UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES

ARAGÓN

Sistema de riego automático para huertos urbanos con interfaz Android

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

P R E S E N T A:

LUIS MIGUEL MEDINA LAGUNES

DIRECTOR DE TESIS:

Dr. ISMAEL DÍAZ RANGEL

CIUDAD NEZAHUALCÓYOTL, ESTADO DE MÉXICO 2018

Agradecimientos

Dedico este trabajo a mis padres: Bertha Lagunes Cordova y Miguel Medina Estrada, los cuales me han brindado su apoyo incondicional a lo largo de mi vida, todo esto acompañado de sus consejos y educación por lo cual les estaré agradecido toda la vida.

Les agradezco a mis hermanas: Diana Medina Lagunes y Juana María Medina Lagunes por brindarme su apoyo y sus consejos a lo largo de mi vida, permitiéndome ser la persona que soy hoy en día.

Le doy las gracias a mi asesor Ismael Díaz Rangel por brindarme su tiempo, conocimiento y sobre todo su apoyo en la realización de este proyecto de tesis.

También a mis compañeros, con quienes compartí gratas experiencias y trabajé lado a lado, remarcando que nunca hay que rendirse y que siempre hay que mantener una actitud positiva.

Finalmente me gustaría dar gracias la Universidad Nacional Autónoma de México por brindarme las herramientas y conocimientos necesarios para formar en mí un universitario y un profesionista.

Contenido

Índice de ilustraciones

Capítulo 1 Introducción

En la actualidad el uso de los huertos urbanos ha ido en incremento a nivel mundial, dando paso a la agricultura urbana y peri-urbana, la cual permite aprovechar al máximo el espacio en las ciudades, disminuir la huella ecológica, contribuir en la economía generando recursos y empleos. Un ejemplo de ello es la ciudad de Boston que en año de 2008 comenzó con 100 huertos urbanos y en un par de años lograron duplicar su número, al mismo tiempo que reducían los costos de abastecimientos con verduras frescas provenientes de sus propios jardines y patios (LIM, 2018).

En México encontramos la segunda ciudad con más huertos urbanos de Latinoamérica, siendo esta la ciudad de México, contando con 244 azoteas verdes, mencionando como los más emblemáticos los huertos de árbol chiquito, centro verde Azcapotzalco, chula verdura, al natural, siembra merced, cultivo de autor y huerto romita (MXCITY, 2017).

La ciudad de México cuenta con el apoyo de la SEDEMA (Secretaria del Medio Ambiente) para la implementación de huertos urbanos brindando talleres introductorios una vez al mes en el periodo de febrero a noviembre en tres diferentes zonas (jardín botánico del bosque de Chapultepec, bosque de san juan de Aragón y el bosque de Tlalpan).

Dadas las ventajas mencionadas de disponer en el hogar de un huerto, en este trabajo se propone implementar un sistema que permita de manera automática mantener la humedad óptima para el crecimiento de las plantas, el cual es configurable desde una interfaz programada en Android, la que también nos mostrará la humedad actual de las plantas, y alertará al usuario si el nivel en el depósito de agua está por debajo del 20%.

1.1 Objetivo general

Diseñar e implementar un huerto urbano con un sistema automático de regulación de humedad de la tierra, para el óptimo crecimiento de plantas aromáticas, incorporando una aplicación Android para la configuración de parámetros.

1.2 Objetivos específicos

- Investigar las características de los huertos urbanos.
- Distinguir y documentar tipos y características de plantas aromáticas.
- Diseño de un sistema para distribución del riego.
- Proponer un modelo de huerto urbano con materiales de bajo costo.
- Identificar tipos y funcionamiento de sensores de humedad.
- Definir e implementar un sistema de control.
- Diseñar e implementar una etapa de potencia.
- Desarrollar una interfaz para el sistema operativo Android.

1.3 Justificación

Debido al estilo de vida acelerado y exigente es muy común escuchar que no se tiene el tiempo para ciertas actividades, una de ellas, el cuidar una planta por no poder o no recordar regarla al menos una vez al día, por lo que se decidió crear un huerto que mantuviera saludables las plantas y que pudiera estar sin supervisión constante por parte del usuario.

Los beneficios de este trabajo van desde disminuir la huella ecológica ayudando al medioambiente con el reciclaje de desechos, contribuir económicamente con productos agrícolas de calidad y de bajo costo, y finalmente la reducción de pesticidas en los productos agrícolas.

1.4 Descripción del capitulado

Capítulo 2. Se describe el marco teórico de la propuesta, comenzando con la explicación de los huertos urbanos y sus características, tipos de cultivos, materiales utilizados para su construcción, sistemas de control, plataformas de desarrollo de aplicación en Android e información de dispositivos eléctricos y electrónicos.

Capítulo 3. Se hablará sobre el desarrollo del prototipo, cómo se elaboró, cómo se desarrolló el algoritmo de la propuesta, las dificultades y/o errores se encontraron y cómo se solucionaron.

Capítulo 4. Se describen las pruebas realizadas a los dispositivos y al prototipo, así como los resultados arrojados en las pruebas.

Por último, se darán las conclusiones y se hablará sobre el trabajo a futuro para mejorar la propuesta.

Capítulo 2 Marco teórico

2.1 Huertos urbanos

El concepto de un huerto urbano se puede definir como un espacio cerrado o no, destinado al cultivo en escala doméstica (ilustración 2.1), sin menospreciar la calidad de los productos, dichos huertos se pueden ubicar en la periferia de la ciudad o en la propia casa ubicándolo dentro del jardín, terraza, azotea, balcón o el ático (Condelmed, S.L, 2017).

Ilustración 2.1 huerto urbano

2.1.1 Tipos de cultivo en huerto urbano

En los huertos urbanos se pueden tener 4 tipos diferentes de cultivo entre los cuales su uso y cuidados varían según su uso.

Cultivo de flores

Este tipo de cultivo se enfoca en plantas ornamentales las cuales a diferencia de los siguientes cultivos no son para consumo propio sino que su uso es decorativo (Ilustración 2.2), por lo que el uso final de dicho cultivo será un florero, los cuidados que se le dan a este tipo cultivo van desde la temperatura, humedad, luz, abono y trasplante; el mayor ejemplo de estos cultivos son: claveles, tulipán, narcisos, nardos, dalias, jancitos, rosas, etc.

Ilustración 2.2 cultivo de flores

Esta clase de cultivo está formada por verduras y legumbres verdes con la finalidad de ser consumidas crudas o preparadas culinariamente (Ilustración 2.3), las cuales están compuestos por agua, glúcidos, vitaminas, minerales, lípidos, proteínas y fibra; los cuidados que se deben tener con esta clase de cultivo son el espacio, el sustrato donde se va a plantar, control de maleza, plagas y el riego. Los cultivos más comunes de este tipo son de: zanahoria, patatas, cebolla, lechuga, tomate, espinacas, etc.

Ilustración 2.3 cultivo de hortalizas

Cultivo de frutales

Este tipo de cultivo es el más restringido dentro de los huertos urbanos debido a la cantidad de espacio disponible y por el clima, entre las características que debe tener cualquier huerto frutal es que debe ser una variedad enana (Ilustración 2.4), autopolinizantes (que no requieren polinización cruzada), la finalidad de este cultivo es el consumo de los frutos, algunos ejemplos comunes de este tipo de cultivo son limonero, mandarino, naranjo, aguacate, etc.

Ilustración 2.4 cultivo de árboles frutales

Cultivo de aromáticas

Este tipo de cultivo está destinado a la producción de finas hierbas (Ilustración 2.5) con el fin de ser utilizadas frescas, secas o deshidratadas para sazonar y condimentar platillos, se debe tener en cuenta que este tipo de cultivo es uno de los más sencillos ya que requieren pocos cuidados solo se necesita conservar la humedad de la tierra y un poco de luz solar; los cultivos más comunes de este tipo son de menta, orégano, apio, perejil, etc.

Ilustración 2.5 cultivo de aromáticas

2.1.2 Tipos de huerto urbano

Existen diferentes tipos de huertos urbanos, estos se diferencian por las zonas en las que se localizan y el fin con el que se construyen, comenzado por:

Huertos privados con fines de lucro

Este tipo de huertos se caracteriza por que los propietarios cultivan y venden a sus clientes, los cuales pueden ser particulares, negocios de restauración, grupos de consumo; por lo que esta clase de huerto requiere un terreno privado con parcelas pequeñas y es principalmente impulsado por la venta de productos ecológicos y limpios.

Huertos privados domésticos

Este tipo de huertos son cultivados en el interior de las casas, patio, jardín, terraza o balcones por lo que este tipo de huerto será utilizado en la propuesta, lo que más caracteriza a este tipo de huerto es que su forma depende del espacio y tiempo disponible por lo que se han creado una gran variedad de huertos tales como mesas de cultivo, huertos verticales, jardinera en terrazas, azoteas, etc.

Huertos de ocio municipales

Son parcelas cedidas o alquiladas por el municipio, ciudad u otra entidad que se encarga de la gestión y mantenimiento con los objetivos de recuperación, conservación de espacios urbanos, la práctica y difusión de la agricultura ecológica.

Huertos urbanos comunitarios

Estos huertos se sitúan en terrenos públicos o propiedades vecinales los cuales son gratuitos y de libre acceso donde los vecinos trabajan para sacarlo adelante y con el objetivo de socializar y compartir conocimientos y experiencia.

Huertos didácticos

Este tipo de huerto tiene como objetivo la educación ambiental, formación agraria y el apoyo de educación básica; algunos ejemplos de estos son los huertos escolares, en dichos huertos las personas que participan con el mantenimiento mientras que organizan actividades y cursos gratuitos.

Huertos terapéuticos

Este tipo de huertos están destinados a ser una alternativa de ocio y recuperación para todo tipo de pacientes y colectivos vulnerables, cuyo único objetivo promover el bienestar físico y psicológico, teniendo como principales clientes las residencias de adultos mayores, centros de inserción social, hospitales, escuelas para discapacitados y prisiones.

Huertos estéticos o de atractivo turístico

Estos huertos son espacios verdes y bonitos que cumplen dos objetivos: la producción de alimentos y la mejora estética del entorno, este tipo de huertos suele verse en hoteles, restaurantes, espacios públicos y privados.

2.1.3 Materiales para la construcción de huertos

Macetas

Las macetas o jardineras son algo esencial para la construcción de un huerto urbano, estas pueden ser de diferentes materiales.

- Macetas de barro y terracota: estas clases de masetas son pesadas, frágiles y el mayor problema es que se secan muy rápidamente por sus paredes porosas y las bajas temperaturas las dañan, además de ser muy costosas.
- Macetas de madera: son ligeras y de mejor aspecto, pero requieren un mayor cuidado como un barniz impermeabilizante y un producto conservante para madera ya que se pudren con facilidad.
- Macetas de plástico o resina: esta clase de maceta fue elegida para la propuesta debido a su bajo peso, son económicas, se limpian fácilmente, y son fáciles de manipular.

Estructuras

Para el desarrollo de la propuesta se desarrollaron varios tipos de estructuras, comenzando con una placa de madera como base con dimensiones de 1.5m x 1.5m x 15mm, se decidió trabajar con este material debido a que es resistente y fácil de manipular, ya que el huerto está diseñado para interiores no hay inconvenientes causados por los factores ambientales.

Además de la placa de madera se construyeron varias estructuras metálicas, comenzando por anillos para sostener las macetas, una repisa para el depósito de agua y finalmente una base para soportar la placa de madera con el resto de estructuras montadas.

2.2 Plantas aromáticas

Las plantas aromáticas son un tipo de plantas cuyo principal uso es la utilización de sus hojas para resaltar los sabores de los platillos; lo que destaca de este grupo de plantas es su facilidad de cultivo, ya que no requieren un gran espacio, no hay necesidad de trasplantarlas y solo hay que cuidar el riego de estas. Este tipo de plantas cuenta con tres diferentes ciclos de vida, comenzando con el de menor duración que son las plantas anuales, las cuales solo viven una estación del año, un ejemplo de ellas son la albahaca y el cilantro; las siguientes son las plantas bianuales, las cuales su tiempo de vida es de dos estaciones entre las cuales se encuentran el perejil y por último las perennes, las cuales pueden vivir durante varios años tales como el romero.

2.2.1 Condiciones óptimas para crecimiento

Una de las razones por las que se decidió por este tipo de plantas para la propuesta es que requieren pocos cuidados, solo se deben cumplir algunos requerimientos básicos los cuales son (infojardin, 2017):

- Luz solar: esta clase de hierbas necesitan entre 4 y 6 horas de luz solar por lo que perfectamente pueden ser cultivadas en interiores, ya que la sobre exposición de luz solar daña las hojas de algunas especies
- Suelo: esta clase de plantas no requieren un suelo con grandes cantidades de nutrientes y solo es necesario abonarlo una vez al año
- Riego: esta clase de plantas requieren tener un suelo húmedo para su buen crecimiento (tabla 2.1), pero esto varía según la especie que se plante, también se debe cuidar de no exceder en agua, ya que eso puede ocasionar que se pudran las raíces y generar hongos, para evitar lo anterior se recomienda que tenga un buen drenado el recipiente.

Tabla 2.1 Porcentaje de humedad

PLANTA	PORCENTAJE DE HUMEDAD
Perejil	**55%**
Apio	**55%**
Epazote o paico	**72%**
Ajo	**70-80%**
Tomillo	**75%**
Romero	**50-60%**
Salvia	**70-80%**
Orégano	**70-80%**
Hierbabuena	**70%**
Menta	**85%**
Estragón	**60-70%**

Cebollino	**80%**
Cilantro	**75%**
Melisa o toronjil	**72%**
Albahaca	**80%**

2.2.2 Familias botánicas

Las plantas aromáticas constan de 3 subfamilias botánicas

- Las aliáceas: estas se distribuyen en regiones templadas, cálidas y subtropicales, su mayor característica es su singular olor a ajo o cebolla, se caracterizan por tener hojas suaves y carnosas y flores sobre los tallos o ramas, algunos ejemplos de ellos son el ajo y la cebolla.
- Las apiáceas: esta subfamilia está compuesta por hierbas y algunos arbustos tradicionalmente llamadas "umbelíferas" por su forma similar a una sombrilla, Presentan un tallo a menudo estirado; con la médula blanda o fistulosa. Hojas alternas, casi siempre con una vaina abrazadora grande, enteras, partidas, graminoides o más frecuentemente recortadas, o, hendidas, muy a menudo divididas, algunos ejemplos de ellas son el cilantro, apio y comino. (Desconocido, 2017)
- Las lamiáceas: esta subfamilia está compuesta por plantas herbáceas y arbustos, constan de tallos tetragonales, hojas opuestas, enteras o divididas, flores hermafroditas, algunas plantas dentro de esta familia son el orégano, la menta y la salvia. (Universitat de les Illes Balears, 2017)

2.3 Sistemas de riego

Dentro de los muchos sistemas de riegos diferentes que existen, los huertos urbanos se encuentran limitados a solo algunos por el espacio y la cantidad de dinero que se suele emplear en ellos.

Riego con manguera o regadera

Esta clase de riego consiste en que personalmente se debe poner agua a las plantas por lo que requiere forzosamente supervisión y más aun con esta clase de riego no se logra una buena uniformidad de humedad.

Riego por goteo

Esta clase de riego consiste en aportar agua de manera constante por medio de goteros o emisores (Ilustración 2.6), los cuales pueden estar integrados en la tubería, goteros de botón y goteo por pinchado, el cual es principal en los huertos verticales, ya que este recicla el agua filtrada de plantas superiores.

Las ventajas del riego por goteo son el ahorro de agua, se mantiene una humedad uniforme y no forma encharcamientos.

Ilustración 2.6 huerto con sistema de riego por goteo

Riego subterráneo

Esta clase de riego es la más moderna, ya que consiste en colocar tuberías perforadas enterradas a una determinada profundidad (Ilustración 2.7), este tipo de riego es el más estético y duradero, pero cuenta con varios inconvenientes, como que solo puede ser utilizado en suelos arenosos o arcillosos y que se limita a una sola especia de planta por sistema de riego, por lo que se optó por automatizar otro tipo de riego ya que este sería demasiado costoso por el uso de electroválvulas o múltiples bombas.

Ilustración 2.7 huerto con sistema de riego subterráneo

Riego propuesto

El sistema de riego propuesto utilizado consiste en el uso de un servomotor, el cual posiciona el flujo de agua en una de los cinco posibles canales de riego, permitiendo regar cinco especies diferentes de plantas con un solo sistema, para evitar los encharcamientos provocados por exceso de agua el sistema regula el encendido y apagado de la bomba por pequeños lapsos para permitir una distribución uniforme de la humedad.

2.3.1 Bombas de agua

Dentro de la propuesta realizada se probaron algunos tipos de bombas de agua, para así seleccionar la más óptima para el trabajo, entre las cuales se tenemos las siguientes:

- Bomba de inmersión para fuente con una potencia de 60 W (Ilustración 2.8): el uso de esta bomba fue rechazado en la propuesta tanto por su elevado costo y por tener demasiada potencia para la propuesta y no contar con más especificaciones de fabricante.

Ilustración 2.8 bomba para fuente

- Bomba de agua de 6 V-12 V (Ilustración 2.9), con una elevación entre 70 y 180 cm y un caudal máximo de 200 l/hora, se decidió no usar esta bomba aunque cumplía con los parámetros el uso de esta bomba requería el uso de una fuente de alimentación diferente, mientras que hay otras que cumplen con los parámetros requeridos incluyendo una alimentación de 5 V.

Ilustración 2.9 bomba de 6V-12V

- Bomba de inmersión de 3.5 V-12 V (ilustración 2.10), con una elevación máxima de 2.2 m, flujo máximo de 350 l/hora, el cuerpo de la bomba este hecho de plástico a prueba de altas temperatura, de alta resistencia y eje de acero inoxidable, esta bomba fue la utilizada en la propuesta tanto por su bajo costo y el cumplir con todos los parámetros de potencia y alimentación.

Ilustración 2.10 bomba de 3.5 V -12 V

2.3.2 Mangueras y conexiones

Dentro de la propuesta se necesitaron dos tipos diferentes de mangueras, uno sería utilizado para los canales de riego, en donde se integran conexiones plásticas para su mejor distribución; y el otro tipo de manguera sería utilizado para la bomba de agua y el servomotor.

- Manguera de plástico utilizado en los canales de riego y desagüe, la cual es flexible y de bajo costo, por lo que fue integrada con conexiones de manguera tipo T, tipo L y boquillas al sistema de riego (ilustración 2.11).

Ilustración 2.11 manguera de plástico y conexiones plásticas

- Manguera de plástico para acuario (ilustración 2.12) con un diámetro de 75 mm, ligeramente flexible y de precio económico, fue rechazada su uso en la propuesta, ya

que no contaba con la flexibilidad suficiente, así que se decidió por utilizar un tipo diferente de manguera.

Ilustración 2.12 manguera para acuario

- Manguera de plástico para gasolina (ilustración 2.13) utilizada para la bomba de agua y el servomotor, donde su flexibilidad fue el aspecto con el que se decidió su uso en la propuesta, ya que esta dio una mejor funcionalidad con el servomotor.

Ilustración 2.13 manguera para gasolina

2.3.3 Servomotor

Los servomotores trabajan con corriente continua y tienen un eje de rendimiento controlado por PWM (*Pulse Width Modulation, Modulación* por Ancho de Pulso) con el cual el motor puede ser posicionado en un ángulo entre su rango de operación, donde normalmente este rango es de 0 a 180 grados. Entre la gran cantidad de servomotores en el mercado se probaron los siguientes modelos.

Servomotor MG996R

Servomotor de alto torque con un rango de voltaje de alimentación de 4.8 V a 7.2 V, con un peso de 55 g, con dimensiones de 40.7 mm x 19.7 mm x 42.9 mm y opera con una PWM de 50 Hz, este servomotor (Ilustración 2.14) fue elegido para la propuesta por su alto torque, con ello permite un buen movimiento a la manguera de riego y opera con la misma frecuencia de PWM que la biblioteca servo.h para Arduino.

Ilustración 2.14 servomotor MG996R

Futaba S3003

Servomotor de la marca Futaba (Ilustración 2.15) con un torque de 3.17 kg-cm, con una alimentación de 5 V, con dimensiones de 39.9 mm x 20.1mm x 36.1 mm y trabaja con una PWM de 30 ms (33.33 Hz), se descartó el uso de este servomotor debido a la frecuencia de PWM con la que trabaja, ya que la biblioteca usada para el control del servomotor trabaja con una frecuencia de 50 Hz.

Ilustración 2.15 servomotor futaba

Tower pro SG92R

Servomotor de la marca Tower pro (ilustración 2.16) con un torque de 2.5 kg-cm, con las dimensiones de 23.1 mm x 12.2 mm x 27 mm, una alimentación de 5 V y trabaja con una PWM de 50 HZ, la razón por la que se descartó el uso de dicho servomotor es por el bajo torque, ya que no podía mover con libertad la manguera de riego.

Ilustración 2.16 servomotor SG92R

2.3.4 Electroválvulas

Las electroválvulas son válvulas electromecánicas (ilustración 2.17), diseñadas para abrir o cerrar el paso el paso de un líquido en un circuito, la apertura y cierre se efectúa a través de un campo magnético generado por una bobina en la base fija que atrae el embolo (Altec Alta Tecnología de Vanguardia, SA de CV. Monterrey, N.L. Mexico, 2017).

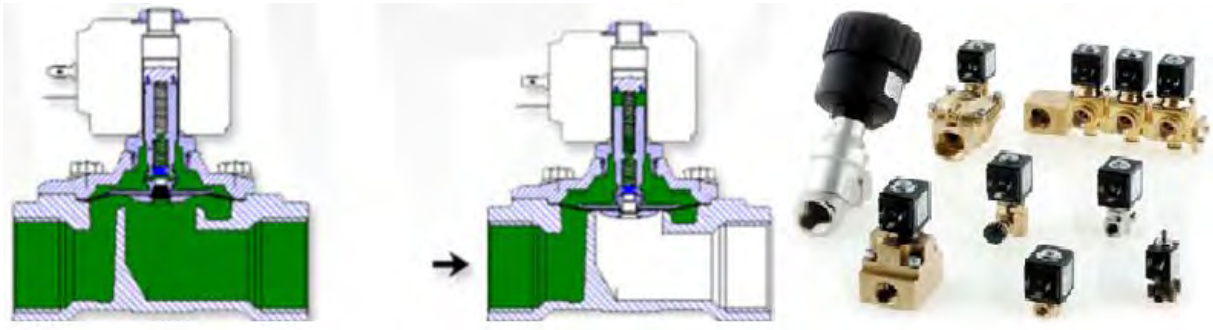

Ilustración 2.17 electroválvulas y su funcionamiento

2.4 Sensores

Dentro de las necesidades del ser humano una de las más importantes ha sido el cuantificar y controlar su entorno, con ello el desarrollo de la electrónica ha ayudado dicho propósito, ya que con el paso de los años se han creado sensores, los cuales son capases de detectar magnitudes físicas o químicas y transformarlas en variables eléctricas las cuales pueden ser interpretadas por otros dispositivos, los cuales pueden tomar decisiones en base a las variables eléctricas que recibe, ha dichos dispositivos se les conoce como dispositivos de control, los cuales permiten la automatización de sistemas.

Entre los sensores utilizados todos ellos comparten características como que son de origen chino, ya que es el principal productor tecnológico gracias al bajo costo de sus dispositivos debido a la mano de obra barata y producción en masa. Otra característica importante es que su alimentación eléctrica está en el rango de 3.3 V- 5 V y lo más importante, es que necesitan un sistema de control para para poder interpretar la información que proporciona el dispositivo.

2.4.1 Sensores de humedad para el suelo

Dentro de la variedad de sensores de humedad en el mercado en la propuesta se usara de un tipo específico, el sensor altamente sensible a la humedad, el cual sirve para medir la humedad en la tierra (ilustración 2.18), consta de dos sondas por las cuales hace pasar un corriente a través del sólido y después deduce la resistencia la resistencia entre las sondas para obtener el nivel de humedad, su funcionamiento es sencillo mientras mayor cantidad de agua exista la resistencia entre las sondas será menor.

Dentro de este tipo de sensores existen ligeras variaciones como el tamaño o cuántas salidas pose el sensor, pero todos constan de dos sondas con recubrimiento para proteger el nikel de la oxidación, lo cual lo hace ideal para monitorear un jardín o un huerto.

Ilustración 2.18 sensor yl-69

Características del sensor utilizado

- Voltaje de entrada: 3.3 – 5 VCD
- Voltaje de salida: 0 ~ 4.2 V
- Corriente: 35 mA
- VCC: Tensión de alimentación
- GND: Tierra
- A0: Salida analógica que entrega una tensión proporcional a la humedad.
- D0: Salida digital; este módulo permite ajustar cuándo el nivel lógico en esta salida pasa de bajo a alto mediante el potenciómetro

2.4.2 Sensores de nivel de agua

En la realización de esta propuesta era necesario mantener un monitoreo constante y preciso de la cantidad de agua disponible en el depósito por lo que se requirió el uso de un sensor en el depósito de agua, para ello se probaron algunos de ellos y finalmente se seleccionó el más óptimo para la propuesta, a continuación, se describe brevemente los sensores utilizados para medir el nivel de agua.

Sensor interruptor de nivel de líquido

El funcionamiento de este sensor (ilustración 2.19) es a través de un interruptor magnético y un imán en forma de anillo dentro de un material menos denso que el agua, el cual se eleva

cuando es sumergido en agua, de esta manera se cierra el circuito y permite detectar la presencia -o ausencia- de agua.

Ilustración 2.19 sensor interruptor de nivel liquido

Características del sensor:

- Potencia máxima: 10 W
- Voltaje máximo: 220 V
- Corriente máxima: 0.5 A
- Resistencia de los contactos: 100 MOhm
- Temperatura de operación: -10~60

La desventaja del sensor es que solo nos regresa como resultado dos posibles respuestas, ausencia o presencia de agua, por lo que se decidió probar con otro sensor que entregará una medición más precisa de la cantidad de agua en el depósito.

Sensor ultrasónico HC-SR04

El funcionamiento de este sensor (ilustración 2.20) es muy sencillo ya que consta de enviar un pulso de disparo ultrasónico y al recibir el eco de dicho disparo se mide el tiempo diferido del pulso de respuesta, el cual es proporcional a la distancia del objeto, un aspecto a tomar en cuenta es la constante de la velocidad del sonido en el aire, ya que esta será la que nos permitirá obtener la distancia del objeto.

Ilustración 2.20 sensor ultrasónico

Características del sensor ultrasónico

- Alimentación (VCC) de 5 V
- *Trigger*: señal de disparo del ultrasónico
- Echo: recepción de la señal ultrasónica
- GND
- Rango de medición: 2 cm a 400 cm
- Corriente de alimentación: 15 mA
- Frecuencia del pulso: 40 KHz
- Apertura del pulso ultrasónico: 15º
- Señal de disparo: 10 µs
- Dimensiones del módulo: 45 mm x 20 mm x15 mm

Se decidió utilizar este sensor en la propuesta, ya que con la medición de distancia entre el nivel actual del depósito y la distancia a máxima capacidad permite obtener una medición precisa de la cantidad de agua, el único aspecto negativo encontrado fue su corta vida útil debido a la humedad del depósito, aun que gracias a su bajo costo es muy fácil remplazarlo.

2.4.3 Módulo *bluetooth*

Para la propuesta se requería la transmisión de datos de forma inalámbrica a un teléfono inteligente (Smartphone) por lo que se decidió utilizar la comunicación por *bluetooth*, la cual se define como un protocolo de comunicaciones de especificaciones industriales para redes inalámbricas de área personal (WPAN) que posibilita la transmisión de datos entre diferentes dispositivos de bajo consumo mediante un enlace de radiofrecuencia de 2.4 GHz (desconocido, 2017).

Dado estos requerimientos se decidió utilizar el dispositivo HC-05 (ilustración 2.21) un módulo *bluetooth* configurado como *slave* (esclavo), lo cual facilita la comunicación con el microcontrolador.

Ilustración 2.21 módulo bluetooth HC-05

Características del módulo HC-05

- Especificación Bluetooth v2.0 + EDR (*Enhanced Data Rate*)
- Modo configurable (*master o slave*)
- Chip de radio: CSR BC417143
- Frecuencia: 2.4 GHz, banda ISM
- Modulación: GFSK (*Gaussian Frequency Shift Keying*)
  Antena de PCB incorporada

- Potencia de emisión: ≤ 6 dBm, Clase 2
- Alcance 5 m a 10 m
- Sensibilidad: ≤ -80 dBm a 0.1% BER
- Velocidad: Asincrónica: 2 Mbps (max.)/160 kbps, sincrónica: 1 Mbps/1 Mbps
- Seguridad: Autenticación y encriptación (contraseña por defecto: 1234)
- Perfiles: Puerto serial Bluetooth
- Módulo montado en tarjeta con regulador de voltaje y 4 pines suministrando acceso a VCC, GND, TXD, y RXD
- Consumo de corriente: 30 mA a 40 mA
- Voltaje de operación: 3.6 V a 6 V
- Dimensiones totales: 1.7 cm x 4 cm aprox.
- Temperatura de operación: -25 ºC a +75 ºC

2.5 Fuentes de alimentación

Las fuentes de alimentación son dispositivos los cuales se dedican a proporcionar el voltaje de alimentación adecuado a los circuitos convirtiendo de corriente alterna (CA) a corriente continua (CC), la fuentes de alimentación de clasifican en dos tipos, fuentes lineales que cuenta con un diseño simple pero poco eficiente en su regulación de voltaje, mientras que las fuentes conmutadas son más pequeñas que las lineales más complejas son mayormente susceptibles a averías (vilardell, 2017).

2.5.1 Fuente de alimentación lineal

Las fuentes lineales son sencillas y todas ellas se elaboran siguiendo las 4 etapas del siguiente esquema. (Electrocomponentes SA, 2017)

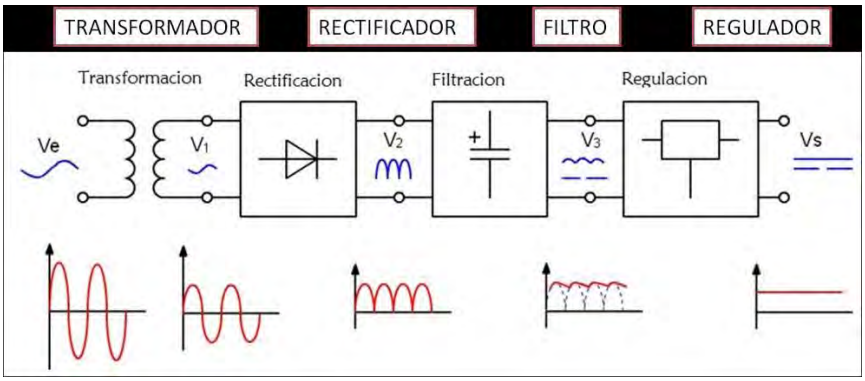

Ilustración 2.22 esquema de fuentes lineales

- **Transformación:** esta etapa consta sencillamente de un transformador el cual convierte la energía eléctrica alterna de la red en alterna de otro nivel de voltaje a través de la acción de un campo magnético.
- **Rectificación:** esta etapa está constituida por diodos rectificadores cuya función es rectificar la señal proveniente de la etapa anterior de transformación
- **Filtrado:** la etapa está constituida por una o varios capacitores que se encargan de eliminar la componente de tensión alterna para entregar un nivel de tensión lo más continuo posible.
- **Regulación:** esta última etapa consiste en el uso de circuitos integrados para disminuir el rizado y proporcionar una tensión de salida ideal.

2.5.2 Fuente de alimentación conmutada

Este tipo de fuentes se caracterizan por el uso de un transistor en su zona de corte y saturación para lograr periodos de tiempo en el que la corriente y la tensión son nulos para convertir una señal de entrada alterna en una cuadrada.

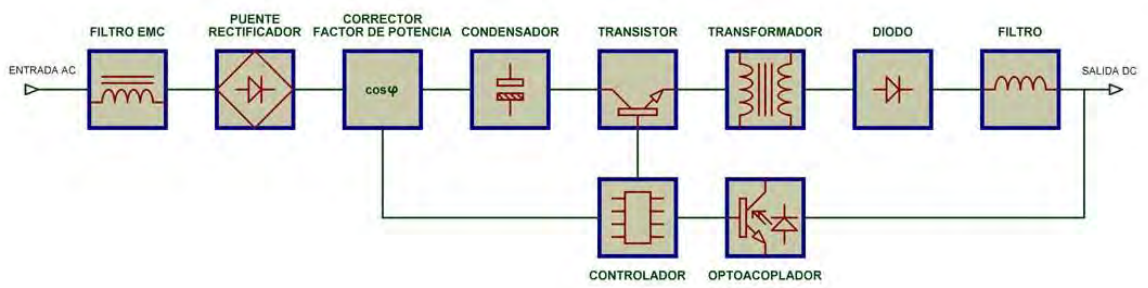

Ilustración 2.23 esquema de fuentes conmutadas

Al igual que las fuentes lineales las fuentes conmutadas pasan por varias etapas, las cuales se explicarán su funcionamiento (vilardell, 2017).

- **Filtro EMC:** Su función es absorber los problemas eléctricos de la red, como ruidos, armónicos, transitorios, etc.
- **Puente rectificador:** Solo deja pasar la corriente en un sentido, de modo que convierte la corriente alterna en corriente continua.
- **Corrector del factor de potencia:** esta etapa se dedica a corregir el factor de potencia cuando este se desfasa y así aprovechar toda la potencia de la red. **Condensador:** Se encarga de eliminar la componente de alterna y dejar solo corriente continua con un valor estable.
- **Transistor:** Se encarga de cortar y permitir el paso de la corriente. De este modo se convierte a la corriente continua en corriente pulsante.
- **Controlador:** se encarga tanto de activar y desactivar el transistor, proporcionar protección contra corto circuitos, sobrecargas, además de controlar el circuito de corrección de factor de potencia.
- **Transformador:** Reduce la tensión, y además aísla físicamente la entrada de la salida.
- **Diodo:** permite el paso de la corriente en una sola dirección rectificando la señal de onda.
- **Filtro:** elimina la componente de alterna de la señal obteniendo una señal de corriente continua.

- **Optoacoplador:** Enlaza la salida de la fuente con el circuito de control, pero manteniéndolos físicamente separados.

2.6 Entorno de programación Arduino

Para la realización de la propuesta se optó por el uso del entorno de programación Arduino, la cual es de código abierto y está basado en *Processing*, en *Wiring* y "C", por lo que es un entorno muy amigable y sencillo de utilizar, que no requiere un gran conocimiento en programación para su uso (ARDUINO, 2017).

El entorno de desarrollo de Arduino (ilustración 2.24) tiene en su barra de menú cinco opciones, las cuales serán explicadas más adelante.

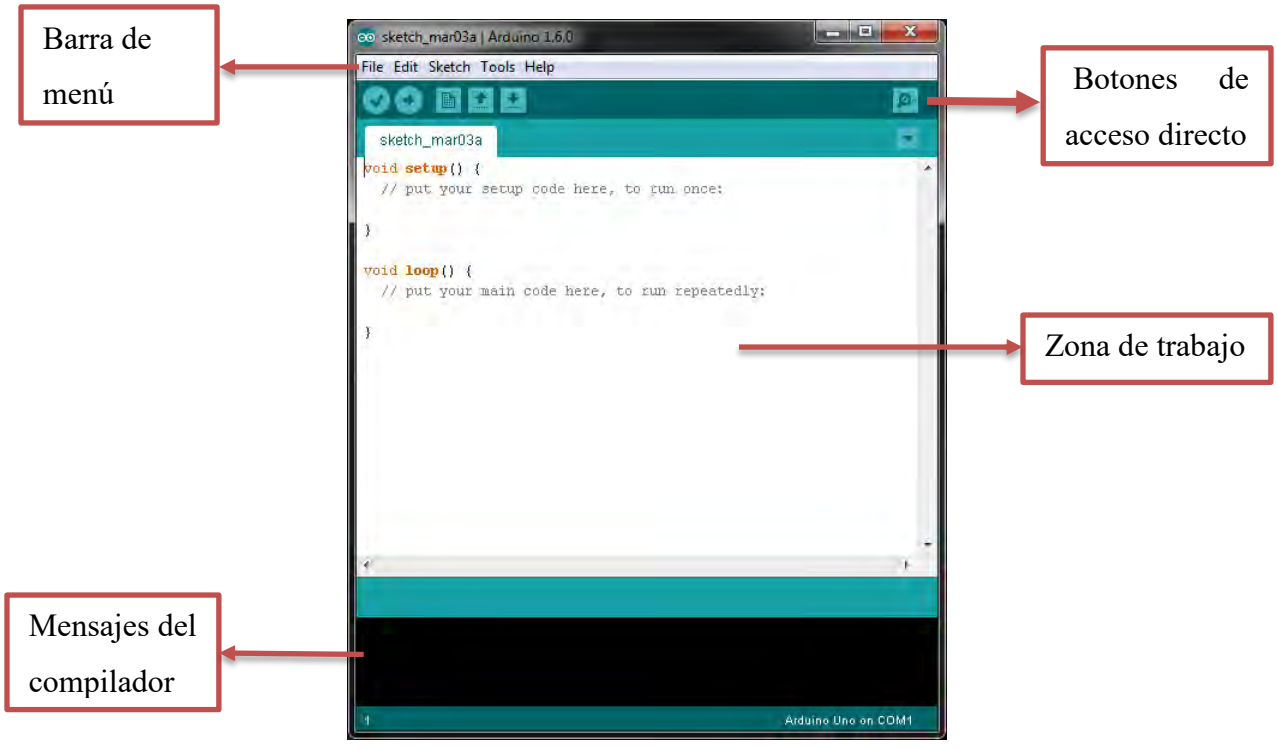

Ilustración 2.24 entorno de programación Arduino

Barra de menú

En esta barra se tienen cinco opciones:

- Archivo: en esta opción encontramos opciones como abrir un nuevo archivo, usar un archivo guardado anteriormente, guardar el programa y cargar ejemplos dentro del entorno.
- Editar: el siguiente apartado cuenta con herramientas para la edición de texto como la sangría, selección de texto y la búsqueda de palabras.
- Programa: este apartado nos da las opciones de compilar el programa, subir el programa a la placa, incluir bibliotecas, añadir ficheros y mostrarnos la ubicación de guardado del programa.
- Herramientas: dentro de este apartado podemos encontrar las opciones de seleccionar el modelo de placa que se está utilizando, el puerto de lectura de datos y el monitor serial.
- Ayuda: en este último apartado encontramos opciones para aclarar dudas respecto al funcionamiento del entorno y sus funciones, así como redirigirnos a foros oficiales de dudas y problemáticas para ser tratados por la comunidad de Arduino.

Botones de accesos directos

Esta sección del entorno cuenta con cinco botones de acciones específicas, las cuales se encuentran en la barra de herramientas, comenzado de izquierda a derecha con el icono de paloma (acierto), nos permite compilar el programa, el icono de flecha apuntando a la derecha nos permite subir el programa a una tarjeta de desarrollo, el icono de un hoja nos permite abrir un nuevo espacio de trabajo en blanco, el icono de flecha apuntando hacia arriba nos permite abrir un archivo previamente guardado en el equipo (.ino), el icono de flecha hacia abajo nos permite guardar nuestro trabajo en el equipo y finalmente el icono de lupa nos permite abrir el monitor serial.

Zona de trabajo

Esta se encuentra en la mitad de la ventana del programa y en esta es donde el usuario escribirá el código del programa, donde se debe respetar las dos funciones incluidas por defecto, comenzando por SETUP, en la cual se declaran las variables principales, bibliotecas y configuraciones que solo se realizaran una vez, la segunda función de nombre LOOP, es donde se escriben las instrucciones del programa principal el cual será ejecutado

repetidamente. Aun si no se utiliza alguno de las dos estructuras mencionadas anteriormente estas deben ser incluidas, de lo contrario el programa no compilará y no podrá ser subido a la placa.

Mensajes de compilador

Esta última sección de la ventana localizada en la parte inferior se dedica a mostrarnos mensajes provenientes del entorno, los cuales pueden ser errores en el código, información de la memoria utilizada del microprocesador, así como hacernos saber que el código se ha compilado y se ha cargado con éxito en la tarjeta.

2.6.1 Estructuras de control

El entorno de desarrollo Arduino soporta varias estructuras de control, cada una para acciones específicas y con su propia sintaxis, estas se dividen en tres tipos diferentes condicionales, bucles, y bifurcaciones o saltos. (ARDUINO, 2017)

Condicionales

Este tipo de estructuras trabajan junto a operadores de comparación (mayor que, menor que, diferente de, etc.), para comprobar si se ha alcanzado cierta condición.

- *If*: es una estructura de selección simple, cuya función es realizar una acción o sentencia de acuerdo a la evaluación de una expresión, la cual puede ser verdadera o falsa, de acuerdo a ello el flujo del programa puede tomar dos direcciones. Su sintaxis es:

  if(condición){
  Acción
  }
  Else{
  Acción 2
  }

- *Switch case*: esta estructura de control es usada para la toma de decisiones múltiples, evaluando el valor de una condición y si esta se encuentra dentro los redactados en su sentencia se ejecuta el código en esa instrucción del case. Su sintaxis es:

switch (variable) {

case a: acción

break;

}

Bucles

Son aquellas estructuras que realiza repetidas veces un fragmento aislado de código, hasta que la condición asignada de dicho bucle deje de cumplirse.

- *For*: es un ciclo utilizado cuando se requiere realizar procesos que requieran una determinada cantidad de repeticiones, su sintaxis es: ***for(inicio, condición, incremento)***
- *while*: este es un ciclo que se repetirá indefinidamente hasta que la condición asignada deje de cumplirse, su sintaxis es: ***while(condición) {código}***

Bifurcaciones o saltos

Este tipo de estructuras son aquellas que interrumpen el flujo del programa y lo redirigen a una dirección especifica.

- *goto*: esta instrucción redirige el flujo del programa a un punto diferente marcado por una etiqueta, su sintaxis es:
  etiqueta:
  goto etiqueta;
- break: esta función es un paro del programa, el flujo del programa es detenido hasta donde se encuentra la función break
- *continue*: la instrucción *continue* trabaja con los ciclos do, *while* y *for* omitiendo el resto de la iteración actual, pero continúa verificando la expresión condicional del bucle.
- *return*: esta instrucción termina una función y regresa un valor de una función a la función llamada.

2.6.2 Variables

El entorno de programación Arduino cuenta con diferentes tipos de variables, cada una con características propias y usos específicos, contando con dos tipos específicos de variables, estas normalmente son declaradas al principio del programa

Variable tipo constante

Este tipo de variables solo puede tener un valor y no se pueden modificar, algunas de estas son:

- HIGH/True: es un estado lógico alto, un 1 lógico o un voltaje mayor a 3V
- LOW/False: es un estado lógico bajo, también se puede interpretar como un 0 lógico o un voltaje menor a 1.5V.
- INPUT/OUTOUT: estas constantes trabajan únicamente con los pines digitales y sirve para definir si estos pines serán entradas o salida de datos.

Variables de tipo dato

Este tipo de variables pueden tomar un amplio rango de valores, algunas de ellas son:

- *Void*: Esta es utilizada para la declaración de funciones e indica que no devuelve ninguna información a la función llamada.
- *Boolean*: es una variable que almacena solo puede almacenar valores de falso o verdadero ocupando 8 bits
- *Char*: es una variable que almacena caracteres que componen el código ASCII
- Byte: es una variable que limita su almacenamiento a 8 bits.
- *Int*: esta variable se utiliza para el almacenamiento de números enteros negativos o positivos, con una capacidad máxima de 16 bits
- *Float*: este tipo de variable es utilizada para valores con punto decimal, otorgando mayor precisión con un máximo de 7 dígitos y un tamaño de 4 bytes o 32 bits.
- *Long*: este tipo de variables es utilizada para el almacenamiento de números de gran tamaño, tomando valores entre -2147483648 y 2147483647 con un tamaño de 4 bytes o 32 bits.

- *Short*: es una variable para almacenar números con un rango de valores de -32.768 a 32.767 utilizando 2 bytes de memoria.
- *String*: este tipo de variables se utiliza para almacenar cadenas de caracteres.

2.6.3 Funciones del entorno

Dentro del entorno de programación Arduino contamos con un amplio catálogo de funciones el cual puede ser aún mayor al incluir bibliotecas de terceros, las cuales se tratarán más adelante; dentro de las funciones incluidas se pueden citar las siguientes:

- pinMode(): esta función es utilizada para la configuración de entrada o salida de datos en los pines digitales de la tarjeta, su sintaxis es la siguiente: ***pinMode(número del pin, configuración);***
- digitalWrite():esta función solo puede ser utilizada en pines configurados previamente como salida y sirve para escribir un valor alto o bajo en un pin, si sintaxis en las siguiente: ***digitalWrite(número del pin, valor alto o bajo);***
- digitalRead():esta función solo es utilizada en pines declarados anteriormente como entrada y su función es leer el nivel del voltaje en el pin y en base a ello entregar un valor alto(voltaje mayor a 3.3V) o un valor bajo(voltaje menor a 1.5V), su sintaxis es la siguiente: ***digitalRead(número del pin);***
- analogWrite(): la función ya mencionada es utilizada para escribir una señal PWM en un pin, el cual puede ir de un rango de 0 a 255 con una frecuencia de 490Hz a excepción de los pines 5 y 6 que trabajan con una frecuencia de 980Hz, su sintaxis es la siguiente: ***analogWrite(número del pin, valor de señal pwm);***
- analogRead(): esta función se utiliza para leer el valor de voltaje de un pin en específico(pines analógicos), dicho valor de voltaje puede ir desde los 5V hasta 0V y genera valores que van entre los 1023 y 0 respectivamente, su sintaxis es la siguiente: ***analogRead(número del pin analógico);***
- delay(): esta función se utiliza para pausar el programa un determinado tiempo, el cual es especificado en milisegundos, si sintaxis es la siguiente: ***delay(cantidad de tiempo en milisegundos);***

- map():esta función matemática se utiliza para reasignar un valor nuevo a un número dentro de un rango de valores a otro rango de valores diferentes, su sintaxis es: ***map(valor, rango mínimo, rango máximo, nuevo valor mínimo, nuevo valor máximo);***
- Serial(): esta función se utiliza para la comunicación serial atreves de los pines TX/RX, esta cuenta con diferentes instrucciones según su sintaxis, las más utilizadas son: ***Serial.begin(número de baudios a los que se transmite);*** esta configura la velocidad de transmisión en baudios, Serial.read(); esta permite leer la información que se recibe de manera serial, ***Serial.print(valores y caracteres que se desean transmitir)*** o ***Serial.println(valores y caracteres que se desean transmitir)*** son instrucciones para transmitir de manera serial información, con lo diferencia de que la segunda función transmite la información y al finalizar manda un salto de línea que puede facilitar la interpretación de dicha información.

2.6.4 Bibliotecas

Una de las facilidades del entorno de programación Arduino son las bibliotecas, las cuales son agrupación de funciones para llevar a cabo un proceso, generalmente son utilizadas para facilitar el uso de dispositivos que requieran una serie de procesos específicos para poder funcionar y también para procesar los datos que estos arrojan, el entorno de programación de manera predeterminada cuenta con algunas bibliotecas instaladas, en caso contrario se puede instalar siguiendo los siguientes pasos:

1. Se busca la biblioteca deseada en internet cuidando que el archivo comprimido tenga la extensión .zip, esto para facilitar su instalación al entorno de programación Arduino.
2. Una vez encontrada la biblioteca se descarga en el ordenador y se guarda en una locación que sea de fácil acceso para el usuario
3. Posteriormente para instalar la biblioteca, desde el entorno de programación Arduino, nos dirigimos a la barra de menú seleccionamos el apartado de programa el cual desplegará una lista, entre la cual existe la opción de incluir una biblioteca (ilustración

2.25), en dicha opción se desplegará una nueva lista con las bibliotecas instaladas y la opción de agregar una nueva.

4. A continuación, nos abrirá un gestor de archivos, en el cual deberemos dirigirnos a la locación donde se guardó la biblioteca y seleccionarla.
5. Ahora volvemos a la barra de herramientas en la sección de programa y la opción de incluir bibliotecas, podemos observar que en la parte inferior se encuentra la biblioteca instalada y lista para usarse.

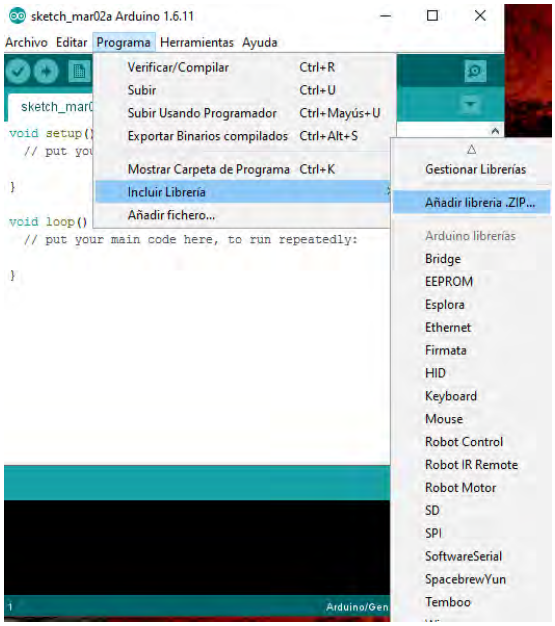

Ilustración 2.25 instalación de nueva biblioteca

2.6.5 Bibliotecas utilizadas

Dentro de la propuesta presentada en este trabajo se necesitó utilizar algunas bibliotecas, las cuales serán explicadas y descritas con sus instrucciones y dispositivo.

EEPROM.h

Esta biblioteca nos permite usar la memoria EEPROM del microcontrolador, la cual podemos utilizar para almacenar valores que sean importantes y no se pierdan, aunque se quite la alimentación del mismo.

La memoria EEPROM es un dispositivo electrónico de alta escala integrativa que sirve para guardar información en forma de unos y ceros en sus celdas, la placa Arduino utiliza el

microntrolador ATmega328P el cual cuenta con una memoria EEPROM de 1Kbyte y una vida útil de aproximadamente 100000 ciclos de lectura o escritura.

Esta biblioteca se utilizó para guardar los valores de configuración de humedad, sus instrucciones son las siguientes:

- ***EEPROM.write(dirección de memoria, variable o valor a almacenar);*** esta función permite escribir información en los bits de una localidad de memoria.
- ***EEPROM.read(dirección de la memoria)=variable donde se desea almacenar la información;*** esta función se utiliza para leer información en una localidad de memoria y asignarla a una variable.
- ***EEPROM.update(dirección, valor o variable);*** esta función actualiza el valor de una variable cuando esta cambia y la almacena en su misma localidad de memoria.
- ***EEPROM.get(dirección, valor o variable);*** esta función obtiene un valor o variable de la memoria EEPROM.
- ***EEPROM.put(dirección inicial, variable o valor);*** esta función guarda de manera automática cuando se proporciona la dirección de memoria, dicha función utiliza la función EEPROM.update, por lo que se recomienda usarla racionalmente, debido a la vida útil de la memoria.

Servo.h

Esta biblioteca se encuentra incluida dentro del entorno Arduino y se utiliza para facilitar el control del dispositivo servomotor, el cual trabaja con una señal PWM, esta biblioteca trabaja con una frecuencia de 50 Hz, por lo que si el servomotor a utilizar no trabaja a dicha frecuencia funcionará inadecuadamente.

Algunas de sus instrucciones son:

- ***Servo "nombre_del_servomotor";*** Esta función se utiliza para crear un objeto de tipo servo con el nombre asignado.
- ***"Nombre_del_servomotor".attach(numero de un pin);*** Esta función se utiliza para asignar un pin al servomotor, por el cual saldrá la señal PWM de control.

- ***"Nombre_del_servo".write(ángulo que se desea posicionar);*** Esta función se utiliza para posicionar el servomotor en un ángulo.

NewPing.h

Esta biblioteca es utilizada como soporte para el ultrasónico, facilitando su configuración y el cálculo de la distancia que este proporciona, sus comandos son los siguientes:

- ***NewPing sonar (TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);*** Esta función se utiliza para configurar los pines y la distancia máxima del sensor.
- ***sonar.ping_median()***; Esta función nos permite obtener la medición del tiempo de viaje del sonido, se recomienda guardar dicho valor en una variable.
- ***US_ROUNDTRIP_CM;*** Esta es una constante la cual dividirá el valor obtenido de la instrucción anterior para obtener la medición en cm de la distancia.

2.7 Diseño y programación de interfaz con Android.

Para el desarrollo de esta propuesta se necesitó el uso de una interfaz gráfica, la cual se decido realizar en el sistema operativo, debido a la gran cantidad de usuarios del sistema operativo y también las libertades con las que cuenta a diferencia de otros sistemas operativos, que requieren licencias u otros permisos.

Una de facilidades del sistema operativo Android es la gran cantidad de herramientas para el desarrollo de aplicaciones, con ayuda de diferentes lenguajes de programación como java, C, BASIC, entre otros, en caso de no tener grandes conocimientos de programación Android cuenta con un entorno de desarrollo de aplicaciones llamado MIT APP Inventor 2 el cual trabaja mediante el armado de bloques el cual facilita el desarrollo de aplicaciones basado en interfaces sencillas.

En el desarrollo de esta propuesta se utilizó el entorno MIT APP Inventor 2 (ilustración 2.26) el cual se describirá más afondo en los puntos siguientes.

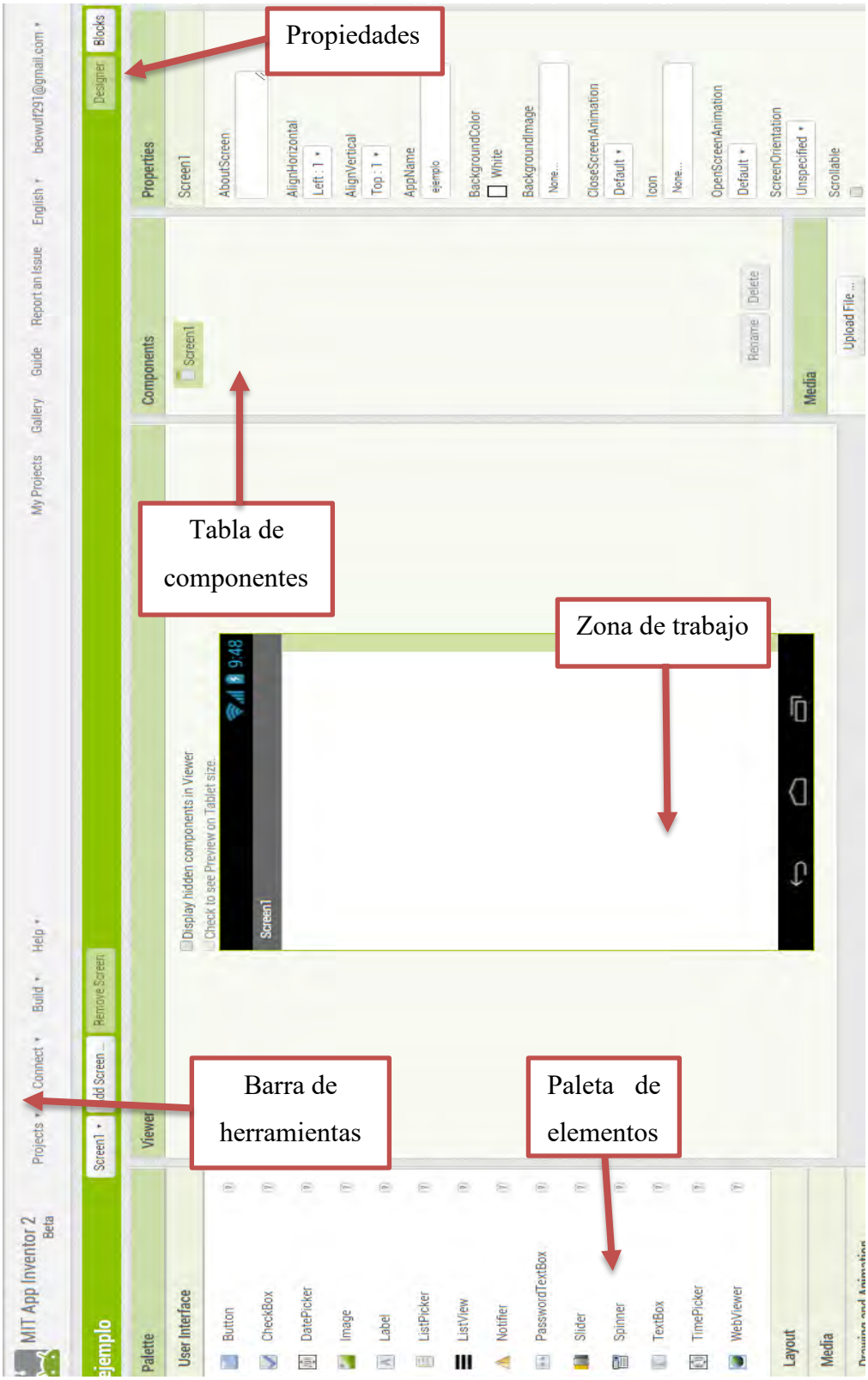

Ilustración 2.26 entorno de programación APP inventor 2

2.7.1 Paleta de elementos

Esta sección del entorno ubicada en la parte izquierda cuenta con una paleta, la cual contiene los siguientes elementos:

- *User interface*: en esta sección se encuentran elementos visuales y de interacción que pueden o no ser alterados, tales como botones, etiquetas, notificaciones e imágenes.
- *Layout*: esta sección cuenta con arreglos horizontales, verticales y en tabla para ayudar a estructurar los elementos dentro de la pantalla
- *Media*: esta sección se utiliza para agregar elementos multimedia como video, sonido, reconocimiento de voz, etc.
- *Drawing and animation*: esta sección permite crear dibujos y animaciones sencillas en pequeñas partes de la zona de trabajo.
- *Sensors*: esta sección se utiliza para habilitar el uso de sensores del teléfono en la aplicación, alguno de ellos son el GPS, giroscopio, reloj, acelerómetro, etc.
- *Social*: esta sección habilita opciones sociales dentro de la aplicación como el compartirlo en redes sociales, realizar llamadas o mensajes de texto.
- *Storage*: esta sección habilita el almacenamiento y recuperación de información por parte de la aplicación
- *Connectivity*: este apartado permite habilitar la comunicación inalámbrica la cual puede ser por *bluetooth*, internet o entre aplicaciones.

2.7.2 Tabla de componentes

La tabla de componentes situada a la parte derecha de la zona de trabajo es donde quedan guardadas todos los componentes visibles y no visibles integrados a la zona de trabajo a través de la paleta de elementos.

La tabla de componentes nos permite seleccionar los elementos y renombrarlos para la comodidad del usuario y también habilitan la zona de propiedades del componente en específico.

2.7.3 Propiedades

La zona de propiedades ubicada en la parte derecha del entorno nos permite modificar las características de los elementos integrados a la aplicación, como el color, tamaño de fuente, ubicación, valores, etc. Dichas características varían según el elemento y se guardan automáticamente.

2.7.4 Zona de trabajo

La zona de trabajo ubicada en la parte central del entorno es donde se lleva a cabo la construcción visual de la aplicación o en otras palabras como se observará una vez cargada al Smartphone.

2.7.5 Bloques

Esta segunda pantalla (ilustración 2.27) a la cual se tiene acceso al presionar el botón blocks, esta nueva sección es donde se realizará la programación de la aplicación mediante construcción de bloques.

Dentro de la lista de bloques encontramos 8 secciones básicas que despliegan una lista de bloques, los cuales pueden aumentar al integrarse elementos extra, las secciones básicas de bloques serán explicadas a continuación:

- *Control*: dentro de esta sección aparecerá una lista de bloques los cuales cuentan con estructuras de control como condicionales (*if*), ciclos de repetición (*for,while*) y otras instrucciones dirigidas a la aplicación como abrir o cerrar una pantalla.
- *Logic*: los bloques en esta sección incluyen dar valores booleanos como verdadero o falso y realizar operaciones lógicas *and* y *or*.
- *Math*: es esta sección se encuentran bloques para realzar operaciones aritméticas, trigonométricas y conversiones.
- *Text*: esta sección de bloques son utilizados para colocar texto en la aplicación, esto puede ser utilizado para enviar o recibir información.
- *Lists*: esta sección de bloques funciona para la creación y gestión de listas dentro de la aplicación.

- *Colors*: esta sección de bloques se utiliza para asignar color a algún elemento, la elección puede estar en la gama básica de 7 colores o definirlo mediante balance de RGB.
- *Variables*: esta sección de bloques cumple con la función de crear variables globales, así como darle valores y utilizarlos en otros bloques.
- *Procedures*: esta última sección básica se utiliza para crear, llamar y pedir resultados de algún procedimiento.

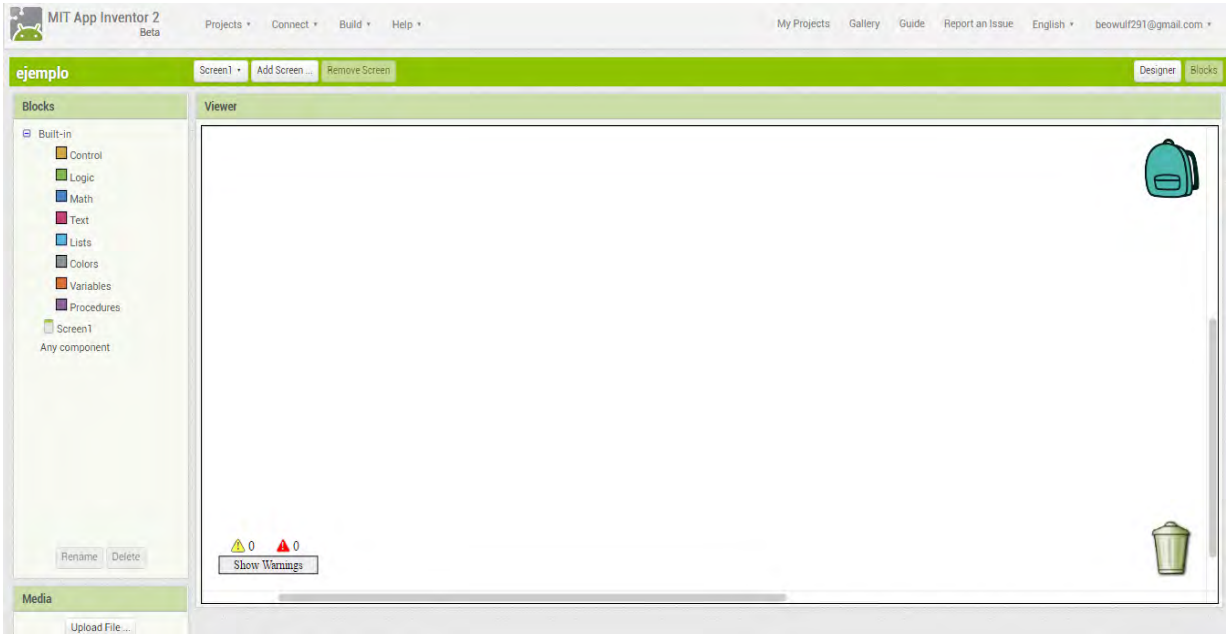

Ilustración 2.27 APP inventor 2 zona de programación de bloques

2.7.6 Barra de herramientas

Esta sección ubicada en la parte superior cuenta con las siguientes opciones:

- *Projects*: esta sección nos despliega las opciones abrir los proyectos realizados, iniciar uno nuevo, guardar o borrar el proyecto, cargar o guardar el proyecto en el equipo con la extensión .aia.
- *Connect*: esta sección despliega las opciones de abrir un simulador, o conectar el Smartphone al entorno de desarrollo.

- *Build*: esta sección nos permite compilar la aplicación y generar un apk en el ordenar para cargarlo al Smartphone o crear un código QR que al escanearlo inicia la descarga del archivo (.apk) en el Smartphone.
- *Help*: esta sección nos despliega un lista de posibles dudas y soluciones acerca del entorno.
- *My projects*: es botón de acceso directo que nos dirige a los proyectos realizados anteriormente.
- *Gallery*: este acceso directo nos muestra aplicaciones publicadas por otros usuarios.
- *Guide*: este acceso directo nos abre una pestaña con la información de actualizaciones, documentos, soporte y otras dudas acerca del entorno.
- *Report an issue*: este acceso directo nos permite reportar algún error dentro del entorno de desarrollo, para que pueda ser corregido.
- Idioma: esta sección nos permite cambiar el idioma del entorno entre diez posibles idiomas.
- Usuario: esta sección nos muestra el correo electrónico del usuario y nos permite gestionar el perfil o salir del sitio.

2.8 Sistemas de control

Un sistema de control se puede definir como un conjunto de dispositivos encargados de controlar un sistema reduciendo probabilidades de fallo y obtener los resultados deseados, su funcionamiento se puede describir simplemente con el siguiente esquema (Alberto Perez, Perez Hidalgo, & Perez Berenguer, 2017).

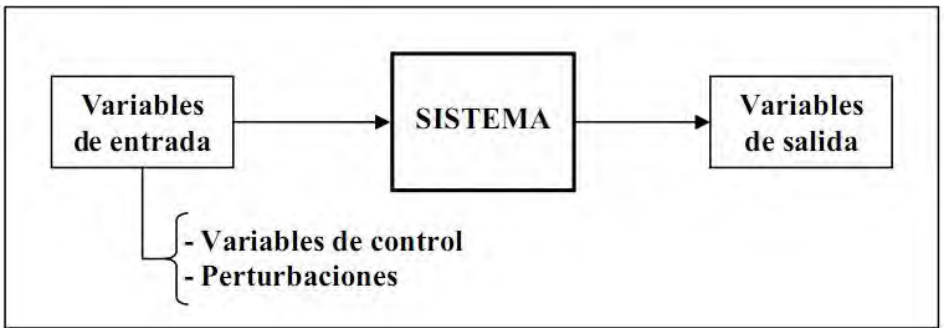

Ilustración 2.28 esquema general de los sistemas de control

Los sistemas de control cuentan con varios tipos, pero estos se pueden clasificar generalmente en solo dos: lazo abierto y cerrado, los cuales serán descritos en los siguientes puntos.

2.8.1 Lazo abierto

Un sistema de lazo abierto se define como un sistema que procesa una señal de entrada y en base a ella genera una señal de salida independiente y esta misma no regresa nada a la etapa de control, por lo que este tipo de sistemas suelen ser sencillos, pero pierden fácilmente estabilidad ante perturbaciones, el esquema general de un sistema de lazo abierto es el siguiente.

Ilustración 2.29 esquema de sistema de lazo abierto

2.8.2 Lazo cerrado

Los sistemas de lazo cerrado trabajan de manera similar que los de lazo abierto, con la diferencia de que la acción de control está en función de la señal de salida, ya que estos retroalimentan al controlador para ajustar la acción de control, estos suelen ser más complejos, pero menos susceptibles a perturbaciones, el esquema general de un sistema de lazo abierto es el siguiente:

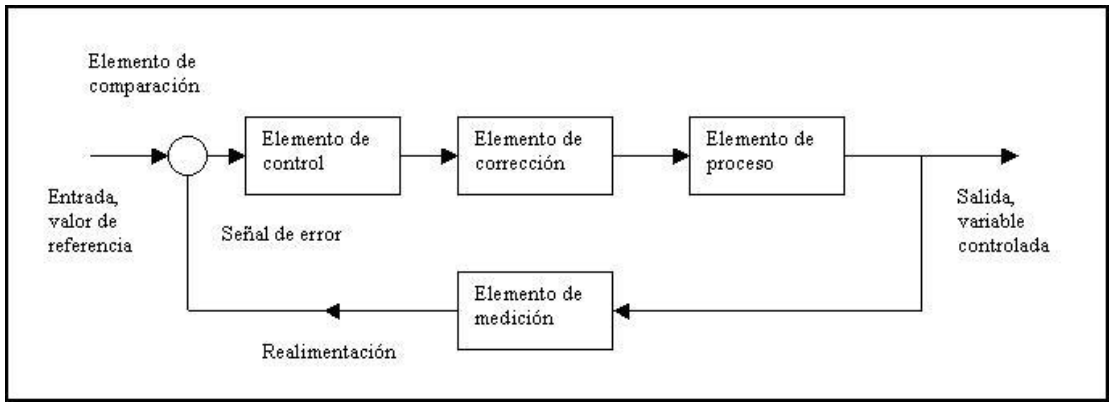

Ilustración 2.30 esquema de sistema de lazo cerrado

2.8.3 Microcontroladores

Los microcontroladores se pueden definir como un circuito integrado programable, conformado por una unidad central de procesamiento (CPU), dos unidades de memorias (RAM y ROM), puertos de entrada y salida y periféricos, todos ellos en un encapsulado, el cual es capaz de ejecutar ordenes programadas en su memoria ROM (Cucho M. , Orihuela Q., Sánchez P. , & Rodríguez p., 2017), dichos microcontroladores pueden variar su cantidad de periféricos, esto dependiendo del modelo.

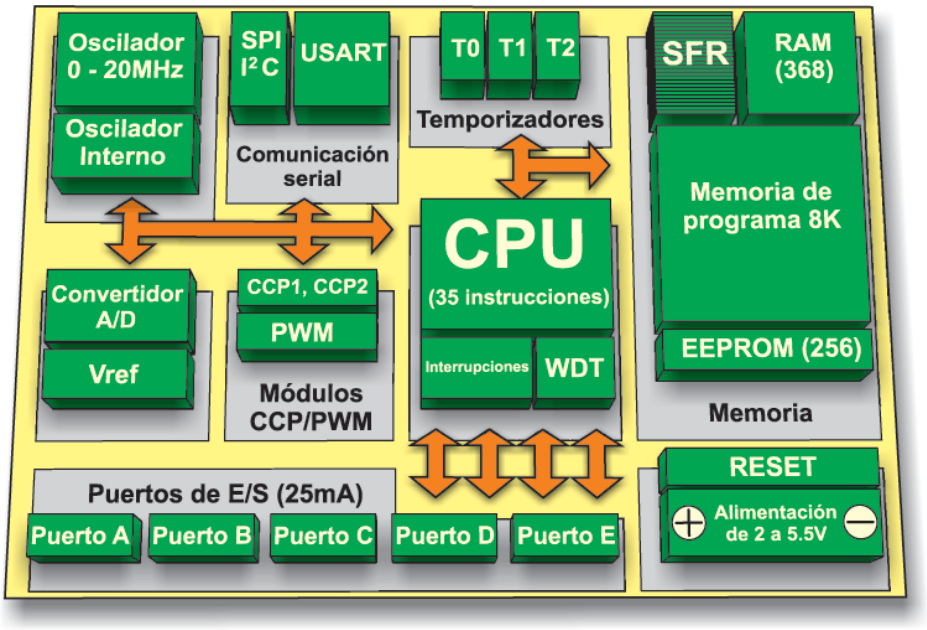

Ilustración 2.31 estructura de un microcontrolador

Como se mencionó anteriormente, los microcontroladores necesitan un programa para trabajar, ya que el propósito del microcontrolador es leer y ejecutar el programa cargado en su memoria, esta actividad requiere conocer el lenguaje de programación, el cual puede variar dependiendo del fabricante y el entorno de programación que se decida utilizar.

2.8.3.1 Tarjetas de desarrollo

Las tarjetas de desarrollo son una herramienta para diseño y prototipado rápido de sistemas digitales o analógicos, que se presenta como un elemento muy útil para el mejoramiento de los procesos de diseño debido a disminución del tiempo de validación de los diseños, así como la posibilidad que ofrece de ser una solución y un producto final (Geusseppe González Cárdenas, 2017).

Dentro del mercado existen una gran cantidad de tarjetas de desarrollo, algunas de ellas son:

- Arduino
- Tessel
- Raspberry pi
- Beaglebone
- ChipKit
- LaunchPad

Ilustración 2.32 tarjetas de desarrollo

De las tarjetas mencionadas anteriormente se decidió trabajar con la tarjeta Arduino modelo uno al cumplir con las siguientes características:

- Microcontrolador ATmega328.

- Voltaje de entrada 7-12V.
- 14 pines digitales de I/O (6 salidas PWM).
- 6 entradas análogas.
- 32k de memoria Flash.
- Reloj de 16MHz de velocidad.

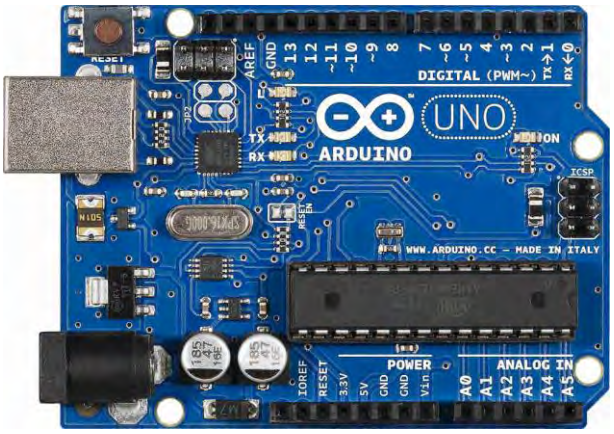

Ilustración 2.33 tarjeta Arduino uno

2.9 Etapa de potencia

En la electrónica existen dispositivos electrónicos los cuales son usados para transformar y controlar voltajes y corrientes elevados, algunos de estos dispositivos son utilizados para controlar la conducción eléctrica y así formar circuitos de control, también son utilizados para alimentar controladamente otros dispositivos (rashid, 2017), algunos de estos dispositivos son los siguientes:

Diodos de potencia

Este dispositivo es un semiconductor cuya única función es permitir el flujo de corriente en un solo sentido, de ánodo a cátodo, al igual que los diodos rectificadores, la característica de estos diodos es que pueden soportar altos niveles de corriente con pequeñas caídas de tensión.

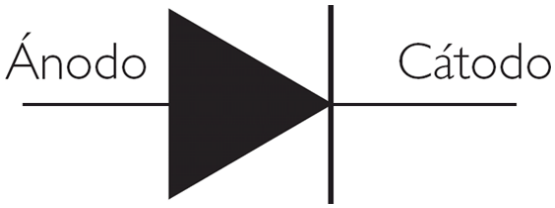

Ilustración 2.34 símbolo eléctrico de un diodo

SCR (rectificador controlado por silicio)

Este dispositivo está formado por cuatro capas de material semiconductor las cuales pueden ser PNPN o NPNP, cuanta con tres terminales ánodo, cátodo y *gate* o puerta, su funcionamiento es similar al de un diodo con la diferencia de que el SCR necesita un voltaje de activación en la terminal de puerta para iniciar la conducción entre ánodo y cátodo.

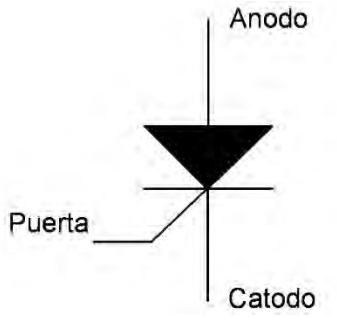

Ilustración 2.35 símbolo eléctrico del SCR

TBJ (transistor de juntura bipolar) de potencia

Estos dispositivos trabajan como interruptores de potencia controlados por corriente, estos permiten una conducción entre sus terminales colector y emisor cuando se excita la terminal de base con una corriente, este dispositivo cuenta con dos tipos diferentes NPN y PNP, las principales características de estos dispositivos son que pueden trabajar con altos niveles de voltaje y corriente, bajos tiempos de respuesta y son más baratos que los FET.

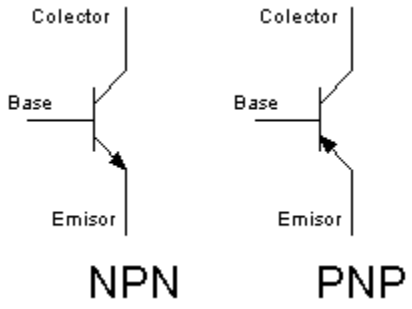

Ilustración 2.36 símbolos eléctricos del TBJ

MOSFET (transistor de efecto de campo metal-oxido-semiconductor)

Los MOSFET al igual que los TBJ trabajan como interruptores de potencia, con la diferencia que estos funcionan con voltaje, dicho dispositivo cuenta con 3 terminales drenador (D), fuente (F) y puerta (G),los cuales pueden ser tipo P o N, al excitarse con un voltaje es en su terminal puerta, este permite el paso de corriente por sus otras dos terminales, una de las ventajas que muestra dicho dispositivo es que tiene un bajo consumo, gran capacidad de integración por su reducido tamaño y una velocidad de conmutación muy alta (nanosegundos), la desventaja que tienen es que son costos y son susceptibles a corrientes, por lo que se deben manejar con cuidado.

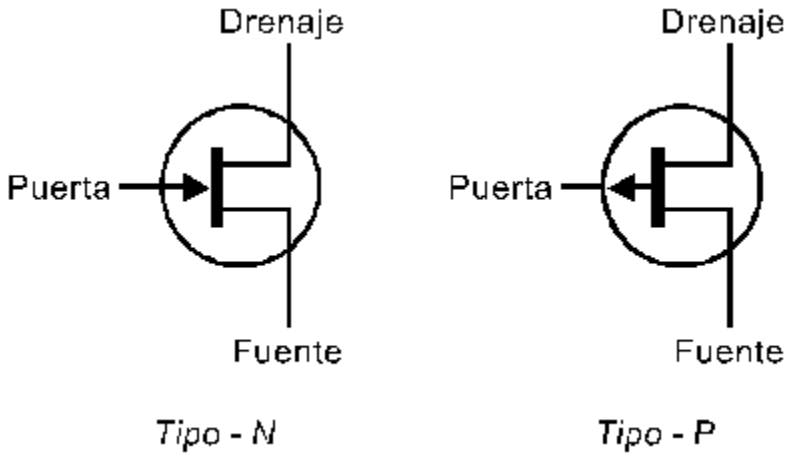

Ilustración 2.37 símbolos eléctricos del MOSFET

TRIAC (triodo para corriente alterna)

Este dispositivo semiconductor es un conmutador bidireccional, es decir es un interruptor de corriente alterna, el dispositivo cuenta con 3 terminales, A1, A2 y puerta, este trabaja cuando

una señal de disparo se aplica en la terminal puerta y permite la conducción entre las terminales A1 y A2, la versatilidad del dispositivo le permite trabajar como interruptor, atenuador, etc.

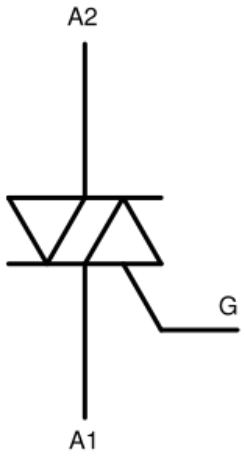

Ilustración 2.38 símbolo eléctrico del TRIAC

Relevador

Este es un dispositivo electromagnético que funciona como un interruptor controlado por medio de una bobina y un electroimán el cual acciona uno o más contactos permitiendo abrir o cerrar circuitos eléctricos separados, por lo que utilizados como un aislante de circuitos que funcionan con diferentes niveles de voltaje o corriente, la ventaja de este dispositivo es que puede ser utilizado fácilmente como dispositivo de control en circuitos de baja y alta potencia, además de tener un bajo costo, su única limitante es que se trata de un dispositivo electromecánico, por lo que su velocidad de conmutación no es tan alta y su desgaste es mayor que el de un semiconductor.

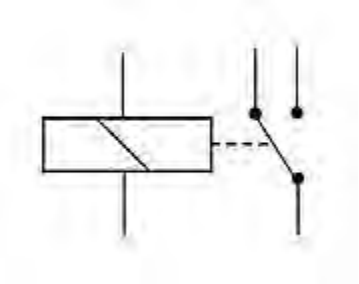

Ilustración 2.39 símbolo eléctrico de un relevador

Capítulo 3 Desarrollo experimental de la propuesta

Para el desarrollo de este proyecto se construyó un huerto urbano, el cual debía de estar distribuido a manera que facilite el riego de las plantas, no dificulte su crecimiento y se pueda montar un sistema de riego automatizado con sus respectivos sensores, actuadores y sistema de control.

A continuación, se explicará a detalle cada aspecto del desarrollo del proyecto desde el diseño y construcción del huerto, hasta el desarrollo de una aplicación para teléfono inteligente para la configuración de los parámetros.

3.1 Construcción del huerto urbano

Para comenzar el proyecto, se construyó un huerto urbano con materiales económicos y fáciles de encontrar; el huerto cuenta con una capacidad máxima de cinco plantas aromáticas.

Los objetivos que debe cumplir son: permitir el libre crecimiento de las cinco plantas aromáticas, soportar el montaje del sistema de control y riego, soporte el traslado a cualquier lugar del hogar de manera sencilla y finalmente, evite el derrame de agua.

3.1.1 Armado de la estructura

Para comenzar la construcción del huerto se armaron 5 anillos de metal (ilustración 3.1) con un diámetro de 17 cm, los cuales soportarán a las macetas, estos a su vez se fijarán a una placa de madera con dimensiones de 1.25 m x 1.25 m x 15 mm; los anillos de metal fueron montados en la placa de madera formando una “V”, con el fin de facilitar el riego, el desagüe y para el libre crecimiento de las plantas.

Ilustración 3.1 anillo de metal

El siguiente paso para la construcción del huerto fue hacer una base de metal con forma de "L" sobre la cual se montó la placa de madera con los demás elementos fijos en ella, esta estructura cuenta con las dimensiones de 1.25 m x 0.62 m x 0.30 m, el montaje completo del huerto mostrado en la siguiente imagen (ilustración 3.2) nos permite colocar las cinco plantas en la parte frontal y así disponer de la parte trasera para montar el sistema de riego automatizado.

Ilustración 3.2 montaje frontal de huerto

3.1.2 Implementación del sistema de riego

Para la implementación del sistema de riego se utilizó el espacio de la parte posterior del huerto y los materiales mencionados en el capítulo anterior como: mangueras, conexiones plásticas, boquillas de plástico, servomotores y bombas de agua.

Se comenzó con la instalación de un depósito de agua, el cual se instaló utilizando una estructura metálica para soportarlo a una altura de 70 cm, debido a que se utilizó una bomba de baja potencia para el riego.

Para la distribución de agua en los diferentes canales, se probaron dos montajes. El primero fue hecho en una placa de acrílico, la cual se perforó y cortó para la instalación del servomotor y las cinco boquillas alrededor del mismo, cubriendo los 180 grados que el servomotor es capaz de rotar. Este primer montaje se colocó en la parte superior del huerto sostenido por una repisa (ilustración 3.3), posteriormente se instaló la bomba de agua con la manguera de gasolina, la cual se fijó al servomotor con ayuda de una placa y una abrazadera de metal. Se decidió no utilizar este montaje debido a que no cumplía con las expectativas de movilidad del servomotor, y tenía goteos que ponían en riesgo el resto de los elementos electrónicos.

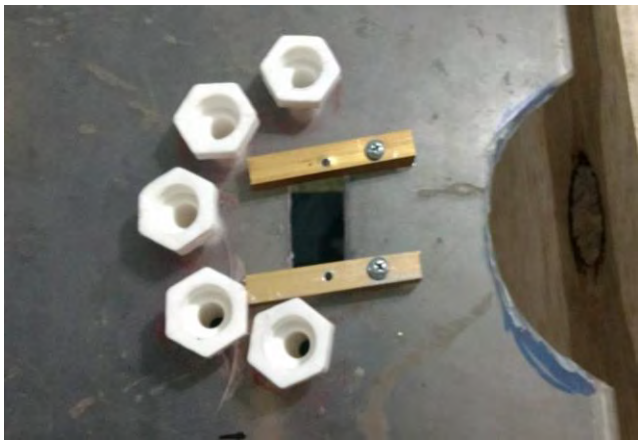

Ilustración 3.3 primer sistema de riego

Por las razones anteriores, se decidió utilizar un montaje diferente, se desechó la placa de acrílico. En el segundo montaje se fijaron las boquillas con abrazaderas de metal a la placa de madera, las cuales fueron fijadas con tornillos, al igual que las macetas fueron colocadas en forma de “V”, esto para facilitar el movimiento del servomotor, el cual fue fijado en la parte superior central de la placa de madera con ayuda de otra abrazadera. Este segundo

montaje mostró resultados satisfactorios en aspectos de movilidad, por lo que se decidió proseguir con el armado del sistema de riego. Para lograr esto, se perforó la placa de madera 3 centímetros arriba de cada maceta, se atravesó por cada agujero la manguera de plástico de media pulgada, la cual se unirá a las cinco boquillas con ayuda de conexiones plásticas en forma de codo para facilitar el paso del agua, teniendo como resultado final la ilustración 3.4.

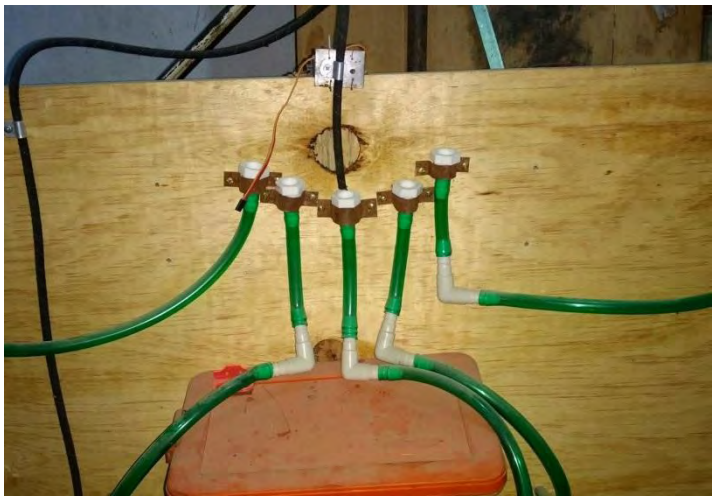

Ilustración 3.4 segundo sistema de riego

3.1.3 Armado de macetas con sistema de desagüe

Para finalizar el armado del huerto urbano se modificaron cinco macetas de plástico, implementado en ellas un sistema de desagüe, para ello se comenzó perforando el centro de la base de las macetas, en el cual se adhirió en la parte exterior una boquilla de plástico y en la parte interior una botella de plástico de 3 litros recortada hasta el límite de la maceta, como se puede apreciar en la ilustración 3.5. Una vez que se montaron las macetas y se verificó que no existían fugas de agua se conectó la manguera con las conexiones plásticas con forma de "T", como se muestra en la ilustración 3.6, todo esto con el fin de dirigir el agua que no es absorbida por la tierra a un segundo depósito de agua. El agua, puede ser reutilizada según el criterio del usuario.

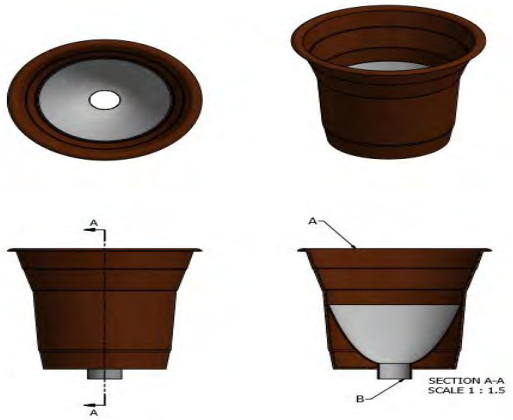

Ilustración 3.5 interior de la maceta modificada

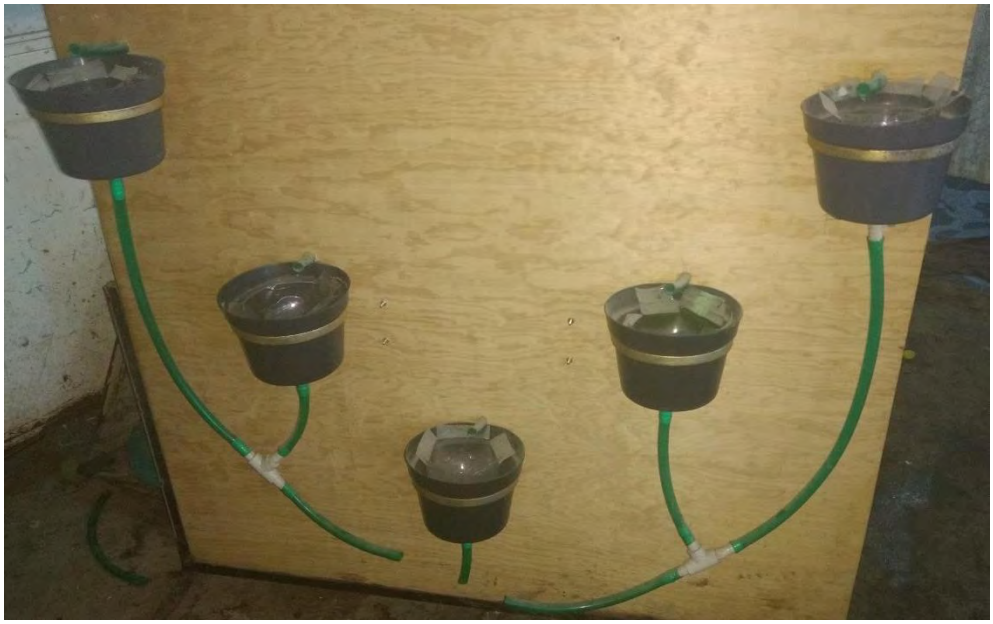

Ilustración 3.6 montaje de las macetas modificadas

3.2 Algoritmo

A continuación, se explicará claramente el algoritmo del proyecto, en el cual se detallan las acciones e instrucciones que realiza el sistema para su correcto funcionamiento, comenzado con la descripción del sistema hasta llegar al código fuente.

3.2.1 Diagrama de bloques

El diagrama de bloques es una representación gráfica del sistema, el cual se forma a partir de las relaciones de entradas con salidas (Porto, 2018), el diagrama de bloques de la propuesta es el siguiente.

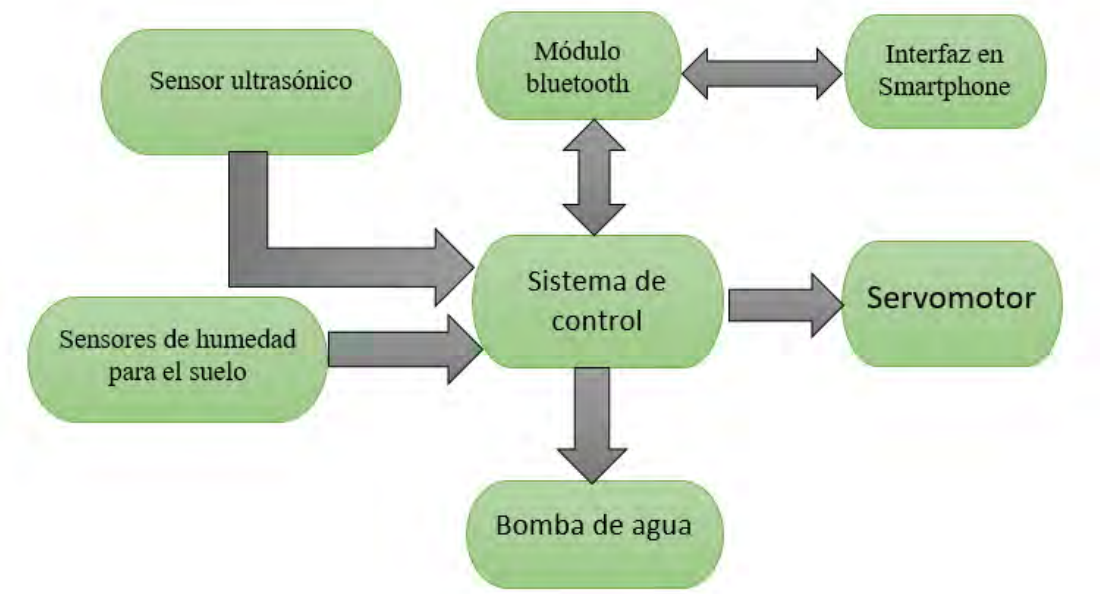

Ilustración 3.7 diagrama a bloques

3.2.2 Diagrama de flujo

El diagrama de flujo es la representación gráfica de los procesos que seguirá el algoritmo para resolver el problema; el algoritmo de esta propuesta se dividió en varios fragmentos, los cuales serán descritos a continuación.

En este primer fragmento del diagrama de flujo (ilustración 3.8) se describen los primeros procesos que realiza el sistema al ser energizado. Se comienza con el posicionamiento del servomotor a un canal de riego inicial, después, se accede a la información guardada en la memoria EEPROM, la cual contiene los valores de humedad óptimos del perejil, menta, cilantro, orégano y apio; finalmente se toman las lecturas de los sensores, las cuales se utilizarán en los siguientes fragmentos del diagrama de flujo.

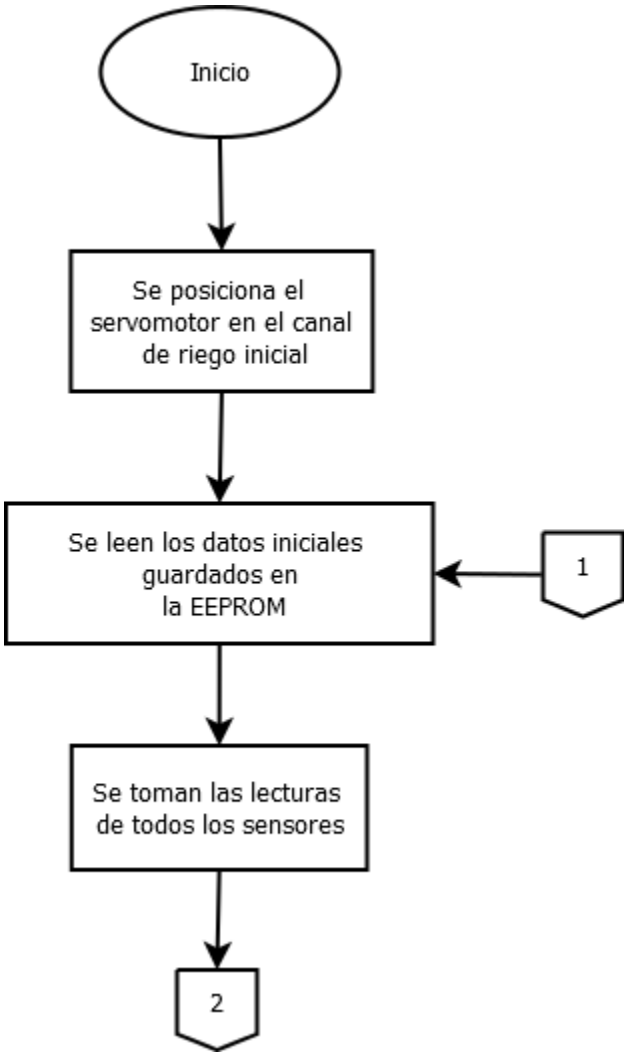

Ilustración 3.8 diagrama de flujo inicial

En el segundo diagrama (ilustración 3.9) se muestra el algoritmo utilizado para la comunicación serial mediante el módulo *bluetooth*. El algoritmo empieza preguntando si hay un dispositivo conectado, en caso de no haber, se dirige hacia las acciones correspondientes al *bluetooth*, y pasará al siguiente diagrama (ilustración 3.10). En caso de que si exista un dispositivo conectado, a este se le enviarán las lecturas de los sensores, además preguntará si el dispositivo conectado ha transmitido nuevos datos de configuración, en ese caso los nuevos datos se guardaran en la memoria EEPROM remplazando a los anteriores; la dirección del diagrama se redirige al primer diagrama de flujo para que se utilicen los nuevos datos guardados, en el caso de que el usuario no envié nuevos datos de configuración se pasará al tercer diagrama de flujo.

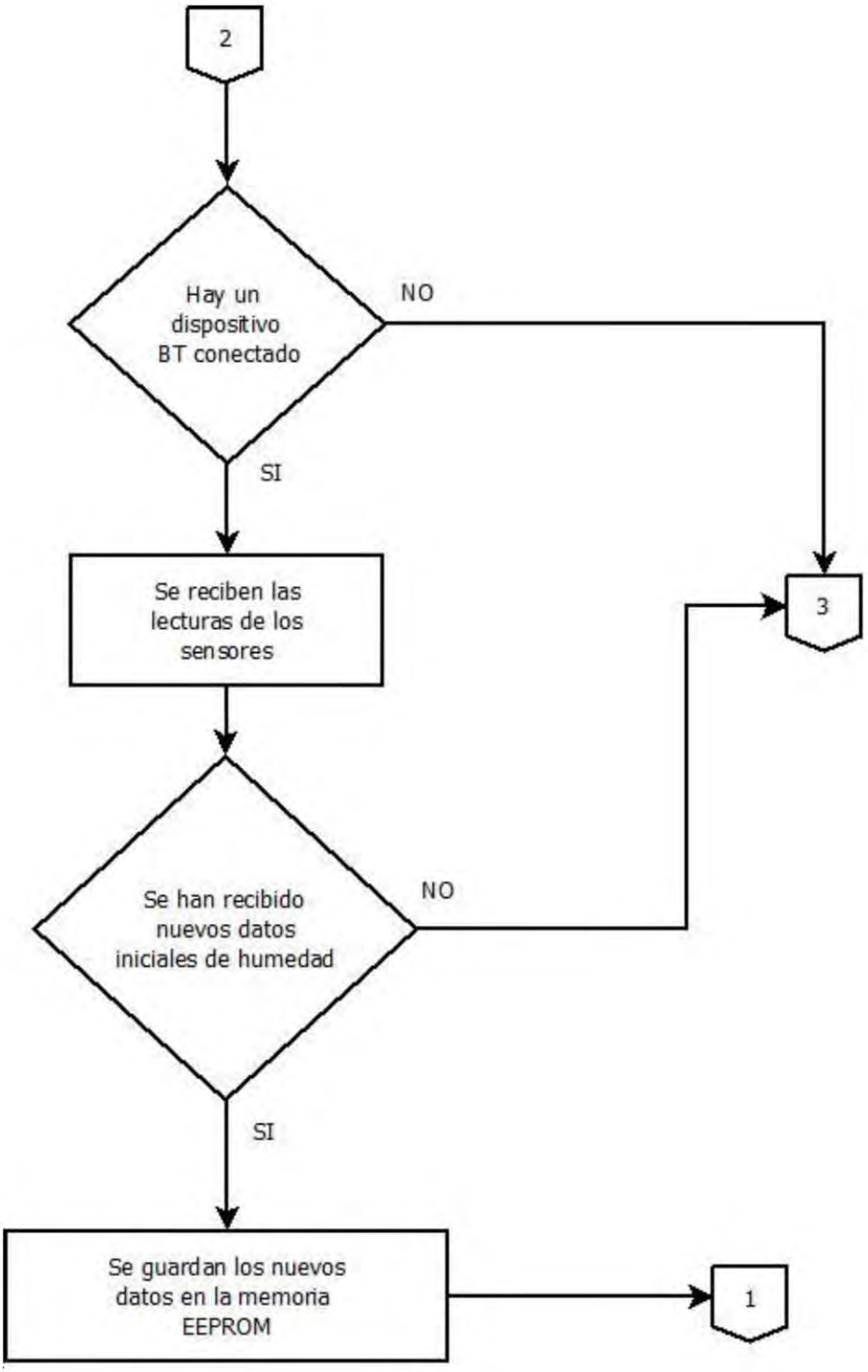

Ilustración 3.9 diagrama de flujo del algoritmo de comunicación

En el tercer diagrama de flujo (ilustración 3.10) se describen las acciones que se toman del sensor ultrasónico, comenzando con la pregunta: ¿el sensor ultrasónico trabaja correctamente? Esto se puede comprobar al comparar la lectura del sensor con la constante 111, si la lectura es igual, el sensor no está trabajando correctamente, por lo tanto, se mandara una señal de alerta al usuario y el flujo del programa, regresará al primer diagrama de flujo; en caso contrario se comparan las lecturas tomadas del sensor ultrasónico con la constante 15, en caso de tener una lectura mayor al 15, el prototipo contará con suficiente agua en su depósito para pasar al quinto diagrama, en caso contrario se pasará al cuarto diagrama.

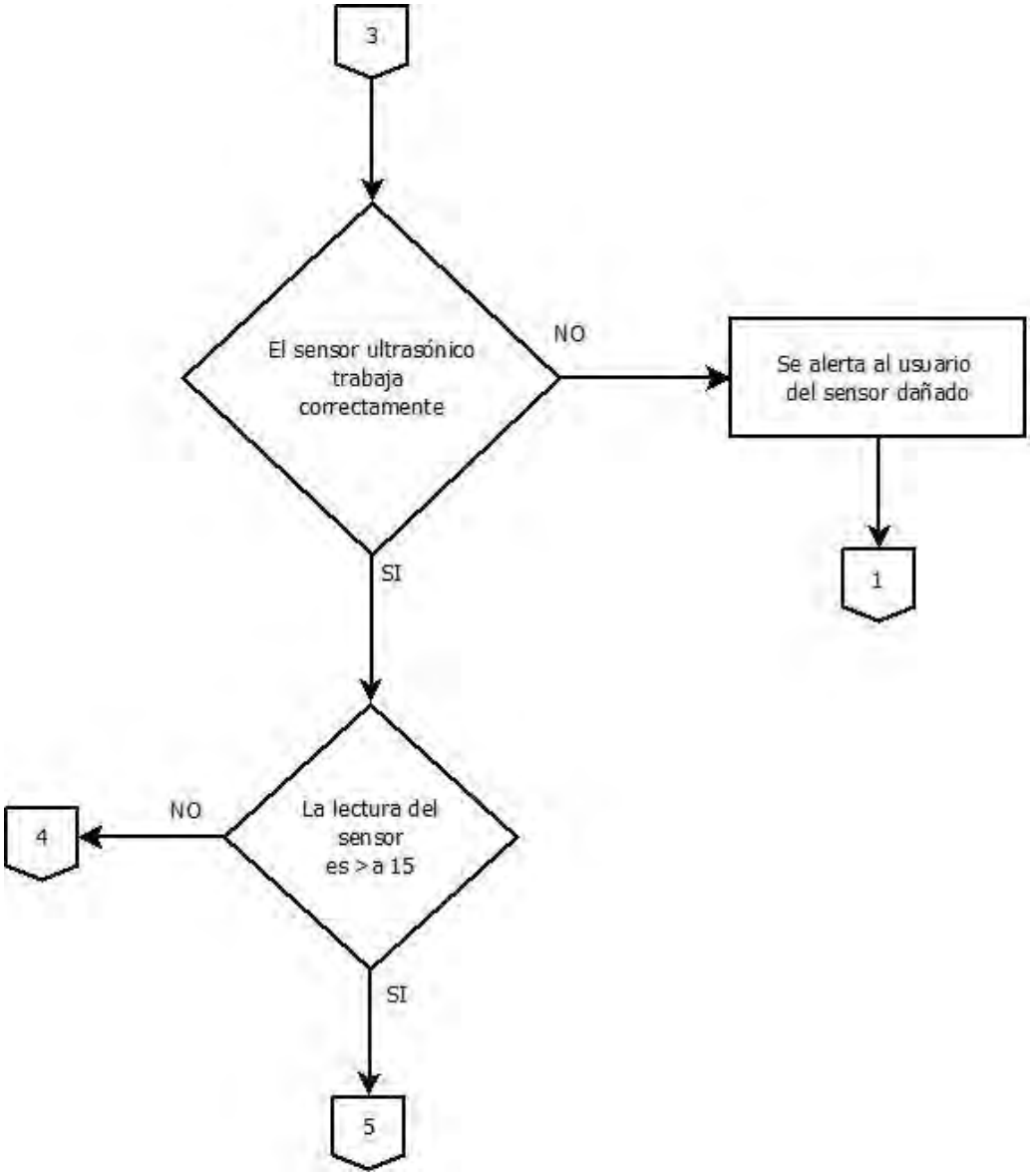

Ilustración 3.10 diagrama de flujo del sensor ultrasónico

El cuarto diagrama de flujo (ilustración 3.11) se describe el algoritmo en el caso en que la medición sea menor a 15, la primera acción es alertar al usuario del bajo nivel de agua, acto seguido el sistema tomara lecturas del sensor ultrasónico y solo podrá continuar hasta que la lectura sea mayor a 80, dirigiendo el flujo al primer diagrama.

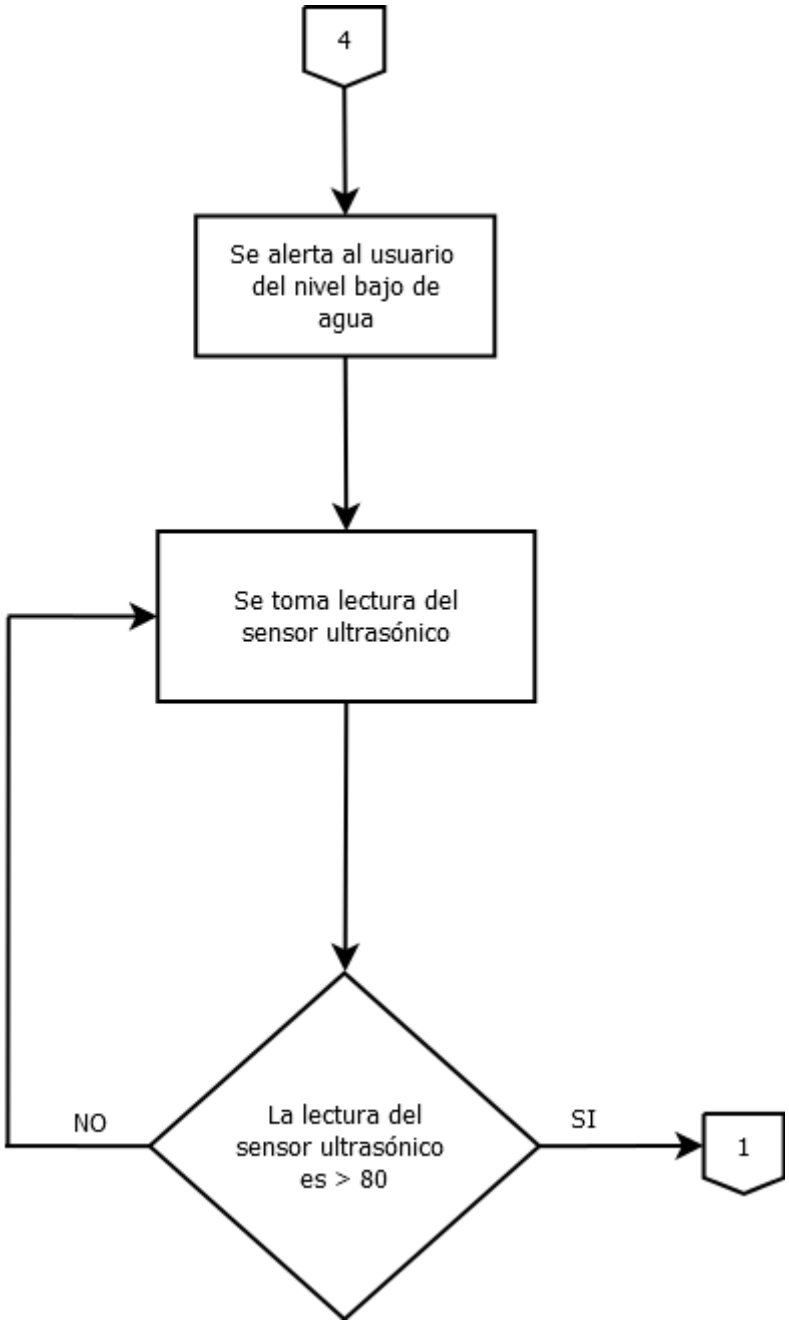

Ilustración 3.11 diagrama de flujo de nivel bajo de agua

El quinto diagrama de flujo (ilustración 3.12) describe el algoritmo del sensor de humedad, en este caso al contar con cinco sensores de humedad se utiliza la variable “i”, la cual inicia con el valor de 1 y se incrementara hasta llegar a 5 para cubrir los sensores que realizan el mismo algoritmo; la primera acción es tomar las lecturas del sensor en curso y con ellas comprobar si el sensor trabaja correctamente; para ello, se comparan dos mediciones del sensor realizadas consecutivamente, y si estas no son iguales el sensor se encuentra dañado, se alerta al usuario e incrementa el valor en 1 de la variable “i”, posteriormente se compara que el valor de la variable i no sea mayor a 5, al tener un valor menor a 5 se toman las lecturas del siguiente sensor y se repite el proceso; en caso de que la variable supere el valor de 5, lo cual implica que se han revisado todos los sensores, y se restablecerá el valor de la variable “i” y se regresará al primer diagrama; en el caso de que el sensor trabaje correctamente se pasará al sexto diagrama que corresponde a la rutina de riego.

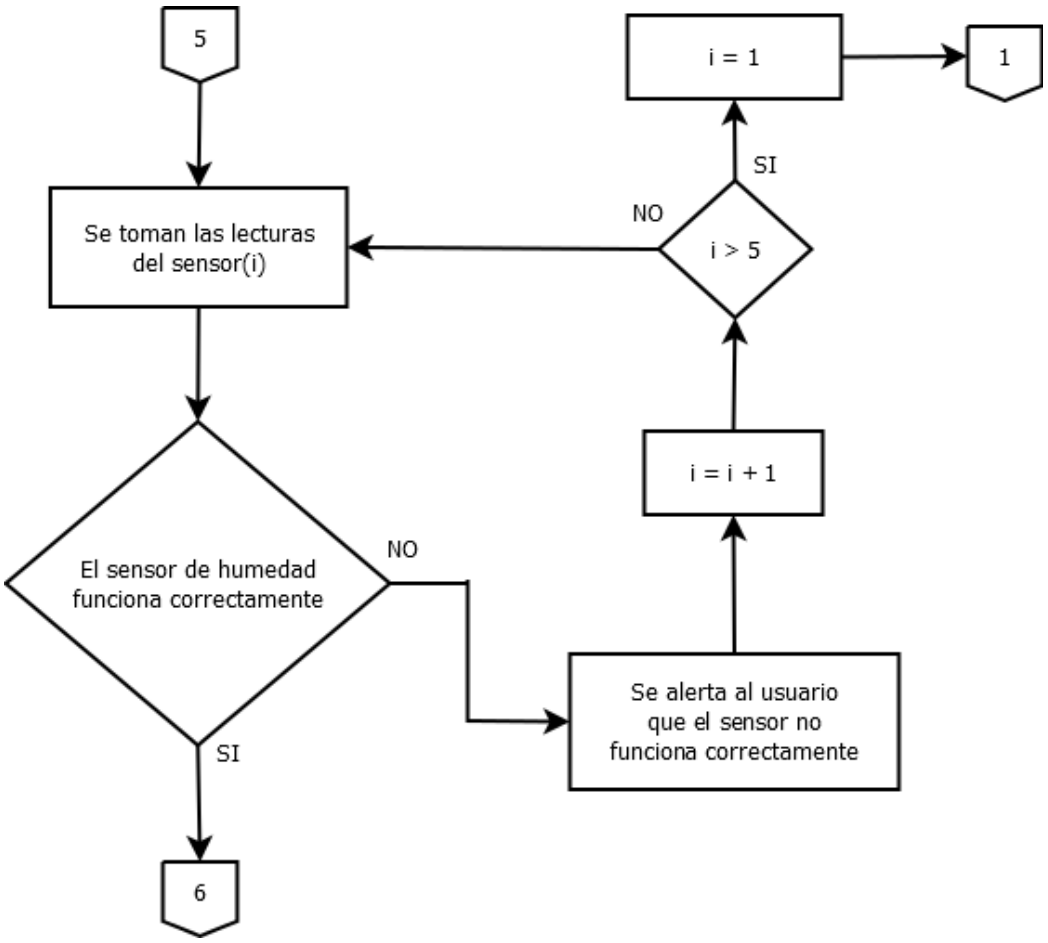

Ilustración 3.12 diagrama de flujo del sensor de humedad

Finalmente, el sexto diagrama de flujo (ilustración 3.13) describe el algoritmo de la rutina de riego, al igual que el diagrama anterior utiliza la variable “i” para identificar el número del sensor que se está revisando; el diagrama comienza comparando la lectura del sensor con un umbral de humedad, el cual se obtiene de los datos guardados en la memoria EEPROM; en caso de que la medición sea menor que el umbral, se inicia el riego de la planta correspondiente al sensor, para esto, se posiciona el servomotor en el canal de riego, se enciende la bomba de agua durante diez segundos y se espera diez segundos más para tomar nuevamente mediciones del sensor y compararla nuevamente, esta acción se repetirá hasta cumplir con la condición anterior; una vez superado el nivel de umbral o desde un principio este fuera mayor, se incrementara en 1 la variable “i” y se comprueba que esta no sea mayor a 5, en caso de ser menor se redirige el flujo al quinto diagrama, de lo contrario ya se habrán revisado todos los sensores, el valor se “i” se restablecerá y se regresara al primer diagrama de flujo.

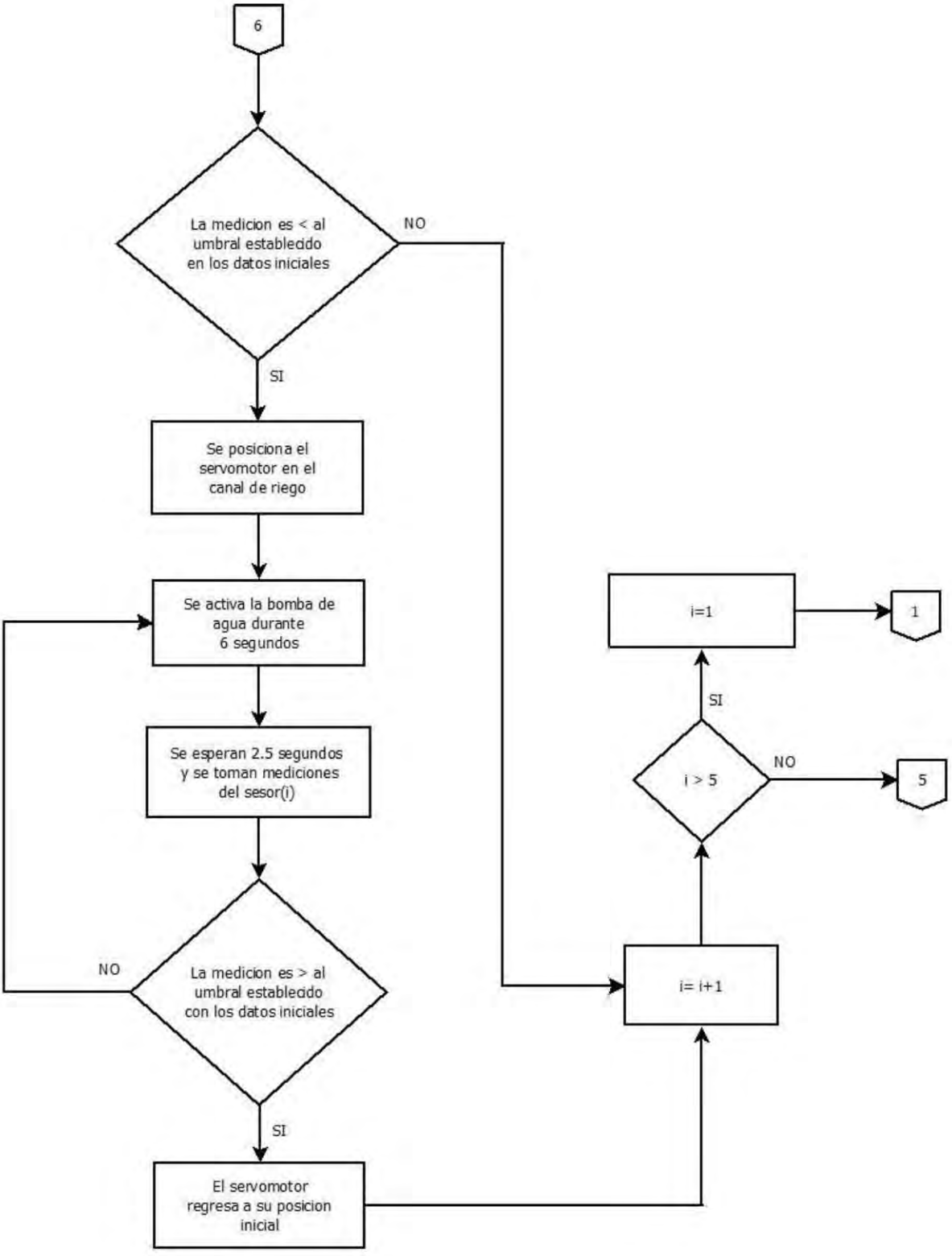

Ilustración 3.13 diagrama de flujo de la rutina de riego

3.2.3 Código en el entorno Arduino

En la propuesta presentada se utilizaron dos programas diferentes, el primer programa se utiliza para guardar datos iniciales en la memoria EEPROM del microcontrolador, los cuales serán utilizados en el segundo programa, el cual se ocupa del monitoreo y funcionamiento completo del prototipo.

3.2.3.1 Programa de carga de valores iniciales

En este primer programa (ilustración 3.13) hacemos uso de la biblioteca <EEPROM.h> para guardar cinco variables de clase entera con los valores de humedad óptima del perejil, menta, cilantro, orégano y apio y a su vez se guardan estos valores en localidades de la memoria EEPROM, para su uso en el segundo programa.

```
#include <EEPROM.h>

void setup() {
int val1=55 ;
int val2=85;
int val3=75 ;
int val4=70 ;
int val5=55 ;

 EEPROM.write(0,(byte)val1);
 EEPROM.write(1,(byte)val2);
 EEPROM.write(2,(byte)val3);
 EEPROM.write(3,(byte)val4);
 EEPROM.write(4,(byte)val5);
}
```

Ilustración 3.14 código del primer programa

3.2.3.2 Programa principal

Este segundo programa inicia declarando las variables globales a utilizar (ilustración 3.15), entre las cuales se encuentran: las de transmisión siendo del tipo carácter con un máximo de 40, las variables de recepción de datos (hpN) de tipo entero, las cuales su única función es recibir los datos transmitidos por el Smartphone, las variables de control de humedad (dhpN) de tipo entero y finalmente las variables para guardar las mediciones de los sensores de humedad (lecN) de tipo entero.

```
int hp1; //variables de recepcion de datos
int hp2;
int hp3;
int hp4;
int hp5;
int dhp1;  // variables para control de humedad
int dhp2;
int dhp3;
int dhp4;
int dhp5;
int lec1;  //variables de lectura de humedad
int lec2;
int lec3;
int lec4;
int lec5;
int lec11; //variables utilizadas para saber el buen
int lec22; //funcionamiento del sensores
int lec33;
int lec44;
int lec55;
char buffer[40];//variables para transmision de informacion
char trans[40];
char env[40];
char dat[40];
char txrx[40];
```

Ilustración 3.15 fragmento de código "declaración de variables"

Continuando con las configuraciones del programa se declaran las bibliotecas a usar con la directiva #include y el nombre de la biblioteca, en caso de las bibliotecas "NewPing" y "servo" se definen los objetos a usar y los pines que utilizaran los dispositivos asociados a la biblioteca.

```
#include <EEPROM.h>  //biblioteca de la eeprom
#include <NewPing.h>   // configuracion del ultrasonico
#define TRIGGER_PIN  9
#define ECHO_PIN     8
#define MAX_DISTANCE 100
 int dis;
 int nivel;
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
#include <Servo.h>  //biblioteca del servomotor
Servo servo;
```

Ilustración 3.16 fragmento de código "declaración de bibliotecas"

Para finalizar las configuraciones (ilustración 3.17) se declaran los pines que activaran los actuadores, señales de alerta con la función "pinMode" declarándolos como pines de salida de información, posteriormente se inicia la comunicación serial del dispositivo con la función

"Serial.begin", en la cual se configura *baud rate* de transmisión que utilizará, finalmente se posiciona el servomotor con la función "servo.write" en su posición inicial en el tercer canal de riego (88 grados), para realizar dicha acción, se energiza el servomotor con la función "digitalWrite" en el pin 3, además se utiliza la función "delay" para que el servomotor tenga el tiempo suficiente de tomar su posición y finalmente se deshabilita el uso del servomotor.

```
void setup() {
pinMode(3,OUTPUT);//activador del motor
pinMode(2,OUTPUT);//activador de la bomba
servo.attach(5); //pin del servomotor
pinMode(6,OUTPUT);//señal de alerta de malfuncionamiento
pinMode(7,OUTPUT);//señal de nivel bajo
  Serial.begin(9600); //configuracion del serial
   digitalWrite(3,1); //posicionamiento del
   delay(200);        //servomotor a una
   servo.write(88);   // posicion inicial
   delay(2000);
   digitalWrite(3,0);
   delay(150);
```

Ilustración 3.17 fragmento de código "configuración de pines y comunicación serial"

Continuando con el algoritmo del programa (ilustración 3.18), se toman las lecturas de los sensores de humedad con la función "analogRead", la cual entrega valores entre 0 y 1023, estos son mapeados para que sean interpretados en porcentaje del 0 al 100%, posteriormente se guardan en las variables (lecN) y a su vez se toma la lectura del sensor ultrasónico, con ayuda de una literal, se obtiene la distancia en cm, la cual también se convierte en porcentaje para facilitar su interpretación por parte del usuario y el programa.

```
lec1= analogRead(A1); //lectura de sensores de humedad
lec2= analogRead(A2);
lec3= analogRead(A3);
lec4= analogRead(A4);
lec5= analogRead(A5);
lec1=map(lec1,0,1023,100,0); //mapeo de lecturas
lec2=map(lec2,0,1023,100,0);
lec3=map(lec3,0,1023,100,0);
lec4=map(lec4,0,1023,100,0);
lec5=map(lec5,0,1023,100,0);
int uS = sonar.ping_median(); //lectura del ultrasonico
dis= uS / US_ROUNDTRIP_CM;
nivel=map(dis,3,30,100,0);
```

Ilustración 3.18 fragmento de código "toma de lecturas y mapeo de los sensores"

Continuando con el programa, se hace uso de la memoria EEPROM con la función "EEPROM.read" se leen los valores almacenados en las cinco localidades y se guardan en las variables dhpN, una vez cargados los valores de la EEPROM se toma una segunda lectura de los sensores de humedad con la diferencia que se almacenaran en variables distintas a la primera muestra (lecNN).

```
dhp1=EEPROM.read(0);
dhp2=EEPROM.read(1);
dhp3=EEPROM.read(2);
dhp4=EEPROM.read(3);
dhp5=EEPROM.read(4);
lec11= analogRead(A1);
lec11=map(lec11,0,1023,100,0);
lec22= analogRead(A2);
lec22=map(lec22,0,1023,100,0);
lec33= analogRead(A3);
lec33=map(lec33,0,1023,100,0);
lec44= analogRead(A4);
lec44=map(lec44,0,1023,100,0);
lec55= analogRead(A5);
lec55=map(lec55,0,1023,100,0);
```

Ilustración 3.19 fragmento de código "segunda toma de lecturas y carga de valores por defecto"

El siguiente fragmento de código (ilustración 3.20), se utiliza la estructura "while(Serial.avaliable()>0)" como condición en la cual si no hay ningún dispositivo conectado de manera serial, las instrucciones que contiene serán ignoradas, en caso contrario

se toman los datos transmitidos por el *Smartphone* con la función “Serial.parseInt()” y los guarda en las variables de recepción de datos (hpN); una vez que termina la transmisión y se recibe un salto de línea (\n) se graban los datos recibidos en la memoria EEPROM con la función “EEPROM.write” cambiando el tipo de variable de entera a byte, finalmente se hace uso de la instrucción “break” para salir del ciclo de while y continuar con el resto del programa.

```
while(Serial.available()>0)  //inicia ciclo de recepcion de datos
 {

    int hp1=Serial.parseInt();// recepcion de datos x parte de la app
    int hp2=Serial.parseInt();
    int hp3=Serial.parseInt();
    int hp4=Serial.parseInt();
    int hp5=Serial.parseInt();

    if(Serial.read()=='\n')//finaliza recepcion de datos
    {
      EEPROM.write(0,(byte)hp1);  //carga de datos recibidos a la eeprom
      EEPROM.write(1,(byte)hp2);
      EEPROM.write(2,(byte)hp3);
      EEPROM.write(3,(byte)hp4);
      EEPROM.write(4,(byte)hp5);
}

 break;
```

Ilustración 3.20 fragmento de código "comunicación serial"

En el siguiente fragmento de código (ilustración 3.21) se revisa el funcionamiento de los sensores comenzando con el ultrasónico, el cual se compara con la literal “111”, si la medición del sensor es igual, el sensor se encuentra dañado y se manda una alerta al huerto, en caso contrario se continua comprobando el estado de los sensores de humedad comparando las dos mediciones tomadas lecN y lecNN (en este caso se muestra el sensor 1), si estas son iguales se transmiten las mediciones en una sola línea de caracteres, agrupando las mediciones con la función “sprintf” y enviándolas con la función “Serial.println”, posteriormente se inicia la rutina de riego la cual se explicará en el siguiente fragmento de código, en caso de que las mediciones no sean idénticas, el sensor se encontrará dañado por lo que se brinca la rutina de riego y se alerta al usuario.

```
if(nivel==111){ //se comprueba el estado del ultrasonico
  sprintf(buffer,"%d,%d,%d,%d,%d,%d,%s,%s",lec1,lec2,lec3,lec4,lec5,nivel,"averiado","ultrasonico");
  delay(500);
 Serial.print(buffer);
 Serial.println(",datos");
  digitalWrite(6,1);
  }
  else{
     if(lec1==lec11){ // sensor 1 operando correctamente

  sprintf(buffer,"%d,%d,%d,%d,%d,%d",lec1,lec2,lec3,lec4,lec5,nivel);
  delay(1000);
 Serial.println(buffer);
     }
     else //sensor 1 dañado
          {
            digitalWrite(6,1);

        sprintf(buffer,"%d,%d,%d,%d,%d,%d,%s,%s",lec1,lec2,lec3,lec4,lec5,nivel,"averiado","sensor 1 averiado");
        Serial.print(buffer);
        Serial.println(",datos");
          }
          }
```

Ilustración 3.21 fragmento de código "comprobación del estado de los sensores"

El siguiente fragmento de código (ilustración 3.22) nos muestra la rutina de riego en la cual se compara la medición del sensor con el valor de las variables de control de humedad menos 2 (dhpN-2), en caso de que la medición sea menor se posiciona el servomotor en el canal correspondiente y se inicia un ciclo de riego, en el cual se activa la bomba de agua por 6 segundos y vuelve a tomar mediciones hasta que sea mayor al valor de la variable de control de humedad más 2 (dhpN+2) y finalmente regresa el servomotor a su posición inicial.

```
if(lec1<=(dhp1-2)){  //rutina de riego planta
       digitalWrite(3,1);
       delay(200);
       servo.write(88);
       delay(2000);
        servo.write(38);
       delay(5000);
       digitalWrite(3,0);
       delay(150);
while(lec1<=(dhp1+2))
     {
       digitalWrite(2,1);//activacion de la bomba
       delay(6000);
       digitalWrite(2,0);
       delay(2500);
       sprintf(buffer,"%d,%d,%d,%d,%d,%d,%s,%s",lec1,lec2,lec3,lec4,lec5,nivel,"optimos","riego");
       lec1=analogRead(A1);
       lec1=map(lec1,0,1023,100,0);
     Serial.print(buffer);
     Serial.println(",datos");
     }
   delay(500); // se regresa el servomotor a su posicion original
   digitalWrite(3,1);
   servo.write(88);
   delay(4000);
   digitalWrite(3,0);
     }
```

Ilustración 3.22 fragmento de código "rutina de riego"

En este último fragmento de código (ilustración 3.23) se compara la medición del ultrasónico con un 15, esto con el motivo de que el dispositivo trabaje solo si el nivel de agua es mayor al 15%, en caso de que el nivel sea menor se manda una señal de alerta de nivel bajo, e inicia un ciclo de espera en el cual se mantendrá hasta que el nivel de agua sea mayor al 80%, evitando así que se dañen los actuadores por trabajar sin agua.

```
if(nivel>=15)//nivel de agua suficiente
  {
    digitalWrite(7,0);
delay(1000);

 else
  {
    digitalWrite(7,1);
    while(nivel>=80)
    {
        int uS = sonar.ping_median();
 dis= uS / US_ROUNDTRIP_CM;
nivel=map(dis,3,30,100,0);
              }
              digitalWrite(7,0);
    }
```

Ilustración 3.23 fragmento de código "nivel del depósito"

3.3 Desarrollo de la interfaz en Android

Para la creación de la interfaz se utilizó el entorno de desarrollo APP inventor 2, el cual es una plataforma en línea; para utilizar este entorno solo se necesita tener o abrir una cuenta de correo electrónico en Gmail para acceder a la plataforma desde el siguiente URL: http://ai2.appinventor.mit.edu/

El desarrollo de la interfaz se puede dividir en diferentes tareas, comenzando con diseñar la apariencia (lo que el usuario podrá observar), esto incluye el icono de la interfaz (ilustración 3.24 Una forma de hoja verde con engranajes dentro de él, 2013), la pantalla principal y de configuraciones, una vez terminada la apariencia se explicará el código de programación utilizado.

Ilustración 3.24 Una forma de hoja verde con engranajes dentro de él

3.3.1 Construcción de la pantalla principal

Para la construcción de la pantalla principal se utilizó un arreglo de tabla, sobre el cual se trabajará la pantalla principal. Uno de los requerimientos de la interfaz es la conexión *bluetooth*, la cual se pierde si se cambia a otra pantalla, por lo que la utilización de los arreglos de tabla nos permite trabajar diferentes pantallas en una sola y así poder mantener la conexión bluetooth.

Continuando con el diseño de la pantalla principal, se utilizaron etiquetas (*Label*) para colocar espacios donde se pudiera mostrar la información del huerto (porcentaje de humedad, nivel de agua y estado de los sensores), además de mostrar la información se agregaron 4 botones, el primero ubicado en la parte superior izquierda el cual nos permite acceder a otra pantalla, en la cual se encuentra una tabla con información de los niveles de humedad óptimos en las plantas aromáticas sugeridas. El segundo botón ubicado en la parte inferior derecha es un botón de salida, el cual se utiliza para cerrar la aplicación. El tercer botón se utiliza para volver visible el campo contraseña y el último botón tiene la función de acceder a la pantalla de configuración, siempre que la contraseña ingresada dentro del campo sea correcta. Finalmente se colocó un selector de lista (*ListPicker*), este con el motivo de mostrar una lista de dispositivos vinculados previamente y realizar una conexión *bluetooth*, la pantalla principal se muestra en la siguiente imagen (ilustración 3.25).

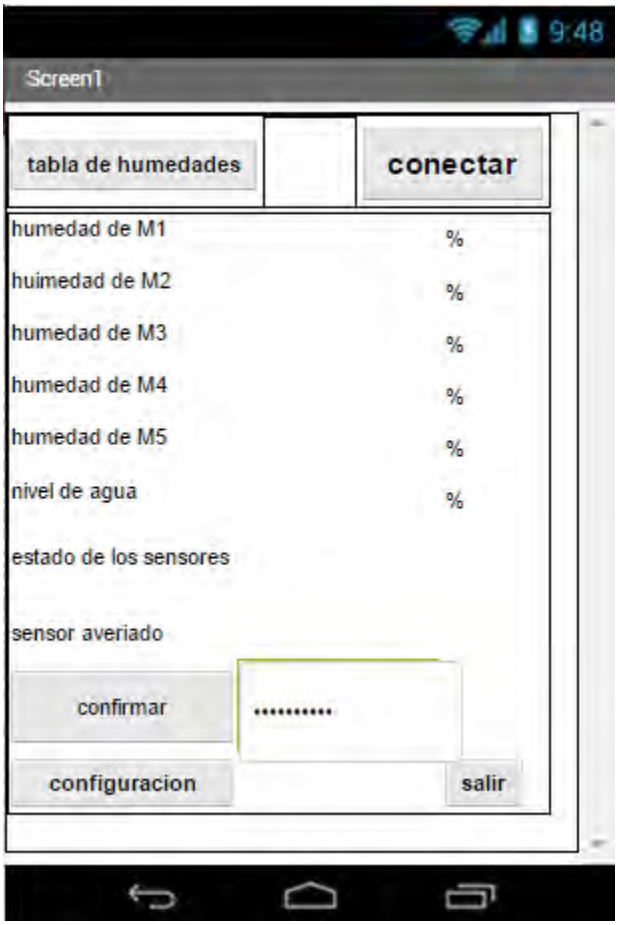

Ilustración 3.25 pantalla principal

3.3.2 Construcción de la pantalla de configuración

La construcción de esta pantalla se basa en que el usuario pueda reconfigurar el nivel de humedad de las cinco plantas aromáticas de la manera más sencilla y cómoda sin que se produzcan errores al introducir valores.

Se utilizó, al igual que en la pantalla anterior, un arreglo de tabla sobre el cual se incorporaron etiquetas (*Labe*l) para denotar el propósito de la pantalla y la humedad de la planta que se está modificando (%PLANTAi). Para modificar el porcentaje de humedad se utilizaron cinco deslizadores (*Sliders*) cuyo valor puede variar de 0 a 100, se agregó un botón de nombre "*enviar*" en la parte inferior derecha para enviar los nuevos datos y retornar a la pantalla principal, finalmente la pantalla de configuración se muestra en la siguiente imagen (ilustración 3.26).

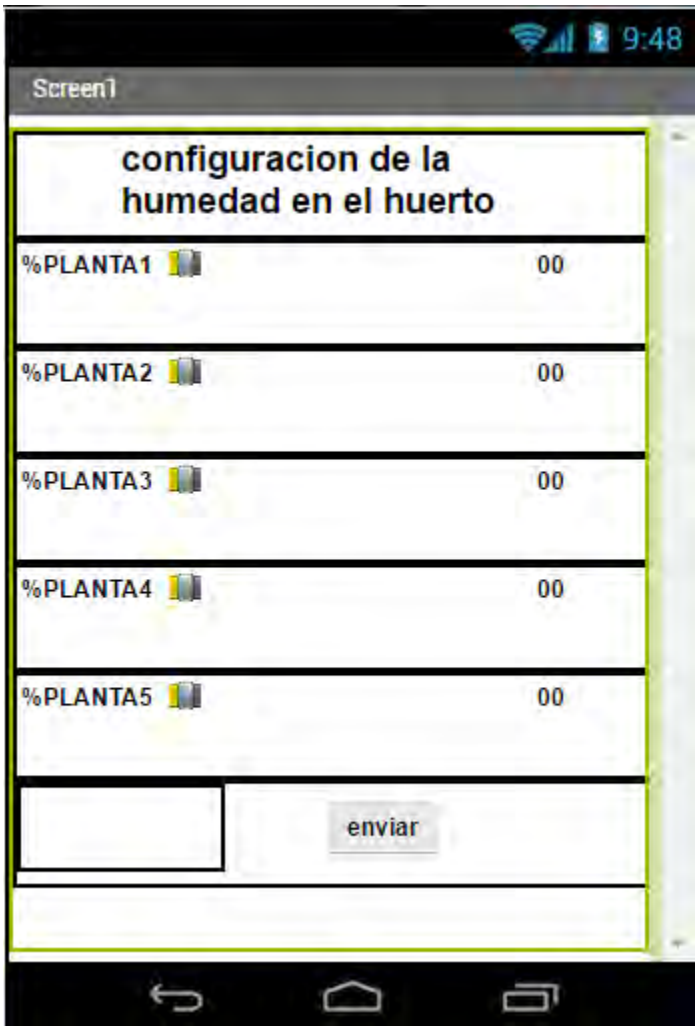

Ilustración 3.26 pantalla de configuración

3.3.3 Construcción de la pantalla de tabla de humedades

La función de esta pantalla es dar información acerca de la humedad óptima de las plantas aromáticas más comunes al usuario, información que será utilizada en la pantalla de configuración.

Para la construcción de esta pantalla se utilizó como base un arreglo de tabla, los elementos que la componen son: una imagen en la cual se muestra una tabla con las humedades optimas de 15 plantas aromáticas y un botón de regreso, el cual permite retornar a la pantalla principal de la interfaz, obteniendo como resultado final la siguiente imagen (ilustración 3.27).

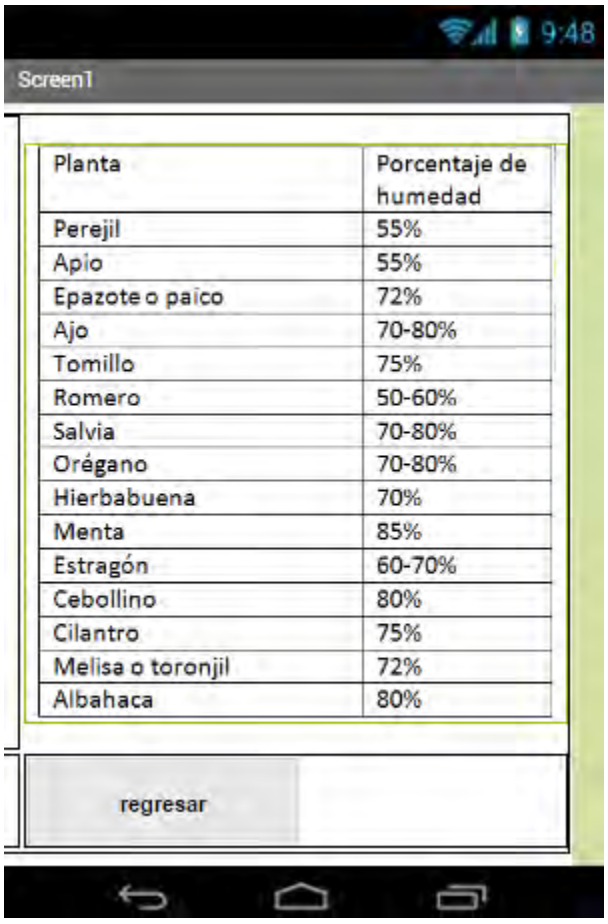

Ilustración 3.27 pantalla de tabla de humedades

3.3.4 Elementos no visibles

Dentro de la aplicación se incluyeron dos elementos no visibles: el primero *BluetoothClient,* este elemento habilita el uso del *bluetooth* para transmitir y recibir información. El segundo elemento es *Clock,* él cual es un reloj interno que cuenta en milisegundos y se desborda al llegar a un límite (por defecto este límite es de 1 segundo).

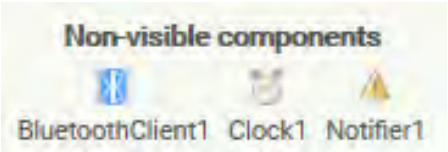

Ilustración 3.28 elementos no visibles

3.3.5 Programación de la interfaz

La programación de la interfaz se realiza con bloques en este entorno de desarrollo en línea llamado APP inventor, la programación se estructuro con el criterio de las tareas que debía cumplir la interfaz.

La primera de las tareas a cumplir era la conexión *bluetooth*, para ello, se usó un *ListPicker*, y se programaron dos bloques, el primero con la condición de que antes de que sea presionado cargue una lista con los dispositivos *bluetooth* sincronizados al Smartphone (ilustración 3.28).

Ilustración 3.29 fragmento del código "bloque de carga de dispositivos sincronizados"

El segundo bloque (ilustración 3.29) se encarga de mostrar la lista de los dispositivos *bluetooth*, previamente sincronizados con el Smartphone, al seleccionar el elemento deseado, se realiza la conexión con el dispositivo *bluetooth* de la lista, en caso de ser exitosa, el texto dentro *ListPicker* (conectar) cambia a color verde.

Ilustración 3.30 fragmento del código "bloque de conexión bluetooth"

En el siguiente fragmento de código (ilustración 3.30) se programan los botones de la aplicación comenzando con el botón de nombre "b_tabla", el cual al ser presionado vuelve visible el arreglo de pantalla de tabla y vuelve invisible el arreglo de tabla correspondiente a la pantalla principal.

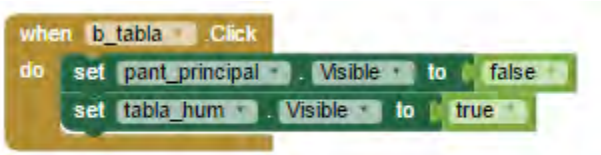

Ilustración 3.31 fragmento del código " botón de acceso a la pantalla de tabla de humedades"

El segundo botón (ilustración 3.31) de nombre “b_regreso” ubicado en la pantalla de tablas de humedades realiza la acción contraria del botón “b_tabla” al volver a ocultar el arreglo perteneciente a la tabla de humedades y volver visible el arreglo de la pantalla principal.

Ilustración 3.32fragmento del código "botón de regreso"

El tercer botón (ilustración 3.32) de nombre “b_confi” se programó para que al ser presionado vuelva visible el botón de “b_autor” y el campo de contraseña *(password*) con el que podemos interactuar.

Ilustración 3.33fragmento del código "botón de configuración"

El cuarto botón (ilustración 3.33) de nombre “b_autor” nos permite acceder a la pantalla de configuración, solo si se cumple la condición de que el texto colocado en el campo de contraseña sea igual a “12345”, mostrando una notificación al usuario que está accediendo a la pantalla de configuración.

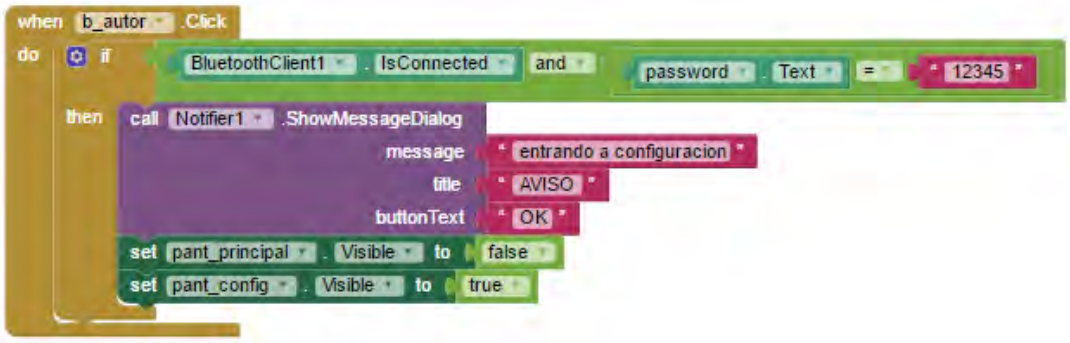

Ilustración 3.34 fragmento del código "botón confirmar"

El último botón ubicado en la pantalla principal es el botón (ilustración 3.34) de nombre "b_out", el cual se programa con una sola función la cual es cerrar la aplicación.

Ilustración 3.35 fragmento del código "botón salir"

Los elementos programados dentro de la pantalla de configuración incluyen 5 *Sliders,* los cuales utilizan los mismos bloques en su programación (ilustración 3.35), variando únicamente el número del slider perteneciente, dicho bloque cumple con la funcion de proporcionar un valor numérico a la etiqueta (*Label*) equivalente a la posición del slider.

Ilustración 3.36 fragmento del código "slider 1"

Finalmente, el botón con nombre (ilustración 3.36) "b_enviar" se programó para que al ser presionado muestre una notificación que dirá "los datos serán enviados". Continuando con el algoritmo, se concatenan todos los datos en una sola línea de caracteres separándolos con comas (,). Finalizando el algoritmo regresa a la pantalla principal limpiando el campo de contraseña y ocultándolo junto con el botón "b_autor".

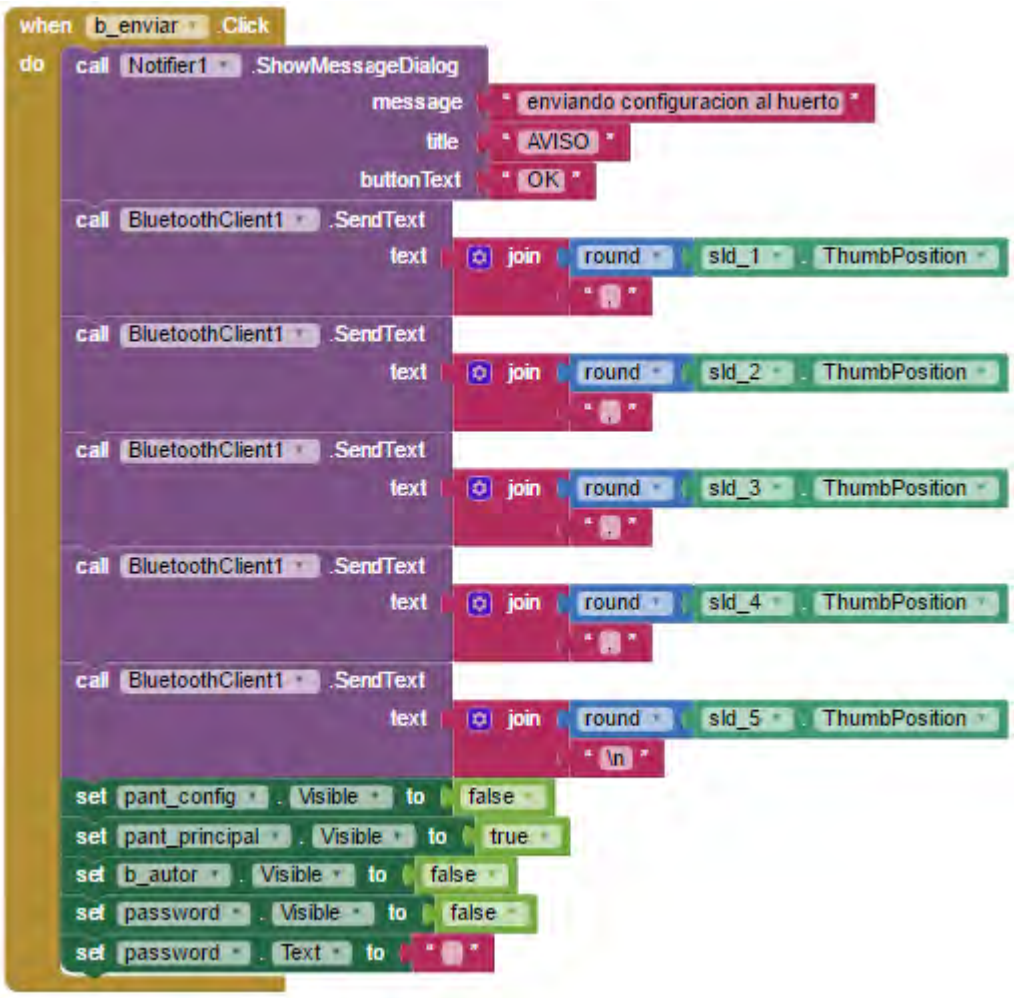

Ilustración 3.37 fragmento del código "botón enviar"

Este último fragmento de código (ilustración 3.37) nos muestra la recepción de la información proveniente del microcontrolador, para ello, se inicializan dos variables globales, con dos condiciones: la primera, debe haber una conexión *bluetooth* y la segunda, la información disponible en dicha conexión sea mayor a cero. Una vez cumplidas estas condiciones, se toma la línea de caracteres recibida y se almacena en la variable global "datos", posteriormente se separa la información en una lista con ayuda de las comas que incluye la línea de caracteres, estos datos separados se guardan en la variable global de nombre lista, finalmente se muestran los datos guardados en la variable lista seleccionando la posición que estos ocupan, este ciclo se repetirá cada vez que se desborde el reloj interno (1 segundo).

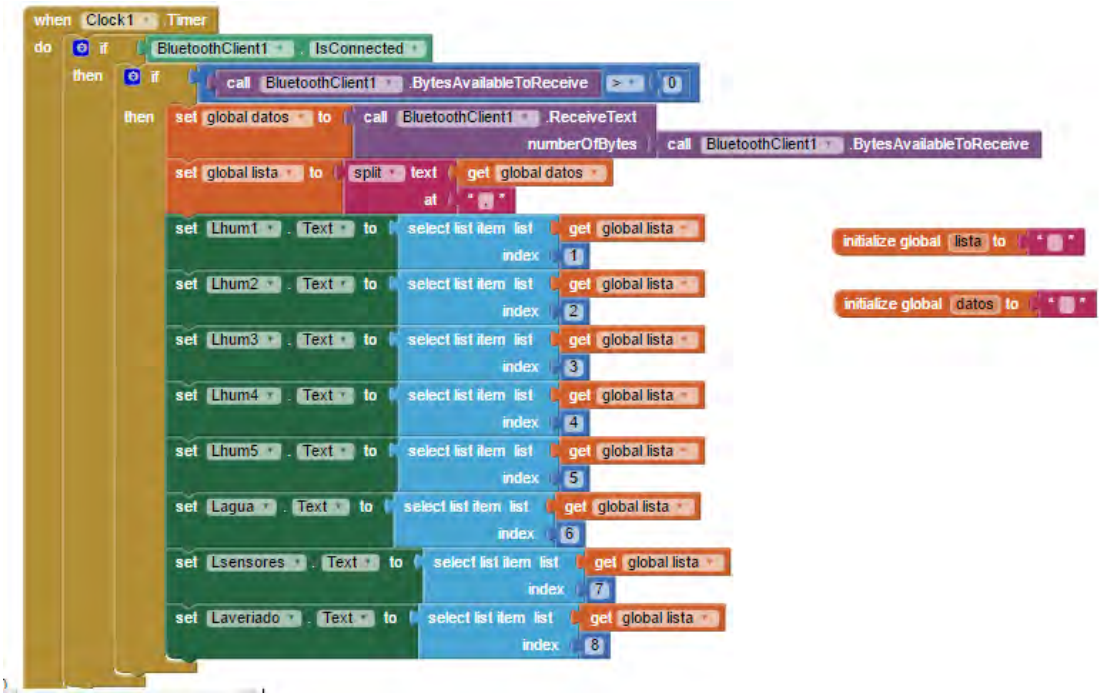

Ilustración 3.38 fragmento del código "bloque de recepción de información"

3.5 Diagramas esquemáticos

A continuación, se mostrarán los diagramas de conexiones realizados en la propuesta, los cuales se dividieron en 3 partes.

3.5.1 Sensores de humedad

En el primer diagrama se muestran las conexiones de los sensores de humedad a la placa Arduino, estos sensores cuentan con cuatro terminales, pero solo utilice tres, dos pines de polarización (5V y tierra) y el tercer pin se conectó a un canal analógico AN comenzando desde A1 hasta A5) de la placa Arduino, ya que este enviará las mediciones de humedad al microcontrolador.

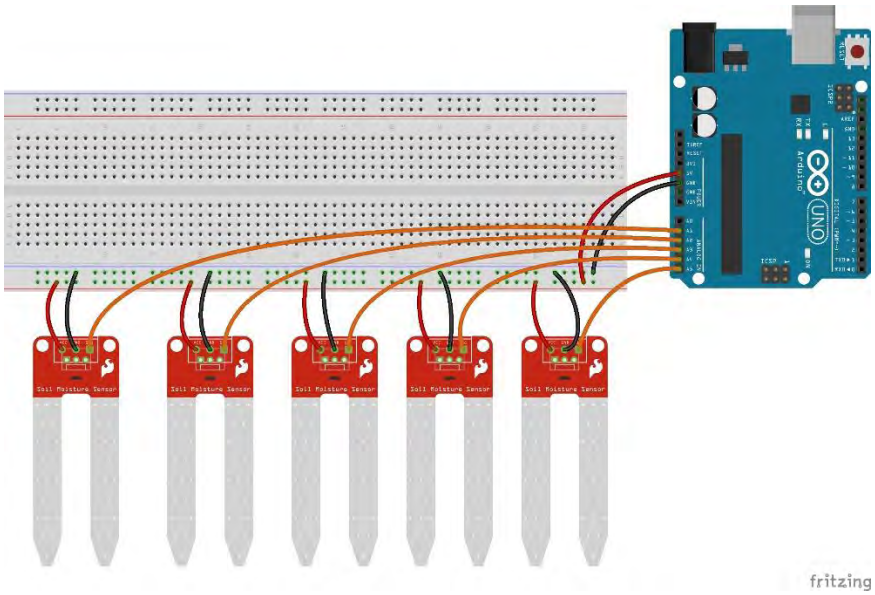

Ilustración 3.39 diagrama de conexión de sensores de humedad

3.5.2 Sensor ultrasónico, módulo bluetooth y alertas

En el segundo diagrama (ilustración 3.39) se muestran las conexiones del resto de sensores, comenzando con el sensor ultrasónico, el cual utiliza cuatro pines: dos de polarización denotados con cables de color rojo y negro y los otros dos *trigger* y echo se conectaron a la placa Arduino en los pines 8 y 9, respectivamente.

Continuando con el módulo *bluetooth*, al igual que el ultrasónico utilizará cuatro pines: dos de polarización, uno de transmisión (TX) y uno recepción (RX) de información, los cuales se conectarán a la placa Arduino en los pines 0 (RX) y 1 (TX) que cumplen con la misma función (Nota: se deben conectar TX-RX y viceversa).

Finalmente, la conexión de las alertas luminosas, las cuales se conectan en los pines 6 y 7. Los leds llevan una resistencia de 220 Ω (para limitar el paso de la corriente) y, por último, se conectan a tierra para cerrar el circuito.

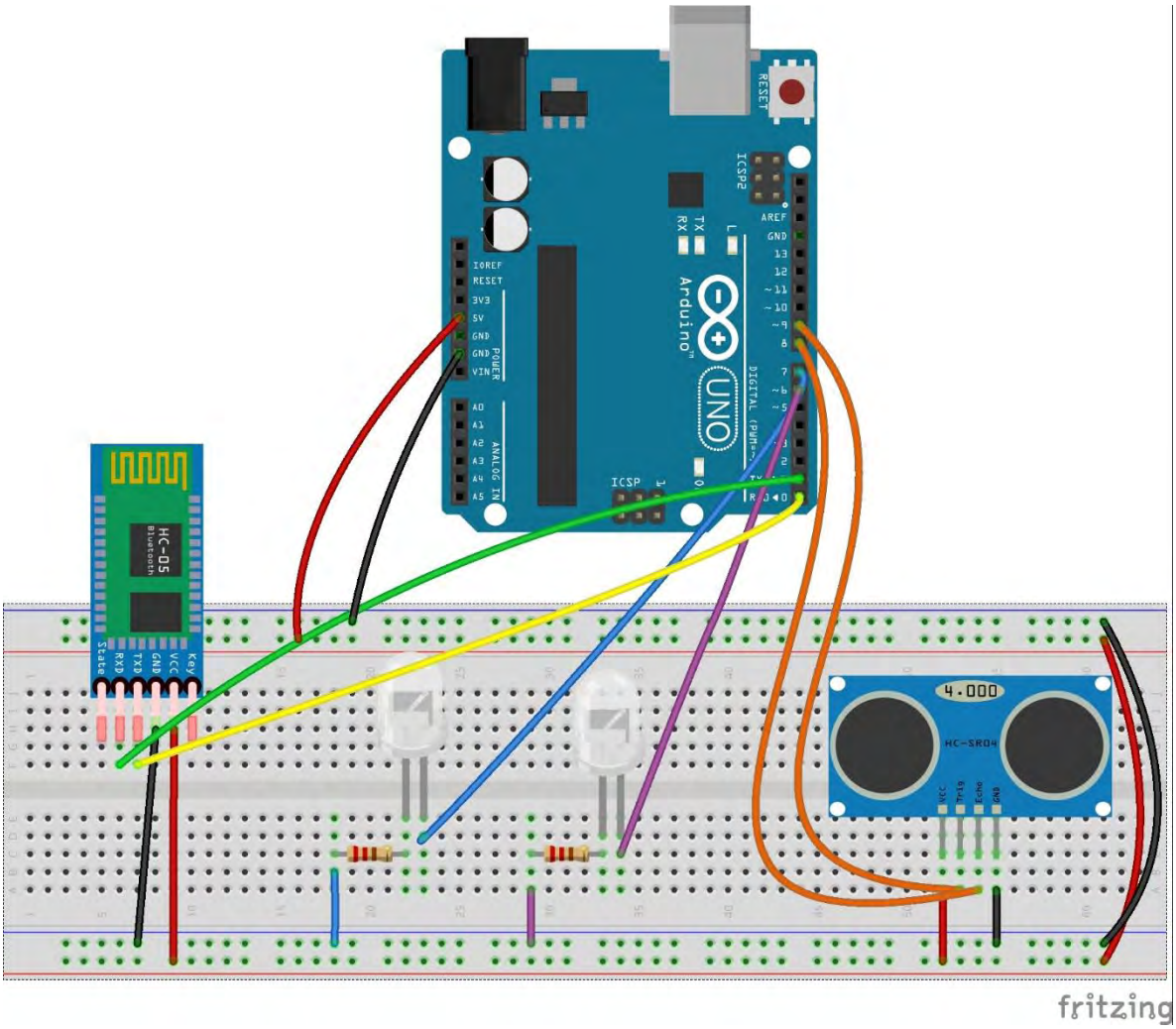

Ilustración 3.40 diagrama de conexión de bluetooth, ultrasónico y leds

3.5.3 Servomotor y bomba de agua

Este tercer y último diagrama (ilustración 3.40) muestra la conexión de los actuadores, bomba de agua y servomotor, cuentan con un circuito de control; la función de este circuito de control es interrumpir la alimentación de cualquier actuador. Esto se lleva a cabo por medio de una señal de control proveniente de la placa Arduino de los pines 2 y 3, los cuales se conectan a una resistencia en la base de un transistor tipo NPN, el transistor permite el paso de corriente y polariza al relevador, el cual cierra o abre el circuito de alimentación del actuador; además se incluye un diodo rectificador para proteger la placa, finalmente el servomotor requiere de una señal del tipo PWM de control proveniente del pin 5 de la placa Arduino.

La polarización de los actuadores se debe realizar con una fuente distinta, ya que la placa Arduino no puede administrar la corriente necesaria para el correcto funcionamiento de los actuadores, sin olvidar conmutar la tierra de la fuente con la que proporciona la placa.

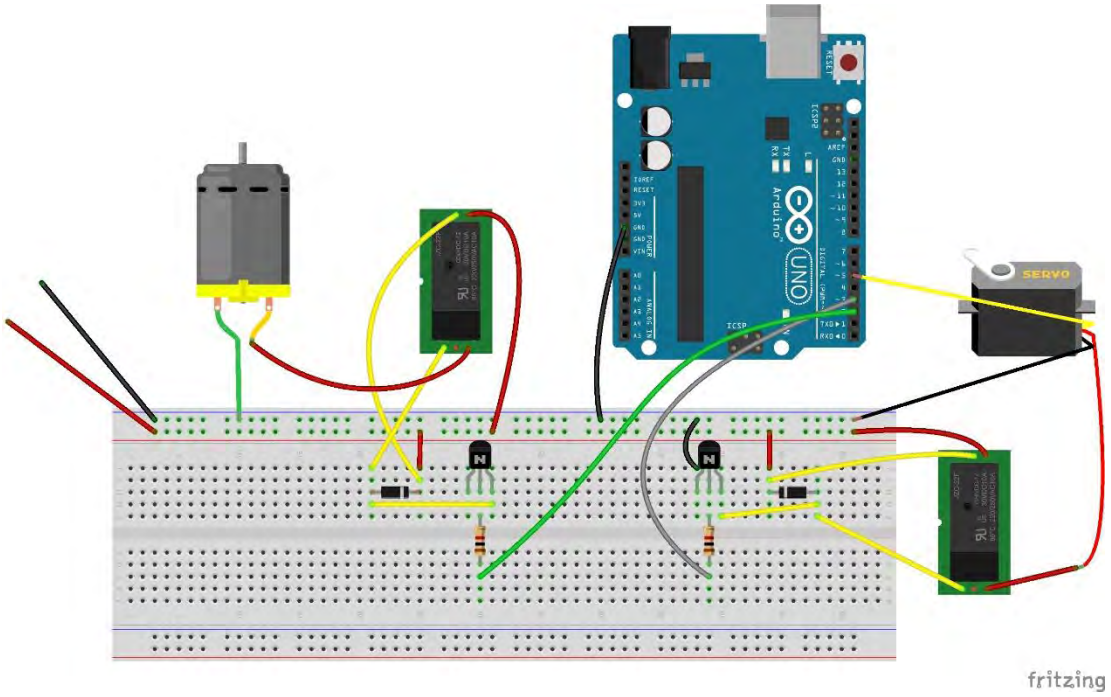

Ilustración 3.41 diagrama de conexión de servomotor y bomba de agua

Capítulo 4 Pruebas y resultados

4.1 Pruebas y resultados del huerto

Al finalizar la construcción del huerto fue sometido a diferentes pruebas, comenzando con las macetas y el sistema de desagüe, para ello se hicieron pruebas vertiendo agua en las macetas, para verificar que no tuvieran fugas. Las macetas que pasaron la prueba se montaron en los anillos del huerto junto con el sistema de desagüe; las macetas que presentaron fugas fueron sometidas al sellado de sus uniones y de nuevo se puso a prueba el sellado, vertiendo agua, hasta que ninguna presentara fugas. Posteriormente se colocó una malla mosquitera para que solo deje pasar agua al sistema de desagüe y la tierra se quede en la maceta.

La segunda prueba realizada al huerto (ilustración 4.1) fue de resistencia, en la cual la estructura del huerto fue sometida al peso que debería soportar, para ello se montaron 5 plantas aromáticas de tamaño considerable y un depósito de agua lleno en la estructura durante una semana, con el propósito de verificar que la estructura no se deforme; la prueba fue exitosa ya que la estructura no sufrió deformaciones por el peso, comprobando que puede soportar el uso del prototipo.

Ilustración 4.1 prueba de peso al huerto

La tercera prueba realizada (ilustración 4.2), fue al sistema de distribución de riego, en específico, los canales de riego, los cuales fueron puestos a prueba vertiendo agua en ellos para buscar posibles fugas. La prueba mostró que ningún canal de riego tenía fugas, pero algunos de ellos mostraron un ligero estancamiento de agua, el cual se redujo con la implementación de conexiones de codo, pero este aún se presenta en los canales de riego.

Ilustración 4.2 prueba a los canales de riego

4.2 Pruebas y resultados de sensores

Los sensores puestos a prueba fueron: los sensores de humedad para la tierra y el sensor ultrasónico, los cuales fueron sometidos a varias pruebas para comprobar su funcionamiento y en el prototipo.

4.2.1 Pruebas del sensor ultrasónico

El sensor ultrasónico fue puesto a prueba (ilustración 4.3) con ayuda de una regla y un objeto plano, para comprobar que la distancia que obtiene el sensor y la medida con la regla fueran cercanas; el sensor mostró buenos resultados, pero tenía dificultades con objetos muy cercanos por lo que para su uso se debía tener una distancia mínima de 5cm y no superar la distancia máxima de este (400 cm).

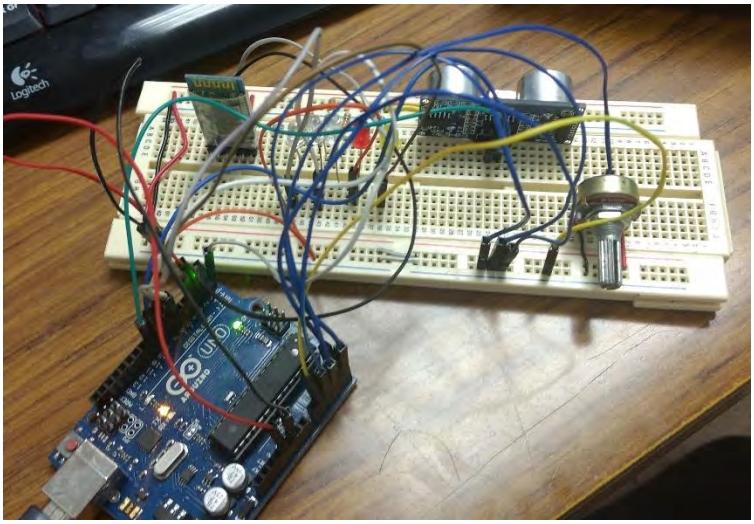

Ilustración 4.3 prueba del sensor ultrasónico

El sensor fue sometido a una segunda prueba con el depósito de agua, en la cual, se tomaron mediciones del depósito con y sin agua. Los resultados obtenidos por las mediciones en el depósito, arrojaron la información necesaria para establecer un nivel mínimo y máximo del depósito de agua.

4.2.2 Pruebas a los sensores de humedad

Los sensores de humedad (ilustración 4.4) fueron puestos a prueba con tierra seca y agua, con el fin de poder obtener los límites de medición de los sensores, las pruebas demostraron que el sensor en ausencia de agua envía una señal analógica de 255, mientras que en presencia de humedad máxima este arroja un valor de 0; además se probó el canal digital, el cual arroja una señal 0 lógico (0 V) hasta superar un cierto nivel de humedad este se puede calibrar en el sensor, al superar el nivel, el sensor arrojara un 1 lógico (5V) y enciende el led del sensor;

se decidió solo usar el canal analógico del sensor (que proporciona la suficiente información) debido a que el canal digital es difícil de calibrar.

Ilustración 4.4 prueba realizada a sensor de humedad

La segunda prueba al sensor fue hecha en las macetas, al variar la distancia a la que el sensor se colocaría del canal de riego, esta prueba mostró que el sensor necesita colocarse a una distancia de 5 cm del centro de la maceta para que mostrara una medición promedio, ya que mientras más se aleja o se acerca el sensor del centro, la medición varia fuertemente debido a la localización el canal de riego.

4.3 Pruebas y resultados de actuadores

Los actuadores puestos a prueba fueron: un servomotor y algunos modelos de bomba de agua, a los cuales se les realizaron pruebas de funcionamiento y pruebas en el prototipo.

4.3.1 Pruebas realizadas al servomotor

La prueba de funcionamiento (ilustración 4.5) realizada en el servomotor consistió en medir y registrar los grados a los que se movía el servomotor y tomar mediciones de la señal de control, el servomotor se movió libremente 180° grados, presento dificultad con movimientos

ligeros, de menos de 10 grados, las mediciones realizadas a la señal de control mostraron cumplir con las condiciones de la señal PWM necesarias para el control del servomotor.

La segunda prueba a la que fue sometido el servomotor fue realizada en el huerto al calibrar los grados que debía moverse para llegar a los cinco canales de riego, la prueba arrojó los grados a los que se posiciona en los canales de riego, pero también mostro que el servomotor tenía dificultades para moverse desde ciertos ángulos, este problema se solucionó al colocar un punto de arranque inicial y agregar movimientos extras para su correcta colocación.

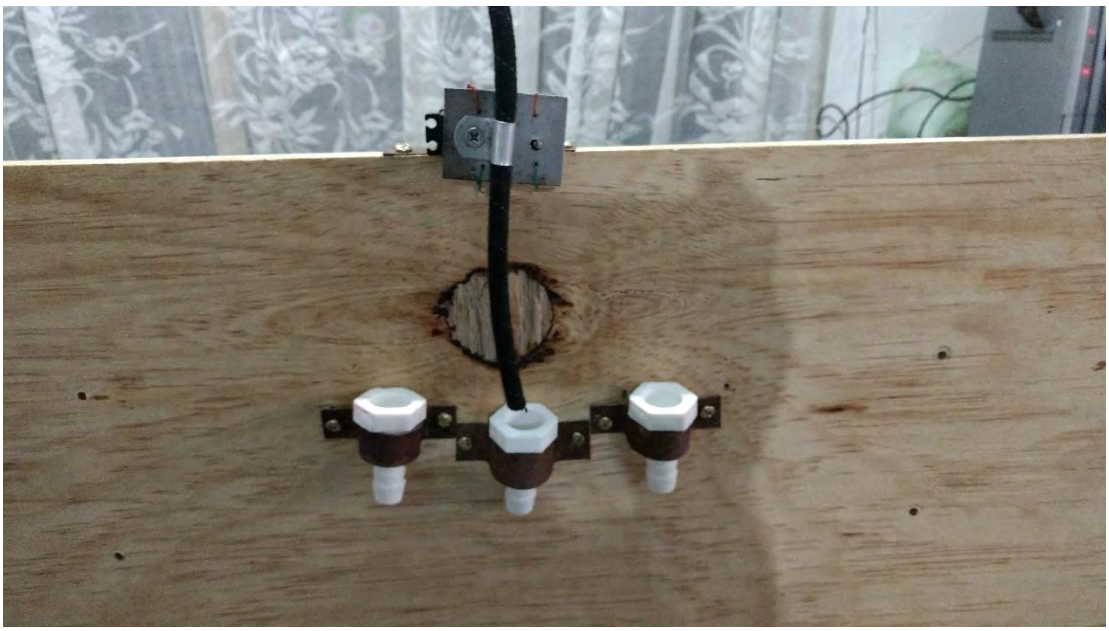

Ilustración 4.5 prueba realizada al servomotor

4.3.2 Pruebas realizadas a bombas de agua

Las bombas de agua fueron puestas a prueba, la primera prueba (ilustración 4.6) consistió en obtener la altura máxima de elevación y el caudal de agua que estas tenían. Los resultados mostraron que mientras se le exigía una mayor elevación del agua el caudal disminuía, mostrando una elevación máxima de un metro con un caudal consistente para el riego, por lo que el depósito de agua se ubicó en la parte trasera-media del huerto, para así obtener las condiciones óptimas para la bomba de agua.

La bomba de inmersión de 3.5 V-12 V (ilustración 2.10), fue la que mostro mejor resultados ya que cumplía con la elevación de agua a un metro con un caudal ligero que cumplía con el

riego sin poner en peligro los componentes electrónicos, a diferencia de las otras bombas que solo cumplían con una de las condiciones ya mencionadas.

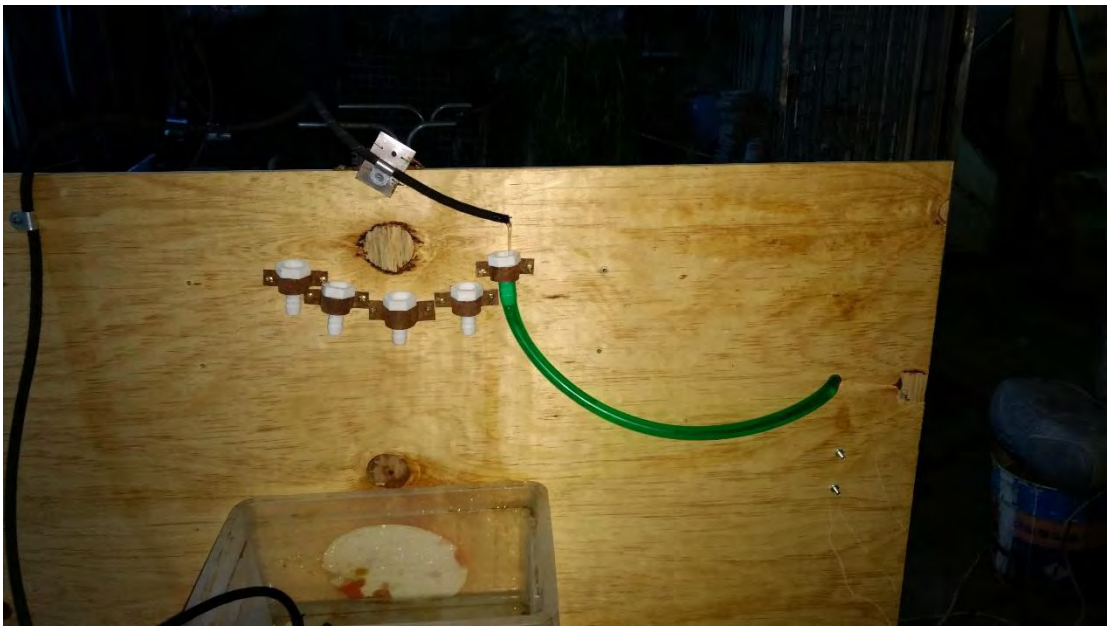

Ilustración 4.6 prueba realizada a la bomba de agua

4.4 Pruebas a la interfaz Android

La interfaz fue puesta a prueba en diferentes tareas, cada una fue realizada por separado, para finalmente probar toda la interfaz en el prototipo.

4.4.1 Prueba de conexión

La primera prueba realizada a la interfaz (ilustración 4.7) fue la de conexión inalámbrica, en la que se debe vincular un dispositivo *bluetooth* seleccionado con el sistema de control, y se debe mantener la conexión hasta que el usuario cierre la aplicación.

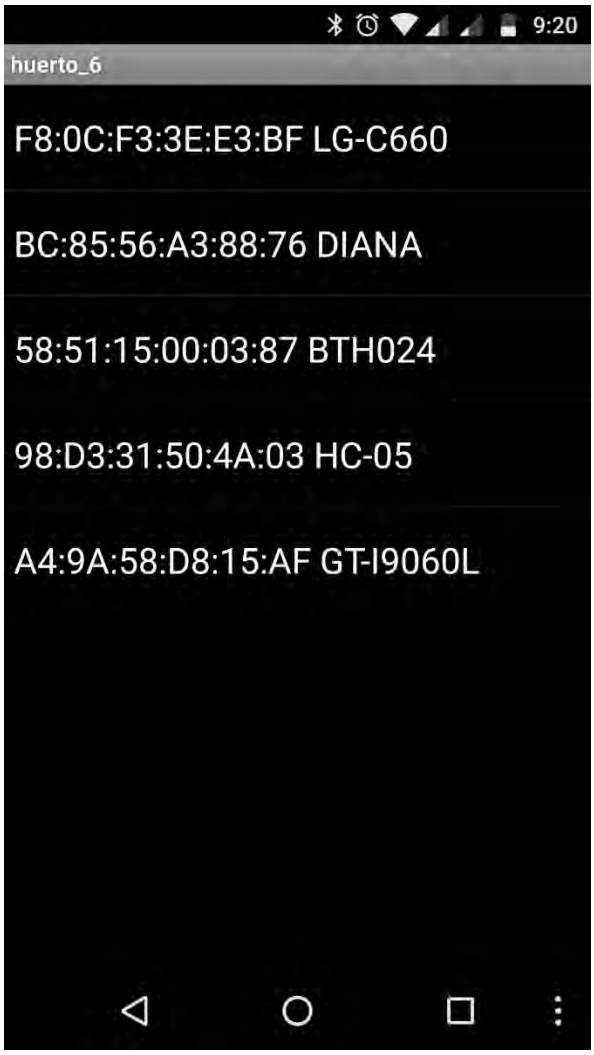

Ilustración 4.7 prueba de conexión (dispositivos vinculados)

Los resultados del primer experimento mostraron que la interfaz lograba establecer una conexión, pero esta se perdía al realizar un cambio de pantalla dentro de la interfaz y cualquier acción que requiriera uso de la conexión mostraba un mensaje de error (ilustración 4.8).

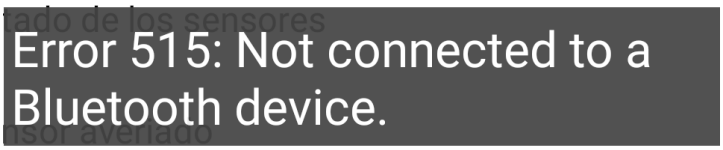

Ilustración 4.8 mensaje de error de conexión

Debido a que se perdía la conexión se modificó el código de la interfaz (ilustración 4.9), el cual guardaba la dirección MAC del dispositivo *bluetooth* y al cambiar de pantalla la comunicación se restablecía con el dispositivo; las pruebas realizadas a esta segunda interfaz mostraron que la mayoría de las veces se recuperaba la conexión al realizar un cambio de pantalla, debido a que la conexión no se mantenía siempre, se decidió modificar nuevamente el código de la interfaz.

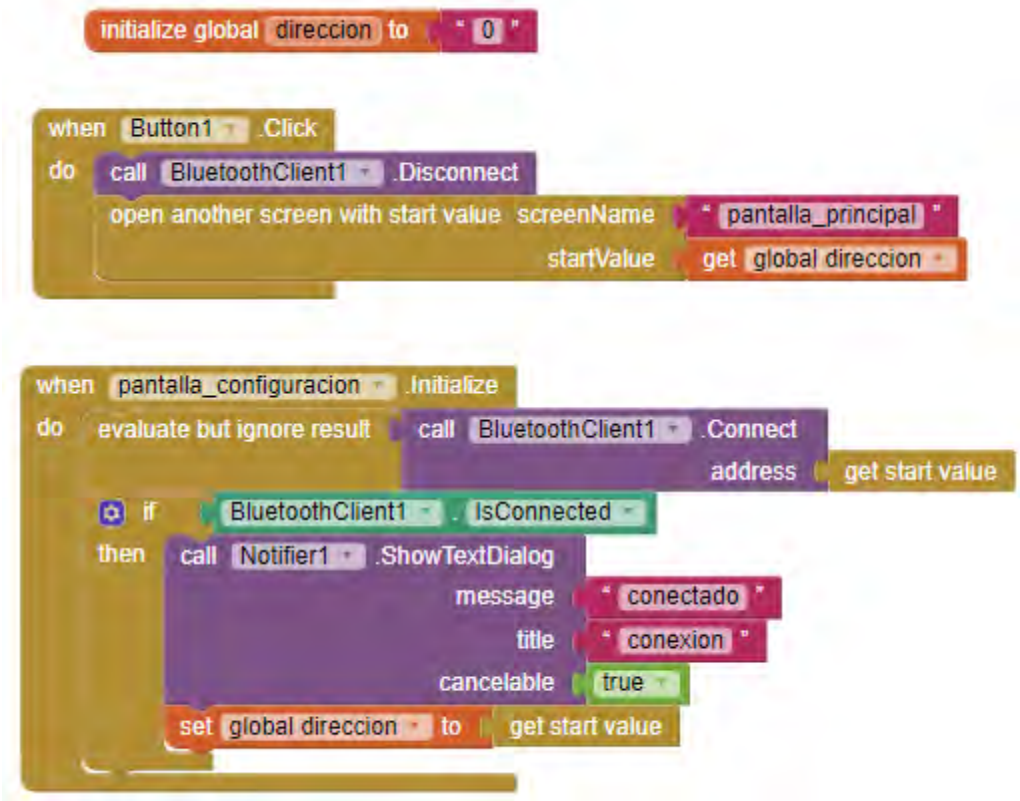

Ilustración 4.9 fragmento de programa donde se guarda y carga la dirección MAC del dispositivo bluetooth

Finalmente se trabajó toda la interfaz en una pantalla y se simuló el cambio con ayuda de arreglos de tabla, esta última interfaz mostró mantener una conexión inalámbrica estable dentro del rango de alcance del dispositivo *bluetooth*.

4.4.2 Prueba de pantalla principal

La pantalla principal (ilustración 4.10) de la aplicación fue sometida a diferentes pruebas, comenzando con la comprobación de las dimensiones de los elementos que la conforman, para esto se visualizó la pantalla en un Smartphone, asegurando que los elementos no excedieran el tamaño de la pantalla, no se encimarán y finalmente que el texto que debía mostrar estuviera completo y fuera legible.

Ilustración 4.10 prueba realizada de la pantalla principal en un Smartphone

Los resultados de esta prueba mostraron que los elementos no excedían el tamaño de la pantalla y que tampoco se encimaban, pero algunos cuadros de texto mostraban la información cortada debido a las dimensiones limitadas del cuadro, esto se solucionó al incrementar el tamaño de estos cuadros de texto sin comprometer los demás elementos.

La siguiente prueba realizada a la pantalla principal (ilustración 4.11) consistió en comprobar la funcionalidad de todos sus elementos. La prueba comenzó realizando la conexión inalámbrica con ayuda del elemento *ListPicker*, al obtener una conexión exitosa se transmitieron datos a la aplicación simulando las lecturas de los sensores con la finalidad de observar si estos llegaban en el orden correcto a la aplicación y que no hubiera perdida de información. Los resultados arrojaron que la pantalla principal trabajaba sin problema alguno recibía correctamente la información y mostró las lecturas de los sensores.

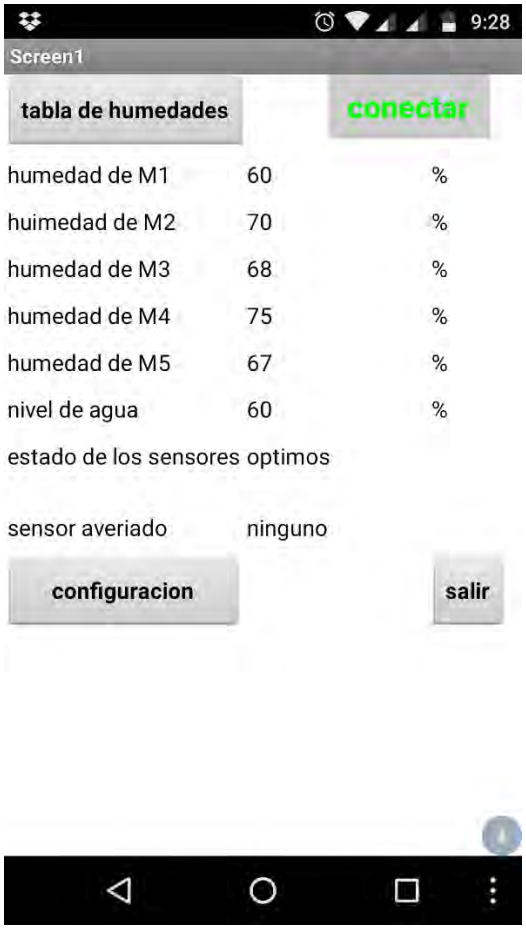

Ilustración 4.11 prueba realizada a la interfaz recibiendo los datos de los sensores

La última prueba realizada fue aplicada a la transición a la pantalla de configuración (ilustración 4.12), esta prueba requería comprobar, sí el usuario deseaba cambiar de pantalla, se habilitaban dos elementos que concedían el acceso, en uno de los elementos el usuario escribía la contraseña y en el otro verificaba la contraseña, al coincidir la contraseña se realizaba la transición de la pantalla principal a la de configuración, la prueba mostró que

solamente al cumplir estas condiciones la transición de las pantallas se realizaba correctamente, la prueba también arrojo que al regresar a la pantalla principal los elementos que permitían la transición de pantallas seguían activos, esto se solucionó al agregar cuadros extras de programación que los ocultaban hasta que el usuario los requiriera nuevamente.

Ilustración 4.12 prueba de transición de pantalla

4.4.3 Prueba de pantalla de configuración

Para la realización de las pruebas en esta pantalla (ilustración 4.13) fue necesario primero haber concluido con éxito las pruebas anteriores.

La primera prueba realizada a la pantalla de configuración, fue que el tamaño de los elementos no excediera la pantalla y que los elementos de texto fueran legibles, al comprobar que los elementos tenían las dimensiones correctas se prosiguió a comprobar la funcionalidad

de los sliders (barras de desplazamiento), la prueba consistió en comprobar la movilidad, la sensibilidad y los límites de los cinco sliders, los cuales debían de poder marcar del cero al cien según la posición en la que estos se encuentren. Los resultados de la prueba mostraron que los cinco sliders podían marcar los valores dentro del rango.

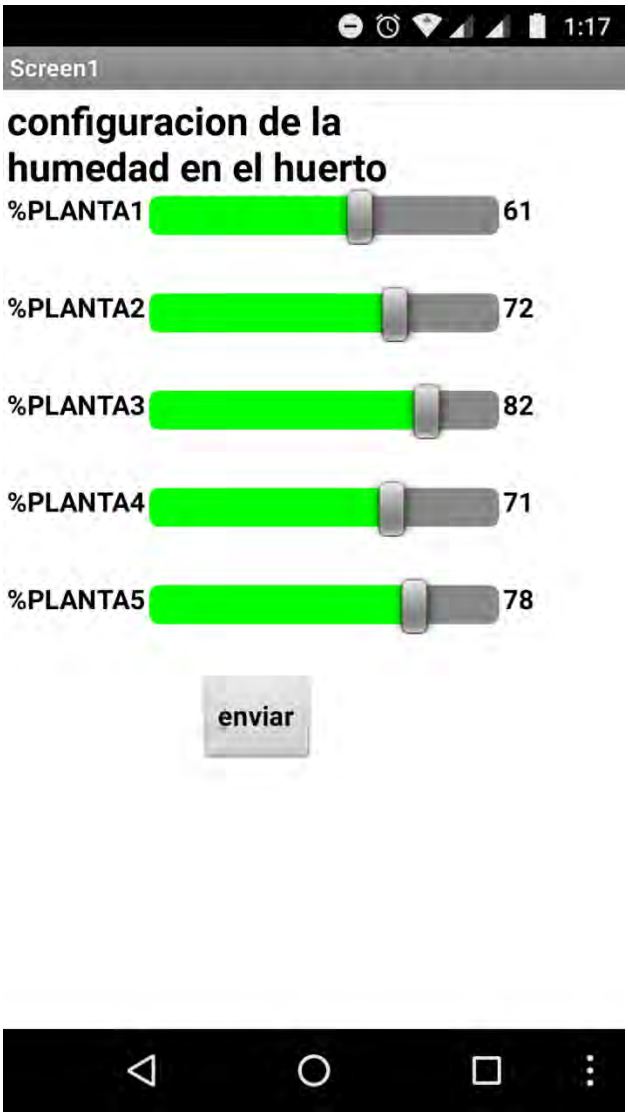

Ilustración 4.13 prueba de la pantalla de configuración

La última prueba realizada a la pantalla fue la transmisión de los nuevos datos al microcontrolador y que después de enviar los nuevos datos, se regresara nuevamente a la pantalla principal. Para su realización se enviaron datos diferentes a los colocados por defecto en el microcontrolador a través de la aplicación, una vez que terminó él envió de nueva información la aplicación regreso a la pantalla principal, en la cual solo se mostraban los

datos de los sensores, se corrigió la falla. Para finalizar la prueba se verificó que el algoritmo trabajara con los nuevos datos transmitidos, esto se logró simulando los sensores de humedad y los actuadores, obteniendo resultados satisfactorios ya que los nuevos datos estaban guardados en el microcontrolador por lo que la prueba de la pantalla de configuración fue exitosa.

4.4.4 Prueba de la pantalla de tabla de humedades

Debido a que los usuarios de la aplicación necesitan conocer los niveles óptimos de humedad de los posibles cultivos se decidió agregar otra pantalla (ilustración 4.14) en la cual se incluyeron los tipos de cultivo y el porcentaje de humedad optimo al que debe estar cada planta.

La primera prueba a la que se sometió fue que se pudiera acceder a esta nueva pantalla a través de la pantalla principal y viceversa, al comprobar el correcto funcionamiento se continuo con el ajuste de tamaño de la tabla; se requería que la tabla fuese lo suficientemente grande y legible pero sin que excediera la pantalla del dispositivo, al ajustar el tamaño de la tabla se concluyen con éxito las pruebas de esta pantalla.

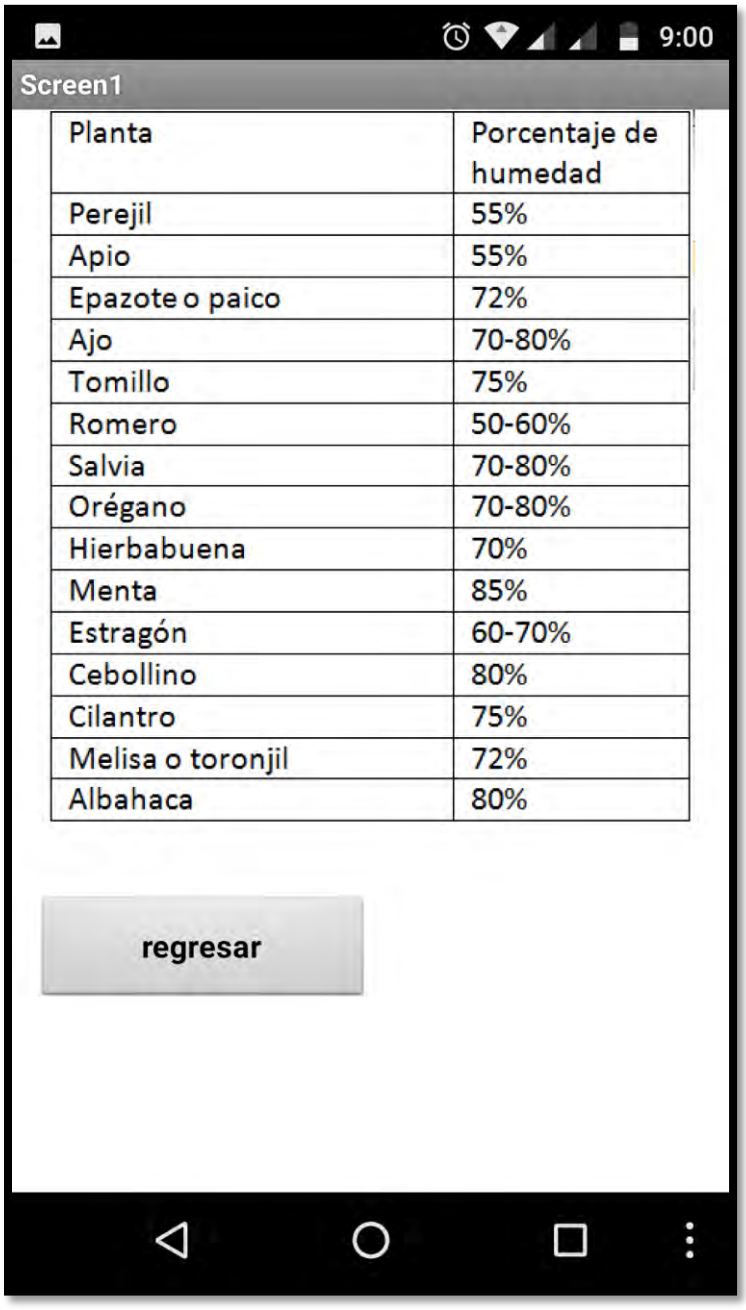

Ilustración 4.14 prueba de pantalla de tabla de humedad

4.5 Pruebas del sistema de riego y drenaje

En esta prueba se comprobó la eficiencia y el correcto funcionamiento de los actuadores integrados en el sistema de riego, para lo cual, se desarrollaron dos prototipos, los cuales, al ser sometidos a pruebas, se eligió el más óptimo.

4.5.1 Prueba del primer prototipo de sistema de riego

El primer prototipo (ilustración 4.15) fue sometido a varias pruebas, comenzando con la de movilidad y precisión del cambio de canales de riego, en donde mostró que aún con el uso de manguera flexible, le costaba al servomotor llegar a los canales de los extremos, también la prueba mostró que no contaba con la precisión suficiente ya que era muy difícil calibrar; la siguiente prueba fue con el uso de la bomba, en la cual se debía comprobar que el agua debía caer en el respectivo canal de riego sin sufrir derrames, el prototipo mostró que los canales de riego colocados en los extremos sufrían pequeños derrames y finalmente se observó que por la forma de la estructura este presentaba goteos cuando terminaba de funcionar la bomba por lo que se optó por usar un prototipo diferente ya que ponía en peligro al servomotor.

Ilustración 4.15 primer prototipo del sistema de riego

4.5.2 Prueba del segundo prototipo del sistema de riego

Al no obtenerse los resultados esperados del primer prototipo, se decidió modificar y utilizar un segundo prototipo (ilustración 4.16), el cual se basa principalmente en abrazaderas fijadas verticalmente al huerto, que a su vez sujetan los canales de riego, al igual que el primer prototipo se realizó una prueba de movilidad la cual mostró una mejora comparado al

anterior prototipo, ya que este podía llegar a todos los canales de riego sin forzar el servomotor, la siguiente prueba fue la precisión, en el cual mostró tener una precisión aceptable aún con el margen de error del servomotor este sistema de riego no presentaba derrames, solo pequeños problemas de calibración que se arreglaron mediante programación, por lo que los resultados del segundo sistema de riego fueron satisfactorios.

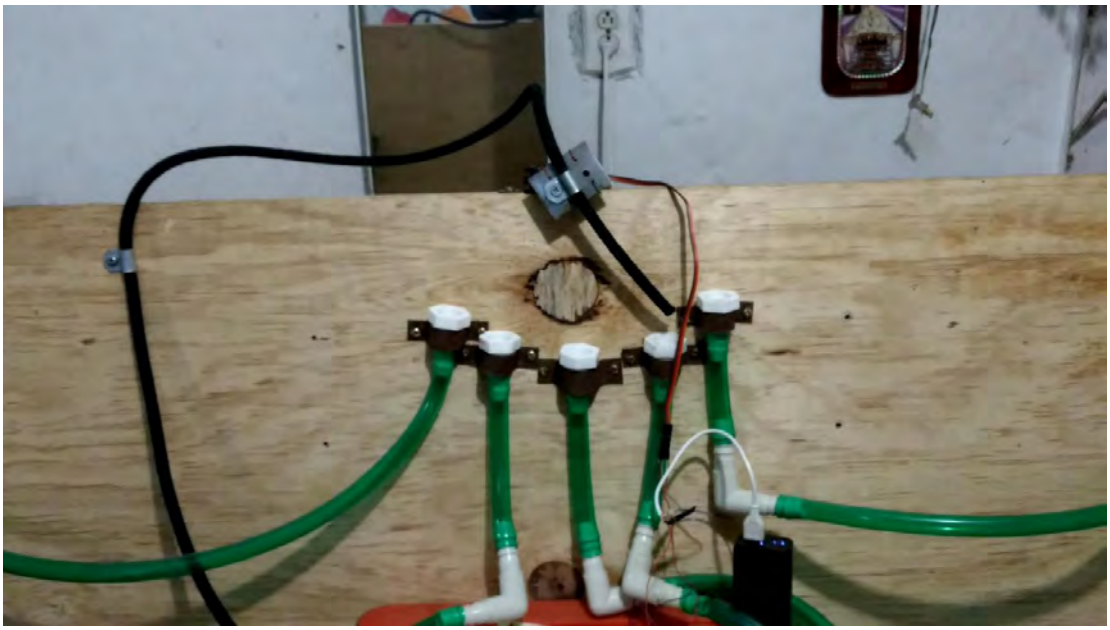

Ilustración 4.16 prueba del segundo prototipo del sistema de riego

4.5.3 Prueba del sistema de drenaje

El sistema de drenaje fue puesto a prueba para prevenir posibles fugas y obstrucciones que puedan comprometer al huerto, para realizar la prueba se vertió agua en los puntos de desagüe de las macetas, al comprobar que no había fugas y el agua no se mantenía estancada se decidió probar el filtro colocado en la boca de los puntos de desagüe para evitar el paso de tierra y que este obstruya los canales, el filtro constó de malla mosquitera y piedra tezontle sobre la boca del canal de desagüe; para la realización de la prueba se colocó tierra de macetas sobre los filtros y se vertió agua sobre ellos, los resultados fueron satisfactorios ya que el filtro cumplía su trabajo de evitar el paso de algún sólido y solo dejaba pasar líquido.

4.6 Prueba del prototipo completo

Finalmente se puso a prueba el prototipo completo con 5 plantas aromáticas diferentes a las que originalmente se montaron en el sistema, por lo que se configuró el prototipo a través de la interfaz con los nuevos valores de humedad y se dejó trabajar durante algunas semanas, revisando el prototipo cada tercer día, y una vez a la semana el depósito de agua.

Durante las primeras semanas se observó el estado de las plantas, donde algunas plantas, especialmente la albaca (ilustración 4.17) mostró quemaduras por una exposición prolongada de luz solar, por lo que el prototipo fue movido ligeramente de la ventana para reducir la exposición de luz solar, lo cual tuvo resultados satisfactorios.

Ilustración 4.17 albaca quemada por el sol

Al cambiar la posición del prototipo se observó una mejora en las plantas hasta el punto en que fueran consumibles otra vez, se observó, que algunas plantas, las cuales aun con los cuidados adecuados tienen un ciclo de vida corto, por lo que es inevitable que se sequen y mueran.

Conclusiones

En este trabajo se investigaron las características de los huertos urbanos, obteniendo los diferentes tipos de cultivos que pueden existir, los tipos de huertos que se pueden construir, los materiales que se utilizan para su construcción, además, obtuve información acerca de los cambios positivos que trae consigo el uso de los huertos urbanos, comenzando con el uso personal hasta ciudades completas que han implementado su uso.

Se identificaron y documentaron los diferentes tipos y características de plantas aromáticas, obteniendo información sobre sus cuidados, familias botánicas a las que pertenecen, dimensiones que pueden alcanzar y tiempo de vida. Para finalmente seleccionar las óptimas para la propuesta.

Se diseñó un sistema de distribución de agua para el riego del huerto con el uso de un servomotor, una bomba de agua y un microcontrolador; el microcontrolador permitía el control de los dispositivos permitiendo seleccionar diferentes canales de riego.

Se construyó un huerto urbano del tipo privado doméstico utilizando los materiales económicos como mangueras, conexiones, boquillas, macetas, bases de metal y botellas de plástico, los cuales son de bajo costo; dejando como mayor gasto la placa de madera donde se montó lo antes mencionado, manteniendo los costos lo más bajo posible.

Se realizaron pruebas con los sensores de humedad (yl-69), en las cuales se comprobó su funcionamiento, así como los resultados que arrojaban (tanto digitalmente como analógicamente). La señal digital necesitaba ser calibrada de manera manual para cada uno de los sensores, por tal motivo se decidió trabajar con la señal analógica.

Se definió e implemento un sistema de control basado en un microcontrolador, ubicado en una tarjeta de desarrollo Arduino, en la cual se crearon funciones en base al nivel de humedad registrado en cada uno de los sensores, los cuales permiten activar y desactivar los actuadores para lograr una humedad óptima en cada uno de ellos.

Se diseñó una aplicación para Smartphone con sistema operativo Android, la cual funciona como interfaz del sistema, permitiendo al usuario visualizar los niveles de humedad del huerto, modificar los valores óptimos de humedad y alertar al usuario de algún sensor dañado;

todo esto se realiza comunicando la interfaz y el sistema mediante *bluetooth*, la cual envía y recibe paquetes de datos de manera inalámbrica por medio de ondas electromagnéticas.

Se implementó una pequeña etapa de potencia con ayuda de un transistor tipo NPN que eleva la corriente de la señal de salida de la tarjeta de desarrollo para activar un relevador y este activa a los actuadores.

Debido a lo anterior, se puede concluir que el proyecto cumple con el objetivo general establecido, al construir e implementar un huerto urbano el cual regula de manera automática la humedad de las plantas aromáticas, permitiendo su óptimo crecimiento, esto ayudado de una interfaz, la cual permite al usuario modificar la humedad optima dependiendo de la especie plantada.

Trabajo a futuro

Como trabajo a futuro se propone reducir las dimensiones del huerto de 1.25 m x 1.25 m x 15 mm a 0.9 m x 0.9 m x 15 mm, así como integrar ruedas para facilitar su traslado al encontrarse armado, además se planea colocar un filtro en el desagüe de agua y un segundo depósito para su reutilización.

Desarrollar modelos 3D de las macetas modificadas en una sola pieza, las cuales aumentarían su resistencia y durabilidad permitiendo una mayor maleabilidad al implementarse; finalmente se adecuara el prototipo a exteriores, lo cual expandirá el repertorio de plantas aromáticas.

Bibliografía

acciona. (20 de enero de 2018). *sostenibilidad.com*. Obtenido de http://www.sostenibilidad.com/huertos-urbanos

Aguaysol, N., Gonzales, V., Robles Téran, L., Lobo Zavalía, R., & Ploper, L. D. (diciembre de 2017). *Detección de Sclerotinia sclerotiorum en cultivos de chía (Salvia hispanica) en Tucumán durante la campaña 2014.* Obtenido de http://hdl.handle.net/11336/15009

Aguilar Murillo, X., Gonzales Rosas , G., Valle Meza , G., & Murillo Amador, B. (2017). *Guía de cultivo de orégano.* La Paz, Baja California Sur, México: Centro de Investigaciones Biológicas del Noroeste.

Alberto Perez, M., Perez Hidalgo, A., & Perez Berenguer, E. (9 de Noviembre de 2017). *INTRODUCCION A LOS SISTEMAS DE CONTROL Y MODELO MATEMATICO PARA SISTEMAS LINEALES INVARIANTES EN EL TIEMPO.* Obtenido de departamento de electronica y automatica U.N.S.J.: http://dea.unsj.edu.ar/control1/apuntes/unidad1y2.pdf

alonso ruíz, n. (29 de noviembre de 2017). *Efecto de la Aplicación de Composta, Lombricomposta y Biodigestados Líquidos en el Crecimiento, Rendimiento y Calidad de follaje en el Cultivo de Cilantro (Coriandum sativum, L.).* Obtenido de http://repositorio.uaaan.mx:8080/xmlui/handle/123456789/6305

Altec Alta Tecnología de Vanguardia, SA de CV. Monterrey, N.L. Mexico. (20 de mayo de 2017). *alta tecnologia de vanguardia*. Obtenido de altec: http://www.altecdust.com/soporte-tecnico/que-son-las-electrovalvulas

ARDUINO. (11 de Junio de 2017). *arduino introduction.* Obtenido de ARDUINO: https://www.arduino.cc/en/guide/introduction

ARDUINO. (16 de Agosto de 2017). *arduino reference.* Obtenido de Arduino: https://www.arduino.cc/reference/en/

Briseño Ruiz, S. E., Aguilar García, M., & Villegas Ezpinoza, J. A. (2017). *El cultivo de la albahaca.* La Paz, Baja California Sur, México.: Centro de Investigaciones Biológicas del Noroeste.

Cañon Angarita, F. A. (18 de julio de 2017). *Efecto del ambiente de cultivo y la densidad de siembra sobre la productividad de dos materiales de romero (Rosmarinus officinalis L) israelí y crespo, en Cajicá –Colombia.* Obtenido de http://hdl.handle.net/10654/13934

Castro Restrepo, D. (2017). *cultivo y producción de plantas aromáticas y medicinales.* fondo editurial universal catolica de oriente.

Condelmed, S.L. (20 de diciembre de 2017). *el huerto urbano*. Obtenido de https://www.elhuertourbano.net/huerto-urbano/

Cucho M. , Z., Orihuela Q., F., Sánchez P. , R., & Rodríguez p., L. (21 de junio de 2017). *microcontroladores.* Obtenido de UNROBOTICA: http://www.unrobotica.com/manuales/SESION_1_ATMEGA8.pdf

Cun G., R., León F., M., & García H., S. (2017). Respuesta del apio (Apium Graveolens l) y perejil (Petroselinum Crispum, mill), a diferentes coeficientes de cultivo en condiciones de organopónicos. *Revista Ciencias Técnicas Agropecuarias, vol. 16, núm. 3,*, 1-5 Q.

delgago baralt, g. a. (26 de octubre de 2017). *DETERMINACIÓN DE LA LÁMINA DE RIEGO PARA EL CULTIVO DE LA ALBAHACA GENOVESA (Ocimum basilicum "Genovese".) A PARTIR DE LA VARIACIÓN DEL COEFICIENTE MULTIPLICADOR DE LA EVAPORACIÓN.* Obtenido de biblioteca digital: http://bibliotecadigital.univalle.edu.co/handle/10893/9017

desconocido. (25 de Noviembre de 2017). *CCM*. Obtenido de CCM: https://es.ccm.net/contents/69-como-funciona-el-bluetooth

Desconocido. (14 de Agosto de 2017). *Ecu Red conocimiento con todos y para todos*. Obtenido de https://www.ecured.cu/Apiaceae

diputacio de valencia. (7 de julio de 2017). *plantas aromaticas para huertos urbanos*. Obtenido de dival: http://www.dival.es/sites/default/files/medio-ambiente/Estudio3.pdf

Electrocomponentes SA. (7 de Noviembre de 2017). *SASE.* Obtenido de Simposio Argentino de Sistemas Embebidos: http://www.sase.com.ar/2011/files/2010/11/SASE2011-Fuentes_de_alimentacion.pdf

Fanlo, M. M. (22 de Agosto de 2017). *CULTIVO DE PLANTAS AROMÁTICAS, MEDICINALES Y CONDIMENTARIAS EN CATALUÑA.* Obtenido de https://www.researchgate.net/profile/Eva_More/publication/230681268_Cultivo_de_plantas_aromaticas_medicinales_y_condimentarias_en_Cataluna_6_anos_de_campos_de_demostracion/links/02e7e539ff9b6d47ac000000.pdf

Geusseppe González Cárdenas, F. A. (14 de agosto de 2017). Diseño e implementación de una Tarjeta de Desarrollo . cancun, Mexico. Obtenido de Diseño e implementación de una Tarjeta de Desarrollo con: http://www.laccei.org/LACCEI2013-Cancun/RefereedPapers/RP157.pdf

Gonzáles Mendoza, M. E. (2 de Septiembre de 2017). *Efecto del estrés hídrico en hierbabuena (mentha piperita) sobre polifenoles y capacidad antioxidante de infusiones.* Obtenido de http://hdl.handle.net/123456789/791

infojardin. (2 de Agosto de 2017). *infojardin*. Obtenido de http://articulos.infojardin.com/aromaticas/cultivo_riego_abonado_aromaticas.htm

iniciarte meleán, c. k. (27 de octubre de 2017). *efecto de la aplicacion de micorrizas sobre el medio suelo.planta para un cultivo de cebollin en sistemas de barbacoa en el municipio maracaibo.* Obtenido de http://tesis.luz.edu.ve/tde_arquivos/182/TDE-2015-03-04T08:28:31Z-5582/Publico/inciarte_melean_carmen_karile.pdf

LIM. (1 de mayo de 2018). *laboratorio de intervencion*. Obtenido de https://laboratoriodeintervencion.wordpress.com/2013/01/05/el-nucleo-duro-de-las-ciudades-04-los-huertos-urbanos/

MXCITY. (12 de Junio de 2017). *mxcity guia insider*. Recuperado el 10 de Enero de 2017, de mxcity guia insider: https://mxcity.mx/2015/04/sabias-que-el-d-f-es-una-de-las-ciudades-con-mas-huertos-urbanos/

Porto, J. P. (21 de Enero de 2018). *Definicion.de*. Obtenido de Definicion.de: https://definicion.de/diagrama-de-bloques/

rashid, m. h. (2017). *electronica de potencia.* MEXICO: prentice hall hispanoamerica, S.A.

UNIVERSIDAD NACIONAL DEL CALLAO. (2009). *ESTUDIO DE LA EXTRACCIÓN Y DETERMINACIÓN DE LA COMPOSICIÓN QUÍMICA DEL ACEITE ESENCIAL DE PAICO.* callao: universitaria. Obtenido de https://unac.edu.pe/documentos/organizacion/vri/eu/CienciaTecnologia12.pdf#page=6

Universitat de les Illes Balears. (22 de Agosto de 2017). *Herbario Virtual del Mediterráneo*. Obtenido de http://herbarivirtual.uib.es/cas-uv/familia/1802.html

vilardell, e. n. (2017). *Fuentes de alimentación conmutadas en la práctica.* Fidestec.

Villegas Ezpinoza, J. A., Briseño Ruiz, S. E., Aguilar Garcia, M., & Sosa y Silva Carballo, R. A. (2017). *Guía de cultivo de tarragón francés.* La Paz, Baja California Sur, México.: Centro de Investigaciones Biológicas del Noroeste.

Bibliografía de imágenes

Ilustración 2.1 huerto urbano, recuperado el 20 de enero de 2018, de http://www.lahuertinadetoni.es/como-empezar-un-huerto-urbano/

Ilustración 2.2 cultivo de flores, recuperado el 20 de enero de 2018, de https://www.jardineriakuka.com/huerto-geotextil/10027-huerto-urbano-vertical-garden-3-bolsillos.html

Ilustración 2.3 cultivo de hortalizas, recuperado el 20 de enero de 2018, de http://www.todohuertoyjardin.es/blog/iniciacion-en-un-huerto-urbano

Ilustración 2.4 cultivo de árboles frutales, recuperado el 20 de enero del 2018, de https://www.guiadejardin.com/2015/07/ideas-para-un-huerto-urbano-melocotones.html

Ilustración 2.5 cultivo de aromáticas, recuperado el 20 de enero del 2018, de http://www.upsocl.com/verde/todo-lo-que-necesitas-saber-para-cultivar-exitosamente-diversas-plantas-aromaticas-en-tu-huerto/

Ilustración 2.6 huerto con sistema de riego por goteo, recuperado el 20 de enero de 2018, de https://viaorganica.org/como-usar-las-botellas-de-plastico-en-la-huerta/

Ilustración 2.7 huerto con sistema de riego subterráneo, recuperado el 20 de enero del 2018, de https://huerta.ojodeltiempo.com/beneficios-huerto-vertical-casa/

Ilustración 2.8 bomba para fuente, recuperado el 25 de enero del 2018, de https://bit.ly/2Mhd3Jz

Ilustración 2.9 bomba de 6v-12v, recuperado el 26 de enero de 2018, de https://www.hazlo2mismo.com/mini-bomba-de-agua-para-fuente-sumergible-electrica-abs-12v-200l-h-400054.html

Ilustración 2.10 bomba de 3.5v-12v, recuperado el 15 de enero de 2018, de autoría

Ilustración 2.11 manguera de plástico y conexiones plásticas, recuperado el 20 de enero de 2018, de autoría.

Ilustración 2.12 manguera de acuario, recuperado el 26 de enero de 2018, de http://www.nascapers.es/tienda/tuberiamanguera-filtro/1245-manguera-transparente-16-22mm.html

Ilustración 2.13 manguera para gasolina, recuperado el 26 de enero de 2018, de https://articulo.mercadolibre.com.ve/MLV-500986293-manguera-para-gasolina-negra-516-gates-usa-80mm-pmetro-_JM

Ilustración 2.14 servomotor MG996R, recuperado el 10 de marzo de 2018, de https://www.ardobot.com/servomotor-mg996r-pi-oneria-metalica.html

Ilustración 2.15 servomotor futaba, recuperado el 10 de marzo de 2018, de https://servodatabase.com/servo/futaba/s3003

Ilustración 2.16 servomotor SG92R, recuperado el 12 de marzo de 2018, de http://www.dx.com/p/tower-pro-sg92r-9g-servo-gear-with-2-5kg-torque-transparent-blue-234002#.W4Y8oOgzbIU

Ilustración 2.17 electroválvulas y su funcionamiento, recuperado el 17 de marzo de 2018, de https://www.altecdust.com/soporte-tecnico/que-son-las-electrovalvulas

Ilustración 2.18 sensor yl-69, recuperado el 17 de marzo de 2018, de http://www.maxelectronica.cl/temperatura-y-humedad/44-sensor-de-humedad-de-suelo-modelo-yl-38-y-sonda-yl-69.html

Ilustración 2.19 sensor interruptor de nivel de nivel liquido, recuperado el 19 de marzo de 2018, de https://solectroshop.com/product-spa-432-Sensor-nivel-liquido-vertical-interruptor.html

Ilustración 2.20 sensor ultrasónico, recuperado el 17 de marzo de 2018, de https://electronilab.co/tienda/sensor-de-distancia-de-ultrasonido-hc-sr04/

Ilustración 2.21 modulo *bluetooth* HC-05, recuperado el 19 de marzo de 2018, de https://alselectro.wordpress.com/tag/bluetooth-hc05/

Ilustración 2.22 esquema de fuentes lineales, recuperado el 21 de marzo de 2018, de http://fallasmonitoreseimpresoras.blogspot.com/

Anexo

Código de carga de valores iniciales

```
#include <EEPROM.h>

void setup() {

  int val1=55;

  int val2=85;

  int val3=75;

  int val4=70;

  int val5=55;

    EEPROM.write(0,(byte)val1);

    EEPROM.write(1,(byte)val2);

    EEPROM.write(2,(byte)val3);

    EEPROM.write(3,(byte)val4);

    EEPROM.write(4,(byte)val5);

}

void loop() {

  }
```

Código completo del proyecto

```
#include <EEPROM.h>
int hp1;
int hp2;
int hp3;
int hp4;
int hp5;
int dhp1;
int dhp2;
int dhp3;
int dhp4;
int dhp5;
int lec1;
int lec2;
int lec3;
int lec4;
int lec5;
int lec11;
int lec22;
int lec33;
int lec44;
int lec55;
#include <NewPing.h>
#define TRIGGER_PIN  9
#define ECHO_PIN     8
#define MAX_DISTANCE 100
 int dis;
 int nivel;
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
```

```
 #include <Servo.h>
Servo servo;
char buffer[40];
char trans[40];
char env[40];
char dat[40];
char txrx[40];
void setup() {
pinMode(3,OUTPUT);//activador del motor
pinMode(2,OUTPUT);//activador de la bomba
servo.attach(5); //pin del servomotor
pinMode(6,OUTPUT);//señal de alerta de malfuncionamiento
pinMode(7,OUTPUT);//señal de nivel bajo
  Serial.begin(9600);
   pinMode(13,OUTPUT);
   digitalWrite(3,1);
   delay(200);
   servo.write(88);
   delay(2000);
   digitalWrite(3,0);
   delay(150);
}
void loop() {
   dhp1=EEPROM.read(0);
   dhp2=EEPROM.read(1);
   dhp3=EEPROM.read(2);
   dhp4=EEPROM.read(3);
   dhp5=EEPROM.read(4);
    lec1= analogRead(A1);
```

```
  lec2= analogRead(A2);
  lec3= analogRead(A3);
  lec4= analogRead(A4);
  lec5= analogRead(A5);
  lec1=map(lec1,0,1023,100,0);
  lec2=map(lec2,0,1023,100,0);
  lec3=map(lec3,0,1023,100,0);
  lec4=map(lec4,0,1023,100,0);
  lec5=map(lec5,0,1023,100,0);
  int uS = sonar.ping_median();
  dis= uS / US_ROUNDTRIP_CM;
 nivel=map(dis,3,30,100,0);
lec11= analogRead(A1);
lec11=map(lec11,0,1023,100,0);
lec22= analogRead(A2);
lec22=map(lec22,0,1023,100,0);
lec33= analogRead(A3);
lec33=map(lec33,0,1023,100,0);
lec44= analogRead(A4);
lec44=map(lec44,0,1023,100,0);
lec55= analogRead(A5);
lec55=map(lec55,0,1023,100,0);
while(Serial.available()>0)
{
  int hp1=Serial.parseInt();//
  int hp2=Serial.parseInt();
  int hp3=Serial.parseInt();
  int hp4=Serial.parseInt();
  int hp5=Serial.parseInt();
```

```
   if(Serial.read()=='\n')
   {
     EEPROM.write(0,(byte)hp1);
     EEPROM.write(1,(byte)hp2);
     EEPROM.write(2,(byte)hp3);
     EEPROM.write(3,(byte)hp4);
     EEPROM.write(4,(byte)hp5);
}
 break;
}
  if(nivel==111){
sprintf(buffer,"%d,%d,%d,%d,%d,%d,%s,%s",lec1,lec2,lec3,lec4,lec5,nivel,"averiado","ult
rasonico");
    delay(500);
   Serial.print(buffer);
   Serial.println(",datos");
    digitalWrite(6,1);
   }
   else{//labe elsex
      if(nivel>=15)
   {
     digitalWrite(7,0);
  delay(1000);
  if(lec1==lec11 && lec2==lec22 && lec3==lec33 && lec4==lec44 && lec5==lec55)
   {
     digitalWrite(6,0);
sprintf(buffer,"%d,%d,%d,%d,%d,%d,%s,%s,",lec1,lec2,lec3,lec4,lec5,nivel,"optimos","ni
nguno");
     delay(500);
```

```
  Serial.print(buffer);
  Serial.println("datos");
 }
 if(lec2==lec22){
 sprintf(trans,"%d,%d,%d,%d,%d,%d",lec1,lec2,lec3,lec4,lec5,nivel);
Serial.println(trans);
    if(lec2<=(dhp1-2)){
              digitalWrite(3,1);
              delay(200);
              servo.write(88);
              delay(2000);
               servo.write(55);//da pos
              delay(5000);
              digitalWrite(3,0);
              delay(150);
            while(lec2<=(dhp1+2))
            {
              digitalWrite(2,1);//activacion de la bomba
              delay(6000);
              digitalWrite(2,0);
              delay(2500);

sprintf(trans,"%d,%d,%d,%d,%d,%d,%s,%s",lec1,lec2,lec3,lec4,lec5,nivel,"optimos","rieg
o");
              lec2=analogRead(A2);
              lec2=map(lec2,0,1023,100,0);
             Serial.print(trans);
             Serial.println(",datos");
           }
           delay(500);
```

```
            digitalWrite(3,1);
            servo.write(88);
            delay(4000);
            digitalWrite(3,0);
              }
 }
          else
          {
           digitalWrite(6,1);
sprintf(trans,"%d,%d,%d,%d,%d,%d,%s,%s",lec1,lec2,lec3,lec4,lec5,nivel,"averiado","sen
sor 2 averiado");
        delay(1000);
        Serial.println(trans);
          }
         if(lec3==lec33){
   sprintf(env,"%d,%d,%d,%d,%d,%d",lec1,lec2,lec3,lec4,lec5,nivel);
  Serial.println(env);
    if(lec3<=(dhp3-2)){
                digitalWrite(3,1);
                delay(200);
                servo.write(60);
                delay(2000);
                 servo.write(87);
                delay(5000);
                digitalWrite(3,0);
                delay(150);
              while(lec3<=(dhp3+2))
                {
                digitalWrite(2,1);
                delay(6000);
```

```
                digitalWrite(2,0);
                delay(2500);
sprintf(env,"%d,%d,%d,%d,%d,%d,%s,%s",lec1,lec2,lec3,lec4,lec5,nivel,"optimos","riego
");
                lec3=analogRead(A3);
                lec3=map(lec3,0,1023,100,0);
              Serial.print(env);
              Serial.println(",datos");
             }
             delay(500);
             digitalWrite(3,1);
             servo.write(88);
             delay(4000);
             digitalWrite(3,0);
               }
 }
          else
          {
            digitalWrite(6,1);
sprintf(env,"%d,%d,%d,%d,%d,%d,%s,%s",lec1,lec2,lec3,lec4,lec5,nivel,"averiado","sens
or 3 averiado");
        delay(1000);
        Serial.println(env);
           }
     if(lec4==lec44){ // sensor 4 operando correctamente
  sprintf(dat,"%d,%d,%d,%d,%d,%d",lec1,lec2,lec3,lec4,lec5,nivel);
 Serial.println(dat);
           if(lec4<=(dhp4-2)){
              digitalWrite(3,1);
              delay(200);
```

```
                servo.write(70);
                delay(2000);
                 servo.write(118);
                delay(5000);
                digitalWrite(3,0);
                delay(150);
              while(lec4<=(dhp4+2))
                {
                digitalWrite(2,1);
                delay(6000);
                digitalWrite(2,0);
                delay(2500);
sprintf(dat,"%d,%d,%d,%d,%d,%d,%s,%s",lec1,lec2,lec3,lec4,lec5,nivel,"optimos","riego"
);
                lec4=analogRead(A4);
                lec4=map(lec4,0,1023,100,0);
               Serial.print(dat);
               Serial.println(",datos");
              }
              delay(500);
              digitalWrite(3,1);
              servo.write(88);
              delay(4000);
              digitalWrite(3,0);
                }
  }
           else
           {
            digitalWrite(6,1);
```

```
sprintf(dat,"%d,%d,%d,%d,%d,%d,%s,%s",lec1,lec2,lec3,lec4,lec5,nivel,"averiado","senso
r 4 averiado");
        delay(1000);
        Serial.println(dat);
          }
        if(lec5==lec55){
   sprintf(txrx,"%d,%d,%d,%d,%d,%d",lec1,lec2,lec3,lec4,lec5,nivel);
  Serial.println(txrx);
            if(lec5<=(dhp5-2)){
               digitalWrite(3,1);
               delay(200);
               servo.write(80);
               delay(2000);
                servo.write(140);
               delay(5000);
               digitalWrite(3,0);
               delay(150);
             while(lec5<=(dhp5+2))
               {
               digitalWrite(2,1);
               delay(6000);
               digitalWrite(2,0);
               delay(2500);
sprintf(txrx,"%d,%d,%d,%d,%d,%d,%s,%s",lec1,lec2,lec3,lec4,lec5,nivel,"optimos","riego
");
               lec5=analogRead(A5);
               lec5=map(lec5,0,1023,100,0);
              Serial.print(txrx);
              Serial.println(",datos");
```

```
            }
            delay(500);
            digitalWrite(3,1);
            servo.write(88);
            delay(4000);
            digitalWrite(3,0);
              }
 }
        else
          {
            digitalWrite(6,1);
sprintf(txrx,"%d,%d,%d,%d,%d,%d,%s,%s",lec1,lec2,lec3,lec4,lec5,nivel,"averiado","sens
or 5 averiado");
        delay(1000);
        Serial.println(txrx);
            }
  if(lec1==lec11){
   sprintf(buffer,"%d,%d,%d,%d,%d,%d",lec1,lec2,lec3,lec4,lec5,nivel);
   delay(1000);
  Serial.println(buffer);
          if(lec1<=(dhp1-2)){
                digitalWrite(3,1);
                delay(200);
                servo.write(88);
                delay(2000);
                 servo.write(38);
                delay(5000);
                digitalWrite(3,0);
                delay(150);
              while(lec1<=(dhp1+2))
```

```
            {
              digitalWrite(2,1);
              delay(6000);
              digitalWrite(2,0);
              delay(2500);
sprintf(buffer,"%d,%d,%d,%d,%d,%d,%s,%s",lec1,lec2,lec3,lec4,lec5,nivel,"optimos","rie
go");
              lec1=analogRead(A1);
              lec1=map(lec1,0,1023,100,0);
            Serial.print(buffer);
            Serial.println(",datos");
              }
          delay(500);
          digitalWrite(3,1);
          servo.write(88);
          delay(4000);
          digitalWrite(3,0);
            }
 }
      else
        {
          digitalWrite(6,1);
sprintf(buffer,"%d,%d,%d,%d,%d,%d,%s,%s",lec1,lec2,lec3,lec4,lec5,nivel,"averiado","se
nsor 1 averiado");
       Serial.print(buffer);
       Serial.println(",datos");
          }
       }
     else
  {
```

```
    digitalWrite(7,1);
    while(nivel>=80)
    {
       int uS = sonar.ping_median();
 dis= uS / US_ROUNDTRIP_CM;
nivel=map(dis,3,30,100,0);
            }
            digitalWrite(7,0);
    }
  }
}
```

Link la aplicación para Smartphone

https://1drv.ms/f/s!AvVBwGrPgXhii0Vtakgx400mBn-h

Link de la placa Arduino y bibliotecas

https://1drv.ms/f/s!AvVBwGrPgXhii0M3uj_iEwikP1TL

https://1drv.ms/f/s!AvVBwGrPgXhii0CuU87wAn0GUZ_x