



IE2090
Professional Engineering Practice and
Industrial Management

2nd Year, 2nd Semester

Final Report

Home Security Robot

Submitted to

Sri Lanka Institute of Information Technology

In partial fulfillment of the requirements for the
Bachelor of Science Special Honors Degree in Information Technology

18/06/2024


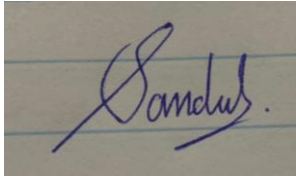
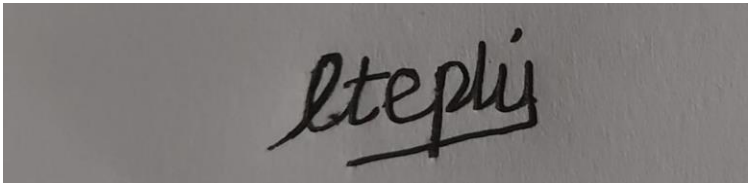
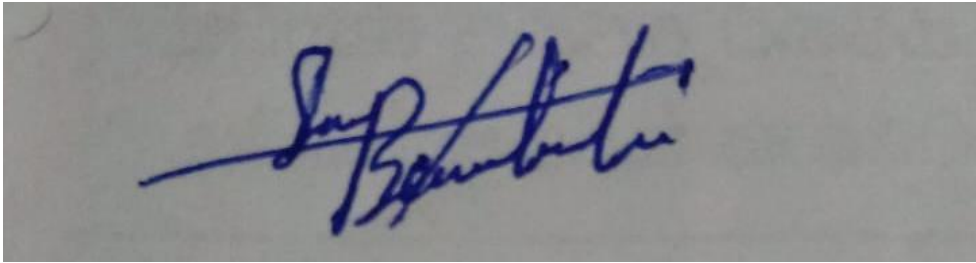
Declaration

We declare that this project report or part of it was not a copy of a document done by any organization, university any other institute or a previous student project group at SLIIT and was not copied from the Internet or other sources.

Project Details

Project Title	Home Security Robot
Project ID	PEP_32

Group Members

Reg. No	Name	Signature
IT22360250	Sahan W A G	
IT22069368	Karunarathna N W A R S	
IT22912770	Fonseka N S W	
IT22132000	Bamunuarachchi S S	

Abstract

The Home Security Robot project aims to develop an autonomous surveillance system designed to enhance security in residential, commercial and industrial settings. This project integrates advanced sensor technology and real-time monitoring capabilities to provide a comprehensive security solution. The robot is equipped with motion detection, fire and gas detection, and a high-resolution camera for live surveillance. Controlled by an Arduino microcontroller, the system offers remote access and instant alerts, enabling swift responses to potential threats. Over a development period of three months and a budget of Rs.20,000, the project focused on designing, building, and testing a robust, cost-effective security robot. The resulting prototype demonstrates the feasibility and effectiveness of using robotic technology to improve safety and surveillance, addressing the growing need for innovative security solutions in various environments.

Acknowledgement

We would like to express our sincere gratitude to our supervisor, Ms. Narmada Gamage, for her guidance, encouragement, and insightful feedback throughout this project. Her expertise and support were instrumental in the successful completion of our Home Security Robot project. Additionally, we thank our families and friends for their unwavering support and understanding during this journey. We also extend our appreciation to the faculty and staff of the Sri Lanka Institute of Information Technology for providing us with the necessary resources and a conducive learning environment.

Table of Contents

1. Introduction.....	1
1.1 Problem Statement.....	1
1.2 Product Scope	1
1.3 Project Report Structure.....	2
2. Methodology	3
2.1 Requirements and Analysis.....	3
2.1.1 Functional Requirements	3
2.1.2 Non-Functional Requirements	6
2.2 Design	9
2.3 Implementation	10
2.4 Testing.....	16
3. Evaluation.....	21
3.1 Assessment of the Project Results	21
3.2 Lessons Learned.....	23
3.3 Future Work.....	25
4. Conclusion	27
5. References.....	28

List of Figures

Figure 1: High Level Circuit Diagram.....	9
Figure 2: Implemented Circuit Diagram.....	10
Figure 3: Implementation of code.....	12
Figure 4: Implementation of code (2)	13
Figure 5: Implementation of code (3)	13
Figure 6: Implementation of code (4)	14
Figure 7: Implementation of code (5)	14
Figure 8: Implementation of code (6)	15
Figure 9: Implementation of code (7)	15

List of Tables

Table 1: Functional Requirement 1.....	3
Table 2: Functional Requirement 2.....	4
Table 3: Functional Requirement 3.....	4
Table 4: Functional Requirement 4.....	5
Table 5: Functional Requirement 5.....	5
Table 6: Motion Detection Implementation.....	16
Table 7: Fire Detection Implementation.....	17
Table 8: Gas Detection Implementation.....	18
Table 9: Image Capture Implementation.....	19
Table 10: Wi-Fi and Communication.....	20

List of Acronyms and Abbreviations

- PIR – Passive Infrared
- GSM – Global System for Mobile
- Wi-Fi – Wireless Fidelity
- SD – Secure Digital
- IDE – Integrated Development Environment
- RISC – Reduced Instruction Set Computer
- UL – Underwrites Laboratories
- CE – Conformance Européenne

1. Introduction

1.1 Problem Statement

In an era where security threats are becoming increasingly sophisticated, traditional security measures often fall short in providing comprehensive protection. Homeowners, businesses, and industrial facilities face numerous security challenges, including unauthorized intrusions, fire hazards, and gas leaks. The limitations of human security personnel, who are prone to fatigue and errors, highlight the need for an innovative solution that can ensure continuous, reliable monitoring and rapid threat detection. The Home Security Robot addresses these challenges by offering an autonomous, technology-driven solution that enhances security and safety across various environments.

1.2 Product Scope

The Home Security Robot project aims to develop a versatile and cost-effective autonomous robot designed to patrol designated areas, detect potential threats, and provide live monitoring to users. Equipped with advanced sensors, including infrared, flame, and gas detectors, the robot will identify intrusions, fire, and gas leaks. It will capture real-time images and videos, allowing users to monitor their surroundings remotely. The project includes the design, development, and integration of hardware and software components, ensuring the robot performs effectively in diverse environments. By leveraging cutting-edge technology, this project seeks to deliver a robust security solution that enhances safety and peace of mind for users.

1.3 Project Report Structure

This report is designed to provide a comprehensive overview of the Home Security Robot project, detailing each phase of development from inception to completion. The report begins with an introduction, outlining the problem statement, product scope, and report structure. Following this, the **Requirements and Analysis** section delves into the specific needs and functionalities identified for the robot, ensuring it meets the security challenges effectively. The **Design** section presents the architectural blueprint of the robot, including hardware and software design considerations. The **Implementation** section documents the process of building and programming the robot, highlighting key technical steps and challenges encountered. The **Testing** section evaluates the robot's performance, ensuring all functionalities operate as intended and meet the predefined requirements. Finally, the **Conclusion** section summarizes the project's outcomes, lessons learned, and potential future work, providing a reflective analysis of the project's impact and areas for further development.

2. Methodology

2.1 Requirements and Analysis

The development of the Home Security Robot involved a meticulous process of requirements gathering and analysis to ensure the system meet its intended functionalities. Documentation and technical feasibility studies were employed to collect and analyze requirements, ensuring a robust and effective design.

2.1.1 Functional Requirements

1. Motion Detection Implementation

- Components: PIR sensor, ATmega328P Arduino board
- Functional Requirement: Detecting unauthorized movement within the designated area and sending a signal to the Wi-Fi module.

Table 1: Functional Requirement 1

Input	Process	Output
Motion	Detecting movement using PIR sensor	Signal to Wi-Fi module

2. Fire Detection Implementation

- Components: Ky-026 flame sensor, ATmega2560 Arduino board
- Functional Requirement: Detecting fire or high-temperature areas and sending a signal to the Wi-Fi module.

Table 2: Functional Requirement 2

Input	Process	Output
Flame/Heat	Detecting flame/high temperature	Signal to Wi-Fi module

3. Gas Detection Implementation

- Components: MQ2 Gas Sensor, ATmega2560 Arduino board
- Functional Requirement: Detecting gas leaks and sending a signal to the Wi-Fi module.

Table 3: Functional Requirement 3

Input	Process	Output
Gas Leak	Detecting gas presence using MQ2 sensor	Signal to Wi-Fi module

4. Image Capture Implementation

- Components: ESP32 Camera Module
- Functional Requirement: Capturing real-time images and storing them on the SD card, as well as streaming video.

Table 4: Functional Requirement 4

Input	Process	Output
Environment	Capture images and video using camera module	Images stored on SD card and video streamed via Wi-Fi module

5. Wi-Fi and Communication

- Components: ESP32 Camera Module
- Functional Requirement: Sending alerts and images to the user via a web interface accessible through an IP address.

Table 5: Functional Requirement 5

Input	Process	Output
Signals from sensors	Establishing communication through a web interface via an IP address, sending alerts	Alerts

2.1.2 Non-Functional Requirements

1. Performance Requirements

- The surveillance robot must be able to process and analyze video feeds in real time with a latency of less than 100 milliseconds.
- The surveillance robot must be able to maintain a stable connection with the remote monitoring station with a minimum bandwidth of 1Mbps.
- The surveillance robot must be able to operate for at least 8 hours on a single charge.
- The surveillance robot must be able to navigate to specified location within a certain time frame, for example, within 5 minutes for 100 meters.
- The surveillance robot must be able to detect and track objects with a minimum accuracy of 90%.

2. Safety Requirements

- The surveillance robot must be equipped with safety features such as emergency stop buttons and collision detection sensors.
- The surveillance robot must comply with all relevant safety regulations and standards, such as UL and CE standards.
- The surveillance robot must be designed to minimize the risk of injury or damage to people, animals, or property.
- The surveillance robot must be able to operate in various environments and weather conditions, while ensuring the safety of the device and its surroundings.
- The surveillance robot must be able to detect and alert operators of any potential safety hazards, such as low battery levels or malfunctioning sensors.

3. Software Quality Attributes

- The surveillance robot must be adaptable to various environments and scenarios, with the ability to add or modify features as needed.
- The surveillance robot must be available in the required time frame.
- The surveillance robot must be correct in its operation, with no errors or bugs that could affect its functionality.
- The surveillance robot must be flexible to accommodate changes in requirements or user needs.
- The surveillance robot must be interoperable with other systems and devices, with the ability to share and communicate with other devices.
- The surveillance robot must be portable, with the ability to run on different platforms and operating systems.
- The surveillance robot must be maintainable, with the ability to update or repair the software as needed.
- The surveillance robot must be reusable, with the ability to use components or features in other applications or systems.
- The surveillance robot must be robust, with the ability to test individual components and features to ensure they are functioning correctly.
- The surveillance robot must be user-friendly, with an intuitive and easy-to-use interface.

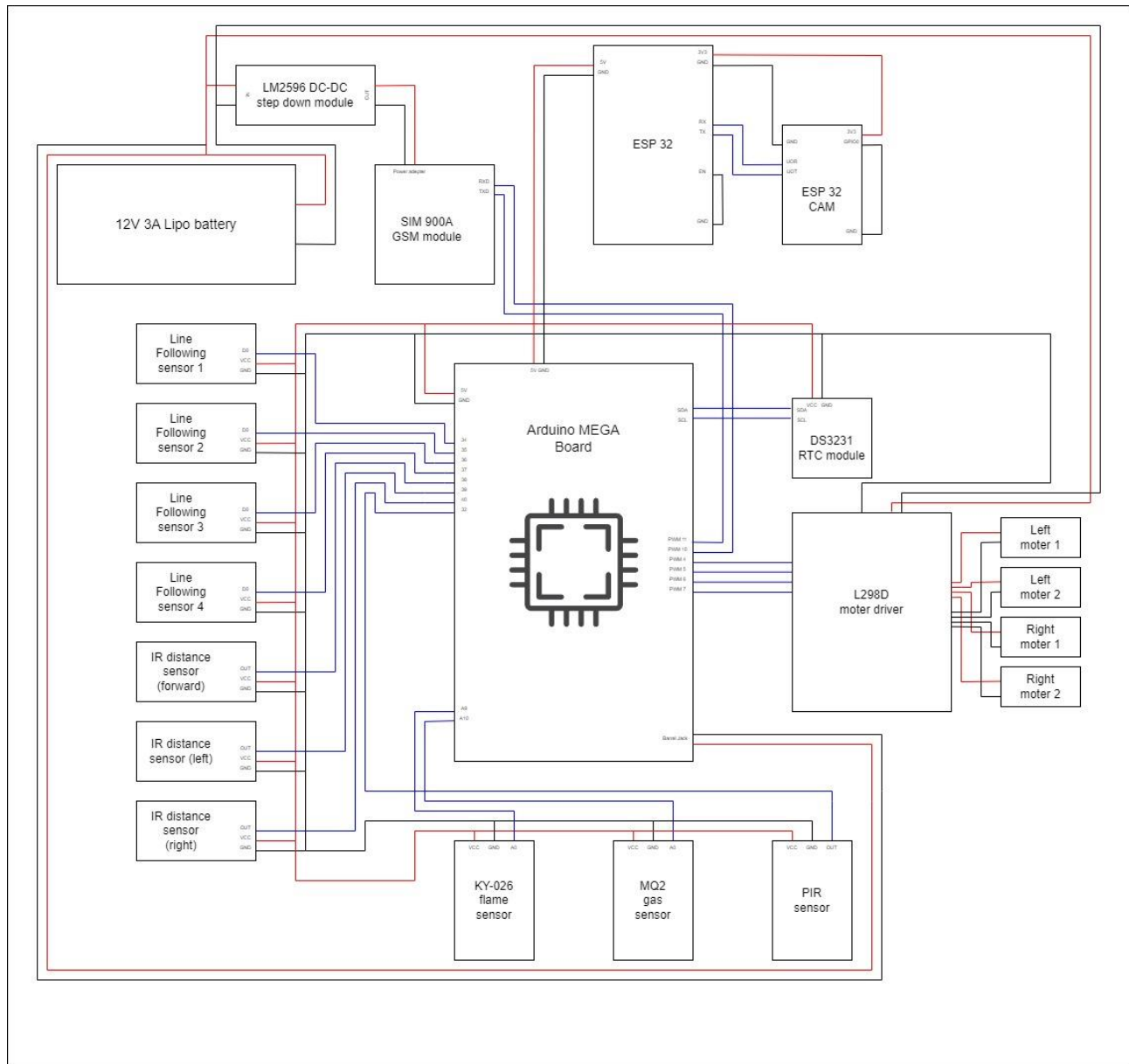
4. Business Rules

- The surveillance robot must only be accessible to authorized users and administrators.
- The surveillance robot must only be used for authorized purposes, such as surveillance and security.
- The surveillance robot must only store and transmit data that is necessary for its operation and its functionality.
- The surveillance robot must only be used in compliance with relevant laws and regulations, such as privacy laws and surveillance regulations.
- The surveillance robot must only be used in accordance with the terms of service and end user license agreement.

2.2 Design

During the design phase, the project team developed a comprehensive blueprint for the Home Security Robot, encompassing hardware components. The hardware design focused on selecting and integrating appropriate sensors, a camera module, and a robust Arduino platform. The robot's structure was designed to ensure stability and mobility, with considerations for power efficiency and durability. The software design involved creating algorithms for sensor data processing, image capture, and remote communication. Detailed schematic diagrams and flowcharts were created to guide the development process, ensuring that each component would function harmoniously within the overall system.

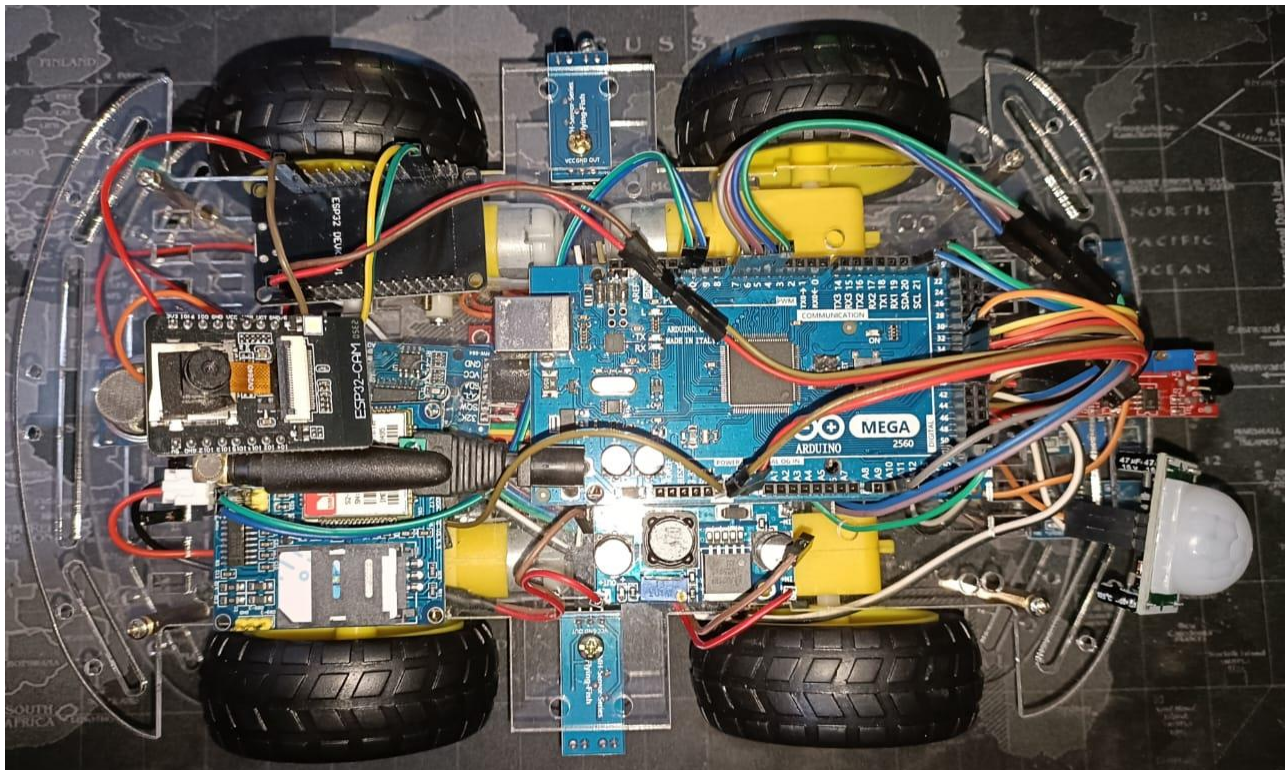
Figure 1: High Level Circuit Diagram



2.3 Implementation

The implementation phase was dedicated to building and programming the Home Security Robot according to the design specifications. This involved assembling the physical components, including PIR sensors for motion detection, a KY-026 flame sensor for fire detection, MQ2 sensor for gas detection, and an OV2840 camera module for image capture. The Arduino microcontroller was programmed to process sensor data, control the robot's movements, and handle communication with the user. The team wrote and tested code to ensure seamless integration of all components, implementing features such as path following, threat detection, and alert notifications via the GSM Module. Regular progress reviews and debugging sessions were conducted to address any technical issues promptly.

Figure 2: Implemented Circuit Diagram



Development Tool: Arduino IDE

The Arduino Integrated Development Environment (IDE) was the primary tool used for programming the microcontrollers in the Home Security Robot project. The Arduino IDE provided a user-friendly interface that allowed developers to write, compile, and upload code to Arduino boards.

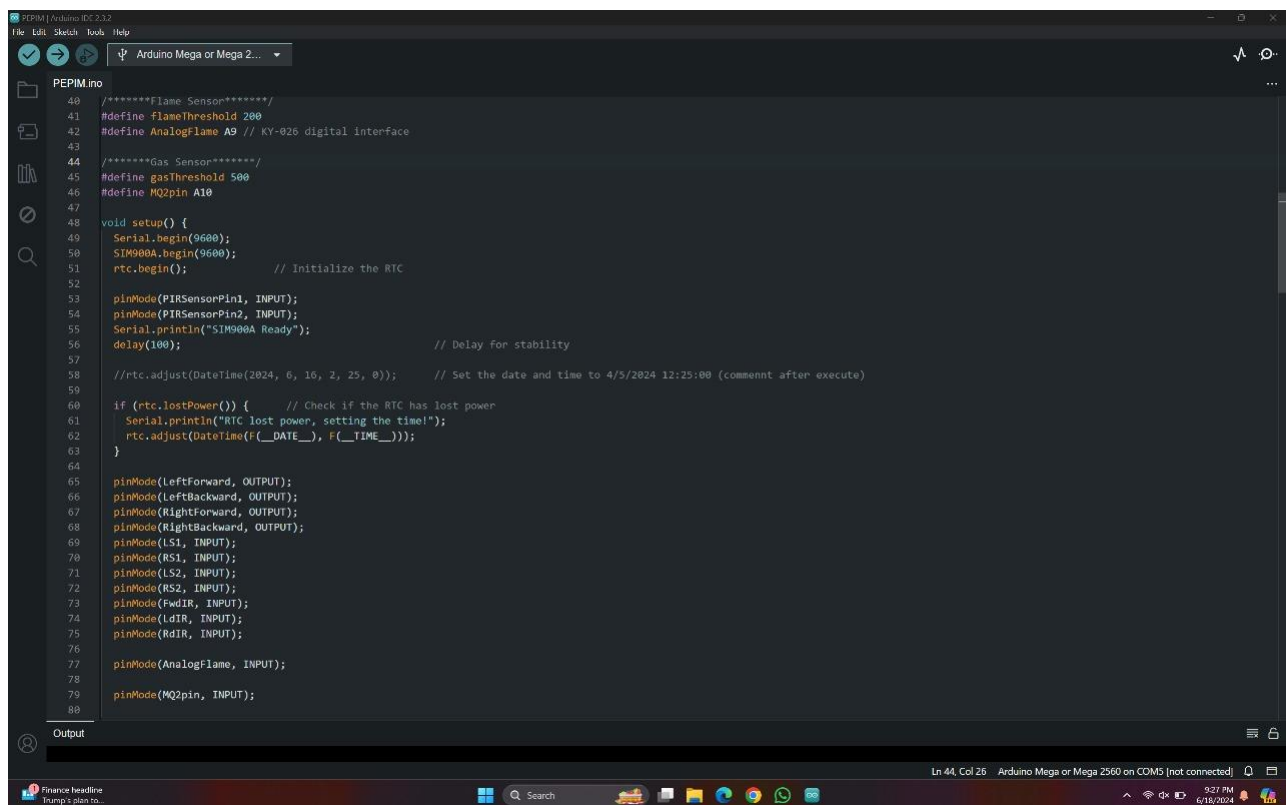
Key Features of Arduino IDE:

1. **User-Friendly Interface:** The Arduino IDE features an intuitive and straightforward interface, making it accessible for both beginners and experienced developers. Its simplicity helps streamline the coding process, allowing for rapid development and iteration.
2. **Extensive Library Support:** The IDE includes a vast library of pre-written code that supports a wide range of sensors and modules. This library support facilitated the integration of various components, such as PIR sensors for motion detection, KY-026 flame sensors for fire detection, MQ2 sensors for gas detection, and the OV2840 camera module.
3. **Cross-Platform Compatibility:** The Arduino IDE is compatible with Windows, macOS, and Linux operating systems, ensuring that the development environment is accessible regardless of the developer's preferred platform.
4. **Serial Monitor:** The built-in Serial Monitor allows developers to debug their programs by providing real-time feedback from the microcontroller. This feature was crucial for testing and verifying sensor readings and other outputs during the development process.
5. **Integrated Libraries and Board Management:** The Arduino IDE's library manager allows for easy installation and updating of libraries, while the board manager simplifies adding support for various Arduino-compatible boards, such as the ESP32.

Using the Arduino IDE, we were able to efficiently program and test the microcontrollers, ensuring seamless integration of the robot's hardware components and enabling the desired functionalities.

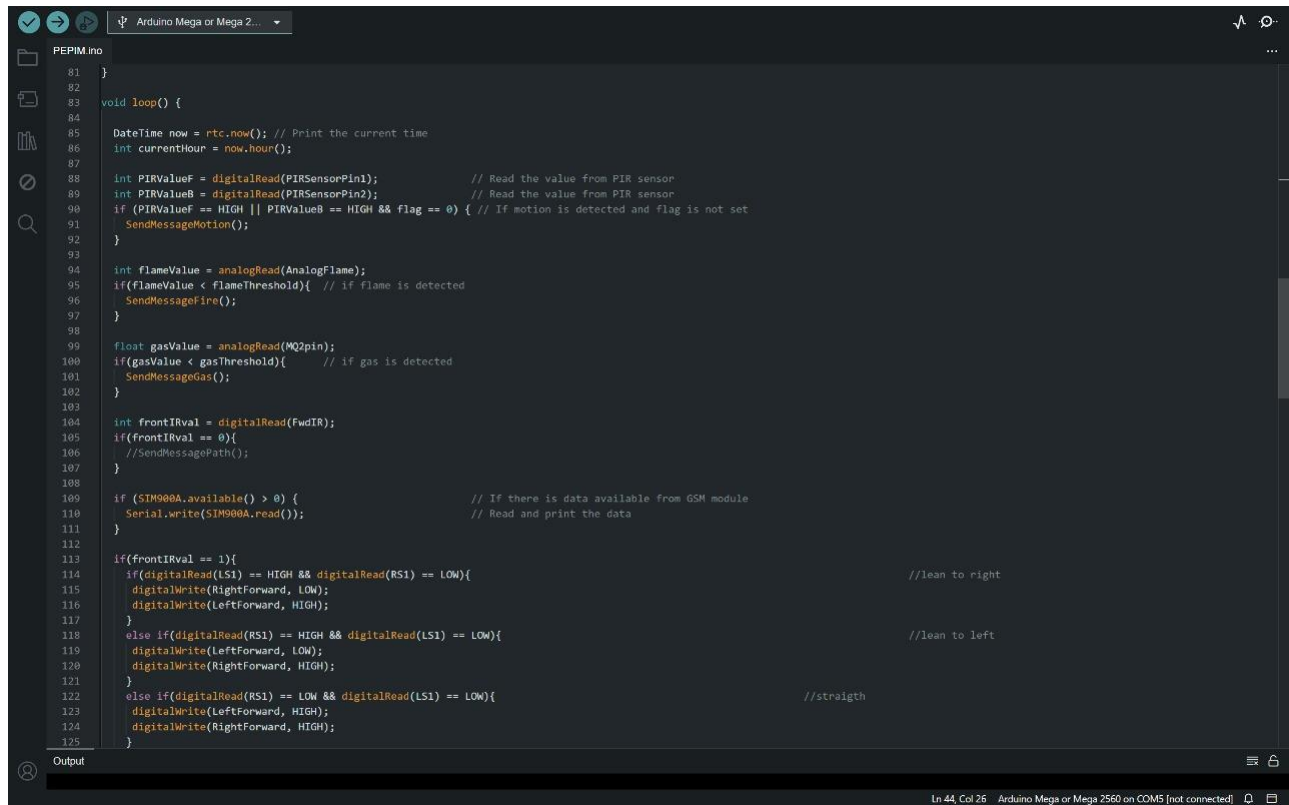
We used various libraries to integrate sensors such as the PIR sensor for motion detection, KY-026 for fire detection, MQ2 for gas detection, and the ESP32 camera module for image capture and storage. The code was structured to handle the sensor inputs, process the data, and send alerts to the user via the GSM module. The IDE's Serial Monitor was invaluable for debugging and ensuring the system's reliability.

Figure 3: Implementation of code



```
PEPIM.ino
40 /*****Flame Sensor*****/
41 #define flameThreshold 200
42 #define AnalogFlame A9 // KY-026 digital interface
43
44 /*****Gas Sensor*****/
45 #define gasThreshold 500
46 #define MQ2pin A10
47
48 void setup() {
49   Serial.begin(9600);
50   SIM900A.begin(9600);
51   rtc.begin(); // Initialize the RTC
52
53   pinMode(PIRSensorPin1, INPUT);
54   pinMode(PIRSensorPin2, INPUT);
55   Serial.println("SIM900A Ready");
56   delay(100); // Delay For stability
57
58   //rtc.adjust(DateTime(2024, 6, 16, 2, 25, 0)); // Set the date and time to 4/5/2024 12:25:00 (comment after execute)
59
60   if (rtc.lostPower()) { // Check if the RTC has lost power
61     Serial.println("RTC lost power, setting the time!");
62     rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
63   }
64
65   pinMode(LeftForward, OUTPUT);
66   pinMode(LeftBackward, OUTPUT);
67   pinMode(RightForward, OUTPUT);
68   pinMode(RightBackward, OUTPUT);
69   pinMode(LS1, INPUT);
70   pinMode(RS1, INPUT);
71   pinMode(LS2, INPUT);
72   pinMode(RS2, INPUT);
73   pinMode(FwdIR, INPUT);
74   pinMode(LdIR, INPUT);
75   pinMode(RdIR, INPUT);
76
77   pinMode(AnalogFlame, INPUT);
78
79   pinMode(MQ2pin, INPUT);
80}
```

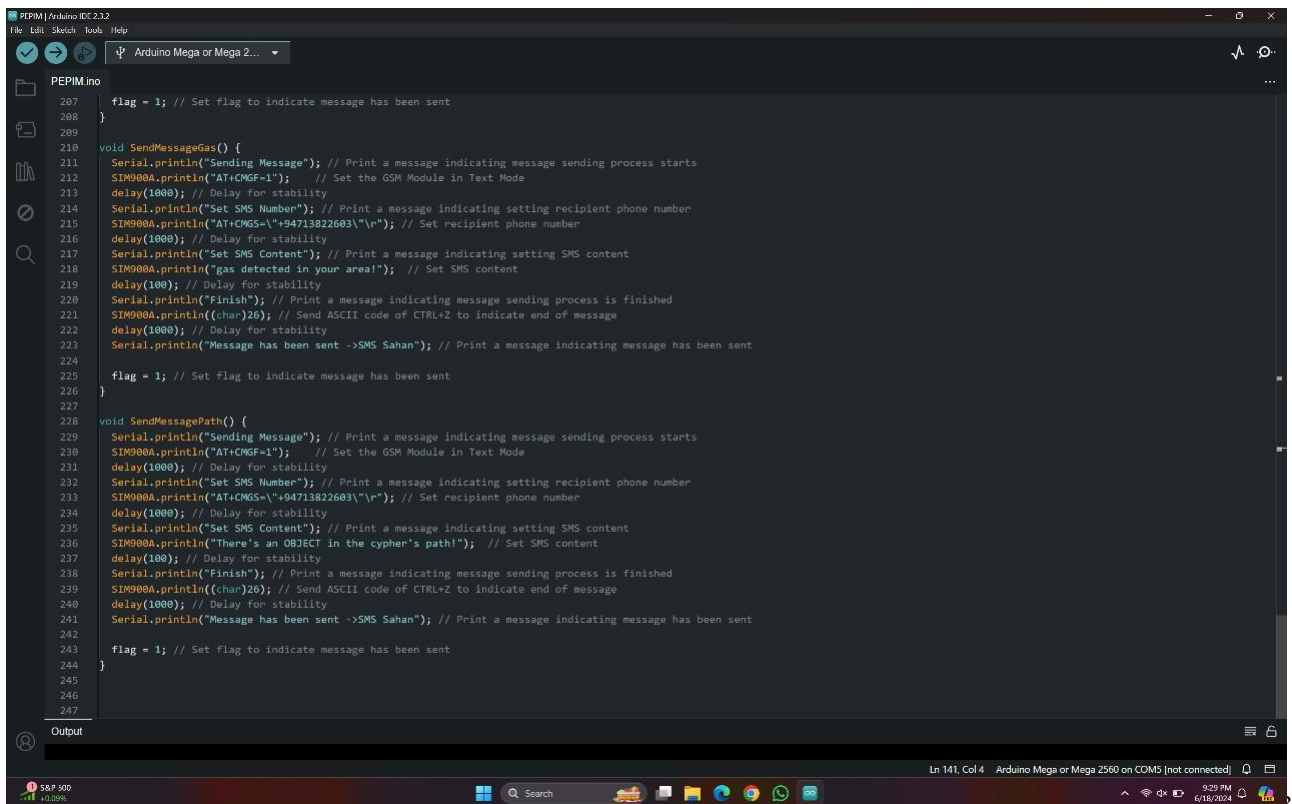
Figure 4: Implementation of code (2)



```
PEPIM.ino
81 }
82
83 void loop() {
84
85   DateTime now = rtc.now(); // Print the current time
86   int currentHour = now.hour();
87
88   int PIRValueF = digitalRead(PIRSensorPin1); // Read the value from PIR sensor
89   int PIRValueB = digitalRead(PIRSensorPin2); // Read the value from PIR sensor
90   if (PIRValueF == HIGH || PIRValueB == HIGH && flag == 0) { // If motion is detected and flag is not set
91     SendMessageMotion();
92   }
93
94   int flameValue = analogRead(AnalogFlame);
95   if (flameValue < flameThreshold) { // if flame is detected
96     SendMessageFire();
97   }
98
99   float gasValue = analogRead(MQ2pin);
100   if (gasValue < gasThreshold) { // if gas is detected
101     SendMessageGas();
102   }
103
104   int frontIRval = digitalRead(FwdIR);
105   if (frontIRval == 0) {
106     //SendMessagePath();
107   }
108
109   if (SIM900A.available() > 0) { // If there is data available from GSM module
110     Serial.write(SIM900A.read()); // Read and print the data
111   }
112
113   if (frontIRval == 1) {
114     if (digitalRead(LS1) == HIGH && digitalRead(RS1) == LOW) { //lean to right
115       digitalWrite(RightForward, LOW);
116       digitalWrite(LeftForward, HIGH);
117     }
118     else if (digitalRead(RS1) == HIGH && digitalRead(LS1) == LOW) { //lean to left
119       digitalWrite(LeftForward, LOW);
120       digitalWrite(RightForward, HIGH);
121     }
122     else if (digitalRead(RS1) == LOW && digitalRead(LS1) == LOW) { //straighth
123       digitalWrite(LeftForward, HIGH);
124       digitalWrite(RightForward, HIGH);
125     }
126   }
127 }
128
129 Output
```

Ln 44, Col 26 Arduino Mega or Mega 2560 on COM5 [not connected]

Figure 5: Implementation of code (3)



```
PEPIM.ino
207 flag = 1; // Set flag to indicate message has been sent
208 }
209
210 void SendMessageGas() {
211   Serial.println("Sending Message"); // Print a message indicating message sending process starts
212   SIM900A.println("AT+CMGF=1"); // Set the GSM Module in Text Mode
213   delay(1000); // Delay for stability
214   Serial.println("Set SMS Number"); // Print a message indicating setting recipient phone number
215   SIM900A.println("AT+CMGS=\"+94713822603\""); // Set recipient phone number
216   delay(1000); // Delay for stability
217   Serial.println("Set SMS Content"); // Print a message indicating setting SMS content
218   SIM900A.println("gas detected in your area!"); // Set SMS content
219   delay(100); // Delay for stability
220   Serial.println("Finish"); // Print a message indicating message sending process is finished
221   SIM900A.println((char)26); // Send ASCII code of CTRL+Z to indicate end of message
222   delay(1000); // Delay for stability
223   Serial.println("Message has been sent ->SMS Sahan"); // Print a message indicating message has been sent
224
225   flag = 1; // Set flag to indicate message has been sent
226 }
227
228 void SendMessagePath() {
229   Serial.println("Sending Message"); // Print a message indicating message sending process starts
230   SIM900A.println("AT+CMGF=1"); // Set the GSM Module in Text Mode
231   delay(1000); // Delay for stability
232   Serial.println("Set SMS Number"); // Print a message indicating setting recipient phone number
233   SIM900A.println("AT+CMGS=\"+94713822603\""); // Set recipient phone number
234   delay(1000); // Delay for stability
235   Serial.println("Set SMS Content"); // Print a message indicating setting SMS content
236   SIM900A.println("There's an OBJECT in the cypher's path!"); // Set SMS content
237   delay(100); // Delay for stability
238   Serial.println("Finish"); // Print a message indicating message sending process is finished
239   SIM900A.println((char)26); // Send ASCII code of CTRL+Z to indicate end of message
240   delay(1000); // Delay for stability
241   Serial.println("Message has been sent ->SMS Sahan"); // Print a message indicating message has been sent
242
243   flag = 1; // Set flag to indicate message has been sent
244 }
245
246
247
248
249 Output
```

Ln 141, Col 4 Arduino Mega or Mega 2560 on COM5 [not connected]

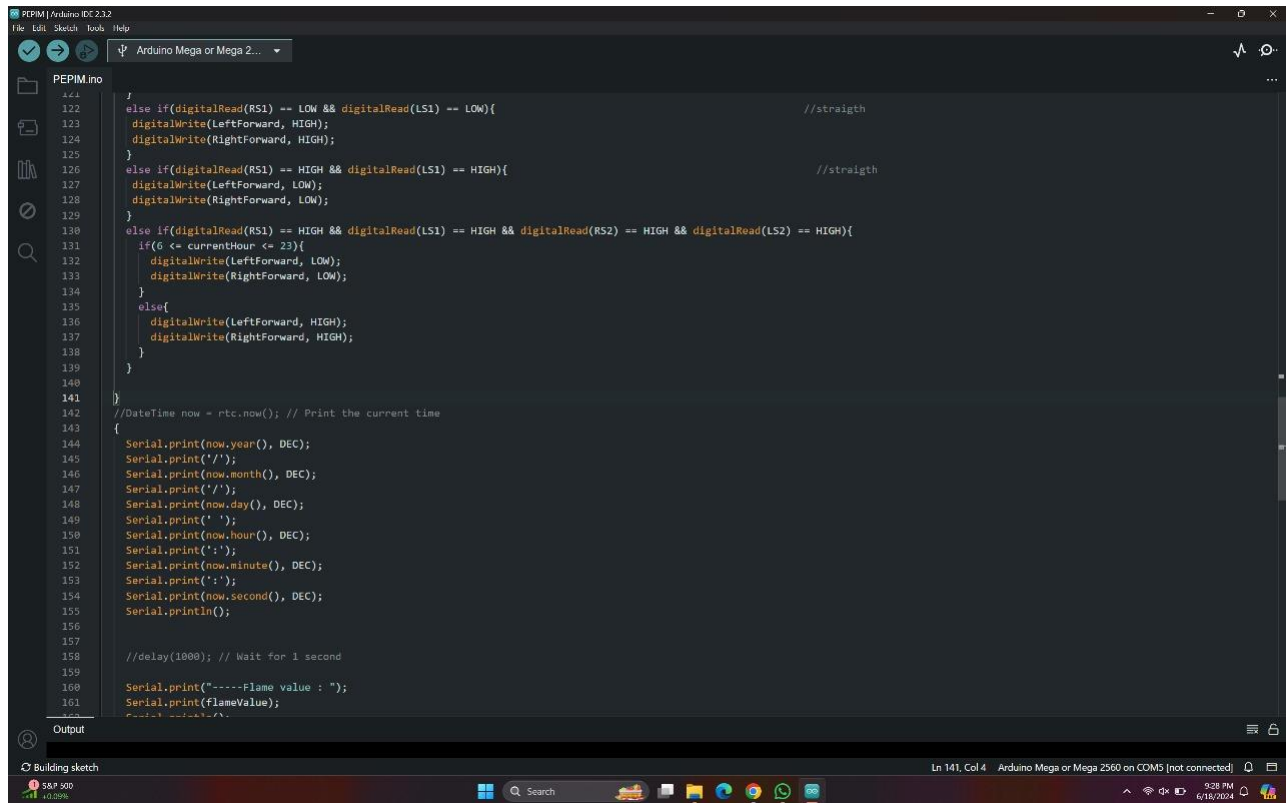
Figure 6: Implementation of code (4)

```
PEPIM.ino
1 #include <RTClib.h>
2 #include <Wire.h>
3 #include <SoftwareSerial.h>
4 #include <Adafruit_BusIO_Register.h>
5 #include <Adafruit_I2CDevice.h>
6 #include <Adafruit_I2CRegister.h>
7 #include <Adafruit_SPIDevice.h>
8
9 /*****GSM module*****/
10 SoftwareSerial SIM900A(10, 11);
11
12 /*****RTC module*****/
13 RTC_DS3231 rtc;
14
15 /*****PIR sensor*****/
16 #define PIRSensorPin1 32
17 #define PIRSensorPin2 4
18 int flag = 0;
19
20 /*****set speed*****/
21 #define speed 100 //set speed
22
23 /*****IR distance sensor*****/
24 #define FwdIR 38 //front distance IR
25 #define LdIR 39 //left distance IR
26 #define RdIR 40 //right distance IR
27
28 /*****line following IR sensors*****/
29 #define LS1 35 //left sensor 1
30 #define LS2 34 //left sensor 2
31 #define RS1 36 //Right sensor 1
32 #define RS2 37 //Right sensor 2
33
34 /*****Motor driver*****/
35 #define Leftforward 4 //LeftMotor1 and LeftMotor2 forward (IM1) 4 30
36 #define LeftBackward 6 //LeftMotor1 and LeftMotor2 backward (IM2) 5 31
37 #define RightForward 7 //Rightmotor1 and RightMotor2 forward (IM3) 6 33
38 #define RightBackward 5 //Rightmotor1 and RightMotor2 backward (IM4) 7 32
39
40 /*****Flame Sensor*****/
41 #define flameThreshold 200
```

Figure 7: Implementation of code (5)

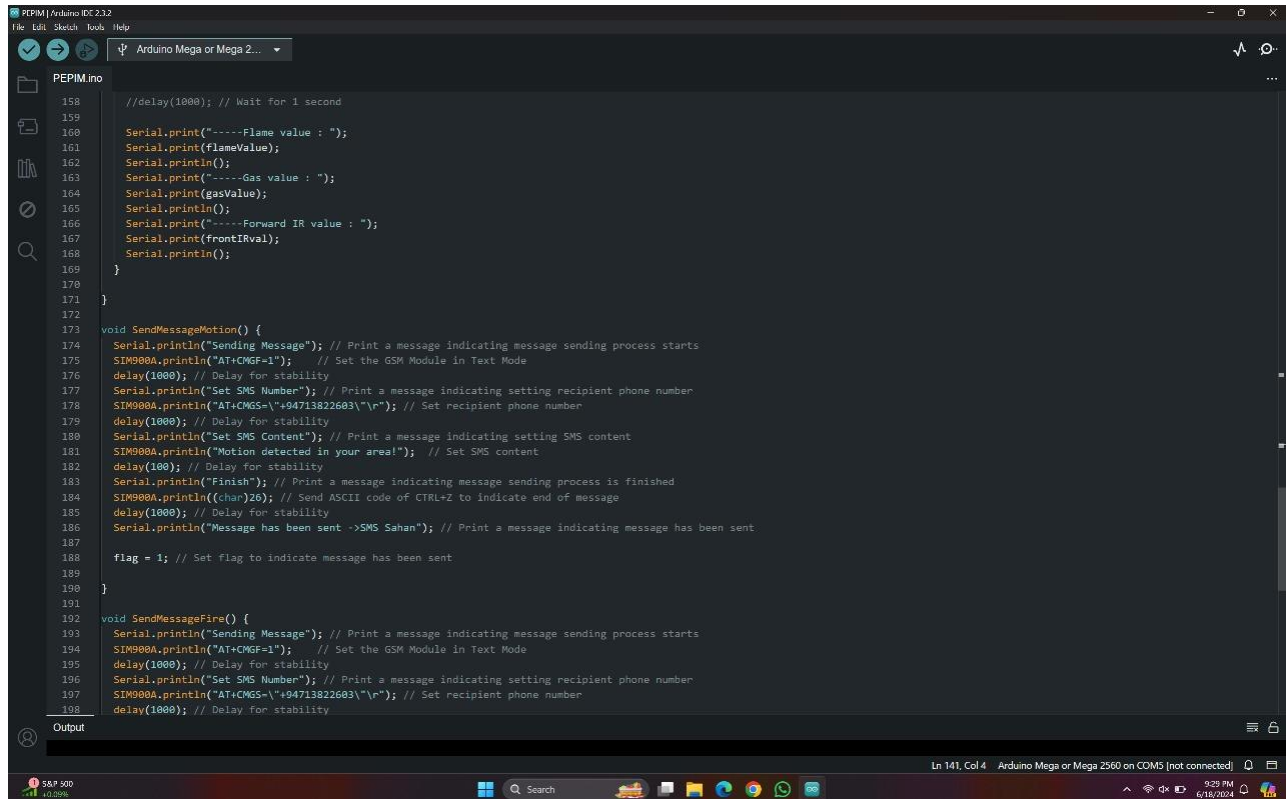
```
PEPIM.ino
173 void SendMessageMotion() {
174   Serial.println("Sending Message"); // Print a message indicating message sending process starts
175   SIM900A.println("AT+CMGF=1"); // Set the GSM Module in Text Mode
176   delay(1000); // Delay for stability
177   Serial.println("Set SMS Number"); // Print a message indicating setting recipient phone number
178   SIM900A.println("AT+CMGS=\"+94713822603\""); // Set recipient phone number
179   delay(1000); // Delay for stability
180   Serial.println("Set SMS Content"); // Print a message indicating setting SMS content
181   SIM900A.println("Motion detected in your area!"); // Set SMS content
182   delay(100); // Delay for stability
183   Serial.println("Finish"); // Print a message indicating message sending process is finished
184   SIM900A.println((char)26); // Send ASCII code of CTRL+Z to indicate end of message
185   delay(1000); // Delay for stability
186   Serial.println("Message has been sent ->SMS Sahan"); // Print a message indicating message has been sent
187
188   flag = 1; // Set flag to indicate message has been sent
189 }
190
191 void SendMessageFire() {
192   Serial.println("Sending Message"); // Print a message indicating message sending process starts
193   SIM900A.println("AT+CMGF=1"); // Set the GSM Module in Text Mode
194   delay(1000); // Delay for stability
195   Serial.println("Set SMS Number"); // Print a message indicating setting recipient phone number
196   SIM900A.println("AT+CMGS=\"+94713822603\""); // Set recipient phone number
197   delay(1000); // Delay for stability
198   Serial.println("Set SMS Content"); // Print a message indicating setting SMS content
199   SIM900A.println("Fire detected in your area!"); // Set SMS content
200   delay(100); // Delay for stability
201   Serial.println("Finish"); // Print a message indicating message sending process is finished
202   SIM900A.println((char)26); // Send ASCII code of CTRL+Z to indicate end of message
203   delay(1000); // Delay for stability
204   Serial.println("Message has been sent ->SMS Sahan"); // Print a message indicating message has been sent
205
206   flag = 1; // Set flag to indicate message has been sent
207 }
208
209 void SendMessageGas() {
210   Serial.println("Sending Message"); // Print a message indicating message sending process starts
211   SIM900A.println("AT+CMGF=1"); // Set the GSM Module in Text Mode
212   delay(1000); // Delay for stability
```

Figure 8: Implementation of code (6)



```
121 }
122 else if(digitalRead(RS1) == LOW && digitalRead(LS1) == LOW){           //straighth
123     digitalWrite(LeftForward, HIGH);
124     digitalWrite(RightForward, HIGH);
125 }
126 else if(digitalRead(RS1) == HIGH && digitalRead(LS1) == HIGH){         //straighth
127     digitalWrite(LeftForward, LOW);
128     digitalWrite(RightForward, LOW);
129 }
130 else if(digitalRead(RS1) == HIGH && digitalRead(LS1) == HIGH && digitalRead(RS2) == HIGH && digitalRead(LS2) == HIGH){
131     if(6 <= currentHour <= 23){
132         digitalWrite(LeftForward, LOW);
133         digitalWrite(RightForward, LOW);
134     }
135     else{
136         digitalWrite(LeftForward, HIGH);
137         digitalWrite(RightForward, HIGH);
138     }
139 }
140 }
141 }
142 //DateTime now = rtc.now(); // Print the current time
143 {
144     Serial.print(now.year(), DEC);
145     Serial.print("/");
146     Serial.print(now.month(), DEC);
147     Serial.print("/");
148     Serial.print(now.day(), DEC);
149     Serial.print(" ");
150     Serial.print(now.hour(), DEC);
151     Serial.print(":");
152     Serial.print(now.minute(), DEC);
153     Serial.print(":");
154     Serial.print(now.second(), DEC);
155     Serial.println();
156 }
157 //delay(1000); // Wait for 1 second
158
159 Serial.print("-----Flame value : ");
160 Serial.print(flameValue);
161 }
```

Figure 9: Implementation of code (7)



```
158 //delay(1000); // Wait for 1 second
159
160 Serial.print("-----Flame value : ");
161 Serial.print(flameValue);
162 Serial.println();
163 Serial.print("-----Gas value : ");
164 Serial.print(gasValue);
165 Serial.println();
166 Serial.print("-----Forward IR value : ");
167 Serial.print(frontIRval);
168 Serial.println();
169 }
170 }
171 }
172 }
173 void SendMessageMotion() {
174     Serial.println("Sending Message"); // Print a message indicating message sending process starts
175     SIM900A.println("AT+CMGF=1"); // Set the GSM Module in Text Mode
176     delay(1000); // Delay for stability
177     Serial.println("Set SMS Number"); // Print a message indicating setting recipient phone number
178     SIM900A.println("AT+CMGS=\"+94713822683\"\r\n"); // Set recipient phone number
179     delay(1000); // Delay for stability
180     Serial.println("Set SMS Content"); // Print a message indicating setting SMS content
181     SIM900A.println("Motion detected in your area!"); // Set SMS content
182     delay(100); // Delay for stability
183     Serial.println("Finish"); // Print a message indicating message sending process is finished
184     SIM900A.println((char)26); // Send ASCII code of CTRL+Z to indicate end of message
185     delay(1000); // Delay for stability
186     Serial.println("Message has been sent ->SMS Sahan"); // Print a message indicating message has been sent
187 }
188
189 flag = 1; // Set flag to indicate message has been sent
190 }
191 }
192 void SendMessageFire() {
193     Serial.println("Sending Message"); // Print a message indicating message sending process starts
194     SIM900A.println("AT+CMGF=1"); // Set the GSM Module in Text Mode
195     delay(1000); // Delay for stability
196     Serial.println("Set SMS Number"); // Print a message indicating setting recipient phone number
197     SIM900A.println("AT+CMGS=\"+94713822683\"\r\n"); // Set recipient phone number
198     delay(1000); // Delay for stability
```


2.4 Testing

The testing phase was critical to ensure the Home Security Robot meets all specified requirements and operated reliably in real-world conditions. Various testing methodologies were employed, including unit testing of individual components, integration testing to verify the interaction between hardware and software, and system testing to evaluate the robot's overall performance. The robot was subjected to a series of simulated scenarios to test its motion detection, fire and gas detection, image capture, and remote monitoring capabilities. User acceptance testing was also conducted to gather feedback and make necessary adjustments.

Table 6: Motion Detection Implementation

Test Scenario ID	Motion Detection Implementation - 1	Test Case ID	Motion Detection - 1
Test Case Description	Motion detection frequency level	Test Priority	High
Pre-Requisite	Power on circuit	Post-Requisite	Connection to GSM module

Test Execution Steps:

S. No	Action	Inputs	Expected Output	Actual Output	Test Result
01	Simulating no motion	None	No change in PIR sensor LED	No change in PIR sensor LED	Pass
02	Simulating motion within the designated area	Motion	PIR sensor LED turns on and sends signal to GSM module	PIR sensor LED turns on and send signal to GSM module	Pass

Table 7: Fire Detection Implementation

Test Scenario ID	Fire Detection Implementation - 1	Test Case ID	Fire Detection - 1
Test Case Description	Fire Detection sensitivity	Test Priority	High
Pre-Requisite	Power on circuit	Post-Requisite	Connection to GSM module

Test Execution Steps:

S. No	Action	Inputs	Expected Output	Actual Output	Test Result
01	Simulating no fire	None	No change in flame sensor LED	No change in PIR sensor LED	Pass
02	Simulating fire in the vicinity	Flame/Hear	Flame sensor LED turns on and sends signal to GSM module	Flame sensor LED turns on and sends signal to GSM module	Pass

Table 8: Gas Detection Implementation

Test Scenario ID	Gas Detection Implementation - 1	Test Case ID	Gas Detection - 1
Test Case Description	Gas leak detection sensitivity	Test Priority	High
Pre-Requisite	Power on circuit	Post-Requisite	Connection to GSM module

Test Execution Steps:

S. No	Action	Inputs	Expected Output	Actual Output	Test Result
01	Simulating no gas leak	None	No change in gas sensor LED	No change in gas sensor LED	Pass
02	Simulating gas leak in the vicinity	Gas Leak	Gas sensor LED turns on and sends signal to GSM module	Gas sensor LED turns on and sends signal to GSM module	Pass

Table 9: Image Capture Implementation

Test Scenario ID	Image Capture Implementation - 1	Test Case ID	Image Capture - 1
Test Case Description	Image capture and storage	Test Priority	High
Pre-Requisite	Power on circuit	Post-Requisite	Connection to GSM module

Test Execution Steps:

S. No	Action	Inputs	Expected Output	Actual Output	Test Result
01	Capturing an image	Environment	Image captured and stored on SD card	Image captured and stored on SD card	Pass
02	Streaming video	Environment	Video streamed via Wi-Fi module	Video streamed via Wi-Fi module	Pass

\

Table 10: Wi-Fi and Communication

Test Scenario ID	Wi-Fi and Communication - 1	Test Case ID	Wi-Fi Communication
Test Case Description	Sending alerts and images via GSM and web interface	Test Priority	High
Pre-Requisite	Connection to GSM module	Post-Requisite	N/A

Test Execution Steps:

S. No	Action	Inputs	Expected Output	Actual Output	Test Result
01	Sending alert for motion detection	Signal from PIR sensor	Alert sent via GSM module	Alert sent via GSM module	Pass
02	Sending alert for fire detection	Signal from flame sensor	Alert sent via GSM module	Alert sent via GSM module	Pass
03	Sending alert for gas detection	Signal from gas sensor	Alert sent via GSM module	Alert sent via GSM module	Pass
04	Accessing captured images	IP address in browser	Images accessible through web interface	Images accessible through web interface	Pass

3. Evaluation

3.1 Assessment of the Project Results

The Home Security Robot project aimed to develop a robust solution capable of detecting intruders, fire, and gas leaks while providing real-time surveillance. The project was successful in several key areas.

1. Functionality of Sensors:

The integration of IR, PIR, KY-026 flame IR, and MQ2 gas sensors allowed the robot to effectively detect human motion, fire, and gas. During testing, the sensors consistently triggered alerts in response to these stimuli, meeting the project's primary objectives.

2. Real-Time Surveillance:

The OV2840 Camera Module provided clear, high-resolution images, which were successfully stored on the SD card. Users could access live feeds, ensuring continuous monitoring of the robot's surroundings.

3. Remote Access and Notifications:

The robot's ability to send notifications to users upon detecting motion, fire, or gas was validated. This feature enhances the responsiveness and security management capabilities of the system.

4. Patrolling Efficiency:

The robot effectively patrolled the designated area between 11 pm and 6 am, adhering to the set patrol schedule. Although it required manual charging, the robot performed its tasks reliably within a single battery cycle.

Overall, the project achieved its goals of enhancing security through autonomous monitoring, real-time data collection, and effective alert mechanisms.

3.2 Lessons Learned

Throughout the project, several important lessons were learned:

1. Sensor Calibration:

Precise calibration of sensors is crucial. Initial tests revealed sensitivity issues, which were resolved through careful adjustment, ensuring accurate detection of motion, fire, and gas.

2. Battery Management:

The assumption that the robot could patrol on a single charge held true. However, future designs should consider incorporating automatic charging stations to reduce manual intervention.

3. Obstacle Avoidance:

The lack of obstacle avoidance was identified as a limitation. Future iterations should integrate ultrasonic sensors or LIDAR for improved navigation and obstacle management.

4. User Interface and Experience:

Ensuring the user interface is intuitive and provides clear instructions is essential. User feedback indicated that detailed manuals and a user-friendly app or control interface could enhance user experience.

5. Cost Management:

Budget constraints were managed effectively, with the project being completed within the estimated Rs.20,000. However, cost-benefit analysis of additional features, such as enhanced obstacle avoidance and autonomous charging, should be considered in future projects.

3.3 Future Work

The Home Security Robot project has laid a strong foundation, but several enhancements can be pursued to increase its effectiveness and user convenience:

1. Obstacle Detection and Avoidance:

Integrating ultrasonic sensors or LIDAR to detect and navigate around obstacles will enable the robot to operate more autonomously and safely.

2. Autonomous Charging:

Developing a docking station for automatic recharging will eliminate the need for manual intervention, ensuring the robot is always operational.

3. Enhanced Connectivity:

Incorporating a WiFi module and developing a comprehensive mobile application will allow users to monitor and control the robot more effectively from remote locations.

4. AI integration:

AI algorithms for smarter threat detection and response could enhance the robot's capability to differentiate between real threats and false alarms.

5. Expanded Surveillance Features:

Adding features such as night vision for the camera module and extending the patrol hours could further improve the robot's surveillance capabilities.

By addressing these areas, the Home Security Robot can evolve into a more versatile and efficient security solution, meeting the increasing demands of modern security systems.

4. Conclusion

The Home Security Robot project aimed to provide an innovative and cost-effective solution for enhancing security in various environments, including residential, commercial, and industrial settings. Over the course of the project, we successfully developed a prototype that integrates multiple advanced sensor technologies and a high-resolution camera, effectively addressing the need for real-time surveillance and threat detection.

Our project's outcomes demonstrated the feasibility and practicality of using a robotic system for security purposes. The robot's ability to detect human motion, fire, and gas, coupled with its real-time alert system and live video feed capabilities, confirmed the project's primary objectives. The system's affordability and ease of implementation further underline its potential as a valuable addition to modern security measures.

Through this project, we gained valuable insights into the complexities of integrating various hardware components and the importance of precise calibration for optimal performance. We also recognized the significance of user-friendly interfaces and the need for autonomous features to enhance user experience and system reliability.

Looking forward, we identified several areas for future improvement, such as incorporating obstacle detection and avoidance, autonomous charging, enhanced connectivity, and AI integration. These enhancements will not only increase the robot's operational efficiency but also expand its applicability across different scenarios.

In conclusion, the Home Security Robot project has proven to be a successful endeavor, providing a solid foundation for further development. The project not only met its initial goals but also opened new avenues for innovation in the field of security robots. By continuing to build on these results, we can contribute to creating safer and more secure environments through technological advancements.

5. References

- [1] U. Hofer and D. Gutmachera, “Fire gas detection,” *Procedia Eng.*, vol. 47, pp. 1446–1459, 2012.
- [2] “Arduino modules - flame sensor,” [projecthub.arduino.cc](https://projecthub.arduino.cc/SURYATEJA/arduino-modules-flame-sensor-e48e97). [Online]. Available: <https://projecthub.arduino.cc/SURYATEJA/arduino-modules-flame-sensor-e48e97>. [Accessed: 16-Apr-2024].
- [3] A. Claudi, F. Di Benedetto, G. Dolcini, L. Palazzo, and A. F. Dragoni, “MARVIN: Mobile autonomous robot for video surveillance networks,” in *2012 Sixth UKSim/AMSS European Symposium on Computer Modeling and Simulation*, 2012, pp. 21–26.
- [4] Wikipedia contributors, “Knightscope,” *Wikipedia, The Free Encyclopedia*, 16-Apr-2024. [Online]. Available: <https://en.wikipedia.org/w/index.php?title=Knightscope&oldid=1219708317>.
- [5] “Arduino SD card Module - Geeetech Wiki,” *Geeetech.com*. [Online]. Available: https://www.geeetech.com/wiki/index.php/Arduino_SD_card_Module. [Accessed: 17-Apr-2024].
- [6] “Line sensors and how to use them,” *FutureLearn*, 08-Aug-2013. [Online]. Available: <https://www.futurelearn.com/info/courses/robotics-with-raspberry-pi/0/steps/75899>. [Accessed: 19-Apr-2024].
- [7] -. Robocraze -, “Interfacing GSM module with arduino,” *Robocraze*, 13-Apr-2022. [Online]. Available: <https://robocraze.com/blogs/post/interfacing-gsm-module-with-arduino>. [Accessed: 19-Apr-2024].
- [8] Last Minute Engineers, “How HC-SR501 PIR sensor works & how to interface it with arduino,” *Last Minute Engineers*, 03-Jul-2018. [Online]. Available: <https://lastminuteengineers.com/pir-sensor-arduino-tutorial/>. [Accessed: 19-Apr-2024]