EE663- Project 2 Report
Controlling Servo Motor

Table of Contents

Overview:

The aim of the project is to control a pair of servo motors using custom interpreted control language.

- Independently control both motors up to six positions.
- Implement custom interpreted user defined recipe. This is like how mnemonic opcode interaction work in an assembly level language program.

Cover Page:

Project title	:	Controlling of Servo Motors
Names	:	Deepak Siddharth Parthipan
E-mail addresses of the authors	:	dp9040@rit.edu.
Date of submission	:	11-14-2016.

Areas of Focus:

Deepak Siddharth Parthipan	: Problem Statement analysis, Functional Design, Developing Code, Test plan and Report.
Tanmay Shinde	: Problem Statement analysis, Functional Design, Test plan.

Analysis / Design:

The basic idea of the project is to do multitasking by means of controlling two servo motors and working w.r.t user inputs from the terminal.

Hardware configuration:

The STM32L476VGT6 kit is quite powerful one with 32-bit ARM Cortex 4 CPU, with a bunch of highly configurable peripherals. It is connected to the computer system via USB. This facilitates three basic functions: connection to IDE, power supply and serial communication. UART port pins TXD and RXD of the

STM32 is used for serial communication with the terminal server. Putty is the host that enables the connection with the program.

The Servo is connected to the output ports of the STM32 device only after verifying. This is because to avoid giving any wrong pulses to the signal inputs of the servo and prevents the damage of the servo motor.

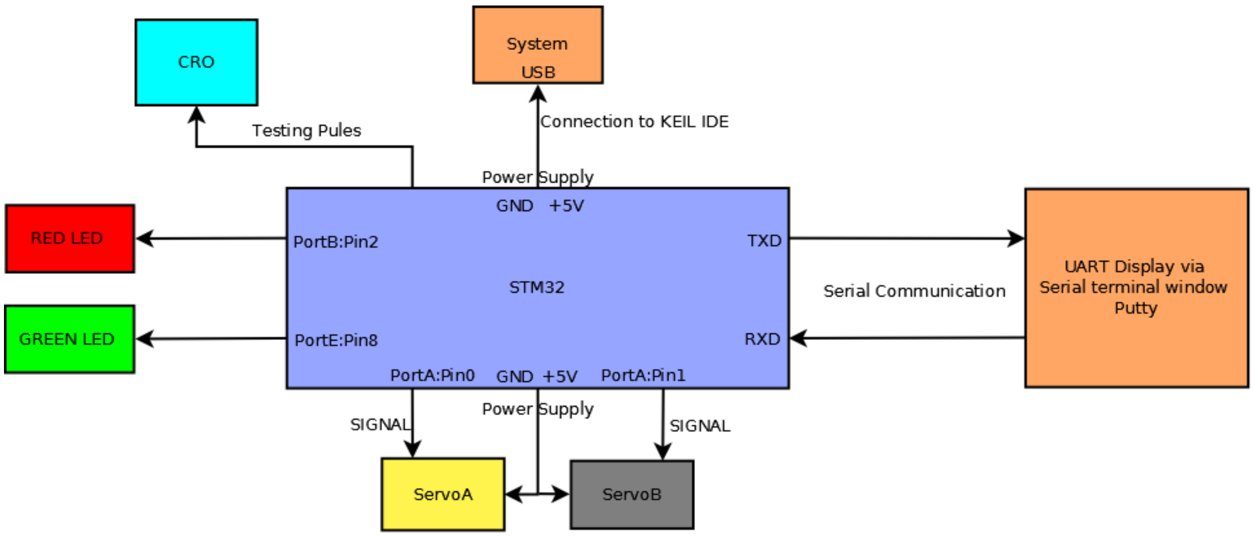


Figure1: Hardware block diagram of hierarchical components

Software Implementation:

Main software implementation involves configuring PWM generation, TIMER5 Interrupt configuration, UART, LED and SYSTEM CLOCK. The software code for UART, LED and SYSTEM CLOCK is reused from the demo code provided in the class. Since there is no Operating system the program must run in an infinite loop (while (1)).

Configuring the PWM and running was the core task of this project. A continuous pulse of 2%-20% of the pulse width will give a complete rotation of the servo. For this purpose, the timer 2 pre-scalar was divided into 100microseconds count. Auto reload register was set to a value of 200. So, the final pulse width was 20ms. Different positions of the servo were controlled using the register CCR2.

The two main operations of the program involve, the periodically receive the user information via terminal and run the sequence of opcode code. Since both are kind of parallel tasks. It was decided that the timer will run the opcode sequence at regular periodic intervals. For this purpose, a timer 5 interrupt was created with a configurable value. Every time the timer interrupt was called the next opcode was executed under runsnipp() using switch cases. Now the user interface program part was designed to be continuously run under the infinite while loop. So, it's kind of polling for the inputs from the user and produces corresponding outputs.

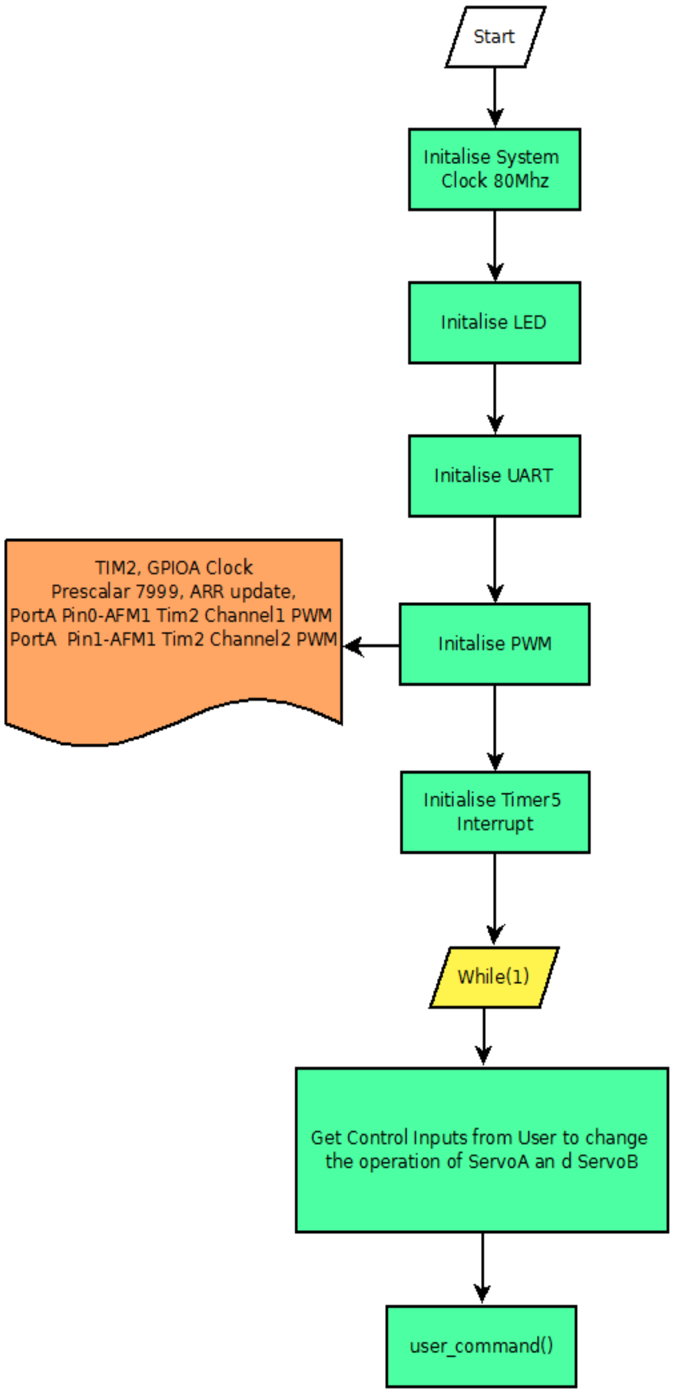

Start
Initalise System Clock 80Mhz
Initalise LED
Initalise UART
Initalise PWM
TIM2, GPIOA Clock
Prescalar 7999, ARR update,
PortA Pin0-AFM1 Tim2 Channel1 PWM
PortA Pin1-AFM1 Tim2 Channel2 PWM
Initialise Timer5 Interrupt
While(1)
Get Control Inputs from User to change the operation of ServoA an d ServoB
user_command()

Figure2: Software Flow chart main.c

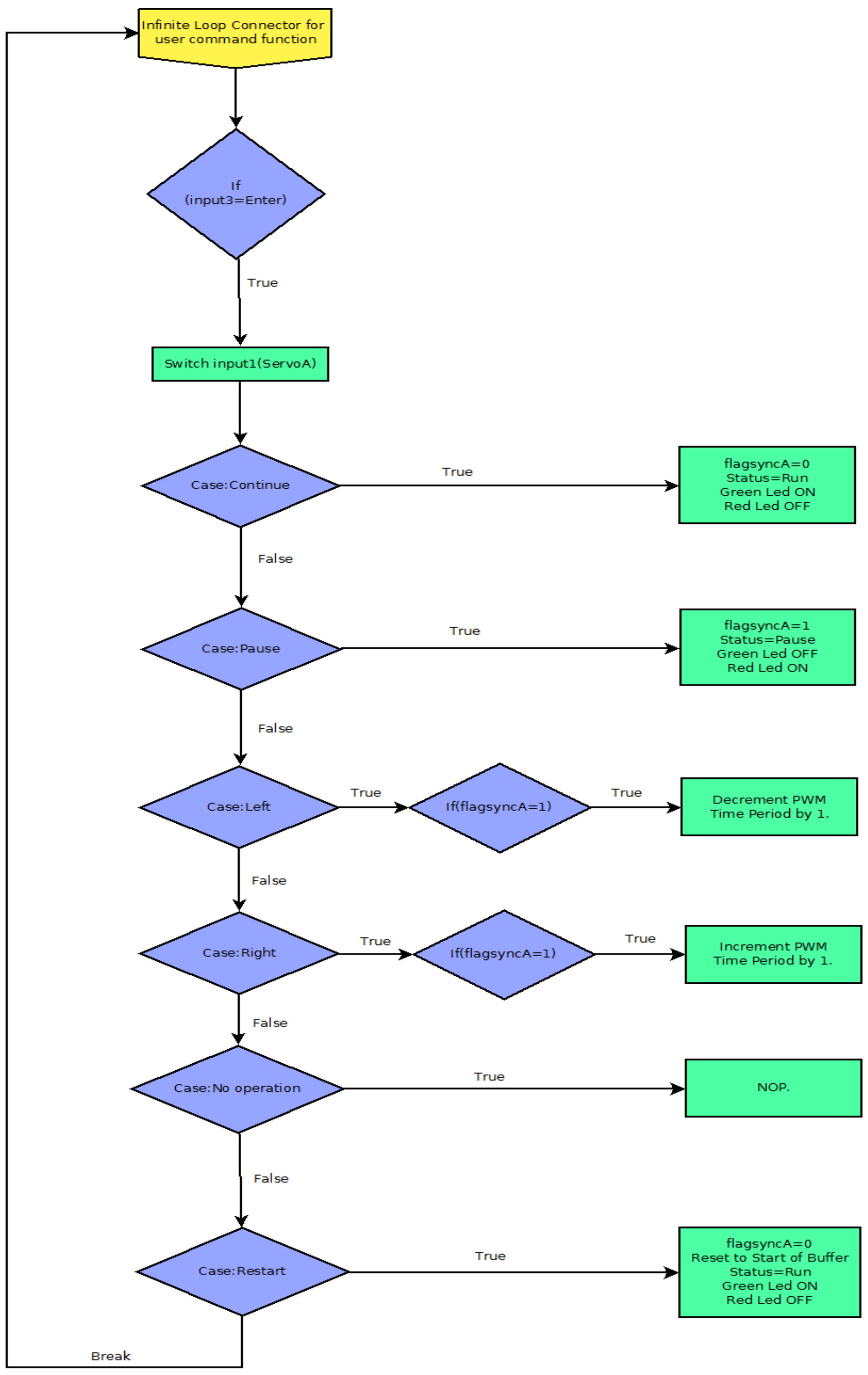


Figure3: Software Flow chart user command

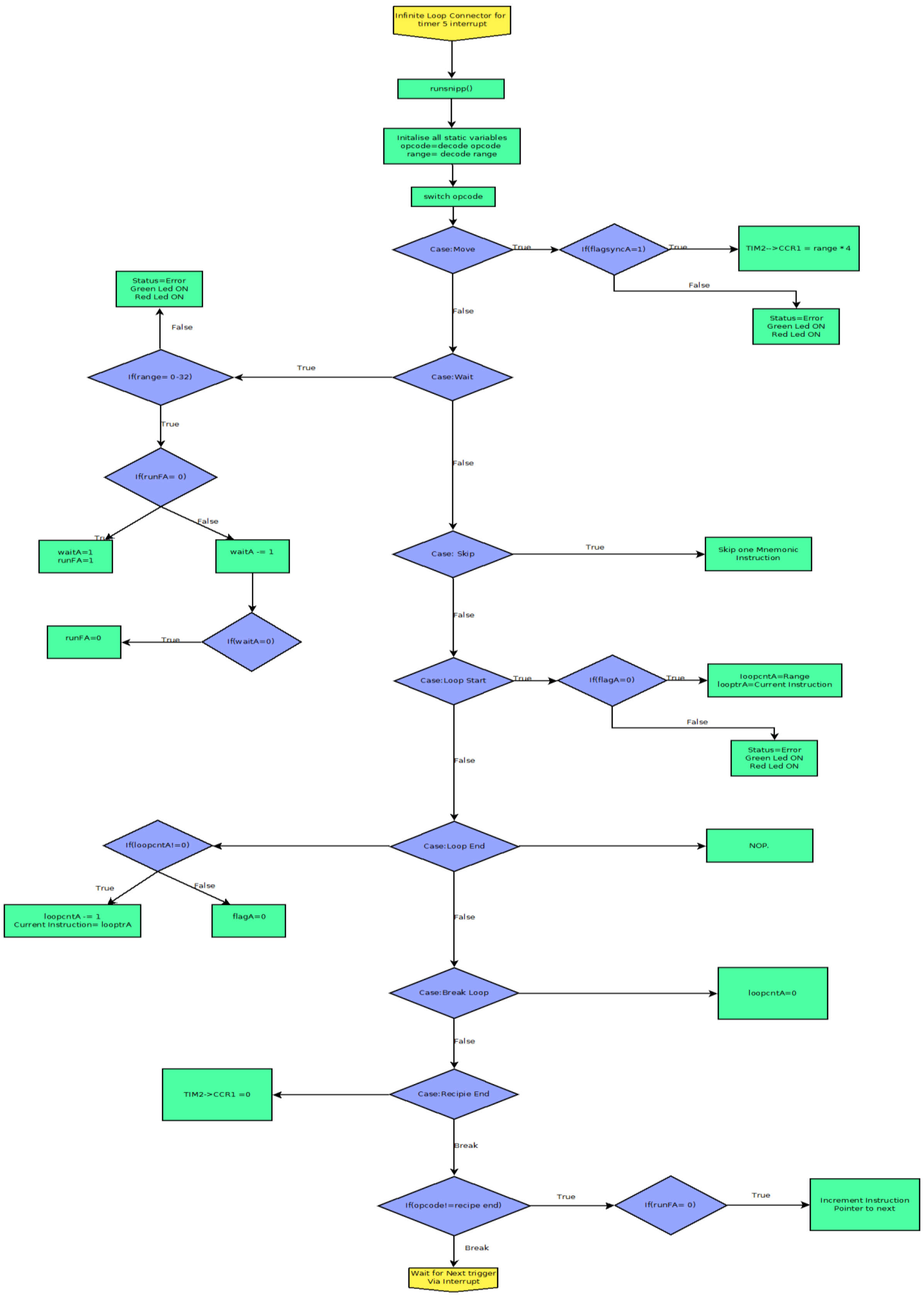
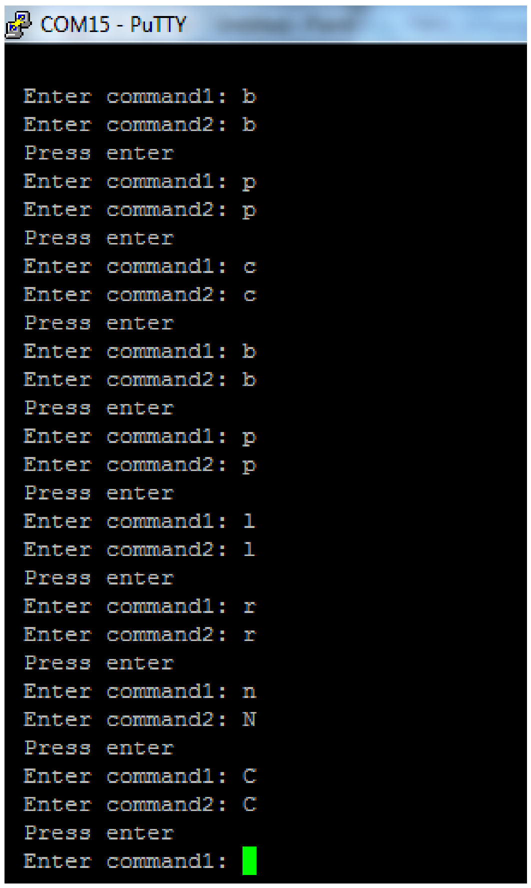


Figure3: Software Flow chart run recipe

Test Plan:

- First test the outputs of the PWM to rotate in all possible six positions with equal intervals in between. Confirm its working through a CRO before plugging the servo.
- After integrating the user interface program give all combinations of commands for testing.
- Check if the RED LED and GREEN LED are giving appropriate information as mentioned in the flowchart.
- Update different test set for each opcodes and test.
- Test positive case and negative case for both opcode and operands.

Project Results:



```
Enter command1: b
Enter command2: b
Press enter
Enter command1: p
Enter command2: p
Press enter
Enter command1: c
Enter command2: c
Press enter
Enter command1: b
Enter command2: b
Press enter
Enter command1: p
Enter command2: p
Press enter
Enter command1: l
Enter command2: l
Press enter
Enter command1: r
Enter command2: r
Press enter
Enter command1: n
Enter command2: N
Press enter
Enter command1: C
Enter command2: C
Press enter
Enter command1: 
```

Figure4: User input screen capture

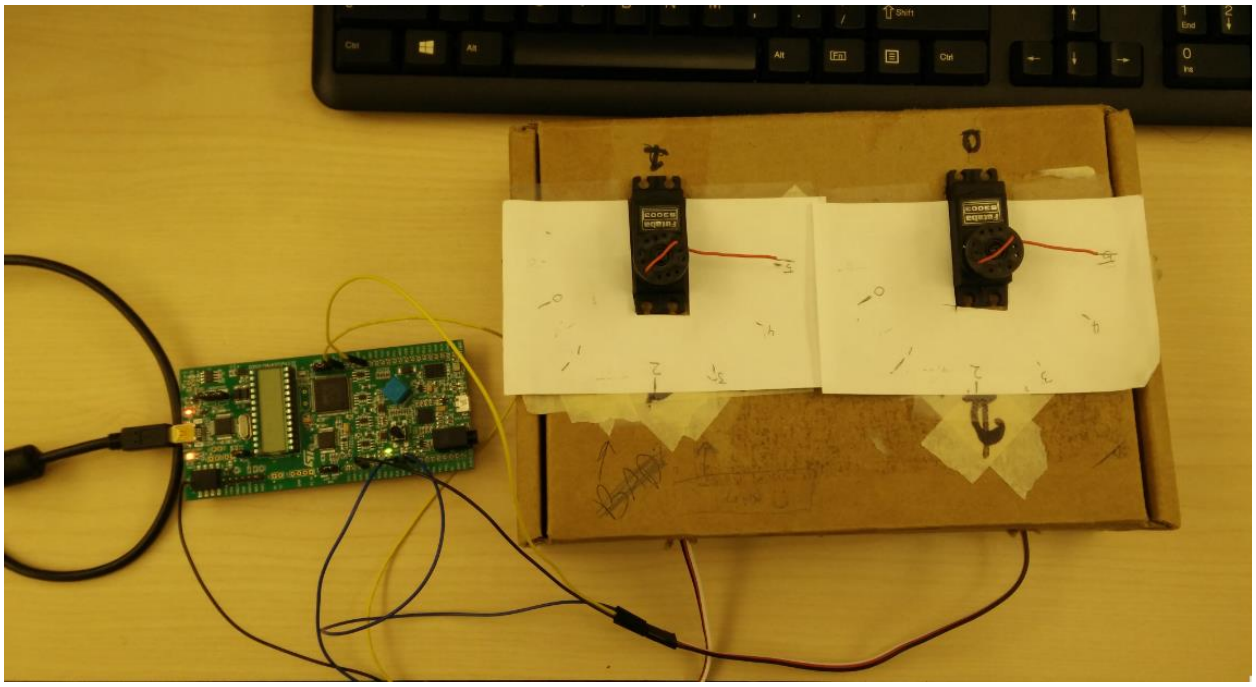


Figure5: Motor connected to the STM32

Lessons Learned:

- Working with PWM software configuration and understanding how it interact with hardware.
- Running program with multiple resource constrains all under while infinite loop.
- Modelling state transitions using the switch case statements.
- Tweaking the interrupt configuration for the timer.
- Debugging the hardware using the CRO.
- Modular coding approach.

Submission:

The controlling of the PWM servo motors using user interface and recipe of opcode was successfully developed and output was tested.