



Security System Project

Stage 3 (Embedded Systems – CIE 408)

Table of Contents:

I.	<i>Introduction and literature survey</i>	<i>(2:3)</i>
II.	<i>Block Diagram</i>	<i>(3)</i>
III.	<i>Components</i>	<i>(4:5)</i>
IV.	<i>Hardware</i>	<i>(6:7)</i>
V.	<i>Code</i>	<i>(8:10)</i>
VI.	<i>Mobile App</i>	<i>(11:12)</i>
VII.	<i>Bonus task</i>	<i>(12:14)</i>
VIII.	<i>FreeRTOS</i>	<i>(14:17)</i>

Belal Gamal -202101237

Abd Alrhman Ahmed - 202101112

Marwan Ahmed - 202101214

Yousef Khaled – 202100191

Mohamed Hosam - 202100975



I. Introduction and Literature Survey

Security systems protect homes, companies, and public spaces by detecting unwanted entrance or suspicious activity. The development of embedded systems and the Internet of Things has led to the integration of real-time monitoring, cloud-based data storage, and automatic warnings into contemporary security solutions. Using the Tiva C Series microcontroller and FreeRTOS for task management, this project aims to create an effective security system that integrates motion and sound sensors to identify possible threats. Real-time remote monitoring is also made possible by the data transmission to Firebase using an ESP8266 NodeMCU module.

Literature Survey:

1. Use of Motion and Sound Sensors

- Motion sensors, such as Passive Infrared (PIR) sensors, detect movement by sensing changes in infrared radiation. PIR sensors are widely used in security applications due to their low power consumption and high reliability.
- Sound sensors detect acoustic signals and can be used to identify anomalies such as unauthorized entry, breaking glass, or suspicious noise levels.

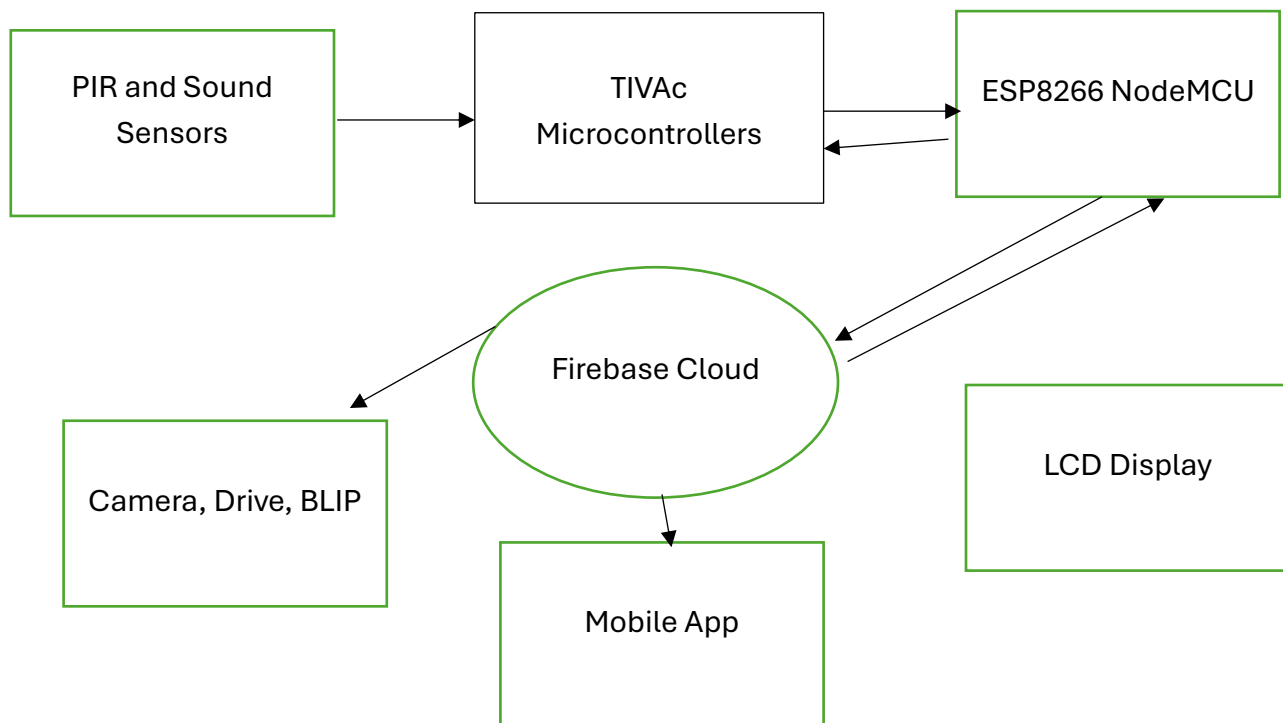
2. Microcontroller-Based Security Systems Research highlights the importance of microcontrollers, such as the Tiva C Series, in security applications. These microcontrollers offer real-time processing capabilities, multiple GPIOs for sensor integration, and efficient power management. FreeRTOS is commonly used in embedded systems.



3. Wireless Communication for Security Systems

- The ESP8266 NodeMCU module provides reliable Wi-Fi connectivity, enabling seamless communication between the microcontroller and cloud-based services.
- Cloud platform -Firebase, disused in Labs- allows secure data storage, real-time updates, and remote access, enhancing the scalability and effectiveness of security systems.

II. Block Diagram





III. Components

Tiva C Series Microcontroller (TM4C123G)	Borrow from the lab
ESP8266 NodeMCU	180 LE
PIR Motion Sensor	60 LE
Sound Sensor Module	65 LE
LCD Display (16x2 with I2C adapter)	95 LE
5V Power Adapter (1A-2A output)	70 LE
LEDs & Buzzer	35 LE
Breadboard & Jumper Wires	25 LE
Resistors & Capacitors	15 LE

Components Functionality:

1. Microcontroller & Communication Modules

- **Tiva C Series Microcontroller (TM4C123G)** → Core processing unit, required for sensor integration and task management.
- **ESP8266 NodeMCU** → Wi-Fi module for sending data to Firebase.

2. Sensors

- **PIR Motion Sensor** → Detects motion within a specific range.
- **Sound Sensor Module** → Detects unusual noise levels.



3. Display & Indicators

- **LCD Display (16x2 with I2C adapter)** → Displays system status or alerts.
- **LEDs & Buzzer** → Provides immediate on-site alerts when a security event is detected.

4. Power Supply

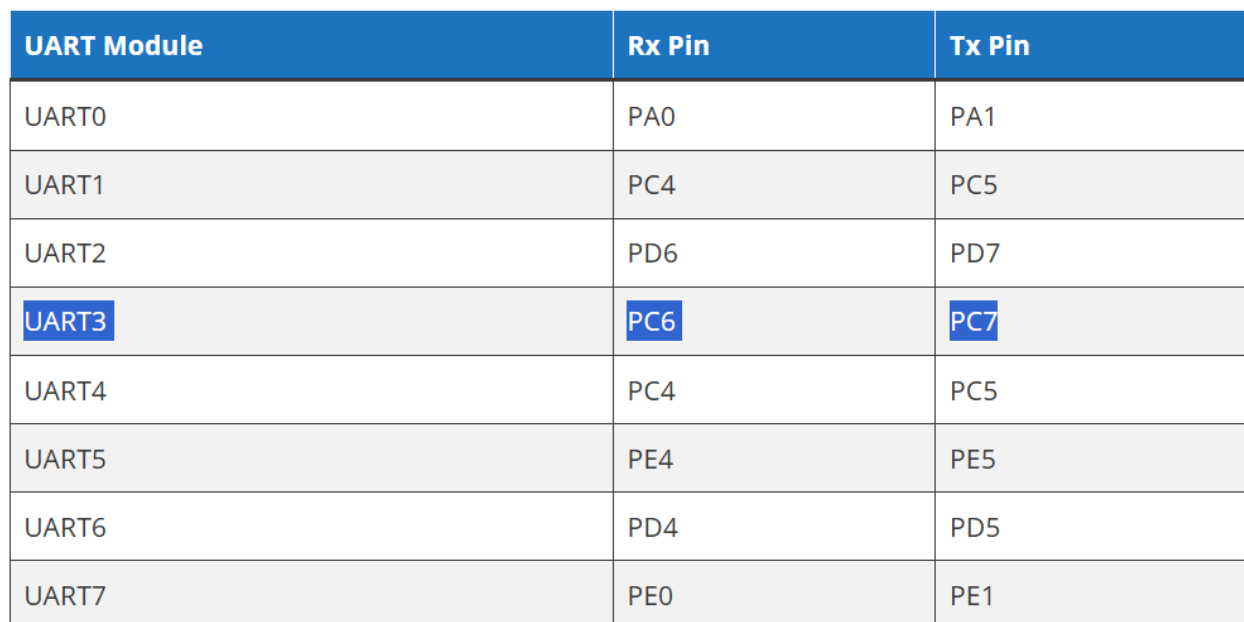
- **5V Power Adapter (1A-2A output) & Voltage Regulators** → Ensures stable power supply for the microcontroller and sensors.

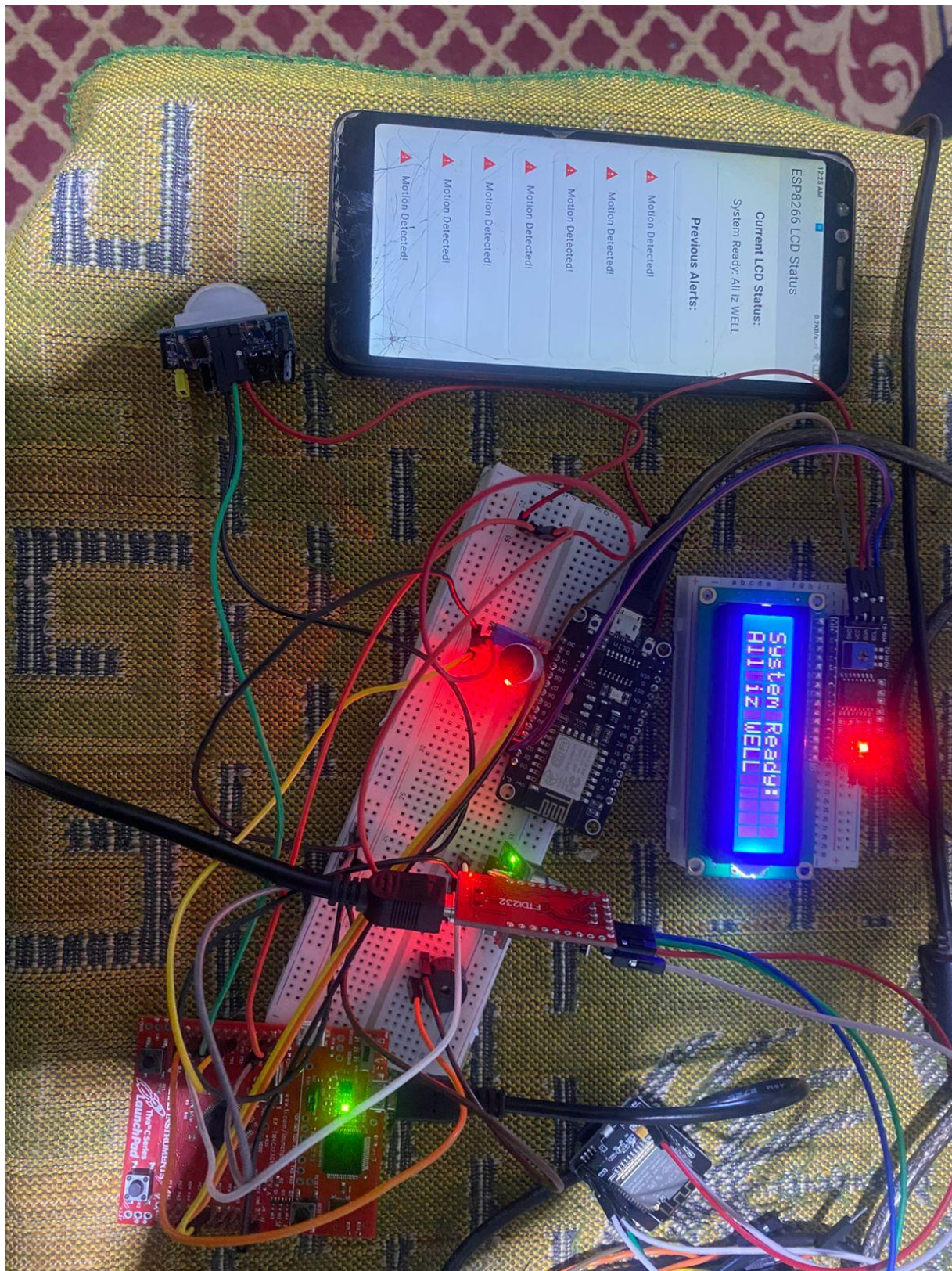
Software Requirements

- **FreeRTOS** (for task management).
- **Firebase Setup** (cloud integration for alerts & monitoring).
- **Mobile App/Web Dashboard** (for remote access).



TivaC connected and handle three sensors (PIR, Sound, Somke). The esp8266 connected to the LCD using I2C, and connected to TivaC using UART3 in tiva.







V. Code

Esp: receiving data from tivac and upload it to the firebase and print it in the LCD

```
1 #include <ESP8266WiFi.h>
2 #include <Firebase_ESP_Client.h>
3 #include <LiquidCrystal_I2C.h>
4
5 // Addons for Firebase token handling
6 #include "addons/TokenHelper.h"
7 #include "addons/RTDBHelper.h"
8
9 // WiFi credentials
10 #define WIFI_SSID "WE87AA46"
11 #define WIFI_PASSWORD "BYOA-20242024"
12
13 // Firebase configuration
14 #define API_KEY "AtzaSyChECeMTdgbSDg5RTKU9QGV1C-fQmwVE5w"
15 #define DATABASE_URL "https://embproject-15e7f-default-rtbd.firebaseio.com/"
16
17 // Firebase setup objects
18 FirebaseData fbdo;
19 FirebaseAuth auth;
20 FirebaseConfig config;
21 bool signupOK = false;
22
23 // LCD setup
24 LiquidCrystal_I2C lcd(0x27, 16, 2);
25
26 unsigned long lastReceiveTime = 0;
27 const unsigned long timeout = 5000; // 5 seconds
28 bool messageReceived = false;
29
30 void setup() {
31   Serial.begin(9600);
32
33   // Connect to WiFi
```

```
34   WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
35   Serial.print("Connecting to WiFi!");
36   while (WiFi.status() != WL_CONNECTED) {
37     delay(500);
38     Serial.print(".");
39   }
40   Serial.println("\nConnected to WiFi");
41
42   // Firebase config
43   config.api_key = API_KEY;
44   config.database_url = DATABASE_URL;
45
46   if (Firebase.signup(&config, &auth, "", "")) {
47     Serial.println("Firebase signup OK");
48     signupOK = true;
49   } else {
50     Serial.printf("Firebase signup failed: %s\n", config.signer.signupError.message.c_str());
51   }
52
53   Firebase.begin(&config, &auth);
54   Firebase.reconnectWiFi(true);
55
56   // LCD initialization
57   lcd.init();
58   lcd.backlight();
59   lcd.setCursor(0, 0);
60   lcd.print("Firebase Ready");
61   delay(1500);
62   lcd.clear();
63
64   lastReceiveTime = millis();
```




TivaC code:

Read from the three sensors and send data to esp through UART3

```
C:\Users\User\Desktop\ZC\IE 3(2)\Embedded\TIVAC\embTiva\proj - µVision [Non-Commercial Use License]
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
RunOnce

Project: embTiva
Target 1
Source Group 1
  startup_rvmdk.S
  masterTiva.c
  masterTiva.h
New Group
  driverlib.lib
  CMSIS

masterTiva.c
1 #include <stdint.h>
2 #include "m40123ghpm.h"
3 #include "masterTiva.h"
4 // === UART3 Initialization ===
5 void UART3_Init(void) {
6     SYSTCL_RCGUART_R |= (1 << 3); // Enable UART3 clock
7     SYSTCL_RCGCGPIO_R |= (1 << 2); // Enable GPIOC clock
8     while ((SYSTCL_FRGPIO_R & (1 << 2)) == 0) {}
9
10    GPIO_PORTC_AFSEL_R |= 0xC0; // PC6, PC7 alternate functions
11    GPIO_PORTC_PCTL_R = (GPIO_PORTC_PCTL_R & 0x00FFFFFF) | (0x11 << 24);
12    GPIO_PORTC_DEN_R |= 0xC0;
13    GPIO_PORTC_AMSEL_R &= ~0xC0;
14
15    UART3_CTL_R &= ~(1 << 0);
16    UART3_IBRD_R = 104;
17    UART3_FBRD_R = 11;
18    UART3_LCRH_R = (0x3 << 5);
19    UART3_CTL_R |= (1 << 0) | (1 << 8) | (1 << 9);
20
21
22    // === UART Send Functions ===
23    void UART3_OutChar(char data) {
24        while ((UART3_FR_R & (1 << 5)) != 0) {}
25        UART3_DR_R = data;
26    }
27
28    void UART3_OutString(const char* str) {
29        while (*str) {
30            UART3_OutChar(*str++);
31        }
32    }
33
34    // === PortF Initialization for SW1, Red and Green LEDs ===
35    void PortF_Init(void) {
36        SYSTCL_RCGCGPIO_R |= (1 << 5);
37        while ((SYSTCL_FRGPIO_R & (1 << 5)) == 0) {}
38
39        GPIO_PORTF_LOCK_R = 0x4C4F4344;
40        GPIO_PORTF_CR_R |= 0x1F;
41        GPIO_PORTF_DIR_R |= (1 << 1) | (1 << 3); // PF1 (Red LED), PF3 (Green LED) outputs
42        GPIO_PORTF_DIR_R &= ~(1 << 4); // PF4 input (SW1)
43        GPIO_PORTF_DEN_R |= 0x1A; // PF1, PF3, PF4 digital enable
44        GPIO_PORTF_PUR_R |= (1 << 4); // Pull-up on PF4
45    }
46
47    // === PortE Initialization for PIR, Smoke Sensors and Buzzer ===
48    void PortE_Init(void) {
49        SYSTCL_RCGCGPIO_R |= (1 << 4); // Enable clock for Port E
50        while ((SYSTCL_FRGPIO_R & (1 << 4)) == 0) {}
51
52        GPIO_PORTE_DIR_R &= ~(1 << 1) | (1 << 2); // PE1 (PIR), PE2 (Smoke) inputs
53        GPIO_PORTE_DIR_R |= (1 << 3); // PE3 output (Buzzer)
54        GPIO_PORTE_DEN_R |= (1 << 1) | (1 << 2) | (1 << 3); // Enable PE1-PE3
55        GPIO_PORTE_DATA_R &= ~(1 << 3); // Buzzer initially OFF
56    }
57
58    // === PortD Initialization for Sound Sensor ===
59    void PortD_Init(void) {
60        SYSTCL_RCGCGPIO_R |= (1 << 3); // Enable clock for Port D
61        while ((SYSTCL_FRGPIO_R & (1 << 3)) == 0) {}
62
63        GPIO_PORTD_DIR_R &= ~(1 << 2); // PD2 input (Sound Sensor)
64        GPIO_PORTD_DEN_R |= (1 << 2); // Enable PD2
65    }
66
Build Output
Stellaris ICD1 L1 C1 CAP. NUM SCRL OVR. R/W
```

```
C:\Users\User\Desktop\ZC\IE 3(2)\Embedded\TIVAC\embTiva\proj - µVision [Non-Commercial Use License]
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
RunOnce

Project: embTiva
Target 1
Source Group 1
  startup_rvmdk.S
  masterTiva.c
  masterTiva.h
New Group
  driverlib.lib
  CMSIS

masterTiva.c
31 }
32 }
33
34 // === PortF Initialization for SW1, Red and Green LEDs ===
35 void PortF_Init(void) {
36     SYSTCL_RCGCGPIO_R |= (1 << 5);
37     while ((SYSTCL_FRGPIO_R & (1 << 5)) == 0) {}
38
39     GPIO_PORTF_LOCK_R = 0x4C4F4344;
40     GPIO_PORTF_CR_R |= 0x1F;
41     GPIO_PORTF_DIR_R |= (1 << 1) | (1 << 3); // PF1 (Red LED), PF3 (Green LED) outputs
42     GPIO_PORTF_DIR_R &= ~(1 << 4); // PF4 input (SW1)
43     GPIO_PORTF_DEN_R |= 0x1A; // PF1, PF3, PF4 digital enable
44     GPIO_PORTF_PUR_R |= (1 << 4); // Pull-up on PF4
45 }
46
47 // === PortE Initialization for PIR, Smoke Sensors and Buzzer ===
48 void PortE_Init(void) {
49     SYSTCL_RCGCGPIO_R |= (1 << 4); // Enable clock for Port E
50     while ((SYSTCL_FRGPIO_R & (1 << 4)) == 0) {}
51
52     GPIO_PORTE_DIR_R &= ~(1 << 1) | (1 << 2); // PE1 (PIR), PE2 (Smoke) inputs
53     GPIO_PORTE_DIR_R |= (1 << 3); // PE3 output (Buzzer)
54     GPIO_PORTE_DEN_R |= (1 << 1) | (1 << 2) | (1 << 3); // Enable PE1-PE3
55     GPIO_PORTE_DATA_R &= ~(1 << 3); // Buzzer initially OFF
56 }
57
58 // === PortD Initialization for Sound Sensor ===
59 void PortD_Init(void) {
60     SYSTCL_RCGCGPIO_R |= (1 << 3); // Enable clock for Port D
61     while ((SYSTCL_FRGPIO_R & (1 << 3)) == 0) {}
62
63     GPIO_PORTD_DIR_R &= ~(1 << 2); // PD2 input (Sound Sensor)
64     GPIO_PORTD_DEN_R |= (1 << 2); // Enable PD2
65 }
66
Build Output
Stellaris ICD1 L101 C:10 CAP. NUM SCRL OVR. R/W
```

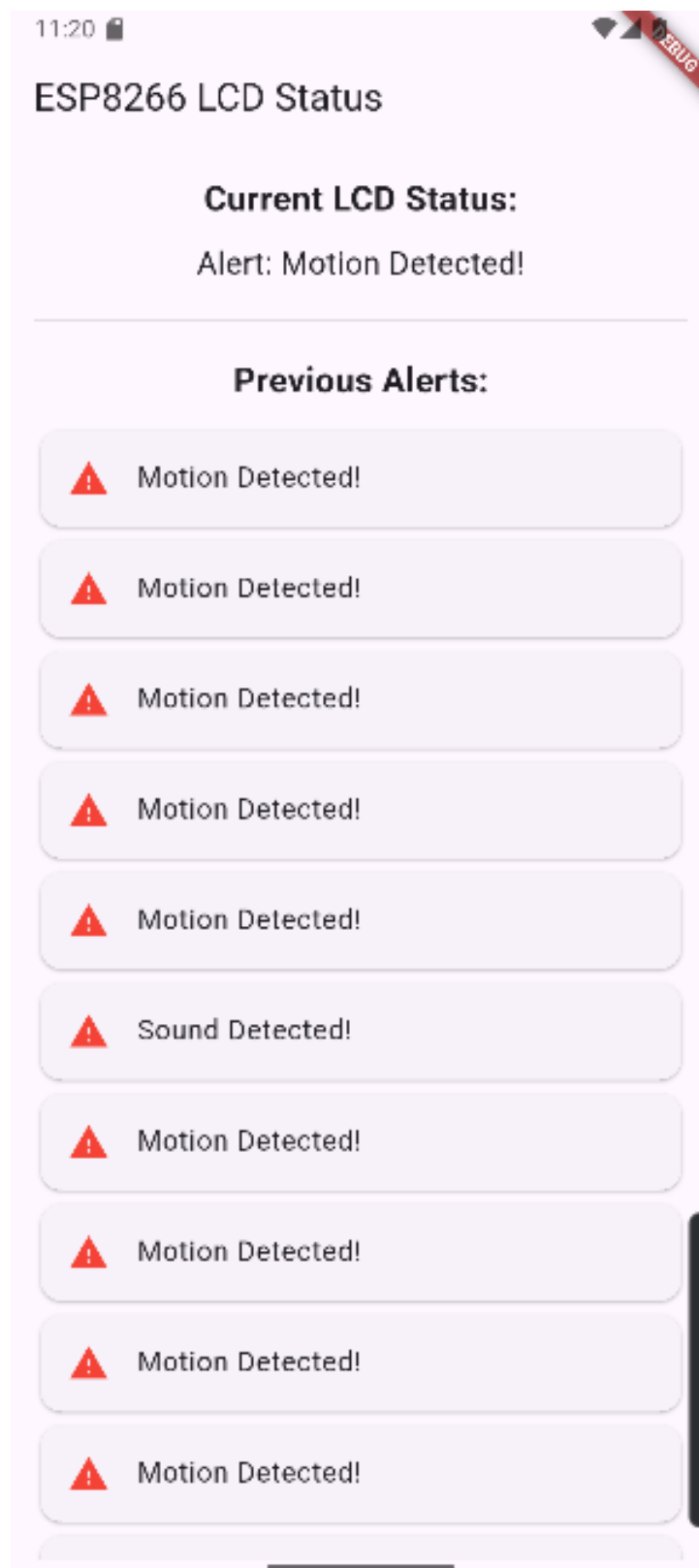


```
67 int main(void) {
68     UART3_Init();
69     PortF_Init();
70     PortE_Init();
71     PortD_Init();
72
73     while (1) {
74         // === Check SW1 Press ===
75         if ((GPIO_PORTF_DATA_R & (1 << 4)) == 0) {
76             UART3_OutString("Button Pressed!\r\n");
77             GPIO_PORTF_DATA_R |= (1 << 3); // Green LED ON
78             while ((GPIO_PORTF_DATA_R & (1 << 4)) == 0);
79         }
80
81         // === Check PIR Motion Sensor on PE1 ===
82         if (GPIO_PORTE_DATA_R & (1 << 1)) {
83             UART3_OutString("Motion Detected!\r\n");
84             GPIO_PORTF_DATA_R |= (1 << 3);
85             while (GPIO_PORTE_DATA_R & (1 << 1));
86         }
87
88         // === Check Smoke Sensor on PE2 ===
89         if (GPIO_PORTE_DATA_R & (1 << 2)) {
90             UART3_OutString("Smoke Detected!\r\n");
91             GPIO_PORTF_DATA_R |= (1 << 1); // Red LED ON
92             GPIO_PORTF_DATA_R |= (1 << 3); // Buzzer ON
93             while (GPIO_PORTE_DATA_R & (1 << 2));
94         }
95
96         // === Check Sound Sensor on PD2 ===
97         if (GPIO_PORTD_DATA_R & (1 << 2)) {
98             UART3_OutString("Sound Detected!\r\n");
99             GPIO_PORTF_DATA_R |= (1 << 3); // Green LED ON
100             while (GPIO_PORTD_DATA_R & (1 << 2));
101         }
102     }
103 }
```

VII. Moblie App (Flutter)

Send notifications about any detection with the time and type detection, also save previous detections

```
1 import 'package:flutter/material.dart';
2 import 'package:firebase_database/firebase_database.dart';
3
4 class LCDStatusPage extends StatefulWidget {
5   @override
6   _LCDStatusPageState createState() => _LCDStatusPageState();
7 }
8
9 class _LCDStatusPageState extends State<LCDStatusPage> {
10   String _lcdStatus = "Loading...";
11   List<String> _alerts = [];
12
13   final _databaseReference = FirebaseDatabase.instance.ref();
14
15   @override
16   void initState() {
17     super.initState();
18     _listenToLCDStatus();
19     _listenToAlerts();
20   }
21
22   void _listenToLCDStatus() {
23     _databaseReference.child('LCD_Status').onValue.listen((event) {
24       final data = event.snapshot.value;
25       if (data != null) {
26         setState(() {
27           _lcdStatus = data.toString();
28         });
29       }
30     });
31   }
32 }
```





VIII. Bonus feature

When the PIR sensor detects motion and uploads it to the firebase. The camera reads the firebase and capture an image and apply Machine learning model (YOLO) and deepseek. Loading the image to google drive and sends the image description to the mobile app.

```
1 import cv2
2 import firebase_admin
3 from firebase_admin import credentials, db
4 import datetime
5 import os
6 import time
7 from ultralytics import YOLO
8 from PIL import Image
9
10 # === Firebase Setup ===
11 cred = credentials.Certificate("C:/Users/user/PycharmProjects/pythonProject/embproject-15e7f-firebase-adminsdk-fb5vc-6776e8acb7.json")
12 firebase_admin.initialize_app(cred, {
13     'databaseURL': 'https://embproject-15e7f-default-rtdb.firebaseio.com/'
14 })
15
16 # === YOLOv8 Model Load ===
17 model = YOLO('yolov8n.pt') # Make sure this file is in your working directory or use full path
18
19 # === Save Paths ===
20 SAVE_DIR = "G:/My Drive/captured_images"
21 os.makedirs(SAVE_DIR, exist_ok=True)
22
23 def detect_with_yolo(image_path):
24     results = model.predict(source=image_path, save=False, conf=0.25)
25
26     detected_classes = set()
27     for result in results:
28         for box in result.boxes:
29             class_id = int(box.cls[0])
30             class_name = model.names[class_id]
31             detected_classes.add(class_name)
32
33     return detected_classes
```



```
File Edit View Navigate Code VCS Help yolo_trial - main.py
Project yolo_trial C:\Users\user\PycharmProjects\yolo_trial
main.py
yolov5npt
External Libraries
Scratches and Consoles

cv2.imwrite(filename, frame)
55
56 print(f"[INFO] Image saved: {filename}")
57 cap.release()
58 cv2.destroyAllWindows()
59 detect_with_yolo(filename)
60 # Push image path as string to Firebase under /Image_Links
61 img_ref = db.reference('/New image name')
62 img_ref.push(f"motion_{timestamp}.jpg (Last image)")
63
64 else:
65     print("Failed to capture image.")
66     cap.release()
67     cv2.destroyAllWindows()
68
69 def listen_firebase():
70     ref = db.reference('/Alerts')
71     processed_keys = set()
72
73     while True:
74         data = ref.get()
75         if data:
76             for key, value in data.items():
77                 if key not in processed_keys and "Motion Detected" in value:
78                     print(f"[ALERT] {value}")
79                     capture_image()
80                     processed_keys.add(key)
81             time.sleep(2)
82
83 if __name__ == "__main__":
84     listen_firebase()
85
detect_with_yolo()
```

console.firebase.google.com/u/0/project/embproject-15e7f/database/embproject-15e7f-default-rtdb/data

Firestore

Project Overview

Project shortcuts

- Realtime Database
- App Distributi...
- App Hosting
- Authentication
- Storage

What's new

- Genkit
- Vertex AI

Product categories

- Build
- Run
- Analytics
- AI

Spark No-cost (\$0/month) Upgrade

Realtime Database

Need help with Realtime Database? Ask Gemini

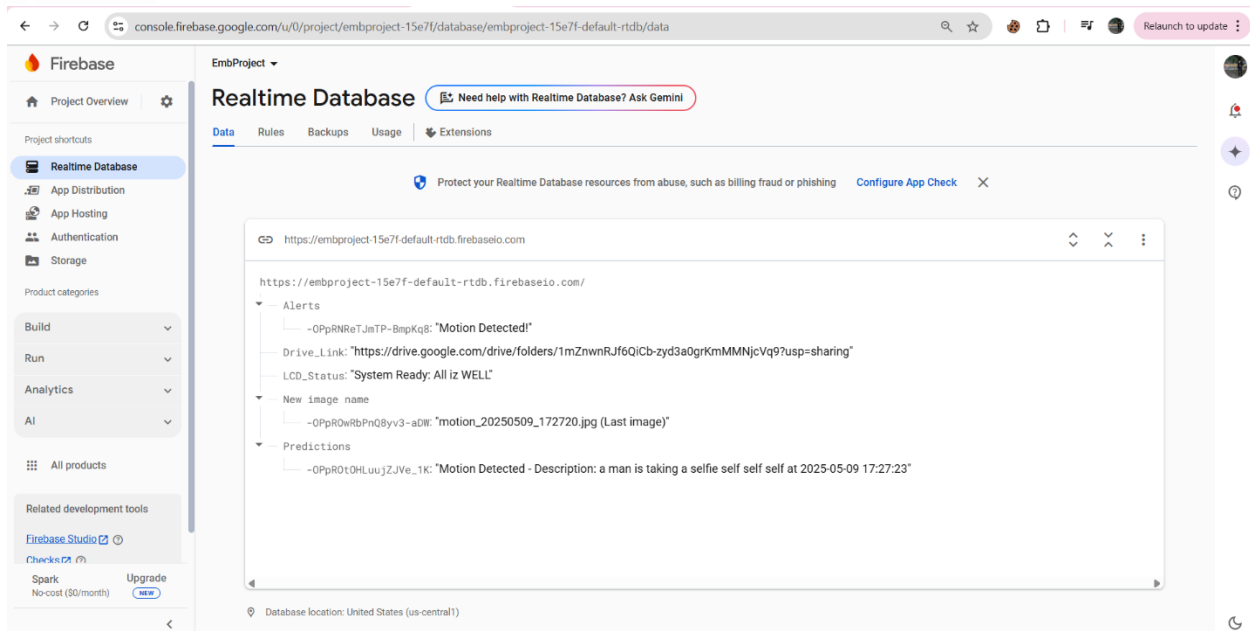
Data Rules Backups Usage Extensions

Protect your Realtime Database resources from abuse, such as billing fraud or phishing. [Configure App Check](#)

https://embproject-15e7f-default-rtdb.firebaseio.com

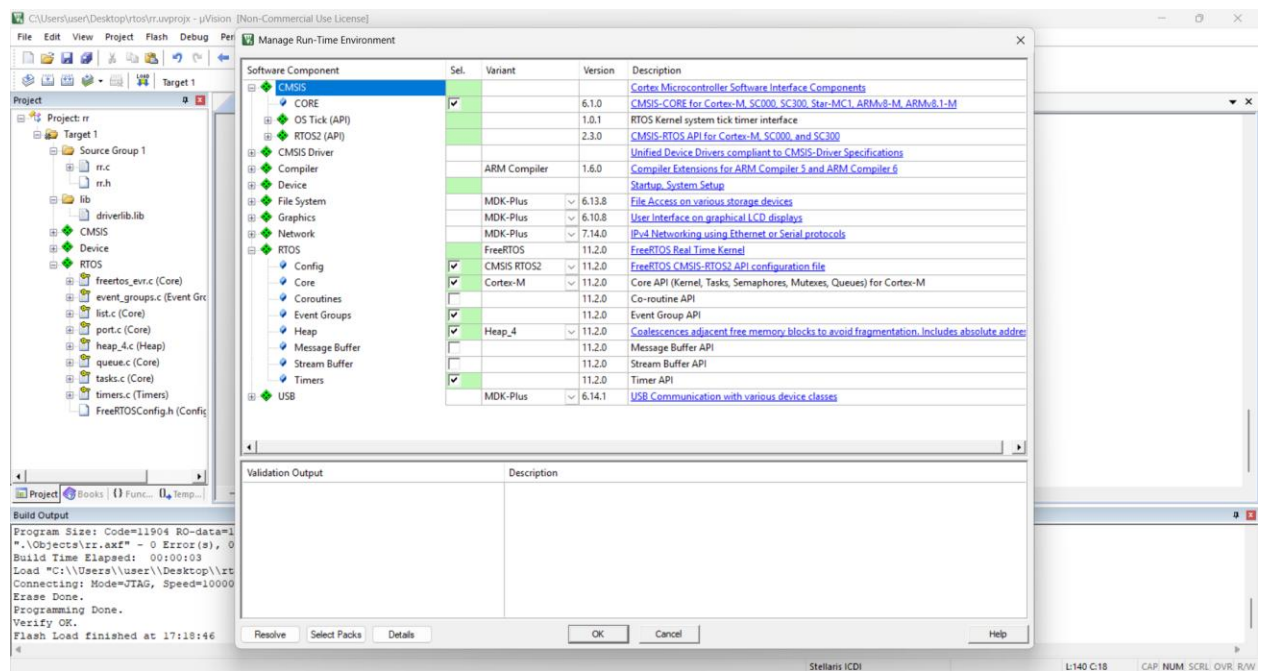
```
https://embproject-15e7f-default-rtdb.firebaseio.com/
├── Alerts
│   ├── Drive_Link: "https://drive.google.com/drive/folders/1mZnwnRJf6QicB-zyd3a0grKmmMnjcVq9?usp=sharing"
│   ├── Images_drive_link
│   ├── LCD_Status: "System Ready: All iz WELL"
│   ├── New image name
│   └── Predictions
```

Database location: United States (us-central1)



VIII. FreeRTOS:

1. FreeRTOS init:

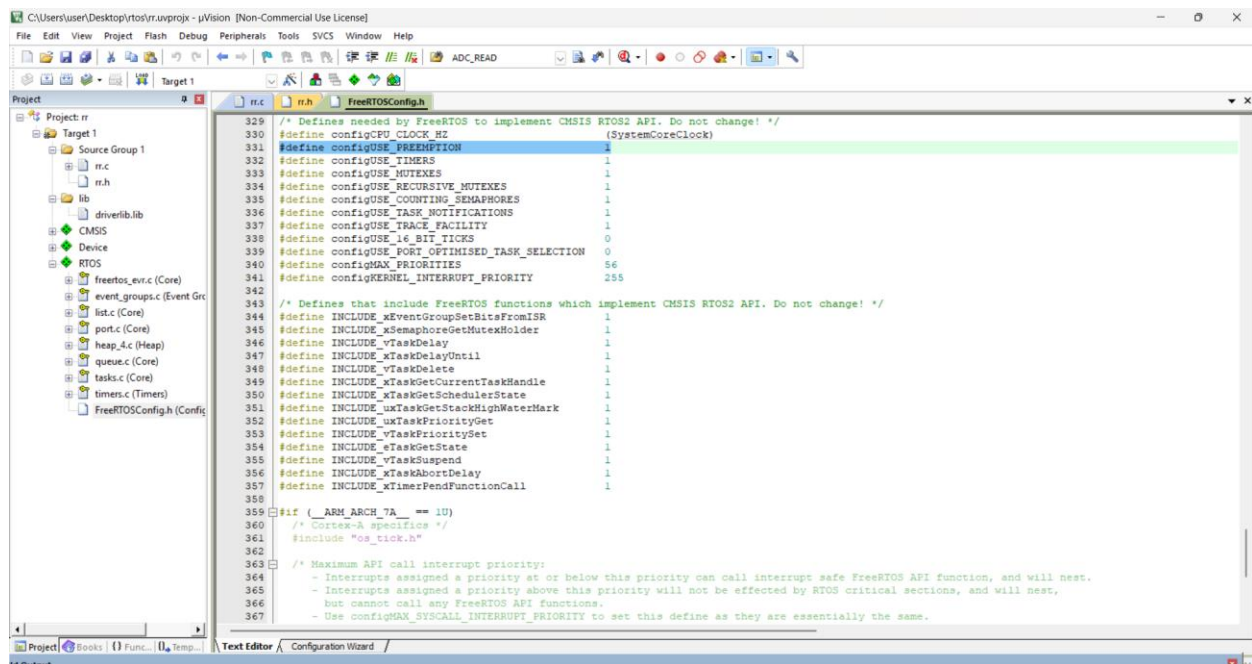




Main()

```
135 int main(void) {
136     SysCtlClockSet(SYSCTL_SYSDIV_4 | SYSCTL_USE_PLL | SYSCTL_OSC_MAIN | SYSCTL_XTAL_16MHZ);
137
138     UART3_Init();
139     PortF_Init();
140     PortE_Init();
141     PortD_Init();
142     void PortD_Init()
143     uartMutex = xSemaphoreCreateMutex();
144     ledMutex = xSemaphoreCreateMutex();
145
146     xTaskCreate(PIR_Task, "PIR", 256, NULL, 2, NULL);
147     xTaskCreate(Smoke_Task, "Smoke", 256, NULL, 3, NULL);
148     xTaskCreate(Sound_Task, "Sound", 256, NULL, 1, NULL);
149     xTaskCreate(Button_Task, "Button", 256, NULL, 1, NULL);
150
151     vTaskStartScheduler();
152
153     while (1);
154 }
```

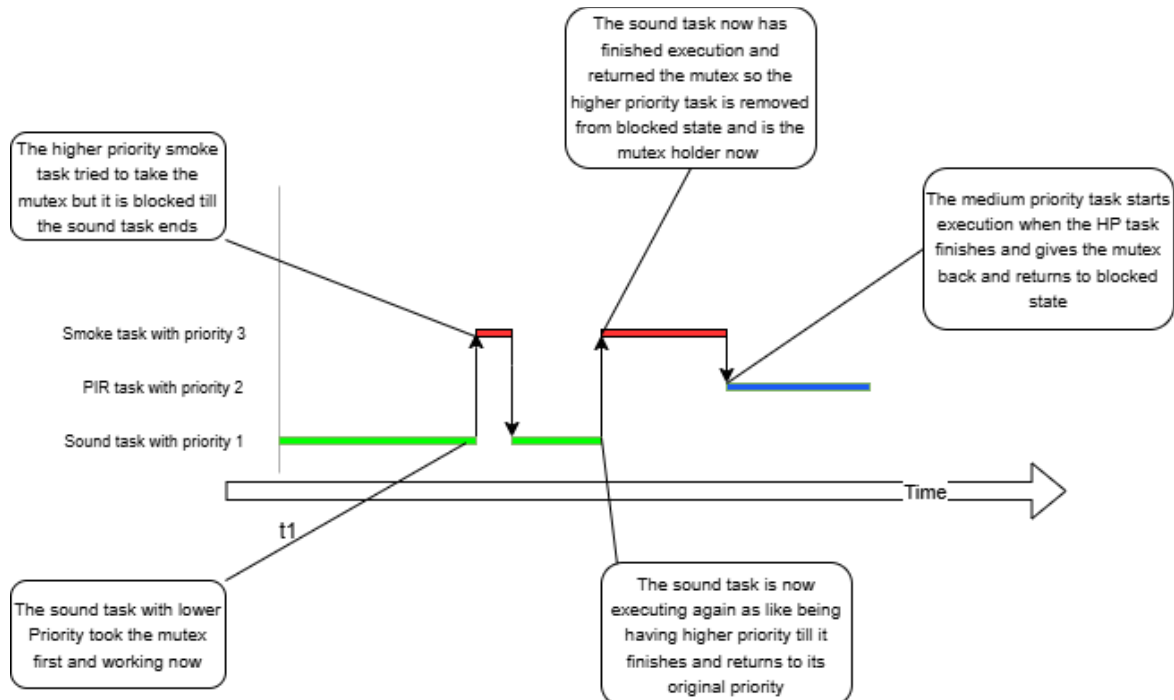
Using Preemptive scheduling



- Smoke Task (priority 3) preempts others.
- PIR Task (priority 2) is next in priority.
- Sound and Button Tasks (priority 1) share CPU time when higher-priority tasks are idle/delayed.
- UART access is **mutex-protected** to avoid overlapping UART writes.

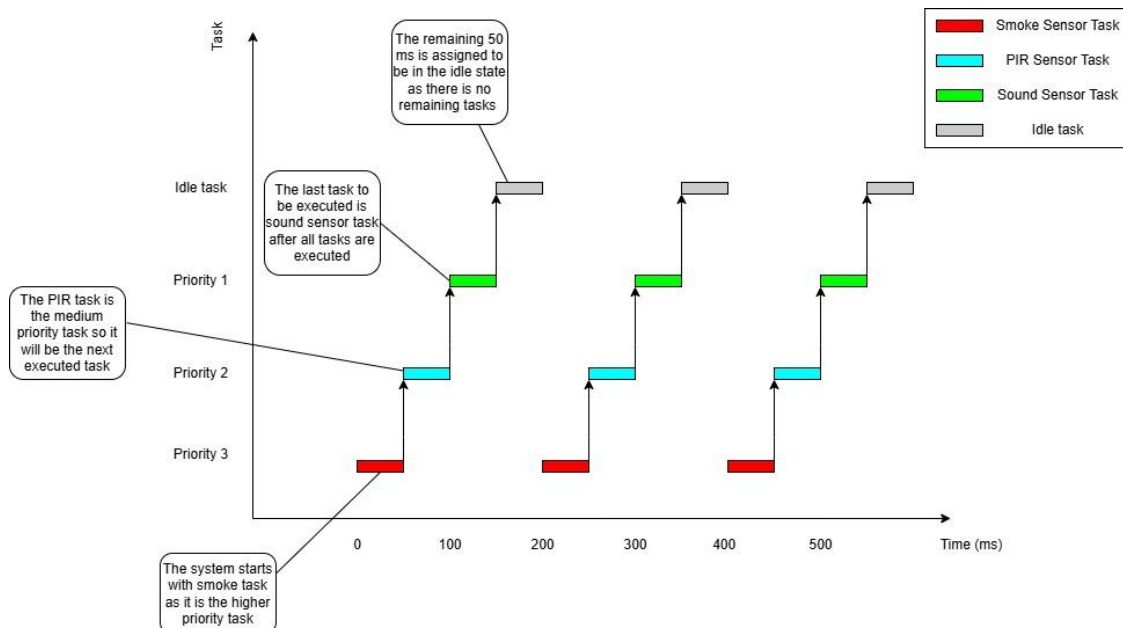


MUTEX: Only one task prints via UART3 at a time.
Prevents overlapping strings, garbled output, and bus conflict.



Timing Diagram:

Task	Schedule d Start Times (ms)	Executio n Time (ms)	Delay Between n Runs (ms)	vTaskDelay Equivalent
Smoke_Task	0, 200, 400	50	200	vTaskDelay(pdMS_TO_TICKS(200))
PIR_Task	50, 250, 450	50	200	vTaskDelay(pdMS_TO_TICKS(200))
Sound_Task	100, 300	50	200	vTaskDelay(pdMS_TO_TICKS(200))



- Smoke Task (priority 3) preempts others.
- PIR Task (priority 2) is next in priority.
- Sound and Button Tasks (priority 1) share CPU time when higher-priority tasks are idle/delayed.
- UART access is **mutex-protected** to avoid overlapping UART writes.

Mutex Use:

- Only **one** task prints via UART3 at a time.
- Prevents **overlapping strings**, garbled output, and bus conflict.



Dr. Ahmed Sayed