# Board Design & Embedded System Programming

Autor 1: Klisver Javier Reyes. Autor 2: Miguel Ángel Aguirre. Autor 3: Juan José Chacón.
Javier.reyes@utp.edu.co, miguel.aguirre12@utp.edu.co, juan.chacon@utp.edu.co
Facultad de Ingenierías, Programa de Ingeniería Electrónica.
Universidad Tecnológica de Pereira, Colombia.

**Resumen – Este documento presenta el desarrollo de un firmware para el módulo ESP32 y el sensor BMP280 que permite medir la temperatura y la humedad, así como también para controlar estos parámetros a través de varios actuadores. El sistema consta de un led para indicar su estado, un control de temperatura y humedad con visualización para cada parámetro. Adicionalmente, el sistema permite transmitir lecturas de la temperatura y la humedad a través de una interfaz serie UART a un terminal de consola serie de USB cada 5 segundos. Finalmente, el sistema envía datos a una plataforma mediante el módulo wifi del microcontrolador ESP32 cada 10 segundos al presionar un botón.**

**Índice de Términos – Actuador, control, ESP32, humedad, sensor, temperatura, Wi-Fi.**

## I. INTRODUCCIÓN

En el ámbito del diseño y desarrollo de soluciones tecnológicas, se presenta un proyecto que aborda la medición precisa de la presión atmosférica y la presión mediante el sensor BMP280. Estos parámetros serán controlados por varios actuadores. Este firmware está constituido por dicho sensor y el microcontrolador ESP32. La comunicación entre estos dos elementos está regida por comunicación I2C.

Como en muchos proyectos de ingeniería, es necesario realizar una aplicación que cumpla ciertos parámetros. Para este caso, se un indicador de estado del sistema mediante un LED que parpadea en un intervalo de 500 milisegundos. Se cumple con un control y visualización de temperatura, donde su funcionamiento se ve reflejado a través de la manipulación de tres LEDs. El sistema cuenta con comunicación UART (por sus siglas en inglés) entre el ESP32 y el monitor de un usuario que quiera visualizar los datos de temperatura y presión que se están midiendo. El envío de información mediante este tipo de comunicación es de 5 segundos. Esta parte de la aplicación fue simulada mediante otro ESP32 que se conectó al sistema. De forma adicional, la aplicación cuenta un pulsador para enviar estas mediciones a internet, mediante un corredor MQTT.

## II. DIAGRAMA DE CIRCUITO

En la figura 1, se presenta el diagrama de circuito para validar el algoritmo desarrollado. En síntesis, éste consta de un sensor BMP280, dos microcontroladores ESP32, tres leds, cuatro resistencias de 270 Ω y un pulsador.

Como se puede apreciar, el sensor permite la comunicación I2C

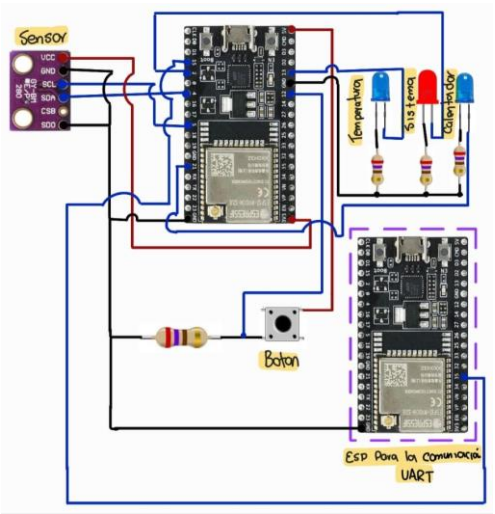


Figura 1. Diagrama de bloques del sistema.

## III. ALGORITMO DESARROLLADO

En la figura 7 se presenta el diagrama de flujo del algoritmo desarrollado.

En este se importan las librerías necesarias para el uso de la comunicación UART, I2C, JSON, tener conexión a internet para enviar información a un servidor MQTT, etc.

Adicionalmente, se definen todas las configuraciones necesarias para la conexión a internet, la configuración del bróker, el sensor, el protocolo de comunicación I2C, la comunicación UART, así como también los diferentes temporizadores requeridos en el diseño del firmware.

Finalmente, el diagrama presenta toda la lógica para dar solución al problema planteado.

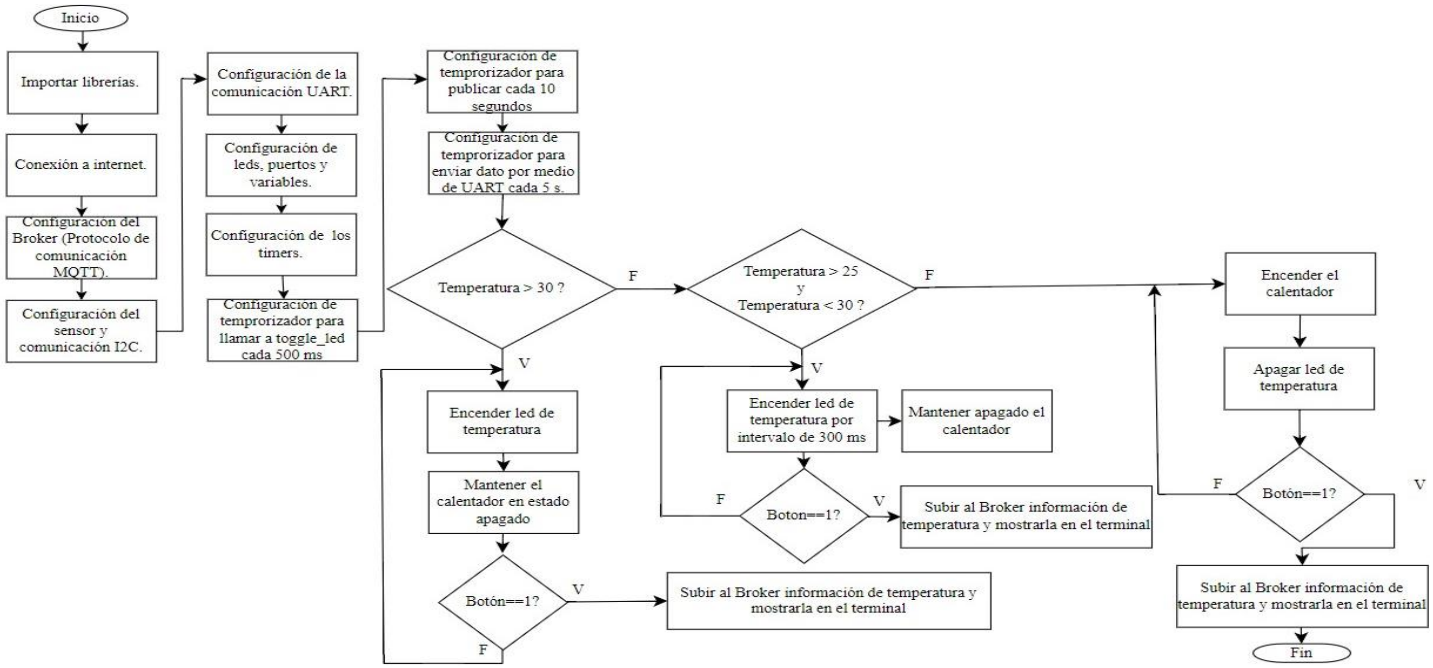


Figura 2. Diagrama de flujo del algoritmo desarrollado.

## IV. RESULTADOS DEL ALGORITMO

La siguiente imagen evidencia el envío de información al servidor MQTT, la cual es enviada cada 10 segundos cuando el sensor BMP280 está captando la temperatura del medio donde se encuentra. Adicionalmente, esta información es enviada por en el terminal del IDE de Pycharm cada vez que se presiona el botón. Se visualiza que la temperatura del ambiente al momento de usar la aplicación fue de 25.35° C.

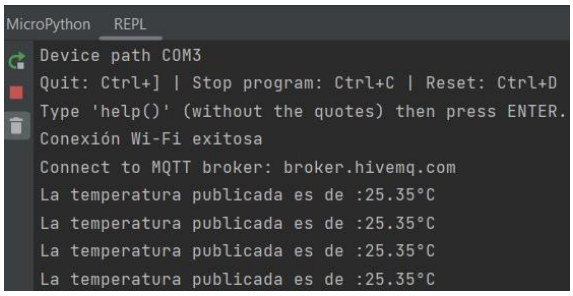


Figura 3. Datos en el terminal de Micropython cuando se presiona el botón.

Como se mencionó en la introducción de este documento, se comprobó el funcionamiento de la comunicación UART mediante la conexión de otro ESP32 (ver figura 1). Considerando lo anterior, las figuras 4 (ESP32 maestro) y 5 (ESP32 esclavo) permiten visualizar el correcto funcionamiento de la comunicación UART por medio del puerto USB tipo C del microcontrolador. Cabe mencionar que los datos transmitidos por este protocolo de comunicación son realizados en intervalos de 5 segundos.

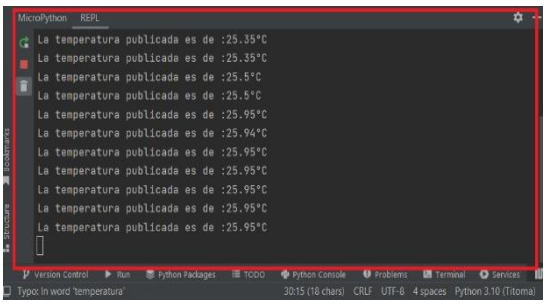


Figura 4. Validación de comunicación UART en el ESP32 maestro.

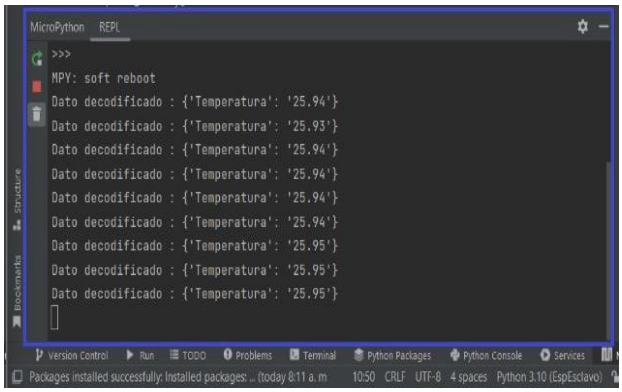


Figura 5. Validación de comunicación UART en el ESP32 esclavo.

## V. RESULTADOS DEL SISTEMA

En esta sección del documento se pretende mostrar la correcta transmisión de datos desde el ESP32 al servidor MQTT presente en internet. La plataforma MQTT usada en la aplicación fue *"HiveMQ"*.

En primer lugar, el servidor muestra el tema (canal de comunicación que se utiliza para transmitir mensajes en MQTT) al que el cliente MQTT está suscrito y publica mensajes. En este caso, el tema se llama "temperatura_miguel".

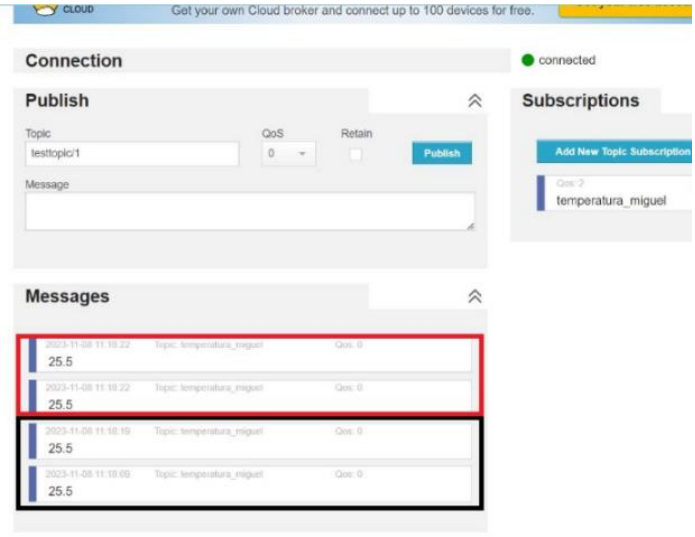


Figura 6. Broker HiveMQ.

El recuadro rojo de la figura anterior indica los datos recibidos por el servidor cuando el botón (ver diagrama de la figura 1) es presionado. Por otro lado, el recuadro negro representa los datos cuando son recibidos en intervalos de 10 segundos.

También es importante exponer el montaje físico de todo el sistema, el cual se permite visualizar en la siguiente imagen. Nótese que en la parte detrás del sensor de temperatura y el diodo LED, está el otro ESP32 usado para la validación de la comunicación UART.

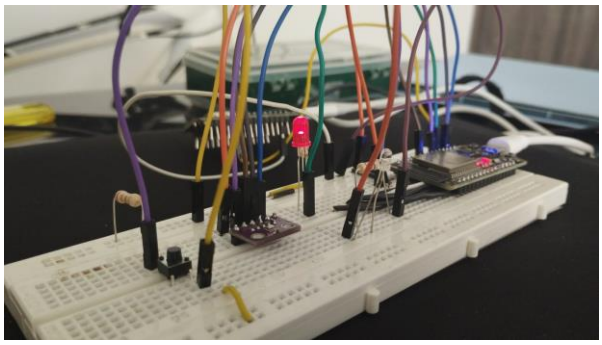

Figura 8. Implementación física del circuito presentado en el diagrama de la figura 1.

## REFERENCIAS

[1] MQTT websocket client. Disponible en: https://www.hivemq.com/demos/websocket-client/ (Fecha de acceso 8 de noviembre del 2023).

[2] ESP32 pinout reference: Which GPIO pins should you use? (2022) Random Nerd Tutorials. Disponible en: https://randomnerdtutorials.com/esp32-pinout-reference-gpios/ (Fecha de acceso 8 de noviembre del 2023).

[3] Micropython-BMP280 GitHub. Disponible en: https://github.com/dafvid/micropython-bmp280/blob/master/README.md#oversampling-setting-see-331-382 (Fecha de acceso 8 de noviembre del 2023).