



General purpose data acquisition system using the STM32 microcontroller and FreeRTOS

LM118 – Bachelor of Engineering in
Electronic and Computer Engineering

Project Interim Report

Qinyuan Liu
20137095

Dr Ian Grout
28/Oct/2024

Abstract

Keywords: RTOS, STM32, Embedded system, data acquisition,

The rapid growth of the complexity of embedded systems has made precise, lightweight, agile data acquisition increasingly important, especially in areas like industrial automation and robotics. Using the STM32 microcontroller combined with FreeRTOS, the project focuses on creating a data acquisition system that will have analog and digital IO, with communication functionality.

STM32_CUB_IDE (C) and NUCLEO-L476RG (Testing platform) will be used for this project, along with Target 3001 Beta-Ver (Schematics and PCB layout).

The primary implementation of this project is the prototype system, which is expected to be done at the end of the autumn semester, The winter semester will be focused on PCB design,

The presentation and final report will be the focus of the final semester.

Declaration

This interim report is presented in part fulfilment of the requirements for the LM118 Bachelor of Engineering in Electronic and Computer Engineering **Bachelors Project**.

It is entirely my own work and has not been submitted to any other University or Higher Education Institution or for any other academic award within the University of Limerick.

Where there has been made use of work of other people it has been fully acknowledged and referenced.

Name

Qinyuan Liu

Signature



Date

28/ OCT/ 2024

Table of Contents

Table of Contents

ABSTRACT	I
DECLARATION	II
TABLE OF CONTENTS	III
CONTENTS	III
LIST OF FIGURES AND DRAWINGS.....	IV
LIST OF EQUATIONS.....	V
LIST OF TABLES.....	VI
CHAPTER 1 INTRODUCTION.....	- 1 -
CHAPTER 1 BACKGROUND	- 3 -
CHAPTER 2 TECHNICAL DETAILS - SOFTWARE	- 5 -
CHAPTER 3 TECHNICAL DETAILS – HARDWARE	- 15 -
CHAPTER 4 ACTION PLAN.....	- 16 -
CHAPTER 5 CONCLUSIONS AND FUTURE WORK.....	- 17 -
CHAPTER 6 REFERENCES	- 18 -
APPENDICES.....	- 20 -
APPENDIX A: PROJECT GANTT CHART	- 22 -
APPENDIX B: INTERIM PRESENTATION SLIDES	- 24 -
APPENDIX C: WEEKLY REPORTS.....	- 28 -
APPENDIX D: DRAWING	- 3 -
APPENDIX E: HEADER TABLE	- 1 -

List of Figures and Drawings

List of Figures:

Figure 1 NUCLEO-L476RG pin layout.....	- 2 -
Figure 2 ICO pin configuration of in the Cube IDE for the RTOS LED toggling demo.....	- 11 -
Figure 3 RTOS configuration in the Cube IDE	- 12 -
Figure 4 . RTOS Task Priority Diagram.....	- 13 -
Figure 5 Project Gantt Chart	- 22 -

List of Drawings:

1. General Data Acquisition System Block Diagram (Appendix D drawing 1)

- Title: General Data Acquisition System Block Diagram
- Author: Qinyuan Liu

2. RTOS Task Priority Diagram (Appendix D drawing 2)

- Title: RTOS Task Priority Diagram
- Author: Qinyuan Liu

3. ADC Module Schematic (Appendix D drawing 3)

- Title: ADC_Module
- Author: Qinyuan Liu

4. DAC Module Schematic (Appendix D drawing 4)

- Title: DAC_and_CS_Module
- Author: Qinyuan Liu

5. Digital Input Module Schematic (Appendix D drawing 5)

- Title: Digital_Input_Module
- Author: Qinyuan Liu

6. Digital Output Module Schematic (Appendix D drawing 6)

- Title: Digital_Output_Module
- Author: Qinyuan Liu

List of Equations

This project has not used any mathematical Equations at this stage.

Equation 1 Sample Equation - Fourier Transform.....v

Sample equation:

Equation 1 Sample Equation - Fourier Transform

$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t} dt$$

List of Tables

Table 1 List of Tables

Table Number	Caption	Page Number
Table 1	PIn_Config_Nuleo	Appendix E, Page 1 of 11
Table 2	DAC_And_CS_Header	Appendix E, Page 4 of 11
Table 3	DAC_OUTPUT	Appendix E, Page 5 of 11
Table 4	ADC_Header	Appendix E, Page 6 of 11
Table 5	ADC_Analog_Output	Appendix E, Page 7 of 11
Table 6	Digital_Output	Appendix E, Page 8 of 11
Table 7	Digital_OUT_Header	Appendix E, Page 9 of 11
Table 8	Digital_Input	Appendix E, Page 10 of 11
Table 9	Digital_In_Header	Appendix E, Page 11 of 11

Chapter 1 Introduction

RTOS has wide applications in embedded system development, from robotic control in the Mars lander to data applications in DSP processing. This system enables an ARM/AVR chip to perform many more complex tasks in a time-accurate task scheduling system. The goal of the project is to develop a general-purpose data acquisition system with the STM32, based on FreeRTOS. RTOS will provide a stable platform for control over timing and communication, and control over an array of ADC/DAC and 2 digital IO chips, handling communication with a PC and potential Peripherals.

As a low-speed general-purpose data acquisition system, this project has a wide range of applications in the industry. The project can be adapted to various uses in the industry, such as automation engineering, civil engineering, machinery, vehicle design, safety systems, and fields.

For example, this project can be put into the battery management function of the battery pack as an application. The ADC array provides the voltage reading of each battery, each DAC provides the control voltage for the power semiconductor of the charging circuit, the digital IO realizes a simple display battery status display, the PC serial port communication provides the error detection and control functions and calculates the battery algorithm based on this system to provide a highly integrated solution. In another example, in industrial vibration detection, many large rotating mechanical equipment require continuous vibration detection, such as wind turbines, thermal power turbines, and large mechanical engines. The resonant frequency of these systems is usually less than 1khz. This project can detect vibration signals through ADC sampling, use the relevant communication protocols to communicate with PLC, provide feedback loops, and realize the real-time monitoring and analysis of mechanical equipment.

The project will be implemented in 3 steps: a breadboard prototype for proof of concept, a PCB prototype for advanced design integration, and a final STM32 chip on board design. The breadboard prototype will be first made as a proof of concept that will provide a solid foundation for the following steps: Following the breadboard prototype, A PCB prototype that connects with the NUCLEO board will be developed as a demonstration of the more advanced design capability of the student, while also providing a better integration of the electronic design of this project. The final step will be the implementation of a full STM32

chip-on-board design, to elevate this project into an industry implementation-ready, advanced electronic design project.

The source of the entire project can be found at:

<https://github.com/Qinyuan72/FYP/tree/master>

[1]

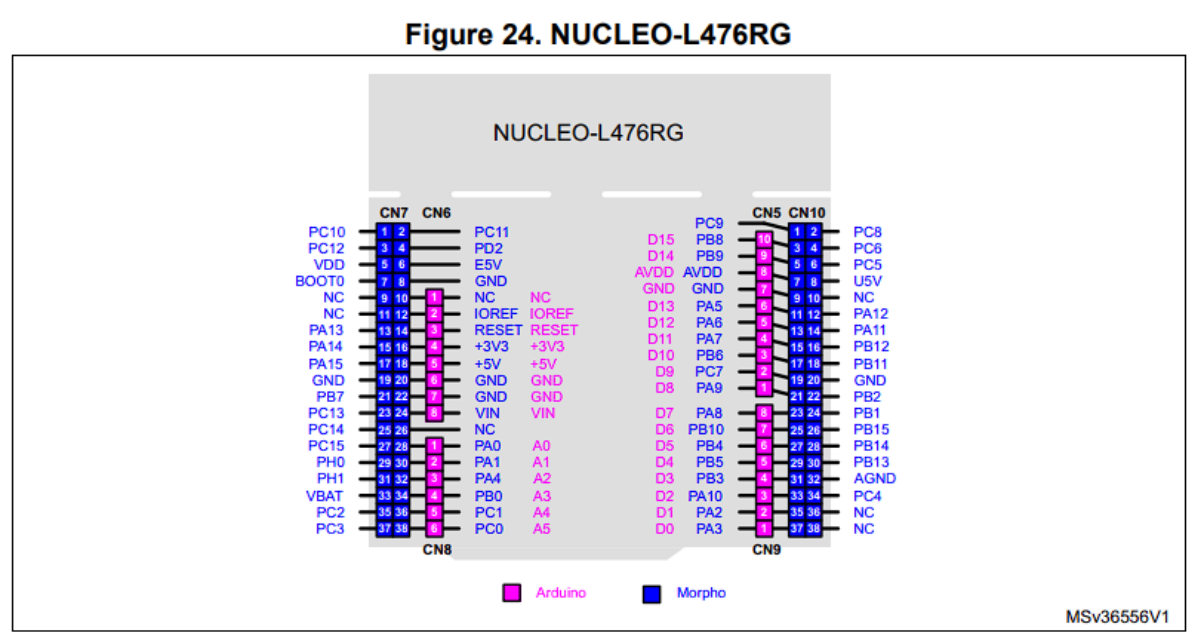


Figure 1 NUCLEO-L476RG pin layout

Chapter 1 Background

RTOS (Realtime operating system) for embedded systems was developed in the 80s with the advancement of microprocessors. The emergence of RTOS, in parallel with other non-time-sensitive systems, proved to be critical for embedded systems in subsequent decades of development. The time sensitivity of the RTOS provides a foundational element for most embedded system designs. With the growing complexity of embedded systems, more dedicated RTOS platforms appeared. In the early 90s, RTOS grew from a core kernel environment into a mature development platform [2].

RTEMS (Real-Time Executive for Missile Systems) is one of the first mature RTOS to emerge in the military industry. It was developed around the late 80s and provided support for the ADA programming language. It is still widely used today [3].

In the 1990s, the development of more powerful microprocessors and the growth of the internet led to the development of more sophisticated RTOS, such as VxWorks and QNX. These systems were designed to handle more complex real-time applications, such as telecommunications, network routing, and multimedia [2].

RTOS today has wide application in embedded systems, with great promotion and adaptability. Its applications involve aerospace, defense, automobile, communication, and multimedia [2].

In our application, we choose CMSIS-RTOS V2 provided by STM32_Cube_IDE as our RTOS platform and develop it on its basis. CMSIS-RTOS V2 provides generic RTOS interfaces for Arm® Cortex® processor-based devices. It offers a standardized API for software components requiring RTOS functionality, and it's well-integrated into our IDE [4].

RTOS is used in this project for two reasons: system time sensitivity and system complexity. As a general-purpose data acquisition system working at a 1kHz scanning rate, the project needs to solve the problem of "jitter" in digital signal processing (DSP). A key characteristic of an RTOS is its consistency in the amount of time it takes to accept and complete an application's task; this variability is known as "jitter." The rigid RTOS scheduling provides an ideal solution for the common problem of jitter in DSP sampling. In terms of system

complexity, this project needs to implement 16 ADC and 16 DAC analog signal inputs, as well as 16 * 16 digital signal I/O. Additionally, USART communication and control functions must be implemented. Such complexity requires a rigorous multi-level priority task management system. Therefore, RTOS's multitasking scheduling, timer management, interrupt management, and inter-task communication and synchronization provide a suitable platform [4].

This project is developed based on STM32, a common microcontroller architecture widely used in embedded development. We use the STM32-Nucleo development board as an experimental platform and the mainstream STM32Cube-IDE for development [1]. The project will also attempt to implement PCB printing technology using Target 3001 to further advance students' capabilities in electronic drawing specifications and PCB design.

Chapter 2 Technical details - Software

STM32 Background and Hardware Configuration

Introduction to STM32 Cube:

The STM32Cube IDE (Cube IDE) is a complete development platform, Cube IDE offers intuitive hardware configuration in the ‘ICO’ interface and generates a skeleton code using the ICO configuration for the device. The Cube IDE used a C-based scripting language, offering HAL (hardware abstraction layers) to simplify hardware interaction by providing an abstraction layer between the application and microcontroller peripherals, which greatly reduced the complexity of the software design of this project.

STM32 and RTOS Integration:

The STM32Cube IDE integrated CMSIS-RTOS in its environment, the project uses the CMSIS-RTOS V2 distribution that’s offered in the IDE as the on-chip operating system of this project. The project uses a 2-tier priority system in the RTOS task priority configuration.

The syntax of the RTOS is relatively intuitive, RTOS skeleton code is generated by the STM32_cube_IDE, and it follows a similar syntax as the interrupt function (ISR) offered by HAL. For instance, a task-creating function can be declared inside #user_code_x, The code snippet below demonstrates how RTOS is configured within the *main.c* file to achieve LED toggling, with the highlighted section showcasing the user-defined RTOS code. The RTOS configuration and Pin layout in the Cube IDE are attached to the end of the snippet as well.

```
/* USER CODE BEGIN Header */
/**
 * *****
 * @file      : main.c
 * @brief     : Main program body
 * *****
 * @attention
 *
 * Copyright (c) 2024 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *
 * *****
 */
/* USER CODE END Header */
/* Includes -----*/
```

```

#include "main.h"
#include "cmsis_os.h"

/* Private includes ----- */
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private typedef ----- */
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ----- */
/* USER CODE BEGIN PD */

/* USER CODE END PD */

/* Private macro ----- */
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ----- */
/* Definitions for defaultTask */
osThreadId_t defaultTaskHandle;
const osThreadAttr_t defaultTask_attributes = {
    .name = "defaultTask",
    .stack_size = 128 * 4,
    .priority = (osPriority_t) osPriorityNormal,
};
/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes ----- */
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
void StartDefaultTask(void *argument);

/* USER CODE BEGIN PFP */
void vTaskLed_0 (void *pvParameters);

void vTaskLed_1 (void *pvParameters);
/* USER CODE END PFP */

/* Private user code ----- */
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/**
 * @brief The application entry point.
 * @retval int
 */
int main(void)
{

/* USER CODE BEGIN 1 */

```

```

/* USER CODE END 1 */

/* MCU Configuration-----*/

/* Reset of all peripherals, Initializes the Flash interface and the Systick. */
HAL_Init();

/* USER CODE BEGIN Init */

/* USER CODE END Init */

/* Configure the system clock */
SystemClock_Config();

/* USER CODE BEGIN SysInit */

/* USER CODE END SysInit */

/* Initialize all configured peripherals */
MX_GPIO_Init();
/* USER CODE BEGIN 2 */

/* USER CODE END 2 */

/* Init scheduler */
osKernelInitialize();

/* USER CODE BEGIN RTOS_MUTEX */
/* add mutexes, ... */
/* USER CODE END RTOS_MUTEX */

/* USER CODE BEGIN RTOS_SEMAPHORES */
/* add semaphores, ... */
/* USER CODE END RTOS_SEMAPHORES */

/* USER CODE BEGIN RTOS_TIMERS */
/* start timers, add new ones, ... */
/* USER CODE END RTOS_TIMERS */

/* USER CODE BEGIN RTOS_QUEUES */
/* add queues, ... */
/* USER CODE END RTOS_QUEUES */

/* Create the thread(s) */
/* creation of defaultTask */
defaultTaskHandle = osThreadNew(StartDefaultTask, NULL, &defaultTask_attributes);

/* USER CODE BEGIN RTOS_THREADS */
/* add threads, ... */
//Create task to manipulate LED_0
if((xTaskCreate(vTaskLed_0, "Task LED
0", configMINIMAL_STACK_SIZE, NULL, 1, NULL)) != pdTRUE)
{
}
//Create task to manipulate LED_1
if((xTaskCreate(vTaskLed_1, "Task LED
1", configMINIMAL_STACK_SIZE, NULL, 1, NULL)) != pdTRUE)
{
}

```

```

}
/* USER CODE END RTOS_THREADS */

/* USER CODE BEGIN RTOS_EVENTS */
/* add events, ... */
/* USER CODE END RTOS_EVENTS */

/* Start scheduler */
osKernelStart();

/* We should never get here as control is now taken by the scheduler */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
/* USER CODE END WHILE */

/* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
}

/**
 * @brief System Clock Configuration
 * @retval None
 */
void SystemClock_Config(void)
{
    RCC_OscInitTypeDef RCC_OscInitStruct = {0};
    RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

    /** Configure the main internal regulator output voltage
     */
    if (HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1) != HAL_OK)
    {
        Error_Handler();
    }

    /** Initializes the RCC Oscillators according to the specified parameters
     * in the RCC_OscInitTypeDef structure.
     */
    RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_MSI;
    RCC_OscInitStruct.MSISState = RCC_MSI_ON;
    RCC_OscInitStruct.MSICalibrationValue = 0;
    RCC_OscInitStruct.MSIClockRange = RCC_MSIRANGE_6;
    RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
    RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_MSI;
    RCC_OscInitStruct.PLL.PLLM = 1;
    RCC_OscInitStruct.PLL.PLLN = 40;
    RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV7;
    RCC_OscInitStruct.PLL.PLLQ = RCC_PLLQ_DIV2;
    RCC_OscInitStruct.PLL.PLLR = RCC_PLLR_DIV2;
    if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
    {
        Error_Handler();
    }

    /** Initializes the CPU, AHB and APB buses clocks

```

```

*/
RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                               |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_4) != HAL_OK)
{
    Error_Handler();
}
}

/**
 * @brief GPIO Initialization Function
 * @param None
 * @retval None
 */
static void MX_GPIO_Init(void)
{
    GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */

    /* GPIO Ports Clock Enable */
    __HAL_RCC_GPIOH_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();

    /*Configure GPIO pin Output Level */
    HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);

    /*Configure GPIO pin : LD2_Pin */
    GPIO_InitStruct.Pin = LD2_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);

/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}

/* USER CODE BEGIN 4 */
void vTaskLed_0 (void *pvParameters)
{
    //Variable Declaration

    //Infinite Loop
    for (;;)
    {
        HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);

        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
    //Delete this task if break out the loop
    vTaskDelete(NULL);
}
void vTaskLed_1 (void *pvParameters)
{

```



```

//Variable Declaration
//Infinite Loop
for (;;)
{
    //HAL_GPIO_TogglePin(User_LED_GPIO_Port, LED_1_Pin);

    //vTaskDelay(1000 / portTick_PERIODS_MS);
}
//Delete this task if break out the loop
vTaskDelete(NULL);
}
/* USER CODE END 4 */

/* USER CODE BEGIN Header_StartDefaultTask */
/**
 * @brief Function implementing the defaultTask thread.
 * @param argument: Not used
 * @retval None
 */
/* USER CODE END Header_StartDefaultTask */
void StartDefaultTask(void *argument)
{
    /* USER CODE BEGIN 5 */
    /* Infinite loop */
    for(;;)
    {
        osDelay(1);
    }
    /* USER CODE END 5 */
}

/**
 * @brief Period elapsed callback in non blocking mode
 * @note This function is called when TIM6 interrupt took place, inside
 * HAL_TIM_IRQHandler(). It makes a direct call to HAL_IncTick() to increment
 * a global variable "uwTick" used as application time base.
 * @param htim : TIM handle
 * @retval None
 */
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    /* USER CODE BEGIN Callback 0 */

    /* USER CODE END Callback 0 */
    if (htim->Instance == TIM6) {
        HAL_IncTick();
    }
    /* USER CODE BEGIN Callback 1 */

    /* USER CODE END Callback 1 */
}

/**
 * @brief This function is executed in case of error occurrence.
 * @retval None
 */
void Error_Handler(void)
{
    /* USER CODE BEGIN Error_Handler_Debug */

```

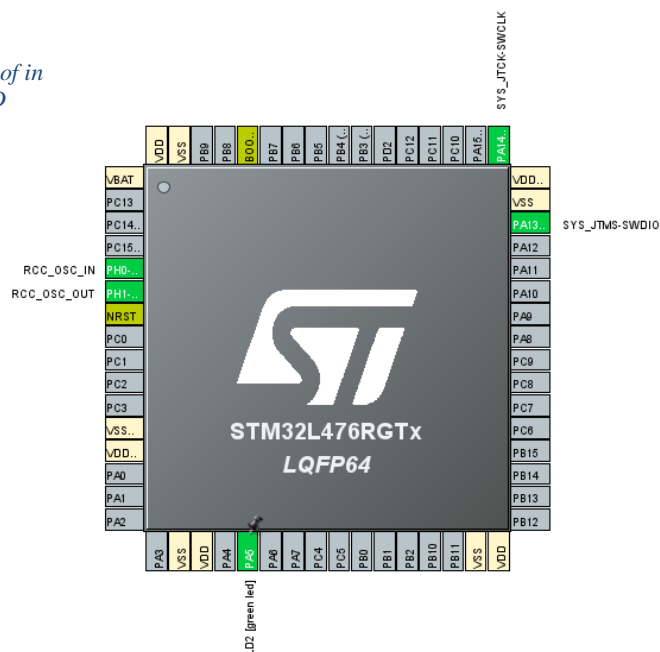
```

/* User can add his own implementation to report the HAL error return state */
__disable_irq();
while (1)
{
}
/* USER CODE END Error_Handler_Debug */
}

#ifdef USE_FULL_ASSERT
/**
 * @brief Reports the name of the source file and the source line number
 * where the assert_param error has occurred.
 * @param file: pointer to the source file name
 * @param line: assert_param error line source number
 * @retval None
 */
void assert_failed(uint8_t *file, uint32_t line)
{
/* USER CODE BEGIN 6 */
/* User can add his own implementation to report the file name and line number,
ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
/* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */

```

Figure 2 ICO pin configuration of in the Cube IDE for the RTOS LED toggling demo



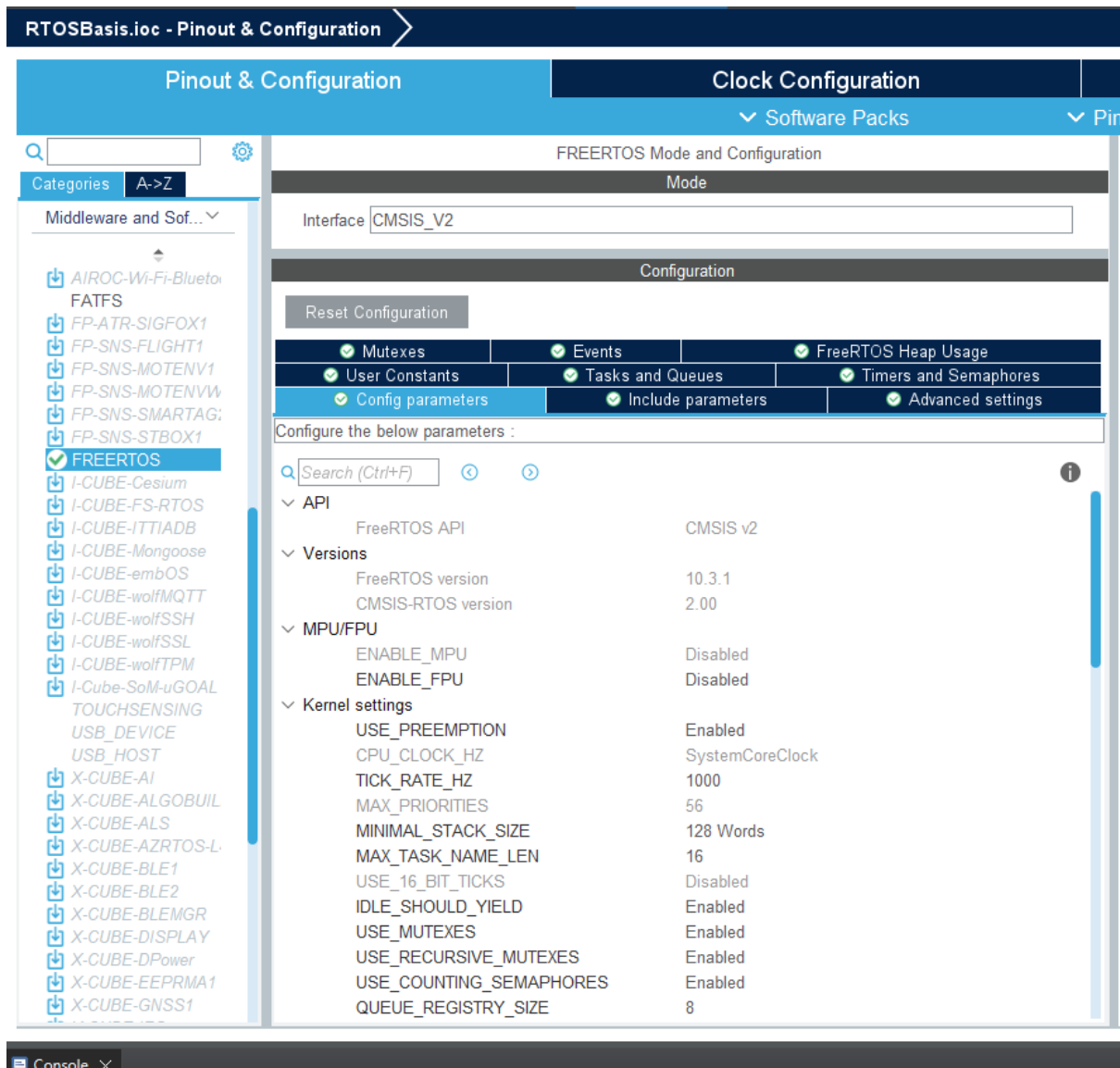


Figure 3 RTOS configuration in the Cube IDE

I plan to use JSON for both synchronized serial communication with the computer and internal data transfer within the RTOS.

The following code is the important function that will be used in the next stage for the Serial communication function.

```
xQueueSend (xQueue, &data, portMAX_DELAY);
```

This will be implemented in week 8 & 9.

Task Priority in RTOS:

The full Task Priority Diagram drawing is shown in Appendicitis C: 2. RTOS Task Priority Diagram

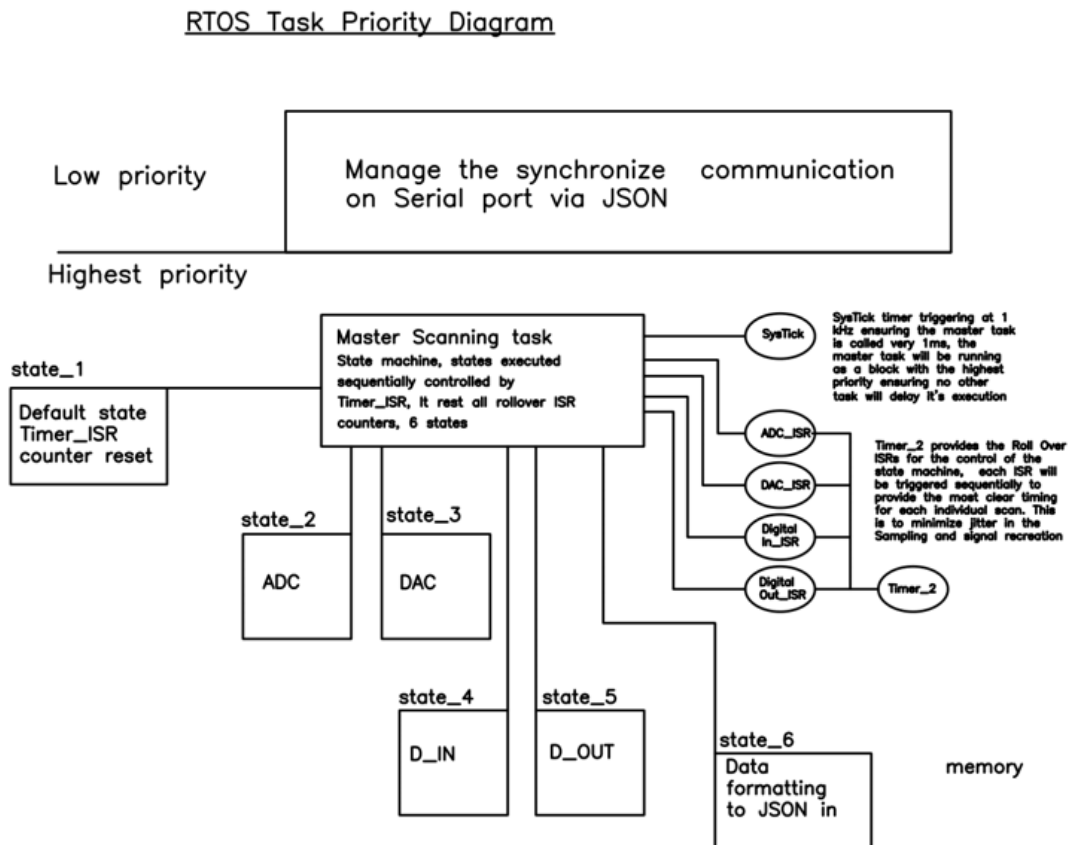


Figure 4 . RTOS Task Priority Diagram

A correct and well-evaluated Task priority setup is critical in an RTOS application, a miss-prioritized task can be critical in the success of a mission. For instance, the Mars lander that was introduced in the background section had the communication task settled in a way that would interfere with the data acquisition task, leaving the lander with a mission-critical problem that must be addressed remotely. [5]

In our application, a 2-level priority is defined: the highest priority Master Scanning task, and the low priority data management and Serial communication task.

Low-Priority Tasks

In the low-priority tasks, it'll have 2 functionalities, it will collect the JSON format from the Master Scanning task and verify its integrity if the data is not corrupted. The data will be sent to the Queue for the communication task, the communication will be managing the serial

communication with the Python code on the PC, providing information for the Python script for data visualization. As well as collecting data from the user input from the Python script and setting up the JSON data in its input/ configuration section for user input purposes. The control data from user input will be set to the **fixed memory location** for the Master scanning Task's direct access. Additionally, for a specific application like the battery management system, a local algorithm could be implemented on this level overriding user input, to achieve an 'Offline' close-loop control system.

High-Priority Tasks: Master Scanning Task

The high-priority task level ensures that the task running on this level will not be interrupted in execution, ensuring timing accuracy to the clock ticking level. Master Scanning Task is controlled by the SysTick timer in 1kHz frequency, meaning it's invoked every 1 millisecond, The Master Scanning Task is essentially a **Finite State Machine (FSM)**, the state of the Master Scanning task FSM is controlled by an internal flag and 4 timers_ISR interrupts. When SysTick is invoked, the FSM will be set to the default state which will reset all the ISR_timer reset the scanning status flags and wait for the first ISR to trigger the FSM to move to the ADC_scanning state, the ADC_scanning state will scan all 2 ADC chips, and save the data into dedicated fixed memory locations. A setup with a similar structure to the ADC is configured for the rest of the chip scanning task. After state 5 (Digital_Out), state 6 data formatting will read the data from the memory location of the privies scanning task and format the data into a JSON string in the dedicated memory location for the low-priority task to access.

Purpose of Configuring the Master Scanning Task as an FSM controlled by ISR

The purpose of configuring the master task as a timer rollover ISR is to provide accurate timing for the Data acquisition system, especially in minimizing jitter in the analog IO, meanwhile provides scalability for explanation as the timer rollover can be easily extended to accommodate more data channels (additional chips).

Chapter 3 Technical details – Hardware

Refers to the drawing in Appendicitis C drawing 1, 3, 4, 5, 6; Appendix E: Header Table

The Hardware prototype is defined by the diagram and the pin header table in the appendicitis. The hardware design consists of three stages:

1. Prototype design.
2. Nucleo board shield.
3. STM-32 onboard integrated system.

Currently. The drawing and pin layout of the hardware prototype design is finished. The prototype drawing provides a solid foundation for developing the physical prototype breadboard, which is trivial for validation of the first stage, the following hardware design will be based on the prototype design and will adapt to the stages' PCB design specification.

Data Acquisition System Modules: (Master Scanning FSM controlled)

In the current design. 4 module is developed for the data acquisition system.

- | | |
|--|--------------------------------|
| 1. DAC module (MCP3008 * 2) | -- Appendix D drawing 4 |
| 2. ADC module (MCP4822 * 8 and 74HC595 as CS) | -- Appendix D drawing 3 |
| 3. Digital Input (74HC165 * 2) | -- Appendix D drawing 5 |
| 4. Digital Output (74HC595 * 2) | -- Appendix D drawing 6 |

The Overview of the Prototype Hardware Configuration:

Appendix D, Drawing 1: *General Data Acquisition System Block Diagram*, illustrates the layout of the project hardware. The project consists of four custom-designed modules to achieve 16×16 analog I/O and 16×16 digital I/O. The selection of chips was made in consultation with Dr. Grout, the project supervisor, and the schematics were thoroughly reviewed and iterated from weeks 4 to 7.

The components have arrived, and all modules will be tested on a breadboard during weeks 8 and 9 for further refinement. The pin configuration details are provided in Appendix E: *Header Table*.

Chapter 4 Action plan

The current development done is as follows: The preliminary work is done in the first 2 weeks, to ensure that the development is correctly installed, and the key concepts have been discussed with the supervisor. Week 2 to week 5 initiates the implementation, including using STM32_CUBE_IDE to familiarize with freeRTOS, produce schematics with Target 3001, and have the component (see schematic) and the NUCLAE0-L476RG ready. And week 5-7 is spent on preparing the intermedia report and finishing the first schematic.

The development will take place over two semesters, using the STM32_CUBE_IDE (C), NUCLEO-L476RG (for testing), and Target 3001 Beta Version for schematics and PCB layout. The initial semester will focus on coding and prototyping, with any additional workload being addressed in the winter semester. The Prototype Breadboard and coding are aimed to be finished by the end of the autumn semester, and the first PCB design is to be sent for printing by the end of the fall. The following winter break will be focused on implementing the semester 2 mentioned above. The final semester will involve system verification and ensuring the project meets industry-level standards for robustness and versatility. In parallel, the presentation of the project will be a key focus, as clear technical communication is crucial both for grading and professional success.

Chapter 5 Conclusions and future work

Significant progress has been made in understanding and developing the foundations of the project, a schematic has been made while the breadboard testing prototype is under development.

The schematic design helped with the decision of chip selection and component wiring, which is critical for prototype design. The breadboard prototype, once complete, will help provide the test bench for the STM32 RTOS implementation, RTOS implementation will be challenging. The above is the current work content. In the next winter vacation, if the prototype is perfected smoothly, we will carry out the PCB design work, which involves schematic to PCB layout, PCB printing, and assembly, among which PCB design will be the most challenging part of the project. If the PCB design goes well and there is time, the PCB design of the onboard STM32 will be carried out during the winter vacation and the first to third weeks of the spring semester.

Chapter 6 References

- [1] STMicroelectronics, "NUCLEO-XXXXCX, NUCLEO-XXXXRX, NUCLEO-XXXXRX-P, NUCLEO-XXXXRX-Q: STM32 Nucleo-64 boards," Data brief, Rev 19, June 2024. Available: <https://www.st.com>
- [2] IntechHouse, "Real-Time Operating System in Embedded Systems," IntechHouse Blog, June 14, 2023. [Online]. Available: <https://intechhouse.com/blog/real-time-operating-system-im-embedded-systems/>. [Accessed: 20-Oct-2024].
- [3] RTEMS Project. *RTEMS 5.1 Documentation* - Doxygen Documentation. <https://ftp.rtems.org/pub/rtems/releases/5/5.1/docs/html/doxygen/RTEMSPreface.html> (Accessed October 28, 2024).
- [4] ARM CMSIS Project. *CMSIS-RTOS V2 API Documentation*. https://arm-software.github.io/CMSIS_5/RTOS2/html/index.html (Accessed October 28, 2024).
- [5] K. Wahome, "The first bug on Mars: OS scheduling, priority inversion, and the Mars Pathfinder," *Medium*, 16-Aug-2020. [Online]. Available: <https://kwahome.medium.com/the-first-bug-on-mars-os-scheduling-priority-inversion-and-the-mars-pathfinder-53586a631525>. [Accessed: 28-Oct-2024].

Additional Acknowledgment

Special thanks to my dad, who provided valuable insight into engineering drawings and introduced me to the world of engineering. A heartfelt thank you to my mom for her immense support during my injury while I was working on this project. I am also deeply grateful to all my friends who helped take care of me during this difficult time. Finally, I extend my gratitude to the dep of ECE for their comprehensive support throughout this project.

Appendices

Appendix A shows the Gantt chart.

Appendix B shows the Interim presentation slides.

Appendix C shows the weekly reports in its original email - communication form.

Appendix D shows the Drawings in its Print export form.

Appendix E shows the Header Table.

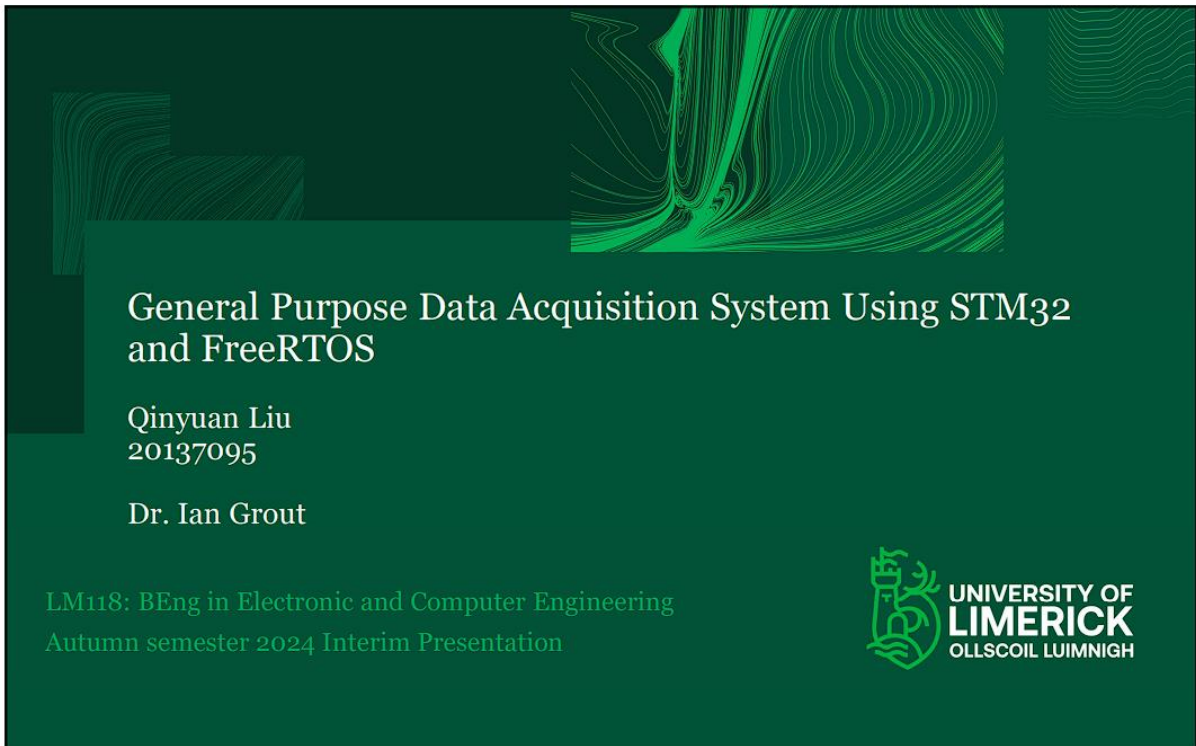
Appendix A: Project Gantt chart

FYP_Qinyuans_Project_implmentaion_Gantt_Chart																
Sem 1		Week														
Task no.	Task	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	Ground work (Background research and learning)	0	0	0	0	0										END
2	Interim Report	-	-	-	-	0	0	0	Dead Line Missed	Behind schedule!!	-	-	-	-	Exam	Exam
3	Schematic Design	-	-	-	-	0	0	0	0	DONE	-	-	-	-	Exam	Exam
4	Autumn Semester Presentation	-	-	-	-	0	0	0	0	Due to injury, I can't spend too much time on it, but it's well	-	-	-	-	Exam	Exam
5	Code Development	-	-	-	-	-	-	-	-	0	0	0	0	0	Exam	Exam
6	PCB Layout	-	-	-	-	-	-	-	-	-	-	-	0	0	Exam	Exam
Winter Break		Week														
Task no.	Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	First PCB Schematic and Layout	Christmas	0	0	Order PCB		END									
2	PCB Testing					0	END									
Sem 2		Week														
Task no.	Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	Final Report Due & Presentation	0	0	0	0	0	0	Final Report Due &		END						
2	PCB with STM32 on board	0	Ord	-	-	-	-	-		END						
3	PCB Testing	-	-	0	0	System Verification	-	-		END						

Figure 5 Project Gantt Chart

https://github.com/Qinyuan72/FYP/blob/master/Weekly%20Report/FYP_Qinyuans_Project_implmentaion_Gantt_Chart.xlsx

Appendix B: Interim presentation slides




General Purpose Data Acquisition System Using STM32 and FreeRTOS

Qinyuan Liu
20137095

Dr. Ian Grout

LM118: BEng in Electronic and Computer Engineering
Autumn semester 2024 Interim Presentation



UNIVERSITY OF
LIMERICK
OLLSCOIL LUIMNIGH

1



Presentation Overview

- Project introduction and objectives
- Hardware System design.
- Software System design
- Gantt chart
- Conclusions and next steps

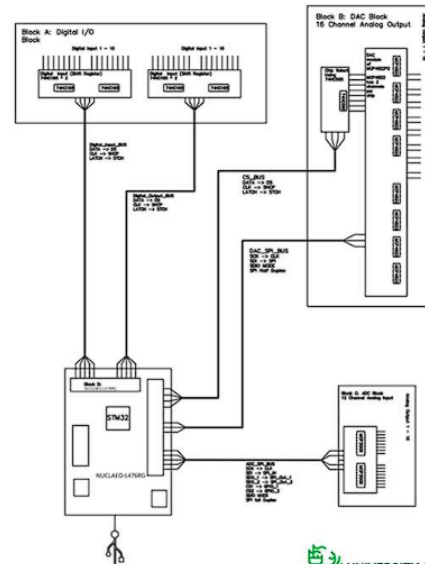


UNIVERSITY OF
LIMERICK
OLLSCOIL LUIMNIGH

2

Project Overview – Hardware

- The project consists of four custom-designed modules to achieve 16×16 analog I/O and 16×16 digital I/O.
- 1. DAC module (MCP3008 * 2)
- 2. ADC module (MCP4822 * 8 and 74HC595 as CS)
- 3. Digital Input (74HC165 * 2)
- 4. Digital Output (74HC595 * 2)
- Connected by header, planning on making 3 PCB board, A shield board, an analog board and a digital board, connected via a Flexible flat cable (FFC)

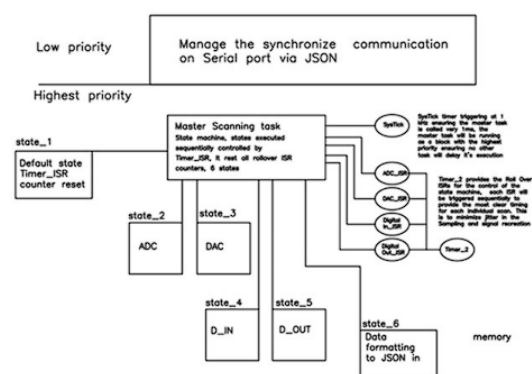


3

Project Overview Software

- In our application, a 2-level priority is defined: the highest priority Master Scanning task, and the low priority data management and Serial communication task.
- Purpose of Configuring the Master Scanning Task as an FSM controlled by ISR

RTOS Task Priority Diagram



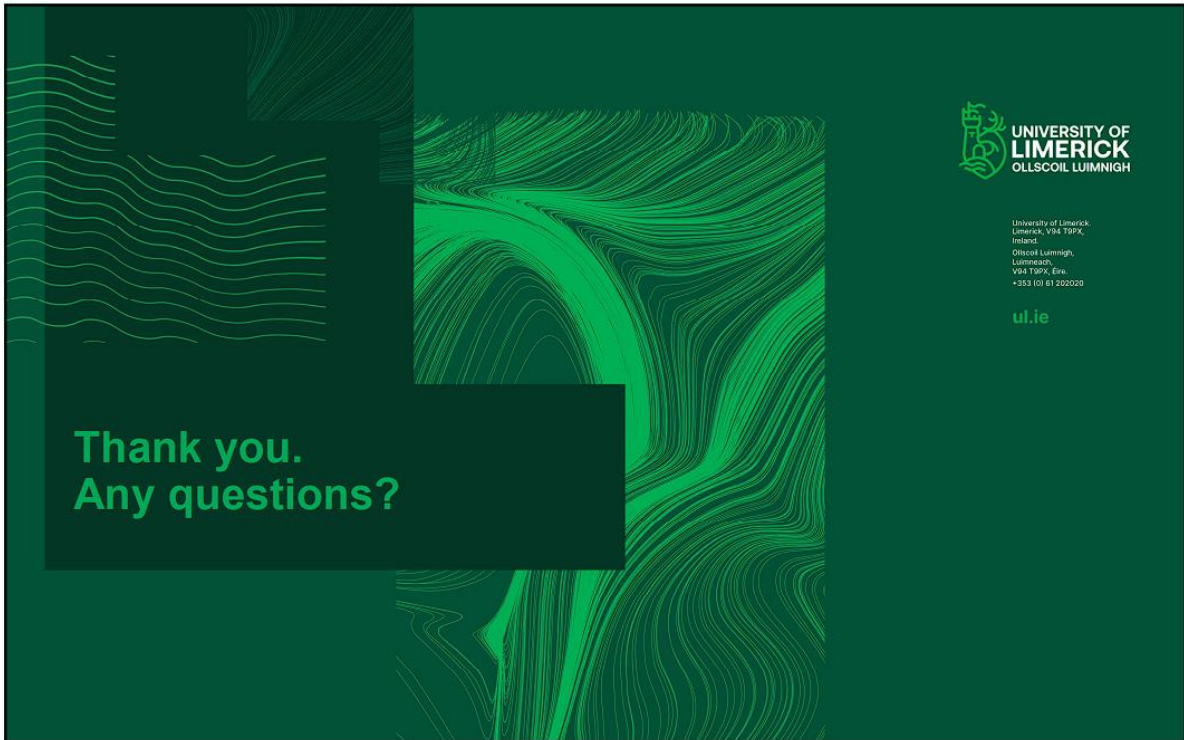
4

Project Gantt chart

FYP_Qinyuans_Project_implmentaion_Gantt_Chart																
Sem 1		Week														
Task no.	Task	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	Ground work (Background research and learning)	0	0	0	0	0										
2	Interim Report	-	-	-	-	0	0	0	Behind schedule!!		-	-	-	-	Exam	Exam
3	Schematic Design	-	-	-	-	0	0	0	DONE		-	-	-	-	Exam	Exam
4	Autumn Semester Presentation	-	-	-	-	0	0	0	Partially completed not finished yet		-	-	-	-	Exam	Exam
5	Code Development	-	-	-	-	-	-	-			0	0	0	0	Exam	Exam
6	PCB Layout	-	-	-	-	-	-	-					0	0	Exam	Exam
Winter Break		Week														
Task no.	Task	-	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	First PCB Schematic and Layout	-	Christmas	0	0	Order PCB		END								
2	PCB Testing	-					0	END								
Sem 2		Week														
Task no.	Task	-	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	Final Report Due & Presentation	-	0	0	0	0	0	0	Final Report Due					END		
2	PCB with STM32 on board	-	0	Ord	-	-	-	-						END		
3	PCB Testing	-	-	-	0	0	Exam development	-	-					END		

Conclusions

- Project summary:
 - Prototype schematics done, the next step will of implementing the prototype will exciting yet challenging, I am expecting imperfection and improvement in the current schematics, as well as challenges in implementing the relatively complexed RTOS code architecture, after prototype, we will be focused on dishing the custom PCB.
- Work completed to-date:
 - Significant progress has been made in understanding and developing the foundations of the project, a schematic has been made while the breadboard testing prototype is under development.
- Next steps:
 - Hard ware: Finish the prototype on a bread board.
 - Soft ware: Finish the prototype some ware that will be the foundation in next step.



Appendix C: Weekly Reports

The weekly report part of the appendix contains a collection of weekly reports that document the progress and deliverables of the project, communicated through regular email updates. The reports were prepared on a weekly basis, typically submitted at the start of each week, summarizing progress made on the tasks from the previous week and setting deliverables for the following week.

****Week 1 (9/9) ****

Hi Ian,

Good morning and happy Monday.

I'm sending a bit of material in preparation for the meeting today, feel free to give it a read if you got bit of time:

****Last Meeting Notes:****

1. **Key Deliverable for Next Monday:**

- Implement a Timer ISR on STM32:
 - Use a similar approach to AVR Dude.
 - Combine HAL functions and direct register manipulation.
 - Watch the Digikey video on Timer ISR.
 - Refer to FreeRTOS and STM32 official documentation for guidance.

2. **Timing Sensitivity Regarding FreeRTOS:**

- Understand Integration:
 - Study how Timer ISR works with FreeRTOS.
 - Use the standard/default FreeRTOS configuration.

3. **Next Meeting:**

- Scheduled for Monday at 12 PM.
- Discuss the first report and request a holistic view of the project.

****To-Do List:****

- **High Priority:**

- Watch the Digikey Video on Timer ISR.
- Implement Timer ISR on STM32: Begin with combining HAL functions and direct register tweaking.

- **Medium Priority:**

- Study FreeRTOS and Timer ISR Integration: Understand how Timer ISR impacts FreeRTOS tasks.
- Refer to Official Documentation: FreeRTOS and STM32 docs for accurate implementation details.

- **Low Priority:**

- Prepare for Next Meeting: Focus on the first report and a holistic view of the project.

****Subject: Follow-up on FYP Project and Key Meeting Notes****

Hi Ian,

I hope you're doing well. I'm sending this email to open a thread regarding the FYP project, in the hope of easing your inbox for future updates. Below are the key notes from our meeting today. Please let me know if you have any additions or corrections.

****Key Notes:****

- ****Key Deliverables:****

- ISR Timer Implementation
- FreeRTOS LED Toggling Demonstration

****Work to be Done:****

1. Study the `static` keyword in C and its implications on ISR using "C How to Program."
2. Implement the timer interrupt routine and understand its relationship with FreeRTOS.
3. Review the real-time system book for insights into system timing and interrupts.

4. Browse Mars Lander FreeRTOS for additional relevant examples and resources.

5. Reflect on the implications of using FreeRTOS and why it's the chosen solution for this project.

6. Review the Analog Development Kit MK1 for relevant insights.

I will do my best to ensure that the deliverables are sent to you before the end of Wednesday, and I'll reach out for any necessary updates. Looking forward to seeing you next Monday.

Thank you again for your time and for this morning's meeting.

Best regards,

Qinyuan

****Week 2 (9/16)****

Hi Ian,

****Minutes from Last Meeting:****

****Key Notes:****

- ****Key Deliverables:****

- ISR Timer Implementation

- FreeRTOS LED Toggling
Demonstration

****Work to be Done:****

1. Study the `static` keyword in C and its implications on ISR using "C How to Program."
2. Implement the timer interrupt routine and understand its relationship with FreeRTOS.
3. Review the real-time system book for insights into system timing and interrupts.
4. Browse Mars Lander FreeRTOS for additional relevant examples and resources.
5. Reflect on the implications of using FreeRTOS and why it's the chosen solution for this project.
6. Review the Analog Development Kit MK1 for relevant insights.

Best regards,
Qinyuan

****Week 3 (9/23)****

Hi Ian,

I'm sending you the list of deliverables for next Monday to review, just in case I've missed or misunderstood anything on my end.

****Deliverables:****

1. ****Produce Schematics:****

- Requirements: SPI for MCP3008 * 2, MCP4822 * 8 (what resolution do we need?)
- Use multiplex, demultiplex, and shift register to reduce pin count.
- Software: Torgit PCB—Beta Layout.
- Potentially I2C capability.

2. ****Learn SPI & I2C Protocols:****

- Digikey has great content for this.

3. ****Gantt Chart:****

- Review the Gantt chart from the Leap Motion project and use it as a template.
- Keep fewer than 10 tasks per semester, and ensure the rest are sub-tasks.

4. ****System Tick/Timing Architecture in FreeRTOS:****

- Focus on FreeRTOS and STM32's documentation page.
- Reference: Real-Time Systems by Jane W.S. Liu.

5. ****Mars Lander Archives:****

- Review how the Mars Lander Viking 1 & 2 handled OS scheduling and priority inversion.

6. ****Miscellaneous:****

- We're using the MK2 as an oscilloscope.

Thank you for your time! I hope you have a great week, and I'm looking forward to our next meeting.

Best regards,
Qinyuan

****Week 4 (9/29)****

Hi Ian,

Happy Monday! The following is the update on the Deliverables for this meeting!

1. ****Produce Schematics: ****

- Done, 74HC959 selected, Shift register solution for CS SPI, more detail needed.

2. ****Learn SPI & I2C Protocols: ****

- Done, reviewed resources:
- [STM32 I2C Example]
(<https://www.digikey.ie/en/maker/projects/getting-started-with-stm32-i2c-example/ba8c2bfef2024654b5dd10012425fa23>)

3. ****Gantt Chart: ****

- Done, but I believe it needs improvement.

4. ****System Tick/Timing Architecture in FreeRTOS:****

- Chapter 12 Operating Systems, time services, and scheduling mechanisms.

5. ****Mars Lander Archives:****

- Done, VxWorks RTOS in Mars Pathfinder Mission had task priority set up wrong and was reprogrammed remotely.

6. ****Miscellaneous:****

- Using the MK2 as an oscilloscope.

Best regards,
Qinyuan

****Week 5 (10/7)****

Hi Ian,

Happy Monday! Here's a summary of today's meeting. The deliverables are listed at the end of the update.

****30/09/2024 [FYP] Meeting Week 4 Sem1:****

****Agenda:****

1. Gantt Chart: In progress

2. Schematics: In progress

****Minutes:****

1. ****Schematics:****

- USB-USART connection
- 16 * 16 digital parallel IO (Shift register & Parallel to serial converter)
- Reference voltage and decoupling capacitor for the DAC chips and check the requirements for other chips.
- Ensure understanding of all analog configurations/requirements for all chips.
- Consider the design of pin connectors.
- Ensure plenty of power and ground (especially ground) connections.
- If approved, order components for breadboard testing.
- Do we need to design isolation and protection for all IO?

2. ****Gantt Chart: ****

- Refer to the FYP guide, translate weekly tasks from [ECE Final Year Project] to the Gantt Chart (Sem_1; Winter Breaks; Sem_2).

3. ****FYP Report: ****

- Develop a high-level structure for the report.
- Follow the guidelines carefully and ensure that the report format is tailored to the specific project.

4. ****Miscellaneous: ****

- Have a bit of think on how the task priority will be set.

- Look deeper into the clock configuration of both internal and external oscillators.

- Pay special attention to the oscillators used for communication and sys-tick.

****Deliverables (Due next Monday):****

1. ****Schematics:**** Complete the overall layout for review. If approved, order breadboard testing chips.

2. ****Gantt Chart:**** Complete the planning for Sem1 at the very least.

3. ****FYP Report:**** Complete the layout and introduction for review.

Let me know if there are any mistakes or if you have anything to add.

Best regards,

Qinyuan

****Week 6 (10/7 – 10/17)****

Hi Ian,

I hope this email finds you well. Below are the updates for this week:

****Minutes:****

1. ****Review of Schematics:****

- Current schematics provide an overview of system functionality but need further refinement.

- Focus on two layout designs: Nucleo-based PCB and custom PCB featuring STM32 on board.

- Key considerations include:

- Modular Design: Modularize each block for easier testing.

- Pin Connectors: Incorporate appropriate connectors for interfacing with Nucleo.

- IC Component Review: Ensure all IC components meet manufacturer specs.

- STM32 Integration: Begin designing the STM32-on-board PCB after Nucleo testing.

- Shield Design: Consider designing the PCB as a shield for easier interfacing with Nucleo.

2. ****Gantt Chart:****

- Current format is acceptable, and I'll have it finalized by next Monday.

3. ****FYP Interim Report:****

- Abstract is nearly done. I can modify the chapter titles to better fit the project structure.

****Deliverables/To-Do for Next Week:****

1. Order components based on schematics.

2. Redraw schematics to meet professional standards for PCB design.

3. Review component specifications, ensuring compatibility with power and signal integrity requirements.

4. Assess clock configuration for communication and SysTick timing.

5. Review STM32-based RTOS projects focusing on ADC/DAC integration.

6. Complete the introduction section of the FYP report for review.

7. Finalize and submit the Gantt Chart.

Please let me know if there are any corrections or additional suggestions.

Best regards,

Qinyuan

****Week 7 (10/17)****

Hi Ian,

****Pin Table on Nucleo:****

- Create a pin configuration table for the Nucleo board.

****Chapter Title:****

- Combine Introduction and Motivation into one section.

****Technical Details:****

- The report should have two sections:
Software and Hardware (already
completed).

****Deliverables:****

1. Finalize schematics for each block.

2. Complete the Gantt chart based on the
action plan.

3. Write with the reader in mind.

4. Complete miscellaneous tasks.

****Next Meeting:****

- Tuesday at 4 PM, online.

Best regards,

Qinyuan

Appendix D: Drawing

1. General Data Acquisition System

Block Diagram

Title: General Data Acquisition
System Block Diagram

Author: Qinyuan Liu

Institution: University of Limerick

Student ID: 20137095

Supervisor: Dr. Ian Grout

Time: 27/10/2024

2. RTOS Task Priority Diagram

Title: RTOS Task Priority Diagram

Author: Qinyuan Liu

Institution: University of Limerick

Student ID: 20137095

Supervisor: Dr. Ian Grout

Time: 27/10/2024

3. ADC Module Schematic

Title: ADC_Module

Author: Qinyuan Liu

Institution: University of Limerick

Student ID: 20137095

Date: 27/10/2024

4. DAC Module Schematic

Title: DAC_and_CS_Module

Author: Qinyuan Liu

Institution: University of Limerick

Student ID: 20137095

Date: 27/10/2024

5. Digital Input Module Schematic

Title: Digital_Input_Module

Author: Qinyuan Liu

Institution: University of Limerick

Student ID: 20137095

Date: 27/10/2024

6. Digital Output Module Schematic

Title: Digital_Output_Module

Author: Qinyuan Liu

Institution: University of Limerick

Student ID: 20137095

Date: 27/10/2024

Link to all the Drawings:

<https://github.com/Qinyuan72/FYP/tree/master/Drawings>

Appendix E: Header Table

1. **Pin_Config_Nucleo:** Overview of pin connections in the Nucleo board configuration. (Page 1 of 11)
2. **DAC_And_CS_Header:** DAC and Control Signals connections. (Page 4 of 11)
3. **DAC_OUTPUT:** output connections of the DACs. (Page 5 of 11)
4. **ADC_Header:** ADC inputs. (Page 6 of 11)
5. **ADC_Analog_Output:** Analog output channels. (Page 7 of 11)
6. **Digital_Output:** Digital output channels. (Page 8 of 11)
7. **Digital_OUT_Header:** Digital output bus connections. (Page 9 of 11)
8. **Digital_Input:** Digital input connections to 74HC165 components. (Page 10 of 11)
9. **Digital_In_Header:** Digital input bus connections, describing data, latch, and clock lines. (Page 11 of 11)