REPUBLIC OF TUNISIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
UNIVERSITY OF TUNIS EL MANAR
NATIONAL ENGINEERING SCHOOL OF TUNIS

FreeRTOS Project Presentation

SMOKE DETECTION AND RESPONSE SYSTEM

Elaborated by :
Meriem Brahem & Salsabil Jaballah
3AGE2

PLAN

PROBLEM STATEMENT

- **Context:**

In residential environments,many homes lack real-time smoke detection systems, leaving them vulnerable to undetected leaks.

- **Problem:**

Without timely detection, that can lead to dangerous concentrations that may result in accidents or health risks.

PROBLEM STATEMENT

Solution

Develop a real-time smoke detection system that automatically responds , providing ventilation, alerts, and notifications to homeowners to prevent accidents.

SYSTEM OVERVIEW

System Behavior:

- Normal Operation: Continuously monitors gas levels.
- Danger Threshold Exceeded: Activates the fan, alerts the homeowner, and displays a warning message.
- De-escalation: Run fan as gas levels increase and stops alerts when the air is safe.

- **FAN:** ACTIVATED TO VENTILATE THE AREA AND IS CONTROLLED BASED ON THE DETECTED GAS LEVELS.

- **LED:** BLINKS RED TO INDICATE DANGER.
- **BUZZER:** SOUNDS AN ALARM WHEN DANGEROUS LEVELS ARE DETECTED.
- **LCD DISPLAY:** SHOWS THE GAS CONCENTRATION AND THE WORD "DANGER" WHEN THE THRESHOLD IS EXCEEDED.

- **NOTIFICATION:** SENDS ALERTS TO THE HOMEOWNER WHEN A GAS LEAK IS DETECTED.

IMPLEMENTATION ON STM32F407

Microcontroller Overview

- ARM Cortex-M4 core.
- Rich peripheral interfaces (UART, I2C, PWM).
- Support for FreeRTOS.

Sensor Integration

- Connect the gas sensor to an analog input of STM32F407.
- Use ADC (Analog-to-Digital Converter) to measure gas concentration.

PWM Control for Fan

- Use a PWM output to control fan speed .

Alert Systems

- Interface the buzzer, LED/lamp, and LCD via GPIO pins for real-time alerting.

Wifi Module Integration

- Sending E_Mail

User friendly Dashboard

COMPONENTS NEEDED FOR THE PROJECT

Hardware Components:

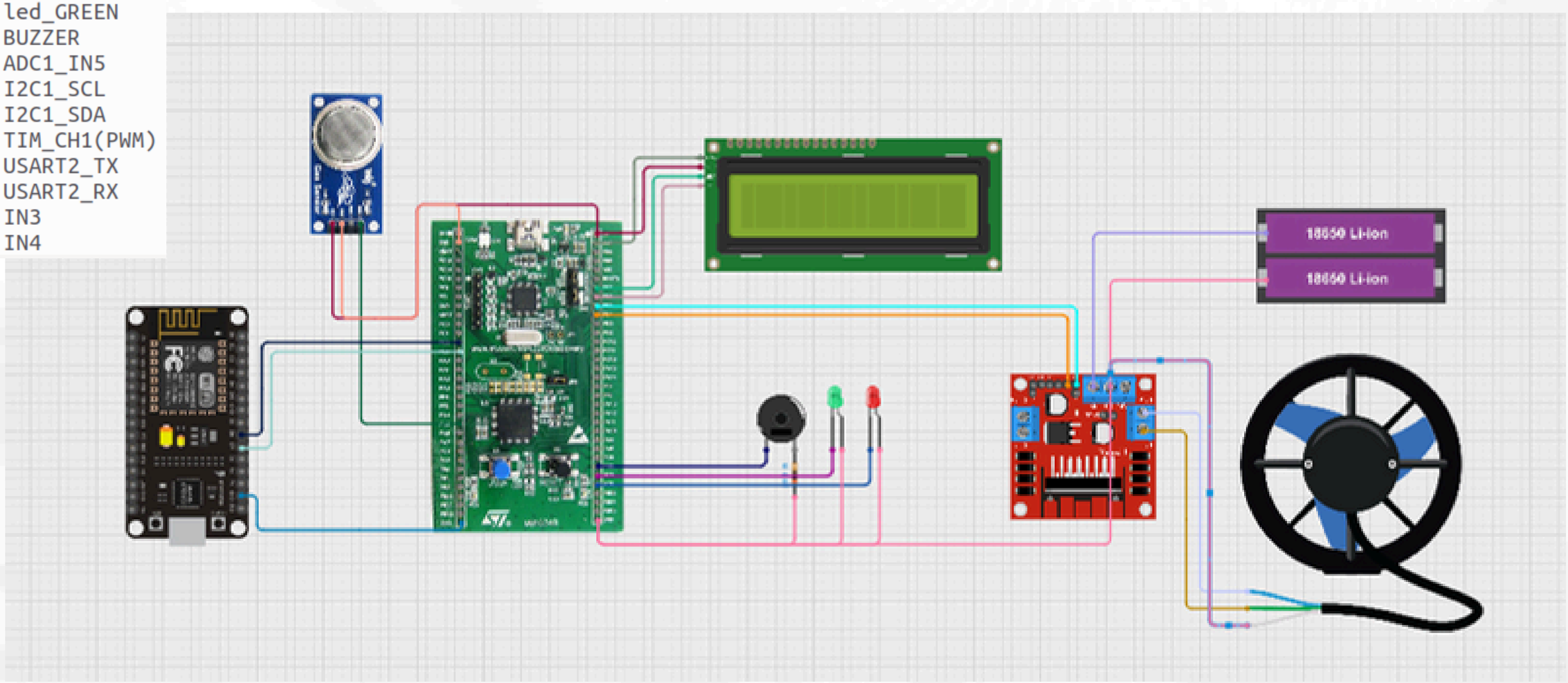
1. Gas Sensor: MQ-2 or MQ-135 for detecting gas concentrations (LPG, methane, smoke).
2. Microcontroller: STM32F407 (for processing and control logic).
3. Fan: DC fan with PWM control to adjust speed based on gas concentration.
4. LCD Display: 16x2 LCD for displaying gas levels and alerts.
5. Buzzer: Piezo buzzer for audible alarms.
6. LED/Lamp: Red LED or warning light for visual alerts.
7. Wifi module .

Software Components:

1. FreeRTOS: For task scheduling and real-time operation.
2. CubeMX for pin configuration
3. CubeIDE for programming the stm32 testing it and debugging.
4. Arduino IDE for programming the esp8266 and interfacing with Blynk.

CIRCUIT DIAGRAM
RT SMOKE DETECTION SYSTEM

PC6 led_RED
PC7 led_GREEN
PC8 BUZZER
PA5 ADC1_IN5
PB6 I2C1_SCL
PB7 I2C1_SDA
PE9 TIM_CH1(PWM)
PA2 USART2_TX
PA3 USART2_RX
PB5 IN3
PB4 IN4



MOTOR CONTROL WITH L298N

- **Functionality:**

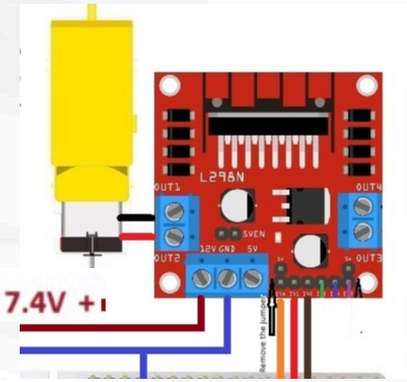
The H-Bridge allows control not only of the motor's direction but also its speed by modulating the voltage applied to the motor.

- **Polarity and Direction:**

Based on the signals sent to the inputs of the H-Bridge (IN1, IN2, IN3, IN4), it will reverse the polarity of the voltage applied to the motor, thereby changing the direction of rotation.

- **Speed (PWM):**

The PWM signal is applied to the ENB (Enable) pin or the speed control pin. This PWM signal modulates the average voltage applied to the motor, depending on the duty cycle



Features:

- Choice of Duty Cycle:
- Interval: The duty cycle value varies between 0 and TIM_PERIOD. For example, if TIM_PERIOD = 10 and dutyCycle = 5, the motor will receive 50% of the average supply voltage, which is approximately 3.7 V (50% of 7.4 V)
- The PWM frequency must be high enough to avoid motor vibrations while remaining compatible with the response times of the electronic transistors in the H-bridge: we chose to set it with a base frequency of 1 MHz (system clock =50MHz and the timer1 prescaler =50) and a period of 10 :

$$f_{PWM} = \frac{1}{10 \times 10^{-6}} = 100\,\text{kHz}$$

Pin Configuration:

- **ENB : PWM Output (TIM1_CH1).**
- **IN3; IN4: Motor Direction(forward/reverse).**
- **GND: Ground.**
- **VCC :Power supply 7.4v**

LCD DISPLAY I2C 2*16

- **Functionality:**

A 16x2 I2C LCD display is a commonly used screen in embedded systems, particularly for displaying small amounts of data like text. The "16x2" refers to the fact that the display has 2 rows and 16 columns.

Pin Configuration:

- **While the LCD itself has 16 pins, the I2C LCD module simplifies the connection by using an I2C interface with an external I2C controller, an PCF8574 . This reduces the number of connections to only 4 essential pins:**
- **VCC (Power Supply): This pin is used to provide power to the LCD.**
- **GND (Ground): The ground pin connects to the system ground.**
- **SDA (Data): This is the data line used to send and receive data between the microcontroller and the LCD.**
- **SCL (Clock): This is the clock line used to synchronize data transmission.**

Features:

- Addressing: I2C LCD modules often have a SLAVE_ADDRESS_LCD 0x4E .
- Basic functionalities of the LCD:

1. Display Text: It can display characters (alphabets, numbers, symbols) on the screen.
2. Clear Screen: You can clear the screen .
3. Cursor Positioning: You can move the cursor to specific locations.

MQ135 Air Quality Sensor

SENSOR INTEGRATION
MQ135 AIR QUALITY SENSOR

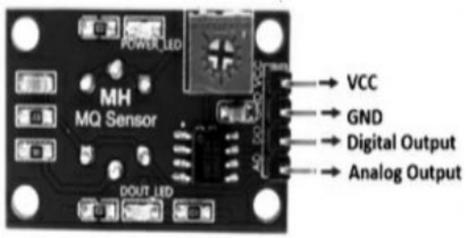
- **Functionality:** Detects gases like NH3, CO2, smoke, alcohol, benzene, and NOx.
- **Operating Voltage:** 5V; analog output voltage range: 0-5V.
- **Sensitivity**: High sensitivity with wide detection scope, faster response, and long-term stability.
- **Threshold Mechanism:**
- Digital Output Pin: Goes high when gas exceeds threshold (adjustable via potentiometer).
- Analog Output Pin: Provides proportional gas level in PPM.

PPM Measurement:

- Uses resistance (Rs) variation with gas concentration.
- Formula: Rs=(Vc/VRL)−1RLRs = \frac{(Vc/VRL) - 1}{RL}Rs=RL(Vc/VRL)−1.
- Calibration with graph between Rs/Ro and PPM.

Features:

- Preheating: 20 seconds required before operation.
- Operating Conditions:
  - Temperature: -10°C to 45°C.
  - Humidity: <95% RH.
- Sensitivity adjustment: Via potentiometer.



MQ135 Air Quality Sensor Pin Configuration

Pin Configuration:

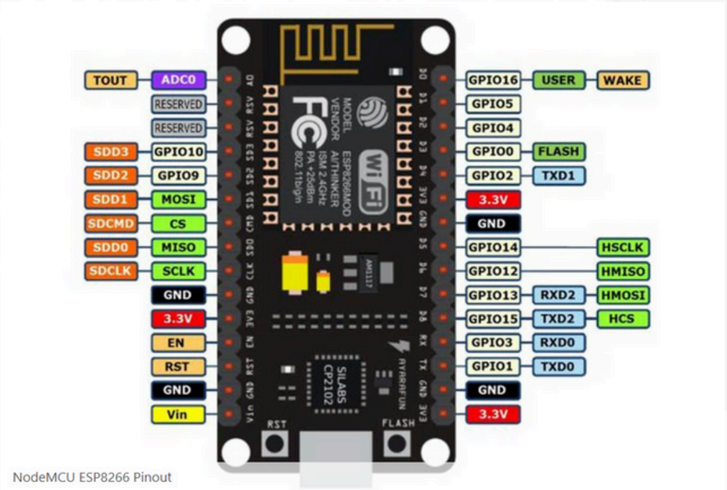
- **VCC: Power supply (+5V).**
- **GND: Ground.**
- **Digital Out (Do): Adjustable threshold digital signal.**
- **Analog Out (Ao): Proportional gas concentration (PPM)**

WIFI CONNECTIVITY
NODEMCU ESP8266

- **NodeMCU** is an open-source Lua based firmware and development board specially targeted for IoT based Applications. It includes firmware that runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module.

- **Pins used for the project :**

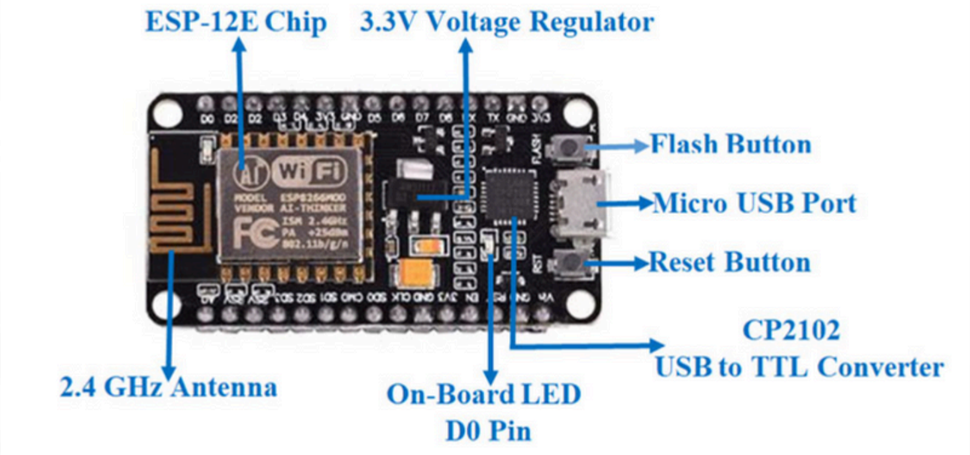
UART Pins	TXD0, RXD0, TXD2, RXD2	NodeMCU has two UART interfaces, UART0 (RXD0 & TXD0) and UART1 (RXD1 & TXD1). UART1 is used to upload the firmware/program.



NodeMCU ESP8266 Pinout

Used to connect the esp8266 to the stm32 with commun GND

Note: while working with arduino IDE , we have to select the right board from board manager. **Tools>Boards>NodeMCU1.0** (ESP-12E Module)



TASK SCHEDULING WITH FREERTOS

Task_1 (The Highest Priority : 3) : Producer of Semaphore
Role:

- Read sensor data, analyze values to detect smoke: check the threshold (2000PPM) to determine whether an alert state or a normal state should be reported.
- Send sensor data to an ESP module.

Tip: The semaphore ensures that each task:

- **Acts only when its corresponding event occurs.**
- **Waits its turn to act based on the global state.**
- **Avoids conflicts by preventing lower-priority tasks from interfering with critical ones.**

This ensures global synchronization of the system, even in a complex multitasking environment.

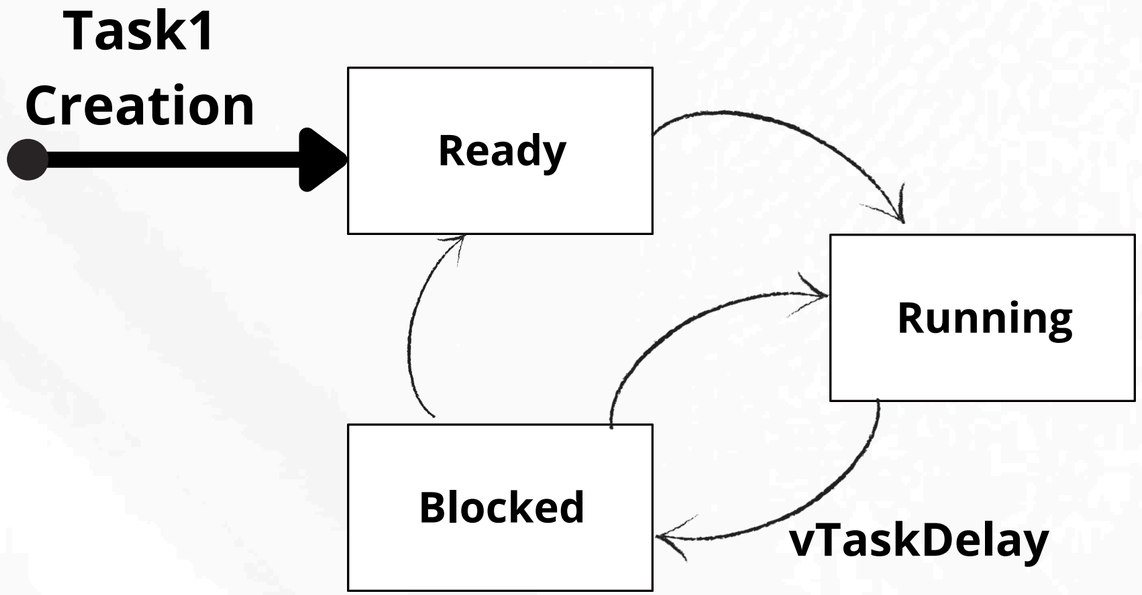
Task_3 (Priority :2) :Consumer of Semaphore
Role:
Primary function:
Manage alerts in case of smoke or fire detection :

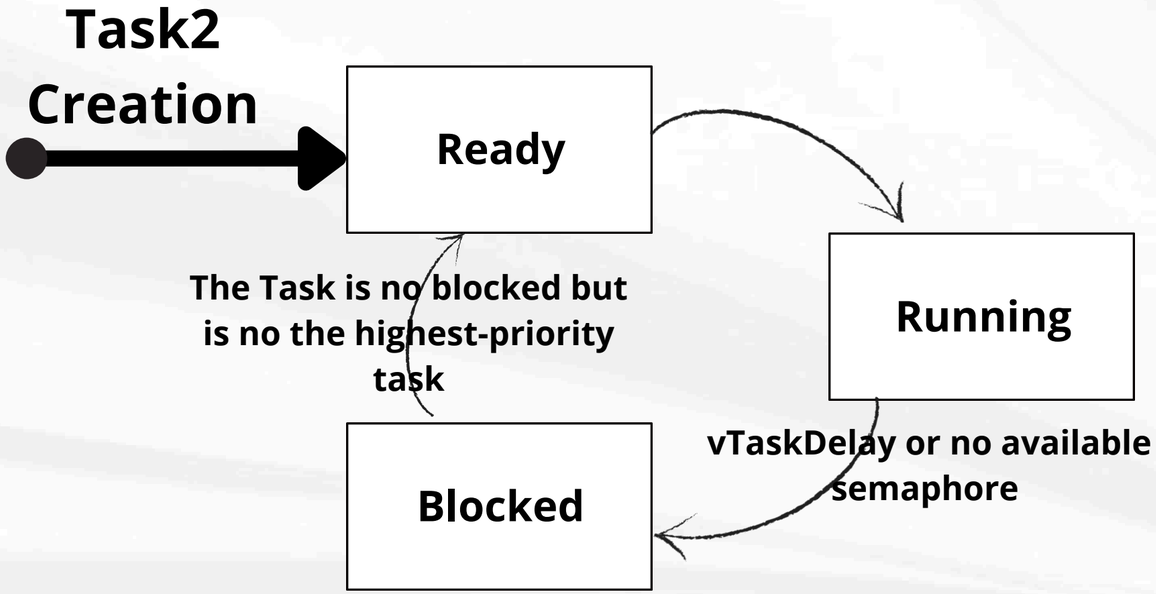
- Execute commands to signal a warning state:
  - Start a motor.
  - Turn on the red LED and the buzzer, and turn off the green LED.

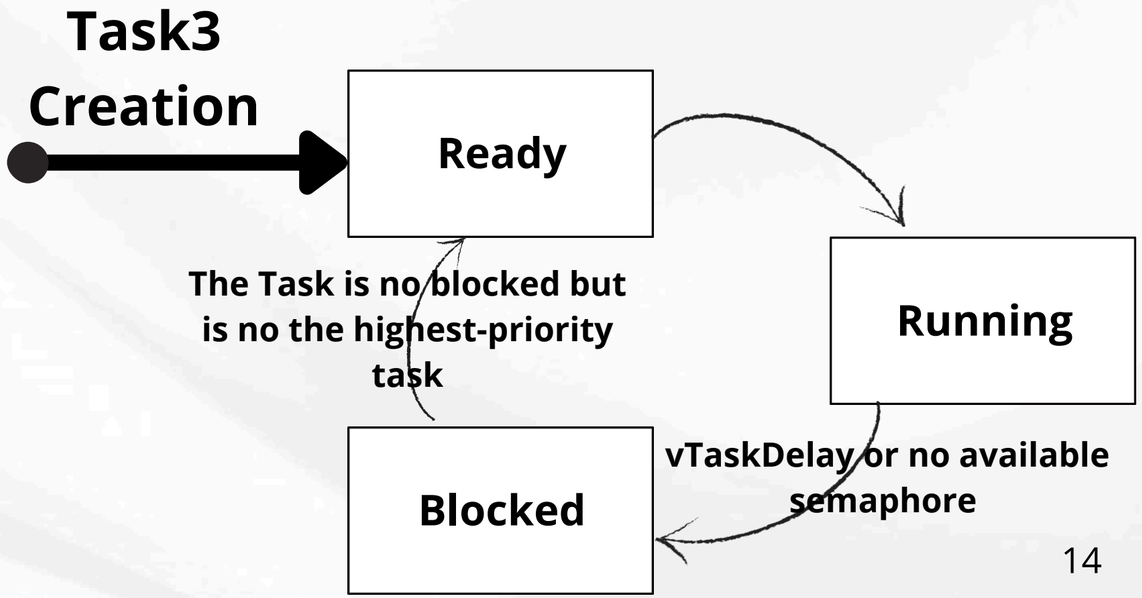
Task_2 (Priority :1) :Consumer of Semaphore
Role:
Primary function:
Handle normal state :

- Execute commands to signal a no_warning state:
  - Stop a motor.
  - Turn on the green LED and turn off the red LED and buzzer.

.


Task1
Creation
Ready
Running
Blocked
vTaskDelay


Task2
Creation
Ready
Running
The Task is no blocked but
is no the highest-priority
task
Blocked
vTaskDelay or no available
semaphore


Task3
Creation
Ready
Running
The Task is no blocked but
is no the highest-priority
task
Blocked
vTaskDelay or no available
semaphore

Normal State

Behavior for each task:

1. **Task_1 (Priority 3):**

- Reads data from the sensor via read_data_sensor().
- value < 2000:
    i. Sets the variable b to 1 (indicating no danger detected).
    ii. Does not give a semaphore.
- Continues sending data to the ESP module via send_data_to_esp().
- Enters a Blocked state for 100 ms.

2. **Task_3 (Priority 2):**

- Remains blocked, waiting for a semaphore with xSemaphoreTake(Binary_Sem, portMAX_DELAY).

3. **Task_2 (Priority 1):**

- Proceeds directly to NORMAL_NO_FIRE(), which handles the normal situation without gas detection.
- Enters a Blocked state for 100 ms.

Warning_State 

Behavior for each task:

1. **Task_1:**

- Reads data from the sensor using read_data_sensor().
- value is greater than 2000:
  i. Sets b = 2 (indicating danger detected).
  ii. Gives a semaphore using xSemaphoreGive(Binary_Sem).
- Sends the data to the ESP module with send_data_to_esp().
- Enters a blocked state for 100 ms.

2. **Task_3:**

- Takes the semaphore using xSemaphoreTake(Binary_Sem, portMAX_DELAY).
- Calls WARNING_HAVE_FIRE(), signaling an alert state.
- Updates the state with state = b.
- Enters a blocked state for 10 ms.

3. **Task_3:**

- Remains in a blocked state, waiting for a semaphore with xSemaphoreTake(Binary_Sem, portMAX_DELAY).

Transition from warning to Normal State

Task_1:

- When value < 2000 (indicating more gas is detected), it sets b to 1.
- If state == 2 && b == 1, this signifies a transition from "gas detected" to "no gas detected". In this case, the variable a is set to 1.

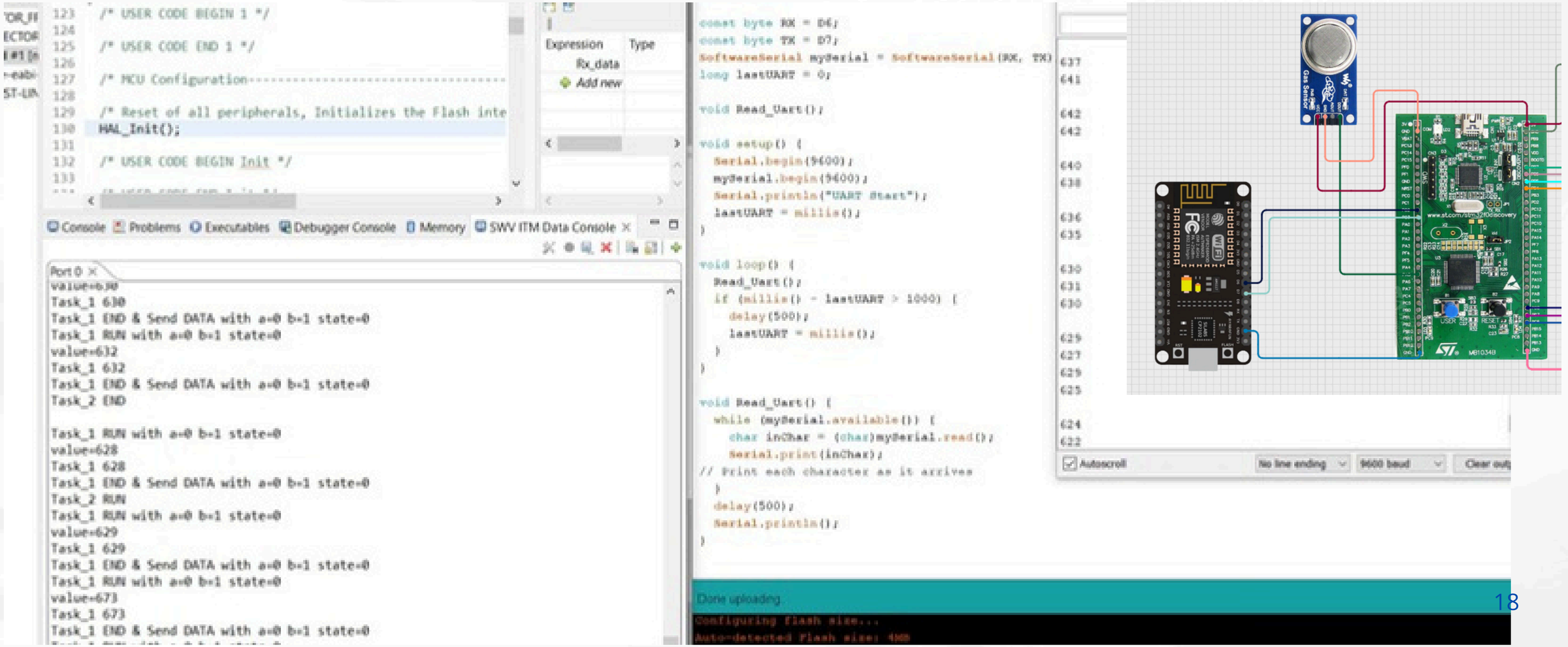
Task_3:

- When a == 1 (indicating the transition from "gas detected" to "no gas detected"), it gives the semaphore using xSemaphoreGive(Binary_Sem) to Task_2 and enters a blocked state via Task_Delay.

Task_2:

- Task_2 takes the semaphore and executes NORMAL_NO_FIRE() to return to a normal state.

UART COMMUNICATION
STM32F407 => ESP8266



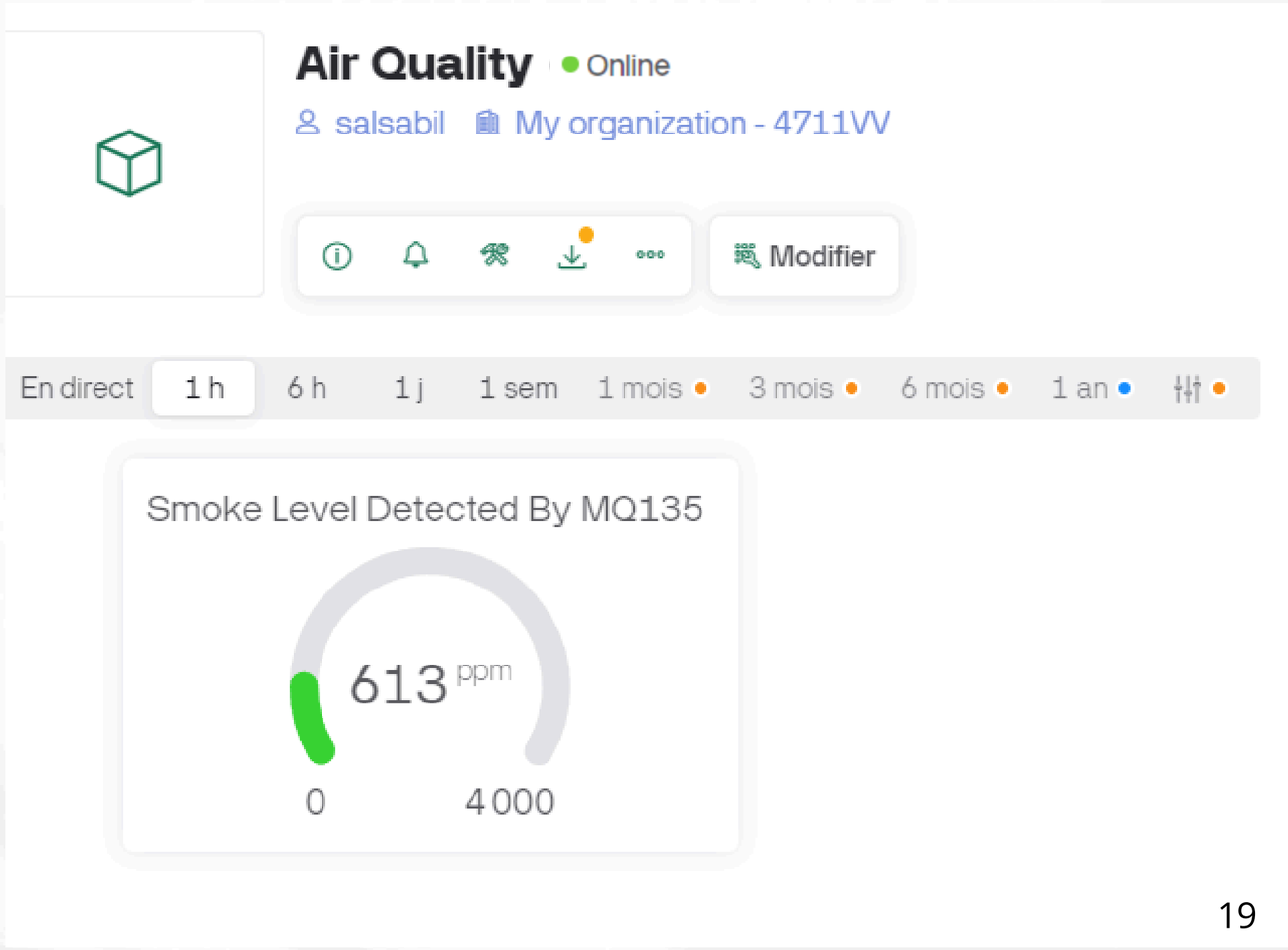
BLYNK DASHBOARD
NODEMCU ESP8266 FOR WIFI

Blynk.Console

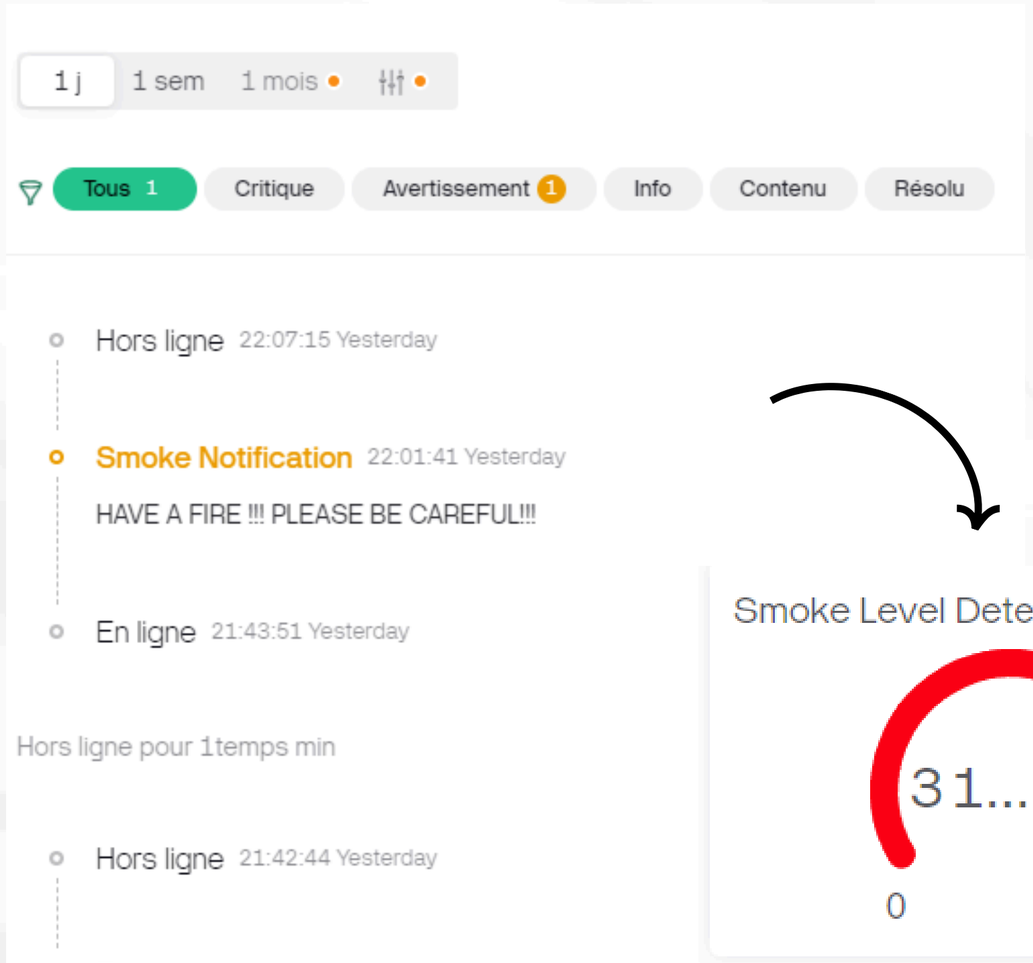
Brief Introduction to Blynk Platform
Blynk is an IoT platform for controlling and monitoring devices via a mobile app or web dashboard.
Key Features:

- Real-Time Monitoring: Tracks sensor data like gas levels.
- Notifications: Sends alerts via push notifications or email.
- User-Friendly: Drag-and-drop widgets for custom dashboards.
- Cloud Integration: Stores and processes data using Blynk Cloud.

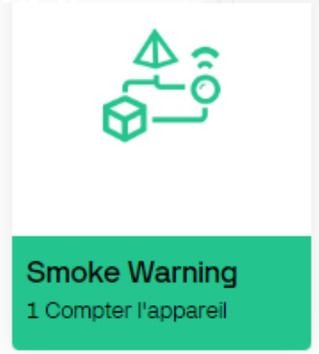
In our Project:
Tracks MQ135 sensor levels and sends alerts when gas exceeds 2000 ppm.

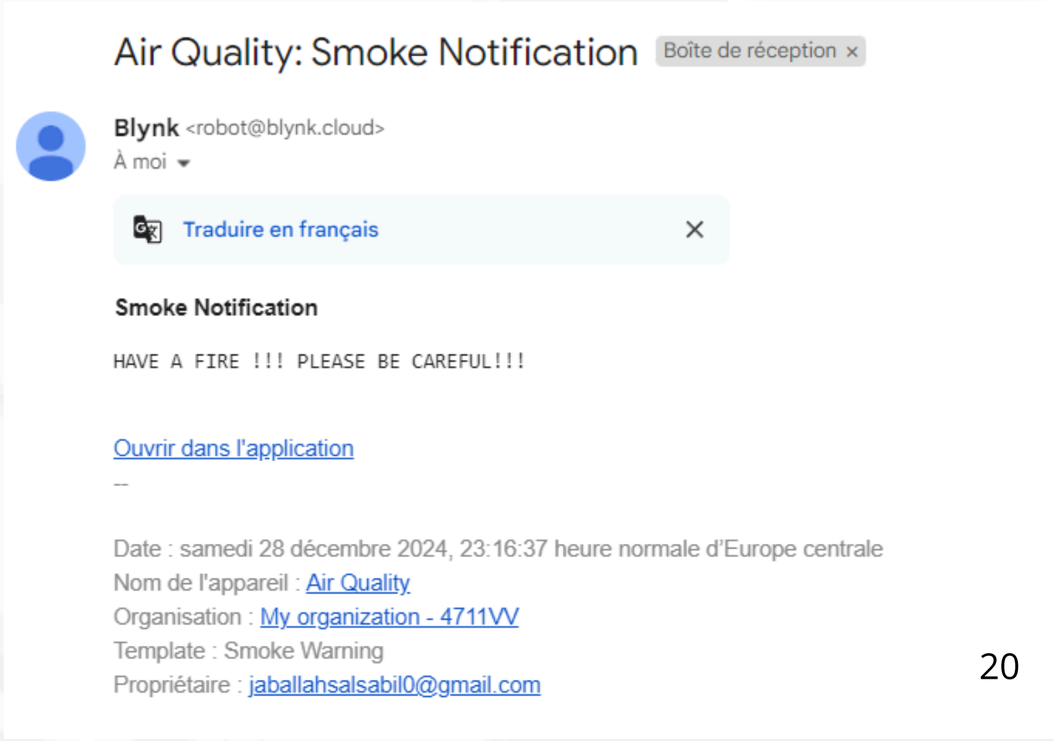


BLYNK DASHBOARD
NODEMCU ESP8266 FOR WIFI



IF THE SENSOR CATCHES MORE THAN 2000 PPM WE'LL GET A WARNING VIA MAIL





EXPLANATION OF THE ESP CODE

```
#include <SoftwareSerial.h>
#define BLYNK_TEMPLATE_ID "TMPL2lcXnxsET"
#define BLYNK_TEMPLATE_NAME "Smoke Warning"
#define BLYNK_AUTH_TOKEN "VfNOJUIMusStNqPhdL-RexIuAIE0OOwk"
#define BLYNK_PRINT Serial

#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <SoftwareSerial.h>

char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "TOPNET_DQGD";
char pass[] = [redacted]

const byte RX = D6;
const byte TX = D7;
SoftwareSerial mySerial = SoftwareSerial(RX, TX);
long lastUART = 0;
BlynkTimer timer;
int numericValue = 0;

void myTimerEvent()
{
  Blynk.virtualWrite(V0, numericValue);

 if (numericValue >2000 ) {
    Blynk.logEvent("smoke_notification", "HAVE A FIRE !!! PLEASE BE CAREFUL!!!");
  }
}
```

- **#include <SoftwareSerial.h>:** Enables serial communication on pins other than hardware UART
- **#define BLYNK_TEMPLATE_ID, BLYNK_TEMPLATE_NAME, BLYNK_AUTH_TOKEN**: Defines project-specific credentials for Blynk Cloud.
- **#define BLYNK_PRINT Serial**: Enables debug messages over hardware Serial.
- **BlynkTimer**: Used to run periodic tasks, such as updating sensor data to the cloud.
- **numericValue:** Stores the sensor's numeric value for processing and alerts.
- **Blynk.virtualWrite(V0, numericValue):** Sends numericValue to the virtual pin V0 in Blynk.
- **if (numericValue > 2000)**: Checks if the smoke concentration exceeds the threshold (2000 ppm).
- **Blynk.logEvent:** Triggers a notification alert.

EXPLANATION OF THE ESP CODE

```
void setup()
{
  Serial.begin(9600);
  mySerial.begin(9600);
  Serial.println("UART Start");
  lastUART = millis();
  // Connexion à Blynk
  Blynk.begin(auth, ssid, pass);
  // Timer pour envoyer des données toutes les 500 ms
  timer.setInterval(500L, myTimerEvent);
}
void loop()
{
  Blynk.run();
  timer.run();
  String uartData = "";
  while (mySerial.available()) {
   char  inChar = (char)mySerial.read();
    uartData += inChar;
  }
   Serial.print(uartData);
   Serial.println();
   delay(500);
   numericValue = uartData.toInt();
   // Envoie les données accumulées au widget virtuel V0
   Blynk.virtualWrite(V0, numericValue);

  }
```

void setup ()

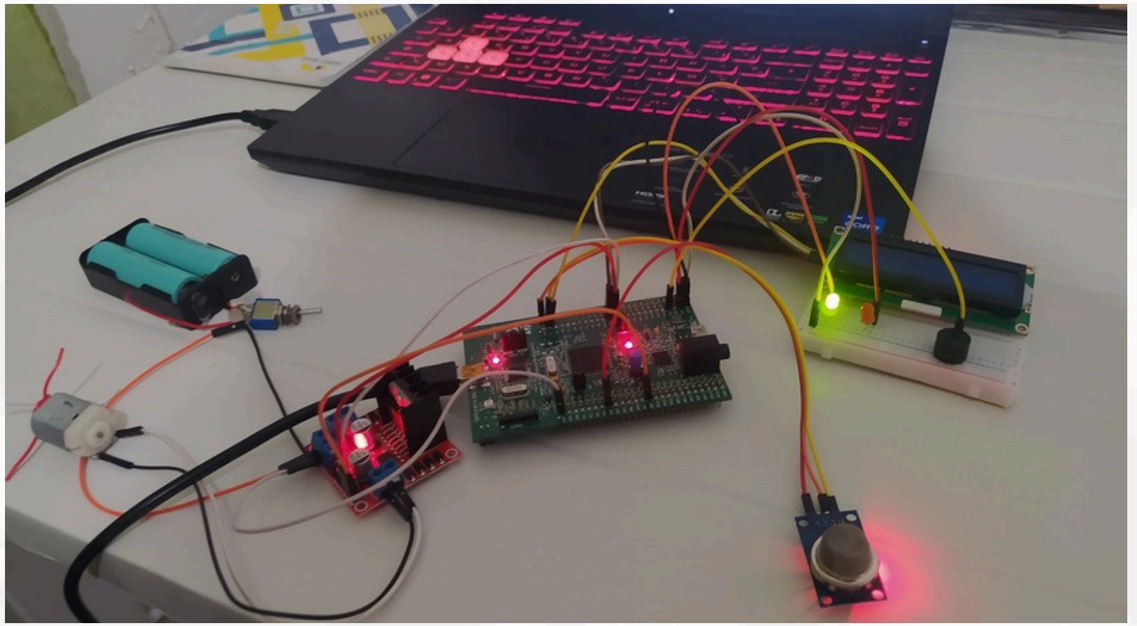
1. **Serial.begin(9600)**: Initializes hardware UART for debugging.
2. **mySerial.begin(9600):** Initializes software UART for the sensor.
3. **Blynk.begin**: Connects the ESP8266 to Blynk using WiFi credentials.
4. **timer.setInterval(500L, myTimerEvent):** Sets a 500 ms timer to run myTimerEvent.

void loop ()

1. **Blynk.run():** Ensures continuous communication with the Blynk cloud.
2. **timer.run()**: Executes tasks at intervals.
3. **while (mySerial.available()):** Reads incoming data from the sensor.
4. **uartData.toInt()**: Converts received string data into an integer for processing.
5. **Blynk.virtualWrite(V0, numericValue):** Updates the Blynk dashboard with the latest sensor value.

CONCLUSION

The Smoke Detection and Response System successfully demonstrates a real-time monitoring solution that detects dangerous smoke levels and provides timely alerts and responses.



FUTURE PLANS

01 Sensor Expansion: Add more sensors from the MQ series (e.g., MQ2, MQ5) to detect a wider range of gases.

02 System Control: Enable user control via the dashboard, such as turning sensors on/off or adjusting thresholds.

03 Advanced Dashboard: Expand the Blynk dashboard to display all sensor readings and provide a more detailed analysis.

04 Motor Control: Automate fan speed adjustments based on cumulative air quality data.

05 AI Integration: Explore using AI to predict hazardous conditions based on sensor trends.

THANK YOU FOR YOUR TIME, ATTENTION, AND VALUABLE GUIDANCE.

We look forward to your feedback and suggestions to further enhance our project.