Security System Project

Stage 3 (Embedded Systems – CIE 408)

Table of Contents:

Belal Gamal -202101237

Abd Alrhman Ahmed - 202101112

Marwan Ahmed - 202101214

Yousef Khaled – 202100191

Mohamed Hosam - 202100975

I. Introduction and Literature Survey

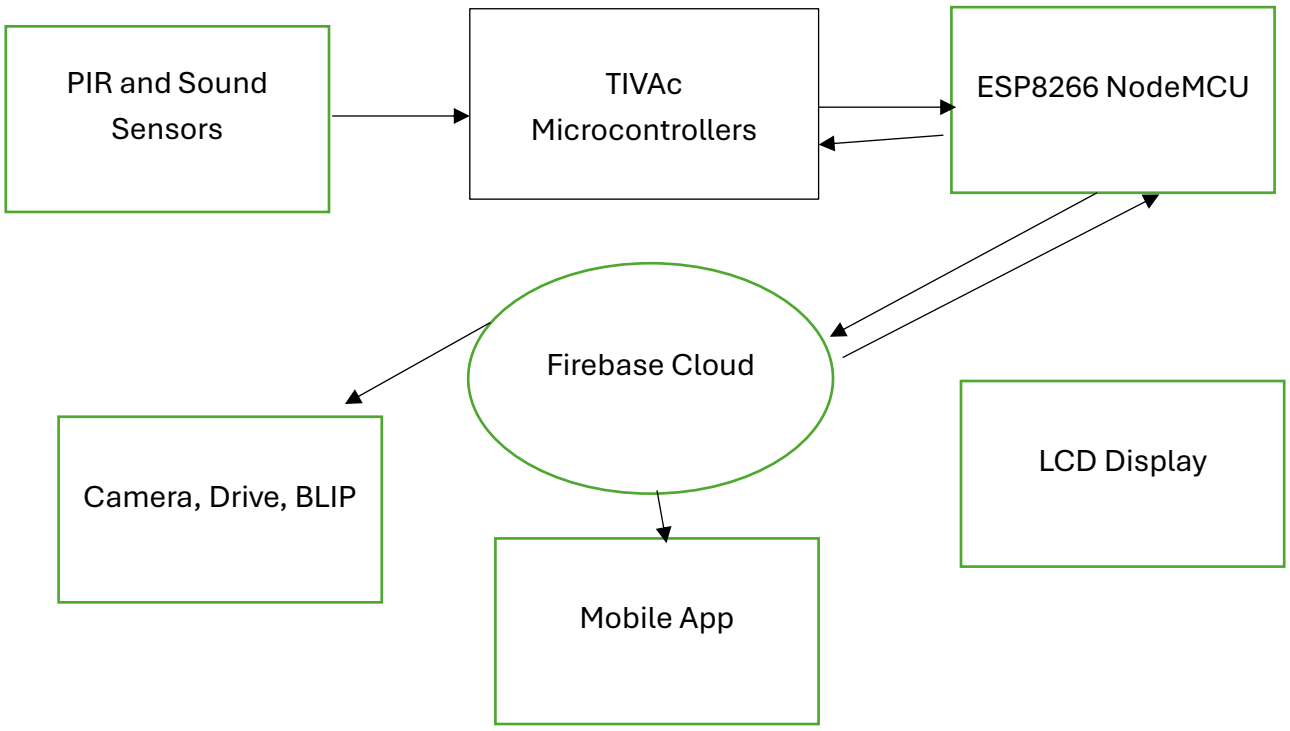
Security systems protect homes, companies, and public spaces by detecting unwanted entrance or suspicious activity. The development of embedded systems and the Internet of Things has led to the integration of real-time monitoring, cloud-based data storage, and automatic warnings into contemporary security 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

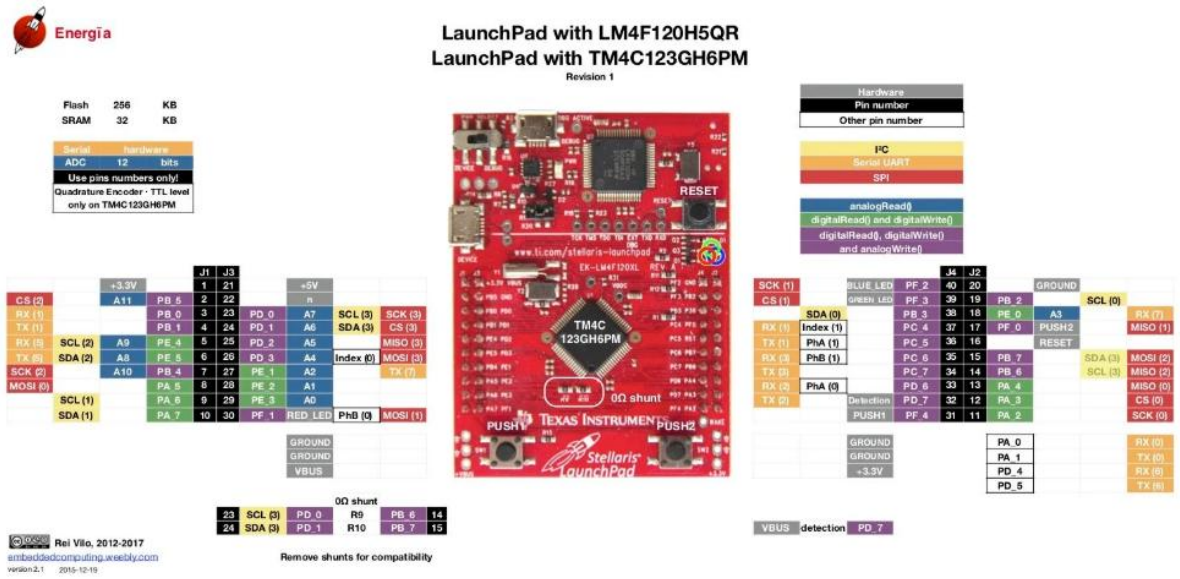
- **5V Power Adapter (1A-2A output) & Voltage Regulators** → Ensures stable power supply for the microcontroller and sensors.

Software Requirements

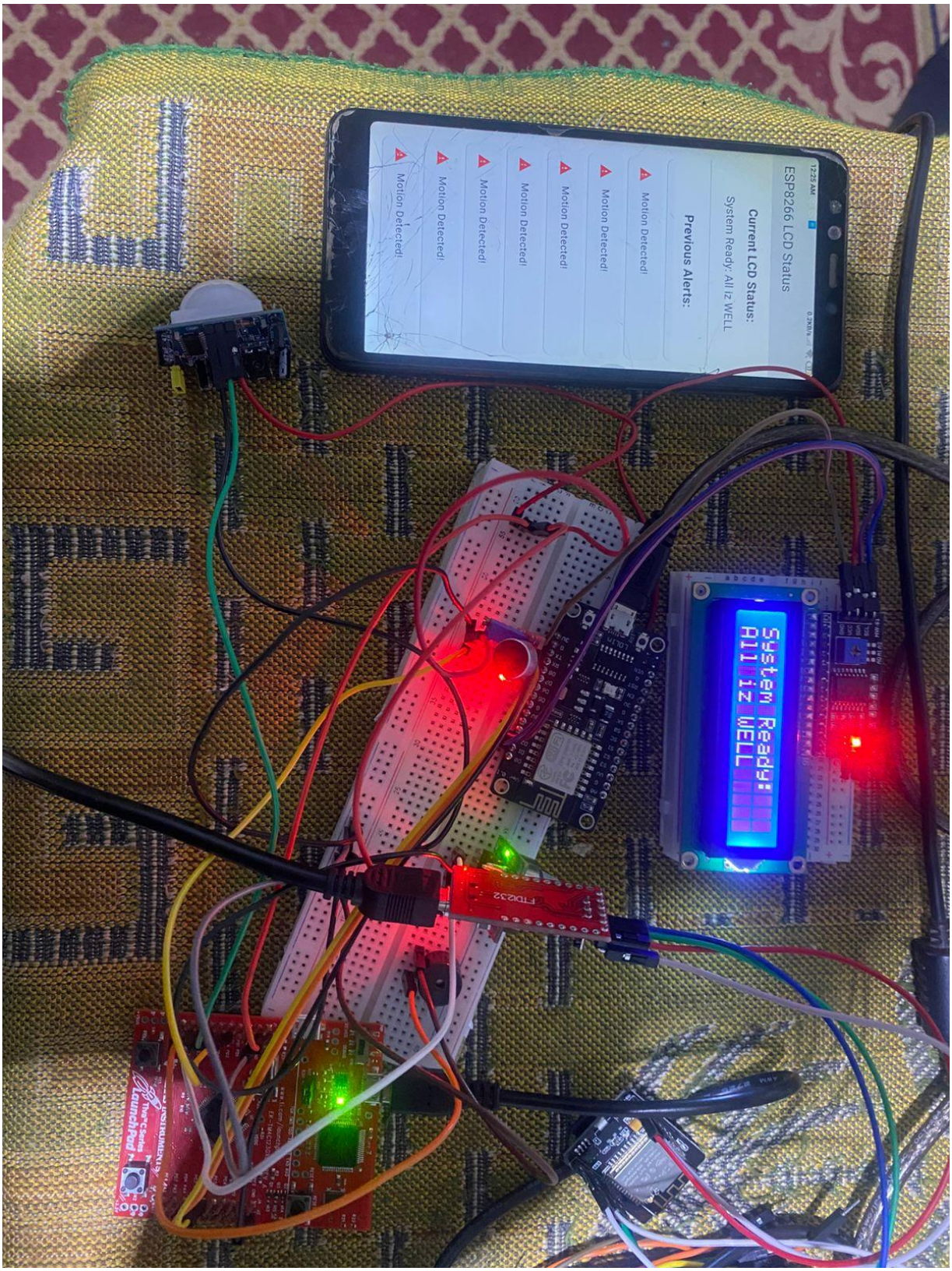
- **FreeRTOS** (for task management).
- **Firebase Setup** (cloud integration for alerts & monitoring).
- **Mobile App/Web Dashboard** (for remote access).

IV. Hardware

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.

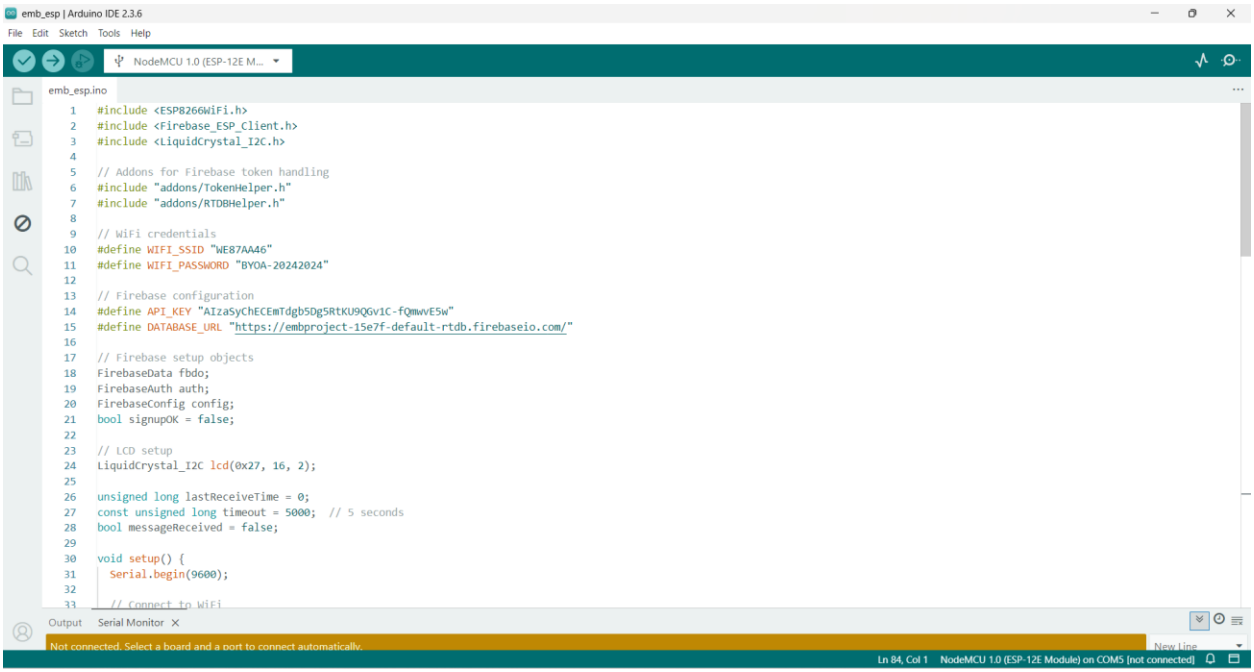


UART Module	Rx Pin	Tx Pin
UART0	PA0	PA1
UART1	PC4	PC5
UART2	PD6	PD7
UART3	PC6	PC7
UART4	PC4	PC5
UART5	PE4	PE5
UART6	PD4	PD5
UART7	PE0	PE1


12:23 AM
ESP8266 LCD Status
Current LCD Status:
System Ready, All iz WELL
Previous Alerts:
Motion Detected!
Motion Detected!
Motion Detected!
Motion Detected!
Motion Detected!
Motion Detected!
System Ready!
All iz WELL
FTDI232

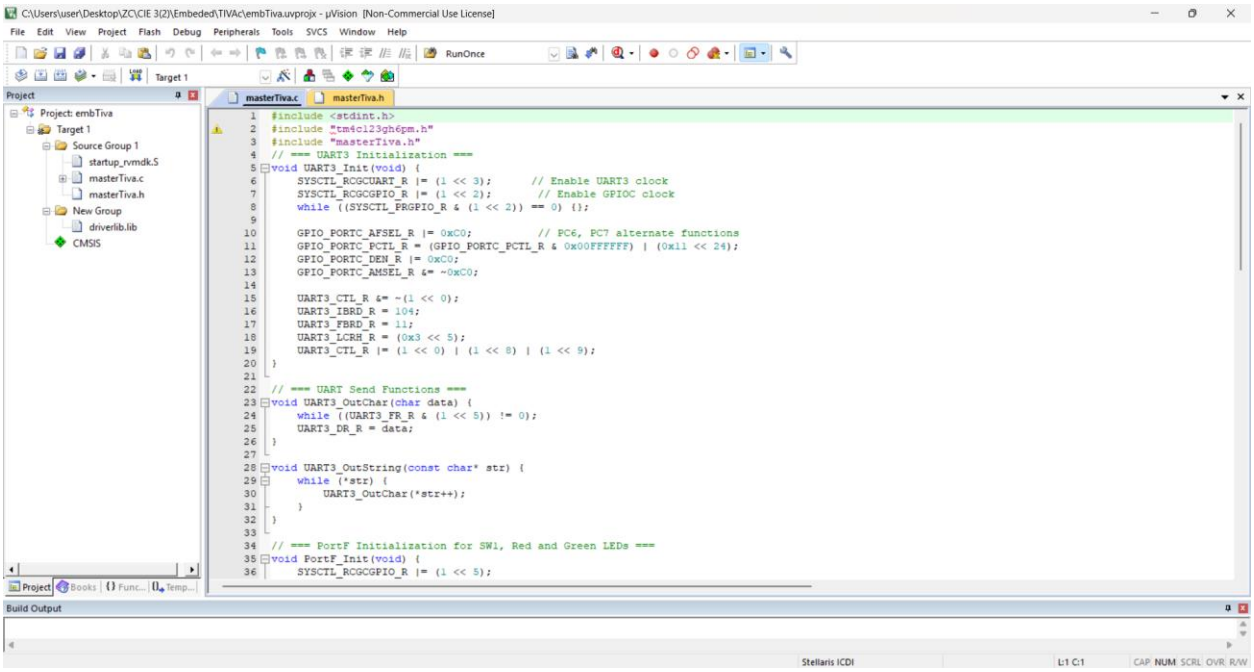
V. Code

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

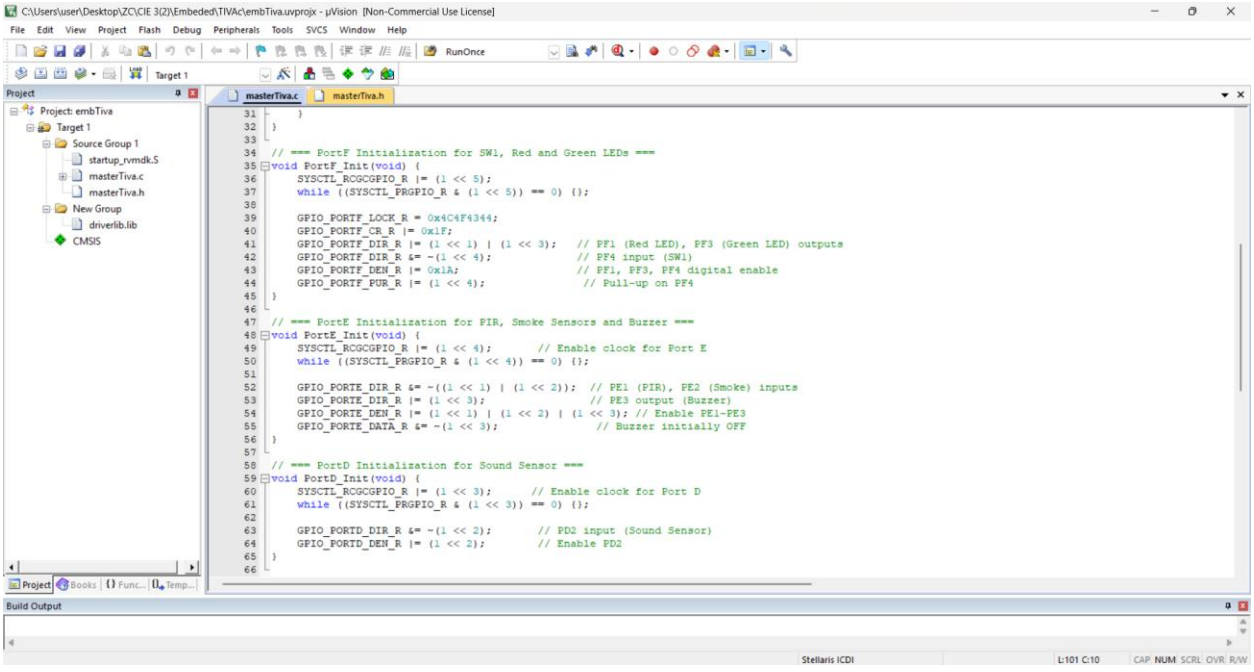


TivaC code:

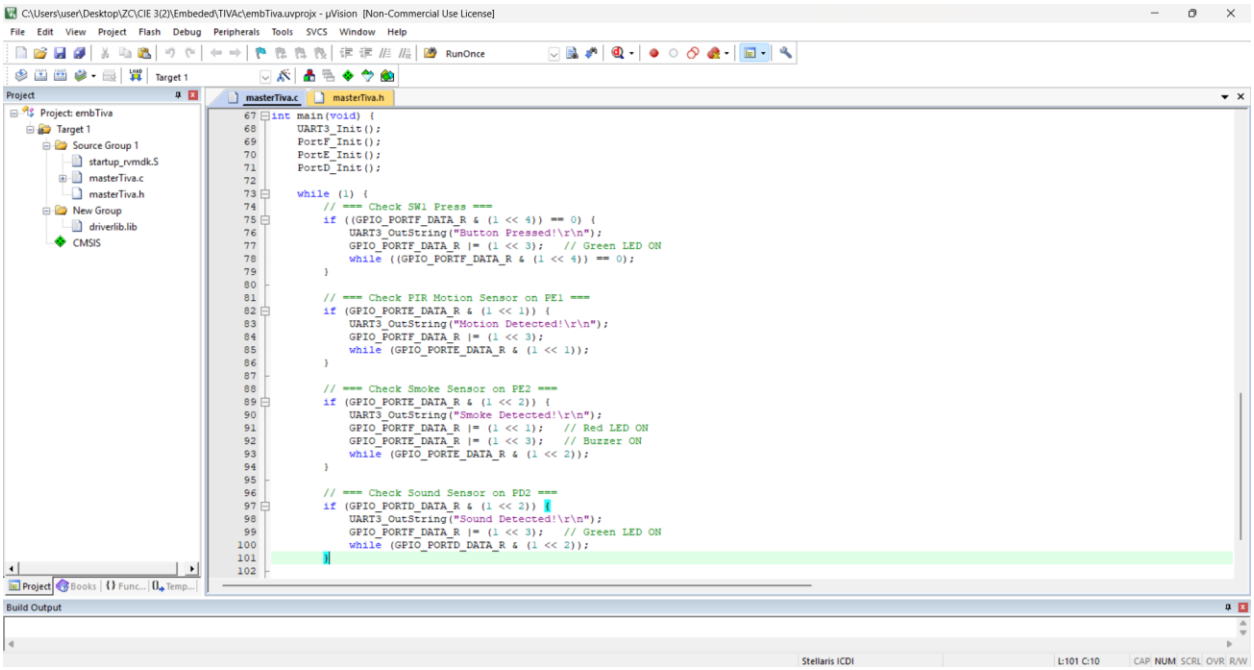
Read from the three sensors and send data to esp through UART3



```
#include <stdint.h>
#include "tm4c123gh6pm.h"
#include "masterTiva.h"
// === UART3 Initialization ===
void UART3_Init(void) {
    SYSCTL_RCGCUART_R |= (1 << 3);      // Enable UART3 clock
    SYSCTL_RCGCGPIO_R |= (1 << 2);       // Enable GPIOC clock
    while ((SYSCTL_PRGPIO_R & (1 << 2)) == 0) {};

    GPIO_PORTC_AFSEL_R |= 0xC0;          // PC6, PC7 alternate functions
    GPIO_PORTC_PCTL_R = (GPIO_PORTC_PCTL_R & 0x00FFFFFF) | (0x11 << 24);
    GPIO_PORTC_DEN_R |= 0xC0;
    GPIO_PORTC_AMSEL_R &= ~0xC0;

    UART3_CTL_R &= ~(1 << 0);
    UART3_IBRD_R = 104;
    UART3_FBRD_R = 11;
    UART3_LCRH_R = (0x3 << 5);
    UART3_CTL_R |= (1 << 0) | (1 << 8) | (1 << 9);
}

// === UART Send Functions ===
void UART3_OutChar(char data) {
    while ((UART3_FR_R & (1 << 5)) != 0);
    UART3_DR_R = data;
}

void UART3_OutString(const char* str) {
    while (*str) {
        UART3_OutChar(*str++);
    }
}

// === PortF Initialization for SW1, Red and Green LEDs ===
void PortF_Init(void) {
    SYSCTL_RCGCGPIO_R |= (1 << 5);
```

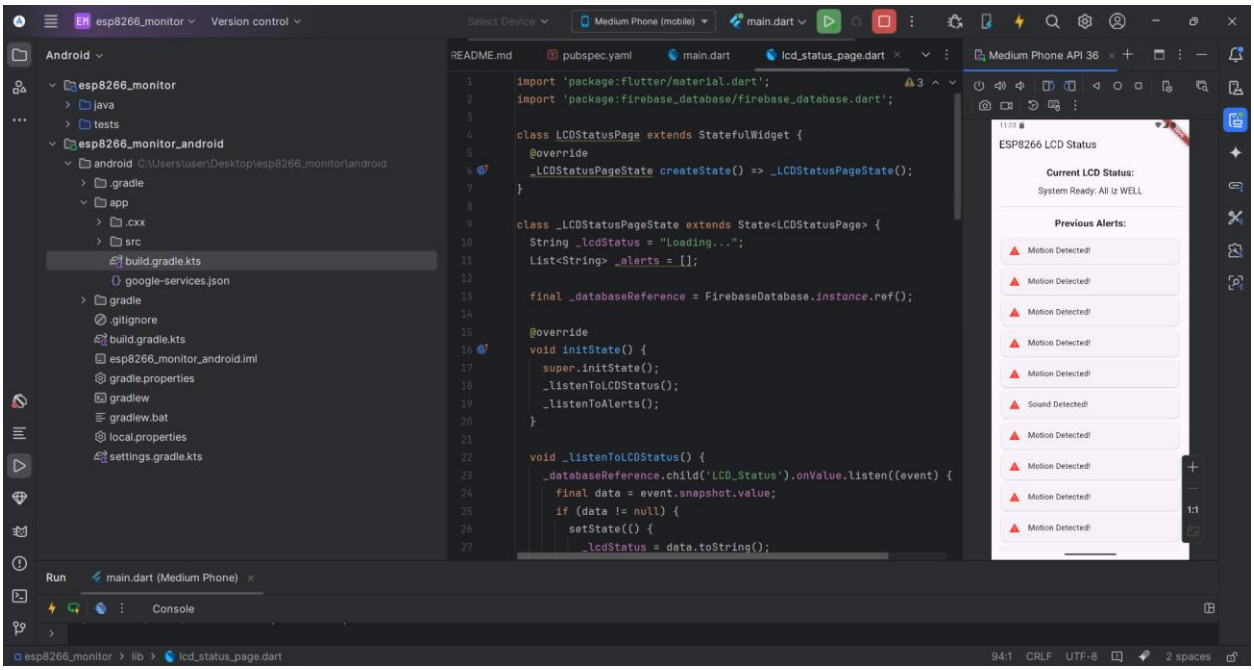


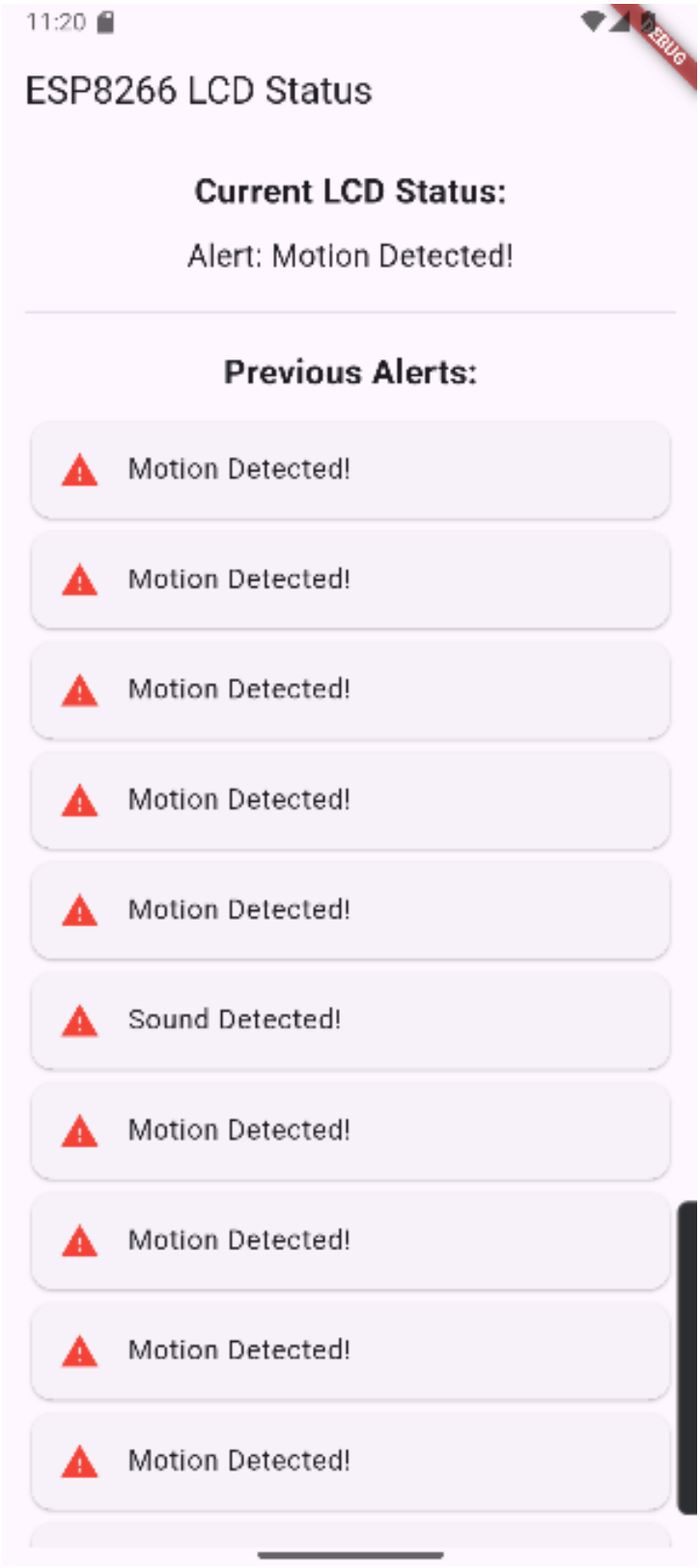
```
    }
}

// === PortF Initialization for SW1, Red and Green LEDs ===
void PortF_Init(void) {
    SYSCTL_RCGCGPIO_R |= (1 << 5);
    while ((SYSCTL_PRGPIO_R & (1 << 5)) == 0) {};

    GPIO_PORTF_LOCK_R = 0x4C4F4344;
    GPIO_PORTF_CR_R |= 0x1F;
    GPIO_PORTF_DIR_R |= (1 << 1) | (1 << 3);   // PF1 (Red LED), PF3 (Green LED) outputs
    GPIO_PORTF_DIR_R &= ~(1 << 4);             // PF4 input (SW1)
    GPIO_PORTF_DEN_R |= 0x1A;                  // PF1, PF3, PF4 digital enable
    GPIO_PORTF_PUR_R |= (1 << 4);               // Pull-up on PF4
}

// === PortE Initialization for PIR, Smoke Sensors and Buzzer ===
void PortE_Init(void) {
    SYSCTL_RCGCGPIO_R |= (1 << 4);       // Enable clock for Port E
    while ((SYSCTL_PRGPIO_R & (1 << 4)) == 0) {};

    GPIO_PORTE_DIR_R &= ~((1 << 1) | (1 << 2));  // PE1 (PIR), PE2 (Smoke) inputs
    GPIO_PORTE_DIR_R |= (1 << 3);                // PE3 output (Buzzer)
    GPIO_PORTE_DEN_R |= (1 << 1) | (1 << 2) | (1 << 3); // Enable PE1-PE3
    GPIO_PORTE_DATA_R &= ~(1 << 3);              // Buzzer initially OFF
}

// === PortD Initialization for Sound Sensor ===
void PortD_Init(void) {
    SYSCTL_RCGCGPIO_R |= (1 << 3);       // Enable clock for Port D
    while ((SYSCTL_PRGPIO_R & (1 << 3)) == 0) {};

    GPIO_PORTD_DIR_R &= ~(1 << 2);       // PD2 input (Sound Sensor)
    GPIO_PORTD_DEN_R |= (1 << 2);        // Enable PD2
}
```



VII. Moblie App (Flutter)

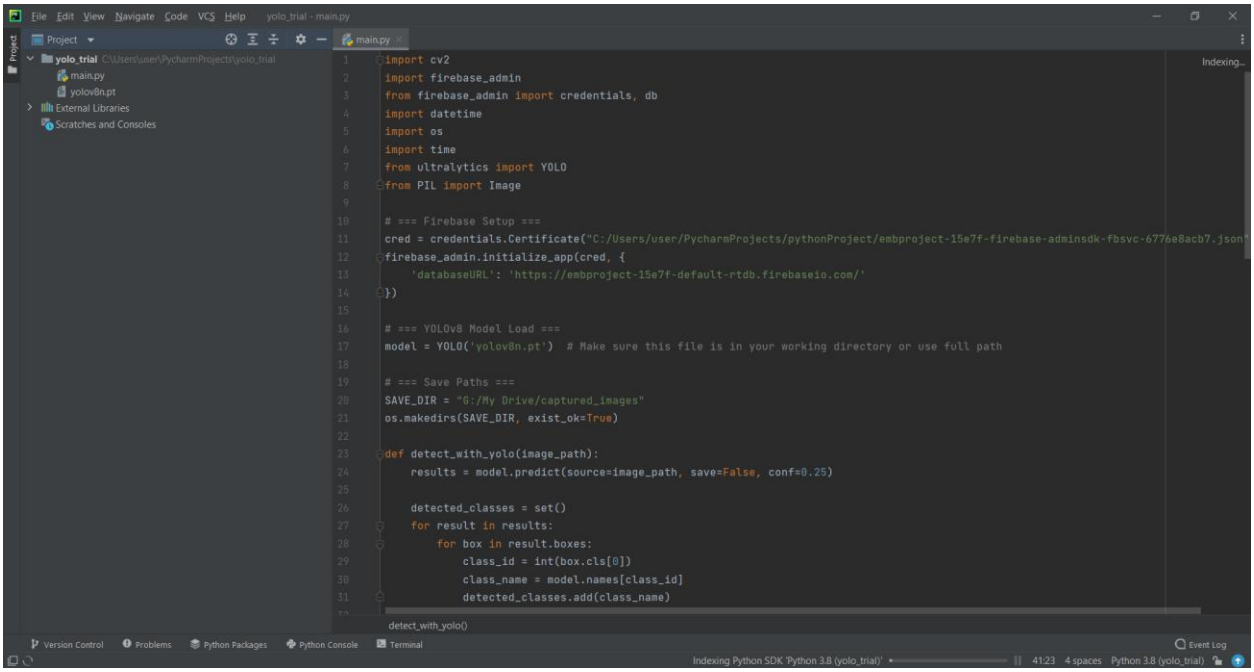
Send notifications about any detection with the time and type detection, also save previous detections

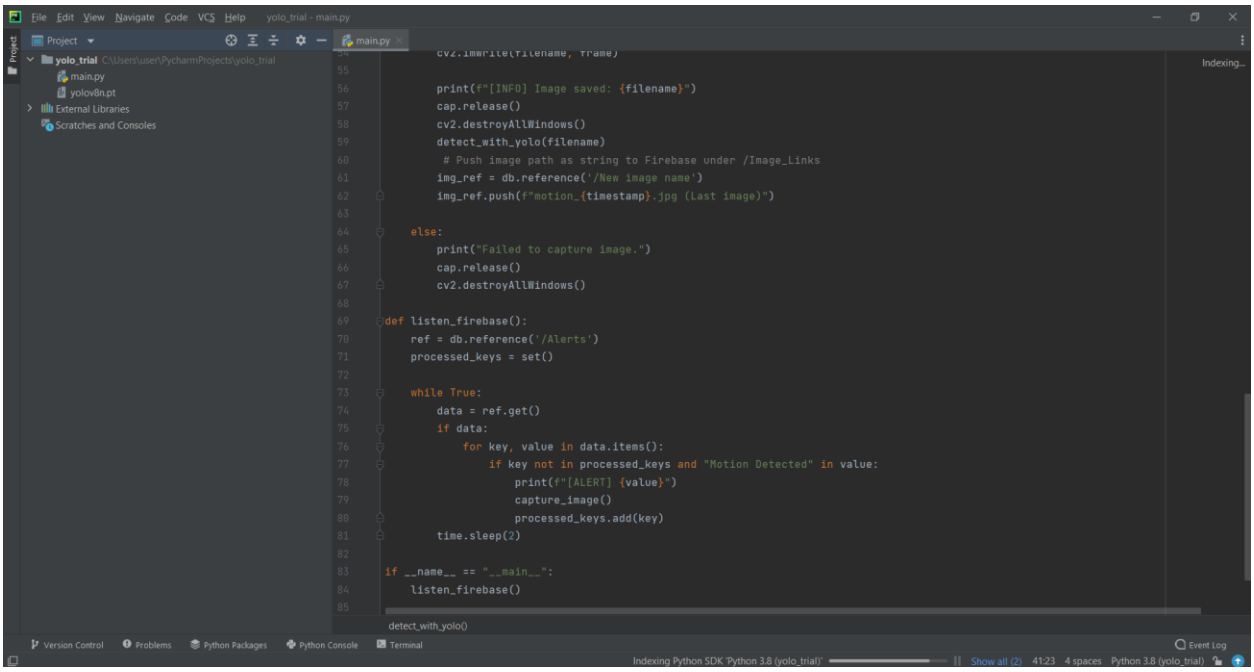


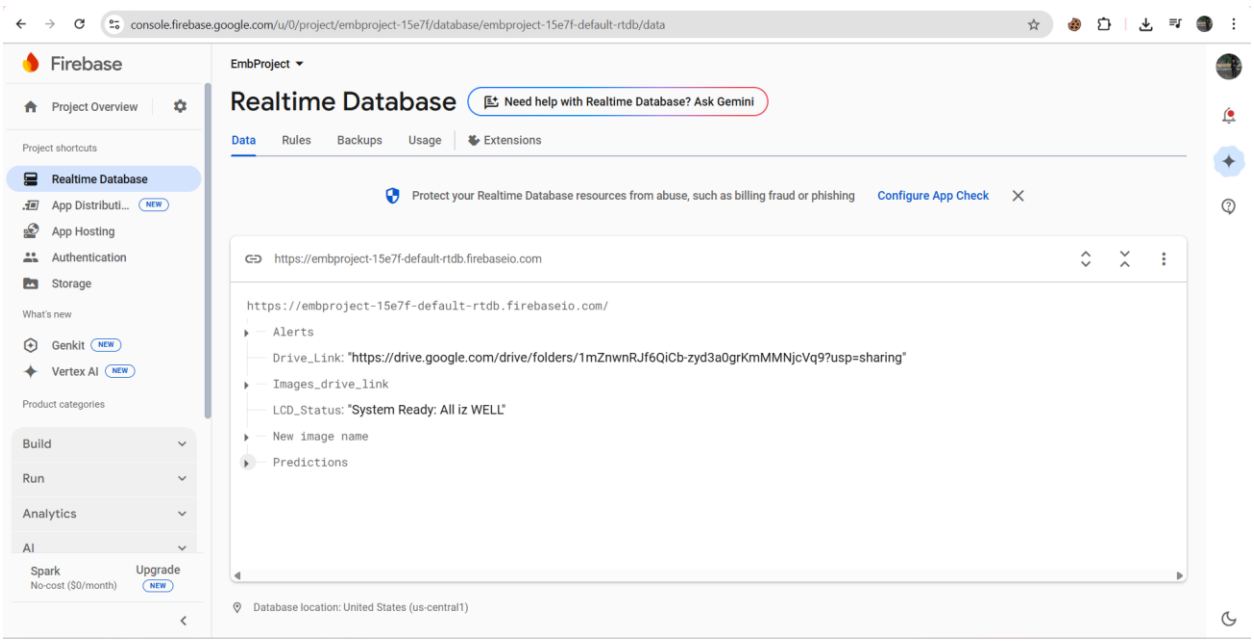

11:20
DEBUG
ESP8266 LCD Status
Current LCD Status:
Alert: Motion Detected!
Previous Alerts:
Motion Detected!
Motion Detected!
Motion Detected!
Motion Detected!
Motion Detected!
Sound Detected!
Motion Detected!
Motion Detected!
Motion Detected!
Motion Detected!

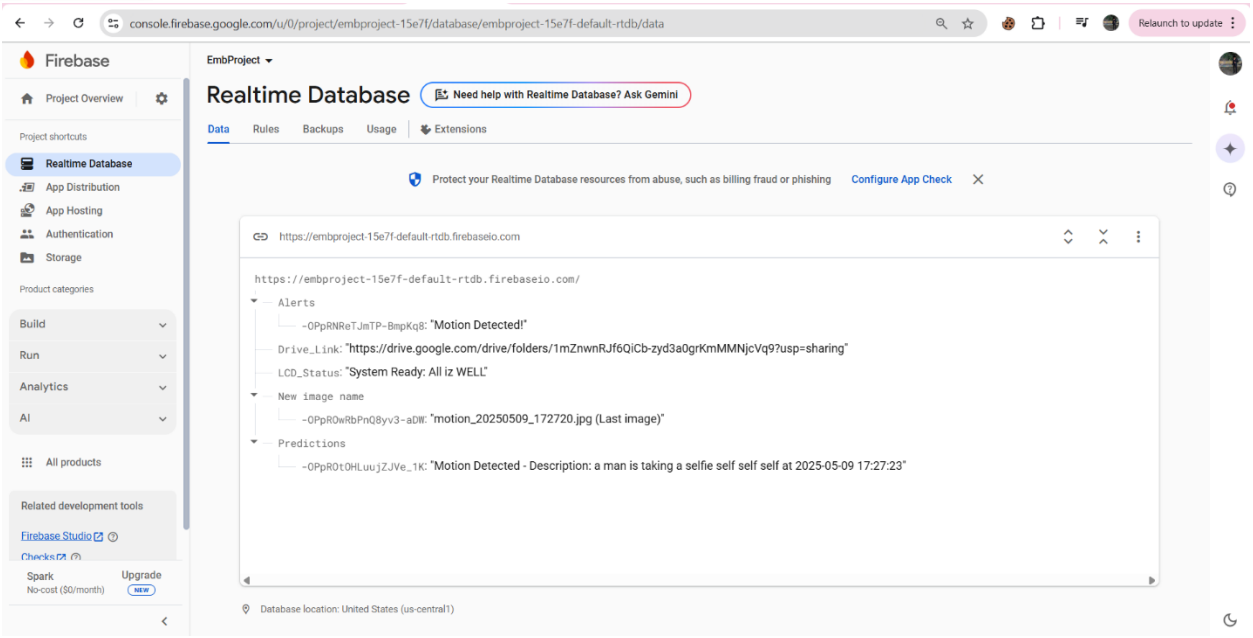
VIII. Bonus feature

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.



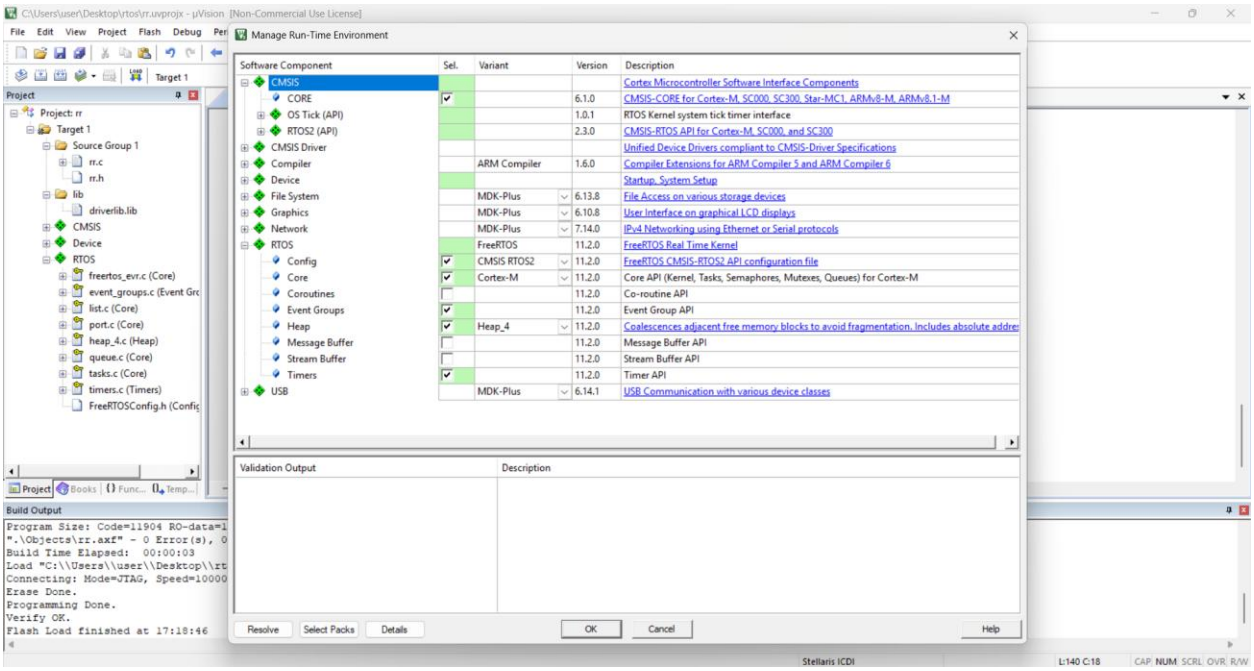

File Edit View Navigate Code VCS Help yolo_trial - main.py
Project
yolo_trial C:\Users\user\PycharmProjects\yolo_trial
main.py
yolov8n.pt
External Libraries
Scratches and Consoles
main.py
print(f"[INFO] Image saved: {filename}")
cap.release()
cv2.destroyAllWindows()
detect_with_yolo(filename)
# Push image path as string to Firebase under /Image_Links
img_ref = db.reference('/New image name')
img_ref.push(f"motion_{timestamp}.jpg (Last image)")
else:
print("Failed to capture image.")
cap.release()
cv2.destroyAllWindows()
def listen_firebase():
ref = db.reference('/Alerts')
processed_keys = set()
while True:
data = ref.get()
if data:
for key, value in data.items():
if key not in processed_keys and "Motion Detected" in value:
print(f"[ALERT] {value}")
capture_image()
processed_keys.add(key)
time.sleep(2)
if __name__ == "__main__":
listen_firebase()
Indexing...
detect_with_yolo()
Version Control
Problems
Python Packages
Python Console
Terminal
Event Log
Indexing Python SDK 'Python 3.8 (yolo_trial)'
Show all (2)
41:23
4 spaces
Python 3.8 (yolo_trial)


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

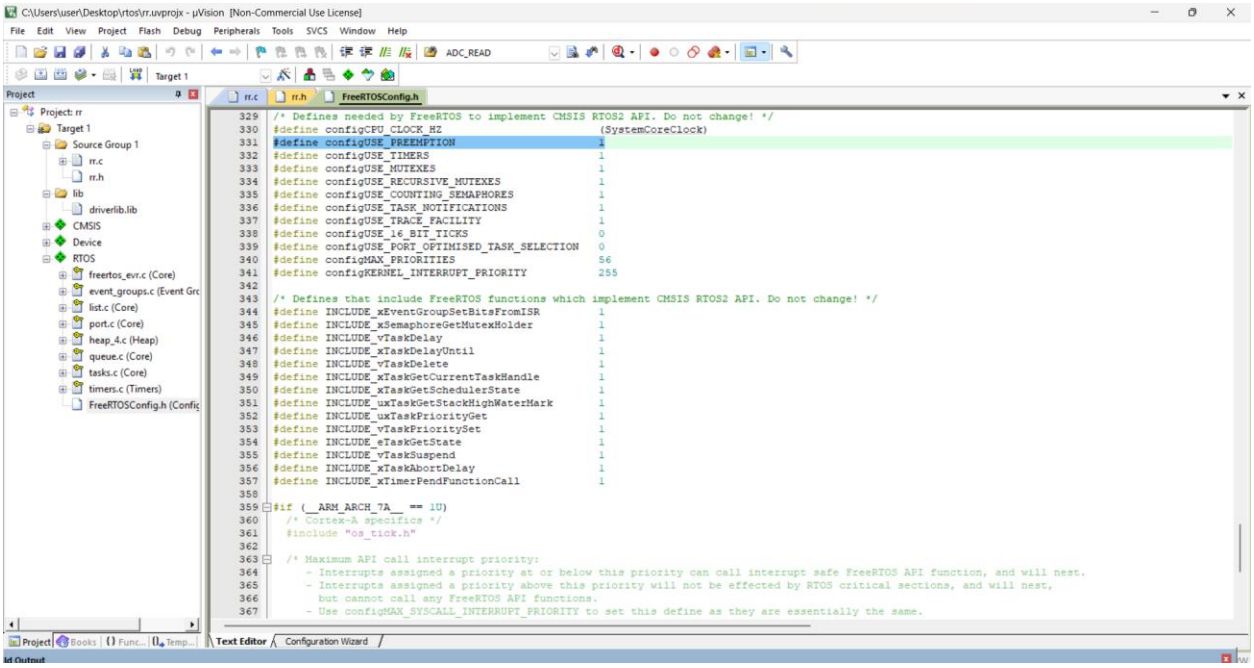
1. FreeRTOS init:



Main()

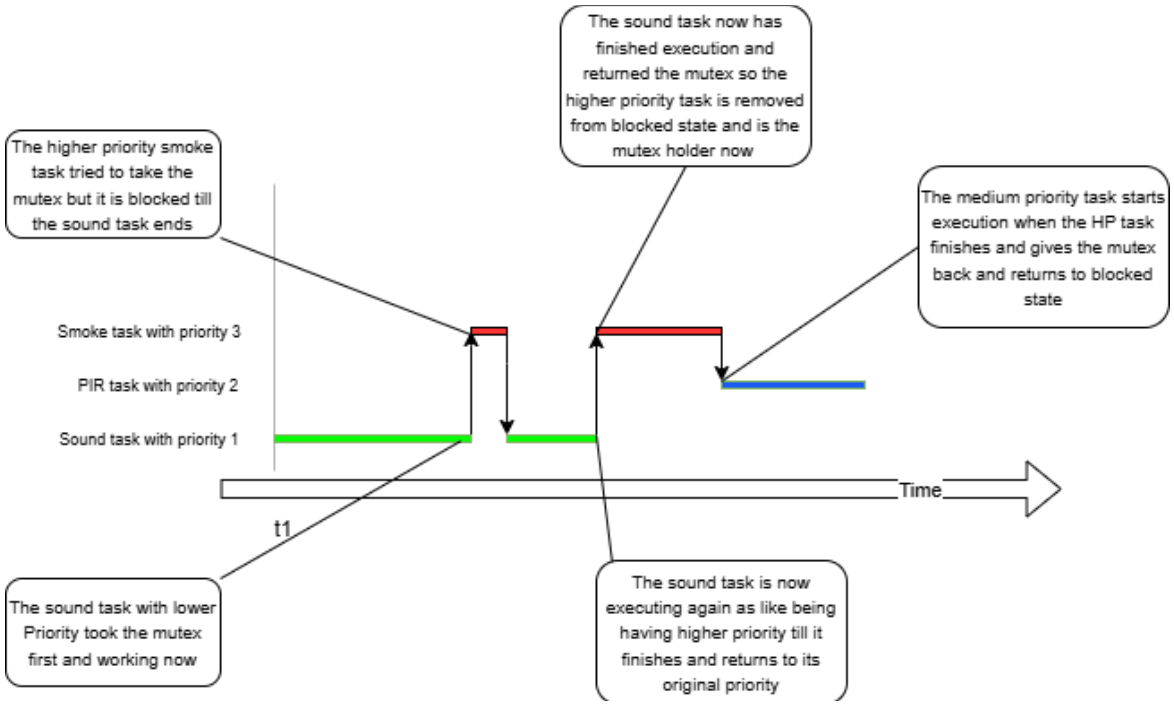
```
int main(void) {
    SysCtlClockSet(SYSCTL_SYSDIV_4 | SYSCTL_USE_PLL | SYSCTL_OSC_MAIN | SYSCTL_XTAL_16MHZ);

    UART3_Init();
    PortF_Init();
    PortE_Init();
    PortD_Init();
    void PortD_Init()
    uartMutex = xSemaphoreCreateMutex();
    ledMutex = xSemaphoreCreateMutex();

    xTaskCreate(PIR_Task, "PIR", 256, NULL, 2, NULL);
    xTaskCreate(Smoke_Task, "Smoke", 256, NULL, 3, NULL);
    xTaskCreate(Sound_Task, "Sound", 256, NULL, 1, NULL);
    xTaskCreate(Button_Task, "Button", 256, NULL, 1, NULL);

    vTaskStartScheduler();

    while (1);
}
```

Using Preemptive scheduling



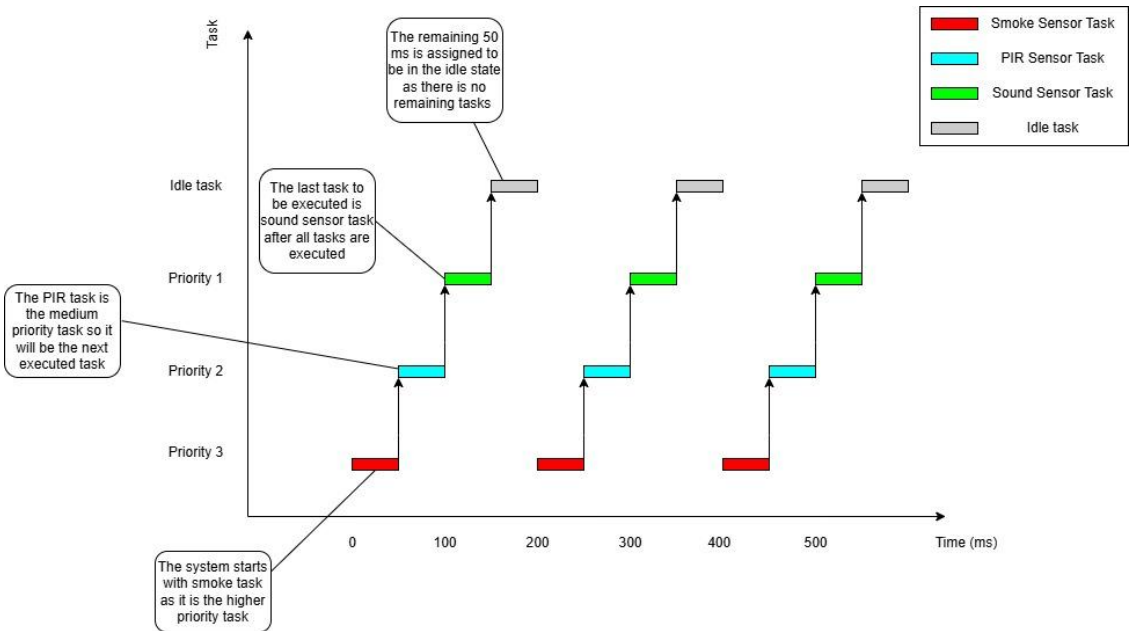
- Smoke Task (priority 3) preempts others.
- PIR Task (priority 2) is next in priority.
- Sound and Button Tasks (priority 1) share CPU time when higher-priority tasks are idle/delayed.
- UART access is **mutex-protected** to avoid overlapping UART writes.

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



Timing Diagram:

Task	Scheduled Start Times (ms)	Execution Time (ms)	Delay Between 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.