



UNIVERSIDAD DE GRANADA

Telémetro

Prácticas de IoT

Domótica

Autor

Pablo Valenzuela Álvarez (pvalenzuela@correo.ugr.es)



INDICE

Primer dispositivo IoT.....	3
Telémetro mejorado.....	4
Ejercicios.....	10
Ejercicio 1.....	10
Ejercicio 2.....	15
Ejercicio 3.....	18
Ejercicio 4.....	19
Ejercicio 5.....	21

Primer dispositivo IoT

En el guión de la práctica se proporciona un código con el cual podemos probar el funcionamiento de las placas de arduino que tenemos disponibles. En nuestro caso se utilizará una placa **ESP8266** para la realización de estas prácticas (ver figura 1).

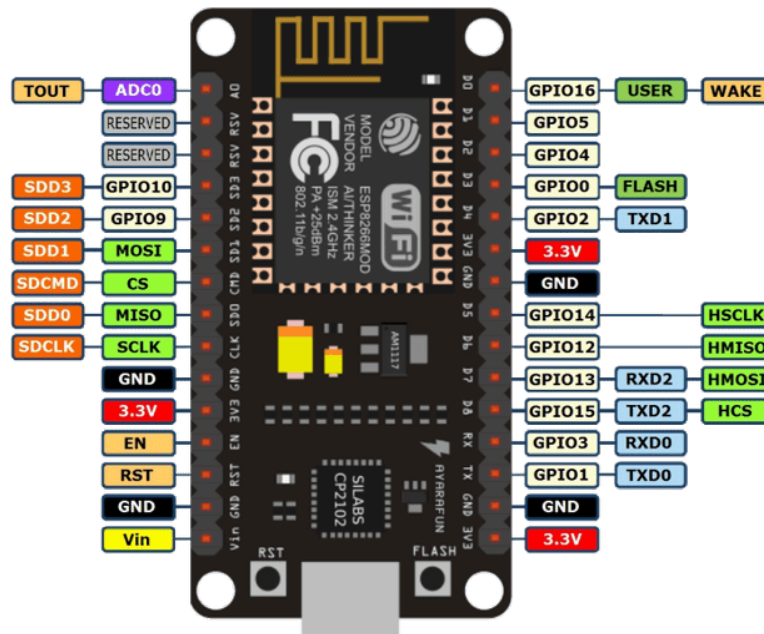


Figura 1. Placa arduino modelo ESP8266.

Una vez montado el dispositivo en la placa y habiendo conectado los sensores necesarios (ver figura 2), al ejecutar el código proporcionado en el guión (ver figura 3) obtendremos los resultados de la figura 4. Ya disponemos de un dispositivo capaz de medir la distancia hasta un obstáculo situado delante suya.

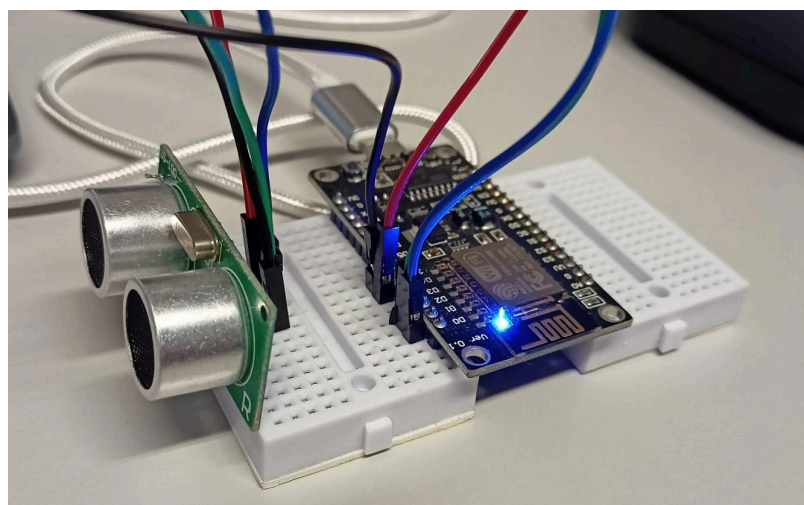


Figura 2. Dispositivo montado.

```

1 // Pines del sensor de ultrasonidos
2 #define trigPin 5
3 #define echoPin 4
4 void setup() {
5     Serial.begin(9600);
6     pinMode ( LED_BUILTIN, OUTPUT );
7     pinMode ( trigPin, OUTPUT );
8     pinMode(echoPin, INPUT);
9     digitalWrite(trigPin, LOW);
10 }
11 void loop() {
12     digitalWrite(LED_BUILTIN, HIGH);
13     delay(500);
14     digitalWrite(LED_BUILTIN, LOW);
15     delay(500);
16     // Medida de la distancia
17     digitalWrite(trigPin, HIGH);
18     delayMicroseconds(10);
19     digitalWrite(trigPin, LOW);
20     float distancia = 0.0172*pulseIn(echoPin, HIGH);
21     // Visualización por el puerto serie
22     Serial.print("Distancia: ");
23     Serial.println(distancia);
24 }

```

Figura 3. Código para un telémetro básico.

Message (Enter to send message to 'NodeMCU 0.9 (ESP-12 Module)' on 'COM3')

```

12:01:43.245 -> Distancia: 8.22
12:01:44.287 -> Distancia: 9.65
12:01:45.284 -> Distancia: 19.04
12:01:46.300 -> Distancia: 11.46
12:01:47.316 -> Distancia: 69.02
12:01:48.294 -> Distancia: 14.98
12:01:49.335 -> Distancia: 36.89
12:01:50.320 -> Distancia: 38.85
12:01:51.332 -> Distancia: 11.44
12:01:52.338 -> Distancia: 31.89
12:01:53.379 -> Distancia: 43.19
12:01:54.405 -> Distancia: 69.45
12:01:55.395 -> Distancia: 70.26

```

Figura 4. Salida del dispositivo.

Telómetro mejorado

Se ha implementado una segunda versión para el telómetro, en la que incluye un **diodo LED**, un sensor **BMP180** para medir la temperatura y la presión atmosférica, y una **pantalla OLED**, a parte del sensor de ultrasonidos incluido en la implementación anterior. En las figuras 8, 9 y 10 se puede ver en funcionamiento..

En el código de la figura 5 se puede observar los pines de la placa ESP8266 que usamos para los sensores. El sensor BMP180 y la pantalla OLED comparten los pines **SDA** y **SCL**.

```
1  #include <Wire.h>
2  #include <Adafruit_GFX.h>
3  #include <Adafruit_SSD1306.h>
4  #include <Adafruit_BMP085.h>
5
6  // Pines del sensor de ultrasonidos
7  #define trigPin 5 // D1
8  #define echoPin 4 // D2
9
10 // Pin diodo led
11 #define ledPin 2 //D4
12
13 // Pines sda y scl para la pantalla y el sensor de temperatura
14 #define sdaPin 13 // D7
15 #define sclPin 12 // D6
16
17 Adafruit_SSD1306 display(128, 64, &Wire, -1); // definicion del display de la pantalla OLED
18 Adafruit_BMP085 bmp; // definicion del sensor bmp
19
```

Figura 5. Pines de los sensores.

En el código de la función setup de la figura 6 se hace la configuración inicial para los sensores.

```

22 void setup() {
23     Serial.begin(9600);
24
25     pinMode ( ledPin, OUTPUT );
26     pinMode ( trigPin, OUTPUT );
27     pinMode(echoPin, INPUT);
28     digitalWrite(trigPin, LOW);
29     digitalWrite(ledPin, LOW);
30
31     Wire.begin(sdaPin, sclPin);
32
33     // Inicializar el sensor BMP180
34     if (!bmp.begin()) {
35         Serial.println("No se pudo encontrar el sensor BMP180");
36         while (true);
37     }
38
39     //Inicializar la pantalla OLED
40     if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
41         Serial.println(F("No se pudo iniciar la pantalla OLED"));
42         while (true);
43     }
44     display.clearDisplay();
45 }

```

Figura 6. Función setup.

Y en el código de la función loop (ver figura 7), imprimimos en la pantalla los valores recogidos por los sensores. Para el diodo LED, hacemos que se encienda si la distancia recogida por el sensor de ultrasonidos sea menor que 10 cm y se apague en el caso contrario.

```

47 void loop() {
48     delay(1000);
49
50     // Medida de la distancia
51     digitalWrite(trigPin, HIGH);
52     delayMicroseconds(10);
53     digitalWrite(trigPin, LOW);
54     float distancia = 0.0172*pulseIn(echoPin, HIGH);
55
56     // Activacion del LED
57     int led = distancia < 10 ? 1 : 0; // 0=desactivado, 1=activado
58     digitalWrite(ledPin, led);
59
60     // Leer temperatura y la presion del BMP180
61     float temperature = bmp.readTemperature();
62     int32_t pressure = bmp.readPressure();
63
64     // Mostrar los datos en la pantalla OLED
65     display.clearDisplay();
66     display.setTextSize(2);
67     display.setTextColor(SSD1306_WHITE);
68     display.setCursor(0, 0);
69     display.print(distancia);
70     display.print(" cm");
71     display.setCursor(0, 25);
72     display.print(temperature);
73     display.println(" C");
74     display.setCursor(0, 50);
75     display.print(pressure / 100.0);
76     display.println(" hPa");
77     display.display();
78
79     // Visualización por el puerto serie
80     Serial.print("Distancia: ");
81     Serial.println(distancia);
82     Serial.print(" Temperatura: ");
83     Serial.println(temperature);
84     Serial.print(" Presion: ");
85     Serial.println(pressure / 100.0);
86 }

```

Figura 7. Función loop.

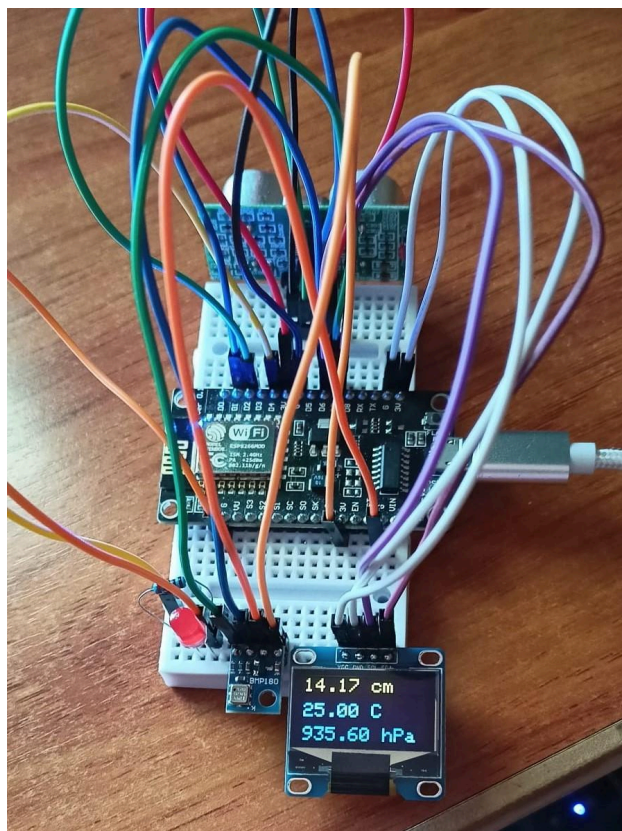


Figura 8. Dispositivo telémetro mejorado.

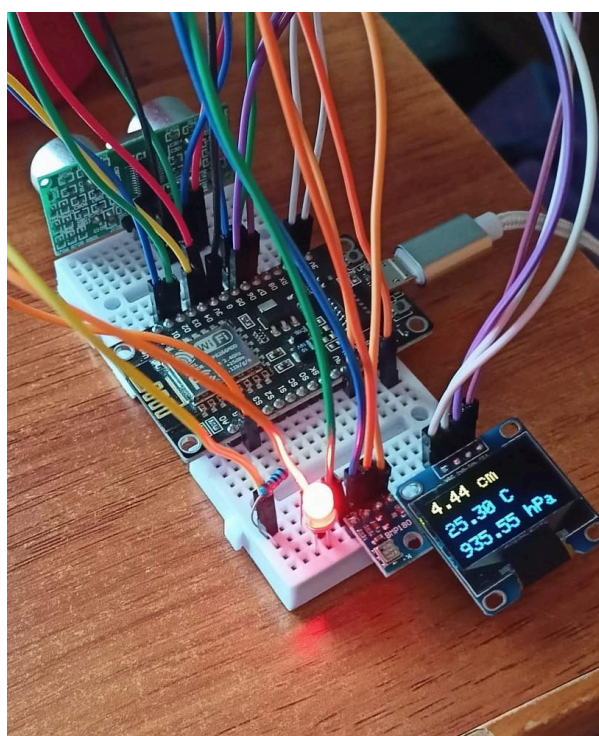


Figura 9. Diodo LED encendido al haber un obstáculo situado a 4.44cm.

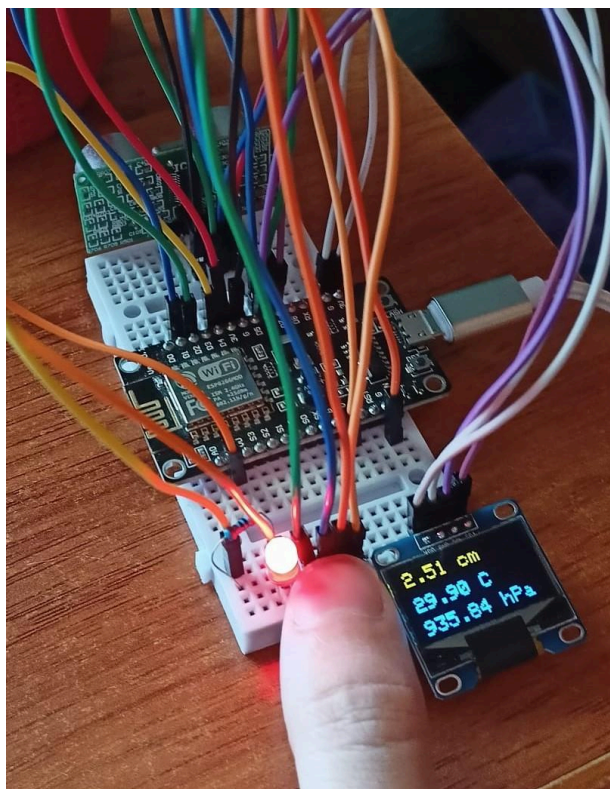


Figura 10. Diodo LED encendido y temperatura aumentada

Ejercicios

Ejercicio 1

En el guión, también se proporciona otro código con el que podemos conectar nuestro dispositivo a la plataforma **ThingSpeak**. Pero para este ejercicio tenemos que modificar ese código para que se conecte usando MQTT.

Una vez tengamos una cuenta creada en la plataforma, necesitaremos crear un canal. En mi caso, vamos a añadir cuatro campos, uno para ver la distancia recogida por el sensor de ultrasonidos, otros dos para la temperatura y la presión del sensor BMP180 y un último donde que indicará si está activo un led (ver figura 11). Enlace al [canal](#).

Channel Settings

Percentage Complete	30%	
Channel ID	2550878	
Name	<input type="text" value="TelemetroPLUS"/>	
Description	<input type="text"/>	
Field 1	<input type="text" value="Distancia"/>	<input checked="" type="checkbox"/>
Field 2	<input type="text" value="Temperatura"/>	<input checked="" type="checkbox"/>
Field 3	<input type="text" value="Presión"/>	<input checked="" type="checkbox"/>
Field 4	<input type="text" value="LED"/>	<input checked="" type="checkbox"/>

Figura 11. Configuración del canal en ThingSpeak.

Seguidamente, añadiremos un dispositivo MQTT, y autorizamos al canal que acabamos de crear para que pueda publicar y suscribirse (ver figura 12).

MQTT Credentials

Use these MQTT credentials to publish and subscribe to ThingSpeak channels. [Learn More](#)

Client ID

HwsQDSArJioXCR03LBY9Chg

Username

HwsQDSArJioXCR03LBY9Chg

Password

●●●●●●●●●●●●●●●●●●●●

Authorize channels to access

-- Select a Channel --

Add Channel

Authorized Channel 	Allow Publish	Allow Subscribe	
TelemetryPLUS (2550878)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	×

Save

Cancel

Figura 12. Añadiendo un dispositivo MQTT.

Ahora toca modificar el código. En la figura 13 se pueden observar algunas de las variables que necesitamos para conectar nuestro dispositivo a la plataforma ThingSpeak como son la red, el servidor MQTT, el canal donde se publicará y los tópicos identificados.

```
24  /* configuracion mqtt thingspeak */
25
26  char ssid[] = "PablOno2";
27  char pass[] = "basKET,2:N";
28
29  const char* publishTopic = "channels/2550878/publish";
30  const char* mqtt_server = "mqtt3.thingspeak.com";
31
32  const char* f1 = "channels/2550878/subscribe/fields/field1";
33  const char* f2 = "channels/2550878/subscribe/fields/field2";
34  const char* f3 = "channels/2550878/subscribe/fields/field3";
35  const char* f4 = "channels/2550878/subscribe/fields/field4";
36
37  WiFiClient espClient;
38  PubSubClient client(espClient);
```

Figura 13. Variables para ThingSpeak.

La función que se usa conectarse al servidor MQTT (ver figura 14) hace uso de un fichero **SECRETS** denominado *"mqtt_secrets.h"*, donde están definidas las variables

SECRET_MQTT_CLIENT_ID,
SECRET_MQTT_PASSWORD.

SECRET_MQTT_USERNAME

y

```
56 // reconectar cliente mqtt
57 void reconnect() {
58     while (!client.connected()) {
59         Serial.print("Intentando la conexión MQTT a ");
60         Serial.print(mqtt_server);
61         Serial.println(" ...");
62         if (client.connect(SECRET_MQTT_CLIENT_ID, SECRET_MQTT_USERNAME, SECRET_MQTT_PASSWORD)) {
63             Serial.println("conectado");
64             client.subscribe(f1);
65             client.subscribe(f2);
66             client.subscribe(f3);
67             client.subscribe(f4);
68         } else {
69             Serial.print("fallo, rc=");
70             Serial.print(client.state());
71             Serial.println(" intentando de nuevo en 5 segundos");
72             delay(5000);
73         }
74     }
75 }
76 }
```

Figura 14. Función para conectarse al servidor MQTT.

Por último, hay que cambiar la función loop para que envíe los datos al servidor MQTT. Cada veinte segundos se publican los datos como se puede ver en las líneas 154 en adelante de la figura 15.

```

106 void loop() {
107
108     if (!client.connected()) reconnect();
109     client.loop();
110
111     // Envío datos cada 20 segundos
112     if(millis() - lastUploadedTime > postingInterval){
113
114         // Medida de la distancia
115         digitalWrite(trigPin, HIGH);
116         delayMicroseconds(10);
117         digitalWrite(trigPin, LOW);
118         float distancia = 0.0172*pulseIn(echoPin, HIGH);
119
120         // Activacion del LED
121         int led = distancia < 10 ? 1 : 0; // 0=desactivado, 1=activado
122         digitalWrite(ledPin, led);
123
124         // Leer temperatura y la presion del BMP180
125         float temperature = bmp.readTemperature();
126         int32_t pressure = bmp.readPressure();
127
128         // Mostrar los datos en la pantalla OLED
129         display.clearDisplay();
130         display.setTextSize(2);
131         display.setTextColor(SSD1306_WHITE);
132         display.setCursor(0, 0);
133         display.print(distancia);
134         display.print(" cm");
135         display.setCursor(0, 25);
136         display.print(temperature);
137         display.println(" C");
138         display.setCursor(0, 50);
139         display.print(pressure / 100.0);
140         display.println(" hPa");
141         display.display();
142
143         // Visualización por el puerto serie
144         Serial.print("Distancia: ");
145         Serial.println(distancia);
146         Serial.print(" Temperatura: ");
147         Serial.println(temperature);
148         Serial.print(" Presion: ");
149         Serial.println(pressure / 100.0);
150         Serial.print(" LED: ");
151         Serial.println(led);
152
153         // Envío de datos al MQTT
154         String topics = String("field1=" + String(distancia) + "&field2=" + String(temperature) + "&field3=" +
155         if (client.publish(publishTopic, topics.c_str())) {
156             Serial.println("Datos enviados correctamente");
157         }
158         else {
159             Serial.println("Error al enviar los datos");
160         }
161         lastUploadedTime = millis();
162     }
163 }
164 }

```

Figura 15. Cambios en la función loop para que envíe los datos al servidor MQTT.

Si lo hemos hecho bien, los datos están siendo enviados a nuestro canal de ThingSpeak. Y como podemos comprobar en las figuras 16 y ,efectivamente están siendo correctamente enviados porque podemos observar que hay cambios en las gráficas.

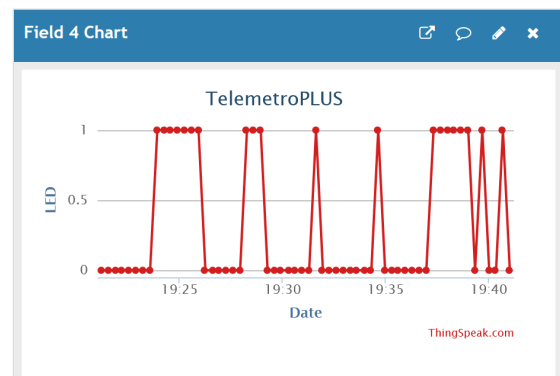
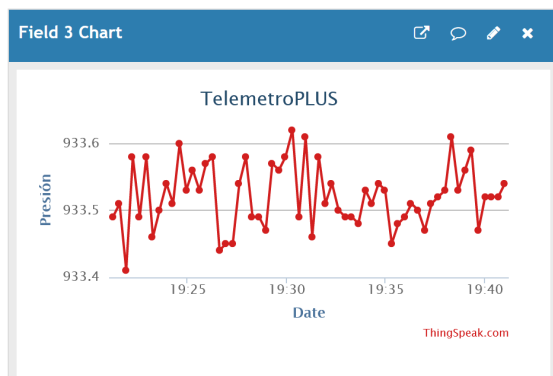
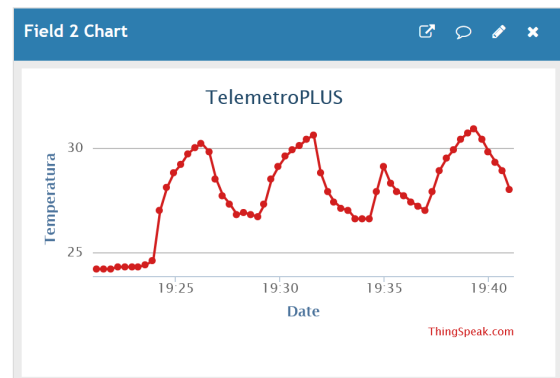
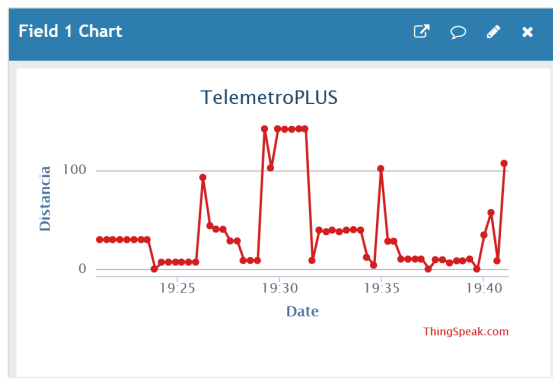


Figura 16. Estadísticas del canal de ThingSpeak.

Ejercicio 2

En este ejercicio tenemos que enviar los datos a otra plataforma MQTT, el gui3n sugiere **HiveMQ**.

Para ello debemos dirigirnos a el [broker p3blico MQTT](#) de HiveMQ y pulsar el bot3n *connect* de la zona derecha(ver figura 17). Hecho esto tenemos que a3adir al c3digo alguna l3neas para que env3e los datos.

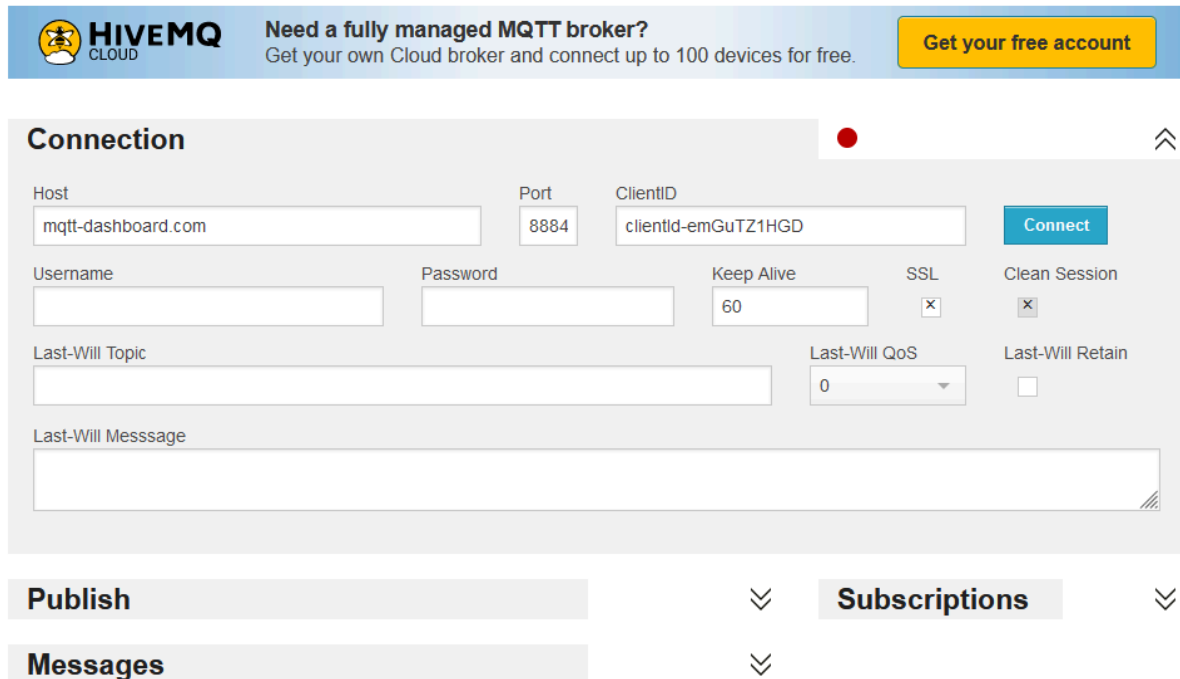


Figura 17. Broker MQTT de HiveMQ.

Hay que declarar el servidor y los campos en los que se subscribir3 y publicar3n los datos. En las siguientes figuras (18, 19 y 20) se muestran fragmentos del c3digo que hay que a3adir.

```
41  /* Setup de HiveMQ*/
42  const char* mqtt_server_hv = "broker.hivemq.com";
43  const char* f1_hv = "telemetroPLUS/distancia";
44  const char* f2_hv = "telemetroPLUS/temperatura";
45  const char* f3_hv = "telemetroPLUS/presion";
46  const char* f4_hv = "telemetroPLUS/led";
```

Figura 18. Servidor MQTT de HiveMQ y t3picos que se enviar3n.


```

91 void reconnect_hv() {
92     while (!client_hv.connected()) {
93         Serial.print("Intentando la conexión MQTT a ");
94         Serial.print(mqtt_server_hv);
95         Serial.println(" ...");
96         if (client_hv.connect("telemetroPLUS")) {
97             Serial.println("conectado");
98             client_hv.subscribe(f1_hv);
99             client_hv.subscribe(f2_hv);
100            client_hv.subscribe(f3_hv);
101            client_hv.subscribe(f4_hv);
102        } else {
103            Serial.print("fallo, rc=");
104            Serial.print(client_hv.state());
105            Serial.println(" intentando de nuevo en 5 segundos");
106            delay(5000);
107        }
108    }
109 }

```

Figura 19. Conexión con el servidor MQTT

```

199 //Envío a HiveMQ
200 if (client_hv.publish(f1_hv, String(distancia).c_str()) &&
201     client_hv.publish(f2_hv, String(temperature).c_str()) &&
202     client_hv.publish(f3_hv, String(pressure/100).c_str()) &&
203     client_hv.publish(f4_hv, String(led).c_str())) {
204     Serial.println("Datos enviados a HiveMQ");
205 } else Serial.println("Error al enviar los datos");

```

Figura 20. Envío de datos al servidor MQTT

Como último paso queda acceder al broker público y ver si se están enviando los datos. En la figura 21 se puede observar que el dispositivo está enviando correctamente los datos (hay que subscribirse a los tópicos en la zona derecha para verlos)

Connection

connected

Publish

Topic

QoS

0

Retain

☐

Publish

Message

Subscriptions

Add New Topic Subscription

Qos: 2

telemetroPLUS/dista...

X

Qos: 2

telemetroPLUS/tem...

X

Qos: 2

telemetroPLUS/presi...

X

Qos: 2

telemetroPLUS/led

X

Messages

2024-05-17 10:13:59	Topic: telemetroPLUS/temperatura	Qos: 0	29.10
2024-05-17 10:13:59	Topic: telemetroPLUS/led	Qos: 0	1
2024-05-17 10:13:59	Topic: telemetroPLUS/presion	Qos: 0	935
2024-05-17 10:13:59	Topic: telemetroPLUS/distancia	Qos: 0	2.46
2024-05-17 10:13:39	Topic: telemetroPLUS/temperatura	Qos: 0	24.00
2024-05-17 10:13:39	Topic: telemetroPLUS/led	Qos: 0	0
2024-05-17 10:13:39	Topic: telemetroPLUS/presion	Qos: 0	935
2024-05-17 10:13:39	Topic: telemetroPLUS/distancia	Qos: 0	36.05
2024-05-17 10:13:19	Topic: telemetroPLUS/temperatura	Qos: 0	24.20
2024-05-17 10:13:19	Topic: telemetroPLUS/led	Qos: 0	0
2024-05-17 10:13:19	Topic: telemetroPLUS/presion	Qos: 0	935
2024-05-17 10:13:19	Topic: telemetroPLUS/distancia	Qos: 0	36.05

Figura 21. Resultados vistos desde el broker MQTT público de HiveMQ.

Ejercicio 3

Ya hemos implementado una mejora en el telémetro vista en [Telémetro mejorado](#).

Ejercicio 4

Para la realización de este ejercicio, hemos usado el widget “*IoT ThingSpeak Monitor Widget*”. Configurando los campos que queremos que muestre, podemos ver en nuestro teléfono los datos reales que está mandando nuestro dispositivo (ver figura 22 y 23). En este widget podemos ver las gráficas de ThingSpeak con las que vemos la evolución de los datos enviados (ver figura 24 y 25).



Figura 22. Medidas de distancia y temperatura.

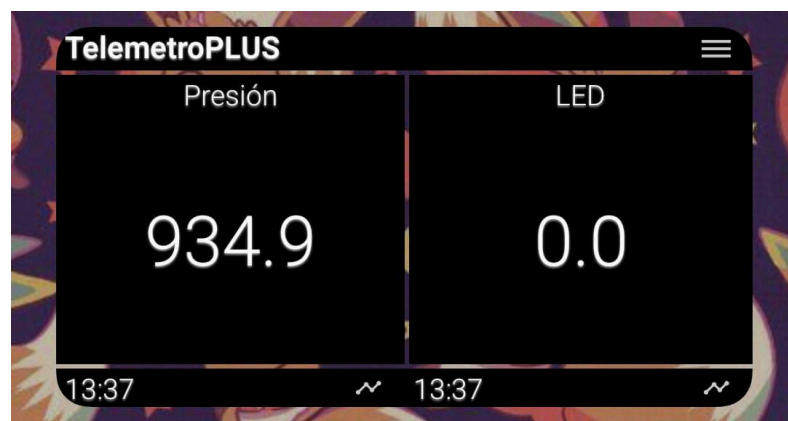


Figura 23. Medidas de presión atmosférica y valor de LED.

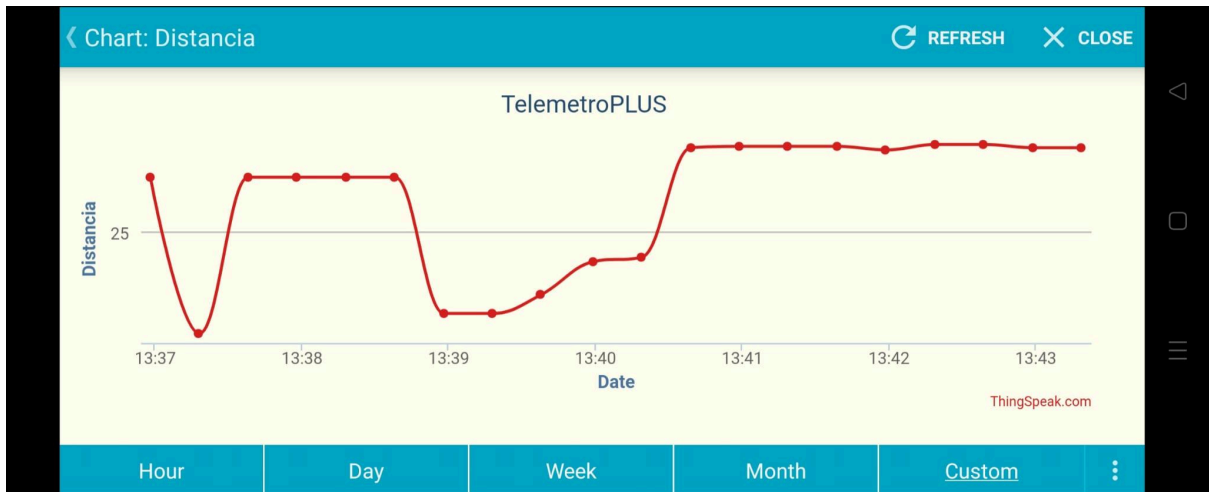


Figura 24. Gráfica de distancia.

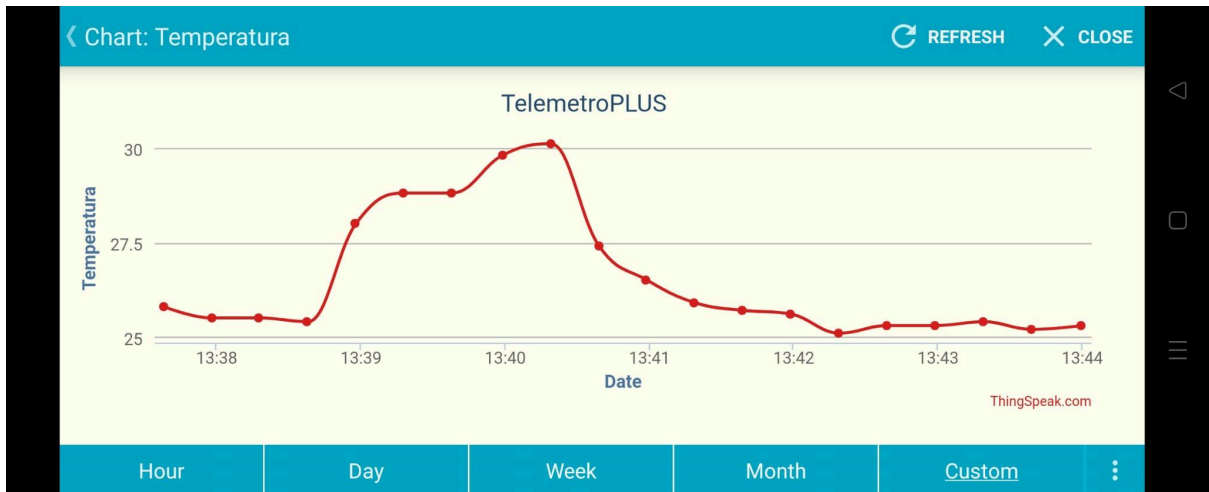


Figura 25. Gráfica de temperatura.

Ejercicio 5

En este último ejercicio vamos a instalar Tasmota en el NodeMCU, de manera que podamos conectarlo a un centro de control domótico como Home Assistant.

La forma más sencilla de instalar esta aplicación es desde su propia página web. Desde la sección [Getting Started - Tasmota](#) podemos seleccionar el firmware que queramos instalar, en mi caso usaré **Tasmota Sensors** debido a que tengo los sensores de ultrasonidos y temperatura (ver figura 26).

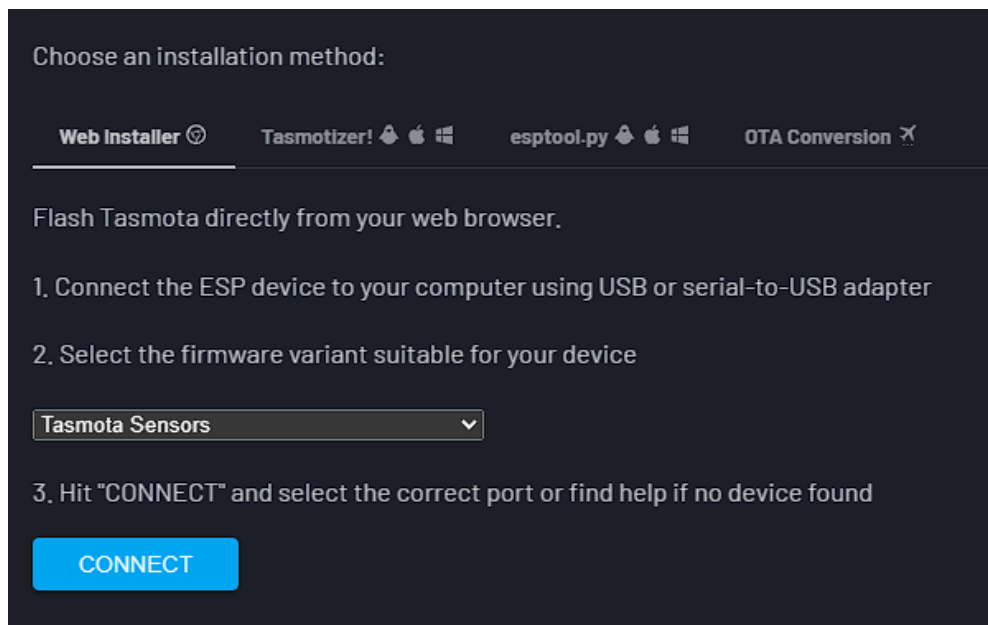


Figura 26. Configuración de la instalación de Tasmota.

Una vez instalado, configuramos los pines como se ve en la figura 27. Como en los ejercicios anteriores de arduino, tenemos que decirle en que pines tenemos los sensores conectados.

Generic

Tasmota

Module parameters

Module type (Sonoff Basic)

Generic (18) ▼

D3 GPIO0	None ▼	
TX GPIO1	None ▼	
D4 GPIO2	Relay ▼	1 ▼
RX GPIO3	None ▼	
D2 GPIO4	SR04 Ech/RX ▼	1 ▼
D1 GPIO5	SR04 Tri/TX ▼	1 ▼
D6 GPIO12	I2C SCL ▼	
D7 GPIO13	I2C SDA ▼	
D5 GPIO14	None ▼	
D8 GPIO15	None ▼	
D0 GPIO16	None ▼	
A0 GPIO17	None ▼	

Save

Configuration

Tasmota 14.0.0 (release-sensors) by Theo Arends

Figura 27. Configuración de los pines de la placa.

Una vez guardada la configuración, si los sensores del dispositivo están funcionando, aparecerá la información que se muestra en la figura 28 en la parte superior.

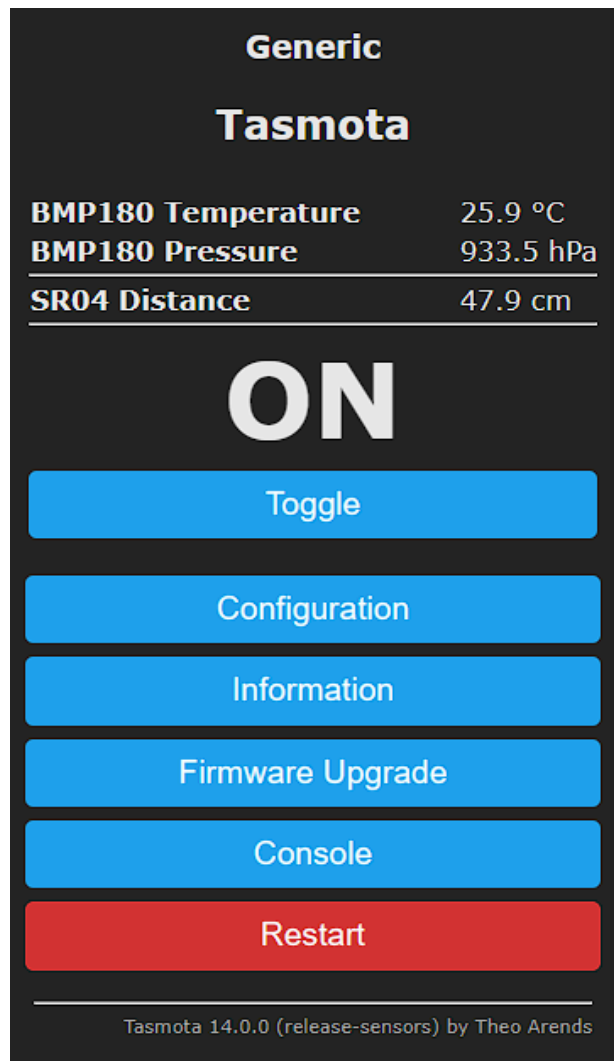


Figura 28. Datos mostrados en la interfaz.

Si queremos conectar nuestro dispositivo a un centro de control doméstico como Home Assistant, debemos proporcionar la dirección IP de dicho dispositivo. Para que funcione correctamente, nuestro dispositivo Home Assistant debe tener un usuario para las conexiones MQTT (ver figura 29).

Generic

TelemetroPLUS

MQTT parameters

Host ()

192.168.0.18

Port (1883)

1883

Client (DVES_A33F4E)

DVES_%06X

User (DVES_USER)

mqttuser

Password ☐

....

Topic = %topic% (tasmota_A33F4E)

tasmota_%06X

Full Topic (%prefix%/%%topic%/)

%prefix%/%%topic%/

Save

Configuration

Tasmota 14.0.0 (release-sensors) by Theo Arends

Figura 29. Conexión con Home Assistant.

Le asignamos un nombre para identificarlo fácilmente en nuestro Home Assistant (ver figura 30).

Generic

Tasmota

Other parameters

Template

```
{"NAME":"Generic","GPIO":[1,1,1,1,1,1,1,1]}
```

☐ Activate

Web Admin Password

....

☒ HTTP API enable
☒ MQTT enable

Device Name (Tasmota)

TelemetroPLUS

Friendly Name 1 (Tasmota)

TelemetroPLUS

Emulation

☐ None
☒ Belkin WeMo single device
☐ Hue Bridge multi device

Save

Configuration

Tasmota 14.0.0 (release-sensors) by Theo Arends

Figura 30. nombre del dispositivo tasmota

Por último, hemos de dirigirnos a nuestro dispositivo Home Assistant, sección Ajustes>Dispositivos y Servicios, y nos tiene que salir algo parecido a la figura 31. Nuestro Home Assistant reconoce el dispositivo Tasmota y nos permite añadir los controles y sensores que tiene a nuestro panel principal (ver figura 32).

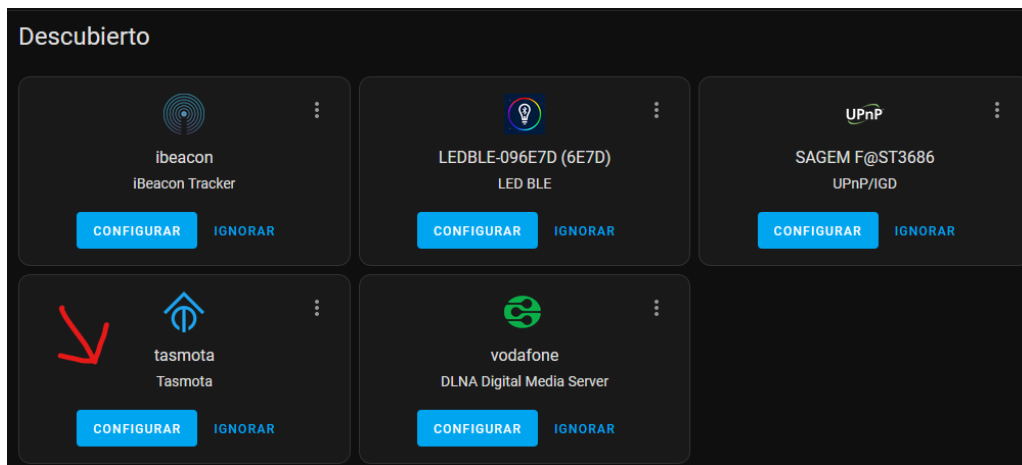


Figura 31. Dispositivo Tasmota descubierto.

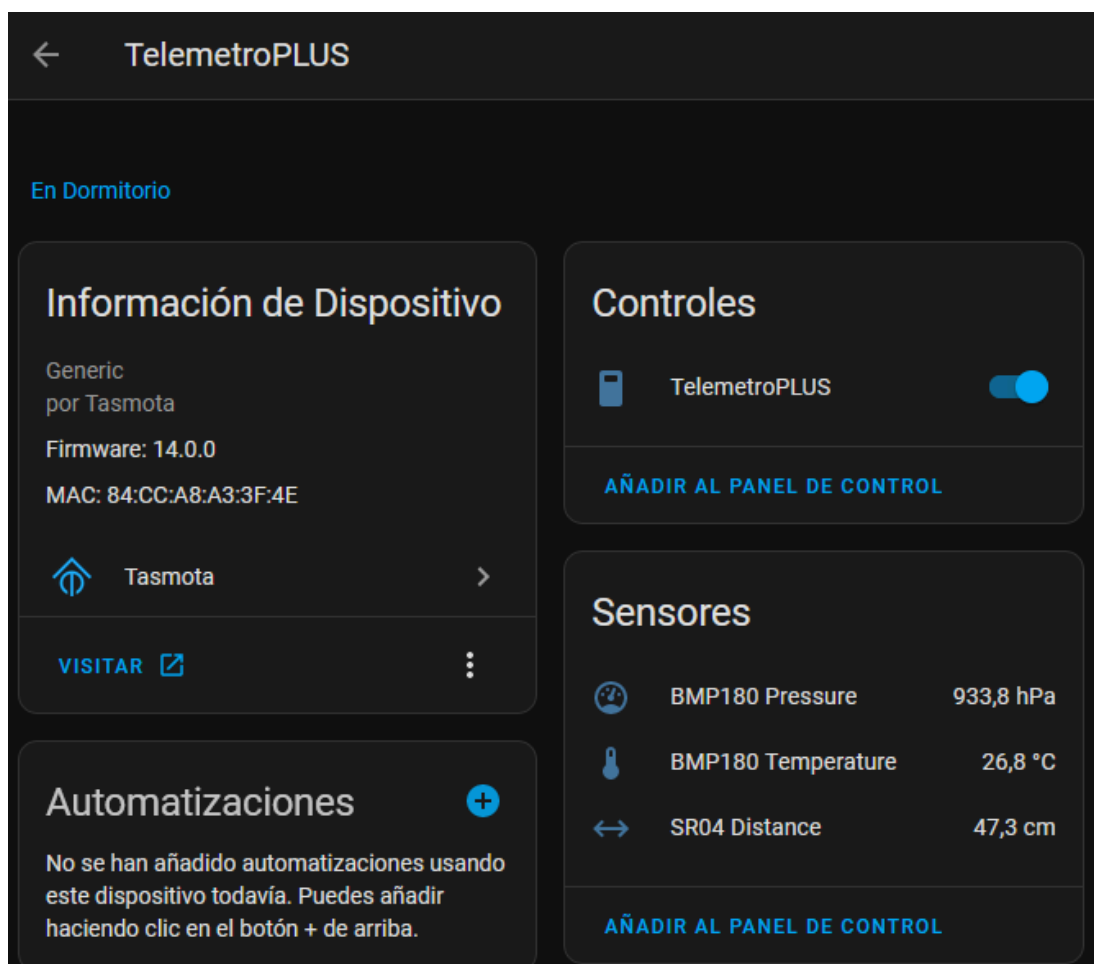


Figura 32. Controles y sensores del dispositivo Tasmota descubierto.

Una vez lo añadimos a nuestro panel principal de Home Assistant, podemos ver los valores que está registrando cada vez que entremos y naveguemos por esta pestaña de la interfaz web (ver figura 33) y de la aplicación móvil (ver figura 34).

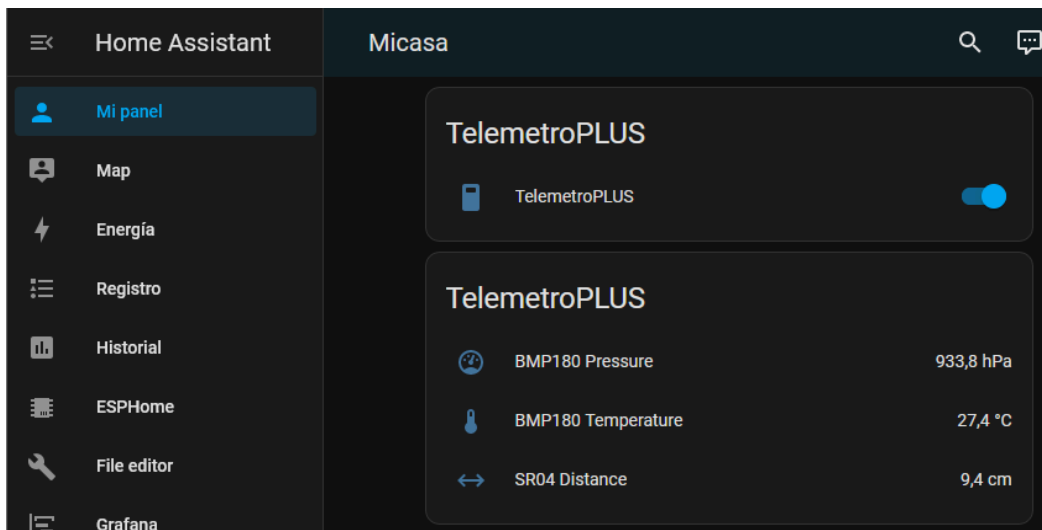


Figura 33. Panel principal de la interfaz web.

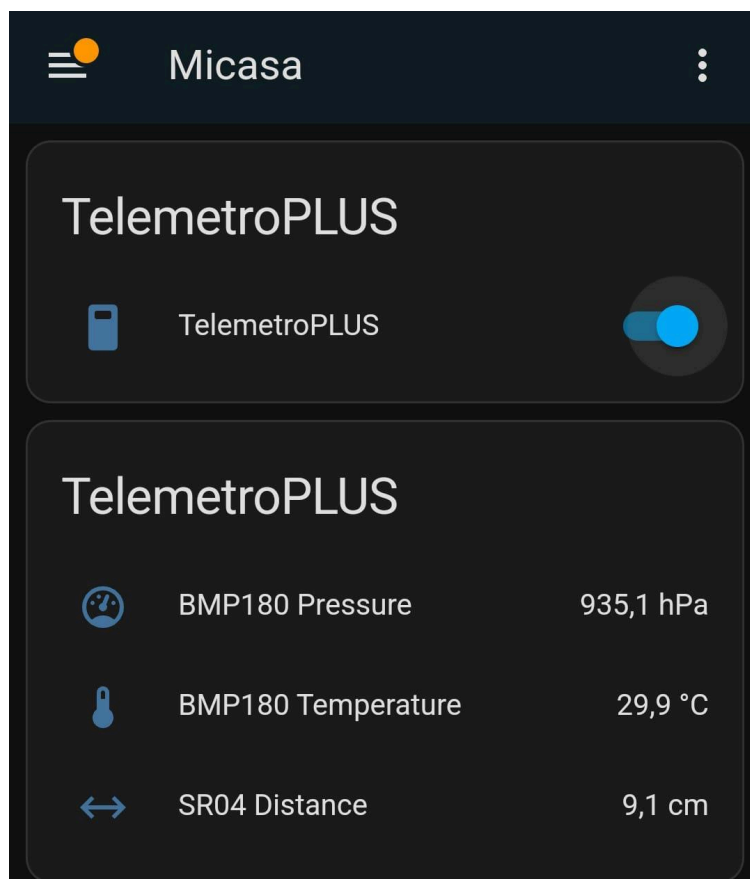


Figura 34. Panel principal de la aplicación móvil.