

Predictor meteorológico

Predictor meteorológico

Autor Mario Monteagudo Asesor digital Centro de Profesorado de Ejea de los Caballeros

Usando IDE Arduino y el sensor Bosch BME280 se puede medir la presión atmosférica, temperatura y humedad y usando un código predictivo **Algoritmo de Zambretti** que usa las diferencias de presiones cada 3 horas, puede hacer pronósticos. Para ello mide la presión cada 5 minutos y tienen en cuenta la altitud del lugar y la temperatura.

El ESP32 tendría que estar en el exterior y como hemos explicado, necesita 3 horas para hacer su primera predicción.

Este es el resultado

<https://www.youtube.com/embed/psGhrktsdQI?t=1s>

Y este es el código

```
#include "arduino_secrets.h"
//Predictor del tiempo mediante el algoritmo de Zambretti*****
//2024*Mario Monteagudo Alda*****mario.monteagudo@cpejea.es*****

#include "arduino_secrets.h"

#include <SimpleBME280.h> //Biblioteca usada para el sensor

const float ALTURA = 497.0; //Altitud del lugar

SimpleBME280 bme280; //Declaración de la instancia del sensor
```

```
float presiones[37]; //Presiones corregidas cincominutales de las 3 últimas horas
int zambretti;      //Número de Zambretti
float temperatura;  //°C
float humedad;      //Humedad relativa en %
float presion;      //Presión atmosférica en mbar
int contador = 0;   //Contador de número de medidas de presión cincominutales tomadas
bool encendido = false; //Bandera para intermitencia del LED
```

```
/*
  Arduino IoT Cloud Variables description

  The following variables are automatically generated and updated when changes are made to the
  Thing
```

```
String boletin;
float fiabilidad;
float tendencia;
```

Variables which are marked as READ/WRITE in the Cloud Thing will also have functions which are called when their values are changed from the Dashboard.

These functions are generated with the Thing and added at the end of this sketch.

```
*/
```

```
#include "thingProperties.h"
```

```
void setup() {
```

```
  pinMode(LED_BUILTIN,OUTPUT);
  digitalWrite(LED_BUILTIN,HIGH);
```

```
  // Inicialización de la comunicación serie y espera
  Serial.begin(19200);
  delay(2000);
```

```
  // Defined in thingProperties.h
  initProperties();
```

```
// Connect to Arduino IoT Cloud and wait
ArduinoCloud.begin(ArduinoIoTPreferredConnection, false);
delay(2000);
/*
    The following function allows you to obtain more information
    related to the state of network and IoT Cloud connection and errors
    the higher number the more granular information you'll get.
    The default is 0 (only errors).
    Maximum is 4
*/
setDebugMessageLevel(2);
ArduinoCloud.printDebugInfo();

bme280.begin(); //Inicialización del sensor y espera
delay(2000);

//Primera lectura del sensor
bme280.update();
presion = bme280.getP();
temperatura = bme280.getT();

//Corrección de la presión según la altitud y temperatura y paso de Pa a mbar
presion = (presion * pow(1 - (0.0065 * ALTURA) / (temperatura + (0.0065 * ALTURA) +
273.15), -5.257 ))/100.0;
for (int i=0; i<=36; i++) { presiones[i]=presion;}

fiabilidad = 0;
}

void loop() {

    bme280.update();
    presion = bme280.getP();
    temperatura = bme280.getT();
    humedad = bme280.getH();
```

```

presion = (presion * pow(1 - (0.0065 * ALTURA) / (temperatura + (0.0065 * ALTURA) +
273.15), -5.257 ))/100.0;

tendencia=presion-presiones[0];

for (int i=0; i<36; i++) { presiones[i]=presiones[i+1];}
presiones[36] = presion;

//Cálculo del número de Zambretti
if ((tendencia <= -1.6) & (presion > 985.0) & (presion < 1050.0)) {zambretti = round(127-
0.12*presion);}

else if ((tendencia >= 1.6) & (presion > 947.0) & (presion < 1030.0)) {zambretti =
round(185-0.16*presion);}

else if ((abs(tendencia) < 1.6) & (presion > 960.0) & (presion < 1033.0)) {zambretti =
round(144-0.13*presion);}

else {zambretti = 0;}

//Boletín con el pronóstico
boletin = "Temperatura: " +      String(temperatura,1) + " °C" + "\n" +
        "Humedad relativa: " + String(humedad,1) + " %" + "\n" +
        "Presión: " +          String(presion,0) + " mbar" + "\n" +
        "Tendencia: " +        String(tendencia,2) + " mbar" + "\n" +
        pronostico(zambretti);

Serial.println(boletin + "\n");

//Espera de cinco minutos para la actualización
unsigned long tiempo = millis();
while ((millis()-tiempo) < 5*60*1000 )
{
    ArduinoCloud.update();
    delay(1000);
    encendido = !encendido;

```

```
        digitalWrite(LED_BUILTIN,encendido);
    }

    contador = min(36, contador + 1);

    //Después de tres horas de medidas, la fiabilidad es del 100%
    fiabilidad = map(contador, 0, 36, 0, 100);

}

//Cadena de texto con el pronóstico meteorológico*****

String pronostico(int z)
{
    switch (z)
    {
        case 1:
            return "Tiempo estable"; break;
        case 2:
            return "Buen tiempo"; break;
        case 3:
            return "Buen tiempo con ligera inestabilidad"; break;
        case 4:
            return "Buen tiempo evolucionando a chubascos"; break;
        case 5:
            return "Chubascos evolucionando a tiempo inestable"; break;
        case 6:
            return "Inestable evolucionando a lluvioso"; break;
        case 7:
            return "Intervalos de lluvia con empeoramiento"; break;
        case 8:
            return "Intervalos de lluvia evolucionando a gran inestabilidad"; break;
        case 9:
            return "Muy inestable con lluvia"; break;
        case 10 :
            return "Tiempo estable"; break;
```

```
case 11:
    return "Buen tiempo"; break;
case 12:
    return "Buen tiempo con posibles chubascos"; break;
case 13:
    return "Buen tiempo con probables chubascos"; break;
case 14:
    return "Chubascos con intervalos despejados"; break;
case 15:
    return "Variable con algo de lluvia"; break;
case 16:
    return "Inestable con intervalos de lluvia"; break;
case 17:
    return "Lluvia a intervalos frecuentes"; break;
case 18:
    return "Muy inestable con lluvia"; break;
case 19:
    return "Tormentoso con lluvia abundante"; break;
case 20 :
    return "Tiempo estable"; break;
case 21:
    return "Buen tiempo"; break;
case 22:
    return "Evolucionando a buen tiempo"; break;
case 23:
    return "Buen tiempo evolucionando a mejor"; break;
case 24:
    return "Buen tiempo con posibles chubascos a primeras horas"; break;
case 25:
    return "Chubascos a primeras horas evolucionando a mejor"; break;
case 26:
    return "Variable evolucionando a mejor"; break;
case 27:
    return "Inestable a primeras horas evolucionando a estable"; break;
case 28:
    return "Inestable con probable mejoría"; break;
```



```
case 29:
    return "Inestable con breves intervalos de buen tiempo"; break;
case 30:
    return "Muy inestable con intervalos de buen tiempo"; break;
case 31:
    return "Tormentoso con probable mejoría"; break;
case 32:
    return "Tormentoso con mucha lluvia"; break;
default:
    return "Tiempo indeterminado";
}
}
```

Revision #1

Created 24 April 2025 11:31:00 by Javier Quintana

Updated 24 April 2025 11:32:13 by Javier Quintana