Lab Assignment 5

Controlling the Robotic Arm by Hardware

Zhenming Yang, Connor McEleney

[yang.zhenm@northeastern.edu](mailto:yang.zhenm@northeastern.edu), [mceleney.c@northeastern.edu](mailto:mceleney.c@northeastern.edu)

Instructor: Prof. Zagieboylo

[d.zagieboylo@northeastern.edu](mailto:d.zagieboylo@northeastern.edu)

Due Date: October 19th, 2023

Submit Date: October 19th, 2023

**Abstract**

In this lab a control circuit was constructed to control a robot arm’s various joints individually using PWM signals aligned with the 50Mhz clock of the DE1-SoC.

# Introduction

The objective of this laboratory experiment is to create a digital circuit capable of generating Pulse Width Modulation (PWM) signals for regulating the Radio Control (RC) servos on a robotic arm. Users will have the ability to adjust the duty cycle of these signals using push buttons and switches. When the robotic arm is linked to the DE1-SoC board, the push buttons will enable users to manipulate the current position of a particular joint on the arm [1].

# Lab Setup

## Pre-Lab

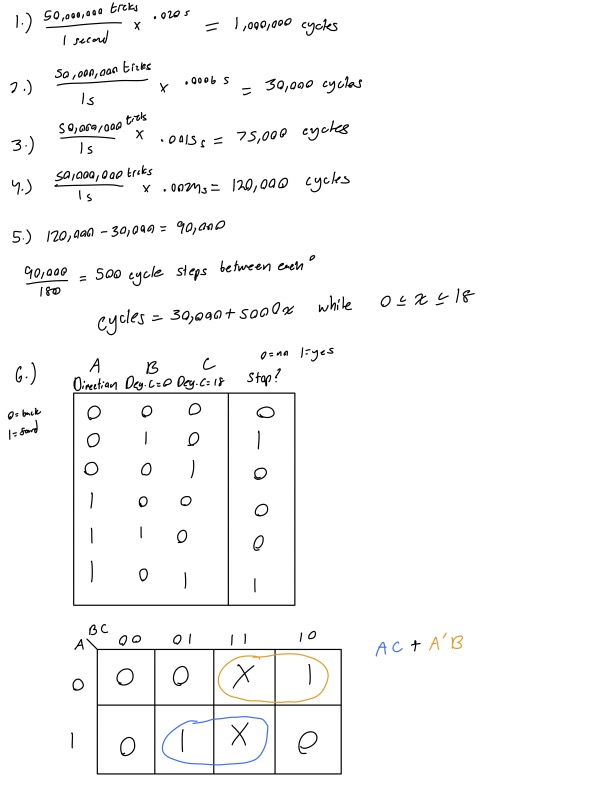


Fig 0: Prelab Work

## Equipment

DE1-SoC:

* The DE1-SoC is a hardware design platform built around the Altera System-on-Chip (SoC) FPGA. The DE1-SoC is designed for experiments on computer organization and embedded systems. It includes embedded processors, memory, audio and video devices, and some simple I/O peripherals.

# Results and Analysis

**Results**

## Part 1: Robotic Arm and DE1-SoC Interface

We utilized the multi-color data cable to establish a connection between the Robotic Arm and the DE1-SoC board by connecting it to the GPIO1 (JP2) port. The end with 5 & 5 pins was attached to the JP2 port on the DE1-SoC board. Notably, the pair of empty pin holes was aligned with the lowest pair of pins on the JP2 port, even though these two pins were not in use. The cable connector was firmly affixed to the lowest set of 6x2 pins on the JP2 port.

We then ensured that the small triangle on the connector was oriented upward, with the two empty pin holes situated on the lower side to the right, although they were not visible.

## Part 2: Controlling One Servo

For this lab segment, we designed an “OneServo" Quartus Schematic to create a PWM signal for servo control. Using a 50MHz DE1-SoC board, we established a 20ms period using an "LPM\_COUNTER" named "pwm\_period" driven by the "CLOCK\_50" input.

The core of our servo control was the "DutyCycleComparator," an "LPM\_COMPARE" block named "DutyCycleComparator." Its purpose was to output logic "1" when "pwm\_period" was less than a specific value, representing the desired duty cycle for servo control.

To achieve precise 90-degree servo movements, we **configured the "DutyCycleComparator" second input to be 7500 based on our calculations in the prelab**. The "PWM\_Out" output signal connected to the servo's control pin, ensuring precise and reliable 90-degree movements (Figure 1).

A computer code with a few lines

Description automatically generated with medium confidence

**Figure 1**: The schematics of rotating the base of the robot arm to the 90-degree position. Noted the “DutyCycle” constant is 7500 based on our calculation in the prelab.

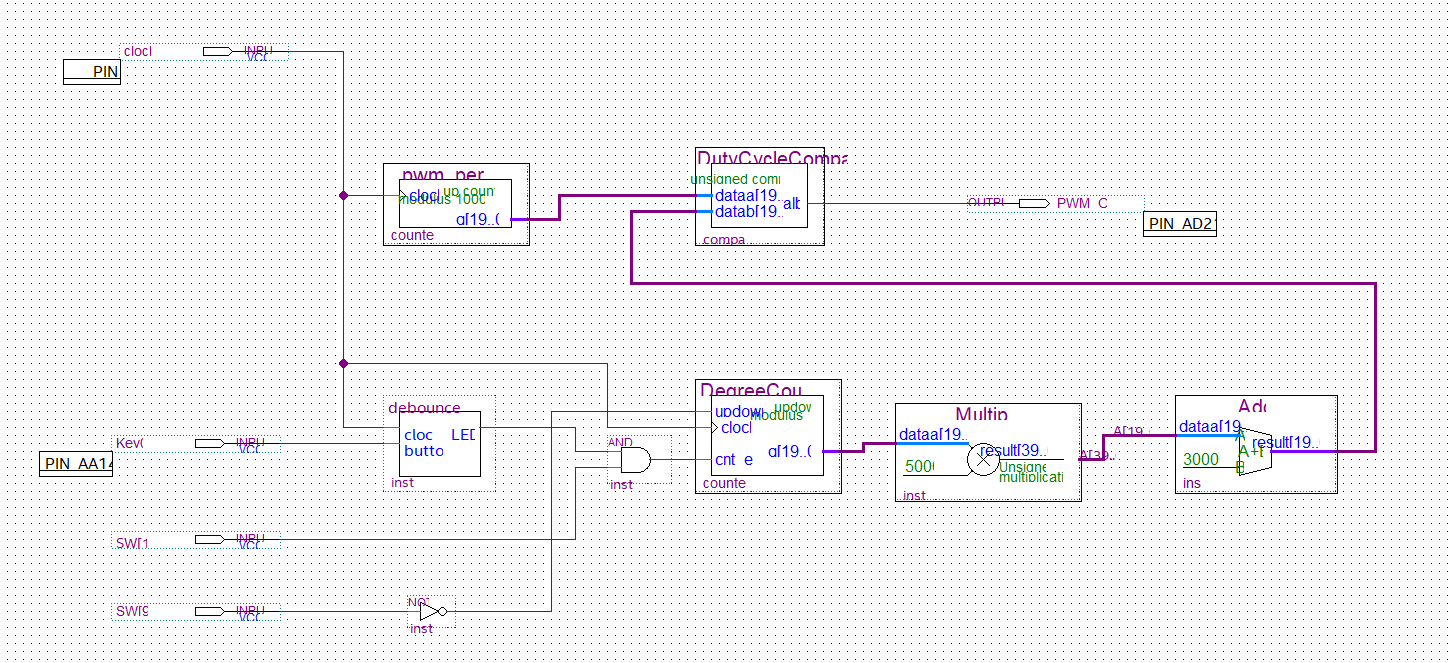
To make the robot turn 180-degree, we simply **changed the value for the "DutyCycleComparator" second input to be 12000**, based on our calculations in the prelab (Figure 2). We tested both scenarios on the robot and they all achieved the correct position as intended.

A diagram of a computer network

Description automatically generated

**Figure 2**: The schematics of rotating the base of the robot ram to the 180-degree position. Noted the "DutyCycle" constant is 1200 based on our calculation in the prelab

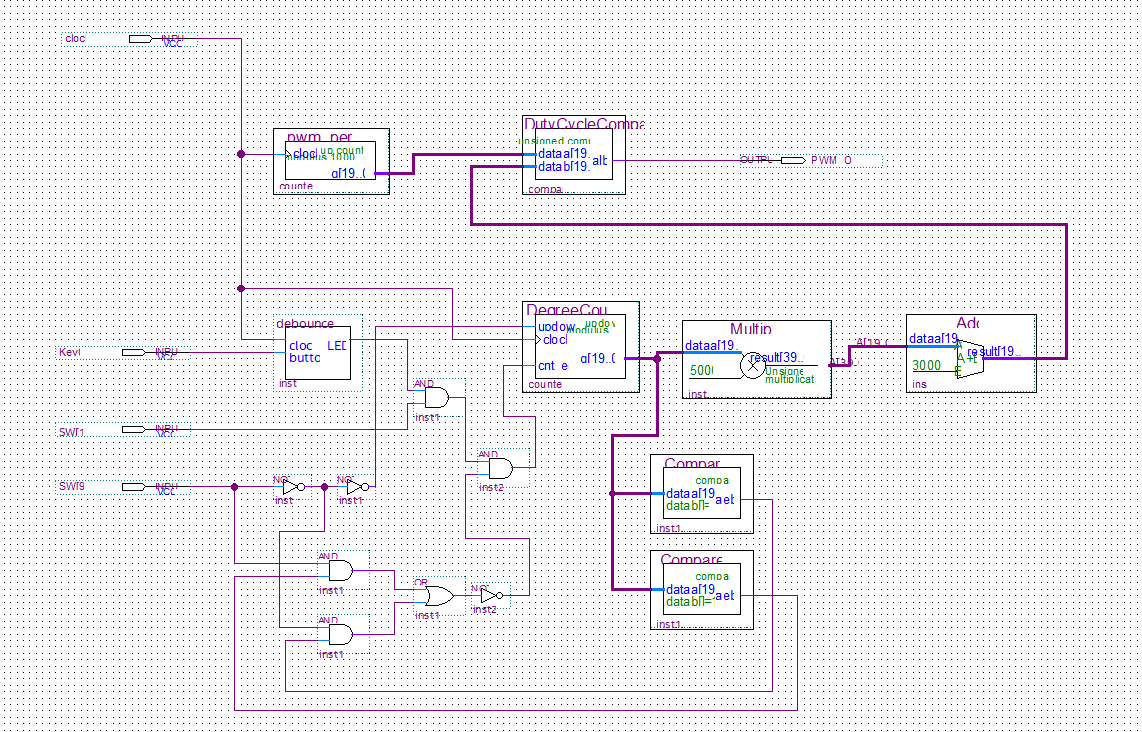
We then modified our schematics to **add the debounce\_p module to integrate the Key button** to control the rotation. Two switches are also used to first enable the movement of the robot and determine the rotation direction. Based on our calculations and logic design in our prelab, we created the schematics of this **function using a degree counter to accumulate signals** and goes **into the equation we got in the prelab to get the position number** we need to feed the it the second input of “DutyCycleCounter.” The **reverse switch is** **connected to the “updown” input of the “DegreeCounter”** and the and the **enable switch is connected to the output of the debounce with an “And” gate** (Figure 3).



**Figure 3**: The schematics of rotating arm using push buttons. The two switches enable the rotation and tell the direction of the rotation.

However, this design has a major flaw. When the robot arm rotates out of the 0-to-180-degree range, the robot arm **will rotate violently to the opposite threshold**. This is because our input for the “DutyCycleComparitor” does not have a range limit and it will just take the modulus of the new number and reset the position.

We implemented a new logic to stop enabling the “DegreeCounter” when the position of the arm is out of the range. Based on our calculations in our prelab, we **implemented two boundary comparators that turn true if the output of the “DegreeCounter” is out of range** (Figure 4).

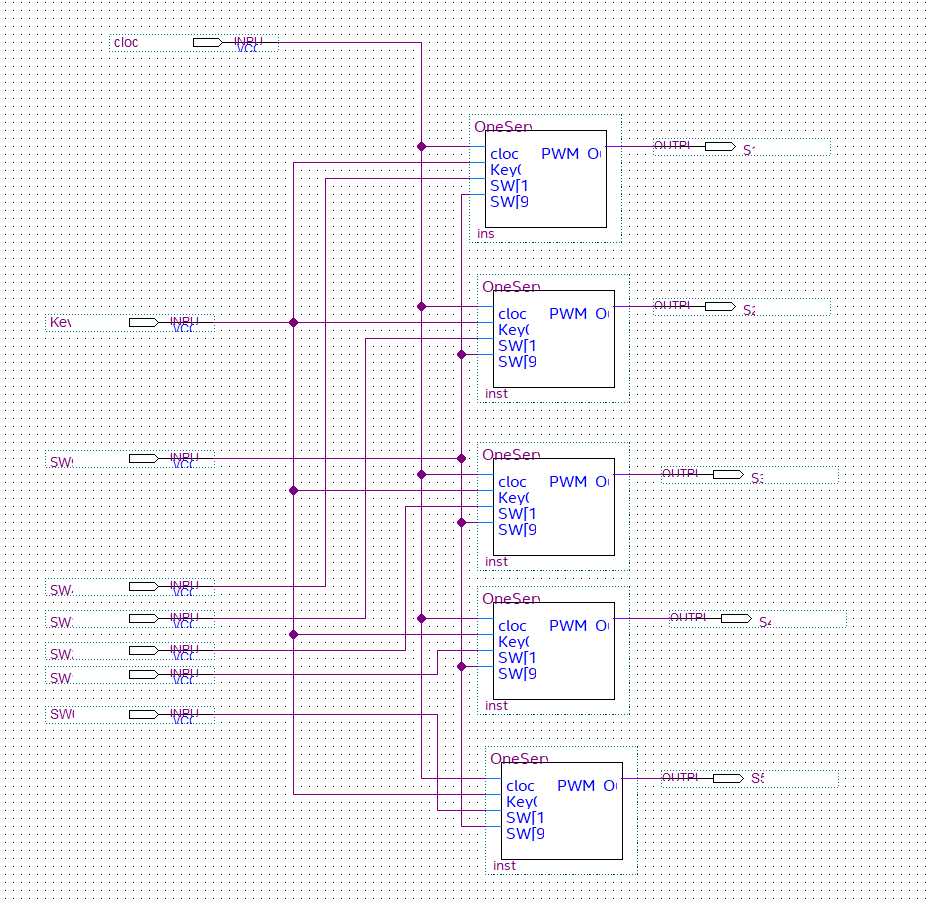


**Figure 4**: The schematics of the push button control with bound detection. Now the robot arm will stop moving when it reaches its maximum degree of rotation.

We tested all our cases on the robot arm and packaged the final one servo control schematic for future usage.

## Part 3: Controlling All Servos

Now after bundling the one servo control, we can implement it in our new design that controls all the servos on the robot arm. We first created a new project and created a new schematic. We imported the one servo control and copied it five times. Each module is connected to an output that we **assigned to as the GPIO of the board** connecting to the robot arm. The **clock input for each module is connected to the same 50Mhz clock** on the board and the enable switch is **connected to five different switches**. They all **share the same push button** and **the same directional switch**, and we connected them accordingly to our design (Figure 5). We tested our design and successfully controlled the robot to pick up a pencil and “stabbed” it into Connor’s hand.



**Figure 5**: The schematics of the 5 servos control that enables us to control the entire robot arm using six switches and one push button.

**Analysis**

We had some trouble figuring out the logic that enabled us to stop the servo from going over its limit because the logic we got from the prelab is a little different from our implementation of the rest of the system. We used a not gate for the directional switch and forgot to not it back to feed into the “updown” input for the “DegreeCounter.” This mistake made the system only turns when it is in over the boundary and is the contrapositive of what we wanted the system to do. We spotted the problem fairly quickly and added the very needed “Not” gate back to reverse our input to make the robot arm work as what we intended.

# Conclusion

This lab introduced us to data representation using PWM signals. Instead of a direct input the duration that a digital signal is on per cycle is correlated with the number of degrees the arm of the robot turns. The specs of the components we used in the lab were calculated in the prelab while integrating the DE1-SoC's 50MHz clock used to generate the PWM waves. Skills and components developed in previous labs were layered on top of one another to achieve full control of the robot's joints with an overextension detector. This lab developed the idea of different means of data representation with PWM signals and further developed our hardware compilation/layering skills.

# References

1. DE1-SoC User Manual