PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE INGENIERÍA

MAESTRÍA EN REDES DE COMUNICACIONES

INFORME FINAL CASO DE ESTUDIO PARA UNIDAD DE TITULACIÓN ESPECIAL

TEMA:

"ESTUDIO DE LAS COMUNICACIONES INALÁMBRICAS BASADAS EN EL ESTÁNDAR IEEE 802.11 PARA LA IMPLEMENTACIÓN DE COMUNICACIONES CON DISPOSITIVOS DE DOMÓTICA - CASO DE ESTUDIO DISPENSAR ALIMENTOS AUTOMÁTICOS."

ROLANDO FABRICIO LARA NARANJO

Quito – 2016

AUTORÍA

Yo, ***Rolando Fabricio Lara Naranjo***, portador de la cédula de ciudadanía No.***0503266827***, declaro bajo juramento que la presente investigación es de total responsabilidad del autor, y que se ha respetado las diferentes fuentes de información realizando las citas correspondientes. Esta investigación no contiene plagio alguno y es resultado de un trabajo serio desarrollado en su totalidad por mi persona.

_____

Rolando Fabricio Lara Naranjo

Contenido

Índice de Figuras

Índice de Tablas

1. Introducción

Este trabajo presenta, un sistema prototipo de un dispensador de alimentos automático para mascotas con envió de información de su funcionamiento y su entorno. El fin principal es realizar un estudio de las comunicaciones inalámbricas basada en el estándar IEEE 802.11 para implementarlos en dispositivos utilizados para domótica que satisfaga las necesidades del usuario utilizando el concepto de “Internet of things (IoT)” en español “El Internet de las cosas” esto se refiere a la interconexión digital de objetos cotidianos con Internet. El prototipo poseerá un sistema de control conformado por sensores, actuadores y controladores lógicos de tal manera que este actué al realizar una configuración inicial utilizando un dispositivo remoto con aplicación Android la cual se comunicará mediante un enlace de red inalámbrica basada en el estándar IEEE 802.11, la configuración inicial pedirá al usuario información de su mascota como: nombre, raza y edad; una vez realizada la configuración inicial, el prototipo se encontrará listo para su funcionamiento y de acuerdo a la configuración dada dispensará alimento cada cierto periodo del día, el prototipo también será capaz de enviar notificaciones en tiempo real al usuario mediante la aplicación Web creada por MICROBLOGGING llamada TWITTER, estas notificaciones informarán al usuario cuando se haya realizado el depósito de alimento, cuando el contenedor de almacenaje este por terminarse, la temperatura ambiente del área en la que se encuentra el dispensador, etc.

Este trabajo persigue desarrollar un prototipo que permita al usuario organizar y optimizar mejor su tiempo dejando de lado tareas como la alimentación de mascotas y al mismo tiempo obtener recordatorios de compra de alimento; cabe destacar que el principio de este prototipo se puede realizar para sistemas dispensadores Industriales y que en el caso de que la

red local falle el dispensador seguirá realizando sus funciones en ese tiempo y de esta manera no afecte considerablemente el funcionamiento normal del prototipo.

La primera parte de este Caso de Estudio de la Maestría en Redes de Telecomunicaciones presenta una descripción del funcionamiento del prototipo, para luego describir la justificación de realizarlo. Una tercera parte señala los antecedentes, la cuarta parte está contenida los objetivos; el diseño y construcción del Sistema Prototipo se encuentra en una quinta parte, los resultados obtenidos se encuentra en la sexta parte, conclusiones y recomendaciones en la séptima; y en la octava parte se encuentra los anexos. Es importante señalar que la bibliografía de este documento está sólidamente sustentada.

2. Justificación

La tecnología se encuentra en todos los ámbitos que se utiliza a diario y ayuda a que las actividades sean mucho más fáciles y rápidas de realizar, es por eso que la tecnología ha llegado a facilitar el entorno de edificios y hogares a lo que se le denomina Domótica, lo cual es un campo de la electrónica que ayuda a crear sistemas automatizados para controlar determinadas tareas, en el Ecuador este campo tecnológico no es explotado al máximo por ser en muchos casos costoso.

La tecnología se encuentra también en los teléfonos móviles en los cuales existen aplicaciones desarrolladas por diferentes sistemas operativos como Android, estas aplicaciones se utilizan para muchas actividades diarias desde el control de distancia recorrida hasta controlar dispositivos de forma remota. Para el estudio del presente proyecto se desarrollará una aplicación bajo la plataforma Android que permita la comunicación con el prototipo dispensador de alimento mediante el estándar IEEE 802.11 y con esto enviar y recibir información del estado del prototipo.

El proyecto pretende realizar un estudio sobre las comunicaciones del estándar IEEE 802.11 implementado en un prototipo que cumpla una actividad diaria como es dispensar alimento para mascotas de forma automatizada y a la vez mantener informado al usuario sobre las actividades que realiza el prototipo. El proyecto prioriza la administración del tiempo y elimina una actividad que debe realizar diariamente el usuario.

3. Antecedentes

En la actualidad la domótica está teniendo un gran espacio en nuestro país en muchas aplicaciones para solventar las necesidades del ser humano en lo referente al ahorro de energía, protección de viviendas e infraestructuras, realizar tareas tanto arduas como monótonas y aumentar el confort del ser humano.

Existen proyectos realizados en años anteriores sobre dispensadores de alimento como por ejemplo “intervención remota a un espacio domotizado para mascotas” en el cual se utiliza equipos costosos para realizar este tipo de proyectos lo cual no es viable y a la vez no cuenta con el estándar IEEE 802.11 el cual se lo va a incorporar en el proyecto propuesto para de esta manera realizar un estudio del comportamiento de este estándar implementado en dispositivos de domótica. (ZAPATA, 2013)

Existen varias plataformas de hardware desarrolladas para este objetivo como por ejemplo la de Arduino la cual se utilizará con un procesador central (ARDUINO, ARDUINO GENUINO, 2015); el hardware seleccionado dependerá de las funciones, de las entradas y salidas tanto analógicas como digitales que se requieren para realizar el control de la actividad que van a realizar; de la misma manera se deberá elegir sensores o posicionadores para verificar el nivel del contenedor de alimento con lo cual se enviará una señal de alerta al controlador cuando el contenedor de alimento este por terminarse y un sensor de temperatura para verificar el entorno en el que se encuentra el dispensador (GONZALEZ, 2014). Se utilizará un módulo de red del estándar IEEE 802.11 (ARDUINO, ARDUINO GENUINO, 2015) con el cual se realizará la comunicación general con todas las plataformas que van a intervenir utilizando tecnología Arduino que incorpora microcontroladores donde se realizará un programa completo (ARDUINO, ARDUINO GENUINO, 2015).

4. Objetivos

4.1. Objetivo General:

Diseñar y construir un prototipo para control y monitoreo de un dispensador inteligente de alimento, empleando comunicación inalámbrica estándar IEEE 802.11 que permita evaluar su comportamiento, optimizando el tiempo del usuario e innovando tecnológicamente la infraestructura en Sistemas Domóticos.

4.2. Objetivos Específicos:

- ✓ Diseñar y construir un prototipo dispensador de alimento automatizado utilizando un módulo lógico programable que permita eliminar la intervención humana en la tarea de alimentación de mascotas.
- ✓ Diseñar y construir un sistema de comunicación inalámbrica estándar IEEE 802.11 que permita realizar un monitoreo remoto del prototipo dispensador de alimento mediante el envío de notificaciones.
- ✓ Diseñar y construir un sistema para medir la temperatura y humedad del entorno del prototipo dispensador de alimento que permita el envío de notificaciones al usuario de la información obtenida.
- ✓ Desarrollar una aplicación móvil basada en tecnología Android que permita la comunicación con el prototipo dispensador de alimento para realizar configuraciones al sistema para un óptimo funcionamiento.
- ✓ Evaluar el comportamiento de la comunicación inalámbrica estándar IEEE 802.11 para monitoreo y control del prototipo dispensador de alimento por un lapso continuo que permita obtener datos relevantes de las acciones del estándar en sistemas Domóticos.

5. Desarrollo Caso de Estudio

Es importante conocer los elementos con los cuales se trabajara y desarrollara el prototipo, a continuación se menciona cada uno de estos al igual que sus características técnicas, cabe mencionar que los elementos a utilizarse son utilizados en muchos casos en la rama de la Robótica y la Electrónica lo que aumenta el interés al utilizar este tipo de elementos para la creación de un sistema Domótico.

5.1. Diseño del Dispensador Inteligente

Es sistema contará con un controlador programable el cual será el cerebro del Sistema, según la programación realizada este controlador tomará determinadas decisiones para cubrir con las necesidades demandadas, a su vez este controlador estará conectado a un sensor de Nivel el cual indicará al controlador el estado del contenedor de alimento, de igual manera el controlador estará conectado a un sensor de Temperatura y Humedad el cual enviará información al controlador del entorno del prototipo, el controlador también estará conectado a un sistema de medición de tiempo el cual indicará la hora y el día actual, un sistema actuador estará conectado al controlador para realizar el depósito del alimento y por ultimo un sistema de comunicaciones estará conectado al controlador el cual tendrá la función de comunicarse con la aplicación móvil y con internet para enviar notificaciones al usuario, en la Figura 1 se indica en un diagrama de bloques como están conectados los Sistemas .

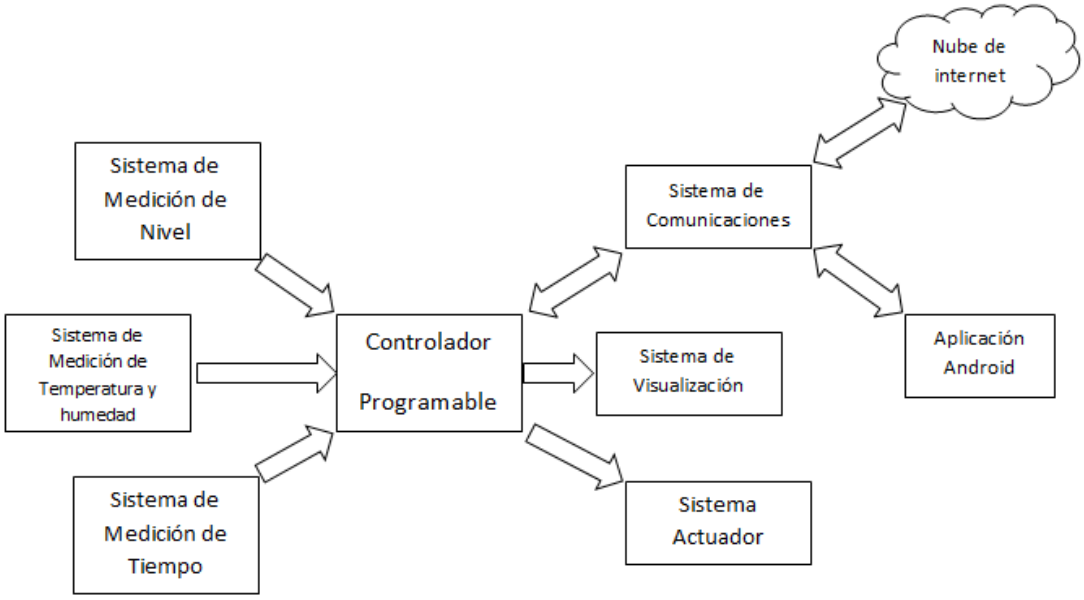


Figura 1. Diagrama de Bloques del Sistema prototipo dispensador de alimento automatizado. Fuente: El Autor.

5.1.1. Diseño del Case del Dispensador

Se realizó un modelado en AutoCAD 3D para visualizar de una manera intuitiva como quedará el trabajo final y de esa manera obtener sus medidas y partes para su ensamblaje, a continuación se indica el modelado 3D.

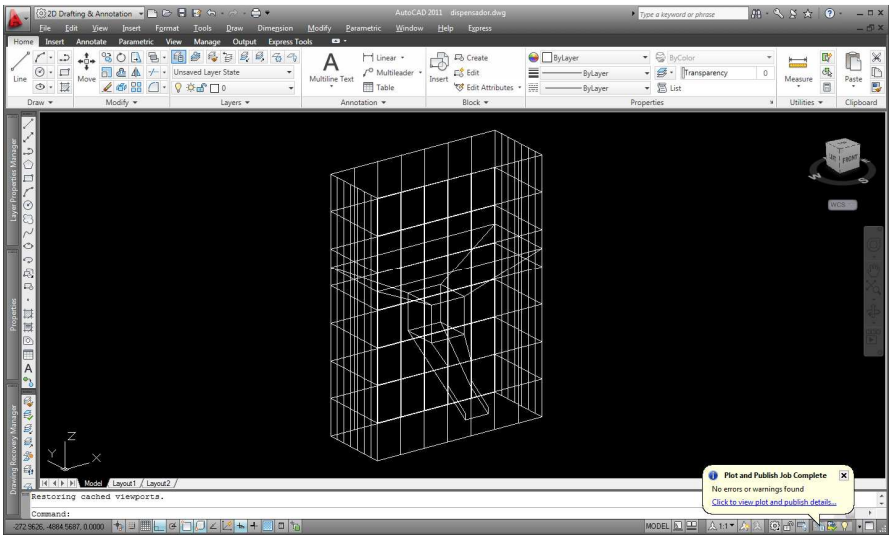


Figura 2. Modelado 3D del Dispensador en AutoCAD 3D. Fuente: El Autor.

La Figura 3 te indica todas las vistas del prototipo para facilitar su construcción.

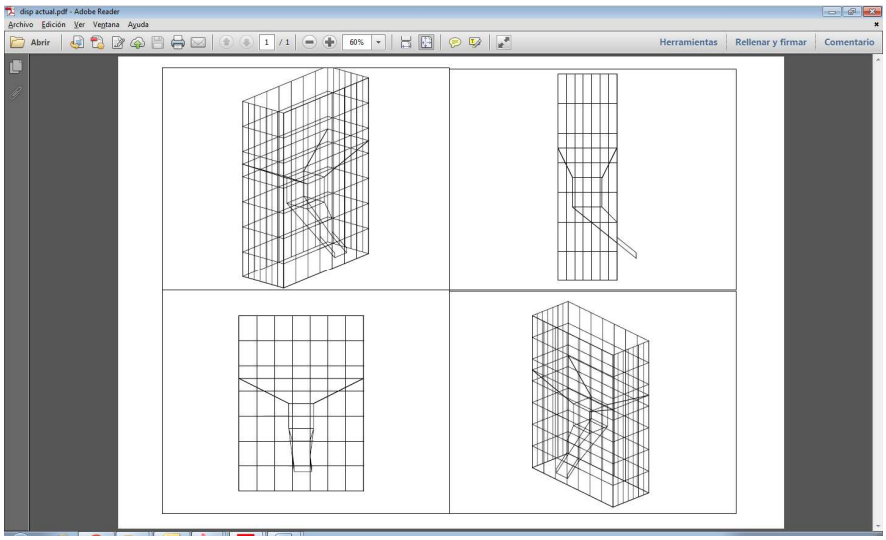


Figura 3. Modelado 3D del Dispensador en AutoCAD 3D. Fuente: El Autor.

La Figura 4 indica las medidas de cada una de las piezas que se necesita para la construcción del dispensador.

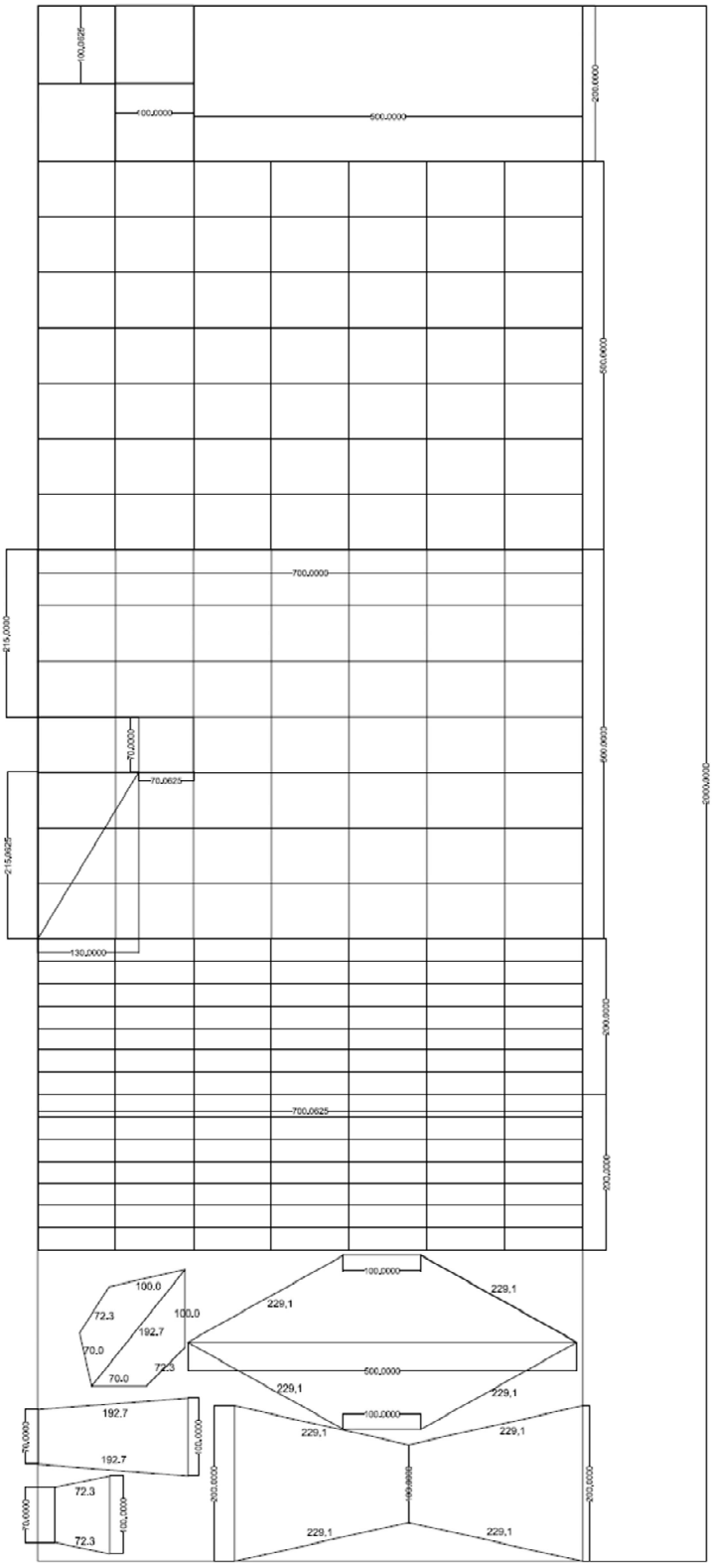


Figura 4. Medidas del Dispensador. Fuente: El Autor.

5.1.2. Diseño del Sistema Controlador Programable

Para el Sistema Controlador Programable se utilizó tecnología Arduino específicamente se utilizó un controlador Arduino Mega 2560, el cual cubre todas las necesidades tanto de entradas digitales, salidas digitales y comunicación utilizando el estándar IEEE 802.11, a continuación se detalla las características más relevantes de este controlador.

5.1.2.1. Arduino Mega 2560.

Arduino Mega es una placa electrónica con un micro controlador ATMega 2560 con todo lo necesario para dar el soporte necesario para proyectos capaces de comunicarse con un software instalado en tu computadora.

Arduino es una plataforma “open-source” basada en una placa de Entradas/Salidas y un entorno de desarrollo que implementa el lenguaje Processing / Wiring. Arduino se puede utilizar para desarrollar objetos interactivos autónomos o puede ser conectado a un equipo por medio de Software. (PIDELECTRONICS, 2015).

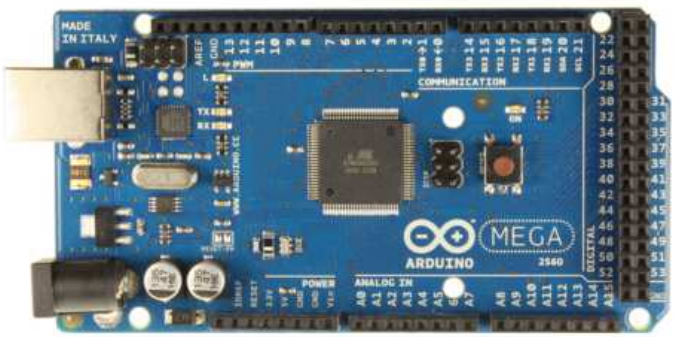


Figura 5. Placa Electrónica Arduino Mega 2560. Fuente: http://flipmu.com/work/chronome/changing-the-serial-number/

5.1.2.2. Características Arduino Mega 2560.

El Arduino Mega es una placa electrónica basada en los ATmega2560. Lleva 54 Entradas /Salidas digitales (de los cuales 14 pueden utilizarse para salidas PWM), 16 entradas analógicas, 4 UARTs (puertas seriales), un oscilador de 16MHz, una conexión USB, un conector de alimentación, un header ICSP y un botón de reset. Contiene todo lo necesario para soporte del microcontrolador, simplemente como conectarlo a un ordenador con un cable USB, o alimentarlo con un adaptador de corriente AC a DC para empezar a utilizarlo. (PIDELECTRONICS, 2015)

En la Tabla 1 se detalla las características del Arduino Mega 2560.

Tabla 1. Especificaciones Arduino Mega 2560. Fuente: www.arduino.cc/en/Main/arduinoBoardMega&prev=search

ESPECIFICACIONES ARDUINO MEGA 2560	
Microcontroladores	ATmega2560
Tensión de funcionamiento	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (límites)	6-20V
Digital pines I / O	54 (de las cuales 15 proporcionan salida PWM)
Pines de entrada analógica	16
Corriente DC por E / S Pin	40 mA
Corriente DC de 3.3V Pin	50 mA
Memoria flash	128 KB de los cuales 4 KB utilizado por el gestor de arranque
RAM	8 KB
EEPROM	4 KB
Velocidad de reloj	16 MHz

Alimentación

El Arduino Mega puede ser alimentado a través de la conexión USB o con una fuente de alimentación externa. La fuente de alimentación se selecciona automáticamente.

Potencia (no USB) externo puede venir con un adaptador de CA a CC o batería. El adaptador se puede conectar al conectar un enchufe de 2,1 mm de centro-positivo en el conector de alimentación de la placa. En las conexiones de una batería se pueden insertar en los cabezales de pin GND y Vin del conector de alimentación.

La placa puede funcionar con un suministro externo de 6 a 20 voltios. Si se suministra con menos de 7V, el pin de 5V puede suministrar menos de cinco voltios y la placa puede ser inestable. Si se utiliza más de 12V, el regulador de voltaje se puede sobrecalentar y dañar la placa. El rango recomendado es de 7 a 12 voltios (ARDUINO, ARDUINO GENUINO, 2015).

Los pines de alimentación son los siguientes:

- **VIN.** La tensión de entrada a la placa Arduino cuando se utiliza una fuente de alimentación externa (por oposición a 5 voltios de la conexión USB u otra fuente de alimentación regulada). Usted puede suministrar tensión a través de este pin.
- **5V.** La fuente de alimentación regulada utilizada para alimentar el microcontrolador y otros componentes de la placa. Esto puede venir de VIN a través de un regulador de tensión o puede ser suministrada por USB u otra fuente de 5V regulada.
- **3V3.** Un suministro de 3,3 voltios generada por el chip FTDI instalado. Su corriente máxima es de 50 mA.
- **GND.** Pines de tierra.

Memoria

El ATmega1280 tiene 128 KB de memoria flash para almacenar el código de programación (de los cuales 4 KB se utiliza para el cargador de arranque), 8 KB de SRAM y 4 KB de EEPROM (que puede ser leído y escrito con la biblioteca EEPROM) (ARDUINO, ARDUINO GENUINO, 2015)

Comunicación

El Arduino Mega tiene una serie de instalaciones para la comunicación con un ordenador, otro Arduino, u otros microcontroladores. El ATmega1280 ofrece cuatro UART hardware para TTL (5V) de comunicación serie. Un FT232RL FTDI en los canales de mesa uno de ellos a través de USB y los drivers FTDI (incluido con el software de Arduino) proporcionan un puerto de comunicación virtual para el software en el ordenador. El software de Arduino incluye un monitor de serie que permite a los datos de texto simples para ser enviados hacia y desde la placa Arduino. Los LEDs RX y TX de la placa parpadean cuando se están transmitiendo datos a través del chip y conexión USB FTDI al ordenador (pero no para la comunicación en serie en los pines 0 y 1). Una biblioteca SoftwareSerial permite la comunicación en serie en cualquiera de los pines digitales de la placa Mega (ARDUINO, ARDUINO GENUINO, 2015).

Programación

El Arduino Mega se puede programar con el software de Arduino. Los ATmega1280 en el Arduino Mega viene precargado con un gestor de arranque que le permite cargar nuevo código a él sin el uso de un programador de hardware externo. Se comunica mediante el protocolo original STK500 (ARDUINO, ARDUINO GENUINO, 2015).

Características físicas y Escudo de compatibilidad

La longitud máxima y la anchura de la placa del Arduino Mega son 4 y 2,1 pulgadas, respectivamente, con el jack conector USB y el poder que se extiende más allá de la dimensión anterior. Tres orificios en la placa permiten fijarse a cualquier superficie. Tenga en cuenta que la distancia entre los pines digitales 7 y 8 es de 160 milésimas de pulgada (0,16"), no un múltiplo par de la separación de 100 milésimas de pulgada de los otros pasadores.

Los Pines digitales 0 a 13 (y la AREF adyacente y pines GND), entradas analógicas 0 a 5, en la cabecera de la energía, y cabecera ICSP están en lugares equivalentes. Además, la UART principal (puerto serie) se encuentra en los mismos pines (0 y 1), así como interrupciones externas 0 y 1 (pines 2 y 3), respectivamente (ARDUINO, ARDUINO GENUINO, 2015).

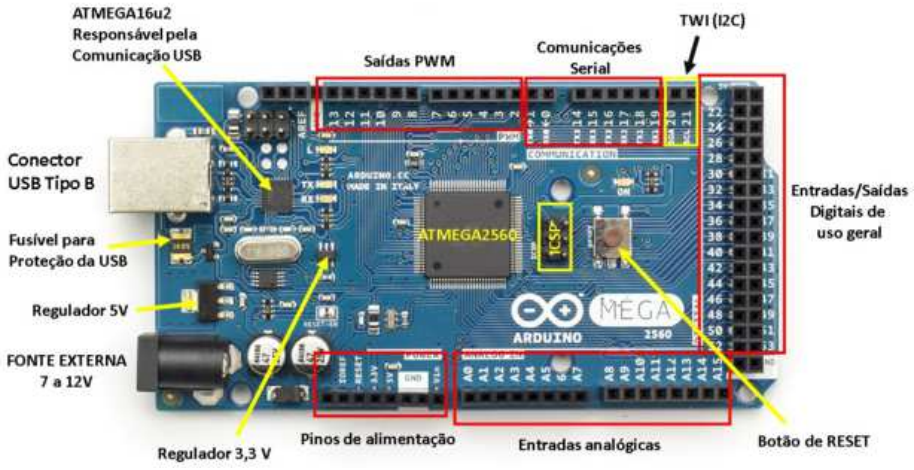


Figura 6. Componentes de la Placa Arduino Mega 2560. Fuente: http://www.embarcados.com.br/arduino-mega-2560/

5.1.3. Diseño del Sistema de Medición de Nivel

Para el diseño se consideró utilizar un final de carrera para la medición de nivel en el contenedor es así que se utilizó el CQC 1E4-T125 el cual mediante un conexionado se conectará al sistema controlador el cual realizará un evento al activarse el sistema de medición de nivel.

El evento a realizar es dar una señal al controlador para q este envié un mensaje vía Twitter al usuario indicando que el contenedor de alimento está por terminarse.

5.1.3.1. Final de Carrera

Los interruptores final de carrera (FdC) son interruptores (switch) convencionales cuya única diferencia con éste último radica en dónde se lo coloca.

Un final de carrera se lo coloca, justamente, al final de un desplazamiento mecánico, y antes que se active el último interruptor que es la parada de emergencia.

Describiendo los FdC de un modo más técnico, son sensores de contacto que muestran una señal eléctrica, ante la presencia de un movimiento mecánico.

Son utilizados ampliamente en ambientes industriales para avisar de la presencia de objetos en una posición específica.

Se utilizan en diversas aplicaciones: Pueden determinar la presencia, ausencia, paso y posicionamiento de un objeto.

En un comienzo se los utilizaba para definir el final del recorrido de un objeto, de ahí que se llamen “interruptores de final de carrera” (ELECTRONICA, 2015).

Figura 7. Final de Carrera. Fuente: https://paletosdelaelectronica.wordpress.com/2015/01/25/interruptores-fin-de-carrera-i/

5.1.3.2. Funcionamiento

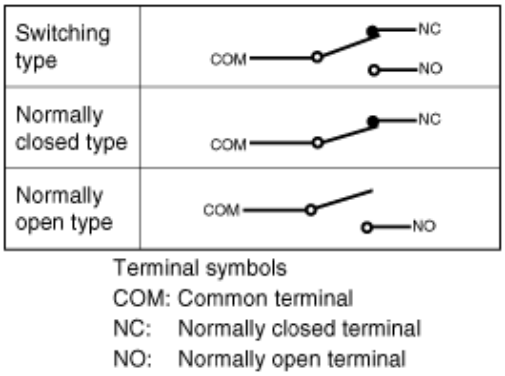
Los interruptores FdC constan de un accionador unido a una serie de contactos. Cuando un objeto entra en contacto con el accionador, el dispositivo activa los contactos para establecer o interrumpir una conexión eléctrica. Están compuestos por dos partes: un cuerpo donde se encuentran los contactos y una cabeza que detecta el movimiento.

Los interruptores de final de carrera están diseñados con dos tipos de cuerpo: enchufable y no enchufable.

Interruptores de carcasa enchufable:

Se abren por la mitad para acceder al bloque de terminales. Si el interruptor sufre daños o se desgasta, basta con quitar el cuerpo del interruptor con su cabeza y enchufar uno nuevo. No hace falta volver a realizar el cableado. Internamente pueden contener interruptores normalmente abiertos (NA o NO en inglés), cerrados (NC) o conmutadores dependiendo de la operación que cumplan al ser accionados (ELECTRONICA, 2015).

Tabla 2. Tipo de conexionados Fd C. Fuente: https://paletosdelaelectronica.wordpress.com/2015/01/25/interruptores-fin-de-carrera- i/



Composición interna:

Accionador: Es la parte del interruptor que entra en contacto con el objeto que se está detectando. Este tiene 2 posiciones, en reposo y posición de operación o punto de disparo. Puede ser:

- Lateral Rotatoria.
- De pulsación lateral o superior.
- De vástago oscilante o bigote de gato.

Cabeza: En la cabeza se encuentra el mecanismo que transforma el movimiento del accionador en movimiento de contacto. Cuando el accionador se mueve correctamente, el mecanismo acciona los contactos del interruptor.

Bloque de contactos: En el bloque de contactos se encuentran los elementos eléctricos de contacto del interruptor. Generalmente hay dos o cuatro pares de contactos.

Bloque de terminales: En el bloque de terminales se encuentran las terminales atornillables. Aquí se realiza la conexión eléctrica (por hilos) entre el interruptor y el resto del circuito de control.

Cuerpo del interruptor: En un interruptor enchufable, el cuerpo del interruptor aloja el bloque de contactos. En un interruptor no enchufable, encontrará el bloque de contactos y el bloque de terminales del interruptor.

Base: En un interruptor enchufable, la base aloja el bloque de terminales. Los interruptores no enchufables no tienen una base aparte (ELECTRONICA, 2015).

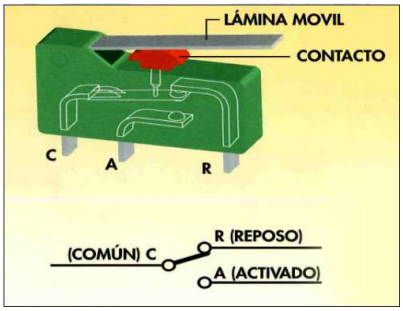


Figura 8. Composición Interna de un Final de Carrera. Fuente: http://www.comohacerturobot.com/Taller/taller-sensorMecanico.htm

5.1.4. Diseño del Sistema de Medición de Temperatura y Humedad

Para el diseño se utilizó el sensor de Temperatura y Humedad DHT11 por su amplio manejo y comunicación con el Arduino Mega 2560, este tipo de sensor tiene una comunicación serial con lo cual envía la información requerida para ser procesada por el controlador.

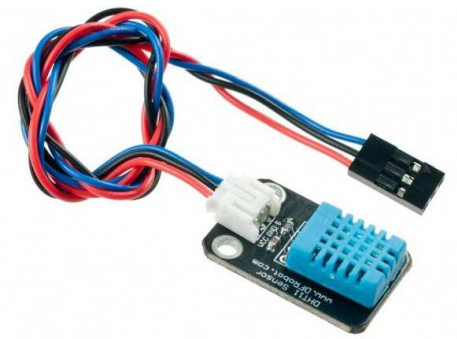


Figura 9. Sensor de Temperatura y Humedad DHT11. Fuente: http://www.tectronix.cl/sensor-de-temperatura-y-humedad-dht11.html#!prettyPhoto[gallery]/0/

5.1.4.1. Características.

Este sensor de temperatura y humedad DHT11 dispone de una salida calibrada de señal digital con la temperatura y el complejo sensor de humedad. Su tecnología garantiza la alta fiabilidad y una excelente estabilidad a largo plazo. Está conectado un microcontrolador de alto rendimiento de 8-bits. Este sensor incluye un elemento resistivo y una sensación de mojado NTC dispositivos de medición de temperatura. Tiene una excelente calidad, rapidez de respuesta, la capacidad anti-interferencia y altas ventajas de rendimiento de costos. Cada DHT11 sensores con calibración extremadamente precisa de la cámara de humedad de calibración (SINETCOM, 2015).

Los coeficientes de calibración almacenados en la memoria de programa OTP, sensores internos detectan señales en el proceso. El sistema de un solo alambre de interfaz está integrado a convertirse de forma rápida y fácil. Tamaño pequeño, de baja potencia, la distancia de transmisión de señal de hasta 20 metros, por lo que es una variedad de aplicaciones e incluso las aplicaciones más exigentes. El producto es un paquete de 4-pin en una sola fila. Conveniente conexión (SINETCOM, 2015).

Tabla 3. Especificación Técnica del sensor de Temperatura y Humedad DHT11. Fuente: http://www.tectronix.cl/sensor-de-temperatura-y-humedad-dht11.html

ESPECIFICACIÓN TÉCNICA DEL SENSOR DHT11	
Tensión de alimentación:	+5 V
Rango de temperatura :	0-50°C error de ± 2°C
Humedad:	20-90% RH Humedad ± 5% RH error
Interfaz:	Digital

5.1.5. Diseño del Sistema de Medición de Tiempo

Para el diseño se utilizó el módulo DS1307 con el cual se puede programar la fecha y hora actual para realizar un sistema que permita la mejor medición del tiempo con el cual se podrá realizar las acciones periódicas que se requieren para el prototipo.

5.1.5.1. Módulo DS1307

Este módulo de reloj en tiempo real está basado en tecnología de Semiconductores, es compatible con placas Arduino, viene totalmente ensamblado y listo para funcionar. Incluye una batería tipo botón de litio (CR1225 a 41mAh) que provee un funcionamiento aproximado durante un periodo de 9 años (generalmente 17 años) sin la alimentación de 5V. Este módulo presenta una interfaz de comunicación de protocolo I2C. Posee detección de caída de voltaje y la función de interruptor de la batería (VISTRONICA, 2015).

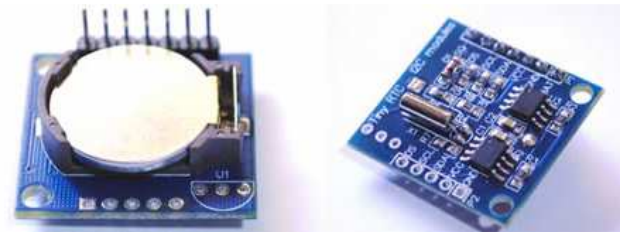


Figura 10. Módulo DS1307. Fuente: http://www.tuelectronica.es/tutoriales/arduino/reloj-rtc-i2c-con-arduino.html#

5.1.5.2. Características

A continuación se presenta un cuadro representativo sobre las características que tiene este módulo.

Tabla 4. Especificación Técnica del Sistema de Medición de Tiempo. Fuente: http://www.vistronica.com/modulos/modulo-reloj-en-tiempo-real-rtc-ds1307-detail.html

CARACTERÍSTICAS MÓDULO DS1307	
Memoria de almacenamiento	24C32 EEPROM 32KB I2C
Batería	LIR2032/CR1225 de litio (recargable)
Interfaz	I2C (controlar RTC DS1307 y AT24C32)
Función	Lectura/Escritura
Uso de tiempo	Alrededor de 1 año (carga completa)
Señal de reloj	Para el microcontrolador (salida de 1Hz para microcontroladores en cascada)
Conexión con otros dispositivos	I2C
Chip Calendario	BCD reloj de 56Bytes RAM no volátil/exacto hasta el 2100 (compensación de año bisiesto)
Cable de conexión	Transmisión de datos/cable serial con dos lí-neas de doble dirección.
Marcación	12 horas (AM/PM)/24 horas/segundo, minuto, hora, dí-a del mes, mes, dí-a de la semana, año
Consumo de energí-a de reserva	<500Ua
Tamaño	2.8cm x 2.5cm x 1.0cm
Peso	7 gramos

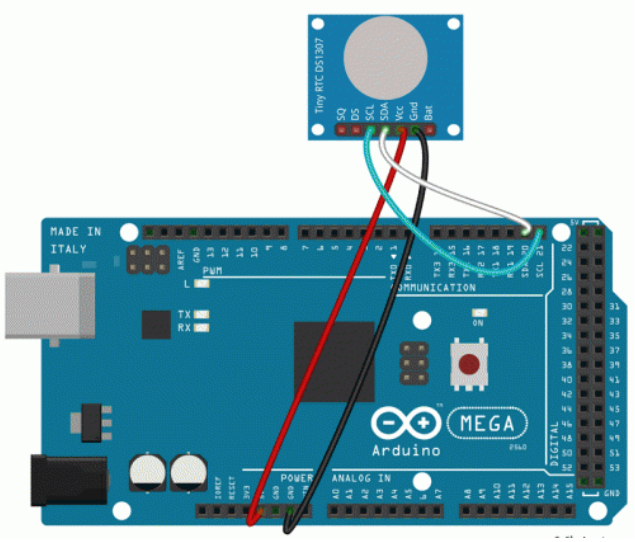


Figura 11. Conexión del Módulo DS1307 con Arduino Mega. Fuente: http://www.tuelectronica.es/tutoriales/arduino/reloj-rtc-i2c-con-arduino.html#

5.1.6. Diseño del Sistema Actuador

Para el prototipo se utilizó un servomotor con alto torque que cumpla con un movimiento fiable para suministrar alimento desde el contenedor, gracias a su fuerza y velocidad moderada el servomotor es un elemento óptimo para realizar este tipo de actividad a continuación se describe las características de este elemento.

5.1.6.1. Servomotor.

Un Servomotor es un dispositivo pequeño que tiene un eje de rendimiento controlado. Este puede ser llevado a posiciones angulares específicas al enviar una señal codificada. Con tal de que una señal codificada exista en la línea de entrada, el servo mantendrá la posición angular del engranaje. Cuando la señala codificada cambia, la posición angular de los piñones cambia. En la práctica, se usan servos para posicionar superficies de control como el movimiento de palancas, pequeños ascensores y timones. Ellos también se usan en radio control y por supuesto, en robots.

Los Servomotores son sumamente útiles en robótica. Los motores son pequeños, tiene internamente una circuitería de control interna y es sumamente poderoso para su tamaño. Un servo normal o Standard como el HS-300 de Hitec tiene 42 onzas por pulgada o mejor 3kg por cm. De torque que es bastante fuerte para su tamaño. También potencia proporcional para cargas mecánicas. Un servo, por consiguiente, no consume mucha energía. Se muestra la composición interna de un servomotor en el cuadro de abajo. Podrá observar la circuitería de control, el motor, un juego de piñones, y la caja. También puede ver los 3 alambres de conexión externa. Uno es para alimentación Vcc (+5volts), conexión a tierra GND y el alambre blanco es el alambre de control (ALBACETE, 2015).

Figura 12. Servomotor. Fuente: https://at89c52proyect.wordpress.com/2010/09/07/servomotores/

5.1.6.2. Funcionamiento.

El motor del servo tiene algunos circuitos de control y un potenciómetro (una resistencia variable) esta es conectada al eje central del servo motor. En la figura se puede observar al lado derecho del circuito. Este potenciómetro permite a la circuitería de control, supervisar el ángulo actual del servo motor. Si el eje está en el ángulo correcto, entonces el motor está apagado. Si el circuito chequea que el ángulo no es el correcto, el motor girará en la dirección adecuada hasta llegar al ángulo correcto. El eje del servo es capaz de llegar alrededor de los 180 grados. Normalmente, en algunos llega a los 210 grados, pero varía según el fabricante. Un servo normal se usa para controlar un movimiento angular de entre 0 y 180.

La cantidad de voltaje aplicado al motor es proporcional a la distancia que éste necesita viajar. Así, si el eje necesita regresar una distancia grande, el motor regresará a toda velocidad. Si este necesita regresar sólo una pequeña cantidad, el motor correrá a una velocidad más lenta. A esto se le llama control proporcional (ALBACETE, 2015).

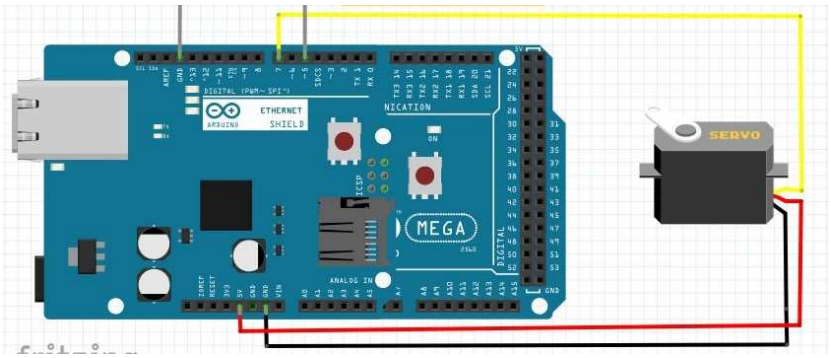


Figura 13. Esquema Gráfico conexión Arduino Mega 2560 - Servomotor. Fuente: http://wiznetmuseum.com/portfolio-items/arduino-ethernet-w5100-and-servo-motor/Funcionamiento.

5.1.7. Diseño del Sistema de Comunicaciones

Para el prototipo se decidió utilizar la comunicación estándar IEEE 802.11 obteniendo esta comunicación mediante un módulo WIFI ESP8266, gracias a su gran versatilidad y su amplio rango de cobertura es un módulo apto para el estudio que se va a realizar.

5.1.7.1. ESP8266 Modulo Wifi

Cabe mencionar que el Esp8266 es un micro en si con 16 pines GPIO. Otra característica en la teoría es su bajo consumo de energía, en modo sleep se habla de uA. Digo en teoría, porque para iniciar, el modulo llega a consumir 200mA, corriente que el Arduino no puede brindarle, y esto puede convertirse en un problema, aunque solucionable de momento con una fuente para protoboard (PATAGONIATEC, 2015).

5.1.7.2. Características

Este módulo incluye todo lo necesario para conectarse a un punto de acceso WIFI mediante comandos de texto AT, vía una puerta serie, que puede ser configurada a diferentes velocidades. Una vez que se lo programa para conectarse a la red WIFI, el modulo es capaz de enviar información que le se remite vía la puerta serie a una dirección IP y puerto que se desea. Cuando se trata de recibir, limpia todo el empaquetado TCPIP y reenvía por la puerta serie la información de datos limpia de polvo y paja, con lo que tiene la enorme virtud de permitir olvidar la gestión del TCPIP y de las demandas de procesador y memoria que suponen. A cambio no es exactamente una conexión WIFI, porque no se tiene acceso al stack o al socket IP pero para el Arduino esto es casi una ventaja (PATAGONIATEC, 2015).

Con una experiencia bastante más avanzada, más que nada en programación se puede hacer uso del microcontrolador que tiene integrado este poderoso módulo, el cual posee 16 GPIO (General Purpose Input Output), los cuales se pueden configurar como entradas o salidas, al igual que con Arduino, como interrupciones para sacar al módulo de modo sleep, y hasta posee un ADC. Los 16 pines, son en total lo que posee este microcontrolador, en algunos de ellos también se encuentran las comunicaciones SPI, UART, entre otras funciones (PATAGONIATEC, 2015).

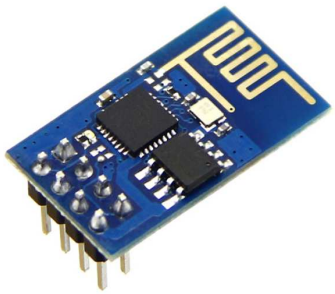


Figura 14. Módulo ESP8266. Fuente: http://saber.patagoniatec.com/esp8266-modulo-wifi/

A continuación se indican las principales características de este módulo que a pesar de su pequeño tamaño es un módulo muy poderoso para aplicaciones complejas.

Tabla 5. Especificación Técnica Modulo ESP8266. Fuente: http://saber.patagoniatec.com/esp8266-modulo-wifi/

ESPECIFICACIÓN TÉCNICA MÓDULO ESP8266	
Protocolos soportados:	802.11 b/g/n
Wi-Fi Direct (P2p), Soft Access Point	Si
Stack TCP/IP integrado	Si
PLL, reguladores y unidades de manejo de energía integrados	Si
Potencia de salida:	+19.5dBm en modo 802.11b
Sensor de temperatura integrado	Si
Consumo en modo de baja energía:	<10 uA
Procesador integrado de 32 bits, puede ser utilizado como procesador de aplicaciones	Si
Wifi 2.4 GHz, soporta WPA/WPA2	Si
Tamaño ultra reducido	(11.5mm x 11.5mm)
Conversor analógico a digital	10-bit
Soporta variedad de antenas	Si
Integrated low power	32-bit MCU
SDIO 2.0, SPI, UART, I2C	Si
Encendido y transmisión de datos	en menos de 2ms
Rango de operación	-40C° ~ 125C°

5.1.8. Diseño del Sistema de Visualización

Para el prototipo se utilizó un LCD 16X2 con un controlador I2C LCD con el cual se comunica con el controlador lógico Arduino Mega 2560 mediante comunicación I2C, a continuación se detalla las características de estos dos elementos utilizados.

5.1.8.1. LCD 16X2

Un display alfanumérico de matriz de puntos (dot-matrix) es un dispositivo de interfaz humano formado por una pantalla de cristal líquido o LCD (Liquid Crystal Display) sobre la que se pueden mostrar mensajes formados por distintos caracteres: letras, números, símbolos, etc. El primer dígito indica el número de filas del display y el segundo el número de columnas, es decir, 2×16 significa que tiene 2 filas y 16 columnas (SUAREZ, 2015). Ver Figura 15.

Figura 15. LCD 16x2. Fuente: http://www.instructables.com/id/Interfacing-16x2-LCD-with-msp430-launchpad-in-8-bi/

Estos dispositivos vienen gobernados por un controlador, que normalmente va incorporado sobre la misma placa de circuito impreso que soporta el LCD (Figura 16). En el mercado es muy habitual encontrarse con el controlador HD44780 de Hitachi. El controlador se encarga de gestionar el display líquido: polarizar los puntos de la pantalla, generar los caracteres, desplazar la pantalla, mostrar el cursor, etc. El usuario se despreocupa de todos estos problemas y simplemente necesita conocer una serie de comandos o instrucciones de alto nivel (limpia display, posiciona cursor, etc.) que le permitirán mostrar mensajes o animaciones sobre la pantalla de forma sencilla. Para comunicarse con el controlador del display se dispone de un

interfaz paralelo al exterior, de fácil conexión a otros microcontroladores o microprocesadores. (SUAREZ, 2015)

Figura 16. Vista trasera de un LCD alfanumérico. Fuente: http://eii.unex.es/profesores/jisuarez/descargas/ip/lcd_alfa.pdf

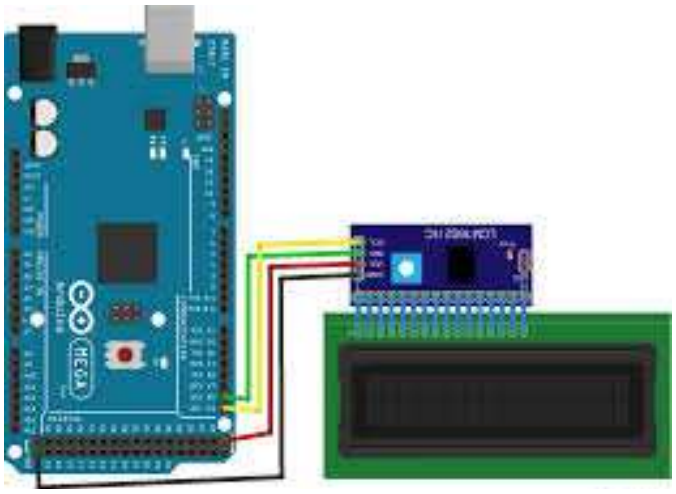


Figura 17. Esquema grafico conexión Arduino Mega 2560 – Controlador LCD I2C – LCD 16x2 . Fuente: http://eii.unex.es/profesores/jisuarez/descargas/ip/lcd_alfa.pdf

5.2. Construcción del Dispensador Inteligente

5.2.1. Construcción del Case del Dispensador

El case o la estructura del dispensador se construyó con plástico policarbonato por su fácil manipulación y resistencia, para su construcción se utilizó el modelado en AutoCAD 3D para obtener sus medidas y partes para su ensamblaje, a continuación se indica su ensamblaje.

Figura 18. Construcción del Dispensador. Fuente: El Autor.

**Las fotografías de su construcción se las puede observar en Anexos.*

5.2.2. Construcción del Sistema Controlador

Para el prototipo se utilizó la placa Arduino Mega 2560 el cual va a ser el cerebro del sistema controlador.

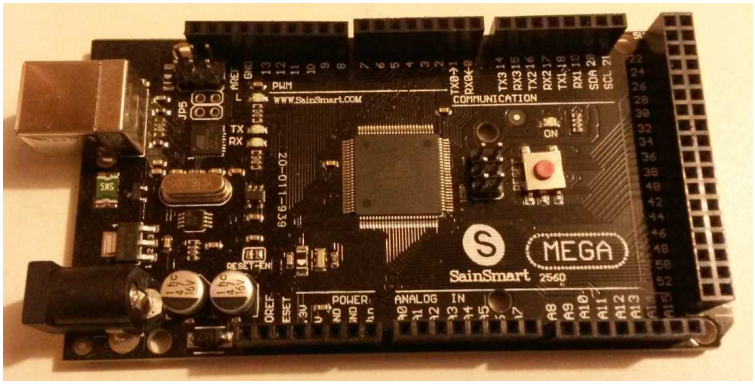


Figura 19. Arduino Mega 2560. Fuente: El Autor.

Se diseñó una placa adicional en la cual comprende los Sistemas de Medición de Nivel, Sistema de Medición de Temperatura y Humedad, Sistema de Medición de Tiempo, sistema

Actuador y el Sistema de Comunicaciones, en la Figura 19 se detalla los pines en los cuales se realizará el conexionado de los diferentes sistemas utilizados para el diseño del prototipo.

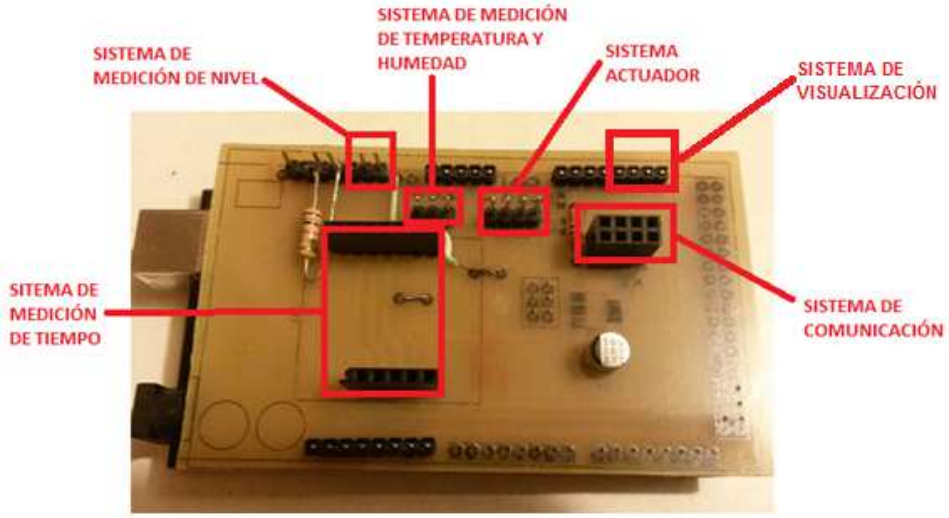


Figura 20. Placa Adicional conectada sobre la Placa Arduino Mega. Fuente: El Autor.

A continuación se encuentra detallada la construcción de cada uno de los sistemas antes mencionados, cada sistema se encuentra conectado a la placa adicional diseñada para este propósito con el objetivo de minimizar el espacio para la construcción del sistema controlador y actuador, esta placa a su vez se conecta con la placa Arduino mega en un coordinado espacio entre pines hembra y macho de la placa Arduino y la placa adicional respectivamente como se indica en la Figura 21.

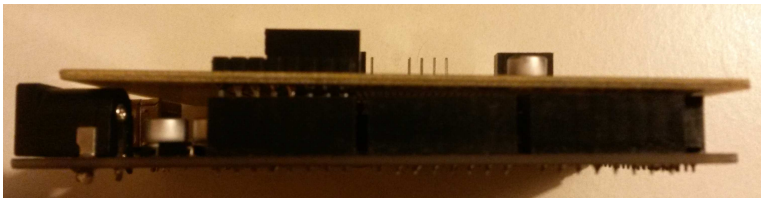


Figura 21. Placa Adicional conectada sobre la Placa Arduino Mega vista lateral. Fuente: El Autor.

5.2.3. Construcción del Sistema de Medición de Nivel

Para la conexión del sistema de medición de Nivel se utilizó los pines del fin de carrera (FdC) como se indica en la siguiente figura.

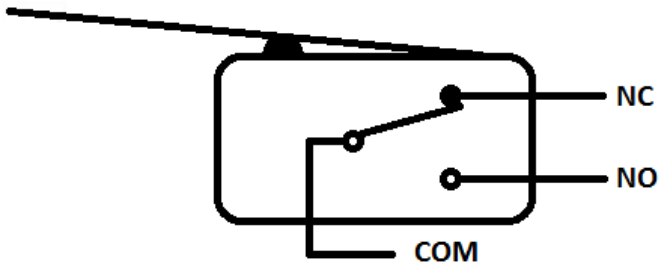


Figura 22. Pines final de carrera. Fuente: El Autor.

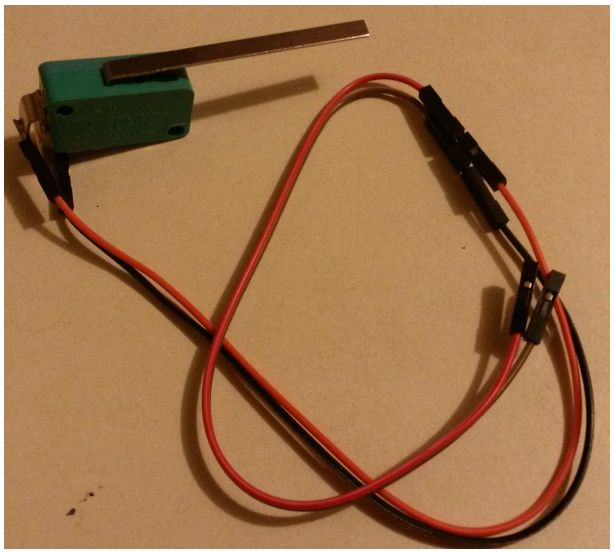


Figura 23. Final de carrera. Fuente: El Autor.

Se utilizó una placa adicional para la conexión de los pines de final de carrera que se conectan directamente a la placa Arduino Mega el pin NC se conectara al pin 12.

El conexionado del FdC se lo realizó como se indica en la Figura 24, se utilizó cables especiales con conexión hembra en el lado del FdC y a su vez una conexión tipo macho en la placa adicional para que el conexionado se lo realice de forma fácil para un mantenimiento a

futuro, es decir si el FdC falla se lo puede sustituir por otro siempre y cuando tenga la misma configuración es decir la pata de normalmente cerrado (NC) conectado a la placa adicional.

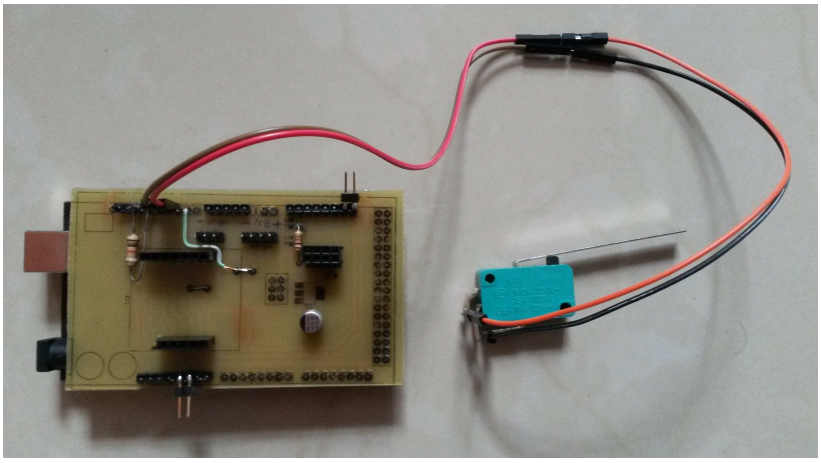


Figura 24. Conexionado del final de carrera en la placa adicional. Fuente: El Autor.

El final de carrera se lo colocó en la esquina inferior dentro del contenedor de alimento como se indica en la Figura 26 y su función será activar una señal que se enviará a la placa Arduino pasando por la placa adicional para indicar que el contenedor de alimento está por terminarse, esto lo realiza de la siguiente manera: cuando el contenedor está lleno el alimento mantiene presionado el contacto del final de carrera pero cuando el alimento disminuye el alimento deja de presionar el contacto y esto hace que el contacto cambie de nivel en estado abierto a nivel estado cerrado lo cual envía una señal al controlador para que este a su vez mediante un programa compilado en el sistema decida las acciones a tomar en este caso será enviar un mensaje al usuario mediante Twitter informando que el contenedor está por terminarse y de esta manera mantiene siempre una comunicación entre el usuario y el prototipo.

Figura 25. Ubicación del final de carrera en el Prototipo. Fuente: El Autor.

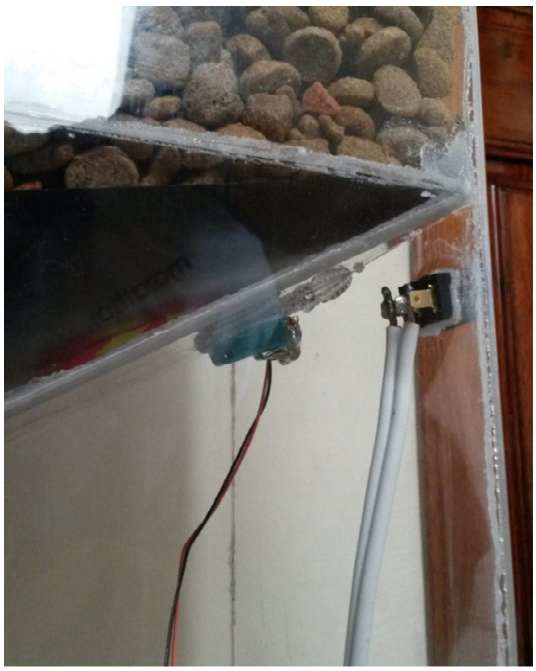


Figura 26. Ubicación del final de carrera en el Prototipo. Fuente: El Autor.

5.2.4. Construcción del Sistema de Medición de Temperatura y Humedad

El sistema de medición de temperatura y humedad se conecta a la placa Arduino Mega utilizando la descripción de pines de la siguiente figura, el pin de salida del módulo DHT11 se conecta al pin 7 de la placa Arduino.

Figura 27. Módulo DHT11. Fuente: https://brainy-bits.com/tutorials/dht11-tutorial/

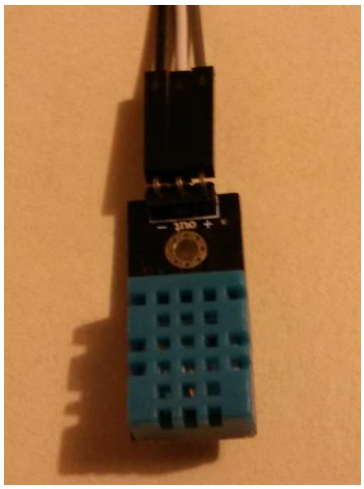


Figura 28. Módulo DHT11. Fuente: El Autor.

Para la conexión se utilizó una placa adicional con la cual el sistema de medición de temperatura y humedad se conecta con la placa Arduino mega.

El conexionado del módulo DHT11 se lo realizó como se indica en la Figura 29, se utilizó cables especiales con conexión hembra en el lado del módulo DHT11 y a su vez una conexión tipo macho en la placa adicional para que el conexionado se lo realice de forma fácil para un mantenimiento a futuro, es decir si el módulo DHT11 falla se lo puede sustituir por otro.

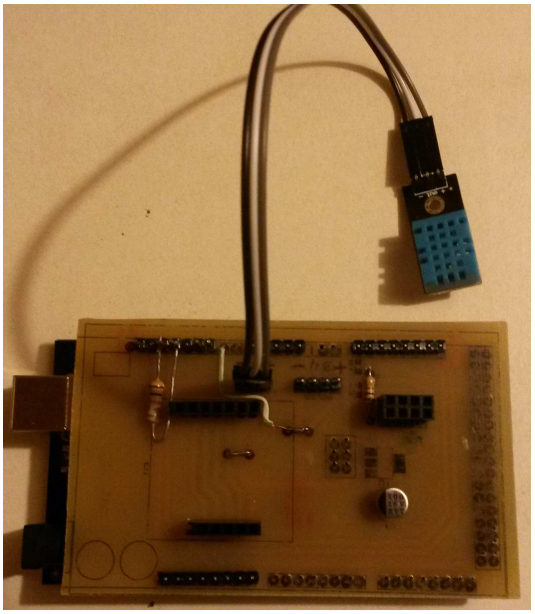


Figura 29. Conexionado del módulo DHT11 en la placa adicional. Fuente: El Autor.

El módulo DHT11 se lo colocó dentro del dispensador por debajo del contenedor de alimento como se indica en la Figura 30 y su función será realizar un sensado de la temperatura y humedad del entorno, esto lo realiza de la siguiente manera: el módulo está constituido de dos sensores: un sensor capacitivo el cual sensa la humedad y un sensor termistor el cual sensa la temperatura ambiente, estas señales entran a un circuito integrado el cual realiza la conversión analógica/digital y a su vez envía la señal digital obtenida a la placa Arduino pasando por la placa adicional para indicar como se encuentra el entorno del prototipo, la placa Arduino a su vez mediante un programa compilado en el sistema decide las acciones a tomar en este caso se enviará un mensaje al usuario mediante Twitter informando como se encuentra el prototipo, el envío de mensajes se lo hará a las 7:00am, 12:00pm, 17:00pm y 22:00pm, de esta manera mantiene siempre una comunicación entre el usuario y el prototipo.

Figura 30. Ubicación del módulo DHT11 en el Prototipo. Fuente: El Autor.

Figura 31. Ubicación del módulo DHT11 en el Prototipo. Fuente: El Autor.

5.2.5. Construcción del Sistema de Medición de Tiempo

Para la conexión del sistema de medición de tiempo se siguió la descripción de los pines como se muestran en la siguiente figura, para el SDA se conectó al Pin 20 y para el SCL al Pin 21 de la Placa Arduino Mega 2650.

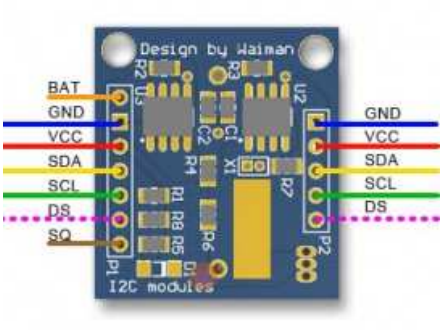


Figura 32. Descripción de Pines de Módulo DS1307. Fuente: http://www.dfrobot.com/wiki/index.php?title=File:Dfr0151_connection.jpg

En la siguiente figura se muestra el módulo DS1307, este módulo se conecta directamente con la placa Arduino Mega 2650 utilizando la placa adicional que se fabricó utilizando una placa impresa.

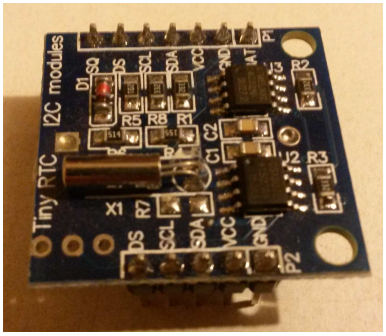


Figura 33. Módulo DS1307. Fuente: El Autor.

El conexionado del módulo DS1307 se lo realizó como se indica en la Figura 34, se utilizó cables especiales con conexión macho en el lado del módulo DS1307 y a su vez una

conexión tipo hembra en la placa adicional para que el conexionado se lo realice de forma fácil para un mantenimiento a futuro, es decir si el módulo DS1307 falla se lo puede sustituir por otro.

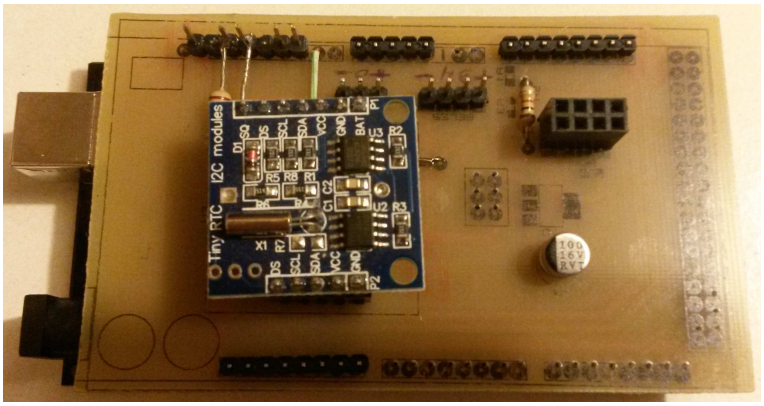


Figura 34. Conexionado del módulo DS1307 en la placa adicional. Fuente: El Autor.

El módulo DS1307 se lo colocó en la placa adicional y esta a su vez conectada a la placa Arduino como se indica en la Figura 35 y su función será obtener la fecha: día, hora, minuto y segundo en tiempo real, esto lo realiza de la siguiente manera: el módulo está constituido de un oscilador que tiene un tiempo de oscilación de un segundo el cual ayuda a un circuito integrado a contar de forma rítmica a los segundos de día, a su vez tiene integrado una pila de 1.5 voltios la cual ayuda al circuito integrado a guardar los datos que se lo programa con la placa Arduino para que se actualice a la fecha y hora actuales, la placa Arduino a su vez mediante un programa compilado en el sistema decide las acciones a tomar en este caso este sistema ayuda a indicar en que momento enviar los mensajes vía Twitter y a su vez a dispensar el alimento cada cierto periodo de tiempo.

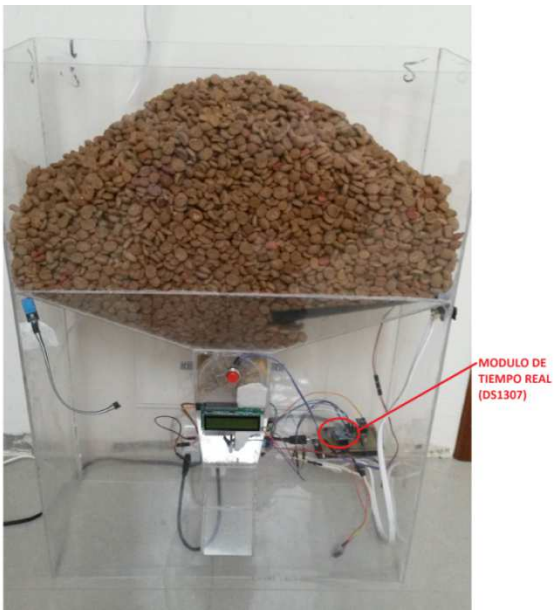


Figura 35. Ubicación del módulo DS1307 en el Prototipo. Fuente: El Autor.

5.2.6. Construcción del Sistema Actuador

El sistema actuador se conecta a la placa Arduino con la descripción de los pines que dependiendo de la marca del servomotor indique, para el caso la marca utilizada es HITEC como se indican en la Figura 36.

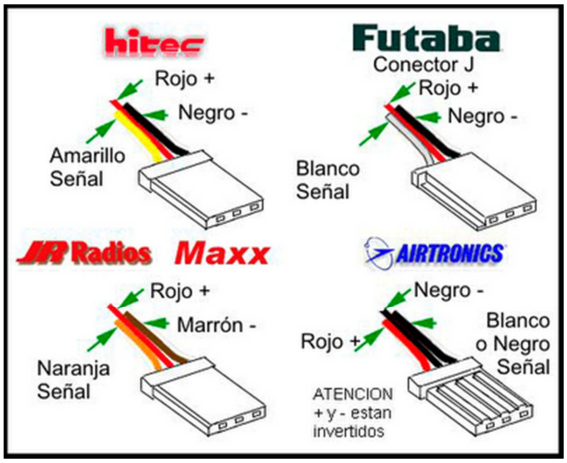


Figura 36. Descripción pines Servomotor. Fuente: http://www.web-robotica.com/arduino/sencillo-brazo-robotizado-con-2-servos-y-controlado-con-arduino-tutorial-paso-a-paso

Para la conexión se utilizó una placa adicional con la cual se conecta directamente a la placa Arduino para la señal del servomotor este se conecta al pin 3.

Figura 37. Servomotor. Fuente: El Autor.

El conexionado del servomotor se lo realizó como se indica en la Figura 38, se utilizó cables especiales con conexión hembra en el lado del servomotor y a su vez una conexión tipo macho en la placa adicional para que el conexionado se lo realice de forma fácil para un mantenimiento a futuro, es decir si el servomotor falla se lo puede sustituir por otro.

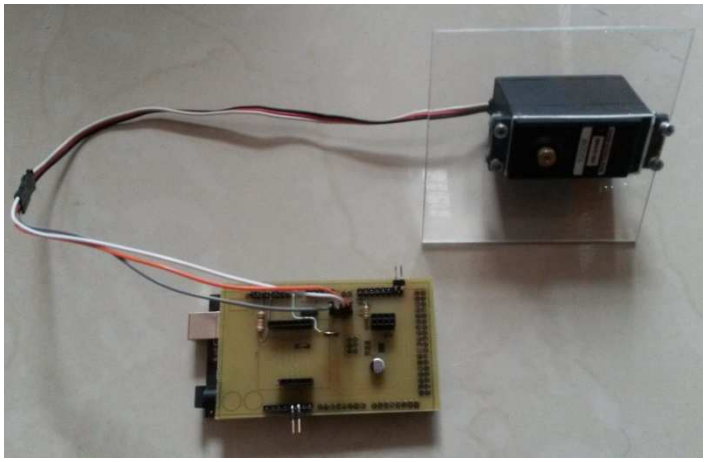


Figura 38. Conexionado del servomotor en la placa adicional. Fuente: El Autor.

El servomotor se lo colocó en la parte inferior del contenedor de alimento como se indica en la Figura 39 y su función será dispensar alimento, esto lo realiza de la siguiente manera: el servomotor está constituido de un sistema de engranes que ayudan a que tenga un alto torque y con la ayuda de un acoplamiento construido especialmente para este prototipo realiza un giro de 0 a 180° para enviar el alimento de la parte superior del acople el cual está dividido en cuatro partes hacia la parte inferior del acople depositándolo en el platillo de la mascota es decir la placa Arduino a su vez mediante un programa compilado en el sistema decide las acciones a tomar en este caso este sistema envía una señal PWM al servomotor para que este actué y cada vez que el servomotor deposite el alimento envía un mensaje vía Twitter de esta manera mantiene siempre una comunicación entre el usuario y el prototipo.

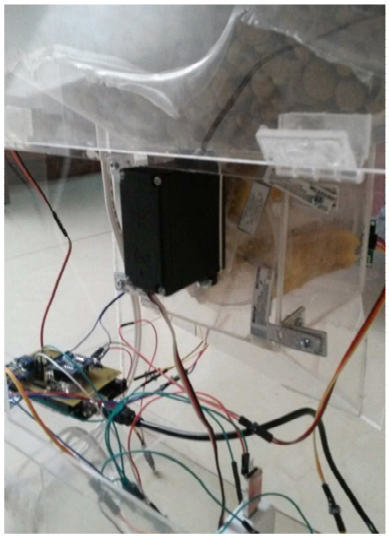


Figura 39. Ubicación del servomotor en el Prototipo. Fuente: El Autor.

5.2.7. Construcción del Sistema de Comunicaciones

El sistema de comunicación utilizando el módulo ESP8266 se conectó utilizando la siguiente descripción de pines, para los pines TX y RX se utilizó los pines 14 y 15 respectivamente en la placa Arduino.

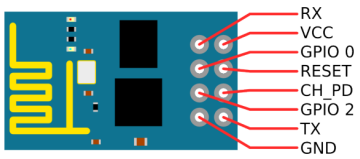


Figura 40. Descripción de Pines de Módulo ESP8266. Fuente: https://elbunuelo.com/hola-mundo-con-esp8266/

Para la conexión se utilizó una placa adicional para la conexión de los pines del módulo ESP8266 a la placa Arduino mega.

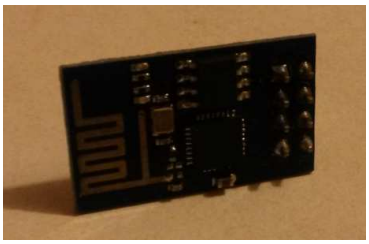


Figura 41. Módulo ESP8266. Fuente: El Autor.

El conexionado del Módulo ESP8266 se lo realizó como se indica en la Figura 42, se utilizó cables especiales con conexión macho en el lado del Módulo ESP8266 y a su vez una conexión tipo hembra en la placa adicional para que el conexionado se lo realice de forma fácil para un mantenimiento a futuro, es decir si el Módulo ESP8266 falla se lo puede sustituir por otro.

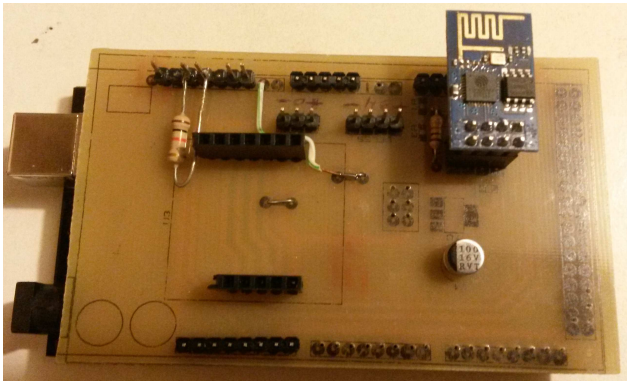


Figura 42. Conexionado del Módulo ESP8266 en la placa adicional. Fuente: El Autor.

El módulo ESP8266 se lo colocó en la placa adicional y esta a su vez conectada a la placa Arduino como se indica en la Figura 43 y su función será comunicar al usuario con el prototipo, esto lo realiza de la siguiente manera: módulo ESP8266 está constituido de la electrónica necesaria para la comunicación Radio Frecuencia en la banda WFI, así como la pila TCPIP y que se comunica con el usuario a través de un puerto serie utilizando Comandos AT con lo cual para la comunicación con la placa Arduino es necesario programarle a una determinada velocidad en este caso se utiliza una velocidad de 115.200 baudios por segundo (bps), es decir mediante la placa Arduino y con un programa compilado en el sistema, este decide las acciones a tomar en este caso este sistema envía la solicitud al router que se desea conectar con el nombre y su contraseña y de esta manera el prototipo se conecta a la red y es capaz de enviar mensajes vía Twitter y comunicarse a un teléfono inteligente para la configuración inicial y el control de esta manera mantiene siempre una comunicación entre el usuario y el prototipo.

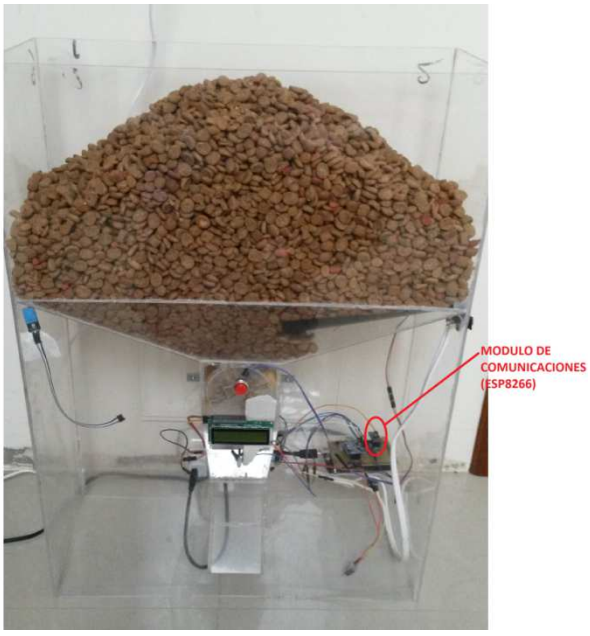


Figura 43. Ubicación del Módulo ESP8266 en el Prototipo. Fuente: El Autor.

5.2.8. Construcción del Sistema de Visualización

Se utilizó un LCD 16x2 con un controlador I2C conectado al controlador Arduino Mega para el sistema de visualización con lo cual se realizará la comunicación visual entre en prototipo y el usuario.

El sistema de visualización utiliza el módulo controlador LCD I2C que se conectó utilizando la siguiente descripción de pines, para los pines SDA y SCL se utilizó los pines 20 y 21 respectivamente en la placa Arduino.

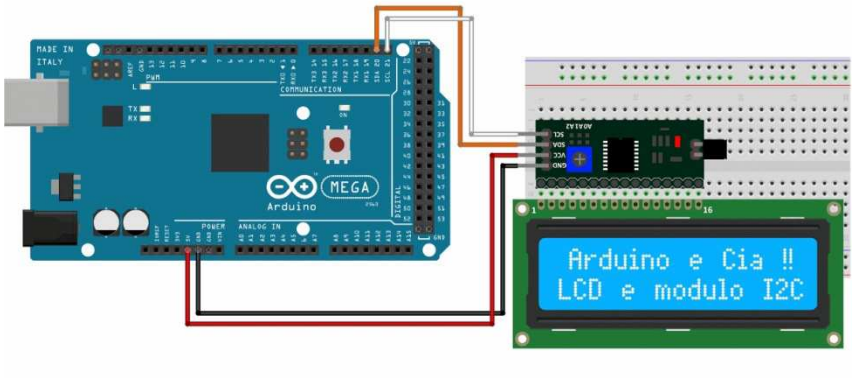


Figura 44. Descripción de Pines de Módulo controlador I2C conectado a LCD 16x2. Fuente: https:// MPE-418494234-modulo-iic-i2c-twi-spi-para-lcd-1602-arduino-_JM

Para la conexión se utilizó una placa adicional para la conexión de los pines del módulo controlador LCD I2C a la placa Arduino mega.

Figura 45. Display LCD 16x2 con Módulo controlador I2C. Fuente: El Autor.

El conexionado del módulo controlador LCD I2C se lo realizó como se indica en la Figura 45, se utilizó cables especiales con conexión macho en el lado del módulo controlador LCD I2C y a su vez una conexión de igual forma tipo macho en la placa adicional para que el conexionado se lo realice de forma fácil utilizando cables tipo hembra en ambos lados para un mantenimiento a futuro, es decir si el módulo controlador LCD I2C o el Display LCD 16x2 falla se los puede sustituir por otro.

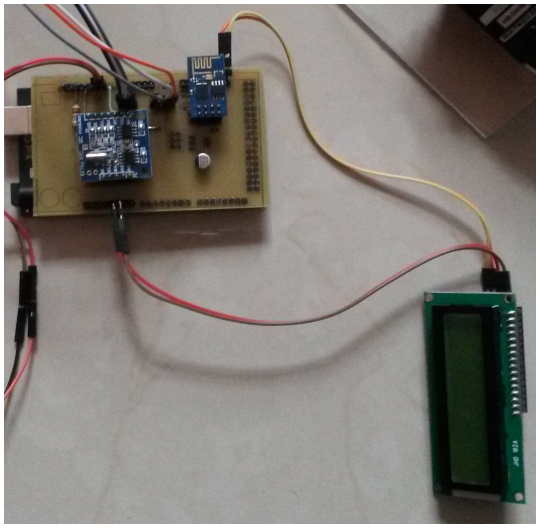


Figura 46. Conexionado del Módulo controlador LCD I2C junto con el Display 16x2 en la placa adicional. Fuente: El Autor.

El Display 16x2 se conectó al módulo controlador LCD I2C y este se conectó a la placa adicional y esta a su vez conectada a la placa Arduino como se indica en la Figura 43 y su función será indicar al usuario mediante el Display LCD 16x2 lo que el controlador Arduino Mega del prototipo está realizando en ese momento, esto lo realiza de la siguiente manera: módulo controlador LCD I2C está constituido de la electrónica necesaria para la comunicación I2C con el Display LCD 16x2 utilizando 16 conectores, el módulo controlador LCD I2C se comunica también con el controlador Arduino Mega mediante comunicación I2C solamente con dos cables, es decir mediante la placa Arduino y con un programa compilado en el sistema envía

las acciones para que se visualice en el Display, es así como el Usuario puede conocer todo lo que está sucediendo en el prototipo como por ejemplo si se conecta o no a la red Wifi, si envía un mensaje a Twitter, si deposita el alimento a la mascota, etc.

Figura 47. Ubicación del Display LCD 16x2 en el Prototipo. Fuente: El Autor.

5.3. Desarrollo de Aplicación Móvil basado en tecnología Android

El desarrollo de la aplicación Android se realizó con App Inventor 2, el cual es una herramienta muy utilizada para el desarrollo de aplicaciones con estructura intuitiva con el usuario, este software implantado en la red se lo puede utilizar sin la necesidad de instalarlo en tu PC, es decir que tú puedes acceder a tus aplicaciones en cualquier parte del mundo donde dispongas del explorador Google Chrome y por supuesto internet, esta herramienta cuenta con dos interfaces la primera es la cual puedes ver la pantalla de tu teléfono inteligente y ubicar determinados objetos que necesites para crear tu aplicación y la segunda interface es en la cual vas a realizar la programación, esta última tiene la ventaja de programar en bloques es decir

puedes construir tu aplicación como si estuvieras armando un rompecabezas, es una forma muy dinámica que amplia mucho la imaginación para crear diferentes aplicaciones que satisfagan tus necesidades, a continuación se mostrará la herramienta App Inventor 2 y la construcción de la aplicación para la comunicación con el dispensador inteligente.

Para el desarrollo de la aplicación se debe diseñar una pantalla de inicio como la mayoría de aplicaciones para teléfonos inteligentes lo tienen, como se indica en la Figura 48 se ubica el nombre de la aplicación, un pequeño dibujo relacionado al tema del diseño, también se encuentra una capsula con la que se indica que la aplicación se está cargando.

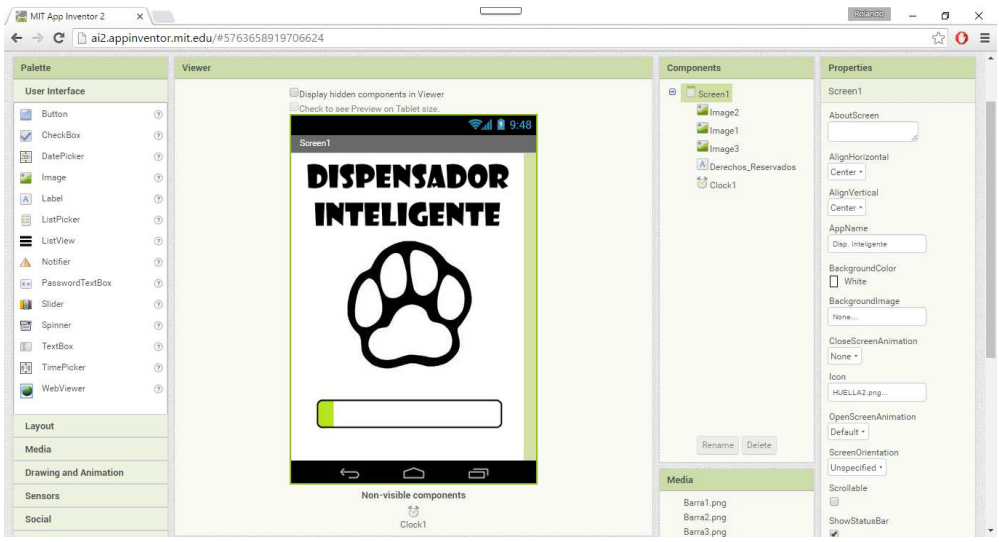


Figura 48. Diseño de la pantalla inicial de la Aplicación Android utilizando App Inventor 2. Fuente: El Autor.

En la segunda pantalla como se indica en la Figura 49 aparece el menú de la aplicación en el cual indica cinco opciones con su correspondiente información, la primera opción ***"Datos Usuario"*** direcciona a otra pantalla ver Figura 52 la segunda opción ***"Datos Mascota"*** direcciona a otra pantalla ver Figura 55, la siguiente opción ***"Dispensar Manualmente"*** direcciona a la pantalla que se indica en la Figura 60, la siguiente opción ***"Biblioteca"***

direcciona a la pantalla ver Figura 63 y por último la opción ***"Acerca de"*** el cual direcciona a la pantalla ver Figura 64.

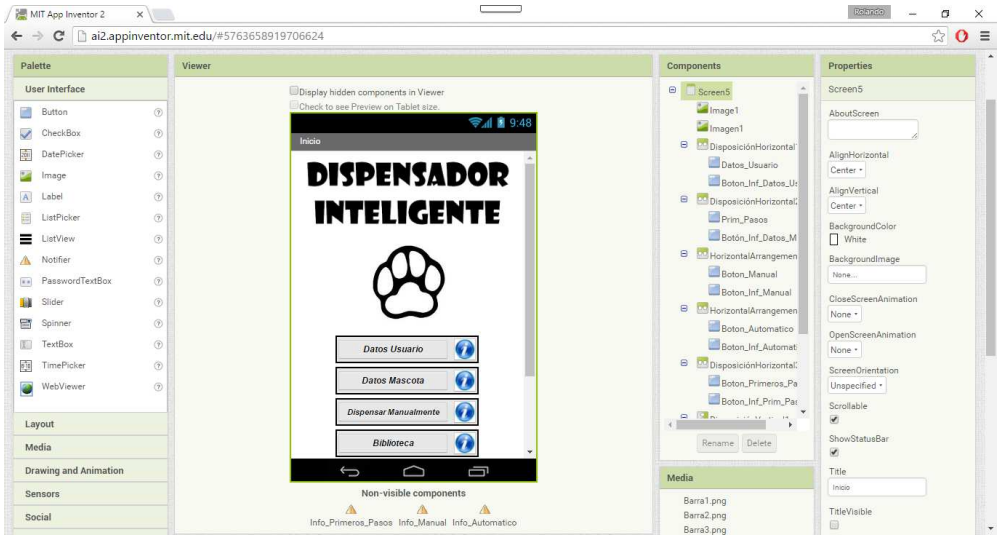


Figura 49. Diseño de la Pantalla "MENU" de la Aplicación Android utilizando App Inventor 2. Fuente: El Autor.

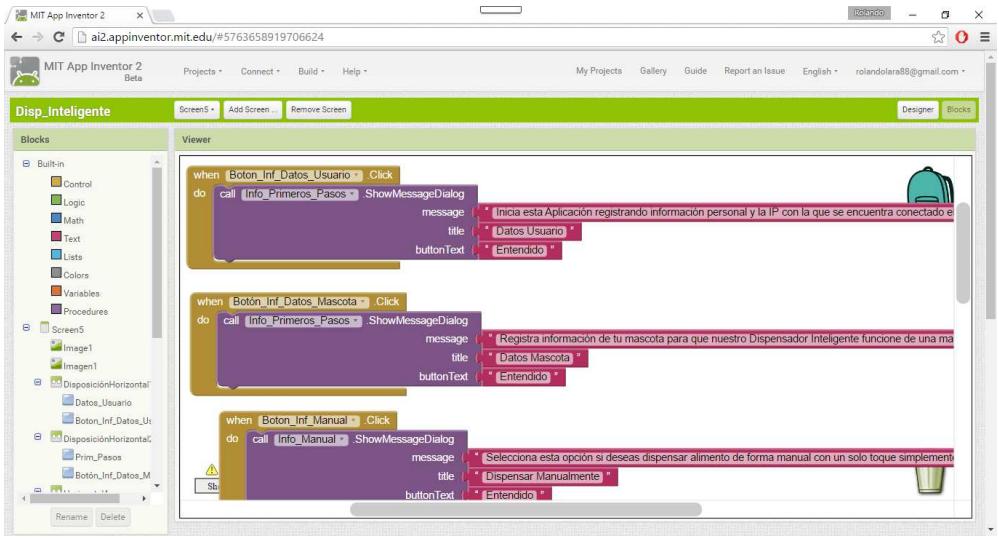


Figura 50. Programación de la pantalla "MENU" de la Aplicación Android utilizando App Inventor 2. Fuente: El Autor.

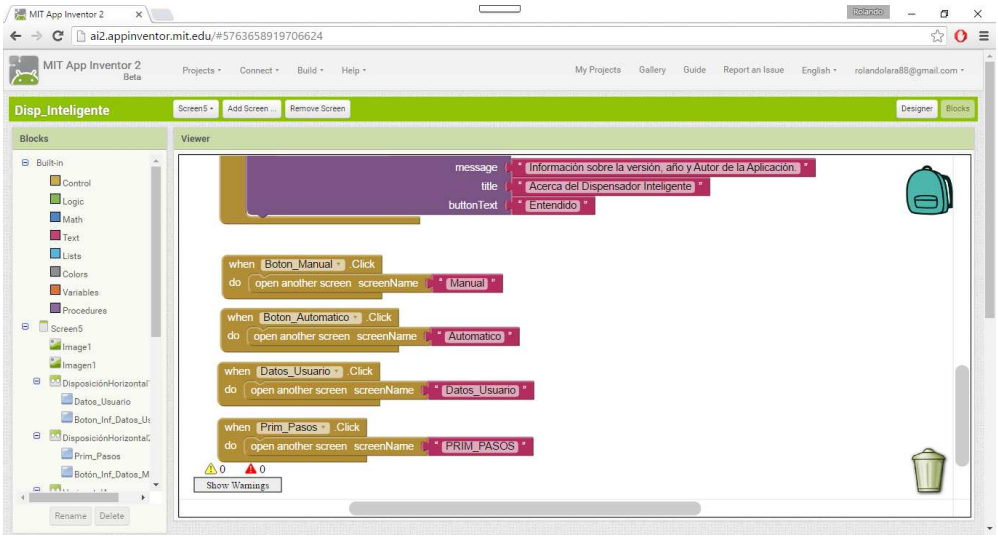


Figura 51. Programación de la pantalla "MENU" de la Aplicación Android utilizando App Inventor 2. Fuente: El Autor.

La pantalla ***"Datos Usuario"*** está diseñado utilizando una etiqueta para ubicar el título, la imagen insignia del dispensador y una serie de etiquetas con sus respectivos entradas de texto para ser llenadas por el Usuario, también se encuentra un botón para enviar la información al prototipo y un visor web para obtener la respuesta del mismo como se indica en la Figura 52.

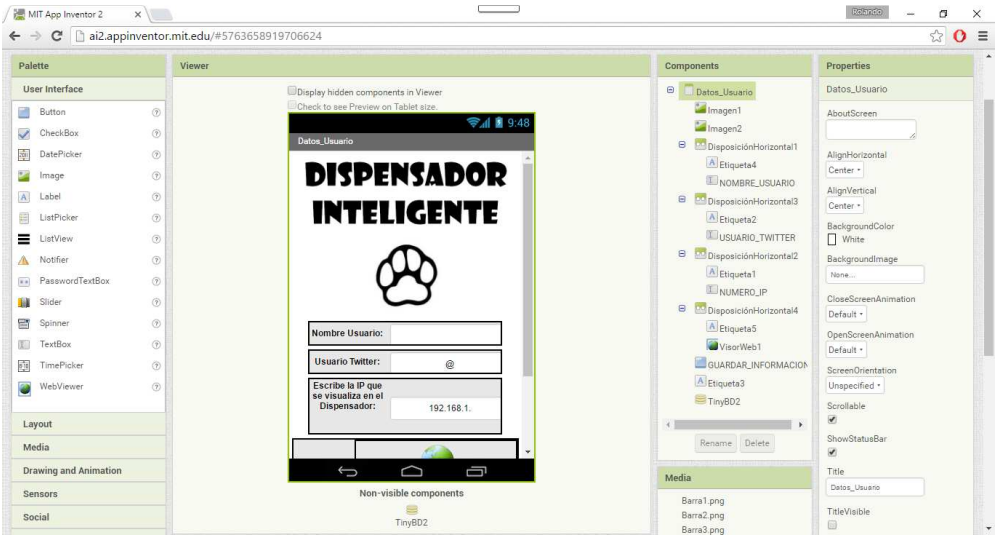


Figura 52. Diseño de la pantalla "DATOS USUARIO" de la Aplicación Android utilizando App Inventor 2. Fuente: El Autor.

En la pantalla de bloques para la programación se guarda los datos en una base de datos para no perderlos al momento de cerrar la aplicación ver Figura 53, estos datos son Nombre del Usuario, Twitter del usuario e IP para conectar con el dispensador, una vez obtenido estos datos en la opción de botón se los ubica para ser enviados al prototipo como se indica en la Figura 54.

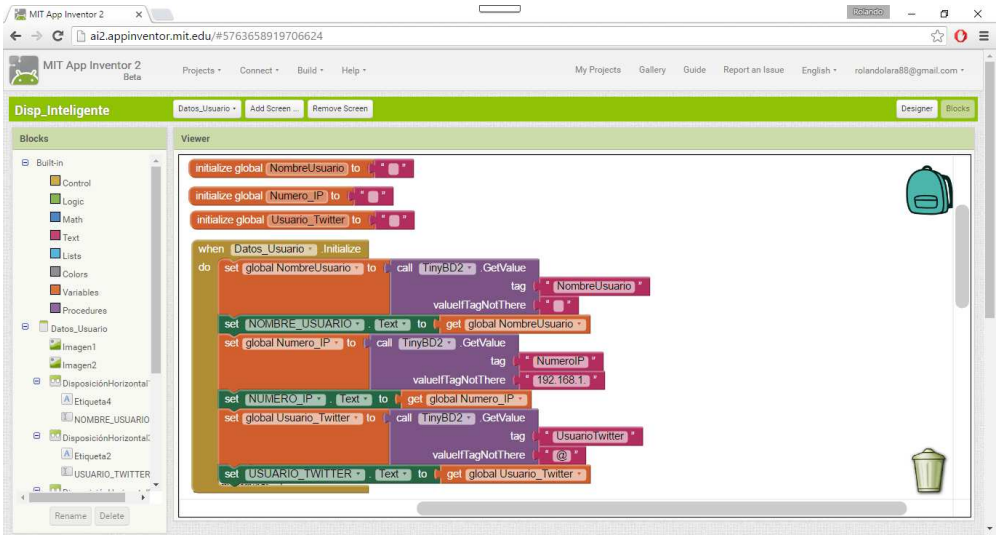


Figura 53. Programación de la pantalla "DATOS USUARIO" de la Aplicación Android utilizando App Inventor 2. Fuente: El Autor.

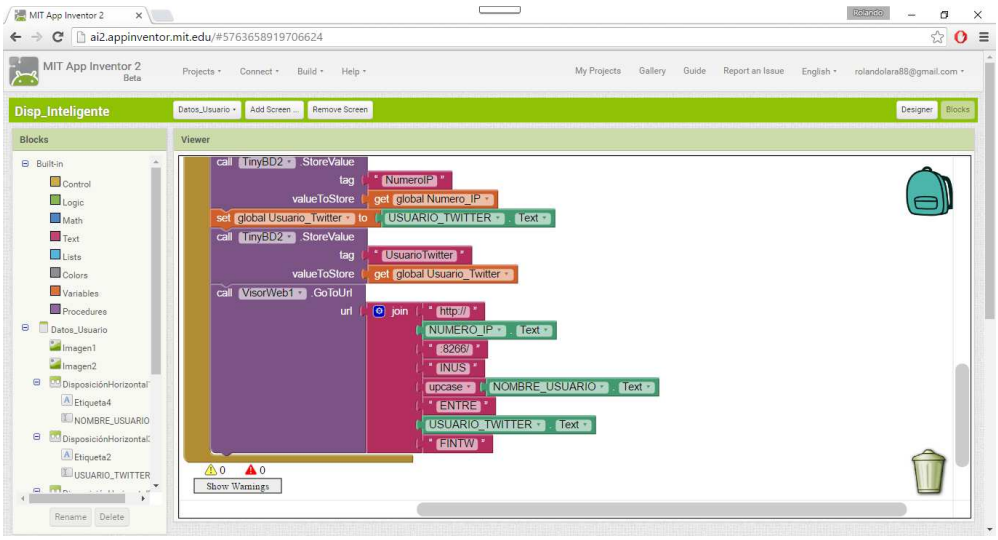


Figura 54. Programación de la pantalla "DATOS USUARIO" de la Aplicación Android utilizando App Inventor 2. Fuente: El Autor.

Para el diseño de la pantalla ***"Datos Mascota"*** se utilizó una etiqueta para el nombre de la aplicación y la descripción de la pantalla, etiqueta con ingreso de texto para el nombre de la mascota, un botón con el cual inicia una serie opciones para seleccionar como por ejemplo tipo de mascota, tamaño de raza y edad de la mascota, se utilizó otro botón para enviar la información obtenida al dispensador y así mismo un visor web para receptar la respuesta del dispensador.

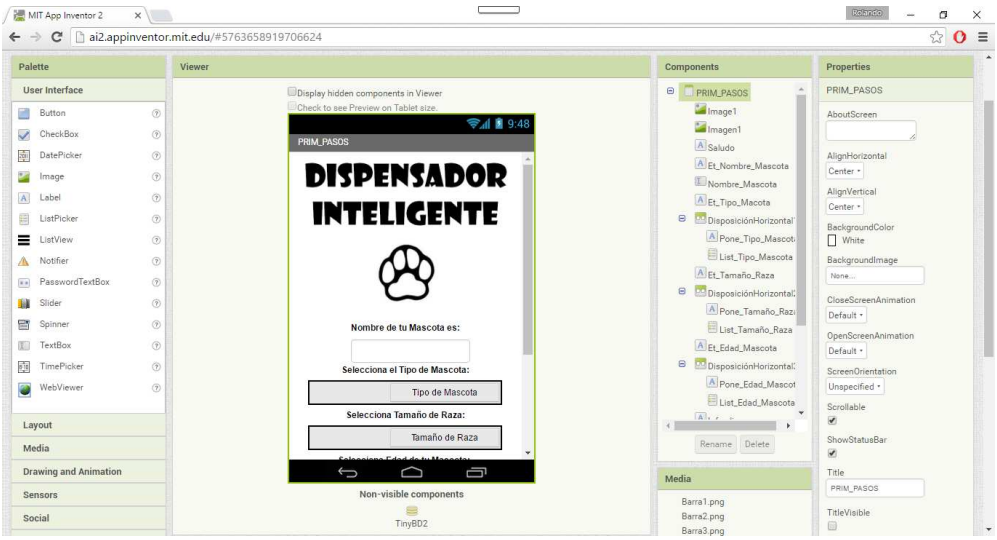


Figura 55. Diseño de la pantalla "DATOS MASCOTA" de la Aplicación Android utilizando App Inventor 2. Fuente: El Autor.

Para realizar la programación de esta pantalla se debe dirigir a la opción bloques y se crea una base de datos la cual se utilizara para guardar las variables que se piden al usuario como: tipo de mascota, nombre, tamaño y edad, de igual manera se obtiene los datos guardados de la pantalla ***"Datos Usuario"*** los cuales serán utilizados para la comunicación con el dispensador como se indica en la Figura 56, Figura 57, Figura 58 y Figura 59.

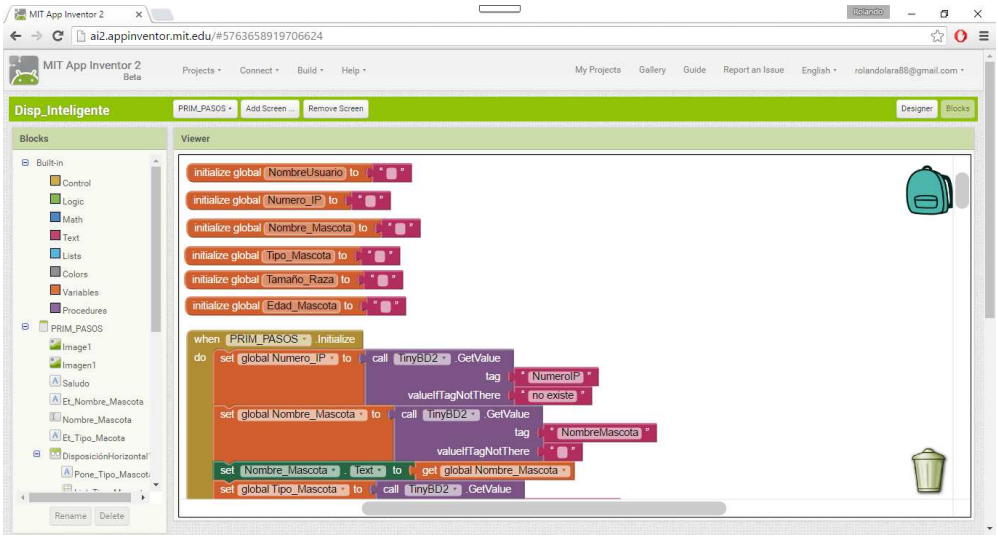


Figura 56. Programación de la pantalla "DATOS MASCOTA" de la Aplicación Android utilizando App Inventor 2. Fuente: El Autor.

Figura 57. Programación de la pantalla "DATOS MASCOTA" de la Aplicación Android utilizando App Inventor 2. Fuente: El Autor.

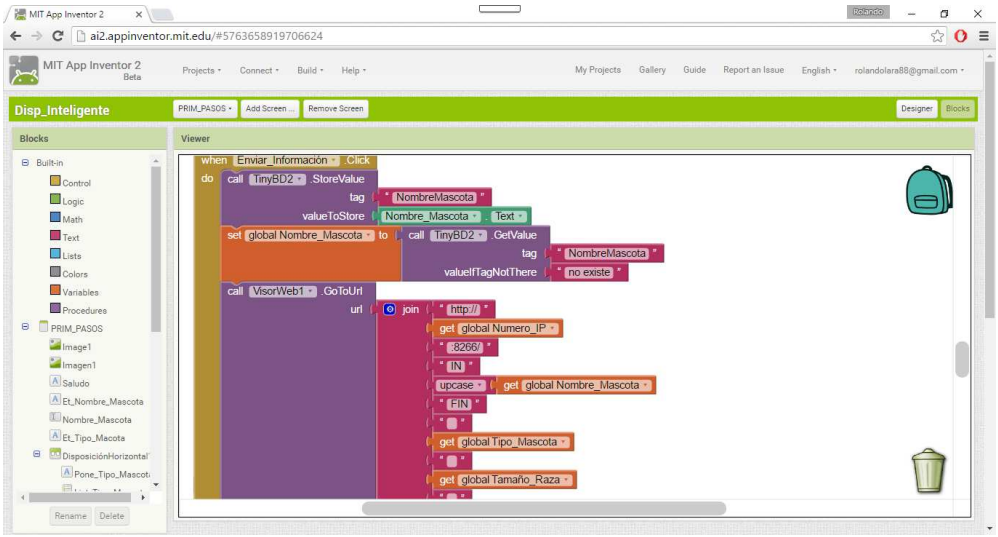


Figura 58. Programación de la pantalla "PRIMEROS PASOS" de la Aplicación Android utilizando App Inventor 2. Fuente: El Autor.

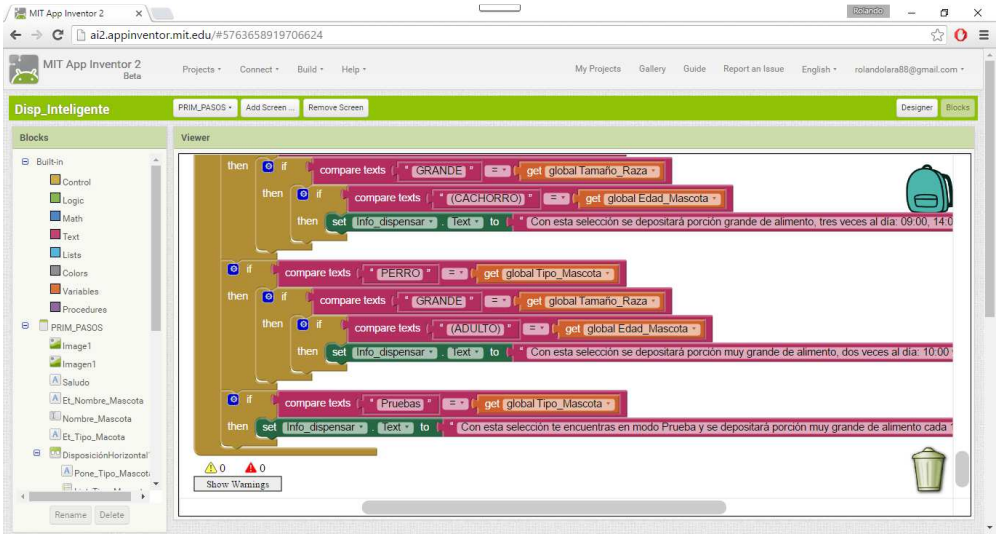


Figura 59. Programación de la pantalla "PRIMEROS PASOS" de la Aplicación Android utilizando App Inventor 2. Fuente: El Autor.

Para el diseño de la pantalla ***"Dispensar Manualmente"*** ver Figura 60 se utilizó una etiqueta para el título de la aplicación, un botón para entrar a seleccionar el tamaño de porción a dispensar, un botón de *pulsa para dispensar* y una visor Web para indicar que el dispensador recibió la orden.

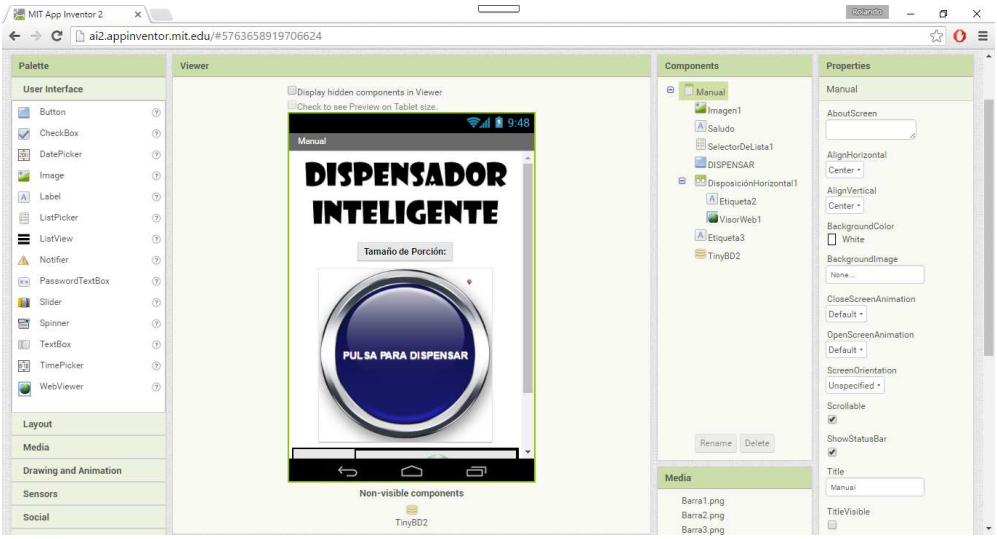


Figura 60. Diseño de la pantalla "DISPENSAR MANUALMENTE" de la Aplicación Android utilizando App Inventor 2. Fuente: El Autor.

Para la programación de esta pantalla ver Figura 61 y Figura 62 se utilizó los bloques "When" para accionar él envió de información al dispensador se unió la dirección IP obtenida de la pantalla ***"Datos Usuario"*** con el puerto que se utilizó. A más de eso el visor Web indica si el dispensador recibió la información enviada.

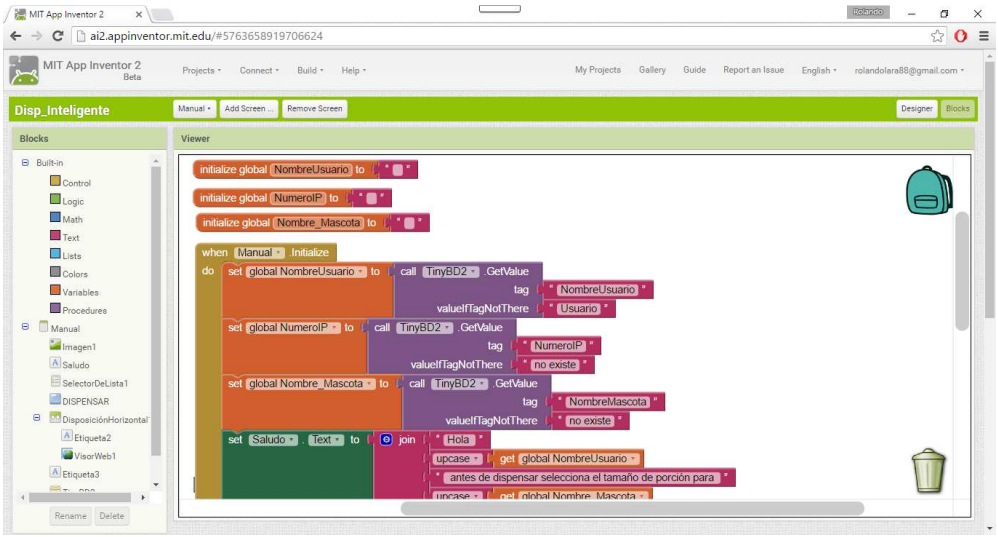


Figura 61. Programación de la pantalla "DISPENSAR MANUALMENTE" de la Aplicación Android utilizando App Inventor 2. Fuente: El Autor.

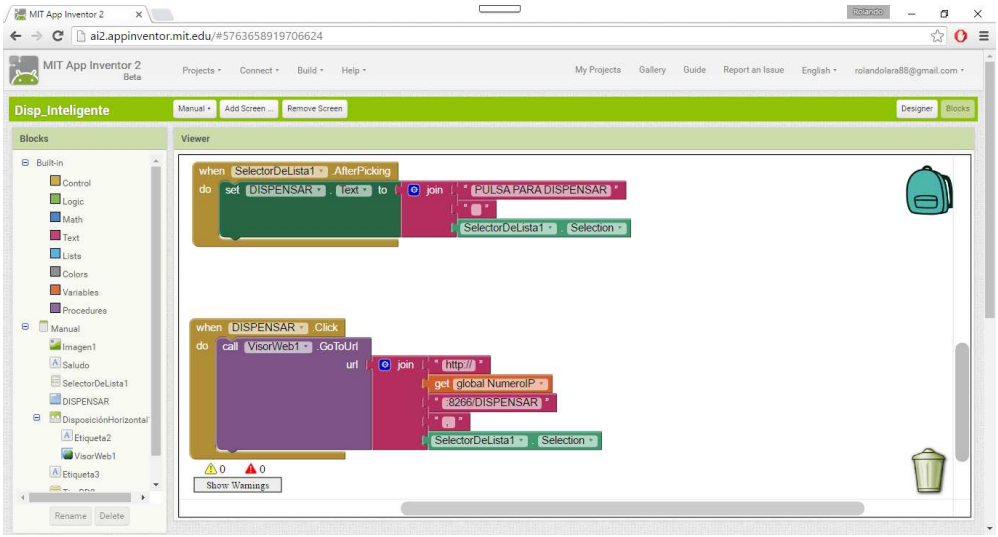


Figura 62. Programación de la pantalla "DISPENSAR MANUALMENTE" de la Aplicación Android utilizando App Inventor 2. Fuente: El Autor.

Para el diseño de la pantalla ***"Biblioteca"*** se utilizó una etiqueta para el título de la aplicación, la imagen del dispensador y varias etiquetas que se utilizaran para ingresar texto sobre los cuidados y alimentación de la mascota con su respectiva fuente de la cual se obtuvo la información como se indica en la Figura 63.

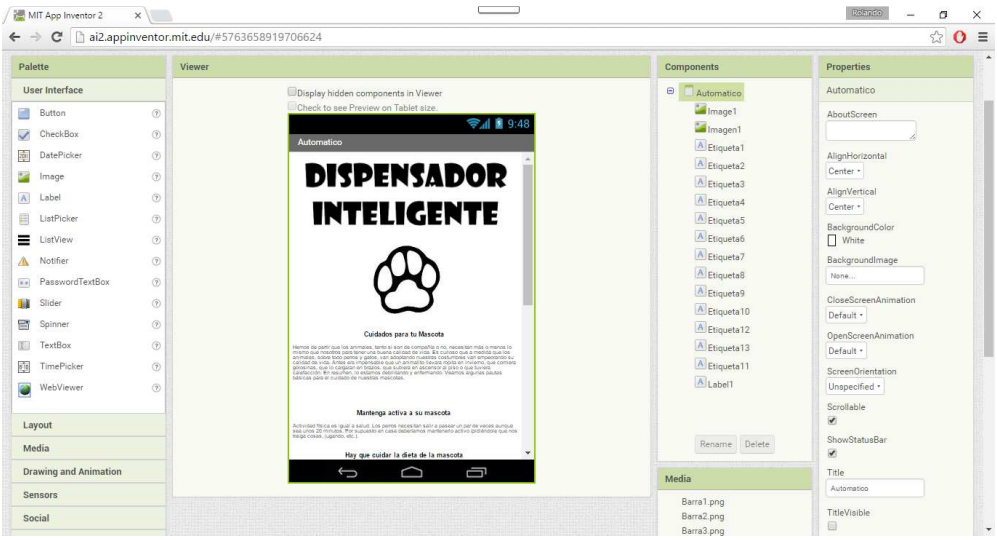


Figura 63. Diseño de la pantalla "BIBLIOTECA" de la Aplicación Android utilizando App Inventor 2. Fuente: El Autor.

Para la programación de la pantalla ***"Acerca de"*** se utilizó una etiqueta para el título de la aplicación, la imagen del dispensador y varias etiquetas que se utilizaran para indicar la versión de la aplicación, su autor, el tema de tesis y el nombre del Director de la misma como se indica en la Figura 64 y Figura 65.

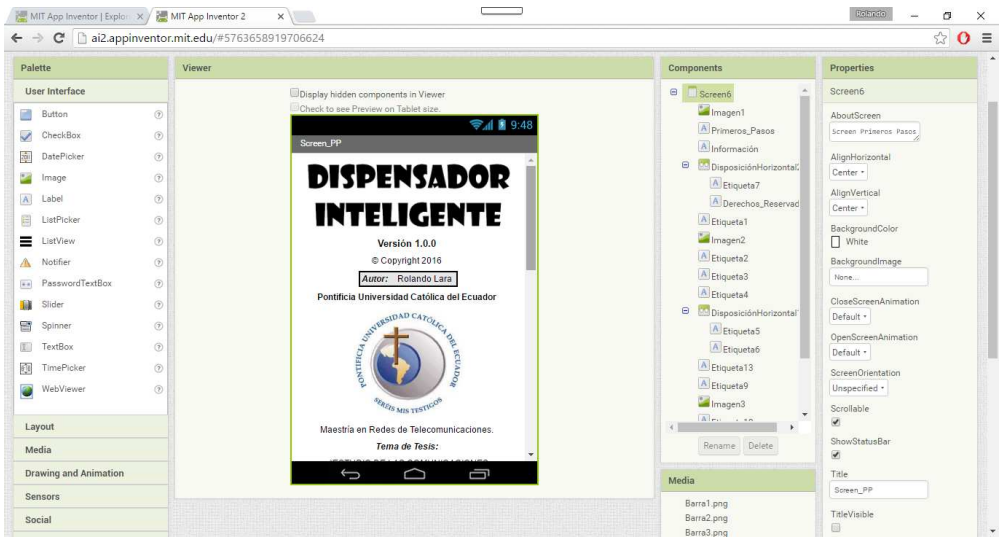


Figura 64. Diseño de la pantalla "ACERCA DE" de la Aplicación Android utilizando App Inventor 2. Fuente: El Autor.

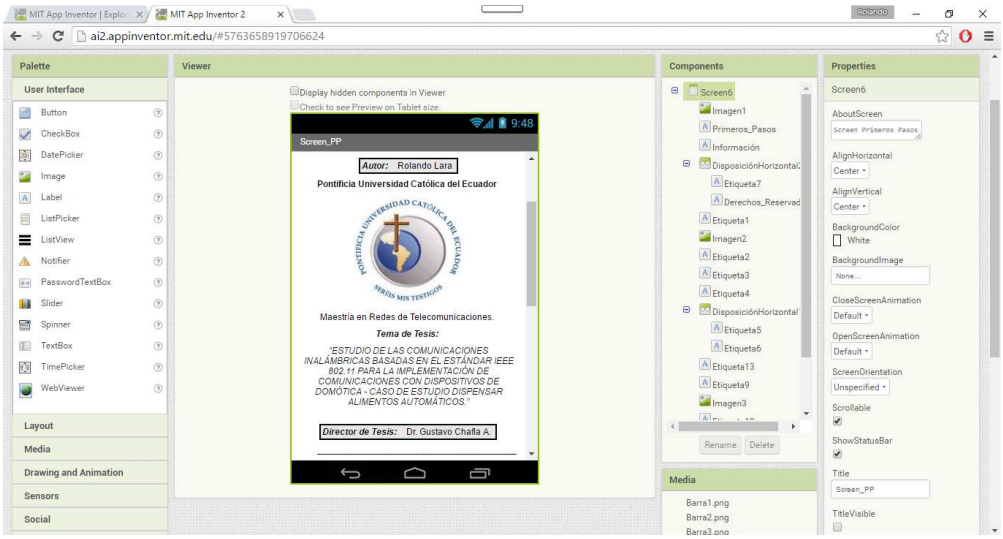


Figura 65. Diseño de la pantalla "ACERCA DE" de la Aplicación Android utilizando App Inventor 2. Fuente: El Autor.

Además en esta pantalla, en la aplicación del desarrollador se encuentra un Código QR mediante el cual utilizando otro dispositivo se lo puede fotografiar para descargar la aplicación a otro dispositivo como se indica en la Figura 66.

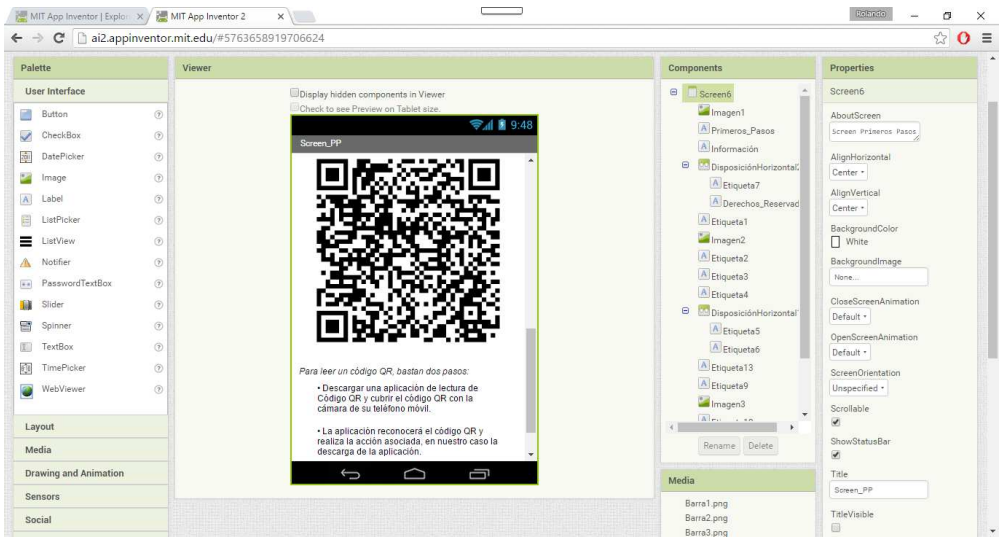


Figura 66. Diseño de la pantalla "ACERCA DE" de la Aplicación Android utilizando App Inventor 2. Fuente: El Autor.

5.4. Programación del Sistema Dispensador Inteligente

Para la programación se utilizó el Arduino Software (IDE) el cual es un software embebido de las placas Arduino, para este caso se utilizará la placa Arduino Mega 2560.

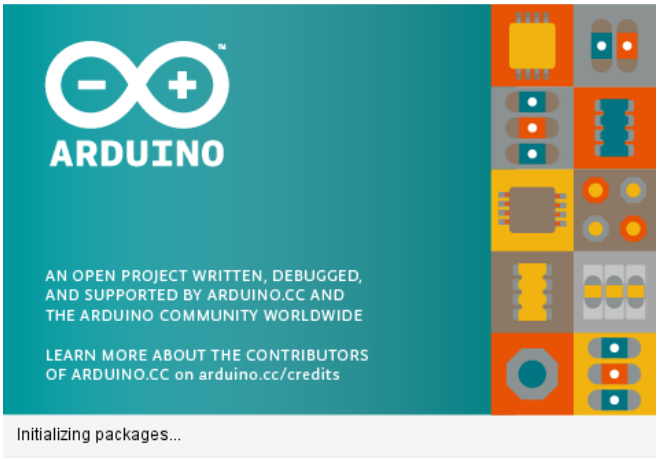


Figura 67. Arduino Software (IDE). Fuente: El Autor.

Se realizó diferentes programas para probar cada uno de los sistemas que se mencionó anteriormente:

Un sistema que también se comprobó su funcionalidad realizando un programa indicado en la Figura 68, fue al sistema de medición de nivel del contenedor utilizando la entrada 10 de la placa Arduino Mega, es decir que cuando el contenedor baja su nivel de alimento por debajo del final de carrera este se activa enviando una señal al controlador y este a su vez envía un mensaje vía Twitter al usuario indicando que el contenedor está por terminarse.

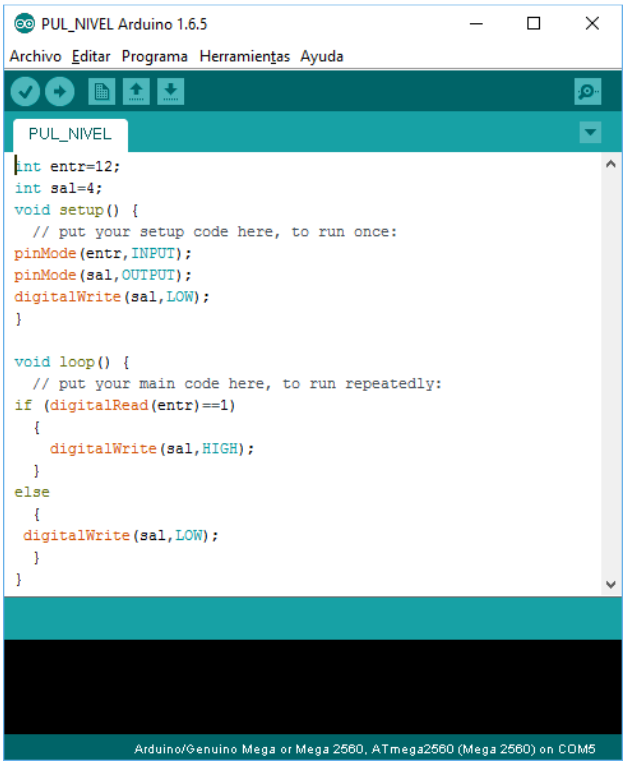


Figura 68. Programación del Sistema de medición de Nivel con Arduino Software (IDE). Fuente: El Autor.

Otro Sistema con el cual se realizó pruebas individuales fue al Sistema de medición de Temperatura y Humedad, se programó directamente a la placa Arduino como se indica en la Figura 69 para configurar la entrada a la cual llegaran los datos del sensor y la velocidad de transmisión.

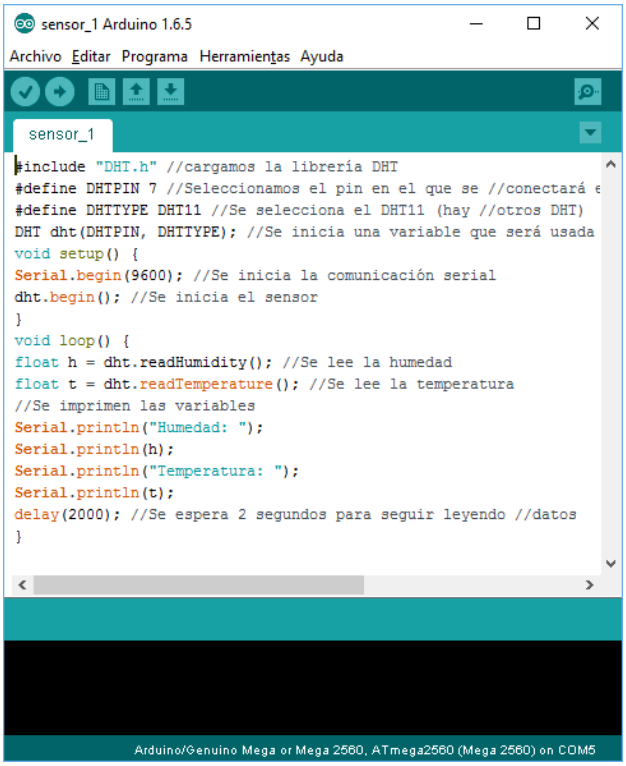


Figura 69. Programación del Sistema de medición de Temperatura y Humedad con Arduino Software (IDE). Fuente: El Autor.

Para configurar el Sistema de medición de Tiempo se realizó un programa de inicialización de fecha, hora, minuto y segundo como se indica en la Figura *70*. El sistema mediante el comando ***Date time now*** obtiene la fecha y hora actual, con esto al programar el Sistema guarda esa información en su memoria eeprom y empieza su conteo en tiempo real.

Figura 70. Programación del Sistema de medición de tiempo con Arduino Software (IDE). Fuente: El Autor.

El sistema actuador de la misma forma se realizó un programa independiente para comprobar la funcionalidad del sistema como se indica en la Figura 71, el programa utiliza la señal PWM para el movimiento del servomotor que gira de 0 a 180°, el programa gira en sentido horario al momento de enviar la primera acción y para la segunda acción el servomotor gira en sentido anti horario eso minimiza el riesgo que el dispensador se obstruya.

Figura 71. Programación del Sistema Actuador con Arduino Software (IDE). Fuente: El Autor.

Se realizó un programa individual para verificar el funcionamiento del Sistema de comunicaciones con el protocolo IEEE 802.11 utilizando el Módulo ESP8266 como se indica en la Figura 72, para esto se delimitaron entradas y salidas; y se configuró la velocidad de transmisión entre la Placa y el Módulo el cual se concluyó que para una mejor transmisión sin perdidas se utilizó una velocidad de transmisión de 115200 bps.

Figura 72. Programación del Sistema de Comunicaciones con Arduino Software (IDE). Fuente: El Autor.

El sistema de comunicación también se programó como se indica en la Figura 73 para realizar una comunicación con la aplicación Android para lo cual se realizó un Web server que dispondrá el dispensador y cuando la aplicación Android desee comunicarse con el dispensador, este enviará con la respuesta correcta según el pedido de la aplicación.

Figura 73. Programación de un WEB SERVER para el prototipo utilizando Arduino Software (IDE). Fuente: El Autor.

5.5. Pruebas de Funcionamiento del Sistema

Para las pruebas de funcionamiento se realizó de forma individual para cada sistema de la siguiente manera:

Se comprobó el correcto funcionamiento del sistema de medición de Nivel conectando el FdC en la placa Arduino y está a una Laptop con un cable de comunicaciones Serial a USB y de igual manera se conecta un diodo led para verificar si el programa realiza las funciones correctas como se indica en la Figura 74, al manipular el FdC enviará una señal a la laptop indicando en la pantalla de visualización del puerto serial que el contenedor está vacío como se indica en la Figura 75.

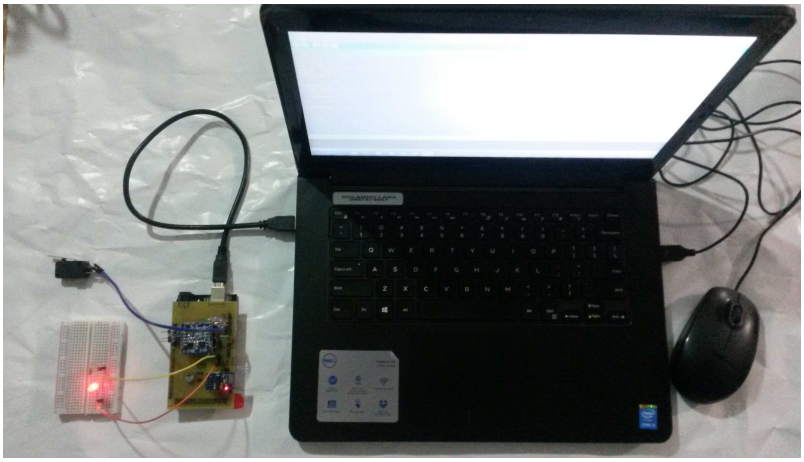


Figura 74. Pruebas de funcionamiento del Sistema de medición de Nivel. Fuente: El Autor.

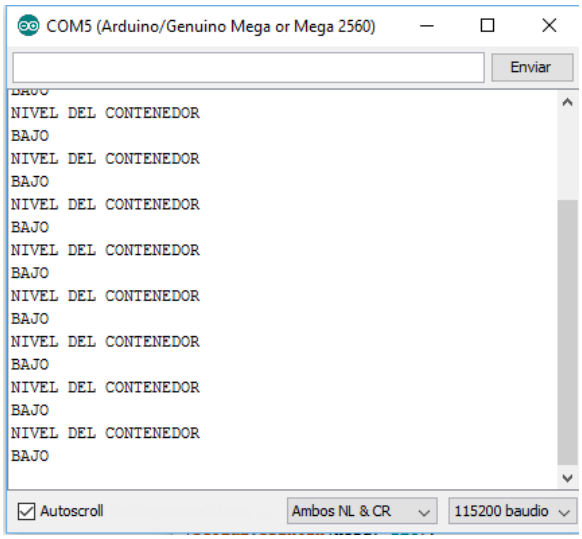


Figura 75. Pantalla de las Pruebas realizadas al Sistema de medición de Nivel. Fuente: El Autor.

Para comprobar el funcionamiento del Sistema de medición de temperatura y humedad se utilizó una laptop conectada a la placa Arduino vía comunicación serial para visualizar los datos que envía el módulo DHT11 como se indica en la Figura 76 y Figura 77.

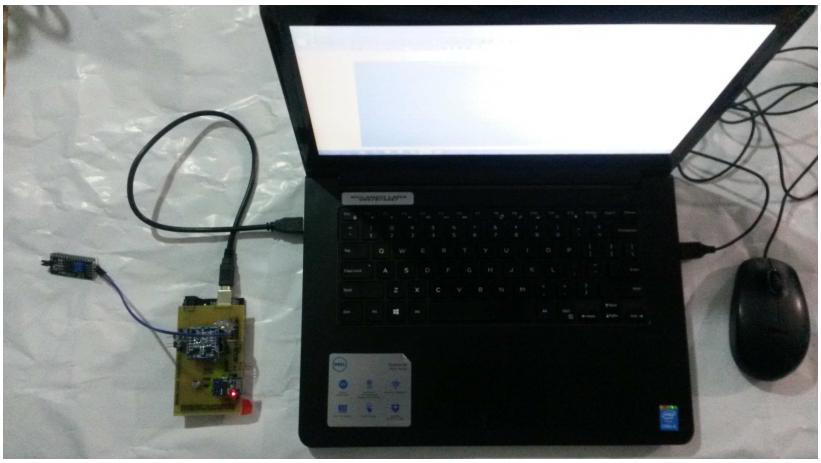


Figura 76. Pruebas de funcionamiento del Sistema de medición de Temperatura y Humedad. Fuente: El Autor.

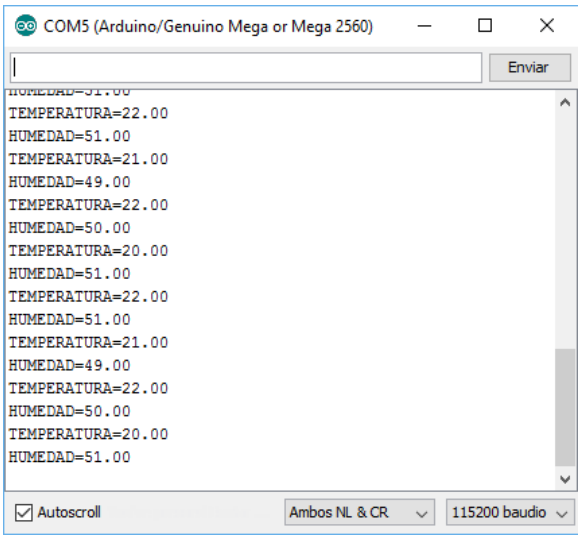


Figura 77. Pantalla de las Pruebas realizadas al Sistema de medición de Temperatura y Humedad. Fuente: El Autor.

Para comprobar el funcionamiento del Sistema de medición de tiempo se utilizó una laptop conectada a la placa Arduino vía comunicación serial para visualizar los datos que envía el módulo DS1307 como se indica en la Figura 78 .

Figura 78. Pruebas de funcionamiento del Sistema de medición de Tiempo. Fuente: El Autor.

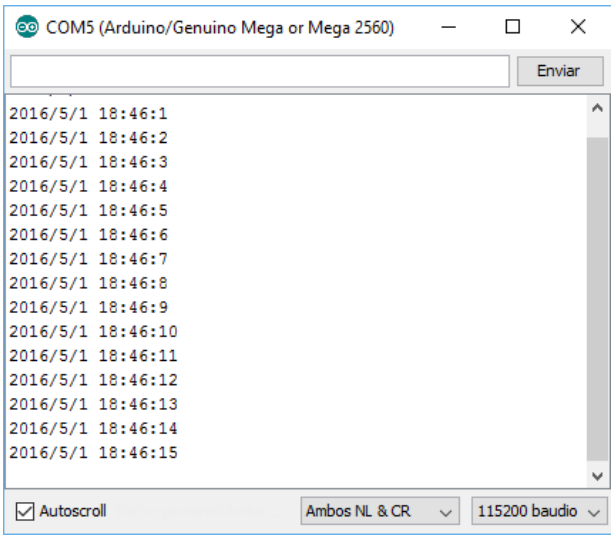


Figura 79. Pantalla de las Pruebas realizadas al Sistema de medición de Tiempo. Fuente: El Autor.

Para comprobar el funcionamiento del Sistema Actuador se utilizó una laptop conectada a la placa Arduino vía comunicación serial la cual simula una señal para activar el servomotor instalado en el dispensador y mediante un movimiento realiza un giro continuo como se indica en la Figura 80.

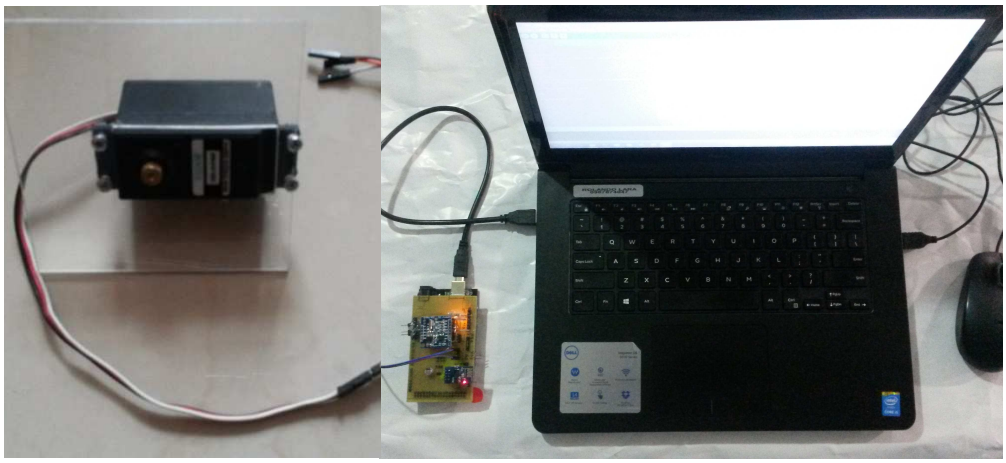


Figura 80. Pruebas de funcionamiento del Sistema Actuador. Fuente: El Autor.

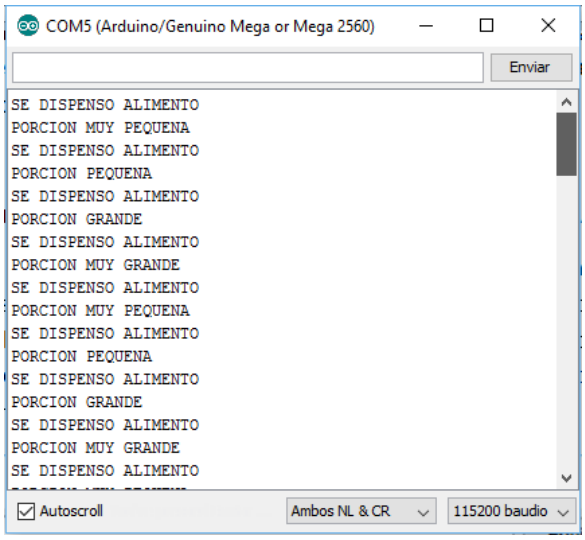


Figura 81. Pantalla de las Pruebas realizadas al Sistema Actuador. Fuente: El Autor.

Para comprobar el funcionamiento del Sistema de Comunicación se utilizó un programa con el cual se conecta a WIFI con el módulo ESP8266 que proporciona internet en el hogar mediante la contraseña antes programada en la placa Arduino, se realizó pruebas en las cuales se puede visualizar cuando el modulo no responde, cuando no puede conectar a WIFI y cuando está conectado a WIFI como se indica en la Figura 82, Figura 83, Figura 84 y Figura 85.

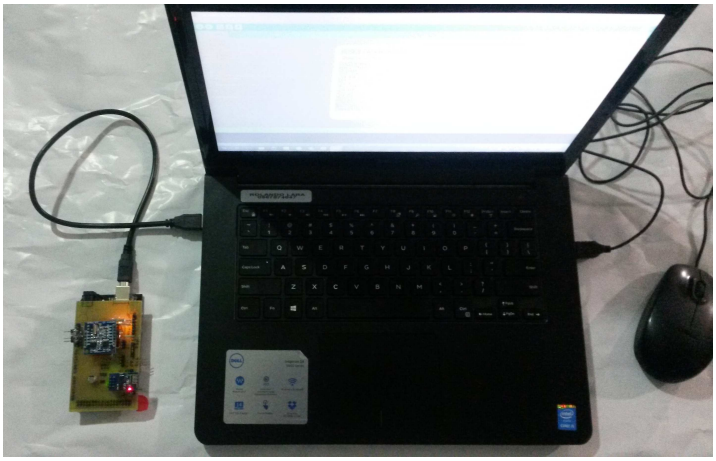


Figura 82. Pruebas de funcionamiento del Sistema de Comunicación. Fuente: El Autor.

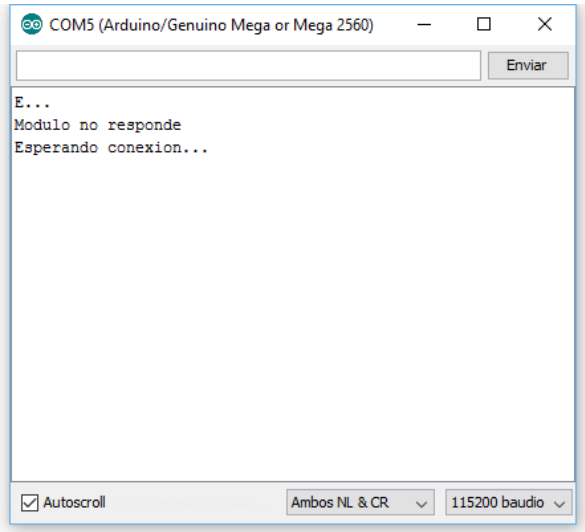


Figura 83. Pantalla de las Pruebas realizadas al Sistema de Comunicación "Modulo no responde". Fuente: El Autor.

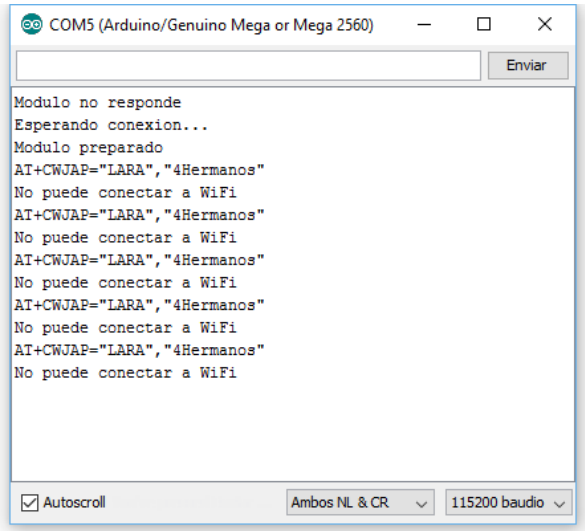


Figura 84. Pantalla de las Pruebas realizadas al Sistema de Comunicación "No puede conectar a Wifi". Fuente: El Autor.

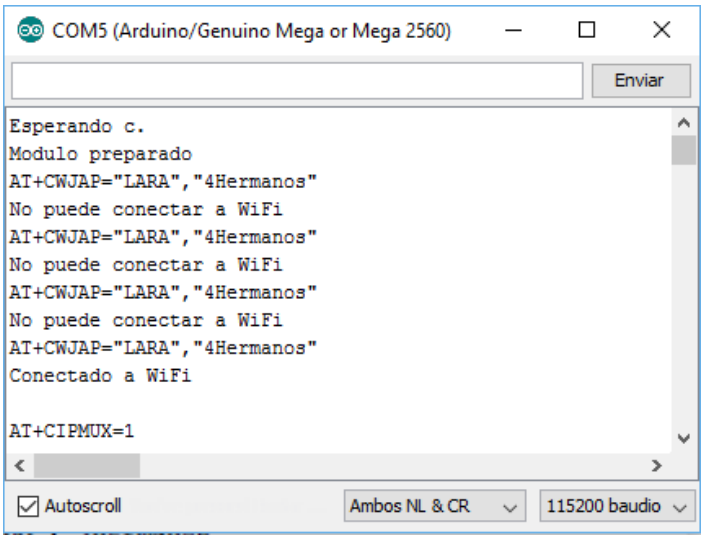


Figura 85. Pantalla de las Pruebas realizadas al Sistema de Comunicación "Conectado a Wifi". Fuente: El Autor.

Para comprobar el funcionamiento de la Aplicación Android se instaló en un Smartphone Samsung Galaxy S4 mini y se realizó las pruebas como ingreso de datos del usuario, datos de la mascota, envío y recepción de información al dispensador como se indica en la Figura 86.

Figura 86. Pruebas de funcionamiento de la Aplicación Android. Fuente: El Autor.

Para comprobar el funcionamiento de todo el prototipo se enlazo todos los sistemas antes mencionados en un solo súper programa que confina cada una de las entradas y salidas necesarias para que los sistemas funcionen correctamente.

En primera instancia se procede a energizar al prototipo, si el sistema corre perfectamente se verá en la pantalla del prototipo que está conectado a WIFI e indicar la IP a la cual está conectado y que el sistema está preparado, ver Figura 87 y Figura 88.

Figura 87. Pantalla del prototipo indicando Sistema conectado a WIFI. Fuente: El Autor.

Figura 88. Pantalla del prototipo indicando la IP del Dispensador. Fuente: El Autor.

Mediante la aplicación Android instalada en el Smartphone se ingresa a la opción “Datos Usuario” el cual pedirá que se llene la información como: Nombre del Usuario, la IP que el prototipo está conectado y la cuenta Twitter del Usuario como se indica en la Figura 89.

Figura 89. Pantalla "Datos Usuario" de la aplicación Android. Fuente: El Autor.

Una vez completados los datos anteriores se dirige a la Pantalla "Primeros Pasos" en la cual se completara información de la mascota como por ejemplo el tipo de mascota, su nombre, tamaño de raza y edad. Ver Figura 90.

Figura 90. Pantalla "Datos Mascota" de la aplicación Android. Fuente: El Autor.

El programa del prototipo está diseñado para dispensar la cantidad de alimento y en el periodo adecuado necesario según los Datos ingresados.

Es decir que para un tipo de mascota Gato las porciones y el periodo de dispensar alimento son las que se muestran en la

Tabla 6, para este tipo de mascota el tamaño de raza no se lo toma en cuenta ya que todas las razas de gatos tienen similar tamaño.

Tabla 6. Tamaño de porción y periodo a dispensar alimento para un Gato. Fuente: El Autor.

Gato	Edad		Periodo
	2-9 meses	**10 m o mayor**	
Porción muy pequeña	√		3 veces al día
Porción pequeña		√	2 veces al día
Porción grande			
Porción muy grande			

Para un tipo de mascota Perro de 2 a 9 meses las porciones y el periodo de dispensar alimento según su tamaño de raza son las que se muestran en la Tabla 7.

Tabla 7. Tamaño de porción y periodo a dispensar alimento para un Perro de 2 a 9 meses. Fuente: El Autor.

Perro	Tamaño Raza			Edad	Periodo
	Pequeño	**Mediano**	**Grande**	**2-9 meses**	
Porción muy pequeña	√			√	3 veces al día
Porción pequeña		√		√	3 veces al día
Porción grande			√	√	3 veces al día
Porción muy grande					

Para un tipo de mascota Perro de 10 meses o mayor las porciones y el periodo de dispensar alimento según su tamaño de raza son las que se muestran en la Tabla 8.

Tabla 8. Tamaño de porción y periodo a dispensar alimento para un Perro de 10 meses o mayor. Fuente: El Autor.

Perro	Tamaño Raza			Edad	Periodo
	Pequeño	**Mediano**	**Grande**	**10 m o mayor**	
Porción muy pequeña					
Porción pequeña	√			√	2 veces al día
Porción grande		√		√	2 veces al día
Porción muy grande			√	√	2 veces al día

El periodo de dispensar cada día se lo realizará como se indica en la Tabla 9.

Tabla 9. Periodo a dispensar alimento diariamente. Fuente: El Autor.

Periodo	Hora	Hora	Hora
2 veces al día	10:00	17:00	X
3 veces al día	09:00	14:00	19:00

Existe otra opción en la aplicación Android que se denomina “Dispensar Manualmente” la cual es su función es dispensar alimento de forma manual eligiendo la porción que se desea y pulsando el botón “Dispensar” el prototipo al recibir esta información dispensará en el momento deseado ver Figura 91.

Figura 91. Pantalla “Dispensar Manualmente” de la aplicación Android. Fuente: El Autor.

El programa de prototipo también está realizado para enviar la temperatura y humedad del entorno del prototipo cada hora mediante un mensaje de Twitter, ver Figura 92.

Figura 92. Mensaje Twitter sobre Temperatura y Humedad recibido en Smartphone. Fuente: El Autor.

El prototipo de la misma manera envía un mensaje Twitter cuando el dispensador realiza el depósito del alimento según el periodo configurado, ver Figura 93.

Figura 93. Mensaje Twitter sobre alimento dispensado recibido en Smartphone. Fuente: El Autor.

El prototipo también envía un mensaje Twitter cuando el contenedor de alimento del dispensador está por terminarse, ver Figura 94.

Figura 94. Mensaje Twitter sobre contenedor por terminarse recibido en Smartphone. Fuente: El Autor.

5.6. Como utilizar el Dispensador Inteligente.

En la siguiente figura se indica el prototipo terminado con todos sus respectivos sistemas conectados para la realización de las actividades correspondientes.

Figura 95. Dispensador Inteligente. Fuente: El Autor.

A continuación se detalla de forma secuencial como utilizar el dispensador inteligente de forma intuitiva para un mejor uso del dispositivo:

- Encender el dispensador inteligente. Ver Figura 96.

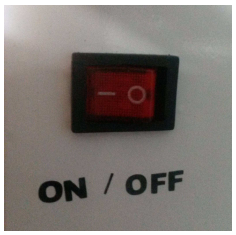


Figura 96. Encendido del Dispensador. Fuente: El Autor.

- Se visualiza la pantalla del Dispensador Indicando que el Dispensador está iniciándose. Ver Figura 97.

Figura 97. Dispositivo Iniciando. Fuente: El Autor.

- Si la pantalla del dispensador te muestra el mensaje como se indica en la Figura 98 esto significa que el Módulo de comunicación no se inició de forma adecuada.

Figura 98. Módulo no Responde. Fuente: El Autor.

- La solución al problema de la Figura 98 es presionar el botón de “REINICIAR” indicado en la Figura 99.

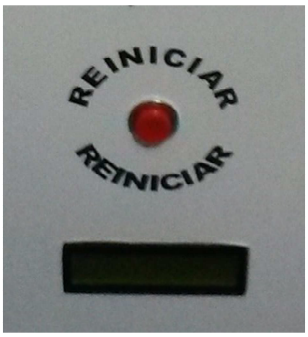


Figura 99. Botón Reinicio del Dispensador. Fuente: El Autor.

- Una vez solucionado el problema o terminando las secuencias anteriores se obtendrá un mensaje en la pantalla del dispensador como se indica en la Figura 100 que se indica que el sistema de comunicación intenta conectarse con el modem de Internet mediante comunicación Wifi.

Figura 100. Dispensador Preparado. Fuente: El Autor.

- Cuando el sistema de comunicación se conecta al modem de internet mediante comunicación Wifi la pantalla muestra el mensaje como se indica en la Figura 101.

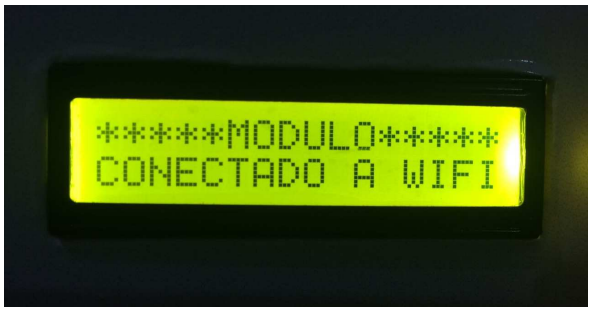


Figura 101. Dispensador conectado a WIFI. Fuente: El Autor.

- Una vez conectado a Internet el sistema del dispensador indica su IP a la cual fue asignado y con la cual se puede comunicar con el Smartphone.

Figura 102. IP del Dispensador. Fuente: El Autor.

Figura 103. Temperatura y Humedad del Dispensador. Fuente: El Autor.

- La aplicación esta creada en formato APK con lo cual al guardarlo en el Smartphone se puede instalar sin dificultades como se indica en la Figura 104.

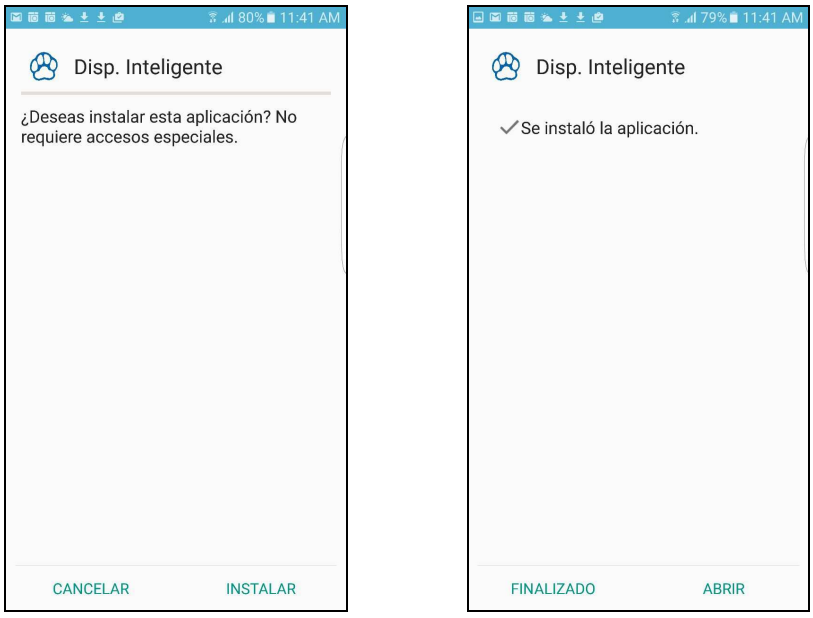


Figura 104. Instalación de la Aplicación Android. Fuente: El Autor.

- Una vez instalada la aplicación en el Smartphone se la inicia y mostrará una pantalla como en la Figura 105.

Figura 105. Pantalla cargando Aplicación. Fuente: El Autor.

- Luego de haberse cargado la aplicación aparecerá un Menú como indica la Figura 106.

Figura 106. Pantalla Menú de Aplicación. Fuente: El Autor.

- Al tocar el botón ***"Datos Usuario"*** enviará a otra pantalla como se indica en la Figura 107.

Figura 107. Pantalla "Datos Usuario" de Aplicación. Fuente: El Autor.

- En la pantalla ***"Datos Usuario"*** se llena la información solicitada como se indica en la Figura 108 y en el espacio que te pide la IP del dispensador escribe la IP que te aparece en la pantalla del dispensador de la Figura 102.

Figura 108. Ingreso de Datos del Usuario. Fuente: El Autor.

- Al tocar el botón ***"Enviar Información al dispensador"*** los datos son enviados al dispensador y si la información llego de forma correcta el dispensador responde como se indica en la Figura 109.

Figura 109. Respuesta del Dispensador en Aplicación. Fuente: El Autor.

- Una vez recibida la información desde la Aplicación Android el dispensador realiza un pequeño saludo al usuario como se indica en la Figura 110.

Figura 110. Saludo al Usuario. Fuente: El Autor.

- Después de realizar esa actividad se regresa al Menú de la aplicación ver Figura 106 y al tocar el botón ***"Datos Mascota"*** enviará a la pantalla como se indica en la Figura 111.

Figura 111. Pantalla "Datos Mascota" de Aplicación. Fuente: El Autor.

- En la pantalla ***"Datos Mascota"*** se llena la información solicitada como se indica en la Figura 112.

Figura 112. Ingreso de Datos de la Mascota. Fuente: El Autor.

- Para el llenado de los datos de la Mascota en el botón ***"Tipo de Mascota"*** que se visualiza en la Figura 112 existe un selector para escoger el tipo de mascota como se indica en la Figura 113. Adicional existe una opción ***(Pruebas)*** la cual se puede escoger para que el dispositivo dispense alimento cada 10 minutos.

Figura 113. Tipo de Mascota. Fuente: El Autor.

- Para el llenado de los datos de la Mascota en el botón ***"Tamaño de Raza"*** que se visualiza en la Figura 112 existe un selector para escoger el tamaño de raza de su mascota como se indica en la Figura 114.

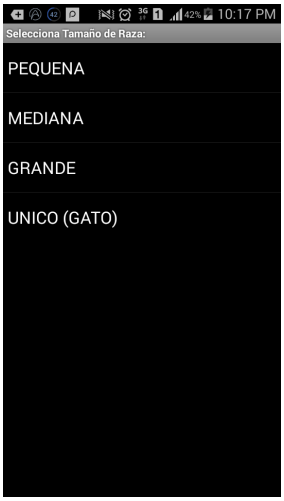


Figura 114. Tamaño de Raza. Fuente: El Autor.

- Para el llenado de los datos de la Mascota en el botón ***"Edad Mascota"*** que se visualiza en la Figura 112 existe un selector para escoger la edad de su mascota como se indica en la Figura 115.

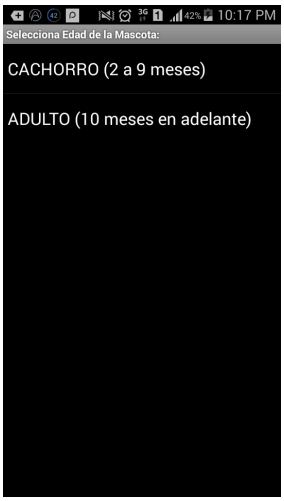


Figura 115. Edad Mascota. Fuente: El Autor.

- Al tocar el botón ***"Enviar Información al dispensador"*** los datos son enviados al dispensador y si la información llego de forma correcta el dispensador responde como se indica en la Figura 116.

Figura 116. Respuesta del Dispensador en Aplicación. Fuente: El Autor.

- Una vez enviada la información al dispensador este indica en su pantalla el nombre y el tipo de la mascota a la cual suministrará alimento, como se indica en la Figura 117.

Figura 117. Nombre y tipo de mascota en dispensador. Fuente: El Autor.

- Después de realizar esa actividad se regresa al Menú de la aplicación ver Figura 106 y al tocar el botón ***"Dispensar Manualmente"*** enviará a la pantalla como se indica en la Figura 118. Con esta pantalla se puede realizar el depósito de alimento con solo seleccionar el tamaño de porción presionando el botón ***"tamaño de Porción"*** como se indica en la Figura 119.

Figura 118. Pantalla "Dispensar Manualmente" de Aplicación. Fuente: El Autor.

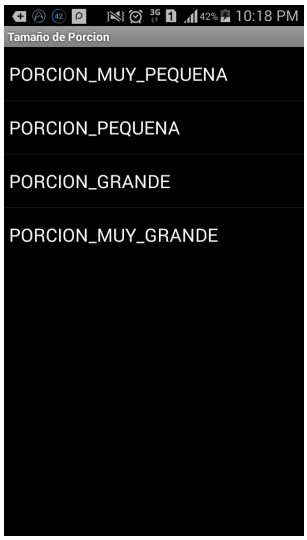


Figura 119. Opciones de Porción de Alimento. Fuente: El Autor.

- Al seleccionar tamaño de porción ***Muy Pequeña*** la pantalla se mostrará como indica la Figura 120, al tocar el botón azul ***(Pulsa para dispensar porción muy pequeña)*** se envía la orden al dispensador.

Figura 120. Dispensar Alimento Porción Muy Pequeña. Fuente: El Autor.

- El dispensador una vez que recibe la información realiza la acción es decir deposita alimento (porción muy pequeña) e indica en su pantalla que acaba de dispensar alimento como se indica en la Figura 121.

Figura 121. Dispenso Porción Muy Pequeña. Fuente: El Autor.

- Al realizar la acción el dispensador envía una respuesta a la aplicación Android indicando que el alimento acaba de ser depositado como se indica en la Figura 122.

Figura 122. Respuesta del Dispensador en Aplicación. Fuente: El Autor.

- Al seleccionar tamaño de porción ***Pequeña*** la pantalla se mostrará como indica la Figura 123, al tocar el botón azul ***(Pulsa para dispensar porción pequeña)*** se envía la orden al dispensador.

Figura 123. Dispensar Alimento Porción Pequeña. Fuente: El Autor.

- El dispensador una vez que recibe la información realiza la acción es decir deposita alimento (porción pequeña) e indica en su pantalla que acaba de dispensar alimento como se indica en la Figura 124.

Figura 124. Dispenso Porción Pequeña. Fuente: El Autor.

- Al realizar la acción el dispensador envía una respuesta a la aplicación Android indicando que el alimento acaba de ser depositado como se indica en la Figura 125.

Figura 125. Respuesta del Dispensador en Aplicación. Fuente: El Autor.

- Al seleccionar tamaño de porción ***grande*** la pantalla se mostrará como indica la Figura 126, al tocar el botón azul ***(Pulsa para dispensar porción grande)*** se envía la orden al dispensador.

Figura 126. Dispensar Alimento Porción Grande. Fuente: El Autor.

- El dispensador una vez que recibe la información realiza la acción es decir deposita alimento (porción grande) e indica en su pantalla que acaba de dispensar alimento como se indica en la Figura 127.

Figura 127. Dispenso Porción Grande. Fuente: El Autor.

- Al realizar la acción el dispensador envía una respuesta a la aplicación Android indicando que el alimento acaba de ser depositado como se indica en la Figura 128.

Figura 128. Respuesta del Dispensador en Aplicación. Fuente: El Autor.

- Al seleccionar tamaño de porción ***muy grande*** la pantalla se mostrará como indica la Figura 129, al tocar el botón azul ***(Pulsa para dispensar porción muy grande)*** se envía la orden al dispensador.

Figura 129. Dispensar Alimento Porción Muy Grande. Fuente: El Autor.

- El dispensador una vez que recibe la información realiza la acción es decir deposita alimento (porción muy grande) e indica en su pantalla que acaba de dispensar alimento como se indica en la Figura 130.

Figura 130. Dispenso Porción Muy Grande. Fuente: El Autor.

- Al realizar la acción el dispensador envía una respuesta a la aplicación Android indicando que el alimento acaba de ser depositado como se indica en la Figura 131.

Figura 131. Respuesta del Dispensador en Aplicación. Fuente: El Autor.

- Cuando el contenedor de alimento está por terminarse el dispensador indica en su pantalla un mensaje para alertar al usuario como se indica en la Figura 132.

Figura 132. El contenedor está por terminarse. Fuente: El Autor.

- El dispensador tiene la característica de enviar mensajes a Twitter indicando en su pantalla como se muestra en la Figura 133.

Figura 133. Enviando Mensaje a Twitter. Fuente: El Autor.

- El dispensador envía mensajes a Twitter cuando:
    - Acaba de depositar alimento (Figura 93).
    - Indica la temperatura y humedad del entorno del dispensador cada cierto periodo (Figura 92).
    - El contenedor de alimento está por terminarse (Figura 94).

- Después de realizar esa actividad se regresa al Menú de la aplicación ver Figura 106 y al tocar el botón ***"Biblioteca"*** enviará a la pantalla como se indica en la Figura 134. En esta pantalla se puede encontrar información relevante sobre el cuidado y alimentación de tu mascota.

Figura 134. Pantalla "Biblioteca" de la aplicación. Fuente: El Autor.

- Después de realizar esa actividad se regresa al Menú de la aplicación ver Figura 106 y al tocar el botón ***"Acerca de"*** enviará a la pantalla como se indica en la Figura 135 y Figura 136. En esta pantalla se puede encontrar información sobre el Autor de la Aplicación, el Tema del Proyecto y el código QR con la cual se puede descargar la aplicación en otros dispositivos

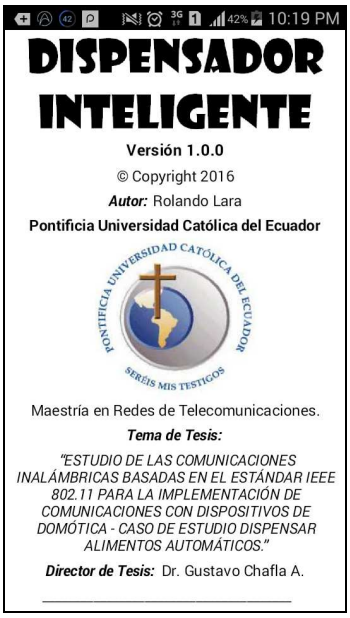


Figura 135. Pantalla "Acerca de" de la aplicación, información Autor y tema de Proyecto. Fuente: El Autor.

Figura 136. Pantalla "Acerca de" de la aplicación, código QR. Fuente: El Autor.

6. Resultados Obtenidos

Para la obtención de los resultados de la investigación se realizaron los siguientes pasos:

- Definición y selección de la muestra.
- Recolección de datos.
- Análisis de datos y elaboración de resultados.

6.1. Definición y selección de la muestra.

La primera muestra se obtuvo a partir de un periodo de trabajo de 15 días del dispensador con la configuración automática inicial como base de: **UN PERRO ADULTO DE RAZA GRANDE,** el dispensado se lo configuro con la ayuda de la aplicación Android como se indica en la Figura 137 con lo cual el dispensador depositará alimento dos veces al día, obteniendo en este periodo 30 depósitos de alimento para su análisis.

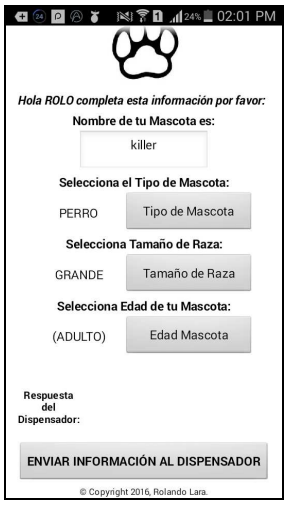


Figura 137. Configuración automática del dispensador para obtención de resultados. Fuente: El Autor.

La segunda muestra se obtuvo a partir de la configuración manual del dispensador con el cual se deposita alimento cada vez que se toca el pulsador azul como se indica en la Figura 138, se realizó 120 pulsaciones para que el dispensador deposite alimento.

Figura 138. Configuración manual del dispensador para obtención de resultados. Fuente: El Autor.

6.2. Recolección de Datos.

Los datos de la primera muestra se recogieron desde el 16 hasta el 30 de septiembre del 2016, la señal Wifi con la que se realizó la toma de muestras se encuentra en el canal 11 a 2462MHZ como se indica en la Figura 139.

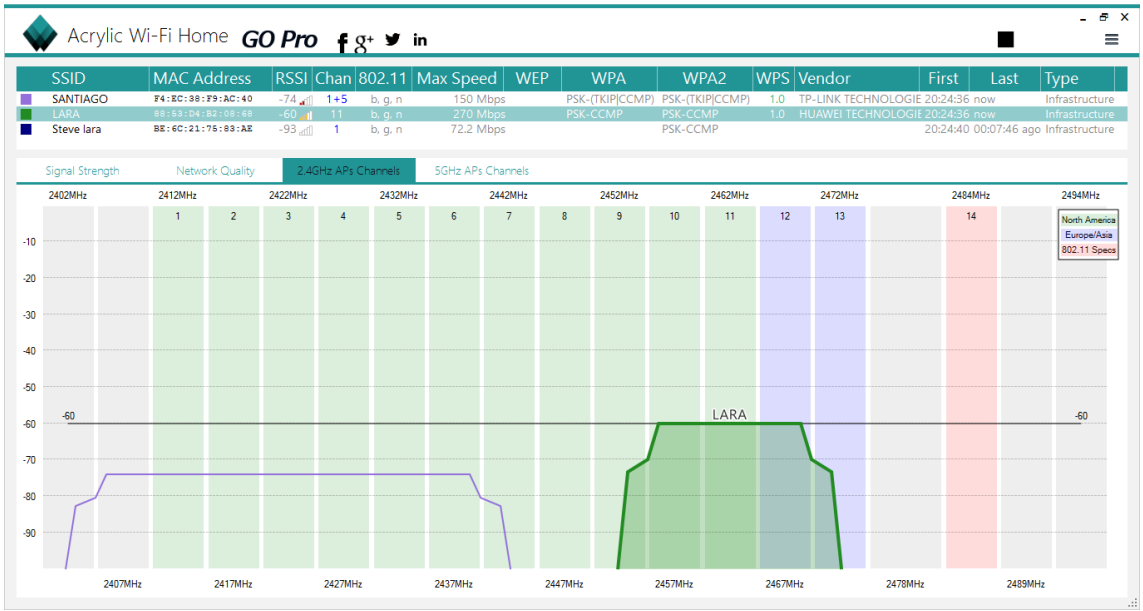


Figura 139. Datos relevantes de la señal Wifi "LARA" para primera muestra. Fuente: El Autor.

Los datos obtenidos con la configuración automática del dispensador se encuentran resumidos en la siguiente tabla.

Tabla 10. Datos obtenidos a partir de configuración automática del dispensador. Fuente: El Autor.

DATOS OBTENIDOS SELECCIÓN AUTOMÁTICA				
DÍA	**FECHA**	**HORA**	**ALIMENTO DISPENSADO**	**TWITTER ENVIADO**
1	16/9/2016	10:00	SI	SI
1	16/9/2016	17:00	SI	SI
2	17/9/2016	10:00	SI	SI
2	17/9/2016	17:00	SI	SI
3	18/9/2016	10:00	SI	SI
3	18/9/2016	17:00	SI	SI
4	19/9/2016	10:00	SI	NO
4	19/9/2016	17:00	SI	SI
5	20/9/2016	10:00	SI	SI
5	20/9/2016	17:00	SI	SI
6	21/9/2016	10:00	SI	SI
6	21/9/2016	17:00	SI	SI
7	22/9/2016	10:00	SI	SI
7	22/9/2016	17:00	SI	SI
8	23/9/2016	10:00	SI	SI
8	23/9/2016	17:00	SI	SI
9	24/9/2016	10:00	SI	SI
9	24/9/2016	17:00	SI	SI
10	25/9/2016	10:00	SI	NO
10	25/9/2016	17:00	SI	SI
11	26/9/2016	10:00	SI	SI

11	26/9/2016	17:00	SI	SI
12	27/9/2016	10:00	SI	SI
12	27/9/2016	17:00	SI	SI
13	28/9/2016	10:00	SI	NO
13	28/9/2016	17:00	SI	SI
14	29/9/2016	10:00	SI	SI
14	29/9/2016	17:00	SI	SI
15	30/9/2016	10:00	SI	NO
15	30/9/2016	17:00	SI	SI

Los datos de la segunda muestra se recogieron el 28 de octubre del 2016, la señal Wifi con la que se realizó la toma de muestras se encuentra en el canal 6 entre 2432MHZ y 2442MHZ como se indica en la Figura 140.

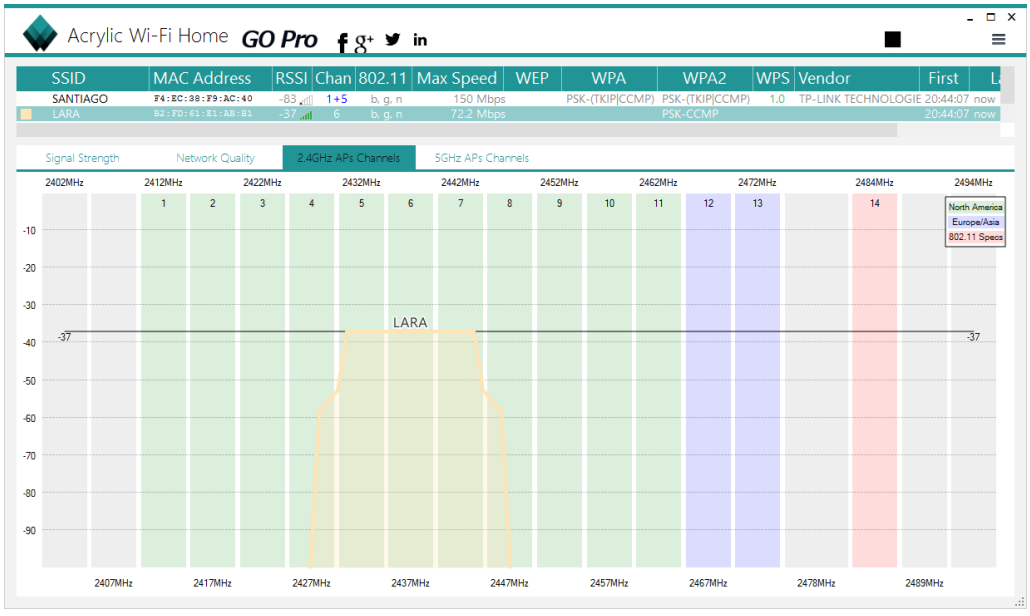


Figura 140. Datos relevantes de la señal Wifi "LARA" para segunda muestra. Fuente: El Autor.

La señal Wifi utilizada para la toma de datos se encuentra en estado “buena” como se indica en la Figura 141.

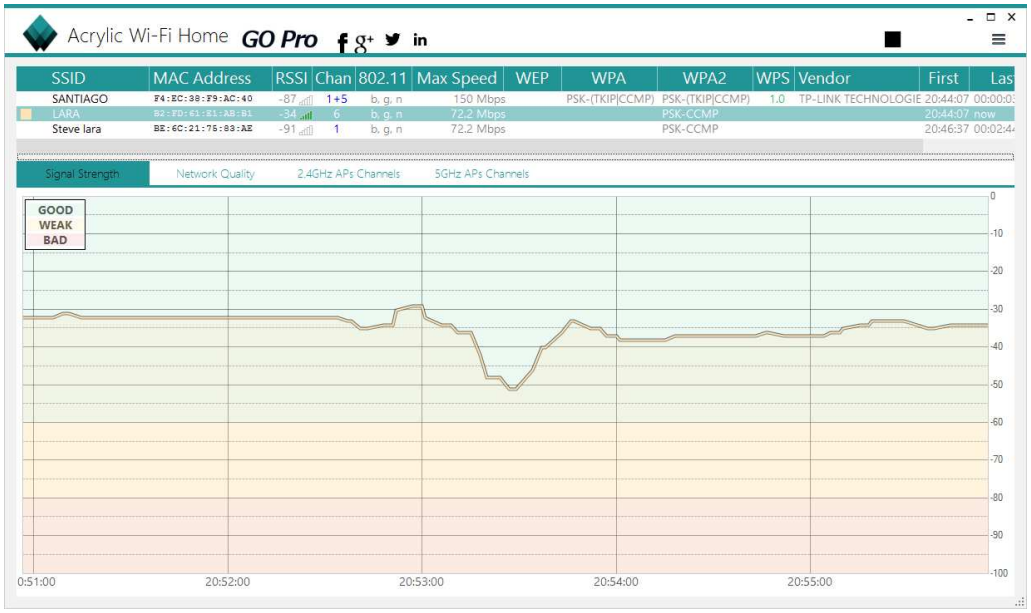


Figura 141. Estado de señal Wifi “LARA” (buena) para segunda muestra. Fuente: El Autor.

Los datos obtenidos con la configuración manual del dispensador se encuentran resumidos en la siguiente tabla.

Tabla 11. Datos obtenidos a partir de configuración manual del dispensador. Fuente: El Autor.

DATOS OBTENIDOS DE SELECCIÓN MANUAL		
SOLICITUD ENVIADA DESDE LA APLICACIÓN	**INFORMACIÓN RECIBIDA**	**INFORMACIÓN NO RECIBIDA**
1	√	-
2	√	-
3	√	-
4	√	-
5	-	√
6	√	-
7	√	-
8	√	-
9	√	-

10	√	-
11	√	-
12	√	-
13	√	-
14	-	√
15	√	-
16	√	-
17	√	-
18	√	-
19	√	-
20	-	√
21	√	-
22	√	-
23	-	√
24	√	-
25	√	-
26	√	-
27	√	-
28	√	-
29	√	-
30	√	-
31	-	√
32	√	-
33	√	-
34	√	-
35	-	√
36	√	-
37	√	-
38	√	-
39	√	-
40	√	-
41	-	√
42	√	-
43	√	-
44	√	-
45	√	-
46	√	-
47	√	-
48	√	-
49	√	-
50	√	-
51	√	-
52	√	-

53	√	-
54	√	-
55	√	-
56	√	-
57	√	-
58	√	-
59	-	√
60	√	-
61	√	-
62	√	-
63	√	-
64	√	-
65	-	√
66	√	-
67	√	-
68	√	-
69	√	-
70	√	-
71	√	-
72	√	-
73	√	-
74	-	√
75	√	-
76	√	-
77	-	√
78	√	-
79	√	-
80	√	-
81	√	-
82	-	√
83	√	-
84	√	-
85	-	√
86	√	-
87	√	-
88	√	-
89	√	-
90	√	-
91	-	√
92	√	-
93	√	-
94	√	-
95	√	-

96	√	-
97	√	-
98	-	√
99	√	-
100	√	-
101	√	-
102	√	-
103	√	-
104	√	-
105	√	-
106	√	-
107	-	√
108	√	-
109	√	-
110	√	-
111	√	-
112	√	-
113	-	√
114	√	-
115	√	-
116	√	-
117	√	-
118	√	-
119	-	√
120	√	-
121	√	-
122	√	-
123	√	-
124	√	-
125	-	√
126	√	-
127	√	-
128	√	-
129	√	-
130	-	√
131	√	-
132	√	-
133	√	-
134	√	-
135	√	-
136	√	-
137	√	-
138	-	√

139	√	-
140	√	-
141	√	-
142	√	-
143	√	-
144	√	-
145	√	-
146	√	-
147	√	-
148	√	-
149	√	-
150	√	-
151	-	√
152	√	-
153	√	-
154	√	-
155	√	-
156	√	-
157	-	√
158	√	-
159	√	-
160	√	-
161	-	√
162	√	-
163	√	-
164	-	√
165	√	-
166	√	-
167	√	-
168	√	-
169	√	-
170	√	-
171	-	√
172	√	-
173	√	-
174	√	-
175	√	-
176	√	-
177	√	-
178	√	-
179	√	-
180	√	-
181	√	-

182	-	√
183	√	-
184	√	-
185	√	-
186	√	-
187	√	-
188	√	-
189	√	-
190	√	-
191	√	-
192	√	-
193	-	√
194	√	-
195	√	-
196	√	-
197	√	-
198	√	-
199	√	-
200	-	√
201	√	-
202	√	-
203	√	-
204	√	-
205	√	-
206	√	-
207	√	-
208	√	-
209	√	-
210	√	-
211	√	-
212	√	-
213	-	√
214	√	-
215	√	-
216	√	-
217	√	-
218	√	-
219	√	-
220	√	-

6.3. Análisis de datos y elaboración de resultados.

Los datos obtenidos de la primera muestra indican que la mayoría de los mensajes de Twitter fueron enviados correctamente, con una muestra de 30 datos obtenidos en 15 días, dos datos enviados por día, a las 10:00 y a las 17:00, las 30 solicitudes fueron realizadas con el correcto depósito de alimento sin embargo de las 30 solicitudes de envío de mensajes a twitter, 26 fueron enviadas correctamente y 4 mensajes no fueron recibidos por el Usuario como se indica en la Figura 142, lo que equivale a un resultado de un 86,67% de efectividad en el envío de forma correcta de mensajes de twitter al Usuario como se indica en la Figura 143.

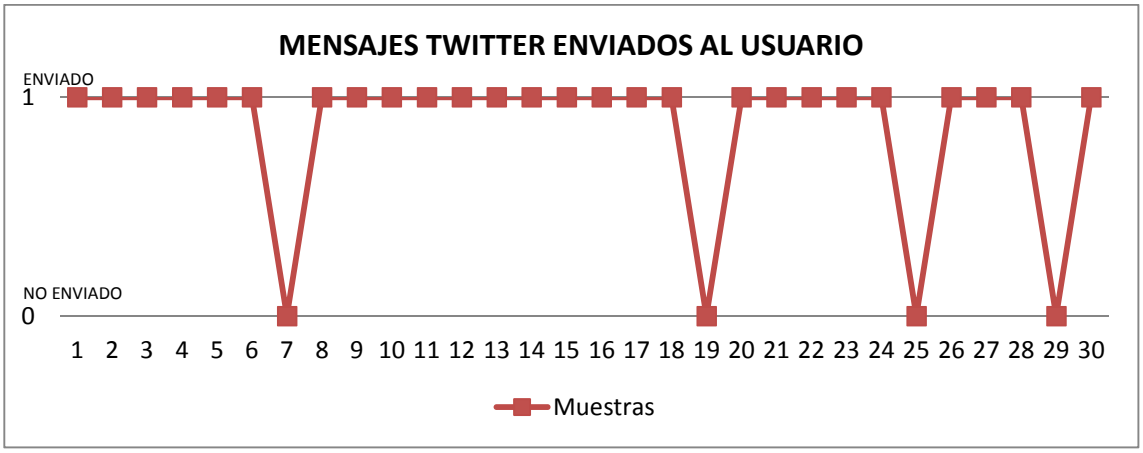


Figura 142. Datos de mensajes Twitter enviados al Usuario utilizando primera muestra. Fuente: El Autor.

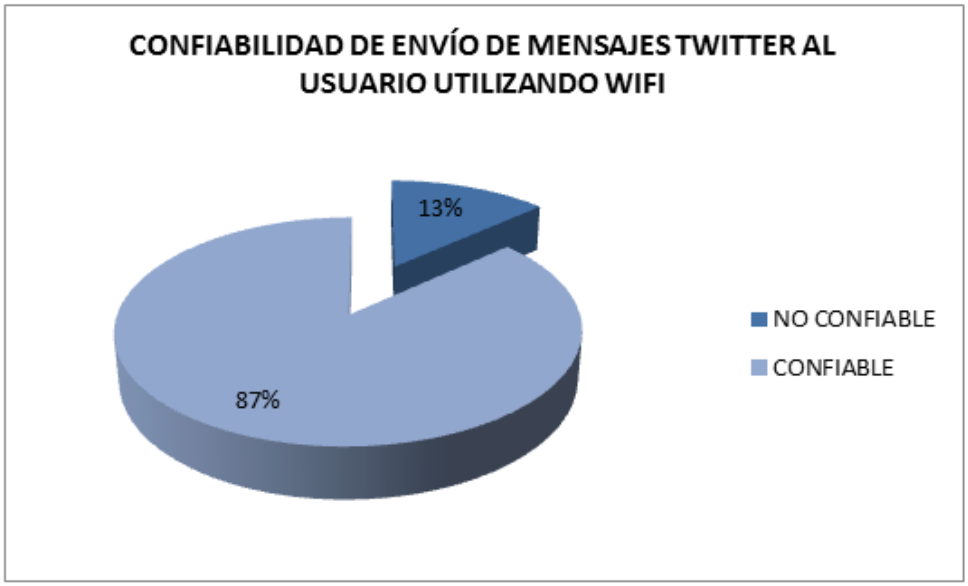


Figura 143. Porcentaje de confiabilidad de Wifi utilizando primera muestra. Fuente: El Autor.

Los datos obtenidos de la segunda muestra indican que la mayoría de las solicitudes de dispensar manualmente fueron recibidas correctamente, con una muestra de 220 datos obtenidos en un solo día, 189 solicitudes fueron realizadas con el correcto depósito de alimento sin embargo 31 solicitudes no fueron recibidas correctamente como se indica en la Figura 144, lo que equivale a un resultado de un 85,90% de efectividad en el envío de forma correcta de solicitudes para depósito de alimento como se indica en la Figura 145.

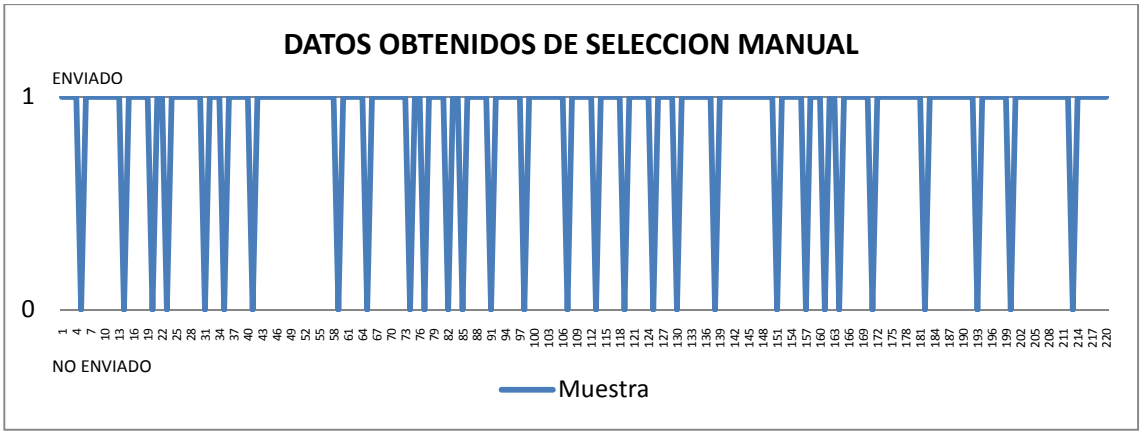


Figura 144. Datos de solicitudes enviados al dispensador utilizando segunda muestra. Fuente: El Autor.

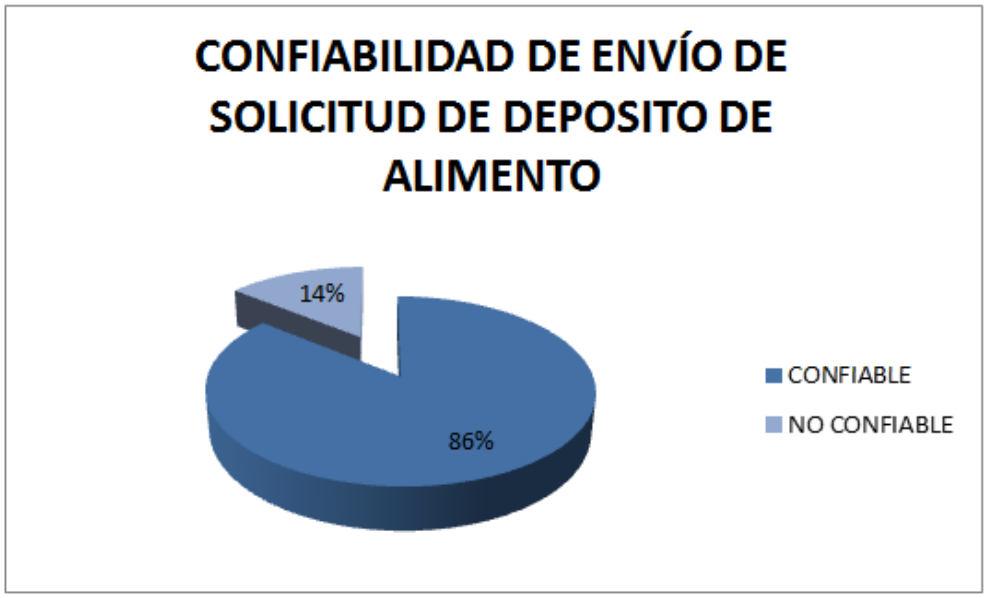


Figura 145. Porcentaje de confiabilidad de Wifi utilizando segunda muestra. Fuente: El Autor.

7. Conclusiones y Recomendaciones

7.1. Conclusiones

- ✓ Se diseñó y construyó un prototipo dispensador de alimentos automatizado con el cual se logró eliminar la intervención humana en la tarea de alimentación de mascotas.
- ✓ Se diseñó y construyó un sistema de comunicación inalámbrica estándar IEEE 802.11 para realizar un monitoreo remoto del prototipo dispensador de alimento utilizando envió de notificaciones lo cual permitió verificar si el dispensador depositaba el alimento, cuando el contenedor estaba por terminarse, la temperatura y humedad del dispensador automático.
- ✓ Se diseñó y construyó un sistema de medición de temperatura y humedad del entorno del prototipo dispensador de alimento mediante un sensor térmico y un sensor capacitivo y de esta manera la información obtenida ingresa a la placa Arduino para su proceso y envío de la información al Usuario mediante mensajes Twitter.
- ✓ Se desarrolló una aplicación móvil basada en tecnología Android que permite la comunicación con el prototipo dispensador de alimento para realizar configuraciones y control del mismo.
- ✓ Se evaluó el comportamiento de la comunicación inalámbrica estándar IEEE 802.11 para el monitoreo y control del prototipo dispensador de alimento utilizando diferentes muestras para dos pruebas la primera para conocer la confiabilidad de envío de mensajes a Twitter y la segunda para dispensar alimento de forma manual con lo cual se obtuvo una confiabilidad de 86,67% y 85,90% lo

que da como media en la confiablidad del estándar IEEE 802,11 de 86,28% que equivale a una confiabilidad aceptable para la realización de este tipo de servicios.

7.2. Recomendaciones

- ✓ Para una comunicación eficiente entre Dispensador-Internet y Dispensador-Aplicación Android se debe programar la placa Arduino con el Usuario y Contraseña para la conexión con el módulo que proporciona Internet en el hogar para de esta manera el módulo ESP8266 reconozca el módulo con el cual se va a conectar.
- ✓ Al prender el Dispensador este te informará si está o no conectado a Wifi y de no ser el caso se debe reiniciar el dispensador apagándolo y volviéndolo a prender para que el sistema se restaure y de esta manera el módulo Wifi se restablezca.
- ✓ El prototipo debe situarse en un ambiente cubierto para evitar daños en el sistema electrónico aunque el prototipo puede resistir polvo y agua se debe evitar que el prototipo este confinado al exceso de estos elementos.
- ✓ El estándar IEEE 802.11 se encuentra en gran avance tecnológico pero no es recomendable utilizarlo para servicios de procesos críticos ya que no es un servicio con alto grado de confiabilidad al momento de enviar y recibir información.

Bibliografía:

ALBACETE, E. P. (2015). *UCLM.* Obtenido de http://www.info-ab.uclm.es/labelec/solar/electronica/elementos/servomotor.htm

ARDUINO. (2015). *ARDUINO GENUINO*. Obtenido de http://www.arduino.cc/en/Products.Compare

ARDUINO. (2015). *ARDUINO GENUINO*. Obtenido de http://www.arduino.cc/en/Main/ArduinoWiFiShield

ARDUINO. (2015). *ARDUINO GENUINO*. Obtenido de http://www.arduino.cc/en/Main/Software,

ARDUINO. (2015). *ARDUINO GENUINO*. Obtenido de http://arduino.cc/en/Main/ArduinoBoardMega2560

ELECTRONICA, J. I. (25 de 01 de 2015). *PALETOS DE LA ELECTRONICA.* Obtenido de https://paletosdelaelectronica.wordpress.com/2015/01/25/interruptores-fin-de-carrera-i/

GONZALEZ, A. G. (8 de FEBRERO de 2014). *PANAMA HITEK*. Obtenido de http://panamahitek.com/dht22-sensor-de-humedadtemperatura-de-precision-para-arduino/.

PATAGONIATEC. (2015). *PATAGONIATEC BLOG*. Obtenido de http://saber.patagoniatec.com/esp8266-modulo-wifi/

PIDELECTRONICS. (20 de 11 de 2015). *PIDELECTRONICS ARDUINO 2560*. Obtenido de http://pidelectronics.com/arduino-mega-2560-r3.html

SINETCOM, G. (2015). *TECTRINIX*. Obtenido de http://www.tectronix.cl/sensor-de-temperatura-y-humedad-dht11.html

SUAREZ, J. I. (2015). *UNIVERSIDAD DE EXTREMADURA.* Obtenido de http://eii.unex.es/profesores/jisuarez/descargas/ip/lcd_alfa.pdf

VISTRONICA. (2015). *VISTRONICA.* Obtenido de TIENDA VIRTUAL ELECTRONICA: http://www.vistronica.com/modulos/modulo-reloj-en-tiempo-real-rtc-ds1307-detail.html

ZAPATA, C. I. (2013). *ESCUELA DE INGENERIA DE ANTOQUIA.* Obtenido de http://repository.eia.edu.co/bitstream/11190/329/1/MECA0111.pdf, Antioquia – Colombia.

Anexos:

Anexo 1:

Fotografías construcción del Dispensador Inteligente.

Figura 146. Construcción del Dispensador (1). Fuente: El Autor.

Figura 147. Construcción del Dispensador (2). Fuente: El Autor.

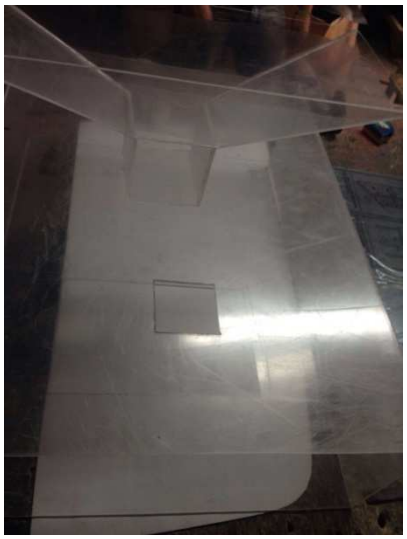


Figura 148. Construcción del Dispensador (3). Fuente: El Autor.

Figura 149. Construcción del Dispensador (4). Fuente: El Autor.

Figura 150. Construcción del Dispensador (5). Fuente: El Autor.

Figura 151. Construcción del Dispensador (6). Fuente: El Autor.

Figura 152. Construcción del Dispensador (7). Fuente: El Autor.

Anexo 2:

Código de Programación para el Sistema Ensamblador Arduino Mega.

```
#include <EEPROM.h>
#include <Servo.h>
Servo miServo;
#include <LCD.h>
#include <LiquidCrystal_I2C.h>
#include <Wire.h>
#define I2C_ADDR    0x27
LiquidCrystal_I2C           lcd(I2C_ADDR,2, 1, 0, 4, 5, 6, 7);
#include "RTClib.h"
RTC_DS1307 RTC;
#include "DHT.h" //carga la librería DHT
#define DHTPIN 7 //Selecciona el pin en el que se //conectará el sensor
#define DHTTYPE DHT11 //Se selecciona el DHT11 (hay //otros DHT)
DHT dht(DHTPIN, DHTTYPE); //Se inicia una variable que será usada por Arduino para
comunicarse con el sensor
#define DEBUG true
#define APIKEY "ANMWSNRABOW9MP3S"//"6383ELOK0YXFH64J"//disp
"2BR31WKNXV102CY6"rolando //  api key your twitter in thingspeak
#define TARGET_IP "184.106.153.149"///        //direccion IP thingspeak
#define TARGET_PORT "80"//---80
```

```
#define SSID "LARA" //El ssid de tu router
#define PASS "4Hermanos" //Tu clave Wifi
String UsuarioIP="";
String NombreUsuario="";
String UsuarioTwitter="";
String NombreMascota="";
String cadena="";
String cadena2="";
String NumeroIP="";
String MensajeTwitter="";
String LCDL1="";
String LCDL2="";
int angulo=10;
int gato=0;
int perro=0;
int Hora = 0;
int Minuto = 0;
int Segundo = 0;
float h;
float t;
int caso1=0;
int caso2=0;
int caso3=0;
```

```
int caso4=0;

int caso5=0;

int caso6=0;

int caso7=0;

int totalcaso=0;

int cada=0;

int cadah=0;

int Ca10=0;

int Ca105=0;

String MensajeServer="";

int Snivel=12;

int Nivel=0;

int Led=4;

int x=0;

int NR=0;

int conectadowifi=0;

int ledrojo=0;

int ledverde=1;

int ledazul=2;

int conled=0;

int estadorojo=0;

int estadoverde=0;

int estadoazul=0;
```

```
int cip=0;
void setup() {
miServo.write(angulo);
pinMode(ledrojo, OUTPUT);
pinMode(ledverde, OUTPUT);
pinMode(ledazul, OUTPUT);
lcd.begin (16,2);    // Inicializar el display con 16 caraceres 2 lineas
lcd.setBacklightPin(3,POSITIVE);
lcd.setBacklight(HIGH);
LCDL1="DISP.INTELIGENTE";
LCDL2="   INICIANDO";
estadorojo=0;
estadoverde=1;
estadoazul=1;
LCDI2C ();
delay (3000);
pinMode(Snivel,INPUT);
pinMode(Led,OUTPUT);
digitalWrite(Led,LOW);
Wire.begin(); // Inicia el puerto I2C (reloj)
RTC.begin(); // Inicia la comunicación con el RTC (reloj)
dht.begin(); //Se inicia el sensor
miServo.attach(3);
```

```
Serial3.begin(115200);
Serial.begin(115200); //can't be faster than 19200 for softserial
Serial3.println("AT+RST");
if(Serial3.find("ready")){
Serial.println("Modulo preparado");
LCDL1="MODULO PREPARADO";
LCDL2=" CONECTANDO.... ";
estadorojo=1;
estadoverde=1;
estadoazul=0;
LCDI2C ();
delay(1000);
boolean connected=false;
for(int i=0;i<5;i++){
  if(connectWiFi()){
   connected = true;
   break;}
}
if (x==5 && connected==false)
  {
LCDL1="MOD. NO RESPONDE";
LCDL2="***REINICIAR****";
estadorojo=0;
```

```
estadoverde=1;
estadoazul=1;
LCDI2C ();
  }
if (!connected){
  while(1);}
delay(5000);
Serial3.println("AT+CIPMUX=1");
delay(1000);
   while ( Serial3.available() ) {
    Serial.write(Serial3.read());
  }
  Serial.print("IP:");
  Serial3.println("AT+CIFSR");
  delay(100);
  while ( Serial3.available()) {
    char e = Serial3.read();
          cadena2 += e;
          int inip=cadena2.indexOf("FSR");
          int finip=cadena2.indexOf("OK");
          Serial.println("CADENA: " + cadena2);
          UsuarioIP=cadena2.substring(inip+6, finip-4);
             Serial.println("IP: " + UsuarioIP);
```

```
        //delay(1000);
  }
LCDL1=" CONECTATE A IP:";
LCDL2="  " + UsuarioIP + "   ";
LCDI2C ();
delay(7000);
//Serial.println("dato IP " +cadena2);
Serial.println("Comenzando Server TCP");
Serial3.println("AT+CIPSERVER=1,8266"); //8266  //Actuar como Server en puerto 8266
delay (1000);
 //Serial3.println("AT+CIFSR");
 //delay(100);
}
else{
Serial.println("Modulo no responde");
LCDL1="MOD. NO RESPONDE";
LCDL2="***REINICIAR****";
estadorojo=0;
estadoverde=1;
estadoazul=1;
LCDI2C ();
delay(5000);
NR=1;
```

```
            }
Serial.println("Esperando conexion...");
  while ( Serial3.available() > 0) {
       char d = Serial3.read();
       cadena2 += d;
       Serial.println("dato IP " +cadena2);
  }
caso1 = EEPROM.read(1);
caso2 = EEPROM.read(2);
caso3 = EEPROM.read(3);
caso4 = EEPROM.read(4);
caso5 = EEPROM.read(5);
caso6 = EEPROM.read(6);
caso7 = EEPROM.read(7);
}
void loop()
{
  while ( Serial3.available()) {
          int inip=cadena2.indexOf("FSR");
          int finip=cadena2.indexOf("OK");
          Serial.println("CADENA: " + cadena2);
          UsuarioIP=cadena2.substring(inip+6, finip-4);
             Serial.println("IP: " + UsuarioIP);
```

```
        //delay(1000);
  }
LCDL1=" CONECTATE A IP:";
LCDL2="  " + UsuarioIP + "   ";
LCDI2C ();
delay(5000);
EEPROM.write(1, caso1);
EEPROM.write(2, caso2);
EEPROM.write(3, caso3);
EEPROM.write(4, caso4);
EEPROM.write(5, caso5);
EEPROM.write(6, caso6);
EEPROM.write(7, caso7);
if (NR==1)
{
LCDL1="MOD. NO RESPONDE";
LCDL2="***REINICIAR****";
estadorojo=0;
estadoverde=1;
estadoazul=1;
LCDI2C ();
estadoleds();
delay (3000);
```

```
}
if (conectadowifi==1)
{
LCDL1="*****MODULO*****";
LCDL2="CONECTADO A WIFI";
estadorojo=1;
estadoverde=0;
estadoazul=1;
LCDI2C ();
delay(5000);
conectadowifi=0;
}
DateTime now = RTC.now(); // Obtiene la fecha y hora del RTC
Hora = now.hour();
Minuto = now.minute();
Segundo = now.second();
h = dht.readHumidity(); //Se lee la humedad
t = dht.readTemperature(); //Se lee la temperatura
char outstr1[15];
dtostrf(t,4, 1, outstr1);
String valor1 = outstr1;   // char to string
char outstr2[15];
dtostrf(h,4, 1, outstr2);
```

```
String valor2 = outstr2;   // char to string
    LCDL1="TEMPERATURA:"+valor1;
    LCDL2="  HUMEDAD:"+valor2;
    estadorojo=1;
    estadoverde=0;
    estadoazul=1;
    LCDI2C ();
 if (cip==0 && Segundo==5)
 {
  LCDL1=" CONECTATE A IP:";
  LCDL2="  " + UsuarioIP + "   ";
  LCDI2C ();
 }
  if (cip==0 && Segundo==15)
 {
  LCDL1="PARA  CONFIGURAR";
  LCDL2="  DISPENSADOR   ";
  LCDI2C ();
 }
    if (cip==0 && Segundo==25)
 {
  LCDL1=" CONECTATE A IP:";
  LCDL2="  " + UsuarioIP + "   ";
```

```
  LCDI2C ();
 }
    if (cip==0 && Segundo==35)
 {
  LCDL1="PARA  CONFIGURAR";
  LCDL2="  DISPENSADOR   ";
  LCDI2C ();
 }
    if (cip==0 && Segundo==45)
 {
  LCDL1=" CONECTATE A IP:";
  LCDL2="  " + UsuarioIP + "   ";
  LCDI2C ();
 }
    if (cip==0 && Segundo==55)
 {
  LCDL1="PARA  CONFIGURAR";
  LCDL2="  DISPENSADOR   ";
  LCDI2C ();
 }
if (digitalRead(Snivel)==1 && Nivel==0)
 {
  Nivel=1;
```

```
  }
if (digitalRead(Snivel)==0)
  {
    Nivel=0;
  }
if (Nivel==1)
  {
     LCDL1="    NIVEL DE    ";
     LCDL2="CONTENEDOR: BAJO";
     estadorojo=0;
     estadoverde=1;
     estadoazul=1;
     LCDI2C ();
     Serial.println("Nivel del contenedor bajo");
     delay (7000);
     ConUnica ();
     MensajeTwitter="El contenedor del Dispensador esta por terminarse T=" + valor1 + " C y
H=" + valor2 + " " + UsuarioTwitter;
     enviartwitter();
     ConMultiple ();
     Nivel=2;

  }
```

```
while (Serial3.available() >0 )
{       char c = Serial3.read();
        cadena += c; //"este es el contenido dejavascript:void(0); la variable";
        int tipogato=cadena.indexOf("GATO");
        int tipoperro=cadena.indexOf("PERRO");
        int tamanopequeno=cadena.indexOf("PEQUENA");
        int tamanomediano=cadena.indexOf("MEDIANA");
        int tamanogrande=cadena.indexOf("GRANDE");
        int edadcachorro=cadena.indexOf("CACHORRO");
        int edadadulto=cadena.indexOf("ADULTO");
        int prueba=cadena.indexOf("Pruebas");
        int dispensar=cadena.indexOf("DISPENSAR");
        int PorcionMP=cadena.indexOf("PORCION_MUY_PEQUENA");
        int PorcionP=cadena.indexOf("PORCION_PEQUENA");
        int PorcionG=cadena.indexOf("PORCION_GRANDE");
        int PorcionMG=cadena.indexOf("PORCION_MUY_GRANDE");
        int InicioNomMascota=cadena.indexOf("/IN");
        int FinNomMascota=cadena.indexOf("FIN");
        int InicioNomUs=cadena.indexOf("INUS");
        int EntUT=cadena.indexOf("ENTRE");
        int FinUT=cadena.indexOf("FINTW");
        digitalWrite(ledrojo, HIGH);
        digitalWrite(ledverde, HIGH);
```

```
digitalWrite(ledazul, LOW);
if (InicioNomUs>0 && FinUT>0)
{
   cip=1;
   NombreUsuario=cadena.substring(InicioNomUs+4, EntUT);
   UsuarioTwitter=cadena.substring(EntUT+5, FinUT);
   Serial.println("Nombre Usuario: " + NombreUsuario);
   Serial.println("Nombre Twitter: " + UsuarioTwitter);
   LCDL1="  HOLA "+NombreUsuario;
   LCDL2=" " +UsuarioTwitter;
   LCDI2C ();
   MensajeServer="HOLA " + NombreUsuario + ", LA INFORMACION FUE
RECIBIDA CORRECTAMENTE";
   webserver();
   InicioNomUs=0;
   EntUT=0;
   FinUT=0;
   c=0;
   cadena="";
}
if (tipogato>0 && edadcachorro>0) //----GATO
  {
   NombreMascota=cadena.substring(InicioNomMascota+3, FinNomMascota);
```

```
tipogato=0;
c=0;
Serial.println("tu mascota es un gato cachorro ");
Serial.println("Nombre Usuario: " + NombreUsuario);
Serial.println("Nombre Twitter: " + UsuarioTwitter);
Serial.println("Nombre Mascota: " + NombreMascota);
edadcachorro=0;
cadena="";
caso1=1;
caso2=0;
caso3=0;
caso4=0;
caso5=0;
caso6=0;
caso7=0;
LCDL1=NombreMascota+" ES UN:";
LCDL2=" GATO CACHORRO  ";
LCDI2C ();
delay (5000);
MensajeServer="HOLA " + NombreUsuario + ", INFORMACION RECIBIDA: TU
MASCOTA ES UN GATO CACHORRO";
webserver();
if (tipogato>0 && edadadulto>0)
```

```
NombreMascota=cadena.substring(InicioNomMascota+3, FinNomMascota);
tipogato=0;
c=0;
Serial.println("tu mascota es un gato adulto ");
Serial.println("Nombre Usuario: " + NombreUsuario);
Serial.println("Nombre Twitter: " + UsuarioTwitter);
Serial.println("Nombre Mascota: " + NombreMascota);
edadadulto=0;
cadena="";
caso1=0;
caso2=1;
caso3=0;
caso4=0;
caso5=0;
caso6=0;
caso7=0;
LCDL1=NombreMascota+" ES UN:";
LCDL2="  GATO  ADULTO  ";
LCDI2C ();
delay (5000);
MensajeServer="HOLA " + NombreUsuario + ", INFORMACION RECIBIDA: TU
MASCOTA ES UN GATO ADULTO";
webserver();
```

```
 }


if (tipoperro>0 && tamanopequeno>0 && edadcachorro>0) //---PERRO
 {
   NombreMascota=cadena.substring(InicioNomMascota+3, FinNomMascota);
   tipoperro=0;
   tamanopequeno=0;
   c=0;
   Serial.println("tu mascota es un perro cachorro de raza pequena");
   Serial.println("Nombre Usuario: " + NombreUsuario);
   Serial.println("Nombre Twitter: " + UsuarioTwitter);
   Serial.println("Nombre Mascota: " + NombreMascota);
   edadcachorro=0;
   cadena="";
   caso1=1;
   caso2=0;
   caso3=0;
   caso4=0;
   caso5=0;
   caso6=0;
   caso7=0;
   LCDL1=NombreMascota+" ES UN:";
   LCDL2=" PERRO CACHORRO ";
```

```
LCDI2C ();
delay (5000);
MensajeServer="HOLA " + NombreUsuario +", INFORMACION RECIBIDA: TU
MASCOTA ES UN PERRO CACHORRO DE RAZA PEQUENA";
webserver();
}
if (tipoperro>0 && tamanopequeno>0 && edadadulto>0)
{
NombreMascota=cadena.substring(InicioNomMascota+3, FinNomMascota);
tipoperro=0;
tamanopequeno=0;
c=0;
Serial.println("tu mascota es un perro adulto de raza pequena ");
Serial.println("Nombre Usuario: " + NombreUsuario);
Serial.println("Nombre Twitter: " + UsuarioTwitter);
Serial.println("Nombre Mascota: " + NombreMascota);
edadadulto=0;
cadena="";
caso1=0;
caso2=1;
caso3=0;
caso4=0;
caso5=0;
```

```
caso6=0;
caso7=0;
LCDL1=NombreMascota+" ES UN:";
LCDL2="  PERRO ADULTO  ";
LCDI2C ();
delay (5000);
MensajeServer="HOLA " + NombreUsuario +", INFORMACION RECIBIDA: TU
MASCOTA ES UN PERRO ADULTO DE RAZA PEQUENA";
webserver();
}

if(tipoperro>0 && tamanomediano>0 && edadcachorro>0)
{
NombreMascota=cadena.substring(InicioNomMascota+3, FinNomMascota);
tipoperro=0;
tamanomediano=0;
c=0;
Serial.println("tu mascota es un perro cachorro de raza mediana ");
Serial.println("Nombre Usuario: " + NombreUsuario);
Serial.println("Nombre Twitter: " + UsuarioTwitter);
Serial.println("Nombre Mascota: " + NombreMascota);
edadcachorro=0;
cadena="";
```

```
caso1=0;
caso2=0;
caso3=1;
caso4=0;
caso5=0;
caso6=0;
caso7=0;
LCDL1=NombreMascota+" ES UN:";
LCDL2=" PERRO CACHORRO ";
LCDI2C ();
delay (5000);
MensajeServer="HOLA " + NombreUsuario +", INFORMACION RECIBIDA: TU MASCOTA ES UN PERRO CACHORRO DE RAZA MEDIANA";
webserver();
}
if (tipoperro>0 && tamanomediano>0 && edadadulto>0)
{
NombreMascota=cadena.substring(InicioNomMascota+3, FinNomMascota);
tipoperro=0;
tamanomediano=0;
c=0;
Serial.println("tu mascota es un perro adulto de raza mediana ");
Serial.println("Nombre Usuario: " + NombreUsuario);
```

```
Serial.println("Nombre Twitter: " + UsuarioTwitter);
Serial.println("Nombre Mascota: " + NombreMascota);
edadadulto=0;
cadena="";
caso1=0;
caso2=0;
caso3=0;
caso4=1;
caso5=0;
caso6=0;
caso7=0;
LCDL1=NombreMascota+" ES UN:";
LCDL2="  PERRO ADULTO  ";
LCDI2C ();
delay (5000);
MensajeServer="HOLA " + NombreUsuario +", INFORMACION RECIBIDA: TU
MASCOTA ES UN PERRO ADULTO DE RAZA MEDIANA";
webserver();
}
if(tipoperro>0 && tamanogrande>0 && edadcachorro>0)
{
NombreMascota=cadena.substring(InicioNomMascota+3, FinNomMascota);
tipoperro=0;
```

```
tamanogrande=0;
c=0;
Serial.println("tu mascota es un perro cachorro de raza grande");
Serial.println("Nombre Usuario: " + NombreUsuario);
Serial.println("Nombre Twitter: " + UsuarioTwitter);
Serial.println("Nombre Mascota: " + NombreMascota);
edadcachorro=0;
cadena="";
caso1=0;
caso2=0;
caso3=0;
caso4=0;
caso5=1;
caso6=0;
caso7=0;
LCDL1=NombreMascota+" ES UN:";
LCDL2=" PERRO CACHORRO ";
LCDI2C ();
delay (5000);
MensajeServer="HOLA " + NombreUsuario +", INFORMACION RECIBIDA: TU MASCOTA ES UN PERRO CACHORRO DE RAZA GRANDE";
webserver();
}
```

```
if (tipoperro>0 && tamanogrande>0 &&edadadulto>0)
 {
   NombreMascota=cadena.substring(InicioNomMascota+3, FinNomMascota);
   tipoperro=0;
   tamanogrande=0;
   c=0;
   Serial.println("tu mascota es un perro adulto de raza grande ");
   Serial.println("Nombre Usuario: " + NombreUsuario);
   Serial.println("Nombre Twitter: " + UsuarioTwitter);
   Serial.println("Nombre Mascota: " + NombreMascota);
   edadadulto=0;
   cadena="";
   caso1=0;
   caso2=0;
   caso3=0;
   caso4=0;
   caso5=0;
   caso6=1;
   caso7=0;
   LCDL1=NombreMascota+" ES UN:";
   LCDL2="  PERRO ADULTO  ";
   LCDI2C ();
   delay (5000);
```

```
    MensajeServer="HOLA " + NombreUsuario +", INFORMACION RECIBIDA: TU
MASCOTA ES UN PERRO ADULTO DE RAZA GRANDE";
    webserver();
   }
  if(prueba>0)
   {
    NombreMascota=cadena.substring(InicioNomMascota+3, FinNomMascota);
    prueba=0;
    c=0;
    Serial.println("opcion de prueba ");
    Serial.println("Nombre Usuario: " + NombreUsuario);
    Serial.println("Nombre Twitter: " + UsuarioTwitter);
    Serial.println("Nombre Mascota: " + NombreMascota);
    cadena="";
    cada=now.minute()+1;
    cadah=now.hour();
    caso1=0;
    caso2=0;
    caso3=0;
    caso4=0;
    caso5=0;
    caso6=0;
    caso7=1;
```

```
        LCDL1=NombreUsuario+" ESTAS";
        LCDL2=" EN MODO PRUEBA ";
        LCDI2C ();
        delay (5000);
        MensajeServer="HOLA " + NombreUsuario +", INFORMACION RECIBIDA: ESTAS
EN MODO PRUEBA";
        webserver();
       }
      if (dispensar>0 && PorcionMP>0)
       {
        Serial.println("Dispensar manualmente porcion muy pequena ");
        DispPorcionMP ();
        c=0;
        cadena="";
        dispensar=0;
        PorcionMP=0;
        MensajeServer="HOLA " + NombreUsuario +", SE DISPENSO PORCION MUY
PEQUENA";
        webserver();
       }
      if (dispensar>0 && PorcionP>0)
       {
        Serial.println("Dispensar manualmente porcion pequena ");
```

```
        DispPorcionP ();
        c=0;
        cadena="";
        dispensar=0;
        PorcionP=0;
        MensajeServer="HOLA " + NombreUsuario +", SE DISPENSO PORCION
PEQUENA";
        webserver();
       }
      if (dispensar>0 && PorcionG>0)
       {
        Serial.println("Dispensar manualmente porcion grande ");
        DispPorcionG ();
        c=0;
        cadena="";
        dispensar=0;
        PorcionG=0;
        MensajeServer="HOLA " + NombreUsuario +", SE DISPENSO PORCION
GRANDE";
        webserver();
       }
      if (dispensar>0 && PorcionMG>0)
       {
```

```
            Serial.println("Dispensar manualmente porcion muy grande ");
            DispPorcionMG ();
            c=0;
            cadena="";
            dispensar=0;
            PorcionMG=0;
            MensajeServer="HOLA " + NombreUsuario +", SE DISPENSO PORCION MUY
GRANDE";
            webserver();
            //angulo=50;
            //angulo=angulo+5;
           }
if (caso1==1)
{
  if (Hora==9 && Minuto==0 && Segundo==0)
    {
     DispPorcionMP ();
     ConUnica ();
     MensajeTwitter="Su Dispensador acaba de depositar alimento, " + UsuarioTwitter;
     enviartwitter();
     ConMultiple ();
    }
   if (Hora==14 && Minuto==0 && Segundo==0)
```

```
    {
      DispPorcionMP ();
      ConUnica ();
      MensajeTwitter="Su Dispensador acaba de depositar alimento, " + UsuarioTwitter;
      enviartwitter();
      ConMultiple ();
    }
    if (Hora==19 && Minuto==0 && Segundo==0)
    {
      DispPorcionMP ();
      ConUnica ();
      MensajeTwitter="Su Dispensador acaba de depositar alimento, " + UsuarioTwitter;
      enviartwitter();
      ConMultiple ();
    }
}
if (caso2==1)
{
  if (Hora==10 && Minuto==0 && Segundo==0)
    {
      DispPorcionP ();
      ConUnica ();
      MensajeTwitter="Su Dispensador acaba de depositar alimento, " + UsuarioTwitter;
```

```
    enviartwitter();
    ConMultiple ();
   }
  if (Hora==17 && Minuto==0 && Segundo==0)
   {
    DispPorcionP ();
    ConUnica ();
    MensajeTwitter="Su Dispensador acaba de depositar alimento, " + UsuarioTwitter;
    enviartwitter();
    ConMultiple ();
   }
}
if (caso3==1)
{
  if (Hora==9 && Minuto==0 && Segundo==0)
   {
    DispPorcionP ();
    ConUnica ();
    MensajeTwitter="Su Dispensador acaba de depositar alimento, " + UsuarioTwitter;
    enviartwitter();
    ConMultiple ();
   }
  if (Hora==14 && Minuto==0 && Segundo==0)
```

```
   {
     DispPorcionP ();
     ConUnica ();
     MensajeTwitter="Su Dispensador acaba de depositar alimento, " + UsuarioTwitter;
     enviartwitter();
     ConMultiple ();
   }
  if (Hora==19 && Minuto==0 && Segundo==0)
   {
     DispPorcionP ();
     ConUnica ();
     MensajeTwitter="Su Dispensador acaba de depositar alimento, " + UsuarioTwitter;
     enviartwitter();
     ConMultiple ();
   }
}
if (caso4==1)
{
  if (Hora==10 && Minuto==0 && Segundo==0)
   {
     DispPorcionG ();
     ConUnica ();
     MensajeTwitter="Su Dispensador acaba de depositar alimento, " + UsuarioTwitter;
```

```
    enviartwitter();
    ConMultiple ();
   }
  if (Hora==17 && Minuto==0 && Segundo==0)
   {
    DispPorcionG ();
    ConUnica ();
    MensajeTwitter="Su Dispensador acaba de depositar alimento, " + UsuarioTwitter;
    enviartwitter();
    ConMultiple ();
   }
}
if (caso5==1)
{
  if (Hora==9 && Minuto==0 && Segundo==0)
   {
    DispPorcionG ();
    ConUnica ();
    MensajeTwitter="Su Dispensador acaba de depositar alimento, " + UsuarioTwitter;
    enviartwitter();
    ConMultiple ();
   }
  if (Hora==14 && Minuto==0 && Segundo==0)
```

```
   {
     DispPorcionG ();
     ConUnica ();
     MensajeTwitter="Su Dispensador acaba de depositar alimento, " + UsuarioTwitter;
     enviartwitter();
     ConMultiple ();
   }
  if (Hora==19 && Minuto==0 && Segundo==0)
   {
     DispPorcionG ();
     ConUnica ();
     MensajeTwitter="Su Dispensador acaba de depositar alimento, " + UsuarioTwitter;
     enviartwitter();
     ConMultiple ();
   }
}
if (caso6==1)
{
  if (Hora==10 && Minuto==0 && Segundo==0)
   {
     DispPorcionMG ();
     ConUnica ();
     MensajeTwitter="Su Dispensador acaba de depositar alimento, " + UsuarioTwitter;
```

```
      enviartwitter();
      ConMultiple ();
    }
  if (Hora==17 && Minuto==0 && Segundo==0)
    {
      DispPorcionMG ();
      ConUnica ();
      MensajeTwitter="Su Dispensador acaba de depositar alimento, " + UsuarioTwitter;
      enviartwitter();
      ConMultiple ();
    }
}
if (caso7==1 && Minuto==0 && Segundo==0)
{
    EnviarDatosSensor ();
    DispPorcionMP ();
    ConUnica ();
    MensajeTwitter="Su Dispensador acaba de depositar alimento T=" + valor1 + " C y H=" +
valor2 + " " + UsuarioTwitter;
    enviartwitter();
    ConMultiple ();
}
if(caso7==1 && Minuto==10 && Segundo==0)
```

```
{
   EnviarDatosSensor ();
   DispPorcionP ();
   ConUnica ();
   MensajeTwitter="Su Dispensador acaba de depositar alimento T=" + valor1 + " C y H=" +
valor2 + " " + UsuarioTwitter;
   enviartwitter();
   ConMultiple ();
}
if(caso7==1 && Minuto==20 && Segundo==0)
{
   EnviarDatosSensor ();
   DispPorcionG ();
   ConUnica ();
   MensajeTwitter="Su Dispensador acaba de depositar alimento T=" + valor1 + " C y H=" +
valor2 + " " + UsuarioTwitter;
   enviartwitter();
   ConMultiple ();
}
if(caso7==1 && Minuto==40 && Segundo==0)
{
   EnviarDatosSensor ();
   DispPorcionMG ();
```

```
   ConUnica ();
   MensajeTwitter="Su Dispensador acaba de depositar alimento T=" + valor1 + " C y H=" +
valor2 + " " + UsuarioTwitter;
   enviartwitter();
   ConMultiple ();
}
if(caso7==1 && Minuto==50 && Segundo==0)
{
   EnviarDatosSensor ();
   DispPorcionMG ();
   //angulo=angulo+5;
   ConUnica ();
   MensajeTwitter="Su Dispensador acaba de depositar alimento T=" + valor1 + " C y H=" +
valor2 + " " + UsuarioTwitter;
   enviartwitter();
   ConMultiple ();
}//
if(Minuto==30 && Segundo==0)
{
   EnviarDatosSensor ();
   DispPorcionG ();
   ConUnica ();
```

```
   MensajeTwitter="Su Dispensador se encuentra a Temperatura=" + valor1 + " C y
Humedad=" + valor2 + " " + UsuarioTwitter;
   enviartwitter();
   ConMultiple ();
}
}
boolean connectWiFi(){
Serial3.println("AT+CWMODE=1");
cmd+=SSID;
cmd+="\",\"";
cmd+=PASS;
cmd+="\"";
Serial.println(cmd);
Serial3.println(cmd);
delay(2000);
if(Serial3.find("OK")){
  Serial.println("Conectado a WiFi");
conectadowifi=1;
conled=1;
  return true;}
else{
  Serial.println("No puede conectar a WiFi");
  x=x+1;
```

```
  return false;}
}
oid DispPorcionMP()
{
for(angulo=10; angulo<70; angulo=angulo+10)
{miServo.write(angulo);
  Serial.print("Angulo:");Serial.println(angulo);
  delay(100);
}
for(angulo=70; angulo>=10; angulo=angulo-10)
{miServo.write(angulo);
  Serial.print("Angulo:");Serial.println(angulo);
  delay(100);
}

  Serial.println("Alimento Dispensado Porcion Muy Pequena");
  LCDL1="DISPENSO PORCION";
  LCDL2="  MUY  PEQUENA  ";
  LCDI2C ();
  delay (2000);
  LCDL1=" PARA " + NombreMascota;
  LCDL2="<<<<<<<..>>>>>>>";
  LCDI2C ();
```

```
  delay (2000);
}
void DispPorcionP()
{
for(angulo=10; angulo<110; angulo=angulo+10)
{miServo.write(angulo);
  Serial.print("Angulo:");Serial.println(angulo);
  delay(100);
}
for(angulo=110; angulo>=10; angulo=angulo-10)
{miServo.write(angulo);
  Serial.print("Angulo:");Serial.println(angulo);
  delay(100);
}

  Serial.println("Alimento Dispensado Porcion Pequena");
  LCDL1="DISPENSO PORCION";
  LCDL2="    PEQUENA     ";
  LCDI2C ();
  delay (2000);
  LCDL1=" PARA " + NombreMascota;
  LCDL2="<<<<<<<..>>>>>>>";
  LCDI2C ();
```

```
  delay (2000);
}
void DispPorcionG()
{
for(angulo=10; angulo<140; angulo=angulo+10)
{miServo.write(angulo);
  Serial.print("Angulo:");Serial.println(angulo);
  delay(100);
}
for(angulo=140; angulo>=10; angulo=angulo-10)
{miServo.write(angulo);
  Serial.print("Angulo:");Serial.println(angulo);
  delay(100);
}
  Serial.println("Alimento Dispensado Porcion Grande");
  LCDL1="DISPENSO PORCION";
  LCDL2="     GRANDE     ";
  LCDI2C ();
  delay (2000);
  LCDL1=" PARA " + NombreMascota;
  LCDL2="<<<<<<<..>>>>>>>";
  LCDI2C ();
  delay (2000);
```

```
}
void DispPorcionMG()
{
for(angulo=10; angulo<170; angulo=angulo+10)
{miServo.write(angulo);
  Serial.print("Angulo:");Serial.println(angulo);
  delay(100);
}
for(angulo=170; angulo>=10; angulo=angulo-10)
{miServo.write(angulo);
  Serial.print("Angulo:");Serial.println(angulo);
  delay(100);
}
  Serial.println("Alimento Dispensado Porcion Muy Grande");
  LCDL1="DISPENSO PORCION";
  LCDL2="   MUY GRANDE   ";
  LCDI2C ();
  delay (2000);
  LCDL1=" PARA " + NombreMascota;
  LCDL2="<<<<<<<..>>>>>>>";
  LCDI2C ();
  delay (2000);
}
```

```
void EnviarDatosSensor()
{
DateTime now = RTC.now();
Serial.print(now.hour(),DEC);
Serial.print(':');
Serial.println(now.minute(),DEC);
Serial.println("Humedad: ");
Serial.println(h);
Serial.println("Temperatura: ");
Serial.println(t);
delay(2000); //Se espera 2 segundos para seguir leyendo //datos
}
void enviartwitter()
{
        LCDL1="ENVIANDO MENSAJE";
        LCDL2="  A  TWITTER  ";
        estadorojo=1;
        estadoverde=1;
        estadoazul=0;
        LCDI2C ();
 for(int x=0; x<=2; x++)
 {
 delay(10000);  //30000/60000
```

```
dtostrf(t,4, 1, outstr1);
String valor1 = outstr1;   // char to string
  char outstr2[15];
dtostrf(h,4, 1, outstr2);
String valor2 = outstr2;   // char to string */
  String webpage = "AT+CIPSTART=\"TCP\",\"";
 webpage += TARGET_IP;
 webpage += "\",80\r\n";
 sendData(webpage,1000,DEBUG);
 String tsData = "api_key=" APIKEY "&status=" + MensajeTwitter ;
 String webpage1 = "POST /apps/thingtweet/1/statuses/update HTTP/1.1\n";
 webpage1+="Host: api.thingspeak.com\n";
 webpage1+="Content-Type: application/x-www-form-urlencoded\n";
 webpage1+="Content-Length: ";
 webpage1+=tsData.length();
 webpage1+="\n\n";
 webpage1+=tsData;
  String cipsend = "AT+CIPSEND=";
  cipsend+= webpage1.length();
  cipsend+="\r\n";
  sendData(cipsend,1000,DEBUG);
  sendData(webpage1,1000,DEBUG);
```

```
    sendData("AT+CIPCLOSE=0\r\n",1500,DEBUG);
    delay(5000);  //100003 seg
  }
}
String sendData(String command, const int timeout, boolean debug)
{
   String response = "";
    Serial3.print(command); // send the read character to the esp8266
    long int time = millis();
    while( (time+timeout) > millis())
   {
     while(Serial3.available())
     {
       char c = Serial3.read(); // read the next character.
       response+=c;
     }
   }
    if(debug)
   {
     Serial.print(response);
   }
    return response;
}
```

```
void http(String output)
{
   Serial3.print("AT+CIPSEND=0,");            // AT+CIPSEND=0, num
   Serial3.println(output.length());
   if (Serial3.find(">"))                    // recibe la peticion del mensaje
     {   Serial.println(output);
         Serial3.println(output);          //Aqui va el string hacia el WIFI
         delay(10);
         while ( Serial3.available() > 0 )
          { if (  Serial3.find("SEND OK") )    // Busca el OK en la respuesta del wifi
                 break;
          }
     }
}
void webserver(void)
   {
      for (int i=0; i<=1; i++)
      {
        http("<!DOCTYPE HTML>");
        http("<html>");
        http("<head><title>DISPENSADOR INTELIGENTE.</title>");
        delay(1);
        }
```

```
        Serial3.println("AT+CIPCLOSE=0");
   }
void ConUnica ()
 {
        Serial3.println("AT+CIPMUX=0");
        delay(1000);
        while ( Serial3.available() ) {
        Serial.write(Serial3.read());}
 }
void ConMultiple ()
 {
        Serial3.println("AT+CIPMUX=1");
        delay(100);
        while ( Serial3.available() ) {
        Serial.write(Serial3.read());}
 }
void LCDI2C ()
{
          digitalWrite(ledrojo, estadorojo);
          digitalWrite(ledverde, estadoverde);
          digitalWrite(ledazul, estadoazul);
          lcd.clear(); //limpia el LCD
          lcd.home ();              // go home primera linea
```

```
        lcd.print(LCDL1);
        lcd.setCursor ( 0, 1 );        // go to the 2nd line
        lcd.print(LCDL2);
}
void estadoleds ()
{
        digitalWrite(ledrojo, estadorojo);
        digitalWrite(ledverde, estadoverde);
        digitalWrite(ledazul, estadoazul);
}
```

Anexo 3:

Fotografías pruebas de Funcionamiento del Dispensador Inteligente.

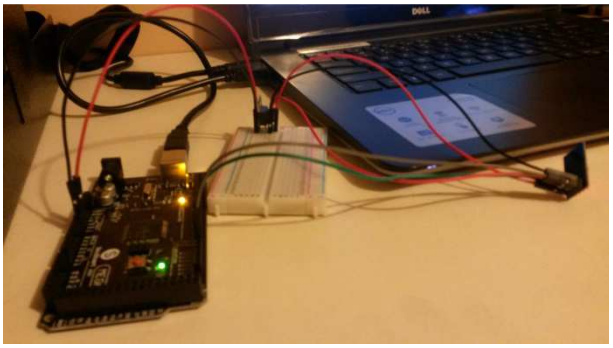


Figura 153. Pruebas de Funcionamiento de los sistemas del Dispensador (1). Fuente: El Autor.

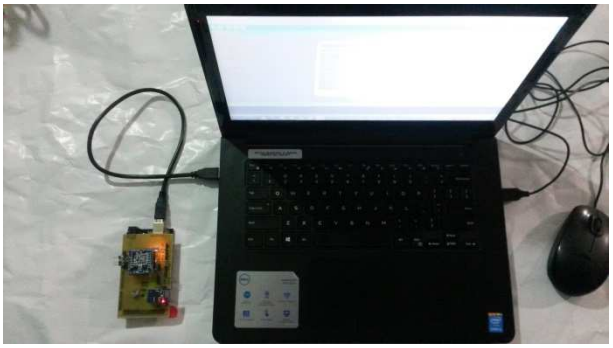


Figura 154. Pruebas de Funcionamiento de los sistemas del Dispensador (2). Fuente: El Autor.

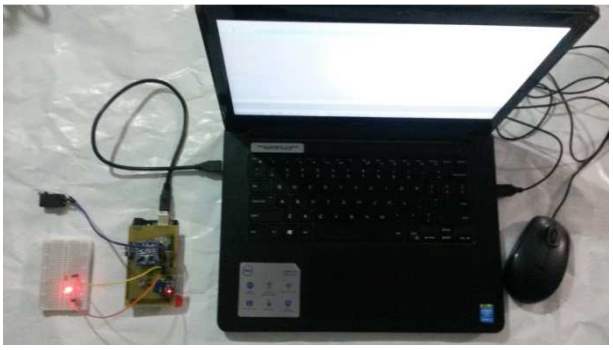


Figura 155. Pruebas de Funcionamiento de los sistemas del Dispensador (3). Fuente: El Autor.

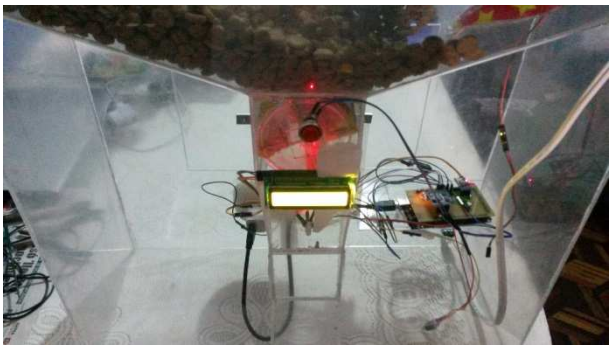


Figura 156. Pruebas de Funcionamiento de los sistemas del Dispensador (4). Fuente: El Autor.

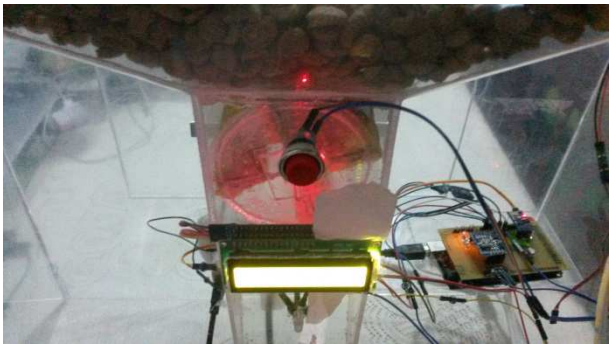


Figura 157. Pruebas de Funcionamiento de los sistemas del Dispensador (5). Fuente: El Autor.

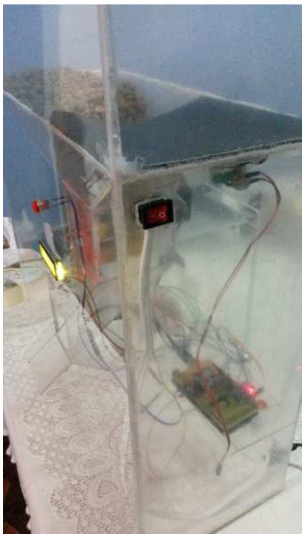


Figura 158. Pruebas de Funcionamiento de los sistemas del Dispensador (6). Fuente: El Autor.

Figura 159. Pruebas de Funcionamiento de los sistemas del Dispensador (7). Fuente: El Autor.

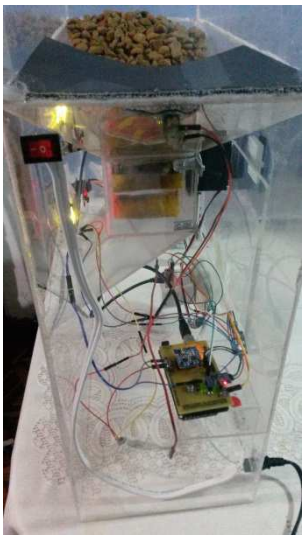


Figura 160. Pruebas de Funcionamiento de los sistemas del Dispensador (8). Fuente: El Autor.

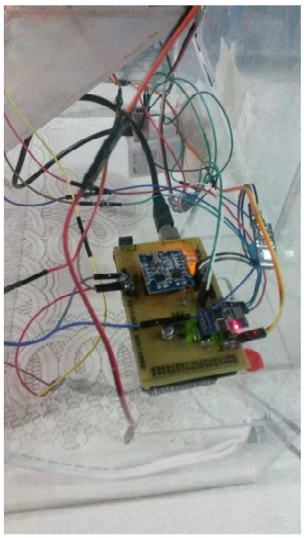


Figura 161. Pruebas de Funcionamiento de los sistemas del Dispensador (9). Fuente: El Autor.

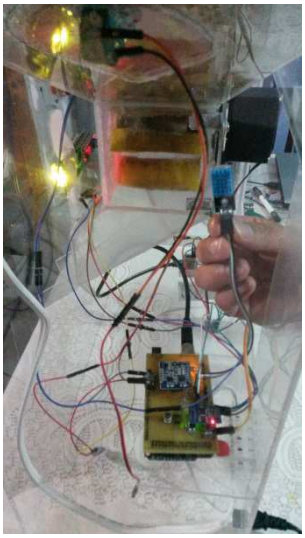


Figura 162. Pruebas de Funcionamiento de los sistemas del Dispensador (10). Fuente: El Autor.

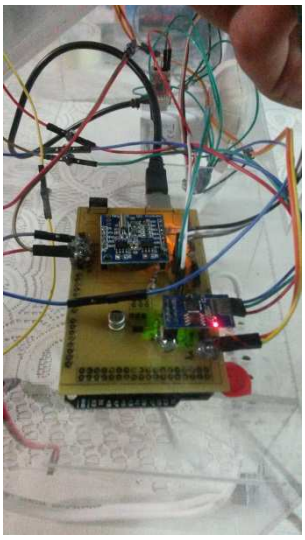


Figura 163. Pruebas de Funcionamiento de los sistemas del Dispensador (11). Fuente: El Autor.

Figura 164. Pruebas de Funcionamiento de los sistemas del Dispensador (12). Fuente: El Autor.

Figura 165. Pruebas de Funcionamiento de los sistemas del Dispensador (13). Fuente: El Autor.

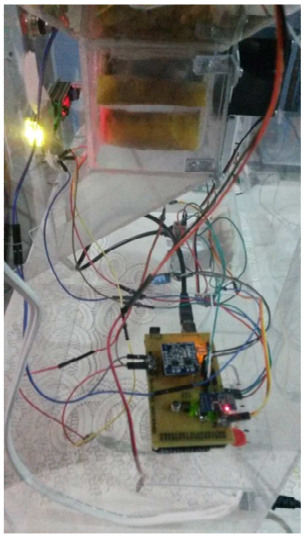


Figura 166. Pruebas de Funcionamiento de los sistemas del Dispensador (14). Fuente: El Autor.

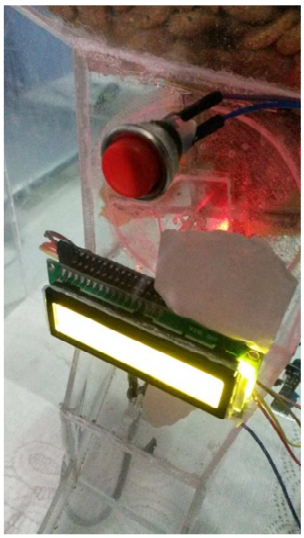


Figura 167. Pruebas de Funcionamiento de los sistemas del Dispensador (15). Fuente: El Autor.

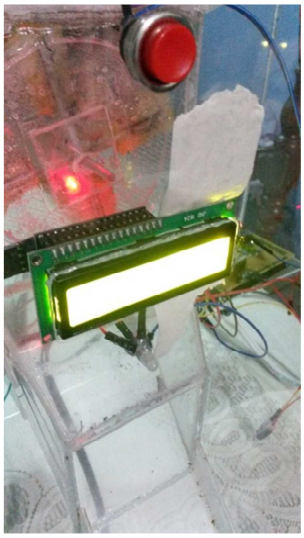


Figura 168. Pruebas de Funcionamiento de los sistemas del Dispensador (16). Fuente: El Autor.

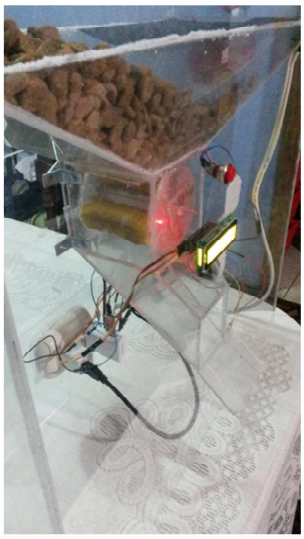


Figura 169. Pruebas de Funcionamiento de los sistemas del Dispensador (17). Fuente: El Autor.

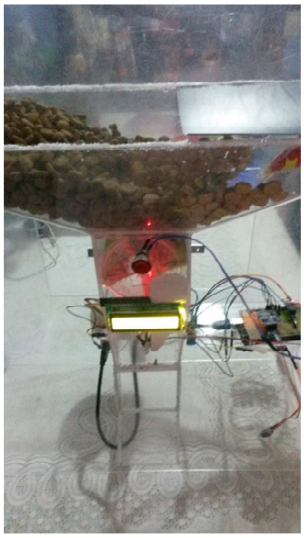


Figura 170. Pruebas de Funcionamiento de los sistemas del Dispensador (18). Fuente: El Autor.

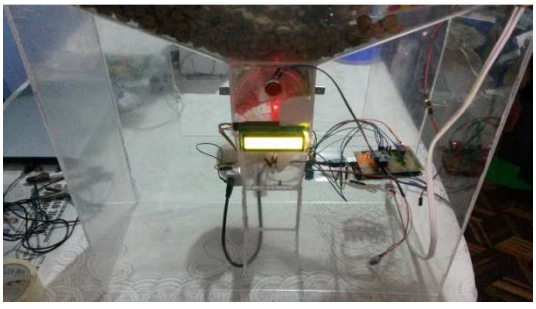


Figura 171. Pruebas de Funcionamiento de los sistemas del Dispensador (19). Fuente: El Autor.

Figura 172. Pruebas de Funcionamiento de los sistemas del Dispensador (20). Fuente: El Autor.

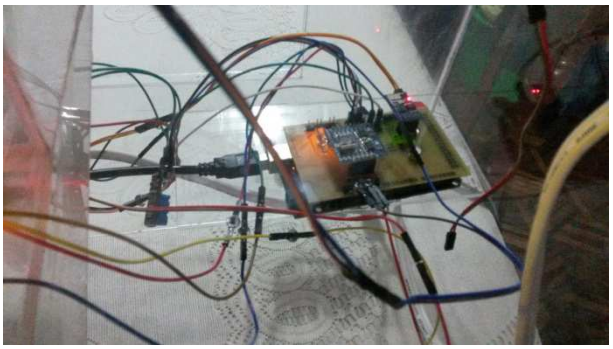


Figura 173. Pruebas de Funcionamiento de los sistemas del Dispensador (21). Fuente: El Autor.

Figura 174. Pruebas de Funcionamiento de los sistemas del Dispensador (22). Fuente: El Autor.

Figura 175. Pruebas de Funcionamiento de los sistemas del Dispensador (23). Fuente: El Autor.

Figura 176. Pruebas de Funcionamiento de los sistemas del Dispensador (24). Fuente: El Autor.

Figura 177. Pruebas de Funcionamiento de los sistemas del Dispensador (26). Fuente: El Autor.

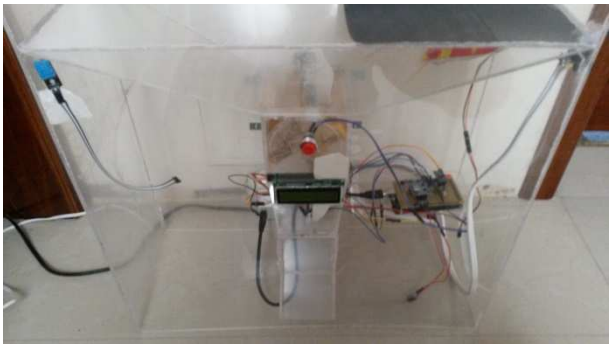


Figura 178. Pruebas de Funcionamiento de los sistemas del Dispensador (27). Fuente: El Autor.

Figura 179. Pruebas de Funcionamiento de los sistemas del Dispensador (28). Fuente: El Autor.

Figura 180. Pruebas de Funcionamiento de los sistemas del Dispensador (29). Fuente: El Autor.

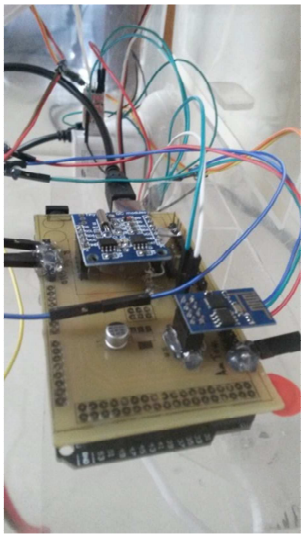


Figura 181. Pruebas de Funcionamiento de los sistemas del Dispensador (30). Fuente: El Autor.

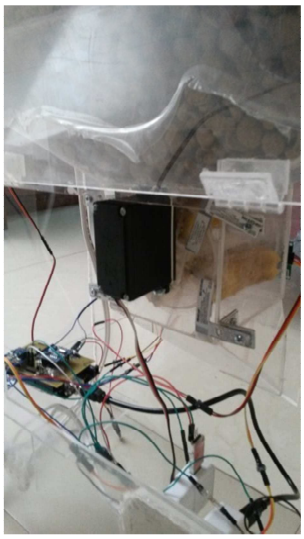


Figura 182. Pruebas de Funcionamiento de los sistemas del Dispensador (31). Fuente: El Autor.

Figura 183. Pruebas de Funcionamiento de los sistemas del Dispensador (32). Fuente: El Autor.

Figura 184. Pruebas de Funcionamiento de los sistemas del Dispensador (33). Fuente: El Autor.