Livestock Gas Monitor - User Manual

Introduction

This document provides detailed instructions for setting up, operating, and maintaining the Livestock Gas Monitoring System using the ESP32 microcontroller, including interfacing, uploading firmware, Wi-Fi provisioning, and managing the AWS IoT infrastructure.

Requirements

- ESP32 Microcontroller
- USB-to-UART cable
- Laptop with ESP-IDF environment

Step-by-Step ESP-IDF Installation

Follow the official Espressif guide to set up ESP-IDF (ESP32 Development Framework) on your device:

ESP-IDF Setup Guide (Windows)

ESP-IDF Setup Guide (MacOS/Linux)

Key installation commands:

```
git clone --recursive https://github.com/espressif/esp-idf.git
cd esp-idf
./install.sh
source export.sh
```

Verify your installation:

```
idf.py --version
```

GitHub Repository and Code Structure

All final code versions are located here:

[ESP32 Gas Monitor Repository]

Repository Structure:

- **main:** Contains primary application logic.
- **tmp:** Contains AWS IoT public, private, and root cert auth keys.
- **components:** Contains reusable software modules.
    - **Note:** The `coremqtt` and `coreMQTT-Agent` libraries are external components required for MQTT functionality. They are not included directly in the repository but must be added manually into the project's `components` folder.

Code Explanation

main

- **Cap_GasMonitor.c:** Contains `app_main()` function, initiates system setup, sensor readings, and data transmission logic.
- **wifi.c/h:** Manages Wi-Fi initialization, connectivity, provisioning, and error-handling.
- **scd4x.c/h:** Driver for I²C digital Temperature, Humidity, and CO2 sensor.
- **i2cdev.c/h:** Allows for the Driver mentioned above to work seamlessly with my project.

components

- Place `coremqtt` and `coreMQTT-Agent` libraries here to ensure MQTT functionality.

tmp

- Client.crt, client.key, root_cert_auth that was obtained by creating a “thing” in AWS IoT.

Uploading Firmware to ESP32

1. **Connect ESP32:**
    - Connect your ESP32 microcontroller to your laptop via USB-to-UART cable (black: ground, yellow: RX, orange: TX).
2. **Build Firmware:**
   idf.py build
3. **Flash Firmware:**
   idf.py flash
4. **Monitor ESP32 Output:**
   idf.py monitor

Wi-Fi Provisioning

Wi-Fi provisioning enables the gas monitor to connect to any available Wi-Fi network.

Provisioning Steps:

1. **Erase Flash Memory:**
   idf.py erase_flash
2. **Upload Firmware (with provisioning enabled):**
   idf.py flash
3. **Monitor Terminal for QR Code:**
   idf.py monitor

A QR code will be generated and displayed in the terminal. Alternatively, visit:

Provisioning QR Code

Use the ESP SoftAP Prov App from the app store:

- Supported Device Type: Soft AP
- Encrypted Communication: ON
- Username: wifiprov

Scan the QR code, click "Join" on the Wi-Fi network, enter your Wi-Fi credentials, and complete provisioning.

AWS IoT Infrastructure

Your Livestock Gas Monitoring System securely transmits data to AWS IoT Core for cloud-based storage, management, and analysis.

AWS Account Handover

Since this project currently runs on Joaquin's AWS account, it is recommended that the sponsor takes over the AWS account to manage billing and maintain infrastructure:

AWS Root User Login:

- **User:** jsalas2002@tamu.edu
- **Password:** ECEN#404capstone

Log in to AWS console: AWS Console

Conclusion

This document aims to assist in smoothly transitioning maintenance responsibilities. Should further assistance be needed, consult the detailed comments in the GitHub repository or contact the original development team.

All the best,

Joaquin Salas - Electrical Engineering Student

(806) 340-9168

salasjoaquin07@yahoo.com