Project Lab 1 – Midterm Progress Report

Dirk Thieme

R11636727

Texas Tech University

March 2022

Abstract

This paper outlines the current iteration of the Final Project for Project Lab 1 at Texas Tech University. While at this moment the project remains unfinished, this document will contain the code and information about the working parts, as well as the ideas and implementation for how the rest of the project is meant to be completed. It will mainly focus on the Hitec HS 422 servo, the Seven-Segment Display of the Basys 3 FPGA Development Board, and the OSRAM SFH 309-5/6 phototransistor, while also discussing future improvements to those features as well as future plans to finish the robot. The Rover is designed by Dirk Thieme, Erik Manis, and Mohamed Ansari.

Table of Contents

1. Introduction
2. Body
    a. Top Module
    b. Servo Controller Module
    c. Seven-Segment Display Controller Module
    d. Future Software Ideas and Implementation
    e. Physical Design
    f. Future Hardware Ideas and Implementation
3. Conclusion
4. Appendices
    a. Budget
    b. Gantt Chart
    c. Code Snippets
5. References

Table of Figures

- Figure I – Assembled Rover – pg. 5
- Figure II – XADC Wizard Configuration – pg. 9
- Figure III – Physical Connections between Pmod port JA (above) and the H-Bridge (below) – pg. 13
- Figure IV – L7805 Circuit – pg. 14
- Figure V – Circuit Simulation Results– pg. 14

List of Tables

- Table I - PWM Output Based on Switch Configuration – pg. 11

1. Introduction

The Final Project is to create a rover that follows a strip of metallic tape and is able to differentiate between a red, 1000Hz LED and a blue, 300Hz LED. The blue LED represents a “friendly”, which is to be distinguished with an “F” displayed on the Seven-Segment Display. The red LED represents an “enemy”, which is to be distinguished with an “E” on the display. If the red LED is detected, the rover is to shoot a rubber band at the target and move on. The rover is to be created using the Rover 5 Robot Chassis as the base, with a Basys 3 FPGA as the brain of the device. The Basys Board talks to the Robot Chassis using an H-Bridge circuit, which translates the low voltage pulses of the Basys Board into stronger signals for the motors. The motors will also be protected by the use of a hardware comparator; however, the design is not finalized so specific parts are not known. The colors of the LEDs will be determined by the use of OSRAM SFH 309-5/6 phototransistors which will be coated in red or blue filter paper to ensure only red or blue light will be visible. A Hitec HS-422 servo will be used to aim at the red lights if detected, and a 6V actuator will be used to fire the rubber band. To follow the tape on the ground, HcW201 IR sensors will be placed in a triangle formation facing the ground.

The Robot Chassis is a simple platform that utilizes two brushed DC motors to run caterpillar treads that move the device at a maximum speed of 25 cm/s [1]. This speed is achieved with the help of the two 86.8:1 gearboxes [1] that give the motors enough torque to move the contraption’s weight along with anything attached.

The brain of the device is the Basys 3 Board provided by Digilent, which is where most of the complexity of this project lies. The board is powered by the Artix-7 FPGA, which features 33,280 logic cells in 5,200 slices, 1,800 Kbits of RAM, five management

tiles, 90 DSP slices, and a 100MHz clock. The Board also includes a multitude of input and output options, most notably a built-in Seven-Segment display which was used to display the direction of the Rover and the current threshold and the Pmod ports which are used to input and output signals to control the motors, read the phototransistors, control the servo, read the tape, and actuate the rubber band gun. The Board was able to interact with the Rover using an L298N H Bridge, a chip which can take in the relatively low voltage signals of the Pmod ports and output the correct forward or reverse signals to the motors. The Basys 3 Board is configured with the use of SystemVerilog, a hardware description and verification language which is used to model and emulate electrical circuits and systems [3]. SystemVerilog is an upgrade to Verilog in the way C++ is an upgrade to C; object-oriented programming is added, along with tons of other useful features that Verilog lacks. The modules are synthesized, implemented, and uploaded to the board using the Xilinx Vivado IDE, which is also where code may be written and checked for errors. The OSRAM SFH 309-5/6 is a phototransistor [6] with a 24° field of vision, that can see an LED from about two to three feet away from itself. These phototransistors will be directly wired into the Basys Board to activate when the color they are assigned to is viewed. Color filter paper will be used to prevent other light from affecting the signal. To aim at the light source once detected, the Hitec HS-422 rotate towards the indicated direction until its OSRAM phototransistor (coated in red) signal it to stop moving and fire. The HS-422 is a servo which is controlled using a PWM signal at a given frequency, in this case 400Hz, to swing between 0° and 180° in either a clockwise or counter-clockwise direction based on the width of the signal. A 400Hz signal corresponds to a 2.5ms period, and to control the position the period within the

range is altered. The datasheet [5] notes the minimum and maximum positions are at periods of 0.9ms and 2.1ms respectively, however it has been noted that in this case it is closer to 0.6ms and 2.4ms for true 0° and 180°.

2. Body

One of the most important distinctions to make before continuing with the discussions about how the Rover was created is the difference between a microcontroller, for example an Arduino or XBee3, and an FPGA, which is what powers the Basys 3 Board. While they seem similar to control, the methods of programming and interfacing with them are very different. A microcontroller, which in this case is an XBee3, has a built in CPU, RAM, and other peripherals to interact with, usually programmed with a standard language such as Python. The CPU and instructions are predefined and not user editable, so while they are still powerful, they cannot be precisely tuned for use like an FPGA. A field programmable gate array, or FPGA, is a collection of ICs which can be defined and configured by the user programming them. This presents an advantage over a standard microcontroller in speed and size, as a user can prune and optimize their design to exactly what they need, with nothing more. The current SystemVerilog modules that are up and running on the Rover consist of a top module that functions as a motherboard of sorts and two sub modules: the Servo Controller and the Seven-Segment Display. Full code samples can be found in Appendix C.

2a. Top Module

The current implementation of the Top Module is relatively small, as it is only taking in the phototransistor for the servo, then passing that through to both the Servo Controller

and the Seven-Segment Controller, along with the clock other output pins. Once complete, the Top Module will be inputting all phototransistors, a comparator signal, and infrared signal, while outputting motor control signals, a slowdown signal, and a fire signal. The prototype for some of these future signals can be seen in Figure I. The wildcard * as seen in Figure II is one of the neater features added in SystemVerilog, instead of having to instantiate each connection every time by hand, the * operator connects all ports with the same name, reducing time and simplifying code.

```
module final_top(
    input clk100,

    /*SERVO CONTROLS*/
    input servoPhotoT,      // JB1
    output servoOut,        // JC1

    /*INFRARED SENSORS*/
    input infraSensorRight, // JB2
    input infraSensorMid,   // JB3
    input infraSensorLeft,  // JB4

    /*PHOTOTRANSISTORS*/
    input RRPhotoT, // JB7
    input RBPhotoT, // JB8
    input LRPhotoT, // JB9
    input LBPhotoT, // JB10

    input [3:0] IN, // motor control pins
    output [6:0] seg,   // output for the segments
    output [3:0] an     // output for the anodes
    );
```

Figure I – Inputs and Outputs for the Basys Board

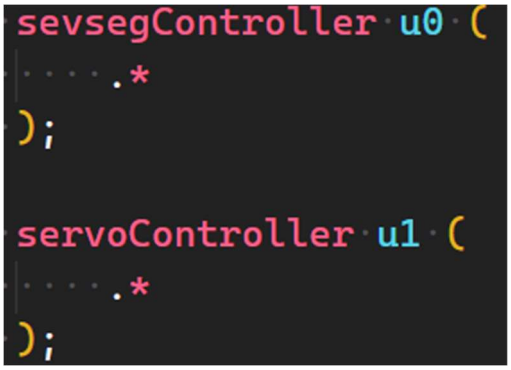

Figure II – Module Instantiation with Wildcard

2b. Servo Controller

The Servo Controller Module is similar to the Motor Controller in which it is driven using a given frequency and modified by changing the width of the pulse in that frequency. As stated previously the Servo is running at 400Hz, which after doing the math corresponds to a counter of 250,000. The pulse for the minimum position corresponds to a count of 60,000, and for the maximum 240,000. This version of the Servo Controller sweeps the servo in a 180° arc back and forth unless the servo phototransistor is triggered, where it then pauses at that location. To count that number, two registers 20 bits wide are declared, with the "pwmSize" being set to the minimum number, and the "pulseCount" begin set to zero. The logic is relatively simple, as all it does to output a position is increment the pulseCount by one until 250,000 is reached. While incrementing, pulseCount is compared to pwmSize, if it is less then a one is output to the servo, other a zero is output. This creates a square wave PWM signal that's on for example for 1.5ms, then off until 2.5ms when it resets. Figure III shows howe a sweeping motion is achieved, the pwmSize is incremented by 100 each time 250,000 is reached

until the maximum position of 240,000 is reached, in which case it is now decremented by 100 until the minimum position of 60,000 is reached, and repeats that. A simple if statement is added to check if the phototransistor is triggered, in which case the pwmSize is blocked from incrementing or decrementing until that signal is removed, effectively stopping it in place.

```
if (pulseCount == MAX_PWM_SIGNAL) begin
    pulseCount <= 20'd0;

    if (pwmSize == MAX_POS) begin
        reverse <= 1'b1;
    end
    else if (pwmSize == MIN_POS) begin
        reverse <= 1'b0;
    end

    if (!servoPhotoT) begin
        case (reverse)
            0: pwmSize <= pwmSize + STEP;
            1: pwmSize <= pwmSize - STEP;
        endcase
    end
end
```

Figure III – Servo Sweeping Motion with Pause

2c. Seven-Segment Controller Module

The final part to complete the Rover is a module which displays the direction and current on the built-in display on the Basys 3 Board. This module inputs the current from the Current Sensor and switch 7, which determines the direction of the motor. A 20-bit

counter is used to drive the four displays at ~95Hz, fast enough to prevent strobing but slow enough to allow the numbers to be clearly read. The display on the Board works by first selecting which anode to output to, then decoding that digit into segments [2]. The Basys 3 Board uses a transistor, which means the desired anode and segments are driven low to be activated, while unwanted anodes and segments are driven high. For example, to display a three into the first digit, the 4-bit number which is used to drive the anodes is set to 4'b0111, and the 7-bit number that is used to drive the segments is set to 7'b0110000. The module takes the 2 MSBs of the 20-bit counter and uses them as a multiplexer to rapidly switch between the anodes, where three possible options can be displayed. The options are derived based on whichever phototransistor is active, if the red coated LEDs are active, an "E" is displayed, if blue coated are active an "F" is displayed, and if neither are active a dash is displayed. Figure IV elaborates the exact method that checks, which is just a simple if else if statement. Figure V is an example of what an "F" looks like displayed on the display in real time.

```
// determine what to output based on which phototransistor sees a light
always @(posedge clk100) begin
    if (RRPhotoT || LRPhotoT) begin
        seg_temp <= 7'b0011110; // E for enemy
    end
    else if (RBPhotoT || LBPhotoT) begin
        seg_temp <= 7'b0001110; // F for friend
    end
    else begin
        seg_temp <= 7'b0111111; // dash for nothing
    end
end
```

Figure IV – Phototransistor Decoding for Segment Display

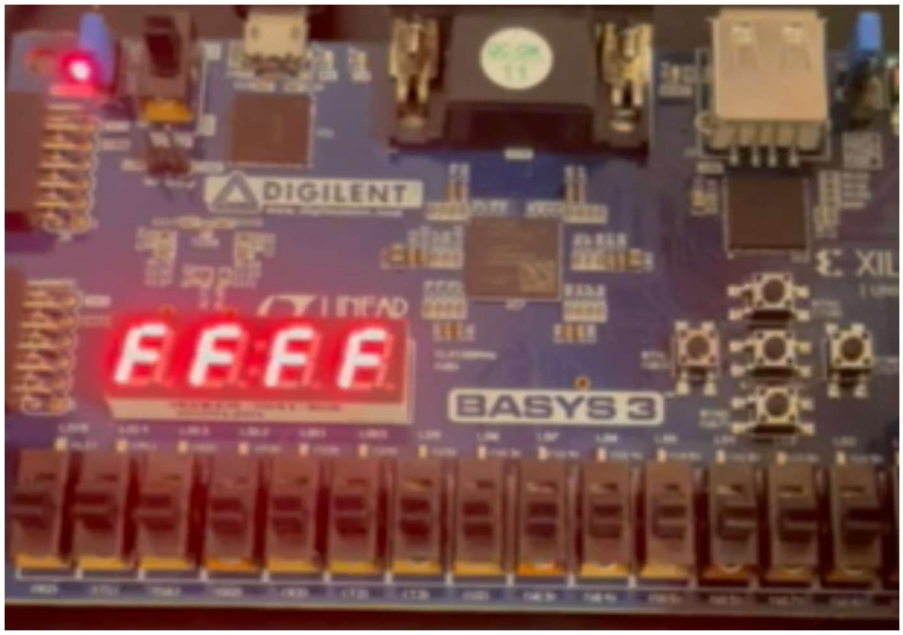

Figure V – Basys Seven-Segment Display in Action

2d. Future Software Ideas and Implementation

Obviously, the requirements of the project cannot be satisfied without much more code to run the various components. As of right now, the flowchart in Figure VI depicts the current iteration of what is believed to be the required amount of code. Completed items are the Seven-Segment Controller and the Servo Controller, which leaves the Driving Module, Firing Module, and Top Module incomplete without further hardware to utilize. The Driving Module will be very similar to the previous Motor Controller Module, in which a PWM signal running at around 762Hz and the speed is controlled by modifying the duty cycle. The Rover will be driving along the track at around 70 percent duty cycle, and when an enemy is detected, will slow down to 15 percent duty cycle until the target is passed. This module will also input the comparator signal and determine if the Rover needs to be turned off to avoid overcurrent. The Firing Module will be triggered when the Servo Controller acquires its target, which will then control an

actuator in order to fire the rubber band gun. After firing, the actuator will retract, reloading the gun for the next shot. Finally, the Top Module will interconnect all the modules and the outside signals.

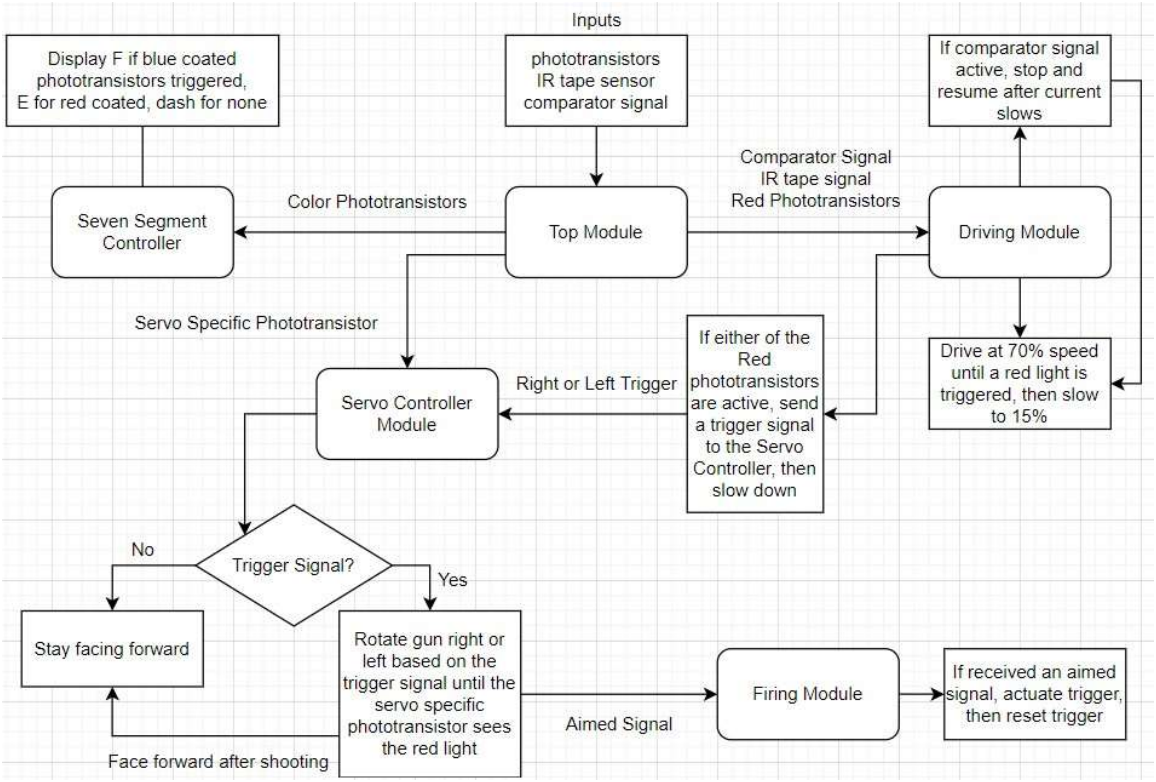

Figure VI – Code Flowchart

2e. Physical Design

The current physical design of the project is relatively incomplete and haphazardly put together, as the only completed hardware is a basic test version of the infrared sensor locations, the H-Bridge talking to the Basys, and the battery powering it all, shown in Figure VII. The main challenge was creating the H-Bridge to power the motors, as it needed to be soldered piece by piece, which is pictured in Figure VIII. Another hurdle which was overcome quickly was the actual wiring of the Pmod ports to

the input ports of the H-Bridge, as little instruction was given as to which lines were for what. Luckily, early on in the design process a wiring diagram was created for which wires were to plug in to which inputs, shown in Figure IX. When testing the phototransistor for use, a simple circuit consisting of power wires, the phototransistor, and a 1MΩ resistor with a signal wire coming from in-between the phototransistor and resistor delivers the information the phototransistor sees to the Basys, pictured in Figure X. The collector of the phototransistor receives 3.3V from the Basys, then the emitter gives a signal to the output wire when a light is triggered.

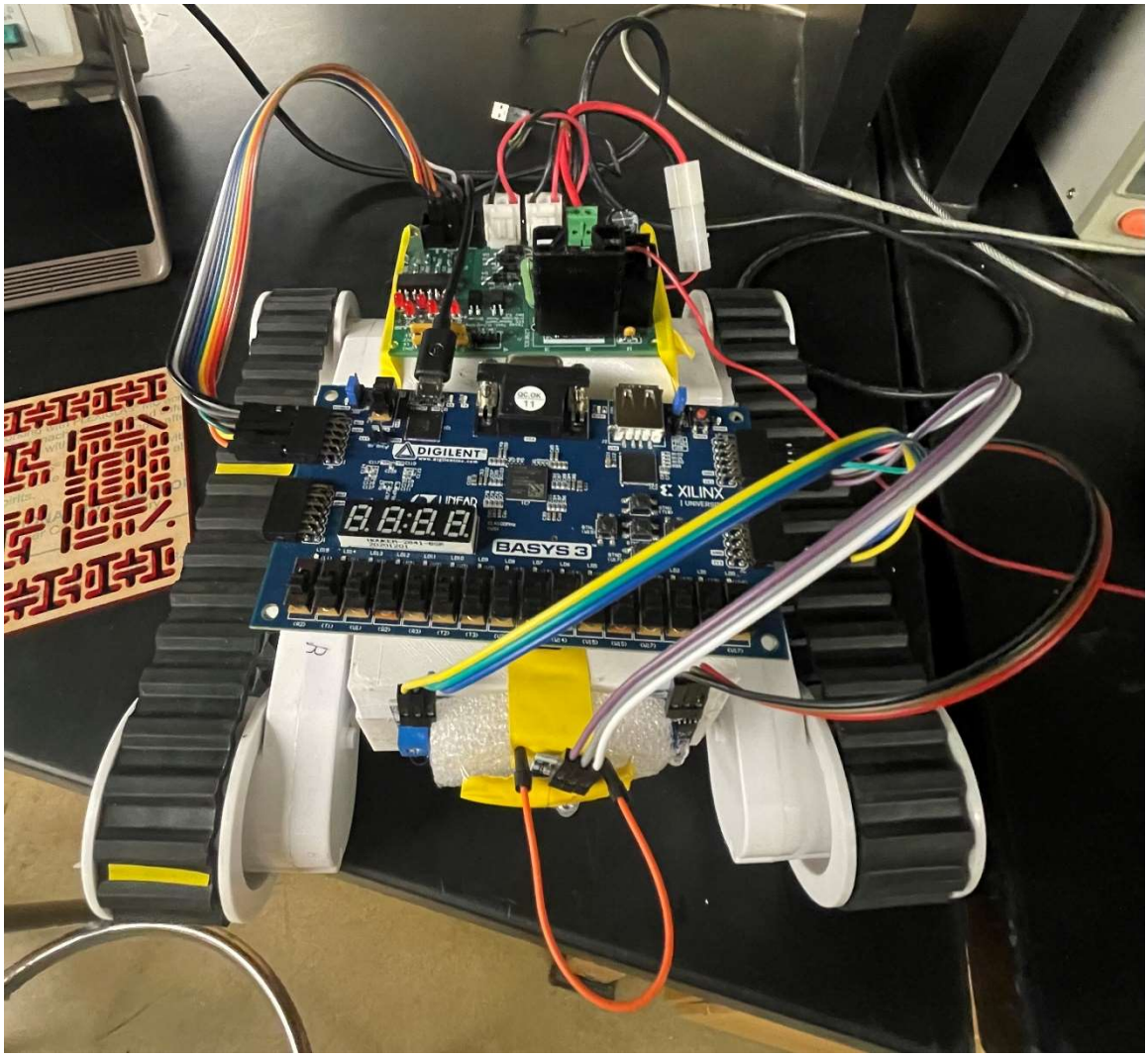

Figure VII – Test Assembly of the Rover

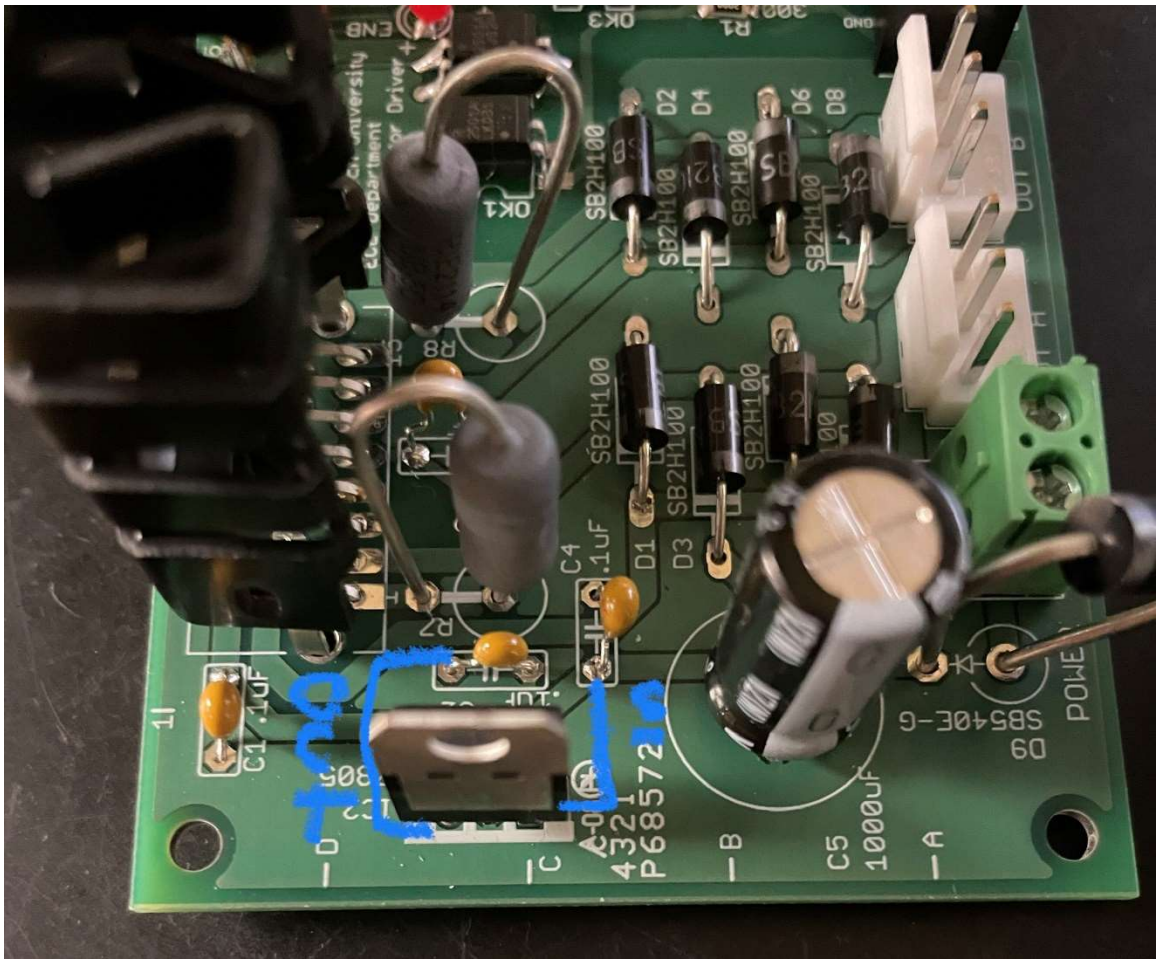

Figure VIII – Completed H-Bridge Circuit

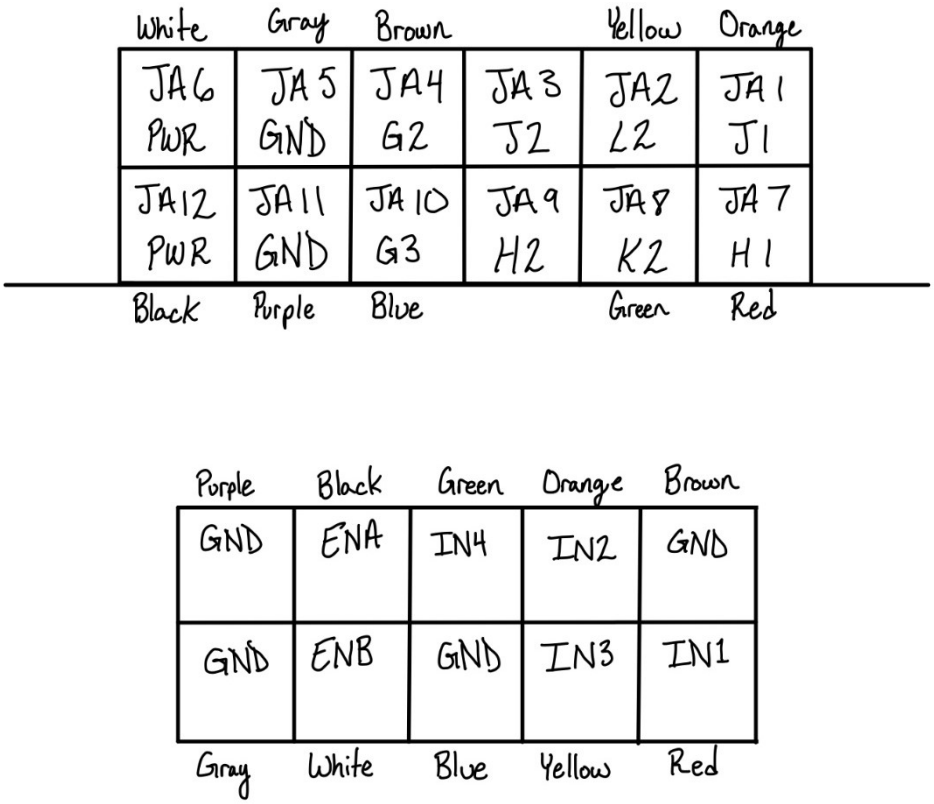

Figure IX– Physical Connections between Pmod port JA and the H-Bridge

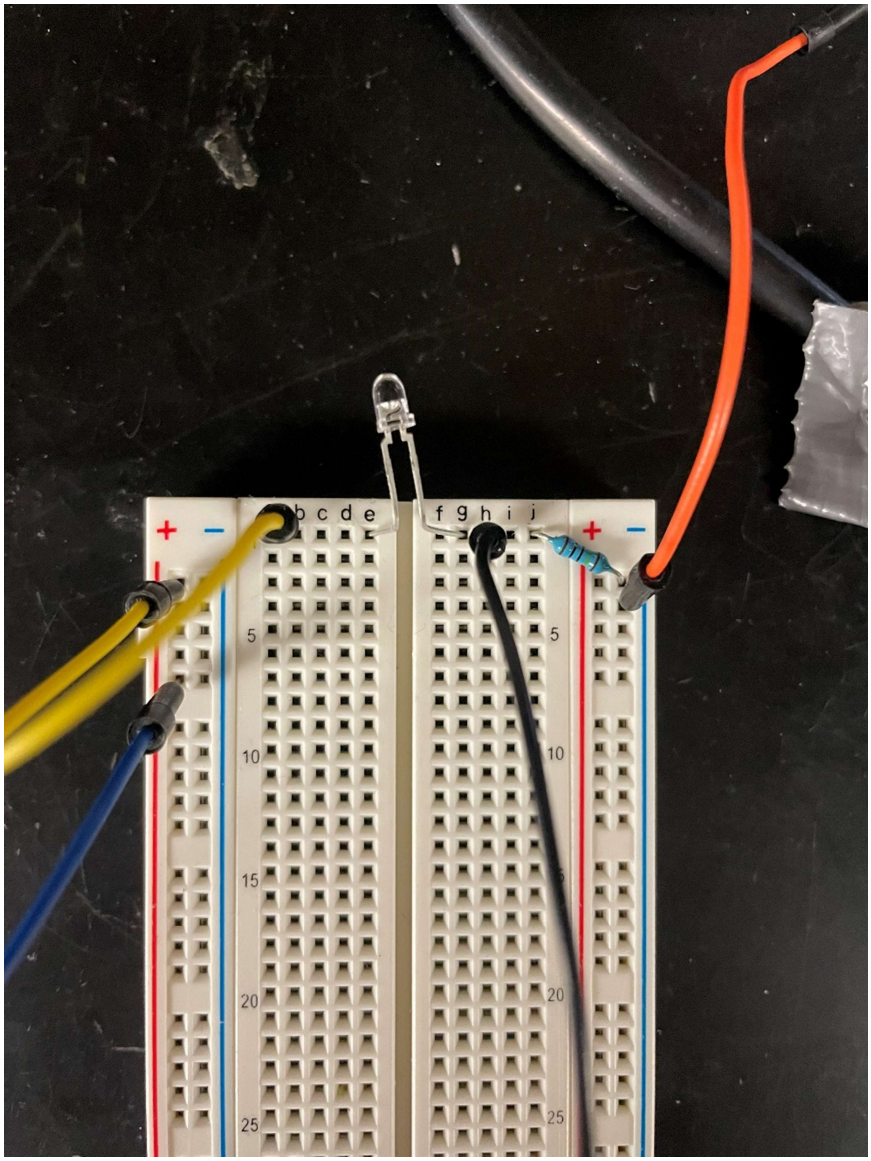

Figure X – Phototransistor Test Circuit

2f. Future Hardware Ideas and Implementation

To be able to see and fire at the targets, more work must be done to add that functionality to the Rover. The first important thing which must be finished is the infrared sensors, mainly final positioning and wiring. Figure VII which is the current version of the placement and wiring is not secure enough, meaning accidental jostles

could knock a sensor off. To mount these and the other parts of the Rover, a new 3D printed design, shown in Figure XI, will be used to hold everything.

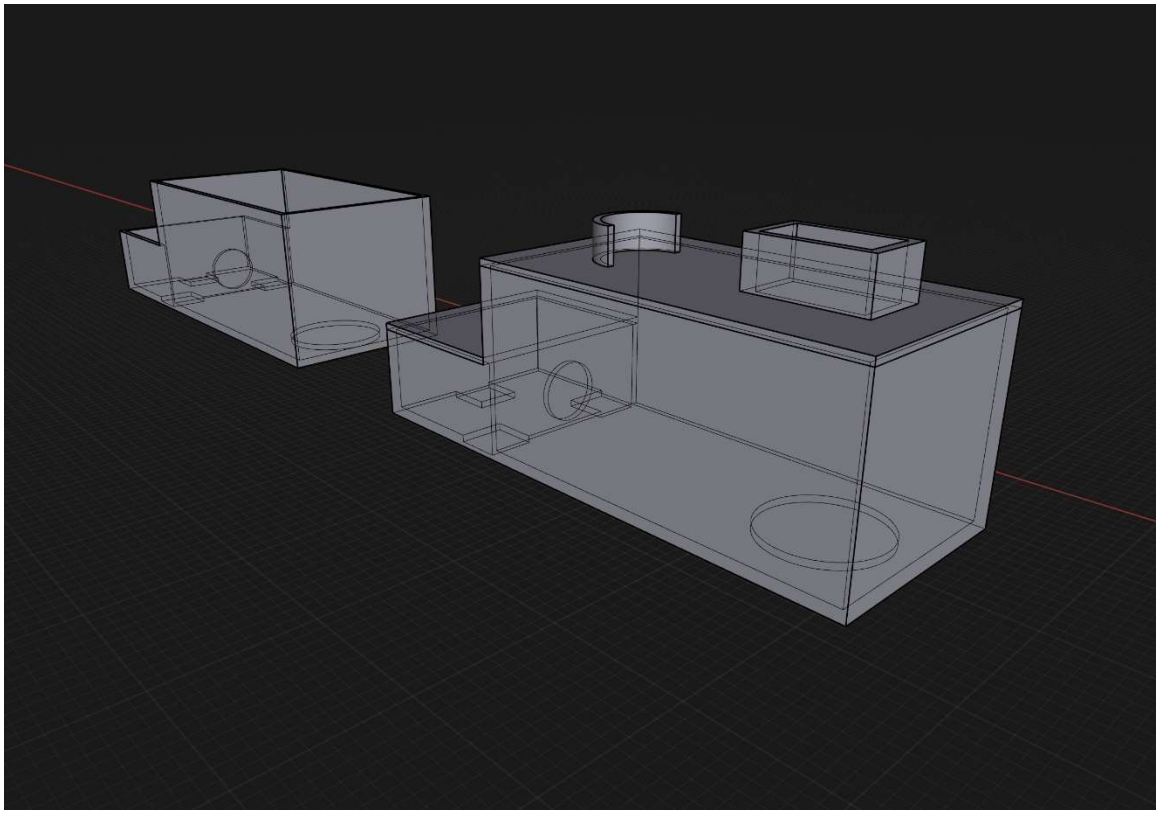

Figure XI – 3D Prototype of Shell for the Rover

Another major piece of hardware to be finished is the PCBs that control the current sensing circuit, power rails, and phototransistors. The current sensor circuit will use a comparator and reference voltage of 0.9V to dictate when the Rover is approaching a dangerous amount of current. The power rails will be a 5V regulator for the Basys and servo, a 3.3V regulator for the phototransistors and the infrared sensors, and a direct line to the battery for the actuator. Finally, the phototransistors will be wired together using a circuit to ensure the connections are secure.

3. Conclusion

While the rover remains incomplete, good work has been done to keep up steady progress and ensure success at the deadline. Around 40 percent of the code is complete and ready for full integration, although unfortunately the hardware is a little behind that. However, the 3D print has been placed in line to be printed and should be ready for testing soon, along with a prototype of the three circuits necessary for use. The current budget and Gantt Chart are noted in Appendix A and B respectively.

Appendix

A. Budget

The budget is progressing slower than expected, spending only $5,018 out of our estimated $29,171. Most of the savings has come from hours, as due to breaks and other events less hours have been spent working than expected. The material costs are also within expectations. The budget, labor hours, and material costs are displayed in Figures XIII, XIV, and XV respectively.

Project Lab 1	Running Total			Total Estimate			Start Date	2/9/2022
Direct Labor(DL):							Today	3/23/2022
Category/Individual	Rate/Hour	Hours	Total	Rate/Hour	Hours	Total		
Dirk Thieme	15	48	$ 720.00	15	160	$ 2,400.00	End Date	5/1/2022
Erik Manis	15	48	$ 720.00	15	160	$ 2,400.00		
Mohammed Ansari	15	48	$ 720.00	15	160	$ 2,400.00		
DL Subtotal:		Subtotal:	$ 2,160.00		Subtotal:	$ 7,200.00		
Labor Overhead	Rate:	100%	$ 2,160.00	Rate:	100%	$ 7,200.00		
Total Direct Labor(TDL)			$ 4,320.00			$ 14,400.00		
Contract Labor								
Lab Assistant	40	0	$ -	40	15	$ 600.00		
Classmates	15	0	$ -	15	15	$ 225.00		
Instructor	200	0	$ -	200	15	$ 3,000.00		
Total Contract Labor(TCL)			$ -			$ 3,825.00		
Direct Material Costs(DMC):			$ 354.38			$ 543.21		
(Material Cost Spreadsheet)								
Total DMC:			$ 354.38			$ 543.21		
Equipment Rental Costs:	Value	Rental Rate		Value	Rental Rate		Total Rental Days to Date	Estimate Rental Days
Oscilloscope	$ 449.99	0.20%	$ 6.30	$449.99	0.20%	$ 27.00	7	30
DMM	$ 415.00	0.20%	$ 5.81	$415.00	0.20%	$ 24.90		
Power Supply	$ 99.99	0.20%	$ 1.40	$99.00	0.20%	5.94		
Soldering Iron	$ 99.99	0.20%	$ 1.40	$99.00	0.20%	$ 5.94		
Total Rental Costs(TRC)			$ 14.91			$ 51.90		
Total TDL+TCL+DMC+TRC			$ 4,689.29			$ 18,820.11		
Business Overhead		55%	$ 2,579.11		55%	$ 10,351.06		
Total Cost		**Current**	$ 7,268.40		**Estimate**	$ 29,171.17		

Key
Current
Estimate
Totals

Figure XIII – Overall Budget

Date	Dirk	Erik	Mohammed	Lab Assistant	Classmate	Instructor		Total Hours	
2/11/2022	0	0	0	0	0	0	Friday	Dirk	48
2/12/2022	0	0	0	0	0	0	Saturday	Erik	48
2/13/2022	0	0	0	0	0	0	Sunday	Mohammed	48
2/14/2022	0	0	0	0	0	0	Monday	Lab Assistant	0
2/15/2022	6	6	6	0	0	0	Tuesday	Classmate	0
2/16/2022	3	3	3	0	0	0	Wednesday	Instructor	0
2/17/2022	0	0	0	0	0	0	Thursday		
2/18/2022	0	0	0	0	0	0	Friday		
2/19/2022	3	3	3	0	0	0	Saturday		
2/20/2022	6	6	6	0	0	0	Sunday	Key	
2/21/2022	0	0	0	0	0	0	Monday	Totals	
2/22/2022	3	3	3	0	0	0	Tuesday		
2/23/2022	3	3	3	0	0	0	Wednesday		
2/24/2022	0	0	0	0	0	0	Thursday		
2/25/2022	0	0	0	0	0	0	Friday		
2/26/2022	0	0	6	0	0	0	Saturday		
2/27/2022	0	6	0	0	0	0	Sunday		
2/28/2022	6	0	0	0	0	0	Monday		
3/1/2022	3	3	3	0	0	0	Tuesday		
3/2/2022	3	3	3	0	0	0	Wednesday		
3/3/2022	0	0	0	0	0	0	Thursday		
3/4/2022	0	0	0	0	0	0	Friday		
3/5/2022	3	3	3	0	0	0	Saturday		
3/6/2022	0	0	0	0	0	0	Sunday		
3/7/2022	0	0	0	0	0	0	Monday		
3/8/2022	6	6	6	0	0	0	Tuesday		
3/9/2022	3	3	3	0	0	0	Wednesday		
3/10/2022	0	0	0	0	0	0	Thursday		
3/11/2022	0	0	0	0	0	0	Friday		
3/21/2022	4	0	0	0	0	0	Monday		
3/22/2022	3	3	3	0	0	0	Tuesday		

Figure XIV – Labor Hours

Name	Cost	Quantity	Note/Website	Total		Total
Rover	$ 49.95	1	Pololu.com	$ 49.95		$ 358.38
H-Bridge	$ 4.67	1	Mouser.com	$ 4.67		
BASYS3	$ 149.00	1	Mouser.com	$ 149.00		
Battery Pack	$ 13.30	1		$ 13.30		
Connectors	$ 3.95	1	Digikey.com	$ 3.95		
Charger	$ 13.99	1	Amazon.com	$ 13.99		Key
Arduino starter kit	$ 38.99	1	Amazon.com	$ 38.99		Totals
3D printed parts	$ 30.00	1	Maker Space	$ 30.00		
Servo Motors (5)	$ 13.00	1	Amazon.com	$ 13.00		
Phototransistors (10)	$ 3.74	1	Mouser.com	$ 3.74		
Infrared Sensors (10)	$ 8.79	1	Amazon.com	$ 8.79		
Actuator	$ 25.00	1	Mouser.com	$ 25.00		
Light Filtration	$ 4.00	1	Home Depot	$ 4.00		

Figure XV – Material Costs

B. Gantt Chart

The Gantt Chart is also progressing as expected, although with a little delay from the hardware. One thing which is very behind is the CPR certificate, which is required to by the end of the semester. While this has not been done yet, research has been done into

accredited classes to make sure money is not wasted. The Gantt Chart, which is basically a timed list of goals, is located in Figure XVI.

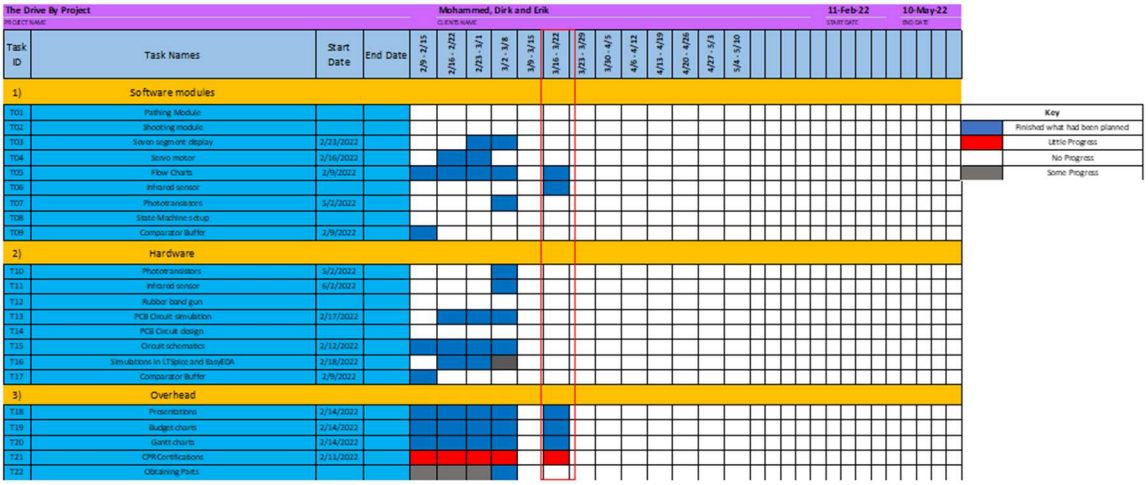

Figure XVI – Gantt Chart

Acknowledgements

I would like to thank Dr. Hemmert for his slides and lectures about what to do, Jake Keidasch for help with choosing a targeting system, Thompson for help with beginning the driving module, and my group for helping get everything done.

References

1. “Pololu - Dagu Rover 5 tracked chassis with encoders,” Pololu Robotics Electronics. [Online]. Available: https://www.pololu.com/product/1551. [Accessed: 10-Feb-2022].
2. “Basys 3 Reference Manual,” Basys 3 Reference Manual - Digilent Reference. [Online]. Available: https://digilent.com/reference/programmable-logic/basys-3/reference-manual. [Accessed: 10-Feb-2022].
3. “SystemVerilog Tutorial,” *ChipVerify*. [Online]. Available: https://www.chipverify.com/systemverilog/systemverilog-tutorial. [Accessed: 23-Mar-2022].
4. “Infrared (IR) sensor HW201, object detection: Malaysia industrial and robotic solutions,” *INRO Electronics*. [Online]. Available: https://www.inro-electronics.com/infrared-object-detection-sensor. [Accessed: 23-Mar-2022].
5. “Hitec HS-422 - deluxe standard servo,” *Hitec HS-422 Servo Specifications and Reviews*. [Online]. Available: https://servodatabase.com/servo/hitec/hs-422. [Accessed: 23-Mar-2022].
6. OSRAM, “SFH 309 Silicon NPN Phototransistor,” SFH 309-5/6 datasheet, Jun. 20, 2018.