Final Report for Project Lab at Texas Tech University

Mason Marnell

Andrew Stelluti (group member)

Justin Sims (group member)

R# 11617302

Texas Tech University

May 4th, 2021

Abstract

This paper describes the workings, mechanics, and technical information regarding a rover system built to navigate a playing field. Said rover must be able to detect and avoid walls, detect randomly placed colored cards on the floor, and sense strobe input from an external light source. Major components used were a Basys 3 board, a dual H-bridge circuit, and a Rover 5 Chassis.

Table of Contents

List of Figures

1. Introduction

The playing field that the rover has the ability to move across is approximately 10 feet by 10 feet but could theoretically be as large as desired. The 3x5" index cards scattered across the floor are either red, blue, or green. The rover must detect what color said cards are and use two different visual indicators for the detection. The rover must also be able to detect flashes of visible light. These flashes should be morse letters that correspond to Go, Start, Stop, and Pause.

The rover discussed in this report uses a relatively simple (yet effective) "bouncing strategy". Instead of hard-programming precise turns and speeds, the rover will simply turn when it senses and gets close enough to a wall. This approach will eventually find all the index cards; it is slower, but there is significantly less room for error.

2. Block Diagram

The block diagram is a comprehensive layout of essentially every physical component of the rover system. Every component that requires power gets it from the "Voltage Regulator & Overcurrent Protection Circuit," which is in turn powered by a 9.6 V battery. PMOD ports are used to interface the Basys board with all data-centric components, and the strobe beacon provides external input to the strobe sensor.

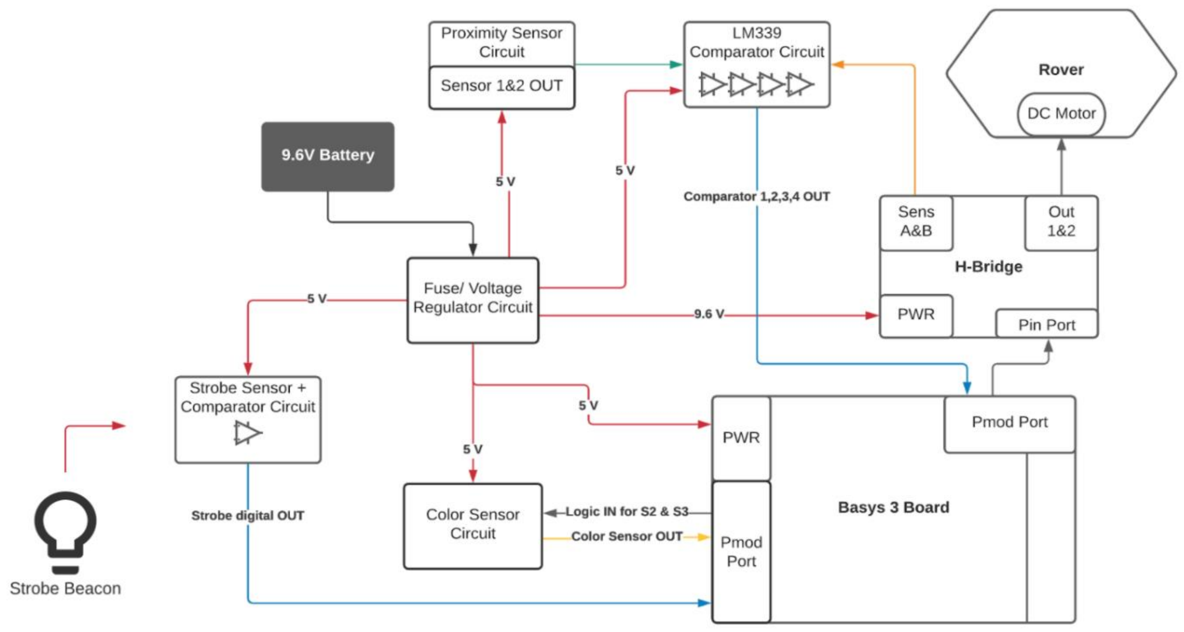


Figure 1: Block Diagram [6]

3. H-Bridge

The “H-Bridge” refers to a circuit layout where a load (in this case, our DC motor) is able to have manually-toggleable bidirectional current flow [1]. The namesake of the circuit is apparent by observing Figure 2. The IN1 - IN4 pins on the soldered version correspond to the switches in the diagram, with S1 being IN1, S2 being IN3, S3 being IN2, and S4 being IN4.

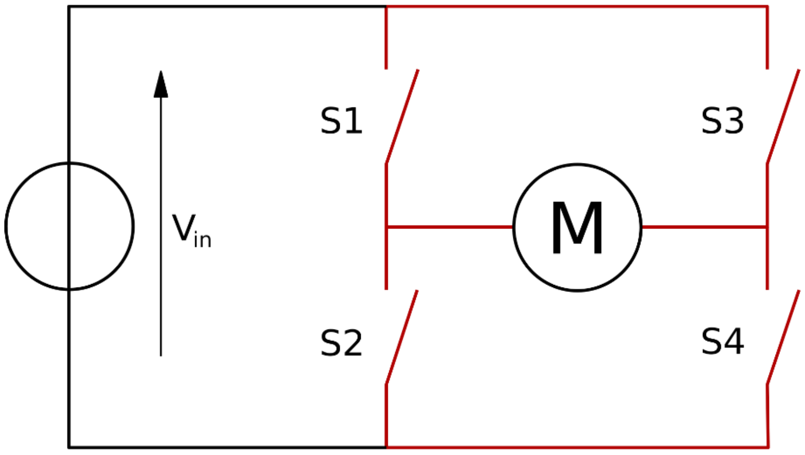


Figure 2: H-bridge Circuit Schematic [2]

A pre-packaged H-bridge circuit kit was soldered together and used for the foundation of the system, seen in Figure 3.

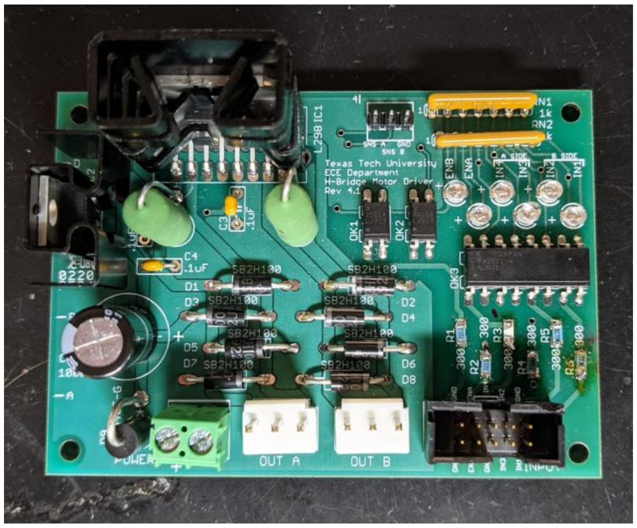


Figure 3: Completed H-Bridge PCB

The method used to test the H-Bridge was to let an ECE Stockroom employee check to make sure the connections were the way they should be using an in-house physical testbench. This could also be done using a handheld multimeter, though in a less effective manner.

4. Basys 3 Board

The Basys 3 Board, produced by Digilent, serves as a motherboard of sorts for the project. As seen in Figure 4, there are 16 user-configurable switches on the board, four 7-segment displays, and five pushbuttons.

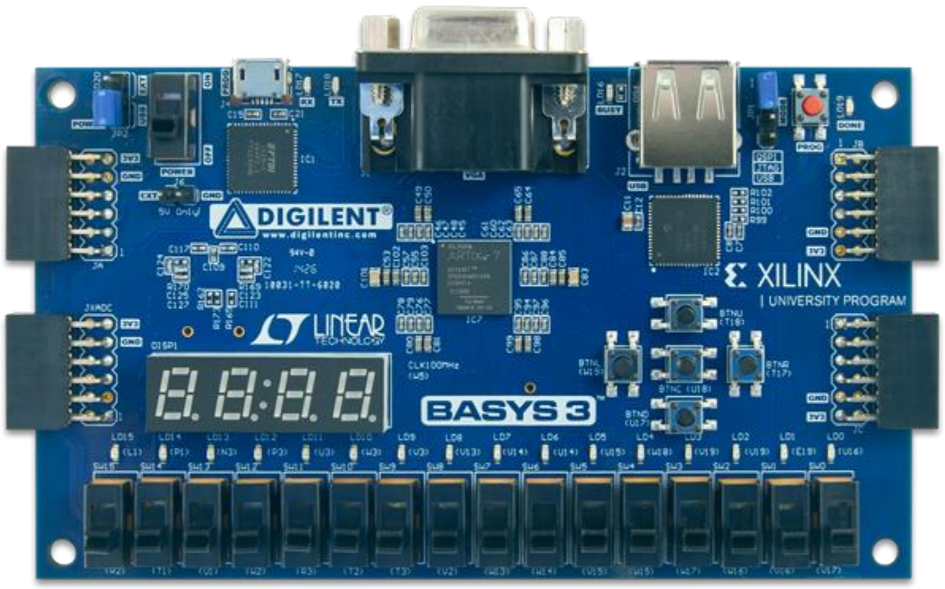


Figure 4: Basys 3 Board Overview [3]

Also seen in Figure 4, there are 4 separate PMOD headers. On each PMOD header, eight pins are reserved for digital logic operations. Twelve of these pins were used: four pins for IN1-IN4, two pins for ENA and ENB (PWM), two for distance input, three for color sensing, and one for strobe input.

5. Comparator Circuit

Comparator circuits are used in many different circuits on the rover so that once a threshold is reached by the input voltage, the output voltage goes from a low to a high. For one of the comparator circuits, a 1 kΩ resistor and a 434Ω resistor were used to make a voltage divider that sets the reference voltage to 1 volt. This can be seen in Figure 5. The LM339 quad differential comparator from TI was used for every comparator circuit.

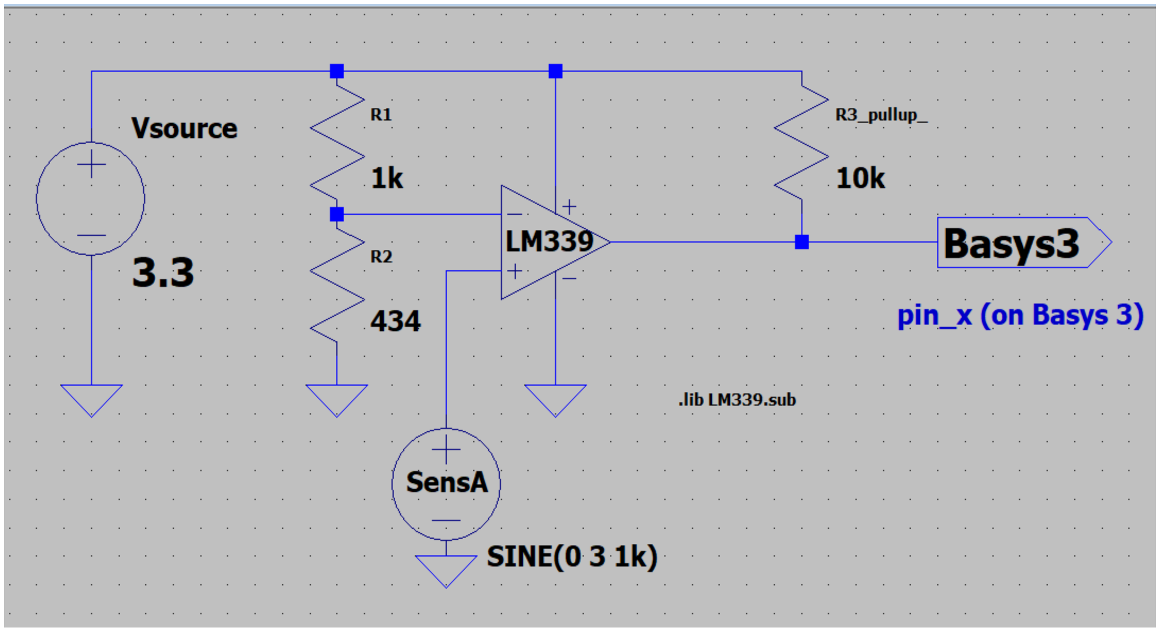


Figure 5: Comparator Circuit Schematic [6]

A simulated oscilloscope image is shown in Figure 6 and a physically tested oscilloscope reading is shown in Figure 7, both showing that the output voltage is set to high when the threshold value of the input voltage is met.

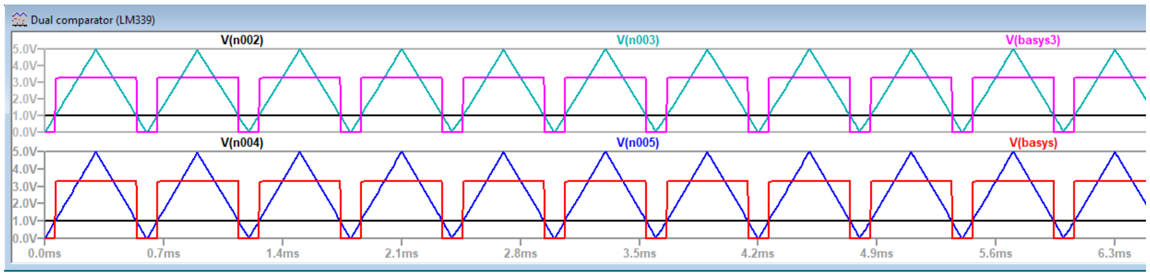


Figure 6: Triangle Wave = Input Voltage, Square Wave = Output Voltage [6]

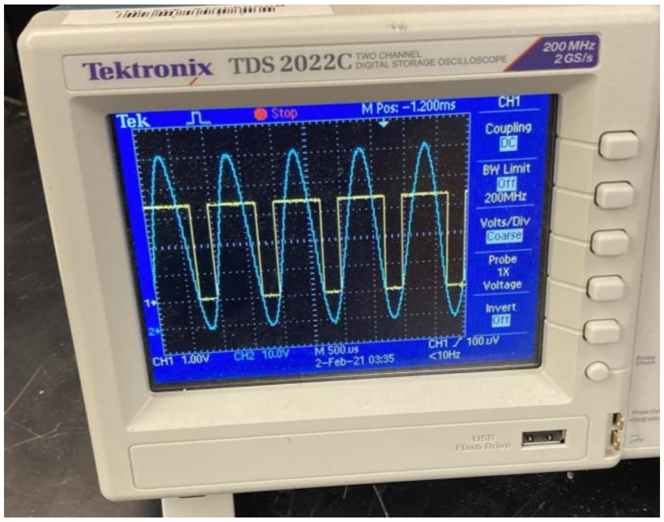


Figure 7: Early Comparator Test with Oscilloscope

6. Proximity Sensor & Circuit

For this project, two Sharp GP2Y0A60SZ0F analog distance sensors were used. The sensor uses infrared light to detect how far away any solid object (other than glass) is. The sensor is analog, though it was decided that it should be used in tandem with a comparator circuit so as to give a high when triggered and a low when not. This piece is superior for short ranges, since it was not required to create an I2C interface like it would have been using something like an ultrasonic distance sensor.

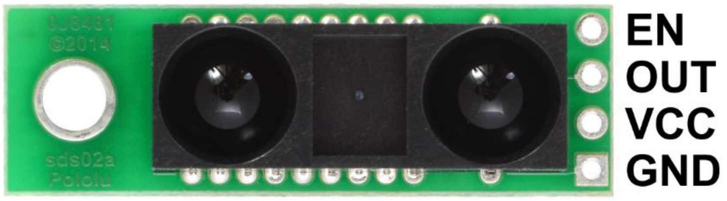


Figure 8: Analog Distance Sensor [4]

Figure 9 demonstrates functionality of the comparator part of the circuit (Blue waveform) and the corresponding analog distance sensor input (orange waveform).

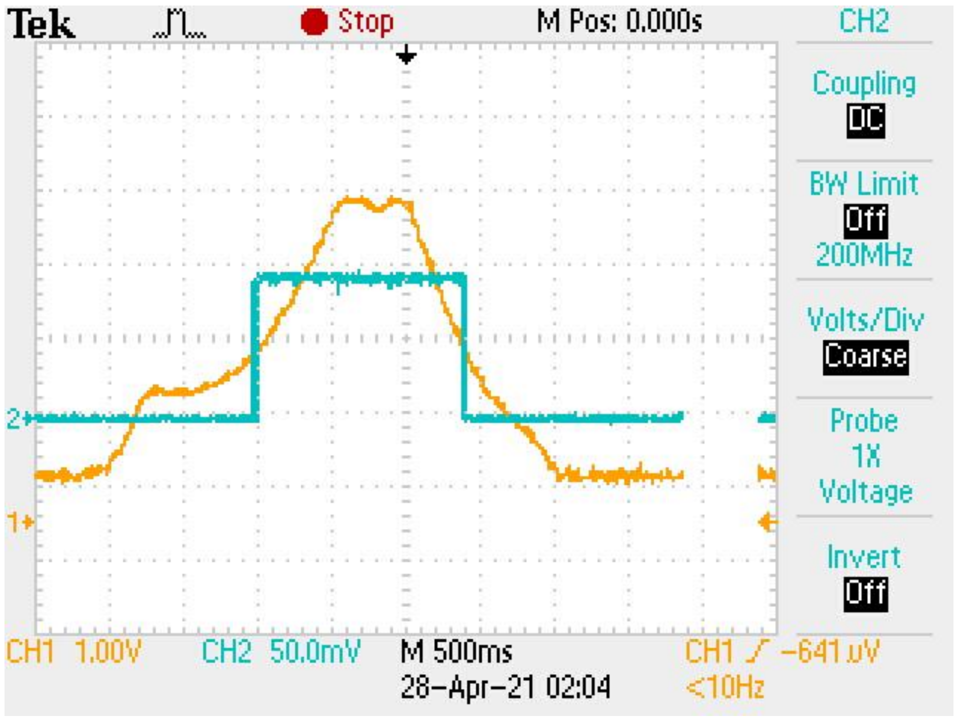


Figure 9: Distance Sensor with Comparator Oscilloscope

7. Color Sensor & Circuit

To be able to sense color, the TCS3200 Color Sensor was chosen for the project. The sensor has four color filters that can be selected: red, blue, green, and clear. These filters can be selected by sending different combinations of 1 and 0 to the TCS3200's S2 and S3 pins from the Basys. The ability to select filters is essentially required to detect color. If the color trying to be detected is, say, blue, then the blue filter would receive the highest amount of blue light (disregarding the filterless sensor). Like the distance sensor, this piece does not require any peripheral interfaces like I2C or SPI. The in-depth software interfacing will be discussed in the coding section.

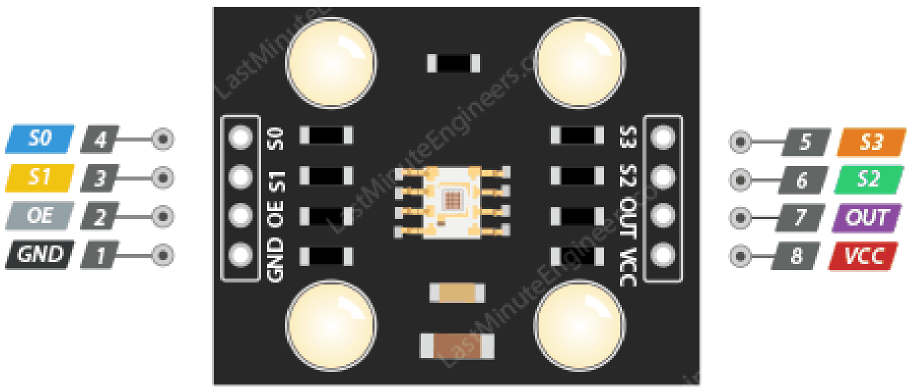


Figure 10: TSC3200 Distance Sensor Pinout [5]

8. Phototransistor Circuit

To be able to give the rover movement commands, an analog phototransistor in tandem with a comparator circuit was chosen to detect the light given off by the strobe beacon, which is covered in the next section.

The OP593A phototransistor was chosen over other solutions (such as a photoresistor) due to this phototransistor's superior light sensitivity and response time. A tin foil cone was added to increase strobe distance, and a potentiometer was added to the comparator circuit to be able to set the reference voltage on-the-fly to account for ambient light in different conditions.

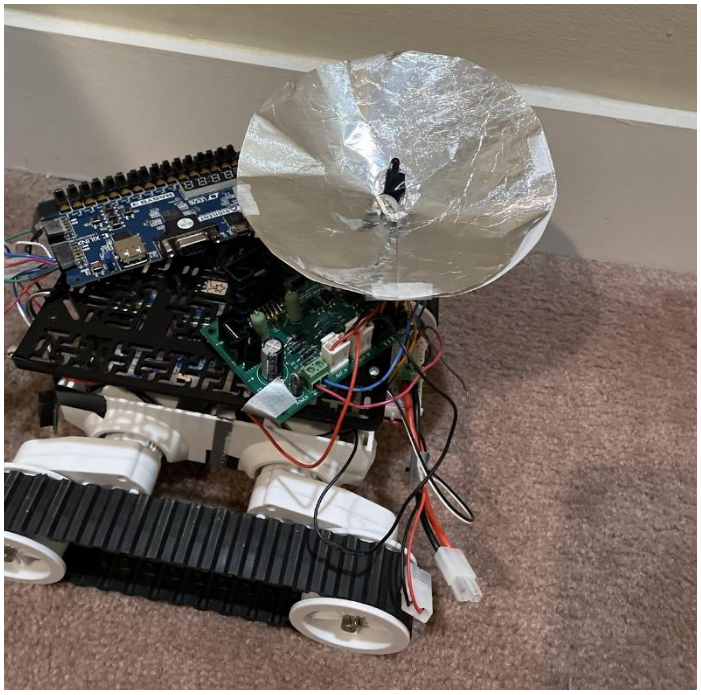


Figure 11: Phototransistor Cone

9. Strobe Beacon and Circuit

The strobe beacon was made using an Arduino, a block of wood, a battery pack, a magnifying glass, a flashlight LED, and an IRFZ44N MOSFET. The MOSFET was used because the LED needed more current than the Arduino could provide to be driven close to its full output power. The circuit is simulated in figure 12.

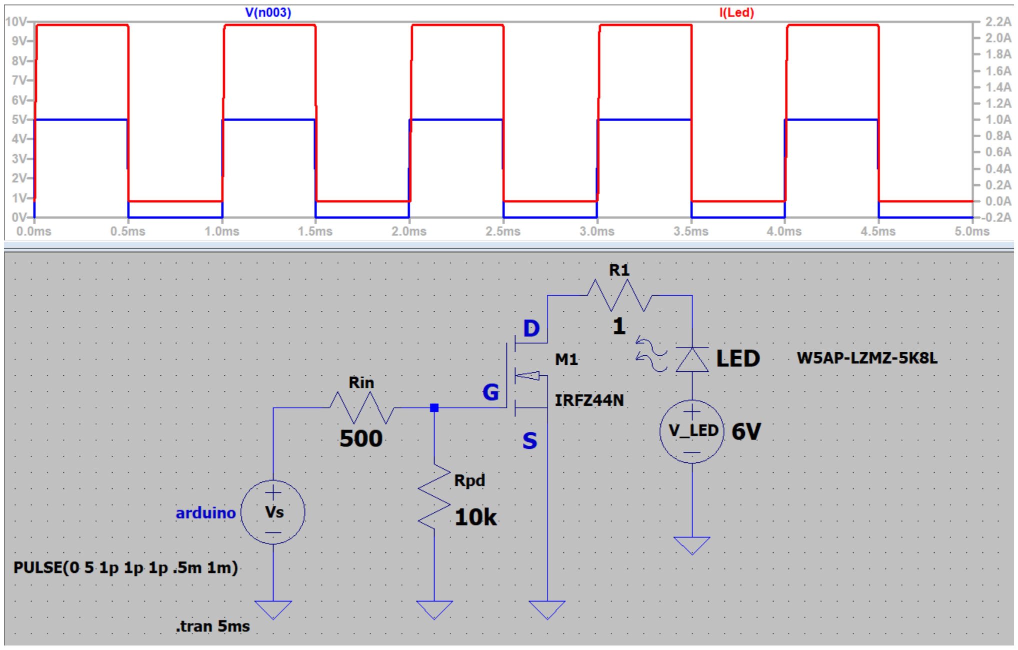


Figure 12: MOSFET/Beacon Circuit Simulation [6]

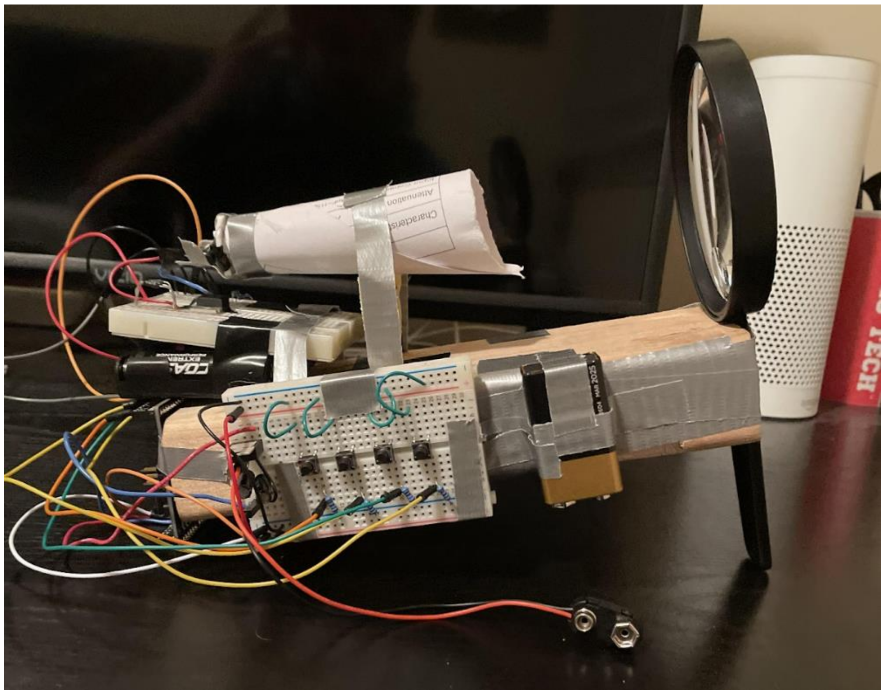


Figure 13: Finished Strobe Beacon

Figure 14 shows two examples of strobe inputs received by the phototransistor.

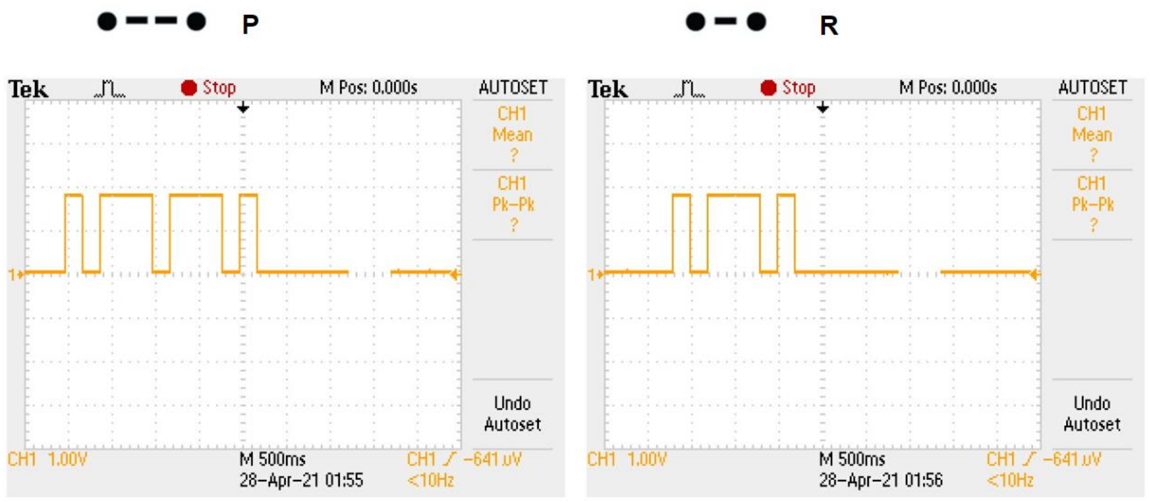


Figure 14 : Strobe Beacon and Phototransistor Oscilloscope Readings[6]

10. Voltage Regulator and Overcurrent Protection Circuit

It is required in the project to have some sort of physical over-current protection. A 5-volt regulator was also indirectly needed since it was required to use a single battery for all power needs. A custom green board was made for this combined circuit, which uses a relatively simple design. A 0.8 Amp fast-blow fuse (chosen based on our testing of the rover's max current draw) was put into a circuit containing both a .1uF and a .33uF capacitor (to give a steady voltage to the regulator) and an LM7805 5V regulator chip. This regulator chip can receive an input voltage from 7 to 40 and regulate it down to 5. This 5 volts is used by the Basys, the strobe circuit, and the color circuit.

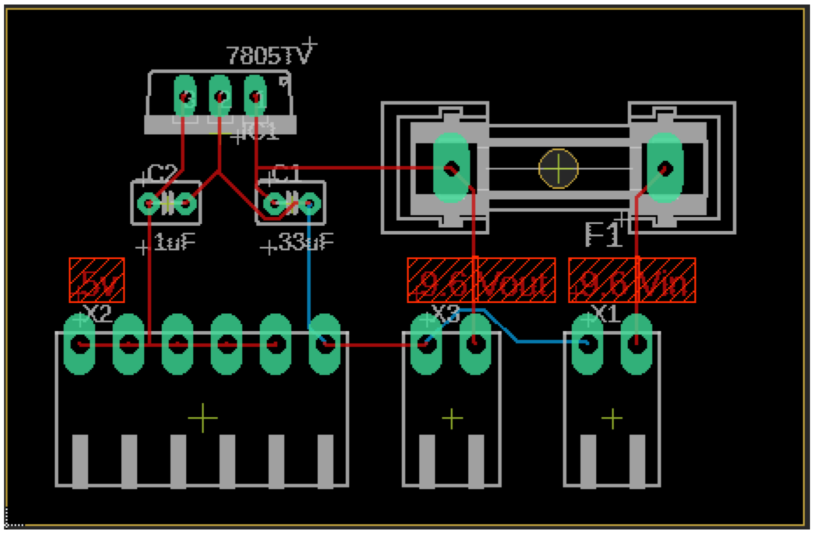


Figure 15: Regulator / Protection Circuit Greenboard Schematic [7]

11. Software/Hardware Interfacing

Moving forward from the hardware side of the project, the software side is equally robust and important. Verilog is the hardware description language (not to be confused with a programming language) used for programming the Basys board. There are 3 main components to the Verilog setup used in this project: the top module, subsequently linked modules, and the constraints file. The top module (in our case ambiguously named "VerilogFile.v") is used to link every other module within the project folder together by a process known as instantiation. In the final version of the project, the top module has connections to the StateMain module, which contains all the other independently tested modules, such as the DISP, PWM, and StrobeSense modules. The constraints file allows an input or output from the top module to be bound to a physical PMOD port or other hardware "slot" on the Basys.

12. State Machine Design

As stated before, Verilog is not a traditional programming language. The Basys 3 is an FPGA, which stands for Field Programmable Gate Array. Due to this seemingly small caveat, nontraditional designs, like a state machine, must be used. The code used for the project is based on a mealy state machine design, in which the next state is determined by the current state as well as external input.

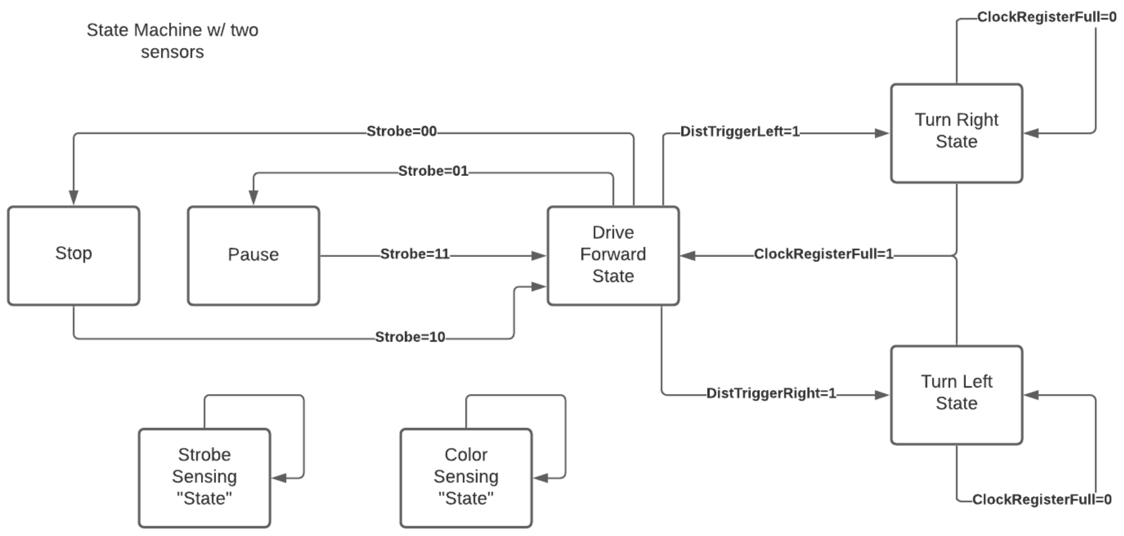
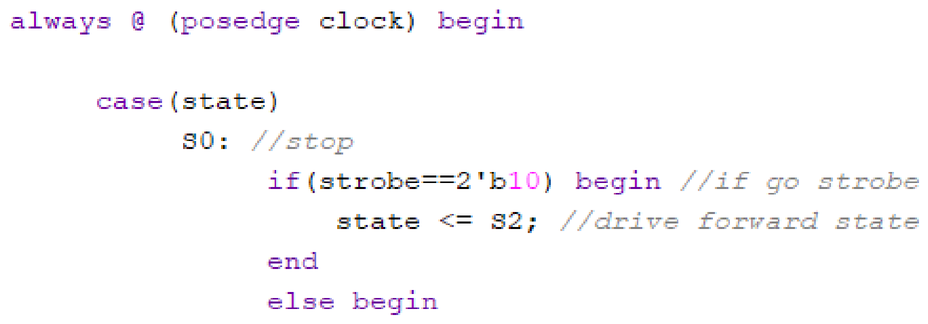


Figure 16: State Machine Diagram

As seen in the flow chart, the next state is almost always dependent on an input. The strobe sensing and color sensing functionality are effectively running separately from the main state machine, with only input from StrobeSense directly affecting the main state machine.

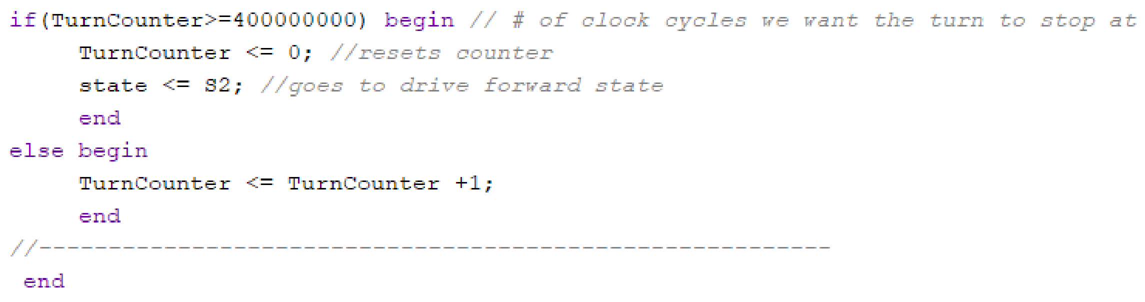


```
always @ (posedge clock) begin

    case(state)
        S0: //stop
            if(strobe==2'b10) begin //if go strobe
                state <= S2; //drive forward state
            end
            else begin
```

Figure 17: State Machine Code Example

As is possibly apparent, what is considered a state machine for this code is essentially just a case statement that gets triggered every clock cycle. Nonetheless, it accomplishes what it needs to and does so concisely. Within each case statement are the triggers for moving to a different state (given the right conditions).



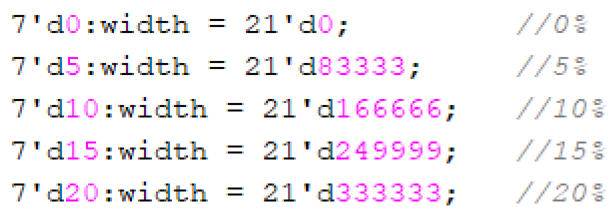
```
if(TurnCounter>=400000000) begin // # of clock cycles we want the turn to stop at
     TurnCounter <= 0; //resets counter
     state <= S2; //goes to drive forward state
     end
else begin
     TurnCounter <= TurnCounter +1;
     end
//----------------------------------------------------------
 end
```

Figure 18: Turn Counter Code

For the right and left turning states specifically, there is a turn counter that allows the system to stay inside of the turning state for a certain amount of clock cycles. After the specified value is reached by the counter, the system returns to the drive state.

13. PWM Module

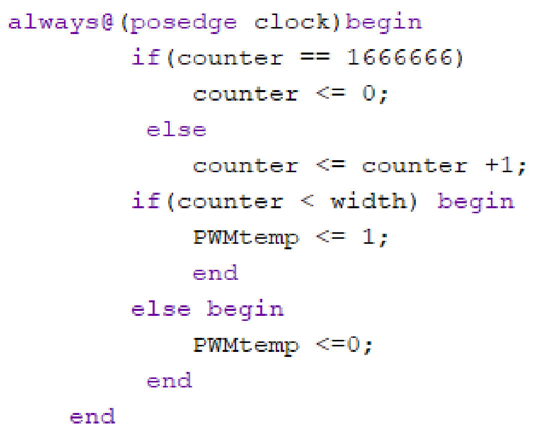
PWM, or Pulse Width Modulation, is a concept used within the project to be able to control the speed of the motors. By cycling the full energy of a signal on and off in a calculated way, the average magnitude of that signal can be granularly controlled through software. This eliminates the need for hardware based DC scaling capabilities.



```
7'd0:width = 21'd0;          //0%
7'd5:width = 21'd83333;      //5%
7'd10:width = 21'd166666;    //10%
7'd15:width = 21'd249999;    //15%
7'd20:width = 21'd333333;    //20%
```

Figure 19: PWM Width Code

The project uses a case statement that compares the decimal value given to the PWM module. The width of the pulse is then set to a predetermined multiple of the counter shown below.



```
always@(posedge clock)begin
        if(counter == 1666666)
            counter <= 0;
         else
            counter <= counter +1;
        if(counter < width) begin
            PWMtemp <= 1;
            end
        else begin
            PWMtemp <=0;
         end
    end
```

Figure 20: PWM Comparison Code

Above, “width” is compared to the counter variable as shown which creates the duty cycle for the output signal.

As seen in Figure 21, the value given to the PWM module is 25. The resulting output is a square wave that is high 25% of the time. The output signal is bound to two pins on the Basys that are connected to ENA and ENB on the H-Bridge. It was found to be easier to "PWM" the enables rather than the 4 inputs due to the enables having to invert to change motor direction.

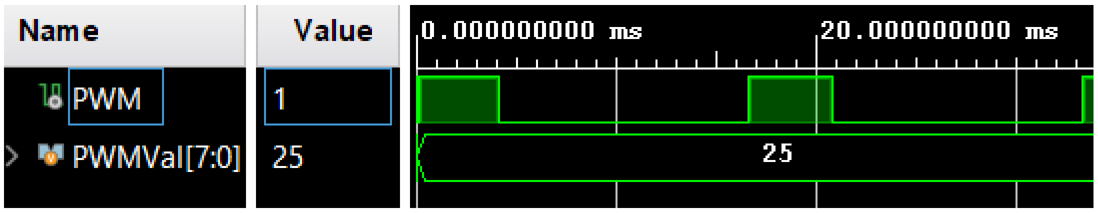


Figure 21: PWM Testbench

Figure 22 shows an oscilloscope reading of the output of the H-bridge's current/voltage sense pin. A 50% PWM duty cycle can be observed.

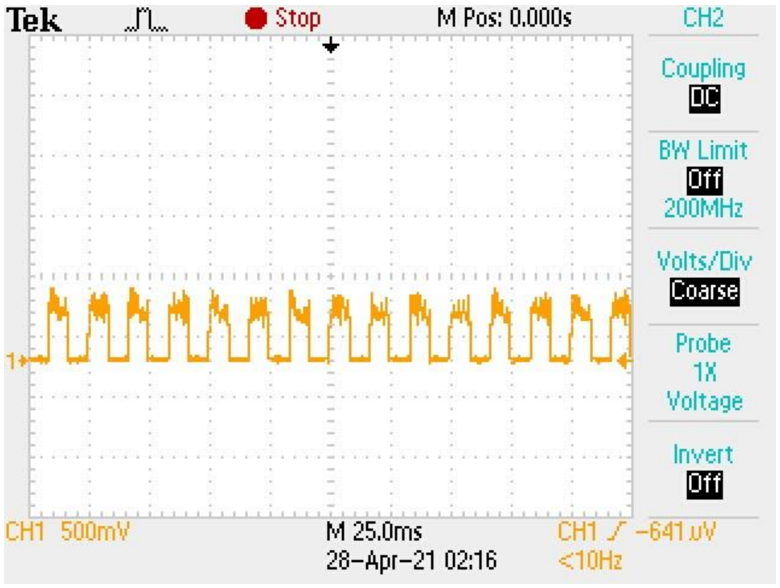


Figure 22: PWM Oscilloscope Shot

14. ColorSense Module

For the color sensing functionality, a frequency counter, along with some case statements, were used. The way the TCS3200 conveys intensity of colored light is through frequency. A higher frequency means more of that color is being detected. As seen in Figure 23, the frequency for each color filter is being read and stored. After every color is stored, a done signal is sent which causes the system to compare frequency values and set color_out to the corresponding value.

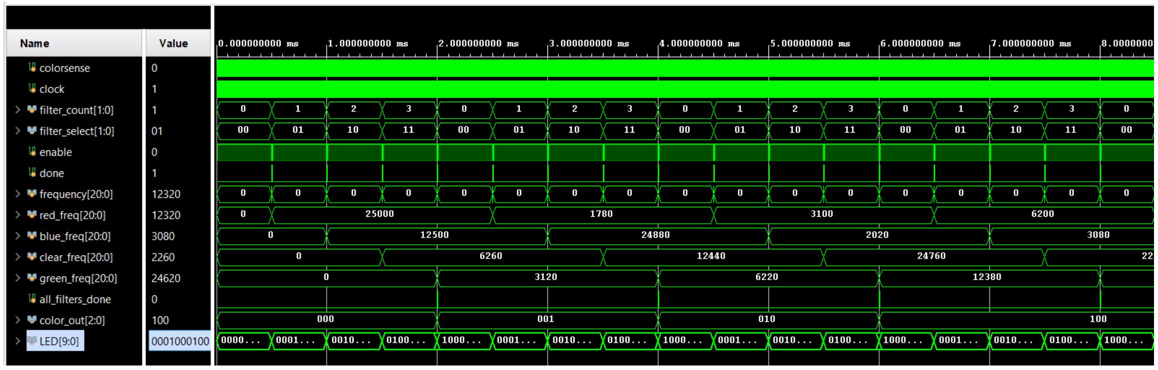
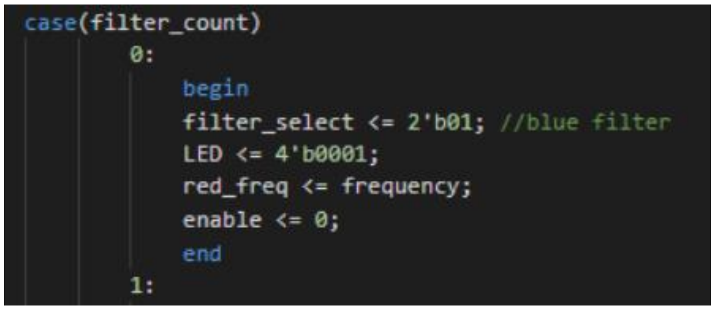


Figure 23: ColorSense Testbench [6]

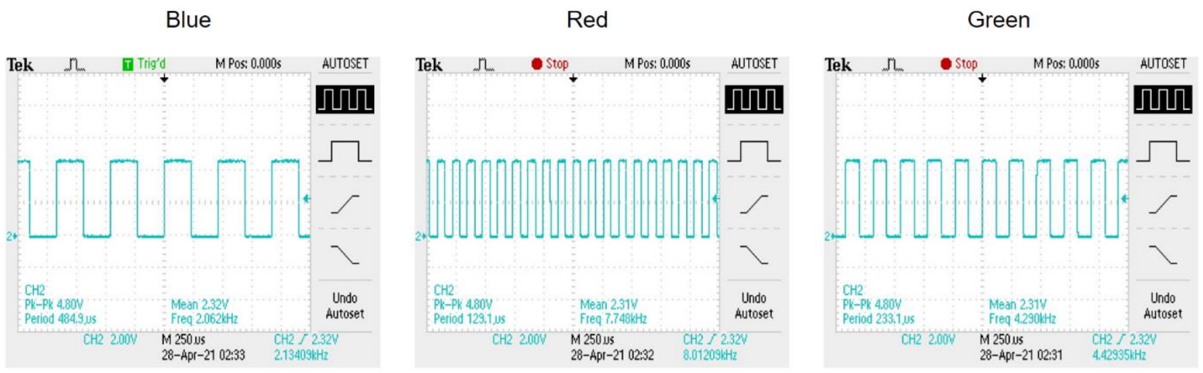
Shown in Figure 24 is a section of code required to switch color filters and store the color frequency value in a respective color variable.



```
case(filter_count)
        0:
            begin
            filter_select <= 2'b01; //blue filter
            LED <= 4'b0001;
            red_freq <= frequency;
            enable <= 0;
            end
        1:
```

Figure 24: ColorSense Code Snippet [6]

Figure 25 shows Oscilloscope readings of the frequencies received when using the red color filter and showing the color sensor the three colors of cards.


Blue
Red
Green
Tek
Trig'd
Stop
M Pos: 0.000s
AUTOSET
Undo Autoset
CH2 Pk-Pk 4.80V Period 484.9µs
Mean 2.32V Freq 2.062kHz
CH2 2.00V M 250µs 28-Apr-21 02:33 CH2 2.32V 2.13409kHz
CH2 Pk-Pk 4.80V Period 129.1µs
Mean 2.31V Freq 7.748kHz
CH2 2.00V M 250µs 28-Apr-21 02:32 CH2 2.32V 8.01203kHz
CH2 Pk-Pk 4.80V Period 233.1µs
Mean 2.31V Freq 4.290kHz
CH2 2.00V M 250µs 28-Apr-21 02:31 CH2 2.32V 4.42935kHz
Figure 25: Color Sensor Oscilloscope Readings [6]

15. StrobeSense Module

The strobe sense module is subjectively the most intuitive module of the project. The basic idea is that the module waits a predetermined amount of clock cycles and then takes a reading from the StrobeIn port on the Basys. It stores that reading in the rightmost (or least significant) bit of an 11 bit register, and then checks if the configuration of bits matches a predetermined value. If there is a match, then it sets the strobe variable to one of the 4 movement options. Whether or not there is a match, the counter continues to count up and loop.

```
if(StrobeCounter > 9999999)
    begin
        StrobeCounter <= 0;
        flip = !flip;
    end
```
Figure 26: StrobeCounter Code

```
always@ (posedge flip) begin

        morse = morse << 1;
        morse[0] = StrobeIn;
```

Figure 27: Morse Reading Code

Figure 28 shows the testbench created to test StrobeSense. StrobeIn is stimulated with the correct number of pulses for a given strobe reading, and StrobeOut is subsequently set by the module.

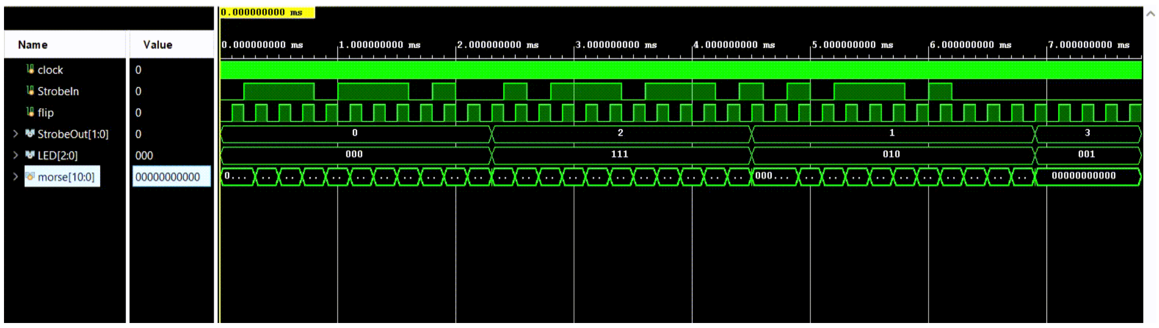


Figure 28: StrobeSense Testbench [6]

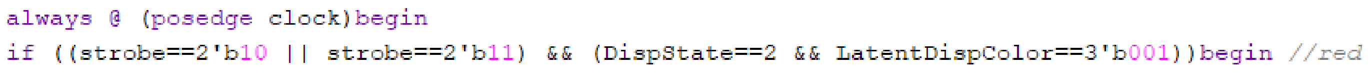
16. DISP Module

Though a small part of the project, the display had to be used to indicate the color being detected by the rover. As partially shown in Figure 29, a state machine was created so that the display would stay on for a set amount of time once a reading is received.

```
always @ (posedge clock)begin //delay counter
    case(DispState)
        0:
                if(color_out==3'b001||color_out==3'b010||color_out==3'b100) begin
```

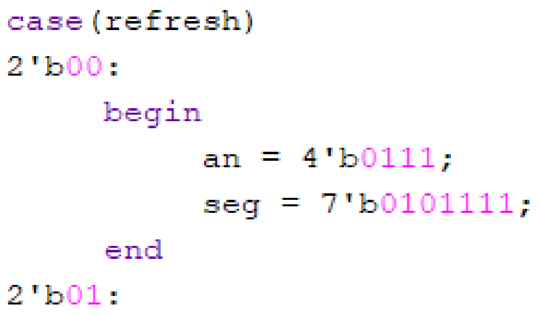
Figure 29: Display State Machine Code

An if statement was used for each color to set the 7-segment display to a certain configuration, shown both in Figure 30 and 31. The Basys 3 display is active low, so a 0 is used to denote an on state and vice versa.



```
always @ (posedge clock)begin
if ((strobe==2'b10 || strobe==2'b11) && (DispState==2 && LatentDispColor==3'b001))begin //red
```

Figure 30: Display If Statement Example



```
case(refresh)
2'b00:
     begin
          an = 4'b0111;
          seg = 7'b0101111;
     end
2'b01:
```

Figure 31: Display Value Setting Code

17. Miscellaneous Programming Concepts

Apart from the code concepts already showcased, there are two others that are worth mentioning. Firstly, due to the binary nature of the comparator used for the distance sensing, effectively no code is required to process those inputs. The raw input from the PMOD port is simply treated as any other input. Secondly, there used to be functionality for sensing whether the current draw was over 1 Amp of current. This was not used as there is now a hardware solution for detecting when too much current is being drawn.

18. Conclusion

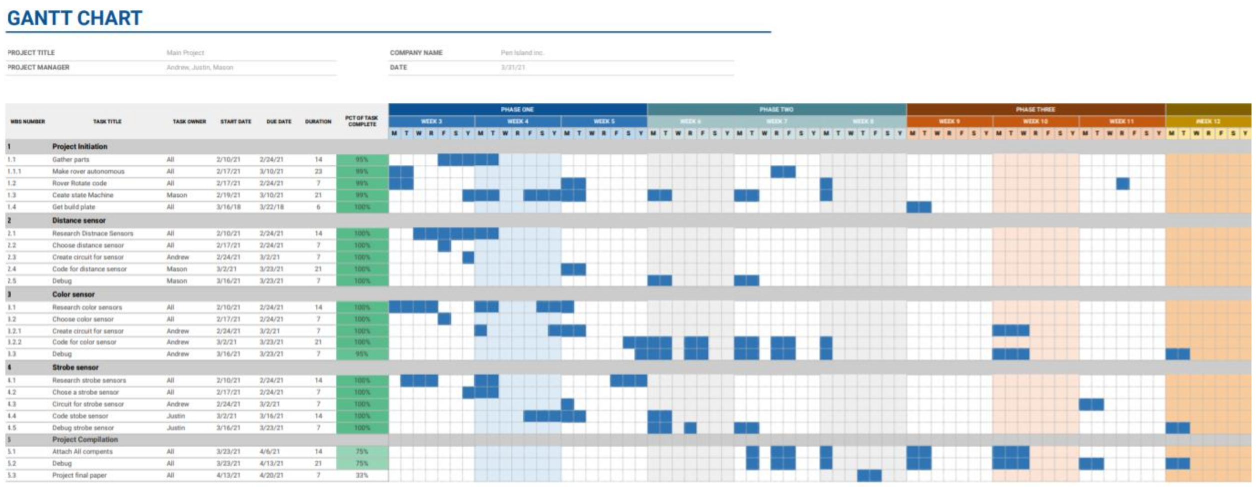
This project rover is able to meet the demands of the requirements set by the project description. It is fully functional and autonomous, and carries out its intended purpose seemingly perfectly. Every subsystem developed independently has been integrated into a wholly functional system that drives, signals, senses, obeys commands, and does not explode. The knowledge and experience gained through the process of developing this rover seems almost tangible; this project lab class has been a valuable use time.

References

1. Dahl, Nydal. “What Is an H-Bridge?,” 5 December 2018, https://www.build-electronic-circuits.com/h-bridge/ (5 February 2021)
2. Wikipedia Images, “H-Bridge.svg,” 9 June 2006, https://en.wikipedia.org/wiki/H-bridge (5 February 2021)
3. Digilent. “Basys 3 Reference,” N/A https://reference.digilentinc.com/basys3/refmanual (5 February 2021)
4. Pololu, “2474.jpg,” N/A, https://www.pololu.com/product/2474# (26 March 2021)
5. Last Minute Engineers, “tcs3200.jpg,” N/A, https://lastminuteengineers.com/tcs230-tcs3200-color-sensor-arduino-tutorial/ (26 March 2021)
6. Stelluti, Andrew. Final Presentation Slides, 28 April 2021
7. Sims, Justin. Final Presentation Slides, 28 April 2021

Appendix A

Gantt Chart



GANTT CHART

PROJECT TITLE	Main Project	COMPANY NAME	Pen Island inc.
PROJECT MANAGER	Andrew, Justin, Mason	DATE	3/31/21

WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE
1	**Project Initiation**					
1.1	Gather parts	All	2/10/21	2/24/21	14	95%
1.1.1	Make rover autonomous	All	2/17/21	3/10/21	23	99%
1.2	Rover Rotate code	All	2/17/21	2/24/21	7	99%
1.3	Ceate state Machine	Mason	2/19/21	3/10/21	21	99%
1.4	Get build plate	All	3/16/18	3/22/18	6	100%
2	**Distance sensor**					
2.1	Research Distnace Sensors	All	2/10/21	2/24/21	14	100%
2.2	Choose distance sensor	All	2/17/21	2/24/21	7	100%
2.3	Create circuit for sensor	Andrew	2/24/21	3/2/21	7	100%
2.4	Code for distance sensor	Mason	3/2/21	3/23/21	21	100%
2.5	Debug	Mason	3/16/21	3/23/21	7	100%
3	**Color sensor**					
3.1	Research color sensors	All	2/10/21	2/24/21	14	100%
3.2	Choose color sensor	All	2/17/21	2/24/21	7	100%
3.2.1	Create circuit for sensor	Andrew	2/24/21	3/2/21	7	100%
3.2.2	Code for color sensor	Andrew	3/2/21	3/23/21	21	100%
3.3	Debug	Andrew	3/16/21	3/23/21	7	95%
4	**Strobe sensor**					
4.1	Research strobe sensors	All	2/10/21	2/24/21	14	100%
4.2	Chose a strobe sensor	All	2/17/21	2/24/21	7	100%
4.3	Circuit for strobe sensor	Andrew	2/24/21	3/2/21	7	100%
4.4	Code stobe sensor	Justin	3/2/21	3/16/21	14	100%
4.5	Debug strobe sensor	Justin	3/16/21	3/23/21	7	100%
5	**Project Compilation**					
5.1	Attach All compents	All	3/23/21	4/6/21	14	75%
5.2	Debug	All	3/23/21	4/13/21	21	75%
5.3	Project final paper	All	4/13/21	4/20/21	7	33%

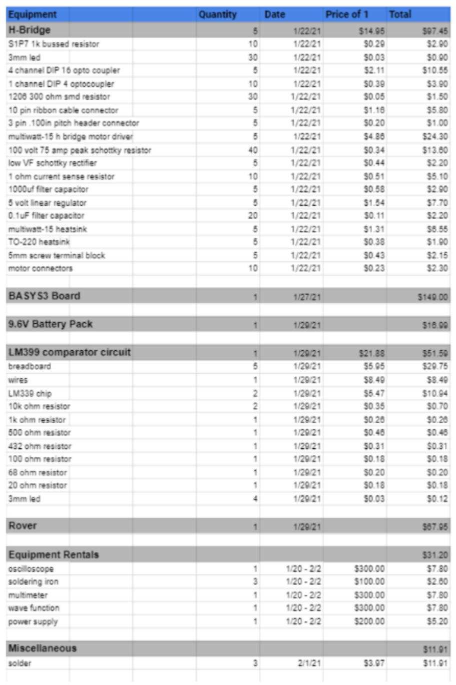
A Gantt chart is used to visually represent time spent on different aspects of a project. Before completion, it is used to gauge how much time should be put into various tasks. After completion, it is used to show that distribution of time in a concise and informative way.

The Gantt chart worked out very well in our case. We managed our time well. We were able to examine what we needed to get done and when to the best of our ability.

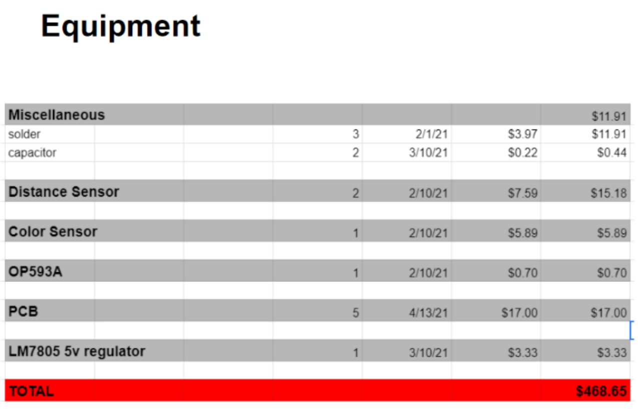
Appendix B

Budget

Week	1	2	3	4	5	6	7	8	9	10	11	12	13	TOTAL
Andrew	12	29	4	7	6	10	15	20	30	8	10	17	4	155
Justin	12	20	7	3	6	10	10	9	12	15	6	12	3	113
Mason	12	27	9	5	4	11	10	12	11	13	11	8	2	148
Total Hours	36	76	20	15	16	31	35	41	53	36	27	37	10	433
Labor Cost	$540	$1140	$300	$225	$240	$465	$525	$615	$795	$540	$405	$555	$150	$6345
Overhead	$540	$1140	$300	$225	$240	$465	$525	$615	$795	$540	$405	$555	$150	$6345
Total Cost	$1080	$2280	$600	$450	$480	$930	$1050	$1230	$1590	$1080	$810	$1110	$300	$12690



Equipment	Quantity	Date	Price of 1	Total
H-Bridge	5	1/22/21	$14.95	$97.45
S1P7 1k bussed resistor	10	1/22/21	$0.29	$2.90
3mm led	30	1/22/21	$0.03	$0.90
4 channel DIP 16 opto coupler	5	1/22/21	$2.11	$10.55
1 channel DIP 4 optocoupler	10	1/22/21	$0.39	$3.90
1206 300 ohm smd resistor	30	1/22/21	$0.05	$1.50
10 pin ribbon cable connector	5	1/22/21	$1.16	$5.80
3 pin .100in pitch header connector	5	1/22/21	$0.20	$1.00
multiwatt-15 h bridge motor driver	5	1/22/21	$4.86	$24.30
100 volt 75 amp peak schottky resistor	40	1/22/21	$0.34	$13.60
low VF schottky rectifier	5	1/22/21	$0.44	$2.20
1 ohm current sense resistor	10	1/22/21	$0.51	$5.10
1000uf filter capacitor	5	1/22/21	$0.58	$2.90
5 volt linear regulator	5	1/22/21	$1.54	$7.70
0.1uF filter capacitor	20	1/22/21	$0.11	$2.20
multiwatt-15 heatsink	5	1/22/21	$1.31	$6.55
TO-220 heatsink	5	1/22/21	$0.38	$1.90
5mm screw terminal block	5	1/22/21	$0.43	$2.15
motor connectors	10	1/22/21	$0.23	$2.30
BASYS3 Board	1	1/27/21		$149.00
9.6V Battery Pack	1	1/29/21		$16.99
LM399 comparator circuit	1	1/29/21	$21.88	$51.59
breadboard	5	1/29/21	$5.95	$29.75
wires	1	1/29/21	$8.49	$8.49
LM339 chip	2	1/29/21	$5.47	$10.94
10k ohm resistor	2	1/29/21	$0.35	$0.70
1k ohm resistor	1	1/29/21	$0.26	$0.26
500 ohm resistor	1	1/29/21	$0.46	$0.46
432 ohm resistor	1	1/29/21	$0.31	$0.31
100 ohm resistor	1	1/29/21	$0.18	$0.18
68 ohm resistor	1	1/29/21	$0.20	$0.20
20 ohm resistor	1	1/29/21	$0.18	$0.18
3mm led	4	1/29/21	$0.03	$0.12
Rover	1	1/29/21		$67.95
Equipment Rentals				$31.20
oscilloscope	1	1/20 - 2/2	$300.00	$7.80
soldering iron	3	1/20 - 2/2	$100.00	$2.60
multimeter	1	1/20 - 2/2	$300.00	$7.80
wave function	1	1/20 - 2/2	$300.00	$7.80
power supply	1	1/20 - 2/2	$200.00	$5.20
Miscellaneous				$11.91
solder	3	2/1/21	$3.97	$11.91



Equipment

Miscellaneous				$11.91
solder	3	2/1/21	$3.97	$11.91
capacitor	2	3/10/21	$0.22	$0.44
Distance Sensor	2	2/10/21	$7.59	$15.18
Color Sensor	1	2/10/21	$5.89	$5.89
OP593A	1	2/10/21	$0.70	$0.70
PCB	5	4/13/21	$17.00	$17.00
LM7805 5v regulator	1	3/10/21	$3.33	$3.33
TOTAL				$468.65

Labor - $12,690

Equipment - $468.65

TOTAL - $13,158.65

Justin

The budget charts were theoretical amounts of money that would be spent to allow us to complete our project had we actually been working for an employer. The total came out to upwards of $13,000 dollars, which is quite large. We recorded equipment used, time spent, and materials used.