Sword Fighting Robot Arm

Project Report

Student name:Fraser Forbes, Michael Wu

Supervised by: Jeff Krahn

Fall/2018

Table of Contents:

1.0 Overview and Introduction

In the Sword Fighting Robot Arm project, we used VHDL to control the position of a robotic arm that was wielding a sword, so that we could have a fencing training partner. This project refined and taught us more about VHDL such as the use of clock cycles, servo motors, ADCs, sensors, and state machines. We use the FPGA to control the servo motors in the arm which made the arm "fight".

Depending on the input of the IR and Force sensors, the FPGA interfaces with other components to move the sword to expected position. Through the state machine we called decider, the FPGA loads and passes the correct PWM to the servo_controller. The servo_controller's primary purpose is to pan and tilt the arm, as well as attack with the sword. While the FPGA outputs signals to the servos, it also getting feedback from the sensor, to decide where to move the arm next. There is one more component that is needed for all the parts to work. We called it wrapper file. Its job is to port map all the inputs and outputs and signals from the different files and to be the top level entity through which everything acts.

The arm consists of five servo motors attached together with machined and folded aluminum segments. There is also a spring on the servo motors that make up the shoulder joint to help compensate for the high levels of gravitational torque on said servo motor.

The sword is a ~60cm long piece of cardboard with a T-shaped cross section. The force sensor along the top of the T and acts as the edge of the blade.

The arm begins holding the sword vertically up in the middle, waiting for an opponent to approach. Once an opponent is within range of the IR sensor, then the sword will begin its attack. If the sword strikes something, the force sensor will send a logic high to the FPGA, which will then tell the servo motors to go back to the position they came from, and try a different attack. If the sword does not make contact with anything, it will move to either the right or the left, raise up, and try another attack. If the opponent leaves the range of the IR sensor, the arm will return to the "alta" position, with the sword held high and in the middle.

2.0 Description of sword fighting robotic arm system

2.1 Specification

Our robotic arm is having following specification

1. Degree of Freedom: 5
2. Voltage range: 4.8-6.8 Volt
3. Amperage range: 0.6 - 3.8 Amps
4. Maximum Reach (without sword): 0.435 m
5. Maximum Reach (with sword): 1.125 m
6. Rated speed:0-0.3 m/s
7. Joint Speed: 0-60 rpm
8. Hardware Interface: USB
9. Control Language: VHDL
10. Shoulder Rotation: 270° maximum, 60° utilized
11. Shoulder Flex Pitch: 180° maximum and utilized
12. Elbow Flex Pitch: 270° maximum, 135° utilized
13. Wrist Flex Pitch: 180° maximum and utilized
14. Wrist Rotation Spin: 180° maximum and utilized

2.2 list of parts & sensors

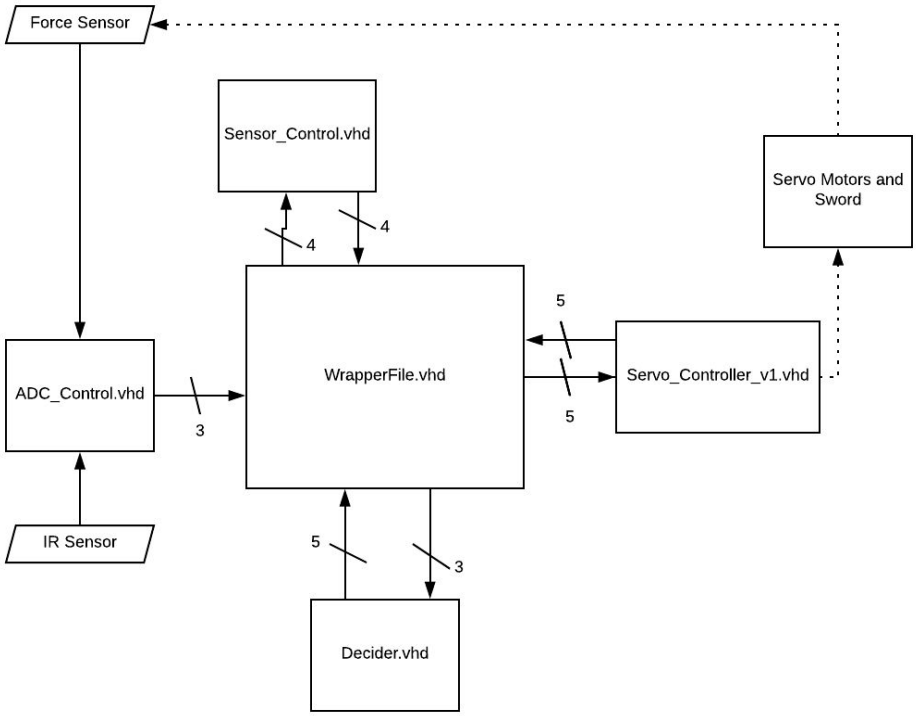
Parts Purchased	Parts Used	Individual Item Cost (CAD)
Ohmite-FSR02CE Force Sensing Resistor (2)	Ohmite-FSR02CE Force Sensing Resistor (2)	18.85
ANNIMOS 25kg Digital Servo (1)	ANNIMOS 25kg Digital Servo (1)	36.99
ANNIMOS 20kg Digital Servo (3)	ANNIMOS 20kg Digital Servo (3)	25.99
	HITEC 805BB+ (3)	28.25
	HITEC 485HB (3)	26.98
DE10-Lite Board (1)	DE10-Lite Board (1)	86.25
	Kuman Electronic Bundle (wires, resistors, etc.)	16.86
	Sharp GP2Y0A60SZLF Analog Distance Sensor(1)	15.60
Total Purchased for this Project		152.66
Total Broken		83.48
Total Cost of Final Arm		250.29
Total Cost (Everything Purchased, broken, and used)		437.06

We unfortunately broke 2 HITEC 805BB+ Servo Motors, and one HITEC 485HB. I was not being careful when I originally set up the circuit for the arm, and because of this I did not have any safety fuses in place in case of a current surge too great for the servo motors to handle.

I am not certain where the mistake was in the circuit, but in any case, too much current got to the 805BB+ acting as the shoulder flexor, and a 485HB acting as the elbow flexor. They stopped responding and never started responding again. The other 805BB+ broke because we had it as the shoulder rotational servo, which rotated in the z-x plane. This was before we had a spring in our system to help alleviate the excess gravitational torque on the arm. This gravitational torque on the shoulder rotator servo caused the rotational restrictor inside the servo to snap. It now will rotate continuously in either direction.

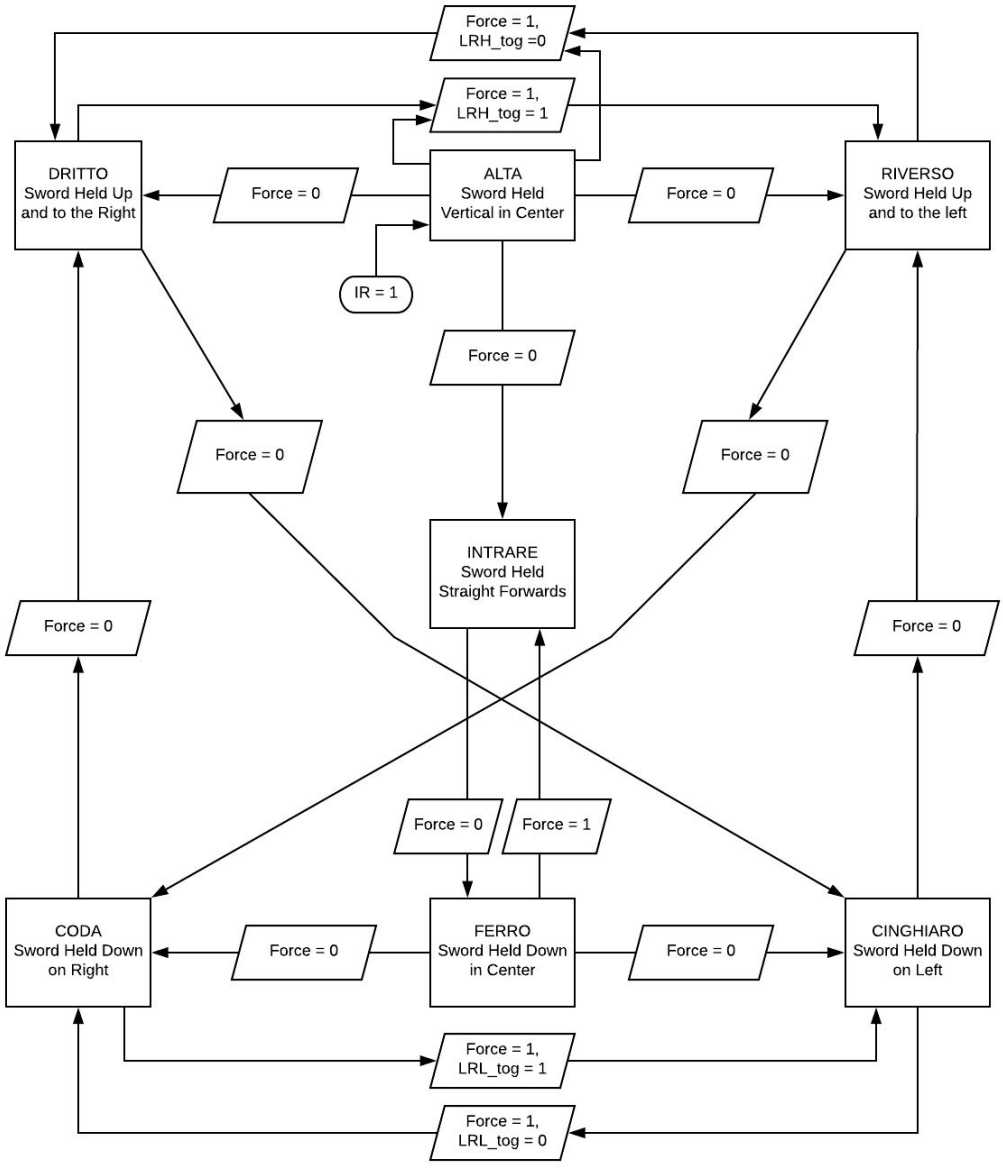
3.0 Methodology/Coding

3.1 Code Block Diagram



The block diagram above is of the .VHDL hierarchy with the WrapperFile.vhd being the top level entity. The wrapper file is what all the components are port-mapped through. Not visible in the block diagram is the pos_selv2.vhd file which we quickly wrote to test the different states/positions of the arm individually without it attack us. When utilizing the pos_selv2.vhd component we would remove decider.vhd from the wrapper file by commenting out the decider component and port map, and uncommenting the pos_selv2.vhd component and port map. This was found to be a quicker solution as opposed to coding a switch that would alternate between the two .vhd components in question.

3.2 State Diagram



Not drawn in the state diagram for ease of reading was a “Force = 1” loop which would loop back to the state the sword was last in, in the case of the sword encountering a force.

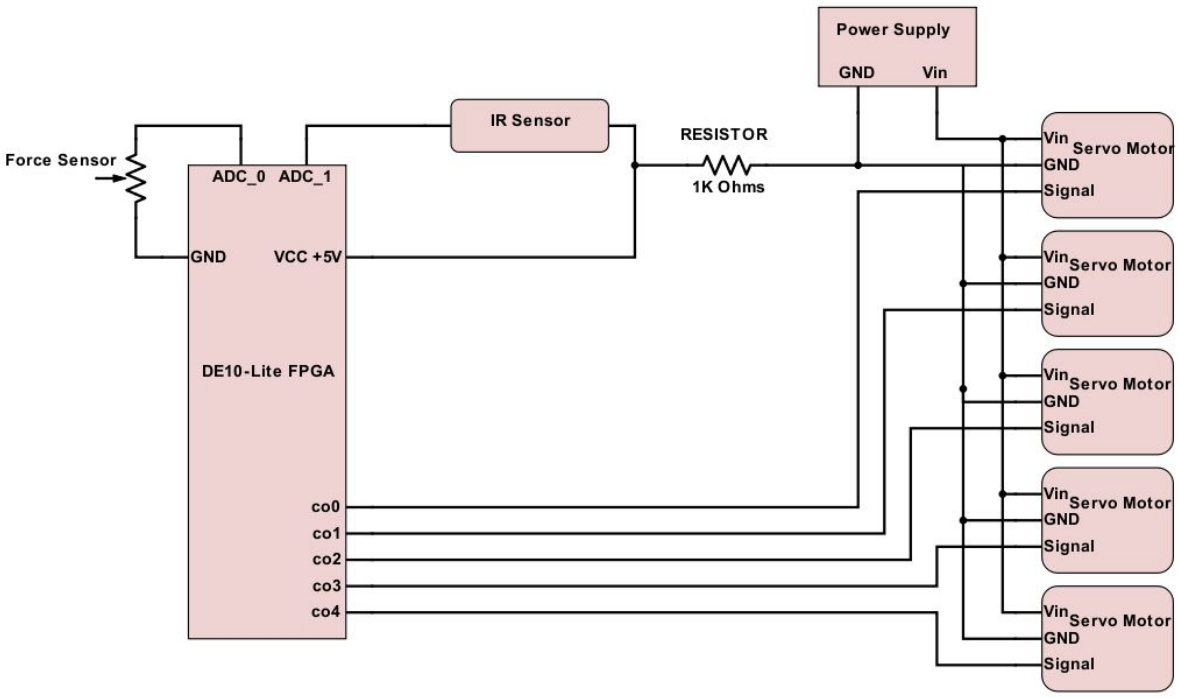
3.3 List of Inputs and Output Pins

To	Input/Output	Value
clkw	Input	PIN_P11
co0 (Clock out to Shoulder Rotational Servo)	Output	PIN_AB5
co1 (Clock out to Shoulder Flexional Servo)	Output	PIN_AB6
co2 (Clock out to Elbow Servo)	Output	PIN_AB7
co3 (Clock out to Wrist Flexional Servo)	Output	PIN_AB8
co4 (Clock out to Wrist Rotational Servo)	Output	PIN_AB9

- clkw: 50MHz clock signal that is port-mapped to all the components so that they can all act on the same rising edge.
- co0: clock out PWM signal to the shoulder rotational servo (servo 0)
- co1: clock out PWM signal to the shoulder flexional servo (servo 1)
- co2: clock out PWM signal to the elbow flexional servo (servo 2)
- co3: clock out PWM signal to the wrist rotational servo (servo 3)
- co4: clock out PWM signal to the wrist flexional servo (servo 4)

4.0 Parts designing and assembly

4.1 Circuit Diagram



Our robotic arm was built using 5 servo motors or varying size, weight, and strength. We had an external DC power supply that would power the servo motors with 5.5V and varying amperage. The force sensor is essentially a variable resistor, where the resistance is based off of the amount of force applied. The IR sensor also acts somewhat like a variable resistor, but in order for it to function properly it needed to be in parallel with a resistor that went to ground.

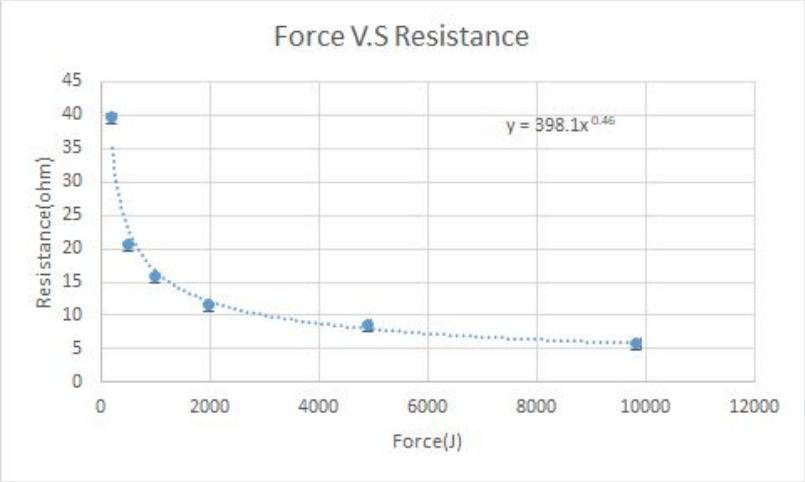
4.2 Interfacing the Sensors

Calibration of Force Sensor

Force Sense Resistor strip Model: OHMITE-FSR02CE
Range: 0-9900J ± 10J

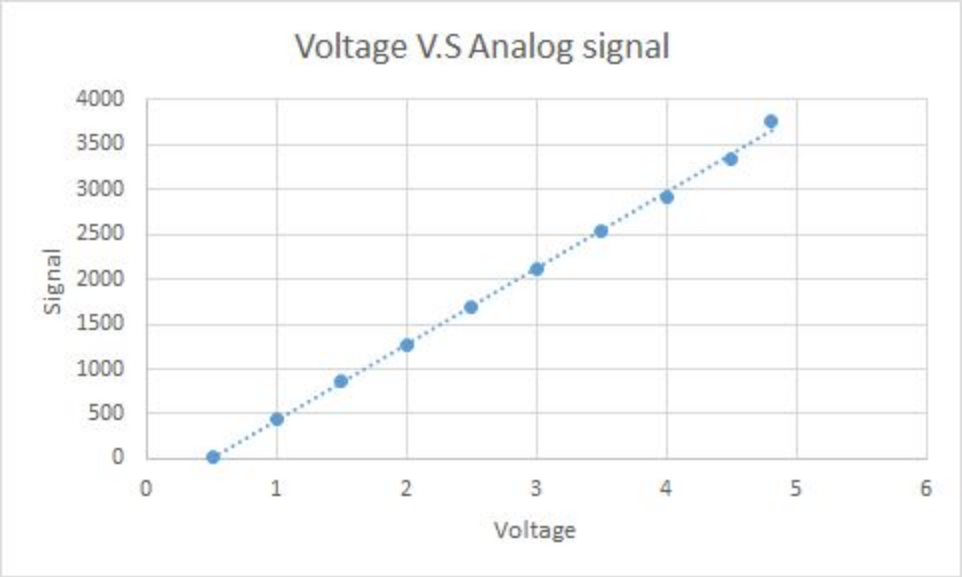
Contrast this FSR to photocells where they act like resistors and thus can be simply tested with a multimeter.

We used a lab-experiment weight set (20-1000 g ± 1%) to calibrate the force range of the sensor, by plotting the result we have the diagram below.



Force V.S Resistance of Force sensor (with error)

We can see that the relationship between the resistance and force is a power function. When no force is exert on the sensor, the resistance is infinity. Since we are using a 5 Volt source we can convert the resistance into voltage so that we can use an ADC to analysis the sensor feedback. By connecting the sensor with a 1000 ohm resistor, we can can get a 0-3.65V ± 0.07V voltage feedback. By using a wood sword against the sensor, we found that the arm should be activate when the incoming force is around 3000J, follow the properties of the ADC generated, we decide that the enable signal is activate when analog signal is greater than 001001111111.



Analog signal v.s Voltage of ADC(with error)

In the demonstration we connected the output of the force sensor with a 1K ohm resistor to FPGA. When no force is exert on the sensor, the voltage is o. When there is a force greater than 3000J exert on the sensor, the output changes to high-impedance and generate a high voltage (up to 3.65volts). Program Part - The sensor will have an Analog signal output send to FPGA when struck, FPGA will read the output, and depending on the "force" the arm will move accordingly

- "0" : Analog signal lower than 001001111111 - no force, arm keeps moving
- "1" : Analog signal higher than 001001111111 - medium force, arm disengage.

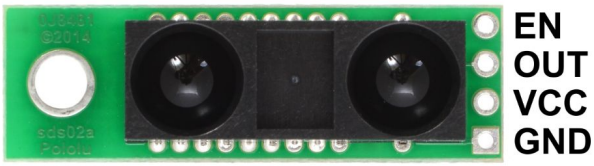
Calibration of IR Sensor

Analog Distance Sensor Model: Sharp-GP2Y0A60SZLF

Range:10-150cm ± 5cm

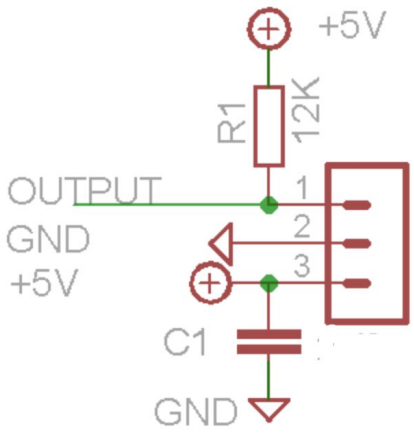
Operating voltage: 3-5 Volt ± 0.15 volt

IR sensors are a low cost, easy to use analog distance sensor. IR sensors produce a constantly updating analog output signal depending upon the intensity of the reflected IR signal, which in turn can be used to detect approaching objects. Because there is a semiconductor/chip inside the sensor, it must be powered with 3 - 5V to function. The GP2Y0A60SZLF requires a 3-5 volt power supply. A 0.1uF bypass capacitor between power and ground is a good idea, but we didn't use it in our demonstration. In the demonstration we connected the output of the sensor directly to FPGA. When an object 12cm in front of the sensor or closer, it trigger the arm and starting attack.



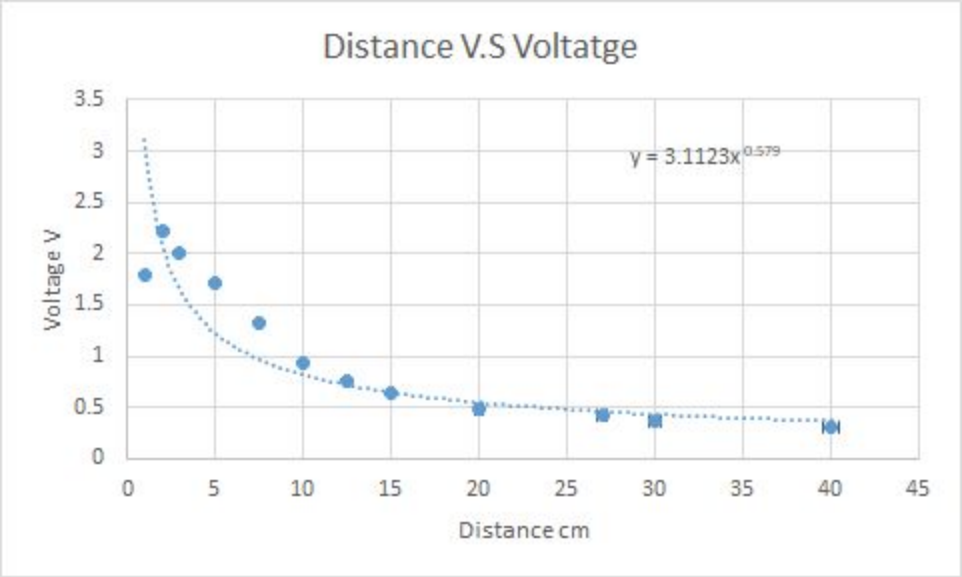
Enable pin can optionally be used to disable the emitter(not used) and the header strip was solder by Fraser.

The GP2Y0A60SZ sensor has two operating modes: 5V and 3V. In 5V mode the output voltage differential over the full distance range is approximately 2.5 V. In 3V mode, the output voltage differential over the full distance range is approximately 1.6 V. We operate the sensor with 5V mode because we want a higher output so that the the accuracy of analog signal is higher.



Circuit Diagram of GP2Y0D21YK（1- out, 2-GND, 3- VCC）

When the operating voltage is 5V and nothing is in front of the sensor, the detector holds the output low (0.40 volts). When there is a object in front of the sensor, the output changes to high-impedance and the pull-up resistor(R1) holds the signal high (up to 2.2 volts).



Output Voltage(V) V.S Distance(cm) of IR sensor

The relationship between the sensor's output voltage and the measured distance is approximately linear over the sensor's range. The graph of the output voltage over the usable distance to a reflective object is below. We can convert the sensor output voltage to an approximate distance by constructing a best-fit line that relates the inverse of the output voltage (V) to distance (cm).
We choose the respond distance range to be 8 cm, following the properties of ADC, when analog signal is 001100111111 or greater it trigger the arm to attack.

Program Part - The IR sensor will have an Analog signal output send to FPGA when someone within the range, FPGA will read the output, and depending on the "enemy" the arm will attack accordingly

- "0" : Analog signal lower than 001100111111 - no enemy within range, arm in "stand-by".
- "1" : Analog signal higher than 001100111111 - detect enemy within attack range , arm attacks.

Power Requirements

- 5 volt input for Force and IR Sensor from the FPGA
- 5 volt input to the servo motors
- Up to 3.8 amps of current drawn by the servo motors while lifting from the low states to the high states while opposed by a force.

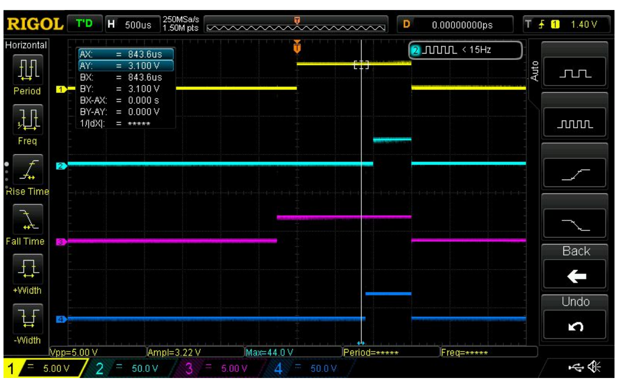
5.0 Execution result

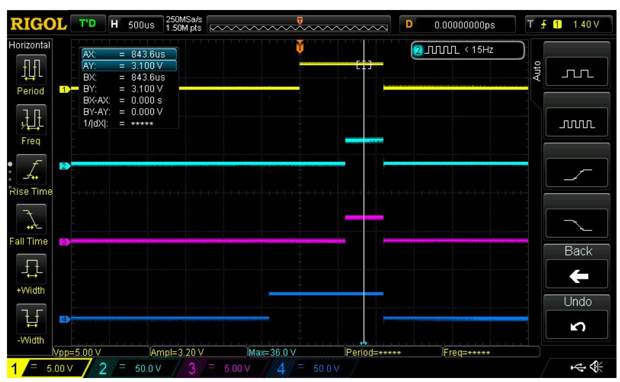
5.1 Oscilloscope data

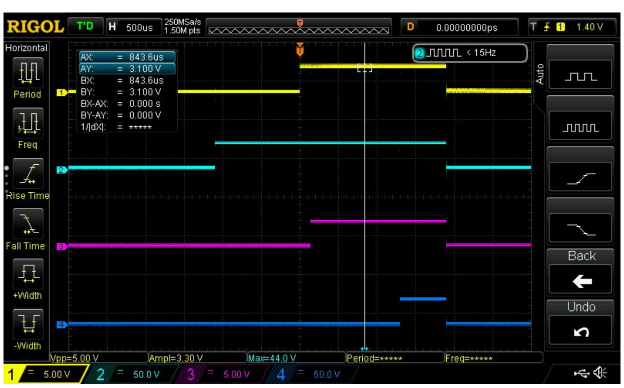
Oscilloscope data for Arm Positions

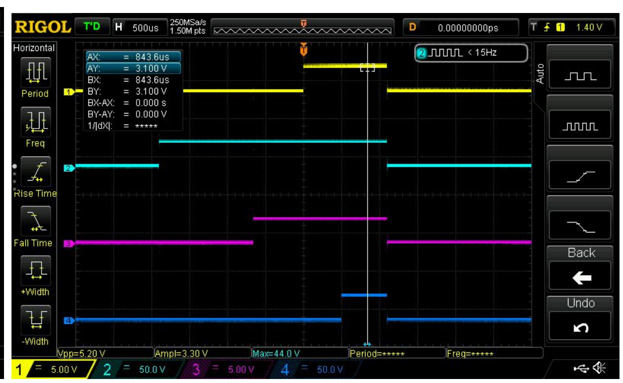
Below are oscilloscope images of the duty cycles being sent from the FPGA board to the servo motors for the 7 different states/arm positions. Listed in reading order: Alta, Intra, Ferro, Dritto, Riverso, Coda, Cing.

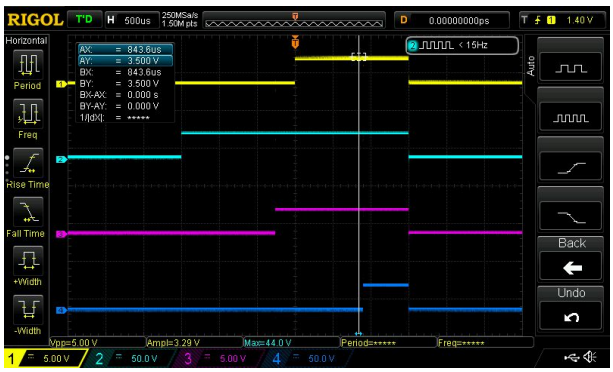
- Yellow is the shoulder rotator signal
- Cyan is the shoulder flex signal
- Magenta is the elbow flex signal
- Navy Blue is the wrist flex signal

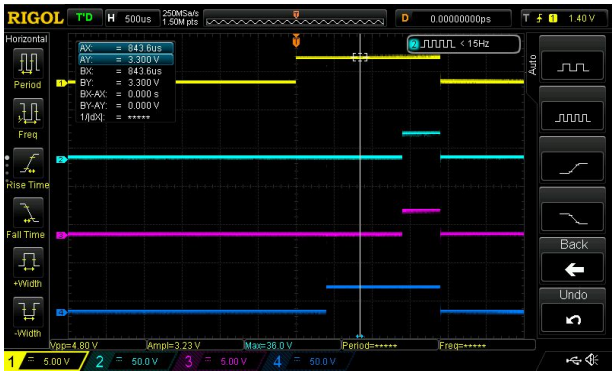






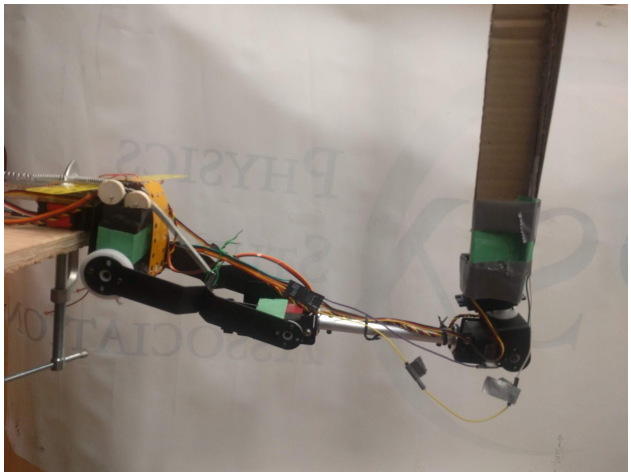






5.2 Arm assembling Process

Luckily we were able to salvage much of our arm from a past project done by Kuomond Lee and Dallas Visagie. From the past project we were able to take the aluminum “bones” they used to hold the servos together. We also used plenty of tape, cardboard, and hot glue to hold the sword together and to the arm. The arm was then clamped to a square piece of plywood, which was then clamped to the table. The final result can be seen to the right.



5.3 Reliability Issue

We had a few problems with reliability of the structure of the cardboard sword, as well as the range of the IR sensor. The cardboard would deteriorate fairly quickly, and required a few reconstructions and patch jobs with hot glue and tape. The IR sensor would occasionally flutter if the human facing off against the robot moved too drastically which would cause the sword arm to reset to the Alta state. We also had problems with the wires unplugging themselves as the arm moved about, which would shut-off certain aspects of the arm such as the elbow motor, or the shoulder rotator, or the force sensor.

During the presentation, the sword was bending too much, and was causing the arm to get stuck in a low guard since it was constantly thinking it was getting struck as the force sensor flexed.

6.0 Problems & Conclusion

We were successful in our goals for this project, which was to construct an arm that could wield a sword and react to an opposing sword. Some of the deviations from the original project proposal were the fact that we did not 3D print any parts as originally planned as we were able to salvage parts from the supplies in the back and also from past projects. Originally we wanted to have different levels of force input from the force sensor to choose different reactions, but we quickly got rid of this idea since we did not want to increase the force on the servos which could have damaged them. These changes seem very minimal to us, and overall our project was a success. An improvement we made was the length of the sword, and how quickly it reacted. I originally expected the arm to have a reaction time of ~4ms, which was the speed of the clock divider, but by keeping the force input outside of the state machine, the arm was able to react as quickly as the 50MHz clock updated.

We did also have some problems with our sword arm. From the beginning of the decider.vhd we had some major problems with the counter and clocks. The issues involved understanding clock cycle. We had the clock that the decider was acting on updating much too quickly but it took us a while to figure that out. We solved this by implementing a clock divider which slowed down the clock signal to the decider, but we kept the ‘force sensor input’ outside of the clock divider since we still wanted the arm to react as quickly as possible. I was also nervous to test the decider.vhd physically since we had many inferred latches and I thought they would damage the physical components of the arm. We also had a little confusion how the use two ADC channel was supposed to be done, but talking to Jeff Krahn he quickly guided us to the solution.

The knowledge we learned from that can be carried on to other systems that require multiple inputs and outputs. It will definitely help on our future 390 project and study of modern control. Robotic arms are also very commonly used in many industries so hopefully this will help us in our future careers.

7.0 References and Acknowledgements

- Jeff Krahn lecture, November 1st, 2018 “Registers and Finite State Machines”. *University of the Fraser Valley - ENPH 320 - Electronics*
- Jeff Krahn lab, “Lab 4: Counters and Frequency Counters”. *University of the Fraser Valley - ENPH 320 - Electronics*
- Jeff Krahn lab, “Lab 6: Analog to Digital Conversion”. *University of the Fraser Valley - ENPH 320 - Electronics*
- Jeff Krahn, “FSM_example_code” . *University of the Fraser Valley - ENPH 320 - Electronics*
- Kuomond Lee and Dallas Visagie, “Lego Colour Sorting Arm” taken apart for supplies.
- Nelson Spies, supplied us with five wire bus.