Project Report

Data logger system(FreeRTOS) using STM32 Nucleo F446RE

Advanced Embedded Systems and Designs

Name: Kartik Khandelwal and Viren Sharma

Roll Number: 21UEC071 and 21UEC143

Instructor: Dr. Deepak Nair

Department of Electronics and Communication Engineering
The LNMIIT Jaipur, Rajasthan

November 30, 2024

Declaration

This report has been prepared based on my work. Where other published and unpublished source materials have been used, these have been acknowledged.

Student Name: Kartik Khandelwal and Viren Sharma
Roll Number: 21UEC071 and 21UEC143
Date of Submission: 30/11/2024

Table of Contents

Abstract

This project presents the design and implementation of a robust, real-time data logging system using the STM32F446RE microcontroller, integrated with FreeRTOS for efficient multitasking. The system collects and logs environmental data using sensors such as System Core Temperature(ADC with DMA), RCWL-0516 (proximity), and MQ135 (ADC). A Real-Time Clock (RTC) ensures precise timestamping, while an OLED SSD1306(I2C) displays real-time data for immediate visualization. Simultaneously data is reliably stored on an SD card(SPI). An interrupt attached to a switch also demonstrates the ability to use interrupts in FreeRTOS using semaphores.

RTOS mechanisms like mutexes, semaphores and queues are employed to synchronize tasks, ensuring safe access to shared resources, such as the OLED and SD card, and efficient data handling. The modular architecture facilitates debugging and future scalability, making the system versatile for various applications. This project emphasizes real-time responsiveness, offering a practical solution for environments requiring continuous monitoring and data archiving.

The implementation was tested rigorously, and the system demonstrated seamless integration of components, robust task synchronization, and reliable data storage, validating its effectiveness for real-world deployment.

Component Name

1. STM32 (Nucleo F446RE) Microcontroller board:

The STM32 Nucleo-F446RE is a powerful and versatile microcontroller development board from STMicroelectronics. It's part of STMs ARM Cortex-M4F-based microcontrollers.

The Nucleo F446RE board features:

- High-Performance Cortex-M4 Core: The 180 MHz ARM Cortex-M4 core comes with a Floating Point Unit (FPU).
- Memory Resources for RTOS: 512 KB Flash Memory provides sufficient storage for FreeRTOS and your application code. It also contains 128 KB SRAM, divided across different SRAM regions, offers flexibility for task stacks, queues, and other FreeRTOS data structures.
- Peripheral Support and Connectivity for Multitasking: Multiple communication interfaces such as USART/UART, I2C, SPI, and CAN allow FreeRTOS tasks to handle concurrent communication processes independently.
- SysTick Timer: Native support for the SysTick timer allows seamless FreeRTOS tick generation, essential for task management.
- GPIO and External Interrupts: Contains configurable GPIO Pins and external interrupts, allows task synchronization based on external events, such as sensors or user inputs, ideal for interrupt-driven FreeRTOS applications.
- Advanced Debugging and Tracing for RTOS using CubeMXIDE: Onboard ST-LINK/V2-1 Debugger in coherence with STM32 CubeMX IDE allows in-depth debugging and real-time task tracking, crucial for RTOS applications where timing and state consistency are critical. Serial Wire Debug (SWD) allows real-time variable inspection and FreeRTOS task state monitoring without halting the system.

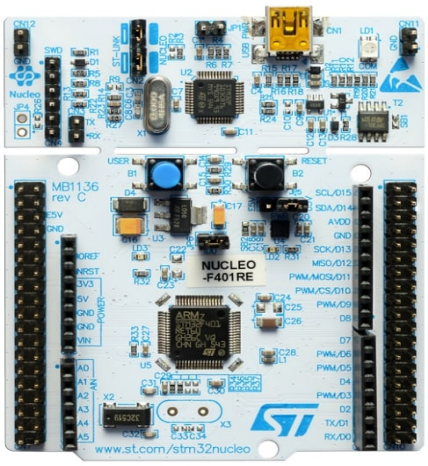


Fig.1. STM32 Nucleo F-446RE

2. RCWL-0516 Motion detection module:

The RCWL-0516 is a microwave radar motion sensor module operating on the principle of doppler effect. The module generates a signal (a series of harmonics) around frequency of 3.18GHz in a omnidirectional sense. A critical function of a doppler radar is to 'mix' the reflected signal with the transmitted signal to arrive at a frequency which is the difference between the transmitted and reflected signal. It gives a logic HIGH (1) or logic LOW (0) as output based on the motion in and around the field of view of the module.

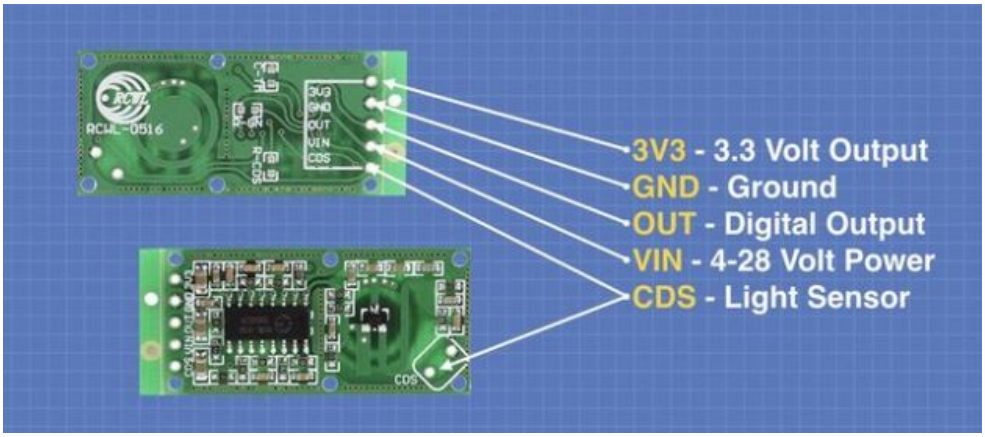


Fig.2. RCWL-0516 Pinout

3. MQ135

The MQ-135 Gas sensor can detect gases like Ammonia (NH3), sulfur (S), Benzene (C6H6), CO2, and other harmful gases and smoke. Similar to other MQ series gas sensor, this sensor also has a digital and analog output pin. When the level of these gases go beyond a threshold limit in the air the digital pin goes high. This threshold value can be set by using the on-board potentiometer. The analog output pin, outputs an analog voltage which can be used to approximate the level of these gases in the atmosphere. Thes sensors has to be powered up for a pre-heat duration for the sensor to warm up before it can start working. This pre-heat time is normally between 30 seconds to a couple of minutes

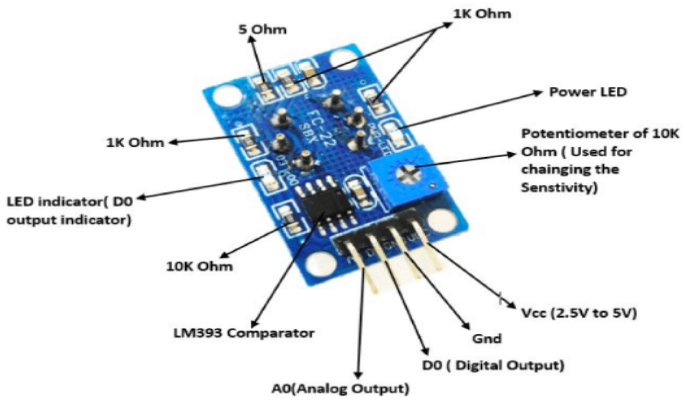


Fig.3. MQ135 Pinout

4. OLED SSD1306:

The **OLED SSD1306** is a widely-used 0.96-inch monochrome OLED display module that operates using the SSD1306 controller IC.

Key Features:

1. **Resolution:** 128x64 pixels.
2. **Display Type:** Monochrome (typically white, blue, or yellow).
3. **Controller:** SSD1306.
4. **Interface:** Supports I2C
5. **Power Supply:** Operates at 3.3V to 5V, though logic levels depend on the interface.
6. **Low Power Consumption:** Ideal for battery-powered devices.
7. **Compact Size:** Typically measures about 0.96 inches diagonally.

Fig.4. OLED Screen

Chapter 1: Proposed Solution and Working

The system uses the STM32 microcontroller to read data from sensors like AHT10, MQ135, and RCWL-0516 at regular intervals. Also the internal core temperature is recorded. Each sensor's data is processed and timestamped using the in-built RTC module, ensuring accuracy. The processed data is displayed on an OLED in real time and queued for logging. Finally, the system writes the data to an SD card securely using mutexes, ensuring reliable storage for analysis. Use of semaphores for utilizing interrupt on a switch in the FreeRTOS architecture.

1.1 Proposed Architecture

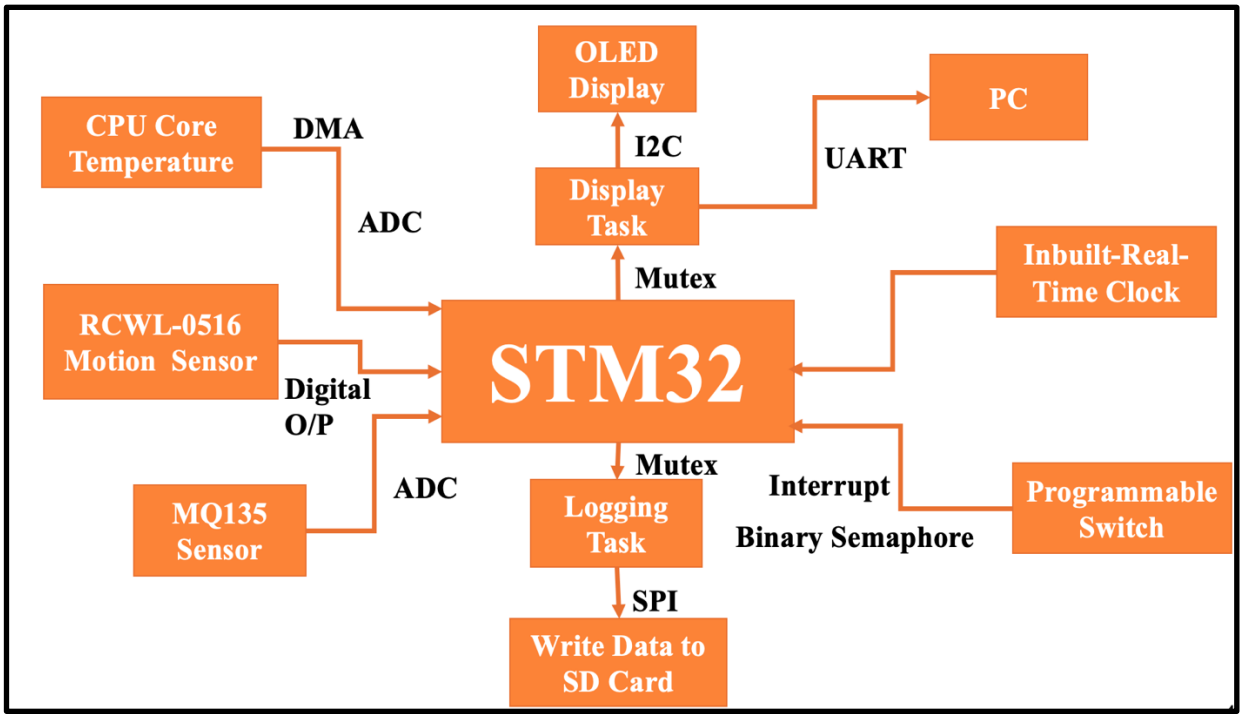


Fig. 5. Proposed architecture with protocols and various RTOS features

1.2 STM32 FreeRTOS Configurations

- Preemption TickRate: 1000Hz
- Minimum Stack Size: 128 words
- Total Heap size: 15360 bytes
- Memory Management Scheme: Heap4
- Trace Facility used for Stack analyser
- SYSCLK TickRate: 84Mhz
- UART baud rate: 115200
- SPI Rate: 10.5 Mbps
- I2C fast mode: 400Kbps
- ADC1 for DMA (Peripheral(switch) to memory)

Chapter 2: Proposed System

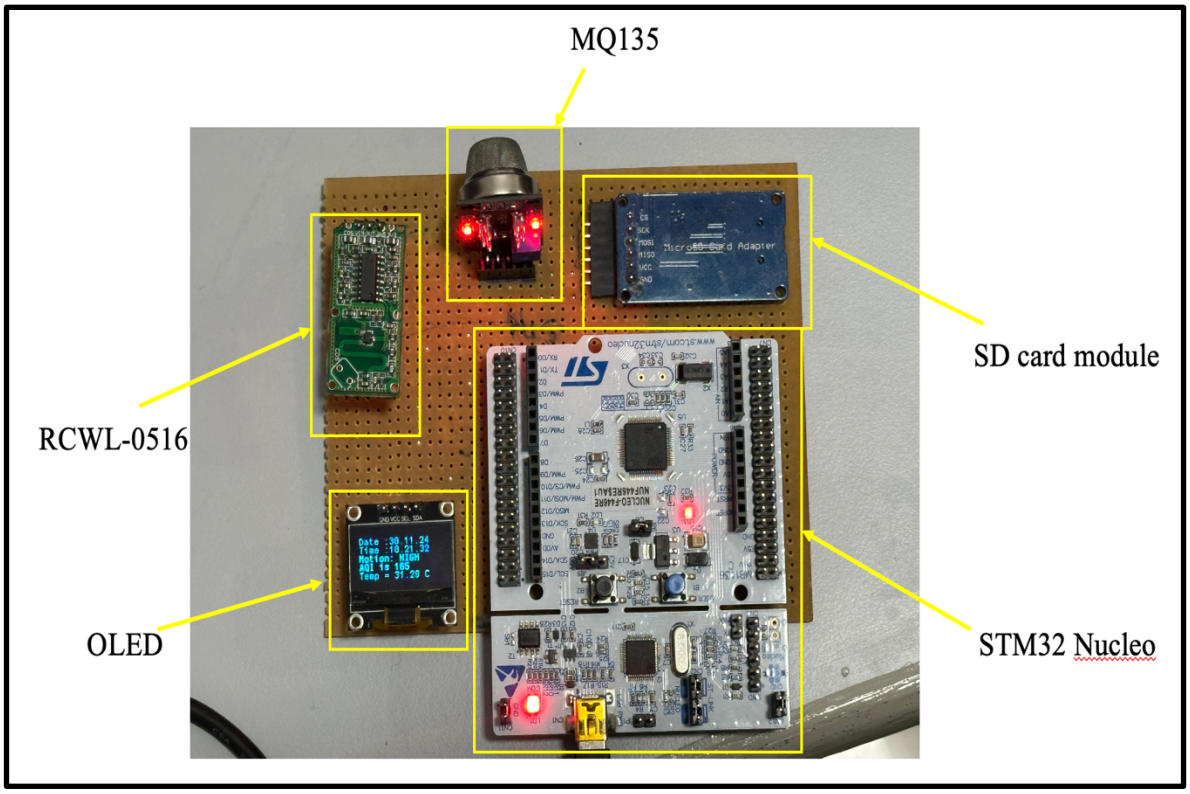


Fig.6. Hardware arrangement of proposed system

2.1 Mutexes and Semaphores

Mutex: Ensuring Safe Resource Access

OLED Display:

- A mutex ensures only one task can write to the OLED display at a time, preventing conflicts and data corruption.

SD Card:

- Same mutex synchronizes access to the SD card, ensuring that only one task writes data at a time to avoid collisions.
- Give and Take mechanism of mutex ensures that a resource such as the SD card and OLED is not simultaneously accessed by multiple tasks.

Binary Semaphore

Switch Task:

- Programmable switch task utilizes semaphore functionality to allow interrupt functionality with FreeRTOS architecture. The semaphore is preferred when utilizing the ISR functionality.

2.2 Pin Configuration

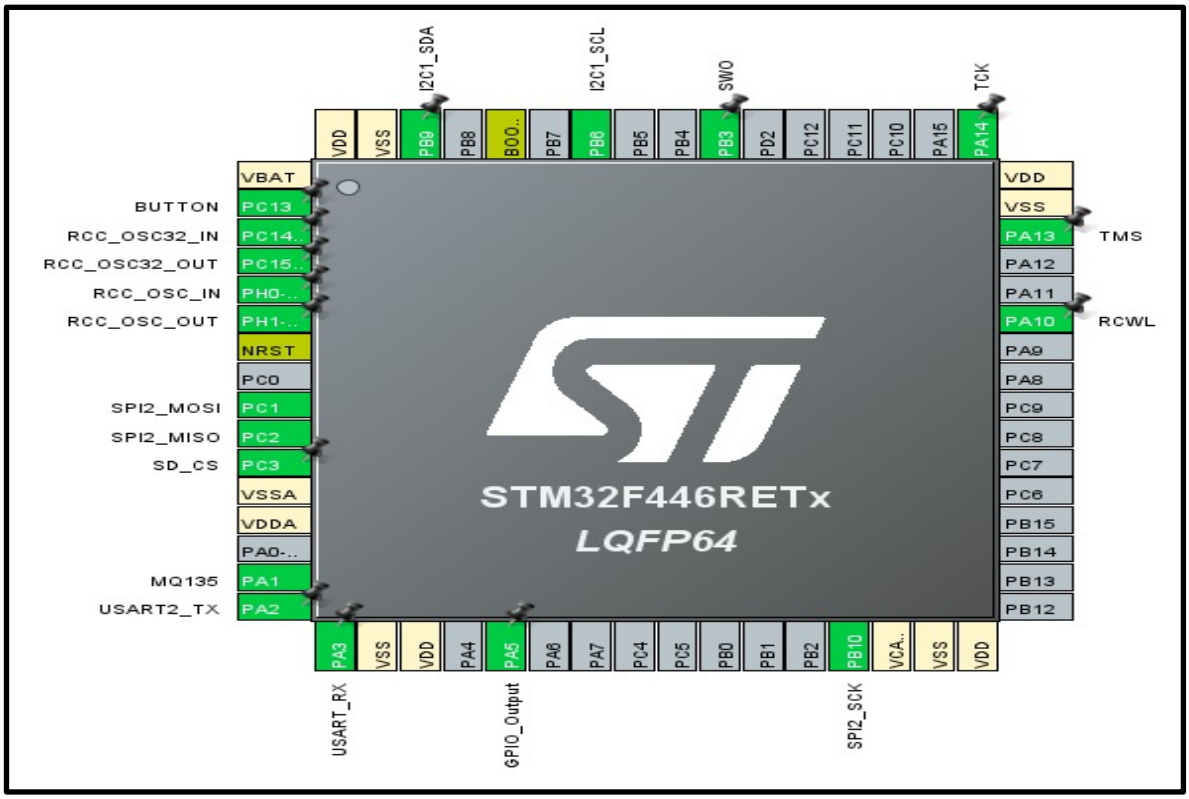


Fig.7. Pin Configuration in CubeMX IOC

2.3 Clock configuration:

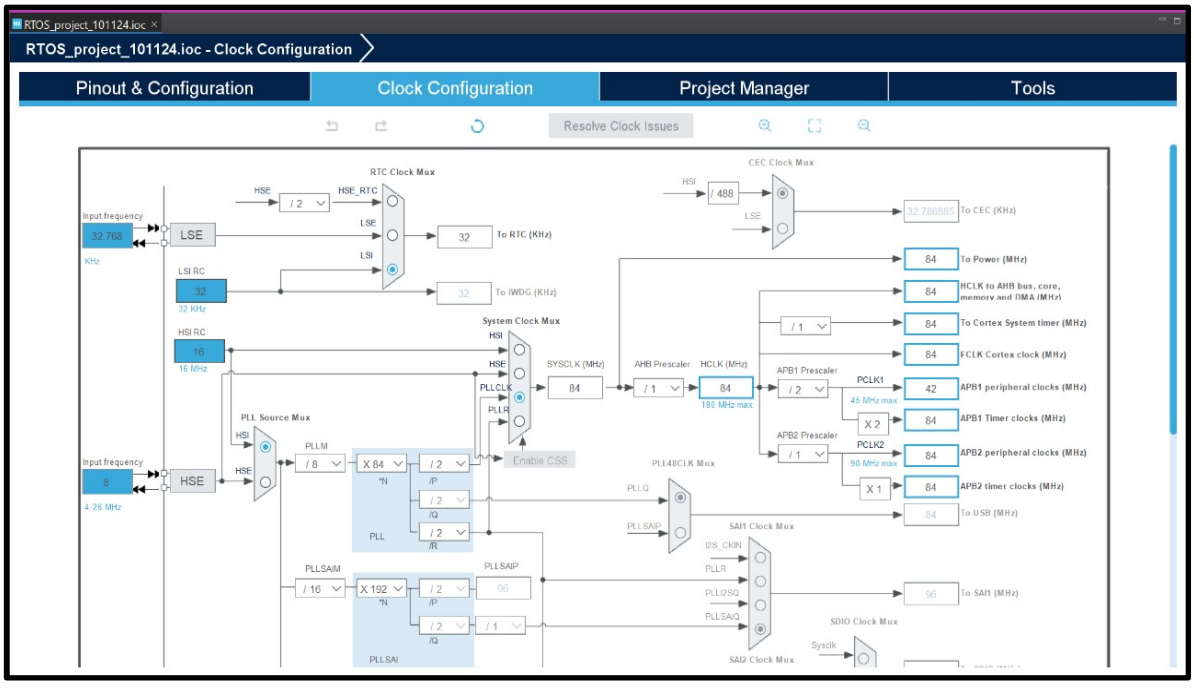


Fig.8. Clock Tree

2.4 Issues Encountered:

- Hardfault Issue: Our program entered hardfault due to memory overflow in the task which upon debugging was altered. Stack size details were analyzed using the CubeMX FreeRTOS task list, which was configured to monitor stack utilization and identify potential overflows or memory mismanagement.
- Priority of OLED task: The comprehensive breakpoint debugging approach allowed in better understanding the data corruption on screen, which was mainly due to simultaneous sending of data onto the OLED via various tasks. To avoid such corruption mutexes were used.

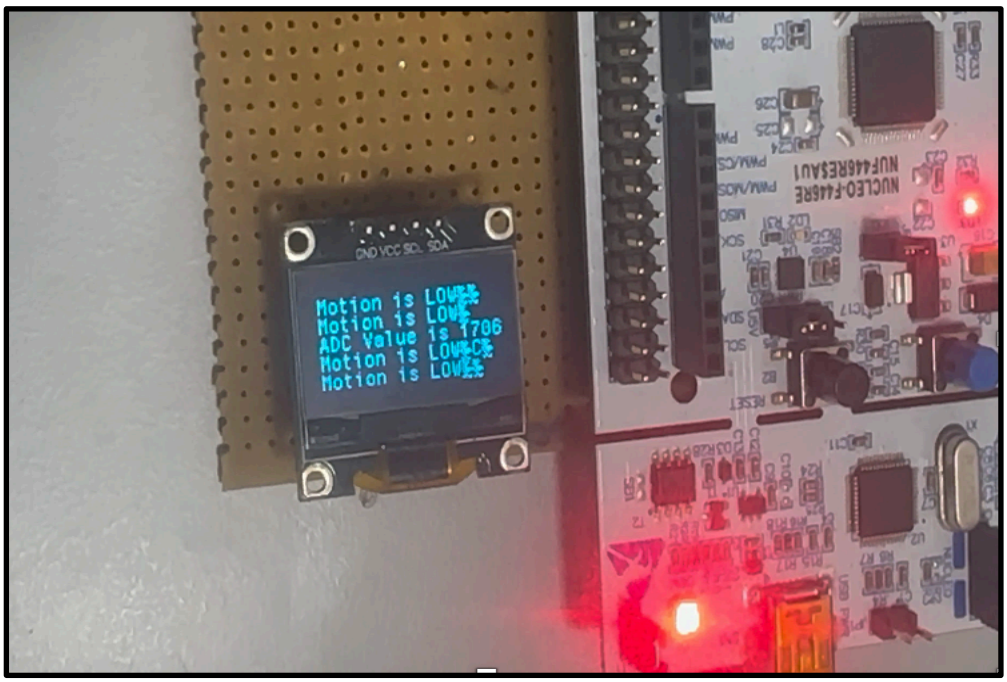
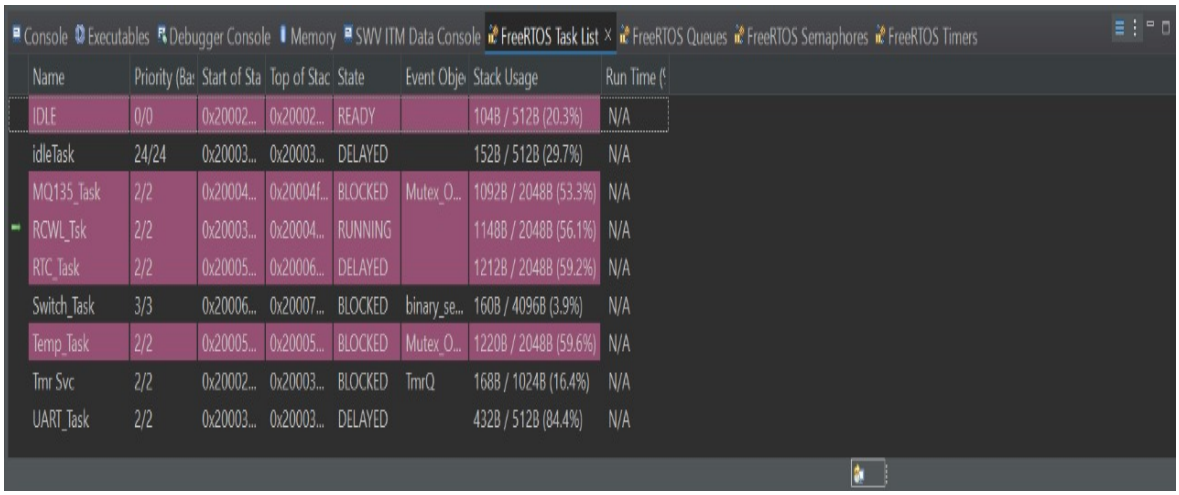


Fig.9. Data corruption on OLED

Chapter 3: Observations & Measurements

3.1 FreeRTOS Task list



Name	Priority (Ba	Start of Sta	Top of Stac	State	Event Obje	Stack Usage	Run Time (%
IDLE	0/0	0x20002...	0x20002...	READY		104B / 512B (20.3%)	N/A
idleTask	24/24	0x20003...	0x20003...	DELAYED		152B / 512B (29.7%)	N/A
MQ135_Task	2/2	0x20004...	0x20004f...	BLOCKED	Mutex_O...	1092B / 2048B (53.3%)	N/A
RCWL_Tsk	2/2	0x20003...	0x20004...	RUNNING		1148B / 2048B (56.1%)	N/A
RTC_Task	2/2	0x20005...	0x20006...	DELAYED		1212B / 2048B (59.2%)	N/A
Switch_Task	3/3	0x20006...	0x20007...	BLOCKED	binary_se...	160B / 4096B (3.9%)	N/A
Temp_Task	2/2	0x20005...	0x20005...	BLOCKED	Mutex_O...	1220B / 2048B (59.6%)	N/A
Tmr Svc	2/2	0x20002...	0x20003...	BLOCKED	TmrQ	168B / 1024B (16.4%)	N/A
UART_Task	2/2	0x20003...	0x20003...	DELAYED		432B / 512B (84.4%)	N/A

Fig.10. FreeRTOS Task Stack space utilization

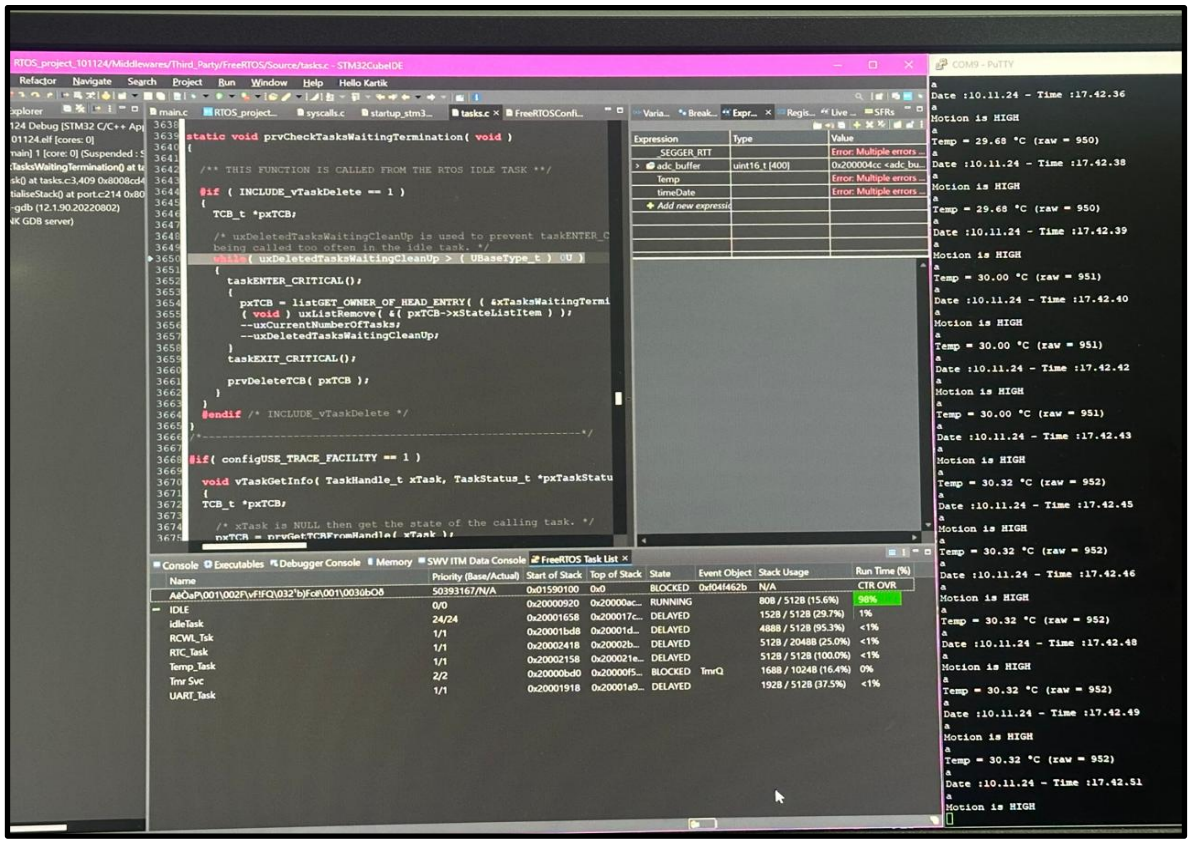


Fig.11. FreeRTOS Task Run time

3.2 FreeRTOS Semaphore list

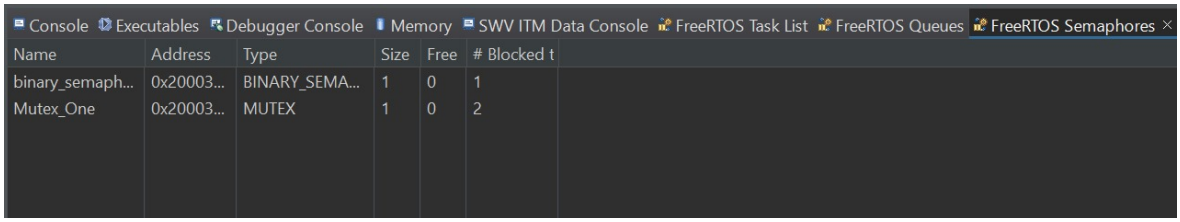


Fig.12. FreeRTOS Semaphore states

3.3 SPI Protocol

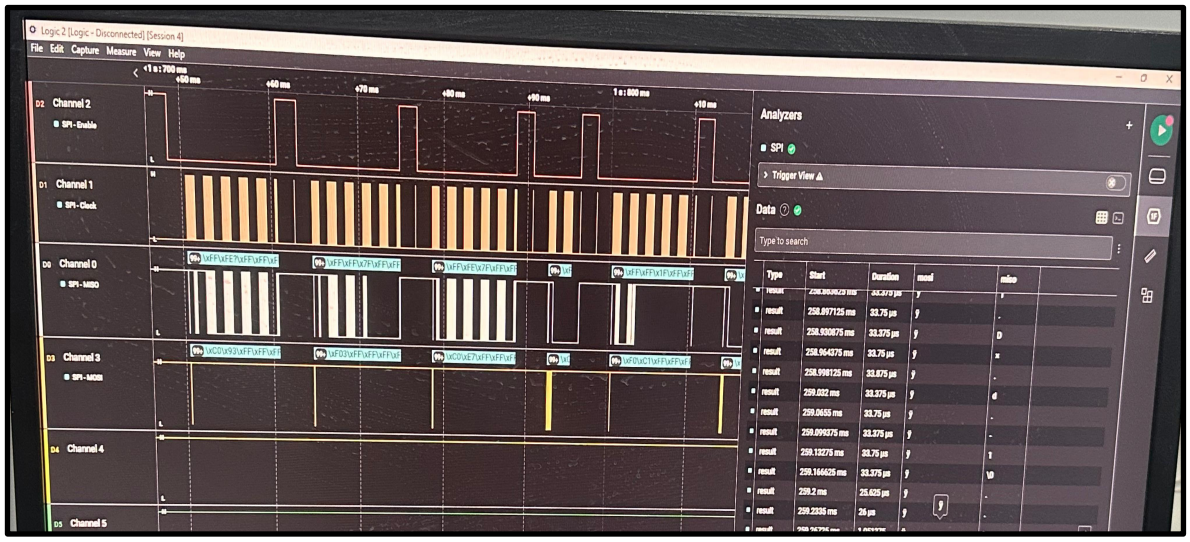


Fig.13. MOSI & MISO pins indicating working of SPI

3.4 UART Protocol

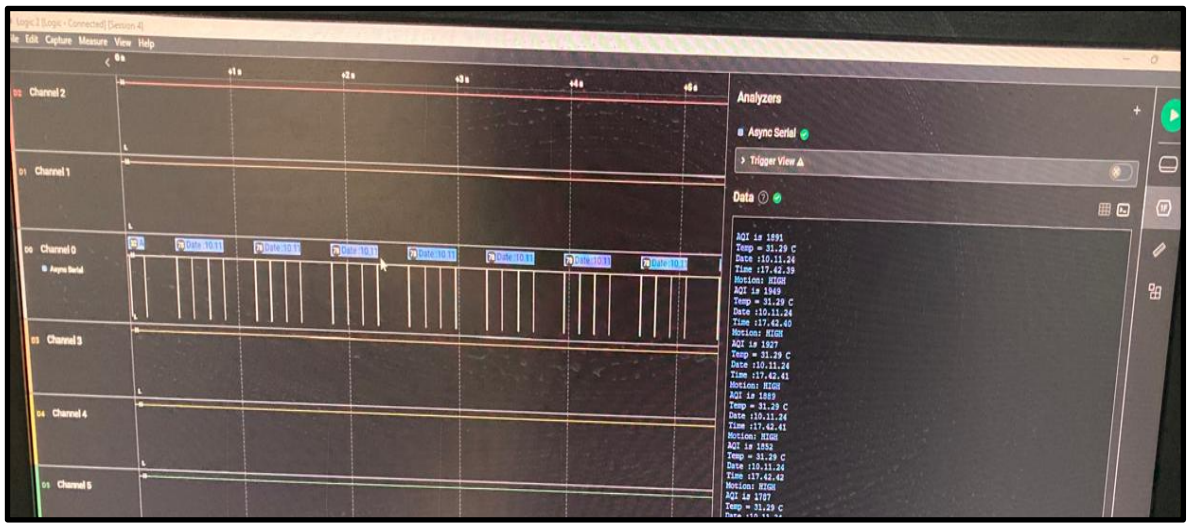


Fig.14. UART communication

3.5 Segger

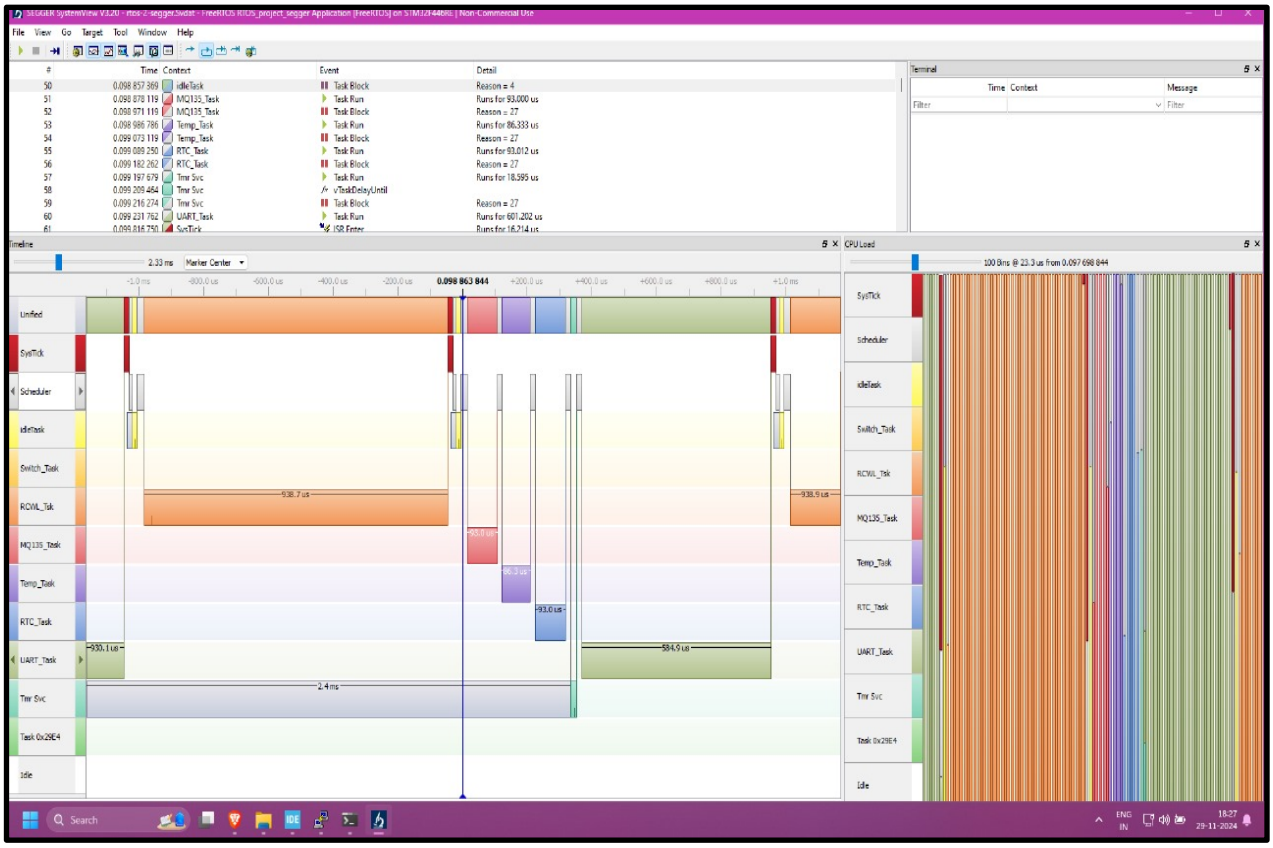


Fig.15. Segger snapshot indicating various tasks appearing with their time slices

3.6 SD Card data

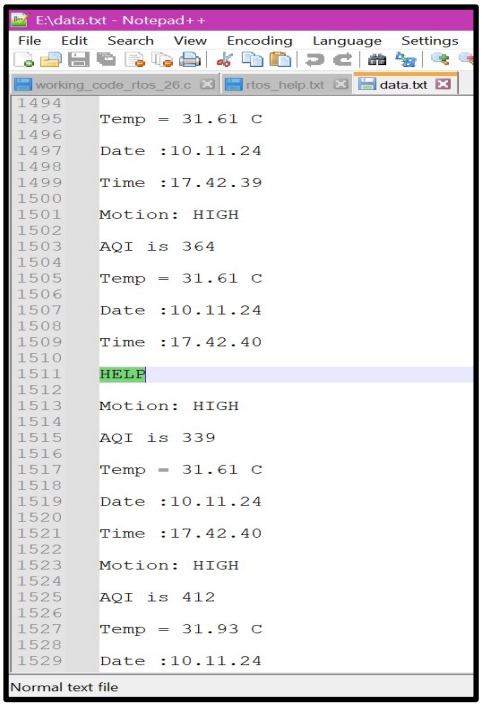


Fig.16. Text file recorded data on SD card

Chapter 4: Pseudo Code

Core Functions and Tasks:

1. **send_OLED(char string, uint8_t y):** This function displays a given string on the OLED screen at a specified row (y). It checks if the help flag is active to determine if updating the display is allowed. The SSD1306_GotoXY function sets the cursor, and the string is printed using SSD1306_Puts, followed by an OLED screen refresh using SSD1306_UpdateScreen.

2. **send_SDCard(char string):** This function appends a string to a file (data.txt) on the SD card using SPI. It increments a counter (count), opens the file in append mode, writes the string followed by a newline, and then closes the file.

3. **vUARTtask(void *pvParameters):** This task periodically increments a counter, formats it into a string, and sends it over UART every 50ms. The send_uart() function is used to transmit the data, and the task uses vTaskDelay() to control the task frequency.

4. **vSwitchtask(void *pvParameters):** This task waits for a button press signal using a semaphore (binarySemaphoreISR). When the button is pressed, it toggles the help state, logs the "HELP" message to the SD card, displays "HELP" on the OLED screen, and sends the event message over UART. After a small delay to allow for debouncing, the help state is restored.

5. **vInternalTemptask(void *pvParameters):** This task measures the internal temperature, formats it, and then logs/displays the temperature on the OLED screen, SD card, and UART. It uses a mutex (MutexOne) to protect shared resources, ensuring safe access to the display and file operations.

6. **vRCWLtask(void *pvParameters):** This task monitors the motion sensor (RCWL) and determines whether motion is detected. It then displays the motion state on the OLED screen, logs it to the SD card, and sends the state over UART.

7. **vRTCtask(void *pvParameters):** This task reads the current date and time from the RTC module, formats the data, and displays it on the OLED screen, logs it to the SD card, and sends it via UART every 10ms. The RTC time and date are retrieved using HAL_RTC_GetTime() and HAL_RTC_GetDate().

8. **vMQ135task(void *pvParameters):** This task reads the air quality index (AQI) value from the MQ135 sensor using ADC readings. It formats the AQI value and logs/displays it on the OLED, SD card, and UART.

Interrupt Handlers:

- **HAL_ADC_ConvHalfCpltCallback:** This callback is triggered when the first half of the ADC conversion buffer is filled. It processes the first half of the buffer to keep the system responsive.

- **HAL_ADC_ConvCpltCallback:** This callback is triggered when the second half of the ADC buffer is filled. It processes the second half of the ADC data to ensure continuous data handling.

- **HAL_GPIO_EXTI_Callback:** This callback is triggered by the external interrupt from the button press. It releases a semaphore (binarySemaphoreISR), signaling the vSwitchtask to process the button press.

Synchronization:

- **Semaphores:** Used for task synchronization, especially to ensure that only one task accesses shared resources at a time (e.g., OLED, SD card).

- **Mutexes:** Used for safe access to shared data and to prevent conflicts in tasks that handle

Project GitHub link: Data logger using FreeRTOS

Bibliography

[1] FreeRTOS Documentation

[2] Mastering FreeRTOS Real Time Kernel

[3] ST Microelectronics Community

[4] Snømann Ingeniør YT channel

[5] UM1724 User manual STM32 Nucleo-64 boards (MB1136)

[6] Nucleo F-446RE Datasheet