

# Conectividad en dispositivos embebidos

Vadym Formanyuk

Tecnología informática y computación, Universidad de alicante,  
San vicente del raspeig, 03690, Alicante, España.

Contributing authors: [vf13@alu.ua.es](mailto:vf13@alu.ua.es);

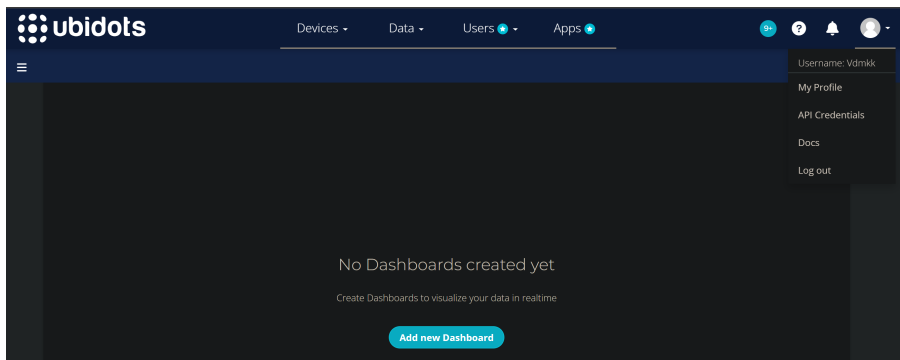
## Abstract

Arduino es una plataforma de hardware libre y de bajo costo que se utiliza para crear prototipos de proyectos electrónicos. MQTT (Message Queuing Telemetry Transport) es un protocolo de comunicación ligero y eficiente diseñado para conectar dispositivos IoT (Internet de las cosas) a través de redes de baja velocidad y alta latencia. La combinación de Arduino y MQTT permite a los dispositivos conectados enviar y recibir mensajes a través de la red, lo que permite la creación de proyectos IoT escalables y distribuidos. Además, la integración de la nube con Arduino y MQTT permite una mayor flexibilidad y facilidad de uso al almacenar y procesar datos de los dispositivos conectados. Al utilizar la nube como intermediario, los dispositivos Arduino pueden comunicarse entre sí sin la necesidad de configurar redes complejas. La nube actúa como un intermediario para enviar y recibir datos entre los dispositivos, lo que permite una fácil gestión y monitoreo de los dispositivos IoT.

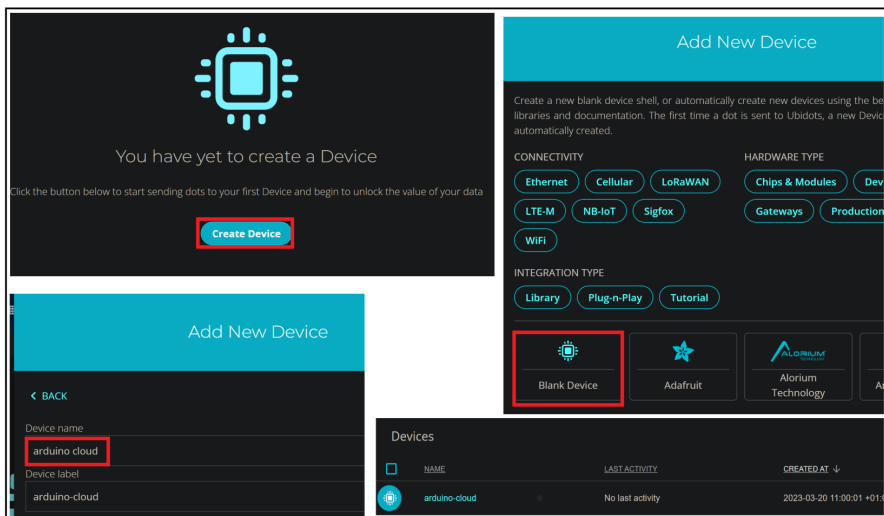
**Keywords:** Arduino, MQTT, Cloud, IoT, Communication

## 1 Configurar Arduino IoT cloud

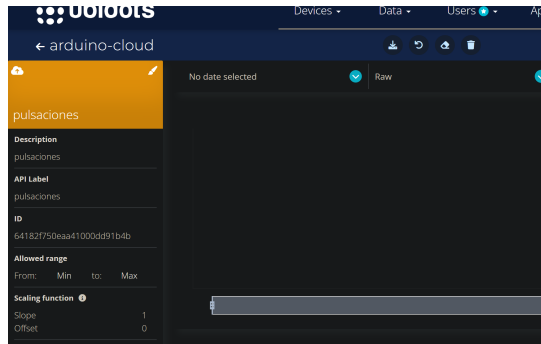
1. Crea una cuenta en el portal de Ubidots stem.



2. Conecta tu dispositivo Arduino 33 IoT a la plataforma Cloud.
  - (a) Crea un dispositivo.

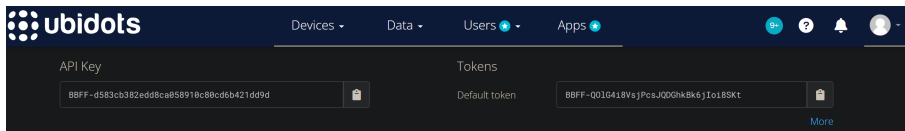


- (b) Crea una variable denominada “pulsaciones” de tipo entero, con permiso “read write” y Actualización “On charge”.



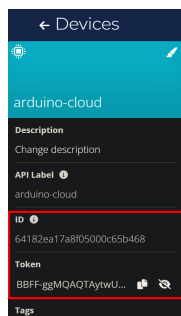
- (c) Crea un dispositivo y guarda las credenciales que te ofrece la plataforma.

### Credenciales plataforma:



- 1 API Key = BBFF-d583cb382edd8ca058910c80cd6b421dd9d
- 2 Default token = BBFF-Q0IG4i8VsjPcsJQDGhkBk6jIoi8SKt

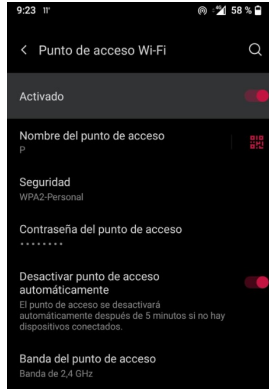
### Credenciales dispositivo:



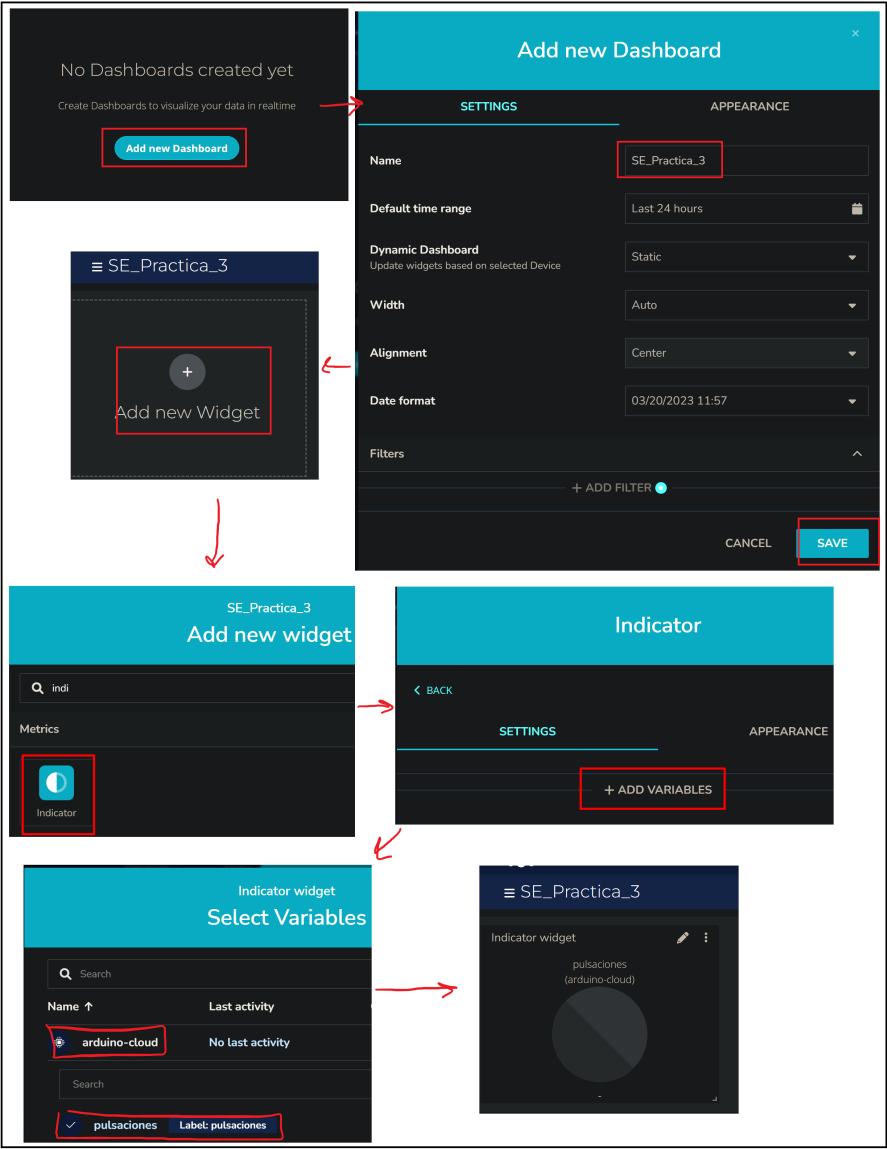
- 1 Device ID = 64182ea17a8f05000c65b468
- 2 Token = BBFF-ggMQAQTAytwU02lJlmwpBOCohONGdK

#### 4 *Conectividad en dispositivos embebidos*

- (d) Establece la conexión “Network” a la Wifi de tu dispositivo móvil o Wifi de casa. En la Universidad es probable que necesites generar una red wifi para poder conectar el dispositivo.



- (e) Crea un cuadro de mando “Dashboard” que muestra en un panel el valor discreto de la variable “pulsaciones”.

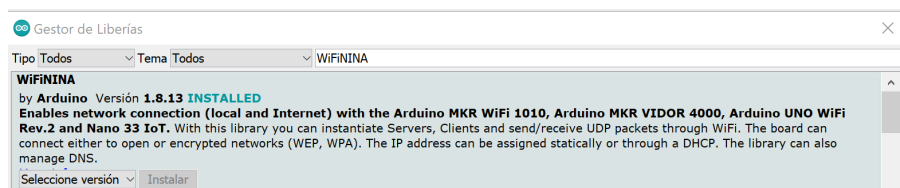


- (f) Crea un código para el dispositivo que se conecte a la red wifi, utiliza la librería `arduino-libraries/WiFiNINA`, y usando las credenciales facilitadas por el portal envíe un entero aleatorio entre 60 y 120 que representa la media de las pulsaciones. Haz que esté mensaje se envíe cada 5 segundos con un valor distinto. Para este ejemplo utiliza la documentación de API REST facilitada por la plataforma. <https://docs.ubidots.com/v1.6/reference/welcome>

## Instalo librerías:

**Arduino SAMD Boards (32-bits ARM Cortex-M0+)**  
 by **Arduino** versión **1.8.12** **INSTALLED**  
 Tarjetas incluidas en este paquete  
 Arduino MKR WiFi 1010, Arduino Zero, Arduino MKR 1000, Arduino MKR Zero, Arduino MKR FOX 1200, Arduino MKR WAN 1300, Arduino MKR WAN 1310, Arduino MKR GSM 1400, Arduino MKR NB 1500, Arduino MKR Vidor 4000, Arduino Nano 33 IoT, Arduino M0 Pro, Arduino M0, Arduino Tian, Adafruit Circuit Playground Express.  
[Online Help](#)  
[More Info](#)

**WiFiNINA\_Generic**  
 by **Khoi Hoang** Versión **1.8.15-1** **INSTALLED**  
**Enables network connection (local and Internet) and WiFiStorage for SAM DUE, SAMD21, SAMD51, Teensy, AVR (328P, 32u4, 16u4, etc.), Mega, STM32F/L/H/G/WB/MP1, nRF52, NINA\_B302\_ublox, NINA\_B112\_ublox, RP2040-based boards, etc. in addition to Arduino MKR WiFi 1010, Arduino MKR VIDOR 4000, Arduino UNO WiFi Rev.2, Nano 33 IoT, Nano RP2040 Connect. Now with fix of severe limitation to permit sending much larger data than total 4K** With this library you can instantiate Servers, Clients and send/receive TCP/UDP packets through WiFiNINA. The board can connect either to open or encrypted networks (WEP, WPA). The IP address can be assigned statically or through a DHCP. The library can also manage DNS.  
[More info](#)



Gestor de Librerías

Tipo Todos Tema Todos WiFiNINA

**WiFiNINA**  
 by **Arduino** Versión **1.8.13** **INSTALLED**  
**Enables network connection (local and Internet) with the Arduino MKR WiFi 1010, Arduino MKR VIDOR 4000, Arduino UNO WiFi Rev.2 and Nano 33 IoT.** With this library you can instantiate Servers, Clients and send/receive UDP packets through WiFi. The board can connect either to open or encrypted networks (WEP, WPA). The IP address can be assigned statically or through a DHCP. The library can also manage DNS.  
 Seleccione versión Instalar

**ArduinoHttpClient**  
 by **Arduino** Versión **0.4.0** **INSTALLED**  
**[EXPERIMENTAL] Easily interact with web servers from Arduino, using HTTP and WebSocket's.** This library can be used for HTTP (GET, POST, PUT, DELETE) requests to a web server. It also supports exchanging messages with WebSocket servers. Based on Adrian McEwen's HttpClient library.  
[More info](#)

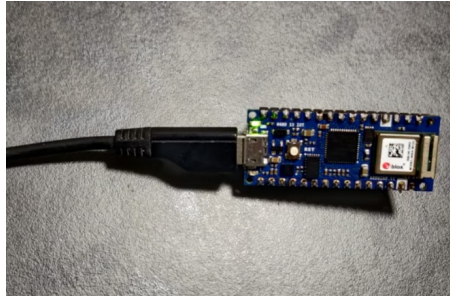
### Creo el código:

```

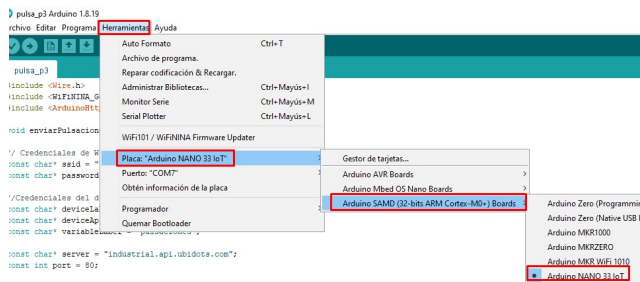
1 #include <Wire.h>
2 #include <WiFiNINA_Generic.h>
3 #include <ArduinoHttpClient.h>
4 void enviarPulsaciones(int);
5
6 // Credenciales de WiFi
7 const char* ssid = "P";
8 const char* password = "sd123456";
9
10 //Credenciales del dispositivo Ubidots
11 const char* device_api = "BBFF-ggMQAQTAytwU02IJ1mwpBOCohONGdK";
12 const char* token = "BBFF-QOlG4i8VsJPcsJQDGhkBk6jIoi8SKt";
13 const char* variableid = "64182f750eaa41000dd91b4b";
14
15 const char* server = "industrial.api.ubidots.com";
16 const int port = 80;
17
18 WiFiClient wifiClient;
19 HttpClient client = HttpClient(wifiClient, server, port);
20
21 void setup() {
22   Serial.begin(9600);
23   while (WiFi.begin(ssid, password) != WL_CONNECTED) {
24     Serial.print("Conectando a ");
25     Serial.print(ssid);
26     Serial.println("...");
27     delay(5000);
28   }
29   Serial.println(" Conectado a la red wifi!");
30 }
31
32 void loop() {
33   int pulsaciones = random(60, 121);
34   enviarPulsaciones(pulsaciones);
35   delay(5000);
36 }
37 void enviarPulsaciones(int pulsaciones) {
38   Serial.print("Enviando pulsaciones: ");
39   Serial.println(pulsaciones);
40
41   String url = "/api/v1.6/variables/" + String(variableid) + "/" +
42     "values";
43   String postData = "token=" + String(device_api) + "&value=" +
44     pulsaciones;
45
46   client.beginRequest();
47   client.post(url.c_str());
48   client.setHeader("X-Auth-Token", token);
49   client.setHeader("Content-Type", "application/x-www-form-
50     urlencoded");
51   client.setHeader("Content-Length", postData.length());
52   client.beginBody();
53   client.print(postData);
54   client.endRequest();
55
56   int statusCode = client.responseStatusCode();
57   String response = client.responseBody();
58
59   if (statusCode == 200 || statusCode == 201) { Serial.println("[
60     Datos enviados correctamente.]");
61   } else { Serial.println("[Error al enviar los datos.]");}
62
63   Serial.println(response);
64
65   client.stop();
66 }

```

## Conecto el dispositivo:



## Selecciono nano 33 IOT:



## Compilo y cargo el programa:

```

El Sketch usa 24644 bytes (9%) del espacio de almacenamiento de programa. El máximo es 262144 bytes.
Las variables Globales usan 4892 bytes (14%) de la memoria dinámica, dejando 27876 bytes para las variables locales. El máximo es 32768 bytes.
Atmel SMART device 0x10010005 found
Device       : ATSAM21G18A
Chip ID      : 10010005
Version      : v2.0 [Arduino:XYZ] Apr 19 2019 14:38:40
Address      : 0192
Pages        : 3968
Page Size    : 64 bytes
Total Size   : 248KB
Planes       : 1
Lock Regions : 16
Locked       : none
Security     : false
Boot Flash   : true
BOD          : true
BOR          : true
Arduino      : FAST_CHIP_ERASE
Arduino      : FAST_MULTI_PAGE_WRITE
Arduino      : CAN_CHECKSUM_MEMORY_BUFFER
Erase flash
done in 0.860 seconds

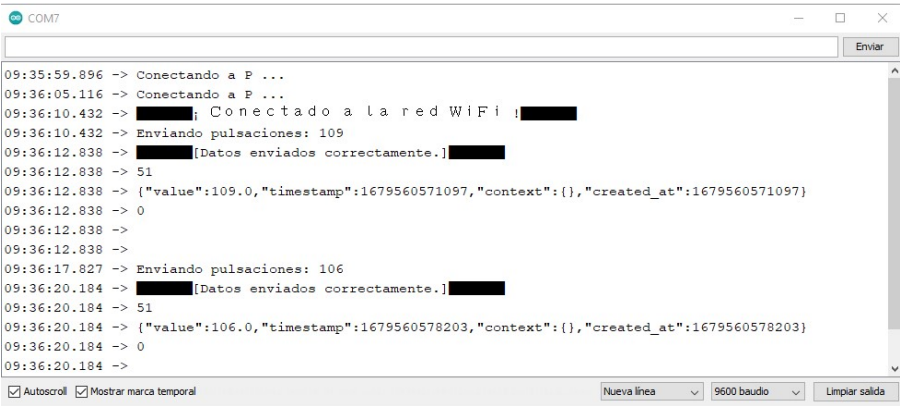
Write 24644 bytes to flash (386 pages)
[=====] 100% (386/386 pages)
done in 0.205 seconds

Verify 24644 bytes of flash with checksum.
Verify successful
done in 0.010 seconds
CPU reset.

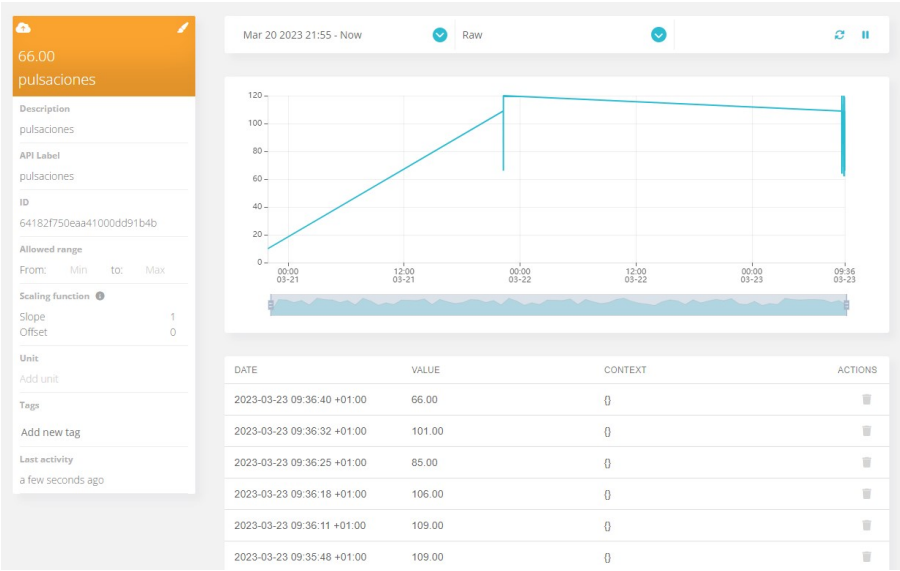
```



Abro el 'Monitor serie':



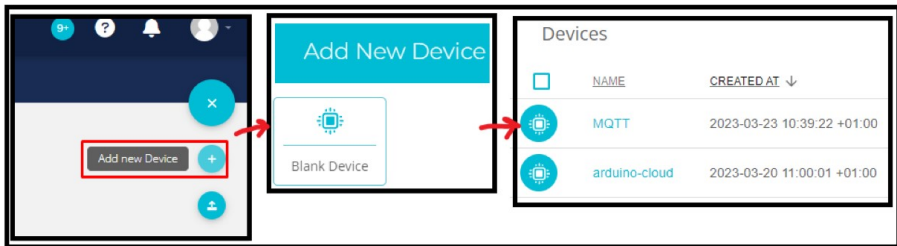
3. Observa los cambios en el valor de la variable en el “Dashboard” a lo largo del tiempo. Añade un control de tipo “chart” en el “Dashboard” para observar de forma gráfica esta evolución.



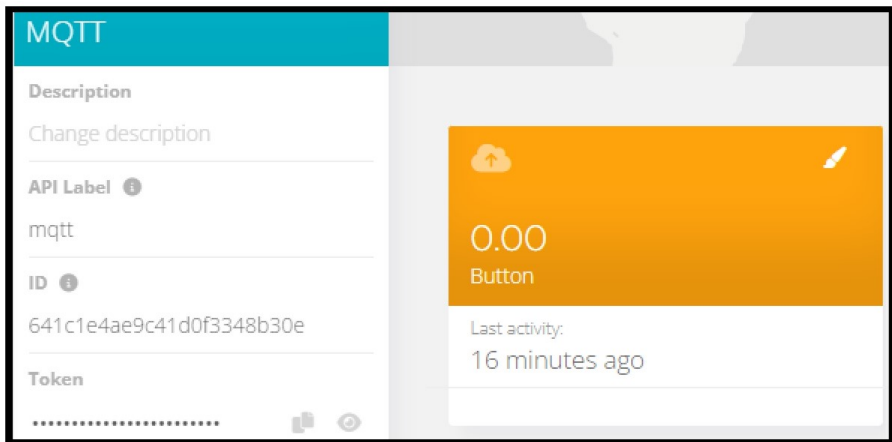
## 2 Enviar y recibir datos por MQTT

1. Crea un botón en el portal de la plataforma y añade el comportamiento para cambiar el valor numérico cuando se pulsa el botón, esto hará que se envíe un mensaje al topic de MQTT correspondiente.

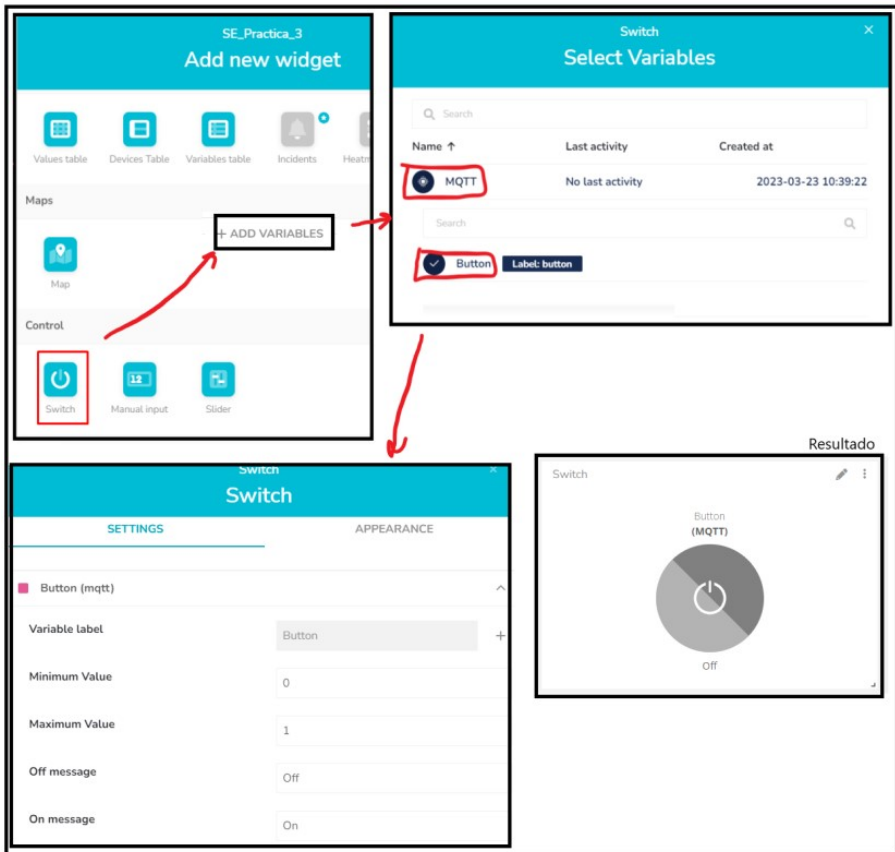
Creo un nuevo device para este ejercicio, llamado 'mqtt':



Creo la variable del boton, llamada 'button':



Creo el widget en el dashboard:



Una vez que se haya creado todo lo anterior, Ubidots STEM crea automáticamente un productor MQTT en el tópico ['/v1.6/devices/mqtt/button/lv'](#). Cada vez que se cambie el valor del botón de 'on' a 'off' o viceversa, se enviará un '1.0' o un '0.0' respectivamente a través de ese tópico.

2. Crea un código para el dispositivo que una vez conectada a la red wifi se conecte al servidor de MQTT de ubidots y haz que se suscriba al topic correspondiente para esperar los mensajes del servidor.

### Instalo:



### Creo el código:

```

1  #include <Wire.h>
2  #include <WiFiNINA_Generic.h>
3  #include <PubSubClient.h>
4
5  void callback(char*,byte*,unsigned int);
6
7  // Credenciales de WiFi
8  const char* ssid = "P";
9  const char* password = "sd123456";
10
11 //Credenciales de cuenta ubidots
12 const char* ubidots_token = "BBFF-QOIG4i8VsJPcsJQDGhkBk6jIoi8SKt";
13 const char* ubidots_password = "@midad2133";
14
15 //Conexion con ubidots
16 const char* mqttServer = "industrial.api.ubidots.com";
17 const int mqttPort = 1883;
18
19 WiFiClient wifiClient;
20 PubSubClient client(wifiClient);
21
22 void setup() {
23   Serial.begin(9600);
24
25   //CONEXION AL WIFI
26   while (WiFi.begin(ssid, password) != WL_CONNECTED) {
27     Serial.print("Conectando a ");
28     Serial.print(ssid);
29     Serial.println(" ...");
30     delay(5000);
31   }
32   Serial.println(" Conectado a la red wifi correctamente!");
33
34   //CONEXION AL BROKER MQTT
35   client.setServer(mqttServer, mqttPort);
36   client.setCallback(callback);
37
38   while (!client.connected()) {
39     Serial.println("Connecting to MQTT...");
40     if (client.connect("ESP32", ubidots_token, ubidots_password)) {
41       Serial.println("Connected to MQTT");
42       client.subscribe("/v1.6/devices/mqtt/button/lv");
43     } else {
44       Serial.print("Failed with state ");
45       Serial.print(client.state());

```

```

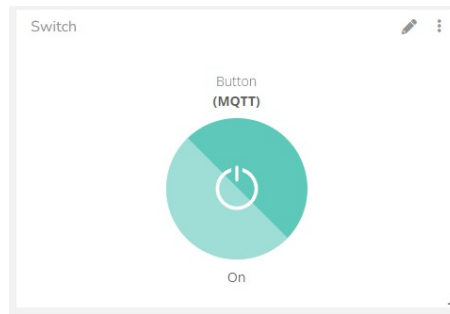
46     delay(2000);
47   }
48 }
49 }
50
51 void loop() { client.loop();}
52
53 void callback(char* topic, byte* payload, unsigned int length){
54   Serial.print("Ha llegado mensaje en el topic");
55   Serial.print(topic);
56   Serial.print("]. Mensaje: ");
57   for(int i=0; i<length; i++){ Serial.print((char)payload[i]);}
58   Serial.println(" ");
59 }

```

Conecto el dispositivo y le cargo el programa.

### Resultados:

Ahora cada vez que apago o desconecto el botón recibo por el topic '1' o '0' dependiendo si está activado o desactivado el botón



```

COM7
17:51:20.150 -> Conectando a P ...
17:51:25.325 -> Conectando a P ...
17:51:30.647 -> ¡Conectado a la red wifi correctamente!
17:51:30.647 -> Connecting to MQTT...
17:51:31.638 -> Connected to MQTT
17:51:32.013 -> Ha llegado mensaje en el topic[/v1.6/devices/mqtt/button/lv]. Mensaje: 0.0
17:51:52.526 -> Ha llegado mensaje en el topic[/v1.6/devices/mqtt/button/lv]. Mensaje: 1.0
17:51:55.397 -> Ha llegado mensaje en el topic[/v1.6/devices/mqtt/button/lv]. Mensaje: 0.0
17:51:56.804 -> Ha llegado mensaje en el topic[/v1.6/devices/mqtt/button/lv]. Mensaje: 1.0
17:51:57.841 -> Ha llegado mensaje en el topic[/v1.6/devices/mqtt/button/lv]. Mensaje: 0.0

```

### 3 Prueba de la función BLE

1. Carga a tu entorno la biblioteca “ArduinoBLE”.



2. Modo emisor: Carga en el entorno el ejemplo “ArduinoBLE/Peripheral/-BatteryMonitor”. Conecta el teléfono móvil con el dispositivo mediante Bluetooth y observa la información suministrada.

#### Cargo el ejemplo 'BatteryMonitor':

```

1 #include <ArduinoBLE.h>
2
3 // Bluetooth Low Energy Battery Service
4 BLEService batteryService("180F");
5
6 // Bluetooth Low Energy Battery Level Characteristic
7 BLEUnsignedCharCharacteristic batteryLevelChar("2A19", // standard
8   16-bit characteristic UUID
9   BLERead | BLENotify); // remote clients will be able to get
10   notifications if this characteristic changes
11
12 int oldBatteryLevel = 0; // last battery level reading from analog
13   input
14 long previousMillis = 0; // last time the battery level was checked
15   , in ms
16
17 void setup() {
18   delay(3000);
19   Serial.begin(9600); // initialize serial communication
20   while (!Serial);
21
22   pinMode(LED_BUILTIN, OUTPUT); // initialize the built-in LED pin
23   to indicate when a central is connected
24
25   // begin initialization
26   if (!BLE.begin()) {Serial.println("starting BLE failed!"); while
27     (1);}
28   else{Serial.println("starting BLE succesful!");}
29
30   BLE.setLocalName("BatteryMonitor");
31   BLE.setAdvertisedService(batteryService); // add the service UUID
32   batteryService.addCharacteristic(batteryLevelChar); // add the
33   battery level characteristic
34   BLE.addService(batteryService); // Add the battery service
35   batteryLevelChar.writeValue(oldBatteryLevel); // set initial value
36   for this characteristic
37
38   BLE.advertise();
39
40   Serial.println("Bluetooth device active, waiting for connections
41     ...");
42 }

```

```

35 void loop() {
36
37     // wait for a Bluetooth Low Energy central
38     BLEDevice central = BLE.central();
39
40     // if a central is connected to the peripheral:
41     if (central) {
42         Serial.print("Connected to central: ");
43         // print the central's BT address:
44         Serial.println(central.address());
45         // turn on the LED to indicate the connection:
46         digitalWrite(LED_BUILTIN, HIGH);
47
48         // check the battery level every 200ms
49         // while the central is connected:
50         while (central.connected()) {
51             long currentMillis = millis();
52             // if 200ms have passed, check the battery level:
53             if (currentMillis - previousMillis >= 200) {
54                 previousMillis = currentMillis;
55                 updateBatteryLevel();
56             }
57         }
58         // when the central disconnects, turn off the LED:
59         digitalWrite(LED_BUILTIN, LOW);
60         Serial.print("Disconnected from central: ");
61         Serial.println(central.address());
62     }
63 }
64
65 void updateBatteryLevel() {
66     int battery = analogRead(A0);
67     int batteryLevel = map(battery, 0, 1023, 0, 100);
68
69     if (batteryLevel != oldBatteryLevel) { // if the battery
70         level has changed
71         Serial.print("Battery Level % is now: "); // print it
72         Serial.println(batteryLevel);
73         batteryLevelChar.writeValue(batteryLevel); // and update the
74         battery level characteristic
75         oldBatteryLevel = batteryLevel; // save the level for
76         next comparison
77     }
78 }

```

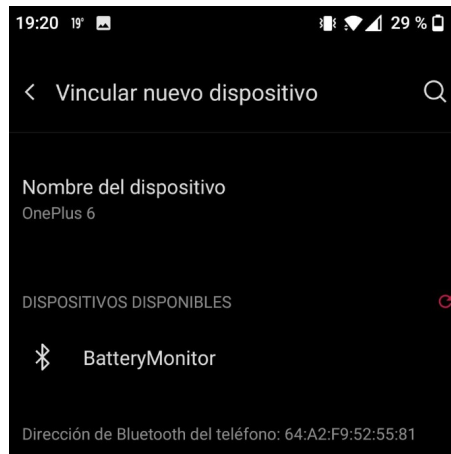
**Activo el monitor:**

```

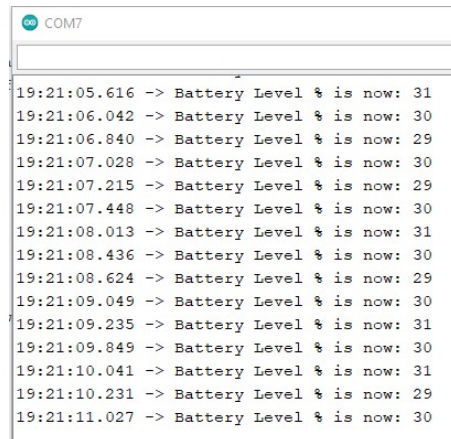
COM7
19:18:51.461 -> Bluetooth® device active, waiting for connections...
19:19:22.412 -> Bluetooth® device active, waiting for connections...

```

Voy a mi móvil y busco el bluetooth:



Me conecto y abro el monitor para ver el nivel de batería de mi móvil:





3. Modo monitor: Carga en el entorno el ejemplo “ArduinoBLE/Central/Scan”. Observa en la consola los dispositivos que se detectan y la información obtenida de ellos. Observa la frecuencia con la que se detectan.

### Carga el ejemplo 'Scan':

```

1 #include <ArduinoBLE.h>
2
3 void setup() {
4   delay(3000);
5   Serial.begin(9600);
6   while (!Serial);
7
8   // begin initialization
9   if (!BLE.begin()) {
10    Serial.println("starting Bluetooth Low Energy module failed!");
11    ;
12    while (1);
13  }
14
15  Serial.println("Bluetooth Low Energy Central scan");
16
17  // start scanning for peripheral
18  BLE.scan();
19 }
20
21 void loop() {
22   // check if a peripheral has been discovered
23   BLEDevice peripheral = BLE.available();
24
25   if (peripheral) {
26     // discovered a peripheral
27     Serial.println("Discovered a peripheral");
28     Serial.println("_____");
29
30     // print address
31     Serial.print("Address: ");
32     Serial.println(peripheral.address());
33
34     // print the local name, if present
35     if (peripheral.hasLocalName()) {
36       Serial.print("Local Name: ");
37       Serial.println(peripheral.localName());
38     }
39
40     // print the advertised service UUIDs, if present
41     if (peripheral.hasAdvertisedServiceUuid()) {
42       Serial.print("Service UUIDs: ");
43       for (int i = 0; i < peripheral.advertisedServiceUuidCount(); i
44       ++){
45         Serial.print(peripheral.advertisedServiceUuid(i));
46         Serial.print(" ");
47       }
48       Serial.println();
49     }
50
51     // print the RSSI
52     Serial.print("RSSI: ");
53     Serial.println(peripheral.rssi());
54
55     Serial.println();
56   }
57 }

```

Abro el monitor y veo los dispositivos que va encontrando:

```
COM7

19:46:01.474 -> Bluetooth® Low Energy Central scan
19:50:16.712 -> Discovered a peripheral
19:50:16.712 -> -----
19:50:16.712 -> Address: 7b:1e:a1:20:6c:34
19:50:16.712 -> RSSI: -92
19:50:16.712 ->
19:50:24.460 -> Discovered a peripheral
19:50:24.460 -> -----
19:50:24.460 -> Address: d6:b0:2c:61:6d:ca
19:50:24.460 -> RSSI: -93
19:50:24.460 ->
```

## 4 Reemplaza en tu prototipo el Arduino UNO usado en la práctica 2 por el dispositivo Arduino nano IoT y verifica que las conexiones son correctas teniendo en cuenta el pinout del nuevo dispositivo.

1. Modifica el código utilizado para la práctica 2 para que se conecte a una red wifi. Realiza también las modificaciones necesarias para que el dispositivo envíe cada 5 segundos los datos del sensor a la plataforma mediante MQTT.

### Código:

```

1  #include <Wire.h>
2  #include "MAX30105.h"
3  #include <WiFiNINA_Generic.h>
4  #include <ArduinoHttpClient.h>
5  #include <PubSubClient.h>
6  #include "heartRate.h"
7  #include <Adafruit_SSD1306.h>
8  #include <string>
9
10 #define SCREEN_WIDTH 128 // OLED display width, in pixels
11 #define SCREEN_HEIGHT 32 // OLED display height, in pixels
12 #define OLED_RESET      -1
13
14 MAX30105 particleSensor;
15 Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
    OLED_RESET);
16
17 const byte RATE_SIZE = 4; //Increase this for more averaging. 4 is
    good.
18 byte rates[RATE_SIZE]; //Array of heart rates
19 byte rateSpot = 0;
20 long lastBeat = 0; //Time at which the last beat occurred
21
22 float beatsPerMinute;
23 int beatAvg;
24
25 //Pantalla
26 int TextSize=1; //tamaño del texto a imprimir
27 bool TextInverted=false; //false=Texto normal, true=Texto invertido
28 String Texto; //Texto a imprimir
29 int TextX; //Ubicación en X del texto
30 int TextY; //Ubicación en Y del texto
31 bool ClearScreen=false; //Limpiar pantalla
32 String Linea1;
33 String Linea2;
34
35 int led = 13;
36
37 //AMPLIACIÓN PRÁCTICA 3
38 // mqtt
39 void suscribeMQTT(char*);
40 void callback(char*,byte*,unsigned int);
41 // Credenciales de WiFi
42 void conectarWifi();
43 const char* ssid = "P";
44 const char* password = "sd123456";
45 //Credenciales de cuenta ubidots
46 const char* ubidots_token = "BBFF-QOIG4i8VsJpCsJQDGhkBk6jIoi8SKt";
47 const char* ubidots_password = "@midad2133";
48 WiFiClient wifiClient;

```

```

49 PubSubClient client(wifiClient); //mqtt
50 const char* mqttServer = "industrial.api.ubidots.com";
51 const int mqttPort = 1883;
52
53 void setup()
54 {
55     delay(3000);
56     Serial.begin(9600);
57     Serial.println("Programa iniciado.");
58
59     conectarWifi();
60     suscribirMQTT((char *)"/v1.6/devices/mqtt/button/lv");
61
62     if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3C
63         for 128x32
64         Serial.print("SSD1306 allocation failed");
65         for(;;); // Don't proceed, loop forever
66     }
67
68     display.display(); //Requerido por Adafruit Copywrite
69     delay(2000); // Pause for 2 seconds
70     display.clearDisplay();
71
72     // Initialize sensor
73     if (!particleSensor.begin(Wire, I2C_SPEED_FAST)) //Use default I2C
74         port, 400kHz speed
75     {
76         Serial.println("MAX30105 was not found. Please check wiring/
77         power. ");
78         while (1);
79     }
80     particleSensor.setup(); //Configure sensor with default settings
81     particleSensor.setPulseAmplitudeRed(0x0A); //Turn Red LED to low
82     to indicate sensor is running
83     particleSensor.setPulseAmplitudeGreen(0); //Turn off Green LED
84
85     display.clearDisplay();
86     PrintText(1,4,3,"Inicializando...");
87     display.display();
88     pinMode(led,OUTPUT);
89     digitalWrite(led,HIGH);
90     delay(3000);
91     digitalWrite(led,LOW);
92     delay(500);
93 }
94
95 void loop()
96 {
97     client.loop();
98
99     printErrorDEDO();
100     long irValue = particleSensor.getIR();
101
102     while(beatsPerMinute < 70){
103         irValue = particleSensor.getIR();
104
105         if (checkForBeat(irValue) == true)
106         {
107             //We sensed a beat!
108             long delta = millis() - lastBeat;
109             lastBeat = millis();
110
111             beatsPerMinute = 60 / (delta / 1000.0);
112
113             if (beatsPerMinute < 255 && beatsPerMinute > 20)
114             {
115                 rates[ratesSpot++] = (byte)beatsPerMinute; //Store this
116                 reading in the array

```

```

112     rateSpot %= RATE_SIZE; //Wrap variable
113
114     //Take average of readings
115     beatAvg = 0;
116     for (byte x = 0 ; x < RATE_SIZE ; x++)
117         beatAvg += rates[x];
118     beatAvg /= RATE_SIZE;
119 }
120 }
121 }
122
123 if(irValue < 50000)
124 {
125     Linea1="ERROR DEDO";
126     Linea2=" ";
127 }
128 else
129 {
130     Linea1="BPM = " + String (int(beatPerMinute));
131     Linea2="Avg = " + String(beatAvg);
132 }
133
134 display.clearDisplay();
135 PrintText(1,4,3,Linea1);
136 PrintText(1,4,18, Linea2);
137 //display.drawRect(1, 1, 126,31, WHITE);
138 display.display();
139
140 //LED
141 if (int(beatPerMinute) < 70){ //FIJO
142     digitalWrite(led,HIGH);
143     delay(5000); //delay 5 segundos
144     digitalWrite(led,LOW);
145 } else{
146     if(int(beatPerMinute) >= 70 && int(beatPerMinute) < 110){ //
147         PARDPADEO CADA SEGUNDO EN 5 SEGUNDOS
148         parpadeo(1000); //2s
149         parpadeo(1000); //2s
150         parpadeoHIGH(1000); //1s
151         digitalWrite(led,LOW);
152     } else{
153         if(int(beatPerMinute) > 110){ //PARDPADO CADA MEDIO SEGUNDO
154             EN 5 SEGUNDOS
155             parpadeo(500); //1s
156             parpadeo(500); //1s
157             parpadeo(500); //1s
158             parpadeo(500); //1s
159             parpadeo(500); //1s
160         }
161     }
162 }
163
164 //P3: Envio los datos del sensor (ya que han pasado los 5 segundos
165 //por los delays del 'if' anterior)
166 if (client.publish((char *)"/v1.6/devices/mqtt/received-text", (
167     std::string("{\n" value \n:}") + std::to_string(beatPerMinute) + "
168     })).c_str())) {
169     Serial.println((std::string("Valor ") + std::to_string(
170         beatPerMinute) + " publicado correctamente.")).c_str());
171 }
172
173     beatPerMinute=0;
174 }
175
176
177 //FUNCIONES////////////////////////////////////
178 void suscribeMQTT(char* topic){
179     client.setServer(mqttServer, mqttPort);
180     client.setCallback(callback);

```

```

174
175 while (!client.connected()) {
176   Serial.println("Connecting to MQTT...");
177   if (client.connect("ESP32", ubidots_token, ubidots_password)) {
178     Serial.println("Connected to MQTT");
179     client.subscribe(topic);
180   } else {
181     Serial.print("Failed with state ");
182     Serial.print(client.state());
183     delay(2000);
184   }
185 }
186 }
187
188 void conectarWifi(){
189   while (WiFi.begin(ssid, password) != WL_CONNECTED) {
190     Serial.print("Conectando a ");
191     Serial.print(ssid);
192     Serial.println(" ...");
193     delay(5000);
194   }
195   Serial.println(" Conectado a la red wifi correctamente!");
196 }
197
198 //MQTT
199 void callback(char* topic, byte* payload, unsigned int length){
200   Serial.print("Ha llegado mensaje en el topic");
201   Serial.print(topic);
202   Serial.print("]. Mensaje: ");
203   for(int i=0; i<length; i++){ Serial.print((char)payload[i]);}
204   Serial.println(" ");
205 }
206
207 //Imprimir Text en pantalla
208 void PrintText(int Size, int X, int Y, String Texto)
209 {
210   display.setTextSize(Size);
211   display.setTextColor(WHITE);
212   display.setCursor(X,Y);
213   display.println(Texto);
214 }
215
216
217 void printErrorDEDO(){
218   display.clearDisplay();
219   PrintText(1,4,3,"ERROR DEDO");
220   digitalWrite(led,HIGH);
221   display.display();
222 }
223
224 void parpadeo(int ms){
225   parpadeoHIGH(ms);
226   parpadeoLOW(ms);
227 }
228
229 void parpadeoHIGH(int ms){
230   digitalWrite(led,HIGH);
231   delay(ms);
232 }
233
234 void parpadeoLOW(int ms){
235   digitalWrite(led,LOW);
236   delay(ms);
237 }

```

## Resultado publicado en ubidots stem:

```
COM11
Programa iniciado.
Conectando a P ...
Conectando a P ...
¡Conectado a la red wifi correctamente!
Connecting to MQTT...
Connected to MQTT
Valor 84.985832 publicado correctamente.
Ha llegado mensaje en el topic[/v1.6/devices/]
Valor 90.361443 publicado correctamente.
```



- Modifica el código y añade la conectividad BLE para que una vez se conecte por BLE un dispositivo al arduino se le envíe el valor del sensor cada 5 segundos.

## Código:

```
1 #include <Wire.h>
2 #include "MAX30105.h"
3 #include <WiFiNINA_Generic.h>
4 #include "heartRate.h"
5 #include <Adafruit_SSD1306.h>
6 #include <string>
7 #include <ArduinoBLE.h>
8
9 #define SCREEN_WIDTH 128 // OLED display width, in pixels
10 #define SCREEN_HEIGHT 32 // OLED display height, in pixels
11 #define OLED_RESET -1
12
13 void parpadeo(int);
14
15 MAX30105 particleSensor;
16 Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
    OLED_RESET);
```

```

17
18 const byte RATE_SIZE = 4; //Increase this for more averaging. 4 is
    good.
19 byte rates[RATE_SIZE]; //Array of heart rates
20 byte rateSpot = 0;
21 long lastBeat = 0; //Time at which the last beat occurred
22
23 float beatsPerMinute = 0;
24 int beatAvg;
25
26 //Pantalla
27 int TextSize=1; //tamaño del texto a imprimir
28 bool TextInverted=false; //false=Texto normal, true=Texto invertido
29 String Texto; //Texto a imprimir
30 int TextX; //Ubicación en X del texto
31 int TextY; //Ubicación en Y del texto
32 bool ClearScreen=false; //Limpiar pantalla
33 String Linea1;
34 String Linea2;
35
36 int led = 13;
37
38 BLEService bleService("19b10010-e8f2-537e-4f6c-d104768a1214"); //
    UUID for the BLE service
39 BLEIntCharacteristic intCharacteristic("2A37", BLERead | BLENotify);
    // Custom characteristic UUID
40
41 void activarBluetooth();
42
43 void setup()
44 {
45     delay(3000);
46     Serial.begin(9600);
47     Serial.println("Programa iniciado.");
48
49     activarBluetooth();
50
51     if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3C
        for 128x32
52         Serial.print("SSD1306 allocation failed");
53         for (;;); // Don't proceed, loop forever
54     }
55
56     display.display(); //Requerido por Adafruit Copywrite
57     delay(2000); // Pause for 2 seconds
58     display.clearDisplay();
59
60     // Initialize sensor
61     if (!particleSensor.begin(Wire, I2C_SPEED_FAST)) //Use default I2C
        port, 400kHz speed
62     {
63         Serial.println("MAX30105 was not found. Please check wiring/
            power. ");
64         while (1);
65     }
66     particleSensor.setup(); //Configure sensor with default settings
67     particleSensor.setPulseAmplitudeRed(0x0A); //Turn Red LED to low
        to indicate sensor is running
68     particleSensor.setPulseAmplitudeGreen(0); //Turn off Green LED
69
70     display.clearDisplay();
71     PrintText(1,4,3,"Inicializando...");
72     display.display();
73     pinMode(led,OUTPUT);
74     digitalWrite(led,HIGH);
75     delay(3000);
76     digitalWrite(led,LOW);
77     delay(500);

```



```

78   Serial.println("Esperando conexión bluetooth...");
79 }
80
81 bool shown = false;
82 void loop()
83 {
84   BLEDevice central = BLE.central();
85   if(central){
86     if(!shown){
87       Serial.print("Connected to central: ");
88       Serial.println(central.address());
89       digitalWrite(LED_BUILTIN, HIGH);
90       shown=true;
91     }
92
93     printErrorDEDO();
94
95     long irValue = particleSensor.getIR();
96
97     irValue = particleSensor.getIR();
98
99     if (checkForBeat(irValue) == true)
100    {
101      //We sensed a beat!
102      long delta = millis() - lastBeat;
103      lastBeat = millis();
104
105      beatsPerMinute = 60 / (delta / 1000.0);
106
107      if (beatsPerMinute < 255 && beatsPerMinute > 20)
108      {
109        rates[rateSpot++] = (byte)beatsPerMinute; //Store this
        //reading in the array
110        rateSpot %= RATE_SIZE; //Wrap variable
111
112        //Take average of readings
113        beatAvg = 0;
114        for (byte x = 0 ; x < RATE_SIZE ; x++)
115          beatAvg += rates[x];
116        beatAvg /= RATE_SIZE;
117      }
118    }
119    if(irValue <50000)
120    {
121      Linea1="ERROR DEDO";
122      Linea2="";
123    }
124    else
125    {
126      Linea1="BPM = " + String (int(beatsPerMinute));
127      Linea2="Avg = " + String(beatAvg);
128    }
129
130    display.clearDisplay();
131    PrintText(1,4,3,Linea1);
132    PrintText(1,4,18, Linea2);
133    //display.drawRect(1, 1, 126,31, WHITE);
134    display.display();
135
136    //LED
137    if (int(beatsPerMinute) < 70){ //FIJO
138      digitalWrite(led,HIGH);
139      delay(5000); //delay 5 segundos
140      digitalWrite(led,LOW);
141    } else {
142      if(int(beatsPerMinute) >= 70 && int(beatsPerMinute) < 110){ //
        PARDPADEO CADA SEGUNDO EN 5 SEGUNDOS
143        parpadeo(1000); //2s

```

```

144         parpadeo(1000); //2s
145         parpadeoHIGH(1000); //1s
146         digitalWrite(led,LOW);
147     }else{
148         if(int(beatsPerMinute) > 110){ //PARDPADEO CADA MEDIO
SEGUNDO EN 5 SEGUNDOS
149             parpadeo(500); //1s
150             parpadeo(500); //1s
151             parpadeo(500); //1s
152             parpadeo(500); //1s
153             parpadeo(500); //1s
154         }
155     }
156 }
157
158 //P4-2: Envio datos por pluetooth
159 intCharacteristic.writeValue(beatsPerMinute);
160 Serial.println("Enviado valor del sensor al dispositivo
conectado mediante BLE!");
161
162 beatsPerMinute=0;
163
164
165 }
166 }
167
168 ///FUNCIONES////////////////////////////////////
169 void activarBluetooth(){
170     if (!BLE.begin()) {
171         Serial.println("Failed to initialize BLE!");
172         while (1);
173     }
174
175     // Set the local name and advertised service UUID:
176     BLE.setLocalName("ArduinoVadim: ej4-2");
177     BLE.setAdvertisedService(bleService);
178
179     // Add the characteristic to the service:
180     bleService.addCharacteristic(intCharacteristic);
181
182     // Advertise the service:
183     BLE.addService(bleService);
184
185     intCharacteristic.writeValue(beatsPerMinute);
186
187     // Start advertising:
188     BLE.advertise();
189
190     Serial.println("Bluetooth activado!");
191 }
192
193 //Imprimir Text en pantalla
194 void PrintText(int Size, int X, int Y, String Texto)
195 {
196     display.setTextSize(Size);
197     display.setTextColor(WHITE);
198     display.setCursor(X,Y);
199     display.println(Texto);
200 }
201
202
203 void printErrorDEDO(){
204     display.clearDisplay();
205     PrintText(1,4,3,"ERROR DEDO");
206     digitalWrite(led,HIGH);
207     display.display();
208 }
209

```

```

210 void parpadeo(int ms) {
211     parpadeoHIGH(ms);
212     parpadeoLOW(ms);
213 }
214
215 void parpadeoHIGH(int ms){
216     digitalWrite(led,HIGH);
217     delay(ms);
218 }
219
220 void parpadeoLOW(int ms){
221     digitalWrite(led,LOW);
222     delay(ms);
223 }

```

Conexión bluetooth con mi telefono al arduino:



Salida monitor:

```

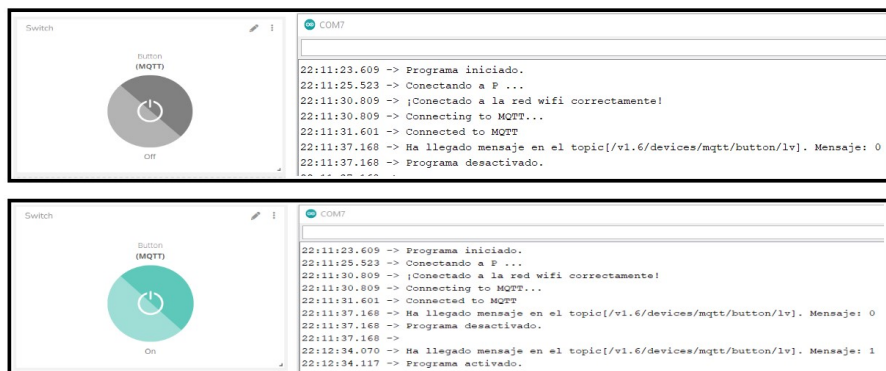
COM7
19:28:02.002 -> Programa iniciado.
19:28:02.990 -> Bluetooth activado!
19:28:08.532 -> Esperando conexión bluetooth...
19:33:32.832 -> Connected to central: 6d:41:b5:fb:73:36
19:33:37.922 -> Enviado valor del sensor al dispositivo conectado mediante BLE!
19:33:43.029 -> Enviado valor del sensor al dispositivo conectado mediante BLE!
19:33:48.131 -> Enviado valor del sensor al dispositivo conectado mediante BLE!
19:33:53.188 -> Enviado valor del sensor al dispositivo conectado mediante BLE!
19:33:58.249 -> Enviado valor del sensor al dispositivo conectado mediante BLE!
19:34:03.397 -> Enviado valor del sensor al dispositivo conectado mediante BLE!

```

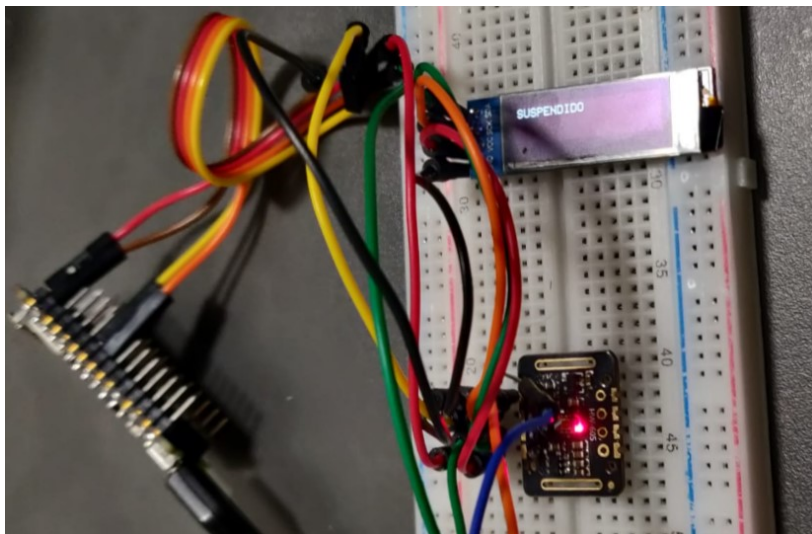
3. Modifica el código para que cuando llegue un mensaje por MQTT el dispositivo quede suspendido, esto quiere decir que no realice ninguna medición y que muestre por pantalla la palabra suspendido. Hasta que se reinicie o llegue otro mensaje.

NOTA: Para realizar esta tarea haré uso del botón creado en ubidots stem en ejercicios anteriores. La idea es que cuando el botón está en "on" deje funcionar el programa, y no cuando está en "off".

**Salida monitor:**



Cuando está en 'off':



**Codigo:**

```

1  #include <Wire.h>
2  #include "MAX30105.h"
3  #include <WiFiNINA_Generic.h>
4  #include <ArduinoHttpClient.h>
5  #include <PubSubClient.h>
6  #include "heartRate.h"
7  #include <Adafruit_SSD1306.h>
8  #include <string>
9  #include <ArduinoBLE.h>
10 #include <sstream>
11
12 #define SCREEN_WIDTH 128 // OLED display width, in pixels
13 #define SCREEN_HEIGHT 32 // OLED display height, in pixels
14 #define OLED_RESET -1
15
16 MAX30105 particleSensor;
17 Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
    OLED_RESET);
18
19 const byte RATE_SIZE = 4; //Increase this for more averaging. 4 is
    good.
20 byte rates[RATE_SIZE]; //Array of heart rates
21 byte rateSpot = 0;
22 long lastBeat = 0; //Time at which the last beat occurred
23
24 float beatsPerMinute;
25 int beatAvg;
26
27 //Pantalla
28 int TextSize=1; //tamaño del texto a imprimir
29 bool TextInverted=false; //false=Texto normal, true=Texto invertido
30 String Texto; //Texto a imprimir
31 int TextX; //Ubicación en X del texto
32 int TextY; //Ubicación en Y del texto
33 bool ClearScreen=false; //Limpiar pantalla
34 String Linea1;
35 String Linea2;
36
37 int led = 13;
38
39 //AMPLIACIÓN PRÁCTICA 3
40 // mqtt
41 void suscribeMQTT(char*);
42 void callback(char*, byte*, unsigned int);
43 // Credenciales de WiFi
44 void conectarWifi();
45 const char* ssid = "P";
46 const char* password = "sd123456";
47 //Credenciales de cuenta ubidots
48 const char* ubidots_token = "BBFF-QOIG4i8VsjPcsJQDGhkBk6jIoi8SKt";
49 const char* ubidots_password = "@midad2133";
50 WiFiClient wifiClient;
51 PubSubClient client(wifiClient); //mqtt
52 const char* mqttServer = "industrial.api.ubidots.com";
53 const int mqttPort = 1883;
54
55 BLEService bleService("19b10010-e8f2-537e-4f6c-d104768a1214"); //
    UUID for the BLE service
56 BLECharacteristic randomChar("19b10011-e8f2-537e-4f6c-d104768a1214",
    BLERead | BLENotify, 2); // UUID for the BLE characteristic,
    2 bytes
57
58 void activarBluetooth();
59
60 bool puedeContinuar=false;
61
62 void setup()

```

```

63 {
64     delay(3000);
65     Serial.begin(9600);
66     Serial.println("Programa iniciado.");
67
68     conectarWifi();
69     suscribeMQTT(("char *")"/v1.6/devices/mqtt/button/lv");
70
71     if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3C
72         for 128x32
73         Serial.print("SSD1306 allocation failed");
74         for(;;); // Don't proceed, loop forever
75     }
76
77     display.display(); //Requerido por Adafruit Copywrite
78     delay(2000); // Pause for 2 seconds
79     display.clearDisplay();
80
81     // Initialize sensor
82     if (!particleSensor.begin(Wire, I2C_SPEED_FAST)) //Use default I2C
83         port, 400kHz speed
84     {
85         Serial.println("MAX30105 was not found. Please check wiring/
86         power. ");
87         while (1);
88     }
89     particleSensor.setup(); //Configure sensor with default settings
90     particleSensor.setPulseAmplitudeRed(0x0A); //Turn Red LED to low
91     to indicate sensor is running
92     particleSensor.setPulseAmplitudeGreen(0); //Turn off Green LED
93
94     display.clearDisplay();
95     PrintText(1,4,3,"Inicializando...");
96     display.display();
97     pinMode(led,OUTPUT);
98     digitalWrite(led,HIGH);
99     delay(3000);
100    digitalWrite(led,LOW);
101    delay(500);
102 }
103
104 void loop()
105 {
106     client.loop();
107
108     if(puedeContinuar){
109         printErrorDEDO();
110
111         long irValue = particleSensor.getIR();
112
113         irValue = particleSensor.getIR();
114
115         if (checkForBeat(irValue) == true)
116         {
117             //We sensed a beat!
118             long delta = millis() - lastBeat;
119             lastBeat = millis();
120
121             beatsPerMinute = 60 / (delta / 1000.0);
122
123             if (beatsPerMinute < 255 && beatsPerMinute > 20)
124             {
125                 rates[rateSpot++] = (byte)beatsPerMinute; //Store this
126                 reading in the array
127                 rateSpot %= RATE_SIZE; //Wrap variable
128
129                 //Take average of readings

```

```

126     beatAvg = 0;
127     for (byte x = 0 ; x < RATE_SIZE ; x++)
128         beatAvg += rates[x];
129     beatAvg /= RATE_SIZE;
130 }
131 }
132
133 if (irValue < 50000){Linea1="ERROR DEDO" ;Linea2="";}
134 else{Linea1="BPM = " + String (int (beatsPerMinute)); Linea2="Avg
    = " + String (beatAvg);}
135
136 display.clearDisplay();
137 PrintText(1,4,3,Linea1);
138 PrintText(1,4,18, Linea2);
139 //display.drawRect(1, 1, 126,31, WHITE);
140 display.display();
141
142 //LED
143 if (int (beatsPerMinute) < 70){ //FIJO
144     digitalWrite(led,HIGH);
145     delay(5000); //delay 5 segundos
146     digitalWrite(led,LOW);
147 } else{
148     if(int (beatsPerMinute) >= 70 && int (beatsPerMinute) < 110){ //
PARDPADEO CADA SEGUNDO EN 5 SEGUNDOS
149         parpadeo(1000); //2s
150         parpadeo(1000); //2s
151         parpadeoHIGH(1000); //1s
152         digitalWrite(led,LOW);
153     } else{
154         if(int (beatsPerMinute) > 110){ //PARDPADEO CADA MEDIO
SEGUNDO EN 5 SEGUNDOS
155             parpadeo(500); //1s
156             parpadeo(500); //1s
157             parpadeo(500); //1s
158             parpadeo(500); //1s
159             parpadeo(500); //1s
160         }
161     }
162 }
163
164 beatsPerMinute=0;
165 }
166 }
167
168 ///FUNCIONES////////////////////////////////////
169 void suscribeMQTT(char* topic){
170     client.setServer(mqttServer, mqttPort);
171     client.setCallback(callback);
172
173     while (!client.connected()) {
174         Serial.println("Connecting to MQTT...");
175         if (client.connect("ESP32", ubidots_token, ubidots_password)) {
176             Serial.println("Connected to MQTT");
177             client.subscribe(topic);
178         } else {
179             Serial.print("Failed with state ");
180             Serial.print(client.state());
181             delay(2000);
182         }
183     }
184 }
185
186 void conectarWifi(){
187     while (WiFi.begin(ssid, password) != WL_CONNECTED) {
188         Serial.print("Conectando a ");
189         Serial.print(ssid);
190         Serial.println(" ...");

```

```

191     delay(5000);
192 }
193 Serial.println(" Conectado a la red wifi correctamente!");
194 }
195
196 //MQTT
197 void callback(char* topic, byte* payload, unsigned int length){
198     Serial.print("Ha llegado mensaje en el topic");
199     Serial.print(topic);
200     Serial.print("]. Mensaje: ");
201     for(int i=0; i<length; i++){
202         Serial.print((char)payload[i]);
203         if((char)payload[i] == '1'){
204             Serial.println(" ");
205             puedeContinuar=true;
206             printErrorDEDO();
207             Serial.println("Programa activado.");
208             break;
209         }
210         if((char)payload[i] == '0'){
211             Serial.println(" ");
212             puedeContinuar=false;
213             printDesactivado();
214             Serial.println("Programa desactivado.");
215             break;
216         }
217     }
218     Serial.println(" ");
219 }
220
221 //Imprimir Text en pantalla
222 void PrintText(int Size, int X, int Y, String Texto)
223 {
224     display.setTextSize(Size);
225     display.setTextColor(WHITE);
226     display.setCursor(X,Y);
227     display.println(Texto);
228 }
229 }
230
231 void printErrorDEDO(){
232     display.clearDisplay();
233     PrintText(1,4,3,"ERROR DEDO");
234     digitalWrite(led,HIGH);
235     display.display();
236 }
237 void printDesactivado(){
238     Linea1=String("SUSPENDIDO");
239     Linea2=" ";
240     display.clearDisplay();
241     PrintText(1,4,3,Linea1);
242     PrintText(1,4,18,Linea2);
243     display.display();
244 }
245
246
247 void parpadeo(int ms){
248     parpadeoHIGH(ms);
249     parpadeoLOW(ms);
250 }
251
252 void parpadeoHIGH(int ms){
253     digitalWrite(led,HIGH);
254     delay(ms);
255 }
256
257 void parpadeoLOW(int ms){
258     digitalWrite(led,LOW);

```



```
259     delay (ms) ;  
260 }
```