Programming Phase

Objectives:

Practice generating **PWM** on the STM32 to control a **servo motor** and an LED. Learn to use timers, the **Capture/Compare Register (CCR)**, **push-button input**, and **ADC input** from a **potentiometer** to adjust servo angle and LED brightness.

Main Task:

For the following subtasks, you are required to build the appropriate circuits on your breadboard using your KIT. The components will include a **servo motor,** a **LED**, a **potentiometer**, and a **push button**. To power on the STM32 bluepill, you may use the TTL cable provided in the KIT.

Make sure to verify the connections with your mentor before powering on the STM32 board to ensure safety and correct wiring.

Subtask 1:

Write a documentation explaining the concept of PWM signal and how it's used to control servo motors, the documentation must contain the following topics:

PWM Signal Explanation

- Define PWM and explain why it is used.
- Describe the key elements: Period (T), Frequency (f), Duty Cycle (%), Signal Shape.
- Explain the concept of Average Voltage (Vavg) and include the mathematical relation.
- Include illustrative waveforms for different duty cycles (e.g., 25%, 50%, 75%).

Servo Motor Explanation

- Define what a servo motor is and mention its main components.
- Explain how a servo motor operates and how it differs from a regular DC motor.
- Mention the range of movement and the role of the internal feedback system.

You may refer to your task 1 technical documentation file.

Servo Motor Control using PWM

- Describe the control signal required by a servo.
- Explain the relation between Pulse Width and rotation angle (angle map).
- Provide a general formula linking Pulse Width to the rotation angle.
- Include an explanation of the Capture/Compare Register (CCR) and how it is used in microcontrollers to generate the PWM signal that controls the servo.

Practical Calculations

- Show how to calculate frequency from the period.
- Show how to calculate Duty Cycle from Pulse Width and Period.
- Provide a reference table with examples (Pulse Width vs. Rotation Angle vs. Duty Cycle).

Fully AI documentations won't be accepted. Support your documentation with visualized content and list all resources used (websites, youtube videos, …).

Subtask 2:

Control the brightness of a LED using a potentiometer and PWM.

- Configure a timer to generate a PWM signal.
- Read the potentiometer value via ADC.
- Map the ADC value to the PWM duty cycle (using CCR) to adjust LED brightness.
- Ensure smooth brightness changes as the potentiometer is rotated.

Subtask 3:

Timer Configuration

- Calculate and configure the correct timer parameters (Prescaler, Auto-Reload Register, and Capture/Compare Register values) to generate a PWM signal with a 20 ms period (50 Hz) suitable for servo motor control.

Servo Control Logic

- Implement a program where each press of the push button increases the servo angle by **45°**.
- Continue increasing the angle until the maximum rotation angle (180°) is reached.
- After reaching 180°, the next button press should reset the servo back to 0° and continue the sequence.
- The sequence should look as follows:
  0° → 45° → 90° → 135° → 180° → 0° → 45° …

Optional Task:

Sub-optional 1:

Toggle an LED with a frequency of 2 Hz using the timer interrupt function.

Sub-optional 2:

Generate PWM signal to control the servo motor without using the built in function, use the timer interrupt function (software PWM).

Hints :

As part of our encouragement for you to not fully rely on AI to complete your tasks, the following are some helpful resources to help you to complete the task:

STM32 PWM Output Example Code (PWM Generation Tutorial)

- Servo Motors, how do they work?
- STM32 Guide #3: PWM + Timers
- Pulse Width Modulation (PWM) - Electronics Basics 23

Note: these are not the only available resources to help you complete the tasks. Also, they are not necessarily explaining the exact requirements of the task, but they contain some helpful information

Servo motor safety rules :

- **Avoid Manually Forcing the Servo Shaft:** Do not try to manually rotate or force the servo motor shaft or gears. Servo motors have delicate internal gears that can break easily when forced by hand.
- **Do Not Block the Servo While Moving:** Avoid physically blocking the servo motor shaft during operation. This can cause the servo to draw its maximum current, potentially overheating and damaging the servo, and in some cases causing overcurrent that may harm the STM32 board.

Submission:

- Record a video explaining the circuit connections and the code and send it to your mentor.
- Send the main.c file to your mentor.
- Complete and submit this task in one PDF file for both main and optional (if done) tasks.
- Add a picture of your circuit connections to the PDF file.
- Add a screenshot of the configurations of IOC to the PDF file.
- Add a screenshot of the code of each subtask to the PDF file (only what you have modified in the generated code).
- Name the PDF files with task5_groupx_your_name, (replace x with your group number).
- This task should be submitted before 26st Aug 10:00pm (Malaysia time), 05:00pm (Makkah time).
- Your mentor must approve your task answer file before submission.
- Submit the PDF file to the Google form.