UNIVERSIDAD CATÓLICA DE CUENCA

UNIVERSIDAD CATÓLICA DE CUENCA

Comunidad Educativa al Servicio del Pueblo

UNIDAD ACADÉMICA DE INGENIERÍA, INDUSTRIA Y CONSTRUCCIÓN

CARRERA DE INGENIERÍA ELÉCTRICA

AUTOMATIZACIÓN DE UN INVERNADERO SEMI HIDROPÓNICO

PROYECTO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO ELÉCTRICO

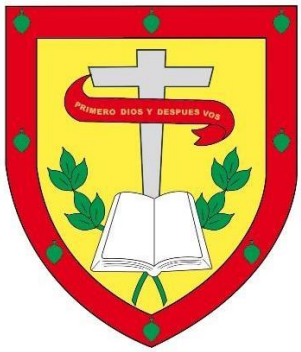
AUTOR: CHRISTIAN RAFAEL COYAGO CALLE

DIRECTOR: ING. JUAN CARLOS COBOS TORRES Ph.D.

CUENCA – ECUADOR

2023

DIOS, PATRIA, CULTURA Y DESARROLLO



UNIVERSIDAD CATÓLICA DE CUENCA

Comunidad Educativa al Servicio del Pueblo

UNIDAD ACADÉMICA DE INGENIERÍA INDUSTRIA Y CONSTRUCCIÓN

CARRERA DE INGENIERÍA ELÉCTRICA

AUTOMATIZACIÓN DE UN INVERNADERO SEMI HIDROPÓNICO

PROYECTO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO ELÉCTRICO

AUTOR: CHRISTIAN RAFAEL COYAGO CALLE

DIRECTOR: ING. JUAN CARLOS COBOS TORRES Ph.D.

CUENCA - ECUADOR

2023

DIOS, PATRIA, CULTURA Y DESARROLLO

Declaratoria de Autoría y Responsabilidad

Christian Rafael Coyago Calle portador de la cédula de ciudadanía Nº 010645361. Declaro ser el autor de la obra: "Automatización de un invernadero semi hidropónico", sobre la cual me hago responsable sobre las opiniones, versiones e ideas expresadas. Declaro que la misma ha sido elaborada respetando los derechos de propiedad intelectual de terceros y eximo a la Universidad Católica de Cuenca sobre cualquier reclamación que pudiera existir al respecto. Declaro finalmente que mi obra ha sido realizada cumpliendo con todos los requisitos legales, éticos y bioéticos de investigación, que la misma no incumple con la normativa nacional e internacional en el área específica de investigación, sobre la que también me responsabilizo y eximo a la Universidad Católica de Cuenca de toda reclamación al respecto.

Cuenca, 19 de abril de 2022

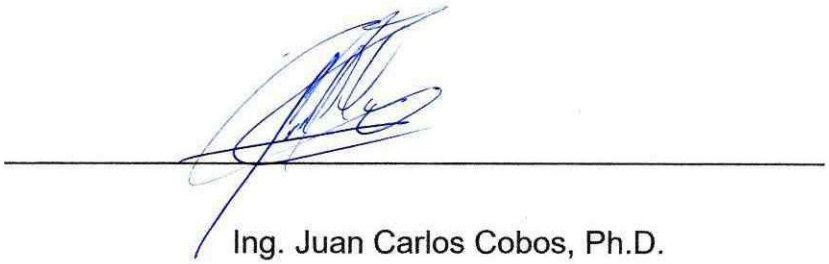
F: .....

Christian Rafael Coyago Calle

0106456361

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Christian Rafael Coyago Calle, bajo mi supervisión.



Ing. Juan Carlos Cobos, Ph.D.

Docente tutor del trabajo de titulación

DEDICATORIA

Dedico mi tesis con todo el cariño a mis padres y todos mis seres queridos por su apoyo incondicional y su compañía en momentos difíciles, brindándome la seguridad para seguir adelante evitando que me dé por vencido.

A mis compañeros y amigos quienes me acompañaron en este camino lleno de dificultades sin esperar nada a cambio.

Dedico este trabajo de titulación a todos los docentes quienes me compartieron sus conocimientos plasmados los conocimientos necesarios para poder cumplir con esta y futuras metas.

Att. Christian Rafael Coyago Calle.

AGRADECIMIENTOS

Al concluir esta etapa de mi vida quiero agradecer a Dios por cada una de las bendiciones que me ha brindado hasta alcanzar mi meta. Con ello espero expresar mis agradecimientos más sinceros a todos quienes fueron un pilar fundamental en mi camino, y a todos mis seres queridos los cuales me brindaron su apoyo incondicional.

A mis padres Claudio Coyago, María Calle, quienes día a día me dieron la confianza, y el empujón que muchas de las veces es el indispensable para seguir adelante.

Mi eterna gratitud a la Universidad Católica de Cuenca, y a su honorable cuerpo docente, de manera especial a mi tutor de tesis Ing. Juan Carlos Cobos, por haberme guiado dentro de este trayecto con paciencia y su sabiduría de igual forma agradezco a cada docente que conforma la Unidad Académica de Posgrado, Ing. Pablo Buestan, Ing. Cristian Morocho y mi cotutor Ing. Manuel Álvarez quienes, con su ayuda y enseñanzas, fueron un apoyo fundamental para culminar con este trabajo de titulación.

Att: Christian Rafael Coyago Calle.

RESUMEN

La automatización de un invernadero semi hidropónico implica la implementación de sistemas tecnológicos, los cuales permiten controlar y monitorear de manera eficaz las condiciones ambientales, suministros hídricos y nutrientes a los cultivos. Permitiendo optimizar el crecimiento de las plantas, mejorando la calidad de los cultivos, reduciendo costos y tiempo de trabajo necesario para la mantención del invernadero. En este trabajo, se realiza un prototipo funcional de bajo costo para la automatización de un invernadero, planteando como principal objetivo la conservación de recursos hídricos. La automatización, se desarrolló mediante herramientas tecnológicas con hardware DIY utilizando microcontroladores y miniordenadores, permitiendo la modificación de nuevos parámetros para el control del invernadero a futuro. El prototipo desarrollado es una nueva herramienta, que busca integrarse para el control de invernaderos pequeños o de mayor escala respetando los parámetros de necesarios para la lectura adecuada de los sensores.

Con el prototipo de automatización, se logró mantener control de riego al 80% de humedad, con una temperatura promedio del invernadero en 25°C , dicho invernadero encuentra en constante monitoreo de los parámetros climáticos; los cuales, son útiles para optimizar el control de riego, con porcentajes apropiados de un nivel de agua y vitaminas de 200 litros de líquido. Permitiendo obtener un desarrollo y crecimiento de un cultivo sano, así como valores positivos en el desarrollo de las plantas, presentando un crecimiento promedio en un mes de 6.42 cm de alto, y finalmente obteniendo el fruto del cultivo.

Palabras clave: Invernadero, Arduino, Rasberry Pi, Python, Cultivo Semi hidropónico.

ABSTRACT

Automation of a semi-hydroponic greenhouse involves implementing technological systems to effectively control and monitor environmental conditions, water supply, and nutrients to the crops. This makes it possible to optimize plant growth, improve crop quality, and reduce costs and labor time required for greenhouse maintenance. In this work, a low-cost, functional prototype for the automation of a greenhouse is developed, with the primary objective of conserving water resources. Automation was developed by technological tools with DIY hardware using microcontrollers and minicomputers, allowing adjustments of new parameters to control the greenhouse in the future. The prototype developed is a new tool that seeks to be integrated to control small or larger-scale greenhouses following the required parameters for the proper reading of the sensors.

With the automation prototype, it was possible to maintain irrigation control at 80% humidity, with an average greenhouse temperature of 25 °C. This greenhouse is constantly monitored by climatic parameters, which are useful for optimizing irrigation control, with appropriate percentages of water level and 200 liters of liquid vitamins. Allowing to obtain healthy crop development and growth, as well as positive values in plant development, showing an average growth in a month of 6.42 cm high, and finally obtaining the fruit of the crop.

Keywords: Greenhouse, Arduino, Raspberry Pi, Python, semi-hydroponic

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

LISTA DE TABLAS

LISTA DE ANEXOS

CAPITULO I

INTRODUCCIÓN

En la actualidad, la sociedad se encuentra preocupada por el tipo de alimentos que se pueden obtener, debido a que en los comercios comunes no brindan información sobre la calidad de los alimentos y mucho menos su procedencia, muchas de las veces se toman en cuenta su apariencia, a su vez considerando su tamaño, tipo de color y consistencia, el presente estudio trató sobre el cultivo de frutillas.

Los mayores países productores de frutilla son China, Estados Unidos de América, México, Turquía y Egipto, los cuales aportan más de setenta por ciento de producción total de frutilla del mundo (Ramírez Padrón, Laura Cecilia Cauich et al., 2020). Siendo así que mayormente la población, se inclina hacia alimentos sanos y orgánicos que sean obtenidos mediante prácticas no nocivas para la salud y el medio ambiente, teniendo como principal inconveniente el tiempo y espacio para llevar a cabo su propio cultivo.

La implementación de la automatización dentro de los invernaderos ayuda de una manera considerable a reducir el tiempo, el espacio y el impacto medio ambiental, debido al uso controlado de recursos necesarios para el sustento del cultivo.

Un invernadero, se define como una estructura cubierta que proporciona a las plantas un ambiente controlado de manera óptima mediante el ajuste de las condiciones climáticas que ayuden a mejorar el crecimiento, para reducir el costo de producción y aumentar el rendimiento de los cultivos (Jain, Nikita Bhakar & Singhal, 2017).

Los sistemas de cultivo en invernadero disponibles en su gran mayoría son sistemas monitoreados por humanos, lo que implica la visita humana continua; puede causar angustia al trabajador y también una disminución en el rendimiento, si la

temperatura y la humedad no se mantienen de manera adecuada y regular. Esto allana el camino para el concepto de automatización de invernaderos (S. Raj, Jennifer J, 2019).

El internet de las Cosas (Internet of Things) está remodelando la agricultura, permitiendo a los agricultores un trabajo de precisión y sostenible en el campo, facilitando el monitoreo en la línea de cultivos, pues se puede supervisar y detectar malezas, nivel del agua, de plagas y controlar el nivel de crecimiento de los cultivos (Sreekantha & Kavya, 2017).

Un sistema de control clásico para invernaderos consiste en sensores, que monitorean todas las variables ambientales que controlan el progreso del cultivo. Dando la oportunidad de optimizar el tiempo y llevar un control de manera remota, mediante dispositivos móviles como celulares y ordenadores, lo cual permite garantizar el debido cuidado y una correcta administración del crecimiento del cultivo (Castañeda, Rodrigo Herrera Ruiz & José, Juan Escalante, 2003). El diseño y desarrollo de un sistema de automatización implica diferentes fases, que incluyen en el estudio de factores ambientales y respuestas de los cultivos (Shamshiri, Ramin Ishak, Wan Ismail, 2013).

La gran mayoría de lugares sufren problemas, generalmente cuando los cultivos crecen en ambientes que no son adecuados o no se tiene un control regulado, se ve afectado su crecimiento y calidad, causando un impacto directo en su valor económico, debido a la falta de entornos adecuados para el control del cultivo como pueden ser los invernaderos (Industria Horticola, 2008).

La automatización de invernaderos permite al usuario despreocuparse del encendido y apagado de equipos comúnmente utilizados para la agricultura, sabiendo que el equipo actuará en determinados periodos preestablecidos (Departamento de

ingeniería rural, 2000). Los cuales pueden ser de gran ayuda para la producción de frutilla que en el Ecuador se encuentra en aumento, debido a las condiciones agroclimáticas, y la variación de temperatura promedio de 22 y 26 °C en el país. (Carrera, Zambrano & Espartaco, 2018).

Por lo expuesto, se realizó el diseño y construcción del prototipo para automatizar un invernadero semi hidropónico utilizando sensores y actuadores, los cuales permiten obtener las variables del entorno tales como: temperatura ambiente y humedad relativa, nivel de humedad en las mangas del cultivo, temperatura y nivel del agua dentro de la cisterna.

La información receptada por los sensores sirve para el accionamiento de actuadores como: bombas de agua, iluminación, ventilación.

El centro de control se estableció directamente al interior del invernadero, siendo ubicado de manera estratégica para la correcta lectura de sensores y accionamiento eficaz de los actuadores, los cuales son los encargados del control riego, el cual se ejecuta directamente en las mangas en donde se encuentra el cultivo. Esto se logró gracias a electroválvulas que se encuentran en cada una de las mangas y módulos relés, los cuales se encuentran ubicados en posiciones estratégicas para su correcto funcionamiento.

1.1 Justificación del problema.

El bajo rendimiento de cultivos de frutilla dentro del Ecuador, se encuentra atribuido a las plagas, falta de cuidado como el riego y más factores ambientales que afectan directamente a la producción, esto ocasiona que los estándares de calidad de la frutilla sean mínimos y se encuentren ineficientes para su exportación y/o comercialización dentro del país.

Uno de los grandes problemas que se manifiestan en la sociedad actual, es el manejo inadecuado de los recursos para la mantención de un invernadero, los mismos que representan una amenaza tanto económica como a la calidad del cultivo a obtener.

En el sector agrícola uno de los principales problemas a afrontar es la implementación de un adecuado sistema de riego, este problema se puede observar con mayor impacto en épocas de verano en cultivos donde el regado se controla de manera manual, lo cual genera un gran desperdicio del recurso hídrico (Salinas Arcos, 2019).

Dado el tedioso cuidado muchos agricultores han optado por abandonar el cultivo de frutilla y priorizar recursos económicos para el cultivo de tomate (Salto Grande, 2012). Por lo mismo, Escaramilla afirma que el mayor enemigo de la agricultura es el cambio brusco de clima, llegando a causar la pérdida del 100% de su producción (Escaramilla, 2019).

La agricultura necesita satisfacer las exigencias de la población de manera acelerada, obligada así a buscar métodos para la rápida producción con altos niveles de calidad, de esta manera la agricultura ha evolucionado buscando nuevas maneras de cultivo promocionan los invernaderos, tomando en cuenta que en un invernadero se puede reducir a la tercera parte del tiempo en el que el fruto germine con respecto a la forma de cultivo tradicional sin un invernadero, tomando en cuenta estos resultados, se afirma que la automatización ayuda a minimizar el tiempo de crianza del cultivo, y aun mejor evitando pérdidas económicas bajo un control sofisticado de las variables climáticas del invernadero.

Se conoce que, se puede aplicar métodos de automatización según investigaciones referidas en artículos y revistas especialistas en cultivos de frutillas,

los cuales dan a conocer los diferentes cuidados que se deben considerar paso a paso desde el cultivo hasta cosecha del fruto.

Existe el interés para minimizar, el mal gasto de recursos que un invernadero conlleva, siendo estos amigables con el medio ambiente, por lo cual la automatización es una de las mejores opciones (Fernández, Milagros, Lorenzo Mínguez, Pilar, Cuadrado Gómez, Isabel MaGiménez Moolhvijzen et al., 2003).

Debido a esto, el usuario podrá tener control sobre el invernadero, obteniendo datos en tiempo real, facilitando así el cuidado del mismo, y teniendo a su vez mayor control sobre el cultivo, maximizando los resultados esperados y minimizando el uso de recursos.

Gracias a investigaciones previas sobre la automatización de invernaderos semi hidropónicos, se procede a realizar una selección de los métodos de automatización más eficaces para el correcto desarrollo del proyecto.

Como una solución al problema del desperdicio de recursos, se presenta el sistema de automatización gracias a que permite tener mejor control de los parámetros climáticos y ambientales dentro del invernadero, y a su vez ayudan a tener más control sobre los recursos a utilizarse para el cultivo de frutillas, recopilado datos en tiempo real y permitiendo tomar acciones oportunas.

En la actualidad, se requiere un gasto excesivo de recursos para mantener un invernadero, generando grandes problemas ambientales y económicos debido a que la mayoría de los invernaderos son manejados de manera empírica y no aprovechando al máximo los recursos utilizados.

El presente trabajo de titulación, se orienta a la automatización del invernadero ubicado en la Unidad Académica de Posgrado de la Universidad Católica de Cuenca.

Cuya automatización se realizó mediante lenguaje de programación Python, con el cual se gestionó el control de diversos sensores esenciales para la vida del cultivo.

Existen soluciones para eliminar el desperdicio innecesario de recursos, pero el más viable es la automatización, debido a que se tiene muchas ventajas frente a los métodos tradicionales, por lo que fue de gran importancia diseñar y construir el prototipo del sistema de automatización, cuyo principal objetivo es tener un control del invernadero de manera acertada y correcta.

Dentro de la automatización del invernadero se realizó el monitoreo y control de:

- Temperatura.
- Hidratación del cultivo.
- Nivel correcto de pH del agua.
- Control del reservorio del agua.
- Control de iluminación.

1.2 DEFINICIÓN DE LA ZONA DE ESTUDIO

La presente investigación se ejecutó en la Unidad Académica de Posgrado de la Universidad Católica de Cuenca, ver Figura 1.

Figura 1: Ubicación del Invernadero

Fuente: Adaptado de Google Earth

1.3 Objetivos.

1.3.1 Objetivo General.

- Automatizar el invernadero ubicado en la Unidad Académica de Posgrado de la Universidad Católica de Cuenca (UCACUE) mediante hardware DIY y Software con Open Source controlando y garantizando las variantes ambientales del cultivo de fresa.

1.3.2 Específicos.

- Revisar la literatura científica sobre invernaderos automatizados mediante gestores científicos para contar con información actualizada y de calidad.
- Levantar información sobre los requerimientos de riego, microclima, monitorización del invernadero mediante una observación y entrevista a los técnicos en agronomía para la determinación de las variables a controlar.
- Diseñar y construir el control, monitorización y comunicación del invernadero mediante software y hardware DIY, contando con un primer prototipo del sistema.
- Probar el sistema instalado mediante múltiples pruebas de funcionamiento y determinación del rendimiento del cultivo de fresa contando con un prototipo funcional para la automatización del invernadero Semi hidropónico.

1.4 METODOLOGÍA

El presente proyecto se basa en el desarrollo de un prototipo funcional para la automatización de un invernadero. Este proceso se divide en tres etapas, ver Figura 2.

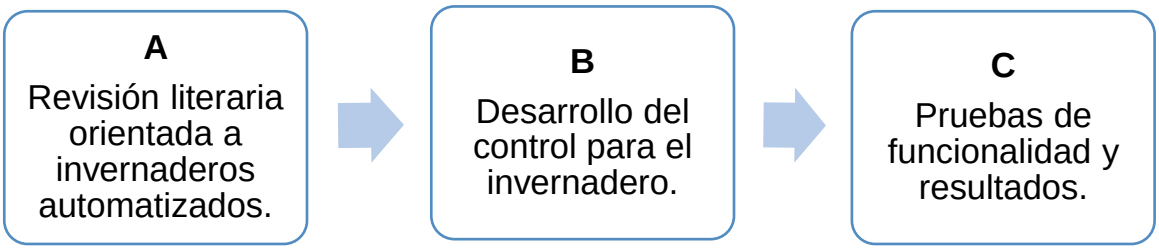


Figura 2: Pasos seguidos para la consecución del prototipo funcional.

Fuente: Autor

A continuación, se realiza un detalle de cada etapa:

A. Esta etapa es una de las más importantes para el correcto desarrollo del proyecto. Gracias a ello, se pudo desarrollar un prototipo adecuado para el invernadero, definiendo los parámetros para un correcto funcionamiento del invernadero. Se realizó simulaciones mediante el software Proteus, mediante los siguientes pasos:

- Planteamiento del prototipo, bosquejo y selección de materiales para su implementación.
- Realización de simulaciones de conexión de sensores y actuadores, los cuales fueron controlados mediante un sistema microprocesado.
- Dimensionamiento de la estructura en donde se realizó la instalación del prototipo, modificando las distancias para el correcto funcionamiento de cada artefacto.

B. Diseño de la conexión eléctrica de acuerdo a los resultados obtenidos en el apartado anterior. Se desarrollo la construcción de un prototipo que cumple con parámetros estrictos de funcionamiento acorde a las necesidades planteadas:

- Se realizó un estudio de los materiales a implementar dentro del invernadero.
- Se diseñó un sistema de control microcontrolado, el cual actúa como sistema de control conjuntamente con la interfaz gráfica realizada en PYQT.
- Se diseñó el sistema de control utilizando como plataforma interactiva dentro del programa PYQT Designer, el cual funciona con lenguaje de programación Python y permitió generar el sistema de control del invernadero.

C. Se realizó las respectivas pruebas de funcionamiento del prototipo dentro del invernadero en busca de errores, para realizar las respectivas correcciones del modelo realizando los siguientes pasos:

- Pruebas preliminares de funcionamiento de los sensores en el circuito.
- Verificación de posibles fallas de funcionamiento.
- Verificación del correcto funcionamiento de los actuadores del circuito.
- Correcciones necesarias para su correcto funcionamiento.
- Se analizó la información obtenida mediante las lecturas de los sensores para evidenciar el correcto funcionamiento.

El presente documento se estructura de la siguiente manera: en el CAPITULO II se detalla el Marco Teórico, en el CAPITULO III se presenta y detalla el desarrollo, análisis y discusión de resultados; finalmente, en el CAPITULO IV se presentan las conclusiones, recomendaciones, bibliografía y anexos.

CAPITULO II

FUNDAMENTACIÓN TEÓRICA

2.1 Invernadero.

Un invernadero consiste en un lugar estático y cerrado, que se encuentra cubierto con plástico y puede incluir vidrio dando como objetivo la obtención de microclimas mediante controles de temperatura, humedad y diversos factores ambientales. Además, se pueden controlar el sistema de riego y ventilación, si es necesario para la obtención de cultivos de forma controlada dando paso a cosechas más frecuentes y de mejor calidad.

De la misma manera, algunos invernaderos pueden contar con diferentes sistemas de control y uno de los más importantes es su sistema de riego, ya que gracias a esto se puede administrar los agroquímicos o abonos necesarios en el cultivo. Cabe recalcar que, uno de los sistemas más eficientes y escogidos para realizar este proyecto es el sistema de riego por goteo.

Este sistema de riego también es conocido como sistema de riego localizado en el pie de cada planta, el cual funciona con un método de irrigación y permite la aplicación de agroquímicos, abonos y por supuesto el agua. Es por ello que presenta una gran ventaja en el ahorro de grandes cantidades de agua, sin generar excesos de riego o estancamientos.

Por lo tanto, se debe conocer sobre los niveles de variación de los parámetros ambientales dentro del invernadero ya que estos repercuten de manera directa al cultivo, para ello se utiliza el control de la temperatura y humedad, el cual se utiliza para poder tener un pronóstico correcto del riego que se da al cultivo, evitando la putrefacción del mismo debido al exceso de humedad en el cultivo dentro del

invernadero. El invernadero, se encuentra situado en la Unidad Académica de Posgrado de la Universidad Católica de Cuenca, ver Figura 3.

Figura 3: Invernadero ubicado en la Unidad Académica de Posgrado (UCACUE).

Fuente: El autor.

Este invernadero está conformado por plástico Polietileno (PE), al cual se le atribuyen propiedades físicas como:

- Peso bajo.
- Alta densidad.
- Gran resistencia.

Envejecimiento tanto físico como radiométrico, el cual se determina por el nivel de luz de trasmisión del material.

Este material también consta con propiedades ópticas y térmicas (Bernabé I Ramos-López; Gabino A Martínez-Gutiérrez; Isidro Morales; Cirenio Escamirosa-Tinoco; & Aleyda Pérez-Herrera, 2017).

Además, es un invernadero con techumbre curva (Túnel) ver Figura 4. Este, tiene ventajas como:

- Alta transmitancia de luz solar.
- Buen volumen interior de aire.
- Construcción de baja complejidad (Serrano Cermeño, 2005).

Figura 4: Principales tipos de invernaderos.

Fuente: (Agro Krebs).

2.2 Sistemas microcontrolados y microprocesados.

Los sistemas microcontrolados utilizan chips, los cuales se integran a un solo dispositivo procesadores, memorias, periféricos de entrada y salida como pueden ser convertidores analógicos y digitales, los cuales son utilizados para una amplia variedad de aplicaciones debido a su flexibilidad, bajo costo y eficiencia energética.

2.2.1 Arduino

Se considera como una plataforma de código abierto (Open Source) basado en hardware y software. Los proyectos de Arduino pueden ser autónomos o pueden comunicarse con software en ejecución en un ordenador, ver sus características en la Figura 5.

ARDUINO MEGA 2560		
 **Figura 4.** Arduino Mega 2560.	***Microcontrolador:***	ATmega2560
	Voltaje de Operación:	5V
	Pines digitales:	28
	Pines PWM:	15
	Pines de entradas análogas:	11
	Corriente DC por cada pin I/O:	20 mA
	Corriente DC en el pin de 3.5V:	50 mA
	Memoria Flash:	256 KB
	Memoria SRAM:	256 KB
	Memoria EEPROM:	4 KB
	Velocidad de reloj:	16 MHZ

Figura 5: Arduino Mega y características.

Fuente: (Arduino y El Internet de Las Cosas - Google Play Libros, 2018)

2.2.2 Raspberry Pi

Un Raspberry Pi, es un miniordenador de muy bajo costo, este puede ser conectado mediante un micro HDMI a cualquier monitor. De la misma manera, que un ordenador común, se le puede conectar periféricos de entrada como son el teclado y un mouse, para su respectivo control, ver Figura 6.

Figura 6: Raspberry Pi (Mini ordenador)

Fuente: (*Ferran Fabregas - Google Libros*, 2020)

Las especificaciones técnicas, se detallan en la Tabla 1.

Tabla 1*: Especificaciones técnicas Raspberry Pi 4*

Modelo	Raspberry Pi 4
Sistema en un chip	Broadcom BCM2711
CPU	Procesador de cuatro núcleos a 1,5 GHz con brazo Cortex-A72
Memoria	1/2/4GB LPDDR4 RAM
Conectividad	802.11ac Wi-Fi / Bluetooth 5.0, Gigabit Ethernet
Vídeo y sonido	2 x puertos micro-HDMI que admiten pantallas de 4K@60Hz a través de HDMI 2.0, puerto de pantalla MIPI DSI, puerto de cámara MIPI CSI, salida estéreo de 4 polos y puerto de vídeo compuesto.
Puertos	2 x USB 3.0, 2 x USB 2.0
Alimentación	5V/3A vía USB-C, 5V vía cabezal GPIO
Expansión	Cabezal GPIO de 40 pines

Fuente: (PcComponentes, 2022)

2.3 Sensores

Los sensores están diseñados para detectar cambios en el entorno físico o químico y convertirlos en señales eléctricas que posteriormente serán interpretadas por ordenadores o microcontroladores como sea el caso.

2.3.1 Módulo de temperatura y Humedad - Sensor DHT22

El sensor DHT22 proporciona la lectura de humedad relativa y temperatura. Es un sensor digital que viene calibrado de fábrica, aunque es preferible realizar una corrección de las medidas, proporcionando fiabilidad y estabilidad en sus lecturas, ver Figura 7. Se puede transmitir aproximadamente hasta a 20 metros de distancia (David Marcelo Oña Salazar & Byron Alexis Vaca Marmol, 2015), mediante su protocolo de comunicación propietario de un solo conductor para la comunicación con el microcontrolador similar al de Dallas semiconductor. Las especificaciones técnicas se detallan en la Tabla 2.

Figura 7: DHT 22

Fuente: (DHT22 Datasheet CodigoElectronica, 2022)

Tabla 2. Especificaciones técnicas DHT 22

Modelo	DHT22
Fuente de alimentación	3.3 – 5v DC. 2.5mA
Señal de salida	Señal digital.
Elemento de detección	Condensador de humedad polímeros y DS1SB20 para la detección de temperatura.
Rango de medición	Humedad 0-100% RH; Temperatura -40 – 125 Celsius.

2.3.2 Sensor humedad de suelo - Capacitive v2.0.

Los sensores de humedad de suelo capacitivos ver (Figura 8), utilizan el principio de funcionamiento de un condensador para poder aproximar la humedad del suelo. Son fabricados con un material anticorrosivo que permite extender su vida útil. Las especificaciones técnicas se detallan en la Tabla 3.

Figura 8: Sensor humedad de suelo

Fuente: (Proto Supplies, 2021)

Tabla 3. Especificaciones técnicas Sensor humedad de suelo

Modelo	Capacitive v2.0
Fuente de alimentación	3.3 – 5.5v DC. A 5mA
Señal de salida	Señal analógica.
Dimensiones	23(ancho)mm x102(largo)mm x7(alto) mm

2.3.3 Sensor de temperatura - DS18B20

El sensor DS18B20, se caracteriza por enviar datos mediante un solo cable. Este sensor, se encuentra blindado para resistir climas adversos, para evitar lecturas fallidas es preciso colocar una resistencia de 4.7 ohmios en medio del pin de datos y la alimentación (AG Electrónica, 2017). Las especificaciones técnicas, se detallan en la Tabla 4.

Figura 9: Sensor de temperatura DS18B20

Fuente:(AG Electrónica, 2017)

Tabla 4. Especificaciones del sensor de temperatura DS18B20

Modelo	DS18B20
Fuente de alimentación	3.0 - 5v DC /1.5mA
Señal de salida	Señal digital
Precisión	+- 0.5 grados
Rango de operación	-50 a 125 grados Centígrados

2.3.4 Modulo Ultrasónico - HC-SR04

El medidor ultrasónico HC-SR04 permite medir la distancia entre un objeto y el sensor, al enviar pulsos ultrasónicos y midiendo el tiempo en que regresan los pulsos enviados, ver Figura 10. Las especificaciones técnicas se detallan en la Tabla 5.

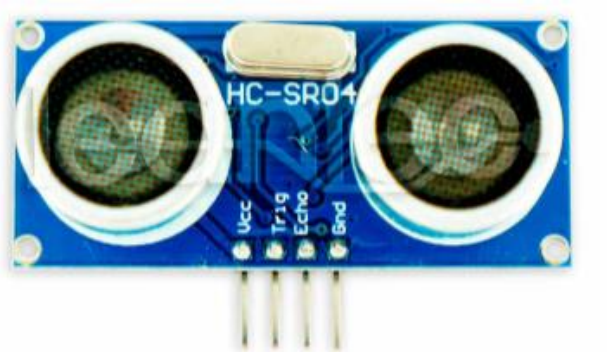


Figura 10: Modulo Ultrasónico

Fuente: (Datasheet HC-SR04, 2022)

Tabla 5. Especificaciones técnicas Modulo Ultrasónico

Modelo	HC-SR04
Fuente de alimentación	5v DC - 15mA
Señal de salida	Digital
Rango de funcionamiento	2 a 500 cm
Angulo de detección	15 a 20 grados
Frecuencia de trabajo	40 KHz

2.3.5 Modulo ultrasónico - SR04M-2

El módulo SR04M-2 es similar al módulo ultrasónico HC-SR04, la diferencia que lo caracteriza es su sensor remoto resistente al agua, asegurando los componentes electrónicos alejándolos de la humedad, del mismo modo este sensor incluye transmisores y receptores para determinar la distancia (Figura 11). Las especificaciones técnicas se detallan en la Tabla 6.

Figura 11: Sensor ultrasónico sumergible SR04M-2

Fuente: (Zamux, 2022)

Tabla 6. Especificaciones técnicas Modulo Ultrasónico SR04M-2

Modelo	SR04M-2
Fuente de alimentación	5v DC - 5mA
Señal de salida	Digital
Distancia máxima	4.5 m
Angulo de detección	Menos de 50 grados
Frecuencia de trabajo	40 KHz
Temperatura de trabajo	-10 a 70 grados Celsius

2.3.6 Sensor de CO_2 - MG-811

El sensor de CO2, como su nombre lo indica se encuentra diseñado para medir el dióxido de carbono del ambiente, este sensor tiene una característica especial debido que necesita calentarse para poder tener una lectura correcta de los datos de CO2 (ver Figura 12) (Rajguru Electronics, 2017).

Figura 12: Sensor CO2

Fuente: (*MG-811 Carbon Dioxide Sensor Module*, 2017)

Tabla 7. Especificaciones técnicas del sensor de CO2

Modelo	MG-811
Fuente de alimentación	6v DC
Señal de salida	Analógica
Rango de detección	350 – 10,000 ppm CO2
Tipo de sensor	Sensor de gas

2.4 Actuadores

Los actuadores son considerados dispositivos que convierten señales eléctricas en movimiento mecánico, siendo fundamental para sistemas de automatización permitiendo a la programación interactuar con el mundo físico.

2.4.1 Modulo relé

Los módulos relevadores o relés, se utilizan comúnmente para el control de cargas de alta potencia mediante señales de baja potencia comúnmente generadas por microcontroladores; su diseño permite controlar cargas que van desde los 10 A y estos pueden variar para voltajes como máximo 250VAC o 30VDC.

El uso del módulo relé permite separar el circuito de control de la carga eléctrica de alta potencia aumentado la seguridad y facilitando el diseño de circuitos (Figura 13). Es importante considerar que la corriente de activación de cada relé es de 10mA a 20mA (Bolaños, 2016). Las especificaciones técnicas se detallan en la Tabla 8.

Figura 13: Modulo relé electromagnético

Fuente: (Bolaños, 2016)

Tabla 8. Especificaciones modulo relé.

Modelo	Relevadores
Voltaje de operación	5v DC - 10mA
Señal de salida	Señal analógica/digital
Tiempo de acción	10ms / 5ms
Voltaje máximo de carga	250VAC/ 30VDC

2.4.2 Válvula solenoide

Este tipo de válvulas se caracterizan por tener dos estados de funcionamiento, denominado abierto o cerrado; está conformado por dos partes: un solenoide y una carcasa plástica. Se utiliza para controlar el flujo de líquidos y se lo controla con un

microcontrolador y un drive de potencia, debido a que la potencia requerida para su activación es mayor a lo que pueden entregar los microcontroladores, ver Figura 14. Las especificaciones técnicas se detallan en la Tabla 9.

Figura 14: Válvula Solenoide

Fuente: (*Válvula Solenoide 1/2" 12VDC (NC)*, 2022)

Tabla 9. Especificaciones *Válvula Solenoide*

Modelo	Electroválvula solenoide
Voltaje de operación	12V DC
Corriente de operación	0.6 A
Potencia de consumo	8 W
Tamaño de conexión	½ Pulgada
Presión de funcionamiento mínima	0.02 MPa (Mínima); 0.8 MPa (Máxima)
Tiempo de respuesta	< 0.15s (Apertura); <0.3s (Cerrado)
Reposo	Normalmente cerrado

CAPITULO III

DESARROLLO DEL PROTOTIPO DE AUTOMATIZACIÓN

Para el desarrollo del prototipo se utilizó los siguientes niveles de investigación:

- Aplicativo: En este nivel de investigación se controla el nivel de riego del cultivo, calibrando los sensores para su óptimo desempeño.
- Explicativo: Se puede demostrar la eficiencia del prototipo desarrollado en esta investigación al compararlo con sistemas de riego complejos.
- Descriptivo: En el parámetro descriptivo se obtiene el promedio de los sensores y se puede estimar un comportamiento promedio mediante la toma de datos previos.
- Exploratorio: Se logró identificar el problema de excesos de riego dentro de los invernaderos sin automatización, de la misma manera se obtiene un control y monitoreo del invernadero.

Además de los niveles de investigación señalados se obtuvo datos

- Transversales: Se obtuvo un muestreo de datos, el universo de datos tomados por el prototipo durante un mes.
- Longitudinales: Se obtuvo una medición del cultivo y su crecimiento al pasar un mes de uso del prototipo.

El prototipo de control para el invernadero comprende en realizar una serie de pasos:

- Inicialmente, se realizó el sistema de control mediante microcontrolador, realizando simulaciones en el software Proteus, verificando la funcionalidad de cada uno de los sensores y actuadores.

- Luego de realizar las correspondientes simulaciones, se realizó el armado del prototipo en un tablero de trabajo realizando las pruebas y correcciones para la lectura de cada uno de los sensores.
- Concluyendo con las pruebas en el tablero de trabajo con el microcontrolador, se procede a realizar la respectiva programación dentro de Python conjuntamente con una Raspberry Pi.
- Obtenida la comunicación entre los dos sistemas se procedió a realizar el diseño de la plataforma de control conjunto, para el cual se procedió a utilizar PYQT Designer; el cual permite crear interfaces amigables para el usuario.
- Finalizada la comunicación serial entre el microcontrolador (Arduino mega) y Raspberry Pi (PYQT). Se realizó el diseño para la instalación del sistema de control dentro del invernadero.

3.1 Sistema de control mediante Microcontrolador - Arduino.

En este apartado se tratará sobre la instalación de cada uno de los sensores en el microcontrolador (Arduino). Para esto, se utiliza el software Proteus el cual permite simular cada uno de los componentes a utilizar en el prototipo de automatización. Como una observación dentro de las simulaciones, los sensores no se conectan directamente, debido a que en el circuito final se incluye diversos actuadores; los cuales serán alimentados por fuentes externas.

3.1.1 Sensor de temperatura y humedad (DHT22)

El sensor DHT22 consta de 3 pines: Vcc: 5v, Gnd y Salida digital su conexión se muestra en la Figura 15.

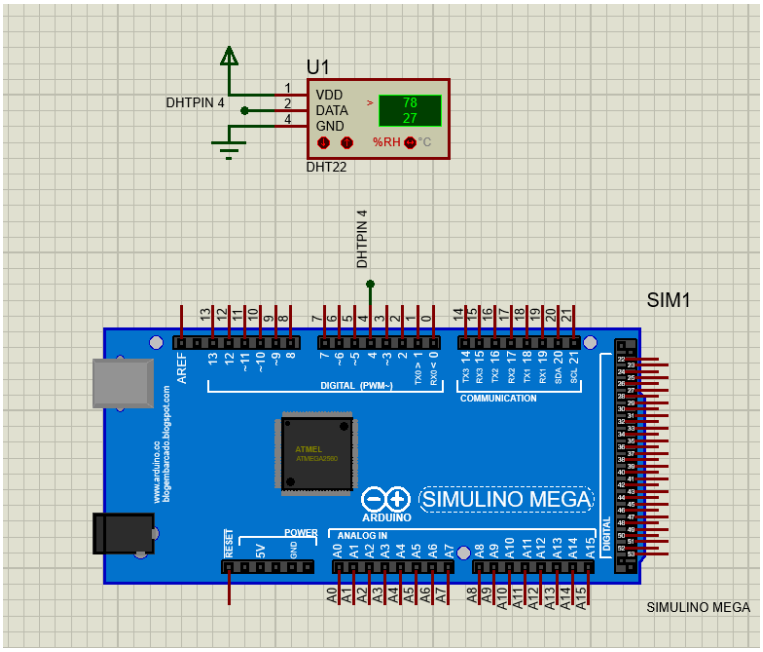


Figura 15: Conexión del sensor DHT22

Fuente: Autor

Este sensor no necesita de ninguna calibración previa, el código (sketch) se muestra en la Figura 16. Para su funcionamiento, es necesario la descarga de la librería para los sensores DHT.h, el cual tiene un funcionamiento tanto para el sensor DHT11 como para el sensor DHT22.

DHT22

```
#include "DHT.h"
#define DHTPIN 4
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  dht.begin();
}

void loop() {
    delay(2000);
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  float f = dht.readTemperature(true);
  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }
  float hif = dht.computeHeatIndex(f, h);
  float hic = dht.computeHeatIndex(t, h, false);
  Serial.print(F("Humidity: "));
  Serial.print(h);
  Serial.print(F("%  Temperature: "));
  Serial.print(t);
  Serial.print(F("°C "));
  Serial.print(f);
  Serial.print(F("°F  Heat index: "));
  Serial.print(hic);
  Serial.print(F("°C "));
  Serial.print(hif);
  Serial.println(F("°F"));
}
```

Figura 16: Código de programación Dht22

Fuente: Autor

3.1.2 Sensor humedad de suelo

De la misma manera, el sensor de humedad de suelo tiene 3 pines de conexión: Vcc(5v), Gnd y señal analógica. Su conexión se muestra en la Figura 17. Este sensor se encuentra representado por resistencias variables, debido a que el sistema propuesto se encuentra conformado por 12 sensores de humedad de suelo los cuales irán ubicados en las mangas; donde se encuentra el cultivo de frutillas.

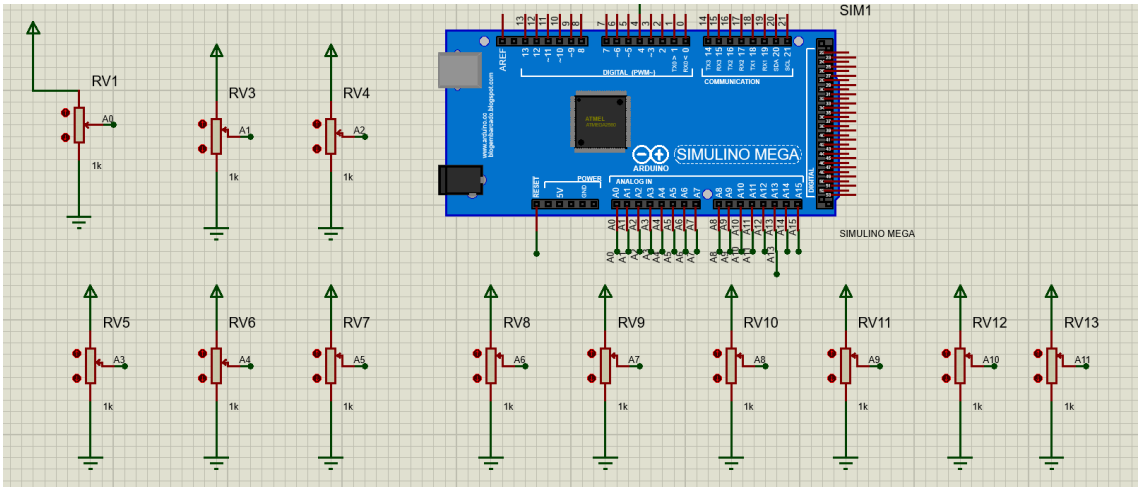


Figura 17: Conexión del sensor de humedad de suelo.

Fuente: Autor

A diferencia del sensor DHT22, este sensor necesita ser calibrado; cuando el sensor está seco o al aire y cuando el sensor se encuentra completamente sumergido en agua.

Al ejecutar el código de la Figura 18 en un ambiente seco, se obtendrá una lectura de valores estableciendo un valor estable, determinando un valor cuando el sensor se encuentra seco, de la misma manera se determina el valor para cuando el sensor se encuentra completamente húmedo.

La calibración solamente es necesaria con un solo sensor de humedad de suelo; pero para una calibración efectiva, se debe calibrar todos los sensores conectados, debido a la variación de voltaje que se produce al conectar el resto de sensores y actuadores.

calibraciondesensorsuelo

```
#define AOUT 0 // Pin analógico "A0" para conectar la salida del sensor de humedad capacitivo

int valor_sensor; // Variable que almacena el valor de salida del sensor de humedad capacitivo


void setup()
{
  Serial.begin(9600); // Comienzo de la comunicación con el monitor serie del IDE de Arduino
}


void loop()
{
  // Leemos el valor de la salida analógica del sensor capacitivo, conectada al pin analógico "A0"
  valor_sensor = analogRead(AOUT);

  // Mostramos el valor de la salida analógica del sensor capacitivo a través del monitor serie
  Serial.print("Valor del sensor de humedad capacitivo: ");
  Serial.println(valor_sensor);

  delay(1000); // Retardo de un segundo entre lecturas del sensor de humedad capacitivo
}
```

Figura 18: Código para la lectura de valores promedio (Seco, Sumergido).

Fuente: Autor

Luego de realizar las conexiones pertinentes, obtenido los valores promedios cuando el sensor se encuentra seco, y determinando el valor cuando el sensor se encuentra sumergido en líquido obtenido en la Figura 18, a continuación; se carga el código en nuestro microcontrolador, el código de lectura para la humedad del sensor ver Figura 19.

humedadsuelo §

```
#define AOUT 0 // Pin analógico "A0" para conectar la salida del sensor de humedad capacitivo

const int Valor_Sensor_Aire = 632;// Valor calculado con el programa de calibración con el sensor al aire
const int Valor_Sensor_Agua = 280; // Valor calculado con el programa de calibración con el sensor sumergido en agua

int valor_sensor = 0; // Variable que almacena el valor de salida del sensor de humedad capacitivo
int porcentaje = 0; // Variable que almacena el porcentaje de humedad relativa del terreno


void setup()
{
  Serial.begin(9600); // Comienzo de la comunicación con el monitor serie del IDE de Arduino
}


void loop()
{
  // Leemos el valor de la salida analógica del sensor capacitivo, conectada al pin analógico "A0"
  valor_sensor = analogRead(AOUT);

  // Se calcula el porcentaje de humedad relativa teniendo en cuenta los dos límites
  porcentaje = map(valor_sensor, Valor_Sensor_Agua, Valor_Sensor_Aire, 100, 0);

  if(porcentaje < 0) porcentaje = 0; // Evita porcentajes negativos en la medida del sensor

  if(porcentaje > 100) porcentaje = 100; // Evita porcentajes negativos en la medida del sensor

  // Se presenta el porcentaje a través del monitor serie
  Serial.print("HUMEDAD: ");
  Serial.print(porcentaje);
  Serial.println("% HR");


  // Se presentan los mensajes, a través del monitor serie, dependiendo del porcentaje de humedad relativa
  if(porcentaje <= 33)
  {
    Serial.println("Suelo seco !!!");
  }

  if(porcentaje > 33 && porcentaje <= 66)
  {
    Serial.println("Suelo humedo !!!");
  }

  if(porcentaje > 66)
  {
    Serial.println("Suelo con exceso de humedad !!!");
  }

  Serial.println(" ");

  delay(1000); // Retardo de un segundo entre lecturas del sensor de humedad capacitivo
}
```

Figura 19: Código de lectura para el sensor de humedad de suelo.

Fuente: Autor

3.1.3 Sensor de temperatura DS18B20

A diferencia de los sensores anteriores, este necesita se energizado de manera normal pero su línea de datos necesita conectarse a una resistencia de 4.7k y su salida

a Vcc. Esto se lo realiza para evitar valores falsos estabilizando el voltaje en grandes distancias, ver Figura 20.

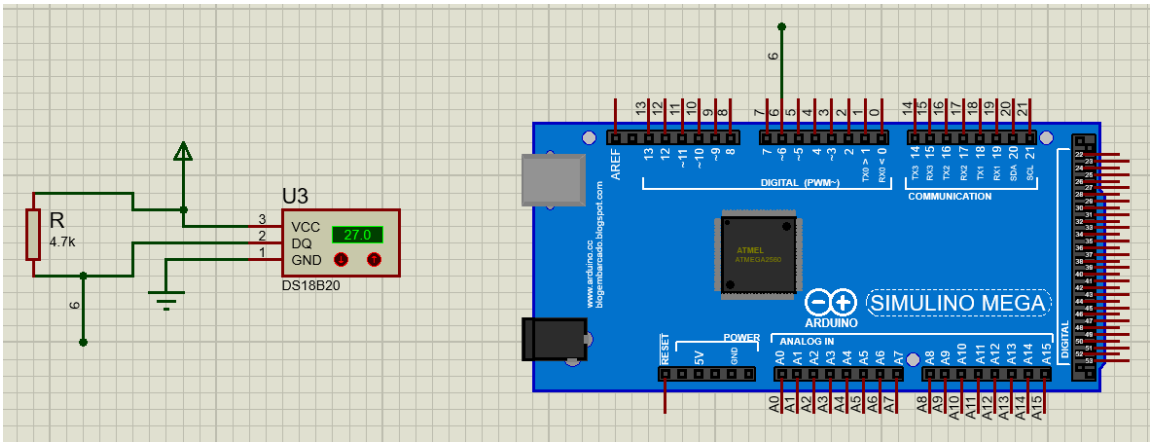


Figura 20: Conexión del sensor de temperatura DS18B20.

Fuente: Autor

Para la ejecución de este sensor, se necesitan las librerías OneWire.h y DallasTemperature.h, como se muestra en la Figura 21, este sensor de temperatura no necesita ninguna calibración previa.

DS18B20

```
#include <OneWire.h>
#include <DallasTemperature.h>

OneWire ourWire(2);                //Se establece el pin 2  como bus OneWire

DallasTemperature sensors(&ourWire); //Se declara una variable u objeto para nuestro sensor

void setup() {
delay(1000);
Serial.begin(9600);
sensors.begin();   //Se inicia el sensor
}

void loop() {
sensors.requestTemperatures();   //Se envía el comando para leer la temperatura
float temp= sensors.getTempCByIndex(0); //Se obtiene la temperatura en °C

Serial.print("Temperatura= ");
Serial.print(temp);
Serial.println(" C");
delay(100);
}
```

Figura 21: Código de lectura para el sensor de temperatura DS18B20.

Fuente: Autor

3.1.4 Modulo Ultrasónico HC-SR04

El módulo ultrasónico es un dispositivo que utiliza ondas sonoras de alta frecuencia para medir distancias, conformado por un transmisor ultrasónico y un receptor ultrasónico, los cuales se encuentran en el mismo dispositivo, su conexión es simple conformado por 4 pines: Vcc , Gnd , Trigger (Transmisor) , Echo (Receptor); ver Figura 22.

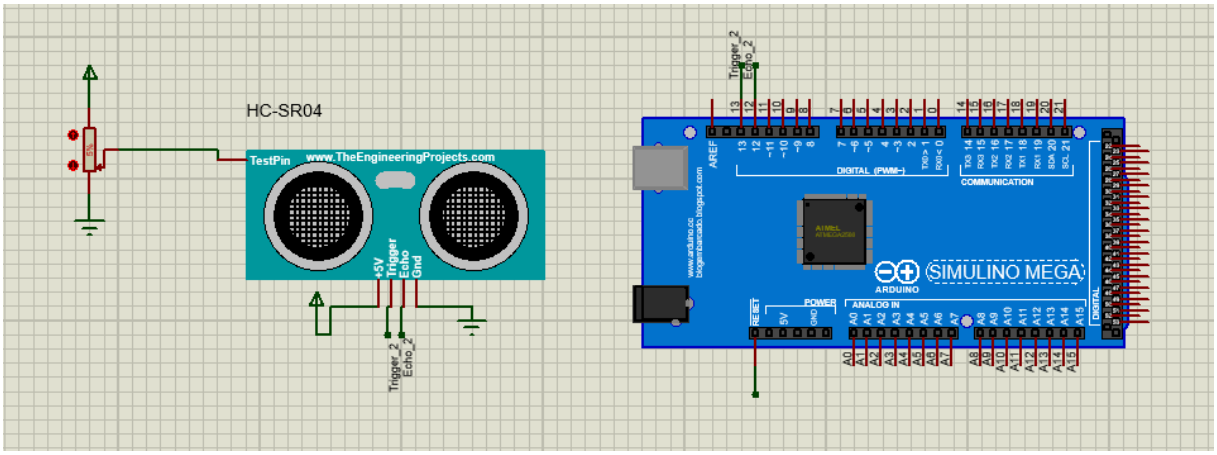


Figura 22: Conexión del Módulo sumergible ultrasónico HC-SR04.

Fuente: Autor

Este sensor necesita de 2 pines digitales para funcionar en donde el primer pin 13 es Trigger y el segundo pin 12 seria Echo. Una vez realizada la conexión, el código para la ejecución se encuentra en la Figura 23.

Figura 23: Código de lectura módulo sumergible ultrasónico SR04M-2.

Fuente: Autor

3.1.5 Modulo ultrasónico SR04M-2

Este módulo a diferencia del tradicional se caracteriza por su resistencia al agua con una protección IP 66 (protección total contra el polvo; protección contra chorros de agua en cualquier dirección), pero cabe enfatizar que no se puede sumergir. La conexión del sensor se detalla en la Figura 24.

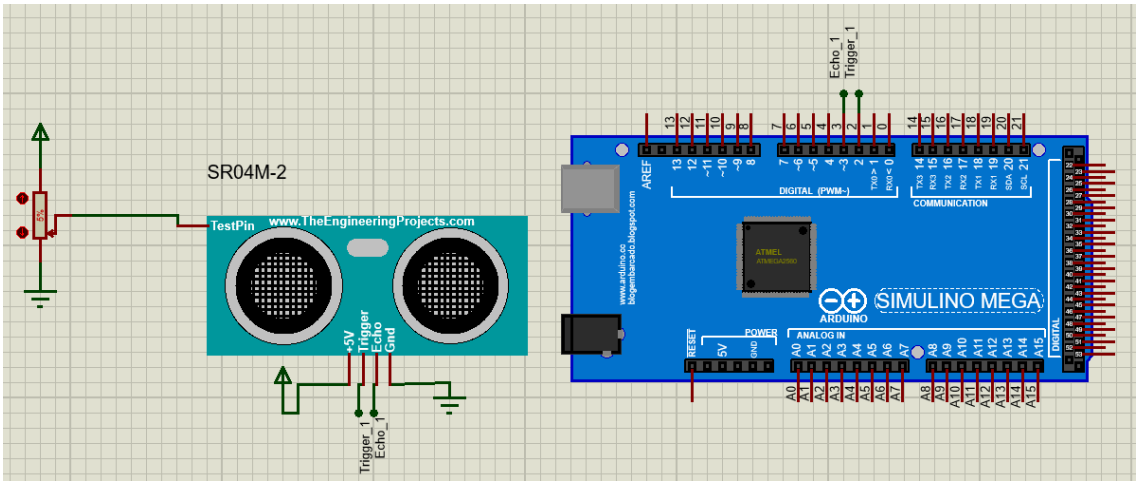


Figura 24: Conexión del Módulo ultrasónico SR04M-2.

Fuente: Autor

Este sensor necesita de 2 pines digitales para su funcionamiento Trigger en el pin 12 ,Echo en el pin 13 y 2 pines de alimentación Vcc y Gnd. ver Figura 25.

Ultrasonicosumergible

```
const unsigned int TRIG_PIN=13; //RX
const unsigned int ECHO_PIN=12; // TX
const unsigned int BAUD_RATE=9600;

void setup() {
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  Serial.begin(BAUD_RATE);
}

void loop() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  const unsigned long duration= pulseIn(ECHO_PIN, HIGH);
  int distance= duration/29/2;
  if(duration==0){
    Serial.println("Warning: no pulse from sensor");
  } else {
    Serial.print("distance to nearest object:");
    Serial.println(distance);
    Serial.println(" cm");
  }
  delay(1000);
}
```

Figura 25: Código de lectura para el modulo sumergible ultrasónico SR04M-2.

Fuente: Autor

3.1.6 Sensor de CO_2

El sensor de CO_2, se encarga de la lectura de dióxido de carbono en el ambiente. La lectura se encuentra representada en ppm. Para su correcto funcionamiento es necesario que el sensor se caliente por determinado tiempo. La conexión del sensor de CO_2 se representa en la Figura 26.

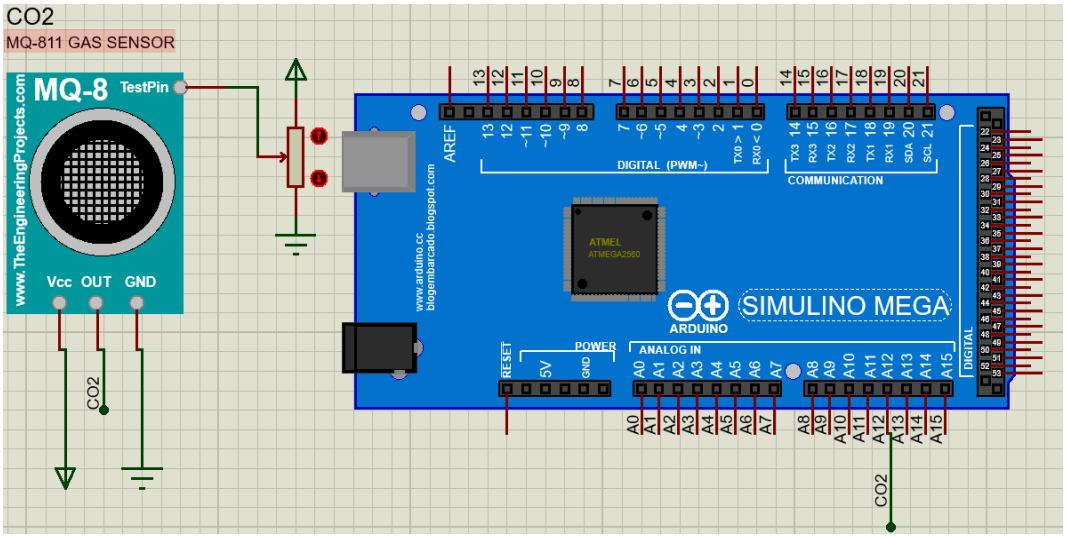


Figura 26: Conexión del sensor de CO2.

Fuente: Autor

Este sensor necesita ser calibrado para su funcionamiento, lo cual se realiza con la librería CO2sensor.h para el módulo MG-811 (Figura 27).

ReadCO2

```
#include "CO2Sensor.h"

CO2Sensor co2Sensor(A0, 0.99, 100);

void setup() {
  Serial.begin(9600);
  Serial.println("=== Initialized ===");
  co2Sensor.calibrate();
}

void loop() {
  int val = co2Sensor.read();
  Serial.print("CO2 value: ");
  Serial.println(val);

  delay(1000);
}
```

Figura 27: Código de lectura para el sensor de CO2.

Fuente: Autor

3.1.7 Modulo reloj y Almacenamiento

Se agregó un módulo reloj RTC DS3231 y un lector de tarjetas secure digital (SD), los cuales sirven para el control de tiempos, y para evitar que la programación varie al sufrir algún tipo de reinicio inesperado.

De la misma manera el lector SD, permite el almacenamiento de datos que brindan los sensores evitando perdida de información. En la Figura 28 , se indica el diagrama de conexión del módulo reloj RTC y del módulo lector micro SD.

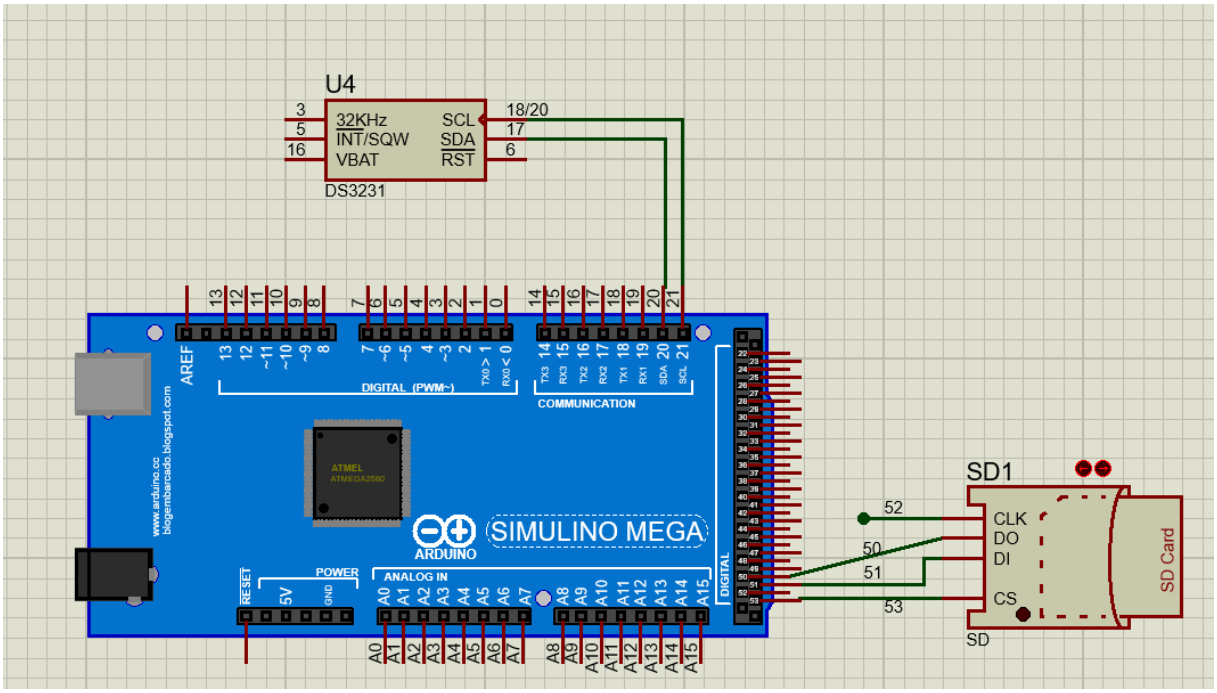


Figura 28: Conexión de modulo reloj y lector SD.

Fuente: Autor

Se realiza una prueba de todos los sensores de manera conjunta para corregir errores y confirmar que todos los sensores se encuentren funcionado de manera correcta, de la misma manera esta sección permite calibrar de manera temporal los sensores, debido a que algunos sensores pueden tener caídas de tensión dentro del circuito real ya instalado en el invernadero. Ver Figura 29.

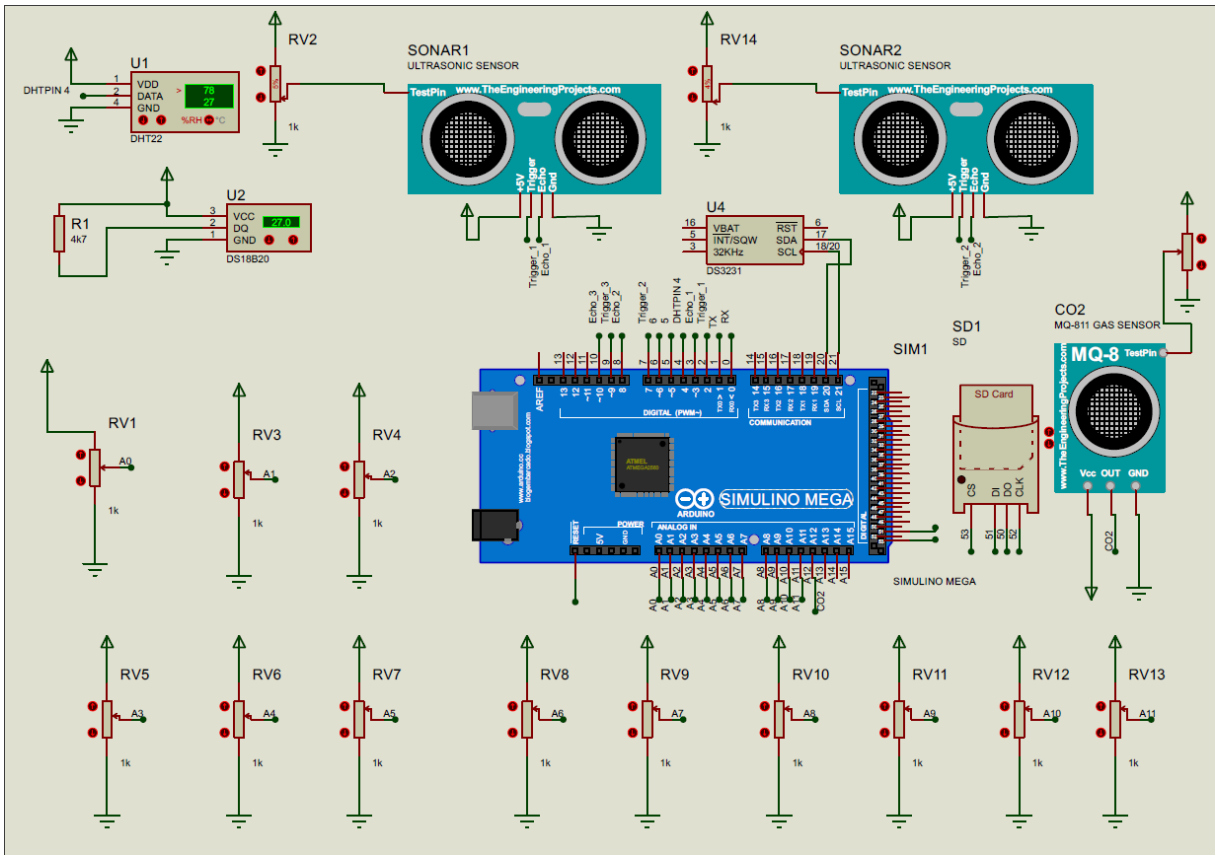


Figura 29: Prueba de sensores utilizando Arduino.

Fuente: Autor

3.2 Comunicación serial (Raspberry Pi y Arduino)

Para realizar la comunicación entre el mini ordenador y el microcontrolador, se realiza de la siguiente manera (Figura 30).

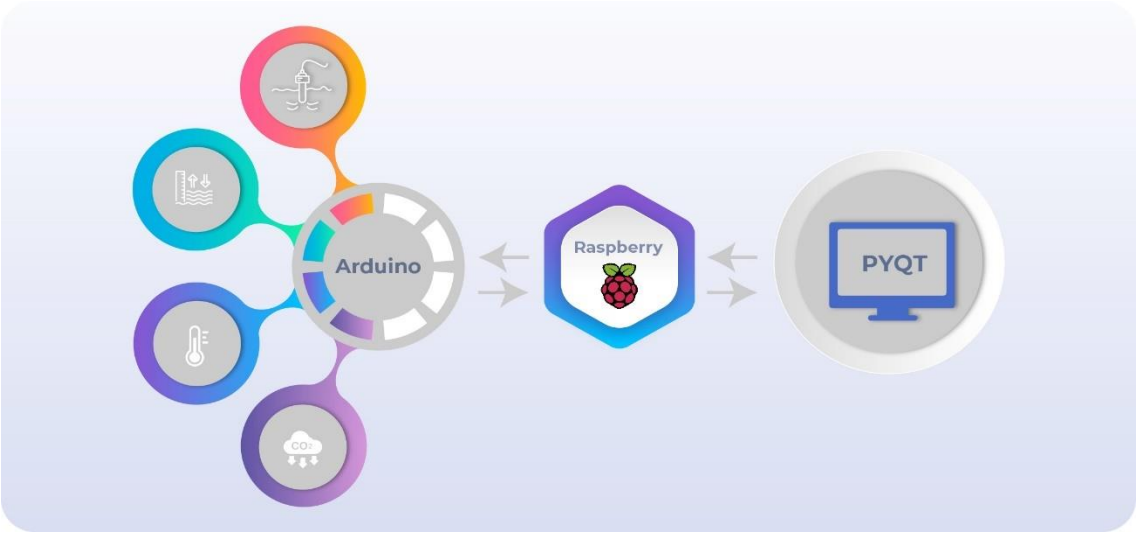


Figura 30: Diagrama de comunicación del prototipo de automatización para el invernadero semi hidropónico.

Fuente: Autor

Como se evidencia en la Figura 30, el enlace se realiza mediante puerto serial entre la Raspberry y el Arduino, cuyos datos se presentan mediante una pantalla táctil, esta pantalla es la encargada de enviar comandos hacia el Arduino, el cual se encarga del accionamiento de actuadores como por ejemplo la iluminación del invernadero. Este control de doble vía se plantea mediante comandos que trabajan en paralelo explicándolo mejor en la Figura 31.

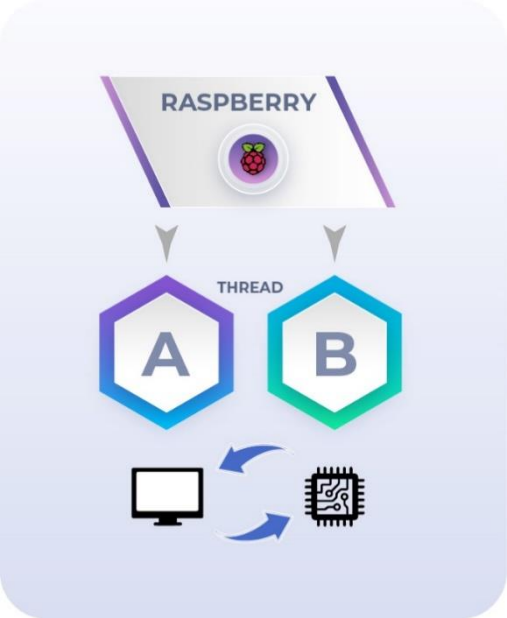


Figura 31: Diagrama de comunicación doble vía para el prototipo.

Fuente: Autor

Entonces para la comunicación tanto del micro ordenador como del microcontrolador fue necesario utilizar sub procesos (THREAD), de esta manera se pudo trabajar de manera simultánea dividiendo el código de la programación en 3 secciones, las cuales se detallan a continuación:

- Transferencia de datos desde el Arduino a la Raspberry Pi.
- Trasferencia de datos desde la Raspberry Pi a la pantalla.
- Transferencia de datos desde la pantalla a Arduino.

3.2.1 Transferencia de datos de Arduino a la Raspberry Pi

Para la transferencia de datos, lo primero que se hizo fue buscar la dirección del puerto serial del Arduino, ingresando al programador del Raspberry Pi ver Figura 32. Para esto, se utilizó el siguiente comando:

dmesg | grep -v disconnect | grep -Eo "tty(ACM|USB)." | tail -1

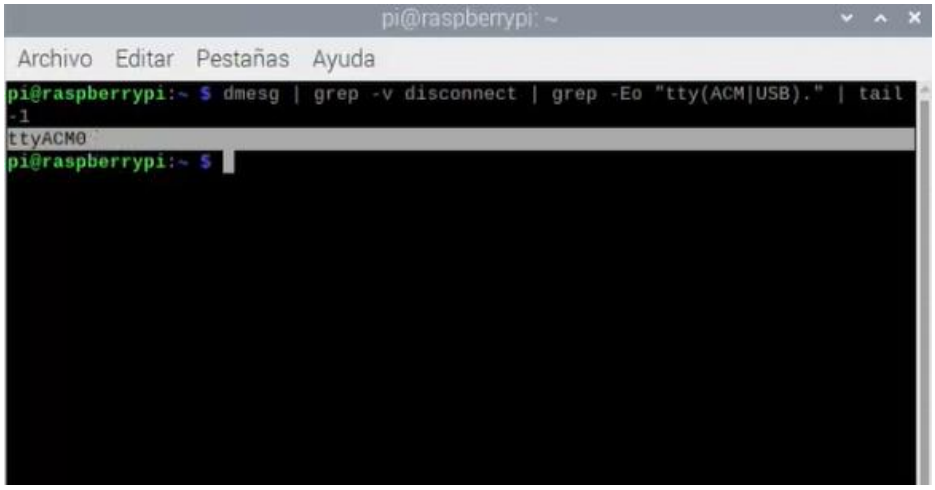


Figura 32: Nombre del puerto serial.

Fuente: Autor

Luego de conocer el nombre del puerto serial, se ajustó la velocidad de transmisión de datos a 9600 baudios. Ver Figura 33.

```
long randomNumber;

void setup(){
  //iniciamos el puerto serie
  Serial.begin(9600); // velocidad de transmisión

}

void loop(){
  randomNumber = random(777,778);
  Serial.println(randomNumber);
}
```

Figura 33: Código prueba.

Fuente: Autor

Se realizó una prueba y se observa que se puede transferir información. Se envía una secuencia de números aleatorios entre 777 y 778 (ver Figura 34).

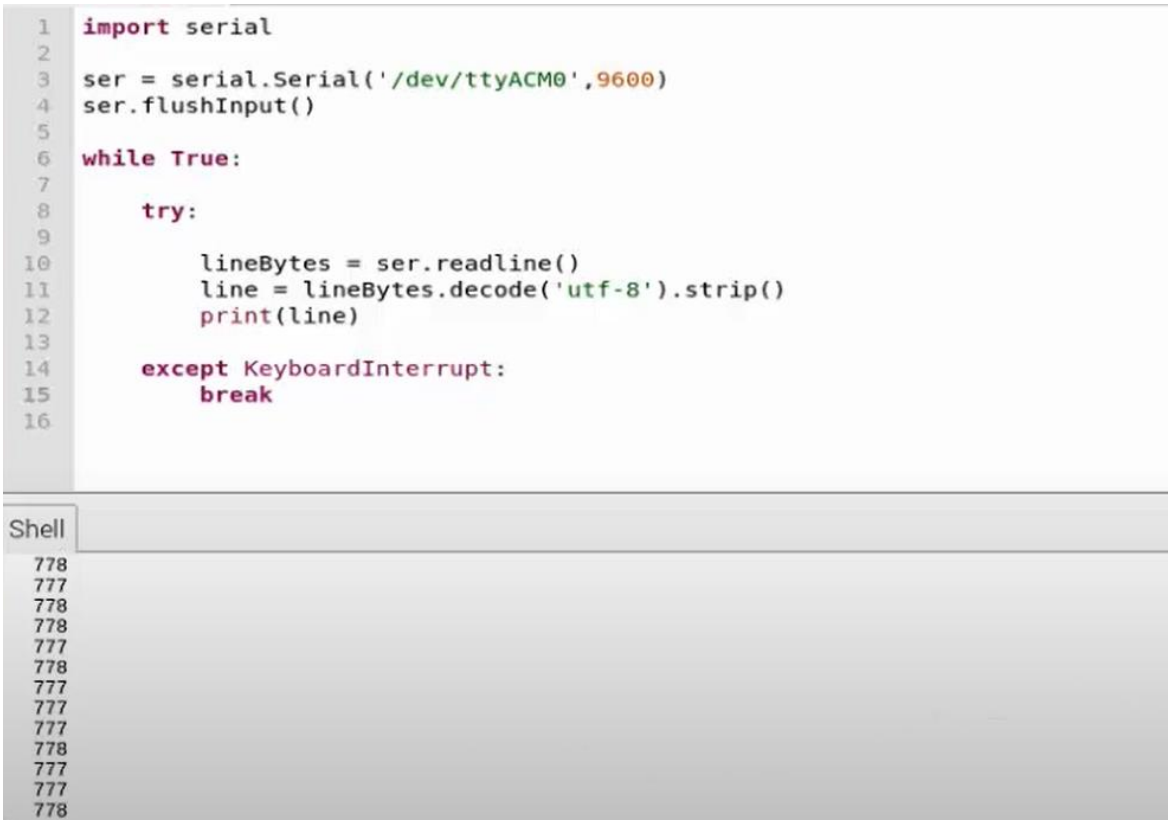
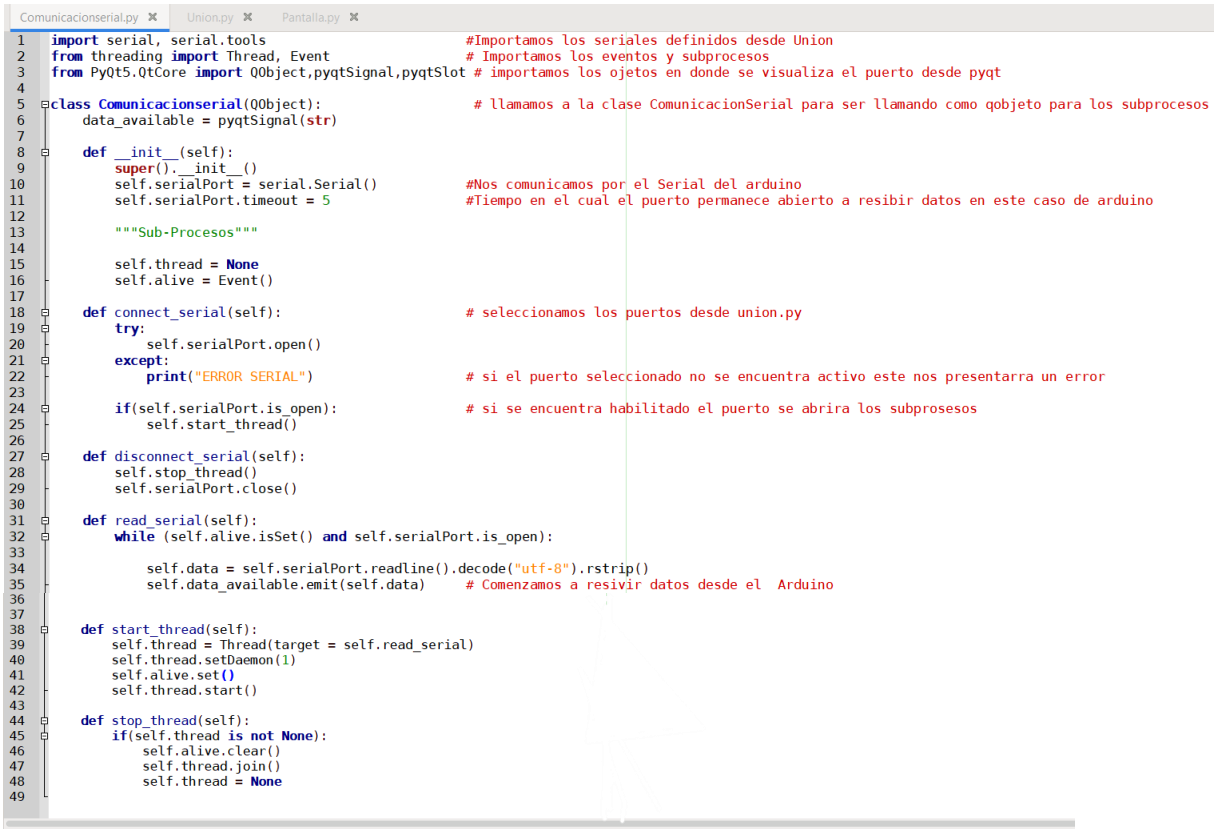


Figura 34: Prueba de puerto y velocidad de transmisión.

Fuente: Autor

Posterior al obtener el nombre del puerto de Arduino Mega, se realizó el primer código para la comunicación serial del arduino ver Figura 35.



```python
import serial, serial.tools                         #Importamos los seriales definidos desde Union
from threading import Thread, Event                 # Importamos los eventos y subprocesos
from PyQt5.QtCore import QObject,pyqtSignal,pyqtSlot # importamos los ojetos en donde se visualiza el puerto desde pyqt

class Comunicacionserial(QObject):                   # llamamos a la clase ComunicacionSerial para ser llamando como qobjeto para los subprocesos
    data_available = pyqtSignal(str)

    def __init__(self):
        super().__init__()
        self.serialPort = serial.Serial()           #Nos comunicamos por el Serial del arduino
        self.serialPort.timeout = 5                 #Tiempo en el cual el puerto permanece abierto a resibir datos en este caso de arduino

        """Sub-Procesos"""

        self.thread = None
        self.alive = Event()

    def connect_serial(self):                       # seleccionamos los puertos desde union.py
        try:
            self.serialPort.open()
        except:
            print("ERROR SERIAL")                   # si el puerto seleccionado no se encuentra activo este nos presentarra un error

        if(self.serialPort.is_open):                # si se encuentra habilitado el puerto se abrira los subprosesos
            self.start_thread()

    def disconnect_serial(self):
        self.stop_thread()
        self.serialPort.close()

    def read_serial(self):
        while (self.alive.isSet() and self.serialPort.is_open):

            self.data = self.serialPort.readline().decode("utf-8").rstrip()
            self.data_available.emit(self.data)     # Comenzamos a resivir datos desde el  Arduino


    def start_thread(self):
        self.thread = Thread(target = self.read_serial)
        self.thread.setDaemon(1)
        self.alive.set()
        self.thread.start()

    def stop_thread(self):
        if(self.thread is not None):
            self.alive.clear()
            self.thread.join()
            self.thread = None
```

Figura 35: Código thread para comunicación serial.

Fuente: Autor

El primer código dentro de la Raspberry Pi es específicamente enfocado a la transmisión de datos, pero no es tan simple debido a que se necesita también la comunicación con la pantalla táctil.

Es por ello que se utilizó los subprocesos, este código se enfoca en habilitar el puerto cumpliendo la función de informar si el puerto se encuentra habilitado o

deshabilitado, de ser el caso en el que el puerto este deshabilitado imprimirá un mensaje de error.

Este subproceso es el más importante debido a que permite la comunicación entre el microcontrolador y el micro ordenador, importando los valores de port y baud desde la segunda parte de la comunicación que es la unión de los sistemas.

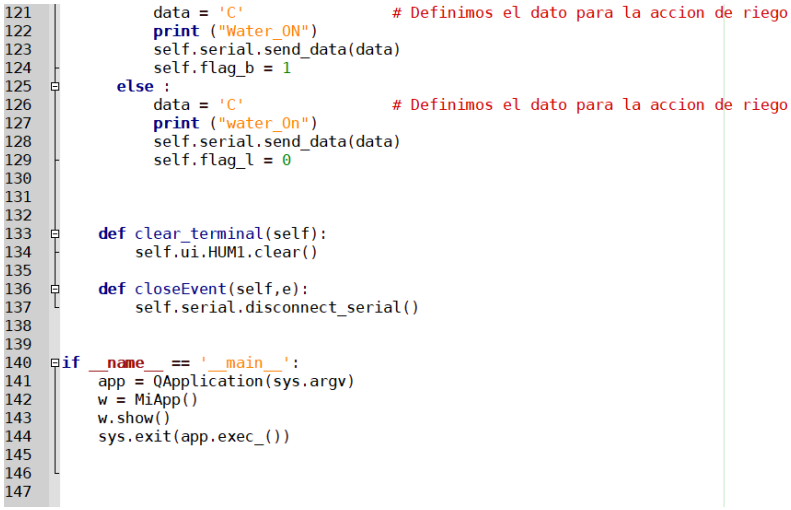
3.2.2 Código puente para la comunicación Arduino pantalla

Esta sección de la programación, se enfoca en crear un puente llamado variables, desde Arduino y transmitiendo dichas variables hacia la interfaz gráfica, lo cual permite interactuar con la pantalla ver Figura 36.

Union.py

```
import sys
from PyQt5.QtWidgets import QMainWindow,QApplication, QTimeEdit # Importamos los archivos pyqt desde pantalla.py,
from GUI import *                                               #importamos pantalla.py
from customSerial import customSerial                           # importamos  Comunicacionserial.py
from PyQt5.QtCore import QTimer,QTime                           # importamos  subprocesos de pyqt como wigets de tiempo
import datetime
import subprocess

class MiApp(QMainWindow):                                       # llamamos a la clase QmainWindow desde PYQT
    def __init__(self):
        super().__init__()
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)
        self.ls = []
        self.flag_l = 0 #Seleccionamos un valor inicial para el pulsante que servira para encender la iluminacion
        self.flag_b = 0 #Seleccionamos un valor inicial para el pulsante que servira para encender el riego del invernadero manual


        #Serial

        self.serial = customSerial()
        self.update_ports()
        port = '/dev/ttyACM0'                                   #Selecionamos el Puerto del arduino
        baud = 9600                                             #Seleccionamos la velocidad de comunicacion de Arduino
        self.serial.serialPort.port = port
        self.serial.serialPort.baudrate = baud
        self.serial.connect_serial()

        #Events
        self.serial.data_available.connect(self.update_HUM1)

        self.ui.BOTILUM.clicked.connect(self.send_data_l) #importamos los botones de accion para encender la iluminacion
        self.ui.BOTBOM.clicked.connect(self.send_data_b)  #importamos los botones de accion para encender el riego

        self.timeEdit_1 = self.findChild(QTimeEdit, "timeEdit_1")#importamos los editores de tiempo desde pyqt
        self.timeEdit_2 = self.findChild(QTimeEdit, "timeEdit_2")#importamos los editores de tiempo desde pyqt
        self.timeEdit_3 = self.findChild(QTimeEdit, "timeEdit_3")#importamos los editores de tiempo desde pyqt
        self.timer = QTimer()
        self.timer.timeout.connect(self.check_time)     # Realiza un tiempo de accion estilo temporizador
        self.timer.start(1000)                          # Comprueba cada segundo
```

```
41
42      def check_time(self):
43          current_time = QTime.currentTime()              #comprobamos la hora del sistema
44
45          if current_time.toString() == self.timeEdit_1.time().toString(): #si la hora del sistema es igual a la hora de los timeEdit
46
47                  data = 'C'          #enviamos el dato C que es el dato de accion para el riego
48                  print ("Water_ON")
49                  self.serial.send_data(data)
50                  self.flag_l = 0
51
52          if current_time.toString() == self.timeEdit_2.time().toString():
53                  data = 'C'
54                  print ("Water_ON")
55                  self.serial.send_data(data)
56                  self.flag_l = 0
57
58          if current_time.toString() == self.timeEdit_3.time().toString():
59                  data = 'C'
60                  print ("Water_ON")
61                  self.serial.send_data(data)
62                  self.flag_l = 0
63
64
65      def update_HUM1(self,data):                     # Enviamos los datos desde pyqt
66          self.ls = data.split(";")
67          print (data)
68          try:
69            self.v_HUM = self.ls[0]               #Estos datos serian enviados en un orden especifico desde arduino consiguiendo una separacion con ;
70            self.v_TEMP = self.ls[1]              # Determinando haci la posicicon de cada uno de los sensores
71            self.v_ULTRASUMER = self.ls[2]        #determinando cada uno dentro de una pocicion en la ventana en pyqt
72            self.v_TEMPSUMER = self.ls[3]
73            self.v_HUM1 = self.ls[4]
74            self.v_HUM2 = self.ls[5]
75            self.v_HUM3 = self.ls[6]
76            self.v_HUM4 = self.ls[7]
77            self.v_HUM5 = self.ls[8]
78            self.v_HUM6 = self.ls[9]
79            self.v_HUM7 = self.ls[10]
80            self.v_HUM8 = self.ls[11]
81              self.v_HUM8 = self.ls[11]
82              self.v_HUM9 = self.ls[12]
83              self.v_HUM10 = self.ls[13]
84              self.v_HUM11 = self.ls[14]
85              self.v_HUM12 = self.ls[15]
86              self.v_CO2 = self.ls[16]
87              self.ui.HUM.setText(self.ls[0])
88              self.ui.TEMP.setText(self.ls[1])
89              self.ui.ULTRASUMER.setText(self.ls[2])
90              self.ui.TEMPSUMER.setText(self.ls[3])
91              self.ui.HUM1.setText(self.ls[4])
92              self.ui.HUM2.setText(self.ls[5])
93              self.ui.HUM3.setText(self.ls[6])
94              self.ui.HUM4.setText(self.ls[7])
95              self.ui.HUM5.setText(self.ls[8])
96              self.ui.HUM6.setText(self.ls[9])
97              self.ui.HUM7.setText(self.ls[10])
98              self.ui.HUM8.setText(self.ls[11])
99              self.ui.HUM9.setText(self.ls[12])
100             self.ui.HUM10.setText(self.ls[13])
101             self.ui.HUM11.setText(self.ls[14])
102             self.ui.HUM12.setText(self.ls[15])
103             self.ui.CO2.setText(self.ls[16])
104           except:
105             pass
106
107
108     def send_data_l(self):
109       if self.flag_l == 0:
110           data = 'A'                  #Se define el dato para aacionar la iluminacion
111           print ("Light_ON")
112           self.serial.send_data(data)
113           self.flag_l = 1
114       else:
115           data = 'B'                  # Definimos el dato para apagar la iluminacion
116           print ("Light_OFF")
117           self.serial.send_data(data)
118           self.flag_l = 0
119
120     def send_data_b(self):
        if self.flag_b == 0:
```



```
            data = 'C'                # Definimos el dato para la accion de riego
            print ("Water_ON")
            self.serial.send_data(data)
            self.flag_b = 1
        else :
            data = 'C'                # Definimos el dato para la accion de riego
            print ("water_On")
            self.serial.send_data(data)
            self.flag_l = 0



    def clear_terminal(self):
        self.ui.HUM1.clear()

    def closeEvent(self,e):
        self.serial.disconnect_serial()


if __name__ == '__main__':
    app = QApplication(sys.argv)
    w = MiApp()
    w.show()
    sys.exit(app.exec_())

```

Figura 36: Código puente para la comunicación entre Arduino y la pantalla.

Fuente: Autor

De esta manera, se logró conectar o crear un puente entre los dos códigos, inicialmente llamando al código de la pantalla el cual funciona con PYQT5. Se importó el archivo llamado pantalla.py y el archivo comunicacionserial.py. Posteriormente, se procedió a declarar el nombre de las variables para la ejecución de los actuadores.

Dentro del código puente, se escogió los valores del puerto y la velocidad de transmisión de datos el cual se utiliza para el enlace con el microcontrolador (Arduino), de la misma manera como el código anterior, este código se encuentra formado por subprocesos o hilos, lo cual permite tener funcionando de manera independiente los códigos por separado y unirlos cuando sea necesario, declarando datos desde la interfaz de acuerdo al número de espacios en el monitor serial del Arduino, Entonces con este orden, se obtiene la visualización de los datos; de esta manera también, se incluye el envido de variables desde la pantalla hacia el Arduino. Esto permitió controlar la hora del riego de las frutillas, se realizó de manera sencilla evitando ingresar al código fuente de Arduino. Es importante aclarar que, todos estos archivos

deben encontrarse en la misma ubicación dentro de la carpeta de la Raspberry Pi para que su ejecución sea exitosa.

3.3 Creación de la interfaz para la pantalla.

Para crear el diseño de la interfaz gráfica, se utilizó el programa de diseño QT Designer; el cual permite crear de manera sencilla interfaces visuales. La pantalla inicial del programa, se muestra en la Figura 37.

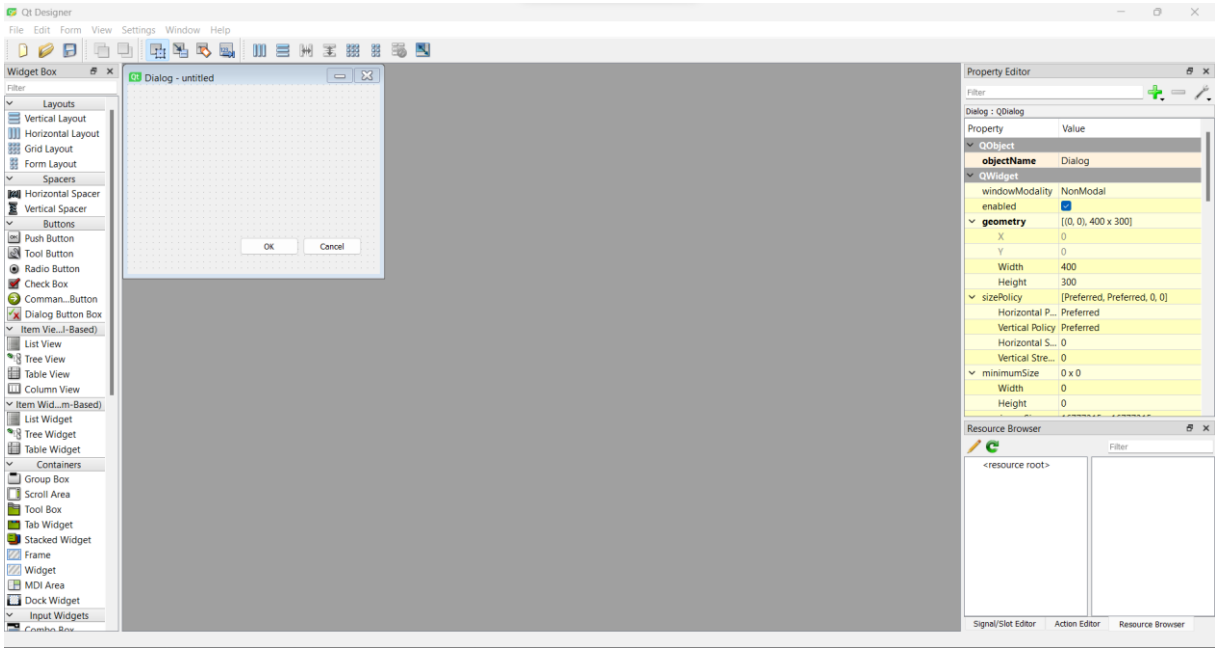


Figura 37: Pantalla de inicio de QT Designer.

Fuente: Autor

La gran ventaja que ofrece Qt Designer es que permite colocar directamente imágenes dentro de su interfaz, para esto es necesario incluir la imagen dentro de la clase correspondiente. Se inicia un nuevo archivo y abre la pantalla central como se observa en la

Figura 38.

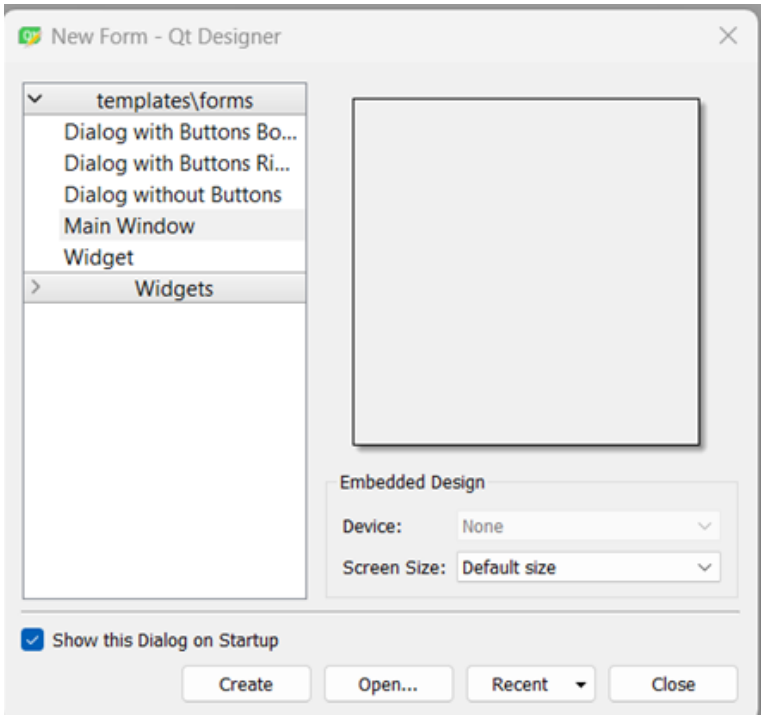


Figura 38: Pantalla central.

Fuente: Autor

Se crea una ventana principal llamada Main Windows, con este nombre se exporta al código puente para la lectura de la interfaz en la ventana creada, ya con la pantalla principal se dirige a la barra de herramientas donde se encuentra un widget llamado en donde se inserta la imagen (Figura 39) de la interfaz deseada, al ubicar el widget se ingresa a este como se puede observar en la Figura 40.

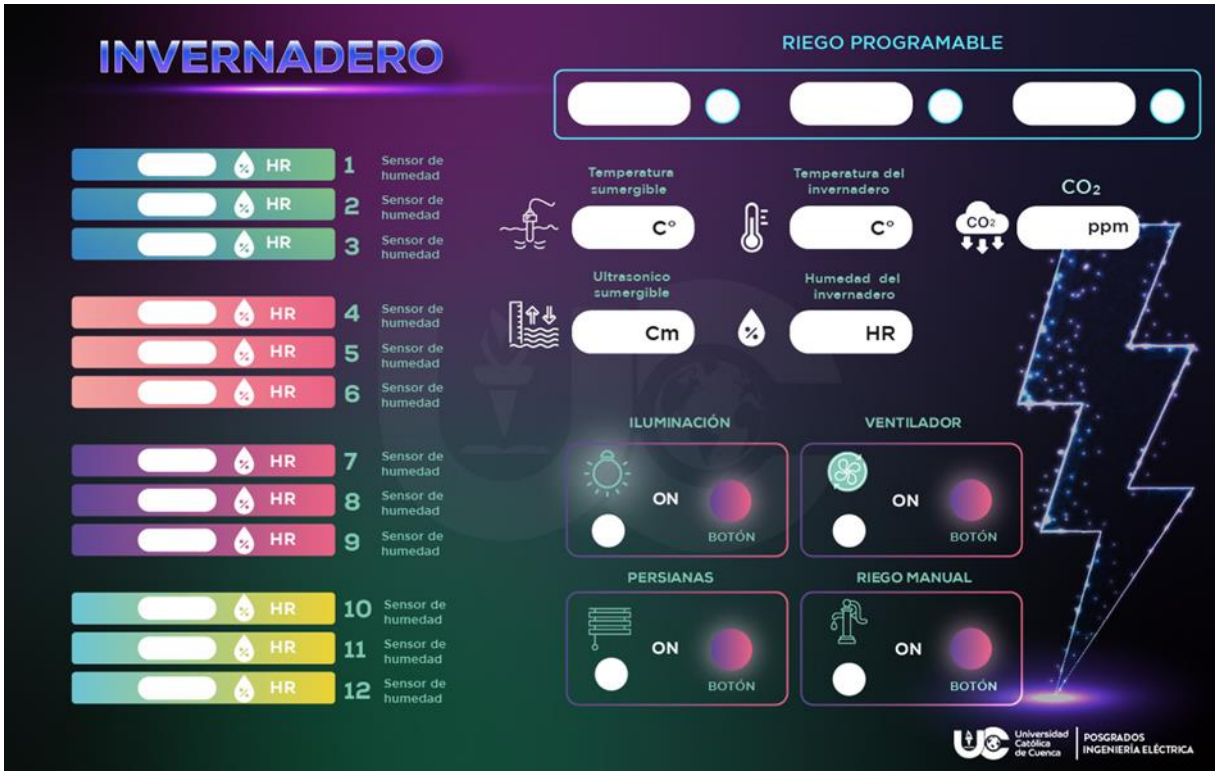


Figura 39: Imagen para insertar en la interfaz.

Fuente: Autor

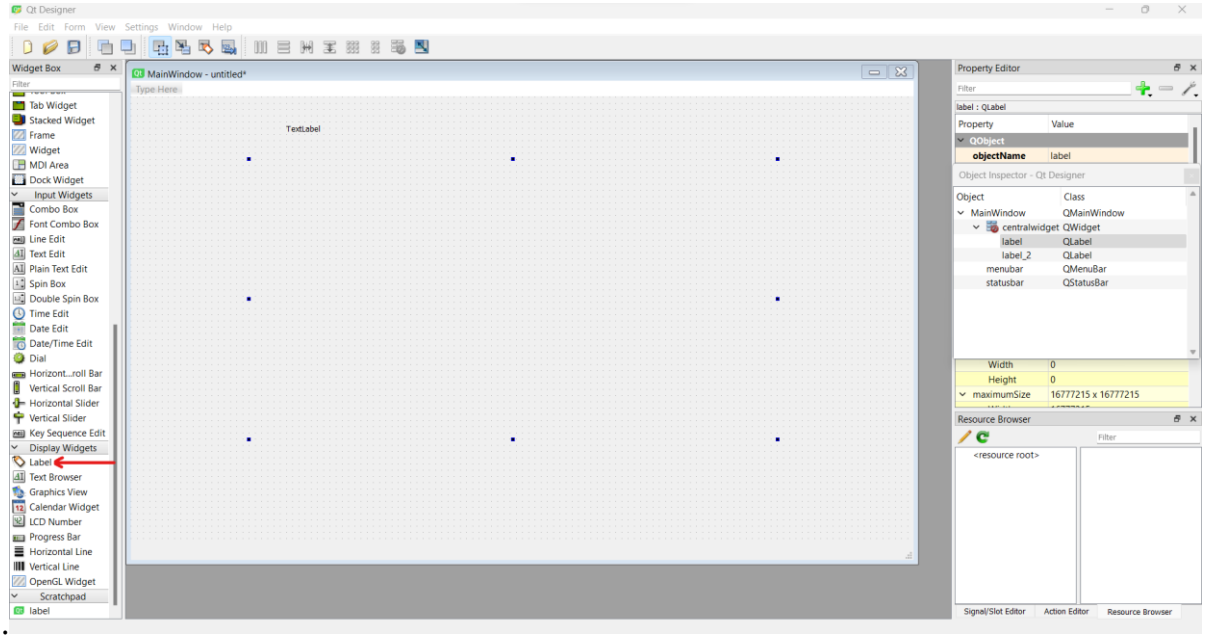


Figura 40: Ubicación de la herramienta Label.

Fuente: Autor

Dentro del **widget label**, se busca la opción **charge rich text**, donde se abre una sub pantalla que permite insertar una imagen deseada para la interfaz (Figura 39). Se deben seguir los pasos señalados para la inserción de la imagen como se puede observar en la Figura 41.

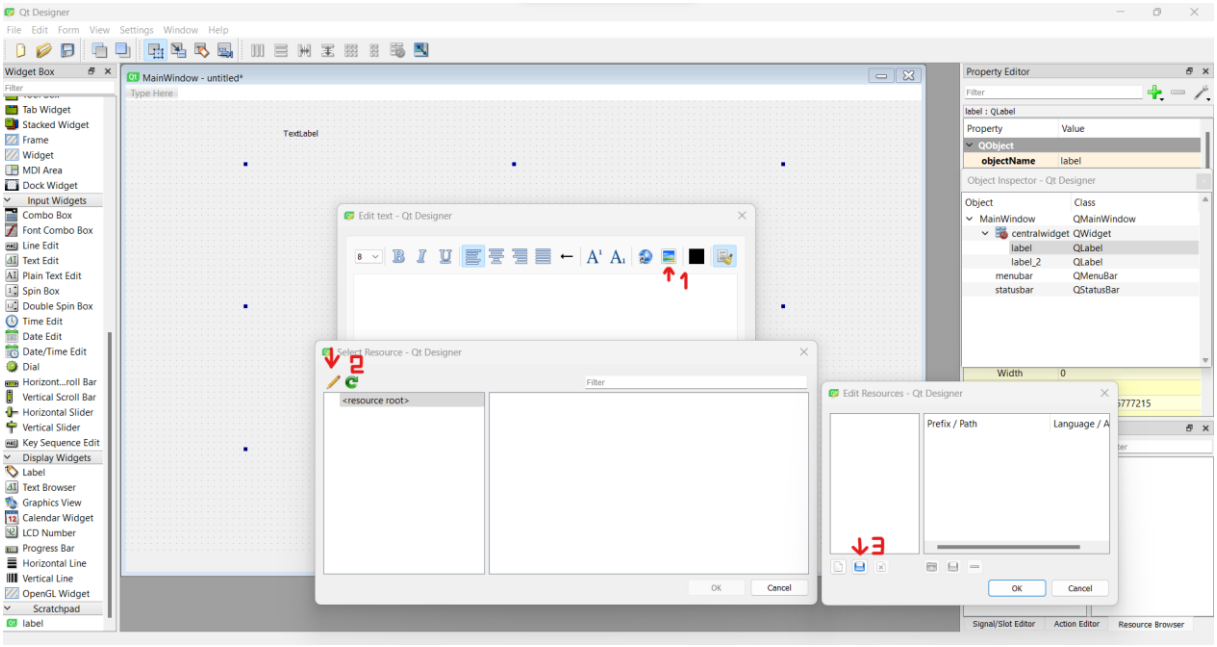


Figura 41: Sub pantalla para insertar una imagen.

Fuente: Autor

Es importante indicar que la imagen (Figura 39) a ser añadida debe tener la extensión (.qrc).

Al insertar la imagen no proporcionará ningún tipo de funcionalidad, hasta determinar las secciones en donde se desea introducir datos y en donde se desean observar los datos de los sensores del prototipo, la inserción de la imagen se puede observar en la Figura 42.

Figura 42: Imagen insertada en la interfaz.

Fuente: Autor

Una vez agregada la imagen en la interfaz, se escoge el comando line edit en cada uno de los recuadros donde se va a visualizar los datos enviados desde Arduino. De la misma manera, se coloca el comando push button como pulsantes los cuales tendrán la función de accionar los actuadores. El comando timeEdit, el cual se encarga de la edición en una interfaz parecida a un reloj digital, con la finalidad de introducir la hora para efectuar el riego.

Cada comando push button, line edit y timeEdit, se los denomina de manera que concuerde con la programación de la Figura 36. En el espacio en donde se represente el dato correcto en la Figura 43.

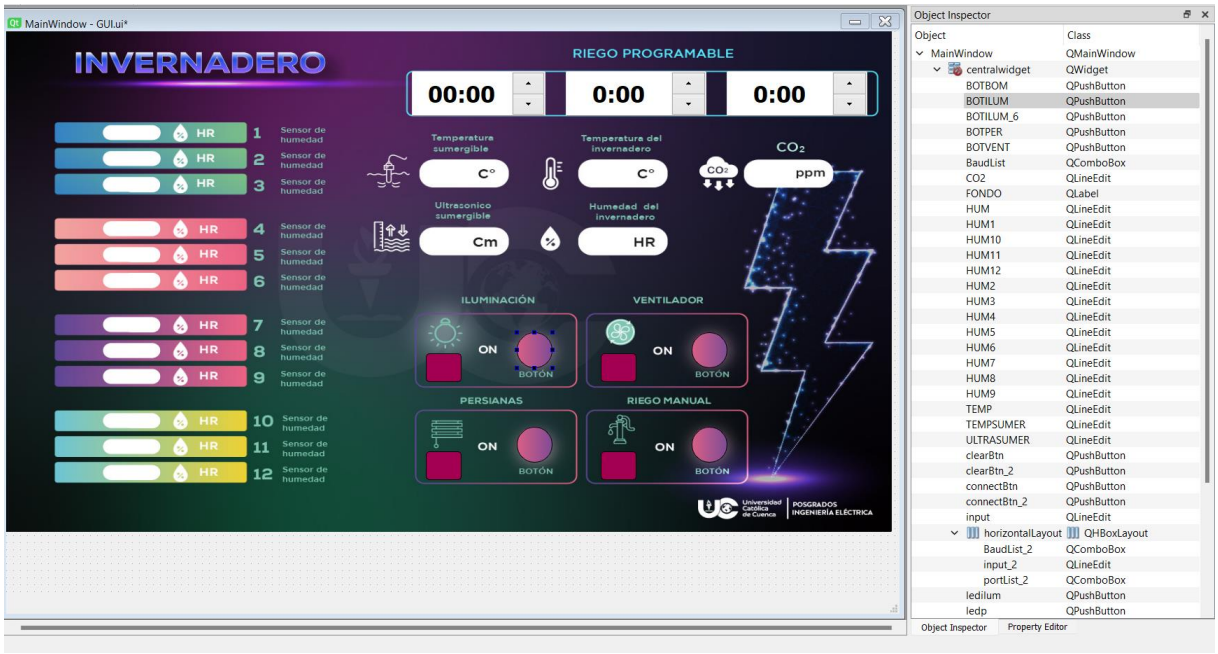


Figura 43: Nombre de los botones dentro de la interfaz.

Fuente: Autor

3.4 Código Arduino para el emparejamiento con la pantalla.

Obtenida la comunicación entre Arduino y la pantalla, se modificó el código de Arduino de acuerdo a los espacios necesarios para su funcionamiento; de tal manera que cada sensor, se encuentre dentro del lugar correcto en la pantalla. En primera instancia, se realizó la declaración de variables de cada uno de los sensores utilizados dentro de la función principal void setup(); donde se realiza la configuración inicial de cada uno de las variables, como se puede evidenciar en la Figura 44 se declara la configuración inicial para el almacenamiento de datos buscando el archivo Invernadero en memoria SD para poder guardar los datos obtenidos por el prototipo, de la misma manera se utiliza el apartado (rtc.adjust) para ajustar los parámetros de hora y fecha el cual se realizara solo una vez al iniciar la programación y

posteriormente este se lo comentará para evitar que la fecha y hora se reinicien cada vez que el microprocesador lea la programación en caso de reinicios espontáneos.

```
void setup()
{
  Serial.begin(9600);            // Iniciamos puerto serial 9600 badius

  if (! rtc.begin()) {
    while (1);
  }
  //rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  // Ajustar solo una vez, Año mes dia hora minuto segundo  despues comentar
  //rtc.adjust(DateTime(2023, 1, 26, 16, 21, 1));

  co2Sensor.calibrate();                      // calibración sensor CO2
  if (!SD.begin(53))                          // Se inicia modulo lector SD card
  {
  }
  if (SD.exists("datalog.csv"))              // Si entra el módulo SD al archivo estación para almacenado de datos
  {
    myFile = SD.open("Invernadero.csv", FILE_WRITE);
    if (myFile)
    {
      myFile.close();
    }
    else
    {
    }
  }
```

Figura 44: Configuración inicial para el módulo reloj y almacenamiento de datos en la SD.

Fuente: Autor

De la misma manera, se realizó la configuración de cada una de las variables como se puede apreciar en la Figura 45 en donde se obtiene un promedio de dos sensores DHT22, los cuales trabajan de forma paralela para conseguir un resultado de temperatura y humedad relativa dentro del invernadero. Se puede observar que, existe un tiempo de retardo de 2 segundos para obtener una lectura ideal del sensor. Para finalizar la configuración e imprimir los datos del muestreo, se utilizan las variables h1 correspondiente a la humedad y t1 correspondiente a la temperatura. Luego, se envía cada uno de los datos separados por un punto y coma, lo cual permite

ordenar adecuadamente dichos valores. Esta separación, se realizó en cada uno de los sensores para la posterior visualización de datos.

```
//dht22
void dht22_1()

{

  if (millis() - tiempoUltimaLectura > 2000)
  {

    humedadi1 = dht1.readHumidity();
    temperaturai1 = dht1.readTemperature();
    humedadi2 = dht2.readHumidity();
    temperaturai2 = dht2.readTemperature();
    h1 = (humedadi1 + humedadi2) / 2;
    t1 = (temperaturai1 + temperaturai2) / 2;
    Serial.print(h1);
    Serial.print(" ;");
    Serial.print(t1);
    Serial.print(" ;");
    tiempoUltimaLectura = millis();              //actualizamos el tiempo de la última lectura
  }
  digitalWrite(13, HIGH);
  delay(100);
  digitalWrite(13, LOW);
  delay(100);
}
```

Figura 45: Configuración inicial para el sensor de humedad dht22.

Fuente: Autor

Dentro de esta configuración inicial, se presentan los primeros condicionantes del prototipo puesto que, luego de obtener el porcentaje de humedad del sensor este activa los actuadores correspondientes a la bomba de agua y electroválvula correspondiente a la sección, estos actuadores se activan si son menores o iguales al 10% de la humedad dentro de la sección a medir, caso contrario si el valor es superior al 10% los actuadores simplemente permanecerán apagados, como se puede evidenciar en la Figura 46. Este código, se replica para 12 sensores de humedad los

cuales se encuentran ubicados en posiciones estratégicas para su correcta lectura y cada uno de ellos se encuentra accionando de manera independiente los actuadores.

```
// Humedad suelo 1
void Humedadsuelo1()
{
  valor_sensor = analogRead(AOUT);
  porcentaje1 = map(valor_sensor, Valor_Sensor_Agua, Valor_Sensor_Aire, 100, 0);
  if (porcentaje1 < 0) porcentaje1 = 0;
  if (porcentaje1 > 100) porcentaje1 = 100;

  Serial.print(porcentaje1);

// Encendemos la bomba de agua y la electrovalvula pertienete al sensor de humedad.
  if (porcentaje1 <= 10)
  {
    digitalWrite (valvula0, LOW);
    digitalWrite (valvula1, LOW);
  }
  if (porcentaje1 > 10 )
  {
    digitalWrite (valvula0, HIGH);
    digitalWrite (valvula1, HIGH);
  }
  Serial.print(" ;");

}
```

Figura 46: Configuración inicial Sensores Humedad suelo para cada manga.

Fuente: Autor

En la Figura 47, se indica la configuración del accionamiento de manera manual para las electroválvulas y la bomba de riego, este accionamiento se presenta cuando dentro de la programación realizada en la interfaz gráfica, se cumplen las condiciones ideales y envía un carácter especial denominado para este caso como “C” este carácter permite que los actuadores se ejecuten por un determinado tiempo. En este

caso, se encontrarían en funcionamiento durante 2 minutos y se desconectarán hasta su posterior accionamiento.

```
void electrovalvulas_f(char Dato_v)
{
  if (Dato_v=='C')                          ////////// Se activa riego manual
  {
    digitalWrite(valvula0, LOW);
    digitalWrite(valvula1, LOW);
    digitalWrite(valvula2, LOW);
    digitalWrite(valvula3, LOW);
    digitalWrite(valvula4, LOW);
    digitalWrite(valvula5, LOW);
    digitalWrite(valvula6, LOW);
    digitalWrite(valvula7, LOW);
    digitalWrite(valvula8, LOW);
    digitalWrite(valvula9, LOW);
    digitalWrite(valvula10, LOW);
    digitalWrite(valvula11, LOW);
    digitalWrite(valvula12, LOW);
    delay (100);

    tiempoEncendido = millis(); // Almacenar el tiempo actual
    }
    if ((millis() - tiempoEncendido) >= 120000)  // Si ha pasado 3  minuto PARA 2 MIN SON 120000 MILISEGUNDOS PARA 3 SON 180000
    {
      digitalWrite(valvula0, HIGH);
      digitalWrite(valvula1, HIGH);
      digitalWrite(valvula2, HIGH);
      digitalWrite(valvula3, HIGH);
      digitalWrite(valvula4, HIGH);
      digitalWrite(valvula5, HIGH);
      digitalWrite(valvula6, HIGH);
      digitalWrite(valvula7, HIGH);
```

Figura 47: Accionamiento de manera manual de electroválvulas desde la pantalla.

Fuente: Autor

La cisterna es uno de los componentes más importantes para el riego del invernadero, debido a ello el sistema de reabastecimiento funciona en conjunto con el sensor ultrasónico que mide el nivel de agua. Permite obtener el nivel de agua dentro del tanque y reabastecerlo cuando sea necesario, el nivel del agua dentro de la cisterna se obtiene desde la distancia en donde se encuentra ubicado el sensor, es decir la cisterna obtendrá un nivel óptimo para el funcionamiento cuando este a 70 cm de distancia del sensor, cuando el nivel del agua se encuentra en 120 cm del sensor la cisterna se encuentra vacía, ahí es cuando se realiza el reabastecimiento del mismo, con lo cual se garantiza que la cisterna siempre cuente con una reserva óptima de agua. La distancia es determinada debido al nivel de agua necesaria para obtener

una mezcla perfecta entre vitaminas y el líquido hidratante. El código para su ejecución se encuentra en la Figura 48.

```
void electrovalvula2()
{

    if (hora == 15 && minuto>1 && minuto<30)  // si la distancia entre el sensor y el agua es mayor a 100 cm o igual  se enciende la elctrovalvula
    {
      if (distance1 >= 100)
       {
        digitalWrite (valvula13, LOW);
       }
    }

   if (distance1 <=70)
   {
     digitalWrite (valvula13, HIGH);
   }


}
```

Figura 48: Código de ejecución para el reabastecimiento de la cisterna.

Fuente: Autor

Se realiza la configuración para el almacenamiento y el orden en el que los datos de los sensores se guardarán dentro de la tarjera SD como se puede observar en la Figura 49.

```
void almacenar()                                          // Funcion del Modulo SD , Almacenamiento de datos
{
  myFile = SD.open("Invernadero.csv", FILE_WRITE);        // Abrimos el archivo Estación para escribir datos
  if (myFile)
  {
    // if (ret1==0)
    // {
    myFile.println("[año];[mes];[día];[hora];[minuto];[segundo];[Humedad %HR];[Temperatura °C];[Nivel de agua (cm)];[Tempe
    //ret1=1;
    //}
    myFile.print(";");
    myFile.print(";");
    myFile.print(yyear);            // Escribiendo Año
    myFile.print(";");
    myFile.print(mes);             //Escribiendo Mes
    myFile.print(";");
    myFile.print(dia);             // Escribiendo dia
    myFile.print(";");
    myFile.print(hora);            // Escribiendo hora
    myFile.print(";");
    myFile.print(minuto);          // Escribiendo minuto
    myFile.print(";");
    myFile.print(segundo);         // Escribiendo Segundo
    myFile.print(";");
    myFile.print(h1);              // Escribiendo humedad
    myFile.print(";");
    myFile.print(t1);              // Escribiendo Temperatura HT22
    myFile.print(";");
    myFile.print(distance1);       // Escribiendo Ultrasonico Sumergible
    myFile.print(";");
```

Figura 49: Código para el almacenamiento de datos en la SD.

Fuente: Autor

Finalmente, como se pudo evidenciar todas las configuraciones anteriores se encuentran en forma de función para su posterior ejecución, en la función principal void loop, en donde se ejecutan de manera infinita observada en la Figura 50.

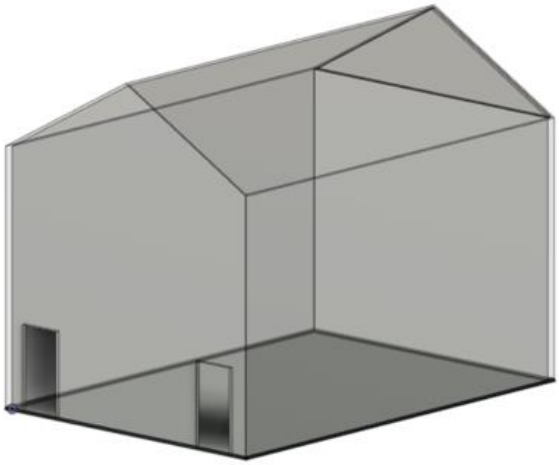
```
void loop()
{
  dht22_1();
  ultrasonicosumegible();
  temperaturasumergible1();
  Humedadsuelo1();
  Humedadsuelo2();
  Humedadsuelo3();
  Humedadsuelo4();
  Humedadsuelo5();
  Humedadsuelo6();
  Humedadsuelo7();
  Humedadsuelo8();
  Humedadsuelo9();
  Humedadsuelo10();
  Humedadsuelo11();
  Humedadsuelo12();
  CO2();
  fecha();
  almacenar();                    // Almacenar datos en la memoria SD
  electrovalvula2();
  iluminacion_f(Dato);
  electrovalvulas_f(Dato);
}
```

Figura 50: Función de ejecución void loop().

Fuente: Autor

3.5 Instalación del prototipo dentro del invernadero.

En la instalación del prototipo es importante mencionar que se ha añadido la instalación de un sistema de vigilancia, cuya finalidad es obtener un monitoreo constante de los sembríos. Dentro del invernadero se instalaron 17 sensores que permiten el monitoreo del entorno para el cultivo y varios actuadores, los cuales actúan al nivel de cambios que tienen los sensores como las electroválvulas y relés, para lo cual se ha realizado el diseño en 3d del invernadero como se puede observar en la Figura 51(a),Para poder realizar la automatización sin ningún tipo de intervención previa ver Figura 51(b).



(a)

(b)

Figura 51: Imágenes del bosquejo del invernadero: (a) Diseño 3D del invernadero; (b) Fotografía del interior del invernadero sin ningún tipo de intervención

Fuente: Autor

Para realizar la instalación fue fundamental conocer las medidas del invernadero, para determinar las distancias a las cuales se instaló los respectivos sensores, garantizando su correcto funcionamiento. Cada sensor de funcionamiento analógico, se encuentran ubicados a 5 metros de la estación de control (sensores de humedad

para cada manga, sensor de CO2),y los sensores de funcionamiento digital (sensores de humedad y temperatura, ultrasónicos, temperatura de la cisterna) se encuentran ubicados a una distancia máxima de 15 metros, y actuadores (electroválvulas).

En cuanto a la distancia de las cámaras, estas se encuentran colocadas a una distancia de 2 metros entre cada cámara. Las cámaras abarcan de manera individual un rango de 4 m2 , lo cual permite tener una visión eficaz del monitoreo del cultivo, para obtener una visión clara de la dimensión del invernadero con la ubicación de las cámaras y sensores se ha realizado un diagrama en AutoCAD como se puede observar en la Figura 52.

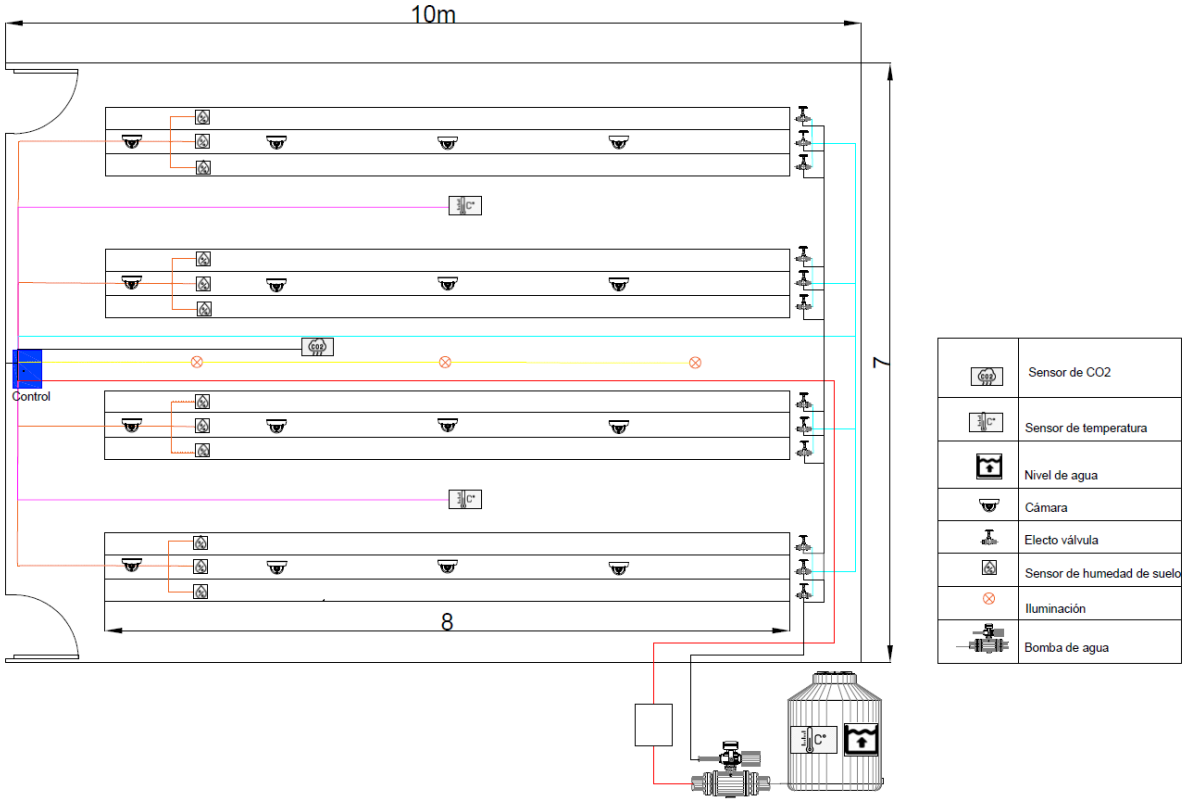


Figura 52: Diagrama de instalación del prototipo.

Fuente: Autor

3.5.1 Instalación de Cámaras e iluminación.

En el circuito de monitoreo, se instalaron 16 cámaras en un circuito cerrado que consta con un sistema de almacenamiento y estación de monitoreo como se observa en la Figura 53. La estación de monitoreo fue ubicada en una estructura física aledaña al invernadero en donde se puede realizar las pruebas del sistema de monitoreo como se aprecia en la Figura 54.

De la misma manera, el circuito de iluminación a diferencia de las cámaras, el control se realiza directamente desde la estación de control del invernadero. Se dispone a la instalación de cámaras e iluminación como primera etapa de acuerdo a la complejidad del trabajo realizado.

Figura 53: Instalación de circuito de monitoreo e iluminación.

Fuente: Autor

Figura 54: Prueba de funcionamiento del sistema de monitoreo.

Fuente: Autor

3.5.2 Instalación de electroválvulas.

En el prototipo de automatización es indispensable tener el control de riego, este control de riego se encuentra conformado por 12 electroválvulas y una bomba de agua trabajando de manera conjunta con los sensores de humedad, lo que permite que el riego del cultivo se realice de manera eficaz, en este caso se utilizó el riego por goteo como se puede observar en la Figura 55.

(a) *(b)*

Figura 55: Instalación Electroválvulas: (a) Proceso de instalación de las electroválvulas; (b) Electroválvulas instaladas.

Fuente: Autor

3.5.3 Instalaciones en la cisterna y sensores aledaños.

En la cisterna y sus aledaños se encuentran 2 tipos de sensores y 2 actuadores:

- Sensor ultrasónico SR04M-2 apto para exteriores

  Dicho sensor, se encuentra en la tapa de la cisterna y está encargada de dar lectura del nivel del agua, a través de la lectura conjuntamente con el microcontrolador se acciona la electroválvula para permitir el paso del agua, para lo cual se ha creado una especie de caja protectora para su modulo, debido que este se encuentra expuesto a niveles de humedad evitado el daño por salpicadura por el agua lluvia como se observa en la Figura 56.

Figura 56: Diseño de caja para la protección del módulo Ultrasónico SR04M-2

Fuente: Autor

- Relé

  Este relé se encuentra aledaño a la cisterna y está encargado de accionar la bomba de agua para permitir el paso del líquido hasta las electroválvulas, este relé se encuentra alojado en una especie de caja plástica para evitar salpicaduras por la lluvia junto al módulo ultrasónico SR04M-2 como se evidencia en la Figura 57.

Figura 57: Instalación del relé para el control de la bomba de agua.

Fuente: Autor

- Electroválvula

  La electroválvula es encargada del control para el subministro de agua para la cisterna, para ello se realizó la adecuación para que dicha electroválvula se encuentre dentro de la cisterna como se observa en la Figura 58(a) Este control se logra mediante la lectura de un sensor ultrasónico SR04M-2 que se encuentra ubicado en la tapa de la cisterna.

- Sensor de temperatura sumergible

  Este sensor se encuentra dentro de la cisterna y se encarga de obtener la temperatura del líquido este sensor se encuentra ubicado al interior de la cisterna el cableado ingresa paralelamente con el tubo de abasto para la cisterna como se evidencia en la Figura 58(b).

(a) (b)

Figura 58: Instalación en la cisterna: (a) Instalación de tubería para la cisterna; (b) Instalación de la electroválvula y sensor de temperatura en la cisterna.

Fuente: Autor

3.5.4 Instalación de sensores de la humedad del suelo (capacitivos).

Estos sensores son los encargados del riego en cada una de las 12 secciones del invernadero, cada sensor activará el accionamiento de las electroválvulas y la bomba de agua. Estos sensores fueron previamente calibrados con el código de la Figura 18, Fue recomendable realizar esta calibración cuando todos los sensores estaban conectados para evitar variaciones indeseadas a futuro.

Estos sensores se encuentran ubicados a los extremos de las mangas para su correcta lectura como se puede ver en la Figura 59.

(a) (b)

Figura 59: Instalación de sensores en las mangas: (a) Sensor de humedad en la manga; (b) Instalación de sensores en las mangas del invernadero.

Fuente: Autor

3.5.5 Instalación de los sensores de temperatura DHT22 y sensor de CO2.

Estos sensores fueron instalados en la parte superior del invernadero, ventajosamente en los alambres base para la instalación de las cámaras, cada sensor tiene distancias cercanas al centro del invernadero para obtener una lectura

adecuada. Para acoplar dichos sensores se realizó un diseño de cajas mediante el software fusión360 el cual se puede observar en la Figura 60.

(a) (b)

Figura 60: Instalación Sensores Dht22 y CO2: (a) Diseño 3D de cajas para protección de los módulos dht22 y CO2; (b) Instalación del sensor de CO2.

Fuente: Autor

3.5.6 Instalación del sistema de control dentro del invernadero.

En el invernadero, se realizó el diseño y construcción de la estructura del sistema de control. Para ello la estructura fue planteada en dos partes como se puede observar en la Figura 61.

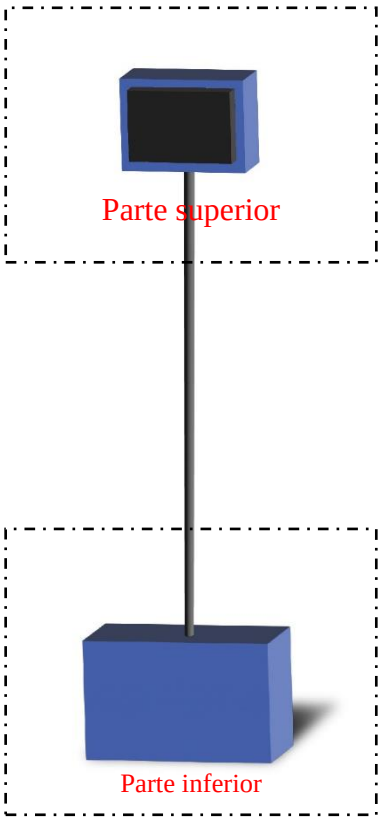


Figura 61: Ilustración del sistema de control del invernadero.

Fuente: Autor

1. La parte inferior estará encargada de albergar la fuente de poder para la alimentación del sistema y el microcontrolador encargado de controlar los sensores y actuadores Ver Figura 62.

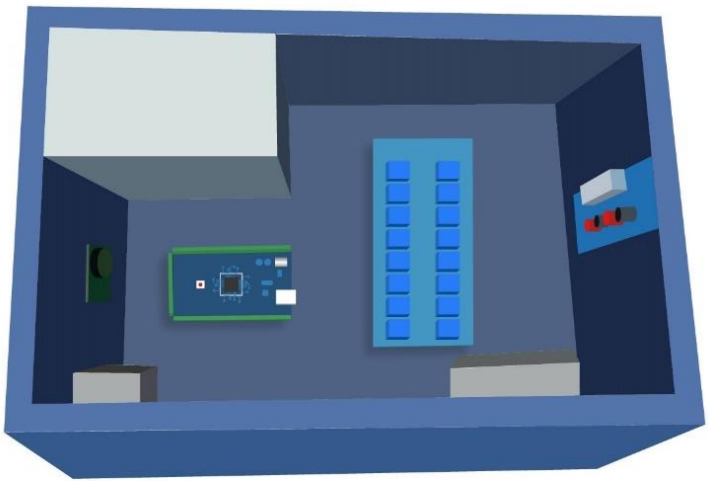


Figura 62: Parte inferior del sistema de control.

Fuente: Autor

2. En la parte superior, se encuentra albergada la pantalla y la Raspberry Pi (ver Figura 63). La comunicación entre los dos sistemas se obtiene mediante un cable de datos desde el microcontrolador de tipo serial, el cual atraviesa por un tubo que une y estabiliza los dos sistemas.

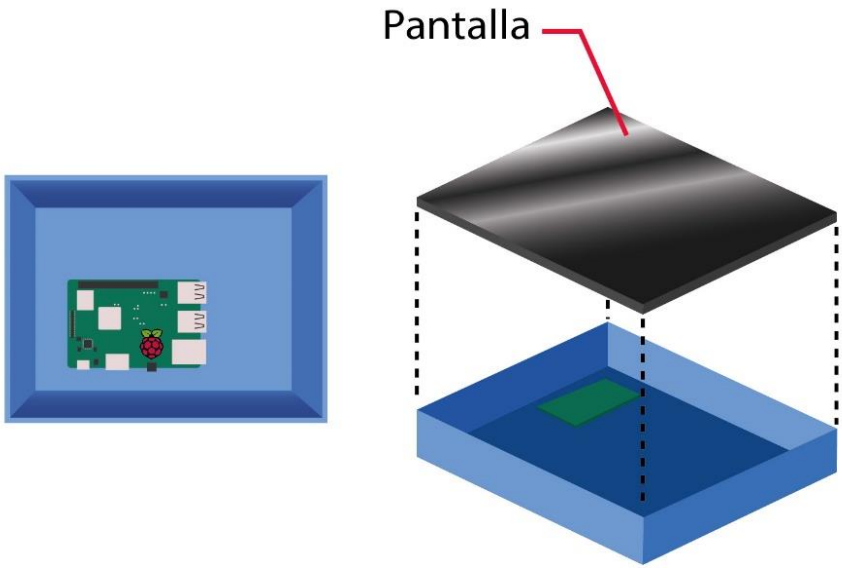


Figura 63: Parte superior del sistema de control.

Fuente: Autor

Tomando en cuenta el modelo planteado se procedió a realizar la construcción de las infraestructuras por secciones como se observa en la Figura 64, dichas estructuras se encuentran creadas por lata moldeada y soldada para dar el aspecto de caja.

(a) (b)

Figura 64: infraestructura para el centro de control del sistema: (a) Caja superior que alberga la pantalla y el raspberry pi ; (b) Caja inferior que alberga el microcontrolador.

Fuente: Autor

Debido a que las infraestructuras, se encuentran realizadas en metal es necesario colocar un aislante entre las partes para evitar cualquier tipo de cortocircuito por contacto. Una vez obtenido las dos cajas se procede a unir las mismas mediante un tubo, con medidas de 120 cm, obteniendo una altura ideal para la interacción con la pantalla del sistema, dicho tubo se encuentra conformado de tal forma que la unión de las cajas con el tubo se pueda acoplar o desacoplar mediante tuercas facilitando, el traslado y la instalación del sistema como se puede observar en la Figura 65.

Figura 65: Estructura completa del sistema.

Fuente: Autor

Una vez terminada la estructura se procedió a instalar los componentes electrónicos.

De la siguiente manera, considerando la complejidad debido al espacio los componentes de mayor tamaño tienen prioridad, se instala la fuente de alimentación con un divisor de voltajes obteniendo (5V ,3V ,12V) necesarios para el sistema y los módulos relés en la caja inferior del sistema como se puede observar en la Figura 66.

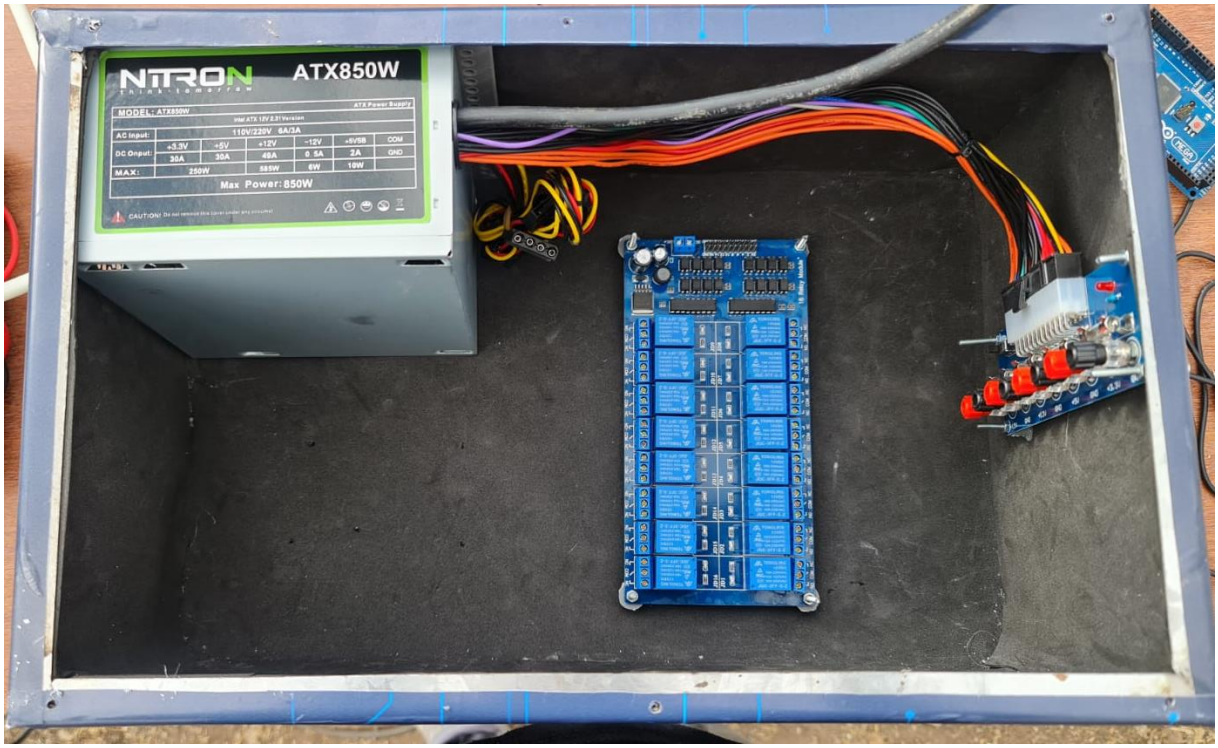


Figura 66: Armado del sistema de control.

Fuente: Autor

Para realizar las conexiones al Arduino y tener unas conexiones estables, lo ideal fue utilizar adaptadores, los cuales permiten atornillar los cables y así estos queden seguros sin riegos de desconexión por vibraciones dichos adaptadores se pueden observar en la Figura 67.

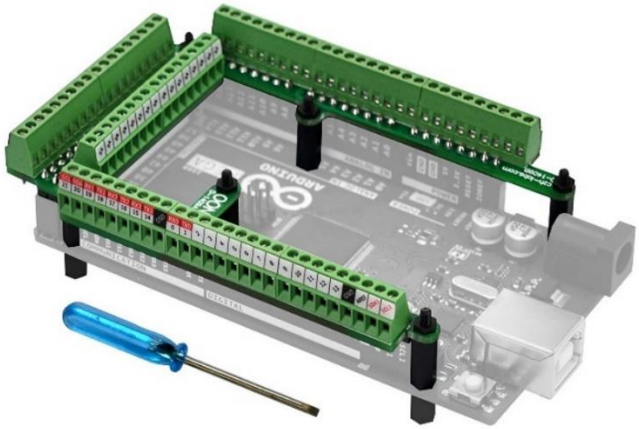
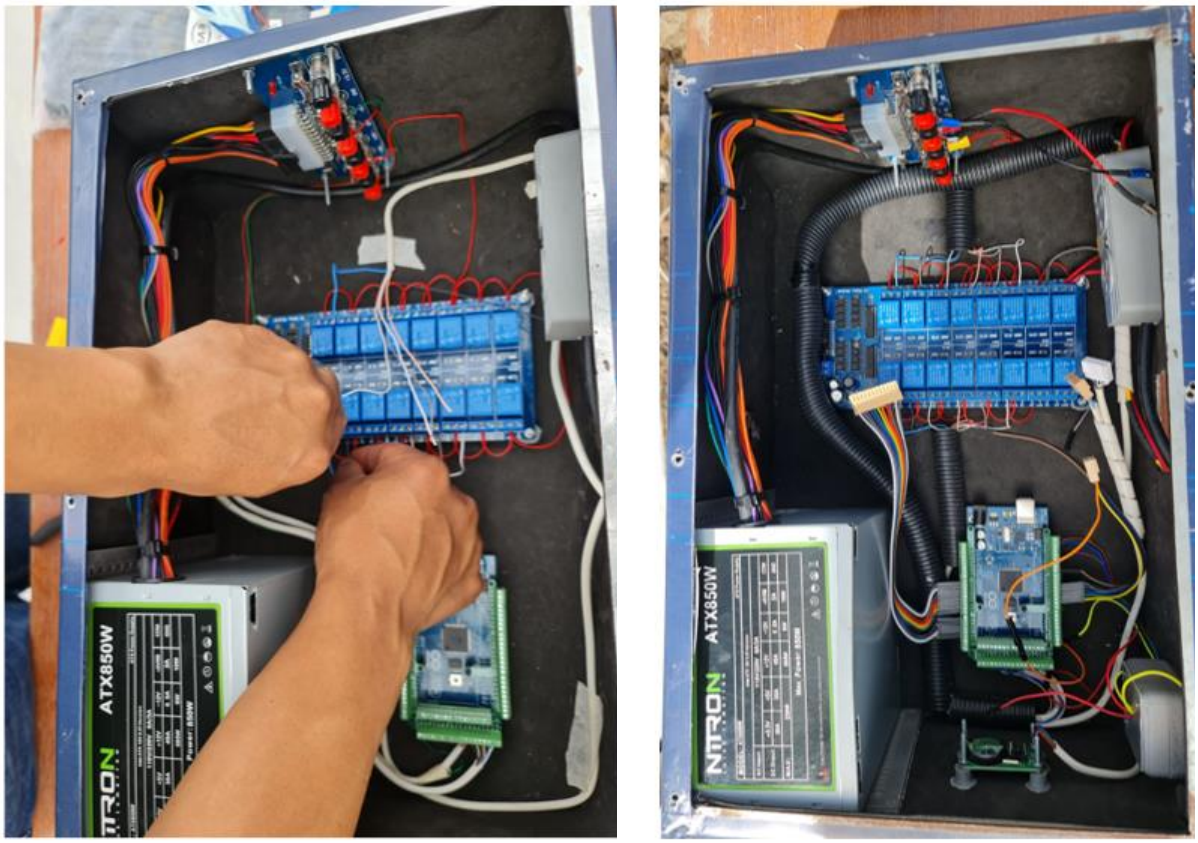


Figura 67: Modulo placa base para Arduino Mega.

Fuente: (Mega-2560, 2023)

Una vez ubicada la base en el Arduino, se procedió a realizar la instalación de los sensores con el microcontrolador y relés respectivamente como requiera el sistema, dicha instalación se puede observar en la Figura 68.



(a) (b)

Figura 68: Conexión del sistema de control: (a) Instalación del módulo relé; (b) Instalación del microcontrolador.

Fuente: Autor

Concluida las conexiones en el sistema de control, se realizó la prueba de funcionamiento y estabilización de la estructura, para posteriormente instalar la interfaz gráfica y realizar las pruebas de funcionamiento.

A) En la Figura 69(a); se realizó la prueba de comunicación de la interfaz con el controlador comprobando que la conexión se ha establecido de manera exitosa.

B) En la Figura 69(b); se realizó la prueba de funcionamiento de los sensores y actuadores, ejecutándolos de manera ordenada para comprobar el funcionamiento de cada uno de ellos y enviar la señal hacia la pantalla.

(a) (b)

Figura 69: Prueba del sistema de control: (a) Prueba del funcionamiento de la interfaz; (b) Prueba de conexión de los sensores.

Fuente: Autor

Finalmente, se coloca la interfaz gráfica en una posición estable para continuar con las pruebas de funcionamiento y toma de datos para el análisis posterior del circuito verificando la fluidez de los datos y presentar la corrección de los mismos de ser necesario como se evidencia en la Figura 70(b).

(a) (b)

Figura 70: Ubicación del sistema de control: (a) Ubicación de la interfaz gráfica; (b) Ubicación del sistema de control completo.

Fuente: Autor

CAPITULO IV

DISCUSIÓN Y RESULTADOS

4.1 Prototipo de invernadero semi hidropónico, frente a los métodos tradicionales.

Un cultivo normal o un cultivo al aire libre hace referencia al cultivo tradicional en el suelo en donde el cultivo depende directamente del suelo y clima pasa su desarrollo. Este tipo de cultivo es uno de los más llamativos por la accesibilidad y economía debido a que el suelo provee de recursos naturales para su crecimiento lo que puede tener algo de beneficio, demostrado el tipo de cultivo en la Figura 71. Sin embargo, como lo mencionado anteriormente el cultivo al depender netamente del clima y suelo, existen limitaciones al cultivar ciertos tipos de plantas, además no se puede tener un control exacto sobre el tiempo de riego y control de clima promoviendo las plagas en el cultivo y en peor de los casos la extinción de la planta.(Monge et al., 2011).

Figura 71: Cultivo en el suelo.

Fuente: (Martínez-Gamiño et al., 2019)

Los invernaderos tradicionales ofrecen beneficios, en donde lo más llamativo es la protección de climas adversos que pueden dañar el cultivo, su estructura se puede observar en la Figura 72. Que Permite mayor variedad de plantas debido a que el entorno es independiente al clima exterior, sin embargo el invernadero tradicional exige mayor cantidad de recursos, como agua y energía para mantener condiciones favorables para el crecimiento del cultivo.(Marín et al., 2017).

Figura 72: Cultivo en invernadero.

Fuente: (Chile, 2020)

Los invernaderos semi hidropónicos, se encuentran dentro de las opciones más populares para el cultivo de plantas en un ambiente controlado. Ver (Figura 73). En donde la planta se cultiva con sustratos, los cuales se los coloca en un nivel de agua que tiene un flujo determinado por sensores que detectan la humedad de la manga o el suelo en donde se encuentran plantados, dichos cultivos absorben los nutrientes disueltos en el líquido, lo que permite un desarrollo saludable. Entre los beneficios de estos sistemas de invernaderos se encuentra el control constante del agua, reduciendo la

necesidad de un riego manual; además de esto, el riego se dará cuando sea estrictamente necesario reduciendo la cantidad de agua necesaria para el cultivo y cuidado de la planta, haciendo que el invernadero sea una opción más llamativa debido a su sostenibilidad y eficiencia en recursos hídricos. A demás del beneficio hídrico también presenta beneficios específicos, como el llevar el monitoreo de las condiciones del cultivo. Entre estos se tiene: el monitoreo del clima, humedad, nivel de agua ajustando el nivel de riego y nutrientes de ser necesarios para tener un crecimiento de manera óptima y controlado para mayor producción (Juan, 2021).

Figura 73: Cultivo invernadero Semi hidropónico.

Fuente: Autor

4.2 Resultados.

Al finalizar la instalación del centro del control en el invernadero, se realizó pruebas de riego, iluminación y lectura de los sensores correspondientes, realizando una recopilación de datos de cada uno de dichos sensores. Y se los presenta de manera gráfica.

4.2.1 Temperatura del invernadero.

En la Figura 74, se encuentra representado el promedio de un día de temperatura interior del invernadero obteniendo como temperatura máxima de 25°C y registrando una temperatura mínima de 9°C. En el Anexo1 se encuentra representado la figura en donde se realizó la toma de datos de un mes sobre la temperatura del invernadero.

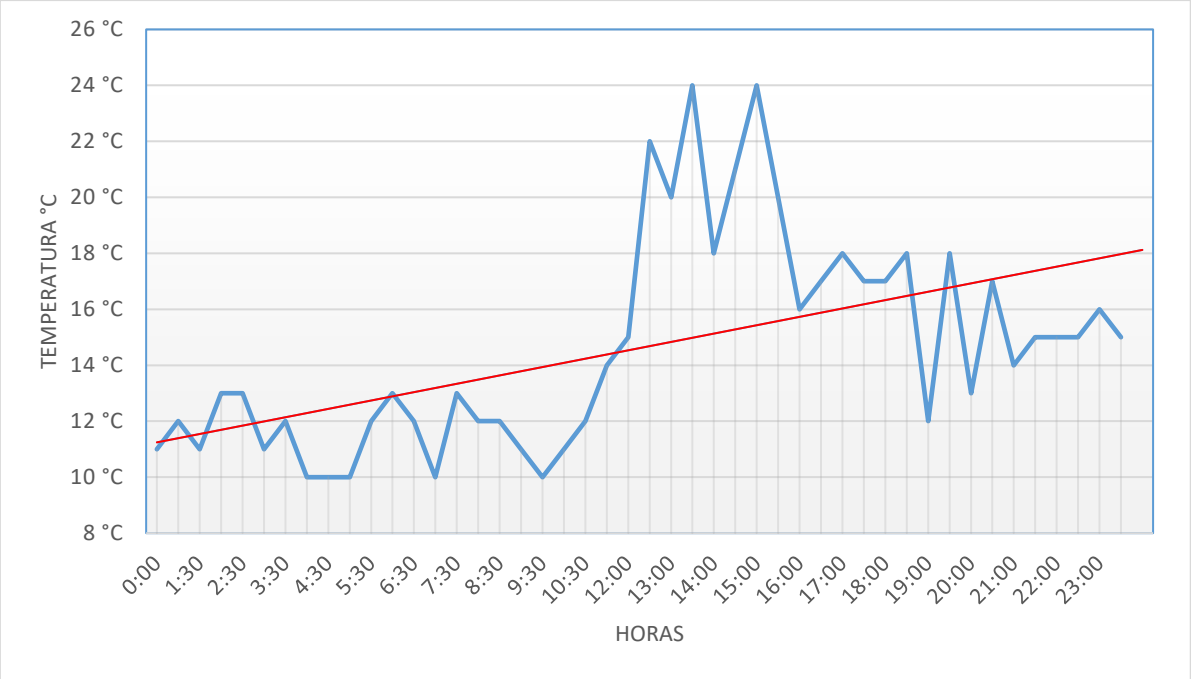


Figura 74: Temperatura del invernadero.

Fuente: Autor

4.2.2 Nivel de agua en la cisterna

La lectura del nivel del agua en la cisterna se realiza con el sensor ultrasónico SR04M-2, efectuando una lectura desde la parte superior de la cisterna para ser exactos en la tapa de la cisterna apuntando hacia la parte inferior o la base del tanque. Cuando el nivel del agua marca 120cm de distancia con el sensor representa que la cisterna se encuentra completamente vacía.

Para realizar la lectura del nivel de agua correspondiente con las vitaminas, se optó por vaciar el tanque y volverlo a llenar para saber los niveles necesarios. En este caso, el tanque se encuentra con 200 litros de agua cuando el nivel del sensor ultrasónico marca 70 cm desde la tapa de la cisterna, llegado a dicho valor se obtiene el nivel de agua necesario para un porcentaje equilibrado con las vitaminas y el líquido. En la Figura 48, se explicó como el código trabaja para lograr el reabastecimiento de la cisterna, pues es importante aclarar que el tanque nunca quedará vacío por protección de la bomba de agua; así que el sistema realiza una comprobación diaria y se efectúa el reabastecimiento cuando el nivel de agua es 100 cm del sensor ultrasónico y continuara llenando la cisterna hasta que el sensor ultrasónico este a 70 cm del sensor la electroválvula se desactivara y la cisterna contara con 200 litros de agua para continuar con el proceso de riego en los cultivos.

En la Figura 75, se puede evidenciar que el agua a llegado a 120cm de distancia del líquido, para ello se realizó un drenado de la cisterna y reabastecimiento de la misma en calidad de prueba. En el Anexo2 se encuentra representado la figura en donde se realizó la toma de datos de un mes.

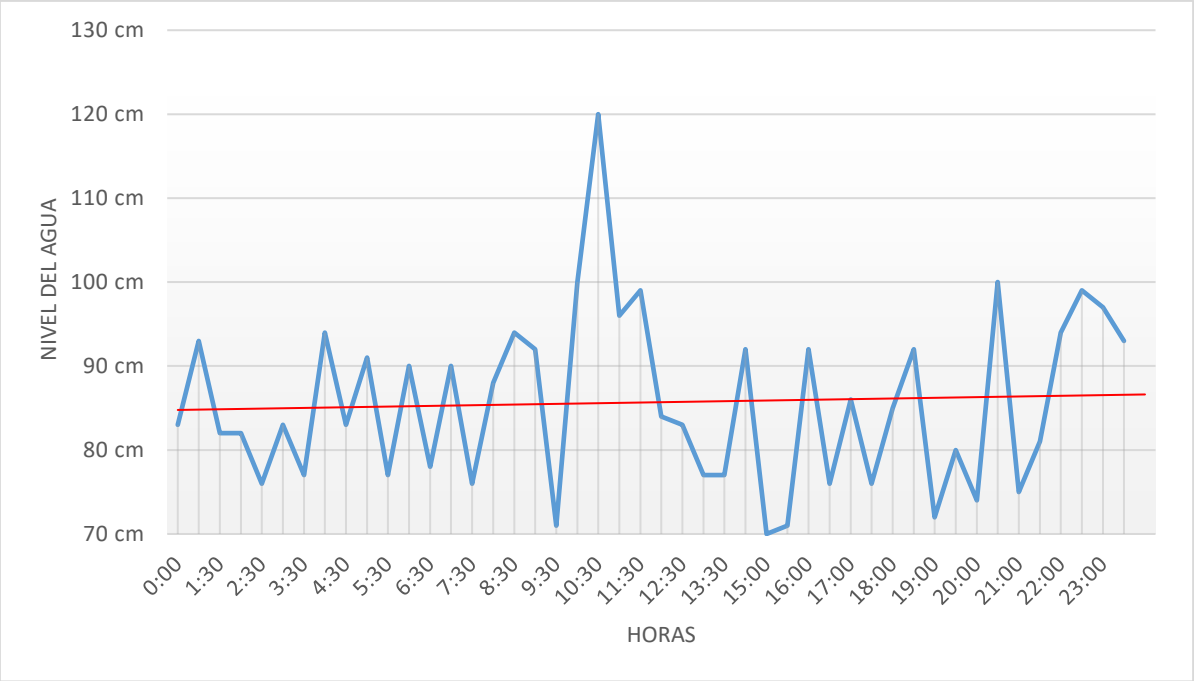


Figura 75: Nivel de agua en la cisterna.

Fuente: Autor

4.2.3 Temperatura de la cisterna

La temperatura hace referencia a la temperatura del agua dentro de la cisterna, como se puede evidenciar en la Figura 76 el comportamiento de la temperatura es diferente a la del invernadero, pues la temperatura mínima llega a 14°C y la temperatura máxima llega hasta los 20°C. En el Anexo3 se encuentra representado la figura en donde se realizó la toma de datos de un mes de la temperatura de la cisterna.

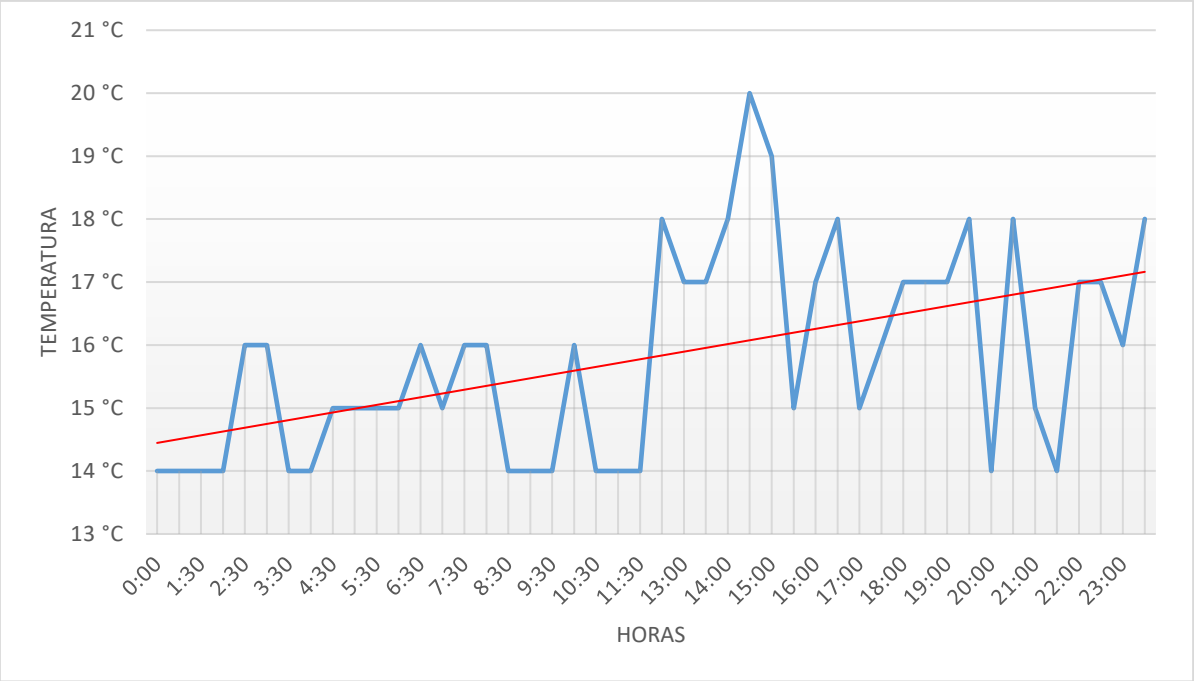


Figura 76: Temperatura de la cisterna.

Fuente: Autor

4.2.4 CO2

El sensor de CO2 presenta una lectura de entre 400 ppm y 420 ppm. Dichas partículas o su variación pueden depender estrictamente por el tipo de cultivo al cual está expuesto el sensor, en este caso el cultivo de frutillas presenta los valores presentados en la Figura 77.

En el Anexo5 se encuentra representado la figura en donde se realizó la toma de datos de un mes de la temperatura de la cisterna.

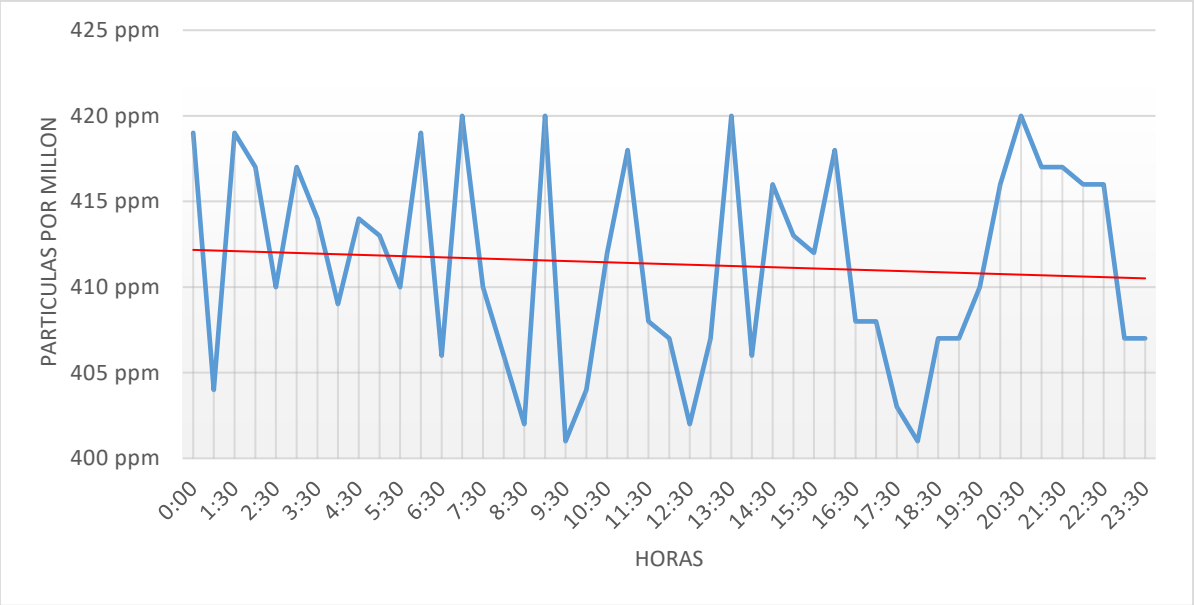


Figura 77: Lecturas del sensor de CO2.

Fuente: Autor

4.2.5 Humedad del invernadero

La Humedad del invernadero se determina tiene un pico máximo de 99 %HR y como mínimo puede llegar a 55%HR. Por lo general, estos valores altos de humedad son frecuentes en las horas de la noche, así como se puede observar en la Figura 78.

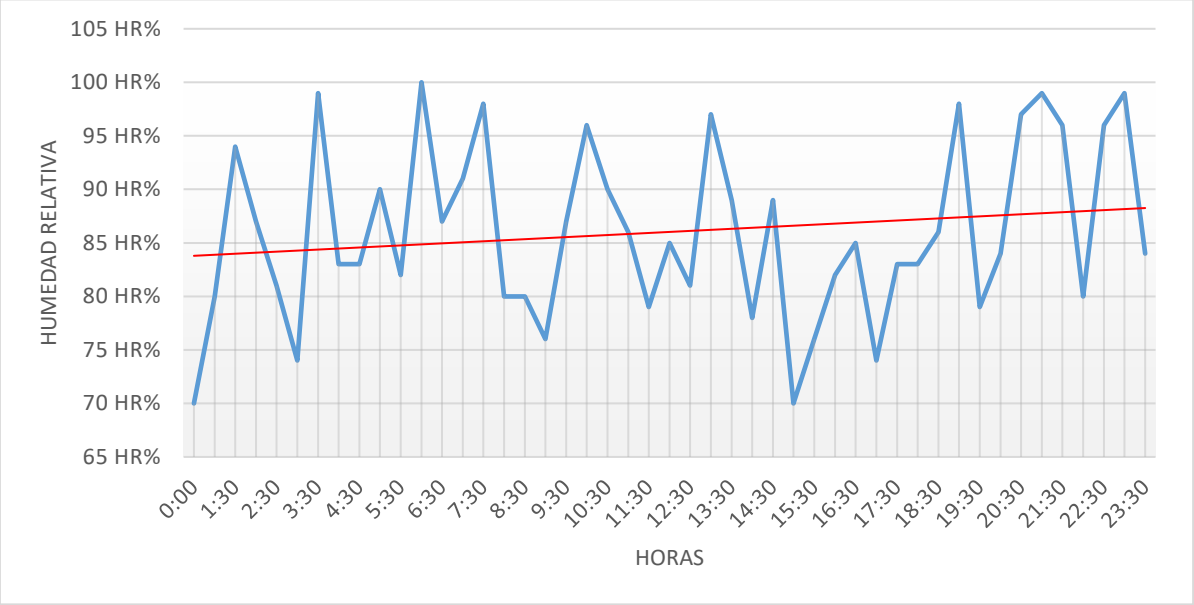


Figura 78: Humedad del invernadero.

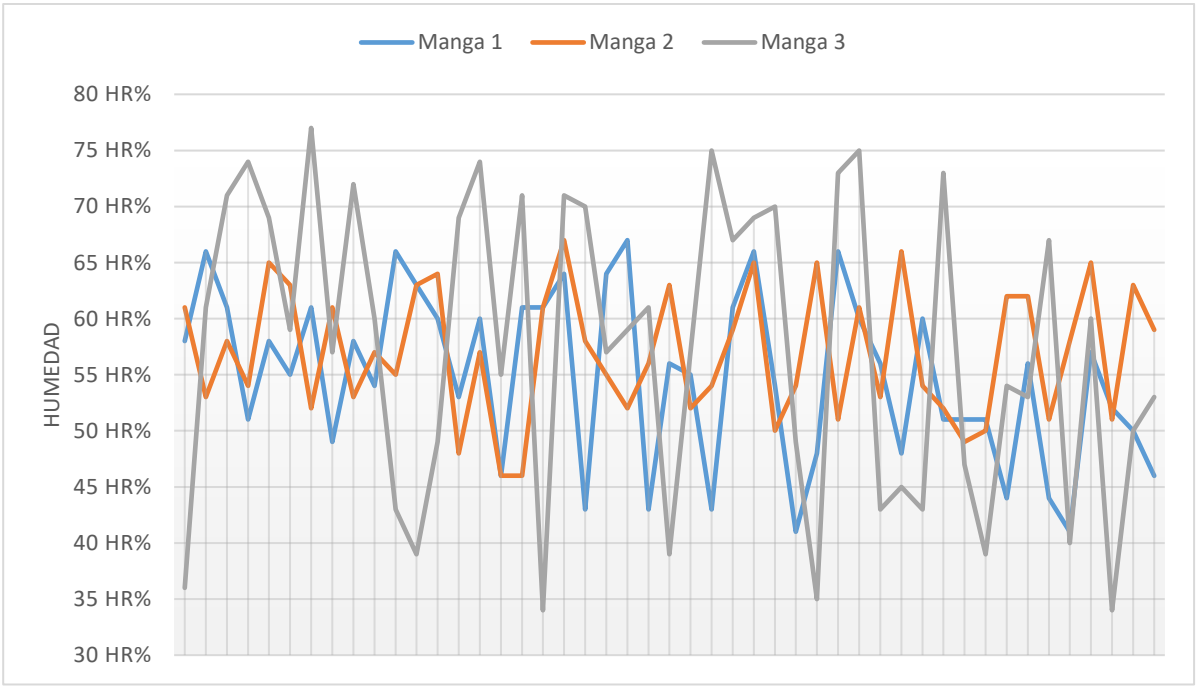
Fuente: Autor

4.2.6 Humedad en las mangas

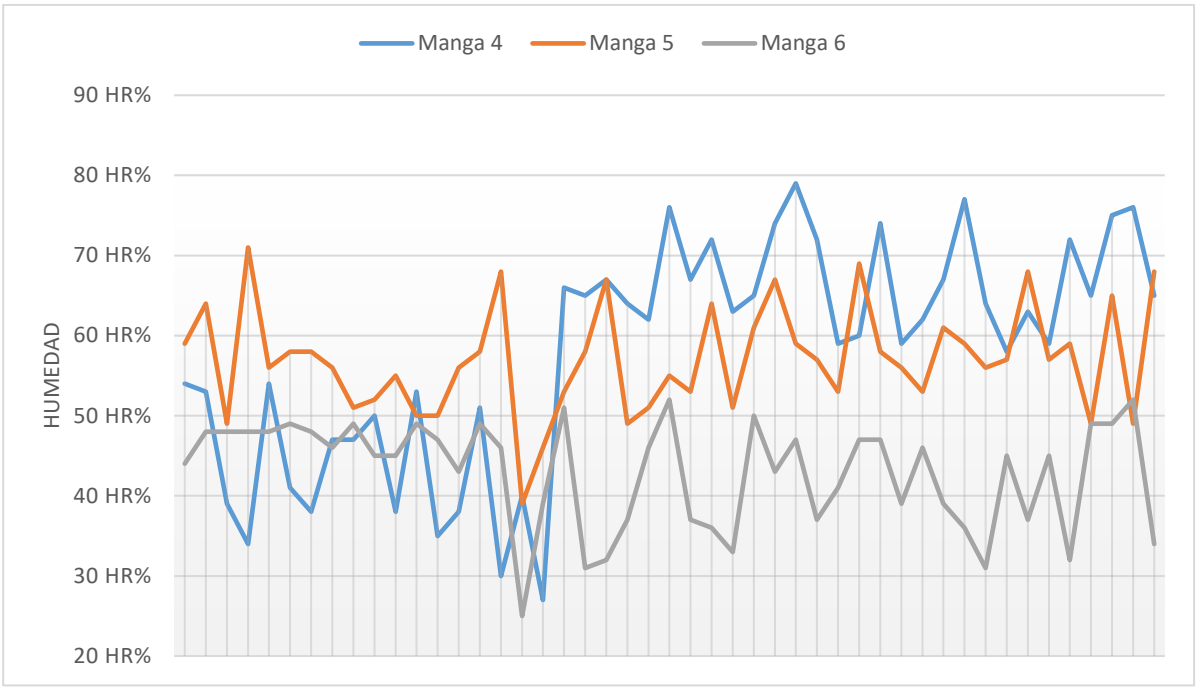
Cada una de las imágenes presenta el nivel de humedad en cada una de las mangas del invernadero, al actuar de manera independiente estos sensores accionarán el riego individualizado de ser necesario, caso contrario se continua con riego automático.

La Figura 79 representa la humedad en las mangas y sus variaciones durante el día.

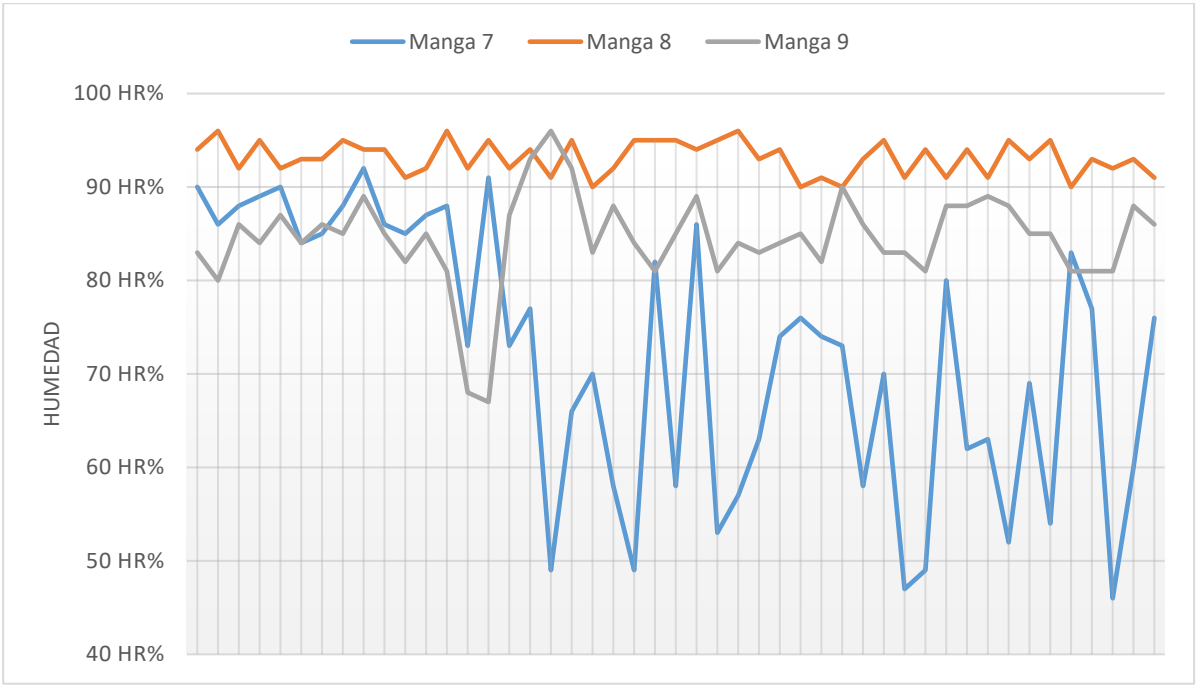
La Figura 79(a), representa a las mangas 1,2,3; La Figura 79(b),representa a las mangas 4,5,6; La Figura 79(c),representa a las mangas 7,8,9; y finalmente la Figura 79(d),representa a las mangas 10,11,12 del invernadero. En el Anexo4 se encuentra representado la figura en donde se realizó la toma de datos de un mes de la humedad en las mangas 1 a la manga 12.



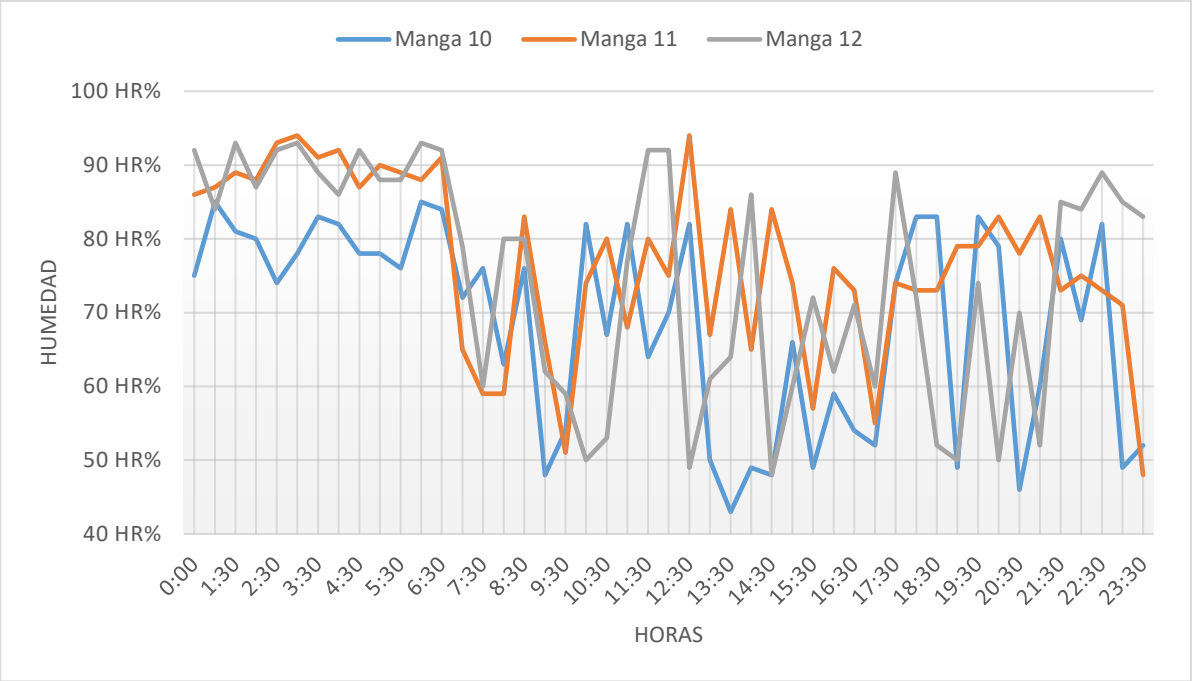
(a)


Manga 4
Manga 5
Manga 6
90 HR%
80 HR%
70 HR%
60 HR%
50 HR%
40 HR%
30 HR%
20 HR%
HUMEDAD

(b)


Manga 7
Manga 8
Manga 9
100 HR%
90 HR%
80 HR%
70 HR%
60 HR%
50 HR%
40 HR%
HUMEDAD

(c)



(d)

Figura 79: Humedad en las mangas: (a) Humedad en las mangas 1,2,3; (b) Humedad en las mangas 4,5,6; (c) Humedad en las mangas 7,8,9; (d) Humedad en las mangas 10,11,12.

Fuente: Autor

4.2.7 Crecimiento longitudinal de las plantas

Se realizó un seguimiento de crecimiento del cultivo en cada una de las 384 plantas sembradas.

El tiempo de comparación de las plantas es de un mes demostrando los efectos positivos de la automatización del invernadero semi-hidropónico puesto que el mes anterior a la automatización los cultivos presentaban 2 cm de crecimiento.

En la Figura 80 se puede observar el crecimiento del cultivo en las mangas en un mes con el uso de la automatización del invernadero.

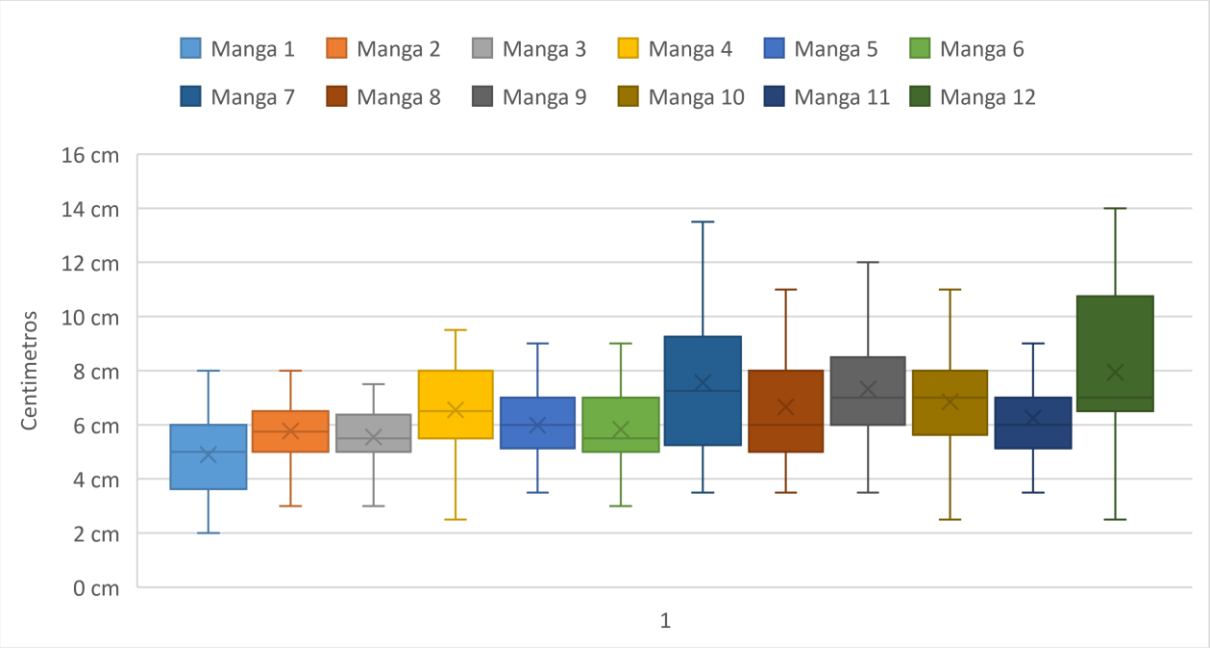


Figura 80: Crecimiento del cultivo en las mangas.

Fuente: Autor

No todas las plantas presentan un crecimiento uniforme por lo que existen plantas que tuvieron un crecimiento de 2.5 cm y otras plantas que presentan un crecimiento máximo de 14cm.

El promedio de crecimiento de todas las plantas de frutilla es de 6.4286 cm de alto.

4.2.8 Funcionamiento del sistema de control

En la Figura 81, se observa la interfaz gráfica la cual permite evidenciar en tiempo real los diferentes tipos de lecturas de los sensores, también permite la interacción con los actuadores debido a que estos se encuentran conectados directamente con el microcontrolador, obteniendo la posibilidad de colocar horas de riego manuales directamente desde la pantalla y evitando de esa manera tener el trabajo de acceder al microcontrolador o modificar el código del mismo.

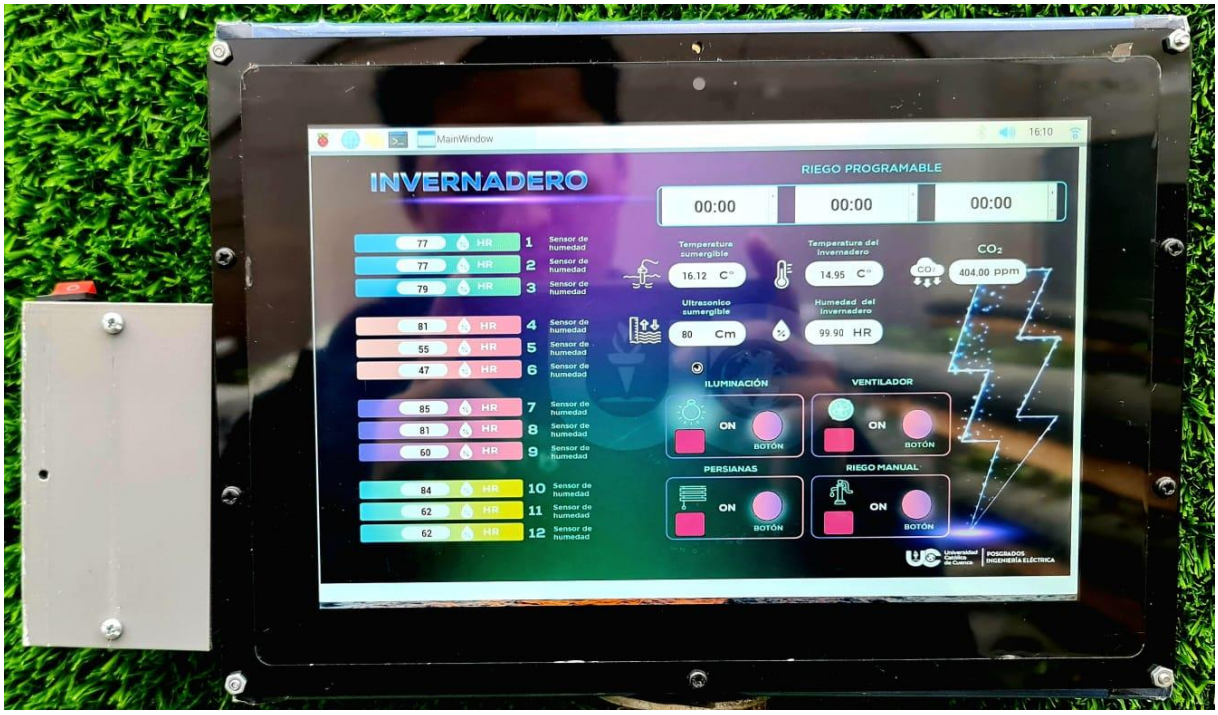


Figura 81: Interfaz grafica.

Fuente: Autor

En la Figura 82, se puede observar el crecimiento de las plantas de frutilla desde su siembra hasta que ya produce su fruto.

En la Figura 82(a) se puede observar el cultivo recién sembrado en las mangas, dicho cultivo llego con medidas de 2cm, desde donde se realizó la toma de datos para realizar el seguimiento de crecimiento de cada planta, en la Figura 82(b) se puede observar el crecimiento progresivo del cultivo con el prototipo de automatización, en la Figura 82(c) se puede observar los primeros frutos de los cultivos en desarrollo, finalmente en la Figura 82(d) la planta comienza a presentar frutos en cada una de las plantas demostrando la eficacia del prototipo de automatización de un invernadero semi-hidropónico.

(a)

(b)

(c)

(d)

Figura 82: Fotografías del crecimiento del cultivo : (a) Cultivo recién plantado en las mangas ; (b) Crecimiento del cultivo; (c) Proceso de crecimiento del fruto; (d) Primeros frutos del cultivo.

Fuente: Autor

DISCUSIÓN

Esta sección trata el análisis económico de los sistemas de automatización para un invernadero semi hidropónico.

Para dicha comparación se encuentra presente el prototipo de automatización realizado en este documento, desarrollado mediante hardware DIY utilizando microcontroladores y microordenadores. Por otra parte, se realiza la comparación económica con un sistema licenciado utilizando sistemas de PLC.

Los dos sistemas comparten los mismos beneficios, pero el prototipo planteado se considera más económico, como se puede observar en la cotización de la Tabla 10, a diferencia de un sistema de PLC que se observa en la Figura 83.

Tabla 10. *Proforma del prototipo de automatización.*

Fuente: Autor

Universidad Católica de Cuenca		
Proforma		
Materiales Para la automatización del invernadero UCACUE		
Ítem	**Materiales**	**Costo**
1	Microprocesador Arduino	30.00
1	Mini ordenador Raspberry Pi	200.00
1	Juego de 16 relés	20.00
1	Modulo Ultrasónico HC-SR04	5.00
12	Módulo de temperatura suelo	48.00
1	Sensor ultrasónico sumergible SR04M-2	10.00
2	Módulo de sensor de humedad y temperatura Sensor DHT22	14.00
1	Sensor de temperatura DS18B20	10.00
12	Electroválvulas	120.00
20m	Manguera PVC	40.00
100m	Cable multipar	40.00
	Total	537.00

CAMEI S.A.
Compañía de Automatización y Modernización Empresarial e Industrial
RUC: 0992710705001

Guayaquil: Avda. J. T. Marengo Km. 4 1/2 C. C. Salibaba Local 26 Teléfono: +593 4 2658 003 Celular: +593 998347226

Quito: Dirección: Vasco de Contreras N35-254 y Mañosca Teléfono: +593 2 2242 288 Celular: +593 998605356

Cuenca: Av. Hurtado de Mendoza 3-11 y Av. Los Andes Teléfono: +593 7 2864 323 Celular: +593 983338980

Cliente: CHRISTIAN COYAGO

Dirección:: ANTONIO LLORET BASTIDAS 4-82 Y CANTON JOYA DE LOS SACHAS

Ciudad: CUENCA

Persona de Contacto: DOCELECTRONICO

Teléfono: +593 0992 709 564

Cotización

No : CAM23_5186
Fecha : 7/Junio/2023

Tiempo de Entrega	Forma de Pago	Garantía	Rep.	Validez	Proyecto
SEGÚN CONDICIONES	30 días	1 AÑO	4	7/Julio/2023	

No.	Referencia	Descripción	Cantidad	V. Unitario	V.Total
1		SCHNEIDER ELECTRIC			
2	TM221CE16R	CPU COMPACTA AC9E/7S RELE ETHERNET	1	282,60	282,60
3	TM3AQ4	MODULO 4S ANALOGICAS	1	331,32	331,32
4	HMISTU855	TERMINAL D22 5.7 COLOR QVGA ETH	1	623,10	623,10
5		OMRON			
6	MEM_003	CP2E CPU, 1 ETHERNET PORT,8 TTL INPUTS 6, OUTPUTS TTL SOURCING, ,1 OPTION PORT, DC POWER	1	238,40	238,40
7	MEM_002	CP1 4IN/2OUT ANALOG 12K RES	1	439,39	439,39
8	MEM_001	NB HMI, 5.6 TFT QVGA, ETHERNET	1	1.049,64	1.049,64
9	DIO_001	CONTROLADOR DE TEMP. E5CC-800 48 × 48 MM 24 VAC/DC 48X48 SAL:VOLTAJE PARA SSR	1	81,69	81,69

CONDICIONES COMERCIALES:
1. Precio en Dólares Americanos (USD), el IVA se cobrará conforme a la tasa vigente al momento de la facturación. El precio responde a las tasas y aranceles vigentes a la fecha, en caso de cambio en las mismas nos reservamos el derecho de modificar los precios.
2. Validez de la oferta: 15 días
3. Tiempo de entrega: 24 HORAS
4. Forma de pago: CREDITO 30 DIAS
5. Garantía: 1 año calendario contra defectos de fabricación.
6. Lugar de Entrega: Oficinas CLIENTE CUENCA
7. Marca-Procedencia: SCHNEIDER - EUROPA
OMRON - EUROPA

PERSONA DE CONTACTO

Dennis Gavilanes C.
Telefono 0998602170
Email: dennis.gavilanes@camei.com.ec

SUBTOTAL:	US$ 3.046,14
IVA (12.0%):	US$ 365,54
TOTAL:	US$ 3.411,68

GAVILANES CASTRO DENNIS MAURICIO

Autorización

Figura 83: Proforma para automatización del invernadero con sistemas licenciados.

Fuente: Autor

Es importante mencionar que en la comparación económica se utilizó los mismos sensores y actuadores, por lo cual se compara solamente el centro de control del sistema.

Sin embargo, es importante recalcar que los sensores, de un sistema licenciado sobrepasan por mucho el costo a comparación de un sistema DIY.

Es importante recalcar que la proforma planteada para realizar el prototipo del invernadero semi hidropónico ya consta con la totalidad de precios. En la proforma de la Figura 83 solo se toma en cuenta la estación de control. Demostrando de esta manera que, el sistema propuesto tiene un valor inferior al de un sistema licenciado por PLC.

Tomando en cuenta solo el sistema de control del sistema planteado en el prototipo este tiene un valor de 230 dólares, realizando la comparativa con un sistema licenciado con similares características puede llegar a tener un valor 3400 dólares.

Realizando el mismo trabajo, con los mismos resultados y de fácil acceso económico.

CONCLUSIONES

- El prototipo de automatización desarrollado en este documento permitirá comparar el cultivo semi hidropónico automatizado con los diferentes métodos de cultivo tradicionales, permitiendo demostrar que el prototipo para el cultivo en el invernadero semi hidropónico es mucho más recomendable; tanto por los recursos hídricos y económicos comparado con otros estilos de siembra, así como por el desarrollo de cultivo. Es importante resaltar que, el uso de tecnología para el desarrollo de cultivos es una herramienta recomendada pues facilita el cultivo y el cuidado de las plantas.
- El uso de los invernaderos semi hidropónico dentro del cultivo no es nuevo, ha sido aplicado en el sector agrícola, logrando resultados favorables. Según la literatura científica sobre invernaderos, este tipo de control no es la excepción ya que se ha demostrado que se puede implementar el sistema en cultivos pequeños y a gran escala. El prototipo es altamente recomendable y de larga duración, que permite modificar, añadir actuadores y sensores sin inconvenientes.
- De acuerdo con los ingenieros en el campo de la agronomía, el tiempo de riego es determinado por el número de plantas y la presión que soportan las mangueras con sistema de goteo, debido que el prototipo de automatización semi-hidropónico cuenta con sensores que detectan la humedad, el riego es determinado por la humedad del sembrío en cada una de las mangas.
- A partir del concepto de un invernadero común, se desarrolló el prototipo de automatización semi hidropónico de este documento, con diferencias

que constituyen ventajas desde su construcción utilizando DIY con sensores y actuadores que se son de fácil adquisición, debido a su costo económico .

- Si bien un invernadero semi hidropónico ha demostrado ser una gran herramienta para el cultivo, es más costoso que el método tradicional de invernaderos o cultivo en el suelo. Este inconveniente se produce por la variedad de sensores que un invernadero semi hidropónico necesita. Ventajosamente el prototipo del sistema semi-hidropónico creado en este documento, el delimitante del costo se omite gracias a que el sistema planteado es mucho más económico.
- Lograr un sistema funcional de automatización para un invernadero semi hidropónico con los requerimientos que se necesitan, se utilizan varias ramas de investigación programas y regulación de los aspectos climáticos, tomando en cuenta que necesariamente se recurre a conocimientos de los expertos en temas de agronomía que posee los docentes de la Unidad Académica de Posgrado de la Universidad Católica de Cuenca, se realizaron las pruebas de funcionalidad con excelentes resultados los cuales se los puede evidenciar desde el cultivo de las frutillas hasta la germinación del fruto.

RECOMENDACIONES

- Es recomendable continuar con la investigación y aceptación de modelos de automatización dentro de los cultivos, con niveles de investigación tomando en cuenta costo económico y la facilidad de implementación, en conjunto con expertos para el cuidado y cultivo de las plantas.
- Se recomienda a la Unidad Académica de Posgrado de la Universidad Católica de Cuenca, continuar con los estudios e investigación para realizar nuevos prototipos e implementar mejoras en prototipos en más cultivos en busca de eficiencia y corrección de posibles errores.
- Es importante tener en cuenta el tipo de aplicaciones y funciones que puede tener un sistema de control para invernaderos semi-hidropónicos, en donde se puede contar e integrar nuevas funciones en la misma propuesta.
- Es recomendable al implementar nuevos estudios o mejorar el prototipo, realizando un estudio previo de los sensores que efectuaran el monitoreo y sus limitaciones, debido a que no todos los sensores poseen las mismas características en cuestión de resistencia a clima y la fiabilidad de datos obtenidos.

BIBLIOGRAFÍA

AG Electrónica. (2017). *DS18B20 CABLE: Sensor de temperatura DS18B20*. http://agelectronica.com/AG/

Arduino y el Internet de las cosas - Google Play Libros. (2018). https://play.google.com/books/reader?id=FIlyDwAAQBAJ&pg=GBS.PA1&hl=es_419

Bernabé I Ramos-López; Gabino A Martínez-Gutiérrez; Isidro Morales; Cirenio Escamirosa-Tinoco;, & Aleyda Pérez-Herrera. (2017). *Consumo de agua y rendimiento de tomate de cáscara bajo diferentes cubiertas de invernaderos*. https://doi.org/10.1590/S0102-053620170218

Bolaños, D. (2016). *Módulo de relés*. *8*, 1–10. http://www.bolanosdj.com.ar/MOVIL/ARDUINO2/moduloRele.pdf

Carrera,Zambrano, L., & Espartaco, C. (2018). *UNIVERSIDAD CATÓLICA DE SANTIAGO DE GUAYAQUIL FACULTAD DE INGENIERÍA*. http://repositorio.ucsg.edu.ec/handle/3317/4691

Castañeda, Rodrigo Herrera Ruiz, G., & José, Juan Escalante, G. (2003). *Núm. 2 Naturaleza y Desarrollo Julio-Diciembre*. *1*.

Chile, I. (2020). *Cultivos en invernadero | Invernaderos | Diseño y Construcción para Cultivo en Invernadero*. https://invernaderoschile.cl/2018/12/20/cultivos-en-invernadero/

Datasheet HC-SR04. (2022). www.leantec.es

David Marcelo Oña Salazar, & Byron Alexis Vaca Marmol. (2015). *DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE MONITOREO DE LAS VARIABLES AMBIENTALES Y ESTADO DE CARGA DEL UPS CON PLATAFORMA DE HARDWARE ARDUINO A TRAVÉS DE SENSORES Y UNA APLICACIÓN ANDROID PARA EL DATA CENTER DE LA EPMAPS.* UNIVERSIDAD POLITÉCNICA SALESIANA SEDE QUITO.

Departamento de ingeniería rural, E. T. S. de I. A. de M. (2000). Sistemas para la automatización de los invernaderos. *Mecanización* . https://www.miteco.gob.es/ministerio/pags/Biblioteca/Revistas/pdf_vrural%2FVrural_2000_118_66_70.pdf

DHT22 datasheet CodigoElectronica. (2022). http://codigoelectronica.com/blog/dht22-datasheet

Escaramilla, R. (2019). *ECORFAN® Editor en Jefe*. *3*. www.ecorfan.org/republicofperu,

Fernández, Milagros, Lorenzo Mínguez, Pilar, Cuadrado Gómez, Isabel MaGiménez Moolhvijzen, M., Agraria., A. J. D. G. de I. y F., Hortimed., Almería., F. para la I. A. en la P. de, Cajamar., & Curso Superior de Especialización sobre Mejora de la Eficiencia en el Uso del Agua en Cultivos Protegidos (7è : 2003). (2003). *Curso superior de especialización sobre mejora de la eficiencia en el uso del agua en cultivos protegidos : del 29 de septiembre al 10 de octubre 2003*.

Ferran Fabregas - Google Libros. (2020). https://books.google.es/books?hl=es&lr=lang_es&id=7EtOEAAAQBAJ&oi=fnd&pg=PA1&dq=raspberry&ots=raurHeHmEA&sig=0uYy9CAN_JKkQSS39fEhDQlJXmY#v=onepage&q=raspberry&f=false

Industria Horticola. (2008). www.agrocomponentes.com

Jain, Nikita Bhakar, S. ., & Singhal, R. (2017). A Review of Greenhouse Climate Control Application for Cultivation of Agriculture products. *International Journal of Engineering Trends and Technology*, *46*(6), 305–308. https://doi.org/10.14445/22315381/ijett-v46p253

Juan, A. R. G. (2021). *UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ Colegio de Ciencias e Ingeniería*.

Marín, S., Bertsch, F., Castro, L., Marín, S., Bertsch, F., & Castro, L. (2017). Efecto del manejo orgánico y convencional sobre propiedades bioquímicas de un Andisol y el cultivo de papa en invernadero. *Agronomía Costarricense*, *41*(2), 26–46. https://doi.org/10.15517/RAC.V41I2.31298

Martínez-Gamiño, M. Á., Osuna Ceja, E. S., Espinosa Ramírez, M., Martínez-Gamiño, M. Á., Osuna Ceja, E. S., & Espinosa Ramírez, M. (2019). Impacto acumulado de la agricultura de conservación en propiedades del suelo y rendimiento de maíz. *Revista Mexicana de Ciencias Agrícolas*, *10*(4), 765–778. https://doi.org/10.29312/REMEXCA.V10I4.1640

Mega-2560, M. de placa de ruptura de bloque de terminales G. ultra pequeño para A. (2023). *Módulo de placa de ruptura de bloque de terminales GPIO ultra pequeño para Arduino Mega-2560 : Amazon.com.mx: Industria, Empresas y Ciencia*. https://www.amazon.com.mx/ruptura-terminales-ultrapequeño-Arduino-Mega-2560/dp/B08XVMBR6P

MG-811 Carbon Dioxide Sensor Module. (2017). https://www.flyrobo.in/mg811-carbon-dioxide-co2-sensor-module

Monge, C., Chaves, C., & Arias, M. L. (2011). Comparación de la calidad bacteriológica de la lechuga (Lactuca sativa) producida en Costa Rica mediante cultivo tradicional, orgánico o hidropónico. *Archivos Latinoamericanos de Nutrición*, *61*(1), 69–73. http://ve.scielo.org/scielo.php?script=sci_arttext&pid=S0004-06222011000100009&lng=es&nrm=iso&tlng=es

PcComponentes. (2022). *Característica Raspberry Pi Model B+*. https://www.pccomponentes.com/caracteristicas-raspberry-pi-4

Proto Supplies. (2021). *Capacitive Soil Moisture Sensor Module - ProtoSupplies*. 1–22. https://protosupplies.com/product/capacitive-soil-moisture-sensor-module/

Rajguru Electronics. (2017). *MG-811 Carbon Dioxide Sensor Module*. https://pdf.indiamart.com/impdf/10268729697/MY-1833510/mg-811-carbon-dioxide-sensor-module.pdf

Ramírez Padrón, Laura Cecilia Cauich, I. C., Fernández, V. G. P., Luis, D. M., & Fernández, A. P. (2020). Análisis de los indicadores de competitividad de las exportaciones de fresa mexicana. *Revista Mexicana de Ciencias Agrícolas*, *11*(4), 815–827. https://doi.org/10.29312/REMEXCA.V11I4.2049

S. Raj, Jennifer J, V. (2019). AUTOMATION USING IOT IN GREENHOUSE ENVIRONMENT. *Journal of Information Technology and Digital World*, *01*(01), 38–47. https://doi.org/10.36548/JITDW.2019.1.005

Salinas Arcos. (2019). *Repositorio Universidad de Guayaquil: Diseño de un prototipo de sistema automatizado con Arduino para riego en el cultivo de fresas.* [Universidad de Guayaquil]. http://repositorio.ug.edu.ec/handle/redug/40625

Salto Grande, I. (2012). *Diciembre 2012-Revista INIA 37 Hortifruticultura*.

Serrano Cermeño, Z. (2005). *Construcción de invernaderos* (Vol. 1). Mundi-Prensa. https://books.google.es/books?hl=es&lr=&id=GIip3Q7T9mEC&oi=fnd&pg=PA3&dq=tipos+de+invernaderos+por+forma&ots=Q2vGqev3rM&sig=Hmtgqkku2QAkG8OVvEL2lD9vRX0#v=onepage&q=tipos de invernaderos por forma&f=false

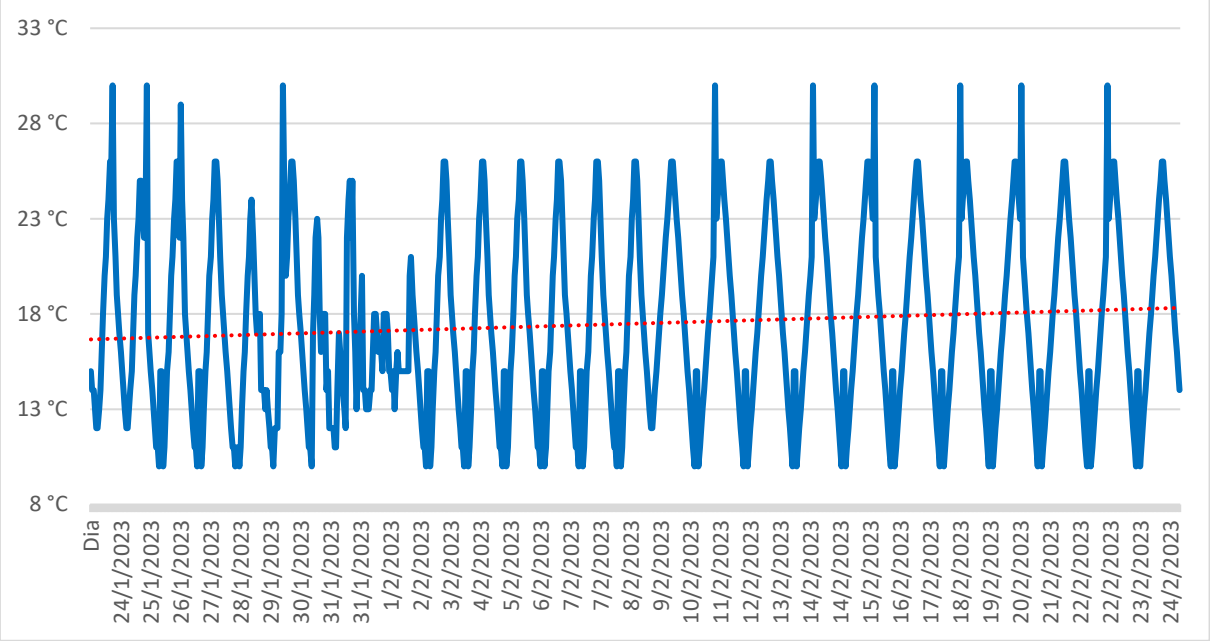
Shamshiri, Ramin Ishak, Wan Ismail, W. (2013). A Review of Greenhouse Climate Control and Automation Systems in Tropical Regions. *Journal of Agricultural Science and Applications (J. Agric. Sci. Appl.) J. Agric. Sci. Appl*, *2*, 176–183. http://aesop.rutgers.edu/~horteng/newsletter/2002/vol17-

Sreekantha, D. K., & Kavya, A. M. (2017). Agricultural crop monitoring using IOT - A study. *Proceedings of 2017 11th International Conference on Intelligent Systems and Control, ISCO 2017*, 134–139. https://doi.org/10.1109/ISCO.2017.7855968

Válvula solenoide 1/2" 12VDC (NC). (2022). https://naylampmechatronics.com/valvulas/314-valvula-solenoide-1p2-pulg-12vdc-nc.html

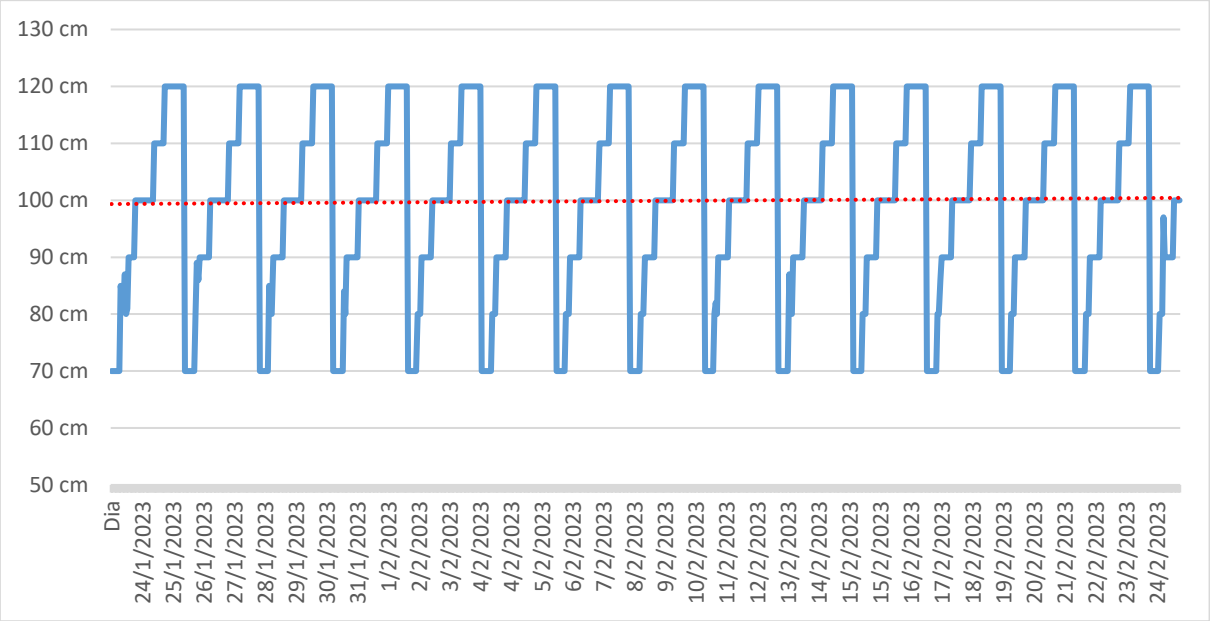
Zamux, B. (2022). *Sensor Ultrasonico Sumergible SR04M -2*. https://www.zamux.co/sensor-ultrasonido-sumergible-sr04m

ANEXOS



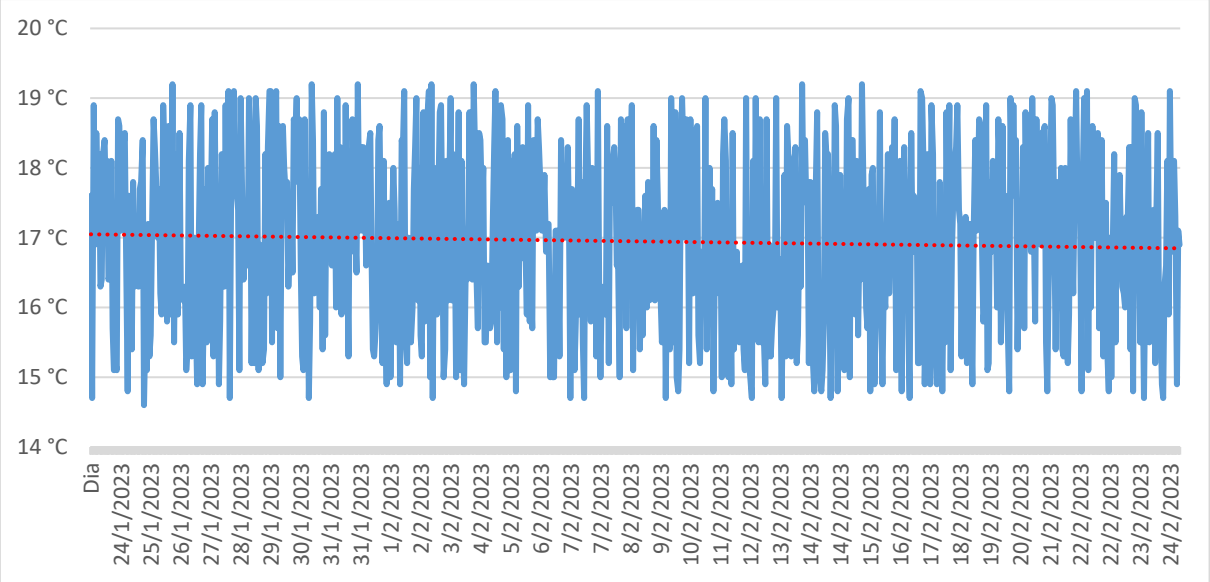
Anexo1:Datos de un mes de la temperatura del invernadero.

Fuente: autor



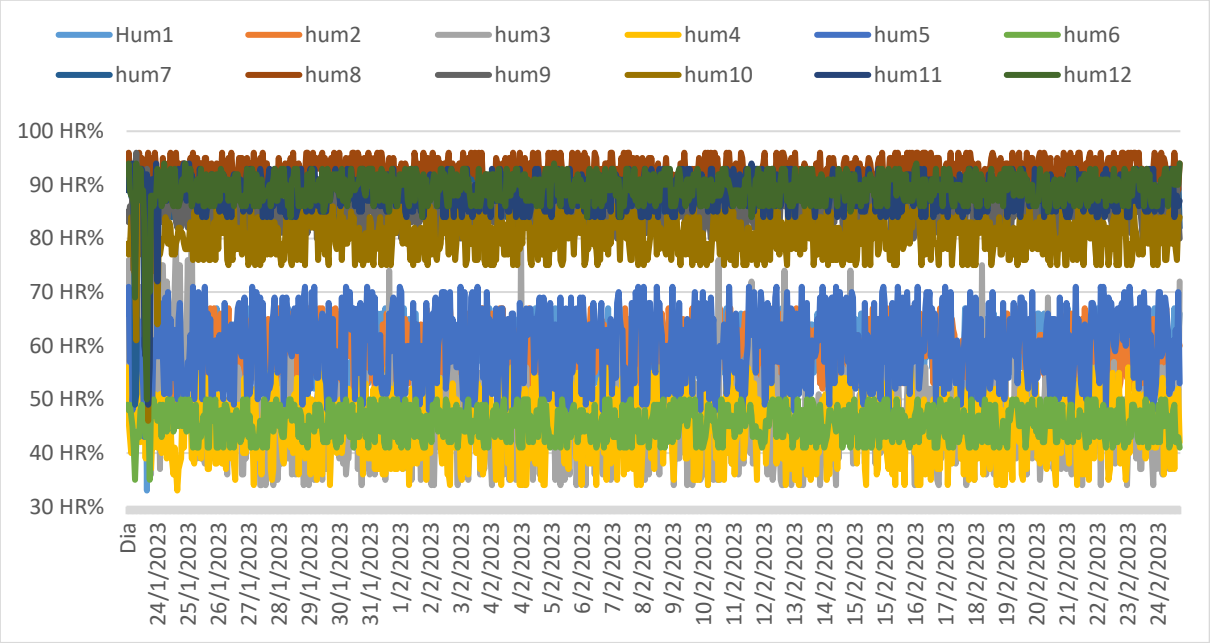
Anexo2:Datos de un mes del nivel del agua en la cisterna.

Fuente: autor



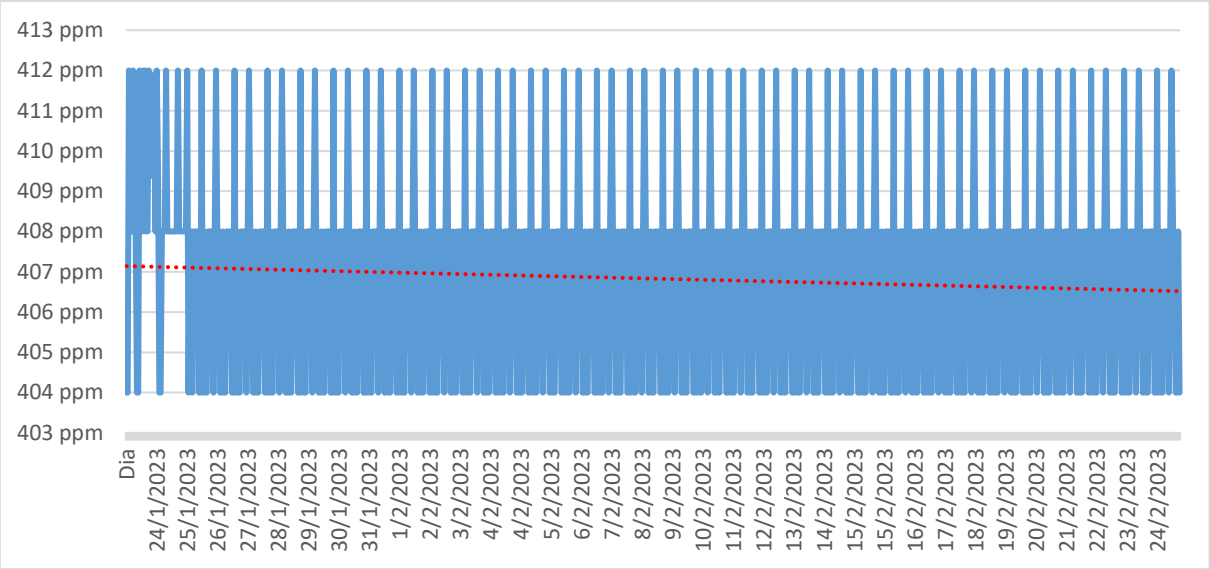
Anexo3:Datos de un mes de la temperatura en la cisterna.

Fuente: autor



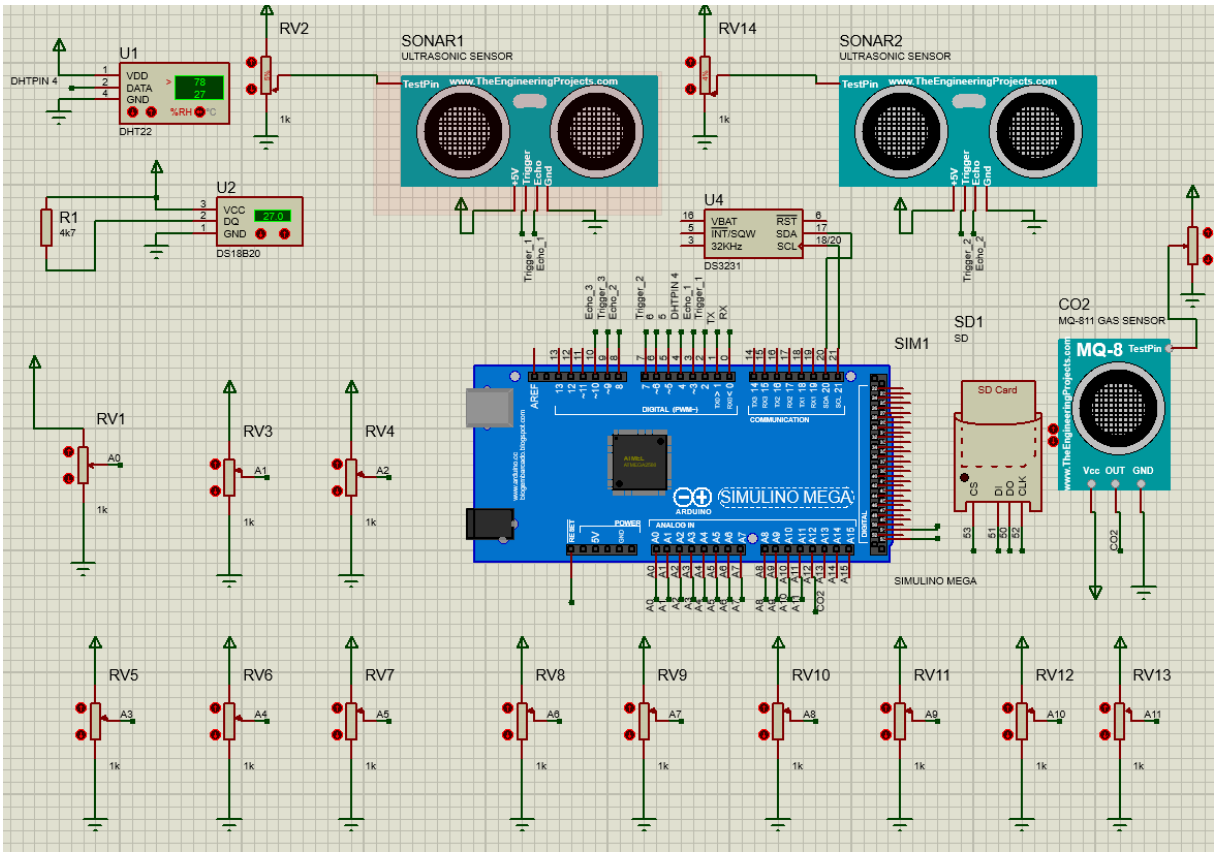
Anexo4:Datos de la Humedad en la manga 1 a la manga 12 en un mes.

Fuente: autor



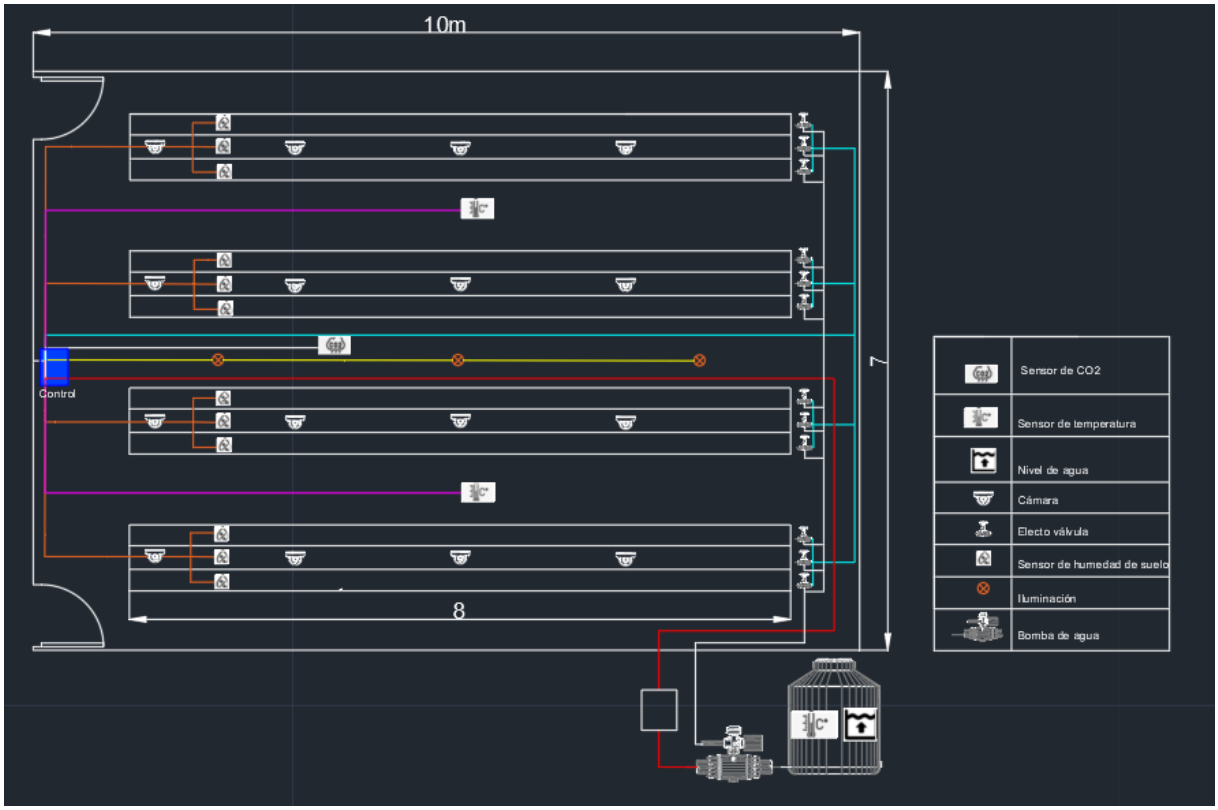
Anexo5:Datos de un mes del sensor de CO2.

Fuente: autor



Anexo6:Simulación del circuito.

Fuente: autor / obtenido a través de Proteus.

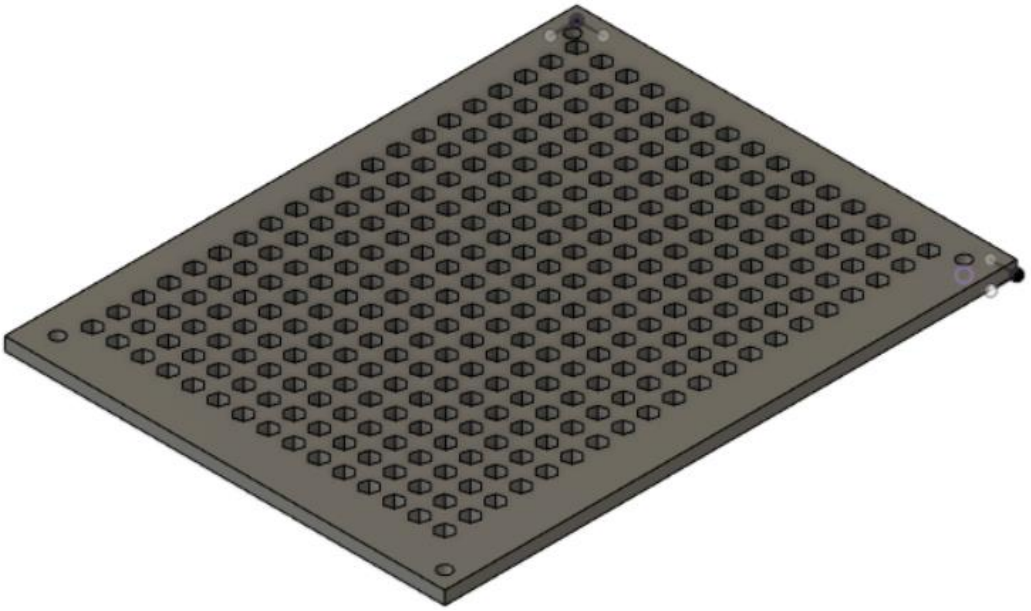


Anexo7:Plano para el circuito de control y monitoreo.

Fuente: autor / obtenido a través de AutoCAD.

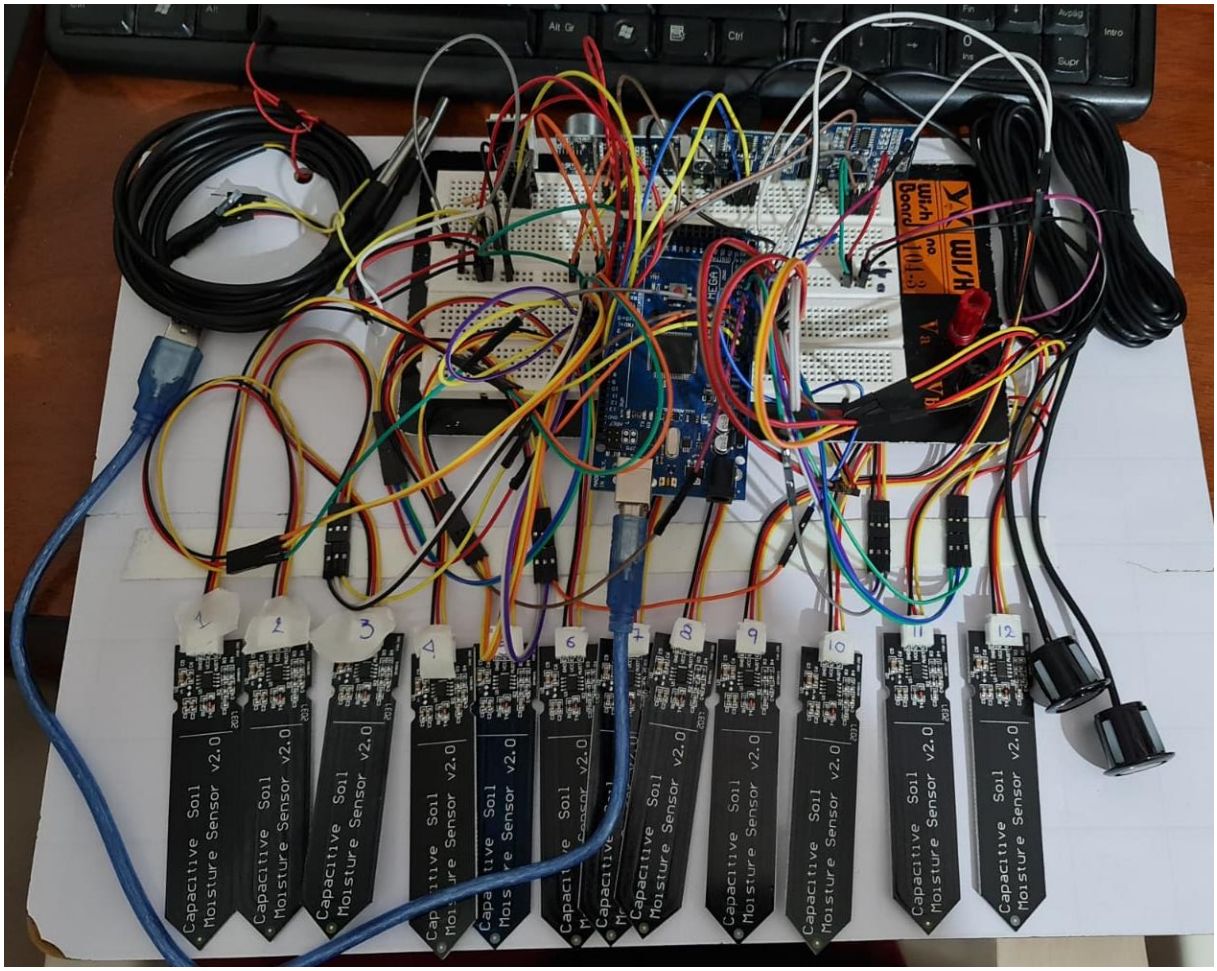
Anexo8:Cajas para cubrir módulos .

Fuente: autor / obtenido a través de Fusion360.



Anexo9:Panelones para cubrir la fuente de control.

Fuente: autor / obtenido a través de Fusion360



Anexo10:Prueba de sensores y código de Arduino.

Fuente: autor

```
// Librería para la comunicación I2C y la RTClib
#include <Wire.h>
#include <RTClib.h>
// Declaramos un RTC DS3231
RTC_DS3231 rtc;

#include <SoftwareSerial.h>

////////////////////////////Libreria sensor de temperatura y humedad ///////////////,
#include <DHT.h>
#define DHTPIN1 2             // Data pin for DHT1
#define DHTPIN2 3             // Data pin for DHT2
#define DHTTYPE DHT22          // DHT 22  (AM2302) AMBOS IGUALES
DHT dht1(DHTPIN1, DHTTYPE);
DHT dht2(DHTPIN2, DHTTYPE);
long tiempoUltimaLectura=0;; //Para guardar el tiempo de la última lectura
int ret1=0;
////////////////////// LIBRERIA MODULO SD ////////////////////////////
#include <SD.h>             //// Libreria micro SD
File myFile;                //// Función micro SD

////////////////////// LIBRERIA SENSOR DE CO2 ////////////////////////////////
#include "CO2Sensor.h"
int pinMG811 = A12;          //// CO2
CO2Sensor co2Sensor(pinMG811, 0.99, 100);


// ultrasonico sumergible
const unsigned int Trigger_2 = 5; // RX
const unsigned int Echo_2 = 4;    // TX

const int Valor Sensor Aire = 600;
```

```
const int Valor_Sensor_Agua = 270;
int valor_sensor = 0;
int porcentaje1 = 0;
int porcentaje2 = 0;
int porcentaje3 = 0;
int porcentaje4 = 0;
int porcentaje5 = 0;
int porcentaje6 = 0;
int porcentaje7 = 0;
int porcentaje8 = 0;
int porcentaje9 = 0;
int porcentaje10 = 0;
int porcentaje11 = 0;
int porcentaje12 = 0;
float humedadi1;
float temperaturai1;
float humedadi2;
float temperaturai2;
float h1;
float t1;
float temp1;
int distance1;
int q=0;
int ret=0;
int cont=1000;

// Humedad suelo 1
#define AOUT 0

// Humedad suelo 2
#define AOUT1 1
```

```
// Humedad suelo 3
#define AOUT2 2

// Humedad suelo 4
#define AOUT3 3

// Humedad suelo 5
#define AOUT4 4

// Humedad suelo 6
#define AOUT5 5

// Humedad suelo 7
#define AOUT6 6

// Humedad suelo 8
#define AOUT7 7

// Humedad suelo 9
#define AOUT8 8

// Humedad suelo 10
#define AOUT9 9

// Humedad suelo 11
#define AOUT10 10

// Humedad suelo 12
#define AOUT11  11
```

```
// temperatura sumergible
#include <OneWire.h>
#include <DallasTemperature.h>

OneWire ourWire1(17);                    //Se establece el pin 17  como bus  Sumergible
DallasTemperature sensors1(&ourWire1);  //Se declara una variable u objeto para nuestro sensor1

//////////////////////////// Declaración variables fecha ////////////////////
float yyear = 0;
float mes = 0;
float dia = 0;
float minuto = 0;
float segundo = 0;
float hora = 0;
float hora1 = 0;
float minuto1 = 0;

////////////// Declaración de variables para las electrovalvulas ////////////
const int valvula0 = 9;
const int valvula1 = 23;
const int valvula2 = 25;
const int valvula3 = 27;
const int valvula4 = 29;
const int valvula5 = 31;
const int valvula6 = 33;
const int valvula7 = 35;
const int valvula8 = 37;
const int valvula9 = 39;
const int valvula10 = 41;
const int valvula11 = 43;
const int valvula12 = 45;
const int valvula13 = 49;     //electrovalvula
////////////// Declaración de variables para iluminación ////////////
const int iluminacion = 47;
int on=0;

void setup()
{
  Serial.begin(9600);  // Iniciamos puerto serial 115200 badius
  if (! rtc.begin()) {
  while (1);
 }
  //rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  // Ajustar solo una vez, Año mes dia hora minuto segundo  despues comentar
  //rtc.adjust(DateTime(2023, 1, 26, 16, 21, 1));
  co2Sensor.calibrate();                    //// calibración sensor CO2
  if (!SD.begin(53))                   //// Se inicia modulo lector SD card
    {
    }
  if(SD.exists("datalog.csv"))          ///// Si entra el módulo SD al archivo estación para almacenado de datos
  {
    myFile = SD.open("ESTACION.csv", FILE_WRITE);
    if (myFile)
    {
      myFile.close();
    }
    else
    {
    }
  }
  dht1.begin();
  dht2.begin();
  pinMode(13, OUTPUT);
```

```
  pinMode(valvula0, OUTPUT);
  pinMode(valvula1, OUTPUT);
  pinMode(valvula2, OUTPUT);
  pinMode(valvula3, OUTPUT);
  pinMode(valvula4, OUTPUT);
  pinMode(valvula5, OUTPUT);
  pinMode(valvula6, OUTPUT);
  pinMode(valvula7, OUTPUT);
  pinMode(valvula8, OUTPUT);
  pinMode(valvula9, OUTPUT);
  pinMode(valvula10, OUTPUT);
  pinMode(valvula11, OUTPUT);
  pinMode(valvula12, OUTPUT);
  pinMode(valvula13, OUTPUT);
  pinMode(iluminacion, OUTPUT);

  digitalWrite(valvula0, HIGH);
  digitalWrite(valvula1, HIGH);
  digitalWrite(valvula2, HIGH);
  digitalWrite(valvula3, HIGH);
  digitalWrite(valvula4, HIGH);
  digitalWrite(valvula5, HIGH);
  digitalWrite(valvula6, HIGH);
  digitalWrite(valvula7, HIGH);
  digitalWrite(valvula8, HIGH);
  digitalWrite(valvula9, HIGH);
  digitalWrite(valvula10, HIGH);
  digitalWrite(valvula11, HIGH);
  digitalWrite(valvula12, HIGH);
  digitalWrite(iluminacion, HIGH);
  digitalWrite (valvula13, HIGH);
}
```

```
 //dht22
void dht22_1()

{

if(millis()-tiempoUltimaLectura>2000)
  {

  humedadi1 = dht1.readHumidity();
  temperaturai1 = dht1.readTemperature();
  humedadi2 = dht2.readHumidity();
  temperaturai2 = dht2.readTemperature();
  h1= (humedadi1+humedadi2)/2;
  t1= (temperaturai1+temperaturai2)/2;
  Serial.print(h1);
  Serial.print(" ;");
  Serial.print(t1);
  Serial.print(" ;");
  tiempoUltimaLectura=millis(); //actualizamos el tiempo de la última lectura
  }
  digitalWrite(13, HIGH);
  delay(100);
  digitalWrite(13, LOW);
  delay(100);
}


// ultrasonico sumegible
void ultrasonicosumegible()
{
  //Ultrasonico sumergible
  pinMode(Trigger 2, OUTPUT);
```

```
  digitalWrite(Trigger_2, LOW);
  delayMicroseconds(2);
  digitalWrite(Trigger_2, HIGH);
  delayMicroseconds(10);
  digitalWrite(Trigger_2, LOW);
  const unsigned long duration = pulseIn(Echo_2, HIGH);
  distance1 = duration / 29 / 2;
  if (duration == 0)
  {
  }
  else
  {

    Serial.print(distance1);
    Serial.print(" ;");
  }
}

// temperatura sumergible
void temperaturasumergible1()
{
  sensors1.begin();   //Se inicia el sensor 1
  sensors1.requestTemperatures();   //Se envía el comando para leer la temperatura
  temp1 = sensors1.getTempCByIndex(0); //Se obtiene la temperatura en °C del sensor 1
  Serial.print(temp1);
  Serial.print(" ;");

}

// Humedad suelo 1
void Humedadsuelo1()
{
```

```
  porcentaje1 = map(valor_sensor, Valor_Sensor_Agua, Valor_Sensor_Aire, 100, 0);
  if (porcentaje1 < 0) porcentaje1 = 0;
  if (porcentaje1 > 100) porcentaje1 = 100;

  Serial.print(porcentaje1);
  Serial.print(" ;");

}

// Humedad suelo 2
void Humedadsuelo2()
{
  valor_sensor = analogRead(AOUT1);
  porcentaje2 = map(valor_sensor, Valor_Sensor_Agua, Valor_Sensor_Aire, 100, 0);
  if (porcentaje2 < 0) porcentaje2 = 0;
  if (porcentaje2 > 100) porcentaje2 = 100;
  Serial.print(porcentaje2);
  Serial.print(" ;");

}

// Humedad suelo 3
void Humedadsuelo3()
{
  valor_sensor = analogRead(AOUT2);
  porcentaje3 = map(valor_sensor, Valor_Sensor_Agua, Valor_Sensor_Aire, 100, 0);
  if (porcentaje3 < 0) porcentaje3 = 0;
  if (porcentaje3 > 100) porcentaje3 = 100;
  Serial.print(porcentaje3);
  Serial.print(" ;");

}
```

```
// Humedad suelo 4
void Humedadsuelo4()
{
  valor_sensor = analogRead(AOUT3);
  porcentaje4 = map(valor_sensor, Valor_Sensor_Agua, Valor_Sensor_Aire, 100, 0);
  if (porcentaje4 < 0) porcentaje4 = 0;
  if (porcentaje4 > 100) porcentaje4 = 100;
  Serial.print(porcentaje4);
  Serial.print(" ;");

}

// Humedad suelo 5
void Humedadsuelo5()
{
  valor_sensor = analogRead(AOUT4);
  porcentaje5 = map(valor_sensor, Valor_Sensor_Agua, Valor_Sensor_Aire, 100, 0);
  if (porcentaje5 < 0) porcentaje5 = 0;
  if (porcentaje5 > 100) porcentaje5 = 100;
  Serial.print(porcentaje5);
  Serial.print(" ;");

}

// Humedad suelo 6
void Humedadsuelo6()
{
  valor_sensor = analogRead(AOUT5);
  porcentaje6 = map(valor_sensor, Valor_Sensor_Agua, Valor_Sensor_Aire, 100, 0);
  if (porcentaje6 < 0) porcentaje6 = 0;
  if (porcentaje6 > 100) porcentaje6 = 100;
  Serial.print(porcentaje6);
```

```
  Serial.print(" ;");

}

// Humedad suelo 7
void Humedadsuelo7()
{
  valor_sensor = analogRead(AOUT6);
  porcentaje7 = map(valor_sensor, Valor_Sensor_Agua, Valor_Sensor_Aire, 100, 0);
  if (porcentaje7 < 0) porcentaje7 = 0;
  if (porcentaje7 > 100) porcentaje7 = 100;
  Serial.print(porcentaje7);
  Serial.print(" ;");
}

// Humedad suelo 8
void Humedadsuelo8()
{
  valor_sensor = analogRead(AOUT7);
  porcentaje8 = map(valor_sensor, Valor_Sensor_Agua, Valor_Sensor_Aire, 100, 0);
  if (porcentaje8 < 0) porcentaje8 = 0;
  if (porcentaje8 > 100) porcentaje8 = 100;
  Serial.print(porcentaje8);
  Serial.print(" ;");

}

// Humedad suelo 9
void Humedadsuelo9()
{
  valor_sensor = analogRead(AOUT8);
  porcentaje9 = map(valor_sensor, Valor_Sensor_Agua, Valor_Sensor_Aire, 100, 0);
```

```
  if (porcentaje9 < 0) porcentaje9 = 0;
  if (porcentaje9 > 100) porcentaje9 = 100;
  Serial.print(porcentaje9);
  Serial.print(" ;");

}
  // Humedad suelo 10
void Humedadsuelo10()
{
  valor_sensor = analogRead(AOUT9);
  porcentaje10 = map(valor_sensor, Valor_Sensor_Agua, Valor_Sensor_Aire, 100, 0);
  if (porcentaje10 < 0) porcentaje10 = 0;
  if (porcentaje10 > 100) porcentaje10 = 100;
  Serial.print(porcentaje10);
  Serial.print(" ;");
}

// Humedad suelo 11
void Humedadsuelo11()
{
  valor_sensor = analogRead(AOUT10);
  porcentaje11 = map(valor_sensor, Valor_Sensor_Agua, Valor_Sensor_Aire, 100, 0);
  if (porcentaje11 < 0) porcentaje11 = 0;
  if (porcentaje11 > 100) porcentaje11 = 100;
  Serial.print(porcentaje11);
  Serial.print(" ;");
}

// Humedad suelo 12
void Humedadsuelo12()
{
  valor_sensor = analogRead(AOUT11);
```

```
  porcentaje12 = map(valor_sensor, Valor_Sensor_Agua, Valor_Sensor_Aire, 100, 0);
  if (porcentaje12 < 0) porcentaje12 = 0;
  if (porcentaje12 > 100) porcentaje12 = 100;
  Serial.print(porcentaje12);
  Serial.print(" ;");

}

void fecha()
{
  DateTime now = rtc.now();
  /*Serial.print(now.day());
  Serial.print('/');
  Serial.print(now.month());
  Serial.print('/');
  Serial.print(now.year());
  Serial.print(" ");
  Serial.print(now.hour());
  Serial.print(':');
  Serial.print(now.minute());
  Serial.print(':');
  Serial.print(now.second());
  Serial.println(" ;");*/
  //delay(100);
  yyear = now.year();
  mes= now.month();
  dia= now.day();
  hora = now.hour();
  minuto = now.minute();
  segundo= now.second();

}
```

```
void electrovalvulas_f(char Dato_v)
{
  if (Dato_v=='C')                                  ////////// Se activa riego manual
  {
    digitalWrite(valvula0, LOW);
    digitalWrite(valvula1, LOW);
    digitalWrite(valvula2, LOW);
    digitalWrite(valvula3, LOW);
    digitalWrite(valvula4, LOW);
    digitalWrite(valvula5, LOW);
    digitalWrite(valvula6, LOW);
    digitalWrite(valvula7, LOW);
    digitalWrite(valvula8, LOW);
    digitalWrite(valvula9, LOW);
    digitalWrite(valvula10, LOW);
    digitalWrite(valvula11, LOW);
    digitalWrite(valvula12, LOW);
    delay (100);
    a=1;
    q=1;
    }
    if ( hora==hora1 && minuto==minuto1+2 && q==1)
    {
      digitalWrite(valvula0, HIGH);
      digitalWrite(valvula1, HIGH);
      digitalWrite(valvula2, HIGH);
      digitalWrite(valvula3, HIGH);
      digitalWrite(valvula4, HIGH);
      digitalWrite(valvula5, HIGH);
      digitalWrite(valvula6, HIGH);
      digitalWrite(valvula7, HIGH);
      digitalWrite(valvula8, HIGH);
```

```
    digitalWrite(valvula8, HIGH);
    digitalWrite(valvula9, HIGH);
    digitalWrite(valvula10, HIGH);
    digitalWrite(valvula11, HIGH);
    digitalWrite(valvula12, HIGH);
    a=0;
    q=0;
    if (hora==1 || hora==2 || hora==3 || hora==4 || hora==5 || hora==6)
    {
      b=1;
    }
    if (hora==13 || hora==15 || hora==16 || hora==17 || hora==18 || hora==19)
    {
      c=1;
    }
    delay(100);
  }

if (a==0)                              ////////// Se activa riego automático
{
  if (b==0)
  {
    if (hora==10 && minuto <3)          // Se enciende las electrovalvulas por 1 minutos
    {
      if (hora == 10 &&  minuto > 1)
      {
        /*digitalWrite(valvula0, LOW);
        digitalWrite(valvula1, LOW);
        digitalWrite(valvula2, LOW);
        digitalWrite(valvula3, LOW);
        digitalWrite(valvula4, LOW);
        digitalWrite(valvula5, LOW);
```

```
      digitalWrite(valvula5, LOW);
      digitalWrite(valvula6, LOW);
      digitalWrite(valvula7, LOW);
      digitalWrite(valvula8, LOW);
      digitalWrite(valvula9, LOW);
      digitalWrite(valvula10, LOW);
      digitalWrite(valvula11, LOW);
      digitalWrite(valvula12, LOW);
      delay (100);*/
    }
  }
  if (hora==10 && minuto ==3)
  {
    /*digitalWrite(valvula0, HIGH);
    digitalWrite(valvula1, HIGH);
    digitalWrite(valvula2, HIGH);
    digitalWrite(valvula3, HIGH);
    digitalWrite(valvula4, HIGH);
    digitalWrite(valvula5, HIGH);
    digitalWrite(valvula6, HIGH);
    digitalWrite(valvula7, HIGH);
    digitalWrite(valvula8, HIGH);
    digitalWrite(valvula9, HIGH);
    digitalWrite(valvula10, HIGH);
    digitalWrite(valvula11, HIGH);
    digitalWrite(valvula12, HIGH);
    delay(100);*/
  }
}
if (c==0)
{
  if (hora==14 && minuto <3)          // Se enciende las electrovalvulas por 1 minutos
```

```
  if (hora == 14 &&  minuto > 1)
  {
    /*digitalWrite(valvula0, LOW);
    digitalWrite(valvula1, LOW);
    digitalWrite(valvula2, LOW);
    digitalWrite(valvula3, LOW);
    digitalWrite(valvula4, LOW);
    digitalWrite(valvula5, LOW);
    digitalWrite(valvula6, LOW);
    digitalWrite(valvula7, LOW);
    digitalWrite(valvula8, LOW);
    digitalWrite(valvula9, LOW);
    digitalWrite(valvula10, LOW);
    digitalWrite(valvula11, LOW);
    digitalWrite(valvula12, LOW);
    delay (100); */
  }
}
if (hora==14 && minuto ==3)
{
  /*digitalWrite(valvula0, HIGH);
  digitalWrite(valvula1, HIGH);
  digitalWrite(valvula2, HIGH);
  digitalWrite(valvula3, HIGH);
  digitalWrite(valvula4, HIGH);
  digitalWrite(valvula5, HIGH);
  digitalWrite(valvula6, HIGH);
  digitalWrite(valvula7, HIGH);
  digitalWrite(valvula8, HIGH);
  digitalWrite(valvula9, HIGH);
  digitalWrite(valvula10, HIGH);
  digitalWrite(valvula11, HIGH);
```

```
        digitalWrite(valvula10, HIGH);
        digitalWrite(valvula11, HIGH);
        digitalWrite(valvula12, HIGH);
        delay(100); */
      }
    }
    if (hora==23)
    {
      b=0;
      c=0;
      d=0;
    }
    hora1=hora;
    minuto1=minuto;
  }
}

void iluminacion_f(char Dato_v)
{

  if (Dato_v=='A')
  {
    digitalWrite(iluminacion, LOW);
  }
  else if (Dato_v=='B')
  {
    digitalWrite(iluminacion, HIGH);
  }
}

void CO2()
{
```

```
void CO2()
{
 ////////////////////////////////////////////////////////////////////////////////

valueCO2 = co2Sensor.read();
Serial.print(valueCO2);
Serial.println(" ;");

}
void electrovalvula2()
{

    if (hora == 15 && minuto>1 && minuto<30)  // si la distancia entre el sensor y e
    {
      if (distance1 >= 100)
       {
        digitalWrite (valvula13, LOW);
       }
    }

   if (distance1 <=70)
   {
     digitalWrite (valvula13, HIGH);
   }


}

////////////////////////////////////////////////////////////////////////////////
char Dato = 'Z';

void serialEvent( )
```

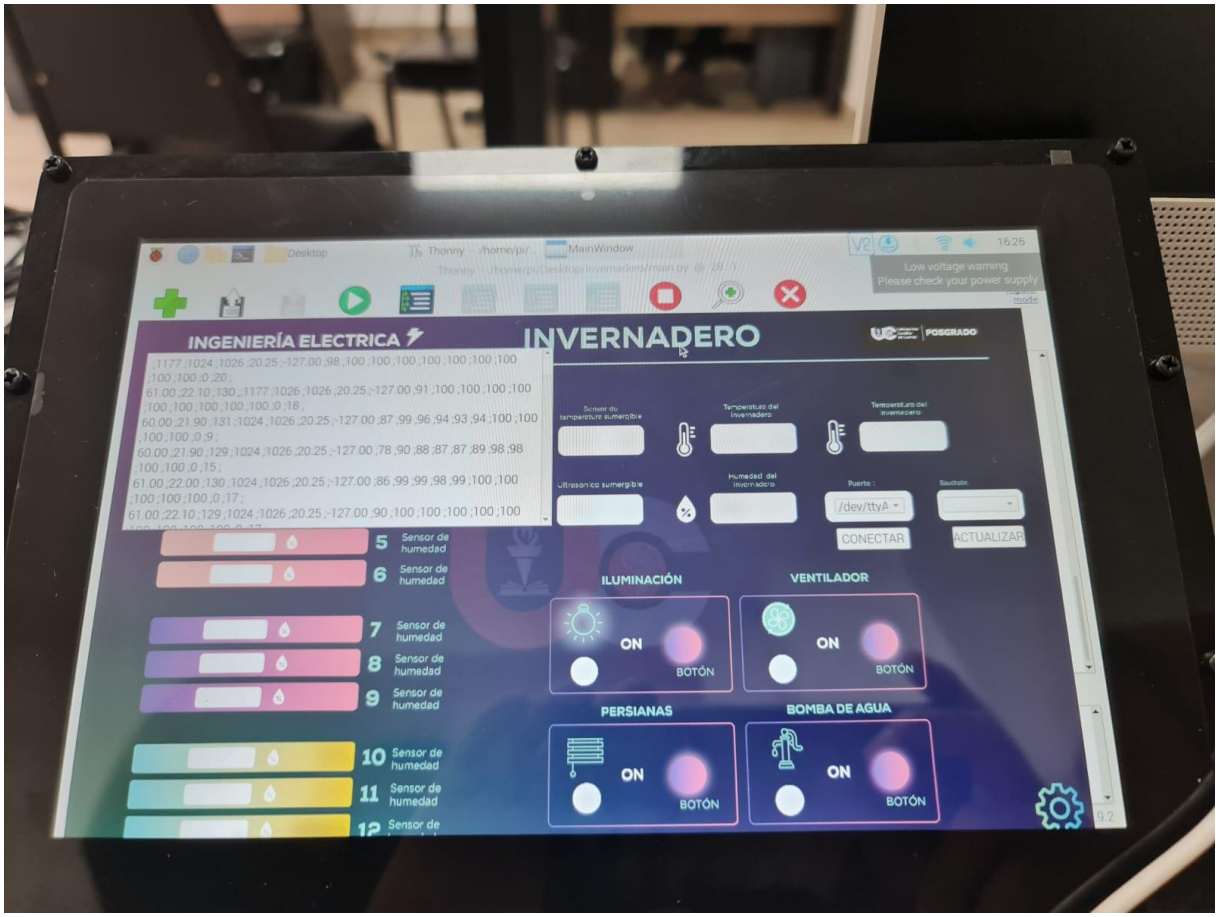
```
{
  if(Serial.available() > 0 )
  {
    Dato = Serial.read();
  }
}
//////////////////////////////////////////// FUNCION MODULO LECTOR MICRO SD //////
void almacenar()          //// Almacenamiento de datos
{
  myFile = SD.open("ESTACION.csv", FILE_WRITE);         // Abrimos el archivo Estac
    if (myFile)
    {
     // if (ret1==0)
     // {
        myFile.println("[año];[mes];[día];[hora];[minuto];[segundo];[Humedad %HR];[
        //ret1=1;
      //}
        myFile.print(";");
        myFile.print(";");
        myFile.print(yyear);        //// Escribiendo humedad
        myFile.print(";");
        myFile.print(mes);        //// Escribiendo humedad
        myFile.print(";");
        myFile.print(dia);        //// Escribiendo humedad
        myFile.print(";");
        myFile.print(hora);        //// Escribiendo humedad
        myFile.print(";");
        myFile.print(minuto);        //// Escribiendo humedad
        myFile.print(";");
        myFile.print(segundo);        //// Escribiendo humedad
        myFile.print(";");
```

```
myFile.print(t1);        //// Escribiendo Temperaturah22
myFile.print(";");
myFile.print(distance1);        //// Escribiendo Ultrasonico Sumergible
myFile.print(";");
myFile.print(temp1);        //// Escribiendo Temperatura Sumergible
myFile.print(";");
myFile.print(porcentaje1);        //// Escribiendo Humedad de suelo 1
myFile.print(";");
myFile.print(porcentaje2);        //// Escribiendo Humedad de suelo 2
myFile.print(";");
myFile.print(porcentaje3);        //// Escribiendo Humedad de suelo 3
myFile.print(";");
myFile.print(porcentaje4);        //// Escribiendo Humedad de suelo 4
myFile.print(";");
myFile.print(porcentaje5);        //// Escribiendo Humedad de suelo 5
myFile.print(";");
myFile.print(porcentaje6);        //// Escribiendo Humedad de suelo 6
myFile.print(";");
myFile.print(porcentaje7);        //// Escribiendo Humedad de suelo 7
myFile.print(";");
myFile.print(porcentaje8);        //// Escribiendo Humedad de suelo 8
myFile.print(";");
myFile.print(porcentaje9);        //// Escribiendo Humedad de suelo 9
myFile.print(";");
myFile.print(porcentaje10);        //// Escribiendo Humedad de suelo 10
myFile.print(";");
myFile.print(porcentaje11);        //// Escribiendo Humedad de suelo 11
myFile.print(";");
myFile.print(porcentaje12);        //// Escribiendo Humedad de suelo 12
myFile.print(";");
myFile.print(valueCO2);        //// Escribiendo Co2
myFile.print(";");
```

```
        myFile.print(";");
        myFile.print(valueCO2);         //// Escribiendo Co2
        myFile.print(";");
        myFile.close();                 //// Cerramos el archivo
    }
}


void loop()
{
  dht22_1();
  ultrasonicosumegible();
  temperaturasumergible1();
  Humedadsuelo1();
  Humedadsuelo2();
  Humedadsuelo3();
  Humedadsuelo4();
  Humedadsuelo5();
  Humedadsuelo6();
  Humedadsuelo7();
  Humedadsuelo8();
  Humedadsuelo9();
  Humedadsuelo10();
  Humedadsuelo11();
  Humedadsuelo12();
  CO2();
  fecha();
  almacenar(); /// Almacenar datos en la memoria SD
  electrovalvula2();
  iluminacion_f(Dato);
  electrovalvulas_f(Dato);
}
```

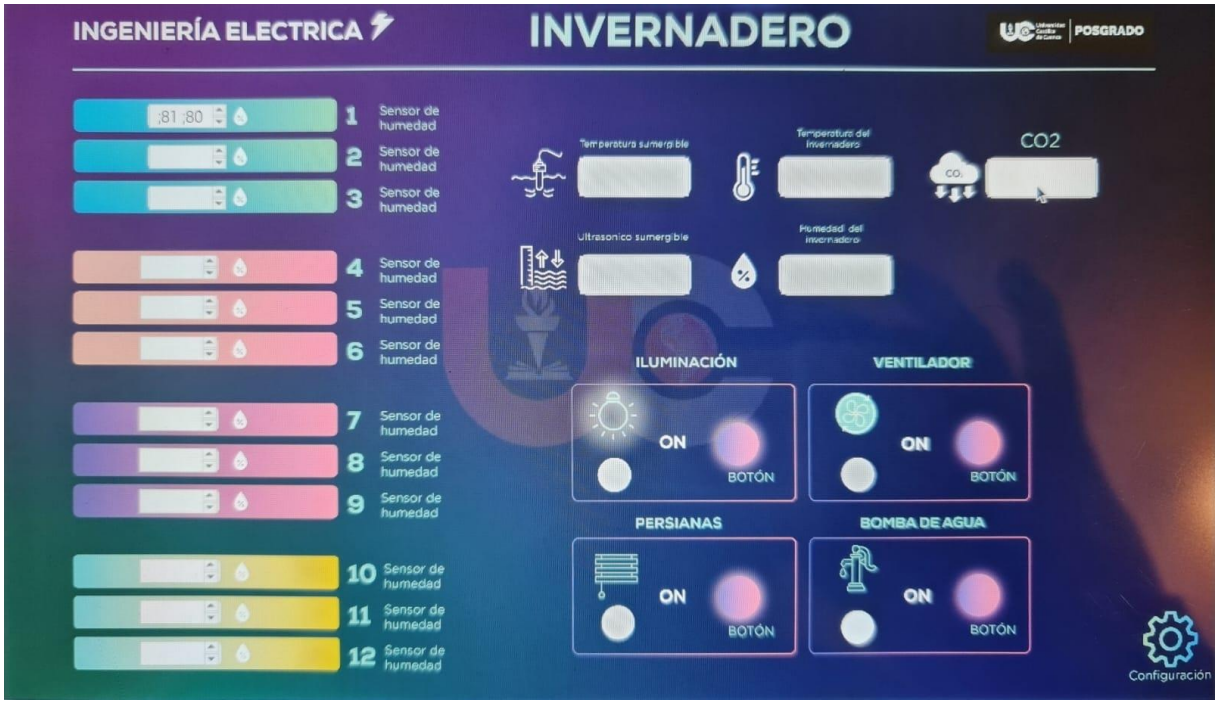
Anexo11:Código de ejecución completo de Arduino.

Fuente: autor



Anexo12:Pruebas de enlace mediante puerto serial.

Fuente: autor



Anexo13:interfaz preliminar a la final.

Fuente: autor

```
from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(1277, 744)
        MainWindow.setStyleSheet("#connectBtn:checked{\n"
"background-color: rgb(115, 210, 22);\n"
"}")
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setEnabled(True)
        font = QtGui.QFont()
        font.setBold(False)
        font.setWeight(50)
        self.centralwidget.setFont(font)
        self.centralwidget.setObjectName("centralwidget")
        self.FONDO = QtWidgets.QLabel(self.centralwidget)
        self.FONDO.setEnabled(True)
        self.FONDO.setGeometry(QtCore.QRect(-5, -5, 1271, 720))
        font = QtGui.QFont()
        font.setBold(True)
        font.setWeight(75)
        self.FONDO.setFont(font)
        self.FONDO.setCursor(QtGui.QCursor(QtCore.Qt.WaitCursor))
        self.FONDO.setFocusPolicy(QtCore.Qt.NoFocus)
        self.FONDO.setStyleSheet("border-image: url(:/fondo/INTERFAZ INVERNADERO-02..jpg);")
        self.FONDO.setText("")
        self.FONDO.setObjectName("FONDO")
        self.connectBtn = QtWidgets.QPushButton(self.centralwidget)
        self.connectBtn.setGeometry(QtCore.QRect(1340, 1080, 93, 28))
        self.connectBtn.setCheckable(True)
        self.connectBtn.setObjectName("connectBtn")
        self.clearBtn = QtWidgets.QPushButton(self.centralwidget)
        self.clearBtn.setGeometry(QtCore.QRect(840, 1370, 93, 28))
        self.clearBtn.setObjectName("clearBtn")
        self.sendBtn = QtWidgets.QPushButton(self.centralwidget)
        self.sendBtn.setGeometry(QtCore.QRect(1200, 1370, 93, 28))
        self.sendBtn.setObjectName("sendBtn")
        self.BaudList = QtWidgets.QComboBox(self.centralwidget)
        self.BaudList.setGeometry(QtCore.QRect(1350, 940, 73, 22))
```

```
        self.BaudList.setObjectName("BaudList")
        self.input = QtWidgets.QLineEdit(self.centralwidget)
        self.input.setGeometry(QtCore.QRect(950, 1370, 237, 22))
        self.input.setObjectName("input")
        self.portList = QtWidgets.QComboBox(self.centralwidget)
        self.portList.setGeometry(QtCore.QRect(1350, 790, 73, 22))
        self.portList.setObjectName("portList")
        self.updateBtn = QtWidgets.QPushButton(self.centralwidget)
        self.updateBtn.setGeometry(QtCore.QRect(1340, 1040, 93, 28))
        self.updateBtn.setObjectName("updateBtn")
        self.layoutWidget = QtWidgets.QWidget(self.centralwidget)
        self.layoutWidget.setGeometry(QtCore.QRect(1400, 670, 211, 32))
        self.layoutWidget.setObjectName("layoutWidget")
        self.horizontalLayout = QtWidgets.QHBoxLayout(self.layoutWidget)
        self.horizontalLayout.setContentsMargins(0, 0, 0, 0)
        self.horizontalLayout.setObjectName("horizontalLayout")
        self.portList_2 = QtWidgets.QComboBox(self.layoutWidget)
        self.portList_2.setObjectName("portList_2")
        self.horizontalLayout.addWidget(self.portList_2)
        self.input_2 = QtWidgets.QLineEdit(self.layoutWidget)
        self.input_2.setObjectName("input_2")
        self.horizontalLayout.addWidget(self.input_2)
        self.BaudList_2 = QtWidgets.QComboBox(self.layoutWidget)
        self.BaudList_2.setObjectName("BaudList_2")
        self.horizontalLayout.addWidget(self.BaudList_2)
        self.portList_2.raise_()
        self.BaudList_2.raise_()
        self.input_2.raise_()
        self.HUM1 = QtWidgets.QLineEdit(self.centralwidget)
        self.HUM1.setEnabled(True)
        self.HUM1.setGeometry(QtCore.QRect(170, 135, 51, 20))
        font = QtGui.QFont()
        font.setBold(True)
        font.setWeight(75)
        self.HUM1.setFont(font)
        self.HUM1.setCursor(QtGui.QCursor(QtCore.Qt.WaitCursor))
        self.HUM1.setFocusPolicy(QtCore.Qt.NoFocus)
        self.HUM1.setLayoutDirection(QtCore.Qt.LeftToRight)
        self.HUM1.setStyleSheet("border: 1px solid;\n"
"border-radius: 10px;\n"
"border-color: rgb(255,255,255);\n"
```

```
"border-color: rgb(255,255,255);\n"
"")
        self.HUM1.setObjectName("HUM1")
        self.HUM2 = QtWidgets.QLineEdit(self.centralwidget)
        self.HUM2.setEnabled(True)
        self.HUM2.setGeometry(QtCore.QRect(170, 172, 51, 20))
        font = QtGui.QFont()
        font.setBold(True)
        font.setWeight(75)
        self.HUM2.setFont(font)
        self.HUM2.setCursor(QtGui.QCursor(QtCore.Qt.WaitCursor))
        self.HUM2.setFocusPolicy(QtCore.Qt.NoFocus)
        self.HUM2.setStyleSheet("border: 1px solid;\n"
"border-radius: 10px;\n"
"border-color: rgb(255,255,255);\n"
"")
        self.HUM2.setObjectName("HUM2")
        self.HUM3 = QtWidgets.QLineEdit(self.centralwidget)
        self.HUM3.setEnabled(True)
        self.HUM3.setGeometry(QtCore.QRect(169, 210, 51, 20))
        font = QtGui.QFont()
        font.setBold(True)
        font.setWeight(75)
        self.HUM3.setFont(font)
        self.HUM3.setCursor(QtGui.QCursor(QtCore.Qt.WaitCursor))
        self.HUM3.setFocusPolicy(QtCore.Qt.NoFocus)
        self.HUM3.setStyleSheet("border: 1px solid;\n"
"border-radius: 10px;\n"
"border-color: rgb(255,255,255);\n"
"")
        self.HUM3.setObjectName("HUM3")
        self.HUM4 = QtWidgets.QLineEdit(self.centralwidget)
        self.HUM4.setEnabled(True)
        self.HUM4.setGeometry(QtCore.QRect(169, 272, 51, 20))
        font = QtGui.QFont()
        font.setBold(True)
        font.setWeight(75)
        self.HUM4.setFont(font)
        self.HUM4.setCursor(QtGui.QCursor(QtCore.Qt.WaitCursor))
        self.HUM4.setFocusPolicy(QtCore.Qt.NoFocus)
        self.HUM4.setStyleSheet("border: 1px solid;\n"
```

```
123     "border-radius: 10px;\n"
124     "border-color: rgb(255,255,255);\n"
125     "")
126             self.HUM4.setObjectName("HUM4")
127             self.HUM5 = QtWidgets.QLineEdit(self.centralwidget)
128             self.HUM5.setEnabled(True)
129             self.HUM5.setGeometry(QtCore.QRect(169, 310, 51, 20))
130             font = QtGui.QFont()
131             font.setBold(True)
132             font.setWeight(75)
133             self.HUM5.setFont(font)
134             self.HUM5.setCursor(QtGui.QCursor(QtCore.Qt.WaitCursor))
135             self.HUM5.setFocusPolicy(QtCore.Qt.NoFocus)
136             self.HUM5.setStyleSheet("border: 1px solid;\n"
137     "border-radius: 10px;\n"
138     "border-color: rgb(255,255,255);\n"
139     "")
140             self.HUM5.setObjectName("HUM5")
141             self.HUM6 = QtWidgets.QLineEdit(self.centralwidget)
142             self.HUM6.setEnabled(True)
143             self.HUM6.setGeometry(QtCore.QRect(169, 345, 51, 20))
144             font = QtGui.QFont()
145             font.setBold(True)
146             font.setWeight(75)
147             self.HUM6.setFont(font)
148             self.HUM6.setCursor(QtGui.QCursor(QtCore.Qt.WaitCursor))
149             self.HUM6.setFocusPolicy(QtCore.Qt.NoFocus)
150             self.HUM6.setStyleSheet("border: 1px solid;\n"
151     "border-radius: 10px;\n"
152     "border-color: rgb(255,255,255);\n"
153     "")
154             self.HUM6.setObjectName("HUM6")
155             self.HUM7 = QtWidgets.QLineEdit(self.centralwidget)
156             self.HUM7.setEnabled(True)
157             self.HUM7.setGeometry(QtCore.QRect(169, 410, 51, 20))
158             font = QtGui.QFont()
159             font.setBold(True)
160             font.setWeight(75)
161             self.HUM7.setFont(font)
162             self.HUM7.setCursor(QtGui.QCursor(QtCore.Qt.WaitCursor))
163             self.HUM7.setFocusPolicy(QtCore.Qt.NoFocus)
```

```
164         self.HUM7.setStyleSheet("border: 1px solid;\n"
165 "border-radius: 10px;\n"
166 "border-color: rgb(255,255,255);\n"
167 "")
168         self.HUM7.setObjectName("HUM7")
169         self.HUM8 = QtWidgets.QLineEdit(self.centralwidget)
170         self.HUM8.setEnabled(True)
171         self.HUM8.setGeometry(QtCore.QRect(169, 446, 51, 20))
172         font = QtGui.QFont()
173         font.setBold(True)
174         font.setWeight(75)
175         self.HUM8.setFont(font)
176         self.HUM8.setCursor(QtGui.QCursor(QtCore.Qt.WaitCursor))
177         self.HUM8.setFocusPolicy(QtCore.Qt.NoFocus)
178         self.HUM8.setStyleSheet("border: 1px solid;\n"
179 "border-radius: 10px;\n"
180 "border-color: rgb(255,255,255);\n"
181 "")
182         self.HUM8.setObjectName("HUM8")
183         self.HUM9 = QtWidgets.QLineEdit(self.centralwidget)
184         self.HUM9.setEnabled(True)
185         self.HUM9.setGeometry(QtCore.QRect(169, 483, 51, 20))
186         font = QtGui.QFont()
187         font.setBold(True)
188         font.setWeight(75)
189         self.HUM9.setFont(font)
190         self.HUM9.setCursor(QtGui.QCursor(QtCore.Qt.WaitCursor))
191         self.HUM9.setFocusPolicy(QtCore.Qt.NoFocus)
192         self.HUM9.setStyleSheet("border: 1px solid;\n"
193 "border-radius: 10px;\n"
194 "border-color: rgb(255,255,255);\n"
195 "")
196         self.HUM9.setObjectName("HUM9")
197         self.HUM10 = QtWidgets.QLineEdit(self.centralwidget)
198         self.HUM10.setEnabled(True)
199         self.HUM10.setGeometry(QtCore.QRect(169, 547, 51, 20))
200         font = QtGui.QFont()
201         font.setBold(True)
202         font.setWeight(75)
203         self.HUM10.setFont(font)
204         self.HUM10.setCursor(QtGui.QCursor(QtCore.Qt.WaitCursor))
```

```
206         self.HUM10.setStyleSheet("border: 1px solid;\n"
207 "border-radius: 10px;\n"
208 "border-color: rgb(255,255,255);\n"
209 "")
210         self.HUM10.setObjectName("HUM10")
211         self.HUM11 = QtWidgets.QLineEdit(self.centralwidget)
212         self.HUM11.setEnabled(True)
213         self.HUM11.setGeometry(QtCore.QRect(169, 583, 51, 20))
214         font = QtGui.QFont()
215         font.setBold(True)
216         font.setWeight(75)
217         self.HUM11.setFont(font)
218         self.HUM11.setCursor(QtGui.QCursor(QtCore.Qt.WaitCursor))
219         self.HUM11.setFocusPolicy(QtCore.Qt.NoFocus)
220         self.HUM11.setStyleSheet("border: 1px solid;\n"
221 "border-radius: 10px;\n"
222 "border-color: rgb(255,255,255);\n"
223 "")
224         self.HUM11.setObjectName("HUM11")
225         self.HUM12 = QtWidgets.QLineEdit(self.centralwidget)
226         self.HUM12.setEnabled(True)
227         self.HUM12.setGeometry(QtCore.QRect(169, 620, 51, 20))
228         font = QtGui.QFont()
229         font.setBold(True)
230         font.setWeight(75)
231         self.HUM12.setFont(font)
232         self.HUM12.setCursor(QtGui.QCursor(QtCore.Qt.WaitCursor))
233         self.HUM12.setFocusPolicy(QtCore.Qt.NoFocus)
234         self.HUM12.setStyleSheet("border: 1px solid;\n"
235 "border-radius: 10px;\n"
236 "border-color: rgb(255,255,255);\n"
237 "")
238         self.HUM12.setObjectName("HUM12")
239         self.TEMPSUMER = QtWidgets.QLineEdit(self.centralwidget)
240         self.TEMPSUMER.setEnabled(True)
241         self.TEMPSUMER.setGeometry(QtCore.QRect(610, 193, 60, 20))
242         font = QtGui.QFont()
243         font.setBold(True)
244         font.setWeight(75)
245         self.TEMPSUMER.setFont(font)
246         self.TEMPSUMER.setCursor(QtGui.QCursor(QtCore.Qt.WaitCursor))
```

```
246             self.TEMPSUMER.setCursor(QtGui.QCursor(QtCore.Qt.WaitCursor))
247             self.TEMPSUMER.setFocusPolicy(QtCore.Qt.NoFocus)
248             self.TEMPSUMER.setStyleSheet("border: 1px solid;\n"
249 "border-radius: 10px;\n"
250 "border-color: rgb(255,255,255);\n"
251 "")
252             self.TEMPSUMER.setObjectName("TEMPSUMER")
253             self.ULTRASUMER = QtWidgets.QLineEdit(self.centralwidget)
254             self.ULTRASUMER.setEnabled(True)
255             self.ULTRASUMER.setGeometry(QtCore.QRect(619, 290, 41, 20))
256             font = QtGui.QFont()
257             font.setBold(True)
258             font.setWeight(75)
259             self.ULTRASUMER.setFont(font)
260             self.ULTRASUMER.setCursor(QtGui.QCursor(QtCore.Qt.WaitCursor))
261             self.ULTRASUMER.setFocusPolicy(QtCore.Qt.NoFocus)
262             self.ULTRASUMER.setStyleSheet("border: 1px solid;\n"
263 "border-radius: 10px;\n"
264 "border-color: rgb(255,255,255);\n"
265 "")
266             self.ULTRASUMER.setObjectName("ULTRASUMER")
267             self.HUM = QtWidgets.QLineEdit(self.centralwidget)
268             self.HUM.setEnabled(True)
269             self.HUM.setGeometry(QtCore.QRect(835, 290, 55, 20))
270             palette = QtGui.QPalette()
271             brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
272             brush.setStyle(QtCore.Qt.SolidPattern)
273             palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.WindowText, brush)
274             brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
275             brush.setStyle(QtCore.Qt.SolidPattern)
276             palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.WindowText, brush)
277             brush = QtGui.QBrush(QtGui.QColor(185, 185, 185))
278             brush.setStyle(QtCore.Qt.SolidPattern)
279             palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.WindowText, brush)
280             self.HUM.setPalette(palette)
281             font = QtGui.QFont()
282             font.setFamily("PibotoLt")
283             font.setBold(True)
284             font.setWeight(75)
285             font.setStyleStrategy(QtGui.QFont.NoAntialias)
286             self.HUM.setFont(font)
```

```
287             self.HUM.setCursor(QtGui.QCursor(QtCore.Qt.WaitCursor))
288             self.HUM.setFocusPolicy(QtCore.Qt.NoFocus)
289             self.HUM.setContextMenuPolicy(QtCore.Qt.DefaultContextMenu)
290             self.HUM.setLayoutDirection(QtCore.Qt.LeftToRight)
291             self.HUM.setStyleSheet("border: 1px solid;\n"
292 "border-radius: 10px;\n"
293 "border-color: rgb(255,255,255);\n"
294 "")
295             self.HUM.setFrame(True)
296             self.HUM.setObjectName("HUM")
297             self.TEMP = QtWidgets.QLineEdit(self.centralwidget)
298             self.TEMP.setEnabled(True)
299             self.TEMP.setGeometry(QtCore.QRect(845, 194, 55, 20))
300             font = QtGui.QFont()
301             font.setBold(True)
302             font.setWeight(75)
303             self.TEMP.setFont(font)
304             self.TEMP.setCursor(QtGui.QCursor(QtCore.Qt.WaitCursor))
305             self.TEMP.setFocusPolicy(QtCore.Qt.NoFocus)
306             self.TEMP.setStyleSheet("border: 1px solid;\n"
307 "border-radius: 10px;\n"
308 "border-color: rgb(255,255,255);\n"
309 "")
310             self.TEMP.setObjectName("TEMP")
311             self.CO2 = QtWidgets.QLineEdit(self.centralwidget)
312             self.CO2.setEnabled(True)
313             self.CO2.setGeometry(QtCore.QRect(1089, 193, 41, 20))
314             font = QtGui.QFont()
315             font.setBold(True)
316             font.setWeight(75)
317             self.CO2.setFont(font)
318             self.CO2.setCursor(QtGui.QCursor(QtCore.Qt.WaitCursor))
319             self.CO2.setFocusPolicy(QtCore.Qt.NoFocus)
320             self.CO2.setStyleSheet("border: 1px solid;\n"
321 "border-radius: 10px;\n"
322 "border-color: rgb(255,255,255);\n"
323 "")
324             self.CO2.setObjectName("CO2")
325             self.updateBtn_2 = QtWidgets.QPushButton(self.centralwidget)
326             self.updateBtn_2.setGeometry(QtCore.QRect(1440, 650, 16, 30))
```

```
326             self.updateBtn_2.setGeometry(QtCore.QRect(1440, 650, 16, 30))
327             self.updateBtn_2.setObjectName("updateBtn_2")
328             self.sendBtn_2 = QtWidgets.QPushButton(self.centralwidget)
329             self.sendBtn_2.setGeometry(QtCore.QRect(1470, 650, 16, 30))
330             self.sendBtn_2.setObjectName("sendBtn_2")
331             self.clearBtn_2 = QtWidgets.QPushButton(self.centralwidget)
332             self.clearBtn_2.setGeometry(QtCore.QRect(1430, 680, 16, 21))
333             self.clearBtn_2.setObjectName("clearBtn_2")
334             self.connectBtn_2 = QtWidgets.QPushButton(self.centralwidget)
335             self.connectBtn_2.setGeometry(QtCore.QRect(1410, 650, 16, 16))
336             self.connectBtn_2.setCheckable(True)
337             self.connectBtn_2.setObjectName("connectBtn_2")
338             self.BOTVENT = QtWidgets.QPushButton(self.centralwidget)
339             self.BOTVENT.setEnabled(True)
340             self.BOTVENT.setGeometry(QtCore.QRect(980, 430, 51, 51))
341             self.BOTVENT.setStyleSheet("background-color: qlineargradient(spr
342 "border: 1px solid;\n"
343 "border-radius: 25px;")
344             self.BOTVENT.setText("")
345             self.BOTVENT.setObjectName("BOTVENT")
346             self.BOTILUM = QtWidgets.QPushButton(self.centralwidget)
347             self.BOTILUM.setEnabled(True)
348             self.BOTILUM.setGeometry(QtCore.QRect(730, 430, 51, 51))
349             self.BOTILUM.setStyleSheet("background-color: qlineargradient(spr
350 "border: 1px solid;\n"
351 "border-radius: 25px;")
352             self.BOTILUM.setText("")
353             self.BOTILUM.setObjectName("BOTILUM")
354             self.BOTPER = QtWidgets.QPushButton(self.centralwidget)
355             self.BOTPER.setEnabled(True)
356             self.BOTPER.setGeometry(QtCore.QRect(730, 565, 51, 51))
357             self.BOTPER.setStyleSheet("background-color: qlineargradient(spre
358 "border: 1px solid;\n"
359 "border-radius: 25px;")
360             self.BOTPER.setText("")
361             self.BOTPER.setObjectName("BOTPER")
362             self.BOTBOM = QtWidgets.QPushButton(self.centralwidget)
363             self.BOTBOM.setEnabled(True)
364             self.BOTBOM.setGeometry(QtCore.QRect(980, 565, 51, 51))
365             self.BOTBOM.setStyleSheet("background-color: qlineargradient(spre
366 "border: 1px solid;\n"
```

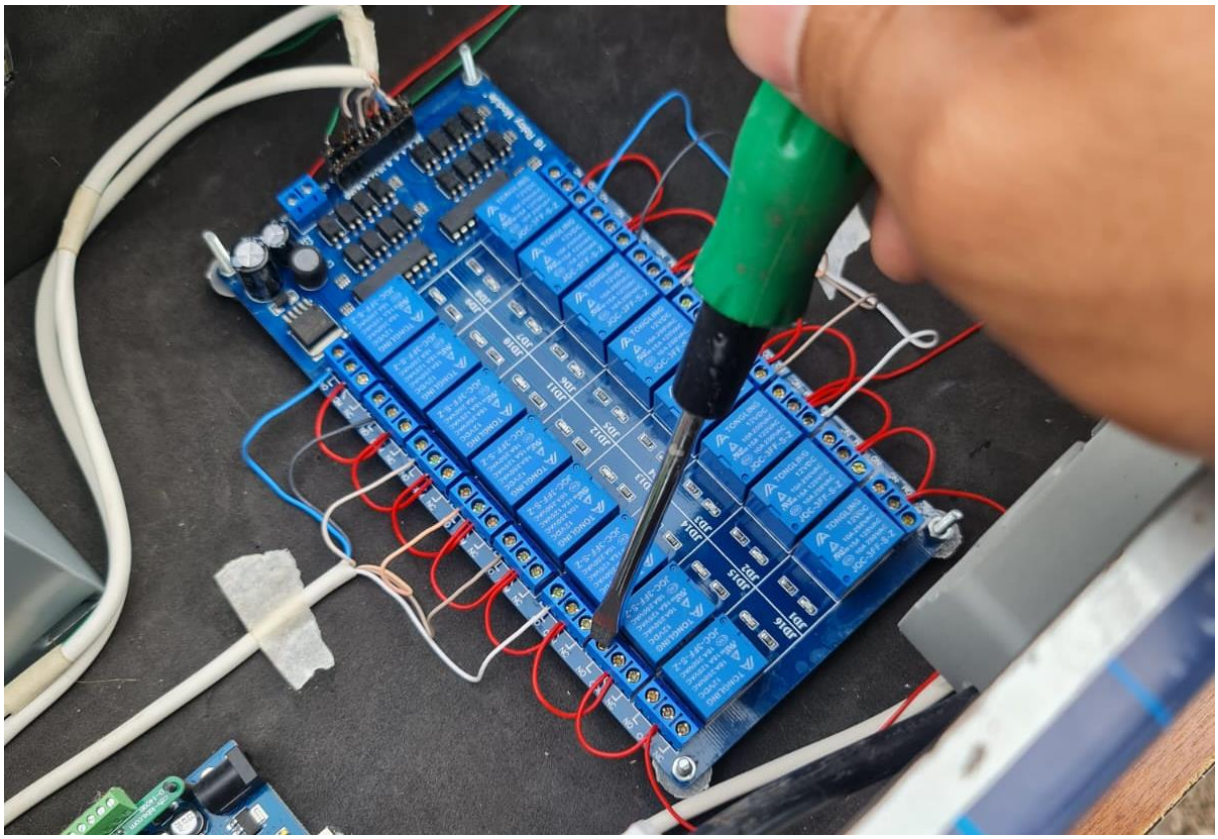
```
366     "border: 1px solid;\n"
367 "border-radius: 25px;")
368             self.BOTBOM.setText("")
369             self.BOTBOM.setObjectName("BOTBOM")
370             self.layoutWidget.raise_()
371             self.FONDO.raise_()
372             self.connectBtn.raise_()
373             self.clearBtn.raise_()
374             self.sendBtn.raise_()
375             self.BaudList.raise_()
376             self.input.raise_()
377             self.portList.raise_()
378             self.updateBtn.raise_()
379             self.HUM1.raise_()
380             self.HUM2.raise_()
381             self.HUM3.raise_()
382             self.HUM4.raise_()
383             self.HUM5.raise_()
384             self.HUM6.raise_()
385             self.HUM7.raise_()
386             self.HUM8.raise_()
387             self.HUM9.raise_()
388             self.HUM10.raise_()
389             self.HUM11.raise_()
390             self.HUM12.raise_()
391             self.TEMPSUMER.raise_()
392             self.ULTRASUMER.raise_()
393             self.TEMP.raise_()
394             self.CO2.raise_()
395             self.updateBtn_2.raise_()
396             self.sendBtn_2.raise_()
397             self.clearBtn_2.raise_()
398             self.connectBtn_2.raise_()
399             self.BOTVENT.raise_()
400             self.BOTILUM.raise_()
401             self.BOTPER.raise_()
402             self.BOTBOM.raise_()
403             self.HUM.raise_()
404             MainWindow.setCentralWidget(self.centralwidget)
```

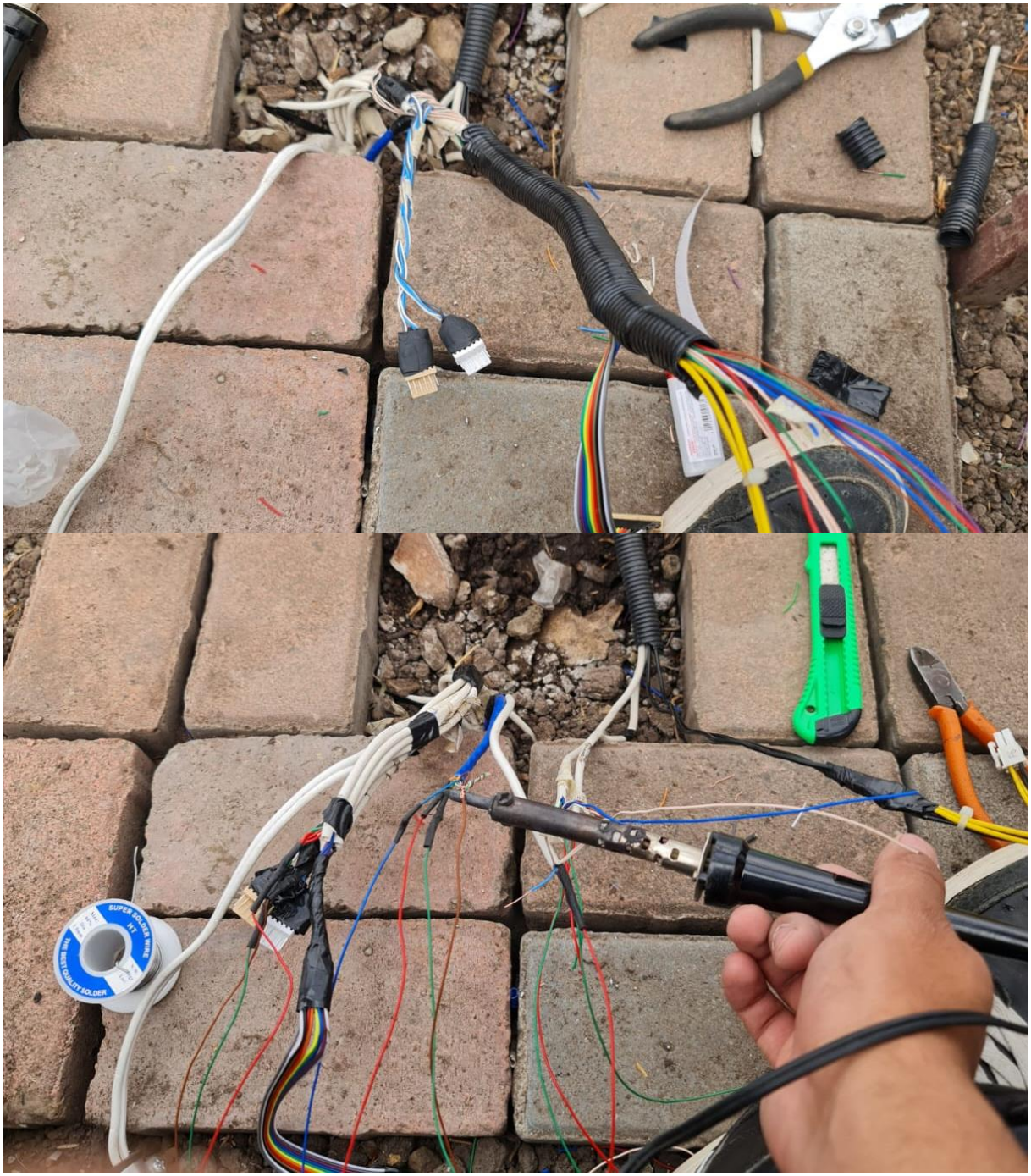
```
373             self.clearBtn.raise_()
374             self.sendBtn.raise_()
375             self.BaudList.raise_()
376             self.input.raise_()
377             self.portList.raise_()
378             self.updateBtn.raise_()
379             self.HUM1.raise_()
380             self.HUM2.raise_()
381             self.HUM3.raise_()
382             self.HUM4.raise_()
383             self.HUM5.raise_()
384             self.HUM6.raise_()
385             self.HUM7.raise_()
386             self.HUM8.raise_()
387             self.HUM9.raise_()
388             self.HUM10.raise_()
389             self.HUM11.raise_()
390             self.HUM12.raise_()
391             self.TEMPSUMER.raise_()
392             self.ULTRASUMER.raise_()
393             self.TEMP.raise_()
394             self.CO2.raise_()
395             self.updateBtn_2.raise_()
396             self.sendBtn_2.raise_()
397             self.clearBtn_2.raise_()
398             self.connectBtn_2.raise_()
399             self.BOTVENT.raise_()
400             self.BOTILUM.raise_()
401             self.BOTPER.raise_()
402             self.BOTBOM.raise_()
403             self.HUM.raise_()
404             MainWindow.setCentralWidget(self.centralwidget)
405             self.statusbar = QtWidgets.QStatusBar(MainWindow)
406             self.statusbar.setObjectName("statusbar")
407             MainWindow.setStatusBar(self.statusbar)
408
409             self.retranslateUi(MainWindow)
410             QtCore.QMetaObject.connectSlotsByName(MainWindow)
411
412 import fondo_rc
413
```

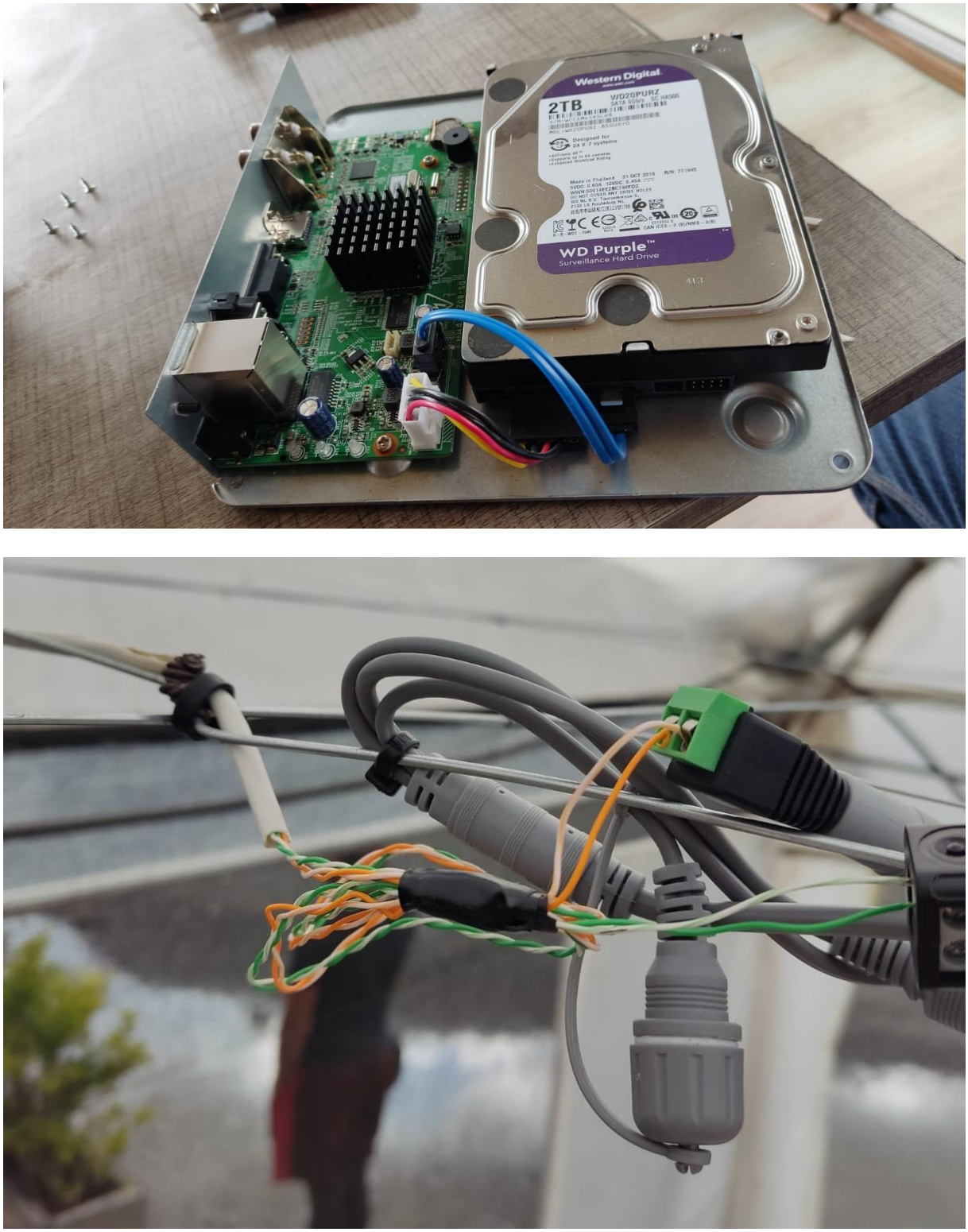
Anexo14:Código completo de Pantalla.py.

Fuente: autor


Pelemix
Soil
Sensor v2.0




SUPER SOLDER WIRE
THE BEST QUALITY SOLDER

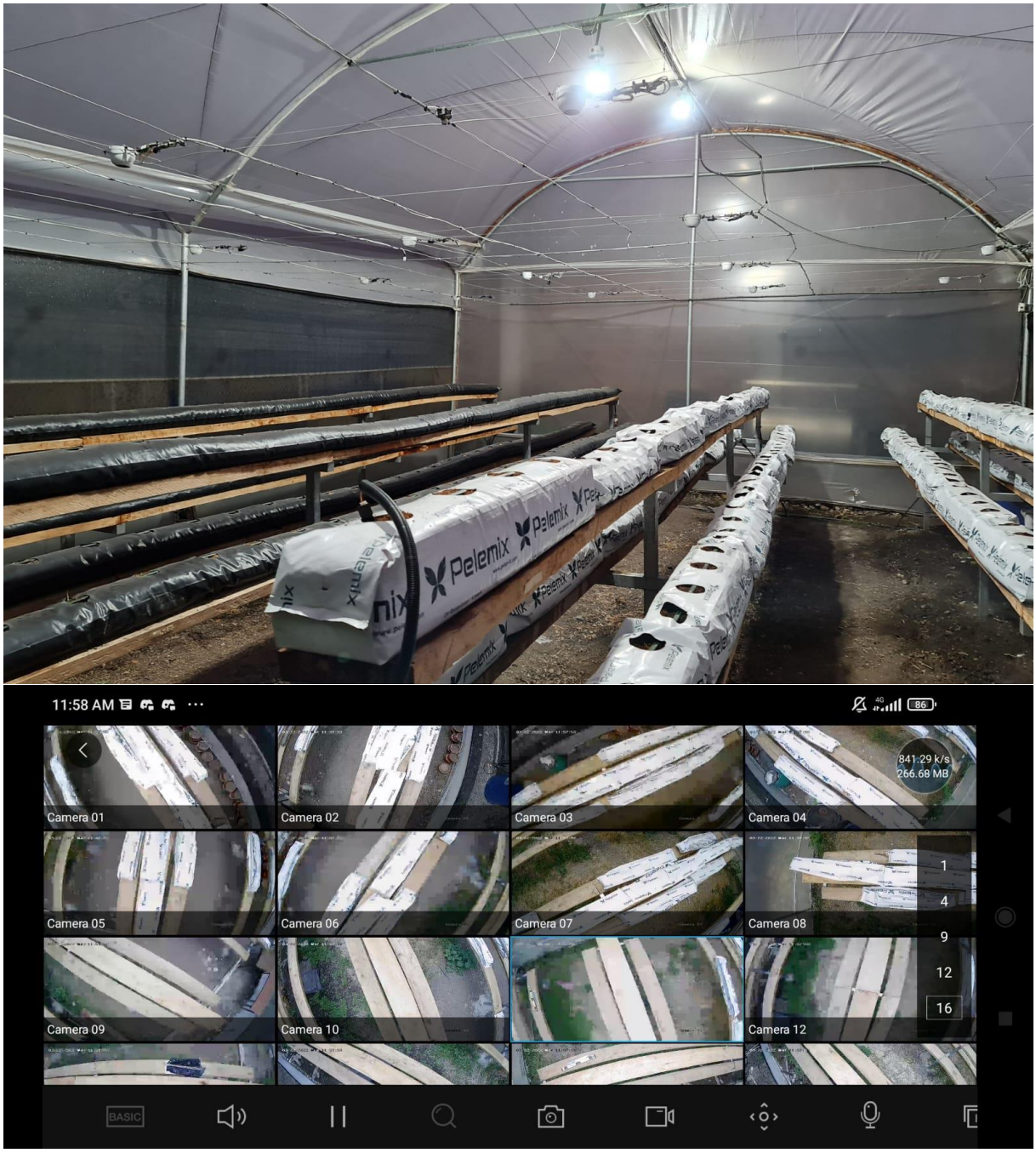


Anexo15:Instalación del circuito de control y monitoreo .

Fuente: autor


/dev/ttyACM0
Papelera
prueba1.py
TesisOnce10...
main.py
prueba
invernadero
Keyboard
sketch_jan23a Arduino 1.8.19
Archivo Editar Programa Herramientas Ayuda
sketch_jan23a
/dev/ttyACM0
Enviar
Fecha y hora actual: 23/1/2023 16:34:12
Fecha y hora actual: 23/1/2023 16:34:13
Fecha y hora actual: 23/1/2023 16:34:14
Fecha y hora actual: 23/1/2023 16:34:15
Fecha y hora actual: 23/1/2023 16:34:16
Fecha y hora actual: 23/1/2023 16:34:17
Fecha y hora actual: 23/1/2023 16:34:18
Fecha y hora actual: 23/1/2023 16:34:19
Fecha y hora actual: 23/1/2023 16:34:20
Fecha y hora actual: 23/1/2023 16:34:21
Fecha y hora actual: 23/1/2023 16:34:22
Fecha y hora actual: 23/1/2023 16:34:23
Fecha y hora actual: 23/1/2023 16:34:24
Fecha y hora actual: 23/1/2023 16:34:25
Fecha y hora actual: 23/1/2023 16:34:26
Autoscroll
Mostrar marca temporal
Nueva línea
9600 baudio
Limpiar salida
Arduino Uno en /dev/ttyACM0
Low voltage warning
Please check your power supply

Pelemix


Pelemix
11:58 AM
841.29 k/s
266.68 MB
Camera 01
Camera 02
Camera 03
Camera 04
Camera 05
Camera 06
Camera 07
Camera 08
Camera 09
Camera 10
Camera 12
1
4
9
12
16
BASIC

Anexo16:Pruebas del sistema de control y monitoreo .

Fuente: autor


Pelemix
www.pelemix.com


Pelemix
www.pelemix.com
UV Estabilizaer : 4 years

Anexo17::Resultados del invernadero semi hidropónico.

Fuente: autor

Anexo18::Resultados del invernadero semi hidropónico fruto.

Fuente: autor

Anexo19:Resultados del invernadero semi hidropónico frutos.

Fuente: autor

AUTORIZACION DE PUBLICACION EN EL REPOSITORIO INSTITUCIONAL

Yo, Christian Rafael Coyago Calle portador de la cédula de ciudadanía N.º 0106456361. En calidad de autor/a y titular de los derechos patrimoniales del trabajo de titulación "Automatización de un invernadero semi hidropónico" de conformidad a lo establecido en el artículo 114 Código Orgánico de la Economía Social de los Conocimientos, Creatividad e Innovación, reconozco a favor de la Universidad Católica de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos, Así mismo; autorizo a la Universidad para que realice la publicación de este trabajo de titulación en el Repositorio Institucional de conformidad a lo dispuesto en el artículo 144 de la Ley Orgánica de Educación Superior.

Cuenca, 19 de abril de 2022

F: .....

Christian Rafael Coyago Calle

0106456361