Práctica 4 : Estación meteorológica usando ESP32

Marlon Sneider Mora Cortes - 20152005034, Sara Valentina Barrero Medina -20191005170
Miguel Ángel Fuentes Ramírez-20182005007
Ingeniería Electrónica, *Universidad Distrital "Francisco José de Caldas"*
Bogotá, Colombia

***Resumen*—En este adocumento se describe la importancia de los sistemas embebidos en la industria y en la vida cotidiana, en particular en la medición y registro de datos meteorológicos. Se presenta un proyecto de elaboración de una estación meteorológica utilizando la tarjeta ESP32 y IoT para la conexión y transmisión de datos. La estación cuenta con cinco sensores para medir la temperatura, humedad, presión atmosférica, nivel de lluvia y velocidad del viento. Para la visualización y monitoreo de los datos, se emplearon las aplicaciones Thingspeak y RemoteXY. Este proyecto es un ejemplo de cómo la tecnología de los sistemas embebidos y el IoT pueden solucionar problemas en diferentes áreas.**

***Palabras Clave*–**Sistemas embebidos, IoT, estación meteorológica, Esp32,Thingspeak, RemoteXY, Bluetooth, Blynk, aplicaciones, web, API.

I. Introducción

En la actualidad, la implementación de sistemas embebidos se ha convertido en una herramienta fundamental para mejorar y automatizar diversas actividades en la industria y en la vida cotidiana. Los sistemas embebidos son dispositivos electrónicos que se integran en otros productos o sistemas para controlar, monitorizar o automatizar una tarea específica. Uno de los campos donde los sistemas embebidos han tenido un gran impacto es en la medición y registro de datos meteorológicos.

El proyecto 4 de la asignatura Sistemas Embebidos 1 propone la elaboración de una estación meteorológica utilizando la tarjeta de desarrollo ESP32 y empleando IoT (Internet de las cosas) como medio para conectar los diferentes sensores y transmitir los datos de forma remota. La estación meteorológica consta de un total de 5 sensores que miden variables como la temperatura, humedad, presión atmosférica, nivel de lluvia y velocidad del viento.

Para el monitoreo y visualización de los datos, se emplearon dos aplicaciones. La primera de ellas es Thingspeak, una plataforma IoT que permite el registro, análisis y visualización de datos en tiempo real. La segunda es RemoteXY, una aplicación para dispositivos móviles que permite controlar y monitorear sistemas embebidos a través de Bluetooth.

La elaboración de esta estación meteorológica es un ejemplo claro de cómo la tecnología de los sistemas embebidos y el IoT pueden ser aplicados para solucionar problemas específicos en distintas áreas. En este informe se presentará la propuesta de solución para el proyecto 4, explicando detalladamente el funcionamiento de la estación meteorológica y las aplicaciones utilizadas para su monitoreo.

II. Formulación del Problema

El problema consiste en diseñar e implementar una estación meteorológica casera haciendo uso de sensores de humedad relativa (valor en %), presión atmosférica (valor en mbar), temperatura (valor en °C), velocidad del viento (valor en m/s) y cantidad de lluvia (valor en mm/h).

El sistema diseñado tomara los datos por cada sensor y deberán ser visualizados gráficamente en la Web por medio de la API gratuita ThingSpeak (o similar) haciendo uso del WiFi del ESP32.

Se deberá diseñar una aplicación en el smartphone utilizando Blynk que permita la comunicación con el ESP32 por medio de Bluetooth que permita la visualización de los datos recibidos, estos datos se mostraran de forma gráfica y numérica, ya sea con barras, donas o similares (en la figura pueden ver una idea lo descrito), para cada caso el widget seleccionado debe ser diferente, el tiempo de adquisición de cada una de las variables deberá ser configurado dentro de la aplicación desarrollada. Cada variable deberá tener un historial gráfico que podrá ser consultado en cualquier momento por el usuario.

III. Diseño y Modelo de solución

A. Conexión sensores

Lo primero que se contemplo para la elaboración del proyecto son los elementos requeridos, para ello es necesario desarrollar una propuesta de solución para la toma de datos en las unidades solicitadas en la guía, para ello se realizo :

1) Sensor de Humedad relativa, Presión atmosférica y Temperatura :: Para medir estas variables se utiliza el sensor Bme280 que se comunica mediante el protocolo I2C.

Figure 1: Caption

2) Sensor de Velocidad del viento ::

3) Sensor de Cantidad de lluvia :: Para realizar la medida de cantidad de lluvia se elaboro un mecanismo de almacenamiento de lluvia y su posterior vaciado, teniendo como medida el tiempo que tardo en llenarse el recipiente. Para esto se uso un sensor de humedad y un servomotor.

Figure 2: Caption

Figure 3: Caption

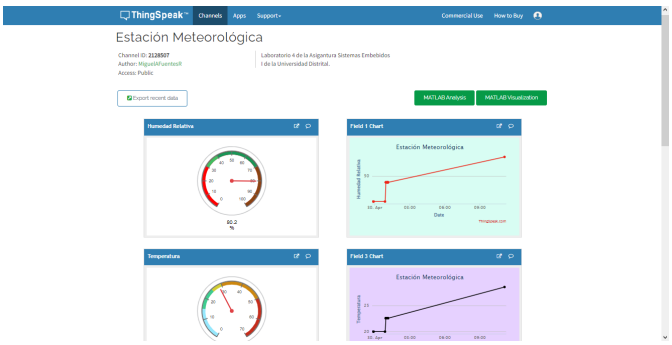
B. Diseño Aplicación ThingSpeak

Se empleo ThingSpeak para enviar y visualizar datos de sensores en tiempo real, el código desarrollado se conecta a una red WiFi, establece los valores de los sensores, los envía a un canal en ThingSpeak y repite el proceso cada cierto tiempo.

Se emplearon las librerías necesarias para utilizar las funciones de WiFi (WiFi.h) y ThingSpeak (ThingSpeak.h)

En la función ”Datos_ThingSpeak()”, se comprueba si la conexión WiFi está activa y si ha pasado un cierto intervalo de tiempo. Si se cumple esta condición, se establecen los valores de los campos del canal en ThingSpeak y se envían mediante la función ThingSpeak.writeFields(). Si el envío se realiza con éxito, se muestra un mensaje por la consola indicando que el canal se ha actualizado. Si hay un problema en el envío, se muestra un mensaje con el código de error HTTP correspondiente. Además, se actualiza el estado del LED y se decrementa el contador de segundos para la siguiente actualización de los valores de los sensores.

Se desarrollo el siguiente canal : Canal ThingSpeak.



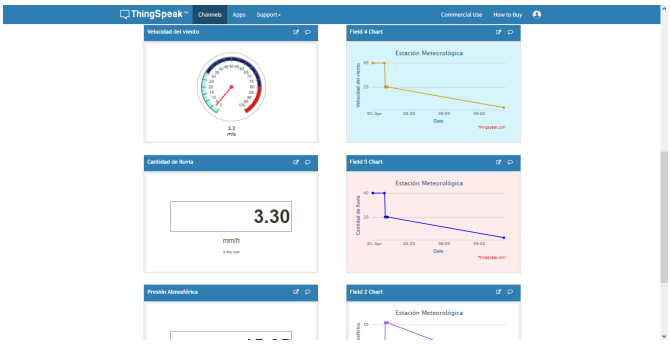


Figure 4: Canal desarrollado en ThingSpeak

C. *Diseño Aplicación RemoteXY*

Como propuesta de alternativa a la App Blynk se empleo la Aplicacion RemoteXY para el diseño de una App Movil que recopila datos de varios sensores, incluidos la humedad relativa, la presión atmosférica, la temperatura, la velocidad de actualización del viento y la cantidad de lluvia, y los muestra en una aplicación móvil conectada por Bluetooth, se emplea la biblioteca BLEDevice para configurar la conexión Bluetooth de baja energía.

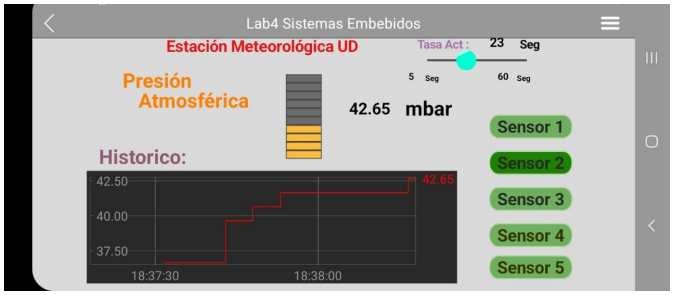
La sección de configuración de RemoteXY define las características del control remoto, incluyendo el nombre Bluetooth, el tamaño y la posición de los controles, y la forma en que se transmiten los datos.

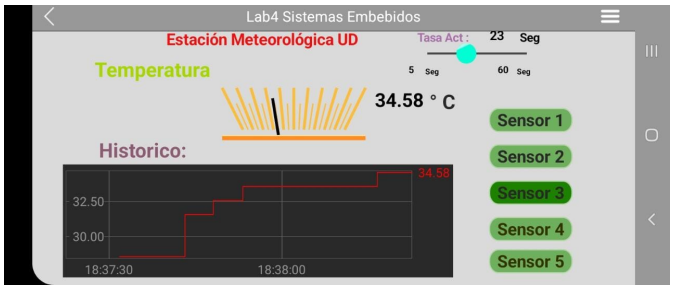
La aplicación configura la conexión Bluetooth usando el modo de conexión ESP32CORE_BLE y el nombre Bluetooth ”Lab4 Sistemas Embebidos”. A continuación, se define la configuración de RemoteXY, que especifica la interfaz de usuario de la aplicación. La interfaz incluye varios controles, como medidores y cuadros de texto, para mostrar los valores de los sensores. La configuración se almacena en una matriz de bytes llamada RemoteXY_CONF empleando el concepto de estructura, por medio de las variables definidas en esta estructura se interactuará con el ESP32.

La aplicación incorpora 3 tipos de visualizaciones para los datos de cada sensor :

1) Visualización empleando algún gráfico (Circular, Barra, Tipo Termómetro, Velocímetro).
2) Visualización numérica de los datos en su respectiva unidad.
3) Visualización de todos los datos recopilados en un gráfico temporal.

La aplicación Bluetooth desarrollada se aprecia en la figura 5.





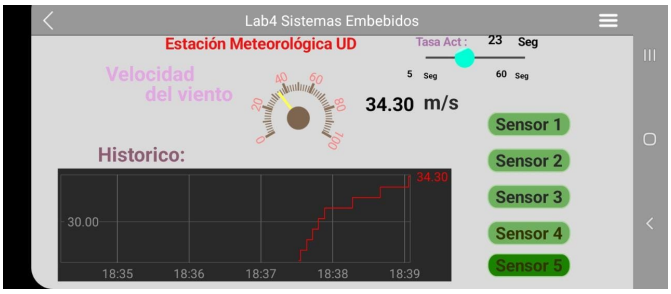


Figure 5: Aplicación Bluetooth Desarrollada

Luego, se llevó a cabo la elaboración e implementación del código necesario para el correcto funcionamiento de la estación meteorológica, haciendo uso del entorno de desarrollo de Arduino IDE. Este código se encuentra disponible para su revisión en el anexo 1.

IV. Resultados

Se desarrollo el siguiente montaje :

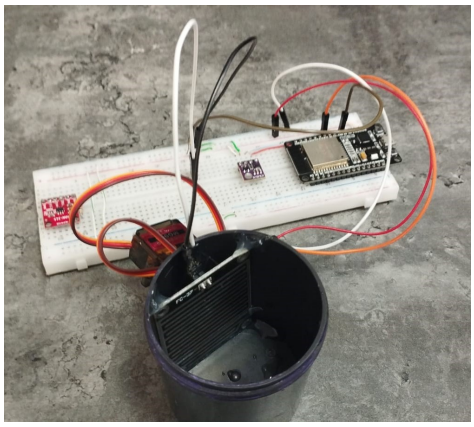


Figure 6: Montaje Final

En la figura 6 se aprecia el montaje final realizado.

V. Conclusiones

- Es necesario establecer un rango de operación para los sensores capacitivos ya que estos tienden a ser muy sensibles al ruido y en consecuencia afectar el comportamiento del sistema.
- La implementación del protocolo I2C para la transmisión de datos, en este caso para el control del RTC, facilita la tarea, ya que solo se requieren dos líneas de bus; una línea de datos en serie (SDA) y una línea de reloj en serie SCL, esto resulta muy útil a la hora de incorporar nuevos módulos en un montaje físico.
- El uso de la plataforma ThingSpeak para enviar y visualizar datos de sensores en tiempo real es una solución conveniente y efectiva así como la elaboración de una aplicación en el smartphone que realice la comunicación con el ESP32 por medio de Bluetooth y que permita la visualización de los datos recibidos, es una forma práctica y accesible para el usuario final.
- Gracias a que esta tarjeta tiene la posibilidad de ser programada en el entorno de desarrollo correspondiente a Arduino IDE el cual utiliza C como lenguaje de programación y nos facilita diversas librerías que son de gran utilidad al momento de usar diferentes características o componentes externos como por ejemplo los sensores empleados, la configuración del WiFi de la tarjeta y el Bluetooth de Baja energía.
- La elaboración de una estación meteorológica utilizando la tarjeta de desarrollo ESP32 y empleando IoT como medio para conectar los diferentes sensores y transmitir los datos de forma remota, es un ejemplo claro de cómo la tecnología de los sistemas embebidos y el IoT pueden ser aplicados para solucionar problemas específicos en distintas áreas.

```
//////////////////////////////////////////////
//        App Bluetooth usando RemoteXY     //
//////////////////////////////////////////////


// RemoteXY select connection mode and include library
#define REMOTEXY_MODE__ESP32CORE_BLUETOOTH
#include <BluetoothSerial.h>

#include <RemoteXY.h>

// RemoteXY connection settings
#define REMOTEXY_BLUETOOTH_NAME "Lab4 Sistemas Embebidos"


// RemoteXY configurate
#pragma pack(push, 1)
uint8_t RemoteXY_CONF[] =   // 672 bytes
{ 255, 1, 0, 57, 2, 153, 2, 16, 30, 4, 131, 5, 89, 46, 21, 6, 1, 12, 88, 83,
  101, 110, 115, 111, 114, 32, 52, 0, 131, 4, 89, 19, 21, 6, 2, 12, 8, 83, 101, 110,
  115, 111, 114, 32, 49, 0, 129, 0, 8, 0, 52, 4, 0, 1, 69, 115, 116, 97, 99, 105,
  195, 179, 110, 32, 77, 101, 116, 101, 111, 114, 111, 108, 195, 179, 103, 105, 99, 97, 32, 85,
  68, 0, 66, 0, 38, 9, 9, 21, 4, 2, 26, 129, 0, 4, 13, 19, 5, 2, 146, 82,
  101, 108, 97, 116, 105, 118, 97, 0, 129, 0, 71, 17, 4, 5, 2, 24, 37, 0, 67, 1,
  49, 15, 20, 5, 4, 24, 26, 101, 129, 0, 249, 8, 22, 5, 2, 146, 72, 117, 109, 101,
  100, 97, 100, 0, 129, 0, 253, 8, 19, 5, 4, 64, 80, 114, 101, 115, 105, 195, 179, 110,
  32, 0, 71, 56, 29, 8, 25, 25, 3, 0, 73, 95, 135, 0, 0, 0, 0, 0, 0, 200,
  66, 0, 0, 160, 65, 0, 0, 32, 65, 0, 0, 0, 64, 39, 0, 129, 0, 246, 7, 29,
  5, 5, 106, 84, 101, 109, 112, 101, 114, 97, 116, 117, 114, 97, 0, 66, 131, 22, 14, 36,
  13, 5, 2, 24, 129, 0, 74, 15, 6, 5, 5, 8, 194, 176, 32, 67, 0, 129, 0, 249,
  7, 23, 5, 3, 238, 86, 101, 108, 111, 99, 105, 100, 97, 100, 0, 129, 0, 3, 12, 23,
  5, 3, 238, 100, 101, 108, 32, 118, 105, 101, 110, 116, 111, 0, 72, 4, 30, 9, 20, 20,
  2, 95, 26, 189, 39, 0, 0, 0, 0, 0, 0, 200, 66, 0, 0, 0, 0, 67, 1, 57,
  16, 13, 6, 2, 24, 26, 101, 131, 4, 89, 54, 21, 6, 3, 12, 88, 83, 101, 110, 115,
  111, 114, 32, 53, 0, 129, 0, 70, 15, 12, 5, 4, 24, 109, 98, 97, 114, 0, 67, 1,
  60, 14, 13, 6, 5, 24, 26, 101, 67, 1, 58, 14, 13, 6, 3, 24, 26, 101, 67, 1,
  52, 16, 13, 6, 1, 24, 26, 101, 129, 0, 73, 14, 6, 5, 3, 8, 109, 47, 115, 0,
  129, 0, 254, 7, 23, 5, 1, 6, 67, 97, 110, 116, 105, 100, 97, 100, 32, 100, 101, 0,
  129, 0, 10, 11, 14, 5, 1, 6, 76, 108, 117, 118, 105, 97, 0, 66, 1, 29, 12, 21,
  16, 1, 193, 26, 129, 0, 72, 17, 8, 5, 1, 8, 109, 109, 47, 104, 0, 131, 4, 89,
  28, 21, 6, 4, 12, 8, 83, 101, 110, 115, 111, 114, 32, 50, 0, 131, 4, 89, 37, 21,
  6, 5, 12, 8, 83, 101, 110, 115, 111, 114, 32, 51, 0, 129, 0, 1, 13, 28, 5, 4,
  64, 65, 116, 109, 111, 115, 102, 195, 169, 114, 105, 99, 97, 0, 68, 17, 237, 33, 100, 28,
  2, 8, 36, 129, 0, 247, 27, 22, 5, 0, 255, 72, 105, 115, 116, 111, 114, 105, 99, 111,
  58, 0, 68, 17, 244, 33, 93, 28, 4, 8, 36, 68, 17, 238, 33, 98, 28, 5, 8, 36,
  68, 17, 238, 33, 98, 28, 1, 8, 36, 68, 17, 237, 33, 99, 28, 3, 8, 36, 4, 128,
  71, 3, 30, 5, 0, 164, 26, 129, 0, 73, 9, 3, 2, 0, 24, 83, 101, 103, 0, 129,
  0, 71, 0, 14, 3, 0, 240, 84, 97, 115, 97, 32, 65, 99, 116, 32, 58, 32, 0, 129,
  0, 96, 9, 3, 2, 0, 24, 83, 101, 103, 0, 67, 1, 87, 255, 8, 4, 0, 24, 26,
  11, 129, 0, 96, 0, 6, 3, 0, 24, 32, 83, 101, 103, 0, 67, 1, 67, 8, 5, 3,
  0, 24, 26, 11, 67, 1, 90, 8, 5, 3, 0, 24, 26, 11, 129, 0, 37, 6, 10, 2,
  4, 24, 49, 48, 48, 48, 32, 109, 98, 97, 114, 0
};

// this structure defines all the variables and events of your control interface
struct {

  // input variables
  int8_t Tasa_Actualizacion; // =0..100 slider position

  // output variables
  int8_t Barra_Presion; // =0..100 level position
  char Valor_Presion[101];  // string UTF8 end zero
  float Barra_VelocidadViento;  // from 0 to 100
  int8_t Barra_Temperatura; // =0..100 level position
  float Barra_Humedad;  // from 0 to 100
  char Valor_Humedad[101];  // string UTF8 end zero
  char Valor_Temperatura[101];  // string UTF8 end zero
  char Valor_Viento[101];  // string UTF8 end zero
  char Valor_Lluvia[101];  // string UTF8 end zero
```

```
  int8_t Barra_Lluvia; // =0..100 level position
  float Graf_Humedad;
  float Graf_Atmosfera;
  float Graf_Temperatura;
  float Graf_LLuvia;
  float Graf_Viento;
  char Valor_Tiempo[11];  // string UTF8 end zero
  char Valor_Tiempo_Minimo[11];  // string UTF8 end zero
  char Valor_Tiempo_Maximo[11];  // string UTF8 end zero

  // other variable
  uint8_t connect_flag;  // =1 if wire connected, else =0

} RemoteXY;
#pragma pack(pop)


//////////////////////////////////////////
//          Aplicacion ThingSpeak       //
//////////////////////////////////////////

#include <WiFi.h>
#include "ThingSpeak.h"

//Credenciales de la red WiFi a conectarse:
const char* ssid = "UDNet_Academia";
const char* password = "6Lj3P_UD_$";


//const char* ssid = "ADM_UDistrital";
//const char* password = "PKFJuurjf986_$";

//const char* ssid = "Invitados_UDistrital";
//const char* password = "1nv1tad0s"
/*
  const char* ssid="Familia Ramirez";
  const char* password="FamiliaRamirezFuentes2020";
*/
unsigned long Channel_ID = 2128507; //N mero del canal en ThingSpeak
const char* WriteAPIKey = "1B9QMCDN2HQUEUMX"; //Api Key de escritura del canal en ThingSpeak
WiFiClient client;

// Definicion de Sensores
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#define SEALEVELPRESSURE_HPA (1013.25)

Adafruit_BME280 bme; // I2C
#define LED_B 2 //Define GPIO2

const uint16_t ledRefresh = 1;
uint16_t dataRefresh = 20; //Intervalo de refresco de los datos (en segundos)

uint16_t contadorSegundos;

// Definicion de Variables

float Sensor_Temperatura = 22.58;
float Sensor_Viento = 20.3;
float Sensor_Presion = 30.65;
float Sensor_Humedad = 40.2;
float Sensor_Lluvia  = 50.1;

//Variables Auxiliares
uint32_t t_actual;
bool Inicio_Bluetooth = true;
int test = 50;
int Tiempo = 5; //Tiempo de actualizacion en Segundos
int Tiempo_Min = 5; //5 Seg minimos
int Valor_Max_Tiempo = 60; // 60 Seg

```

```
142 //****************** TIMMER ******************************
143 //Variables Timmer
144 int Velocidad_Actualizacion ;
145 hw_timer_t *timer1 = NULL;
146
147 void IRAM_ATTR Int_Timer1() {
148   if (test < 100) {
149     Sensor_Viento++;
150     Sensor_Lluvia++;
151     test++;
152   } else {
153     Sensor_Viento = 0;
154     Sensor_Lluvia = 0;
155     test = 50;
156   }
157   Actualizar_Datos(); //Actualiza los Datos de la App Bluetooth
158 }
159
160
161 void setup() {
162   timer1 = timerBegin(1, 80, true); //Configura Timer 1 con preesc.x80 (=1MHz) y cambio por flanco
        de bajada
163   Velocidad_Actualizacion = 5000000; // Actualizacion de datos cada 5 Segundos
164
165   Serial.begin(115200);
166
167   //Inicializacion de Pines Sensores
168   pinMode(LED_B, OUTPUT);
169   Iniciar_Sensores();
170
171   RemoteXY_Init ();
172   Serial.print("Emparejando");
173   digitalWrite(LED_B, HIGH);
174   t_actual = millis();
175
176
177   //Conexion  Wifi
178 }
179
180 void loop() {
181
182
183
184   //Iniciar Bluetooth
185   Iniciar_Bluetooth();
186
187   //--------------- Lectura de Valores Sensores -------------------
188
189
190   Sensor_Temperatura = bme.readTemperature();
191   // El sensor mide en Pascales por tanto =  1 pascal = 0.01 mbar
192   Sensor_Presion = bme.readPressure() * 0.01;
193   Sensor_Humedad = bme.readHumidity();
194   //------------ Actualizacion de Datos Bluetooth ------------------
195
196   Tasa_Actualizacion();
197   dataRefresh = Tiempo;
198   //Activar_Timer();
199   Datos_ThingSpeak();
200
201
202 }
203
204 void Iniciar_Bluetooth() {
205
206   RemoteXY_Handler ();
207   if (!RemoteXY.connect_flag) {
208     Serial.print('.');
209     delay(1000);
210   }
211   if (RemoteXY.connect_flag && Inicio_Bluetooth) {
```

```
    Serial.println(' ');
    Serial.println("Bluetooth Conectado");
    digitalWrite(LED_B, LOW);
    Inicio_Bluetooth = false;
    Iniciar_Wifi();
  }
}

void Iniciar_Wifi() {
  Serial.println("Conectando a ");
  Serial.println(ssid);
  WiFi.begin(ssid, password); //Conectarse a la red WiFi con ssid y contrase a indicada
  while (WiFi.status() != WL_CONNECTED) { //Esperar hasta que se realice la conexi n con el router
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi conectado");

  ThingSpeak.begin(client); //Iniciar cliente en ThingSpeak


  pinMode(LED_B, OUTPUT);
  digitalWrite(LED_B, LOW);
  contadorSegundos = 0;
}

void Iniciar_Sensores() {
  Serial.println(F("BME280 test"));
  bool status;
  // default settings
  // (you can also pass in a Wire library object like &Wire2)
  status = bme.begin(0x76);
  if (!status) {
    Serial.println("Could not find a valid BME280 sensor, check wiring!");
    while (1);
  }
}

void Datos_ThingSpeak() {
  if (WiFi.status() == WL_CONNECTED && contadorSegundos <= 0) { //Si est  conectado...
    contadorSegundos = dataRefresh;
    ThingSpeak.setField(1, Sensor_Humedad); //Establecer el campo 1 en Canal de ThingSpeak -->
    Humedad Relativa
    ThingSpeak.setField(2, Sensor_Presion); //Establecer el campo 2 en Canal de ThingSpeak -->
    Presi n Atmosf rica
    ThingSpeak.setField(3, Sensor_Temperatura); //Establecer el campo 3 en Canal de ThingSpeak -->
    Temperatura
    ThingSpeak.setField(4, Sensor_Viento); //Establecer el campo 4 en Canal de ThingSpeak -->
    Velocidad del viento
    ThingSpeak.setField(5, Sensor_Viento); //Establecer el campo 5 en Canal de ThingSpeak -->
    Cantidad de lluvia
    Actualizar_Datos(); //Actualiza los Datos de la App Bluetooth

    int httpCode = ThingSpeak.writeFields(Channel_ID, WriteAPIKey); //Enviar los datos al servidor
    de ThingSpeak
    if (httpCode == 200) {
      Serial.println("Canal actualizado!!");
    } else {
      Serial.println("Problema al actualizar el canal. Codigo de error HTTP: " + String(httpCode));
    }
  }

  delay(ledRefresh * 1000);

  digitalWrite(LED_B, !digitalRead(LED_B));
  contadorSegundos -= ledRefresh;
}


void Tasa_Actualizacion() {
```

```
277   if (RemoteXY.Tasa_Actualizacion == 0) {
278     //Velocidad_Actualizacion = Tiempo_Min * 1000000;
279     Tiempo = 5;
280   } else {
281     //Velocidad_Actualizacion = (RemoteXY.Tasa_Actualizacion * Valor_Max_Tiempo * 10000); // Para
        ajustar el Timmer
282     Tiempo = (int)(RemoteXY.Tasa_Actualizacion * 3 / 5);
283   }
284   itoa (Tiempo, RemoteXY.Valor_Tiempo, 10);
285   itoa (Tiempo_Min, RemoteXY.Valor_Tiempo_Minimo, 10);
286   itoa (Valor_Max_Tiempo, RemoteXY.Valor_Tiempo_Maximo, 10);
287 }
288
289
290 void Activar_Timer() {
291   timerAttachInterrupt(timer1, Int_Timer1, true); //Hab. int del Timer 1
292   timerAlarmWrite(timer1, Velocidad_Actualizacion, true);
293   timerAlarmEnable(timer1); //Habilitar el Timer 1
294 }
295 void Desactivar_Timer() {
296   timerRestart(timer1);
297   timerAlarmDisable(timer1); //Habilitar el Timer 1
298 }
299 void Actualizar_Datos() {
300
301   //Actualizacion de los Strings
302   dtostrf(Sensor_Humedad, 0, 2, RemoteXY.Valor_Humedad);
303   dtostrf(Sensor_Temperatura, 0, 2, RemoteXY.Valor_Temperatura);
304   dtostrf(Sensor_Viento, 0, 2, RemoteXY.Valor_Viento);
305   dtostrf(Sensor_Lluvia, 0, 2, RemoteXY.Valor_Lluvia);
306   dtostrf(Sensor_Presion, 0, 2, RemoteXY.Valor_Presion);
307
308   //Actualizacion de la Barra de Humedad Relativa %
309   RemoteXY.Barra_Humedad =   Sensor_Humedad;
310   //Actualizacion de la Barra de Temperatura  Grados Celcius
311   RemoteXY.Barra_Temperatura  =  Sensor_Temperatura;
312   //Actualizacion de la Barra de Viento
313   RemoteXY.Barra_VelocidadViento  =  Sensor_Viento;
314   //Actualizacion de la Barra de Lluvia
315   RemoteXY.Barra_Lluvia = (int)(Sensor_Lluvia) ;
316   //Actualizacion de la Barra de Presion
317   RemoteXY.Barra_Presion = (int)(Sensor_Presion / 10) ; //Escalarlo para mostrarlo de una escala de
        0 - 100
318
319   // Actualizar las Graficas
320   Actualizar_Graficas();
321 }
322 void Actualizar_Graficas() {
323
324   //A adir Valores al Historico
325   RemoteXY.Graf_Humedad = Sensor_Humedad;
326   RemoteXY.Graf_Atmosfera = Sensor_Presion;
327   RemoteXY.Graf_Temperatura = Sensor_Temperatura;
328   RemoteXY.Graf_LLuvia = Sensor_Lluvia;
329   RemoteXY.Graf_Viento = Sensor_Viento;
330
331 }
```

Listing 1: Código Proyecto 4