# CHAPTER THREE

## SYSTEM ANALYSIS AND DESIGN

### 3.0 Methodology

The methodology for this project follows a structured engineering design and prototyping process to develop and validate an IoT-based power fault detection system. The approach is divided into distinct phases: system design and component selection, software development, hardware implementation, and system testing. This systematic process ensures that each component of the system is thoughtfully selected, properly integrated, and rigorously tested to meet the project's objectives of creating a reliable, low-cost, and scalable fault detection solution. The system is designed to directly monitor the electrical health of the power line and report anomalies in real-time using a combination of specialized sensors and communication technologies.

The development lifecycle involves the following key stages:

1. **System Design:** A multi-layered architecture was designed to segregate the core functions of data acquisition, local processing, and remote alerting. The choice of a master-slave configuration for the microcontrollers and the selection of SPI for inter-device communication were central to this design phase.
2. **Component Selection and Justification:** Hardware components were selected based on a balance of functionality, cost-effectiveness, and suitability for the task. The Arduino platform was chosen for its accessibility and robust community support. The ZMPT101B and ACS712 sensors were selected for their ability to provide direct and accurate measurements of voltage and current, respectively. The SIM900A GSM module was chosen for its reliability and the ubiquity of cellular networks for remote data transmission.
3. **Prototyping and Implementation:** The system was first simulated using Proteus software to verify the circuit design and logic. Following successful simulation, a physical prototype was assembled. Software for the master and slave Arduinos was developed in the Arduino IDE, focusing on efficient data sampling, reliable SPI communication, and the implementation of a precise fault detection algorithm.
4. **Testing and Validation:** The completed prototype will be subjected to a series of tests to validate its performance. Fault conditions, including overvoltage, undervoltage, and overcurrent, will be simulated using a variable power supply to test the accuracy of the detection algorithm and the reliability of the GSM alert system. Key performance metrics such as detection latency and alert delivery time will be measured.

### 3.1 System Architecture

This system is designed as a multi-layered IoT architecture for the timely detection and resolution of power faults. The architecture is divided into three functional layers: the Sensor and Edge Processing Layer for data acquisition, the Local Aggregation and Communication Layer for data processing and short-range communication, and the Remote Monitoring and Alerting Layer for long-range data transmission and user notification.

#### 3.1.1 Sensor and Edge Processing Layer

This layer forms the data collection backbone of the system. It consists of multiple specialized sensor nodes deployed at various points along a power line to monitor key electrical parameters that directly indicate a fault condition.

* **Sensor Nodes:** The system uses **Voltage Sensors (ZMPT101B)** and **Current Sensors (ACS712)** to collect real-time data on the electrical characteristics of the power grid. These sensors provide direct measurements of the line's operational status.
* **Edge Microcontrollers (Slaves):** Each sensor set (one voltage and one current sensor) is connected to a dedicated Arduino microcontroller. In this configuration, these Arduinos function as "slaves," responsible for reading the raw analog data from their assigned sensors and converting it into digital values for transmission.

#### 3.1.2 Local Aggregation and Communication Layer

This layer acts as the local data processing hub, where information from the sensor layer is collected, analyzed, and prepared for remote transmission.

* **Master Microcontroller:** A fourth Arduino serves as the "master" unit. Its primary role is to aggregate data from the three slave Arduinos.
* **SPI Communication:** The slave Arduinos transmit their collected voltage and current data to the master Arduino using the Serial Peripheral Interface (SPI) communication protocol. SPI is a synchronous interface that allows for high-speed, simultaneous data exchange over short distances, making it ideal for real-time communication between the microcontrollers.
* **Fault Detection and Display:** The master Arduino runs a fault detection algorithm that analyzes the aggregated data to identify specific fault conditions such as overvoltage, undervoltage, or overcurrent. System status and immediate fault indications are shown on a connected LCD display.
* **Data Relay:** After processing, the master Arduino relays the data and any generated alerts to the GSM module for long-range communication.

#### 3.1.3 Remote Monitoring and Alerting Layer

This layer manages the crucial task of transmitting fault alerts and data from the local site to a central monitoring station and relevant personnel.

* **GSM Module:** The system integrates a SIM900A GSM module to enable wireless communication over long distances using the cellular network. This makes the system effective for remote installations where other internet infrastructure may be unreliable.
* **Data Transmission:** The GSM module sends data to a central server. It can transmit critical alerts as SMS messages or send data packets using GPRS.
* **Central Server & Stakeholder Notification:** A central server receives and processes the data transmitted by the GSM module. Once a fault is confirmed, the system automatically sends alerts to designated stakeholders, such as maintenance teams, allowing for a rapid response to mitigate downtime and prevent potential damage.

Figure 3.1: Revised System Architecture Diagram

Description: The diagram shows three stacked horizontal blocks representing the system's layers. The bottom layer, "Sensor & Edge Processing," contains three sets of sensor icons (voltage and current) each connected to a "Slave Arduino." Arrows point up from the slaves to the middle layer, "Local Aggregation & Communication." This layer shows a central "Master Arduino" connected to an LCD display. The Master Arduino is also connected via an arrow to the top layer, "Remote Monitoring & Alerting." This top layer contains an icon for the "SIM900A GSM Module," which sends alerts via a cellular network to icons representing a "Central Server" and "Stakeholders/Maintenance Team." All connections between Arduinos are labeled "SPI."

### 3.2 Working and Connection Methodology

The working and connection methodology of the "IoT Based Power Fault Detection system in Mesh Networks" is as follows, utilizing three sensor nodes (each equipped with a voltage and current sensor), SPI, four Arduinos, and a SIM900A GSM module:

1. Connect the three sensor nodes (each with a ZMPT101B voltage sensor and an ACS712 current sensor) to their respective slave Arduinos. The Arduinos will continuously collect voltage and current data.
2. Connect the four Arduinos (three slaves, one master) using the SPI communication protocol. SPI will facilitate the seamless and high-speed exchange of sensor data from the slaves to the master.
3. Integrate the SIM900A GSM module with the master Arduino to enable wireless communication over long distances. The GSM module will transmit fault alerts from the master controller to remote stakeholders.
4. Implement a fault detection algorithm on the master Arduino. The algorithm will analyze the incoming voltage and current data to identify anomalies such as overvoltage, undervoltage, and overcurrent conditions.
5. Once a power fault is detected, the system will trigger an immediate alert, displaying the fault type on the local LCD and sending an SMS notification via the GSM module to the relevant stakeholders.
6. The system enables proactive measures by providing real-time data on the electrical health of the grid, helping to prevent widespread outages and reduce downtime. The complete connection of these components is shown in the schematic diagram in Figure 3.3.

### 3.3 Serial Peripheral Interface (SPI)

The serial peripheral interface (SPI) is a communication interface used to send data between multiple devices. These devices are organized into a master and slave configuration, in which the master has control over the slaves and the slaves receive instruction from the master. The most common implementation of SPI consists of a configuration in which a single device is the master, and the remainder of the devices are slaves. SPI is a synchronous communication protocol that transmits and receives information simultaneously with high data transfer rates and is designed for board-level communication over short distances.

#### 3.3.1 SPI Signals

There are four signals required to implement SPI communication, as listed in the table below, with all but the MISO line controlled by the master. Chip Select (CS), sometimes referred to as Slave Select (SS), is also often denoted as CS¯ or SS¯ because a particular chip/slave is active when that line is pulled low by the master.

| **Signal** | **Description** |
| --- | --- |
| MOSI | Data: Master-Out Slave In |
| MISO | Data: Master-In Slave Out |
| SCLK | Serial Clock |
| CS | Chip Select |

**Table 3.1:** SPI Signals

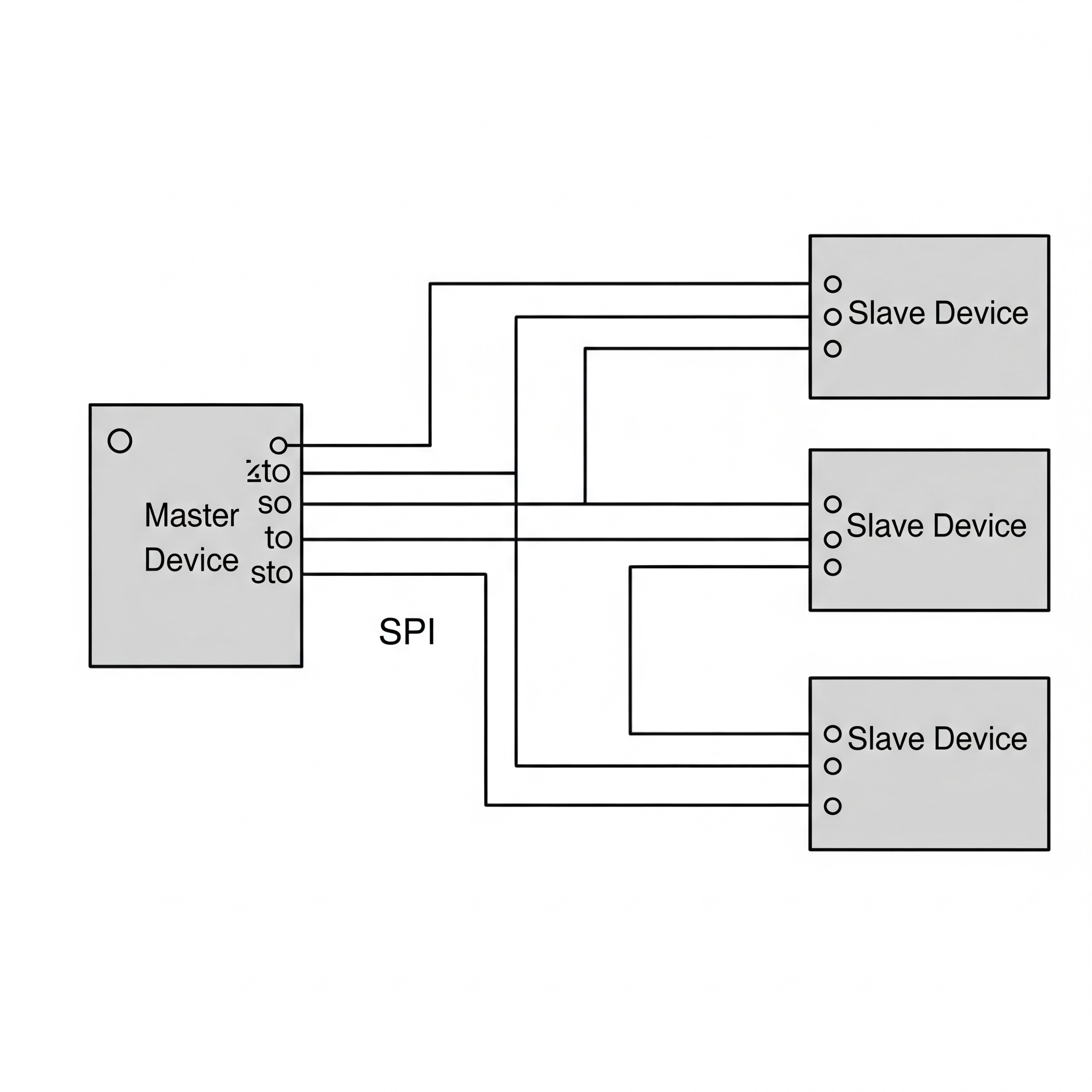


Figure 3.2: Master and Slave Inputs and Outputs

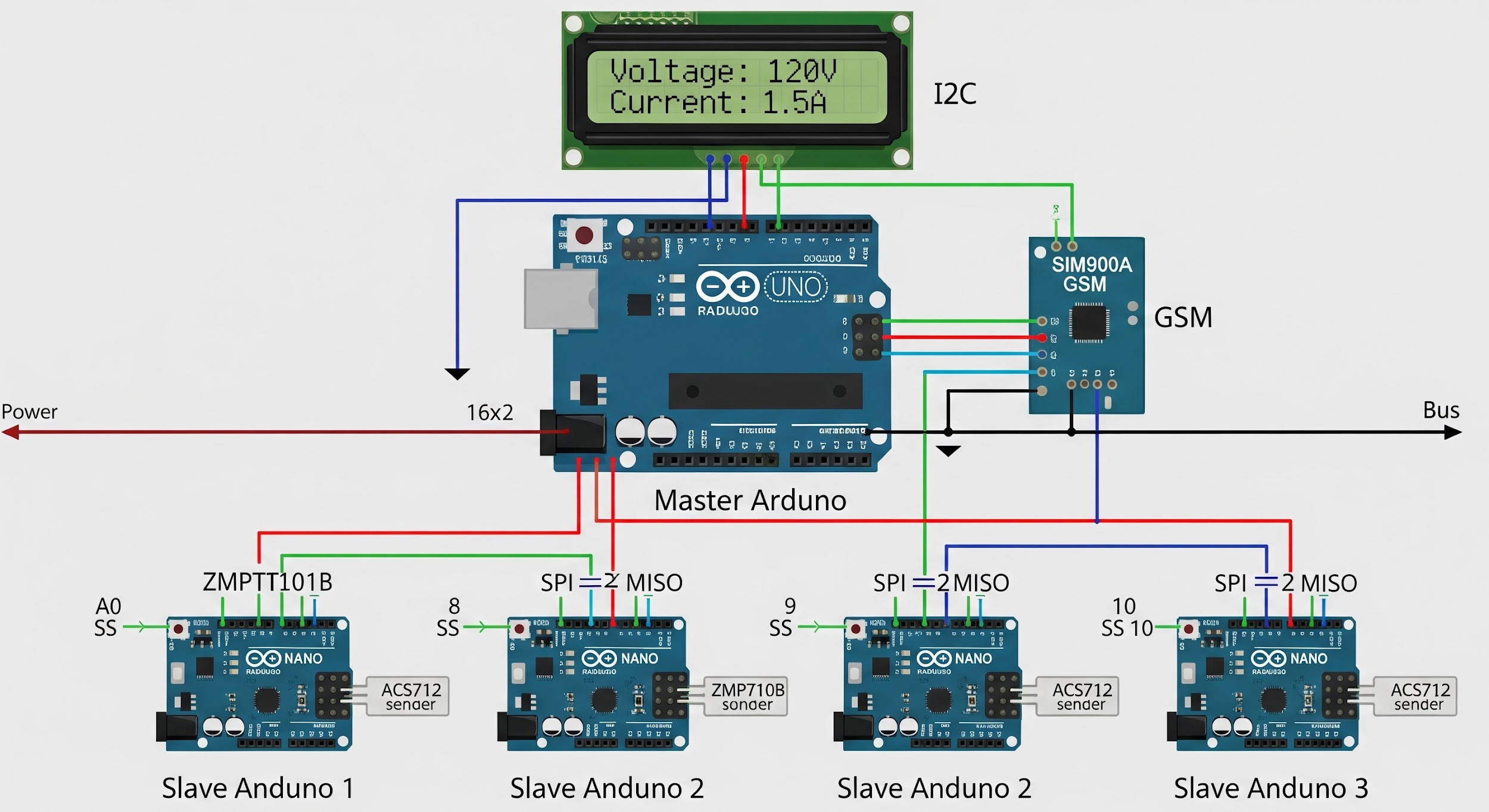


Figure 3.3: System Schematic Diagram

### 3.4 How the 4 Arduinos Work Together

The four Arduinos function as the distributed intelligence of the system. One serves as the master controller, while the remaining three act as slave nodes connected to the sensors.

1. **First, Second, and Third Arduinos (Slaves):** Each of these Arduinos is connected to a dedicated sensor node (comprising a ZMPT101B voltage sensor and an ACS712 current sensor). Their sole responsibility is to continuously read the analog values from these sensors, convert them to digital format, and prepare them for transmission.
2. **Fourth Arduino (Master):** This Arduino is connected to the three slave Arduinos via the SPI communication protocol. It polls each slave sequentially to collect the latest voltage and current readings.
3. The master Arduino then analyzes this collected data using the fault detection algorithm.
4. Finally, the master Arduino transmits the system status and any fault alerts to the GSM module for remote notification.

This distributed approach allows the master Arduino to focus on analysis and communication while the slave Arduinos handle the continuous, high-frequency task of data acquisition.

### 3.5 How the Sensor Nodes Detect Power Faults

The sensor nodes in the mesh network detect power faults by directly measuring the core electrical parameters of the power line: voltage and current.

* **Voltage Sensor (ZMPT101B):** This sensor measures the AC voltage. The Arduino can detect:
  + **Overvoltage:** A reading significantly above the standard utility voltage (e.g., > 240V).
  + **Undervoltage (Brownout):** A reading significantly below the standard voltage (e.g., < 200V).
  + **Power Outage (Blackout):** A voltage reading of or near zero.
* **Current Sensor (ACS712):** This sensor measures the current flowing through the line. The Arduino can detect:
  + **Overcurrent:** A reading that exceeds the safe limit for the circuit, often indicating a short circuit or severe overload.

The collected data from these sensors is transmitted via SPI to the master Arduino. A fault detection algorithm then compares these real-time values against predefined safe operating thresholds. If any reading falls outside these thresholds, a fault is registered, and an alert is immediately generated. This method provides direct, unambiguous detection of electrical faults, unlike indirect environmental measurements.



Figure 3.4: ZMPT101B Voltage Sensor



Figure 3.5: ACS712 Current Sensor

### 3.6 The SIM900A GSM Module

The GSM module in the power fault detection system plays a crucial role in enabling wireless communication over long distances. It acts as the bridge between the local IoT network and remote stakeholders.

1. **Communication Interface:** The GSM module uses a serial interface to communicate with the master Arduino.
2. **SIM Card:** The module requires a standard SIM card to connect to the cellular network.
3. **Data Transmission:** When the master Arduino detects a fault, it formats an alert message (e.g., "FAULT: Overvoltage at Node 2") and sends it to the GSM module.
4. **SMS Transmission:** The GSM module transmits this message as an SMS to pre-programmed phone numbers belonging to maintenance teams or system administrators.

This ensures reliable and timely fault notification, regardless of the location of the monitoring site, as long as cellular service is available.

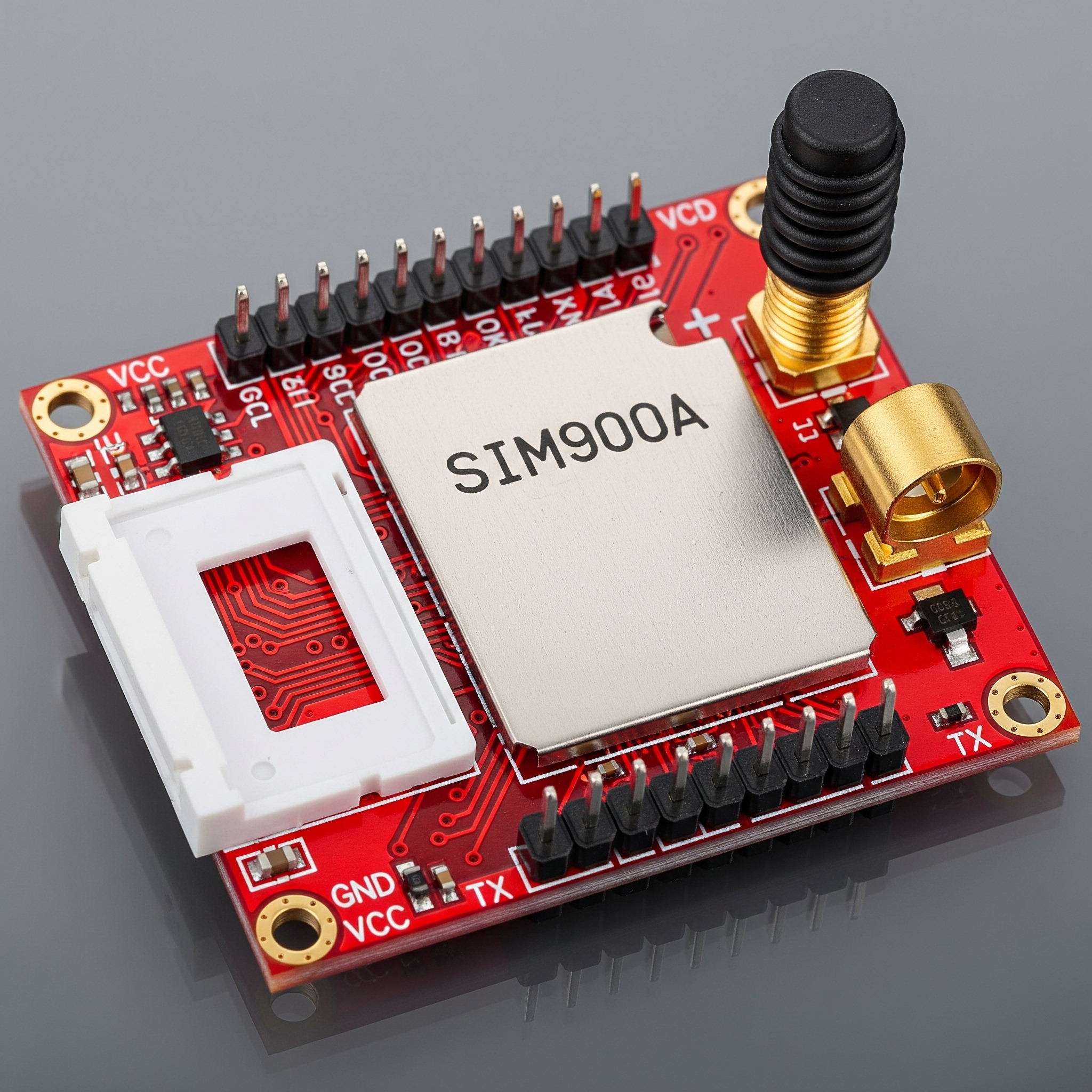


Figure 3.6: SIM900A GSM Module

Description: This is a photo of the SIM900A module, a circuit board (often red or blue) with a SIM card holder on one side. It features the SIM900A chip under a metal shield, a small helical antenna or a brass connector for an external antenna, and header pins for power and serial communication.

### 3.7 The Fault Detection Algorithm

The fault detection algorithm is implemented on the master Arduino. It continuously analyzes the data received from the slave nodes to identify electrical anomalies.

**Algorithm Steps:**

1. **Initialize Variables & Thresholds:**
   * V\_max = 245.0 (Maximum safe voltage)
   * V\_min = 195.0 (Minimum safe voltage)
   * I\_max = 15.0 (Maximum safe current, example value)
   * faultDetected = false
2. **Request and Read Data:**
   * Sequentially request and receive currentVoltage and currentCurrent readings from each slave node via SPI.
3. **Check for Faults:**
   * For each node's data, perform the following checks:
     + if (currentVoltage > V\_max): Set fault type to "Overvoltage".
     + else if (currentVoltage < V\_min): Set fault type to "Undervoltage".
     + else if (currentCurrent > I\_max): Set fault type to "Overcurrent".
4. **Alert Generation:**
   * If a fault is detected:
     + Construct an alert message (e.g., "FAULT: Overcurrent at Node 1").
     + Send the message to the GSM module for SMS transmission.
     + Display the alert on the local LCD.
5. **Repeat:**
   * Loop back to Step 2 to continuously monitor the power line.

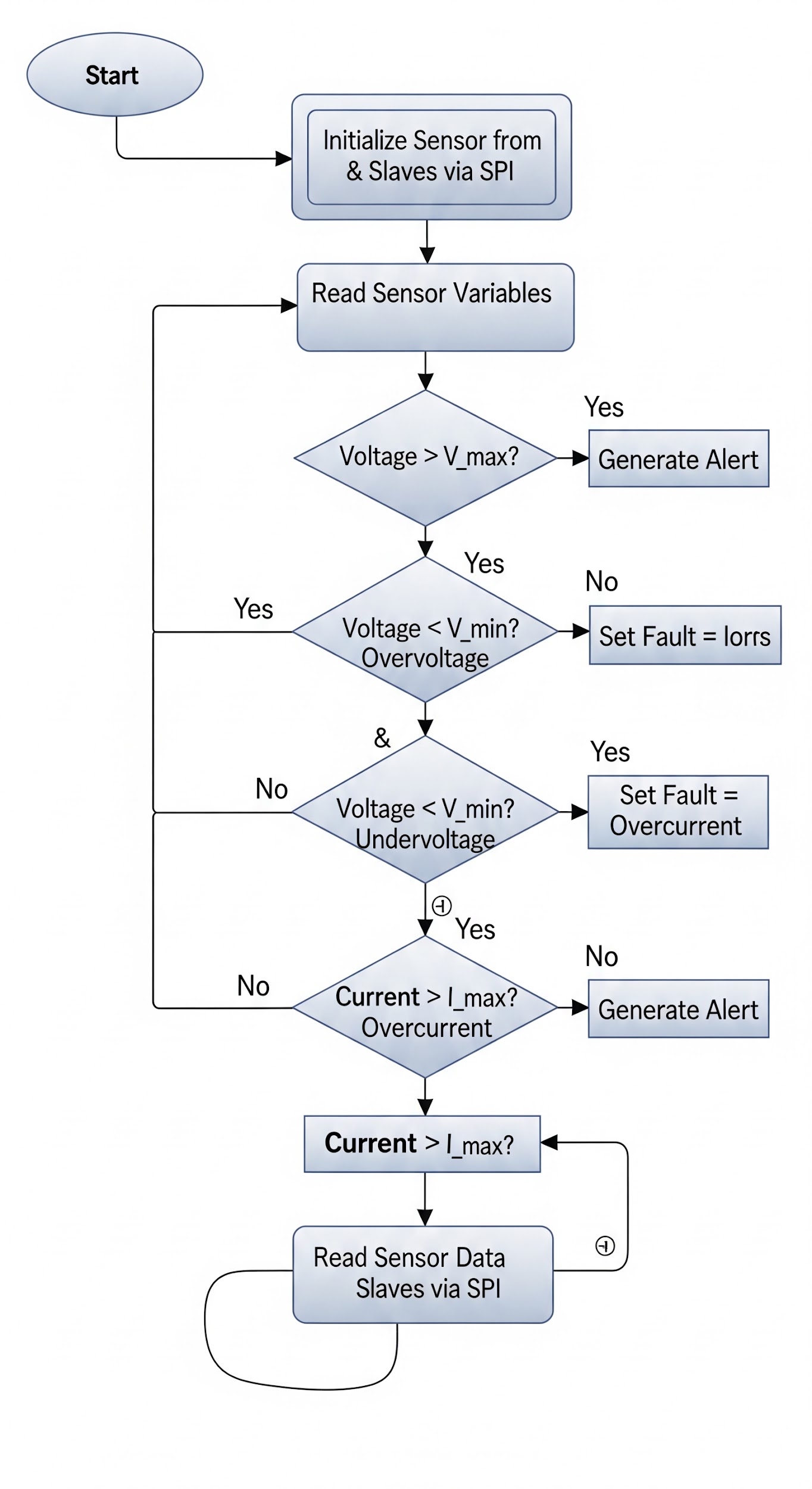


Figure 3.7: The Fault Detection Flowchart

### 3.8 System Requirements

To run efficiently and produce the required results, the IoT-based power fault detection system requires the following:

#### 3.8.1 Hardware Requirements

* Arduino Microcontrollers (4 units)
* Wireless Communication Module (SIM900A GSM module and SIM card)
* Voltage Sensors (e.g., ZMPT101B, 3 units)
* Current Sensors (e.g., ACS712, 3 units)
* LCD Display
* Power supplies for all components.

#### 3.8.2 Software Requirements

* Arduino IDE for programming the microcontrollers.
* Proteus simulation software (for design and validation).
* A suitable Operating System (OS) for the development environment.

### 3.9 Advantages of the Proposed System

The proposed IoT-based power fault detection system offers significant improvements in fault responsiveness and situational awareness by directly monitoring critical electrical parameters.

* **High Accuracy and Relevance:** By using voltage and current sensors, the system detects actual electrical faults, not indirect environmental indicators. This provides accurate, actionable information.
* **Rapid Fault Detection:** The system provides continuous, real-time monitoring, enabling immediate detection and notification, which significantly reduces the time between fault occurrence and response.
* **Remote Monitoring and Resilience:** The integration of a GSM module ensures alerts are delivered across wide geographical areas. The underlying mesh network topology provides resilience against single-node failures, ensuring the monitoring fabric remains operational.
* **Cost-Effectiveness:** The use of standard, off-the-shelf hardware components (Arduino, ZMPT101B, ACS712, SIM900A) reduces capital and maintenance expenditures, making the solution accessible for a wide range of applications.
* **Scalability:** The architecture is inherently scalable. Additional sensing nodes can be introduced into the mesh network without significant redesign to cover larger areas.
* **Improved Safety and Reduced Downtime:** Early detection and precise notification reduce Mean Time To Repair (MTTR). Maintenance crews can be dispatched with prior knowledge of the fault type, shortening restoration cycles, minimizing economic impact, and improving safety.