EE212-Microprocessors Laboratory Assignment 4

Due Date: April 29, 2024 13.30

1 Assignment Details

For this lab assignment, you will implement **Pulse Width Modulated (PWM)** signals using timers of the FRDM-KL25 board. Some pins of FRDM-KL25 board are capable of generating PWM signals which are nothing but periodic square waves with two different parameters: duty cycle (%) and periodic frequency (Hz). To understand and implement PWM, you will utilize a SG90 servo motor to rotate its servomotor. With different duty cycles, you will rotate the servo periodically from left-to-middle and middle-to-left and when the button is pressed you will change the pattern of the periodic rotation to right-to-middle and middle-to-right.

Following are to be implemented in this lab assignment:

1. (**30 points**) Using timers of FRDM-KL25 board (TPM timers), generate PWM signals on the pin of FRDM-KL25. Generated signal will be measured from this pin by using oscilloscope probe. Therefore, you will need a probe for this assignment. By using cursors, please check if generated signals have correct frequency and duty cycle values as your input values. Errors up to % 3-5 percent are acceptable. During the check, you can stop the oscilloscope to check a single duty cycle.

2. (**40 points**) You will connect FRDM-KL25 with SG90 servo motor for continuous rotation of the motor. Firstly, motor should continously follow: $\angle 0 \rightarrow \angle 30 \rightarrow \angle 60 \rightarrow \angle 90 \rightarrow \angle 60 \rightarrow \angle 30 \rightarrow \angle 0$. The interval of angle change should be 0.5 seconds.

3. (**30 points**) You will implement and use Interrupt Service Routine (ISR) to act when the button is pressed. When the button is pressed, motor should follow the new pattern with a similar time intervals of 0.5 seconds. The new pattern will be: $\angle 180 \rightarrow \angle 150 \rightarrow \angle 120 \rightarrow \angle 90 \rightarrow \angle 120 \rightarrow \angle 150 \rightarrow \angle 180$. So, the new pattern is symmetric to the previous pattern. When the button is pressed again, pattern should be revert back to the original case.

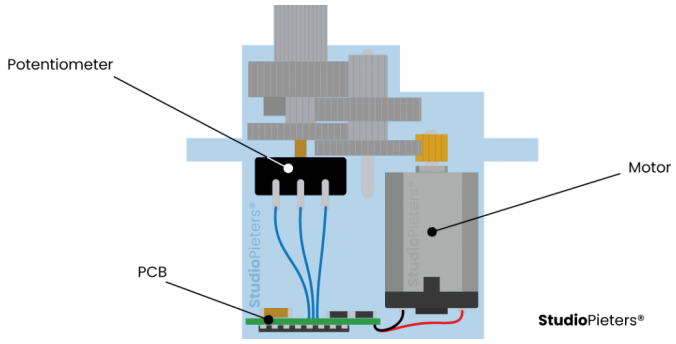

Figure 1: Illustration of the internal structure of SG90.

2 Configuration Details

2.1 SG90 Servo Motor

SG90 contains a small DC motor that is connected to the output shaft through the gears (see Fig.1). The output shaft drives a servo arm and is also connected to a potentiometer (pot). The potentiometer provides position feedback to the servo control unit where the current position of the motor is compared to the target position. According to the error, the control unit corrects the actual position of the motor to match the target position.

You can control the servo motor by sending a series of pulses to the signal line. A conventional analog servo motor expects to receive a pulse approximately every 20 milliseconds (the signal should be 50 Hz). The length of the pulse determines the position of the servo motor (see Fig.2).

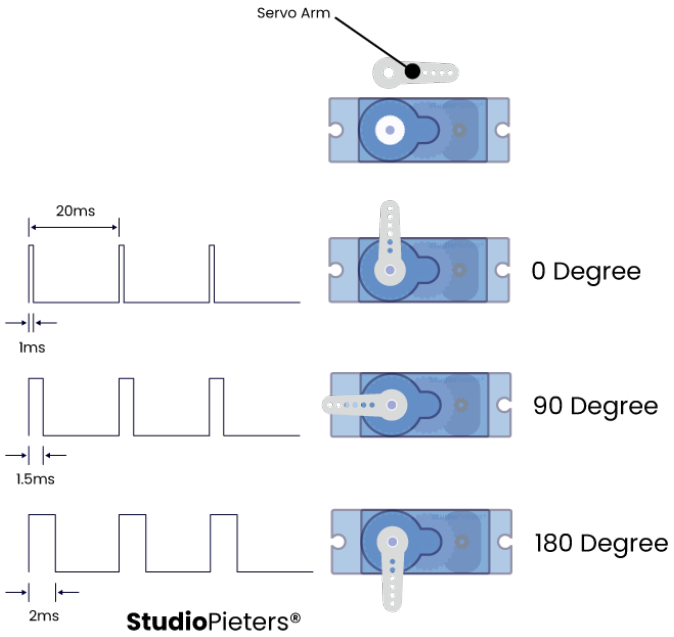

Figure 2: Illustration on how duty cycle of the signal affects the servo rotation.

If the pulse is high for 1ms, then the servo angle will be ∠0. If the pulse is high for 1.5ms, then the servo will be at its center position. If the pulse is high for 2ms, then the servo will at ∠180. Pulses ranging between 1ms and 2ms will move the servo shaft through the full ∠180 of its travel. The duration of the pulses may sometimes vary with different brands and they can be 0.5ms for ∠0 and 2.5ms for ∠180. You will need to check it during the implementation.

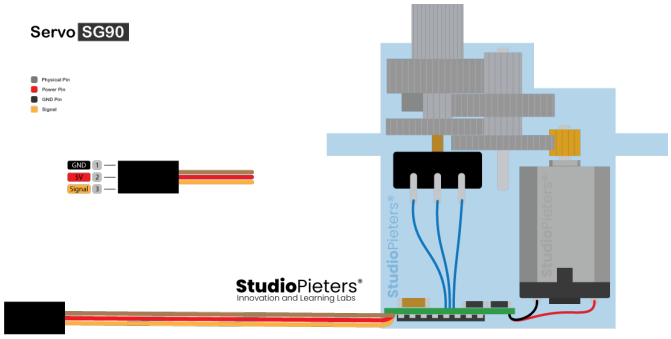

Figure 3: SG90 pinout.

2.2 Interrupt Service Routine

Defining an ISR (Interrupt Service Routine) to act when the button is pressed is a crucial step in the implementation process. This will allow the direction pattern of the servo motor to be switched between left-to-middle/middle-to-left and right-to-middle/middle-to-right as the button is pressed. To achieve this, it may be necessary to define a global variable to keep track of the current pattern.

3 Guidelines

Follow the guidelines given below for implementation:

- Please refer to Mazidi's book Freescale ARM Cortex-M Embedded Programming. Chapter 5 - Section 2 is dedicated for general configuration of TPM timers, Chapter 11- Section 2 is dedicated for generating PWM signals, Chapter 2 is dedicated to GPIO.
- If you are to use PWM signals generated by the board, you should choose pins which are capable of generating these signals (For example PORTE Pin 20 or PORTE Pin 31 may be good option for generating PWM signals). Please refer to the manual in this case.
- TPM timers are used to generate PWM signals. So use TPM timers instead of Systick Timer; this will ease your process. Also note that, there is no restriction about how many timers you should use.