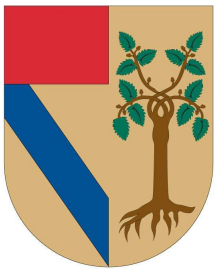
FreeRTOS Course

July 20, 2024



UNIVERSIDAD
PANAMERICANA

Contents

1 Course Overview **3**
- 1.1 Instructor Information . . . 3
- 1.2 Prerequisites . . . 3
- 1.3 Assessment and Evaluation . . . 3
- 1.4 Objectives . . . 3

2 Agenda **4**

3 Course Content **7**
- 3.1 Module 1: Introduction to Real-Time Systems . . . 7
  - 3.1.1 What exactly means Free? . . . 7
  - 3.1.2 What is an Operating System? . . . 7
  - 3.1.3 What is Real-Time? . . . 7
  - 3.1.4 What does RTOS standfor? . . . 7
  - 3.1.5 Critical systems . . . 7
- 3.2 FreeRTOS Architecture . . . 8
  - 3.2.1 Basic Concepts . . . 8
  - 3.2.2 Concurrent procesing in FreeRTOS . . . 8
- 3.3 Hands on ESP32 . . . 8
  - 3.3.1 Short story about spressif microcontrollers . . . 8
  - 3.3.2 Install visual studio code . . . 8
  - 3.3.3 WSL instalation for espressif FreeRTOS SDK . . . 8

3.4 How to do UML and planning for FreeRTOS programming 9
3.5 FreeRTOS Programming 9

4 Final project **10**
4.1 Software Requirements: 11

5 Materials and Resources **11**

1 Course Overview

1.1 Instructor Information

Name: Luis Emilio Tonix Gleason

Contact: ltonix@up.edu.mx

This course provides comprehensive training on FreeRTOS, targeting embedded systems programmers who are looking to utilize the FreeRTOS real-time operating system to its full potential in cricitcal system where safety is a key role for a responsable project. Remeber safety first.

1.2 Prerequisites

Participants are expected to have:

1. Basic knowledge of C programming
2. Microcontroller architectures.
3. Linux basics commands
4. Esp32 microcontroller
5. Windows/Linux Setup. MacOs not guarenteed to be supported
6. Git/Github: I will review your code and diagrams in this tool. We can learn good coding standars and practices, and you will generate a porfolio.

1.3 Assessment and Evaluation

Assessment Component	**Percentage**
Class Participation	10%
Homework Assignments	20%
Midterm Examination	20%
Project Design and Requirements First Delivery	20%
Final Project Delivery	30%

Table 1: Breakdown of assessment components and their corresponding weights in the course.

Class participation involves responding to the professor's questions on the topic currently being studied.

Some homework assignments are assessed through quizzes using the Kahoot format.

Winners of the Kahoot quizzes receive additional bonus points that can be used towards projects or exams.

1.4 Objectives

- Understand the core concepts of real-time operating systems.
- Learn how to configure,develop and run FreeRTOS on ESP32.
- Design and management of systems that must compute and provide results within strict time constraints.
- Designing and implementing effective emergency procedures and recovery plans to handle potential disasters or failures.

- Study of systems where failure could cause loss of life, significant property damage, or environmental harm.

2 Agenda

Week 1: Introduction to Real-Time Systems

- What is an operating system?
- Main features of an OS
- Safety
- Security
- What is real time?
- Introduction to Real-Time Operating Systems (RTOS) and their relevance in embedded solutions
- MIT License, GNU, MIT, and GPL
- Understanding Copyleft
- Comparison of Open Source Licenses

Week 2: Critical Systems and Applications of RTOS

- Discussion of RTOS applications in critical systems across various industries
- Aerospace and Aviation (e.g., General Electric, Hydra, DJI)
- Automotive (e.g., Continental, NXP, Avnet)
- Healthcare (e.g., Plexus, Baxter)
- Power Generation (e.g., Baker Hughes, CERN)
- Telecommunications (e.g., Cinvestav, Qualcomm)
- Defense and Military (e.g., Hydra, A2E)

Week 3: Continuous Integration and Professional Workflow

- Overview of standard industry practices.
- Importance of documentation and diagrams in maintaining scalable and maintainable code.
- Software Development Guide (SDG)
- Key components of an effective SDG.
- Software Detailed Document (SDD)
- Requirements
- Exploring the role and structure of SDD.
- Unified Modeling Language (UML)
- Development process, good practices, and coding standards
- Version control process GIT
- Test plan document
- Whitebox testing and manual testing

- Understanding Continuous Integration
- Benefits of integrating CI into the software development lifecycle.

Week 4: Continuous Integration and Professional Workflow (continued)

- Practical perspective on applying CI strategies in development projects.

Week 5: FreeRTOS Architecture (Part 1)

- Introduction to heap/stack memory management in RTOS.
- Detailed discussion on function pointers and the concept of callbacks.
- Overview of interrupt service routines and timer ISR.

Week 6: FreeRTOS Architecture (Part 2)

- In-depth look at concurrent processing in FreeRTOS:
- Scheduler dynamics
- Task management and priority settings
- Queues, semaphores, and mutexes

Week 7: Hands-on with ESP32 (Setup and Basic Examples)

- Introduction to Espressif microcontrollers, with a focus on ESP32.
- Setting up the development environment, including Visual Studio Code.
- Configuring WSL for the Espressif FreeRTOS SDK.
- Practical examples: Running basic GPIO LED control tasks.

Week 8: Advanced Hands-on with ESP32

- Techniques for effective serial debugging.
- WIFI drivers and code examples

Week 9: FreeRTOS Programming (Part 1)

- Techniques for task creation and management.
- Control functions such as delay, suspend, and resume.
- FreeRTOS configuration overview

Week 10: FreeRTOS Programming (Part 2)

- Comprehensive management of queues and semaphores.

Week 11: Midterm Review and Examination

- Review of all topics covered thus far.
- Midterm examination to assess knowledge and practical application.

Week 12: IoT MQTT Programming Part 1 (TCP and WIFI)

- Introduction to MQTT protocols
- Setting up MQTT broker and clients

- Implementing MQTT with TCP and WiFi on ESP32

Week 13: IoT MQTT Programming Part 2 (Adafruit IO)

- Integrating IoT devices with Adafruit IO using MQTT
- Developing interactive IoT applications
- Practical examples and project ideas

Week 14: Project Design, Requirements, UML and Planning

- Detailed discussion on final project design and requirements.
- Initial project planning and milestone setting.
- Utilizing Unified Modeling Language (UML) to plan and model projects.
- Creating sequence diagrams and behavioral architectures for real-time systems.

Week 15: FreeRTOS Programming (Part 3)

- Detailed handling of software timers and event groups.
- Advanced scheduling techniques.

Week 16: Final Project workshop (Part 1)

- Practical implementation of the final projects.
- Final presentations and review sessions.

Week 17: Final Project workshop (Part 2)

- Continued implementation and final adjustments.
- Concluding presentations and evaluations.

Week 18: Final Project workshop (Part 3)

- Completion of project implementations.
- Final evaluations and feedback.

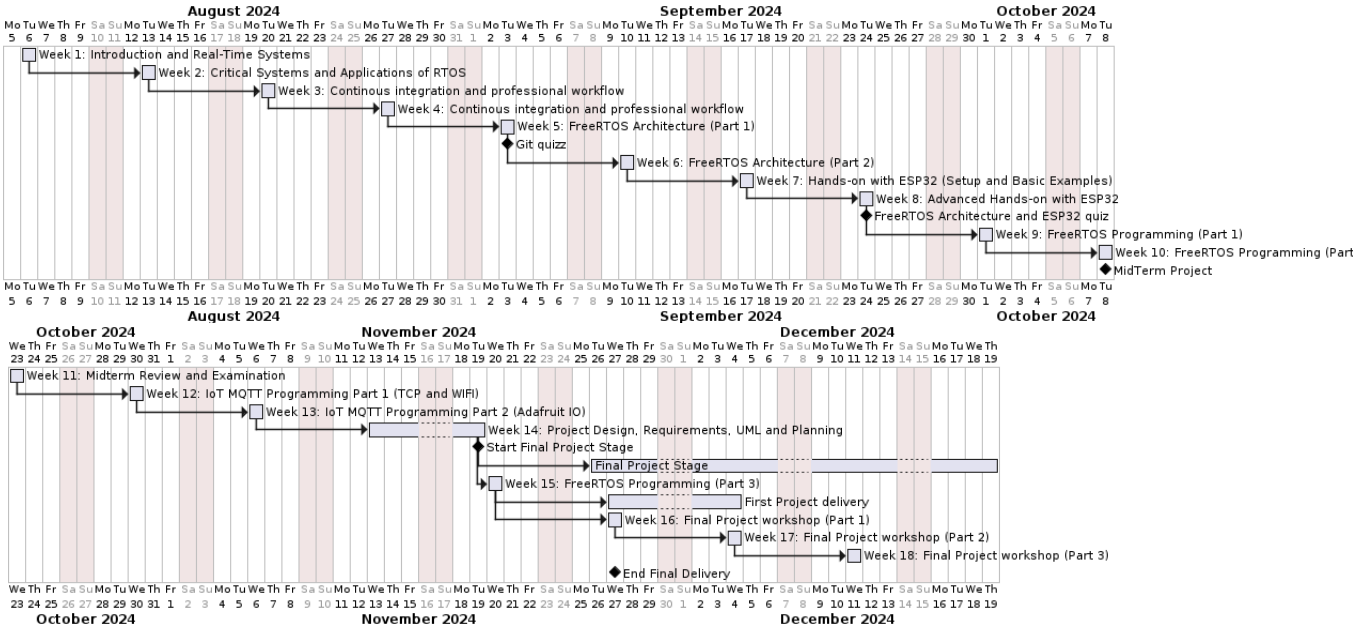


Figure 1: Course Gannt

3 Course Content

3.1 Module 1: Introduction to Real-Time Systems

3.1.1 What exactly means Free?

Overview of the implication of having an MIT license. What you should and not do with this type of licence.

3.1.2 What is an Operating System?

A generic overview of an operating sytem

3.1.3 What is Real-Time?

Diagrams to undertand the key concept of real time

3.1.4 What does RTOS standfor?

How RTOS are part for embbeded solutions

3.1.5 Critical systems

Use cases for RTOS enviroments

- Aerospace and Aviation (General Electic) (Hydra) (Dji)
- Automotive (Continental) (NXP) (Avnet)

- Healthcare (Plexus) (Baxter)
- Power Generation (Baker Hughges)(Cern)
- Telecommunications (Cinvestav)(Qualcomm)
- Defense and Military (Hydra)(A2E)

3.2 FreeRTOS Architecture

3.2.1 Basic Concepts

- Heap/Stack Memory managment
  - We need to know how this memory is managed because the RTOS task creating will require to understand this
- Function pointers
  - Review of the callback concept
- Interrupt service routine [Review]
- Timer ISR

3.2.2 Concurrent procesing in FreeRTOS

- Scheduler
- Task, priorities
- Queues
- Semaphores
- Mutexes

3.3 Hands on ESP32

3.3.1 Short story about spressif microcontrollers

3.3.2 Install visual studio code

3.3.3 WSL instalation for espressif FreeRTOS SDK

- Download git FreeRTOS SDK project
- Explore and open basic example
- Configure COM ports to program
- How to serial debug microcontroller
- Run basic examples with GPIO LED

3.4 How to do UML and planning for FreeRTOS programming

3.5 FreeRTOS Programming

https://www.freertos.org/a00106.html

- Configuration
  - FreeRTOS configuration overview
- Task Creation
  - xTaskCreate vs xTaskCreateStatic
  - vTaskDelete
  - What task parameters mean?
  - Create one task that blink a led to high frequency
  - Create other task that blink a led with low frequency
- Task Control
  - vTaskDelay
  - vTaskDelayUntil
  - vTaskSuspend
  - vTaskResume
  - xTaskResumeFromISR
- Kernel Control
  - vTaskStartScheduler
  - vTaskStepTick
  - vTaskStartScheduler
- Task Utilities
  - vTaskList
  - xTaskGetSchedulerState
- Queue Management
  - xQueueCreate
  - xQueueCreateStatic
  - vQueueDelete
  - xQueueSend
  - xQueueSendFromISR
  - xQueueReceive
  - xQueueReceiveFromISR
- Semaphores
  - xSemaphoreCreateBinary

  - xSemaphoreCreateBinaryStatic
  - xSemaphoreGive
  - xSemaphoreGiveFromISR
- Software Timers
  - xTimerCreate
  - xTimerCreateStatic
  - vTimerSetReloadMode
  - xTimerStart
  - xTimerStop
  - xTimerChangePeriod
  - xTimerDelete
  - xTimerReset
- Event Groups
  - vEventGroupDelete
  - xEventGroupClearBits
  - xEventGroupClearBitsFromISR
  - xEventGroupCreate
  - xEventGroupCreateStatic
  - xEventGroupGetBits
  - xEventGroupGetBitsFromISR
  - xEventGroupGetStaticBuffer
  - xEventGroupSetBits
  - xEventGroupSetBitsFromISR

4 Final project

For a final project in a FreeRTOS class that involves the implementation of a critical system, you could consider designing a "Smart Emergency Response System." This project would integrate various sensors, communication modules, and control outputs to create a comprehensive emergency management system suitable for environments like industrial facilities, schools, or public buildings. The SERS will use FreeRTOS to manage the concurrent operations of various sensors and actuators to detect and respond to emergencies such as fires, gas leaks, and unauthorized access. The system will prioritize tasks based on the severity and type of emergency, ensuring that the most critical responses are handled first.

4.1 Software Requirements:

1. Design
   (a) Software squence diagram, representing task comunication
   (b) Behavioral architecture
   (c) Software Detail Design Document and the critical constrains to be met.
2. FreeRTOS Configuration:
   (a) Implement tasks
   (b) Queues
   (c) Semaphores
   (d) Mutexes.
3. Task Scheduling:
   (a) High priority tasks for immediate threats (e.g., fire detection and response).
   (b) Lower priority tasks for non-immediate alerts (e.g., unauthorized access).
4. Interrupt Service Routines (ISR): For immediate processing of critical sensor data.
5. Communication Protocol: Implementation for sending real-time data and alerts.

5 Materials and Resources

https://www.freertos.org/Documentation/Mastering-the-FreeRTOS-Real-Time-Kernel.v1.0.pdf