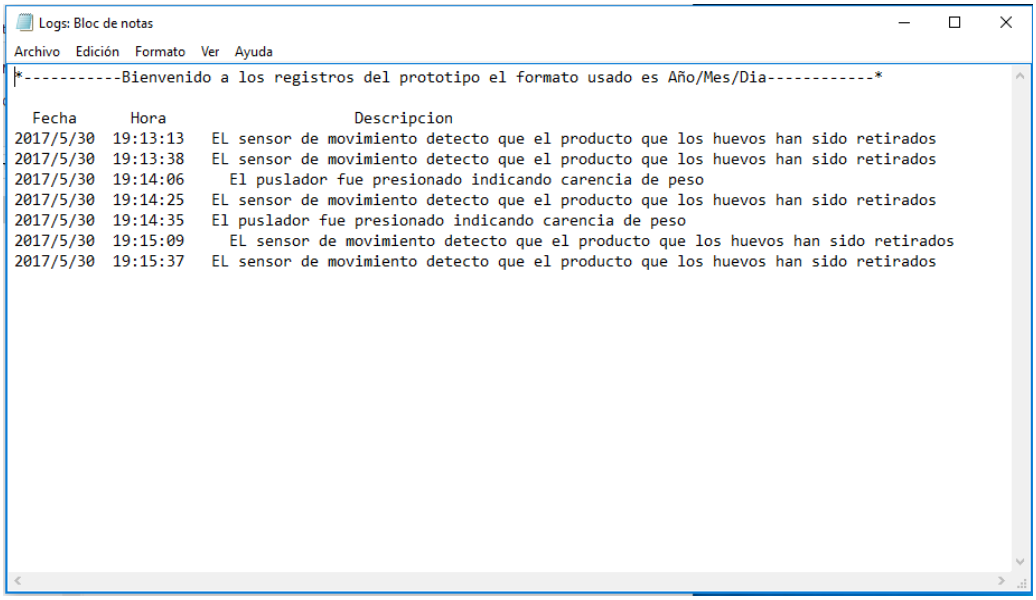

Figura 62. Registro de eventos (Logs) en la computadora

3.5.3. Almacenamiento en la Base de Datos

Para tener un registro de todos los eventos se optó también usar por crear en el servidor de base de datos una tabla que lleva por nombre Proyecto para almacenar la fecha y hora con la descripción de cual sensor fue el que se activó, la estructura de la tabla es:

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	Id	int(11)			No	*Ninguna*		AUTO_INCREMENT
2	Descripcion	varchar(200)	latin1_swedish_ci		No	*Ninguna*		
3	Anio	int(11)			No	*Ninguna*		
4	Mes	int(11)			No	*Ninguna*		
5	Dia	int(11)			No	*Ninguna*		
6	Hora	int(11)			No	*Ninguna*		
7	Minuto	int(11)			No	*Ninguna*		
8	Segundo	int(11)			No	*Ninguna*		

Figura 63. Campos de la base de datos.

3.6. Implementación y Pruebas

La implementación del proyecto consta de una maqueta que fue diseñada para simular a una refrigeradora y serán colocados agua, huevos y embutidos para su respectivo inventario.

Figura 64. Maqueta diseñada para la implementación del proyecto.

El diagrama de conexiones sería el siguiente:

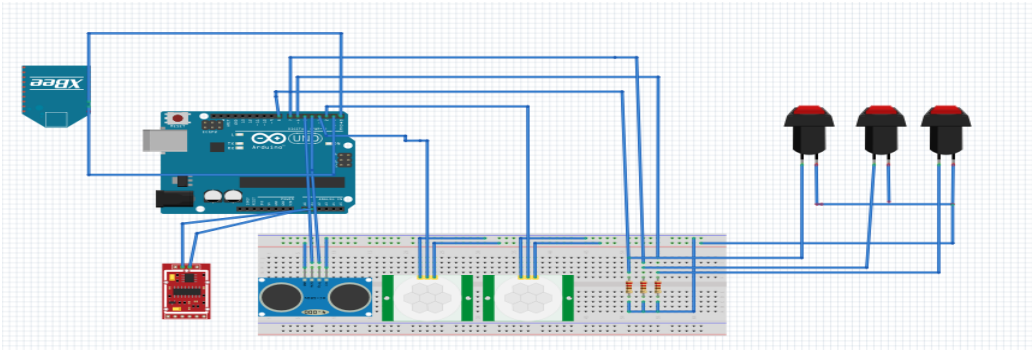

Figura 65. Diagrama de Conexión en Fritzing

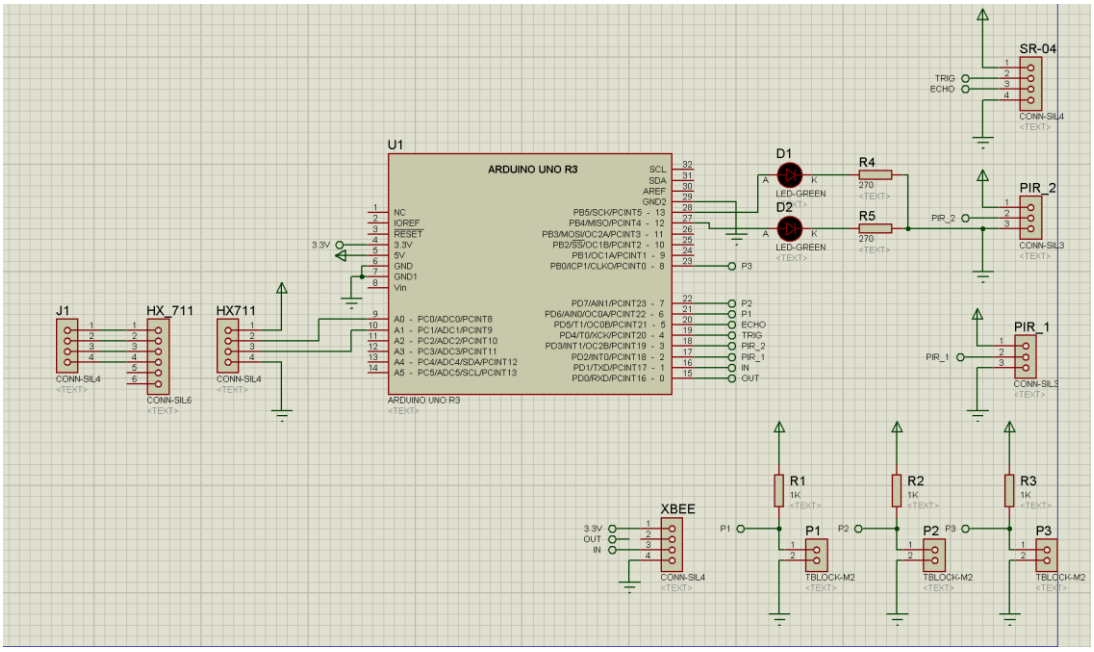

Figura 66. Diagrama de Conexión en ISIS

Para la lectura de sensores es necesario probar el funcionamiento en una placa de pruebas (protoboard) con el diagrama de conexión especificado en la Figura 65.

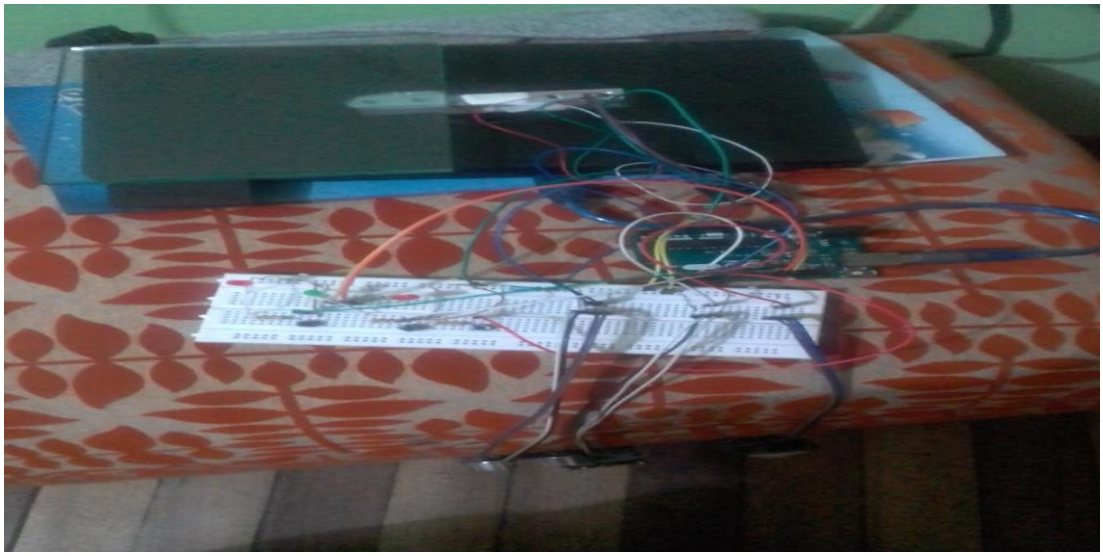

Figura 67. Implementación del circuito para pruebas reales

Comprobando el buen funcionamiento en el protoboard, se diseñó e implementó una placa electrónica, mediante Proteus y su herramienta Ares fue posible la creación de la placa.

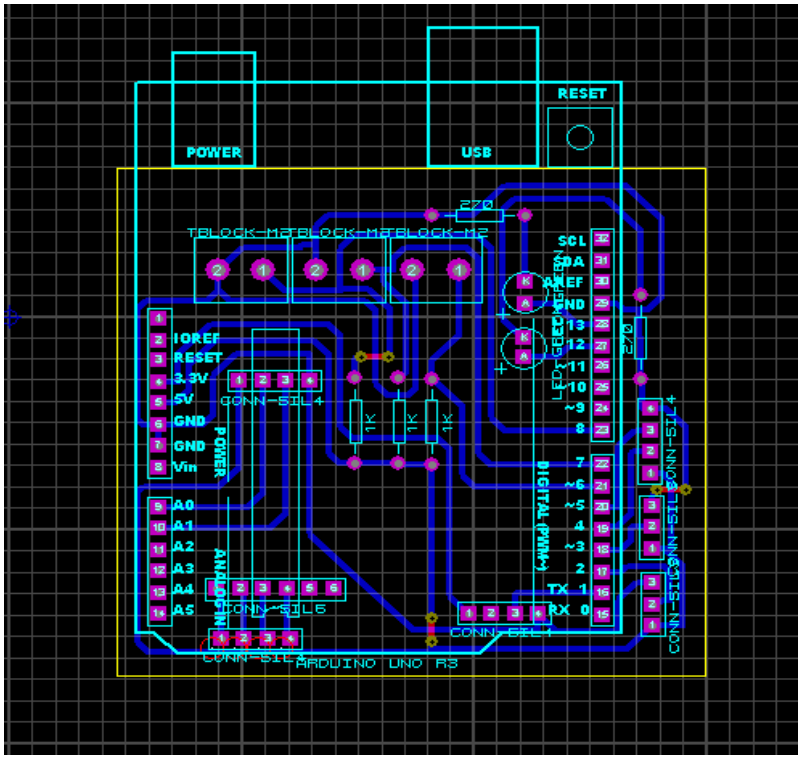

Figura 68. Placa electrónica diseñada en ARES

Para la implementación del circuito en una placa electrónica se deben seguir los siguientes procedimientos:

- Para la impresión usar una impresora Láser
- Imprimir en un papel de transferencia calórica
- Lijar la baquelita en la parte superior, donde se encuentra el cobre.
- Con el uso de una plancha, colocar el impreso en la parte lijada de la baquelita, una vez terminado el proceso sumergir la placa en agua fría durante 5 minutos y quitar el restante de papel.
- Poner la baquelita en percloruro férrico, una vez terminado el proceso lijar y verificar continuidad en el circuito.

- Mediante un taladro y con una broca fina, realizar los agujeros en los lugares donde los componentes electrónicos serán colocados.
- Con el uso de un cautín y estaño, soldar los componentes.

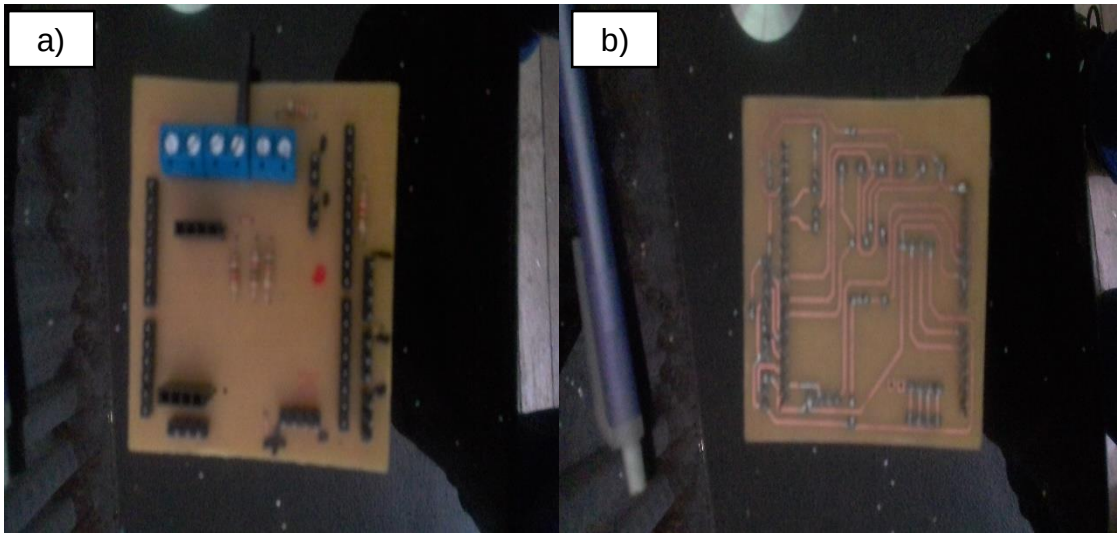

Figura 69. Placa soldada

a) vista superior
b) vista inferior

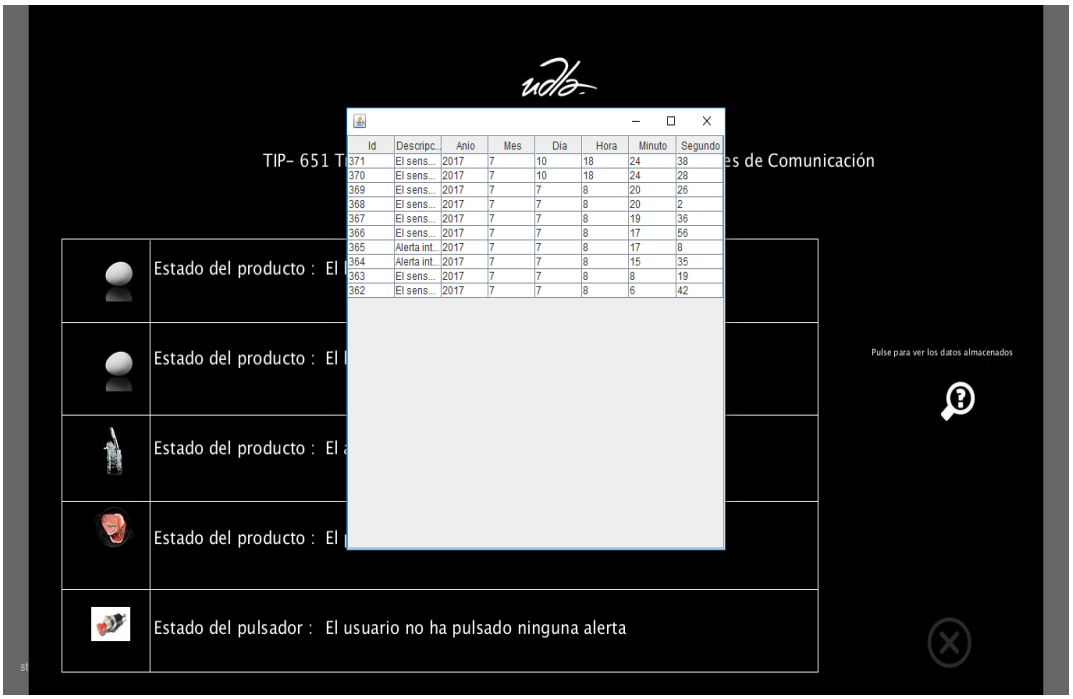

Figura 70. Interfaz del prototipo con los últimos eventos

Figura 71. Ubicación de productos en los sensores

4. CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

La implementación del prototipo se lo realizó en una maqueta, utilizando productos de tamaño y peso adecuados para calibrar los sensores y se puedan obtener datos en un entorno simulado exceptuando factores de temperatura y humedad, logrando informar al usuario si alguno de los productos se encuentra escaso o si no está en su lugar después del tiempo programado, que para efectos demostrativos se estableció una espera de 10 segundos , pasado ese tiempo el prototipo envía la notificación al proveedor y además el prototipo fue capaz de documentar los eventos ocurridos en la maqueta en un archivo de texto y en la base de datos .

La interfaz gráfica con el usuario debe ser intuitiva y fácil de manejar, por lo que se eligió la programación de la misma en Processing, por su amplia difusión en sistemas de adquisición de datos desde Arduino y la facilidad de información disponible. Esta interfaz permite monitorear el estado del producto y además interpreta los datos enviados desde el prototipo al host bajo las condiciones de stock de los productos. Se investigaron y aplicaron las librerías necesarias para que Processing pueda enviar notificaciones vía correo electrónico, y que este envío sea controlado, debido a que en pruebas iniciales se detectaron envíos de correos repetitivos cuando la lectura de los datos detectaba escasez de productos, por ejemplo, al encontrarse el agua a una distancia de 31 cm con el sensor HR-SR04 (sensor de presencia) se enviaban correos repetitivos hasta que la condición de distancia cambie a menor a 29 cm.

El envío de datos entre el prototipo y el host requieren de un protocolo que permita el envío y recepción de un volumen de datos bajo, por lo que se eligieron los módulos de comunicación inalámbrica Xbee, aprovechando además su bajo consumo de potencia y tamaño reducido para la

implementación en un prototipo real, ofrece un alcance adecuado para un ambiente doméstico por lo que no existe problema en la ubicación del host dentro de cualquier ambiente de una casa.

Para la adaptación de las señales de los sensores fue necesaria su calibración previa con muestras de productos, en algunos casos se probó más de un tipo de sensor hasta encontrar el que mejor se adapte a las necesidades del prototipo, finalmente se configuraron e instalaron los siguientes sensores: el sensor PIR, capaz de detectar si los huevos fueron retirados de la base donde fueron ubicados , el sensor de peso que consta del módulo HX711 y la celda de carga interpreto el peso del producto , el sensor HRSR-04 permitió medir el nivel de líquido colocado en el recipiente , y de igual manera los pulsadores fueron capaces de enviar notificaciones al ver directamente el usuario que un producto en específico escaseo.

4.2. Recomendaciones

Para que el funcionamiento del prototipo sea óptimo se debe ser muy cuidadoso en la colocación de los productos para que los sensores sean capaces de realizar la lectura adecuada.

El sensor de peso deberá ser calibrado por primera vez antes de iniciar el funcionamiento normal, con un peso conocido para su correcto funcionamiento.

Se debe evitar la exposición de los sensores a temperaturas extremas o mojarlos y en caso de que estas condiciones sean inevitables se deben proteger en algún contenedor.

Se puede complementar el prototipo con una aplicación móvil para que el usuario decida enviar o no el correo electrónico al proveedor de alimentos cuando éste se termine.

En aplicaciones críticas como el control de stock en un restaurant, se debe tener un enlace de backup del servicio de Internet para mantener la comunicación con el proveedor de alimentos.

La distancia entre el prototipo y el host no debe exceder los 40 metros con línea de vista, por lo que se recomienda ubicarlos a distancias menores a esta o dentro del mismo ambiente.

REFERENCIAS

5 Hertz. (s.f.). *Celda de Carga*. Recuperado el 2 de noviembre de 2016 de http://5hertz.com/tutoriales/?p=690

5 Hertz. (s.f.). *Sensores de Ultrasonido*. Recuperado el 14 de octubre de 2016 de http://5hertz.com/tutoriales/?p=659

Area Tecnológica. (s.f.). *¿Qué es Domótica?*. Recuperado el 3 de marzo de 2017 de http://www.areatecnologia.com/electricidad/domotica.html

Blogspot. (2011). *Módulos Xbee*. Recuperado el 23 de febrero de 2017 de http://alvarounal.blogspot.com/2011/10/modulos-xbee-parte1.html

Botboss. (s.f.). *Tutorial Arduino, medir distancia con sensor ultrasónico HC-SR04*.Recuperado el 15 de enero de 2017 de http://bot-boss.com/tutorial-arduino-medir-distancia-sensor-ultrasonico-hc-sr04/

Buioli, I. Pérez Marín, J. (2013). *Processing: Un lenguaje al alcance de todos*. Recuperado el 1 de diciembre de 2016 de http://laurence.com.ar/artes/comun/Processing_un_lenguaje_al%20alcance_de_todos.pdf

Casares, J. (2016). *Puente de Wheatstone*. Recuperado el 5 de Marzo de 2017 de: http://josecasares.com/puente-de-wheatstone/

Cedom. (s.f). *Que es domótica.* Recuperado el 1 de diciembre del 2016 de http://www.cedom.es/sobre-domotica/que-es-domotica

Compututorials. (2011). *Módulos Xbee …Lo básico*. Recuperado el 23 de febrero del 2017 de http://compututorials.blogspot.com/2011/11/modulos-xbee-lo-basico.html

Consinfin. (s.f). *¿Que es la comunicación inalámbrica Wireless?*. Recuperado el 12 de noviembre de 2016 de http://consinfin.com/que-es-la-comunicacion-inalambrica-wireless/

Crespo, E. (2017). *Programación Arduino.* Recuperado el 15 de marzo de 2016 de https://aprendiendoarduino.wordpress.com/2017/01/23/programacion-arduino-5/

Del Ángel, J. (2015). *Identificación de las aplicaciones de la mecatrónica en el desarrollo tecnológico*. Recuperado el 24 de octubre del 2016 de http://iosmando.blogspot.com/2015/05/instalacion-de-sistemas-mecatronicos.html?view=flipcard

Electronics Hacking. (2016). *EVERYTHING YOU NEED TO KNOW ABOUT ARDUINO(GENUINO)*. Recuperado el 23 de marzo de 2017 de http://electronicshacking.com/everything-you-need-to-know-about-arduino/

Electrón perdido. (s.f.). *Celda de carga 5 kg*. Recuperado el 2 de noviembre de 2016 de http://electronperdido.com/shop/arduinocompatible/sensores/celda-de-carga-5kg-czl635/

Fabrica Digital. (s.f.). *Entradas analógicas con el potenciómetro.* Recuperado el 5 de noviembre de 2016 de https://fabricadigital.org/leccion/entradas-analogicas-con-el-potenciometro/

Gobierno de Canarias. (2013). *Características Arduino.* Recuperado el 7 de noviembre de 2016 de http://www3.gobiernodecanarias.org/medusa/ecoblog/ralvgon/files/2013/05/Caracter%C3%ADsticas-Arduino.pdf

Hubor Proteus. (s.f.). *Herramienta grafica para gestión de encapsulados*. Recuperado el 5 de marzo de 2017 de http://hubor-proteus.com/proteus-pcb/isis/14-herramienta-gr%C3%A1fica-para-la-gesti%C3%B3n-de-los-encapsulados.html

iReformas. (2015). *Domótica y mecanismos*. Recuperado el 5 de marzo de 2017 de http://www.ireformas.com/servicios/electricidad-instalaciones-electricas/2015-02-19-07-14-43.html

Llamas, L. (2015). *Detector de Movimiento con Arduino y Sensor PIR*. Recuperado el 4 de diciembre de 2016 de https://www.luisllamas.es/detector-de-movimiento-con-arduino-y-sensor-pir/

Llamas, L. (2015). *MEDIR DISTANCIA CON ARDUINO Y SENSOR DE ULTRASONIDOS HC-SR04*. Recuperado el 4 de diciembre de 2015 de https://www.luisllamas.es/medir-distancia-con-arduino-y-sensor-de-ultrasonidos-hc-sr04/

Logicbus. (s.f.). *Celdas de carga tipo botón*. Recuperado el 4 de marzo de 2017 de http://www.logicbus.com.mx/celdas_miniatura.php

Martínez, F. (2015). *Tutorial Arduino: IDE Arduino*. Recuperado el 4 de octubre de 2016 de https://openwebinars.net/tutorial-arduino-ide-arduino/

Molina, F. (s.f.). *Cálculos de la primera zona de Fresnel.* Recuperado el 15 de febrero de 2017 de https://www.franciscomolina.cl/798/calculos-primera-zona-de-fresnel

Naylamp Mechatronics. (s.f.). *Celda de Carga y módulo Hx711 Arduino*. Recuperado el 27 de febrero de 2017 de http://www.naylampmechatronics.com/blog/25_tutorial-trasmisor-de-celda-de-carga-hx711-ba.html

Pallás Areny, R. (2003). *Sensores y Acondicionamiento de la Señal*. (4ta Edición). [versión electrónica]. Recuperado el 7 de octubre de 2016 de https://books.google.es/books?hl=es&lr=lang_es&id=Eevyk28_fVkC&oi=fnd&pg=PR11&dq=que+es+un+sensor&ots=JWmN56Fvae&sig=y1frydaGSXexTUDhqpuPrKCw-F4#v=onepage&q=que%20es%20un%20sensor&f=false

Pérez, D. (s.f.). *Sensores de Distancia por ultrasonidos*. Recuperado el 7 de octubre de 2016 de http://www.alcabot.com/alcabot/seminario2006/Trabajos/DiegoPerezDiego.pdf

Recuero, A. (s.f.). *Estado Actual y Perspectivas de la Domótica*. Recuperado el 20 de octubre de http://informesdelaconstruccion.revistas.csic.es/index.php/informesdelac onstruccion/article/view/827/912

Santos, N. (2015). *Zona de Fresnel.* Recuperado el 24 de febrero de 2017 de http://zonafresnel-edu.blogspot.com/

Sparkfun. (s.f.). *Ultrasonic Sensor HR-SR04*. Recuperado el 1 de noviembre de 2016 de https://www.sparkfun.com/products/13959

Sparkfun. (s.f.). *SparkFun Load Cell Amplifier - HX711*. Recuperado el 5 de diciembre de 2016 de https://www.sparkfun.com/products/13879

Talos Electronics. (s.f.) .*Módulo A/D 24 Bits HX711 para celdas de carga*. Recuperado el 4 de marzo de 2017 de: https://www.taloselectronics.com/producto/modulo-a7d-24-bits-hx711-para-celdas-de-carga

Tecmake Electronics. (s.f.). *Tutorial - Sensor de Distancia Ultrasónico HC-SR04*.Recuperado el 16 de diciembre de 2016 de: http://www.techmake.com/tutorialhcsr04

Tecnológico BJ12. (2011). *Xbee*. Recuperado el 5 de noviembre de 2016 de : http://tecnologicobj12.blogspot.com/2011/09/que-es-xbee.html

Todo Robótica. (s.f.). *HX711*. Recuperado el 2 de noviembre de 2016 de http://tdrobotica.co/circuito-amplificador-de-celda-de-carga-hx711/395.html

Toledo Mettler. (2001). *Manual de Sistemas de Módulos de Peso*. Recuperado el 5 de Marzo de 2017 de http://www.mt.com/dam/mt_ext_files/Editorial/Generic/5/WM_eng_spec_tension_appl_0x0002464400026aa2000603d5_files/tensionsp.pdf

Tutoelectro. (s.f). *Definición de Proteus*. Recuperado el 3 de febrero de 2017 https://tutoelectro.wikispaces.com/file/view/CAPITULO+1.pdf/230147400/ CAPITULO+1.pdf

UTN. (s.f.). *X-CTU CONFIGURATION & TEST UTILITY SOFTWARE*. Recuperado el 2 de noviembre de 2015 de http://repositorio.utn.edu.ec/bitstream/123456789/1057/3/04%20RED%20O12%20-5%20MANUAL%20XCTU.pdf

Valdés, E. Pérez, F. Arias, O. (2013). *Sistema de Adquisición con comunicación inalámbrica*. Recuperado el 5 de diciembre de 2016 de: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S1815-59282013000300007

Vega, J. Salgado, G. Lagos, M. Tapia, V. Sánchez, F. (2014). *Red inalámbrica de sensores de presencia*. Pag. 3. Recuperado el 2 de noviembre de http://somi.ccadet.unam.mx/somi29/memoriassomi29/PDFS/electronica/46-SGSOMI-66-46.pdf

Villapol, M. (2010). *Introducción a las Redes Móviles e Inalámbricas*. Recuperado el 2 de noviembre de 2016 de http://www.ciens.ucv.ve/redesmov/Documentos/IntroduccionRMI.pdf

Zona Maker. (s.f.). *Sensor Ultrasónico HR-SR04*. Recuperado el 6 de marzo del 2017 de https://www.zonamaker.com/arduino/modulos-sensores-y-shields/ultrasonido-hc-sr04.

ANEXOS

ANEXO A

```
#include "HX711.h"
#include <Ultrasonic.h>



HX711 scale (A1, A0);
const int led = 13;
const int led2 = 12;
const int sensor = 2;
const int sensor2 = 3;
Ultrasonic ultrasonic (4,5);
const int pinButton=6;
const int pinButton2=7;
const int pinButton3=8;
int cambio=0;
int cambio2=0;
int valor=0;
int valor2=0;
int valor3=0;
int entero;


void setup () {

    Serial.begin(9600);
    pinMode (led, OUTPUT);
    pinMode (led2, OUTPUT);
```

```
    pinMode (sensor, INPUT);
    pinMode (sensor2, INPUT);
    pinMode (pinButton, INPUT);
    pinMode (pinButton2, INPUT);
    pinMode (pinButton3, INPUT);


  Serial.println("Beginning");
  scale.set_scale (206363); // valor obtenido al calibrar el sensor
  scale. Tare ();
  delay (1000);
  Serial.println("Now put upper plate");




 }

void loop () {

   int stateButton = digitalRead(pinButton);
   int stateButton2 = digitalRead(pinButton2);
   int stateButton3 = digitalRead(pinButton3);
     if (stateButton == 1) {
     valor=0;
     Serial.print(valor);
     Serial.print(",");
     } else {
```

```
    valor=1;
    Serial.print(valor);
    Serial.print(",");
}



    if (stateButton2 == 1) {
    valor2=0;
    Serial.print(valor2);
    Serial.print(",");
    } else {
    valor2=1;
    Serial.print(valor2);
    Serial.print(",");
}



  if (stateButton3 == 1) {
    valor3=0;
   Serial.print(valor3);
   Serial.print(",");
} else {
  valor3=1;
   Serial.print(valor3);
   Serial.print(",");
}

int f= (scale.get_units (10) *100);
```

```
if (digitalRead(sensor)) {
      cambio=1;
      cambio2=0;
      digitalWrite (led, HIGH);
      delay (1000);
      Serial.print(cambio);
      Serial.print(",");
      Serial.print(cambio2);
      Serial.print(",");
      Serial.print(ultrasonic. Ranging(CM));
      Serial.print(",");
      Serial.println(f);
     scale. power_down ();
     scale. power_up ();

      delay (300);

 }


 else {

      cambio=0;
      cambio2=0;
      digitalWrite(leedlo);
      delay (1000);
```

```
          Serial.print(cambio);
          Serial.print(",");
          Serial.print(cambio2);
          Serial.print(",");
          Serial.print(ultrasonic. Ranging(CM));
          Serial.print(",");
          Serial.println(f);
          scale. power_down ();
         scale. power_up ();

         delay (1000);




  }




if (stateButton == 1) {
      valor=0;
    Serial.print(valor);
    Serial.print(",");
 } else {
   valor=1;
    Serial.print(valor);
    Serial.print(",");
 }
```

```
if (stateButton2 == 1) {
      valor2=0;
     Serial.print(valor2);
     Serial.print(",");
  } else {
    valor2=1;
     Serial.print(valor2);
     Serial.print(",");
  }


    if (stateButton3 == 1) {
      valor3=0;
     Serial.print(valor3);
     Serial.print(",");
  } else {
    valor3=1;
     Serial.print(valor3);
     Serial.print(",");
  }


    if (digitalRead(sensor2)) {

          cambio=0;
          cambio2=1;
          digitalWrite (led2, HIGH);
```

```
    delay (1000);
    Serial.print(cambio);
    Serial.print(",");
    Serial.print(cambio2);
    Serial.print(",");
    Serial.print(ultrasonic. Ranging(CM));
    Serial.print(",");
    Serial.println(f);
    scale. power_down ();
   scale. power_up ();

    delay (1000);

}

else {
    cambio=0;
    cambio2=0;
    digitalWrite (led2, LOW);
    delay (1000);
    Serial.print(cambio);
    Serial.print(",");
    Serial.print(cambio2);
    Serial.print(",");
    Serial.print(ultrasonic. Ranging(CM));
    Serial.print(",");
   Serial.println(f);
   scale. power_down ();
```

```
        scale. power_up ();


        delay (1000);




    }
}
```

ANEXO B

```
importa javax. swing. JOptionPane;     //librería para cuadros de dialogo
import javax. swing. JFrame;               //librería para cuadros de dialogo
import processing.serial.*;               // libreria para el uso del puerto serial
import javax.mail.Authenticator;          // Libreria para el envio de correos
import javax.mail.PasswordAuthentication;   // Libreria para el envio de correos
import javax.mail.*;                      // Libreria para el envio de correos
import javax.mail.internet.*;             // Libreria para el envio de correos
import java.util.Properties;              // Libreria para el envio de correos
import javax.mail.Message;                // Libreria para el envio de correos
import javax.mail.MessagingException;      // Libreria para el envio de correos
import javax.mail.Session;                // Libreria para el envio de correos
import javax.mail.Transport;              // Libreria para el envio de correos
import javax.mail.internet.InternetAddress;  // Libreria para el envio de correos
import javax.mail.internet.MimeMessage;       // Libreria para el envio de correos
PImage foto,foto1,foto2,foto3,foto4,foto5,foto6;



Serial puerto;   // variable para lectura de puerto
int sensorpir=0;                          // variable para lectura del sensor pir n°1
int sensorpir2=0;                         // variable para lectura del sensor pir n°2
int sensorultrasonico=0;                      // variable para lectura del sensor
ultrasonico
int variable =0;                          // variable de control (acumulador)
int variable2=0;                          // variable de control (acumulador)
int correo=0;                             // variable de control (acumulador)
int correo2=10;                           // variable de control (acumulador)
int correo3 =10;                          // variable de control (acumulador)
float sensorpeso;                           // variable para lectura del sensor de
peso
```

```
int pulsador=0;                          // variable para lectura de pulsador N°1
int pulsador2=0;                          // variable para lectura de pulsador N°2
int pulsador3=0;                          //variable para lectura de pulsador N°3
int change=0;
int change2=0;
int change3=0;
int ventana=0;
int ventana2=0;
int ventana3=0;
int variable_logs;
PrintWriter logs;
ControlP5 cp5;

void setup()
{
  //Formato de pantalla
  size(1300, 768);   //Tamaño

  foto=loadImage("logo_udla1.jpg");              // carga de imagenes en interfaz
grafica
  foto1=loadImage("huevo(1).png");
  foto2=loadImage("agua.png");
  foto3=loadImage("huevo(2).png");
  foto4=loadImage("res.jpeg");
  foto5=loadImage("pulsador.jpg");
  foto6=loadImage("boton salir_opt.png");
  background(0);

    puerto= new Serial(this, Serial.list()[0], 9600);   // apertura del puerto serial
    puerto.bufferUntil('\n');           // lectura del buffer hasta encontrar un
espacio
```

```
    logs = createWriter("Escritorio/Logs.txt");
    logs.println("*-----------Bienvenido a los registros del prototipo el formato usado es Año/Mes/Dia----------*");
    logs.println("");
    logs.println("  Fecha  "+"    "+"Hora " +"      "+"            Descripcion    ");


  }


  void controlEvent(ControlEvent theEvent)
{




}



  void serialEvent(Serial puerto){
   String inString= puerto.readStringUntil('\n');

    if (inString !=null){
    inString= trim(inString);
    int [] sensors = int(split(inString, ","));        // separa bits mediante comas
    if (sensors.length>=7){
    pulsador=sensors[0];                              // definicion de sensores
    pulsador2=sensors[1];
    pulsador3=sensors[2];
    sensorpir=sensors[3];
    sensorpir2=sensors[4];
    sensorultrasonico=sensors[5];
```

```
    sensorpeso=sensors[6];


   }
   }
}




void draw()
{
  background (0,0,0);                                    // Desarrollo de interfaz
grafica



  image (foto,600,0);
  image (foto1,100,270);
  image (foto1,100,370);
  image (foto2, 100, 470);
  image (foto4 , 100 ,570);
  image (foto5, 100 ,650);
  image (foto6 ,1150,680);



      if (mousePressed)
      if(mouseX>1150 && mouseX<1211 && mouseY>680 && mouseY<740){
      Cerrar();
      }
 if(mouseX>1150 && mouseX<1211 && mouseY>680 && mouseY<740){
 textSize(10);
 text("Pulse para salir de la aplicacion",1112,678);
  }
```

```
    textSize(20);
  text("TIP- 651 Trabajo de Titulación - Ingeniería Electrónica y Redes de Comunicación",300,180);
  text("Sistema de Control de productos faltantes",450,230);


  text("Estado del producto : ",160,300);      // Control del sensor pir
  text("Estado del producto : ",160,400);      // Control del sensor pir2
  text("Estado del producto : ",160,500);      // Control del sensor ultrasonico
  text("Estado del producto : ",160,600);      // Control del sensor de peso
  text("Estado del pulsador : ",160,680);



    //sensorpeso=sensors[2];

 //------------------------------COMANDOS DEL SENSOR PESO-------------------------------------------


   float sensorpesolibras=(sensorpeso/100);
   if (sensorpesolibras<1){     //condicion sensor de peso si el peso es mayor a 100 acumuladores aumentan
   text("El producto ha sido retirado o su peso es bajo , " + sensorpesolibras + "lbs" ,380,600);  // mensaje en interfaz grafica


   correo3--;
   //delay (1000);
   try {
            Thread.sleep(1000); //
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
```

```
  if (correo3<=0){
  text(" " ,380,625);


  }
  else{
text("Enviando correo en , " + correo3 + "  segundos",380,640);
  }


  if(sensorpesolibras<1 && correo3==0){
    print("Se detecto carencia de producto , enviando correo");



    JFrame frame7 = new JFrame();
       frame7.setContentPane(new JOptionPane("Se detecto Carencia de
embutidos ,enviando correo..."));
       frame7.setSize(100, 100);
       frame7.setLocationRelativeTo(null);
       frame7.setVisible(true);


       frame7.pack();
       frame7.setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);


       try {
           Thread.sleep(2000);
       } catch (InterruptedException e) {
           e.printStackTrace();
       }
       frame7.setVisible(false);
    sendMail();          // llamada al metodo de envio de correo
    variable_logs=1;
    Logs();
```

```
    }
    }

    if (sensorpesolibras>=1) {
  text("El producto esta en su lugar  "+ sensorpesolibras + "lbs",380,600 ) ;
// mensaje en interfaz grafica
  correo3=10;

    }

 //--------------------------------COMANDOS DEL SENSOR DE MOVIMIENTO-------
-----------------------------------------------
  if (variable==1 && variable2==1){

     JFrame frame = new JFrame();
       frame.setContentPane(new JOptionPane("Se detecto Carencia de huevos
,enviando correo..."));
       frame.setSize(100, 100);
       frame.setLocationRelativeTo(null);
       frame.setVisible(true);

       frame.pack();
       frame.setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);

       try {
           Thread.sleep(3000);
       } catch (InterruptedException e) {
           e.printStackTrace();
       }
       frame.setVisible(false);
  }
```

```
 if (variable==1 && variable2==1){

  println("Se detecto carencia de huevos , Enviando correo");
  int correo=1;
  //delay(1000);
  try {
        Thread.sleep(100);
     } catch (InterruptedException e) {
        e.printStackTrace();
     }

  if (correo==1){

    sendMail();
    correo=0;
    variable=0;
    variable2=0;
    variable_logs=2;
    Logs();



  }
}
    if (sensorpir == 1){
    //delay(500);
    try {
        Thread.sleep(100);
     } catch (InterruptedException e) {
```

```
        e.printStackTrace();
    }
 variable=1;
 }


  if (sensorpir2 == 1){

 //delay(500);
 try {
        Thread.sleep(100);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }

 variable2=1;
 }



 if (variable==1){
 text("El huevo se ha retirado ",380,300);
ventana=1;

 }else{
 text("El huevo sigue en la refrigeradora",380,300);
      }
 if (variable2==1){

 text("El huevo se ha retirado ",380,400);
```

```
    ventana2=1;
     }else{
     text("El huevo sigue en la refrigeradora",380,400);

     }
 //----------------------------------COMANDOS DEL SENSOR DE DISTANCIA----------
---------------------------------------------------

    if (sensorultrasonico >= 29){
    text("El agua se encuentra en niveles bajos :",380,500);
    correo2--;
    //delay(1000);
    try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }



    if(correo2<=0){
      text("",380,525);
    }

    else{
text("Enviando correo en , " + correo2 + "  segundos",380,545);
   }

 if ( sensorultrasonico>=29 && correo2==0){
   print("Se detecto carencia de producto , enviando correo");
 JFrame frame100 = new JFrame();
```

```
        frame100.setContentPane(new JOptionPane("Se detecto que el agua se
retiro ,enviando correo..."));
        frame100.setSize(100, 100);
        frame100.setLocationRelativeTo(null);
        frame100.setVisible(true);

        frame100.pack();
        frame100.setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);

        try {
            Thread.sleep(2000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        frame100.setVisible(false);
        sendMail();
        variable_logs=3;
        Logs();
 }



    }else{
    text("El agua se encuentra en niveles aceptables :"+sensorultrasonico +
"cm",380,500);
    correo2=10;
    }

  //--------------------------------COMANDOS PULSADORES----------------------------
-----------------------------------------

  if (pulsador ==0 && pulsador2==0 && pulsador3==0){
  text("El usuario no ha pulsado ninguna alerta ",380,680);
```

```
 }


 if (pulsador==1){
  text("Alerta activada por el usuario , carencia de huevos ",380,680);
  print("Alerta activada por el usuario , enviando correo... ");
  change++;
  if (change==2)
  change--;
  ventana++;


 if(ventana==2){
  JFrame frame2 = new JFrame();
      frame2.setContentPane(new JOptionPane("Alerta activada por el usuario , enviando correo con carencia de huevos"));
      frame2.setSize(100, 100);
      frame2.setLocationRelativeTo(null);
      frame2.setVisible(true);
      frame2.pack();
      frame2.setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
      try {
          Thread.sleep(2000);
      } catch (InterruptedException e) {
          e.printStackTrace();
      }
      frame2.setVisible(false);
      ventana=0;
      sendMail();
      variable_logs=4;
      Logs();
 }
```

```
 }


 if (pulsador2 == 1){
   text("Alerta activada por el usuario , carencia de agua ",380,680);
   print("Alerta activada por el usuario , enviando correo... ");
   change2++;
   if (change2==2)
   change2--;
   ventana2++;


  if(ventana2==2){
  JFrame frame3 = new JFrame();
      frame3.setContentPane(new JOptionPane("Alerta activada por el usuario , enviando correo con carencia de agua"));
      frame3.setSize(100, 100);
      frame3.setLocationRelativeTo(null);
      frame3.setVisible(true);
      frame3.pack();
      frame3.setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
      try {
          Thread.sleep(2000);
      } catch (InterruptedException e) {
          e.printStackTrace();
      }
      frame3.setVisible(false);
      ventana2=0;
      sendMail();
      variable_logs=5;
      Logs();
 }
 }
```

```
  if (pulsador3 == 1){
  print("Alerta activada por el usuario , enviando correo... ");
  text("Alerta activada por el usuario , carencia de embutidos ",380,680);
  change3++;
  if (change3==2)
  change3--;
  ventana3++;
  if(ventana3==2){
  JFrame frame4 = new JFrame();
      frame4.setContentPane(new JOptionPane("Alerta activada por el usuario , enviando correo con carencia de embutidos"));
      frame4.setSize(100, 100);
      frame4.setLocationRelativeTo(null);
      frame4.setVisible(true);
      frame4.pack();
      frame4.setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);

      try {
          Thread.sleep(2000);
      } catch (InterruptedException e) {
          e.printStackTrace();
      }
      frame4.setVisible(false);
   ventana3=0;
   sendMail();
   variable_logs=6;
   Logs();
  }
 }
```

```
  //-------------------------------Impresion de datos por consola----------------------------
---------------------------------
  println("Lectura sensor de distancia = "+ sensorultrasonico + "cm");
  println("Lectura sensor de movimiento N°1= "+ variable);
  println("Lectura sensor de movimiento N°2= "+ variable2);
  println("Lectura sensor de peso= "+ sensorpesolibras + "lbs");
  println("Lectura de pulsador =" + pulsador);
  println("Lectura de pulsador =" + pulsador2);
  println("Lectura de pulsador =" + pulsador3);



  println("---------------------------------");
}




//Funcion para que el usuario sea validado
public class Auth extends Authenticator
{
    public Auth()
    {
      super();
    }
    public PasswordAuthentication getPasswordAuthentication()
    {
      String username, password;
```

```
        username = "informacion.faltante.udla@gmail.com";
        password = "laelegancia";
        //System.out.println("authenticating. . ");
        return new PasswordAuthentication(username, password);
    }
}

//Esta función es la que se encarga de enviar el correo

void sendMail() {
    // Create a session
    String host="smtp.gmail.com";
    Properties props=new Properties();
    // SMTP Session
    props.put("mail.transport.protocol", "smtp");
    props.put("mail.smtp.host", host);
    props.put("mail.smtp.port", "587");
    props.put("mail.smtp.auth", "true");

   props.put("mail.smtp.starttls.enable","true");

   Session session = Session.getDefaultInstance(props, new Auth());
   try  {

     MimeMessage message = new MimeMessage(session);

     message.setFrom(new
InternetAddress("informacion.faltante.udla@gmail.com", "Informacion de
productos faltantes"));

     message.setRecipients(Message.RecipientType.TO,
InternetAddress.parse("supermercado.udla@gmail.com", true));
```

```
if (pulsador ==1 ){
message.setSubject("Alerta activada por el usuario(HUEVOS)");
message.setText("El usuario ha presionado el boton instalado en el prototipo , indicando la carencia de huevos");
pulsador=0;
Transport.send(message);
}

if (variable==1 && variable2==1){
message.setSubject("Ausencia de huevos");
message.setText("Han sido retirados los huevos de la refrigeradora , por favor llevar nuevos para la reposicion");
Transport.send(message);
}

if (pulsador3 ==1 ){
message.setSubject("Alerta activada por el usuario (EMBUTIDOS)");
message.setText("El usuario activo las alerta de carencia del producto");
pulsador3=0;
Transport.send(message);
}

float sensorpesolibras=(sensorpeso/100);
if (sensorpesolibras<1 && correo3==0){

message.setSubject("Ausencia de Carne");
message.setText("Estimado Cliente , el sensor instalado ha detectado que el frasco con carne ha sido retirado o que se encuentra en el limite ");
Transport.send(message);
}
```

```
      if (pulsador2 ==1 ){
      message.setSubject("Alerta activada por el usuario (AGUA)");
      message.setText("El usuario activo el boton de la refrigeradora indicando
que el agua ha terminado");
      pulsador2=0;
      Transport.send(message);
      }

      if (sensorultrasonico >=29){
      message.setSubject("Ausencia de Agua");
      message.setText("Estimado Cliente , el sensor instalado ha detectado que
el agua en el envase esta en niveles bajos");
      Transport.send(message);
      }

    }  catch(Exception e)  {
      e.printStackTrace();
    }

}

void Logs(){

  logs.print(year()+"/"+month()+"/"+day()+"  " );
```

```
  if (minute()<10 && second()<10){
  logs.print(hour()+":"+"0"+minute()+":"+"0"+second()+"     ");
  }else if(minute()<10){
  logs.print(hour()+":"+"0"+minute()+":"+second()+"     ");
  }
  else if(second()<10){
  logs.print(hour()+":"+minute()+":"+"0"+second()+"     ");
  }



  else{


  logs.print(hour()+":"+minute()+":"+second()+ "    ");
  }
  if(variable_logs==1){
  logs.println("EL sensor de peso detecto que el producto se retiro o que tiene un peso menor a 1 libra");
  variable_logs=0;
  }
  if(variable_logs==2){
  logs.println("EL sensor de movimiento detecto que el producto que los huevos han sido retirados");
  variable_logs=0;
  }
  if(variable_logs==3){
  logs.println("EL sensor de ultrasonico detecto que el agua fue retirada o se encuentra con un nivel bajo de liquido");
  variable_logs=0;
  }
  if(variable_logs==4){
  logs.println("El puslador fue presionado indicando carencia de huevos ");
```

```
  variable_logs=0;
  }
  if(variable_logs==5){
  logs.println("El puslador fue presionado indicando carencia de agua");
  variable_logs=0;
  }
  if(variable_logs==6){
  logs.println("El puslador fue presionado indicando carencia de peso");
  variable_logs=0;
  }

  logs.flush();

}

void Cerrar(){
exit();
}
```