Project Report: Simple IoT Alcohol Monitoring System

Project Title: Real-Time Environmental Alcohol Concentration Monitoring using IoT
Platform: Arduino Mega 2560 Microcontroller

Key Components: MQ-3 Alcohol Sensor, GSM Module (SIM900), ThingSpeak IoT Platform

1. Introduction

This project created a **real-time system designed to detect and monitor airborne alcohol vapor concentration**. The primary goal is to enhance **safety assurance** in controlled settings, such as industrial fermentation areas, chemical laboratories, or to serve as a foundational component for advanced personal breathalyzers. The final system is a hybrid solution that combines two critical functions:

1. **Immediate Local Warning:** Provides an instant, localized hazard signal using an **LED and buzzer**.
2. **Continuous Remote Monitoring:** Performs **remote data logging** by sending measurements via a GSM/GPRS module to the ThingSpeak cloud platform.

This Internet of Things (IoT) capability allows supervisory personnel to track environmental safety trends and respond to alerts from any remote location.

2. System Architecture and Hardware

The system is built around the **Arduino Mega 2560** microcontroller. We chose the Mega because it has enough internal communication ports (UARTs) to handle the GSM module without interfering with the main port used for debugging.

Component	Function	Interface Pins
MQ-3 Alcohol Sensor	Detects alcohol vapor and gives an analog voltage output.	Analog Pin A0
LED Indicator	Visual alarm light.	Digital Pin 7
Buzzer	Auditory alarm.	Digital Pin 8
GSM Module (SIM900)	Provides mobile internet access (GPRS).	Hardware Serial 1 (Pins 18/19)

Sensor Data Conversion

The MQ-3 sensor produces a raw reading (0-1023). To make this value meaningful, the system converts it into **Parts Per Million (PPM)**, which is the standard unit for gas concentration.

The conversion process involves three steps:

1. **Calibration (R_o):** Reading the sensor's baseline resistance in clean air.
2. **Ratio:** Calculating the resistance ratio of the sensor when exposed to gas (R_s) versus its baseline (R_o).

3. **Mapping:** Using the sensor's known characteristics (a datasheet formula) to accurately convert this ratio into the final PPM value.

The system is set to trigger an alarm if the concentration exceeds a predetermined level, such as **200 PPM (**$\text{ALARM\_PPM\_THRESHOLD}$**)**.

3. Software Implementation and Data Flow

The software is written in C and runs two main processes simultaneously: managing the local alarm and handling remote data transmission.

3.1 Local Monitoring and Alarm Logic

The code constantly monitors the PPM level. The alarm logic is designed to prevent annoying, continuous ringing when alcohol levels are consistently high:

- If the PPM exceeds the threshold, the system triggers the **LED and buzzer for a single, precise 3-second alert**.
- After the 3-second alert, the system enters a **"safe" pause state**. It will not trigger the alarm again, even if the alcohol concentration is still high.
- The alarm will only be **reset and re-armed** once the concentration has dropped **below** the safe threshold, confirming the hazard has cleared.

3.2 GSM Internet Connection and Data Upload

The GSM module establishes an internet connection using a series of specialized commands known as **AT Commands** sent over the serial port. The general communication flow is:

1. **Network Setup:** Commands are sent to configure the mobile network and obtain a GPRS (mobile internet) connection.
2. **Server Connection:** A command is issued to establish a stable **TCP/IP link** directly to the **ThingSpeak** server.
3. **Data Transmission:** The system then formats the sensor data into an **HTTP GET request**—a standard web protocol—which includes the latest PPM value and the private ThingSpeak API key. This request is packaged and sent to the cloud, ensuring the data is logged in real-time.

4. Challenges and Future Improvements

4.1 Development Challenges

The project faced two main hurdles:

1. **Hardware Failure:** A major recurring issue was the inability to upload new code to the Arduino Mega due to a persistent **bootloader failure** (an $\text{stk500v2\_getsync()}$ timeout error). This was a fundamental hardware problem that could not be solved with software and required replacing the main microcontroller board.
2. **GSM Instability:** The GSM module required a **dedicated external power supply** because it draws too much current for the Arduino board. Additionally, achieving reliable network connection required painstaking fine-tuning of the **timing and sequence** of the AT commands.

4.2 Future Development

The system can be enhanced with:

- **Faster Responsiveness:** Currently, the system delays all processing while waiting for the GSM module. Refactoring the code to use **non-blocking communication** will allow the sensor to monitor at full speed while the data uploads in the background.
- **Redundant Alerts:** Implementing a feature to send an **SMS text message** alert to a specified number when the concentration is dangerously high, providing a critical secondary alert channel.
- **Permanent Calibration:** Storing the sensor's calibration settings in the board's **EEPROM memory**, allowing the system to be recalibrated without needing to recompile and re-upload the entire program.

5. Conclusion

The IoT Alcohol Monitoring System successfully combines localized safety measures with remote, cloud-based data tracking. Despite setbacks caused by the hardware failure, the project validated the effective integration of the MQ-3 sensor, local alarm logic, and the reliable GSM-ThingSpeak data pipeline. This solution demonstrates a foundation for robust, connected environmental monitoring applications.