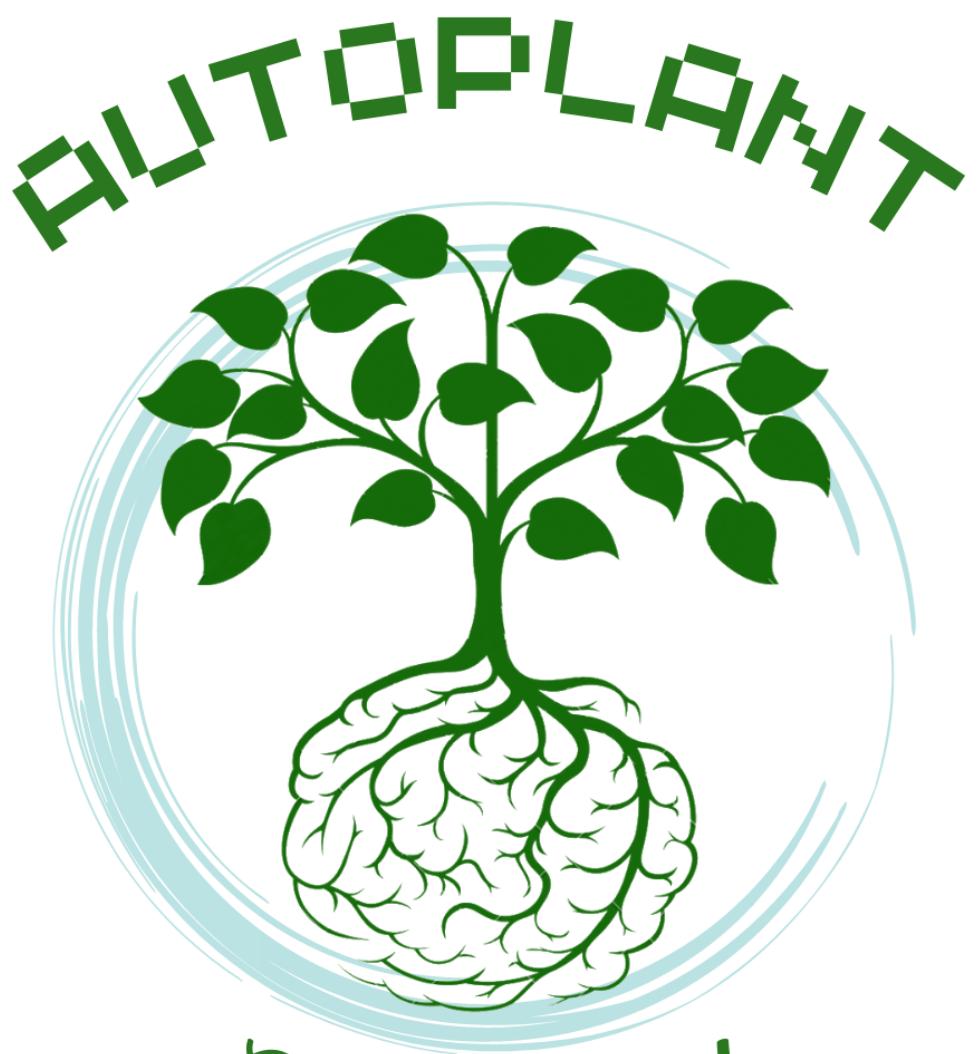


Proyecto integrador 2



Grow Smart

Integrantes:

Nicolas Nemmer 221907
Ingenieria en electronica



Juan Pablo Peyroulou 181251
Ingenieria en electronica



Tutores:

Prof. Emiliano Espíndola González

Prof. Juan Pedro Silva Gallino

Introducción	5
Objetivos	5
Objetivo principal:	5
Objetivos opcionales:	6
Funcionalidades del sistema	7
Casos de uso	7
Elementos físicos del sistema	7
Microcontroladores	8
ESP 32	8
ESP 8266	9
Sensores	10
Módulo sensor de agua	10
Módulo sensor de temperatura y humedad DHT 11	10
Módulo sensor de humedad del suelo	10
Actuadores	11
Ventilador 60x60x10 DC12V	11
Moto bomba	11
LED WS2812 16bit anillo	11
Fuente	12
FUENTE SWITCHING 12V 50W	12
Step down	12
Relay 4 canales	12
Software del sistema	12
Microcontroladores	12
Esp 32	12
Función setup wifi	13
Conexión por MQTT a thingsboard	13
Lectura de sensores	14
Envío de telemetría	15
Reporte de atributo	16
Esp 8266	16
Función setup wifi	16
Conexión por MQTT a thingsboard	17
Conexión por MQTT a thingsboard	18
ThingsBoard	19
Dashboard	19
Rule Chain	20
Prototipos	22
Sistema de control de temperatura y humedad	22
Sistema de control de riego	23
Sistema de control de Luz	23

Plan de trabajo:	24
Pruebas	25
Comunicación placa nube	25
Censo de temperatura y humedad	27
Censo de humedad terrestre	28
Censo de agua del tanque	30
Luces led	31
Rule Chain	33
Encendido de ventiladores	33
Alarms	33
Control de riego	34
Control de luz	34
Control de temperatura y humedad	34
Prueba final	34
Diseño y construcción	36
Diseño inicial	36
Armando de estructura	36
Colocación microcontrolador y fuente	37
Fuente	37
Microcontroladores	37
Step Down	37
Relay	37
Colocación de sensores y actuadores	37
Ventiladores	37
Luces	37
Sensor de humedad y temperatura	38
Sensor de humedad terrestre	38
Tanque de agua	38
Dificultades	38
Conclusiones	39
Mejoras a futuro y defectos a mejorar	39

Introducción

Para este proyecto se tuvo como consigna, realizar un sistema que contenga una red de objetos físicos que se interconectan entre ellos a través de internet. Este tipo de tecnología se llama IoT, Internet de las cosas. De esta manera, podemos conectar diferentes dispositivos que contengan: sensores, actuadores, software, etc. con un software en la nube y de esta manera intercambiar datos para realizar una tarea.

Con esta modalidad de conectividad hemos diseñado nuestro sistema llamado AUTOPLANT. Este proyecto se enfoca en resolver el problema del cuidado de un cultivo de plantas en interior a distancia y de manera automatizada. Este problema, es uno de los principales problemas de cualquier cultivador en interior, ya que no es posible estar el cien por ciento del tiempo observando las plantas o simplemente no es lo ideal. Incluso muchas veces es necesario contratar a alguien para esta tarea, por ejemplo en el caso de irse de vacaciones.

AUTOPLANT propone para solucionar este problema, una aplicación en la nube en la cual se programan perfiles para el estado ideal del cultivo. El sistema, desde la nube, se comunica con dispositivos que sensan el estado actual, el cual se muestra en pantalla, y actúa sobre otros dispositivos para controlar que este estado sea siempre el ideal.

Objetivos

Nuestro objetivo general es ofrecer una solución inteligente y automatizada para un cultivo. La idea es controlar la mayor cantidad de variables posibles para lograr un máximo desempeño del cultivo, estas incluyen luz, humedad terrestre, humedad ambiente, temperatura, oxígeno y CO₂, plagas, entre otras.

Objetivo principal:

Nuestro primer objetivo es enfocarnos en las variables de mayor impacto (luz, humedad, temperatura y riego) para un cultivo “indoor” en sustrato de tierra. En cuanto a luz, se colocará una luz led, de la cual se controlará el horario de encendido/apagado con un cronograma programable desde la nube e intervenible por el usuario, tanto el cronograma como el encendido/apagado en tiempo real.

Para el caso de la temperatura y humedad, se colocará un sensor que mida tanto temperatura como humedad. Este censo se subirá a la nube, la cual contendrá un cronograma con los límites máximos y mínimos de temperatura y humedad acordes a la fase de la planta. En caso que los límites sean superados, se accionará el sistema de control de temperatura y/o humedad, que idealmente debería ser un aire acondicionado que puede tanto bajar como subir la temperatura y la humedad, pero en nuestro caso usaremos una combinación de un ventilador interno y 2 extractores. Los cuales se controlarán para

lograr el objetivo deseado y en caso de no poder lograrlo, se dará un aviso al usuario a través de la interfaz.

Por último, el control de riego se realizará a través de un cronograma combinado con un sensor de humedad terrestre en cada maceta. Por ejemplo, se tiene una planta que por cronograma se dice que debe ser regada cada 3 días. Pero también se conoce que dicha planta necesita que la humedad terrestre no descienda del 10% y que a la hora de regar la humedad esté por debajo del 30%. Teniendo en cuenta esto, se ajustará el cronograma para cumplir con los requerimientos de la planta. Además, se debe tener en cuenta cuando el tanque suministrador de agua se vacíe y avisar al usuario a través de la interfaz para que lo rellene.

Como conclusión habremos logrado cumplir nuestro objetivo principal. Ofreciendo una solución a distancia y controlada desde la nube. La cual permitirá al usuario tener la tranquilidad de que su cultivo estará bien cuidado en su ausencia.

Objetivos opcionales:

Como primer objetivo opcional tenemos expandir la solución para cultivos que no sean cien por ciento en interior, por ejemplo un invernadero. Contando con un sensor de intensidad de luz, el cual de “día” accionará una luz en caso de que la energía solar no sea suficiente para el cultivo. Por ejemplo, un día nublado.

Como segundo objetivo opcional tenemos instalar un sensor de CO2 con el objetivo de reducir los ciclos de ventilación y de esta manera ahorrar energía o en caso de querer potenciar el nivel de CO2 con una bombona. En este último el sensor controlará la llave de apertura del suministro de la bombona para mantener el nivel deseado.

Como tercer objetivo opcional tenemos expandir la solución para sistemas hidropónicos. Aquí se cambiará el sistema de riego por un sistema hidropónico teniendo en cuenta, el cambio a macetas de hidroponía, la colocación de tubería de retorno de agua al tanque y por último la necesidad de un oxigenador en el tanque de agua.

Como cuarto objetivo opcional tenemos la inclusión de aspersores para control de plagas por cronograma y/o manual. Se deberá instalar en el techo de la habitación un aspersor conectado a un tanque con plaguicida. El cual se accionará desde la nube.

Como último objetivo opcional tenemos hacer un Upgrade del sistema de temperatura y/o humedad si el cultivo lo requiere. En caso de tener una planta que requiere una temperatura mayor a la media del ambiente, se debería instalar un sistema de calefacción. Por otro lado, si se desea cultivar una planta que necesite una temperatura menor, se debería instalar un sistema de refrigeración. Lo único que sería necesario cambiar, sería el sistema de temperatura y/o humedad para poder cumplir con los requerimientos del cultivo, pero la lógica debería ser la ya programada.

Funcionalidades del sistema

La funcionalidad principal de este sistema es el control de las variables de un cultivo. Para lograr esto el proyecto se compone de los siguientes sub-sistemas:

- Sistema de control de temperatura y humedad
- Sistema de control de riego
- Sistema de control de luz
- Sistema de reporte de alarmas por medio del dashboard
- Sistema de reporte de humedad, temperatura, humedad terrestre y nivel de tanque de riego
- Sistema de reporte de estado de extractores, ventilador, luces y bomba de agua
- Sistema de accionado de extractores, ventilador, luces y bomba de agua de forma manual
- Sistema de selección y customización de cultivo

Casos de uso

- El usuario no está presente en el cultivo pero aun así puede ver desde la aplicación el estado actual del mismo.
- En el momento que el usuario quiera retirarse del lugar del cultivo, puede setear perfiles para su cuidado y seleccionarlos en la aplicación para controlar el estado actual.
- En caso de no tener instalado un sistema lo suficientemente potente para cumplir con el perfil seleccionado, el usuario va a recibir alarmas en la aplicación indicando que está donde está la carencia.

Elementos físicos del sistema

A continuación describiremos los elementos de hardware que componen nuestro sistema.

Microcontroladores

ESP 32

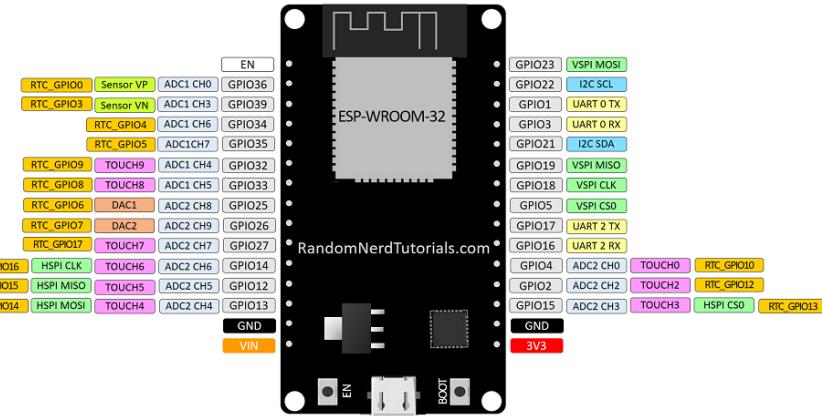
El ESP32 DEVKIT V1 es un microcontrolador de bajo costo y consumo de energía, cuenta con tecnología Wi-Fi y Bluetooth de modo dual integrada que permite controlar todo tipo de sensores, módulos y actuadores; es el sucesor del microcontrolador ESP8266.

Permite generar proyectos de Internet de las cosas “IoT” de forma eficiente y económica, ya que integra internamente una gran cantidad de periféricos incluyendo: sensores táctiles capacitivos, sensor de efecto Hall, amplificadores de bajo ruido, interfaz para tarjeta SD,

Ethernet, SPI de alta velocidad, UART, I2S e I2C. Tiene un CPU de dos núcleos de hasta 240Mhz que se pueden controlar independientemente.

ESP32 DEVKIT V1 – DOIT

version with 30 GPIOs



Especificaciones y características:

- Serie: ESP32 DEVKIT V1
- Chip USB-Serial: CP2102
- Voltaje de Alimentación (USB): 5V DC
- Voltaje de Entradas/Salidas: 3.3V DC
- Consumo de energía de 5 μ A en modo de suspensión
- Pines Digitales GPIO: 24 (Algunos pines solo como entrada)
- Conversor Analógico Digital: Dos ADC de 12bits tipo SAR, soporta mediciones en hasta 18 canales, algunos pines soporta un amplificador con ganancia programable
- Antena en PCB
- Tipo: Módulo Wifi + Bluetooth
- Wifi: 802.11 b/g/n/e/i (802.11n @ 2.4 GHz hasta 150 Mbit/s)
- Bluetooth: 4.2 BR/EDR BLE Modo de control dual
- CPU principal: Tensilica Xtensa 32-bit LX6
- Memoria: 448 KByte ROM, 520 KByte SRAM, 6 KByte SRAM en RTC y QSPI admite múltiples chips flash /SRAM
- Procesador secundario: Permite hacer operaciones básicas en modo de ultra bajo consumo
- Desempeño: Hasta 600 DMIPS
- Frecuencia de Reloj: hasta 240Mhz
- Seguridad: IEEE 802.11, incluyendo WFA, WPA/WPA2 y WAPI
- Criptografía acelerada por hardware: AES, SHA-2, RSA, criptografía de curva elíptica (ECC), generador de números aleatorios (RNG)

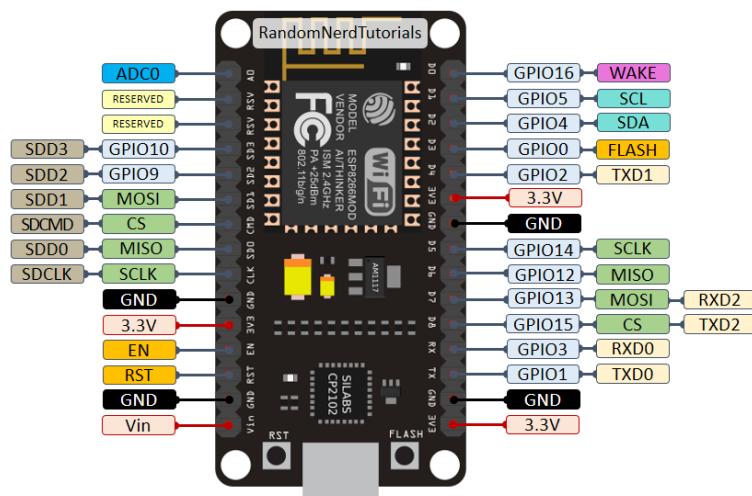
Esta tarjeta se utilizará para conectar los sensores ya que cuenta con hasta 18 entradas analógicas. La misma reportará todos los datos recolectados a la nube por medio de WiFi.

ESP 8266

NodeMCU es un plataforma IoT open source que integra un chip ESP8266, en concreto el ESP-12E. El firmware usa el lenguaje de script Lua.

El chip ESP8266 del módulo WiFi incluye un potente procesador de 32 bits, capaz de ser programado desde el IDE de NodeMCU usando el lenguaje interpretado Lua e incluso desde el IDE de Arduino usando C/C++ compilado. Incluye antena grabada en la propia placa, no necesita antena externa.

Esta plataforma permite programar y usar el ESP8266 directamente sin necesidad de conversores USB o cableado adicional. Incluye pins directos a placa de prototipado y conector micro USB para conexión directa al ordenador y alimentación.



Especificaciones y características:

- Procesador: ESP8266 @ 80MHz (3.3V) (ESP-12E)
- 4MB de memoria FLASH (32 MBit)
- WiFi 802.11 b/g/n
- Regulador 3.3V integrado (500mA)
- Conversor USB-Serial CH340G / CH340G
- Función Auto-reset
- 9 pines GPIO con I2C y SPI
- 1 entrada analógica (1.0V max)
- 4 agujeros de montaje (3mm)
- Pulsador de RESET
- Entrada alimentación externa VIN (20V max)

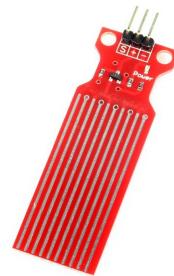
Esta tarjeta tendrá la responsabilidad de controlar los actuadores del sistema según lo indicado por la nube. Esta tarjeta es de precio más reducido que la ESP32 y cumple con nuestros requisitos.

Sensores

Módulo sensor de agua

Este sensor estima el nivel de agua a través de una serie de alambres paralelos expuestos; Su valor de salida analógica puede ser leído directamente por tarjeta de desarrollo Arduino y por lo tanto activar la alarma de nivel de agua.

Tienen tres pines de conexión, dos para 5V y GND, y otro que conectaremos a una entrada analógica de nuestro ESP32.



La utilización de este sensor es para indicar si hay o no agua en el tanque de riego

Módulo sensor de temperatura y humedad DHT 11

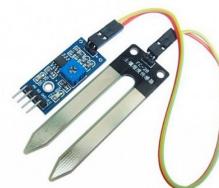
El DHT11 es un sensor básico digital de medición de temperatura y humedad. Este sensor está basado en un termistor que sirve para medir el aire circundante (temperatura) e implementa un sensor interno capacitivo para la medición de humedad. Este dispositivo funciona mediante el uso de tres terminales, Vcc, Gnd y DATA. A través del pin 3 (DATA) se obtiene una señal digital que es tratada a través del microcontrolador ESP32.



Este sensor lo utilizaremos para la medición de temperatura y humedad ambiente dentro de la carpeta.

Módulo sensor de humedad del suelo

Este sensor es un sensor básico que se encarga de medir la humedad del suelo por medio de la resistencia generada entre sus electrodos. Este tiene un pin VCC y GND, un A0 que es la salida analógica y un D0 que es una salida digital regulable.



Este sensor va a ser utilizado para medir la humedad del sustrato del cultivo para indicar cuando debe ser regado el mismo.

Actuadores

Ventilador 60x60x10 DC12V

Este es un pequeño pero potente ventilador de 12v que genera un flujo de aire de 19.8CFM a una velocidad de 3800 RPM



Usaremos dos de estos ventiladores como extractores de la carpeta y uno como ventilador interno.

Moto bomba

Esta pequeña moto bomba tiene un rendimiento bastante importante. Alimentada con 3V en una hora puede entregar 80 litros y a 4.5V 100 litros.

Esta será utilizada para bombear el agua del depósito a la planta y poder regar el sustrato.



Model	Voltage Scope (DC)	Current (A)	Power (W)	Max Water Head (M)	Max Flow Rate (L/H)	Starting Voltage	waterproofing grade
JT-DC3L-3	3V	0.12	0.36	0.35	80	1	IP68
JT-DC3L-4.5	4.5V	0.18	0.91	0.55	100	1	IP68

LED WS2812 16bit anillo

Neopixel circular WS2812 tiene un encapsulado 5050 (5mm x 5mm) de 4 patas que incluye 3 LEDs superbrillantes (Rojo, Verde y Azul) junto a un circuito controlador (WS2812) responsable de recibir los datos del LED anterior y enviarlos al LED siguiente.

Mediante esta forma de transmisión en serie se consigue que todos los LEDs se controlen desde una sola línea de datos.



Cuando un led recibe un flujo de bytes, guarda los 3 últimos y transmite el resto al siguiente led, finalmente la señal "Reset" indica a los leds que muestren el valor almacenado.

Cuatro de estos anillos serán los encargados de iluminar el cultivo para su fotosíntesis.

Fuente

FUENTE SWITCHING 12V 50W

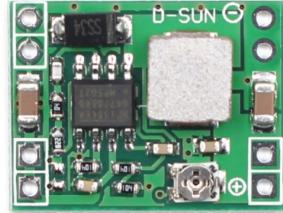
Fuente de voltaje de 12 voltios de 50W, que será utilizada para alimentar la carpeta.



Step down

Módulo conversor dc - dc 3A tipo buck. Acepta un voltaje de entrada de 4.5 - 28V y permite reducirlo hasta un mínimo de 0.8V. El conversor es controlador lo que quiere decir que a pequeños cambios de voltaje en la fuente de entrada el voltaje de salida se mantiene constante

Este modulo se utilizará para reducir el voltaje de la fuente de 12v a 5v para alimentar los microcontroladores, luces y algunos sensores.



Relay 4 canales

Módulo relay optoacoplado de 4 canales con bobinas de 5V.

Este modulo relay será usado para encender y apagar los extractores/ ventiladores.



Software del sistema

Microcontroladores

A continuación se detalla los códigos usados en los microcontroladores

Esp 32

Para la programación de esta placa partimos de un código de ejemplo brindado por los tutores que reporta una telemetría a la nube. Este código estaba diseñado para una placa de desarrollo ESP8266 por lo que tuvimos que adaptarlo.

Resumidamente este código ejecuta una conexión a la red WiFi para luego conectarse con thingsboard. Luego de esto toma las medidas de los sensores y genera el mensaje que envía a thingsboard.

Función setup wifi

Esta función es utilizada para la conexión por wifi. La misma utiliza una biblioteca llamada WiFi.h que nos facilita la conexión.

```

// Inicializar la conexión WiFi
void setup_wifi() {

    delay(10);
    Serial.println();
    Serial.print("Conectando a: ");
    Serial.println(ssid);

    WiFi.mode(WIFI_STA); // Declarar la ESP como STATION
    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    randomSeed(micros());

    Serial.println("");
    Serial.println("Conectado!");
    Serial.print("Dirección IP asignada: ");
    Serial.println(WiFi.localIP());
}

}

```

La parte del código que realmente conecta la placa a wifi es "WiFi.begin(ssid, password);", el resto es para la interfaz.

Conexión por MQTT a thingsboard

Para esta conexión se utiliza la librería PubSubClient.h que nos resuelve la misma de la siguiente manera

```

// Host de ThingsBoard
const char* mqtt_server = "demo.thingsboard.io";
const int mqtt_port = 1883;

client.setServer(mqtt_server, mqtt_port); // Establecer los datos para la conexión MQTT

```

Y luego utiliza la función reconnect

```

// Establecer y mantener la conexión con el servidor MQTT (En este caso de ThingsBoard)
void reconnect() {
    // Bucle hasta lograr la conexión
    while (!client.connected()) {
        Serial.print("Intentando conectar MQTT...");
        if (client.connect("ESP8266", token, token)) { //Nombre del Device y Token para conectarse
            Serial.println("Conectado!");

            // Una vez conectado, suscribirse al tópico para recibir solicitudes RPC
            client.subscribe("v1/devices/me/rpc/request/+");

        } else {

            Serial.print("Error, rc = ");
            Serial.print(client.state());
            Serial.println("Reintentar en 5 segundos...");
            // Esperar 5 segundos antes de reintentar
            delay(5000);

        }
    }
}

```

Con esto aseguramos la conexión y el mantenimiento de la misma.

Lectura de sensores

```
unsigned long now = millis();
if (now - lastMsg > msgPeriod) {
    lastMsg = now;

    temperature = dht.readTemperature(); // Leer la temperatura
    humidity = dht.readHumidity(); // Leer la humedad
    humedad_terrestre1 = analogRead(hterrestre);

    humedad_terrestre = map(humedad_terrestre1, 4095, 300, 0, 100);

    valorTanque = analogRead(pinTanque);

    Serial.println(valorTanque);
    if (valorTanque > valorReserva) {
        estadoTanque = false;
    } else {
        estadoTanque = true;
    }
}
```

La lectura de sensores se realiza cada 2 segundos. Para obtener ese intervalo se utilizan las 3 primeras líneas, luego se miden los valores y se mapea el valor de humedad terrestre para convertirlo en porcentaje. También el valor del tanque se convierte en true o false.

Envío de telemetría

```
unsigned long now2 = millis();
if (now2 - lastMsg2 > msgPeriod2) { //!isnan(temperature) && !isnan(humidity) && !isnan(humedad_terrestre)
    lastMsg2 = now2;
    if (true) {
        // Publicar los datos en el tópico de telemetría para que el servidor los reciba
        DynamicJsonDocument resp(256);
        resp["temperatura"] = temperature;
        resp["humedad"] = humidity;
        resp["humedad_terrestre"] = humedad_terrestre;

        char buffer[256];
        serializeJson(resp, buffer);
        client.publish("v1/devices/me/telemetry", buffer); // Publica el mensaje de telemetría

        Serial.print("Publicar mensaje [telemetry]: ");
        Serial.println(buffer);
    } else {
        Serial.print("Publicar mensaje [telemetry]: ");
        Serial.println("Failed to read from DHT sensor!");
    }
} else {
    if (true) {
        DynamicJsonDocument resp(256);
        resp["temperatura"] = temperature;
        resp["humedad"] = humidity;

        char buffer[256];
        serializeJson(resp, buffer);
        client.publish("v1/devices/me/telemetry", buffer); // Publica el mensaje de telemetría

        Serial.print("Publicar mensaje [telemetry]: ");
        Serial.println(buffer);
    } else {
        Serial.print("Publicar mensaje [telemetry]: ");
        Serial.println("Failed to read from DHT sensor!");
    }
}

}
}
```

El envío de telemetría se hace también cada dos segundos para la humedad y temperatura y cada 10 segundos para la humedad terrestre. Esto se hace viendo la diferencia de tiempo al igual que la lectura de sensores.

Si la diferencia de tiempo es mayor a 10 segundos se envían las 3 lecturas, en caso contrario se envían solamente la humedad y temperatura.

Este código se repite cada 2 segundos dado a que se encuentra dentro del if de la lectura de sensores.

El envío de datos se hace a través de una publicación de telemetría. Esta publicación está compuesta de un Jason que contiene el nombre de la variable y el valor.

```
DynamicJsonDocument resp(256);
resp["temperatura"] = temperature;
resp["humedad"] = humidity;
resp["humedad_terrestre"] = humedad_terrestre;

char buffer[256];
serializeJson(resp, buffer);
client.publish("v1/devices/me/telemetry", buffer); // Publica el mensaje de telemetría
```

Reporte de atributo

El valor del tanque se comunica modificando un atributo llamado tanque vacío. Este se modifica con un true o false que se seteo anteriormente.

```
//-----REPORTE DE ATRIBUTOS-----
DynamicJsonDocument resp(256);
resp["TanqueVacio"] = estadoTanque;
char buffer[256];
serializeJson(resp, buffer);
client.publish("v1/devices/me/attributes", buffer);
Serial.print("Publish message [attribute]: ");
Serial.println(buffer);
```

Esta sección de código se encuentra dentro del if de la lectura de sensores, por lo que se repite cada dos segundos, haciendo que se actualice el atributo cada dos segundos.

Esp 8266

Para la programación de esta placa también partimos del código de ejemplo brindado por los tutores que reporta una telemetría a la nube.

Resumidamente este código ejecuta una conexión a la red WiFi para luego conectarse con thingsboard. Luego de esto escucha los mensajes de la nube y acciona los actuadores indicados.

Función setup wifi

Esta función es prácticamente igual a la función setup wifi de la esp32. Utiliza también una biblioteca pero llamada ESP8266WiFi.h que también nos facilita la conexión.

```

void setup_wifi() {

    delay(10);
    // We start by connecting to a WiFi network
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(ssid);

    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    randomSeed(micros());

    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
}

```

Conexión por MQTT a thingsboard

Al igual que en la placa anterior, para esta conexión se utiliza la librería PubSubClient.h que nos resuelve la misma de la siguiente manera

```

const char* mqtt_server = "demo.thingsboard.io";
const char* token = "11QHcVqS0n3q8Ov022k1";

client.setServer(mqtt_server, 1883);

```

Y luego utiliza la misma función reconnect

```

//-----FUNCION RECONNECT-----
void reconnect() {
    // Loop until we're reconnected
    while (!client.connected()) {
        Serial.print("Attempting MQTT connection...");
        // Attempt to connect - client.connect(DEVICE_ID, TOKEN, TOKEN)
        if (client.connect("NODEMCU Nuevo", token, token)) {
            Serial.println("connected");
            // Once connected, subscribe to rpc topic
            client.subscribe("v1/devices/me/rpc/request/+");
        } else {
            Serial.print("failed, rc=");
            Serial.print(client.state());
            Serial.println(" try again in 5 seconds");
            // Wait 5 seconds before retrying
            delay(5000);
        }
    }
//-----

```

Con esto aseguramos la conexión y el mantenimiento de la misma.

Conexión por MQTT a thingsboard

El callback a continuación se encarga de administrar todos los mensajes que llegan a nuestro sistema y activar los actuadores correspondientes. Los mensajes llegan con un “topic” y un “payload” que contienen toda su información.

```
void callback(char* topic, byte* payload, unsigned int length) {  
  
    //log to console  
    Serial.print("Message arrived [");  
    Serial.print(topic);  
    Serial.print("] ");  
    for (int i = 0; i < length; i++) {  
        Serial.print((char)payload[i]);  
    }  
    Serial.println();  
  
    //convert topic to string to parse  
    String _topic = String(topic);  
  
    if (_topic.startsWith("v1/devices/me/rpc/request/")) {  
        //We are in a request, check request number  
        String _number = _topic.substring(26);  
  
        //Read JSON Object  
        deserializeJson(incoming_message, payload);  
        String metodo = incoming_message["method"];  
  
        //-----CHECK STATUS-----  
        if (metodo == "checkStatus") { //Check device status. Expects a response to the same topic number with status=true.  
  
            char outTopic[128];  
            ("v1/devices/me/rpc/response/" + _number).toCharArray(outTopic, 128);  
        }  
    }  
}
```

Esta función toma los mensajes que tienen a request como tópico. Luego toma el nombre del método y busca en una serie de if el mismo método.

Si el método es “setExOkStatus” ingresa en el siguiente if y cambia el estado del extractor Ok.

```
//-----SET ExOk STATUS-----  
if (metodo == "setExOkStatus") { //Set led status and update attribute value via MQTT  
  
    boolean estado = incoming_message["params"];  
  
    if (!estado) {  
        digitalWrite(FAN1, LOW); //turn on led  
    } else {  
        digitalWrite(FAN1, HIGH); //turn off led  
    }  
  
    //Attribute update  
    DynamicJsonDocument resp(256);  
    resp["EXOK"] = estado;  
    char buffer[256];  
    serializeJson(resp, buffer);  
    client.publish("v1/devices/me/attributes", buffer);  
    Serial.print("Publish message [attribute]: ");  
    Serial.println(buffer);  
}
```

Luego de cambiar el estado del extractor actualiza el atributo del extractor para informar a la nube el cambio de estado.

Lo mismo sucede para el otro extractor, el ventilador, las luces y la bomba de agua.

ThingsBoard

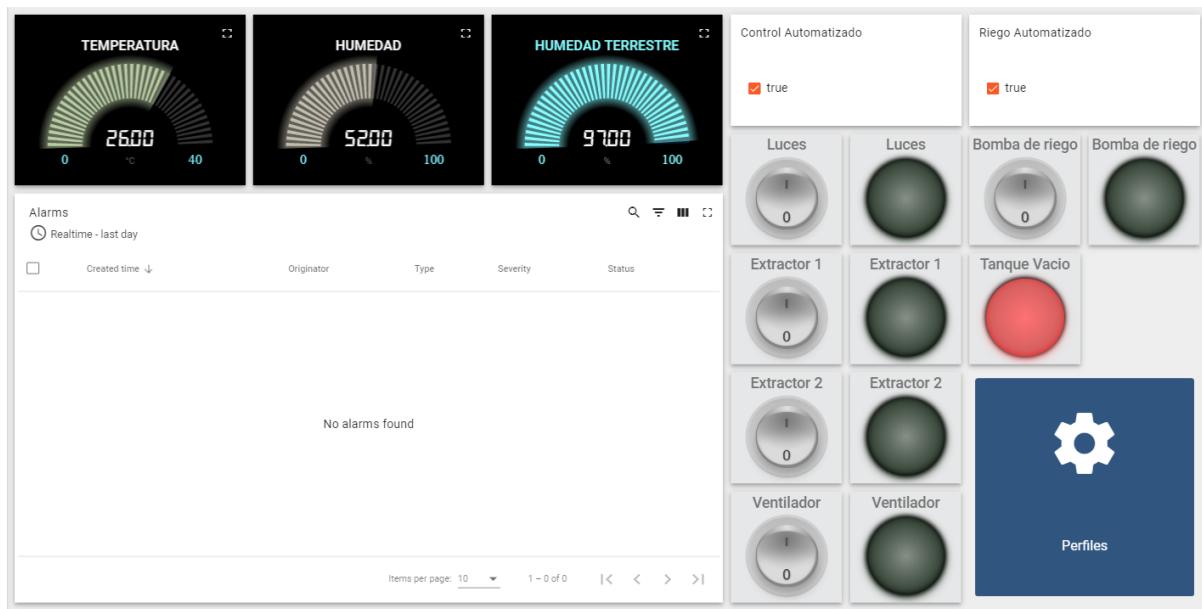
ThingsBoard es una plataforma de IoT de código abierto para monitoreo, procesamiento de datos, visualización de datos junto con administración de dispositivos. Es compatible con los protocolos IoT estándar de la industria: MQTT, CoAP y HTTP. ThingsBoard combina escalabilidad, tolerancia a fallas y rendimiento para capturar los datos del dispositivo para su procesamiento y monitoreo. Proporciona su servidor de puerta de enlace que puede comunicarse con los dispositivos conectados.

Dashboard

En el dashboard se programaron medidores digitales para mostrar la temperatura, humedad y humedad terrestre, lo cuales cambian de color segun su estado. Estos medidores obtienen los datos de los reportes de telemetría del dispositivo de sensores.

También se programaron llaves que prenden o apagan, por ejemplo, las luces. Las llaves leen su estado del atributo al cual están ligadas y al ser accionadas envían un método RPC al dispositivo para cambiar ese atributo.

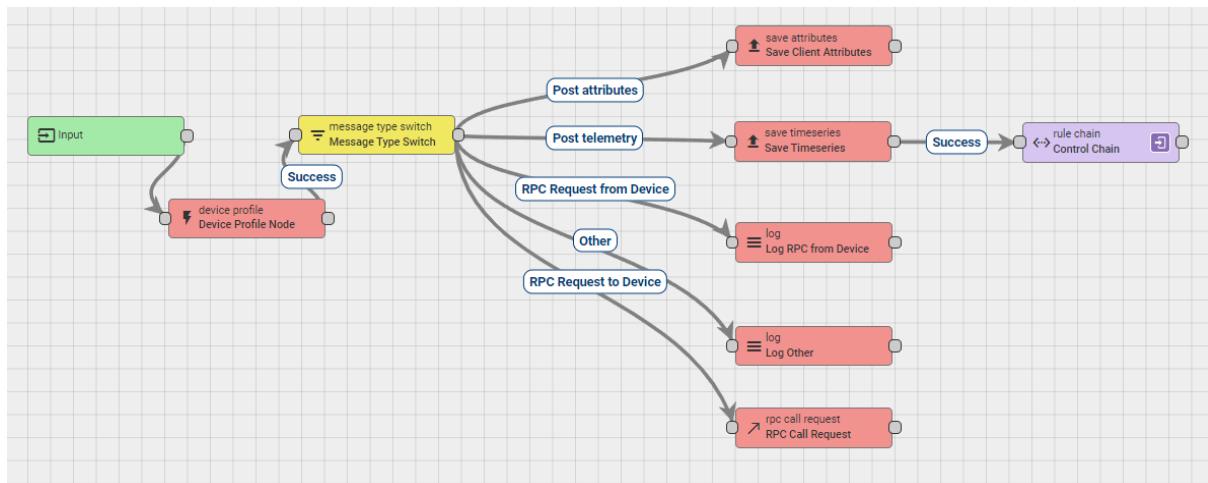
Por otro lado tenemos las luces las cuales tienen un comportamiento similar a las llaves ya que leen su estado del atributo al cual están ligadas. Se programaron dos checkbox los cuales están completamente ligados a sus respectivos atributos de servidor, los cuales se usan en la rule chain. y por último tenemos el cuadro que nos lleva al dashboard de perfiles.



En este dashboard se setean y se muestran los atributos de los perfiles. Estos atributos se guardan independientemente como atributos de servidor y en la rulechain se utilizan para setear los umbrales con los cuales se va a trabajar acorde al número de perfil seleccionado.

Perfil 1 TemperaturaMax 28	Perfil 2 TemperaturaMax 30	Perfil 3 TemperaturaMax 25	Perfil 4 TemperaturaMax 24	Perfil 5 TemperaturaMax 30	Perfil Actual Value * 1
TemperaturaMin 18	TemperaturaMin 15	TemperaturaMin 15	TemperaturaMin 22	TemperaturaMin 28	
HumedadMax 70	HumedadMax 60	HumedadMax 75	HumedadMax 99	HumedadMax 100	
HumedadMin 65	HumedadMin 35	HumedadMin 60	HumedadMin 80	HumedadMin 90	
HumedadTerrestreMax 80	HumedadTerrestreMax 80	HumedadTerrestreMax 80	HumedadTerrestreMax 40	HumedadTerrestreMax 90	
HumedadTerrestreMin 40	HumedadTerrestreMin 40	HumedadTerrestreMin 10	HumedadTerrestreMin 10	HumedadTerrestreMin 70	
HoraEncendido 22:00	HoraEncendido 22:00	HoraEncendido 22:00	HoraEncendido 22:00	HoraEncendido 0	
HoraApagado 23:16	HoraApagado 10:00	HoraApagado 16:00	HoraApagado 12:00	HoraApagado 0	
<input type="button" value="Undo"/>	<input type="button" value="Save"/>	<input type="button" value="Undo"/>	<input type="button" value="Save"/>	<input type="button" value="Undo"/>	<input type="button" value="Save"/>

Rule Chain



Como se puede observar en la imagen superior, se agregó el acceso a una nueva rule chain luego de el salvado de telemetrías que denominamos “Control Chain”. Desde aqui, cada vez que la placa reporta el estado de los sensores, el sistema calcula las acciones que debe

realizar para mantener estos estados dentro de los deseados.



Al entrar a esta rule chain, lo primero que se hace es concatenar en el metadata del mensaje todos los atributos de servidor. Los cuales corresponden a todos los perfiles. En el bloque siguiente se crea un nuevo metadata con los atributos correspondientes solo al perfil seleccionado. De aquí en adelante se trabaja solo con este perfil. Luego se divide en 3 caminos, control de riego, control de luz y control de temperatura y humedad.

El control de riego trabaja con el censo de la humedad terrestre. Cuando está baja del umbral inferior se prende la bomba de riego y al alcanzar el nivel superior se apaga.

El control de luz funciona a través de un cronograma por horarios. Para ello utilizamos el “ts” (timestamp) del mensaje. Este nos da la fecha y hora en la cual se reportó el mensaje. De esta manera podemos saber la hora actual del mensaje y prender o apagar las luces de ser necesario.

El control de temperatura y humedad activa y desactiva, 2 extractores y un ventilador.

Acorde al estado actual de los sensores de temperatura y humedad, reportados por la placa, se selecciona un estado de los actuadores. Para ello creamos una tabla, detallada más abajo, en donde según la combinación de temperatura y humedad actual se decide que es lo óptimo para mantener o llevar el ambiente al deseado. A su vez, este control tiene previsto un reporte de alarmas en caso de que las condiciones excedan la capacidad de control del sistema. Es decir, en casos donde la temperatura o humedad supere los umbrales, lo cual significa que la capacidad o potencia del sistema no pudo mantener estos márgenes en los parámetros programados, se le avise al usuario para que el mismo tome otras acciones.

Habiendo explicado la funcionalidad de los bloques lo que falta abarcar es como el sistema actúa sobre los propiamente dicho, actuadores. Ya que el mensaje, reporte de los sensores, se hace desde una placa y los actuadores están en otra. Es necesario usar los bloques llamados “change originator”. Los cual nos permite cambiar con qué dispositivo se está comunicando. Y de esta manera poder enviar un request RPC al dispositivo de los actuadores.

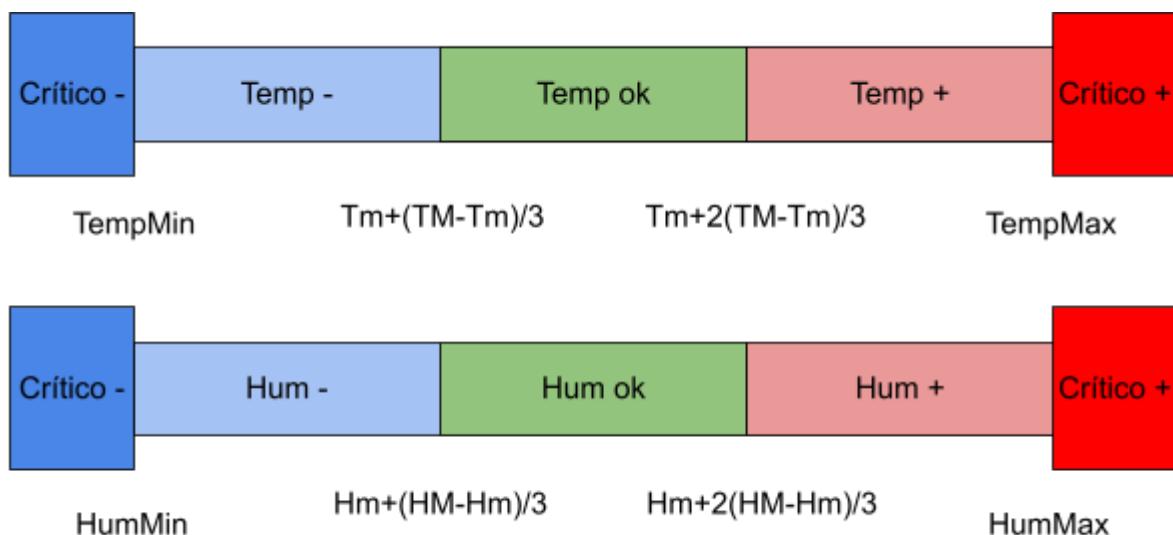
Prototipos

Sistema de control de temperatura y humedad

Este sistema es el encargado de regular la temperatura y humedad por medio de los extractores y el ventilador.

El mismo cuenta con un sensor de temperatura y humedad ambiente DHT11, dos extractores que llamaremos ExOk y Ex1 y un ventilador interno.

Con los valores de temperatura máxima y mínima y humedad máxima y mínima ingresados por el usuario se generan los siguientes intervalos.



Estos intervalos son usados para crear la siguiente tabla que contiene qué actuadores encender o apagar según los intervalos de temperatura y humedad.

	Tmin	T-	T ok	T+	T max
Hmin	-	-	-	-	ExOk + Ex1 + Vent
H-	-	-	-	ExOk	ExOk + Ex1 + Vent
Hok	-	-	ExOk	ExOk + Vent	ExOk + Ex1 + Vent
H+	-	ExOk	ExOk + Ex1	ExOk + Ex1 + Vent	ExOk + Ex1 + Vent
Hmax	-	ExOk + Ex1 + Vent			

Con esto buscamos mantener tanto la humedad como la temperatura en los rangos solicitados por el usuario.

En este sistema con estos actuadores existen claros riesgos. Estos riesgos son cuando la temperatura o la humedad se sale de los rangos marcados por el usuario. Esto se puede

deber a la temperatura o humedad ambiente fuera de la carpa, que puede no estar en el rango marcado por el usuario lo que generaría que sea imposible llevar las variables a dicho rango.

Este riesgo no existiría si se utilizara un humidificador/deshumidificador y un aire acondicionado ya que estos no tendrían problemas en regular las variables.

Sistema de control de riego

Este sistema es el encargado del riego del cultivo según los valores de humedad terrestre indicados por el usuario. El mismo está compuesto por un sensor de humedad terrestre y una moto bomba dentro de un depósito de agua.

Cuando la humedad terrestre llega al mínimo se activa la bomba de agua hasta que el valor de humedad terrestre llegue al máximo, en ese momento el sistema apaga la bomba de agua hasta que la humedad vuelva a llegar al mínimo.

El depósito de agua contiene un sensor que indica al usuario cuando hay poca agua.

El mayor riesgo de este sistema es que el sensor de humedad no se encuentre cerca del cultivo y por lo tanto indique valores de humedad para una zona alejada del mismo. Esto generaría que el cultivo no tenga la humedad correcta.

Otro riesgo es que el depósito se vacíe cuando el usuario no se encuentre cerca. Este se puede solucionar conectando este sistema al agua corriente y cambiando la bomba por una electroválvula o utilizando un sistema de llenado automático.

Sistema de control de Luz

Este sistema se encarga de generar el día o la noche artificial dentro de la carpa. Este sistema es vital para el cultivo ya que sin luz el cultivo no se desarrollará de la mejor manera.

El sistema está constituido simplemente por cuatro aros de luces que se encienden y se apagan según el cronograma en el dashboard.

El riesgo que tiene este sistema es que la placa se desconecte del wifi provocando que la luz quede encendida o apagada y el cultivo no reciba la cantidad de energía lumínica necesaria.

Plan de trabajo:

1. Conexión del microcontrolador a la nube. De esta manera resolver la comunicación, que es crucial para poder realizar las siguientes partes.
 - 1.1. Conectar el microprocesador con la nube.
 - 1.2. Enviar un random hacia la nube y chequear correcta recepción.
 - 1.3. Mostrar información en el dashboard.
 - 1.4. Crear botón en dashboard.
 - 1.5. Enviar al microcontrolador la señal del botón.
 - 1.6. Con esa señal prender una luz.

Con esto habremos adquirido los conocimientos básicos necesarios sobre MQTT

2. Instalación del sistema de control de temperatura y humedad. Esto incluye conectar un sensor a un microprocesador, filtrar señal en el microprocesador, subir la información a la nube, procesar dicha información, enviar acción desde la nube al microprocesador, procesar acción y ejecutarla.
 - 2.1. Conectar el sensor al microcontrolador y chequear su funcionamiento.
 - 2.2. Procesar/filtrar lectura del sensor en el microcontrolador y enviarla a la nube.
 - 2.3. Mostrar en dashboard la lectura recibida.
 - 2.4. Configurar acción de acuerdo a la lectura del sensor.
 - 2.5. Conectar ventilador y extractores al microcontrolador y chequear su funcionamiento.
 - 2.6. Enviar acción al microcontrolador y ejecutarla.
3. Instalación del sistema de riego. La cual consta de 2 partes: la instalación del kit de riego y la instalación de un sistema de control del nivel del tanque de riego (2.2).
 - 3.1. Instalación del sensor de humedad terrestre y bomba de riego.
 - 3.1.1. Conectar sensor de humedad y bomba de riego al microcontrolador y chequear su funcionamiento.
 - 3.1.2. Enviar información del lector de humedad terrestre a la nube y mostrarla en dashboard.
 - 3.1.3. Procesar lectura del sensor en la nube y enviar acción al microcontrolador acorde a lo procesado.
 - 3.2. Instalación del sistema de control del nivel de líquido en el tanque de riego
 - 3.2.1. Diseñar un tanque acorde para que la contenga la bomba de riego y el sensor de nivel.
 - 3.2.2. Conectar sensor de nivel al microcontrolador y verificar su funcionamiento.
 - 3.2.3. Enviar lectura del sensor a la nube
 - 3.2.4. Mostrar lectura recibida en el dashboard
 - 3.2.5. Crear alarma en caso de que el nivel sea menor al necesario.
4. Instalar sistema de control de luz.
 - 4.1. Conectar luces al microcontrolador y verificar funcionamiento.

- 4.2. Enviar acción de encender las luces desde la nube al microcontrolador y procesarla.
5. Armado de maqueta y montaje del sistema. Se deba construir la maqueta y diseño donde se montaran los sensores y actuadores.
6. Programar en la nube el cronograma el cual se utilizará para el control de luces y el riego. Para el control de luces el usuario deberá ingresar el horario en el cual las luces deben permanecer encendidas. En cuanto al riego sería necesario que el usuario indique cada cuánto tiempo desea regar junto con los valores límites de humedad terrestre.
7. Terminar detalles estéticos de la maqueta y del dashboard previamente programado

Pruebas

Comunicación placa nube

Para realizar la prueba de comunicación tomamos el código de ejemplo brindado por los docentes el cual contenía un código de reporte de telemetría. Al mismo tiempo hicimos un usuario en things board y creamos un nuevo dispositivo el cual asignamos a nuestra esp8266.

Con el token brindado por el dispositivo creado en thingsboard pudimos activar la comunicación entre ambas partes.

```
const char* mqtt_server = "demo.thingsboard.io";
const char* token = "11QHcVqS0n3q8Ovo22k1";
```

Luego de conectado programamos en la esp8266 un código de prueba que reporta una telemetría.

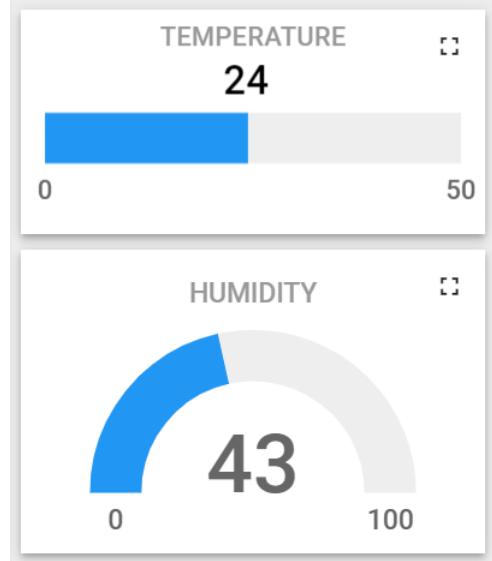
```
//-----PUBLICACION TELEMETRIA-----
if (!isnan(temp) && !isnan(hum)) {
    DynamicJsonDocument resp(256);
    resp["humedad"] = hum;
    resp["temperatura"] = temp;
    char buffer[256];
    serializeJson(resp, buffer);
    client.publish("v1/devices/me/telemetry", buffer);

    Serial.print("Publish message [telemetry]: ");
    Serial.println(buffer);
}

//-----
```

Con esto pudimos ver y corroborar la correcta recepción de la telemetría en thingsboard.

Con esto creamos unos widgets en thingsboard para mostrar la información en el dashboard



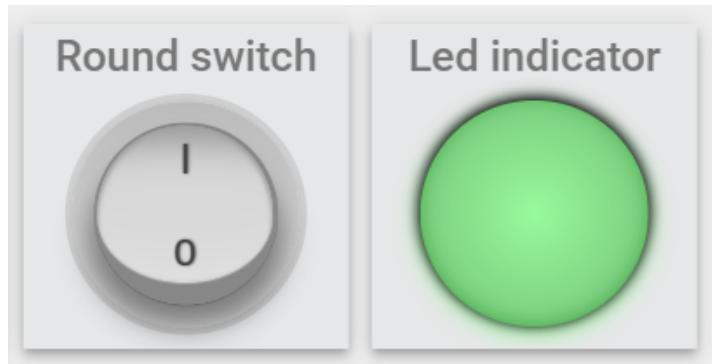
Luego de esto continuamos con la comunicación inversa, osea de thingsboard a la placa, utilizando un request RPC para encender y apagar el led integrado de la placa esp8266.

Utilizamos el siguiente fragmento del código de ejemplo. Este se encuentra dentro de una función callback que se llama cada vez que recibe un mensaje e identifica el tópico para procesar el mensaje. Luego se extrae, para el caso de request, el nombre del método y se ejecuta el siguiente fragmento si el mismo coincide.

```
if (metodo=="setLedStatus") {  
  
    boolean estado = incoming_message["params"]; // Leer los parámetros del método  
  
    if (estado) {  
        digitalWrite(LED_BUILTIN, LOW); // Encender LED  
        Serial.println("Encender LED");  
    } else {  
        digitalWrite(LED_BUILTIN, HIGH); // Apagar LED  
        Serial.println("Apagar LED");  
    }  
  
    // Actualizar el atributo relacionado  
    DynamicJsonDocument resp(256);  
    resp["estado"] = !digitalRead(LED_BUILTIN);  
    char buffer[256];  
    serializeJson(resp, buffer);  
    client.publish("v1/devices/me/attributes", buffer); //Topico para actualizar atributos  
    Serial.print("Publish message [attribute]: ");  
    Serial.println(buffer);  
}  
}
```

Este código cambia el estado del led, si se encuentra encendido lo apaga y si se encuentra apagado lo enciende. Luego actualiza un atributo, en este caso llamado estado, para que la nube sepa el nuevo estado del led.

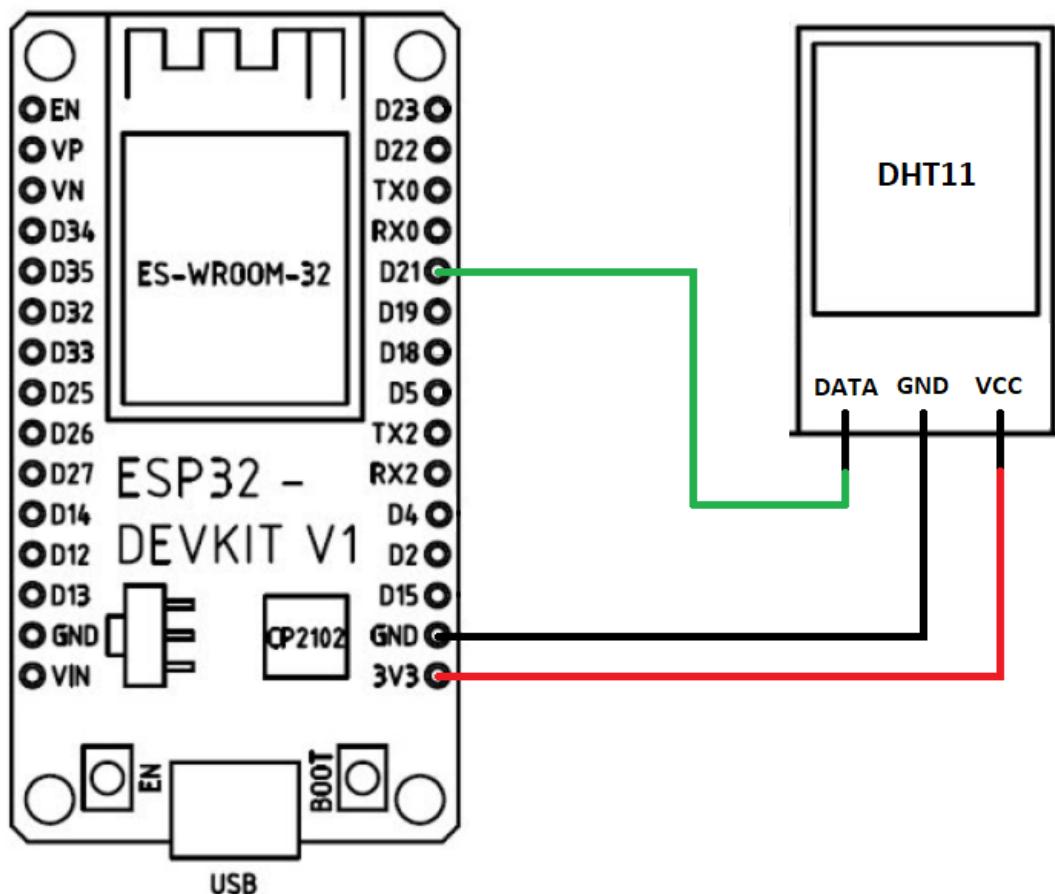
En el dashboard creamos dos widgets para ver y controlar el estado del led. El indicador refleja el estado del atributo mientras que el switch manda un rpc con el método setLedStatus



Con esto tenemos lo básico de la comunicación placa-nube necesaria para crear los sub-sistemas del proyecto.

Censo de temperatura y humedad

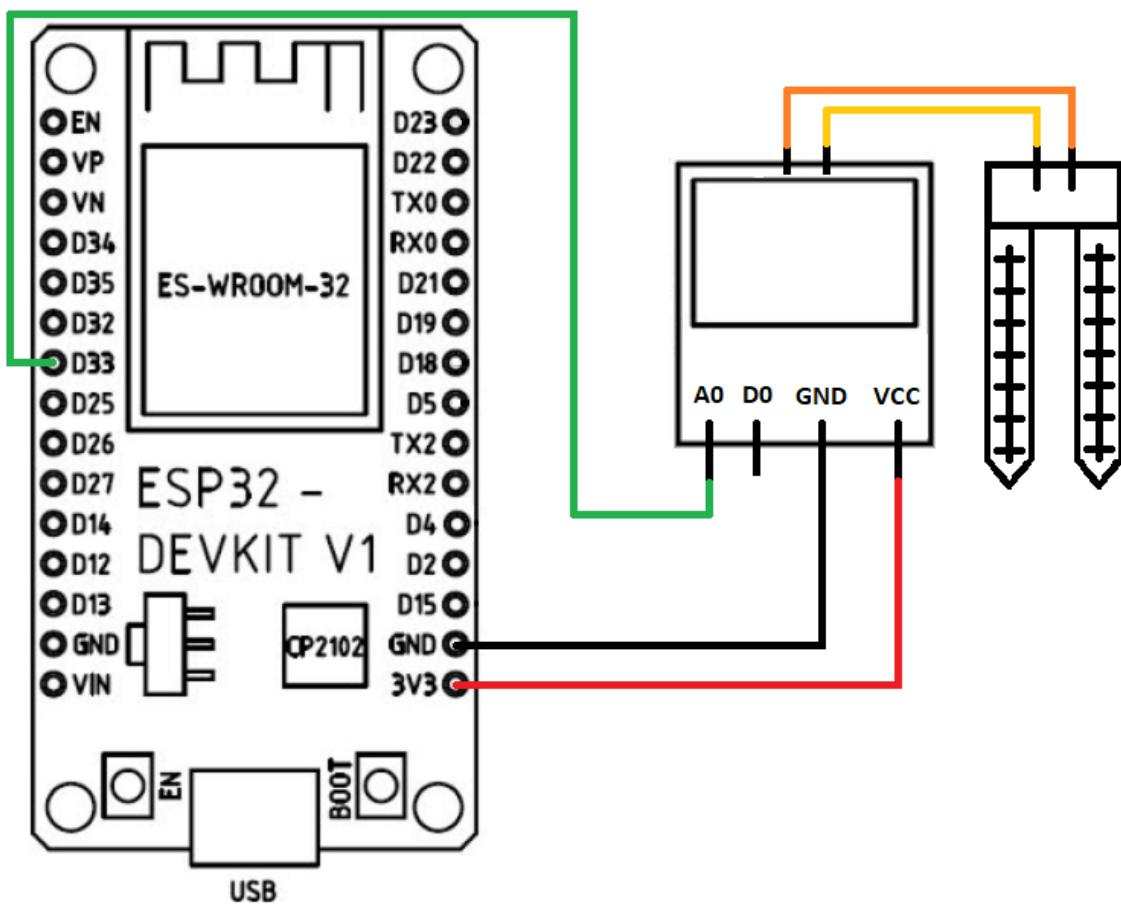
Para comenzar con este censo es necesario incluir un módulo sensor de humedad y temperatura, más específicamente el DHT11. Este lo hemos conectado con el microprocesador de la siguiente manera.



Una vez hecho esto se instaló la librería para controlar el mismo y se procedió a mostrar en consola los reportes del sensor para chequear la correcta lectura de las variables. Ya teniendo una lectura correcta, se ligó esto con el código de la prueba anterior y en lugar de reportar en la telemetría un número random como temperatura y humedad, se reportó el censo. Así obtuvimos en el dashboard de thingsboard nuestros primeros dos widgets que nos muestran el estado del sensor.

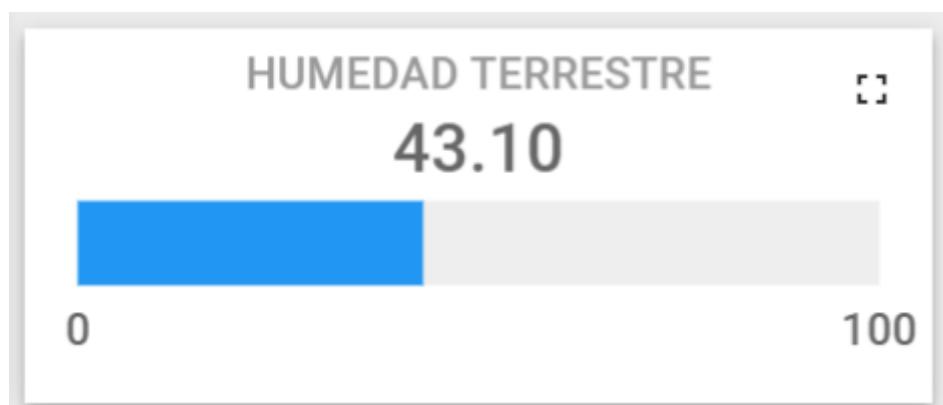
Censo de humedad terrestre

Nuevamente para esto lo primero es incluir un sensor higrómetro que mide la humedad del suelo por medio de la resistividad generada entre sus electrodos. Este lo conectamos a la ESP 32 de la siguiente manera



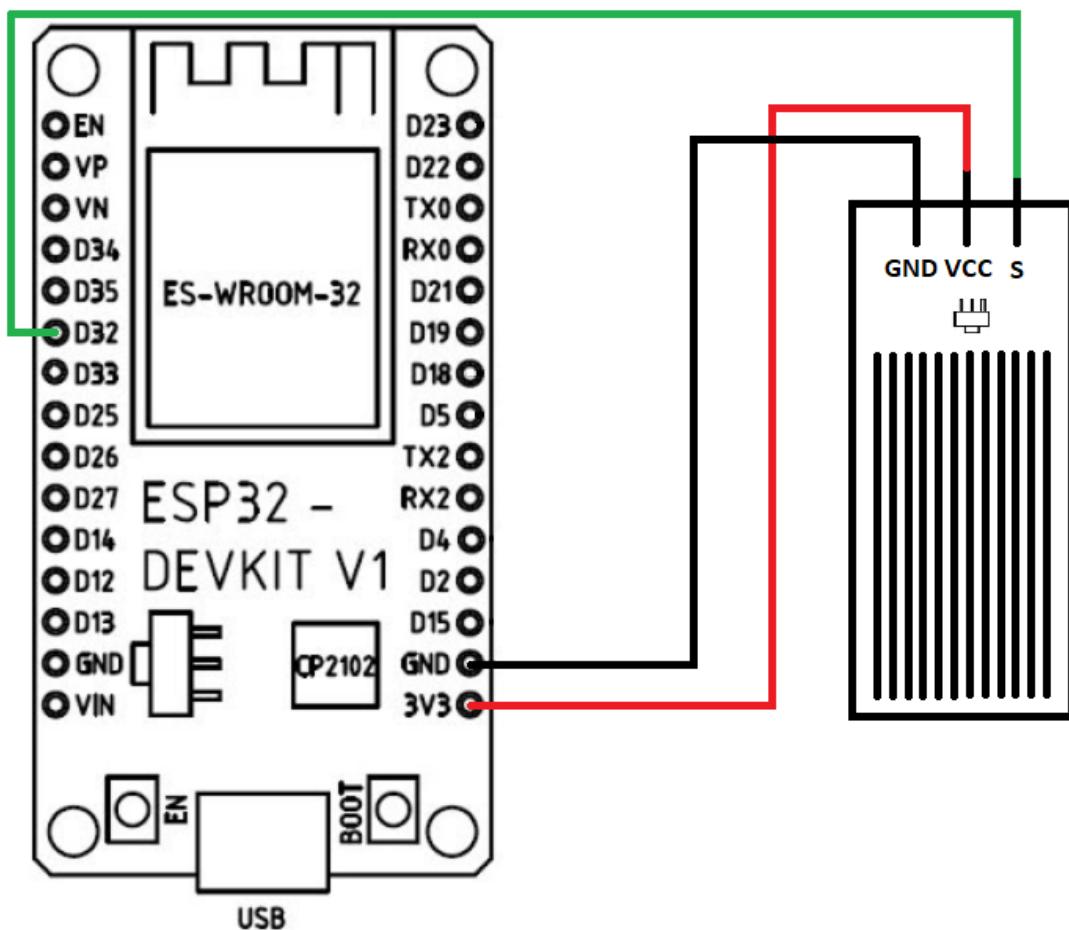
Una vez conectado leímos los datos del sensor y los imprimimos en consola. Al imprimirla nos dimos cuenta que si el sustrato está muy húmedo el valor que muestra es pequeño, y si está totalmente seco el valor que muestra es 4095 por lo que convertimos este rango en un rango de 0 a 100 para tratarlo como porcentaje. Esto lo logramos con una función de arduino llamada map().

Luego, utilizando el código de envío de telemetría que utilizamos en la prueba de comunicación placa nube, enviamos el dato leído a thingsboard y con un widget lo mostramos en el dashboard.



Censo de agua del tanque

Para este censo utilizamos el módulo sensor de agua que es un módulo diseñado para medir el nivel de un líquido en un tanque. Conectamos este sensor a la ESP32 de la siguiente manera



Luego comenzamos a mostrar las lecturas en consola pero nos dimos cuenta que el largo de la pista era muy corto por lo que no íbamos a poder saber el nivel del tanque en todo momento. Optamos por utilizarlo como un detector de tanque vacío, osea que envíe un true si el tanque contiene agua y un false si contiene muy poca agua.

Para realizar esto definimos un valor límite el cual, por medio de un if, indica false si el valor del sensor está por encima de este o true si el actual está por debajo.

```

valorTanque = analogRead(pinTanque);
Serial.println(valorTanque);
if (valorTanque > valorReserva) {
    estadoTanque = false;
} else {
    estadoTanque = true;
}

```

Ahora tenemos un detector de tanque vacío el cual, por medio de un atributo, vamos a enviar a thingsboard.

```

//-----REPORTE DE ATRIBUTOS-----
DynamicJsonDocument resp(256);
resp["TanqueVacio"] = estadoTanque;
char buffer[256];
serializeJson(resp, buffer);
client.publish("v1/devices/me/attributes", buffer);
Serial.print("Publish message [attribute]: ");
Serial.println(buffer);

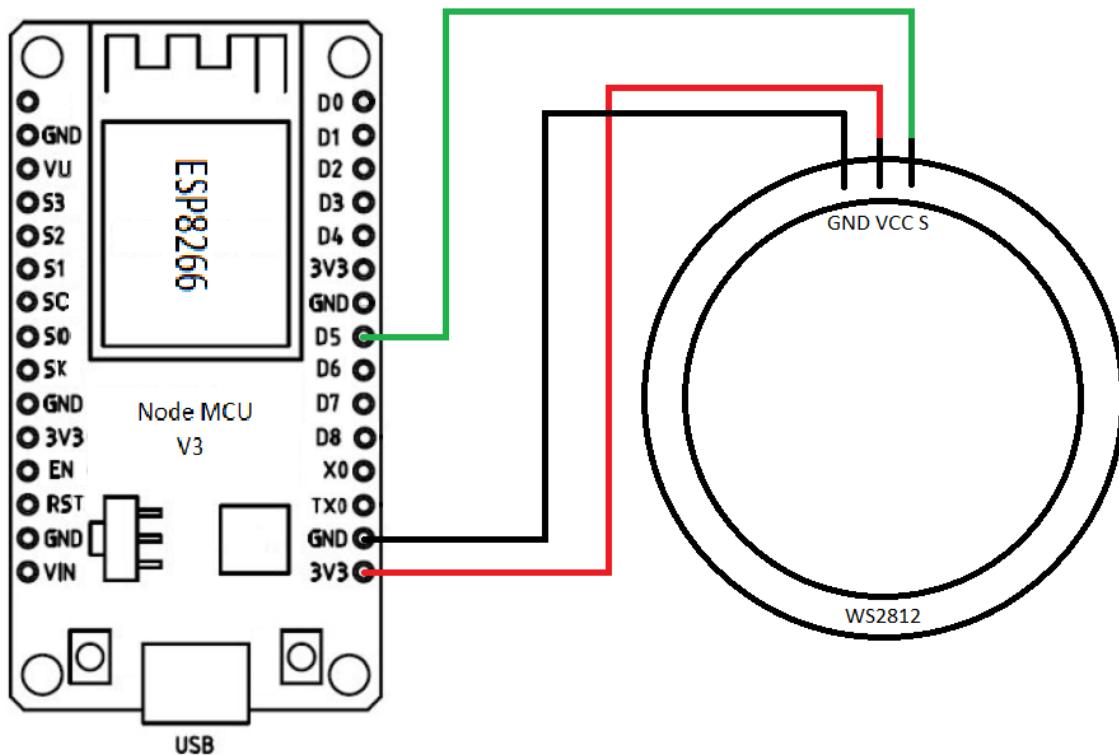
```

En el dashboard de thingsboard colocamos un indicador led rojo que anuncia si el tanque está vacío.



Luces led

Para el sistema de luces utilizamos cuatro aros led WS2812 que permite controlar el color rgb de cada led por medio de un tren de bits. En nuestro caso vamos a utilizar, en primera instancia, luz de color blanco. Comenzamos conectando un solo aro led a nuestra ESP8266 de la siguiente manera.



Luego instalamos una librería llamada “Adafruit_NeoPixel” que nos facilita con el arreglo de bits.

Para mantener un color blanco en todos los led se debe indicar el color RGB para cada led por medio de un for, como se ve en el siguiente código.

```

for(int i=0; i<NUMPIXELS; i++) { // For each pixel...
    pixels.setPixelColor(i, pixels.Color(255,255, 100));
}
pixels.show();

```

Para la comunicación modificamos la función callback, que ya contenía un método request llamado setLedStatus, insertando en encender el led el fragmento de código anterior y en apagar led utilizamos pixels.clear() que deja en 0 todos los led.

```

-----SET LED STATUS-----
if (metodo == "setLedStatus") { //Set led status and update attribute value via MQTT

    boolean estado = incoming_message["params"];

    if (estado) {
        for(int i=0; i<NUMPIXELS; i++) { // For each pixel...
            pixels.setPixelColor(i, pixels.Color(255,255, 100));
        }
        pixels.show();
        //digitalWrite(LED1, HIGH); //turn on led
    } else {
        pixels.clear();
        pixels.show();

        // digitalWrite(LED1, LOW); //turn off led
    }
}

```

En el dashboard colocamos el interruptor y el indicador led de la misma manera que en la prueba de comunicación placa nube.

Rule Chain

Encendido de ventiladores

La primer prueba a realizar en la rule chain fue el envío de un request RPC cuando un sensor supere cierto valor. Ya que el sensor de temperatura y humedad fue el primero que instalamos, decidimos usar este para la prueba. Entonces, cuando la placa reporta a la cola de telemetría el estado actual del sensor, la rulechain lo guardará y luego abrirá una nueva rule chain que más adelante se convertirá en la “Control chain”.

Dentro de esta rule chain hicimos una primer prueba simple. Incluimos un bloque script que cuando la temperatura supere un número X, mande un request RPC que cambie el estado de un atributo. De la misma manera que hicimos en la prueba de conectividad con la llave y el led indicador pero esta vez en lugar de prender un led en la placa, asociamos ese atributo a una salida digital donde conectamos un ventilador.

Logrado esto, subimos el nivel de complejidad queriendo mandar el mismo request pero esta vez al otro microcontrolador. Haciendo esto nos encontramos que para poder hacer una acción sobre otra placa diferente a la que inicio el mensaje, era necesario primero asociar los dispositivos a través de un “asset” dentro de thingsboard y utilizar esto en el bloque “change originator” para cambiar el curso del mensaje hacia la otra tarjeta. Después del “change originator” se agrega el bloque de send request RPC, igual que con un solo dispositivo.

Ahora ya tenemos, que al reportar el estado de un sensor desde un dispositivo, el mensaje, después de ser salvado, pase por un filtro y con el resultado de este poder accionar sobre otro dispositivo.

Alarms

La siguiente prueba a realizar son las alarmas. Aprovechando el filtro de la prueba anterior, utilizamos ese resultado para que en caso de ser positivo cree una alarma y en caso de ser

negativo borre la misma alarma. Para ello se utilizaron los bloques “create alarm” y “clear alarm” para crear y borrar la alarma. Y en el dashboard se agregó un “alarm widget” para poder visualizar la misma.

Control de riego

Para este control la prueba fue muy simple ya habiendo realizado las pruebas anteriores. Solamente es necesario cambiar el sensor que se quiere leer y agregar un script de filtrado que tenga dos salidas. Luego es lo mismo que ya se probó. Se conecta un filtro para cada salida, uno el cual envía un request RPC para prender la bomba de riego y otro para apagarla. Tampoco el hecho de cambiar de actuador genera una dificultad ya que es simplemente prender una salida digital, que ya se probó con éxito previamente.

Control de luz

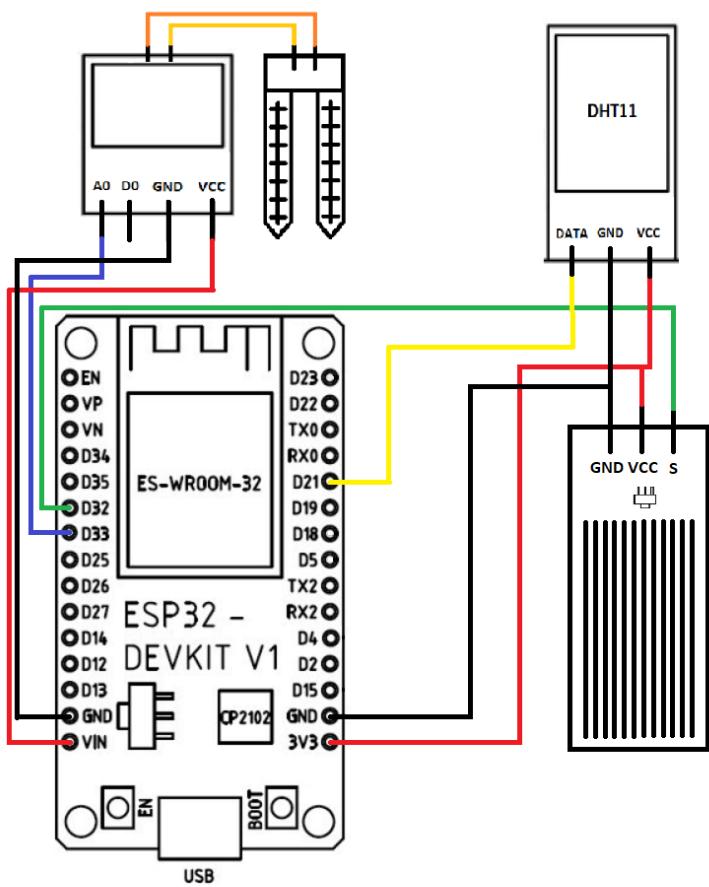
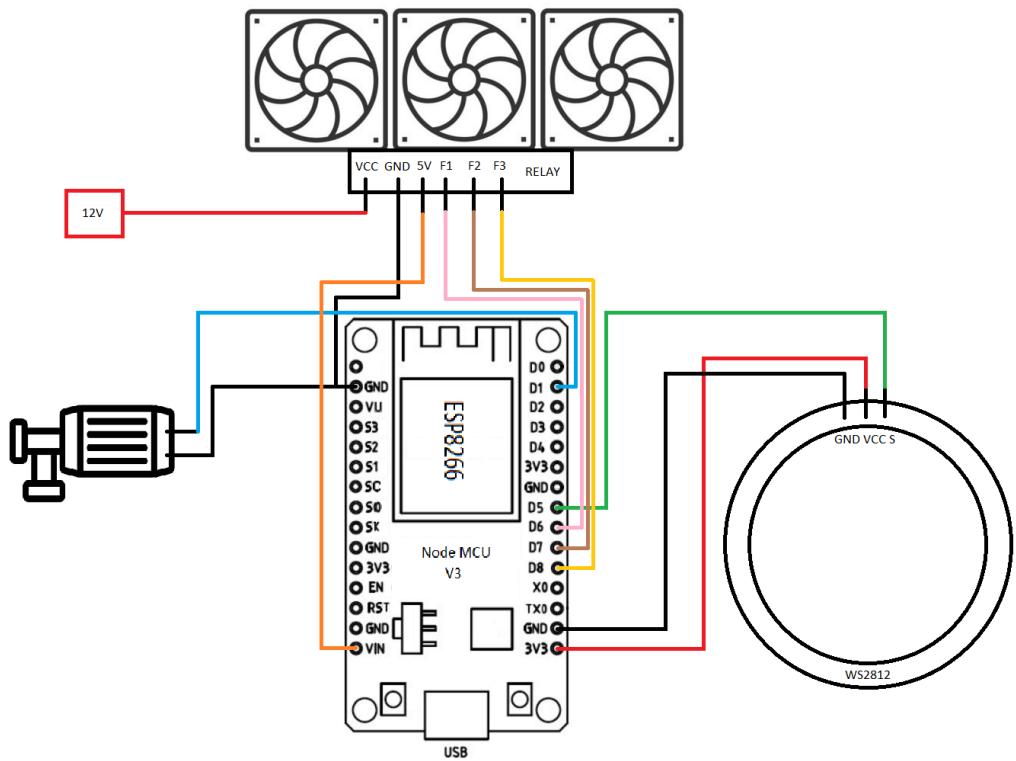
La prueba adicional que se hizo en este caso fue la correcta utilización de timestamp para saber el momento en el que se envió ese mensaje y de esta manera saber si se deben encender o apagar las luces o no hacer nada. Otra vez recurrimos a un bloque script en el cual se utilizó Date de javascript, el cual nos convierte el “ts” al un formato de fecha y hora. ya teniendo la hora y minutos del mensaje se filtraron para que a la hora y minutos, con un delta de 2 minutos, se encienda o se apague la luz. El resto del control se probó previamente. Con la salida del filtro se manda un request RPC a la placa de actuadores donde se enciende y apaga la luz. Tal cual como con los ventiladores.

Control de temperatura y humedad

Para este control se tuvo que probar el funcionamiento de la matriz de control, detallada previamente. Lo importante aquí era que según una combinación de condiciones se prendiera o apagará el ventilador y los extractores. Entonces envió un nuevo mensaje con las órdenes para todos los actuadores y después agregando un filtro y el envío de una request RPC, que ya se había hecho y probado antes, se probó la matriz.

Prueba final

Con todas las pruebas individuales realizadas hicimos una prueba final con todas estas partes funcionando en conjunto. Hicimos las siguientes correcciones para corroborar el funcionamiento del sistema

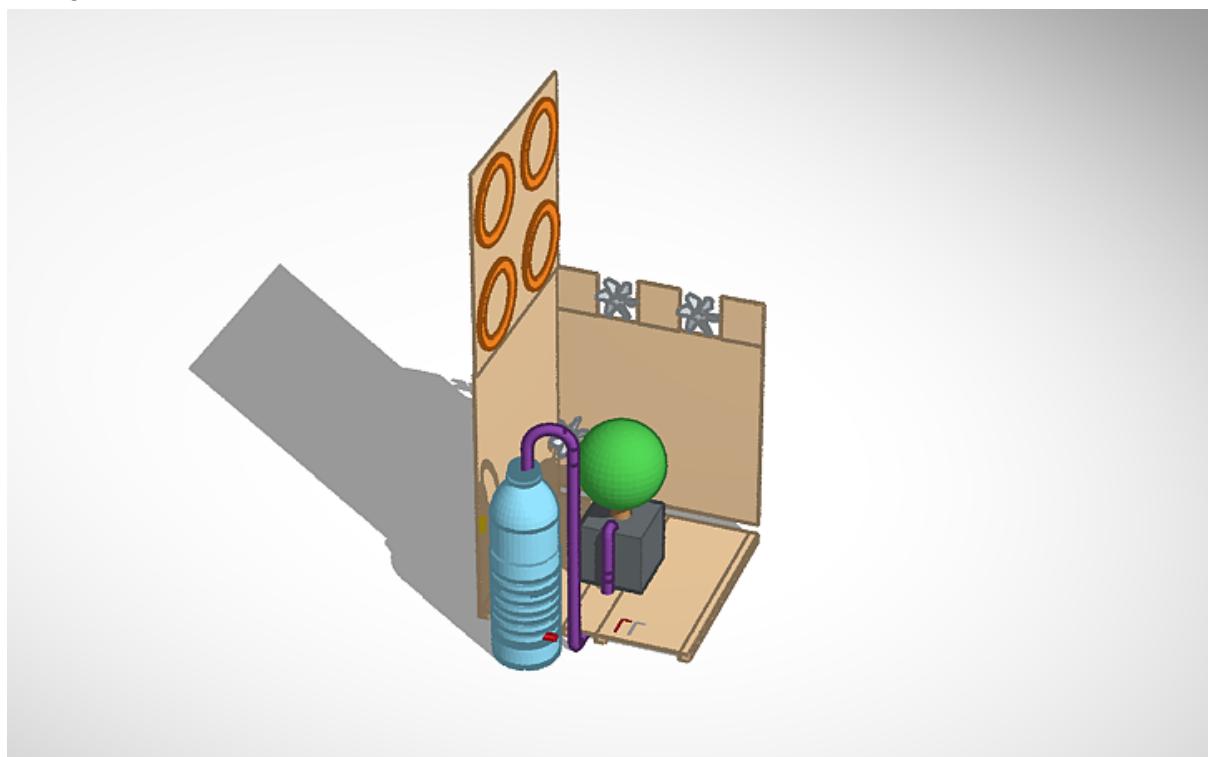


Utilizando [estos códigos](#) en cada placa pudimos comprobar el correcto funcionamiento del sistema. También probamos lo realizado en las rule chains verificando el correcto desempeño.

Diseño y construcción

Diseño inicial

Luego de definido las funciones y sensores a utilizar se realizó un modelo 3D de nuestro sistema. El mismo se puede observar en la imagen de abajo o en el link donde se puede navegar por el sistema.



<https://www.tinkercad.com/things/4tUs5qRp10x>

Armado de estructura

Comenzamos por la caja que contiene el cultivo y el sistema. Inicialmente era un cubo de 50cm de lado que luego cambiamos a un prisma de base cuadrada de 40x40x60cm. Este cambio fue realizado debido a que los cultivos tienden a ser más altos que anchos por lo que quisimos mantener ese formato para tener más comodidad.

Realizamos el armado uniendo las placas de madera con una alfajia y tornillos. La alfajia también la colocamos debajo de la caja para generar un espacio y poder colocar los microcontroladores.

Colocación microcontrolador y fuente

Fuente

La fuente se atornilló en la zona inferior de la caja donde se dejó el espacio para colocar los microcontroladores. A esta le conectamos un cable de alimentación con un enchufe macho de tres en línea.

Microcontroladores

Los microcontroladores se colocaron en dos protoboard los cuales se fijaron al suelo de la caja al lado de la fuente.

Step Down

El step down se colocó debajo de la caja y se enchufó la entrada a la fuente de 12 voltios. La salida se colocó en un lateral de la protoboard para acceder fácilmente a los 5 voltios.

Relay

La tira de relay se fijo fijo al suelo de la caja por su parte inferior y se realizaron las conexiones necesarias de alimentación.

Colocación de sensores y actuadores

Ventiladores

Siguiendo con el modelo 3d, para el ensamblado de los extractores realizamos dos agujeros en uno de los laterales con una mecha copa y colocamos ambos ventiladores. Estos fueron cableados hasta los relay que colocamos en la parte inferior de la caja.

También pegamos el ventilador interno a uno de los cantos de la alfajia que soporta las paredes de la caja. Este también fue cableado hasta los relay.

Luces

Colocamos las luces tal como se diseñó inicialmente, estas fueron soldadas entre sí ya que contienen un pin de entrada de señal y uno de salida para enlazar varias luces. Con esto nos ahorraremos pines y el cableado es menor.

Luego fueron pegadas con silicona al techo de la caja simétricamente y cableadas hasta la parte inferior de la caja donde se encuentran los microcontroladores.

Sensor de humedad y temperatura

Este sensor se colocó en una de las alfajías a media altura para captar la temperatura del medio de la habitación. También se cableó hasta la ESP32 ubicada en la parte inferior.

Sensor de humedad terrestre

Este módulo consta de dos partes, la primera es una tarjeta que contiene los electrodos y la segunda es una placa acondicionadora de señal.

La tarjeta con los electrodos fue insertada en el sustrato del cultivo directamente, y cableada hasta la zona inferior de la caja donde se colocó la placa acondicionadora de señal la cual conectamos a la ESP32.

Tanque de agua

Para el tanque de agua se utilizó un frasco de plástico con tapa en el cual se colocó el sensor de nivel de agua y la bomba de agua. El sensor de nivel de agua se colocó verticalmente en la parte inferior del frasco y se aisló con silicona caliente y termocontraible para evitar problemas con el agua. La bomba de agua, al ser sumergible, se pegó directamente en el fondo del tanque sin ningún paso previo.

Tanto el sensor como la bomba se cablearon hasta el fondo de la caja y se conectaron con sus respectivos microcontroladores.

Dificultades

Nuestra primer dificultad a la hora de encarar el proyecto fue la conexión vía MQTT con thingsboard, ya que era algo completamente nuevo para nosotros. Gracias a la clase introductoria de los tutores sobre thingsboards y MQTT y el ejemplo que nos brindaron, esto ya no fue más un problema. Realizamos la conexión con éxito en pocos minutos y empezamos a reportar a la cola de telemetria.

Ya habiendo avanzado en el proyecto nos topamos a la hora de accionar una salida en una placa desde el filtrado del reporte de una telemetría que comenzó en otra tarjeta. Esto lo intentamos resolver leyendo la documentación y ejemplos de thingsboard pero eran muy poco claros y engorrosos de entender. Conversamos con otros colegas que se estaban enfrentado al mismo problema y también con los tutores pudiendo llegar a la solución.

Más adelante a la hora de pensar en los perfiles que finalmente realizamos, primero teníamos que cambiar los valores fijos que utilizamos al principio por atributos. Estos atributos los asociamos a la placa que reportaba el mensaje como “Client attributes” y los concatenamos en la metadata del mensaje. Esto también nos presentó una dificultad ya que no sabíamos cómo trabajar con los mensajes de thingsboard y mensajes de estilo JSON. Lo

cual el acceder al valor final de los atributos previamente concatenados en la metadata no hizo perder mucho tiempo a la hora de programar y dejar funcionando los filtros de control.

Continuando con los perfiles, ahora varios, se tenía que concatenar varios atributos de cliente que estaban asociados a la placa de cenco. Pensando también que esta misma era la que debería guardarlos y al momento de encenderse volver al estado que debería estar el sistema, nos dimos cuenta que no era la mejor opción. Optamos por cambiar los mismo a atributos de servidor, los cuales se guardan en el mismo y están asociados al sistema pero no a ninguno de los dispositivos en sí. Entonces resolvimos el problema del estado inicial desde thingsboard. Guardando las variables ahí y al momento de la conexión sea el encargado de volver al estado deseado.

Entonces ahora para setear los atributos de los perfiles ya no era necesario enviar un request RPC como debería de ser en el caso de haberlos hecho como atributos de cliente. Esto nos resultó una ventaja porque también pudimos resolver la dificultad que nos generaba el guardar estos atributos de forma amigable para el usuario desde el dashboard.

Conclusiones

Podemos concluir que se pudo realizar el proyecto con éxito. El haber realizado un buen plan de trabajo previo nos ayudó a mantenernos en orden y llevar adelante el plazo de entrega. Se podría decir que lo único para detallar del mismo fue que subestimamos un poco la dificultad de la programación de la rule chain y en particular el dejar en funcionamiento la la matriz de control de temperatura y humedad. Pero finalmente pudimos resolverlo a tiempo.

Por último podemos decir que quedamos conformes con el proyecto y el prototipo. Creemos que se asemeja mucho a la realidad y que la escalabilidad de este proyecto no sería una dificultad mayor, tanto para cultivos pequeños como para cultivos industriales.

Mejoras a futuro y defectos a mejorar

Una de las cosas que quedó para solucionar sería qué pasa si uno de los dispositivos se apaga o se desconecta. En este caso, estaria bueno agregar una alarma en thingsboard para avisar al usuario. De todas maneras hay ciertas cosas que se hicieron al respecto. En la rule chain de control, luego de enviar un request RPC, por ejemplo para prender las luces, si el bloque no tiene respuesta del dispositivo, este tiene una respuesta de "Failure". Con esto se creó una alarma que avisa al usuario sobre este comportamiento.

Continuando con las alarmas, algo que se podía incluir a futuro seria el envio de esta alarma via mail o algo del estilo, más personal del usuario, para que reciba, si lo desea, notificaciones directas de la alarma, con toda la información correspondiente para que se tomen las medidas que se crean necesarias.

Otras posibles mejoras serían en la potencia del sistema para poder controlar el estado deseado. Para casos donde se necesiten humedades más altas, sería necesaria tal vez colocar un humidificador. En caso contrario, de necesitar humedades más bajas, colocar un deshumidificador. Para el caso de la temperatura se solucionaría con aire acondicionado. Pero lo más importante es que para cualquiera fuese el caso, esto no requeriría una dificultad elevada para el sistema, será más bien un tema de costos. En nuestro sistema simplemente se conectarán estos nuevos dispositivos a la placa de actuadores y se accionarán bajo request RPC que se comandará desde los bloques de control en la rule chain.

Siguiendo con dispositivos a mejorar tenemos que incluir el sensor de humedad terrestre. Con el que se tiene actualmente solo se puede saber cuando el suelo está totalmente seco o totalmente mojado. Esto no nos permite tener diferentes perfiles de humedad terrestre como se tenía pensado. Esto sería bueno incorporarlo con un cronograma de riego por tiempo.