Documentation: ESP32 Gas Sensor Monitor with FreeRTOS, AWS IoT, and BLE

Overview

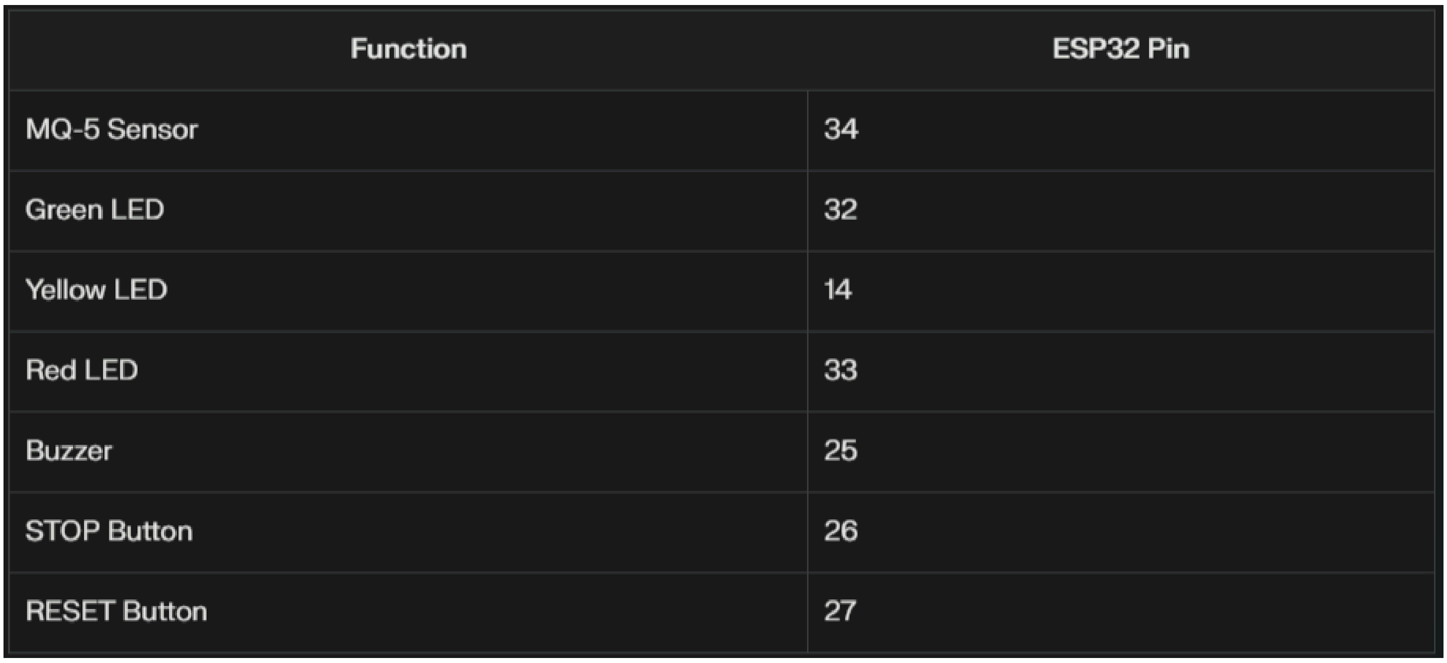
This project implements a multi-interface gas monitoring system on an ESP32 using FreeRTOS. It measures gas concentration with an MQ-5 sensor, displays readings on an LCD, signals alerts via LEDs and a buzzer, communicates over BLE, and publishes data/events to AWS IoT Core via MQTT. The system features robust alerting, remote control (reset/stop buzzer), and a web UI for monitoring.

System Architecture

Key Components:

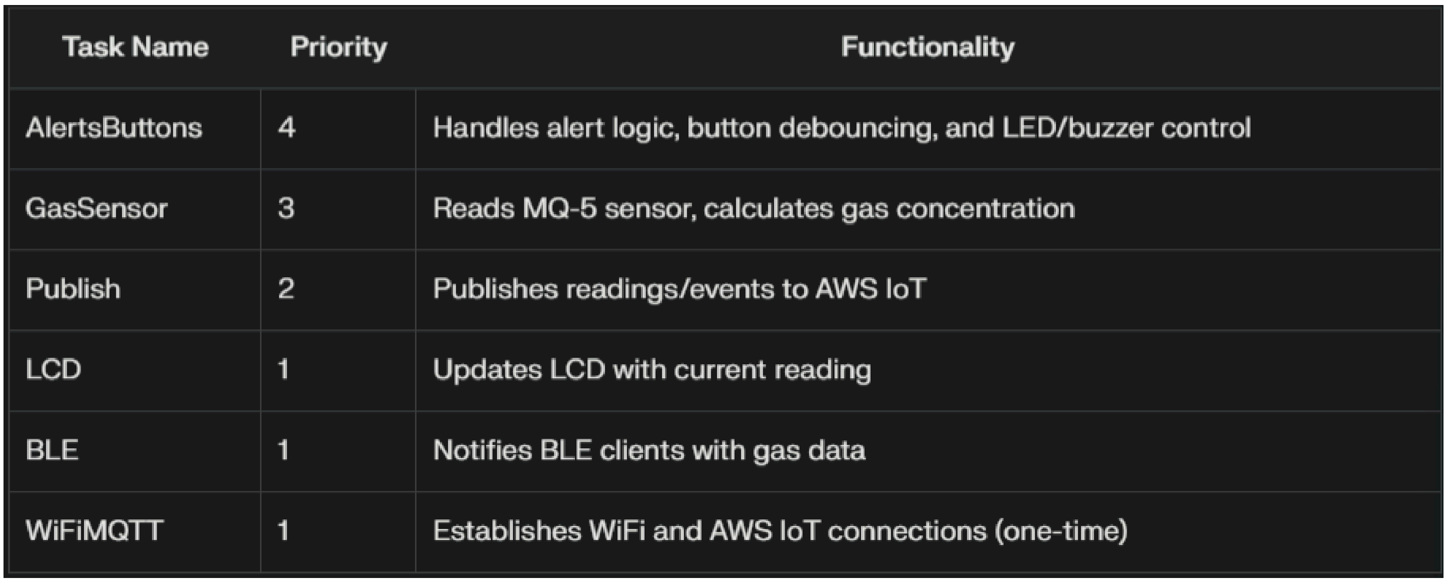
- **ESP32 Microcontroller**: Central controller running FreeRTOS.
- **MQ-5 Gas Sensor**: Detects gas concentration.
- **LCD Display**: Shows real-time gas readings and status.
- **LEDs (Green/Yellow/Red)**: Visual gas-level indicators.
- **Buzzer**: Audible alarm for dangerous gas levels.
- **Buttons (STOP/RESET)**: Manual controls for alert muting and device reset.
- **BLE**: Broadcasts gas data to nearby devices.
- **WiFi + AWS IoT**: Publishes readings/events to the cloud.
- **Web Interface**: Real-time remote monitoring and control.

Pin Assignments



Function	ESP32 Pin
MQ-5 Sensor	34
Green LED	32
Yellow LED	14
Red LED	33
Buzzer	25
STOP Button	26
RESET Button	27

FreeRTOS Task Structure



Task Name	Priority	Functionality
AlertsButtons	4	Handles alert logic, button debouncing, and LED/buzzer control
GasSensor	3	Reads MQ-5 sensor, calculates gas concentration
Publish	2	Publishes readings/events to AWS IoT
LCD	1	Updates LCD with current reading
BLE	1	Notifies BLE clients with gas data
WiFiMQTT	1	Establishes WiFi and AWS IoT connections (one-time)

Core Functional Modules

1. Sensor Calibration & Reading

- **Calibration (calibrateMQ5)**:
  - Samples MQ-5 sensor 50 times in clean air.
  - Calculates baseline resistance (Ro) using clean air ratio.
- **Reading (taskGasSensor)**:
  - Continuously samples sensor, computes resistance, and calculates gas concentration (as PPM).

2. Alert Logic (updateAlerts)

- **Thresholds**:
  - **Safe**: Below 200 PPM (Green LED)
  - **Warning**: 200–400 PPM (Yellow LED, intermittent buzzer)
  - **Danger**: Above 400 PPM (Red LED, continuous buzzer)
- **Manual Mute**:
  - STOP button disables buzzer/alerts until gas level is safe again.
- **Automatic Reset**:
  - RESET button re-calibrates sensor, reconnects WiFi/AWS, and restores normal operation.

3. AWS IoT Integration

- **WiFi Connection (connectToWiFi)**:
  - Connects to local WiFi, displays status on LCD.
- **MQTT Setup (connectAWS)**:
  - Establishes secure MQTT connection to AWS IoT Core using device certificates.

- **Publishing (taskPublish/publishMessage)**:
  - Sends JSON payloads (gas concentration + event info) to AWS every second.

4. BLE Integration

- **Service/Characteristic**:
  - BLE service broadcasts gas readings (read/notify).
- **Callbacks**:
  - LCD and serial log connection/disconnection events.

5. LCD UI (taskLCD)

- **Display**:
  - Shows current gas concentration and system status.

6. Web Interface (HTML/JS)

- **Features**:
  - Real-time gas reading display with color-coded status.
  - Connection status indicator.
  - Buttons for remote STOP/RESET (if implemented).
  - Uses WebSocket or MQTT for live updates

Key Functions and Their Roles

setupBLE(): Initializes BLE server, service, and characteristic.

calculateResistance(): Converts analog sensor value to resistance.

calibrateMQ5(): Performs sensor calibration and calculates Ro.

publishMessage(): Publishes gas data/events to AWS IoT Core.

connectToWiFi(): Connects to WiFi, handles failures with restart.

connectAWS(): Sets up secure MQTT connection to AWS.

updateAlerts(): Controls LEDs/buzzer based on gas level and STOP state.

performReset(): Re-calibrates sensor, reconnects WiFi/AWS, resets alerts.

handleButton(): Debounces and handles STOP button.

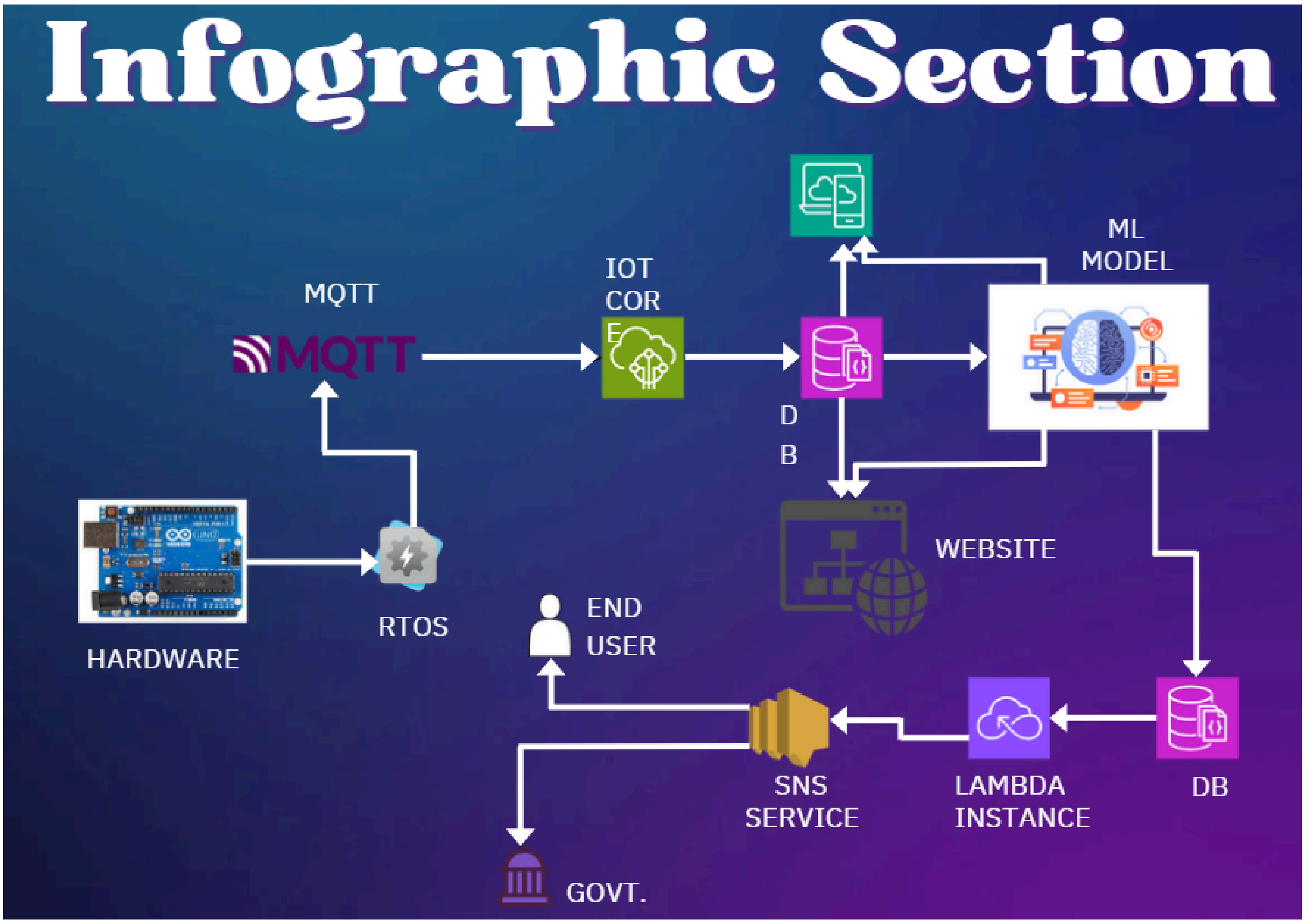
handleResetButton(): Debounces and handles RESET button.

Initialization and Task Scheduling

- In setup():
  - Serial, LCD, BLE, pins, and PWM for buzzer are initialized.
  - MQ-5 is calibrated.
  - All FreeRTOS tasks are created with appropriate priorities.

- In loop():
  - Empty; all logic is handled by FreeRTOS tasks.

Workflow



Setting-up AWS IoT Core

Create a "Thing"

1. Go to the **AWS IoT Core** console.
2. In the left menu, click **"Manage" → "Things" → "Create things"**.
3. Choose **"Create a single thing"**.
4. Enter a **Thing Name** (e.g., mq6_esp32_device).

5. Click **Next**.

Create Certificates and Keys

1. Choose **"Auto-generate a new certificate"**.
2. Download the following files:
    - **Device Certificate**
    - **Private Key**
    - **Public Key**
    - **Amazon Root CA 1**

Attach a Policy

1. After certificate generation, click **"Attach a policy"**.
2. If you don’t have one, click **"Create a policy"**.
3. Attach the policy to the certificate.

Attach Certificate to the Thing

1. Go to **"Things" → Your Thing**.
2. Under the **"Security"** tab, click **"Attach"** to link the certificate you just created.

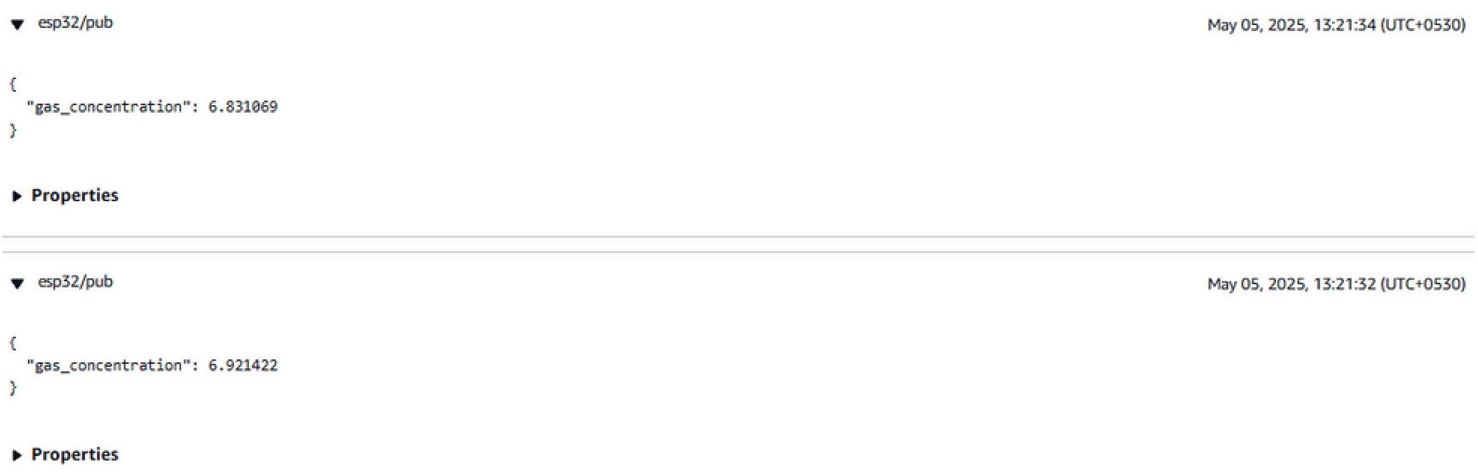
Note Your MQTT Endpoint

1. In the AWS IoT Core console, go to **"Settings"**.
2. Copy the **"Endpoint"** – you’ll need this in the ESP32 code (usually looks like: a3k7odshdkjf.iot.us-west-2.amazonaws.com).

Step 6: Test with MQTT Test Client

1. Go to **"MQTT test client"** in the AWS IoT console (left sidebar).
2. Under **"Subscribe to a topic"**, enter the topic your device will publish to (e.g., mq6/sensor/data).
3. Click **Subscribe**.
4. Once your ESP32 is running and publishing, messages will appear here in real-time.

Sample AWS IoT JSON Payload



▼ esp32/pub May 05, 2025, 13:21:34 (UTC+0530)

```
{
  "gas_concentration": 6.831069
}
```

▶ Properties

▼ esp32/pub May 05, 2025, 13:21:32 (UTC+0530)

```
{
  "gas_concentration": 6.921422
}
```

▶ Properties

Web Interface Features

- **Live Reading**: Color-coded (safe/warning/danger).
- **Status**: Online/offline indicator.
- **Control**: STOP buzzer and RESET device buttons (if supported).
- **Connection**: WebSocket or MQTT for real-time updates.

Extensibility

- **Add more sensors** by expanding the sensor reading task.
- **Enhance web UI** for historical data, charts, or user authentication.
- **Integrate more cloud services** (e.g., SMS/email alerts via AWS Lambda).

References

FreeRTOS

- **Official site:** https://www.freertos.org/
- **ESP32 + FreeRTOS:** ESP-IDF FreeRTOS

AWS IoT Core

- **AWS IoT Docs:** https://docs.aws.amazon.com/iot/latest/developerguide/
- **MQTT:** https://docs.aws.amazon.com/iot/latest/developerguide/mqtt.html

FreeRTOS + AWS IoT

- **CoreMQTT Lib:** https://github.com/FreeRTOS/coreMQTT
- **Getting Started:** AWS FreeRTOS User Guide