Lab 5
Alarm System

GROUP 1
SEG4145 REAL TIME SYSTEMS

01

Introduction

Introduction

Alexander Choukeir

Alexis Verana

Gavin Gao

Hened Saade

Jayden Lachhman

Overview

The main goal of this project is to create an alarm system.

The alarm system should be able to let user set a password to arm or disarm the alarm system. As well as detect any motion.

The alarm system consists of:

- LED display screen
- Keypad
- Motion detector
- Buzzer
- LED Lights

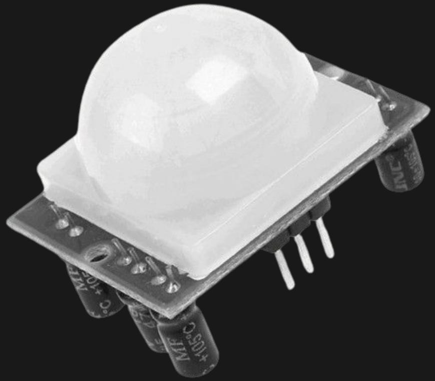
02

Problem Statement & Solution Overview

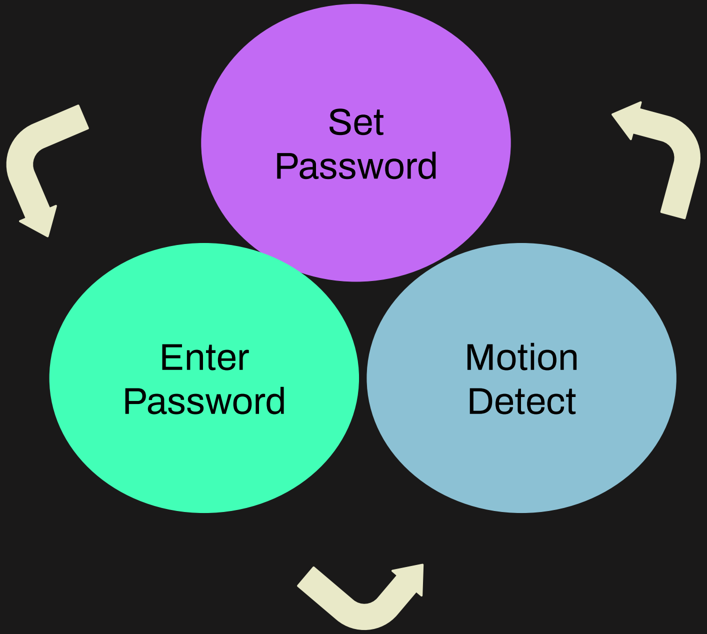
Security Challenges

The security challenges that are addressed are:

- Password restrictions
- Password detection
- Motion detection

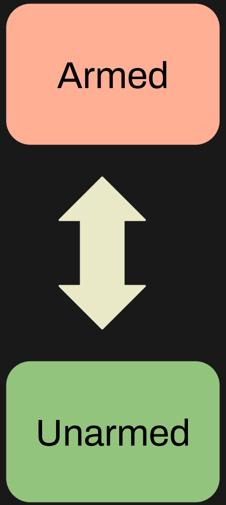


System Overview

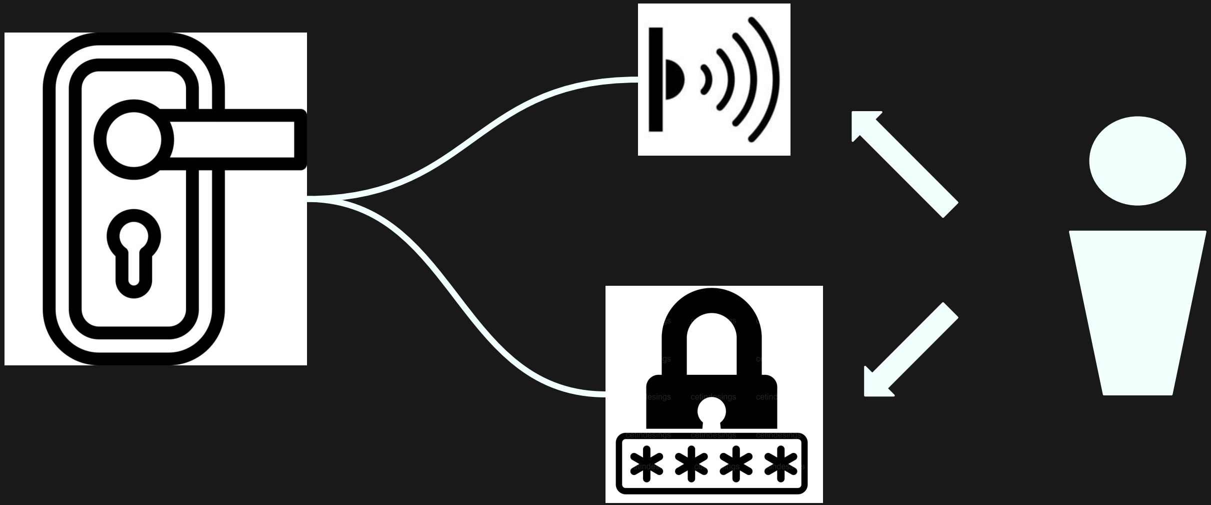


The entire system is built based on three tasks: setPassword, Enter Password and motion Detect.

The system status can switch between unarmed and armed after set password task and Enter password complete



System Overview



03

System Architecture and Design

Hardware Components

Interfaces

- Nucleo-F446RE Development Board
- Circuit Board
- Wires
- Resistors

Input

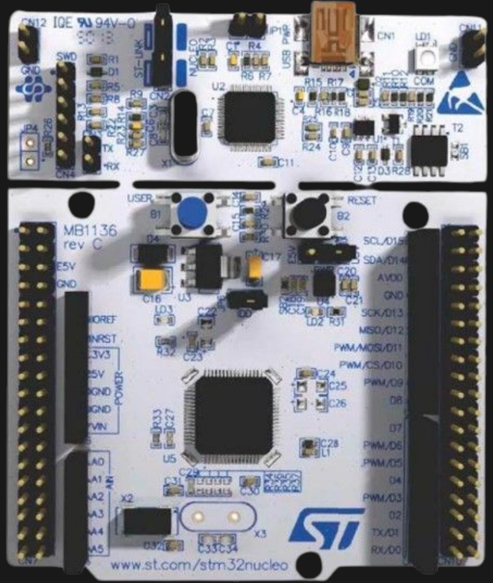
- 4x4 Keypad
- PIR Motion Sensor

Output

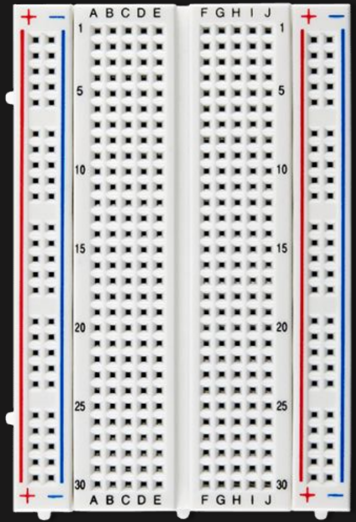
- LEDs
- OLED Display Module
- Buzzer

Interfaces

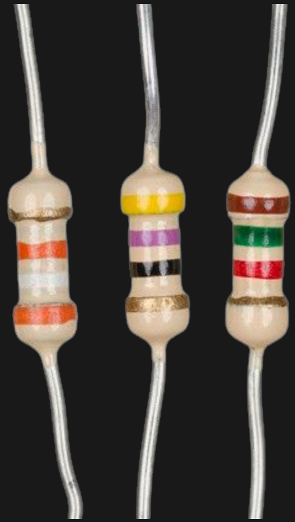
Nucleo-F446RE Development Board



Circuit Board



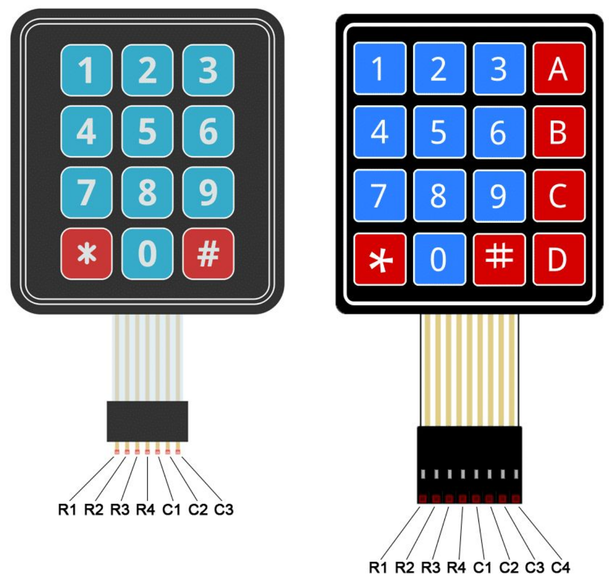
Resistors



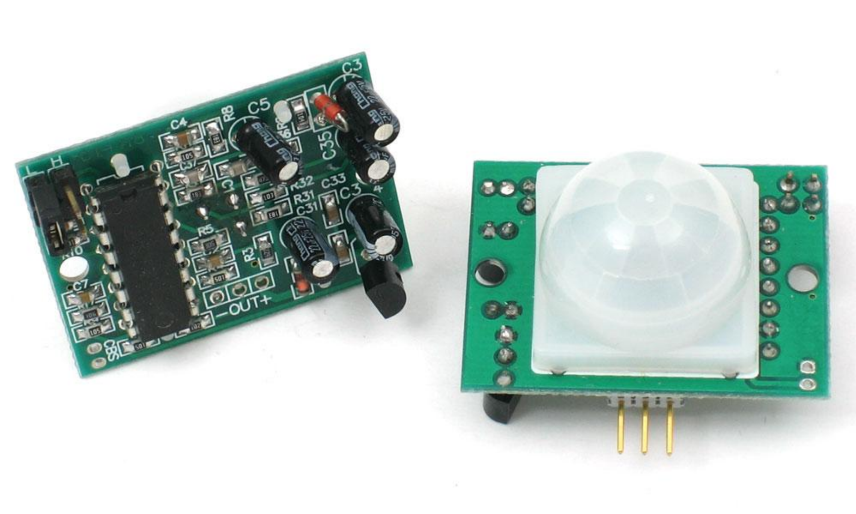
Wires

Input

4x4 Keypad

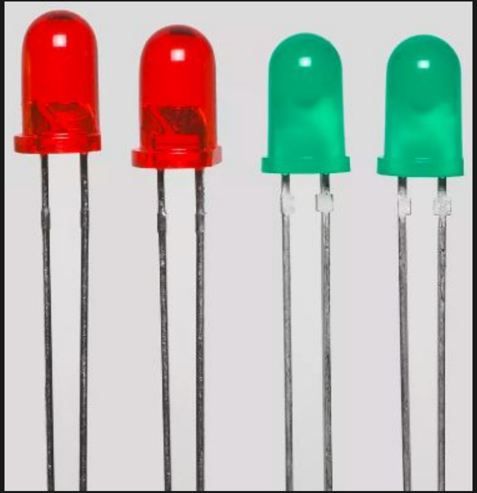


PIR Motion Detector

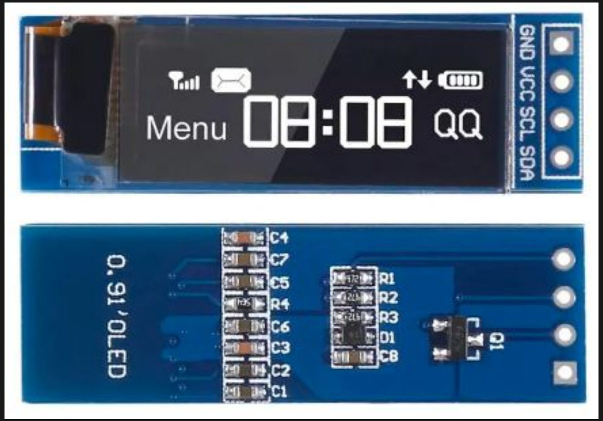


Output

LEDs



OLED
Display Module



Buzzer

Software Components

Utilizes FreeRTOS

- Software libraries:
  - CMSIS_V2
  - HAL
  - SSD1306
  - Keypad4x4
- Pins:
  - LEDs
  - LCD
  - Keypad
  - PIR Motion Detector
  - Buzzer

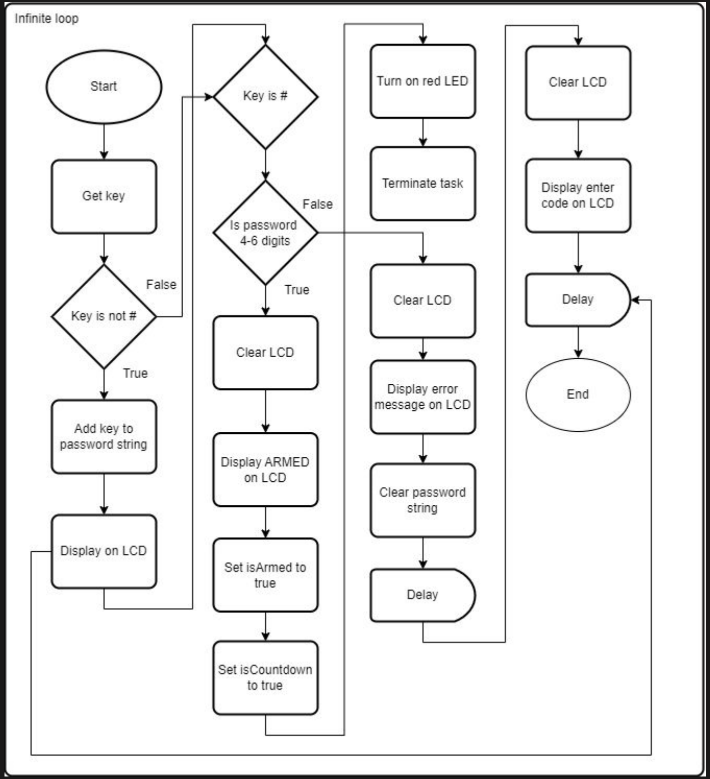
Software Components

Using # key for enter and * key for rearm

Three tasks:

- SetPasswordTask
  - Priority: High
- EnterPasswordTask
  - Priority: Normal
- DetectMotionTask
  - Priority: Normal

SetPasswordTask



Infinite loop

Start

Get key

Key is not #

False

True

Add key to password string

Display on LCD

Key is #

Is password 4-6 digits

False

True

Clear LCD

Display ARMED on LCD

Set isArmed to true

Set isCountdown to true

Turn on red LED

Terminate task

Clear LCD

Display error message on LCD

Clear password string

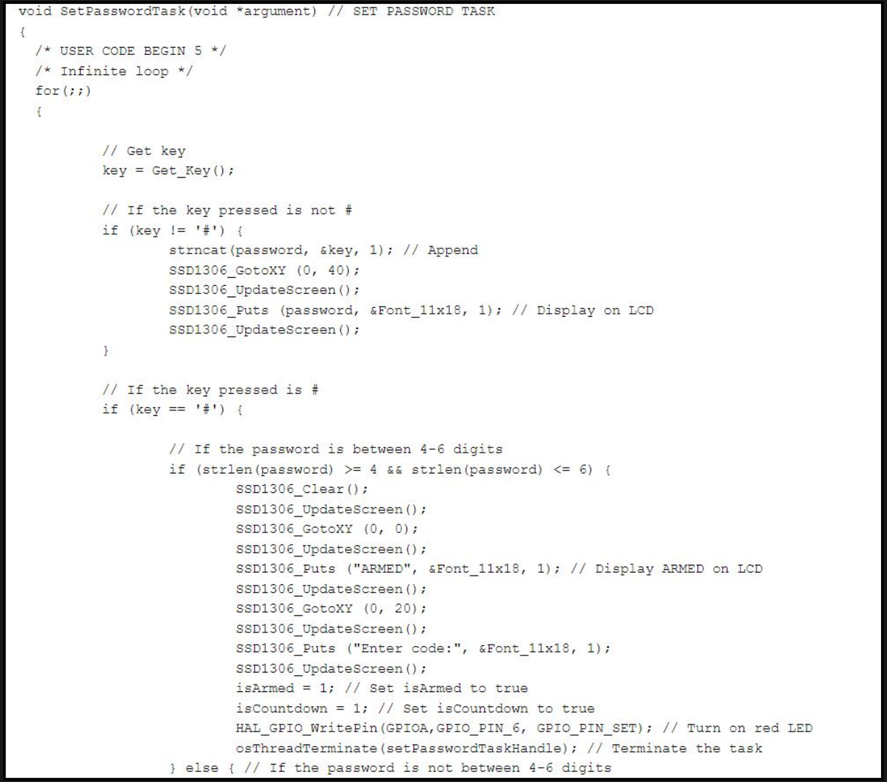
Delay

Clear LCD

Display enter code on LCD

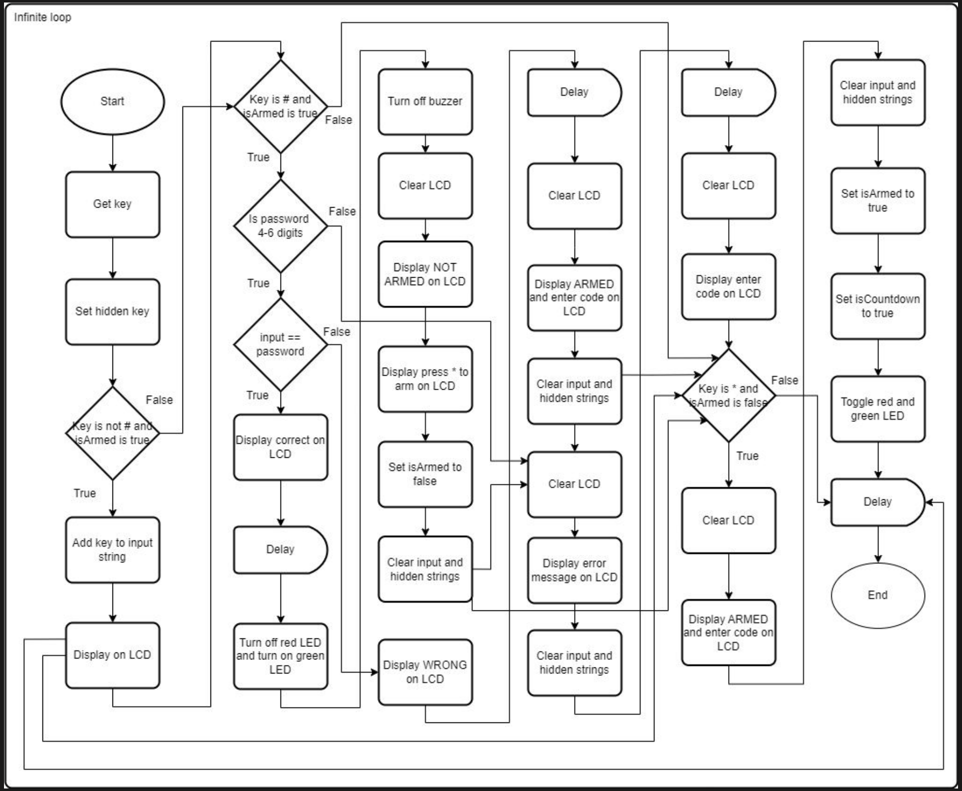
Delay

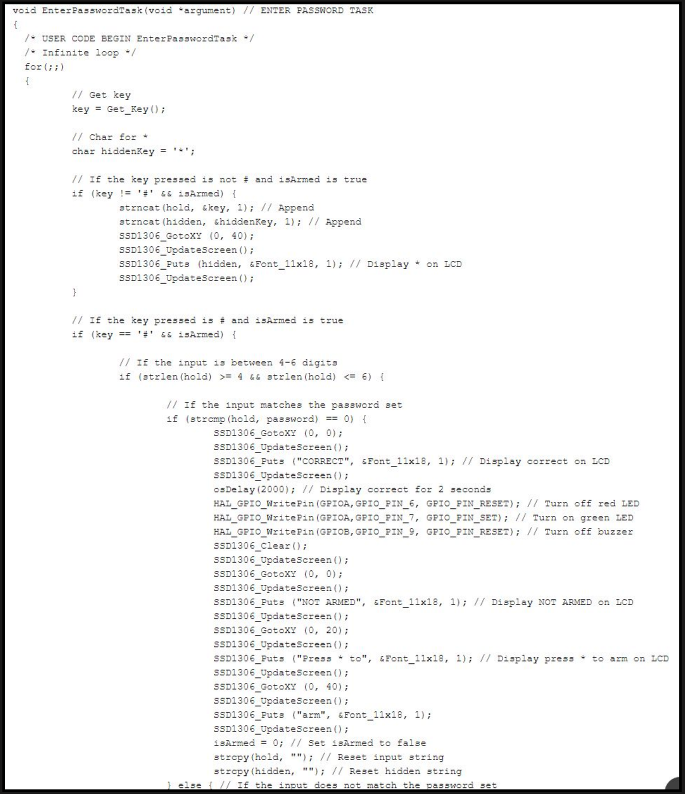
End



```
void SetPasswordTask(void *argument) // SET PASSWORD TASK
{
  /* USER CODE BEGIN 5 */
  /* Infinite loop */
  for(;;)
  {

          // Get key
          key = Get_Key();

          // If the key pressed is not #
          if (key != '#') {
                  strncat(password, &key, 1); // Append
                  SSD1306_GotoXY (0, 40);
                  SSD1306_UpdateScreen();
                  SSD1306_Puts (password, &Font_11x18, 1); // Display on LCD
                  SSD1306_UpdateScreen();
          }

          // If the key pressed is #
          if (key == '#') {

                  // If the password is between 4-6 digits
                  if (strlen(password) >= 4 && strlen(password) <= 6) {
                          SSD1306_Clear();
                          SSD1306_UpdateScreen();
                          SSD1306_GotoXY (0, 0);
                          SSD1306_UpdateScreen();
                          SSD1306_Puts ("ARMED", &Font_11x18, 1); // Display ARMED on LCD
                          SSD1306_UpdateScreen();
                          SSD1306_GotoXY (0, 20);
                          SSD1306_UpdateScreen();
                          SSD1306_Puts ("Enter code:", &Font_11x18, 1);
                          SSD1306_UpdateScreen();
                          isArmed = 1; // Set isArmed to true
                          isCountdown = 1; // Set isCountdown to true
                          HAL_GPIO_WritePin(GPIOA,GPIO_PIN_6, GPIO_PIN_SET); // Turn on red LED
                          osThreadTerminate(setPasswordTaskHandle); // Terminate the task
                  } else { // If the password is not between 4-6 digits
```

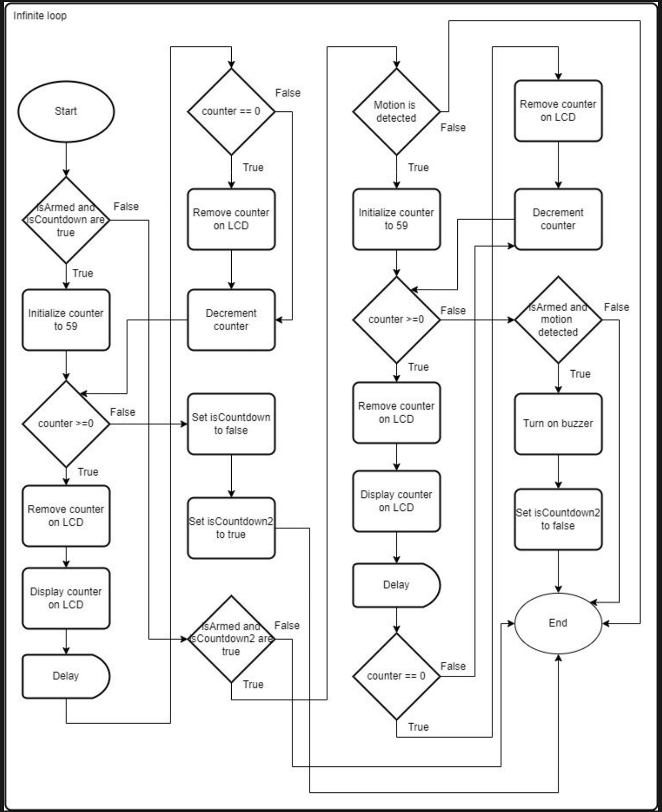
EnterPasswordTask

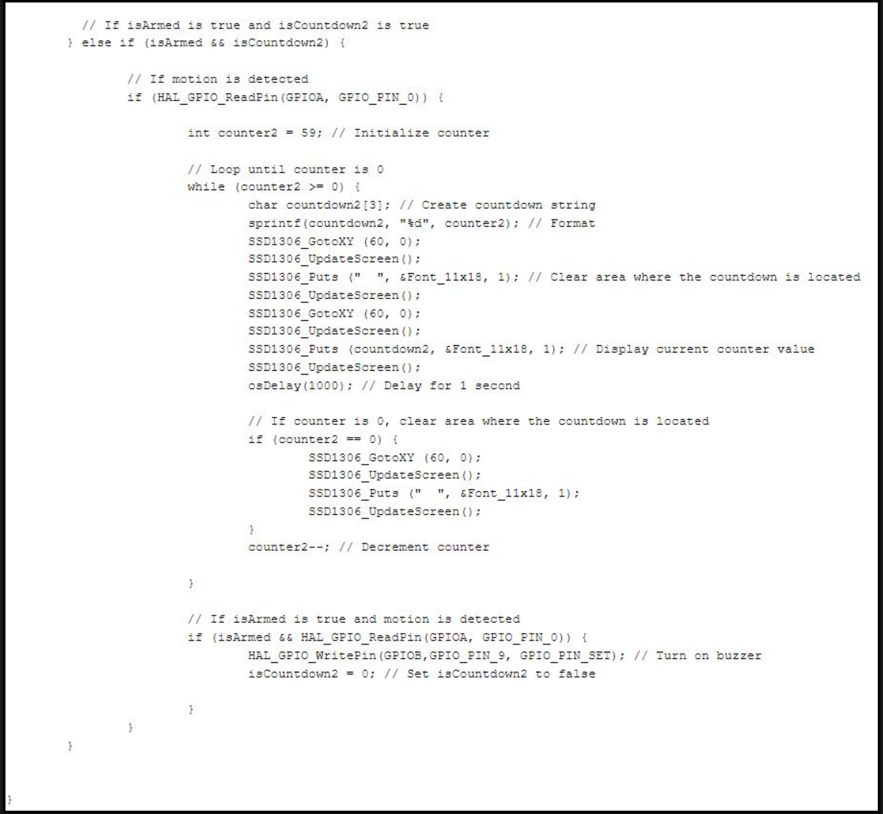

Infinite loop
Start
Get key
Set hidden key
Key is not # and isArmed is true
False
True
Add key to input string
Display on LCD
Key is # and isArmed is true
False
True
Is password 4-6 digits
False
True
input == password
False
True
Display correct on LCD
Delay
Turn off red LED and turn on green LED
Turn off buzzer
Clear LCD
Display NOT ARMED on LCD
Display press * to arm on LCD
Set isArmed to false
Clear input and hidden strings
Display WRONG on LCD
Delay
Clear LCD
Display ARMED and enter code on LCD
Clear input and hidden strings
Clear LCD
Display error message on LCD
Clear input and hidden strings
Delay
Clear LCD
Display enter code on LCD
Key is * and isArmed is false
False
True
Clear LCD
Display ARMED and enter code on LCD
Clear input and hidden strings
Set isArmed to true
Set isCountdown to true
Toggle red and green LED
Delay
End



```
void EnterPasswordTask(void *argument) // ENTER PASSWORD TASK
{
  /* USER CODE BEGIN EnterPasswordTask */
  /* Infinite loop */
  for(;;)
  {
          // Get key
          key = Get_Key();

          // Char for *
          char hiddenKey = '*';

          // If the key pressed is not # and isArmed is true
          if (key != '#' && isArmed) {
                  strncat(hold, &key, 1); // Append
                  strncat(hidden, &hiddenKey, 1); // Append
                  SSD1306_GotoXY (0, 40);
                  SSD1306_UpdateScreen();
                  SSD1306_Puts (hidden, &Font_11x18, 1); // Display * on LCD
                  SSD1306_UpdateScreen();
          }

          // If the key pressed is # and isArmed is true
          if (key == '#' && isArmed) {

                  // If the input is between 4-6 digits
                  if (strlen(hold) >= 4 && strlen(hold) <= 6) {

                          // If the input matches the password set
                          if (strcmp(hold, password) == 0) {
                                  SSD1306_GotoXY (0, 0);
                                  SSD1306_UpdateScreen();
                                  SSD1306_Puts ("CORRECT", &Font_11x18, 1); // Display correct on LCD
                                  SSD1306_UpdateScreen();
                                  osDelay(2000); // Display correct for 2 seconds
                                  HAL_GPIO_WritePin(GPIOA,GPIO_PIN_6, GPIO_PIN_RESET); // Turn off red LED
                                  HAL_GPIO_WritePin(GPIOA,GPIO_PIN_7, GPIO_PIN_SET); // Turn on green LED
                                  HAL_GPIO_WritePin(GPIOB,GPIO_PIN_9, GPIO_PIN_RESET); // Turn off buzzer
                                  SSD1306_Clear();
                                  SSD1306_UpdateScreen();
                                  SSD1306_GotoXY (0, 0);
                                  SSD1306_UpdateScreen();
                                  SSD1306_Puts ("NOT ARMED", &Font_11x18, 1); // Display NOT ARMED on LCD
                                  SSD1306_UpdateScreen();
                                  SSD1306_GotoXY (0, 20);
                                  SSD1306_UpdateScreen();
                                  SSD1306_Puts ("Press * to", &Font_11x18, 1); // Display press * to arm on LCD
                                  SSD1306_UpdateScreen();
                                  SSD1306_GotoXY (0, 40);
                                  SSD1306_UpdateScreen();
                                  SSD1306_Puts ("arm", &Font_11x18, 1);
                                  SSD1306_UpdateScreen();
                                  isArmed = 0; // Set isArmed to false
                                  strcpy(hold, ""); // Reset input string
                                  strcpy(hidden, ""); // Reset hidden string
                          } else { // If the input does not match the password set
```

DetectMotionTask

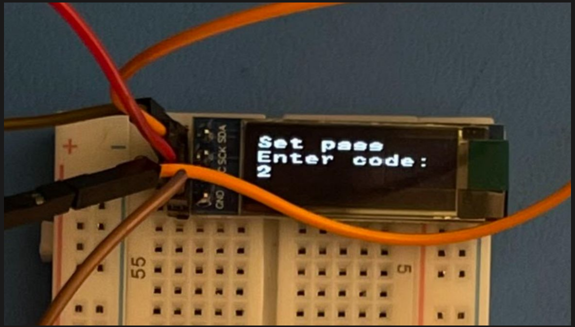


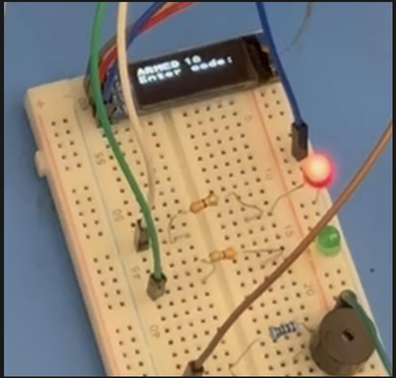


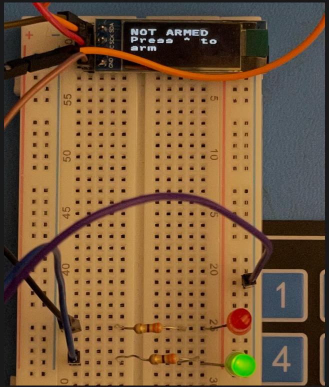
```
      // If isArmed is true and isCountdown2 is true
    } else if (isArmed && isCountdown2) {

        // If motion is detected
        if (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0)) {

            int counter2 = 59; // Initialize counter

            // Loop until counter is 0
            while (counter2 >= 0) {
                char countdown2[3]; // Create countdown string
                sprintf(countdown2, "%d", counter2); // Format
                SSD1306_GotoXY (60, 0);
                SSD1306_UpdateScreen();
                SSD1306_Puts ("  ", &Font_11x18, 1); // Clear area where the countdown is located
                SSD1306_UpdateScreen();
                SSD1306_GotoXY (60, 0);
                SSD1306_UpdateScreen();
                SSD1306_Puts (countdown2, &Font_11x18, 1); // Display current counter value
                SSD1306_UpdateScreen();
                osDelay(1000); // Delay for 1 second

                // If counter is 0, clear area where the countdown is located
                if (counter2 == 0) {
                    SSD1306_GotoXY (60, 0);
                    SSD1306_UpdateScreen();
                    SSD1306_Puts ("  ", &Font_11x18, 1);
                    SSD1306_UpdateScreen();
                }
                counter2--; // Decrement counter

            }

            // If isArmed is true and motion is detected
            if (isArmed && HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0)) {
                HAL_GPIO_WritePin(GPIOB,GPIO_PIN_9, GPIO_PIN_SET); // Turn on buzzer
                isCountdown2 = 0; // Set isCountdown2 to false

            }
        }
    }


}
```

User interface design

- System status at the top
- Under this is a message to enter code
- Code entered is displayed under this
- Beside system status at the top is the countdown
- If the system is not armed, LCD will display message telling the user to press * to arm







04

Performance Metrics and Future Enhancements

Performance Evaluation

In evaluating the performance of our alarm system, two key performance metrics were considered:

- Response time:
  - Rapid response time to motion detection
  - Consistently responds within milliseconds
- Accuracy:
  - Minimal instances of false positives/negatives
  - Accurately distinguishes between genuine threats and environmental noise

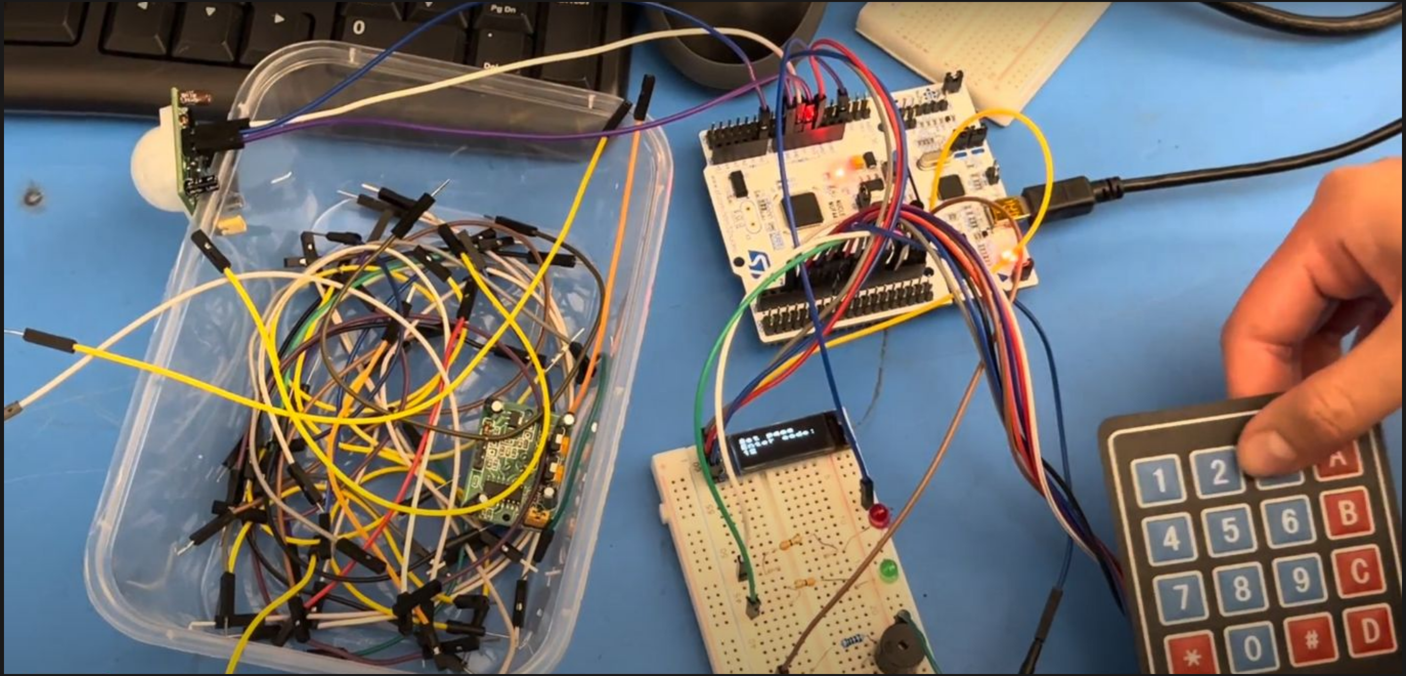
Future Enhancements

- Integration with IoT devices:
  - Remote monitoring and control via smartphones or smart devices
- Enhanced user interface:
  - Intuitive interface with touchscreen controls or mobile app
- Advanced motion detection algorithms:
  - Employ machine learning for improved accuracy
- Integration with home automation systems:
  - Coordination with other smart home devices for comprehensive automation

05

Conclusion

Conclusion



Thank you.