Anexos

A. Sensores de suelo disponibles en el mercado

Tal como ya se mencionó en este proyecto se buscó medir 4 variables las cuales son: temperatura, humedad, conductividad eléctrica y pH del suelo, es de resaltar que existen distintos métodos de medición de las variables del suelo, como puede ser capacitiva, inductivas, resistivas o espectroscopia, cada uno con sus ventajas o cualidades, dado que el cultivo está sembrado en materas es importante buscar sensores que se ubiquen en puntos fijos y envíen información confiable al sistema, debido a esto se deben descartar varios sensores basados espectroscopia o inducción que son diseñados para movilizar a lo largo del cultivo y realizar un mapeo del estado general, aunque cuentan con excelentes características no serían aptos para las necesidades del proyecto.

A continuación, se presenta un resumen de los sensores comerciales que miden las variables plateadas en este proyecto, donde se resumen sus características, métodos de medición, confiabilidad, disponibilidad y costos, esto con el fin de identificar los sensores más adecuados para el proyecto, adicionalmente es de resaltar que los sensores que se encontraron son fabricados en otros países, convirtiendo la disponibilidad de estos en una característica importante al momento de la elección del sensor, la información presentada a continuación fue extraída directamente de los manuales o páginas oficiales de los fabricantes.

JXBS-3001-TDR

Este sensor es fabricado por la empresa JXCT, con sede en Shandong – China, la tecnología utilizada es la reflectometría en el dominio del tiempo o TDR por sus siglas en inglés, en el cual se mide la constante dieléctrica del suelo a distintas profundidades con el fin de evaluar la humedad y conductividad eléctrica, adicionalmente otra característica que ofrece es que está diseñado para que los electrodos no entren en contacto directo con el suelo, aumentando considerablemente la vida útil del dispositivo, este sensor se puede encontrar en dos tamaños, en el pequeño se toman tres mediciones a 10, 20 y 30 cm de profundidad, y en el grande se toma una medida adicional a los 40 cm de profundidad.

Figura 1 Sensor JXBS-3001-TDR adaptado de *(JXCT, 2021)*

Este sensor puede medir otras variables adicionales a la humedad, temperatura y conductividad eléctrica del suelo, como lo son las concentraciones de nitrógeno, potasio y fosforo, dependiendo de la referencia solicitada al fabricante puede ser adquirido con cierta cantidad de mediciones, adicionalmente tiene incorporado RS-485 con protocolo Modbus, NB-IoT y Lora, lo cual facilita su implementación en distintos proyectos de IoT o de la industria, en la Figura 1 se muestra el dispositivo y los intervalos de medición, debido a que el sensor es fabricado en otra país fue necesario consultar su precio por una comercializadora internacional como Aliexpress, donde es de aproximadamente 350 dólares y con un tiempo de entrega de un mes, en la Tabla 1 se muestran las especificaciones técnicas del dispositivo.

Tabla 1 Especificaciones técnicas JXBS-3001-TDR adaptado de *(JXCT, 2021)*

Especificación	**Valor**
Voltaje de alimentación	12 V
Rango humedad	0-100%
Precisión humedad	0-53% (±3%) / 53-100% (±5%)
Resolución humedad	1%
Rango temperatura	-20 a 80 °C
Precisión temperatura	±0.5 °C
Resolución temperatura	0.1 °C
Rango CE	0 a 10000 µS/cm
Precisión CE	± 10 µS/cm
Resolución CE	2 µS/cm
Comunicación	RS485/Lora
Grado de protección	IP67
Temperatura de operación	-20 a 80 °C

Humedad relativa	0 A 95 %
Costo aproximado	350 USD

Una de las desventajas que tiene este sensor es que en ninguna de sus referencias mide el pH del suelo, pero por otra parte su resolución y precisión en las demás variables son buenas y puede dar confiabilidad al sistema, otra cualidad muy importante es que con un solo sensor se puede monitorear distintas profundidades y tener una mejor aproximado del movimiento del agua en el suelo, controlando con un solo sensor la lixiviación en la matera, aun así sus principales desventajas para el presente proyecto son su disponibilidad y elevado costo, en el cual se deberían adquirir mínimo tres sensores con un costo aproximado de 1.050 USD, haciendo que el proyecto pierda viabilidad en la inversión.

GroPoint Pro SDI-12 soil moisture sensor

Este sensor es fabricado por la empresa Riot Technology Corp, con sede en North Saanich – Canadá, la tecnología utilizada es la transmisión en el dominio del tiempo o TDT por sus siglas en inglés, esta tecnología ha sido patentada por la misma empresa y es una mejora de la TDR, ofreciendo una mejor precisión en la medición, el sensor tiene un tamaño de 20 cm de largo y puede ser instalado de forma vertical u horizontal, en el primer caso promedia la humedad de una capa de aproximadamente 15 cm y por medio de un sensor frontal indica cuando el agua ha llegado a la zona radicular, en el segundo caso mide la humedad en una profundidad específica, el fabricante recomienda que el sensor sea de uso permanente o semipermanente.

Figura 2 Sensor GroPoint Pro adaptado de *(Riot Technology Corp, 2021)*

Este sensor puede medir la humedad, temperatura y conductividad eléctrica del suelo, adicionalmente la medición del frente de humectación indicará cuando el agua ha llegado al fondo de la sonda lo que le permite detener el riego precisamente en el momento óptimo para garantizar que solo se aplique el agua necesaria, su diseño interno consta de una antena tejida la cual no entra en contacto directo con el suelo, aumentando así su vida útil, adicionalmente cada medición envía 400.000 pulsos para generar datos a los cuales se les aplica un filtrado avanzado con el objetivo de eliminar ruido y promediar la salida, por ultimo utiliza como salida el protocolo SDI-12 para la comunicación, lo cual permite tener un longitud de cable de 610 metros para 10 sensores, con aproximadamente 61 metros para cada sensor esto siendo lo recomendado por el fabricante, en la Figura 2 se muestra el dispositivo, debido a que el sensor es fabricado en otra país fue necesario consultar su precio por una comercializadora internacional como TEquipment, donde es de aproximadamente 500 dólares y con un tiempo de entrega de un mes, en la Tabla 2 se muestran las especificaciones técnicas del dispositivo.

Tabla 2 Especificaciones técnicas GroPoint Pro adaptado de *(Riot Technology Corp, 2021)*

Especificación	Valor
Voltaje de alimentación	6 a 14 VDC
Rango humedad	0-100%
Precisión humedad	±2.0%
Resolución humedad	0.2%
Rango temperatura	-20 a 70 °C
Precisión temperatura	±0.5 °C
Rango CE	0 a 4 dS/m
Precisión CE	± 3%
Comunicación	SDI-12 v.1.3
Grado de protección	IP68
Temperatura de operación	-20 a 70 °C
Costo aproximado	500 USD

El sensor Gro Point aunque cuenta con excelentes características en la medición de la humedad, temperatura y conductividad eléctrica del suelo, no mide pH en ninguna de sus referencias, adicionalmente en la hoja técnica no se especifica la resolución de las mediciones, por otra parte si este sensor es utilizado de forma horizontal puede entregar una medida general del estado de una profundidad específica, aunque esto significaría que se deberían comprar 6 sensores, algo que no sería económicamente viable para el proyecto, y

en caso de que se usara de forma vertical el promedio de 15 cm de profundidad podría no ser tan preciso debido a que se está trabajando en sustratos, en general aunque este sensor utiliza una tecnología de alta precisión sus principales desventajas serian la disponibilidad y costos para el proyecto.

TBSMP03 SDI-12 soil moisture/temperature probe

Este sensor es fabricado por la empresa Tekbox Digital Solutions, con sede en Ho Chi Minh – Vietnam, la tecnología utilizada es la reflectometría en el dominio del tiempo o TDR por sus siglas en inglés, en el cual se mide la constante dieléctrica con el fin de estimar la humedad del suelo, una de las principales cualidades de este sensor son sus modos de calibración, por configuración de fábrica este viene con una calibración aire/agua, la cual hace referencia que el sensor marcara 0% en el aire y 100% en agua, también puede ser calibrado por dos métodos adicionales: el primero es mínimo y máximo especifico del suelo, en este método el sensor debe ser colocado en un volumen seco de suelo para que registre el valor mínimo para después saturar este mismo volumen y registrar su valor máximo; el segundo método es la calibración multipunto donde se toman varias mediciones a distintos niveles de humedad para calibrar la sonda, el fabricante recomienda esta calibración para suelos con alto contenido orgánico.

Figura 3 Sensor TBSMP03 SDI-12 adaptado de *(Tekbox Digital Solutions, 2021)*

Este sensor solo mide humedad volumétrica y temperatura del suelo, aunque es de resaltar que sus modos de calibración avanzada hacen que este alcance una alta exactitud y confiabilidad para el sistema, cuenta con compensación por temperatura y no se ve afectado por el nivel de salinidad del suelo, su diseño lo hace robusto con un diámetro de 32 mm y

largo de 17.5 cm, convirtiéndolo en una de los sensores menos invasivos para las materas, por otra parte cuenta con una interface SDI-12, la cual además de permitir tener una lectura de las variables, es usada en la calibración por medio de distintos comandos establecidos en el manual del fabricante, en la Figura 3 se muestra el dispositivo, en este caso el precio del sensor se encuentra especificado en el sitio web del fabricante, aunque no se especifica el costo del envió ni el tiempo de entrega, con costo de 99 dólares, en la Tabla 3 se muestran las especificaciones técnicas del dispositivo.

Tabla 3 Especificaciones técnicas TBSMP03 SDI-12 adaptado de *(Tekbox Digital Solutions, 2021)*

Especificación	Valor
Voltaje de alimentación	6 a 17 VDC
Rango humedad	0-100%
Resolución humedad	0.1%
Rango temperatura	-20 a 65 °C
Precisión temperatura	±0.5 °C
Resolución temperatura	0.1 °C
Comunicación	SDI-12 v.1.3
Grado de protección	IP67
Temperatura de operación	-20 a 65 °C
Costo aproximado	100 USD

El sensor TBSMP03 SDI-12 mide con gran exactitud la humedad volumétrica de distintos suelos y su temperatura, aun así, este no tiene implementado ninguna medición de la conductividad eléctrica o salinidad del suelo, y aunque su costo en comparación de los otros sensores es más bajo, la implementación de un sensor que solo mida conductividad aumentaría considerablemente los costos de la inversión del proyecto y adicionalmente aumentaría la invasión de sensores en las materas, por otra parte el uso de este sensor en la zona de lixiviación donde solo se desea conocer la humedad, tampoco sería muy necesario ya que en esta profundidad no se requiere una alta precisión y la relación costo-necesidad no sería adecuada.

Drill and Drop probe accurately measures soil

Este sensor es fabricado por la empresa Sentek, con sede en Stepney – Australia del Sur, la tecnología utilizada es la reflectometría en el dominio de la frecuencia o FDR por sus siglas en inglés, es un sensor totalmente encapsulado en forma cónica, diseñado para ser insertado

en el suelo de forma vertical, existen 5 tamaños, en los que varía la cantidad de puntos medidos, para este proyecto el más adecuado sería el de 30 cm de largo el cual incluye 3 sensores, cada uno a 10 cm de profundidad, los resultados de cada medición pueden ser monitoreados de forma individual, es importante resaltar que su instalación se debe realizar con un equipo especializado comercializado por el mismo fabricante con el fin de obtener una medición de alta calidad, adicionalmente la vida útil del equipo es asegurada debido a que la electrónica y los sensores están protegidos por la cubierta plástica sin contacto directo al suelo.

Figura 4 Sensor Drill and Drop adaptado de *(Sentek, 2021)*

Este sensor es capaz de medir temperatura, humedad y conductividad eléctrica del suelo, este último es importante solicitarlo al momento de realizar el pedido, debido a que es una variable opcional, adicionalmente el fabricante ofrece las siguientes interfaces de comunicación: SDI-12, RS232, RS485 o Sentek Bluetooth, cada uno con su respectivo instructivo, en la Figura 4 se muestra el dispositivo, debido a que el sensor es fabricado en otro país fue necesario consultar su precio por una comercializadora internacional como TEquipment, donde es de aproximadamente 600 dólares pero no se especifica su tiempo de entrega o disponibilidad, en la Tabla 4 se muestran las especificaciones técnicas del dispositivo.

Tabla 4 Especificaciones técnicas Drill and Drop adaptado de *(Sentek, 2021)*

Especificación	Valor
Voltaje de alimentación	3.5 a 6 V
Rango humedad	0-100%
Precisión humedad	±0.03%
Resolución humedad	0.001%
Rango temperatura	-20 a 60 °C

Precisión temperatura	±2 °C
Resolución temperatura	0.3 °C
Rango CE	0 a 3000 µS/cm
Precisión CE	± 10 µS/cm
Resolución CE	3.3 µS/cm
Comunicación	SDI-12 /RS232 /RS485
Temperatura de operación	-20 a 60 °C
Humedad relativa	0 A 95 %
Costo aproximado	600 USD

El sensor Drill and Drop ofrece excelentes características en la medición de la humedad, temperatura y conductividad eléctrica, además su diseño hace que su vida útil e instalación sean favorables para sensores permanentes, aun así, su principal desventaja es su disponibilidad y alto costo, sería necesaria la compra de mínimo 3 sensores alcanzando un coste de 1.800 USD, una inversión que no sería viable para el presente proyecto.

MEC-10 soil moisture, temperature, EC y pH sensor

Este sensor es fabricado por la empresa Dalian Endeavour Technology Co., Ltd, con sede en Dalian – China, la tecnología utilizada es la reflectometría en el dominio de la frecuencia o FDR por sus siglas en inglés, en el catálogo del fabricante se ofrecen distintas combinaciones de medidas entra las que se encuentran el MEC-10 con humedad volumétrica, temperatura y conductividad eléctrica; y el soil pH Sensor que mide únicamente pH, existen otros capaces de medir hasta 7 variables: humedad, temperatura, CE, pH, nitrógeno, fosforo y calcio, aunque el costo de estos aumenta considerablemente, el diseño de este sensor consta de dos partes, tres electrodos y un encapsulado donde se encuentran el chip y los circuitos, los electrodos son fabricados con una aleación que puede soportar impactos físicos, y es resistivo a la corrosión, permitiendo su instalación permanente.

Figura 5 MEC-10 soil moisture, temperature, EC sensor adaptado de *(Dalian Endeavour Technology Co., Ltd, 2021)*

Una de las características que ofrece el fabricante es la de seleccionar tanto el voltaje de alimentación, como la interface de salida, los rangos de voltaje que se pueden elegir son: 2.7 a 16V, 5 a 30V y de 12 a 30V, para el sensor MEC-10 la única interface de salida disponible es RS485 con protocolo Modbus, mientras que para el Soil Ph Sensor hay la disponibilidad de salidas de RS485, 4 a 20mA, 0 a 5V y 0 a 10V, lo que permite que sea utilizado en diferentes aplicaciones, el protocoló Modbus implementado en el sensor MEC-10 permite además de la lectura de las tres variables, configurar dos coeficientes de salinidad y TDS, que después son utilizados para calcular estas dos variables en base a la conductividad eléctrica del suelo, en la Figura 5 se muestra el dispositivo, debido a que el sensor es fabricado en otra país fue necesario consultar su precio por una comercializadora internacional como Aliexpress, donde es de aproximadamente 45 dólares y con un tiempo de entrega de 20 días, en las Tabla 5 y Tabla 6 se muestran las especificaciones técnicas de ambos dispositivos.

Tabla 5 Especificaciones técnicas MEC-10 Soil adaptado de *(Dalian Endeavour Technology Co., Ltd, 2021)*

Especificación	Valor
Voltaje de alimentación	5 a 30 V
Rango humedad	0-100%
Precisión humedad	0-50% (±2%) / 50-100% (±3%)
Resolución humedad	1%
Rango temperatura	-40 a 80 °C
Precisión temperatura	±0.5 °C
Resolución temperatura	0.1 °C
Rango CE	0 a 20000 µS/cm
Precisión CE	± 5 µS/cm
Resolución CE	10 µS/cm
Comunicación	RS485

Grado de protección	IP68
Temperatura de operación	-40 a 80 °C
Costo aproximado	45 USD

Tabla 6 Especificaciones técnicas Soil pH Sensor adaptado de *(Dalian Endeavour Technology Co., Ltd, 2021)*

Especificación	**Valor**
Voltaje de alimentación	5 a 30 V
Rango pH	3 a 9
Precisión pH	± 3
Resolución pH	1
Comunicación	RS485
Grado de protección	IP68
Temperatura de operación	-40 a 80 °C
Costo aproximado	45 USD

Los sensores MEC-10 y soil pH ofrecen excelentes características de medición y adicionalmente sus rangos de voltaje los convierten en dispositivos que se pueden usar en distintas aplicaciones, una de las principales recomendaciones del fabricante es, que a profundidades mayores de 20 cm, los sensores sean instalados de forma horizontal, el tamaño de este sensor es relativamente pequeño en comparación a los anteriores y podría ser instalado con facilidad en las materas, por último su principal ventaja es el costo, siendo necesario la compra de tres dispositivos por cada sensor con una inversión aproximada de 270 USD.

B. Validación y calibración de sensores

Con el objetivo de asegurar la integridad del sistema automatizado se realizó la validación de las variables establecidas para el proyecto en los sensores elegidos anteriormente, inicialmente se asignó una dirección modbus a cada uno de los sensores y un identificador:

- Sensor MEL-10 A: Sensor matera 1
- Sensor MEL-10 B: Sensor matera 2
- Sensor MEL-10 C: Sensor matera 3
- Sensor pH Soil A: Sensor matera 1
- Sensor pH Soil B: Sensor matera 2
- Sensor pH Soil C: Sensor matera 3
- Sensor Capacitivo A: Sensor matera 1
- Sensor Capacitivo B: Sensor matera 2
- Sensor Capacitivo C: Sensor matera 3
- Sensor TDS agua: Fuente de agua
- Sonda Temperatura sumergible: Fuente de agua

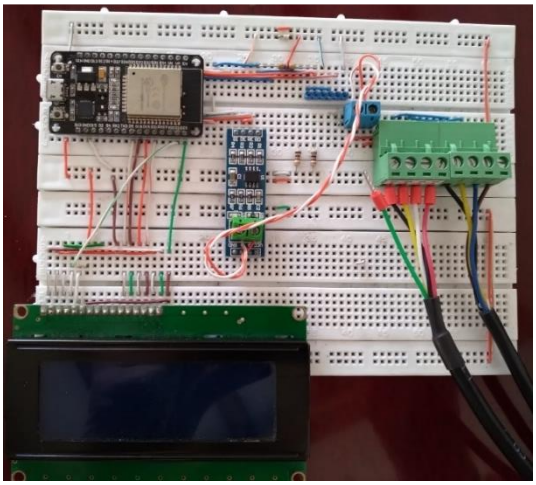

Figura 6 Circuito para lectura de sensores en el Laboratorio

Para facilitar los procedimientos en el laboratorio se implementó el circuito mostrado en la Figura 6 en protoboard con un módulo ESP-32 doit kit dev V-1, un módulo MAX-485 y una pantalla LED 4x20 que permitiese tener una lectura rápida y confiable de cada una de

las variables, el cual fue configurado para visualizar las lecturas dependiendo de la validación que se deseara realizar.

Validación sensor de humedad volumétrica MEL-10

A continuación, se describen los materiales y el procedimiento que se siguió en el laboratorio de ingeniería ambiental para validar la medición de humedad del sensor MEL-10. Las especificaciones técnicas del sensor para la variable de la humedad volumétrica especificadas por el fabricante son:

- Rango: 0-100 %
- Resolución: 0.03 %
- Precisión: ± 1%

Teniendo en cuenta que la humedad volumétrica es la relación entre el volumen de agua y el volumen del sustrato seco, dada por la siguiente ecuación (Flores & Alcalá, 2010):

$$\theta = \frac{V_w}{V_S} * 100\% \quad (35)$$

Se hizo uso de los siguientes materiales:

- Horno secador a 110 °C
- Recipiente a 200 ml
- Peso
- Probeta 20 ml
- 1 Botella de Agua destilada
- Circuito lectura sensores con pantalla LCD
- Sensor MEL10
- Muestras de Sustrato 100 gramos

Se siguieron los siguientes pasos:

1. Tomar una muestra de sustrato de 100 gramos en una bolsa Ziploc de cierre hermético con el fin de evitar que la humedad afecte el contenido, Figura 7.

Figura 7 Muestra de 100 gramos de sustrato en bolsa Ziploc hermética

2. Secar la muestra en una estufa a 110 °C durante 1 hora eliminando cualquier rastro de humedad.
3. Pasar la muestra a un recipiente hasta alcanzar un volumen de 200 ml. (Asegurarse de no comprimir la muestra con el fin de no reducir el espacio poroso) Figura 8.

Figura 8 Muestra de sustrato seco en recipiente de 200 ml

4. Añadir 5 ml de agua destilada con la probeta, con el fin de que el agua no se acumule en un solo punto, esta se aplicó con movimientos circulares buscando uniformidad, Figura 9.

Figura 9 Probeta con 5 ml de agua destilada

5. Se midió la humedad volumétrica con el sensor MEL-10 y se pesó la muestra de sustrato, Figura 10.

Figura 10 Medición del sensor MEL-10 A para la primera aplicación de 5 ml

6. Repetir los pasos 4 y 5 hasta alcanzar la saturación del sustrato (Se define como punto de saturación cuando al humedecer el sustrato este no incrementaría más de tamaño y todo su espacio poroso estarían lleno de agua, se evidencia por la lámina de agua que se observa en su parte superior o inferior) Figura 11.

Figura 11 Muestra de suelo saturada a un volumen de aproximadamente 150 ml

7. Se alcanzó la saturación del sustrato al aplicar 50 ml de agua destilada, se detuvo la medición dado que el sustrato no tiene la capacidad de retener más agua.

El procedimiento se llevó a cabo en las instalaciones de la Escuela Colombiana de Ingeniería Julio Garavito con acompañamiento del encargado del laboratorio de ingeniería ambiental, los resultados del sensor MEL-10 A se muestran en la Tabla 7, los datos teóricos fueron calculados con el volumen de sustrato al alcanzar su saturación el cual es aproximadamente 100 ml, aunque se midió con la probeta la cantidad de agua aplicada, esta se calculó por medio de la diferencial de peso antes y después de la aplicación para alcanzar una mayor precisión.

Tabla 7 Datos validación humedad volumétrica del sensor MEL-10 A

Agua Aplicada (ml)	Mediad sensor MEL-10 (%)	Peso de la muestra (g)	Humedad Volumétrica Teórica (%)	Error de medición
0	0,00	36,2990	0,00	0,00
5	4,13	41,2715	4,97	-0,84
10	10,20	45,9703	9,67	0,53
15	14,32	51,3582	15,06	-0,74
20	20,58	56,2283	19,93	0,65
25	25,25	60,8766	24,58	0,67
30	30,36	66,115	29,82	0,54
35	34,80	70,9021	34,60	0,20
40	40,76	76,5153	40,22	0,54
45	44,70	81,8654	45,57	-0,87
50	49,57	86,1529	49,85	-0,28

Partiendo de lo observado en la tabla de resultados se puede afirmar que el sensor adquirido para este proyecto mantiene sus características para el sustrato utilizado en el cultivo y sus mediciones podrán ser utilizadas para la caracterización y su posterior supervisión en campo.

Validación sensor de Conductividad eléctrica MEL-10

A continuación, se describen los materiales y el procedimiento que se siguió en el laboratorio de ingeniería ambiental para validar la medición de conductividad eléctrica de los sensores MEL-10. Las especificaciones técnicas del sensor para la variable de CE entregadas por el fabricante son:

- Rango: 0-20000 µS/cm.
- Resolución: 10 µS/cm de 0-10000 µS/cm y 50 µS/cm de 10000-20000 µS/cm.
- Precisión: ± 3% de 0-10000 µS/cm y ± 5% de 10000-20000 µS/cm.

La validación de estos sensores se realizó por medio de dos patrones de conductividad eléctrica disponible en el laboratorio y con los cuales se calibran equipos como el multi parámetro HQ40d, se hizo uso de los siguientes materiales:

- Recipiente a 200 ml.
- 1 Botella de agua destilada.

- Patrón de conductividad a 1000 µS/cm a 25 °C BiopharChem.
- Patrón de conductividad a 1413 µS/cm a 25 °C Oakton.
- Circuito lectura sensores con pantalla LCD.
- Sensor MEL10.

Se siguieron los siguientes pasos:

1. Se aplicó una pequeña cantidad de agua destilada en el recipiente de 200 ml con el fin de eliminar posibles causas de desviación en la solución calibradora.
2. Se limpiaron los electrodos del sensor con agua destilada
3. Se colocó el sensor en el recipiente
4. Se agregó la solución calibradora hasta que cubriera los electrodos del sensor, la cual fue aproximadamente 200 ml, Figura 12.

Figura 12 Sensor MEL-10 A en solución calibradora a 1413 µS/cm

5. Se alimentó el circuito y sensor, y se esperó a que se estabilizara la lectura, Figura 13.

Figura 13 Medición del sensor MEL-10 A para solución calibradora a 1413 µS/cm

Inicialmente se usó un recipiente en el que justo ingresaban los electrodos del sensor y donde los extremos estaban próximos al borde del recipiente, al realizar la medición los

valores registrados no eran correctos, incluyendo la humedad volumétrica, la cual señalaba 68.15% a pesar de estar totalmente sumergido en un líquido, esto permitió identificar que no solo se estaba censando la parte interior del recipiente, si no que el volumen de medición del sensor estaba midiendo el aire que lo rodeaba, razón por la cual para obtener lecturas correctas se procedió a usar el recipiente de 200 ml que cubriera la totalidad del volumen, según el fabricante dicho volumen tiene forma cilíndrica con un diámetro de aproximadamente 5 cm tal como se muestra en la Figura 14.

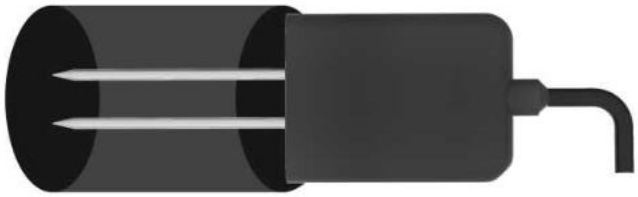

Figura 14 Volumen de medición sensores de suelo *(Dalian Endeavour Technology Co., Ltd, 2021)*

Los resultados del sensor MEL-10 A se muestran en la Tabla 8, es importante resaltar que la conductividad eléctrica del patrón varía según la temperatura y esta fue calculada haciendo uso de su ficha técnica, para finalizar se concluye que los sensores funcionan correctamente con la precisión indicada por el fabricante.

Tabla 8 Datos validación conductividad eléctrica del sensor MEL-10 A

Patrón CE a 25°C (µS/cm)	Medida sensor MEL-10 (µS/cm)	Temperatura sensor MEL-10 (°C)	Valor patrón a temperatura (µS/cm)	Error relativo de medición
1000	912	16,22	906	0,7%
1413	1188	16,34	1184	0.3%

Validación sensor pH Soil

A continuación, se describen los materiales y el procedimiento que se siguió en el laboratorio de ingeniería ambiental para validar la medición de acidez y alcalinidad de los sensores pH Soil. Las especificaciones técnicas del sensor para la variable de pH entregadas por el fabricante son:

- Rango: 0-9 pH.

- Resolución: 0.1.
- Precisión: ± 0.3 pH.

La validación de estos sensores se realizó por medio de dos soluciones tampón de acidez y alcalinidad disponible en el laboratorio y con los cuales se calibran equipos como el multi parámetro HQ40d, se hizo uso de los siguientes materiales:

- Recipiente a 200 ml.
- 1 Botella de agua destilada.
- Solución tampón pH 4.00 Chemi.
- Solución tampón pH 7.00 Chemi.
- Circuito lectura sensores con pantalla LCD.
- Sensor pH Soil.

Se realizaron los siguientes pasos:

1. Se aplicó una pequeña cantidad de agua destilada en el recipiente de 200 ml con el fin de eliminar posibles causas de desviación en la solución calibradora.
2. Se limpiaron los electrodos del sensor con agua destilada.
3. Se colocó el sensor en el recipiente.
4. Se agregó la solución calibradora hasta que cubriera los electrodos del sensor, la cual fue aproximadamente 200 ml.

Figura 15 Sensor pH Soil A en solución tampón pH 7.00

5. Se alimentó el circuito y el sensor y se esperó a que se estabilizara la lectura.

Figura 16 Medición del sensor pH Soil A para solución tampón pH 4.00

Al igual que el sensor de conductividad eléctrica este sensor tiene un volumen de medición que se debe tener presente al momento de realizar la validación, también es importante resaltar que el tiempo de respuesta en estos sensores fue mucho más alto y su estabilización se alcanzó alrededor de los 15 segundos, aunque según la hoja técnica su tiempo de respuesta debería ser de 10, en la Tabla 9 se muestran los valores de las dos soluciones tapón usadas para validar la medición, las lecturas del sensor y el error absoluto de medición, donde se puede observar que ninguno de los errores sobrepasa el establecido por el fabricante.

Tabla 9 Datos validación pH del sensor pH Soil A

Patrón pH (pH)	Medida sensor pH (pH)	Error de medición
4,00	4,00	0,0
7,00	6,90	-0,1

Calibración sensor conductividad eléctrica TDS Meter v1.0

A continuación, se describen los materiales y el procedimiento que se siguió en el laboratorio de ingeniería ambiental para validar la medición de conductividad eléctrica del TDS Meter v1.0. Las especificaciones técnicas del sensor entregadas por el fabricante son:

- Rango TDS: 0-1000 ppm.
- Precisión: ± 10% a 25 °C.
- Rango CE: 0-2000 µS/cm.
- Voltaje de salida: 0 - 2.3 V.

La validación de este sensor se realizó por medio del patrón de conductividad eléctrica disponible en el laboratorio y el cual es recomendado por el fabricante, se hizo uso de los siguientes materiales:

- Recipiente a 200 ml.
- 1 Botella de agua destilada.
- Patrón de conductividad a 1413 µS/cm a 25 °C Oakton.
- Circuito lectura sensores con pantalla LCD.
- Sensor TDS Meter v1.0.
- Sonda Temperatura sumergible DS18B20.

Se realizaron los siguientes pasos:

1. Se aplicó una pequeña cantidad de agua destilada en el recipiente de 200 ml con el fin de eliminar posibles causas de desviación en la solución calibradora.
2. Se limpiaron los electrodos del sensor y la sonda de temperatura con agua destilada
3. Se colocó el sensor en el recipiente.
4. Se agregó la solución calibradora hasta que cubriera los electrodos del sensor, la cual fue aproximadamente 200 ml, Figura 17.

Figura 17 Sensor TDS Meter y sonda en solución calibradora a 1413 µS/cm

5. Se alimentó el circuito y el sensor y se esperó a que se estabilizara la lectura, Figura 18.

Figura 18 Medición del sensor TDS Meter para solución calibradora a 1413 µS/cm

En la Tabla 10 se muestran los resultados, evidenciando que el algoritmo y las especificaciones dadas por el fabricante se cumplen, con lo cual este sensor podrá ser usado para la medición de la calidad del agua en la fuente.

Tabla 10 Datos validación conductividad eléctrica del sensor TDS Meter

Patrón CE a 25°C (µS/cm)	Medida sensor TDS (µS/cm)	Temperatura Sonda (°C)	Valor patrón a temperatura (µS/cm)	Error relativo de medición
1413	1173,5	16,94	1199	-2,1%

Calibración sensor capacitivo de humedad V1.2

A continuación, se describen los materiales y el procedimiento que se siguió en el laboratorio de ingeniería ambiental para calibrar la medición de humedad del sensor capacitivo de Humedad V1.2. Las especificaciones técnicas del sensor entregadas por el fabricante son:

- Voltaje de Alimentación: 3.3 Voltios.
- Voltaje de Salida: 0 – 3 voltios.

Haciendo uso del sensor MEL-10 previamente verificado, se realizó la curva de calibración para el sensor capacitivo:

Se hizo uso de los siguientes materiales:

- Horno secador a 110 °C.
- Recipiente a 200 ml.
- Probeta 20 ml.
- 1 Botella de Agua destilada.
- Multímetro.
- Sensor capacitivo de humedad V1.2.
- MEL-10.
- Muestra de Sustrato 100 gramos.

Se realizaron los siguientes pasos:

1. Tomar una muestra de sustrato de 100 gramos en una bolsa Ziploc de cierre hermético con el fin de evitar que la humedad afecte el contenido.
2. Secar la muestra en una estufa a 110 °C durante 1 hora eliminando cualquier rastro de humedad.
3. Pasar la muestra a un recipiente hasta alcanzar un volumen de que cubra el sensor. (Asegurarse de no comprimir la muestra con el fin de no reducir el espacio poroso). Figura 19.

Figura 19 Sensor capacitivo de humedad V1.2 en sustrato seco

4. Añadir 20 ml de agua destilada con la probeta, con el fin de que el agua no se acumule en un solo punto esta se aplicó con movimientos circulares buscando uniformidad.
5. Se midió la Humedad volumétrica con el sensor MEL-10 esperando a que se estabilice.
6. Se midió el voltaje de salida del sensor capacitivo de humedad V1.2 con el multímetro, Figura 20.

Figura 20 Voltaje de salida sensor capacitivo de humedad V1.2

7. Se repitieron los pasos 5 y 6 hasta alcanzar el punto donde la salida no vario.

8. Los sensores tuvieron un rango de medición entre 0% y 60% de humedad volumétrica.

Tabla 11 Datos calibración sensor capacitivo de humedad V1.2 A

Medida	Medida sensor MEL-10 (%)	Voltaje (V)
1	0,00	2,24
2	6,20	2,08
3	15,42	1,80
4	20,40	1,72
5	23,78	1,64
6	29,19	1,59
7	35,11	1,50
8	44,14	1,29
9	51,20	1,08
10	59,48	0,97

En la Tabla 11 se muestran los resultados para el sensor capacitivo de humedad V1.2 A, para su calibración se graficaron los puntos y se realizó una regresión lineal, la cual alcanzó un coeficiente de correlación del 98.86%, en la Figura 21 se muestran los resultados.

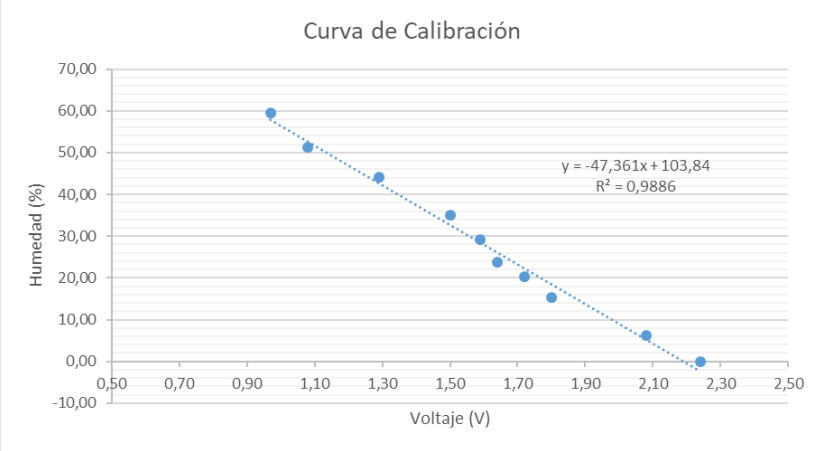

Figura 21 Curva sensor capacitivo de humedad V1.2

C. Pruebas organolépticas al sustrato

Para esta etapa se utilizó una bolsa de sustrato sin cultivo, la cual fue llevada a un estado de humedad cercano a la saturación, y posteriormente cubierta con una lona para evitar la evapotranspiración y realizar así una prueba de drenaje interno. Se decidió monitorear tres profundidades con el fin de conocer el comportamiento del agua a lo largo del perfil vertical de la bolsa: la primera fue a 15 cm, punto en el que se encuentra la mayor densidad de raíces para plantas jóvenes menores a dos años; la segunda fue a 25 cm, en la cual se encontrara la mayor densidad de raíces para plantas adultas; y por ultimo a 35 cm donde se ubican los orificios de lixiviación.

Figura 22 Bolsa de sustrato sin cultivo a tres profundidades

En la Figura 22 se muestra la bolsa de sustrato en la que se instalaron los tres sensores MEC-10, y adicionalmente dos tensiómetros a distintas profundidades con el fin de conocer el esfuerzo de la planta para acceder al agua, la humedad fue monitoreados durante 4 días con una frecuencia de un minuto, mientras que la presión fue tomada cada dos horas. Los datos fueron recolectados con un circuito compuesto por un módulo ESP-12F y XY-17, estos enviaban la información por mensajes TCP/IP al ordenador de placa reducida, la cual los almacenaba en una base de datos.

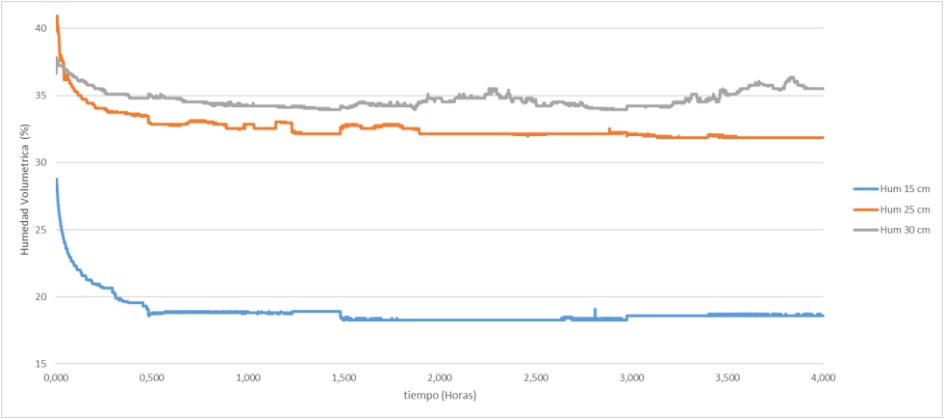

Figura 23 Resultados humedad volumétrica drenaje interno

En la Figura 23 se muestran los resultados del experimento de drenaje interno en los 4 días, se consideró saturación el momento en que al aplicar agua se observaba una cantidad lixiviada parecida a la suministrada, donde se observó cómo durante las primeras 12 horas el sustrato presenta el mayor cambio de humedad hasta alcanzar un punto de estabilidad en cada una de las profundidades, es importante resaltar que a mayor profundidad se acumulará una mayor cantidad de agua y será más fácil saturar el sustrato.

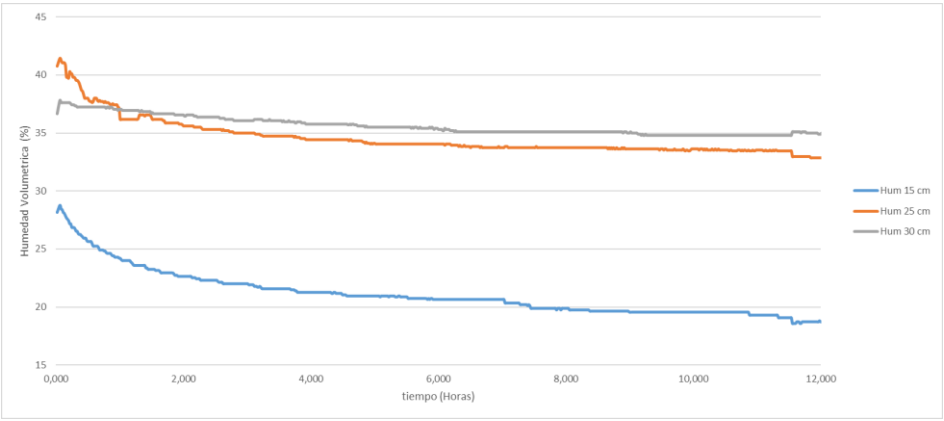

Figura 24 Variación de humedad primeras 12 horas

Para hallar la curva de conductividad hidráulica del sustrato se usaron los datos de las primeras 12 horas a una profundidad de 15 cm dado que estas presentan la mayor variación y tendrán un delta de humedad más amplio, la primera suposición que se realiza es que la conductividad será una función exponencial de la humedad volumétrica (López, Duarte, González, & Cid, 2008) y se hallaran los parámetros con el procedimiento propuesto por Libardi et al. (1980).

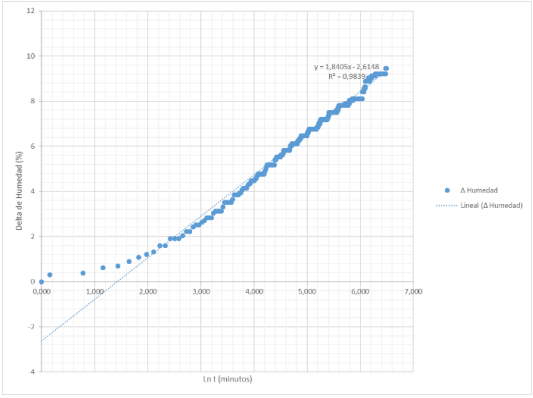

Figura 25 Delta de humedad vs logaritmo del tiempo en minutos

La linealización de la curva mostrada en la Figura 25 permitió hallar los parámetros que ajustan la función de conductividad hidráulica (36) para la profundidad $L = 15\,cm$ especificada, haciendo uso de las ecuaciones (37) y (38), esta es una aproximación para valores inferiores a un $\theta_0 = 28.77\%$.

$$K = K_0 * e^{\beta(\theta - \theta_0)} \tag{36}$$

$$m = \frac{1}{\beta} \tag{37}$$

$$b = \frac{1}{\beta} * \ln\left(\frac{\beta * K_0}{L}\right) \tag{38}$$

Hallando los coeficientes de la ecuación se traza la curva de conductividad hidráulica aproximada para valores inferiores a 28.77% en la Figura 26, pero dado que esta función no es adecuada para valores superiores, se realizó un segundo ajuste utilizando los datos de retención y humedad siguiendo el procedimiento propuesto por Van Genuchten et al. (1991).

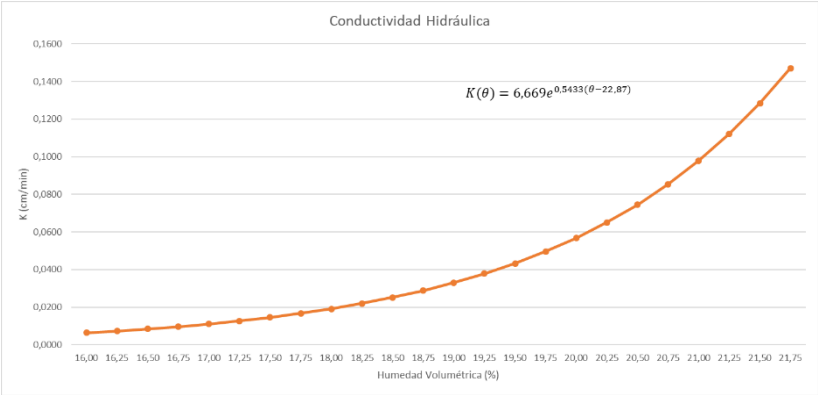

Figura 26 Conductividad Hidráulica procedimiento de Libardi et al. *(1980)*

Para el cálculo de la curva característica se hizo uso de los datos de tensión, humedad volumétrica y la curva de conductividad anteriormente ajustada, el procedimiento requirió de la aproximación de 5 parámetros para la ecuación (4) usando el software RETC (RETention Curve) v.6.02, los resultados se muestran en el Anexo D y se resumen en la Tabla 12, obteniendo un coeficiente de determinación del 0.953 respeto a los valores observados.

Tabla 12 Parámetros de las curvas ajustados

θ_r	θ_s	α	n	l	K_s
0,001	0,3776	0,0174	2.5212	0,5	1.8357

D. Resultados del ajuste parámetros

```
*****************************************************************
*                                                               *
*     Analysis of soil hydraulic properties                     *
*                                                               *
*     Welcome to RETC                                           *
*                                                               *
*     Mualem-based restriction, M=1-1/N                         *
*     Simultaneous fit of retention and Conductivity data       *
*     MType= 3     Method= 1                                    *
*                                                               *
*****************************************************************

 INITial values of the coefficients
 ==================================
 No      Name         INITial value   Index
  1      ThetaR           .1000         1
  2      ThetaS           .3800         1
  3      Alpha            .0270         1
  4      n               1.2300         1
  5      m                .1870         0
  6      l                .5000         0
  7      Ks               .0020         1

 Observed data
 =============
Obs. No.    Pressure head     Water content     Weighting coefficient
     1          22.434             .3656             1.0000
     2          23.454             .3587             1.0000
     3          23.454             .3531             1.0000
     4          24.473             .3500             1.0000
     5          24.473             .3473             1.0000
     6          25.493             .3443             1.0000
     7          26.513             .3384             1.0000
     8          28.552             .3374             1.0000
     9          30.592             .3353             1.0000
    10          32.631             .3304             1.0000
    11          33.651             .3304             1.0000
    12          33.651             .3304             1.0000
    13          34.670             .3286             1.0000
    14          34.670             .3286             1.0000
    15          34.670             .3254             1.0000
    16          35.690             .3254             1.0000
    17          35.690             .3254             1.0000
    18          35.690             .3254             1.0000
    19          34.670             .3297             1.0000
    20          36.710             .3215             1.0000
    21          36.710             .3215             1.0000
    22          61.183             .2718             1.0000
```

23	62.203	.2525	1.0000
24	62.203	.2430	1.0000
25	63.223	.2338	1.0000
26	64.242	.2265	1.0000
27	64.242	.2231	1.0000
28	65.262	.2201	1.0000
29	65.262	.2158	1.0000
30	66.282	.2118	1.0000
31	67.302	.2066	1.0000
32	69.341	.1965	1.0000
33	71.380	.1956	1.0000
34	73.420	.1882	1.0000
35	73.420	.1882	1.0000
36	73.420	.1882	1.0000
37	76.479	.1882	1.0000
38	77.499	.1882	1.0000
39	77.499	.1882	1.0000
40	78.518	.1882	1.0000
41	78.518	.1873	1.0000
42	78.518	.1873	1.0000
43	79.538	.1873	1.0000
44	79.538	.1891	1.0000
45	79.538	.1891	1.0000
46	80.558	.1891	1.0000
47	80.558	.1827	1.0000
48	80.558	.1827	1.0000
49	81.578	.1827	1.0000
50	81.578	.1827	1.0000

	Water content	Conductivity	Weighting coefficient
51	.1600	.6500E-02	1.0000
52	.1625	.7400E-02	1.0000
53	.1650	.8500E-02	1.0000
54	.1675	.9700E-02	1.0000
55	.1700	.1110E-01	1.0000
56	.1725	.1280E-01	1.0000
57	.1750	.1460E-01	1.0000
58	.1775	.1670E-01	1.0000
59	.1800	.1920E-01	1.0000
60	.1825	.2200E-01	1.0000
61	.1850	.2520E-01	1.0000
62	.1875	.2880E-01	1.0000
63	.1900	.3300E-01	1.0000
64	.1925	.3780E-01	1.0000
65	.1950	.4330E-01	1.0000
66	.1975	.4960E-01	1.0000
67	.2000	.5680E-01	1.0000
68	.2025	.6510E-01	1.0000
69	.2050	.7460E-01	1.0000
70	.2075	.8540E-01	1.0000
71	.2100	.9790E-01	1.0000

72	.2125	.1121E+00	1.0000
73	.2150	.1284E+00	1.0000
74	.2175	.1471E+00	1.0000
75	.2200	.1685E+00	1.0000

Weighting coefficients
======================
W1= .10000 W2= 5.04146 W12= .50415

NIT	SSQ	ThetaR	ThetaS	Alpha	n	Ks
0	.47701	.1000	.3800	.0270	1.2300	.0020
1	.32424	.0680	.3775	.0305	1.2647	9532.8051
2	.12711	.0599	.4050	.0891	1.2799	6923.5893
3	.10939	.0575	.4049	.1047	1.2836	5844.6152
4	.10309	.0255	.4515	.1255	1.3248	94.3212
5	.07152	.0151	.4611	.1183	1.3320	1170.4995
6	.06264	.0073	.4666	.1126	1.3388	1360.4651
7	.05916	.0029	.4714	.1095	1.3427	1347.1510
8	.05716	.0001	.4749	.1086	1.3455	1277.6475

wcr is less then 0.001: Changed to fit with wcr=0.0

NIT	SSQ	ThetaS	Alpha	n	Ks
0	.34186	.3800	.0270	1.2300	.0020
1	.16870	.3745	.0326	1.2840	521.0134
2	.06221	.4218	.0676	1.3629	288.7654
3	.03233	.4789	.0655	1.4796	166.3888
4	.01753	.5189	.0609	1.5731	146.4704
5	.01403	.5374	.0591	1.6244	124.4997
6	.01320	.5441	.0575	1.6513	113.0856
7	.01299	.5449	.0563	1.6653	104.4578
8	.01289	.5430	.0551	1.6735	97.3693
9	.01280	.5400	.0540	1.6793	91.0690
10	.01251	.5239	.0494	1.6965	66.5516
11	.01234	.5202	.0484	1.7048	63.8391
12	.01203	.5046	.0441	1.7257	46.0202
13	.01182	.5008	.0431	1.7359	44.0283
14	.01150	.4860	.0393	1.7612	31.5312
15	.01127	.4823	.0383	1.7739	30.0754
16	.01115	.4787	.0374	1.7831	27.7344
17	.01103	.4751	.0365	1.7925	25.5986
18	.01091	.4715	.0356	1.8022	23.6136
19	.01079	.4679	.0348	1.8124	21.7705
20	.01066	.4643	.0339	1.8231	20.0612
21	.01054	.4608	.0331	1.8343	18.4781
22	.01041	.4573	.0323	1.8460	17.0142
23	.01029	.4538	.0315	1.8583	15.6624
24	.01016	.4504	.0307	1.8711	14.4159
25	.01003	.4471	.0300	1.8845	13.2683
26	.00991	.4437	.0292	1.8985	12.2131

```
27      .00978      .4405      .0285     1.9131    11.2443
28      .00966      .4373      .0279     1.9283    10.3560
29      .00953      .4341      .0272     1.9441     9.5426
30      .00941      .4311      .0266     1.9605     8.7989
31      .00930      .4281      .0260     1.9775     8.1198
32      .00918      .4251      .0254     1.9950     7.5004
33      .00907      .4223      .0248     2.0132     6.9362
34      .00896      .4195      .0243     2.0319     6.4230
35      .00886      .4168      .0238     2.0511     5.9567
36      .00876      .4142      .0233     2.0707     5.5335
37      .00867      .4117      .0228     2.0907     5.1500
38      .00858      .4092      .0224     2.1111     4.8029
39      .00850      .4069      .0220     2.1317     4.4890
40      .00842      .4047      .0216     2.1524     4.2056
41      .00835      .4025      .0212     2.1733     3.9500
42      .00828      .4005      .0209     2.1941     3.7198
43      .00822      .3986      .0206     2.2147     3.5127
44      .00817      .3968      .0203     2.2351     3.3267
45      .00812      .3951      .0200     2.2551     3.1598
46      .00800      .3890      .0190     2.3253     2.5996
47      .00795      .3810      .0178     2.4425     1.9988
48      .00784      .3778      .0174     2.5146     1.8433
49      .00784      .3777      .0174     2.5204     1.8383
51      .00784      .3776      .0174     2.5212     1.8357

Correlation matrix
==================
     ThetaS     Alpha      n          Ks
        1          2          3          4
 1   1.0000
 2    .9690     1.0000
 3   -.9172     -.9714     1.0000
 4    .9397      .9629     -.9626     1.0000

RSquated for regression of observed vs fitted values =  .95332858
================================================================

Nonlinear least-squares analysis: final results
===============================================
                                                  95% Confidence limits
Variable      Value      S.E.Coeff.     T-Value      Lower        Upper
ThetaS       .37764       .00931         40.56       .3591        .3962
Alpha        .01743       .00120         14.55       .0150        .0198
n           2.52122       .17163         14.69     2.1790       2.8634
Ks          1.83572       .52074          3.53      .7973       2.8741

End of problem
==============
```

E. Materiales PCB control de velocidad y módulos de medición

Ítem	Designador	Descripción	Comentario	Cantidad x placa
1	C1	Capacitor_SMD:CP_Elec_4x4.5	10uF	1
2	> C2-C5, C8	Capacitor_SMD:C_0805_2012Metric	100nF	5
3	> C6, C7	Capacitor_SMD:C_0805_2012Metric	27pF	2
4	> D1-D4	Diode_THT:D_DO-41_SOD81_P10.16mm_Horizontal	1N4004	4
5	> D5-D7	Diode_THT:D_DO-35_SOD27_P7.62mm_Horizontal	1N4148	3
6	IC1	Modules:DIP1543W58P254L5175H635Q40N	DSPIC30F4011-30I_P	1
7	J1	Connector_PinSocket_2.54mm:PinSocket_1x05_P2.54mm_Vertical	Pickit3	1
8	J2	TerminalBlock_Phoenix:TerminalBlock_Phoenix_MKDS-1,5-2-5.08_1x02_P5.08mm_Horizontal	Motor	1
9	J3	TerminalBlock_Phoenix:TerminalBlock_Phoenix_MKDS-1,5-2-5.08_1x02_P5.08mm_Horizontal	Válvula 1	1
10	J4	TerminalBlock_Phoenix:TerminalBlock_Phoenix_MKDS-1,5-2-5.08_1x02_P5.08mm_Horizontal	Válvula 2	1
11	J5	TerminalBlock_Phoenix:TerminalBlock_Phoenix_MKDS-1,5-2-5.08_1x02_P5.08mm_Horizontal	Válvula 3	1
12	J6	TerminalBlock_Phoenix:TerminalBlock_Phoenix_MKDS-1,5-3-5.08_1x03_P5.08mm_Horizontal	Presión	1
13	J7	TerminalBlock_Phoenix:TerminalBlock_Phoenix_MKDS-1,5-3-5.08_1x03_P5.08mm_Horizontal	Flujo	1
14	J8	Connector_PinSocket_2.54mm:PinSocket_1x07_P2.54mm_Vertical	Raspberry	1
15	J9	TerminalBlock_Phoenix:TerminalBlock_Phoenix_MKDS-1,5-2-5.08_1x02_P5.08mm_Horizontal	Fuente	1
16	Q2	Package_TO_SOT_THT:TO-220-3_Vertical	IRF530N	1
17	> Q1, Q3	Package_TO_SOT_SMD:SOT-23	2N2222	2
18	> Q4-Q6	Modules:TO254P483X1010X1985-3P	TIP31	3
19	> Q7-Q12	Package_TO_SOT_THT:TO-92_Inline	2N7000	6
20	> R1, R5, R9-R20	Resistor_SMD:R_1206_3216Metric	10K	14
21	R2	Resistor_SMD:R_1206_3216Metric	1,8K	1
22	R3	Resistor_SMD:R_1206_3216Metric	3,3K	1
23	> R4, R6-R8	Resistor_SMD:R_1206_3216Metric	200	4
24	SW1	Button_Switch_SMD:SW_SPST_CK_RS282G05A3	SW_Push	1
25	U1	Modules:MP23070N	MP23070N	1
26	Y1	Crystal:Crystal_HC49-4H_Vertical	10000MHz	1

Ítem	Designador	Descripción	Comentario	Cantidad x placa	Cantidad Total
1	BT1	Battery:BatteryHolder_Keystone_1042_1x18650	Battery_Cell	1	4
2	C1	Capacitor_SMD:C_0805_2012Metric	10uF	1	4
3	C2	Capacitor_SMD:C_0805_2012Metric	100nF	1	4
4	IC1	RF_Module:ESP-12E	esp-12f	1	4
5	J1	Modules:TerminalBlock_TE_282834-2_1x02_P2.54mm_Horizontal	Panel	1	4
6	J2	Modules:TerminalBlock_TE_282834-4_1x04_P2.54mm_Horizontal	PH-Soil	1	6
7	J3	Modules:TerminalBlock_TE_282834-4_1x04_P2.54mm_Horizontal	MEL-10	1	
8	J4	Modules:TerminalBlock_TE_282834-3_1x03_P2.54mm_Horizontal	Sen_ana	1	8
9	J5	Modules:TerminalBlock_TE_282834-3_1x03_P2.54mm_Horizontal	Sen_dig	1	
10	J6	Modules:TerminalBlock_TE_282834-3_1x03_P2.54mm_Horizontal	Puerto Serial	1	4
11	Q1	Package_TO_SOT_SMD:SOT-23	2N3906	1	4
12	Q2	Package_TO_SOT_SMD:SOT-23	2N2222	1	4
13	R1	Resistor_SMD:R_0603_1608Metric	470K	1	4
14	R4	Resistor_SMD:R_0603_1608Metric	200K	1	4
15	> R2, R3, R5, R6	Resistor_SMD:R_0603_1608Metric	10K	4	4
16	R7	Resistor_SMD:R_0603_1608Metric	470	1	4
17	> R8, R9	Resistor_SMD:R_0603_1608Metric	100	2	8
18	R10	Resistor_SMD:R_0603_1608Metric	4.7K	1	4
19	SW1	Button_Switch_SMD:SW_SPST_TL3342	Reset	1	4
20	SW2	Button_Switch_SMD:SW_SPST_TL3342	Flash	1	4
21	U1	Module:AC02-YF3	AC02-YF3	1	4
22	U2	Module:HW-0519	HW-0519	1	3
23	U3	Module:MP23070N	MP23070N	1	4

F. Analogía a cultivos de mayor extensión

Dado que en este proyecto se busca crear una solución integral que pueda ser aplicada a distintos cultivos, a continuación, se realizó una breve descripción de cómo se llevaría a cabo el control PID en un cultivo donde su extensión es mucho mayor y es necesario el uso de bombas con capacidades mayores a las del prototipo, como ejemplo se utilizará la información de un cultivo ubicado en el municipio de la Calera el cual cuenta con 9 unidades de riego y un total de 30245 plantas, la bomba instalada en este cultivo se muestra en la Figura 27, mientras que en la Tabla 13 y 22 se indican las especificaciones del motor y la bomba respectivamente.

Figura 27 Motor y bomba de riego cultivo la Calera

Con el fin de asegurar el correcto funcionamiento del sistema, se propone la elección de un variador teniendo en cuenta el voltaje de alimentación, corriente nominal y potencia del motor. El tipo de aplicación es uno de los factores que toma mayor relevancia en la decisión de implementar un variador de frecuencia, debido a que la dinámica entre la velocidad del motor y el torque, para bombas centrifugas es cuadrática, el decremento o incremento en la velocidad se verá reflejado en el torque al cuadrado y en la potencia al cubo.

Tabla 13 Especificaciones técnicas motor WEG TE1BFOXO

Especificación	Valor
Numero de fases	3
Tensión nominal de operación	220/380/440 V
Régimen de servicio	S1
Grado de protección	IP55
Clase de aislamiento	F
Frecuencia	60 Hz
Potencia nominal	7,5 KW
Velocidad nominal	3515 RPM
Corriente nominal	25,0/14,5/12,5 A
Factor de potencia	0,88
Temperatura ambiente	40 °C

Tabla 14 Especificaciones técnicas bomba HE 2 100-2

Modelo	Referencia	Succión	Descarga	Potencia	Etapas	Fases	H max	Q max
HE 2 100-2	1E0546	2" NPT	2" NPT	10,0 HP	2	3	124 mca	130 gpm

Después de haber identificado las especificaciones de los equipos instalados en el cultivo se procedió con la búsqueda de los variadores ofertados comercialmente, se consultaron cuatro marcas reconocidas mencionadas a continuación: Siemens, Schneider Electric, WEG y Danfoss, la razón por la cual se priorizaron estas marcas fue debido a su reconocimiento en el campo de los variadores de velocidad, su trayectoria en el diseño y así mismo el soporte técnico que prestan a sus clientes.

A continuación, se presenta un breve resumen de un equipo que cumple con las características adecuadas para la aplicación.

CWF500

El modelo pertenece a la marca WEG y hace parte de la gama diseñada especialmente para las aplicaciones de bombeo con par variable, en este caso el dispositivo viene equipado con una función softPLC la cual permite una programación mucho más robusta en aplicaciones complejas, adicionalmente el proveedor facilita una guía de aplicación para el control de bombas centrifugas en las que se desea obtener una presión constante llamada Pump Genius, la cual puede ser utilizada para controlar una sola bomba o varias bombas con el mismo variador, esta puede ser configurada por medio del panel frontal o por medio del

software WEG Ladder Programmer, en la Tabla 15 se muestran las descripciones técnicas del equipo.

Tabla 15 Especificaciones técnicas CWF500

Especificación	Valor
Referencia	CFW500D28P0T2
Voltaje de alimentación	230 V
Potencia nominal de salida	7,5 KW
Corriente nominal de salida	28 A
Grado de protección	IP20
Salida Relé	1
HMI	Alfa-numérico
Entradas analógicas	1
Salidas analógicas	1
Entradas digitales	4
Salidas digitales	1
Comunicación	Modbus
Software	WEG Ladder Programmer
Controlador	PID
Costo aproximado	$ 2.500.000

Otra ventaja que ofrece este variador y la aplicación Pump Genius, es la habilitación de funciones ya establecidas para la correcta marcha del sistema como son: El llenado de tubería, el cual asegura que al encender la bomba esta no inicie a máxima velocidad debido a la falta de presión, sino que entre en una rampa de velocidad hasta que la tubería alcance una presión mínima de trabajo para el control PID; Detección de ruptura de tubería o fuga; protección de bomba en seco; monitoreo de cavitación y desatascamiento para la limpieza del impulsor.

Por último, el CWF500 aunque es un equipo que cuenta con 4 entradas digitales y tan solo una analógica, es así mismo el más apto para el proyecto debido a su bajo precio en comparación con otras referencias, y adicionalmente cuenta con las funciones necesarias para asegurar el correcto desempeño del sistema de presión constante, comunicación Modbus y control PID de lazo cerrado.

G. Calibración sensor de Flujo

Para la calibración del sensor de flujo, se utilizó el montaje de la Figura 28, en el cual se varió el ciclo del PWM y la apertura de la válvula para obtener flujos constantes que permitieran tener un punto de referencia, los cuales fueron verificados mediante el uso de un cronometro y un recipiente de un litro de capacidad con marcas cada 0.25 Litros.

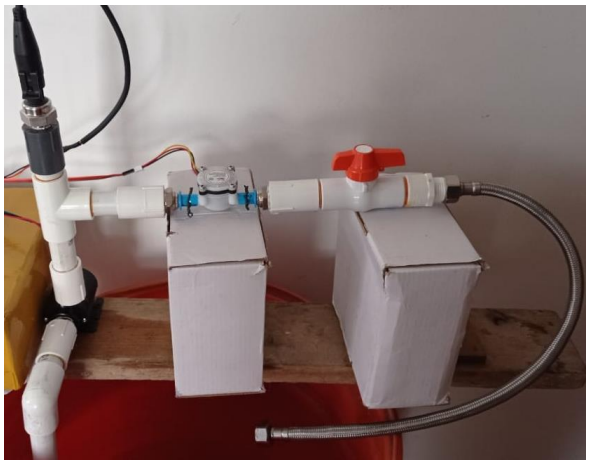

Figura 28 Montaje para la identificación de la curva característica

Se midió el tiempo entre marcas para cada una de las pruebas y se promedió el número de pulsos entregados por el microcontrolador en ese tiempo, para posteriormente hallar su inverso. Los resultados de este proceso se muestran en la Figura 29, donde se pudo encontrar la recta que describe la relación entre la frecuencia de salida y el flujo con coeficiente de correlación de 0,99.

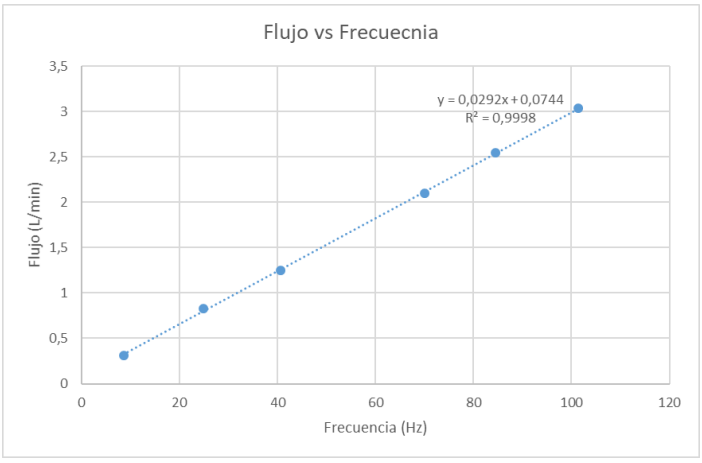

Figura 29 Calibración del sensor de flujo YF-201

H. Bomba de riego

La elección de la bomba de riego se realizó partiendo de las necesidades del cultivo y proyecto, teniendo en cuenta que hay 20 plantas, y cada una requiere de un flujo de 8 litros por hora, se debe garantizar que la bomba de un caudal mínimo de 160 litros por hora. Otro factor fue la alimentación, la cual dada la necesidad de controlar su velocidad y la baja demanda de flujo, se escogió una bomba de corriente continua a 12V, a continuación, se muestra en la Figura 30 el plano del equipo con sus respectivas medidas, inicialmente este fue instalado con un arranque directo operado mediante un temporizador el cual permitía al operador programar los tiempos de encendido y apagado de la bomba, esta instalación no contaba con ningún tipo de control y hacía que la bomba trabajara al 100% de su capacidad a lazo abierto.

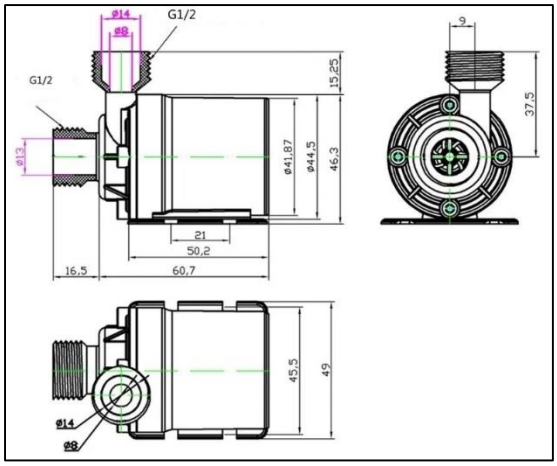

Figura 30 Planos y cotas de la electrobomba de riego

Las especificaciones técnicas de la bomba de riego se muestran en la Tabla 16, es importante resaltar que el caudal máximo es superior al demandado por el cultivo, con el fin de asegurar un correcto suministro de agua, para cada una de las plantas.

Tabla 16 Especificaciones técnicas bomba de agua 12V

Especificación	Valor
Voltaje de alimentación	12 VDC
Potencia	19 W
Corriente	1,58 A
Caudal máximo	800 L/h
Cabezal máximo de agua	5 m

Temperatura máxima del agua	60 °C
Succión	1/2" NPT
Descarga	1/2" NPT

Debido a que el fabricante de la bomba no suministra la curva característica se decidió trazar una curva de presión vs flujo, mediante el montaje mostrado en la Figura 28 , donde se midió la presión a la salida de la bomba en psi y el flujo en litros/min, con el fin de verificar las capacidades del equipo.

La presión fue variada a través del registro que se encuentra ubicado previo a la salida mientras que el voltaje aplicado a la bomba fue de 12 voltios con un PWM al 100%, las mediciones fueron registradas mediante el circuito previamente diseñado, pero enviadas a un computador con un convertidor serial TTL a usb, las cuales fueran recibidas y almacenadas en un Excel con la ayuda de un script en Python. El resultado de este ejercicio se muestra en la Figura 31.

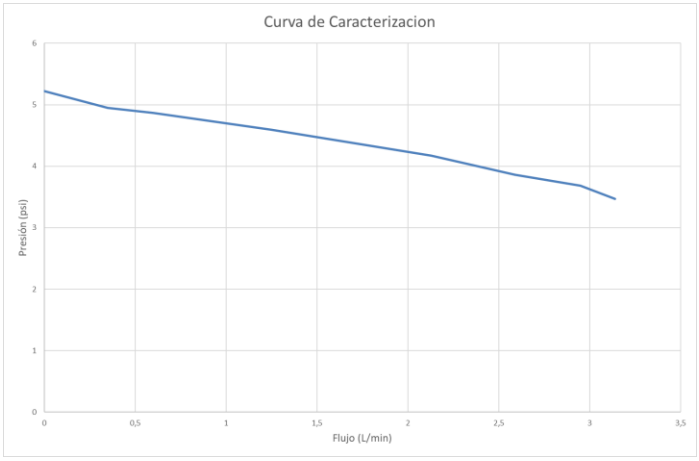

Figura 31 Curva característica de presión vs flujo

Se puede observar cómo la presión máxima cuando el flujo está totalmente restringido es de 5,2 psi, mientras que el flujo máximo con el registro totalmente abierto es de 3,14 L/min, valor más que suficiente para suplir las necesidades de cada una de las líneas de riego, aun así, la presión máxima generada por la bomba limita el uso a goteros auto compensados, pero no antidrenantes los cuales requieren de una presión superior.

I. Comparación de las redes de comunicación

Debido a que el ordenador se encuentra instalado en el cabezal de riego con el control de velocidad, es importante plantear cómo esta podría comunicarse con los sensores instalados en campo, y por tal razón se propuso dos métodos de comunicación diferentes con el fin de compararlos y tomar la elección más adecuada para el proyecto, teniendo en cuenta características como costos de implementación, complejidad, confiabilidad, mantenimiento y sostenibilidad, los dos métodos propuestos son un bus RS485 que se comunique por medio de protocolo Modbus RTU y una red Wifi que comunique módulos Wifi con ordenador por medio de mensajes TCP/IP. a continuación, se presenta el estudio de cada una de las propuestas para posteriormente tomar la elección más conveniente a las necesidades del proyecto.

Bus RS485 con protocoló Modbus RTU

El protocolo Modbus RTU fue desarrollado por Modicon en 1979, es uno de los más usados a nivel industrial dado que facilita la comunicación entre controladores y gran variedad de sensores, este protocolo solo cumple con 3 de las 7 capas del modelo OSI (Defas & Guzmán, 2017), es nórmamele implementados en RS485, un bus de datos que permite conectar hasta 32 dispositivos sin necesidad de repetidores y distancias de 1000 m, fue diseñado para no verse afectado por ruido electromagnético presente en entornos industriales.

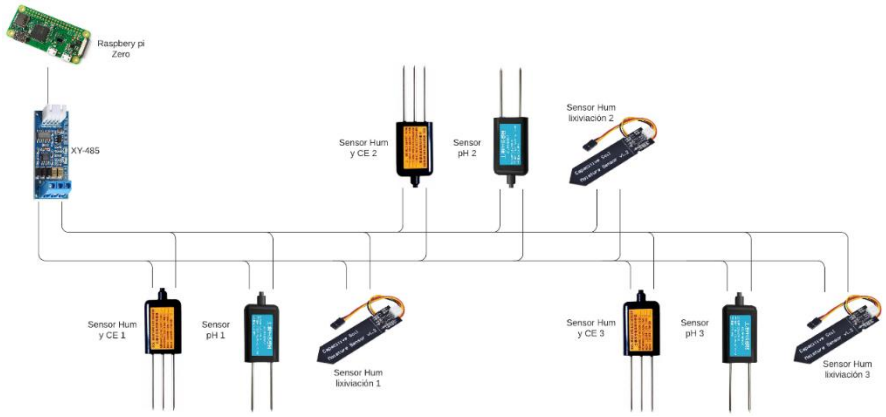

Figura 32 Topología bus red RS-485 con protocolo Modbus

La principal razón por la que se evaluó la implementación de esta red es que los sensores elegidos en las anteriores etapas tienen implementado este protocolo y su integración se

facilitaría, a continuación, se describe el desarrollo de la red para el presente proyecto, para iniciar es necesario la implementación de un módulo XY-485 que convierta la interface RS485 a serial TTL el cual es compatible con el ordenador en el cabezal de riego, la cual será la encargada de funcionar como maestro de la red, mientras que cada uno de los sensores de las materas funcionaria como un esclavo, en la Figura 32 se muestra la topología de bus para la red.

Dado que no todos los sensores utilizados en el proyecto tienen implementado el protocolo Modbus, se requerirá de la compra o desarrollo de microprocesadores que funcionen como Gateway y permitan el acoplamiento de estos al bus de datos, dado que esta red necesita del despliegue de cables también se planteó alimentar los sensores en campo por medio de estos y evitar la inversión en baterías o paneles.

Red Wifi con protocolo TCP/IP

El principal objetivo con la implementación de esta red es que no sea necesario el despliegue de cables a lo largo de la unidad, lo cual implicaría el uso de baterías para alimentar los sensores y un método de recarga, una de las finalidades de esta red sería aprovechar el entorno a campo abierto para utilizar energías renovables y crear módulos que trabajen de forma independiente y se comuniquen por un medio inalámbrico con el ordenador en el cabezal de riego, a continuación, se describirían los dispositivos requeridos para el despliegue de la red, con sus respectivas características.

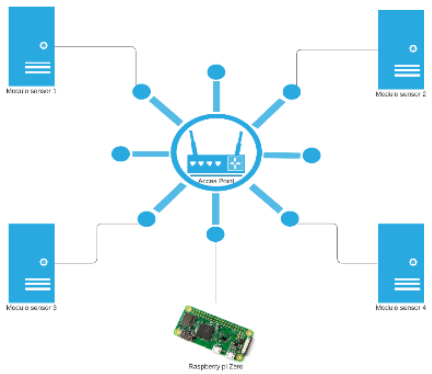

Figura 33 Topología estrella red Wifi con protocolo TCP/IP

En esta configuración se requiere de un Access Point para dar acceso a los distintos módulos, un dispositivo que cumple esta necesidad es el TP-Link The reliable choice modelo TL-WA701ND, el cual cuenta con una antena de 5 dBi a 2,4 GHz, fuente de alimentación a 9 v PoE, soporta hasta 50 dispositivos y con un alcance de aproximadamente 200 m, características más que suficientes, es importante resaltar que las especificaciones de este equipo dependerán de las condiciones del cultivo, en casos donde no existan una ubicación segura de instalación se deben evaluar modelos que sean diseñados para exteriores como el TP-Link Omada EAP110, en la Figura 33 se muestra la topología estrella para la red.

Para la lectura de los sensores en cada una de las materas se evaluaron dos opciones, la primera es la implementación de un módulo ESP32 el cual cuenta con un sistema operativo embebido que permite su programación por medio del IDE de Arduino, haciendo uso de una gran variedad de librerías que simplificarían su desarrollo, como ejemplo la librería ModbusRtu. permite su integración con un módulo MAX485 que convierte la interface Uart a RS485, adicionalmente cuenta con conversor análogo digital y una interface I2C para comunicación con sensores temperatura y humedad relativa, existe una gran variedad de modelos para las tarjetas ESP32, pero una de las más comerciales y completas para desarrollo de proyectos en placa reducida es el "DOIT KIT DEV V-1", en la Tabla 17 se muestran sus especificaciones técnicas.

Tabla 17 Especificaciones técnicas ESP32 DOIT KIT DEV V-1

Especificación	Valor
Voltaje de alimentación	5 V
Voltajes in/out	3,3 V
Consumo activo	80-180 mA
Pines GPIO	24
Entradas ADC-12 bit	12
Wifi	802.11n @ 2.4 GHz hasta 150 Mbit/s
Bluetooth	4.2 BR/EDR
Memoria	448 KByte ROM, 520 KByte SRAM

Después de revisar las especificaciones técnicas de este equipo se puede concluir que estaría sobre dimensionado para el proyecto, debido a que, no se estaría utilizando toda su capacidad y se perdería espació y energía en periféricos que no serían aplicados. La segunda

opción que se evaluó es el uso del módulo ESP-12, el cual puede ser programado con el entorno de arduino y adicionalmente implementado con protocolo TCP/IP. Lo anterior reduciría tanto el tamaño de la placa resultante como el consumo de energía de cada módulo.

Por último, se requiere de una fuente de alimentación la cual permita cargar las baterías y mantener un suministro constante al módulo de medición, dado esto se plantea la implementación de paneles solares de 5 voltios y módulos de carga para baterías de litio, lo cual permitiría que en cultivos de mayor envergadura los módulos de medición se puedan instalar sin limitaciones de disponibilidad de energía.

J. Ordenador de placa reducida

Para elegir el ordenador de placa reducida adecuado, primero se evaluaron las necesidades del proyecto: debía contar con un software que permitiera solicitar la información de los diferentes módulos por el medio del protocolo TCP/IP, adicionalmente ser capaz de almacenar una base de datos, la cual posteriormente sirvió para toma de decisiones y ser capaz de implementar distintas leyes de control por medio de lenguajes de programación como Python, aunque en el mercado existen distintos modelos y marcas se decidió usar la Raspberry pi Zero, debido a su costo, fácil instalación de distintos sistemas operativos y trae incorporado su módulo de conexión Wifi.

La Raspberry requiere de una microSD, la cual sirve como almacenamiento del sistema operático y de los datos recolectados, otra decisión de vital importancia fue el sistema operativo que se instaló en la placa, aunque existen gran variedad de opciones se eligió Debían, el cual fue desarrollado por los mismos fabricantes, algunas de las ventajas de este son: la compatibilidad con la placa, la documentación y que tiene incluidos algunos IDE que permiten el desarrollo con el lenguaje Python como Thonny, aunque que para este proyecto se usó la versión con escritorio para facilitar y agilizar los procesos, sería recomendable usar un versión sin interface gráfica la cual permita optimizar los recursos del procesador.

Dado que la placa utilizada como base de datos y procesador de información es una Raspberry Pi Zero, y aunque esta cuente con un conector para una pantalla externa, no sería eficiente para el proyecto el uso de esta característica, ya que en campo no se contaría con una estructura para realizar dichas conexiones. Por lo anterior se utilizó el protocolo SHH para establecer una conexión remota segura con un ordenador y con ayuda del software VNC Viewer generar un servidor para visualizar y utilizar de forma remota el escritorio de la placa reducida.

El algoritmo que se ejecutó en la Raspberry Pi Zero fue desarrollado en lenguaje Python y buscó que se ejecutaran tres hilos, uno por cada línea de riego, con una frecuencia de tres minutos de ejecución, a continuación, se nombran las librerías utilizadas y su función en el código:

- Serial: Utilizada para enviar y recibir datos por el puerto serial de la Raspberry hacia el control de velocidad.
- Time: Usada para obtener la hora y fecha de lectura.
- Ctypes: Usada para manejo de datos en diferentes formatos.
- Threading: Utilizada para crear los hilos de ejecución.
- Schedule: Utilizada para ejecutar tareas como el cambio de día a una hora especifica.
- sqlite3: Usada para realizar búsquedas y consultas en la base de datos
- socket: Utilizada para crear socket y comunicarse con los módulos de medición.
- RPi.GPIO: Utilizada para el uso de los pines de propósito general integrados en la Raspberry.
- Pandas: Utiliza para manejar los datos extraídos de la base de datos como dataframes, con el fin de facilitar su procesamiento.
- Control: Utilizada para obtener respuestas de modelos de referencia.
- Skfuzzy: Utilizada para crear un control difuso.
- numpy: Utilizada para manejo de datos.

K. Resultados conectividad red Wifi

Con el fin de verificar el correcto funcionamiento de la red implementada y la conectividad entre los distintos dispositivos, se procedió con la conexión de todos los módulos y la Raspberry, y mediante un ordenador adicional se ejecutó el comando ping, este comando es usado para determinar el estado de un host remoto utilizando el protocolo ICMP, el cual envía al host un determinado datagrama para solicitar su respuesta y se ocupa de verificar errores en las redes TCP/IP. Al utilizar el comando, se buscó verificar el estado y su disponibilidad de conexión IP.

Lo anterior sirvió como herramienta para evaluar y visualizar el estado de la red y la cantidad de paquetes perdidos en la comunicación, el comando utilizado fue: ping -t 192.168.0.94 > M1.txt, este fue ejecuta para los tres módulos de censado en cada uno de las líneas y el módulo ubicado en la fuente de agua, una de las ventajas de este comando es que guarda la información del proceso en un archivo de texto plano para su posterior análisis.

```
Estad¡sticas de ping para 192.168.0.94:
    Paquetes: enviados = 601, recibidos = 601, perdidos = 0
    (0% perdidos),
Tiempos aproximados de ida y vuelta en milisegundos:
    M¡nimo = 3ms, M ximo = 130ms, Media = 6ms
```

Figura 34 Resultados comando ping 192.168.94

La Figura 34 muestra los resultados para el módulo ubicado en la línea 3, en este evidenciamos como el porcentaje de mensaje o información perdida es igual a 0%, esto garantiza que la información solicitada por la Raspberry y recolectada por los módulos se transmitirá y recibirá correctamente.

L. Algoritmo implementado en el micro-controlador dsPIC304011

```
#include "xc.h"

#define FCY 20000000UL
#include <libpic30.h>
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <stdbool.h>
#include <ctype.h>
#include <string.h>
#include <libq.h>
#include "Config.h"

#include<adc10.h>
#include "uart.h"
#include<timer.h>
#include<qei.h>
#include<pwm.h>
#include<InCap.h>
#include<ports.h>

void config_ICM7();
void config_QEI();
void config_PWM();
void config_UART();
void config_IO();
void config_TM1();
void config_INT();
void config_ADC();
void send_float(float dato);
void __attribute__((__interrupt__)) _T1Interrupt(void);
void __attribute__((__interrupt__)) _IC7Interrupt(void);
void __attribute__((__interrupt__)) _U1RXInterrupt(void);
void __attribute__((interrupt,auto_psv)) _INT1Interrupt(void);

unsigned int timer_first_edge=0, timer_second_edge=0,Count_flow=0;
unsigned int result[8], Velocidad, i;
unsigned int pos_value, flag;
unsigned int dutycyclereg;
unsigned int dutycycle=0;
unsigned char updatedisable;
float Send=0.0, Dato, Flujo=0,Flow=0, Consumo=0, Litros=0.0, Contador=0,time_delta=0,
Presion=0.0,ciclosc=0.0;
float Flujo_Filter=0.0, Presion_Filter=0.0;
unsigned char *chptr;
unsigned int Rh=0, Rl=0, fr=0;
unsigned int Ciclos=0;
char flag_Encendido=0;
```

```
int Llenado=0;
float SP=0,error_act=0,error_ant=0, control_act=0, control_ant=0;
float kp=13.6980656763313, ki=10.8935718223359, Ts=0.125;
const float Sovol=5.0;
const float alpha=0.18,alpha_P=0.15;
unsigned char *Receiveddata;
unsigned int Datarem;


int main(void) {
    chptr = (unsigned char *)&Send;
    config_INT();
    config_TM1();
    config_ICM7();
    config_IO();
    config_UART();
    config_ADC();
    config_PWM();
    while(1){
        if(PORTDbits.RD0==1){
            LATEbits.LATE1=PORTDbits.RD1;
            LATEbits.LATE2=PORTDbits.RD2;
            LATEbits.LATE3=PORTDbits.RD3;
            SP=3.4;
            Llenado=0;
            Contador=0;
            Litros=0;
            flag_Encendido=1;
            IEC1bits.INT1IE=1;  // Habilito interrupcion de la entrada INT1
            IEC0bits.T1IE=1;     // Habilito Timer 1
            while(flag_Encendido==1){}
            __delay_ms(10000);
            IEC1bits.INT1IE=0;  // Habilito interrupcion de la entrada INT1
            IEC0bits.T1IE=0;     // Desabilito Timer 1
            Flujo=0;
            Presion=0;
            Flujo_Filter=0;
            Presion_Filter=0;
            Contador=0;
            Ciclos=0;
            Litros=0;
            fr=0;
            Rh=0;
            Rl=0;
        }
    }
    return 0;
}
void config_INT(){
    unsigned int config;
```

```
 99      config=FALLING_EDGE_INT & EXT_INT_PRI_7 & EXT_INT_ENABLE & GLOBAL_INT_ENABLE;
100      ConfigINT1(config);
101      IEC1bits.INT1IE=0;  // desabilito interrupcion de la entrada INT1
102  }
103
104  void config_ICM7(){
105      unsigned int config;
106      OpenTimer2(T2_ON & T2_GATE_OFF & T2_PS_1_256  & T2_SOURCE_INT, 0xFFFF);
107      IC7CON=0;
108      ConfigIntCapture7(IC_INT_PRIOR_6 & IC_INT_ON);
109      config=IC_IDLE_STOP & IC_TIMER2_SRC & IC_EVERY_RISE_EDGE & IC_INT_1CAPTURE;
110      OpenCapture7(config);
111      Count_flow=0;
112      IEC1bits.IC7IE=1; // Desabilita interrupcion
113  }
114
115  void config_PWM(){
116      unsigned int period;
117      unsigned int sptime;
118      unsigned int config1;
119      unsigned int config2;
120      unsigned int config3;
121      period=0x1FFF;      //0x1FFF
122      sptime=0;
123      config1=PWM_EN & PWM_OP_SCALE1 & PWM_IPCLK_SCALE1 & PWM_MOD_UPDN;
124      config2=PWM_MOD1_IND & PWM_PDIS1H & PWM_PEN1L & PWM_PDIS2H & PWM_PDIS2L &
125  PWM_PDIS3H & PWM_PDIS3L;
126      config3=0;
127      OpenMCPWM(period, sptime, config1,config2, config3);
128      dutycycle = 0x0000; //
129  }
130
131  void config_UART(){
132      int baud = 10;      //Baute rate 115000
133      unsigned int Reg1_Uart, Reg2_Uart;
134
135      //Configuracion pines Salidas
136      TRISFbits.TRISF3=0;     //Salida TX
137      TRISFbits.TRISF2=1;     //Entrada RX
138      //Configuracion Serial
139      ConfigIntUART1(UART_RX_INT_EN & UART_RX_INT_PR6 & UART_TX_INT_DIS &
140  UART_TX_INT_PR2);
141      Reg1_Uart= UART_EN & UART_IDLE_CON & UART_DIS_WAKE & UART_DIS_LOOPBACK & UART_RX_TX
142  & UART_DIS_ABAUD & UART_NO_PAR_8BIT &
143      UART_1STOPBIT;
144      Reg2_Uart= UART_INT_TX & UART_TX_PIN_NORMAL & UART_TX_ENABLE & UART_INT_RX_CHAR &
145      UART_ADR_DETECT_DIS & UART_RX_OVERRUN_CLEAR;
146      //Inicializacion de UART
147      OpenUART1(Reg1_Uart, Reg2_Uart, baud);
148
```

```
149     //IFS0bits.U1RXIF = 0;
150
151 }
152
153 void config_IO(){
154     ADPCFG=0xffff; //Todo Diigtal
155     TRISEbits.TRISE0=0;  // Puerto E todo salida
156     TRISEbits.TRISE1=0;
157     TRISEbits.TRISE2=0;
158     TRISEbits.TRISE3=0;
159     TRISDbits.TRISD0=1;  // Puerto D todo Entrada
160     TRISDbits.TRISD1=1;
161     TRISDbits.TRISD2=1;
162     TRISDbits.TRISD3=1;
163 }
164
165 void config_TM1(){
166     unsigned int match_value;
167     ConfigIntTimer1(T1_INT_PRIOR_6 & T1_INT_ON);
168     WriteTimer1(0);
169     match_value = 9240;  //Muestreo a 250 ms 65535*250/841=19481.27
170     OpenTimer1(T1_ON & T1_GATE_OFF & T1_IDLE_STOP & T1_PS_1_256 & T1_SYNC_EXT_OFF &
171 T1_SOURCE_INT, match_value);
172 }
173
174 void config_ADC(){
175     unsigned int Channel, PinConfig, Scanselect;
176     unsigned int Adcon3_reg, Adcon2_reg, Adcon1_reg;
177     ADCON1bits.ADON = 0;         /* turn off ADC */
178     Channel = ADC_CHX_NEG_SAMPLEA_NVREF & ADC_CHX_NEG_SAMPLEB_NVREF &
179 ADC_CHX_POS_SAMPLEA_AN0AN1AN2 &
180     ADC_CHX_POS_SAMPLEB_AN0AN1AN2 & ADC_CH0_POS_SAMPLEA_AN3 & ADC_CH0_NEG_SAMPLEA_NVREF
181 & ADC_CH0_POS_SAMPLEB_AN3
182     & ADC_CH0_NEG_SAMPLEB_NVREF;
183     SetChanADC10(Channel);              //Configura los 4 Canales
184     ConfigIntADC10(ADC_INT_DISABLE);    //Desabilita la Interrrupcion
185     Scanselect = SKIP_SCAN_AN4 & SKIP_SCAN_AN5 & SKIP_SCAN_AN6 & SKIP_SCAN_AN7 &
186 SKIP_SCAN_AN8;
187     Adcon1_reg= ADC_MODULE_ON & ADC_IDLE_STOP & ADC_FORMAT_INTG & ADC_CLK_MANUAL &
188 ADC_SAMPLE_SIMULTANEOUS & ADC_AUTO_SAMPLING_ON;
189     Adcon2_reg= ADC_VREF_AVDD_AVSS & ADC_SCAN_OFF & ADC_CONVERT_CH_0ABC;
190     Adcon3_reg= ADC_CONV_CLK_SYSTEM & ADC_CONV_CLK_4Tcy;
191     PinConfig  = ENABLE_AN0_ANA & ENABLE_AN1_ANA & ENABLE_AN2_ANA & ENABLE_AN3_ANA;
192     OpenADC10(Adcon1_reg, Adcon2_reg, Adcon3_reg,PinConfig, Scanselect);
193 }
194
195 void __attribute__((interrupt,auto_psv)) _INT1Interrupt(void){
196     LATEbits.LATE1=0;
197     LATEbits.LATE2=0;
198     LATEbits.LATE3=0;
```

```
199     flag_Encendido=0;   // Apago el motor
200     IFS1bits.INT1IF=0;  // Bandera en 0 de la entrada INT1
201 }
202
203 void __attribute__((__interrupt__)) _IC7Interrupt(void){
204     timer_first_edge=timer_second_edge;
205     ReadCapture7(&timer_second_edge);
206     if(timer_second_edge>=timer_first_edge){
207        time_delta=timer_second_edge-timer_first_edge;
208     }else{
209        time_delta=(65535-timer_first_edge)+timer_second_edge;
210     }
211     if(time_delta>600 || time_delta<9000){
212         if(time_delta==0){
213             Flujo=0;
214         }else{
215             Flow=(2100/time_delta)+0.074;
216             if(Flow<6){
217                 Count_flow=0;
218                 Flujo=Flow;
219             }
220         }
221     }
222     IFS1bits.IC7IF = 0;
223 }
224
225 void __attribute__((__interrupt__)) _T1Interrupt(void){
226     Contador++;
227     Llenado++;
228     if(Contador>Ciclos){
229         LATEbits.LATE1=0;
230         LATEbits.LATE2=0;
231         LATEbits.LATE3=0;
232         SP=0.0;
233     }
234     ADCON1bits.SAMP = 1;
235     while(!ADCON1bits.SAMP);
236     ConvertADC10();
237     while(!BusyADC10());
238     //-------Lectura sensores--------
239     for(i=1;i<3;i++){
240         result[i] = ReadADC10(i);
241     }
242     Dato=result[1]*(Sovol/1023);
243     Presion=(2.5*Dato)-1;
244     if(Presion<0){
245         Presion=0.0;
246     }
247     Presion_Filter=(alpha_P*Presion)+((1.0-alpha_P)*Presion_Filter);
248     error_act=SP-Presion_Filter;
```

```
249     //-------Control Motor--------
250     if(Llenado>120 || Presion_Filter>2.6){
251         control_act=(kp*(error_act-error_ant))+(ki*Ts*error_ant)+control_ant;
252         if(control_act<0){
253             control_act=0;
254         }else if(control_act>100){
255             control_act=100;
256         }
257         Velocidad=(control_act*16383/100);
258         control_ant=control_act;
259 
260         Llenado=161;
261     }else{
262         if(flag_Encendido==1){
263             Velocidad=0x3332;
264             control_ant=80;
265         }else{
266             Velocidad=0x0000;
267         }
268     }
269     error_ant=error_act;
270     Dato=Velocidad*(100.0/16383.0);
271     //-------Filtrado Media Movil Esponencial
272     Flujo_Filter=(alpha*Flujo)+((1.0-alpha)*Flujo_Filter);
273     Consumo=Flujo_Filter*0.125/60;
274     Litros=Litros+Consumo;
275 
276     Count_flow++;
277     if(Count_flow>4){
278        Flujo=0;
279     }
280     send_float(Presion_Filter);
281     send_float(Dato);
282     send_float(Flujo_Filter);
283     send_float(Litros);
284     send_float(Contador);
285     dutycycle = Velocidad;
286     dutycyclereg = 1;
287     updatedisable = 0;
288     SetDCMCPWM (dutycyclereg, dutycycle, updatedisable);
289     WriteTimer1(0);
290     IFS0bits.T1IF = 0;    /* Clear Timer interrupt flag */
291 }
292 void send_float(float dato){
293     Send=dato;
294     while(BusyUART1());
295     WriteUART1(*(chptr+3));
296     while(BusyUART1());
297     WriteUART1(*(chptr+2));
298     while(BusyUART1());
```

```
299     WriteUART1(*(chptr+1));
300     while(BusyUART1());
301     WriteUART1(*(chptr));
302     while(BusyUART1());
303 }
304 void __attribute__((__interrupt__)) _U1RXInterrupt(void){
305
306     Datarem = getsUART1(1, &Rh, 200);
307     __delay_ms(10);
308     Datarem = getsUART1(1, &Rl, 200);
309     Ciclos=(Rh*256)+Rl;
310     IFS0bits.U1RXIF = 0;
311 }
```

M. Algoritmo implementado en las ESP-12F Módulos Materas

```
#include <ModbusRTU.h>
#include <ESP8266WiFi.h>
#include <SoftwareSerial.h>
#include "DHTesp.h"
#define SLAVE_ID_CE 2
#define FIRST_REG_CE 0
#define REG_COUNT_CE 6
#define SLAVE_ID_PH 3
#define FIRST_REG_PH 0
#define REG_COUNT_PH 1
#define SCOUNT  20           // sum of sample point
#define LED_BUILTIN 2
#define CONTROL_5 14
#define CONTROL_3 5
#define VREF 3.3 // analog reference voltage(Volt) of the ADC

float time_sleep=0.0;
String Datos;

//Variables sensot de lixiviacion

//const int Valor_Sensor_Aire = 651; // Valor calculado con el programa de calibración
con el sensor al aire
//const int Valor_Sensor_Agua = 256; // Valor calculado con el programa de calibración
con el sensor sumergido en agua

int analogBuffer[SCOUNT];    // store the analog value in the array, read from ADC
int analogBufferTemp[SCOUNT];
int analogBufferIndex = 0,copyIndex = 0;
float averageVoltage = 0;
#define VREF 3.3      // analog reference voltage(Volt) of the ADC

float Temp, Hum, CE, Salinidad,TDS, Eps, pH, Hum1;
float humidity, temperature, errores=0;
SoftwareSerial S(12,13);
ModbusRTU mb;
/* ADC1_0*/
const int analogInPin = A0;
int sensorValue = 0, filtro=0;
bool CEF=false, PHF=false, HUF=false, AMF=false;

/*Sensor DHT*/
DHTesp dht;

//const char *ssid = "Familia ganti";
//const char *password = "Samuel2020*";

const char *ssid = "RedCultivo";
```

```
49 const char *password = "arandanos2022";
50
51 //const char *ssid = "Galaxy A114607";
52 //const char *password = "shan8693";
53
54 int getMedianNum(int bArray[], int iFilterLen)
55 {
56       int bTab[iFilterLen];
57       for (byte i = 0; i<iFilterLen; i++)
58       bTab[i] = bArray[i];
59       int i, j, bTemp;
60       for (j = 0; j < iFilterLen - 1; j++)
61       {
62       for (i = 0; i < iFilterLen - j - 1; i++)
63           {
64         if (bTab[i] > bTab[i + 1])
65             {
66         bTemp = bTab[i];
67             bTab[i] = bTab[i + 1];
68         bTab[i + 1] = bTemp;
69          }
70       }
71       }
72       if ((iFilterLen & 1) > 0)
73     bTemp = bTab[(iFilterLen - 1) / 2];
74       else
75     bTemp = (bTab[iFilterLen / 2] + bTab[iFilterLen / 2 - 1]) / 2;
76       return bTemp;
77 }
78
79 bool cb_ce(Modbus::ResultCode event, uint16_t transactionId, void* data) { // Callback
80 to monitor errors
81   if (event != Modbus::EX_SUCCESS) {
82     Serial.print("Request result: 0x");
83     Serial.println(event, HEX);
84     CEF=true;
85   }else{
86     Serial.println("Lectura correcta");
87     CEF=false;
88   }
89   return true;
90 }
91
92 bool cb_ph(Modbus::ResultCode event, uint16_t transactionId, void* data) { // Callback
93 to monitor errors
94   if (event != Modbus::EX_SUCCESS) {
95     Serial.print("Request result: 0x");
96     Serial.println(event, HEX);
97     PHF=true;
98   }else{
```

```
99      Serial.println("Lectura correcta");
100     PHF=false;
101   }
102   return true;
103 }
104
105 WiFiServer server(2020);
106
107 void Conectar_red(){
108   WiFi.setSleepMode(WIFI_NONE_SLEEP);
109   Serial.print("Connecting to ");
110   Serial.println(ssid);
111   WiFi.mode(WIFI_STA);
112   WiFi.begin(ssid, password);
113
114   while (WiFi.status() != WL_CONNECTED) {
115     delay(500);
116     Serial.print(".");
117   }
118   Serial.println("");
119   Serial.println("WiFi connected");
120   Serial.println("IP address: ");
121   Serial.println(WiFi.localIP());
122   server.begin();
123 }
124
125 void setup() {
126   Serial.begin(115200);
127   S.begin(9600, SWSERIAL_8N1);
128   pinMode(LED_BUILTIN, OUTPUT);
129   pinMode(CONTROL_5, OUTPUT);
130   pinMode(CONTROL_3, OUTPUT);
131   digitalWrite(CONTROL_5, HIGH);  // Turn the LED off by making the voltage HIGH
132   digitalWrite(CONTROL_3, HIGH);  // Turn the LED off by making the voltage HIGH
133   dht.setup(4, DHTesp::DHT22);
134   mb.begin(&S);
135   mb.master();
136
137   Conectar_red();
138 }
139
140 void loop() {
141   uint16_t res[REG_COUNT_CE];
142   uint16_t res1[REG_COUNT_PH];
143   WiFiClient client = server.available(); // Intenta crear un objeto de cliente
144   if (client) // Si el cliente actual está disponible
145   {
146       Serial.println("[Client connected]");
147       String readBuff;
148       while (client.connected()) // Si el cliente está conectado
```

```
149       {
150         if (client.available()) // Si hay datos legibles
151           {
152               char  c = client.read(); // Leer un byte
153                                        // También puede utilizar otros métodos como
154 readLine ()
155               if(c == 'l') // Retorno de carro recibido
156               {
157                 if (!mb.slave()) {    // Check if no transaction in progress
158                   mb.readHreg(SLAVE_ID_PH, FIRST_REG_PH, res1, REG_COUNT_PH, cb_ph); //
159 Send Read Hreg from Modbus Server
160                   while(mb.slave()) { // Check if transaction is active
161                     mb.task();
162                     delay(10);
163                   }
164                   if(PHF==true){
165                     res1[0]=0.0;
166                   }else{
167                     pH=res1[0]/10.0;
168                   }
169                   Serial.print("Flag:");
170                   Serial.println(PHF);
171                   }
172                   if (!mb.slave()) {    // Check if no transaction in progress
173                     mb.readHreg(SLAVE_ID_CE, FIRST_REG_CE, res, REG_COUNT_CE, cb_ce);
174 // Send Read Hreg from Modbus Server
175                     while(mb.slave()) { // Check if transaction is active
176                       mb.task();
177                       delay(10);
178                    }
179                    if(CEF==true){
180                       res[0]=0;
181                       res[1]=0;
182                       res[2]=0;
183                       res[3]=0;
184                       res[4]=0;
185                       res[5]=0;
186                     }else{
187                       // Conversiones de los sensores Modbus
188                       Temp=(res[0]/100.0)-15.0;
189                       Hum=res[1]/100.0;
190                       CE=res[2];
191                       Salinidad=res[3];
192                       TDS=res[4];
193                       Eps=res[5]/100.0;
194                     }
195                     Serial.print("Flag:");
196                     Serial.println(CEF);
197                   }
198
```

```
                  // read the analog in value
                  for(int i=0;i<SCOUNT;i++){
                     analogBuffer[i] = analogRead(analogInPin);    //read the analog
value and store into the buffer
                     delay(40);
                  }
                  for(copyIndex=0;copyIndex<SCOUNT;copyIndex++)
                    analogBufferTemp[copyIndex]= analogBuffer[copyIndex];
                  filtro=getMedianNum(analogBufferTemp,SCOUNT);
                  averageVoltage = filtro * (float)VREF / 1024.0; // read the analog
value more stable by the median filtering algorithm, and convert to voltage value


                  Hum1=(-47.8129*averageVoltage)+104.75;
                  if(Hum1< 0.0 || Hum1>80.0){
                    HUF=true;
                    Hum1=0.0;
                  }else{
                    HUF=false;
                  }
                  // print the readings in the Serial Monitor
                  Serial.print("sensor = ");
                  Serial.println(Hum1);
                  Serial.print("Voltaje = ");
                  Serial.println(averageVoltage);
                  Serial.print("Flag:");
                  Serial.println(HUF);

                  humidity = dht.getHumidity();
                  temperature = dht.getTemperature();
                  if(humidity>=0 && temperature>=0){
                    AMF=false;
                  }else{
                     AMF=true;
                     humidity = 0.0;
                     temperature = 0.0;
                  }
                  Serial.print(humidity);
                  Serial.print("\t\t");
                  Serial.print(temperature);
                  Serial.println("\t\t");
                  Serial.print("Flag:");
                  Serial.println(AMF);

                  Serial.println("Received: " + readBuff); // Imprimir desde el puerto
serie
                  readBuff = "";
                  boolean Flag = true;
                  errores=0.0;
                  if(CEF){
```

```
                    errores=errores+1;
                  }
                  if(PHF){
                    errores=errores+2;
                  }
                  if(HUF){
                    errores=errores+4;
                  }
                  if(AMF){
                    errores=errores+8;
                  }
                  Datos=(String(Temp)+" "+String(Hum)+" "+String(CE)+"
"+String(Salinidad)+" "+String(TDS)+" "+String(pH)+" "+String(Hum1)+"
"+String(humidity)+" "+String(temperature)+" "+String(errores));
                  while(Flag){
                    client.print(Datos); // Enviar al cliente
                    delay(10);
                    Serial.println("Waiting confirmation...");
                    while (!client.available()){ // Espera confirmacion de recepcion
                      Serial.print(".");
                      delay(50);
                    }
                    char f = client.read();
                    Serial.println(f);
                    if(f=='a'){
                      Serial.println("Confirmado");
                      Flag=false;
                    }
                    if(f=='b'){
                      Serial.println("Error");
                    }
                  }


              }else if(c == 's'){
                  while (!client.available()){ // Espera confirmacion de recepcion
                     Serial.print(".");
                     delay(50);
                  }
                  String Sleep_t = client.readStringUntil('\r');
                  time_sleep=Sleep_t.toFloat();
                  Serial.println(time_sleep);
              }
          }
      }
      client.stop(); // Finalizar la conexión actual:
      Serial.println("[Client disconnected]");
      if(time_sleep!=0.0){
        digitalWrite(CONTROL_5, LOW);  // Turn the LED off by making the voltage HIGH
        WiFi.setSleepMode(WIFI_MODEM_SLEEP);
```

```
299         delay(time_sleep*1000);
300         Conectar_red();
301         time_sleep=0.0;
302         digitalWrite(CONTROL_5, HIGH);  // Turn the LED off by making the voltage HIGH
303         Serial.println("Despertando");
304       }
305     }
306 }
```

N. Algoritmo implementado en las ESP-12F Módulo Fuente

```
#include <ESP8266WiFi.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#define LED_BUILTIN 2
#define CONTROL_5 14
#define CONTROL_3 5
#define VREF 3.3 // analog reference voltage(Volt) of the ADC

float time_sleep=0.0;
String Datos;

#define TdsSensorPin A0
#define VREF 3.3      // analog reference voltage(Volt) of the ADC
#define SCOUNT  30           // sum of sample point
int analogBuffer[SCOUNT];    // store the analog value in the array, read from ADC
int analogBufferTemp[SCOUNT];
int analogBufferIndex = 0,copyIndex = 0;
float averageVoltage = 0,tdsValue = 0, TempA=0;

OneWire ourWire(4);                //Se establece el pin 2 commo bus de datos
DallasTemperature DS18B20(&ourWire); //Se declara una el objeto tipo
DallasTemperature

//const char *ssid = "Familia ganti";
//const char *password = "Samuel2020*";

const char *ssid = "RedCultivo";
const char *password = "arandanos2022";

WiFiServer server(2020);

void Conectar_red(){
  WiFi.setSleepMode(WIFI_NONE_SLEEP);
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  server.begin();
}
```

```
49
50  void setup() {
51    Serial.begin(115200);
52    pinMode(LED_BUILTIN, OUTPUT);
53    pinMode(CONTROL_5, OUTPUT);
54    pinMode(CONTROL_3, OUTPUT);
55    digitalWrite(CONTROL_5, HIGH);  // Turn the LED off by making the voltage HIGH
56    digitalWrite(CONTROL_3, HIGH);  // Turn the LED off by making the voltage HIGH
57
58    pinMode(TdsSensorPin,INPUT);
59    DS18B20.begin();    //Se inicia el el sensor DS18b20
60
61    Serial.print("Connecting to ");
62    Serial.println(ssid);
63    WiFi.mode(WIFI_STA);
64    WiFi.begin(ssid, password);
65
66    while (WiFi.status() != WL_CONNECTED) {
67      delay(500);
68      Serial.print(".");
69    }
70    Serial.println("");
71    Serial.println("WiFi connected");
72    Serial.println("IP address: ");
73    Serial.println(WiFi.localIP());
74    server.begin();
75  }
76
77  void loop() {
78    WiFiClient client = server.available(); // Intenta crear un objeto de cliente
79    if (client) // Si el cliente actual está disponible
80    {
81        Serial.println("[Client connected]");
82        String readBuff;
83        while (client.connected()) // Si el cliente está conectado
84        {
85          if (client.available()) // Si hay datos legibles
86            {
87                char  c = client.read(); // Leer un byte
88                                         // También puede utilizar otros métodos como
89  readLine ()
90                if(c == 'l') // Retorno de carro recibido
91                {
92                    for(int i=0;i<20;i++){
93                       analogBuffer[i] = analogRead(TdsSensorPin);    //read the analog
94  value and store into the buffer
95                       delay(40);
96                    }
97                    DS18B20.requestTemperatures();    //Se envía el comando para leer la
98  temperatura
```

```
                    TempA= DS18B20.getTempCByIndex(0); //Se obtiene la temperatura en
°C
                    TempA= 22.1;
                    for(copyIndex=0;copyIndex<20;copyIndex++)
                      analogBufferTemp[copyIndex]= analogBuffer[copyIndex];
                    averageVoltage = getMedianNum(analogBufferTemp,20) * (float)VREF /
4096.0; // read the analog value more stable by the median filtering algorithm, and
convert to voltage value
                    float compensationCoefficient=1.0+0.02*(TempA-25.0);
//temperature compensation formula: fFinalResult(25^C) =
fFinalResult(current)/(1.0+0.02*(fTP-25.0));
                    float compensationVolatge=averageVoltage/compensationCoefficient;
//temperature compensation

tdsValue=(133.42*compensationVolatge*compensationVolatge*compensationVolatge -
255.86*compensationVolatge*compensationVolatge + 857.39*compensationVolatge)*0.5;
//convert voltage value to tds value
                    Serial.print("TDS Value:");
                    Serial.print(tdsValue,2);
                    Serial.println("ppm");
                    float CE=2*tdsValue;
                    Serial.println("Received: " + readBuff); // Imprimir desde el
puerto serie
                    readBuff = "";
                    boolean Flag = true;
                    Datos=(String(TempA)+" "+String(tdsValue)+" "+String(CE));
                    while(Flag){
                      client.print(Datos); // Enviar al cliente
                      delay(10);
                      Serial.println("Waiting confirmation...");
                      while (!client.available()){ // Espera confirmacion de recepcion
                        Serial.print(".");
                        delay(10);
                      }
                      char f = client.read();
                      if(f=='a'){
                        Serial.println("Confirmado");
                        TempA=0.0;
                        tdsValue=0.0;
                        CE=0.0;
                        Flag=false;
                      }
                      if(f=='b'){
                        Serial.println("Error");
                      }
                    }
                }else if(c == 's'){
                    while (!client.available()){ // Espera confirmacion de recepcion
                       Serial.print(".");
                       delay(50);
```

```
                    }
                    String Sleep_t = client.readStringUntil('\r');
                    time_sleep=Sleep_t.toFloat();
                    Serial.println(time_sleep);
                }
            }
        }
        client.stop(); // Finalizar la conexión actual:
        Serial.println("[Client disconnected]");
        if(time_sleep!=0.0){
          digitalWrite(CONTROL_5, LOW);  // Turn the LED off by making the voltage HIGH
          WiFi.setSleepMode(WIFI_MODEM_SLEEP);
          delay(time_sleep*1000);
          Conectar_red();
          time_sleep=0.0;
          digitalWrite(CONTROL_5, HIGH);  // Turn the LED off by making the voltage
HIGH
          Serial.println("Despertando");
        }
    }
}
int getMedianNum(int bArray[], int iFilterLen)
{
        int bTab[iFilterLen];
        for (byte i = 0; i<iFilterLen; i++)
        bTab[i] = bArray[i];
        int i, j, bTemp;
        for (j = 0; j < iFilterLen - 1; j++)
        {
        for (i = 0; i < iFilterLen - j - 1; i++)
            {
          if (bTab[i] > bTab[i + 1])
              {
          bTemp = bTab[i];
              bTab[i] = bTab[i + 1];
          bTab[i + 1] = bTemp;
           }
        }
        }
        if ((iFilterLen & 1) > 0)
      bTemp = bTab[(iFilterLen - 1) / 2];
        else
      bTemp = (bTab[iFilterLen / 2] + bTab[iFilterLen / 2 - 1]) / 2;
        return bTemp;
}
```

O. Algoritmo implementado en la Raspberry Pi Zero

```
import serial
from time import sleep
from ctypes import *
import threading
import schedule
import sqlite3
import time
import socket
from datetime import datetime,date, timedelta
import RPi.GPIO as GPIO
import pandas as pd
import control as co
import skfuzzy as fuzz
from skfuzzy import control as ctrl
import numpy as np

Start=7
Valvula1=11
Valvula2=13
Valvula3=15
Tiempo_Adicional=210.0

GPIO.setmode(GPIO.BOARD)
GPIO.setup(Start, GPIO.OUT)
GPIO.setup(Valvula1, GPIO.OUT)
GPIO.setup(Valvula2, GPIO.OUT)
GPIO.setup(Valvula3, GPIO.OUT)

Datos= [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0] #Recepcion Datos matera
value=[0.0, 0.0, 0.0, 0.0, 0.0]    #Recepcion datos motor
valor=[0.0, 0.0, 0.0]    #Recepcion datos motor
Ocupado=False
Leyendo=False
Wait=0.0
Id_Dia=0
regar_1=False

def tiempo():
    formato = '%d-%m-%Y'

    # Obtiene fecha/hora local como tupla struct_time
    st_tiempo = time.localtime()

    # Convierte fecha/hora a segundos
    tiempo = time.mktime(st_tiempo)

    # Convierte fecha/hora a cadena
    str_tiempo = time.strftime(formato, st_tiempo)
```

```
    return tiempo, str_tiempo

def binario_a_decimal(numero_binario):
        numero_decimal = 0

        for posicion, digito_string in enumerate(numero_binario[::-1]):
                numero_decimal += int(digito_string) * 2 ** posicion

        return numero_decimal

def enviar_durancion(Duracion):
    Ciclos=int(Duracion/0.125)
    dato = bin(Ciclos)[2:].zfill(16)
    Rh=dato[:8]
    Rl=dato[8:]
    #print(Rh, Rl)
    rList = [binario_a_decimal(Rh), binario_a_decimal(Rl)]
    arr = bytes(rList)
    print(arr)
    Enviado=serialConnection.write(arr)
    print(Enviado)

def convert(s):
    i = int(s, 16)                   # convert from hex to a Python int
    cp = pointer(c_int(i))           # make this into a c integer
    fp = cast(cp, POINTER(c_float))  # cast the int pointer to a float pointer
    return fp.contents.value         # dereference the pointer, get the float


def Riego(nombre, dia,comienzo, Inicio_riego, Duracion,Linea,Id_Riego,flag,
Detenido):
    global Ocupado
    global Wait
    inicio=time.time()
    inicio_p=inicio-comienzo
    Ocupado=True
    #-------Inicia Riego
    if(flag==True):
        serialConnection.reset_input_buffer()
        if(Linea==1):
            GPIO.output(Valvula1,1)
        elif(Linea==2):
            GPIO.output(Valvula2,1)
        elif(Linea==3):
            GPIO.output(Valvula3,1)
        print("Inicio riego... linea: ", Linea)
        GPIO.output(Start,1)
        Inicio_riego=Set_riego_Inicio(Id_Riego, comienzo)
        Parar=False
    else:
```

```
        if(inicio_p-Inicio_riego-Tiempo_Adicional>Duracion):
            print("Saliendo riego...")
            Parar=True
        elif(inicio_p-Inicio_riego>Duracion and Detenido==False):
            print("Termina riego...")
            GPIO.output(Start,0)
            GPIO.output(Valvula1,0)
            GPIO.output(Valvula2,0)
            GPIO.output(Valvula3,0)
            Parar=False
            Detenido=True
        else:
            Parar=False
    Wait=Duracion-(inicio_p-Inicio_riego)+Tiempo_Adicional
    if(Wait<0):
        Wait=0
    print(Wait)
    #-----Conexion Wifi----
    s = socket.socket()
    Flag_conec=True
    if(Linea==1):
        ip="192.168.0.97"
    elif(Linea==2):
        ip="192.168.0.95"
    elif(Linea==3):
        ip="192.168.0.94"
    while(Flag_conec):
        try:
            s.connect((ip,2020))
            s.settimeout(3)
            Flag_conec=False
        except:
            print("Esperando conexion")
    b="l"
    s.send(b.encode())
    Step_dato=round((time.time()-comienzo),3)
    print(Step_dato)
    Flag_rece=True
    while(Flag_rece):
        try:
            mensa=s.recv(1024)
            try:
                Lectura=(mensa.decode().split())
                for i in range(10):
                    Datos[i]=float(Lectura[i])
            except ValueError:
                Datos[1]=0.0
            print(Datos)
            b="a"
            s.send(b.encode())
```

```
            Flag_rece=False
        except socket.error as socketerror:
            print("Error: ", socketerror)
            e="b"
            s.send(e.encode())
    s.close()
    data_left = serialConnection.inWaiting()
    Datos_wait=int(data_left/20)
    conexion=sqlite3.connect("Cultivo.db")
    for i in range(Datos_wait):
        for j in range(5):
            Lector=serialConnection.read(4)
            try:
                value[j]=round(float(convert(Lector.hex())),3)
            except:
                value[j]=0;
        Step_Motor=round((((value[4])*0.125)+Inicio_riego),3)
        conexion.execute("""insert into Control_Velocidad(Id_Riego,Presion, PWM,
Flujo, Consumo, Hora_medicion_velocidad) values
(?,?,?,?,?,?)""",(Id_Riego,value[0],value[1],value[2],value[3],Step_Motor))
    conexion.execute("""insert into
Mediciones_Matera(Id_Riego,Temperatura,Humedad,CE,Salinidad,TDS,pH,Lixiviacion,Hum_am
biente,
                     Temp_ambiente, Error, Hora_medicion_matera) values
(?,?,?,?,?,?,?,?,?,?,?,?)""",(Id_Riego,Datos[0],Datos[1],

Datos[2],Datos[3],Datos[4],Datos[5],Datos[6],Datos[7],Datos[8],Datos[9],Step_dato))
    conexion.commit()
    conexion.close()
    fin=time.time()
    delta=2.000-(fin-inicio)
    if(Parar!=True):
        threading.Timer(delta,Riego, ('Lectura de Sensores', dia, comienzo,
Inicio_riego, Duracion,Linea,Id_Riego, False, Detenido)).start()
    else:
        Set_riego_Consumo(Id_Riego, value[3])
        Lectura_fuente('Hilera', comienzo, Id_Riego)
        error_2, error_medio, error_final=Hallar_Errores(Id_Riego)
        Set_Errores(Id_Riego, error_medio, error_final, error_2)
        Ocupado=False
        Wait=0.0
        Supervision('Hilera', comienzo,Linea ,Id_Riego, 0)

def Supervision(nombre,comienzo, Linea, Id_Riego, contador):
    global Ocupado
    global Wait
    global Leyendo
    global Id_Dia
    global regar_1
    if(Ocupado==False):
```

```
199         while(Leyendo==True):
200             print("Leyendo otra linea")
201         Leyendo=True
202         inicio=time.time()
203         Humedad, error= Lectura_Sensores(nombre, comienzo, Linea, Id_Riego)
204         Leyendo=False
205         if(Linea==1):
206             if(regar_1==True):
207                 Ocupado=True
208                 Wait=Tiempo_Adicional
209                 Duracion=150.0
210                 enviar_durancion(Duracion)
211                 regar_1=False
212                 Id_Riego, comienzo=Crear_Riego(Id_Dia, Linea, Duracion)
213                 Crear_Control(Id_Riego, 0, 0, 0, 0, 0)
214                 Lectura_fuente(nombre, comienzo, Id_Riego)
215                 Riego('Lectura de Sensores', Id_Dia,comienzo, inicio, Duracion,
216 Linea,Id_Riego, True, False)
217             else:
218                 contador=contador
219                 fin=time.time()
220                 delta=180.00-(fin-inicio)
221                 threading.Timer(delta,Supervision, ('Hilera',comienzo,Linea,Id_Riego,
222 contador)).start()
223         elif(Linea==2):
224             if(Humedad<30 and error!=1 and contador>40):
225                 Ocupado=True
226                 Wait=Tiempo_Adicional
227                 contador=0
228                 Error_Medio,Error_Final, Error_Cuadratico,
229 Delta_Error=get_Errores(Id_Riego)
230                 Dt=Contro_difuso(Error_Medio, Delta_Error, Error_Final)
231                 Set_Delta_t(Id_Riego, Dt)
232                 Duracion=get_duracion(Id_Riego)+Dt
233                 if(Duracion>100):
234                     Duracion=100
235                 enviar_durancion(Duracion)
236                 Id_Riego, comienzo=Crear_Riego(Id_Dia, Linea, Duracion)
237                 Crear_Control(Id_Riego, 0, 0, 0, Error_Medio, 0)
238                 Lectura_fuente(nombre, comienzo, Id_Riego)
239                 Riego('Lectura de Sensores', Id_Dia,comienzo, inicio, Duracion,
240 Linea,Id_Riego, True, False)
241             else:
242                 fin=time.time()
243                 contador=contador+1
244                 delta=180.00-(fin-inicio)
245                 threading.Timer(delta,Supervision, ('Hilera',comienzo,Linea,Id_Riego,
246 contador)).start()
247         elif(Linea==3):
248             if(Humedad<30 and error!=1 and contador>40):
```

```
                Ocupado=True
                Wait=Tiempo_Adicional
                contador=0
                Error_Medio,Error_Final, Error_Cuadratico,
Delta_Error=get_Errores(Id_Riego)
                Dt=Control_p(Error_Cuadratico,Error_Medio,Error_Final)
                Set_Delta_t(Id_Riego, Dt)
                Duracion=get_duracion(Id_Riego)+Dt
                if(Duracion>100):
                    Duracion=100
                enviar_durancion(Duracion)
                Id_Riego, comienzo=Crear_Riego(Id_Dia, Linea, Duracion)
                Crear_Control(Id_Riego, 0, 0, 0, Error_Medio, 0)
                Lectura_fuente(nombre, comienzo, Id_Riego)
                Riego('Lectura de Sensores', Id_Dia,comienzo, inicio, Duracion,
Linea,Id_Riego, True, False)
            else:
                fin=time.time()
                delta=180.00-(fin-inicio)
                contador=contador+1
                threading.Timer(delta,Supervision, ('Hilera',comienzo,Linea,Id_Riego,
contador)).start()
    else:
        print("Esperando...", Wait)
        threading.Timer(Wait+2+Linea,Supervision, ('Hilera',comienzo, Linea,Id_Riego,
contador)).start()

def Lectura_Sensores(nombre, comienzo, Linea, Id_Riego):
    s = socket.socket()
    Flag_conec=True
    if(Linea==1):
        ip="192.168.0.97"
    elif(Linea==2):
        ip="192.168.0.95"
    elif(Linea==3):
        ip="192.168.0.94"
    while(Flag_conec):
        try:
            s.connect((ip,2020))
            s.settimeout(2)
            Flag_conec=False
        except:
            print("Esperando conexion")
    b="1"
    s.send(b.encode())
    inicio=time.time()
    Step_dato=round((inicio-comienzo),3)
    conexion=sqlite3.connect("Cultivo.db")
    sleep(0.1)
    Flag_rece=True
```

```
299     while(Flag_rece):
300         try:
301             mensa=s.recv(1024)
302             try:
303                 Lectura=(mensa.decode().split())
304                 for i in range(10):
305                     Datos[i]=float(Lectura[i])
306             except ValueError:
307                 Datos[i]=0.0
308             print("Leyendo linea:", Linea, "tiempo: ", Step_dato)
309             print(Datos)
310             b="a"
311             s.send(b.encode())
312             Flag_rece=False
313         except socket.error as socketerror:
314             print("Error: ", socketerror)
315             e="b"
316             s.send(e.encode())
317     s.close()
318     conexion.execute("""insert into
319 Mediciones_Matera(Id_Riego,Temperatura,Humedad,CE,Salinidad,TDS,pH,Lixiviacion,Hum_am
320 biente,
321                     Temp_ambiente, Error, Hora_medicion_matera) values
322 (?,?,?,?,?,?,?,?,?,?,?)""",(Id_Riego,Datos[0],Datos[1],
323
324 Datos[2],Datos[3],Datos[4],Datos[5],Datos[6],Datos[7],Datos[8],Datos[9],Step_dato))
325     conexion.commit()
326     conexion.close()
327     return Datos[1], Datos[9]
328
329 def Lectura_fuente(nombre, comienzo,Riego):
330     s = socket.socket()
331     Flag_conec=True
332     while(Flag_conec):
333         try:
334             s.connect(("192.168.0.96",2020))
335             s.settimeout(2)
336             Flag_conec=False
337         except:
338             print("Esperando conexion")
339     b="l"
340     s.send(b.encode())
341     inicio=time.time()
342     Step_fuente=round((inicio-comienzo),3)
343     print(Step_fuente)
344     conexion=sqlite3.connect("Cultivo.db")
345     sleep(0.1)
346     Flag_rece=True
347     while(Flag_rece):
348         try:
```

```
            mensa=s.recv(1024)
            try:
                Lectura=(mensa.decode().split())
                for i in range(3):
                    valor[i]=float(Lectura[i])
            except ValueError:
                valor[i]=0.0
            print(valor)
            b="a"
            s.send(b.encode())
            Flag_rece=False
        except socket.error as socketerror:
            print("Error: ", socketerror)
            e="b"
            s.send(e.encode())
    s.close()
    conexion.execute("""insert into Mediciones_Fuente(Id_Riego, Temperatura,
TDS_Agua, CE_agua, Hora_medicion_fuente)
                    values (?,?,?,?,?)""" ,(Riego,valor[0],valor[1],valor[2],
Step_fuente))
    conexion.commit()
    conexion.close()

def dormir_sensores(nombre, Linea, Sleep):
    s = socket.socket()
    Flag_conec=True
    while(Flag_conec):
        try:
            s.connect(("192.168.0.97",2020))
            s.settimeout(3)
            Flag_conec=False
        except:
            print("Esperando conexion")
    b="s"
    s.send(b.encode())
    Sleep_str=str(Sleep)+"\r"
    s.send(Sleep_str.encode())
    s.close()

def Crear_Riego(Id_Dia, Linea, Duracion):
    Inicio_riego=time.time()
    conexion=sqlite3.connect("Cultivo.db")
    cursor=conexion.execute("""select Hora_inicio FROM Dia where
Id_Dia=?""",(str(Id_Dia),))
    comienzo,=cursor.fetchone()
    Inicio_riego=round(time.time()-comienzo,3)
    conexion.execute("""insert into Riego(Id_Dia, Id_Linea, Hora_inicio, Duracion)
                        values (?,?,?,?)""" ,(Id_Dia,Linea,Inicio_riego, Duracion))
    conexion.commit()

```

```
399     cursor=conexion.execute("""select Id_Riego FROM Riego where Id_Linea=? ORDER BY
400 Id_Riego DESC""",(str(Linea),))
401     Id,=cursor.fetchone()
402     cursor=conexion.execute("""select Hora_inicio FROM Dia where
403 Id_Dia=?""",(Id_Dia,))
404     comienzo, =cursor.fetchone()
405     conexion.close()
406     return (Id, comienzo)
407 
408 def Crear_Control(Id_Riego, Error_medio, Error_final, Error_cuadratico, Delta_e,
409 Delta_t):
410     conexion=sqlite3.connect("Cultivo.db")
411     conexion.execute("""insert into Control(Id_Riego, Error_Medio, Error_Final,
412 Error_Cuadratico, Delta_Error,
413                         Delta_Tiempo)values (?,?,?,?,?,?)""" ,(Id_Riego, Error_medio,
414 Error_final, Error_cuadratico,
415                                                                 Delta_e, Delta_t))
416     conexion.commit()
417 
418 def get_duracion(Id_Riego):
419     conexion=sqlite3.connect("Cultivo.db")
420     cursor=conexion.execute("""select Duracion FROM Riego where
421 Id_Riego=?""",(str(Id_Riego),))
422     Duracion,=cursor.fetchone()
423     conexion.commit()
424     conexion.close()
425     return Duracion
426 
427 def Set_riego_Inicio(Id_Riego, comienzo):
428     Inicio_riego=round(time.time()-comienzo,3)
429     conexion=sqlite3.connect("Cultivo.db")
430     conexion.execute("""UPDATE Riego SET Hora_inicio=? WHERE Id_Riego=?"""
431 ,(Inicio_riego,str(Id_Riego)))
432     conexion.commit()
433     conexion.close()
434     return Inicio_riego
435 
436 def Set_riego_Consumo(Id_Riego, consumo):
437     conexion=sqlite3.connect("Cultivo.db")
438     conexion.execute("""UPDATE Riego SET Consumo=? WHERE Id_Riego=?"""
439 ,(consumo,str(Id_Riego)))
440     conexion.commit()
441     conexion.close()
442 
443 def get_Errores(Id_Riego):
444     conexion=sqlite3.connect("Cultivo.db")
445     cursor=conexion.execute("""select Error_Medio, Error_Final, Error_Cuadratico,
446 Delta_Error FROM Control where
447                             Id_Riego=?""",(str(Id_Riego),))
448     Error_M, Error_F, Error_C, Delta_E,=cursor.fetchone()
```

```
449     conexion.commit()
450     conexion.close()
451     return Error_M, Error_F, Error_C, Delta_E
452 
453 def Set_Delta_t(Id_Riego, Delta_t):
454     conexion=sqlite3.connect("Cultivo.db")
455     conexion.execute("""UPDATE Control SET Delta_Tiempo=? WHERE Id_Riego=?"""
456 ,(Delta_t,str(Id_Riego)))
457     conexion.commit()
458     conexion.close()
459 
460 def Set_Errores(Id_Riego, Error_Medio, Error_Final, Error_Cuadratico):
461     conexion=sqlite3.connect("Cultivo.db")
462     cursor=conexion.execute("""select Delta_Error FROM Control where
463 Id_Riego=?""",(str(Id_Riego),))
464     Error_ant,=cursor.fetchone()
465     Delta_Error=Error_Medio-Error_ant
466     conexion.execute("""UPDATE Control SET Error_Medio=?, Error_Final=?,
467 Error_Cuadratico=?, Delta_Error=? WHERE
468                         Id_Riego=?""" ,(Error_Medio, Error_Final, Error_Cuadratico,
469 Delta_Error,str(Id_Riego)))
470     conexion.commit()
471     conexion.close()
472 
473 def Iniciar_dia():
474     global Id_Dia
475     dia, str_dia = tiempo()
476     print('COMIENZO:', str_dia)
477     comienzo=time.time()
478     conexion=sqlite3.connect("Cultivo.db")
479     conexion.execute("""insert into Dia(Fecha, Hora_inicio)
480                       values (?,?)""" ,(str_dia,comienzo))
481     conexion.commit()
482     cursor=conexion.execute("""select Id_Dia FROM Dia ORDER BY Id_Dia DESC""")
483     Id_Dia, =cursor.fetchone()
484     print("Inicia dia ",Id_Dia)
485     conexion.close()
486 
487 def Regar_1():
488     global regar_1
489     regar_1=True
490 
491 def Inicializar_Riego(Id_Dia, Linea, Duracion):
492     conexion=sqlite3.connect("Cultivo.db")
493     cursor=conexion.execute("""select Id_Riego FROM Riego where Id_Linea=? ORDER BY
494 Id_Riego DESC""",(str(Linea),))
495     try:
496         Id,=cursor.fetchone()
497         Existe=True
498     except:
```

```
        Existe=False
    if(Existe):
        cursor=conexion.execute("""select Id_Dia FROM Riego where
Id_Riego=?""",(Id,))
        Id_Dia,=cursor.fetchone()
        cursor=conexion.execute("""select Hora_inicio FROM Dia where
Id_Dia=?""",(Id_Dia,))
        comienzo, =cursor.fetchone()
    else:
        Id, comienzo =Crear_Riego(Id_Dia, Linea,Duracion)
        Crear_Control(Id, 0, 0, 0, 0, 0)
    conexion.close()
    return (Id, comienzo)

def HumedadvsPulso(Riego):
    Resultados=[]
    Flujo=[]
    conexion=sqlite3.connect("Cultivo.db")
    cursor=conexion.execute("""select Humedad, Hora_medicion_matera from
Mediciones_Matera,
                            Riego where Mediciones_Matera.Id_Riego=? and
Mediciones_Matera.Id_Riego=Riego.Id_Riego""",(Riego,))
    for fila in cursor:
        Resultados.append(fila)
    df_results=pd.DataFrame(Resultados,
            columns= ['Humedad', 'Step'])
    df_results['Flujo'] = 0.0
    cursor=conexion.execute("""select Flujo, Hora_medicion_velocidad from
Control_Velocidad, Riego where
                        Control_Velocidad.Id_Riego=? and
Control_Velocidad.Id_Riego=Riego.Id_Riego""",(Riego,))
    for fila in cursor:
        Flujo.append(fila)
    conexion.close()
    df_flujo=pd.DataFrame(Flujo,
            columns= ['Flujo', 'Step'])
    In=0
    for i in range(len(df_results)):
        Step_1=df_results.loc[i,"Step"]
        for j in range(In,len(df_flujo)):
            Step_2=df_flujo.loc[j,"Step"]
            if((Step_2>Step_1-0.0625)and(Step_2<Step_1+0.0625)):
                df_results.loc[i,"Flujo"]=df_flujo.loc[j,"Flujo"]
                In=j
                break
    return df_results

def Contro_difuso(error, delta, error_final):
    Con.input['Error_Medio']=error
    Con.input['Delta_Error_Medio']=delta
```

```
549     Con.input['Error_Final']=error_final
550     Con.compute()
551     return Con.output['Delta_Tiempo']
552 
553 def Control_p(error_Cuadratico, error, error_final):
554     if(error<0):
555         error_Cuadratico=(-1)*error_Cuadratico
556     Delta=(-0.9476*error_Cuadratico)+(-1.1514*error_final)
557     if(Delta>15):
558         Delta=15
559     if(Delta<-15):
560         Delta=-15
561     return Delta
562 
563 def Hallar_Errores(Id_Riego):
564     delay=40
565     t_step=210
566     nd = [-.4588,0.2144]
567     dn = [1,0.2118,0.008917]
568     FOPDT = co.tf(nd,dn)
569     df_1=HumedadvsPulso(Id_Riego)
570     Hum_ini=df_1.loc[0,"Humedad"]
571     t_ini=df_1.loc[0,"Step"]
572     tr=np.array(df_1["Step"])
573     tr=tr-t_ini
574     hr=np.array(df_1["Humedad"])
575     hr=hr-Hum_ini
576     hs=hr.copy()
577     i=0
578     j=0
579     ti  = np.arange(0, t_step+0.1 , 0.1)
580     t1,y1= co.step_response(FOPDT,ti)
581     y1=y1*0.133
582     error_acumulado=0
583     error_acumulado_cua=0
584     for tn in tr:
585         if(0<=tn and tn<=delay):
586             hs[i]=0
587             error_acumulado=error_acumulado+(hr[i]-hs[i])
588             error_acumulado_cua=error_acumulado_cua+((hr[i]-hs[i])**2)
589             i=i+1
590         elif(delay<tn and tn<=t_step):
591             t=int((round(tn, 1)-delay)*10)
592             hs[j+i]=y1[t]
593             error_acumulado=error_acumulado+(hr[j+i]-hs[j+i])
594             error_acumulado_cua=error_acumulado_cua+((hr[j+i]-hs[j+i])**2)
595             j=j+1
596         elif(tn>t_step):
597             error_final=hr[j+i-1]-hs[j+i-1]
598             break
```

```
599      error=error_acumulado/(j+i+1)
600      error_2=error_acumulado_cua/(j+i+1)
601      td = np.split(tr,[i+1,j+i+1])
602      return  error_2, error, error_final
603
604  serialConnection = serial.Serial ("/dev/ttyS0", 115000)    #Open port with baud rate
605  conexion=sqlite3.connect("Cultivo.db")
606  cursor=conexion.execute("""select Id_Dia FROM Dia ORDER BY Id_Dia DESC""")
607  Id_Dia, =cursor.fetchone()
608  cursor=conexion.execute("""select Fecha FROM Dia where Id_Dia=?""",(Id_Dia,))
609  str_dia, =cursor.fetchone()
610  conexion.close()
611
612  ## Declara el controlador Difuso
613
614  #Declaracion del universo de las entradas
615  EM = ctrl.Antecedent(np.arange(-3, 8, 0.1),'Error_Medio')
616  DEM = ctrl.Antecedent(np.arange(-6, 6,0.1),'Delta_Error_Medio')
617  EF = ctrl.Antecedent(np.arange(-3, 8, 0.1),'Error_Final')
618  DT = ctrl.Consequent(np.arange(-10, 10, 0.1),'Delta_Tiempo')
619
620  EM['NA']=fuzz.sigmf(EM.universe,-2.5, -8)
621  EM['NM']=fuzz.gaussmf(EM.universe,-1.8, 0.2)
622  EM['ZE']=fuzz.gaussmf(EM.universe,0, 0.6)
623  EM['PM']=fuzz.gaussmf(EM.universe,3, 0.7)
624  EM['PA']=fuzz.sigmf(EM.universe,5, 5)
625
626  DEM['NA']=fuzz.sigmf(DEM.universe,-3, -2)
627  DEM['ZE']=fuzz.gaussmf(DEM.universe,0, 1)
628  DEM['PA']=fuzz.sigmf(DEM.universe,4,2)
629
630  EF['N']=fuzz.sigmf(EF.universe,-1.5, -3)
631  EF['ZE']=fuzz.gaussmf(EF.universe,0, 0.5)
632  EF['PM']=fuzz.gaussmf(EF.universe,2, 0.5)
633  EF['PMA']=fuzz.gaussmf(EF.universe,4, 0.5)
634  EF['PA']=fuzz.sigmf(EF.universe,5.5, 3)
635
636  DT['NA']=fuzz.sigmf(DT.universe,-7, -1.5)
637  DT['NM']=fuzz.gaussmf(DT.universe,-4, 1)
638  DT['ZE']=fuzz.gaussmf(DT.universe,0, 1)
639  DT['PM']=fuzz.gaussmf(DT.universe,4, 1)
640  DT['PA']=fuzz.sigmf(DT.universe,7, 1.5)
641
642  rule1= ctrl.Rule(EM['NA'] & DEM['NA'] & EF['N'], DT['PM'])
643  rule2= ctrl.Rule(EM['NM'] & DEM['NA'] & EF['N'], DT['PM'])
644  rule3= ctrl.Rule(EM['ZE'] & DEM['NA'] & EF['N'], DT['PM'])
645  rule4= ctrl.Rule(EM['PM'] & DEM['NA'] & EF['N'], DT['NM'])
646  rule5= ctrl.Rule(EM['PA'] & DEM['NA'] & EF['N'], DT['NM'])
647  rule6= ctrl.Rule(EM['NA'] & DEM['ZE'] & EF['N'], DT['PA'])
648  rule7= ctrl.Rule(EM['NM'] & DEM['ZE'] & EF['N'], DT['PA'])
```

```
649 rule8= ctrl.Rule(EM['ZE'] & DEM['ZE'] & EF['N'], DT['PM'])
650 rule9= ctrl.Rule(EM['PM'] & DEM['ZE'] & EF['N'], DT['NA'])
651 rule10= ctrl.Rule(EM['PA'] & DEM['ZE'] & EF['N'], DT['NA'])
652 rule11= ctrl.Rule(EM['NA'] & DEM['PA'] & EF['N'], DT['PM'])
653 rule12= ctrl.Rule(EM['NM'] & DEM['PA'] & EF['N'], DT['PM'])
654 rule13= ctrl.Rule(EM['ZE'] & DEM['PA'] & EF['N'], DT['PM'])
655 rule14= ctrl.Rule(EM['PM'] & DEM['PA'] & EF['N'], DT['NM'])
656 rule15= ctrl.Rule(EM['PA'] & DEM['PA'] & EF['N'], DT['NM'])
657 rule16= ctrl.Rule(EM['NA'] & DEM['NA'] & EF['ZE'], DT['PM'])
658 rule17= ctrl.Rule(EM['NM'] & DEM['NA'] & EF['ZE'], DT['ZE'])
659 rule18= ctrl.Rule(EM['ZE'] & DEM['NA'] & EF['ZE'], DT['ZE'])
660 rule19= ctrl.Rule(EM['PM'] & DEM['NA'] & EF['ZE'], DT['NM'])
661 rule20= ctrl.Rule(EM['PA'] & DEM['NA'] & EF['ZE'], DT['NM'])
662 rule21= ctrl.Rule(EM['NA'] & DEM['ZE'] & EF['ZE'], DT['PA'])
663 rule22= ctrl.Rule(EM['NM'] & DEM['ZE'] & EF['ZE'], DT['PM'])
664 rule23= ctrl.Rule(EM['ZE'] & DEM['ZE'] & EF['ZE'], DT['ZE'])
665 rule24= ctrl.Rule(EM['PM'] & DEM['ZE'] & EF['ZE'], DT['NA'])
666 rule25= ctrl.Rule(EM['PA'] & DEM['ZE'] & EF['ZE'], DT['NA'])
667 rule26= ctrl.Rule(EM['NA'] & DEM['PA'] & EF['ZE'], DT['PM'])
668 rule27= ctrl.Rule(EM['NM'] & DEM['PA'] & EF['ZE'], DT['ZE'])
669 rule28= ctrl.Rule(EM['ZE'] & DEM['PA'] & EF['ZE'], DT['ZE'])
670 rule29= ctrl.Rule(EM['PM'] & DEM['PA'] & EF['ZE'], DT['NM'])
671 rule30= ctrl.Rule(EM['PA'] & DEM['PA'] & EF['ZE'], DT['NM'])
672 rule31= ctrl.Rule(EM['NA'] & DEM['NA'] & EF['PM'], DT['PM'])
673 rule32= ctrl.Rule(EM['NM'] & DEM['NA'] & EF['PM'], DT['ZE'])
674 rule33= ctrl.Rule(EM['ZE'] & DEM['NA'] & EF['PM'], DT['NM'])
675 rule34= ctrl.Rule(EM['PM'] & DEM['NA'] & EF['PM'], DT['NM'])
676 rule35= ctrl.Rule(EM['PA'] & DEM['NA'] & EF['PM'], DT['NM'])
677 rule36= ctrl.Rule(EM['NA'] & DEM['ZE'] & EF['PM'], DT['PM'])
678 rule37= ctrl.Rule(EM['NM'] & DEM['ZE'] & EF['PM'], DT['ZE'])
679 rule38= ctrl.Rule(EM['ZE'] & DEM['ZE'] & EF['PM'], DT['NM'])
680 rule39= ctrl.Rule(EM['PM'] & DEM['ZE'] & EF['PM'], DT['NA'])
681 rule40= ctrl.Rule(EM['PA'] & DEM['ZE'] & EF['PM'], DT['NA'])
682 rule41= ctrl.Rule(EM['NA'] & DEM['PA'] & EF['PM'], DT['PM'])
683 rule42= ctrl.Rule(EM['NM'] & DEM['PA'] & EF['PM'], DT['ZE'])
684 rule43= ctrl.Rule(EM['ZE'] & DEM['PA'] & EF['PM'], DT['NM'])
685 rule44= ctrl.Rule(EM['PM'] & DEM['PA'] & EF['PM'], DT['NM'])
686 rule45= ctrl.Rule(EM['PA'] & DEM['PA'] & EF['PM'], DT['NM'])
687 rule46= ctrl.Rule(EM['NA'] & DEM['NA'] & EF['PMA'], DT['ZE'])
688 rule47= ctrl.Rule(EM['NM'] & DEM['NA'] & EF['PMA'], DT['PM'])
689 rule48= ctrl.Rule(EM['ZE'] & DEM['NA'] & EF['PMA'], DT['NM'])
690 rule49= ctrl.Rule(EM['PM'] & DEM['NA'] & EF['PMA'], DT['NM'])
691 rule50= ctrl.Rule(EM['PA'] & DEM['NA'] & EF['PMA'], DT['NM'])
692 rule51= ctrl.Rule(EM['NA'] & DEM['ZE'] & EF['PMA'], DT['ZE'])
693 rule52= ctrl.Rule(EM['NM'] & DEM['ZE'] & EF['PMA'], DT['PM'])
694 rule53= ctrl.Rule(EM['ZE'] & DEM['ZE'] & EF['PMA'], DT['NM'])
695 rule54= ctrl.Rule(EM['PM'] & DEM['ZE'] & EF['PMA'], DT['NA'])
696 rule55= ctrl.Rule(EM['PA'] & DEM['ZE'] & EF['PMA'], DT['NA'])
697 rule56= ctrl.Rule(EM['NA'] & DEM['PA'] & EF['PMA'], DT['ZE'])
698 rule57= ctrl.Rule(EM['NM'] & DEM['PA'] & EF['PMA'], DT['PM'])
```

```
699 rule58= ctrl.Rule(EM['ZE'] & DEM['PA'] & EF['PMA'], DT['NM'])
700 rule59= ctrl.Rule(EM['PM'] & DEM['PA'] & EF['PMA'], DT['NM'])
701 rule60= ctrl.Rule(EM['PA'] & DEM['PA'] & EF['PMA'], DT['NM'])
702 rule61= ctrl.Rule(EM['NA'] & DEM['NA'] & EF['PA'], DT['ZE'])
703 rule62= ctrl.Rule(EM['NM'] & DEM['NA'] & EF['PA'], DT['ZE'])
704 rule63= ctrl.Rule(EM['ZE'] & DEM['NA'] & EF['PA'], DT['NM'])
705 rule64= ctrl.Rule(EM['PM'] & DEM['NA'] & EF['PA'], DT['NM'])
706 rule65= ctrl.Rule(EM['PA'] & DEM['NA'] & EF['PA'], DT['NA'])
707 rule66= ctrl.Rule(EM['NA'] & DEM['ZE'] & EF['PA'], DT['ZE'])
708 rule67= ctrl.Rule(EM['NM'] & DEM['ZE'] & EF['PA'], DT['ZE'])
709 rule68= ctrl.Rule(EM['ZE'] & DEM['ZE'] & EF['PA'], DT['NM'])
710 rule69= ctrl.Rule(EM['PM'] & DEM['ZE'] & EF['PA'], DT['NA'])
711 rule70= ctrl.Rule(EM['PA'] & DEM['ZE'] & EF['PA'], DT['NA'])
712 rule71= ctrl.Rule(EM['NA'] & DEM['PA'] & EF['PA'], DT['ZE'])
713 rule72= ctrl.Rule(EM['NM'] & DEM['PA'] & EF['PA'], DT['ZE'])
714 rule73= ctrl.Rule(EM['ZE'] & DEM['PA'] & EF['PA'], DT['NM'])
715 rule74= ctrl.Rule(EM['PM'] & DEM['PA'] & EF['PA'], DT['NM'])
716 rule75= ctrl.Rule(EM['PA'] & DEM['PA'] & EF['PA'], DT['NM'])
717
718 C1= ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5, rule6, rule7, rule8,
719 rule9, rule10, rule11, rule12, rule13, rule14,
720                         rule15, rule16, rule17, rule18, rule19, rule20, rule21,
721 rule22, rule23, rule24, rule25, rule26, rule27,
722                         rule28, rule29, rule30, rule31, rule32, rule33, rule34,
723 rule35, rule36, rule37, rule38, rule39, rule40,
724                         rule41, rule42, rule43, rule44, rule45, rule46, rule47,
725 rule48, rule49, rule50, rule51, rule52, rule53,
726                         rule54, rule55, rule56, rule57, rule58, rule59, rule60,
727 rule61, rule62, rule63, rule64, rule65, rule66,
728                         rule67, rule68, rule69, rule70, rule71, rule72, rule73,
729 rule74, rule75 ])
730 Con= ctrl.ControlSystemSimulation(C1)
731
732 schedule.every().day.at("00:01").do(Iniciar_dia)
733 schedule.every().day.at("07:00").do(Regar_1)
734 schedule.every().day.at("19:00").do(Regar_1)
735
736 GPIO.output(Start,0)
737 GPIO.output(Valvula1,0)
738 GPIO.output(Valvula2,0)
739 GPIO.output(Valvula3,0)
740
741 Linea=1
742 Id_Riego, comienzo =Inicializar_Riego(Id_Dia, Linea,90)
743 Supervision('Hilera',comienzo, Linea, Id_Riego, 0)
744 Linea=2
745 Id_Riego, comienzo =Inicializar_Riego(Id_Dia, Linea,30)
746 Supervision('Hilera',comienzo, Linea, Id_Riego, 40)
747 Linea=3
748 Id_Riego, comienzo =Inicializar_Riego(Id_Dia, Linea,30)
```

```
749 Supervision('Hilera',comienzo, Linea, Id_Riego, 41)
750
751 while True:
752     schedule.run_pending()
753     t=1
754
755 final, str_final = tiempo()
756 print('FINAL   :', str_final)
```

P. Riegos en segunda y tercera línea

A continuación, la Tabla 18 muestra un resumen general de cada uno de los riegos realizados para identificar la respuesta óptima para el riego:

Tabla 18 Resumen riegos varios

Id	Línea	Duración	Delay	Delta	Error Cuadrático	Error Medio	Error Final
1	2	30	40	0,4	3,82	-1,5147	-2,79
2	3	40	40	1	2,95	-1,3937	-2,25
3	3	40	60	2	1,75	-1,0729	-1,2539
4	2	60	25	7,8	16,18	3,636	4,352
5	3	60	40	10	24,7	3,9	7,2
6	3	40	40	8	6,6138	1,8	4,56
7	2	40	25	11	88,91	8,3	7,9
8	3	40	50	6	1,63	0,39	2,46
9	2	20	20	3	1,14	0,5	-0,14
10	3	30	25	3	1,65	-0,87	0,27
11	2	30	35	7	7,23	2,05	4,15
12	2	60	26	6,5	16,61	3,5	3,09
13	2	30	210	0	5,08	-1,75	-3,19
14	3	30	75	2	3,13	-1,408	-1,67
15	2	30	10	1,8	1,17	-0,6	-1,34
16	3	30	50	1,9	1,33	-0,93	-1,33
17	3	30	50	2	0,92	-0,78	-1,23
18	2	30	15	6,5	23,9	4,26	3,73
19	2	30	20	10	58,4	6,7	7
20	3	30	210	0	4,9	-1,75	-3,11
21	3	30	210	0	4,89	-1,74	-3,11
22	3	30	45	3	2,13	-1,12	-0,58
23	2	25	0	1,6	3,26	-1,47	-1,62
24	2	25	20	3,2	0,43	0,34	0,1
25	3	30	25	2,4	2,39	-1,17	-1,1
26	3	30	20	2,8	1,4	-0,94	-0,53
27	2	35	50	2,8	0,71	-0,67	-0,33
28	3	35	40	6,2	4,25	1,64	3,14
29	3	30	40	1,8	3,16	-1,5	-1,5
30	3	30	40	3,3	1,09	-0,75	-0,03
31	3	25	5	2,5	1,59	-1	-0,65
32	3	25	15	3,9	0,55	-0,35	0,48
33	2	25	35	4	0,63	0,7	0,82
34	2	25	40	1,5	3,49	-1,56	-1,7

35	2	25	40	2,2	1,94	-1,16	-1
36	2	25	25	4	0,39	0,57	0,81
37	2	20	100	0,8	5,14	-1,91	-2,5
38	2	20	100	1	4,84	-1,86	-2,22
39	2	20	25	8	27,84	4,78	4,74
40	2	20	25	7	28,74	4,83	4,31
41	3	40	40	7	5,04	1,54	3,64
42	3	40	30	7,5	12,45	3,06	4,3

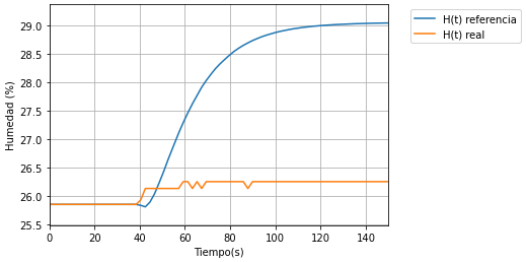

Figura 35 Respuesta humedad Riego Id 1

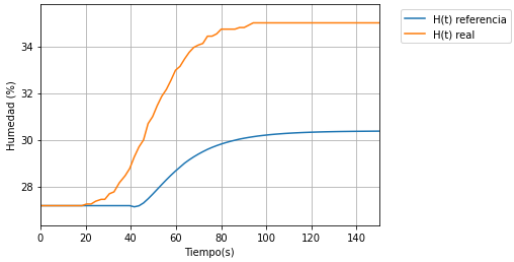

Figura 38 Respuesta humedad Riego Id 4

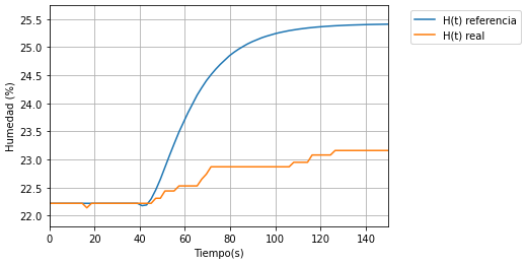

Figura 36 Respuesta humedad Riego Id 2

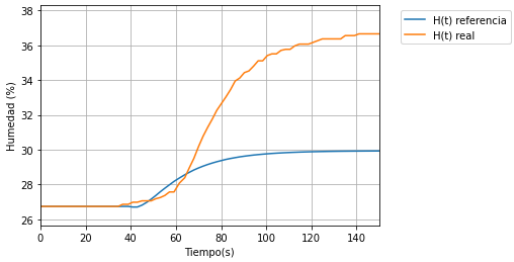

Figura 39 Respuesta humedad Riego Id 5

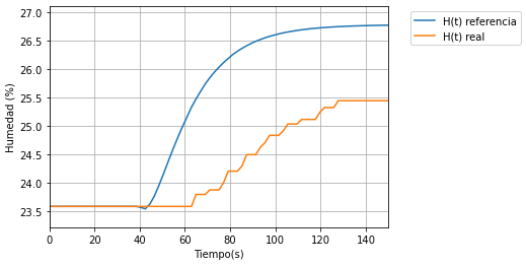

Figura 37 Respuesta humedad Riego Id 3

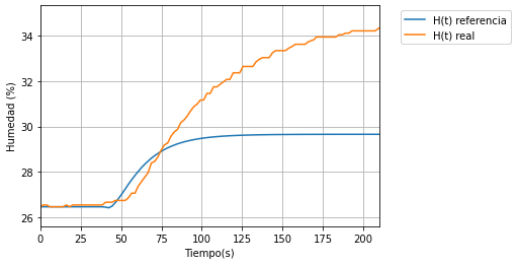

Figura 40 Respuesta humedad Riego Id 6

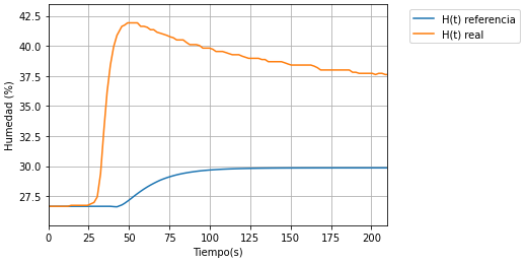

Figura 41 Respuesta humedad Riego Id 7

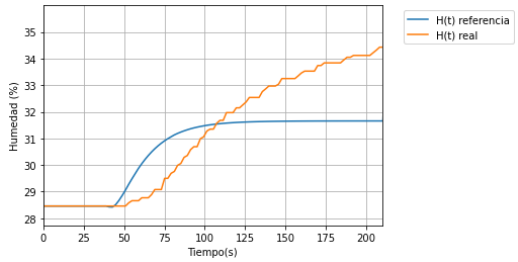

Figura 42 Respuesta humedad Riego Id 8

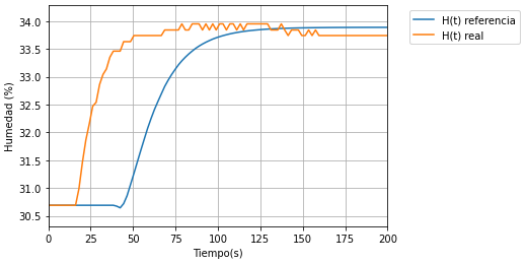

Figura 43 Respuesta humedad Riego Id 9

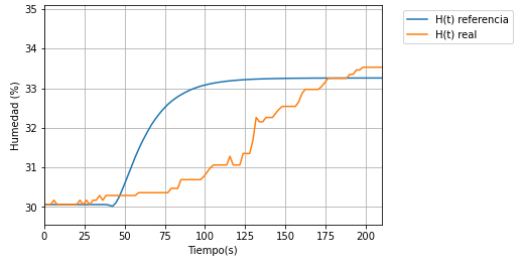

Figura 44 Respuesta humedad Riego Id 10

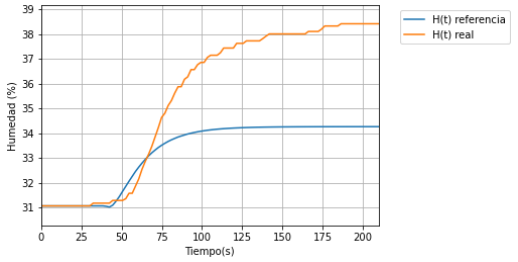

Figura 45 Respuesta humedad Riego Id 11

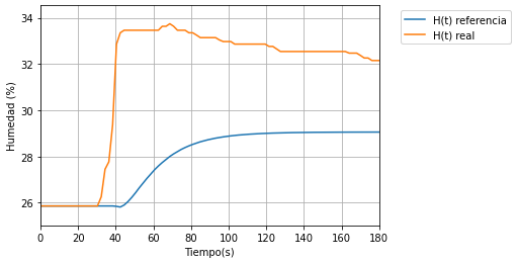

Figura 46 Respuesta humedad Riego Id 12

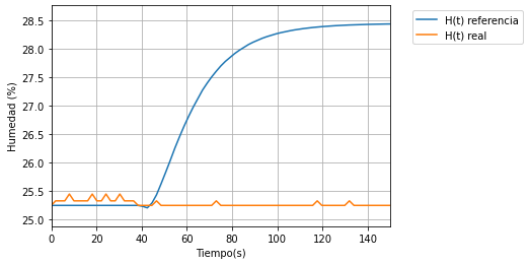

Figura 47 Respuesta humedad Riego Id 13

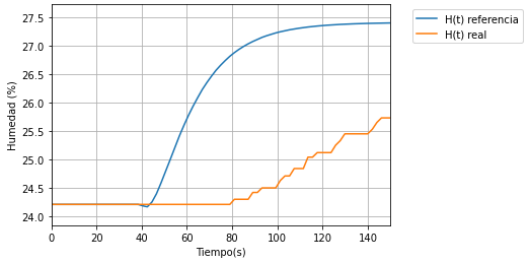

Figura 48 Respuesta humedad Riego Id 14

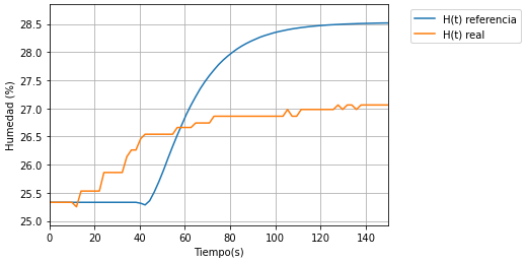

Figura 49 Respuesta humedad Riego Id 15

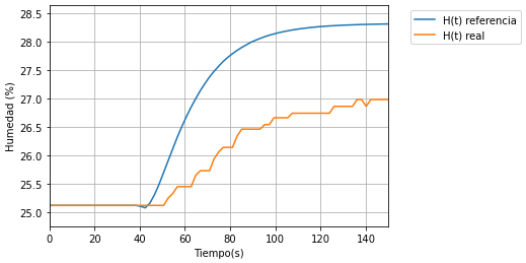

Figura 50 Respuesta humedad Riego Id 16

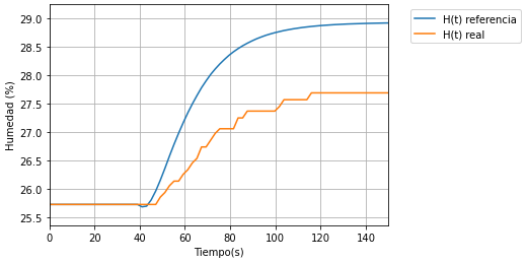

Figura 51 Respuesta humedad Riego Id 17

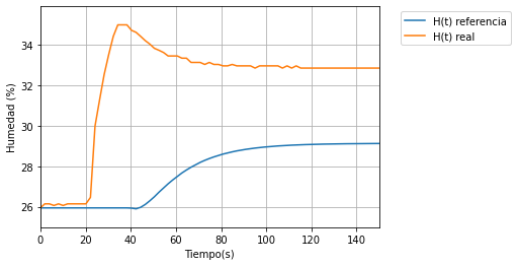

Figura 52 Respuesta humedad Riego Id 18

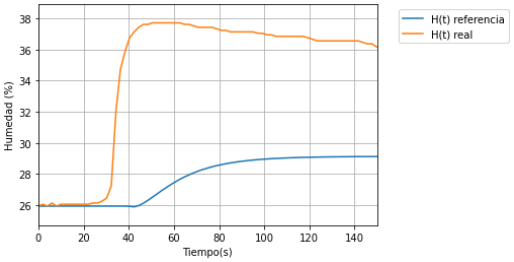

Figura 53 Respuesta humedad Riego Id 19

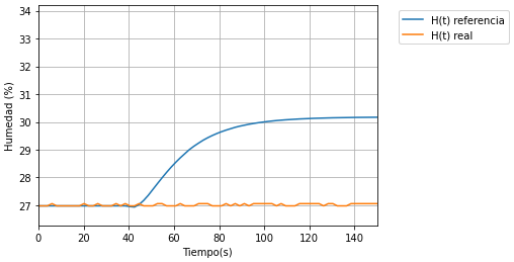

Figura 54 Respuesta humedad Riego Id 20

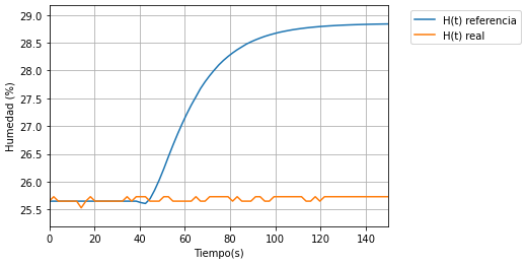

Figura 55 Respuesta humedad Riego Id 21

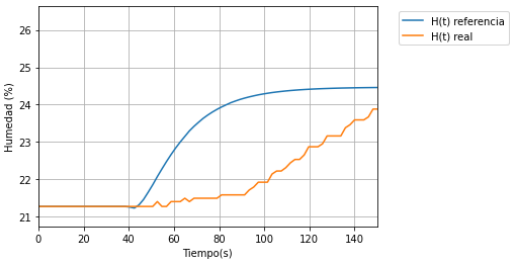

Figura 56 Respuesta humedad Riego Id 22

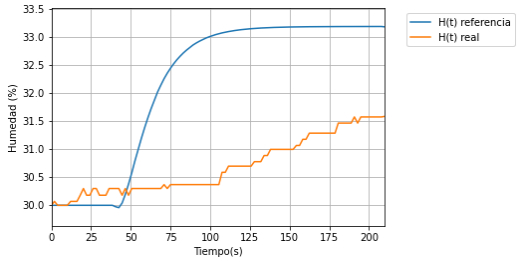

Figura 57 Respuesta humedad Riego Id 23

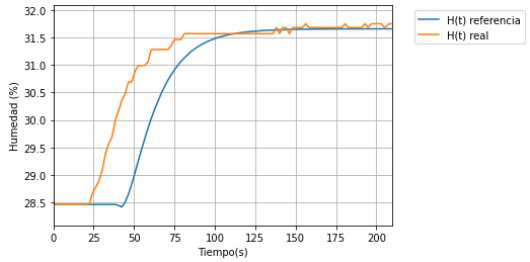

Figura 58 Respuesta humedad Riego Id 24

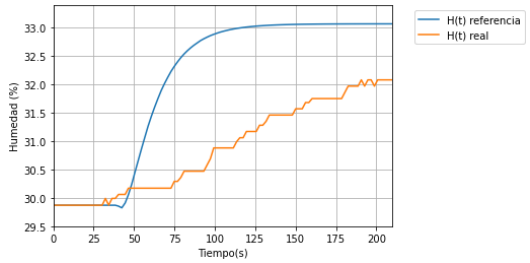

Figura 59 Respuesta humedad Riego Id 25

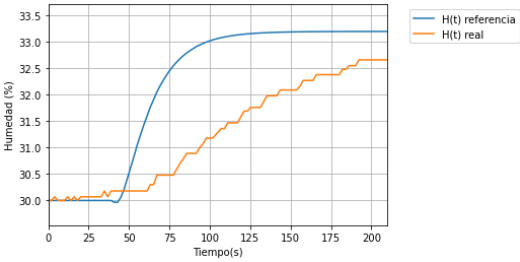

Figura 60 Respuesta humedad Riego Id 26

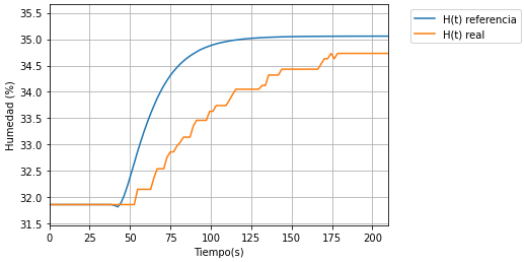

Figura 61 Respuesta humedad Riego Id 27

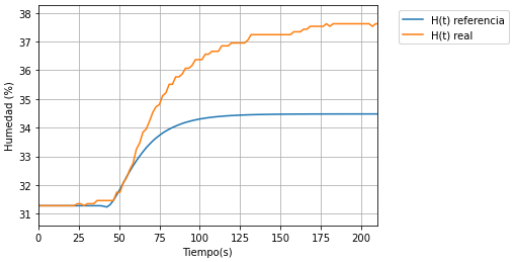

Figura 62 Respuesta humedad Riego Id 28

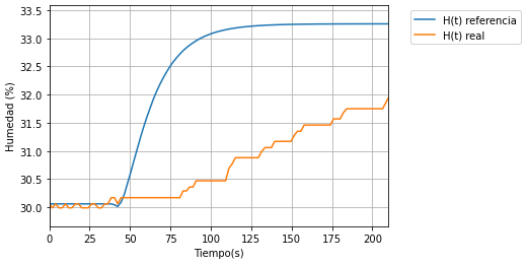

Figura 63 Respuesta humedad Riego Id 29

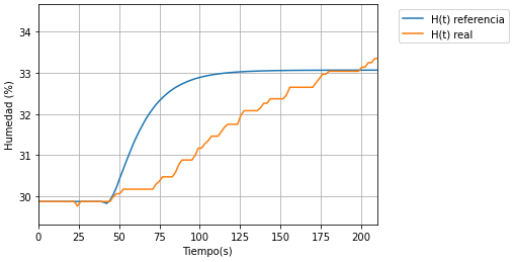

Figura 64 Respuesta humedad Riego Id 30

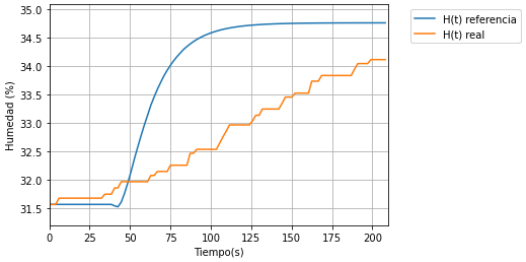

Figura 65 Respuesta humedad Riego Id 31

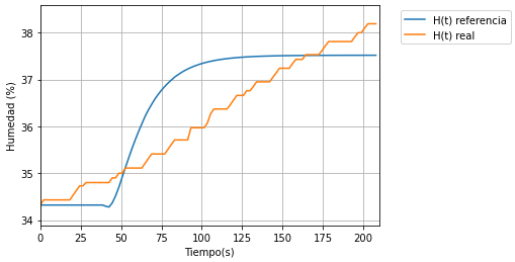

Figura 66 Respuesta humedad Riego Id 32

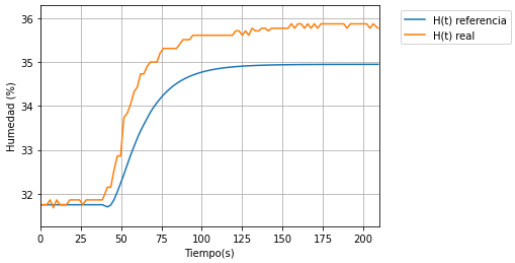

Figura 67 Respuesta humedad Riego Id 33

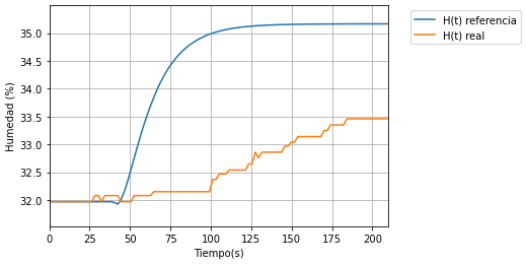

Figura 68 Respuesta humedad Riego Id 34

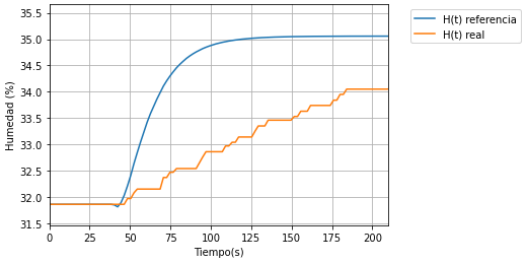

Figura 69 Respuesta humedad Riego Id 35

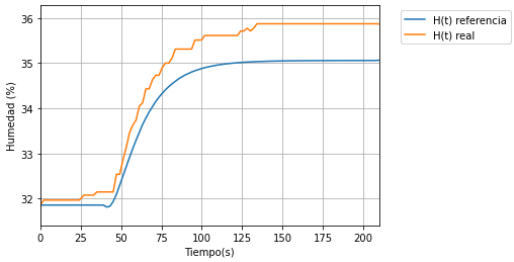

Figura 70 Respuesta humedad Riego Id 36

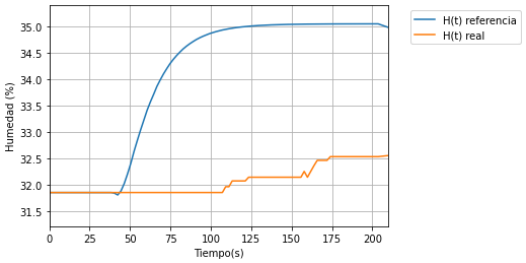

Figura 71 Respuesta humedad Riego Id 37

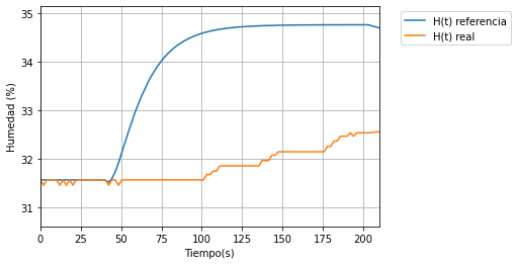

Figura 72 Respuesta humedad Riego Id 38

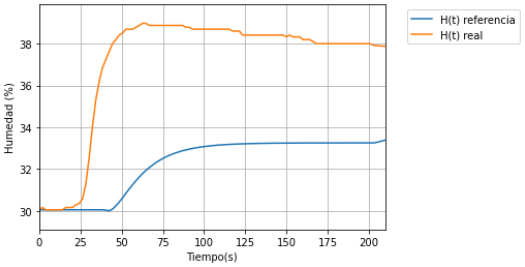

Figura 73 Respuesta humedad Riego Id 39

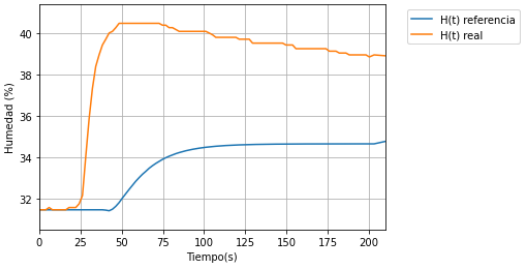

Figura 74 Respuesta humedad Riego Id 40

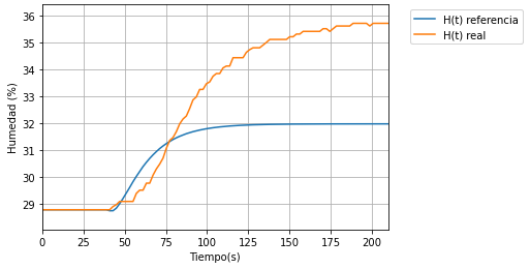

Figura 75 Respuesta humedad Riego Id 41

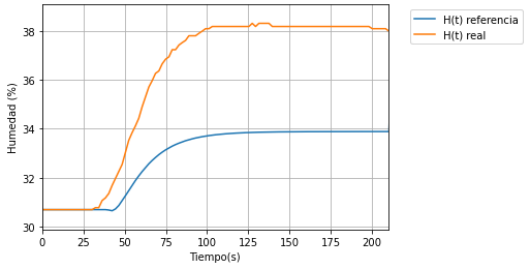

Figura 76 Respuesta humedad Riego Id 42

Q. Simulación de los controladores

En la presente sección se muestran las simulaciones realizadas en lenguaje Python para los controladores propuestos. Continuando, a partir de la planta se verifico cuantas iteraciones requería cada uno de los controladores para alcanzar un error mínimo al iniciar con un tiempo de 30 segundos. Debido a que la planta utilizada en esta simulación presenta diferente forma a la deseada, no se buscó que los controladores lograran tener una respuesta perfecta, sino cual era la respuesta optima que lograba obtener cada uno.

Dado que se utilizaron el error medio y el error final entre ambas curvas, se priorizo que el que tendiera más rápido a 0 fuera el medio, dado que este era el principal indicador de lixiviación y el que más aportaría a reducirlo.

Para el caso del controlador difuso se necesitaron un total de 5 iteraciones para lograr un error medio de 0.02, y un error final de 0.56, con un tiempo de riego de 27.3 segundos, pero al continuar con más iteraciones se observó que en la décima obtuvo un error medio de -0.02, y un error final de 0.48, con un tiempo de riego de 26.8 segundos. La Figura 77 muestra el resultado de humedad al riego y la curva de referencia.

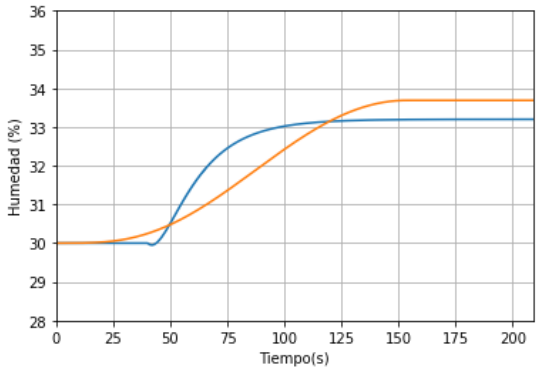

Figura 77 Resultado respuesta control difuso con tiempo de 26,8 s

Se evidencia como el controlador actúa inicialmente para obtener una curva aproximada utilizando el error medio y después con ayuda del error final realiza un ajuste fino con el fin de obtener una respuesta más cercana a la deseada. Sin embargo, dadas las notables diferencia entre la forma de las dos curvas no se obtiene un aproximado perfecto.

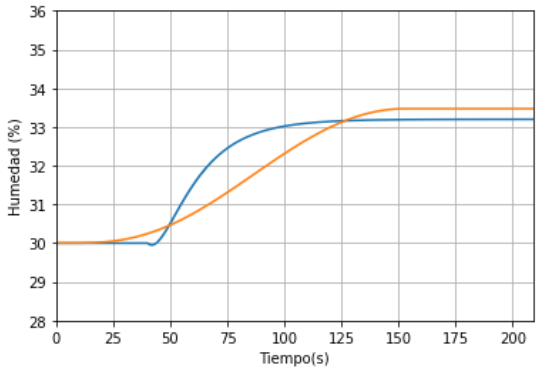

Figura 78 Resultado respuesta control proporcional con tiempo de 25,3 s

Para el caso del controlador proporcional se necesitaron un total de 5 iteraciones para lograr un error medio de -0.08, y un error final de 0.4, con un tiempo de riego de 26.0 segundos, pero al continuar con más iteraciones se observó que en la décima obtuvo un error medio de -0.13, y un error final de 0.27, con un tiempo de riego de 25.3 segundos. La Figura 78 muestra el resultado de humedad al riego y la curva de referencia.

Se evidencia que al igual que el controlador difuso la acción sobre el tiempo primero busca disminuir la diferencia entra la forma de las curvas, y posterior con ayuda del error final realiza un ajuste fino. Aunque para este caso fue más agresivo dado que el error utilizado por este controlador fue el cuadrático, haciendo que necesitara un menor número de iteraciones.

Estas simulaciones permitieron identificar como la interacción de los errores propuestos para ambos controladores, facilitaron una correcta acción que tuvo en cuenta tanto la forma de la curva de respuesta como su valor final.