Western Washington University
Engineering and Design Department

EECE 344 – Embedded Microcontrollers II

STM32L4 Mini Project (10%)

Done in Teams of 2

Due on Nov. 30th at 11 pm

Submission to GitLab

Project Overview

For this mini project, students will design and implement an embedded application on the STM32L4 microcontroller, showcasing the use of at least one peripheral, a timer, and interrupt. This project allows students to explore a real-world application by integrating peripherals and using timers to manage time-based events or data sampling. Students are encouraged to be creative and design an application that demonstrates core embedded systems concepts, such as data acquisition, communication, or control.

Reusing code from previous labs is **encouraged**. This project should build on your previous work with peripherals, timers, and interrupts, allowing you to integrate and expand on your knowledge in a creative way.

Project Requirements

- **Microcontroller**: STM32L476RG.
- **Peripherals**: Use at least one peripheral (e.g., ADC, GPIO, I2C, UART, SPI, DAC, etc.).
- **Timer**: Incorporate a timer to manage periodic tasks or control event timing.
- **Interrupts**: Use interrupts to respond to specific events, enhancing the project's real-time responsiveness.

Examples of Suitable Peripherals

Below are some peripherals that can be integrated into the project. Students can choose any one or more based on the requirements of their application:

1. **Analog-to-Digital Converter (ADC)**: For measuring sensor values, such as temperature or light intensity.
2. **Digital-to-Analog Converter (DAC)**: For generating analog signals from digital values.
3. **General-Purpose Input/Output (GPIO)**: For controlling LEDs, reading button states, or toggling relays.
4. **Universal Asynchronous Receiver/Transmitter (UART)**: For serial communication with other devices or a computer.

5. **Inter-Integrated Circuit (I2C)**: For communicating with I2C-enabled modules, such as displays or sensors.
6. **Serial Peripheral Interface (SPI)**: For high-speed communication with sensors, displays, or other microcontrollers.

Suggested Applications

While students have the freedom to design their own projects, here are some application ideas for inspiration:

1. **Environmental Monitor**: Measure temperature, humidity, or light levels using sensors, and display readings on an OLED display.
2. **Digital Clock**: Use a timer to keep time, display it on an LCD, and provide button controls to set the clock.
3. **Signal Generator**: Generate different waveform signals (sine, square) using the DAC, with a timer controlling the output frequency.
4. **Smart Home Control Panel**: Control appliances or lights with buttons and a display, using UART or I2C for communication with other devices.

Project Requirements Checklist

- **Peripherals**: Select and configure at least one peripheral in STM32CubeIDE.
- **Timer Configuration**: Set up at least one timer for periodic tasks or delays.
- **Interrupts**: Use interrupts to handle events such as button presses, sensor triggers, or timer events.
- **Documentation**: Include a brief project presentation covering:
    - The purpose and functionality of the project.
    - How each peripheral, timer, and interrupt is used.
    - Code explanations for key sections.

Suggested Project Steps

1. **Project Planning**:
    - Define the objective and functionality of your project.
    - Identify which peripherals you will use and how they will interact in your application.
2. **Peripherals and Timer Setup**:
    - Use STM32CubeMX within STM32CubeIDE to configure the selected peripherals and timer.
    - Define the role of each peripheral in the project and map them to suitable pins.
3. **Coding and Implementation**:
    - Implement interrupt handlers for time-based events or peripheral-specific events.
    - Use a timer interrupt to create a periodic action, such as updating a display or sampling data.

    - Integrate the peripherals to work together within the main loop or through interrupt-driven routines.
4. **Testing and Validation**:
    - Test each peripheral individually, then integrate them to form the complete application.
    - Debug using breakpoints and serial output to ensure the application behaves as expected.
5. **Demonstration and Documentation**:
    - Prepare a short demonstration of the project, explaining how it works and the role of each component.
    - Submit a project presentation detailing your design, code, and testing process.

Evaluation Criteria

Projects will be evaluated based on:

- **Functionality**: Completeness and reliability of the application.
- **Peripherals and Timer Usage**: Effective configuration and integration of peripherals, timers, and interrupts.
- **Code Structure**: Code readability, use of comments, and adherence to embedded programming practices.
- **Documentation**: Clear explanations of design choices, functionality, and code operation.

This project provides an opportunity to explore and apply embedded programming skills in a creative and hands-on way. Remember to make use of both timers and interrupts effectively to demonstrate real-time capabilities.

As part of this project, students will deliver a presentation to showcase their embedded application, covering the design, implementation, and operation of their project. The presentation allows you to share your design process, demonstrate technical knowledge, and reflect on the challenges and solutions you encountered.

Presentation Requirements

1. **Project Overview (1-2 slides)**
    - **Project Objective**: Summarize the purpose of your project, the problem it solves, or the application it demonstrates.
    - **Functional Overview**: Briefly explain the main functionalities of your embedded system, including how it integrates peripherals, timers, and interrupts.
2. **Hardware and Peripherals (1-2 slides)**
    - **Peripherals Used**: List the peripherals you used (e.g., ADC, GPIO, I2C) and briefly describe their roles in your project.
    - **Hardware Setup**: Provide a diagram or photo of your hardware connections, explaining how different components are connected to the STM32L4.

3. **System Design and Configuration (2-3 slides)**
    - **Timer and Interrupt Configuration**: Explain how you configured timers and interrupts, including their purpose and how they contribute to the functionality of your project.
    - **Peripheral Configuration**: Briefly describe the configuration steps for each peripheral used, focusing on important settings (e.g., ADC sampling rate, I2C clock speed).
    - **Software Flow**: Include a flowchart or diagram of your code structure, showing how the main loop and interrupt service routines work together.
4. **Code Explanations (1-2 slides)**
    - Highlight key sections of your code, such as interrupt service routines or peripheral initialization functions.
    - Explain how you handled specific tasks like data processing, peripheral communication, or real-time event handling.
5. **Challenges and Solutions (1-2 slides)**
    - Describe any significant challenges you faced during the project (e.g., debugging interrupts, configuring peripherals).
    - Share the solutions you implemented to overcome these challenges and how they improved your project.
6. **Demonstration of Functionality (1-2 slides)**
    - Include a short video or live demonstration of your project in action, showing the primary features and how the peripherals and timer are used.
    - Ensure that your demonstration highlights key functionality like data acquisition, communication, or control features that involve timers and interrupts.
7. **Conclusion and Reflection (1 slide)**
    - Summarize the main takeaways from your project experience.
    - Discuss what you learned about embedded systems, STM32 peripherals, and using timers and interrupts.
    - Reflect on how this project could be expanded or applied to other real-world applications.

Tips for a Successful Presentation

- **Clarity**: Make sure each slide is clear and concise, with minimal text and informative visuals.
- **Visual Aids**: Use diagrams, flowcharts, and code snippets to visually explain your system design and code structure.
- **Practice**: Ensure you're comfortable with the technical details so you can explain them confidently, especially regarding how your peripherals, timer, and interrupts are used.