

PROJECT REPORT
ON
RTOS based Fire & Gas safety critical system for commercial use

Carried Out at



**CENTRE FOR DEVELOPMENT OF ADVANCED COMPUTING
ELECTRONIC CITY, BANGALORE.**

UNDER THE SUPERVISION OF

**Nikhil Yadav
Project Engineer
C-DAC Bangalore**

Submitted By
Varun S Patil(230950130035)
Abhishek N G(230950130003)
Aman Chaurasia(230950130006)
Saksham Tak(230950130028)
Naradham Dharanija(230950130019)

**PG DIPLOMA IN EMBEDDED SYSTEM AND DESIGN
C-DAC, BANGALORE**

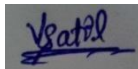
Candidate's Declaration

We hereby certify that the work being presented in the report entitled **RTOS based Fire & Gas safety critical system for commercial use** in the partial fulfillment of the requirements for the award of **Post-Graduate Diploma** and submitted in the department of **PG-DESD** of the C-DAC Bangalore, is an authentic record of our work carried out during the period 1st June 2018– 13th July 2018 under the supervision of **Nikhil Yadav**, Project Engineer at C-DAC Bangalore.

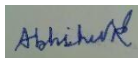
The matter presented in the report has not been submitted by us for the award of any degree of this or any other Institute/University.

(Name and Signature of Candidate)

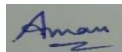
Varun S Patil(230950130035)



Abhishek N G(230950130003)



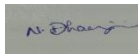
Aman Chaurasia(230950130006)



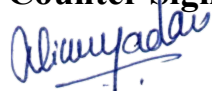
Saksham Tak(230950130028)



Naradham Dharanija(230950130019)



Counter Signed by



Nikhil Yadav

CERTIFICATE

This is to certify that this is a bonafide record of project work on the **RTOS based Fire & Gas safety critical system for commercial use** done by Varun S Patil(30035), Abhishek N G(30003), Saksham Tak(30028), Aman Chaurasia(30006) and Naradham Dharanija(30019) under the guidance of Mr.Nikhil Yadav in partial fulfilment of the requirement of a PostGraduate Diploma in Embedded System Design at C-DAC Bangalore for the academic session of September 2023.



Mr.Nikhil Yadav

CDAC, Electronic City,
Bangalore-560100, India

ACKNOWLEDGMENT

We take this opportunity to express my gratitude to all those people who have been directly and indirectly with us during the competition of this project.

We pay thanks to Nikhil Yadav who has given guidance and a light to us during this major project. His versatile knowledge about “**RTOS based Fire & Gas safety critical system for commercial use**” has eased us in the critical times during the span of this Final Project.

We acknowledge our debt to those who contributed significantly to one or more steps. I take full responsibility for any remaining sins of omission and commission.

Student's Name

Varun S Patil(230950130035)

Abhishek N G(230950130003)

Aman Chaurasia(230950130006)

Saksham Tak(230950130028)

Naradham Dharaniya(230950130019)

ABSTRACT

Did you know that fire and gas hazards pose significant risks in commercial settings? Every year, numerous accidents occur due to these hazards, leading to loss of life and property. To address these challenges, the "RTOS based Fire & Gas Safety Critical System for Commercial Use" project aims to develop a sophisticated safety system that can detect and respond to potential threats in real time.

This project utilizes advanced technologies, including the INDUS Development Board, STM32 CubeIDE, and FreeRTOS, to create a robust safety solution. By integrating various sensors such as LPG, flame, and CO sensors, the system can accurately detect gas leaks and fire hazards. Additionally, the system includes a buzzer for immediate alerting in case of emergencies.

The project's primary objective is to enhance safety measures in commercial environments, such as kitchens and fuel storage facilities. By leveraging the capabilities of RTOS, this system can provide timely and effective responses to potential threats, minimizing the risk of accidents and ensuring the safety of individuals and property.

Overall, this project demonstrates the importance of leveraging advanced technologies, such as RTOS, in developing safety-critical systems for commercial use. By providing a reliable and efficient safety solution, this project aims to reduce the incidence of accidents and enhance safety standards in commercial environments.

TABLE OF CONTENTS

SECTION I

TABLE OF CONTENTS	2
LIST OF FIGURES	3
LIST OF TABLES	4
ABBREVIATIONS & ACRONYMS	5
CHAPTER 1: INTRODUCTION	6
CHAPTER 2: LITERARY SURVEY	7
2.1 Background	7
2.2. Related Work	7
CHAPTER 3: SOFTWARE REQUIREMENT SPECIFICATION	9
3.1 Hardware Component Specification	9
3.1.1 INDUS board Microcontroller	9
3.1.2 ADS1115 MODULE (16 BIT-ADC)	10
3.1.3 OLED Display (SSD1306 168x64)	10
3.1.4 LPG Sensor (MQ-5)	11
3.1.5 DFR0076 flame sensor	11
3.1.6 CO Sensor (MQ-7)	12
3.1.7 BUZZER (ON-BOARD)	12
3.1.8 Stepper Motor	13
3.2 Communication Protocols	13
3.2.1 I2C (Inter-Integrated Circuit)	13
3.2.2 CAN FD Protocol	14
3.3 Software Specification	15
3.3.1 Development Environment:	15
STM32 CubeIDE:	16
FreeRTOS (CMSIS V1):	16
3.3.2 RTOS Configuration:	17
CHAPTER 4: ARCHITECTURE	18
CHAPTER 5: SYSTEM DESIGN	22
5.1 Flowchart	22
5.2 STM32Cube-MX .IOC design:	25
CHAPTER 6: IMPLEMENTATION & FUTURE SCOPE	26
6.1 Implementation	26
6.2 Future Scope	28

SECTION II

CHAPTER 7: CONCLUSION	29
CHAPTER 8: REFERENCES	30

LIST OF FIGURES

Fig.1 INDUS IoT Development Kit	09
Fig.2 ADS1115 Module	10
Fig.3 OLED Display	10
Fig.4 LPG Sensor	11
Fig.5 Flame Sensor	11
Fig.6 CO Sensor	12
Fig.7 Buzzer	12
Fig.8 . ULN2003 Driver Module & Stepper Motor	13
Fig.9 Block Diagram	18
Fig 10. Flowchart	22
Fig 11. FreeRTOSConfig.h	23
Fig 12. Structure for reentrant task	23
Fig 13. Re-entrant Task	23
Fig 14. Task Creation.....	24
Fig 15. High Priority Task.....	24
Fig 16. CAN FD Tx_Header Parameters.....	25
Fig 17. CAN FD filter function at receiver.....	25
Fig 18. Super loop code at receiver, that displays alert on OLED	25
Fig 19 Main INDUS .ioc	26
Fig 20 Node INDUS .ioc	26
Fig 21 Implemented Project	27

LIST OF TABLES

Table 1. Sensor Threshold for comparison19

ABBREVIATIONS & ACRONYMS

CAN FD: Controller Area Network Flexible Data-rate

ADC: Analog to Digital Converter

RTOS: Real Time Operating System

I2C: Inter-Integrated Circuit

CMSIS : Common Microcontroller Software Interface Standard

ARM: Advanced RISC Machine

RISC: Reduced Instruction Set Computer

HAL: Hardware Abstraction Layer

OLED: Organic Light-Emitting Diode

Chapter-1 INTRODUCTION

The rapid advancement in technology has led to the development of sophisticated safety-critical systems, particularly in commercial settings such as restaurants, fuel storage facilities, and industrial kitchens. Among these, fire and gas safety systems play a pivotal role in ensuring the safety of individuals and the protection of assets. In this context, the project focuses on the design and implementation of a real-time operating system (RTOS) based Fire & Gas safety critical system for commercial use.

The system utilizes the INDUS Development Board (STM32G431MBT6) as the core platform and integrates various hardware components including a Stepper Motor, ADS1115 MODULE (16 BITADC), OLED Display (SSD1306 168x64), LPG Sensor (MQ-5), Flame Sensor (DFR0076), CO Sensor (MQ-7), and Buzzer (ON-BOARD). Software development is done using STM32 CubeIDE and FreeRTOS (CMSIS V1), with version control managed through GitHub.

The primary objective of the project is to create a versatile device that can be employed in both restaurant kitchen and fuel storage facilities. The system is designed to detect and respond to fire and gas hazards promptly, ensuring the safety of occupants and minimizing the risk of property damage. By implementing a robust RTOS-based solution, the project aims to provide a reliable and efficient safety-critical system suitable for commercial applications.

The following sections of this report will provide a detailed analysis of the literature survey, software requirement specification, system architecture, design implementation, and conclusion. The project also explores future scope for enhancement and expansion of the system's capabilities to address emerging safety challenges in the commercial environment

LITERARY SURVEY

2.1 Background

Safety in commercial environments, particularly those involving the storage and use of flammable gases such as LPG, is a critical concern. Accidents related to fire and gas leaks can lead to devastating consequences, including loss of life, injuries, and extensive property damage. Traditional safety systems often rely on manual intervention or basic automation, which may not provide sufficient protection in emergency situations.

Real-Time Operating Systems (RTOS) offer a promising solution for enhancing safety-critical systems in commercial settings. RTOS provides deterministic behavior, ensuring that critical tasks are executed within specified time constraints. This capability is crucial in safety-critical applications where timely responses are essential to prevent accidents and mitigate risks.

The "RTOS based Fire & Gas Safety Critical System for Commercial Use" project aims to address these challenges by developing a sophisticated safety system that can detect and respond to potential fire and gas hazards in real-time. By leveraging the capabilities of RTOS, the system will be able to provide timely and reliable monitoring and control, enhancing overall safety in commercial environments.

We referred to various safety standards from Honeywell Safety Services GasBook.

<https://sps.honeywell.com/content/dam/honeywell-edam/sps/his/en-us/documents/services/sps-safety-services-gas-book.pdf>

2.2. Related Work

Luca de Oliveira Turci, "Real-Time Operating System FreeRTOS Application for Fire Alarm Project in Reduced Scale", International Journal of Computing and Digital Systems 6(4):198-204, July 2006

In the proposed project, an algorithm was developed by using Arduino Nano and FreeRTOS open source kernels in order to accomplish such a task in a reduced scale project. Some tests, such as jitter, latency, and worst case response time are also carried out to evaluate the performance of the real-time system proposed in the present project.

7

V Ramya, Balasubramaniam Palaniappan, "Embedded system for Hazardous Gas detection and Alerting", International Journal of Distributed and Parallel systems 3(3), May 2012

The main objective of the work is designing a microcontroller based toxic gas detecting and alerting system. The hazardous gases like LPG and propane were sensed and displayed each and every second in the LCD display.

M. Abdullah Khan, "Gas Detection Using ESP32 and Fire Alarm Project Report", September 2023

The "Gas Detection Using ESP32 and Fire Alarm" project aims to create a comprehensive system for detecting harmful gases and potential fire hazards in indoor environments. By utilizing the capabilities of the ESP32 micro controller and various gas sensors, the project enhances safety measures and environmental awareness.

Ganesh Gathiya, Krishna Kumar Patel and Kranti Yadav, "Home & Industrial Safety Using Fire & Gas Detection System", EasyChair, May 8, 2021

The system detects the leakage using the sensors. We have used the gas sensor to detect the leakage of the gas and we have also used the flame sensor to detect the flames. The MQ-2 and flame sensor simultaneously collect data from the environment and then transfer it to the Arduino UNO board in the form of analog inputs. The Arduino board then check the inputs and acts according to it.

SOFTWARE REQUIREMENT SPECIFICATION

3.1 Hardware Component Specification

3.1.1 INDUS board Microcontroller

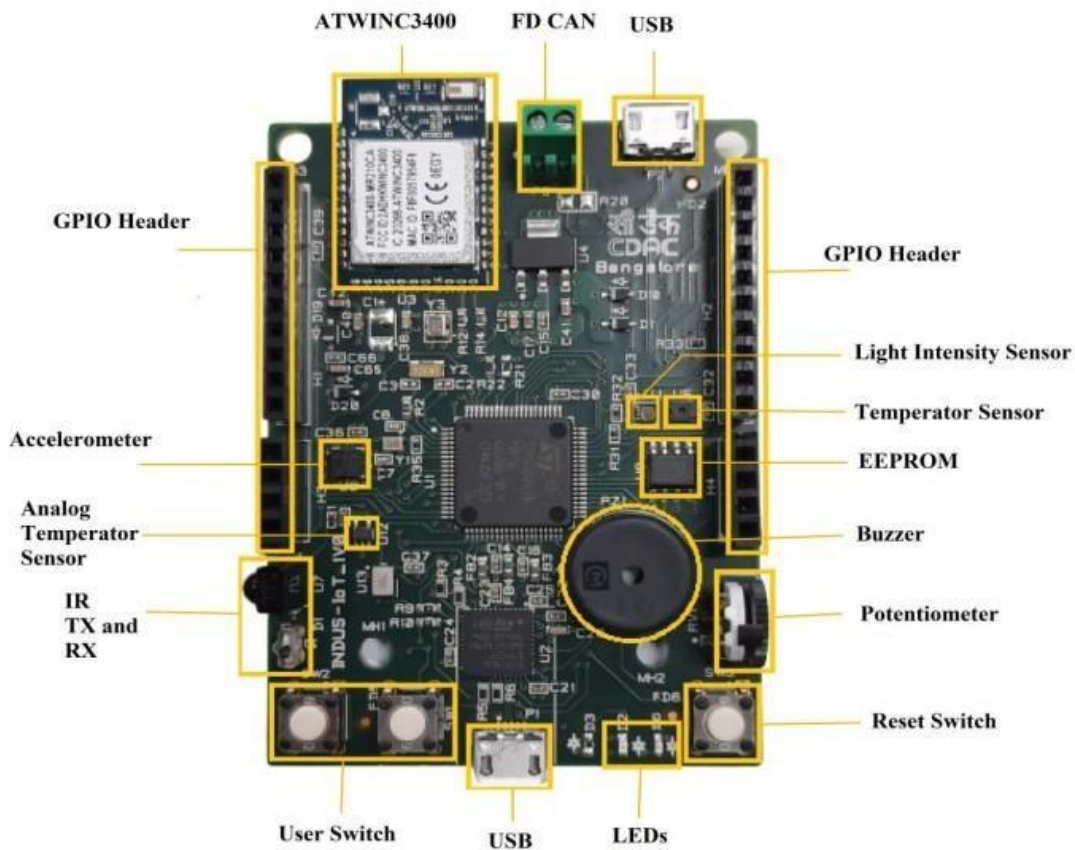


Fig 1. INDUS IoT DEVELOPMENT KIT

The INDUS Development Board features the STM32G431MBT6 processor, a powerful and versatile device suitable for a wide range of applications.

- Key specifications include:
 - 32-bit Arm Cortex-M4 core running at up to 170 MHz.
 - 128 KB Flash memory and 32 KB SRAM.
 - Comprehensive set of peripherals including GPIO, UART, SPI, I2C, ADC, DAC, and more.
 - Integrated USB controller for connectivity.

- Low-power operation and multiple power-saving modes.
- Compatibility with STM32 CubeIDE for development

3.1.2 ADS1115 MODULE (16 BIT-ADC)

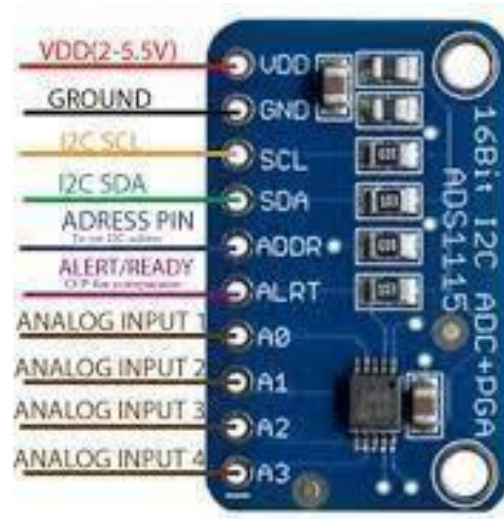


Fig 2. ADS1115 MODULE

Specifications

- It offers high precision and accuracy for sensor data acquisition.
- Key specifications include:
 - Resolution: 16-bit
 - Number of Channels: 4
 - Interface: I2C
 - Input Voltage Range: 5 V
 - Programmable Data Rate: 8 to 860 SPS
 - Operating Voltage: 2.0 to 5.5

3.1.3 OLED Display (SSD1306 168x64)

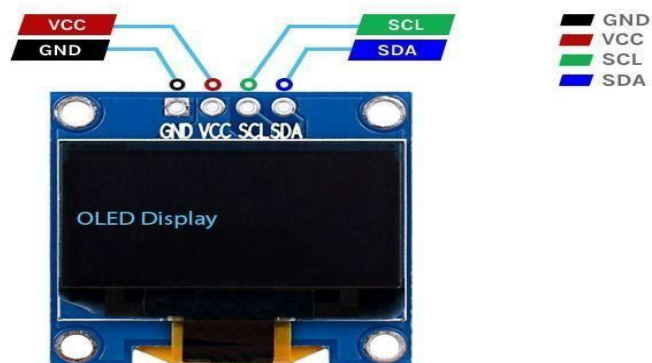


Fig 3. OLED Display

Specifications

- Display Type: OLED
- Resolution: 168x64 pixels
- Colors: Monochrome
- Interface: I2C
- Operating Voltage: 3.3V to 5V • Operating Current: ~20m

3.1.4 LPG Sensor (MQ-5)

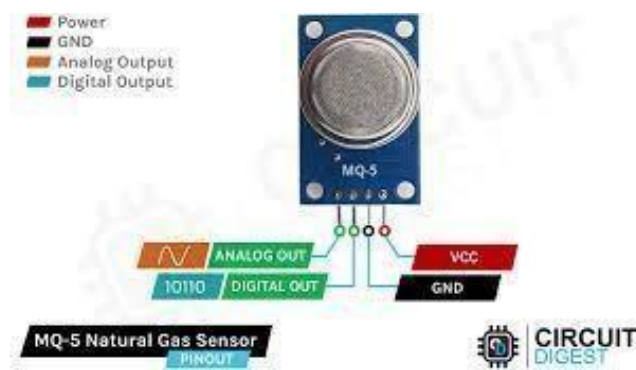


Fig 4. LPG Sensor

The MQ-5 gas sensor is designed to detect LPG, natural gas, and coal gas, providing an early warning system for potential hazards. Specifications include:

- Type: Gas Sensor
- Detection Gas: LPG, natural gas, coal gas
- Sensitivity: Adjustable
- Operating Voltage: 5V
- Heating Resistance: $31\Omega \pm 3\Omega$
- Load Resistance: 20K Ω

3.1.5 DFR0076 flame sensor



Fig 5. Flame Sensor

The DFR0076 flame sensor is used to detect the presence of flames, providing additional safety measures in the event of a fire. Specifications include:

- Type: Flame Sensor.
- Detection Range: 760 nm to 1100 nm.
- Operating Voltage: 3.3V to 5V.
- Detection Angle: 60 degrees.
- Response Time: < 20 ms.
- Digital (High/Low).

3.1.6 CO Sensor (MQ-7)

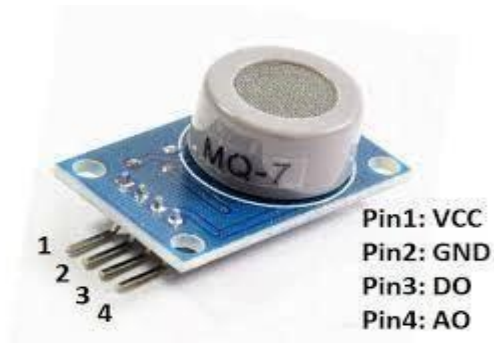


Fig 6. MQ-7

Specifications

- Detection Gas: Carbon Monoxide (CO)
- Sensitivity: Adjustable
- Operating Voltage: 5V
- Heating Resistance: $33\Omega \pm 3\Omega$
- Load Resistance: 20K

3.1.7 BUZZER (ON-BOARD)



Fig 7. ON-Board Buzzer

The on-board buzzer provides audible alerts and alarms in response to certain events or conditions detected by the system. It is used to alert users to potential hazards, such as the presence of gas leaks or fires, ensuring timely and appropriate responses.

3.1.8 Stepper Motor

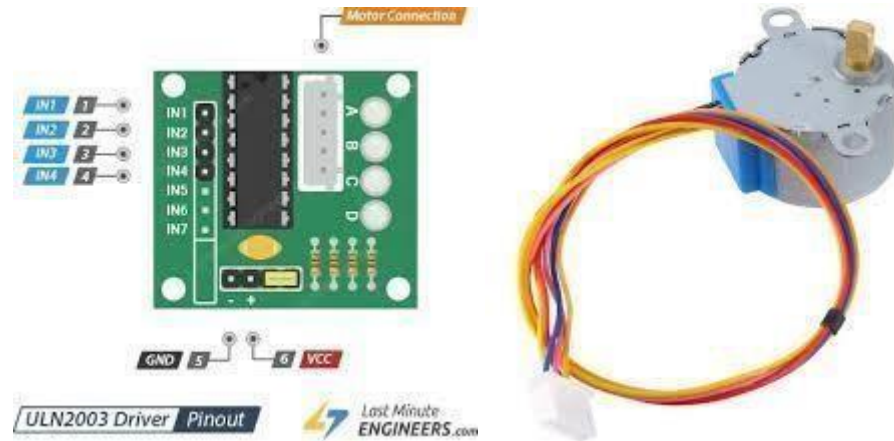


Fig 8. ULN2003 Driver Module & Stepper Motor

The stepper motor's rotation is controlled by the microcontroller, converting digital pulses into precise mechanical movements. This makes it ideal for applications requiring accurate positioning. The microcontroller sends step pulses to the motor driver, which energizes the motor coils in a specific sequence for rotation. By adjusting the number and sequence of pulses, the microcontroller can control the speed and direction of rotation, ensuring precise movement control. This capability makes stepper motors suitable for applications like industrial automation, robotics, and 3D printing, where precise positioning is crucial.

3.2 Communication Protocols

3.2.1 I2C (Inter-Integrated Circuit)

I2C operates using a two-wire serial interface, which consists of a clock line (SCL) and a data line (SDA). The master device, typically a microcontroller, initiates and controls the communication by generating clock pulses and transmitting data to the slave devices connected to the bus. The slave devices respond by sending data back to the master.

In the context of the INDUS Board, it acts as the master device, communicating with the ADS1115 ADC Module that has the slave address of 0x48 and has Gravity Analog Flame Sensor, MQ-7 CO Sensor & MQ5 LPG sensor to each of its analog input channel. Each sensor has a unique register number, allowing the master to address and communicate with specific sensors individually.

One of the key features of I2C is its acknowledgment mechanism. After transmitting a byte of data, the master waits for an acknowledgment from the slave device. If the slave receives the data correctly, it sends an acknowledgment bit back to the master. This ensures that data is transmitted and received accurately, minimizing the risk of data corruption.

I2C offers several advantages over other communication protocols, particularly for embedded systems and real-time applications:

1. **Reduced Wiring Complexity:** I2C requires only two wires (SCL and SDA), significantly simplifying the hardware setup and reducing potential signal interference compared to protocols that require multiple wires.
2. **Multi-Device Compatibility:** I2C allows for the integration of multiple sensors or devices on the same bus, each with its unique address. This enables the creation of complex systems with various sensors and actuators communicating seamlessly.
3. **Low Power Consumption:** I2C is designed with energy efficiency in mind, making it suitable for applications where power consumption is a critical factor. It operates at low voltages and minimizes power consumption during idle periods.
4. **Scalability:** The ability to support multiple devices on the same bus makes I2C highly scalable, allowing for the easy addition or removal of sensors or devices as needed.

Overall, I2C is a versatile and reliable communication protocol that simplifies the development of embedded systems and real-time applications, providing an efficient and low-power solution for inter-device communication.

3.2.2 CAN FD Protocol

CAN FD (Controller Area Network Flexible Data-rate) is a widely used communication protocol in automotive and industrial applications. It represents an evolutionary step from the classic CAN protocol, offering higher data rates and larger payload sizes while maintaining compatibility with existing CAN networks. CAN FD operates using a differential two-wire interface (CANH and CANL) and supports both standard and extended frame formats.

In the context of the INDUS Board project, CAN FD plays a vital role in facilitating communication between the master device and peripheral devices. Each device connected to the CAN FD bus can act as a node, actively transmitting and receiving messages based on a predefined protocol.

A key advantage of CAN FD is its ability to support significantly higher data rates compared to traditional CAN protocols, reaching up to 8 Mbps. This enhanced data rate enables faster and more efficient data exchange, making CAN FD suitable for applications requiring real-time communication and high-speed data transfer.

CAN FD is renowned for its robustness and reliability. It employs a priority-based message arbitration scheme to ensure deterministic and timely message delivery, even in congested network conditions. Additionally, CAN FD incorporates error detection and correction mechanisms, such as cyclic redundancy checks (CRCs), to detect and mitigate data errors, ensuring reliable communication in noisy environments.

CAN FD stands out for its scalability and flexibility. It allows for easy integration of new devices into existing networks, enabling seamless expansion or reconfiguration as needed. This flexibility

makes CAN FD adaptable to changing system requirements and facilitates the addition of new nodes or sensors without disrupting the overall network functionality.

The versatility and reliability of CAN FD make it a suitable communication protocol for safetycritical systems where fast and reliable communication is crucial.

3.3 Software Specification

3.3.1 Development Environment:

STM32 CubeIDE:

- **Overview:**
 - STM32 CubeIDE is an Eclipse-based Integrated Development Environment (IDE) specifically designed for STM32 microcontrollers.
 - It offers a comprehensive suite of tools for developing, debugging, and managing STM32-based projects.
- **Features:**
 - **User-Friendly Interface:**
 - Intuitive graphical user interface for easy navigation and code editing.
 - Built-in editor with syntax highlighting, code completion, and error checking.
 - **Project Management:**
 - Project templates for quick and easy project setup.
 - Support for multiple projects and workspaces.
 - Version control integration with Git.
 - **Debugging:**
 - On-chip debugging using ST-Link or Seggar J-Link debug probes.
 - Comprehensive debugging features such as breakpoints, watchpoints, and variable inspection.
 - Integrated terminal for console output and command execution.
- **Toolchain:**
- **Overview:**
 - STM32 CubeIDE includes the GCC compiler toolchain for compiling, linking, and debugging STM32 projects.
 - The toolchain provides the necessary compiler, linker, and debugger tools for developing embedded applications.
- **Features:**
 - **GCC Compiler:**
 - Supports all major C and C++ language features.
 - Optimizations for code size and execution speed.
 - Built-in libraries for peripheral access and standard C functions.
 - **Linker:**

- Generates executable files (.elf) and downloadable images (.hex, .bin). ■ Supports linker scripts for memory mapping and section placement.
- **Debugger:**
 - GDB-based debugger with command-line interface.
 - Supports debugging of both C and assembly code.
 - Breakpoint setting, variable inspection, and call stack navigation.

FreeRTOS (CMSIS V1):

- **Overview:**
 - FreeRTOS is an open-source, portable, and robust Real-Time Operating System (RTOS) designed for embedded systems.
 - It provides a preemptive, priority-based scheduler for managing multiple tasks efficiently.
 - CMSIS V1 is a standardized hardware abstraction layer for ARM Cortex-M processors.
 - It defines standardized APIs for accessing hardware peripherals, making it easier to port applications across different STM32 devices.
- **Features:**
 - **Task Management:**
 - Supports multiple tasks running concurrently.
 - Each task has a unique priority level, determining its scheduling order.
 - Tasks can be created, deleted, and suspended dynamically.
 - **Inter-Task Communication:**
 - Provides mechanisms like queues, semaphores, and mutexes for safe data exchange and task coordination.
 - Queues for message passing between tasks.
 - Semaphores for resource synchronization.
 - Mutexes for mutual exclusion.
 - **Tickless Idle Mode:**
 - Reduces power consumption by entering a low-power sleep mode when no tasks are ready to run.
 - Wakes up when a task becomes ready or an interrupt occurs.

3.3.2 RTOS Configuration:

- **Overview:**
 - The RTOS configuration in STM32 CubeIDE allows you to manage tasks, queues, and interrupts efficiently.
- **Features:**
 - **Task Configuration:**

- Define task names, priorities, and stack sizes.
- Assign tasks to specific cores in multi-core STM32 devices.
- Set task affinities for optimal scheduling. ○

Queue Configuration:

- Create queues for inter-task communication.
- Specify queue sizes and message types.

○ **Interrupt Configuration:**

- Configure interrupts for peripherals and external events.
- Associate interrupt handlers with tasks for efficient handling.

- **Task Management:**

- **Overview:**

- STM32 CubeIDE provides a comprehensive task management system for creating, scheduling, and managing multiple tasks in an RTOS-based application.

- **Features:**

- **Task Creation:**

- Create tasks with unique names, priorities, and stack sizes.
- Specify task entry points and arguments.

- **Task Scheduling:**

- The RTOS scheduler assigns CPU time to tasks based on their priorities.
- Higher-priority tasks preempt lower-priority tasks when they become ready to run.

- **Task Synchronization:**

- Tasks can synchronize their execution using synchronization primitives such as semaphores and mutexes.
- Semaphores for counting resources and preventing race conditions. ■
Mutexes for mutual exclusion and critical section protection.

ARCHITECTURE

Block Diagram

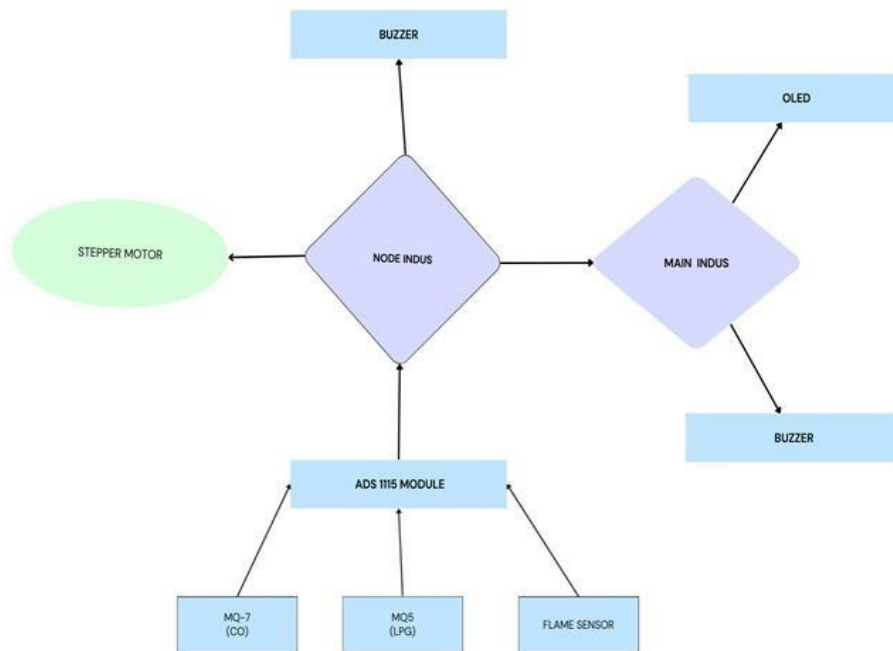


Fig 9 Block Diagram

The block diagram depicts the architecture of a fire & gas detection system that utilizes multiple sensors and microcontrollers for functionality.

Key Components and their Roles:

Gas Sensors (MQ-7, MQ5):

- **MQ-7:** Designed to detect carbon monoxide (CO) in the environment. It operates by sensing the concentration of CO and generating an analog voltage signal proportional to the detected level.
- **MQ-5:** Primarily utilized to detect flames, particularly liquefied petroleum gas (LPG). Similar to the MQ-7, it produces an analog output signal corresponding to the presence and intensity of a flame.

ADS1115 ADC:

- Responsible for converting the analog voltage signals received from the gas sensors into digital data. This conversion allows microcontrollers to process and interpret the sensor readings more effectively.

Microcontrollers (NODE INDUS, MAIN INDUS):

Serve as the core of the system, performing crucial tasks, the Node INDUS present at critical location performs tasks such as:

- **Reading Sensor Data:** Continuously receive and process digital data from the ADC.
- **Threshold Comparison:** Compare the received sensor data against predefined safety thresholds or limits to determine if hazardous levels are reached.
- **Stepper Motor Control:** Potentially used to control the positioning of the gas knob at various locations for broader coverage and increased safety by reducing risk of explosions(not entirely but can make a lot of difference).

CAN FD Communication: Utilize the Controller Area Network Flexible Data-rate (CAN FD) protocol for transmitting processed sensor data, alerts, and warnings from other microcontrollers on the network to the Main INDUS present at security.

Output Control: Control peripherals like a buzzer and OLED display based on sensor readings and threshold conditions at Main INDUS. For instance, triggering an audible alarm when gas or flame levels exceed safety limits.

Sensor Name	Threshold Value (Voltage)
MQ5	3.5
MQ7	1.0
DFR0076	4.5

Table 1. Sensor Threshold for comparison

Stepper Motor:

- Facilitates the movement of the fuel source(LPG Cylinder) knob, thus ensuring adequate safety so as to prevent fire explosions and largely reduce the risk.

Buzzer:

- Emits audible alerts to notify personnel working at the critical location & to the security & monitoring personnel when gas,CO or flame levels exceed predetermined safety limits, providing an immediate indication of potential danger.

OLED Display:

- Serves as a visual display, presenting real-time gas readings, alerts or warnings, and potentially device status information. This allows users to monitor the system's status and take appropriate actions if necessary.

CAN FD Protocol and Communication:

This is used to communicate the type of alert & the node number i.e. number of the INDUS corresponding to the critical location thereby helping security to identify the situation and respond correctly.

- The CAN FD protocol offers several advantages in this system:
 - **Efficiency:** CAN FD enhances data transfer efficiency by supporting larger data packets (up to 64 bytes) and variable data rates. This enables the transmission of more extensive sensor data and comprehensive device state information between microcontrollers in a shorter timeframe.
 - **Flexibility:** The variable data rate capability of CAN FD allows for switching speeds during message transmission. This flexibility ensures faster

communication of critical alerts, enabling prompt action when thresholds are exceeded.

- **Inter-Board Communication:** The CAN FD protocol facilitates seamless communication between the NODE INDUS and MAIN INDUS microcontrollers.

This distributed architecture enables various functions, including:

- **Sensor Data Sharing:** One microcontroller might collect sensor data and transfer it to the other for analysis or centralized alerts.
- **Synchronized Actions:** Microcontrollers can coordinate actions like buzzer activation and OLED display updates to provide a cohesive response to detected gas or flame levels.
- **System Redundancy:** In the event of a microcontroller failure, CAN FD communication helps maintain safety functions on the other board, ensuring continued monitoring and alerting capabilities.

Potential Use Cases:

- **Restaurants:** Such systems are designed to detect, respond to, and mitigate fire and gas-related risks to ensure the safety of occupants and property.
- **Industrial Safety:** Monitoring environments with potential hazards, such as carbon monoxide or LPG leaks, and providing timely alerts to personnel to prevent accidents.
- **Smart Home:** Integration with smart home systems for gas leak detection in residential settings or automated ventilation control based on gas sensor readings.
- **Environmental Monitoring:** Utilizing the stepper motor to enable multi-point ambient gas monitoring in outdoor environments, providing valuable data for environmental research or air pollution monitoring.

SYSTEM DESIGN

5.1 Flowchart

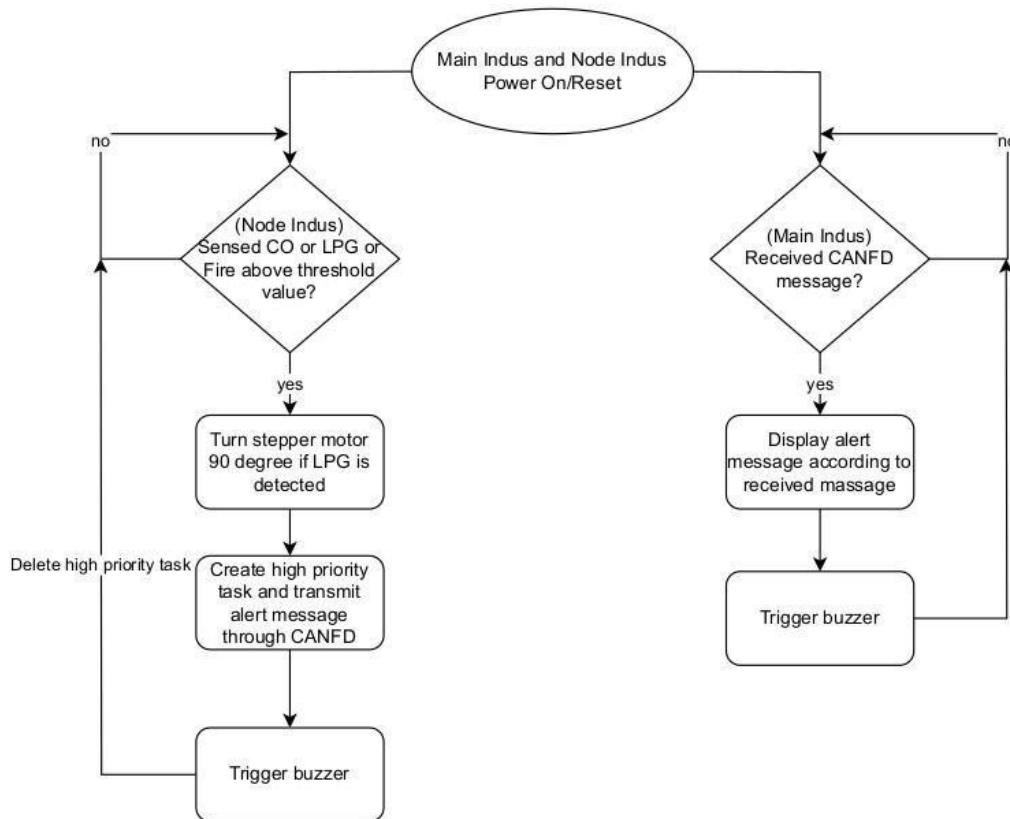


Fig 10. Flowchart

The above flow-chart explains the working of the our project/system, it is divided into 2 parts:

1. Node INDUS:

This is the sensor interfaced microcontroller module designed to be placed at the critical locations such as fuel storage areas, kitchen, industrial vents etc. They can be single or multiple based on the requirement of the customer.

In these modules three tasks having equal priorities that are used to get sensor data from the ADS1115 ADC module are running. The scheduling policy is Cooperative scheduling i.e. where we use “vTaskDelay()” to yield the CPU control as per required duration. The

preemption parameter is also “ON” in ‘FreeRTOSConfig.h’ file as it will be explained in code snippets below along with increase in heap size to accommodate multiple reentrant tasks.

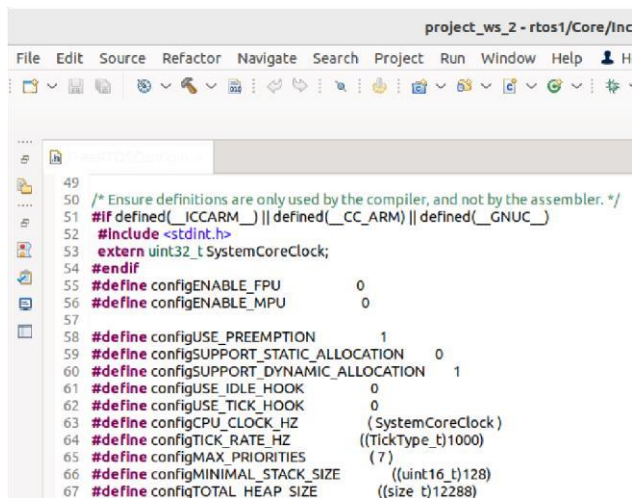


Fig 11. FreeRTOSConfig.h



Fig 12. Structure for reentrant task

This reentrant task will take parameters from pointer to structure and will fetch data via I2C and compare it with critical voltage values(after converting the RAW data into voltage values & then ppm values if needed), if not then it will repeat this again & again.

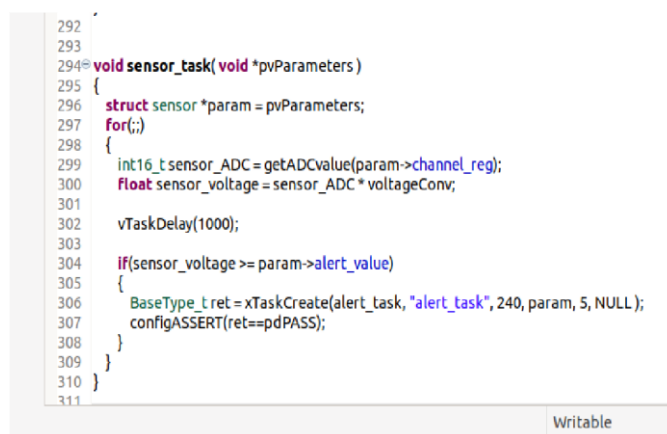
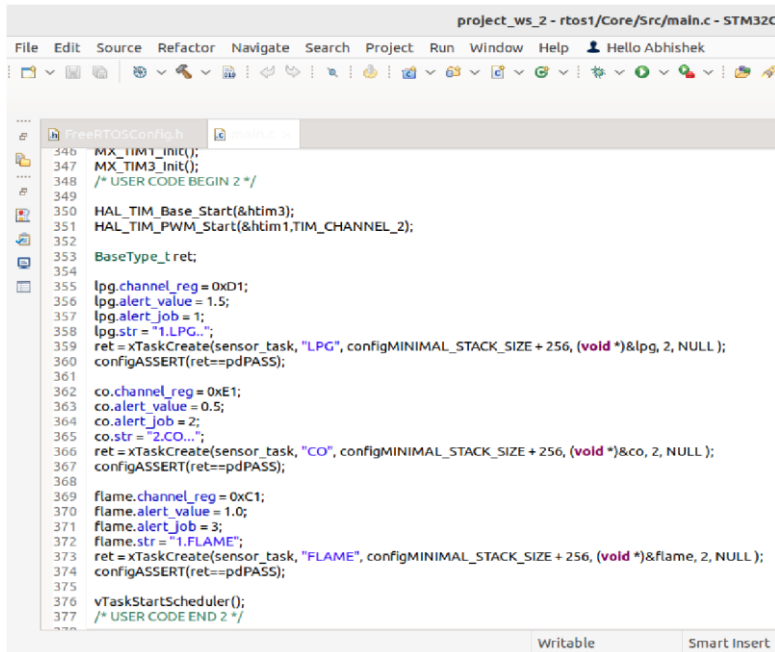


Fig 13. Re-entrant Task

If the reading is greater than critical value then, this task will create a high priority task(refer code implementation/snippet section below) meant for transmitting the type of alert messages & node number using CAN FD. After this it turns “ON” the on-board buzzer using a PWM signal after which the task gets deleted using “vTaskDelete(NULL)” so as to reduce wastage of resources.

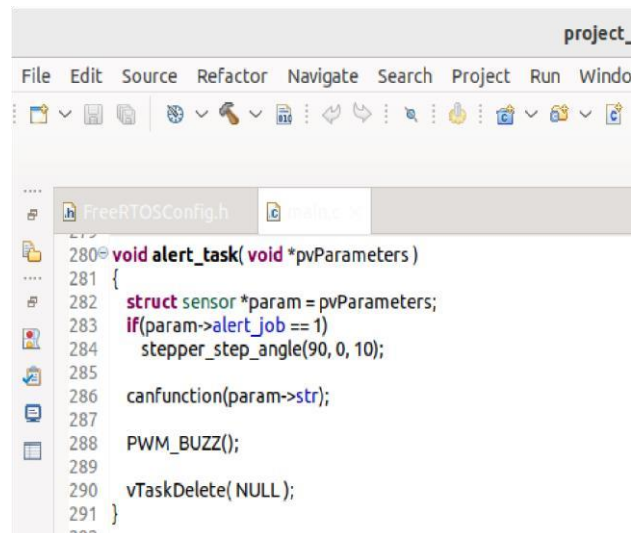


```

346 MX_TIM1_Init();
347 MX_TIM3_Init();
348 /* USER CODE BEGIN 2 */
349
350 HAL_TIM_Base_Start(&htim3);
351 HAL_TIM_PWM_Start(&htim1,TIM_CHANNEL_2);
352
353 BaseType_t ret;
354
355 lpg.channel_reg = 0xD1;
356 lpg.alert_value = 1.5;
357 lpg.alert_job = 1;
358 lpg.str = "1.LPG..";
359 ret = xTaskCreate(sensor_task, "LPG", configMINIMAL_STACK_SIZE + 256, (void *)&lpg, 2, NULL);
360 configASSERT(ret==pdPASS);
361
362 co.channel_reg = 0xE1;
363 co.alert_value = 0.5;
364 co.alert_job = 2;
365 co.str = "2.CO..";
366 ret = xTaskCreate(sensor_task, "CO", configMINIMAL_STACK_SIZE + 256, (void *)&co, 2, NULL);
367 configASSERT(ret==pdPASS);
368
369 flame.channel_reg = 0xC1;
370 flame.alert_value = 1.0;
371 flame.alert_job = 3;
372 flame.str = "1.FLAME";
373 ret = xTaskCreate(sensor_task, "FLAME", configMINIMAL_STACK_SIZE + 256, (void *)&flame, 2, NULL);
374 configASSERT(ret==pdPASS);
375
376 vTaskStartScheduler();
377 /* USER CODE END 2 */

```

Fig 14. Task Creation



```

280 void alert_task( void *pvParameters )
281 {
282     struct sensor *param = pvParameters;
283     if(param->alert_job == 1)
284         stepper_step_angle(90, 0, 10);
285
286     canfunction(param->str);
287
288     PWM_BUZZ();
289
290     vTaskDelete( NULL );
291 }
292
293 /* USER CODE END 2 */

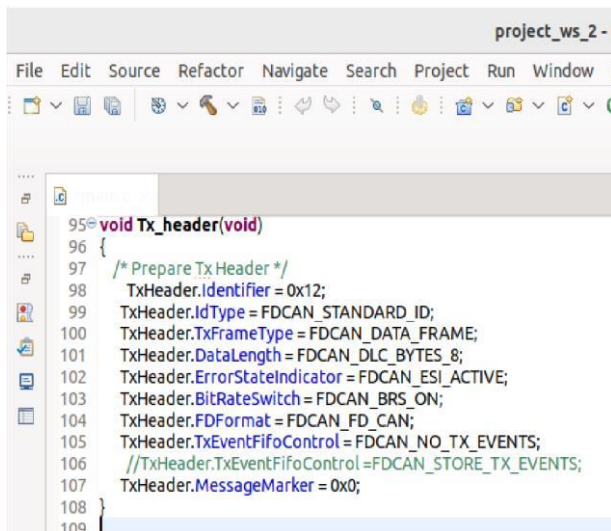
```

Fig 15. High Priority Task

To stop the buzzer one needs to manually reset it after crisis is resolved or wait for few minutes, though even if the buzzer is “ON”, the three tasks will still run and take readings for any scenarios like “fire + CO” or “LPG + CO” leaks occur can alert as well.

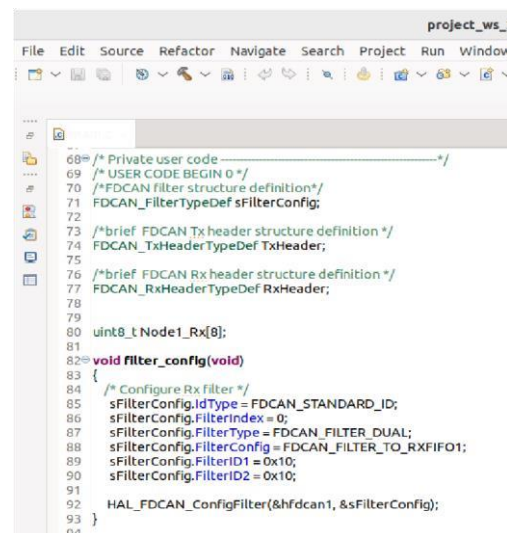
2. Main INDUS:

This is a single module present at security personnel that will alert the personnel in case of crisis by first displaying the type of alert and node number on the OLED display and synchronously turn on the buzzer for effectiveness. This data will allow the personnel to take appropriate action as per the situation thereby reducing the risks and losses, by a large margin.



```
project_ws_2 - I
File Edit Source Refactor Navigate Search Project Run Window
95 void Tx_header(void)
96 {
97     /* Prepare Tx Header */
98     TxHeader.Identifier = 0x12;
99     TxHeader.IdType = FDCAN_STANDARD_ID;
100     TxHeader.TxFrameType = FDCAN_DATA_FRAME;
101     TxHeader.DataLength = FDCAN_DLC_BYTES_8;
102     TxHeader.ErrorStateIndicator = FDCAN_ESI_ACTIVE;
103     TxHeader.BitRateSwitch = FDCAN_BRS_ON;
104     TxHeader.FDFormat = FDCAN_FD_CAN;
105     TxHeader.TxEventFifoControl = FDCAN_NO_TX_EVENTS;
106     //TxHeader.TxEventFifoControl = FDCAN_STORE_TX_EVENTS;
107     TxHeader.MessageMarker = 0x0;
108 }
109
```

Fig 16. CAN FD Tx_Header Parameters



```
project_ws_2
File Edit Source Refactor Navigate Search Project Run Window
68 /* Private user code */
69 /* USER CODE BEGIN 0 */
70 /* FDCAN filter structure definition */
71 FDCAN_FilterTypeDef sFilterConfig;
72
73 /*brief FDCAN Tx header structure definition */
74 FDCAN_TxHeaderTypeDef TxHeader;
75
76 /*brief FDCAN Rx header structure definition */
77 FDCAN_RxHeaderTypeDef RxHeader;
78
79
80 uint8_t Node1_Rx[8];
81
82 void filter_config(void)
83 {
84     /* Configure Rx filter */
85     sFilterConfig.IdType = FDCAN_STANDARD_ID;
86     sFilterConfig.FilterIndex = 0;
87     sFilterConfig.FilterType = FDCAN_FILTER_DUAL;
88     sFilterConfig.FilterConfig = FDCAN_FILTER_TO_RXFIFO1;
89     sFilterConfig.FilterID1 = 0x10;
90     sFilterConfig.FilterID2 = 0x10;
91
92     HAL_FDCAN_ConfigFilter(&hfdcan1, &sFilterConfig);
93 }
94
```

Fig 17. CAN FD filter function at receiver

The above are the functions and code snippets that are to be added to the main.c at receiver side to facilitate reception. The “tx-header” will define the parameters of the transmitter & the “filter_config” will ensure that messages will not overcrowded at the receiver side, ensuring clean data reception.

IMPLEMENTATION & FUTURE SCOPE

6.1 Implementation

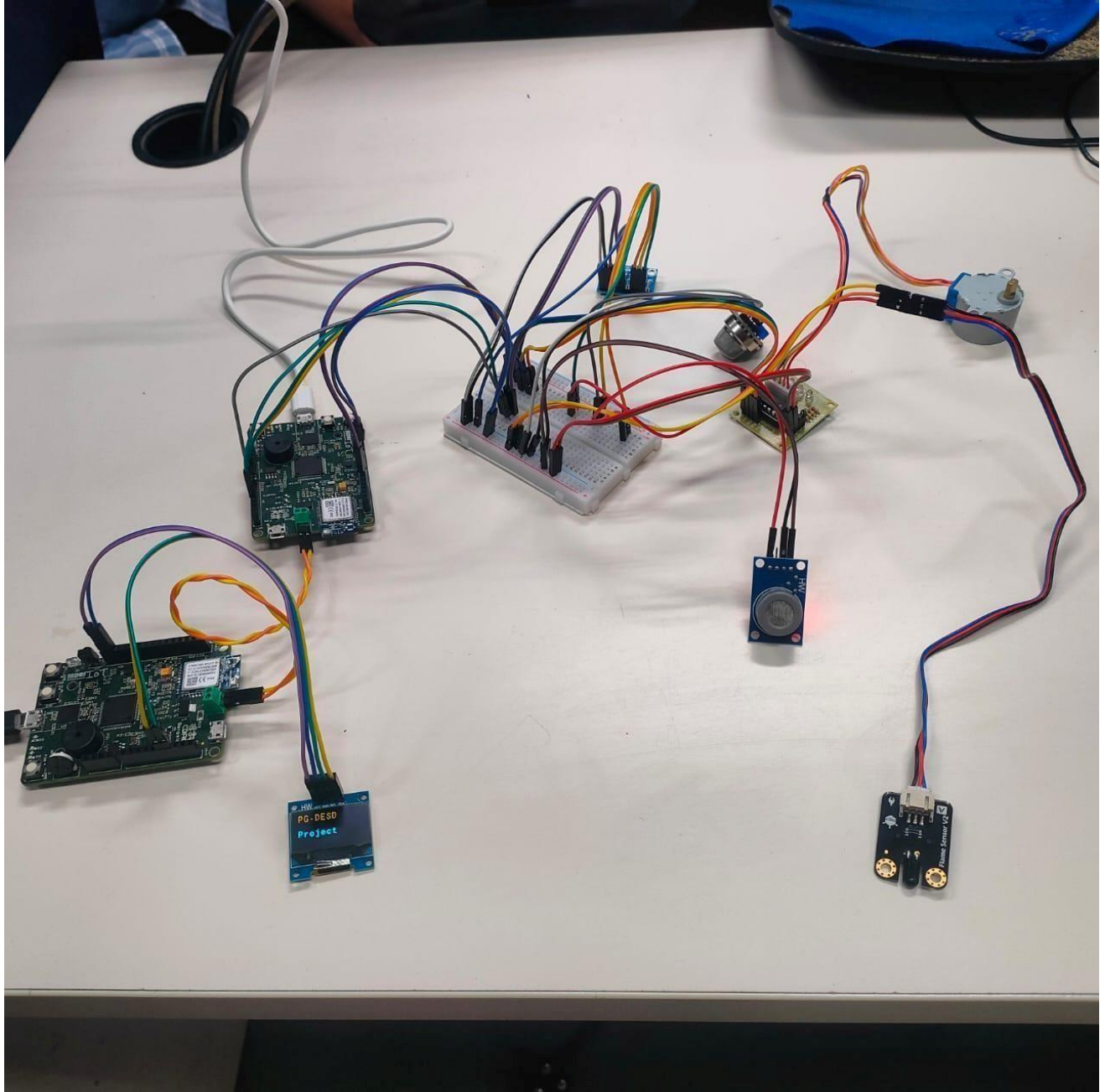


Fig 21. Implemented Project

The project has been implemented as shown in the above picture, the Main INDUS connected to an OLED display displays the initial message upon power on, and turns off to save power. Then it turns on only after a threat is detected i.e. when an alert message is transmitted via any of the CAN FD

connected nodes, to display the type of alert and the node number corresponding to the critical location. It has several modes such as:-

Normal Operation:

During normal operation, the ADS1115 ADC module continuously samples data from its sensors, including the MQ7 for CO, MQ5 for LPG, and the flame sensor for fire detection. Each sensor reading is processed within the FreeRTOS scheduler, ensuring that tasks related to CO, LPG, and fire detection are all active and running concurrently. These tasks are of equal priority, allowing the system to respond promptly to any hazardous conditions that may arise. Meanwhile, the display node remains in a standby state, ready to receive and display any relevant information transmitted from the sensor node.

Detection of CO, LPG, or Fire:

When any of the sensors on the sensor node detect a hazardous condition exceeding predefined thresholds, the corresponding task is triggered. For instance, if high levels of CO or LPG are detected, or if the flame sensor detects a fire, the alert task is activated. This task immediately executes, first performing any necessary physical responses, such as activating the stepper motor to turn 90 degrees clockwise to turn off the knob if LPG is detected. Subsequently, the alert task sends a message via the CAN FD protocol to the display node, specifying the type of hazard detected. This message transmission ensures that the display node receives immediate notification of the hazard. Simultaneously, the alert task triggers the buzzer on the sensor node to sound an alarm, providing localized warning.

Reception and Display on Display Node:

Upon receiving the hazard detection message from the sensor node, the display node interprets the message and initiates corresponding actions. It activates its own buzzer to provide additional audible notification and updates the OLED display to visually represent the type of hazard detected. This real-time display of hazard information ensures that users are promptly informed and can take appropriate action to address the situation.

End of Alert Condition:

After completing its tasks, including message transmission and local alarm activation, the alert task is deleted to free up system resources. This deletion ensures that other sensing tasks can resume execution without delay, maintaining continuous monitoring of environmental conditions. Meanwhile, the system returns to a normal operational state, with sensors continuing to monitor for any further hazards.

Manual Buzzer Stop:

Once activated, the buzzer on either the sensor node or display node can only be stopped manually after resetting both nodes. This manual intervention ensures that users are actively engaged in acknowledging and addressing the detected hazard, promoting safety and accountability within the system.

6.2 Future Scope

The future scope of your project, "RTOS based Fire & Gas Safety Critical System for Commercial Use," could include the following aspects:

Enhanced Sensor Integration: Integrate more advanced sensors for detecting various gases and fire types, improving the system's accuracy and reliability.

Data Logging and Analysis: Implement data logging capabilities to store sensor data over time & upload to cloud via wifi module, allowing for analysis and trend identification to improve safety measures.

Fault Detection and Diagnostics: Develop algorithms for fault detection and diagnostics to identify and mitigate system failures or malfunctions.

Energy Efficiency: Implement energy-efficient algorithms and power management techniques to reduce power consumption and extend the system's battery life.

User Interface Enhancement: Improve the user interface of the system to make it more intuitive and user-friendly, enhancing user experience and ease of operation.

Expandability and Scalability: Design the system to be easily expandable to include more number of nodes that can cover a wide range of critical locations i.e. scalable, allowing for future upgrades and additions to meet evolving safety needs.

MPU : To include an MPU in the RTOS, so as to increase the robustness of the system.

CONCLUSION

This project, focused on developing an RTOS-based fire and gas safety system for commercial applications, effectively demonstrates the inherent suitability of real-time operating systems for such safety-critical implementations. The system's core competency – the ability to detect potential hazards like gas leaks and fires and initiate prompt responses – showcases the value of RTOS-driven designs within the commercial safety domain. By prioritizing real-time responsiveness, the system has the potential to significantly mitigate risks in environments such as restaurant kitchens and fuel storage facilities, where timely intervention is paramount for preventing accidents and potential loss of life.

The choice of the INDUS Development Board (STM32G431MBT6) underscores the importance of selecting a microcontroller platform capable of handling the real-time demands of a safety system. The STM32G4 series' blend of processing power, peripheral support, and power efficiency made it well-suited to this application. The integration of sensors specifically tailored to detect LPG, flames, and carbon monoxide, coupled with the actuation capability of a stepper motor, provided the system with the necessary 'senses' and reactive mechanisms to fulfill its safety purpose. The OLED display served as a crucial interface for system status and alerts.

Software architecture played a pivotal role in the system's success. The adoption of FreeRTOS ensured deterministic task scheduling and predictable system behavior – essential characteristics within the realm of safety-critical applications. The utilization of established communication protocols like CAN FD and I2C streamlined the interaction between various hardware components and the central microcontroller, ensuring reliable data transfer and control. The project's successful implementation highlights the proficiency achieved in integrating hardware and software components under the constraints of a real-time operating system.

While the project achieved its primary goals, there remains ample scope for future enhancements and exploration. Integrating more sophisticated and specialized sensors could further improve the system's detection accuracy and response spectrum. For example, the inclusion of infrared temperature sensors might enable preemptive detection of potential fire hazards. The user interface could undergo refinement to present information more intuitively, particularly during critical situations. Networking capabilities, such as Wi-Fi or Ethernet connectivity, could enable remote monitoring and centralized logging, enhancing situational awareness for broader safety management. Additionally, exploring machine learning techniques could allow for predictive maintenance and anomaly detection, moving towards a more proactive safety paradigm.

This project aligns with the ongoing trend within the embedded systems domain to leverage the power and flexibility of RTOS-based solutions in increasingly complex and safety-sensitive applications. The successful outcome reinforces the effectiveness of this approach, offering a compelling example of how RTOS-driven designs can elevate safety standards in commercial and industrial environments. As safety regulations become more stringent and technological capabilities expand, projects like this will undoubtedly pave the way for the next generation of intelligent, responsive, and highly reliable safety-critical systems.

REFERENCES

- D. Birla, R. P. Maheshwari, and H. O. Gupta, "A New Non-linear Directional Overcurrent Relay Coordination Technique, and Banes and Boons of Near-end Faults Based Approach," IEEE Transactions on Power Delivery, vol. 21, no. 3, pp. 1176-1182, July 2006.
- S. M. Alavi, A. D. Raj, and H. G. Naik, "Implementation of Adaptive Fuzzy Logic Controller for Temperature Control of a Heat Exchanger," IEEE Transactions on Industrial Electronics, vol. 58, no. 10, pp. 4787-4794, Oct. 2011.
- A. K. Sahoo and M. P. Dave, "Design and Implementation of a Fire Detection System Using Raspberry Pi," 2018 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI), Chennai, India, 2018, pp. 469-473.
- M. A. Naeem, S. U. Khan, and A. M. Siddiqui, "Design and Implementation of a Gas Leakage Detection and Control System Using IoT," 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Halifax, NS, Canada, 2018, pp. 1227-1232.
- D. J. De Sousa, S. S. Prabhu, and S. S. Rao, "Real-Time Embedded Systems: Design Principles and Engineering Practices," CRC Press, 2015.
- S. S. Prabhu and K. K. Shukla, "Real-Time Operating Systems: Concepts and Implementation," McGraw-Hill Education, 2009.
- Warren Gay, "Beginning STM32: Developing with FreeRTOS, libopenm3 and GCC", ISBN 978-1-4842-3624-6, Apress, 2018.
- Michael J. Pont, "The Engineering of Reliable Embedded Systems (Second Edition)", ISBN 978-14842-3624-6, Apress, 2016.