`

SYSC3310 Lab 9 – Using PWM with a Servo Motor

Fall 2018

Objectives:

- Pulse Width Modulation
- Use of the Servo Motor

Equipment:

- MSP432 P401R LaunchPad Development Kit
- MKII BoostPack
- Servo Motor

Submission: Lab9b.c and Lab9c.c.

References and Reading Material

- Demonstration program called Demo-PWM-Servo
- Demonstration program called Demo-MKII-Buttons-Interrupt

Background

Your lab kit contains a servo motor: the Tower Pro SG92R MicroServo.

"Tiny little servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds you're used to but smaller. …..

To control with an Arduino, … Position "0" (1.5ms pulse) is middle, "90" (~2ms pulse) is all the way to the right, "-90" (~1ms pulse) is all the way to the left.

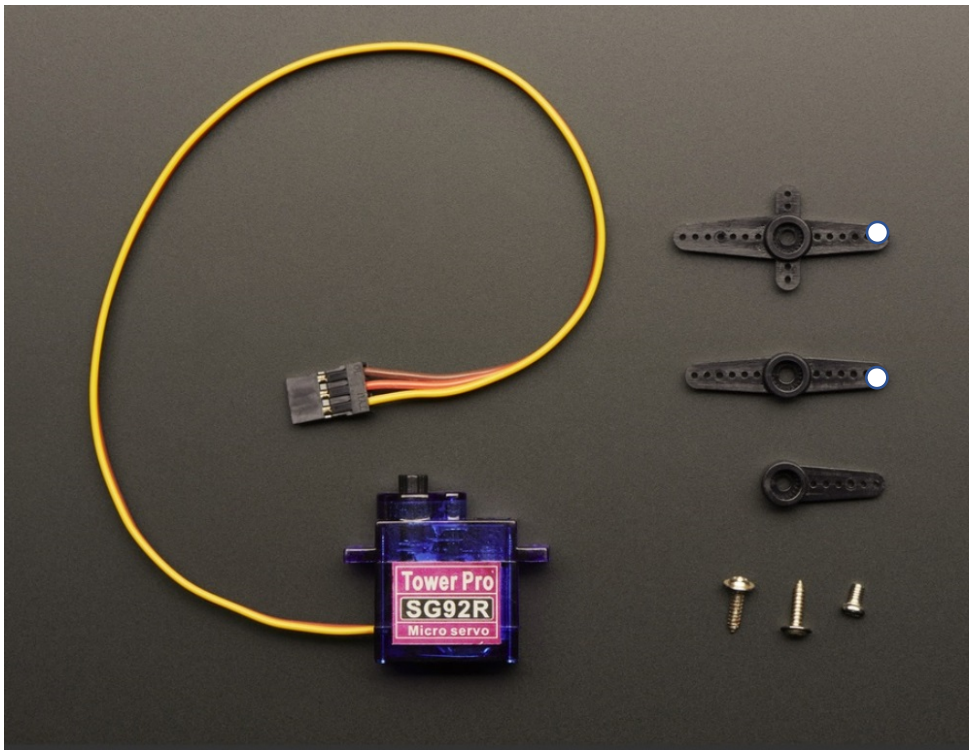
Note that the default servo pulse widths (usually 1ms to 2ms) may not give you a full 180 degrees of motion. In that case, check if you can set your servo controller to custom pulse lengths and try 0.75ms to 2.25ms. You can try shorter/longer pulses but be aware that if you go too far you could break your servo!"

Reference: https://www.adafruit.com/product/169

Its connector has three wires:

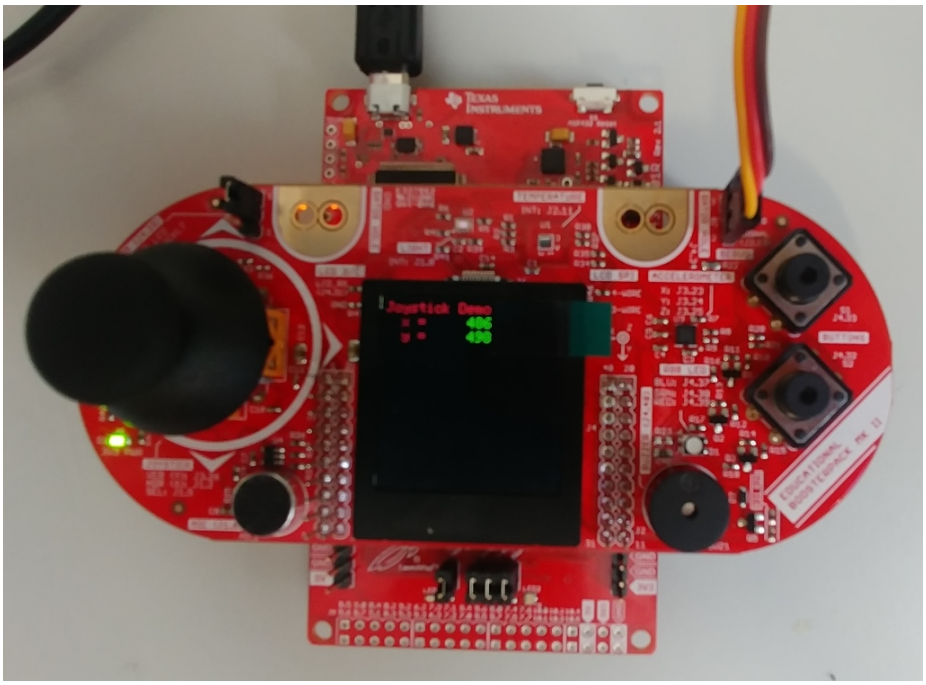
- Brown: Ground
- Orange: Power
- Yellow: Signal (connected to a port that provides PWM)

`



Tip: Put a white dot of Liquid Paper on one end of these rotors for enhanced visibility

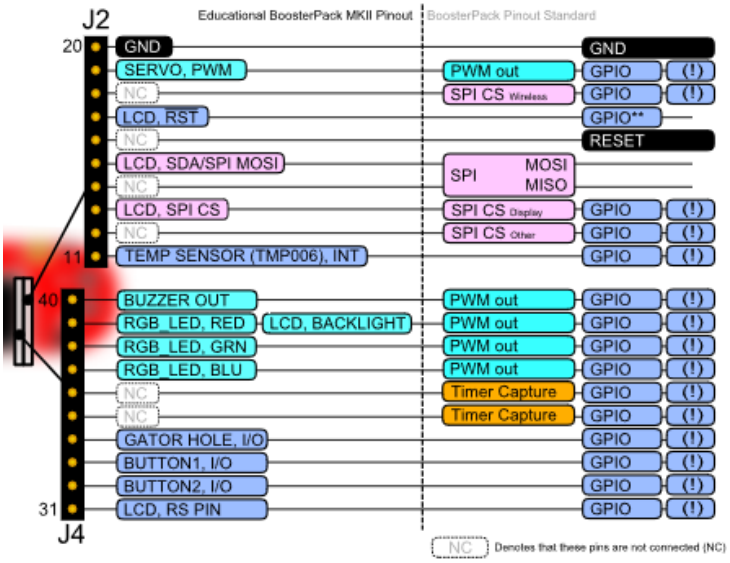
The TI Educational BoosterPack MKII has exposed 3 pins that readily accept our 3-wire servo motor in its upper-right corner. The three pins are: GND at the top, POWER in the middle and SIGNAL at the bottom. Match up the colours of the servo motor's wires to the labels of the three pins on the BoosterPack MKII.



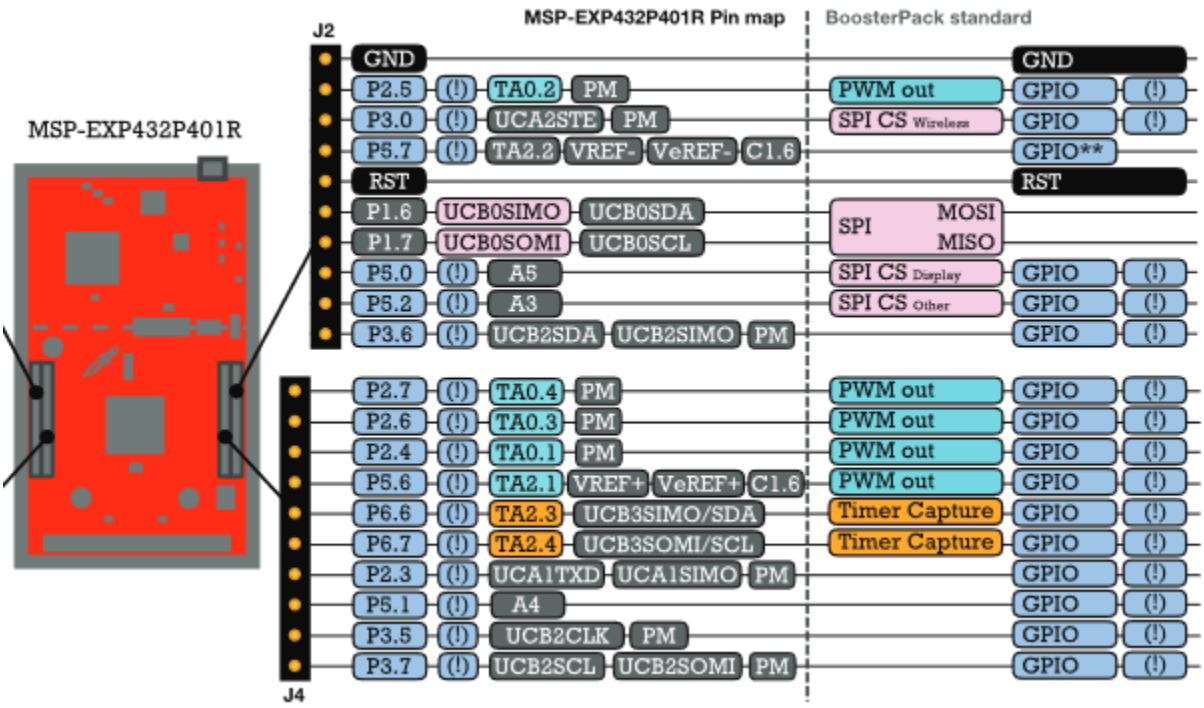
`

Before you can write code, you must know what hardware you are using. Answer the following questions and keep the answers handy.

1. What is the pin number of the Servo SIGNAL pin on the Educational Boostpack MKII?



2. To what pin on the MSP432P401R is it connected? Take note of both the Primary and Secondary and Tertiary functions of this pin.



Answer: SIGNAL is J2.19 which corresponds on the MSP432P401R to P2.5 (TA0.2, Primary Mode)

`

Part A: Running the Demonstration Program

- *Create a project called **Lab9a***
- *Tip: If not already done, put a bit of LiquidPaper on one end of the rotor, so that the direction of movement is visible.*

You are provided with a demonstration program called **Demo-PWM-Servo**. Run this program to prove to yourself that your connection is correct and your servo motor is working. There should be continuous movement of the motor from +90 degrees to -90 degrees and back again. If you detect any pauses in motion, your motor's limits are being exceeded and you need to study the code to change some of the loop limits therein. Specifically, change the values of **PWM_PULSE_MAX** and **PWM_PULSE_MIN** (increasing or decreasing them in steps of 100) until there is smooth continuous movement on your servo.

Part B: Preparing the Skeletal Structure of a Servo Control Program

- *You are to use a project called **Lab9b** for your solution.*

Instead of continuous motion of the motor, we want to be able to steer our motor to three different positions

- MKII's Button 1: Turn Motor Left (-90 degrees)
- MKII's Button 2: Tur Motor Right Right (+90 degrees)
- MKII's Joystick SELECT: Centre at home (0 degrees)

You will write a completely interrupt-driven program.

Demonstration (and Typical Exam) Questions:

1) What are the MSP432P401R ports for each of the three input controls?
2) How many ISRs will you need? Justify with a hardware rationale based on the ports.

In this Part B, **you are to add only the skeleton framework for your interrupt handlers** to (a copy of) the demonstration program used in Part A. The intent is for you to prove that you are writing code incrementally. In the next Part C, you will complete the logic of the program.

- *Tip: You can re-use parts of an earlier lab that used these same buttons.*
- When you run your program, the motor should behave identically to Part A. When you push on the buttons, nothing should happen yet (but the program shouldn't crash either)
- To prove that you have installed the ISRs correctly, you will demonstrate to the TA the **setting of breakpoints** on the first statement of **each** ISR. Pushing of the corresponding button will cause the breakpoint to be reached; hitting RUN will let the program continue on so you can test the other buttons.
- Don't forget that minimally, each ISR must clear the flag associated with its interrupt source.

`

Answer to Posed Question above (just in case you need it): B1 is P3.5, B2 is P5.1 and SELECT is P4.1 Because the buttons are on 3 different ports, three ISRs are needed; if the buttons had been on the same port (such as SW1 and SW2 both on Port 1), then only one ISR would be needed.

Part C: Completing the implementation of the Servo Control Program

- *You are to use a project called **Lab9c** for your solution.*

You are to now complete the Servo Control Program described in Part B.

- Mostly, you simply have to move pieces of the existing code Part A's demonstration program from the main() loop to the appropriate ISR
    - By the end, the main() function must have an empty loop (or simply contain WFI)
- The simplest solution is to use global (i.e. static) variables shared by all ISRs to store the current high-or-low pulse values.

Discussion for afterwards: The ISRs will have loops within them, to incrementally adjust the pulse width to effect the change in position of the motor's rotor, as well as a delay loop. This all takes time, human-length time. What if you push a different button while the ISR for a previous button is still running? Is there another solution we could try?

Part D: [Optional Challenge Exercise] A Joystick Servo Control Program

Write a program so that the servo motor mimics the directions of the joystick, with the SELECT button being used to reset the motor to its home (centre) position.

`

Marking Scheme -13 marks

Part A: 1 mark for Demonstration. No submission needed

Part B: (7 marks)

Demonstration: 2 marks. Breakpoints in each ISR, execution stops when button pressed; in the meantime, motor continuously oscillates in the background.

Inspection: 5 marks

- 2 marks: Installation of all ISRs (NVIC and PxIES, PxIFG) as well as GPIO init of the three switches
- 2 marks: ISR must contain ONLY the code to clear the particular interrupt, and optionally a static counter; Zero marks for Part B if ISRs contain any logic for controlling the servo
- 1 marks: Usual Overall marks for style

Part C: (5 marks)

Demonstration: 2 marks: When no button is pushed, the motor is quiet. When a button is pushed, correct action performed on motor. If button is pushed a second time, the servo motor should remain in its place.

Inspection:

- 2 marks: ISR must be the same as Part B plus code for moving motor to desired direction
    - Code may use a loop (for slow movement) or directly change the PWM duty cycle (for quick direct movement)
- 1 mark: Usual Overall marks for style (use of variables and constants instead of hard-coded values)

Overall marks for style:

Comments, indentation, and well-named functions and variables

- Removal of all extra code (no commented out sections)