LAB TEAM 6

Project Lab 1

Texas Tech University

ECE 3331

Project Notebook

Table of Contents

Table of Figures

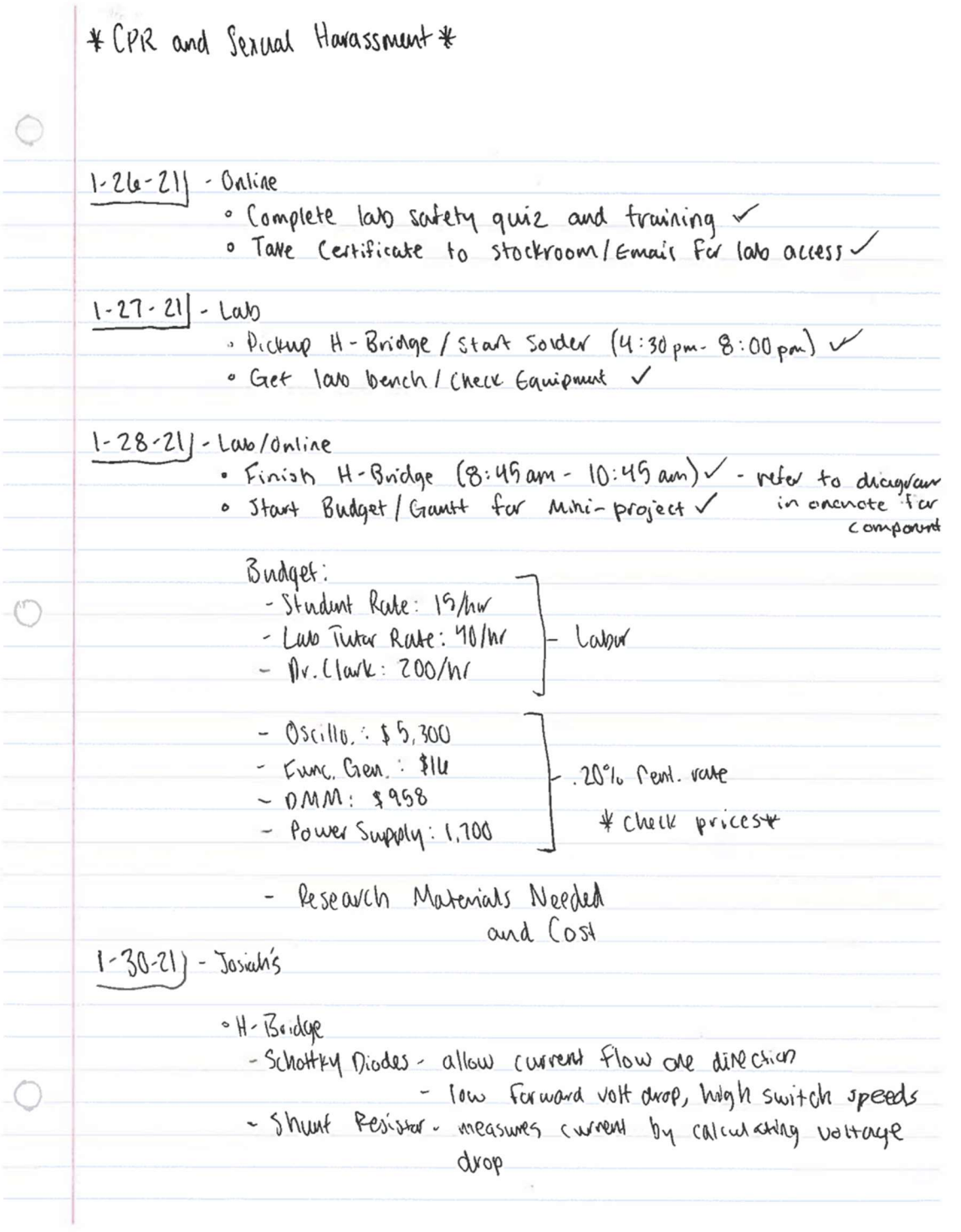
Table of Tables

Table of Graphs

List of Charts

Division I

A. Brendan Blacklock – Notes



* CPR and Sexual Harassment *

1-26-21 - Online
- Complete lab safety quiz and training ✓
- Take Certificate to stockroom / Email for lab access ✓

1-27-21 - Lab
- Pickup H-Bridge / Start Solder (4:30 pm - 8:00 pm) ✓
- Get lab bench / Check Equipment ✓

1-28-21 - Lab / Online
- Finish H-Bridge (8:45 am - 10:45 am) ✓ - refer to diagram in oneNote for components
- Start Budget / Gantt for Mini-project ✓

Budget:
- Student Rate: 15/hr
- Lab Tutor Rate: 40/hr
- Dr. Clark: 200/hr

} Labor

- Oscillo.: $5,300
- Func. Gen.: $116
- DMM: $958
- Power Supply: 1,700

} 20% rent. rate
* check prices *

- Research Materials Needed and Cost

1-30-21 - Josiah's
- H-Bridge
  - Schottky Diodes - allow current flow one direction
    - low forward volt drop, high switch speeds
  - Shunt Resistor - measures current by calculating voltage drop

- Finish Gantt Chart
- Research/watch verilog videos
- Tank for Budget

- H-Bridge
  - Voltage Regulator
    - Max Voltage - 35 V
    - Regulates to 5 V - Basys needs 5V power supply

Operates at 3.3, 1.8 or 1 V

- Vivado - ~~bit-stream~~ regulations for quicker transfer
- Basys 3
  - 7-Seg - Multiplexing required
    - Uses transistors to drive current into anode, therefore, enables for anodes are inverted. (driven low when active)

ex. Number	Common Anode	CA	CB	CC	CD	Cathode
0	high	low	low	low	low	7'b0000

CE	CF	CG
low	low	high

- PMod Type 5 (H-Bridge)

Pin	Signal	Direction
1	DIR	Out
2	EN	Out
3	SA	In
4	SB	In
5	GND	
6	VCC	

DIR - Motor Direction
EN - Enable Motor, active high
SA - Feedback sense A
SB - Feedback sense B

OR

Pin	Signal	Direction
1	DIR1	Out
2	EN1	Out
3	DIR2	Out
4	EN2	Out
5	GND	
6	Vcc	

DIR1 - Motor 1 Dir
EN1 - Enable mot
DIR2 - " 2
EN2 - "

- Basys3 - cont'd.
  - 33,280 logic cells in 5200 slices (slice = 4 6-input LUT's and 8 flip-flo
  - 1,800 Kbits of fast block RAM
  - 100 MHz Oscillator
  - FPGA - Artix 7 cpg236

- Vivado
  - Create Project
  - Add Source, Code Modules
  - Save, Run Synthesis
  - Create Testbench, Used in Simulation ☑
    - Step 1: Copy I/O, change I to Reg and O to Wire
    - Step 2: Instantiate module you want to test (control one module → inside another)
    - Step 3: Stimulate, assign registers
    - Step 4: Run Simulation (Simple way)
    - Step 5: Add constraints (Master XDC), Save, Re-run synthesis
    - Step 6: Setup QSPI, Tools → settings → bitstream → bin.file
    - Step 7: Synthesize, open design, tools → device properties
    - Step 8: Generate Bitstream
    - Step 9: Open hardware target, (Open Hardware manager) → Config rate = 33 Enable bitstream comp = true; master SPI x4
    - Step 10: Add Config Mem Device (s25fl032p) → change clock to 30 MHz

2-1-21 Testing H-Bridge w/o Basys
- Power supply
- function Generator (useful) (Waveform right?)
- Oscilloscope - (triggering) (coupling)

2-3-21 • LM339 ~~Comparator~~ Comparator - used for sensing current ← overcurrent protection

• How to implement readings in verilog?

2-5-21 Verilog Basics

• Data types - wire and reg
  • wire = continuous assignments
  • reg = procedural assignments
• Delays
  • #5 A=1; , At 5ns A=1
  • #5 A=2; , At 10ns A=2 } Blocking
  • #5 A<=1; At 5ns A=1
  • #10 A<=2; At 10ns A=2 } Non-blocking

• Blocking and Non-blocking

```
reg A=1;          reg A=1;
reg B=0;          reg B=0;
A<=B; //A=0       A=B; // A=0
B<=A; // B=1      B=A; // B=0
```

• Always @ (sensitivity)

```
begin
end
```

Always @ (posedge clk)

```
begin
end
```

```
reg[2:0] state=S0;
case(state)
  S0:
  S1:
  default:
end
```

• Case (A)

```
1:
2:
default:
end
```

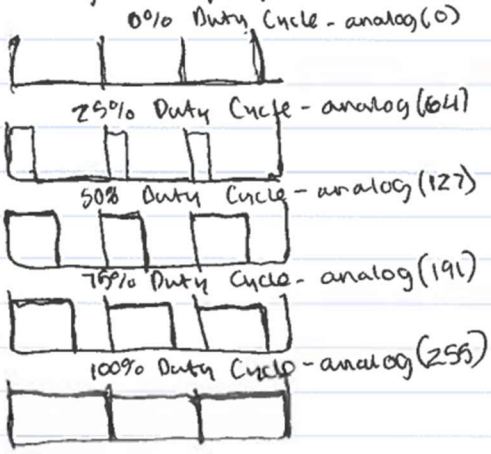
If A=1, that will execute

```
casex(thing)
  4'bxxx1:
  default:
end
```

If LSB of thing is 1, 4'bxxx1 will execute

PWM

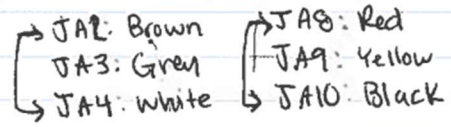
- Reducing average power by chopping signals up



Verilog: Set Period (counter)
Current Width (duty cycl

- le different PWMs for each motor? (switches)

PMOD: Blue/orange/green/purple] - ground



Software

1. Adjust seg display for current sense
2. Current sense module
3. Pmod interfacing - XDC
4. Switches

2-16-21) Pmod XADC
- Used to input analog signals to ADC inside Artix-7

2-16-21) Software Overcurrent
- Logic/code worked when input wires were plugged into GND and 3.3 V pmod ports.
- Tested sensA output, only read above 1V for PWM 100
  - averaged around .150 volt w/o holding for all pwm
  - Input from 3.3V to comparator is dividing to a voltage of 1 V.
  - Output of comparator stays low

2-25-21) Main-Project
- Color sensing: Research TCS3200 Color Sensor ← Display some how w/visual indicator (Flag?)
- Wall Avoidance: Sharp GP2Y0A60SZ0F Sensor (IR)
- Command: Communicate Go, Pause, Resume, Stop (Morse-code)
  - Photo diode or Photo transistor?

3-1-21) - Wall Avoidance: Sharp GP2Y0A60SZLF (can be affected by direct light)
- 60 Hz update rate, capable of measuring distances (4 in - 60 in)
- Infrared sensor ↑ Lenses
- Operating Voltages:
- 5V - 2.7V to 5.5V (Output 3.6V - .6V)
- 3V - 2.7V to 3.6V (Output 1.9V - .3V) } Analog

← on by default through 10 pull-up

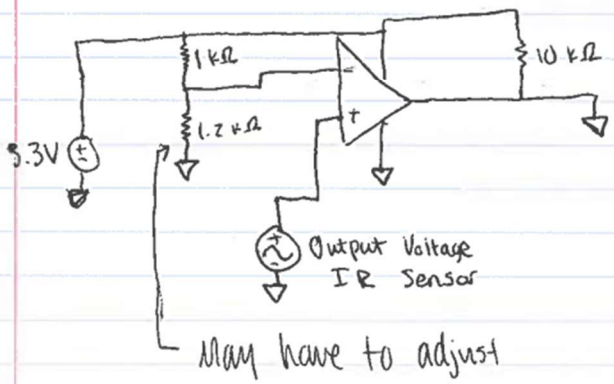
* Implement output w/ reference voltage and LM339

- 1 sensor in front, maybe 1/2 more on side

IR Distance Sensor 3-4-21

Distance (in)	50	40	30	25	20	15	10	8	6	4	2	0
Output Voltage (V)	1.03	1.14	1.32	1.48	1.61	1.7	1.85	1.98	2.38	3.25	N/A	N/A

- Want to stop around 10 in or 1.85 V
  - Comparator circuit w/ output of IR Sensor



Low when inverting V < 1.85 V
High when inverting V > 1.85 V

* May want to implement capacitor to limit noise*

May have to adjust

3-6-21

- Implement State machine

If state = 0 (sensor = 0)
- Continue, set state = 1

Else If state = 1
- Check sensor = 1, if so state = 2
- Otherwise, drive forward, state = 1

Else If state = 2
- Stop for period, turn, state = 3

Else If state = 3
- Based on 'random' counter, turn more, state = 0

* Code works in terms of detecting wall turning until sensor ≠ 1, then turning more

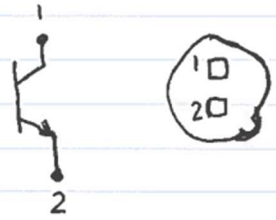
- Need to play w/ counters/pwm

3-7-21

More on OP593 Phototransister and Morse Code

- Need to be able to send signal from atleast 5 feet.
- Look into High-power LED

- OP593
  - 130° viewing angle
  - Collector-Emitter Voltage = 30 V
  - Emitter-Collector Voltage = 5 V
  - Continuous Collector Current = 50 mA

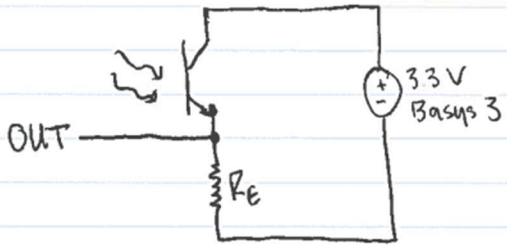


Pin 1 = Collector
Pin 2 = Emitter

- Must be hit with light at top of phototransistor

- Low Voltage = No Flash, High Voltage = Flash

- Common-Collector Amplifier?



R_E = 470 Ω?

- Won't amplify voltage over 3.3 V

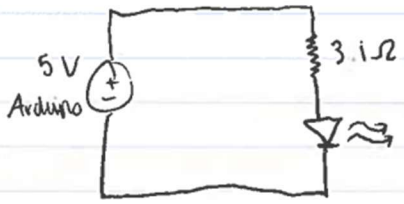
3-16-21

- OP593 common collector amplifier works well, but iPhone flashlight does not have the distance thought.

- Use Arduino w/ high-power LED?
- LIHX - High-power LED
  - $V_F = 2.83\ V$
  - Forward current = 700 mA

  - Arduino outputs 5 V from logic pins

$$\therefore \quad R_1 = \frac{5 - 2.83}{.7} = 3.1\ \Omega$$



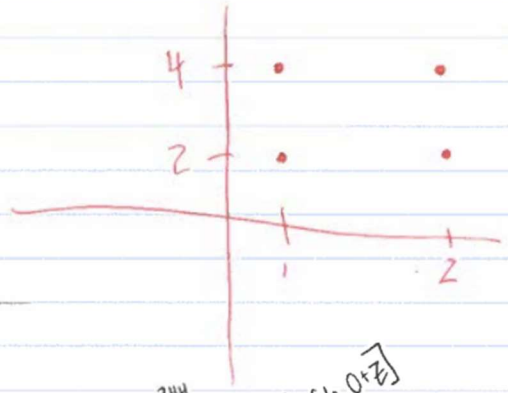
- LIHX ordered ✓

3-22-21

- Wall Avoidance - Snaking or Controlled-degree turns?
  - May need to add more sensors for snaking

  - 3, 90° turns, and 4th turn is random degree
    - Work on python simulation
      - Loop through for how many times hit
      - Dot Product to calculate next points from start
      - Py Cairo

3-27-21

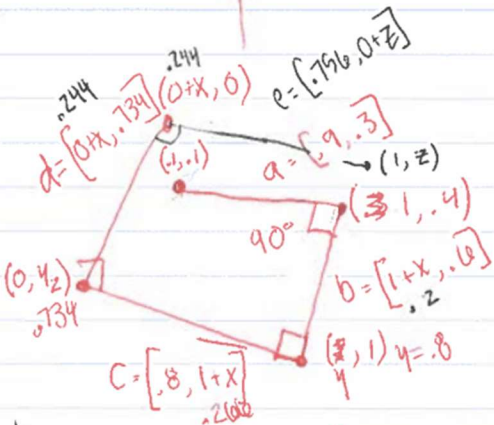


$a_x \cdot X + a_y b_y = 0$

$X = -\frac{a_y b_y}{a_x}$

$(.756 \cdot .244) + (0+z) \cdot 0$

bottom left to right



$a_x X + a_y \cdot b_y$

$b = [.2, .6]$

$a_y X + a_x \cdot b_x = \cos(100°)$

$a \cdot b = .9 \cdot (1+x) + .3 \cdot .6$

$= .9 + .9x + .18$

$|a| = \sqrt{.9^2 + .3^2} = .9487$

$|b| = \sqrt{(1+x)^2 + .6^2} = \sqrt{x^2 + 2x + 1.36}$

$\cos(90) = \frac{.9 + .9x + .18}{.9487 \cdot \sqrt{x^2 + 2x + 1.36}}$

$X = -1.2$

$\frac{a_x \cdot .2(1+x) + a_y \cdot b_y}{\sqrt{a_x^2 + a_y^2} \cdot \sqrt{b_x^2 + b_y^2}}$

$= (1+x)$

$\cos(90) =$

$\frac{a_x + a_x X + a_y \cdot b_y}{}$

$\frac{(a_x + a_x X + a_y \cdot b_y)^2}{(a_x^2 + a_y^2)(x^2 + 2x + 1 + b_y^2)}$

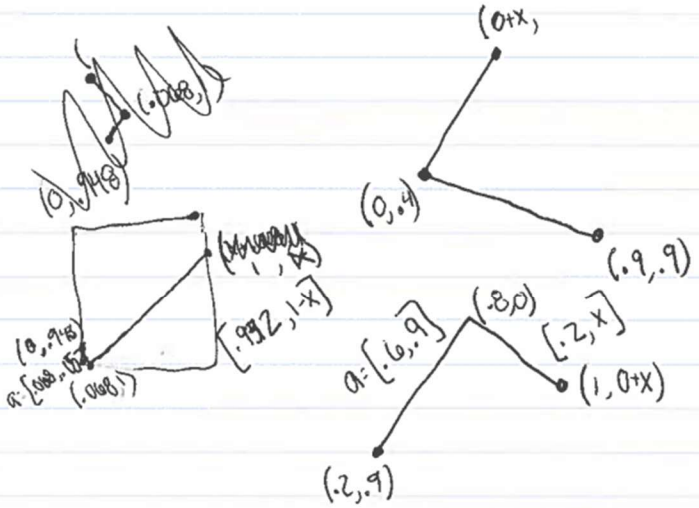
$a_x + a_x X$

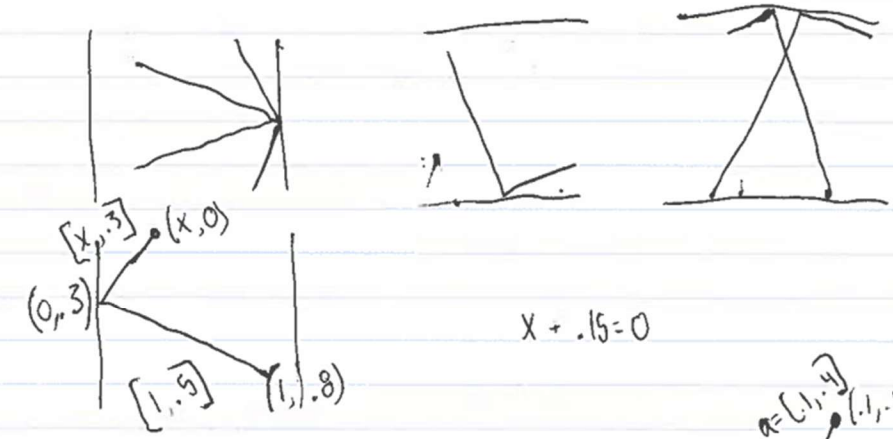
$= 2a_x^2 X + a_x^2 x^2 + 2(a_x a_y b_y) + 2(a_x a_y b_y X + a_x^2 + a_y^2 b_y^2$

$a_x (1+x) + a_y b_y = 0$

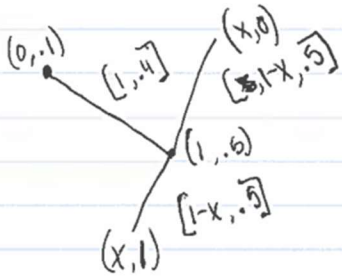
$a_y b_y = -a_x (1+x)$

$a_y b_y + a_x = -a_x X$

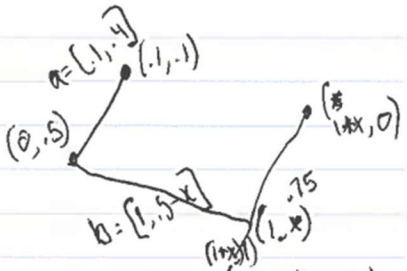




$x + .15 = 0$



$.1 + .2 - .4x = 0$



$a_x \cdot b_x + (a_y \cdot (b_x - x)$

$a_x \cdot b_x + a_y curY - a_y x$

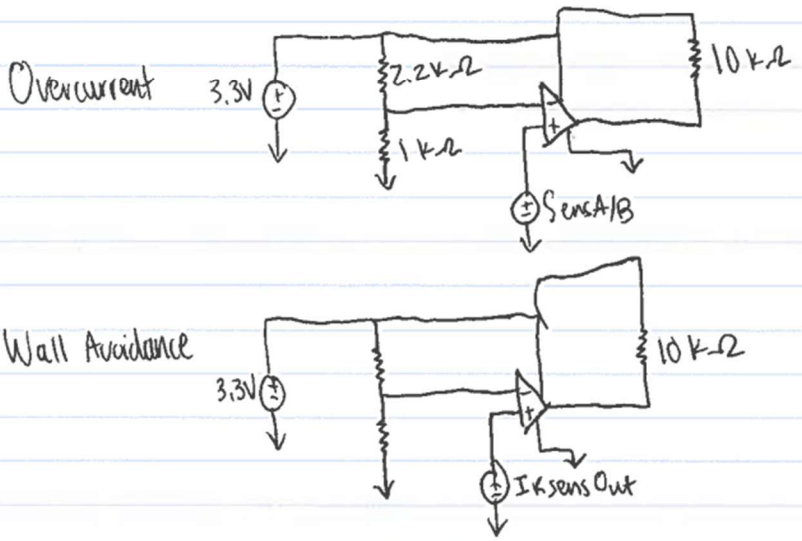
4-1-21

- Python Vector Simulation works
  - Previous idea proved a pattern of not reaching the center of the playing field.

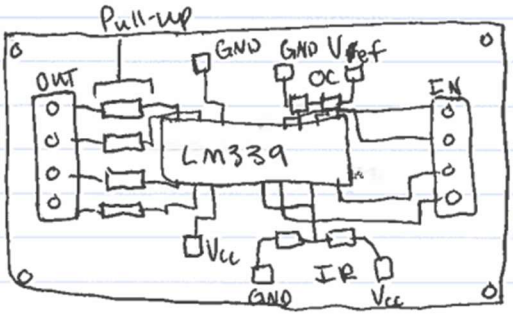
- Whats next? - Wall avoidance
  - Do 3, 90° degree, random 4th but turn after reading a color card
  - JX ADC - snaking (check parallel, 90° turn, then 180°)
  - Find one solid degree turn
    - Michael says 1160°

- PCB Design - Easy EDA?

  - Want to implement LM339 Comparator to include overcurrent protection and wall avoidance circuit.



- PCB - Continued
  - 1-layer w/ soldering mask
  - must have proof of order by April 5th
  - EasyEDA
    - Easiest to use and quick order from JLCPCB



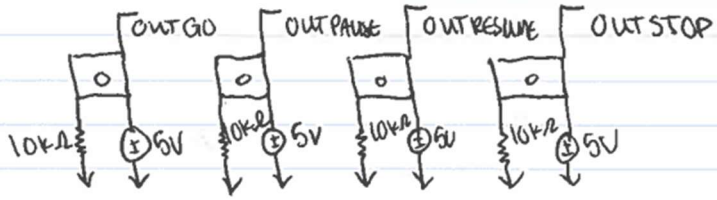
Rough sketch

- PCB Ordered ✓
- Jacob refined frequency ranges for TCS3200 and color sensor is now reading all colors with no false positives

4-8-21

- LIHX High-Power LED is not what we thought
- New beacon?
  - Alter flashlight
    - Replace switch w/ NPN transistor
    - Base current from Arduino based on buttons

- Push buttons and Arduino

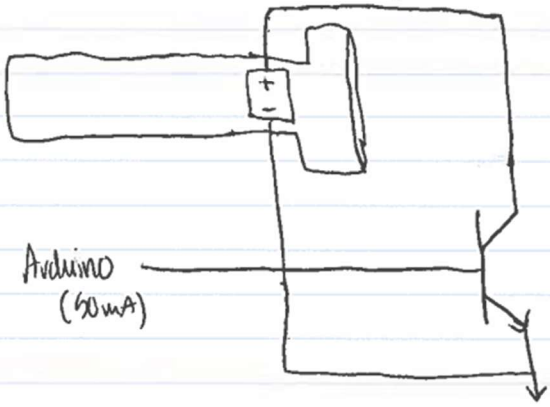


- Arduino code
  - Checks state of buttons,
    If button was pressed,
    (e.g., GO pressed → Dash, Dash, Dot)

```
Dash( );
{ High for .7 s;
  Low for 1 s;
}
```

```
Dot( );
{ High for .35 s;
  Low for 1 s;
}
```

- Transistor, ensure 50 mA is enough base current to allow flow through emitter



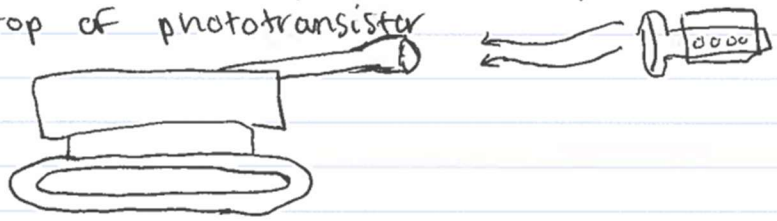
- 1200 Lumen Flashlight had 3 modes that were difficult to remove, find simple flashlight.

4-14-21

- Safety flashlight found
  - Implemented into strobe beacon circuit and functioned as intented.

  - Only 100 Lumens, so don't get range from further than 5 Ft.

  - Implement cone design to focus light on top of phototransistor



4-16-21

- Morse Code State-Machine

- State zero
  - Count length of high signal and low signal
  - Check if commands have been given
  - If low sign > 1s reset dot and dash count

- State One
  - If length of high = dot
    dot = dot + 1
  - Else length of high = dash
    dash = dash + 1

- State 2-5
  - Check dot and dash count, if command matches, tell the rover to go, pause, resume, and stop

Go → Pause → Resume → Stop

4-18-21

- Morse Code State-Machine tested w/ testbench and functioned as expected.

- After testing, the rover proved functionality but it was difficult to hit the phototransistor while the rover was moving.

  - Maybe shorten delay and length of dash and dot.
  - Less time ☺ Less chance of error?

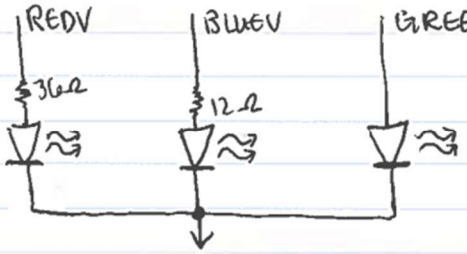
- Tay and I soldered all the components onto the PCB and tested using the multimeter and dc voltage supply.
  - PCB functioned as expected

- Current-State of Project
  - Wall Avoidance – Jacob working w/ JXADC for sensing
  - Color Sensing – reads colors, need to implement RGB LED.
  - Morse Code – all components work, work on precision
  - PCB – DONE (overcurrent)
  - Final Mounting and Transfer circuits to solderable breadboard

4-21-21

- Shortened times for morse code dot and dash
  - Lots more precision when moving
  - Implement L-brackets and Funnel/cone.

- Color Sensor Indicator
  - LED - Red, Blue, and Green
    - Red: 36Ω
    - Blue: 12Ω
    - Green: 0Ω
  - Implement on bread board with OPB93



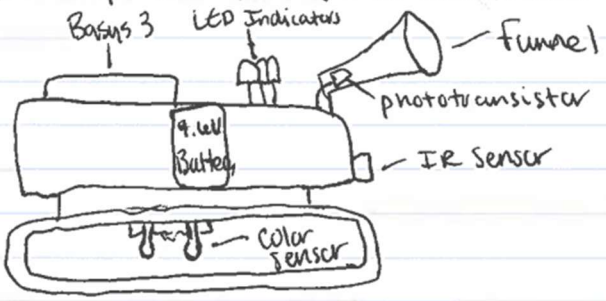
  - After construction and testing, the visual indicators displayed the correct color it was reading.

- With only a few days left, we are abandoning the snaking algorithm.

- We will still utilize JXADC for wall avoidance, but will do 1160° turn when the rover hits the distance threshold.

4-24-21

- All circuits were put on solderable breadboards and tested. Each circuit worked as intended.
- The color sensor was mounted underneath w/ velcro. The visual indicators are on the top of the junction box.
- The PCB and H-bridge circuit are secured inside the junction box.
- A voltage divider was implemented because the JXADC only reads between 0-1 V (IR Sensor). This divider is on the top of the junction box on the same board as the LEDs and OP593.
- The OP593 is mounted on the top of the junction box with a Funnel directing light to the top of the phototransistor.



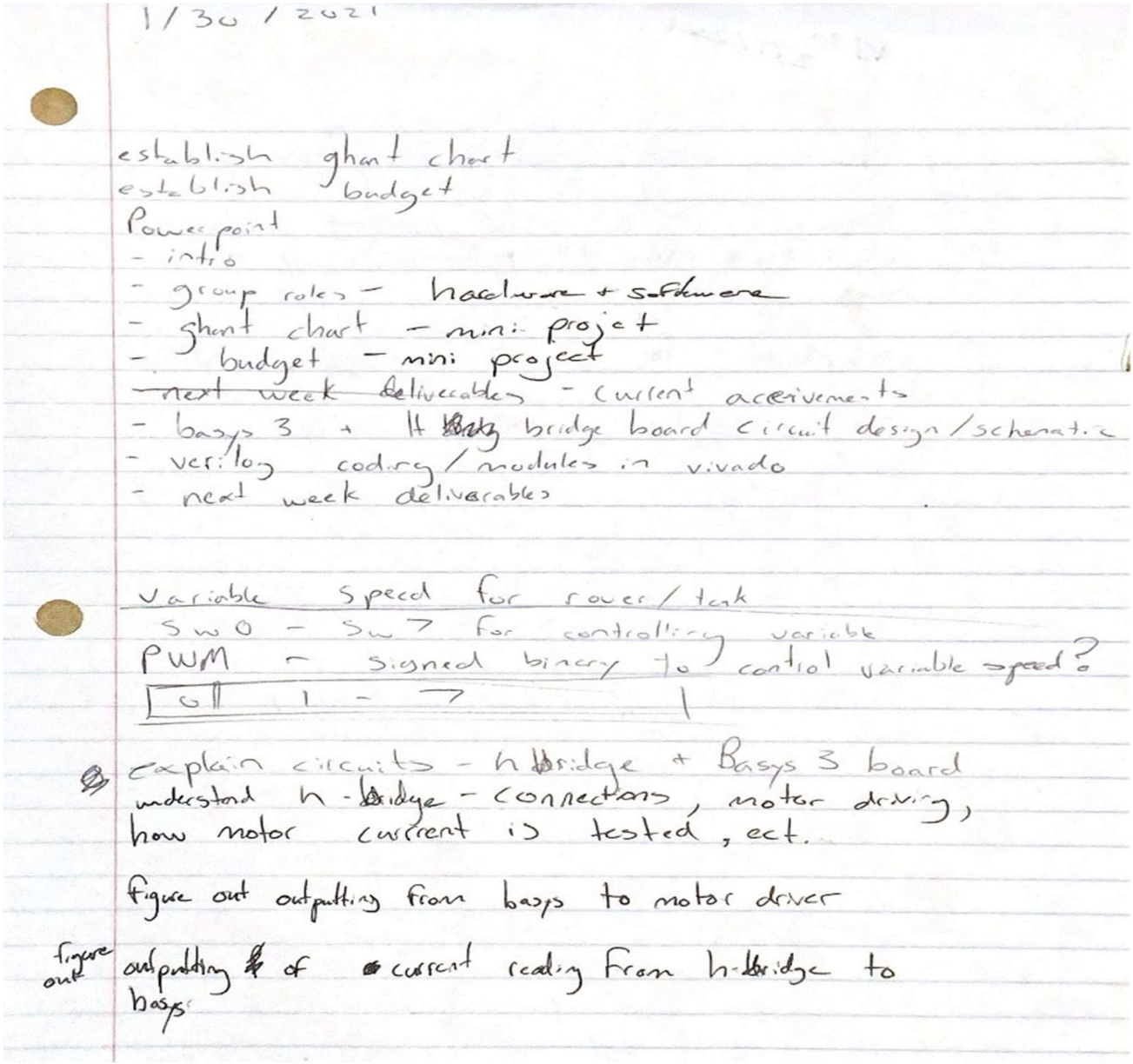
4-25-21

- Testing in Rawls Demo Room
  - Wall Avoidance ✓
    - Works good, might increase threshold
  - Color Sensing ✓
    - No false detections, reads card correctly and LED flashes
  - Morse-Code ✓
    - Works great, distance is further w/ cone
  - Overcurrent ✓
    - Works at 100% PWM

- May increase PWM on turn due to faulty motor gear on Rover 5 chassis.

- ALL SYSTEMS ARE A GO ✓

B. Jacob Holyoak – Notes



1/30/2021

establish ghant chart
establish budget
Powerpoint
- intro
- group roles - hardware + software
- ghant chart - mini project
- budget - mini project
- ~~next week deliverables~~ - current accomplishments
- basys 3 + H bridge board circuit design/schematic
- verilog coding/modules in vivado
- next week deliverables

variable speed for rover/tank
sw 0 - sw 7 for controlling variable
PWM - signed binary to control variable speed?
0 || 1 - >

explain circuits - h bridge + Basys 3 board
understand h-bridge - connections, motor driving, how motor current is tested, ect.

figure out outputting from basys to motor driver

figure out outputting of current reading from h-bridge to basys

2/11/2021

Shunt resistor - voltage drop across

Slide 6 talk about ~~how~~ our H bridges boards & how many we have working

Comparator for current sensing & controlling

Voltage regulator - 7805

Function generator - generate pwm type signal
oscilliscope AC vs DC voltage
DC coupling
oscilliscope - NO Autoset
research - triggering + coupling

Not Negotiable - figure out pwm frequency - what is a motor
what is in the motor
motor has capacitance

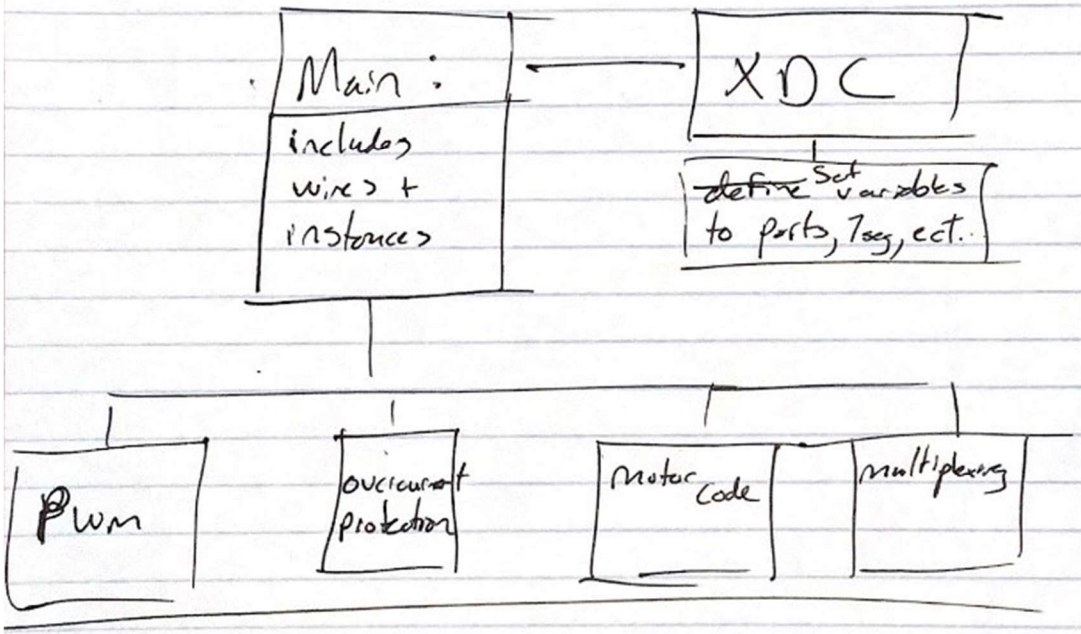
2/1/2021

github:

create group section for saving working versions of code.
upload working versions of modules or full code

---

code hierarchy:



Pwm - 4 settings can be set using 2 switches
as binary 00, 01, 10, 11
equating to 25%, 50%, 75%, 100%.

to achieve higher current we can increase this pwm to its max (100%)

2/7/2021

multiplexing
clock - oscillator
Clock counts up based on the 100 megahertz clock on basys 3 board 18 bit counter allows us to multiplex at 1000 Hz

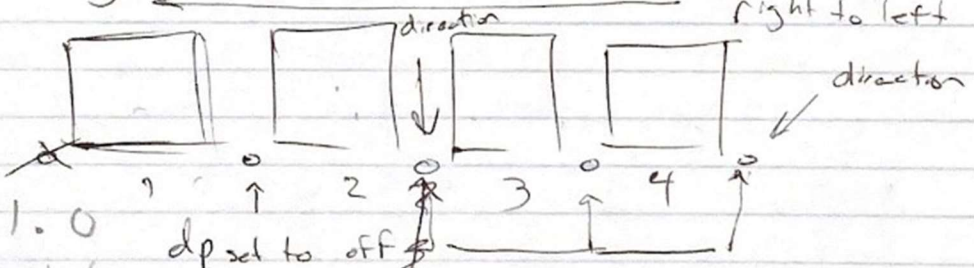
SSeg - 0-9 active low on enables and individual led outputs

7bit reg Sseg - ~~[illegible]~~ register holds the data to output

4 bit an_temp register for enables

7 bit reg sseg temp

Verilog does not allow manipulation of outputs



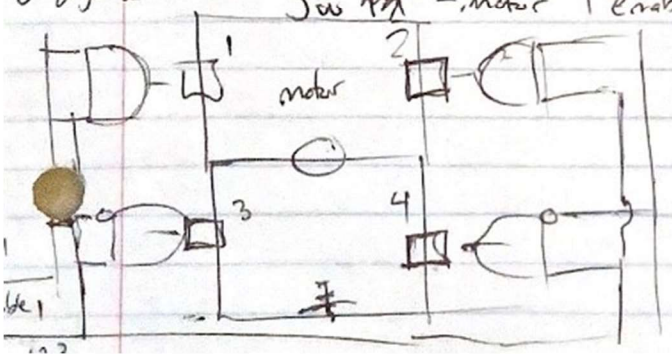
< 1.0
at 1.0 shutdown
dp set to off

7 4
Sw 15 - Sw 8

3
Sw 7 - Sw 0

Sw 15 - motor 1 direction
~~Sw 14 - motor 1 enable~~
Sw 14 - motor 1 enable

Sw 7 - motor 2 direction
~~Sw 6 - motor 2 enable~~
Sw 6 - motor 2 enable



[4:0]
reg CounterCurrent ← initialize 2/20/2021
reg [4:0] CounterCurrent_limit

always @ (posedge clock)
begin
if (currentSenseA == 1)
counterCurrent <= CounterCurrent + 1
else if (currentSenseB == 0)
counterCurrent <= 0
~~else if (counterCurrent == counterlimit)~~
~~enableA_temp <= 0~~
~~enableB_temp <= 0~~

case ((currentSenseA)
case ((currentSenseB

B

B

pwm always @ motorA enableA
begin
case
~~begin~~ ... pwm enable code
endcase
if (currentCounterA <= CurrentCounterB_limit)
enableA_temp = 0
~~else~~ if (currentCounterB == currentCounter_limit)

if	pwm messes with turning off motor enable - in software protection - make logical variable that stays high until counter reaches limit and resets to 1 when logical currentSenseA reads 0 again
else if	
if	
else if	

import -0 for current sense A +B logical variable that is 1. that variable sets to zero when counter in software protection meets the limit. ~~AND~~ that variable with pwm reset variable back to 1 when currents sense A + B read 0 again

3/10/2021

Arduino readings for color frequency half pulse

Red card:

red	min	46
	max	58
blue	min	129
	max	144
green	min	174
	max	191

Blue Card

red	min	192
	max	227
blue	min	125
	max	151
green	min	80
	max	108

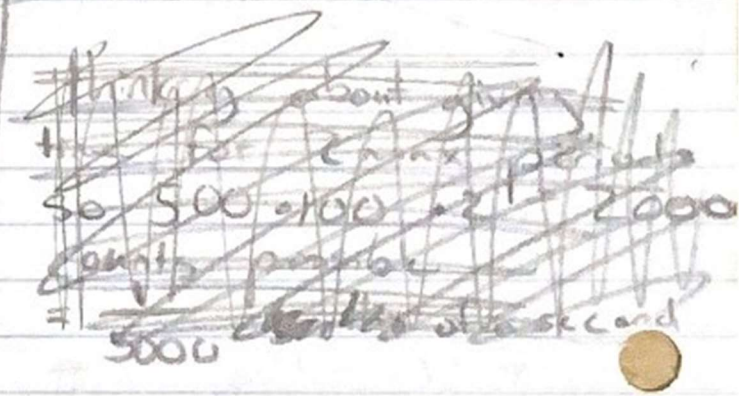
Green Card

red	min	174
	max	223
blue	min	34
	max	53
green	min	80
	max	110

Microsecond = 100 counts on 100MHz clock

Counter 0-100

another counter increments every time first counter hits max



every 100 counts on fpga is equal to 1 count on the frequency read through arduino

might be some slight offset due to arduino clock being smaller

10

March 2021

Arduino testing:

Using Arduino uno tested 2 versions of the tcs3200 color to frequency sensor to read the ranges of the frequencies outputted for each photodiode type

by testing the ranges for each color 100 times and expanding if we get a value indicating a higher range then we can get an accurate range for the photodiode type readings for each color

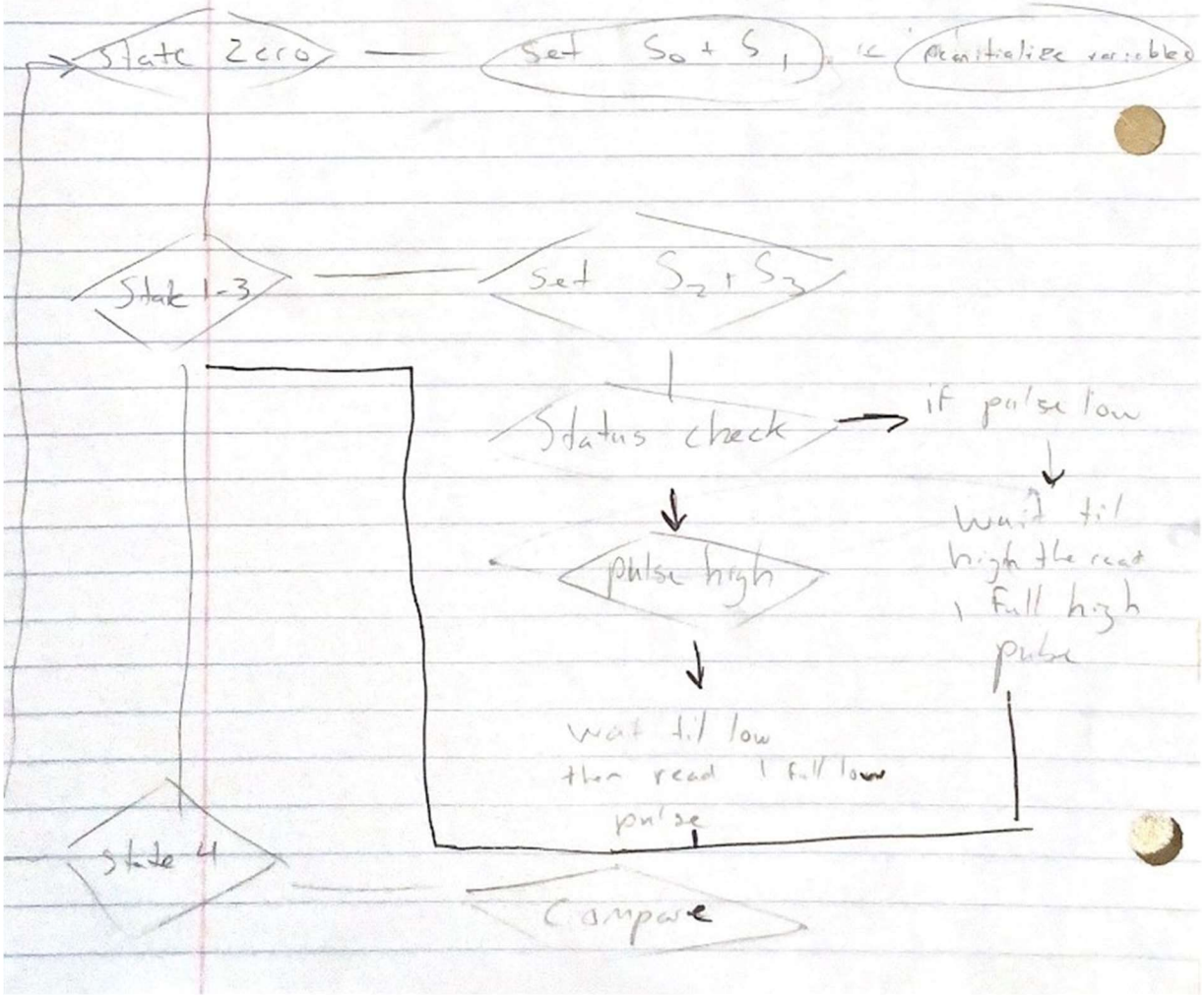
this can be implemented into fpga afterwards and is much more reliable than using oscilliscope readings

3/24/2011

Set S_0 + S_1 at beginning for 20% Freq
~~Status~~
Set S_2 + S_3 to photodiode type 2 check for "status check"
10 uS delay →
- if pulse is high we wait until low + count for length of low pulse
- if pulse is low we wait until high + count for length of high pulse

do this for each photo diode type then compare in state 4

State 1-3 photodiode Red, blue, green check

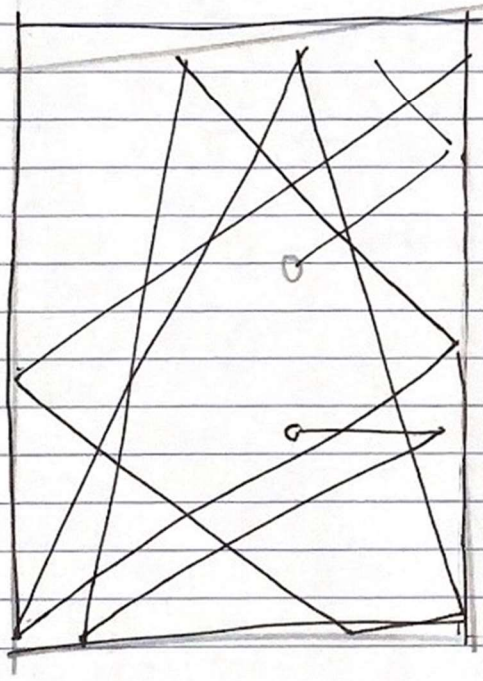


3/29/2021

demo room:
90° TURN

timing turns for various surfaces such as lab and demo room floor

turns of 160 seemed to make best coverage as observed through python simulation
turns at angles of 160:



50% pwm - 2.66 secs

50% - 2.59
~~2.86~~
2.59
2.59
2.65

75% - 1.49
1.40
1.38

since it is easier to time turns of 90° then convert that ratio for 160° we are ~~calculating~~ checking for 90° turns

after some test it does appear the battery charge has a large effect on turn + movement speed

tread inaccuracy/clicking also causes the rover to shift left constantly

4/2/2021

Color Sensor

color sensing filters through 3 photodiode types: green, red, & blue. each time it checks the state of the pulse before it starts reading. IF the pulse is high it waits until the pulse is low and reads that full low pulse. If the pulse is low it waits until the pulse is high then reads that full high pulse. This is to ensure we get a full half pulse for our frequency reading. we can use a half pulse because the pulses are always 50% pwm. The length of the frequency is determined by a counter that increments will reading the pulse. Using the arduino readings we can know the ranges necessary for each photodiode to determine if we are over a red, blue, or green. If all photodiode readings fit the qualifications for the ranges for a color then the fpga will identify that as that color and output to the 7 seg + display LEDs.

8x8 photodiode array consist of a mix of red, blu, green, + clear photodiodes used to determine a color. the clear can be used to decrease the factor of ambient light or the readings but for this project it was not necessary.

4 / 10 / 2021

Modules

Main

main hierarchal module used to instance all other modules + pass variables through wires

Pwm

sets variable speed through switches + feeds output to Motor Code

Color sensor

detects color + outputs to Motor code, 7 segment, + LEDs (Blue, green, Red) Motor code stops for a second when a color is detected + 7 segment shows the color

Motor Code + Morse

implements navigation of field area + Morse communication motor responds to inputs from light signal, IR sensor, even the color sensor, + Pwm

7 Segment display

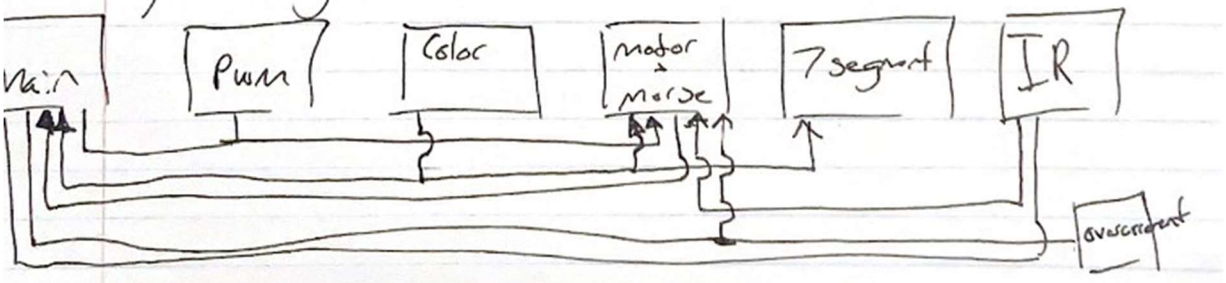
displays the color outputted from color sensor module + overcurrent module

IR sensor + ADC wizard

using the ADC_wiz to configure the mux to take an analog input for the distance. outputted to Motor code

Over Current protection - Software

takes input from h-bridge for current over motor outputs to motor code + stops rover if system goes overcurrent



4/13/2021

IR sensor JXADC:

pmod port JXADC is a dual usage port allowing for simple digital inputs + outputs and configuration to take on analog voltage signals of 0-1 volts

Using the JXADC wizard to configure the daddr address for the muxer the jx[redacted] ADC can understand a analog input on a scale of 0-1 volts every approximately 26th posedge of the clock where the muxer outputs a 16 bit number taking the 12 highest bit gives a reasonably high resolution without allowing for huge fluxuations from the sensor readings

this 12 bit number is the voltage reading for the distance to the wall

since the max output of the IR sensor is 3.3v this must be scaled down using a voltage division

using a paired port on the jxadc we ground one port as the negative signal / lower bound and max the other the output of the sensor these values are run through the adc wizard to the muxer

resolution of 12 bit number on scale of 0-1 volts is 244 micro volts

4/17/2021

XDC port assignments

use JA, JB, JC, & JXADC

all grounds + 3.3 Vcc in use for color sensor, hbridge board, pcb, and soldered vector board containing the photo transistor, display LED, and IR sensor!

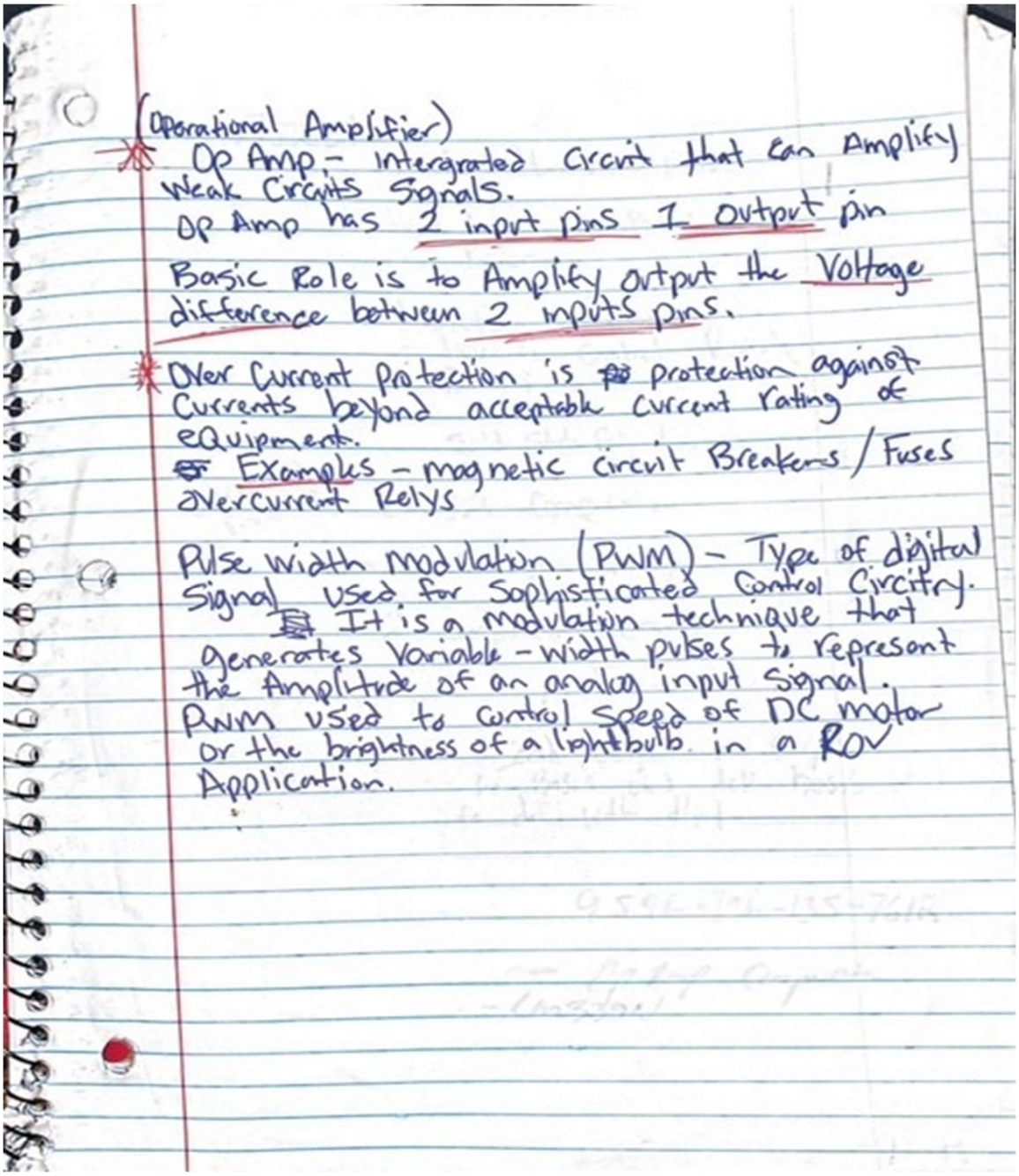
JA:
Current Sense A + B, Light Signal
Red LED, Blue LED, Green LED

JB: color sensor_input
S0, S1, S2, S3 for color sensor
frequency settings + photodiode selector

JC: ~~Hbridge~~ Rf bridge
enable A + B
inputs 1-4 for motor movements
inputs_tmp = 0000 - stop & 0110 - forward
1010 - right 0101 - left

JXADC IR sensor positive + negative
paired port to receive ~~ADC~~ Analog signal
and convert to digital

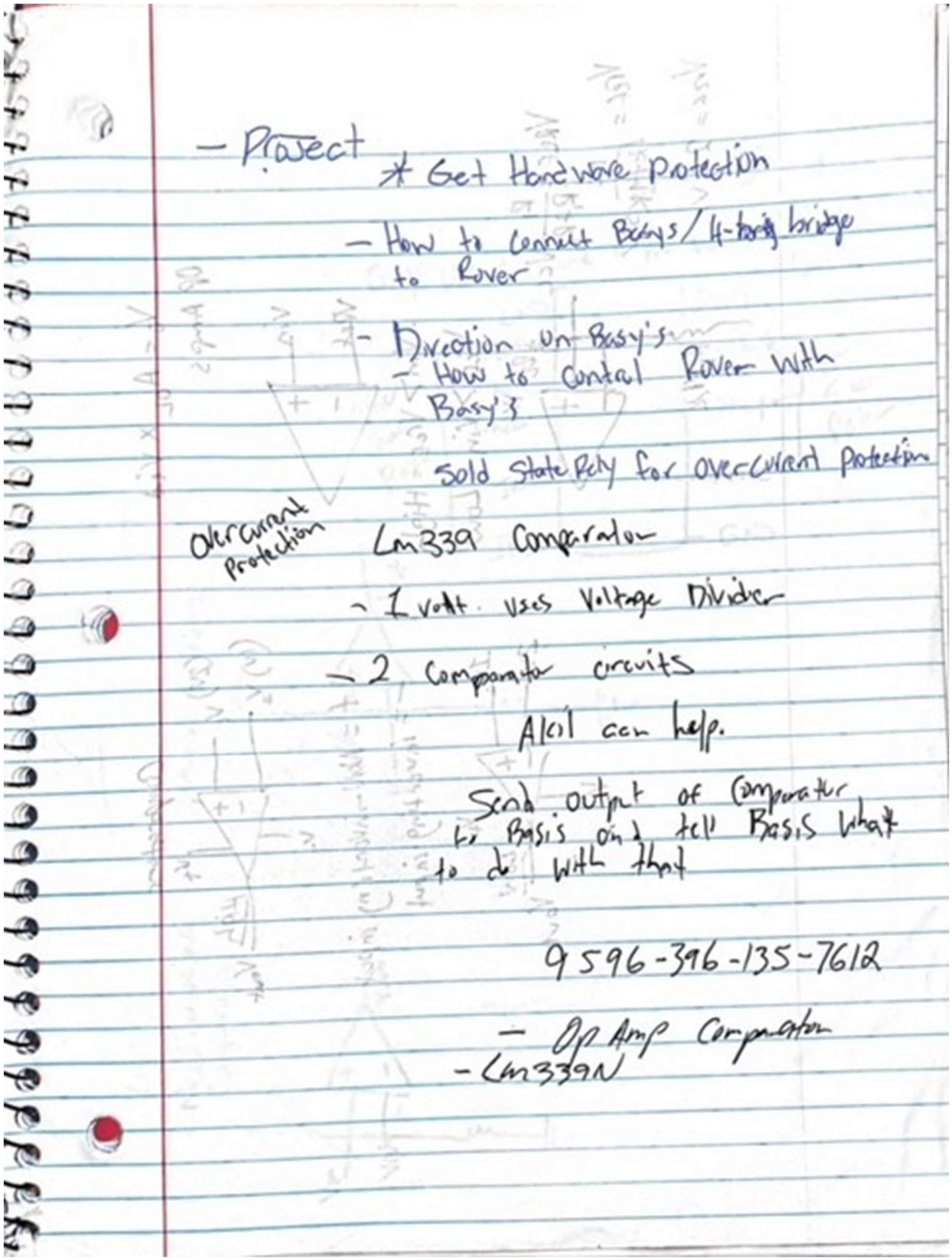
C. Josiah Ramirez – Notes

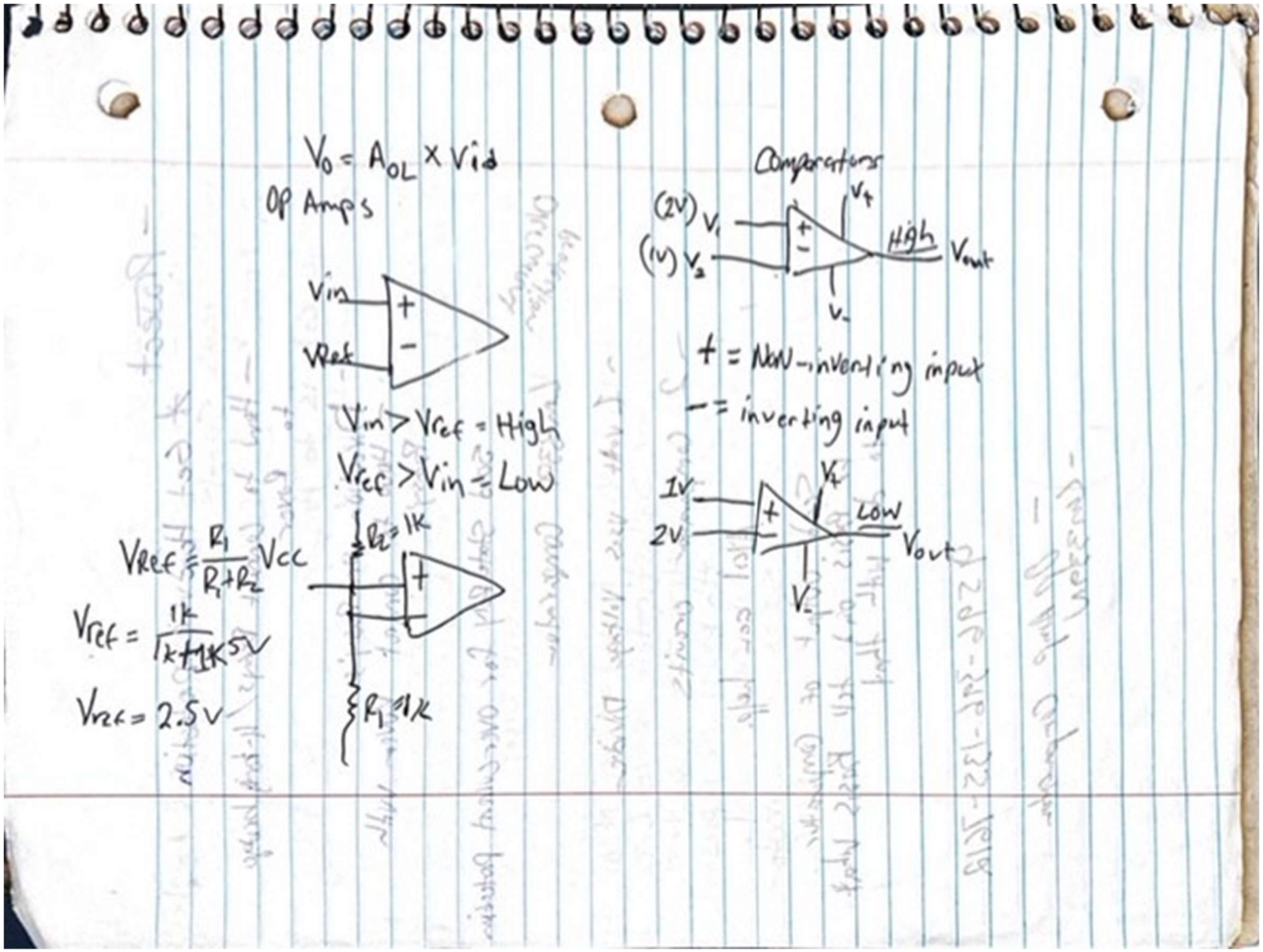


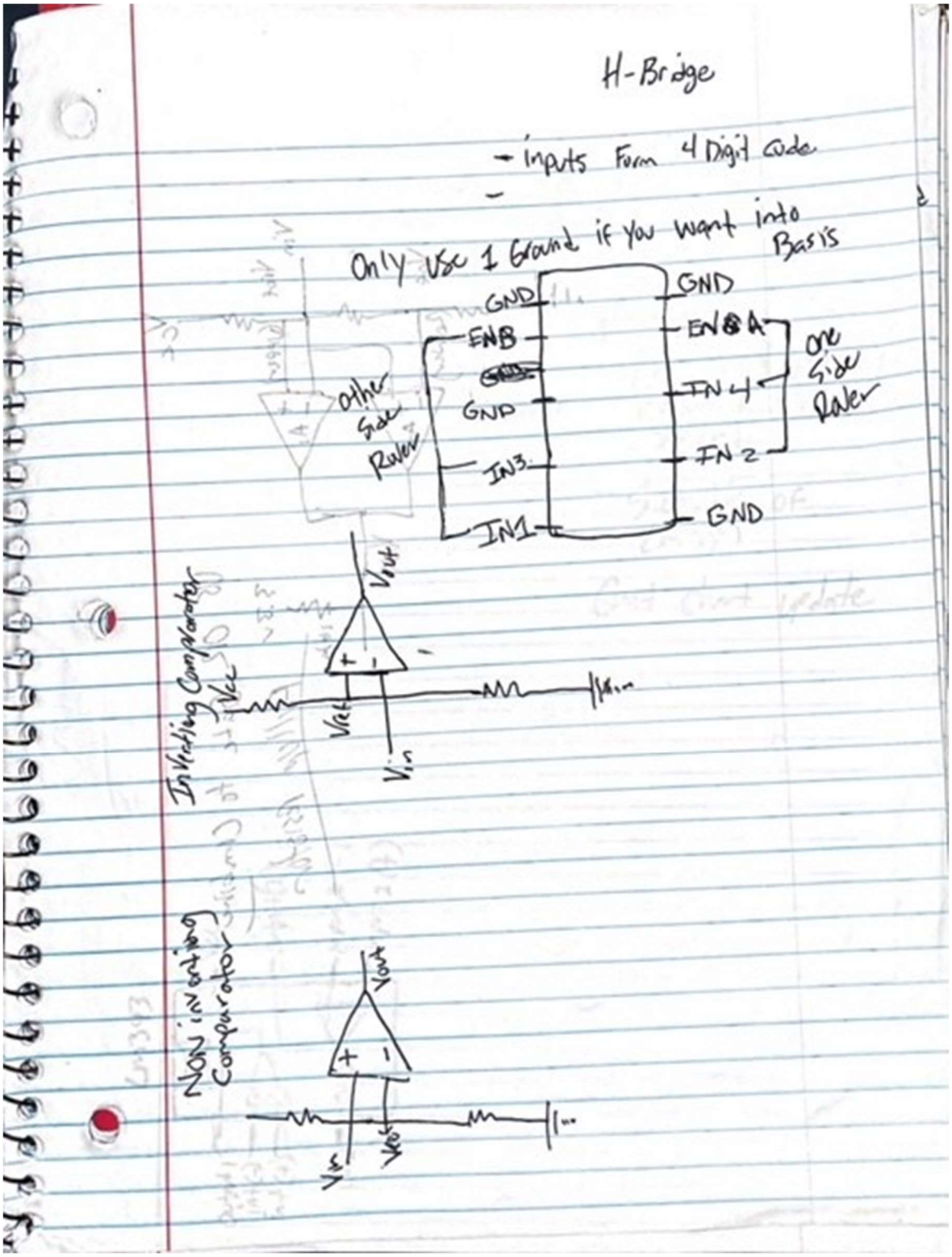
(Operational Amplifier)
Op Amp - intergrated circuit that can Amplify weak Circuits Signals.
Op Amp has 2 input pins 1 output pin
Basic Role is to Amplify output the Voltage difference between 2 inputs pins.

Over Current protection is protection against Currents beyond acceptable current rating of equipment.
Examples - magnetic circuit Breakers / Fuses
Overcurrent Relys

Pulse width modulation (PWM) - Type of digital Signal used for Sophisticated Control Circitry.
It is a modulation technique that generates Variable - width pulses to represent the Amplitude of an analog input signal.
PWM used to control speed of DC motor or the brightness of a lightbulb in a ROV Application.


- Project
* Get Hardware Protection
- How to connect Basys / H-bridge bridge to Rover
- Direction on Basy's
- How to control Rover with Basy's
Solid State Rely for overcurrent protection
Overcurrent Protection
Lm339 Comparator
- 1 volt. uses Voltage Divider
- 2 Comparator circuits
Alol can help.
Send output of Comparator to Basis and tell Basis what to do with that
9596-396-135-7612
- Op Amp Comparator
- Lm339N


$V_o = A_{OL} \times Vid$
Op Amps
Vin
Vref
Vin > Vref = High
Vref > Vin = Low
$V_{REF} = \frac{R_1}{R_1+R_2} V_{CC}$
$V_{ref} = \frac{1k}{1k+1k} 5V$
$V_{ref} = 2.5V$
R_2 = 1K
R_1 = 1K
Comparators
(2V) V_1
(1V) V_2
V_+
V_-
High
V_{out}
+ = Non-inverting input
− = inverting input
1V
2V
Low


H-Bridge
- Inputs Form 4 Digit code
Only Use 1 Ground if you want into Basis
GND
ENB
GND
IN3
IN1
GND
ENA
IN 4
IN 2
GND
One Side Rover
Other Side Rover
Inverting Comparator
Vout
Vcc
Vref
Vin
Non inverting Comparator
Vout
Vin
Vref

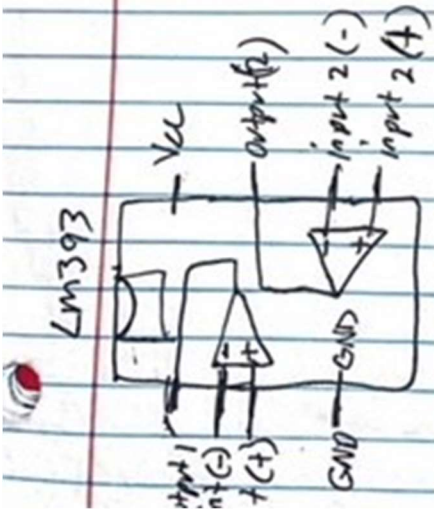
- Hardware Protection for H-bridge and Basys
  * Fuses ?
  * Switches for Breadboard ?

powerpoint

- Circuit Breaker
  - Lm339 * Comparator
  - Verilog modules
  - Simulations

- Shematic of Lm339

Gant chart update

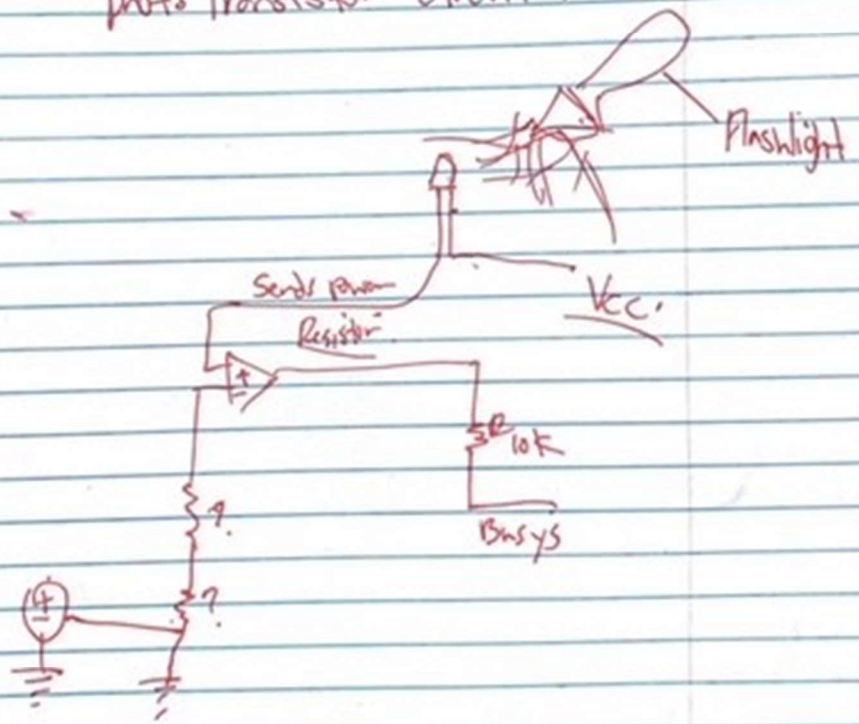


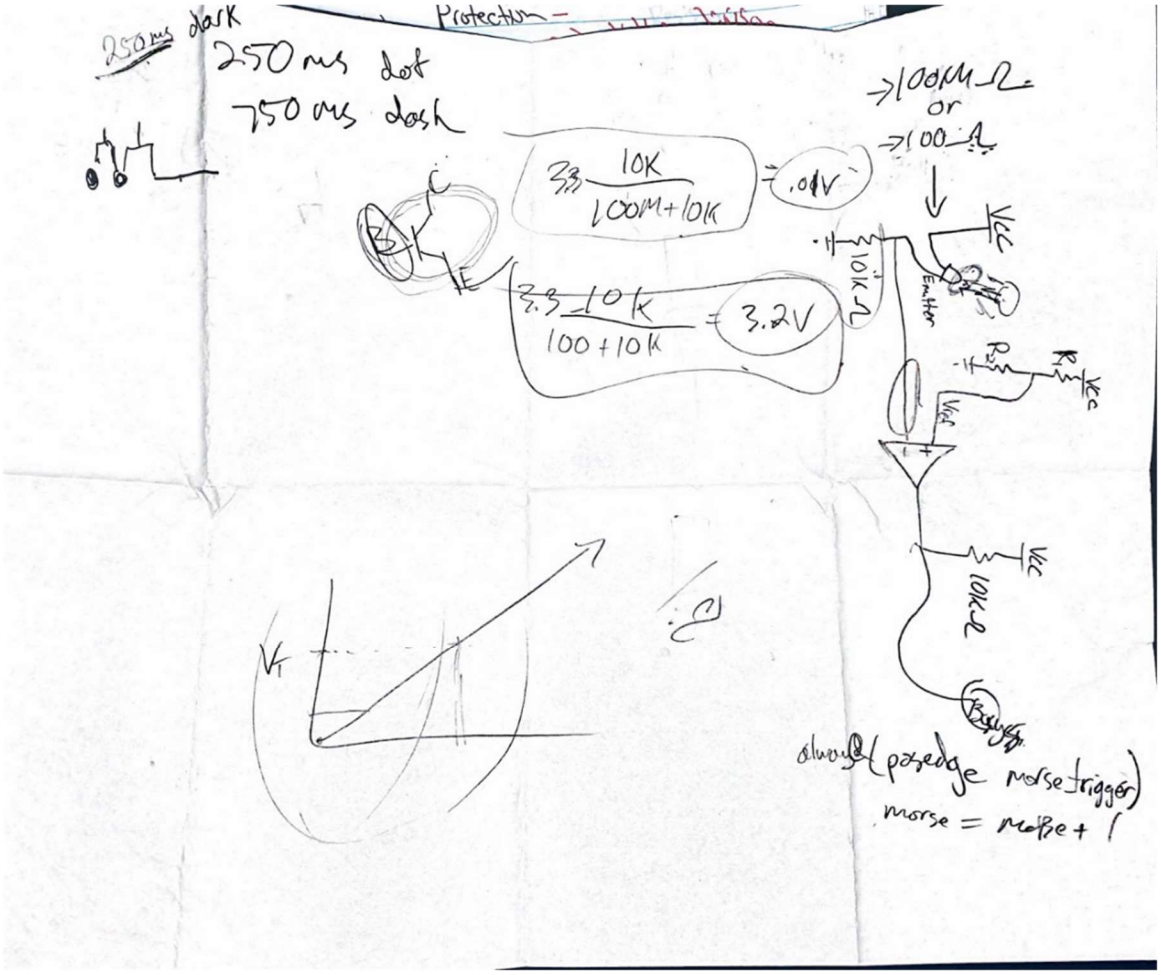
Thursday
2-18-21

Having 1K ohm / 430 Ω Resistors.
With 10K pull up for some Reason I am having Trouble with the Comparator.

Pick up and bring Micro USB to project Lab Tommorrow.

Photo Transistor Circuit?




250 ms dash
250 ms dot
750 ms dash
Protection
→100MΩ
or
→100 μA
10K
100M+10K
≈.04V
10K
100+10K
3.2V
10KΩ
Vcc
Emitter
Vcc
10KΩ
always @(posedge morse_trigger)
morse = morse + 1

Division II

A. Diagrams and Photographs

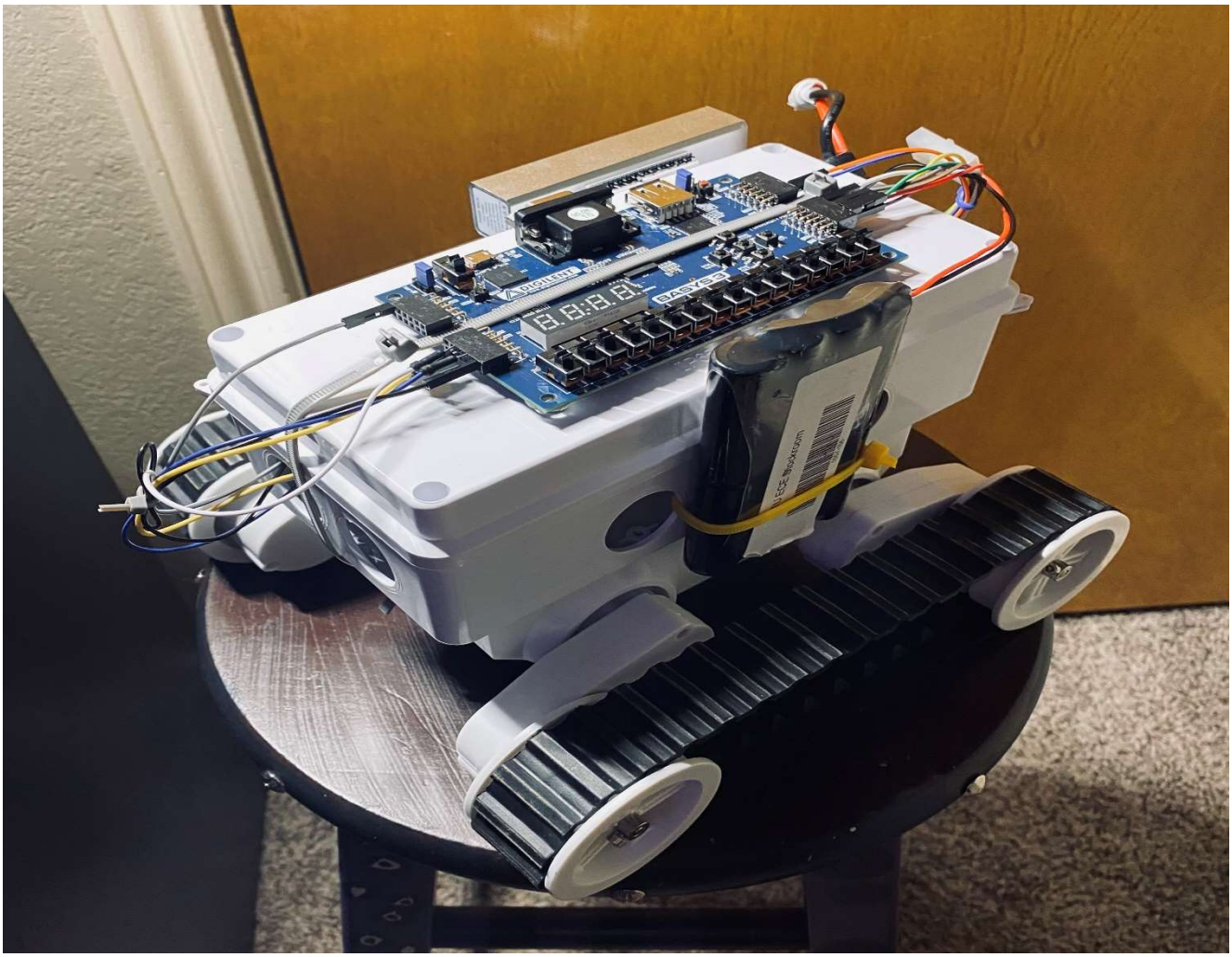


Figure 1: Junction Box Rover Design – Mini-Project

Note. For the main project, the junction box housed the H-bridge circuit and the PCB for overcurrent protection. The TCS3200 color sensor was mounted on the underside and the Sharp IR sensor was mounted on the front of the rover. The color indicating LEDs were mounted on the top of the rover along with the OP593 phototransistor.

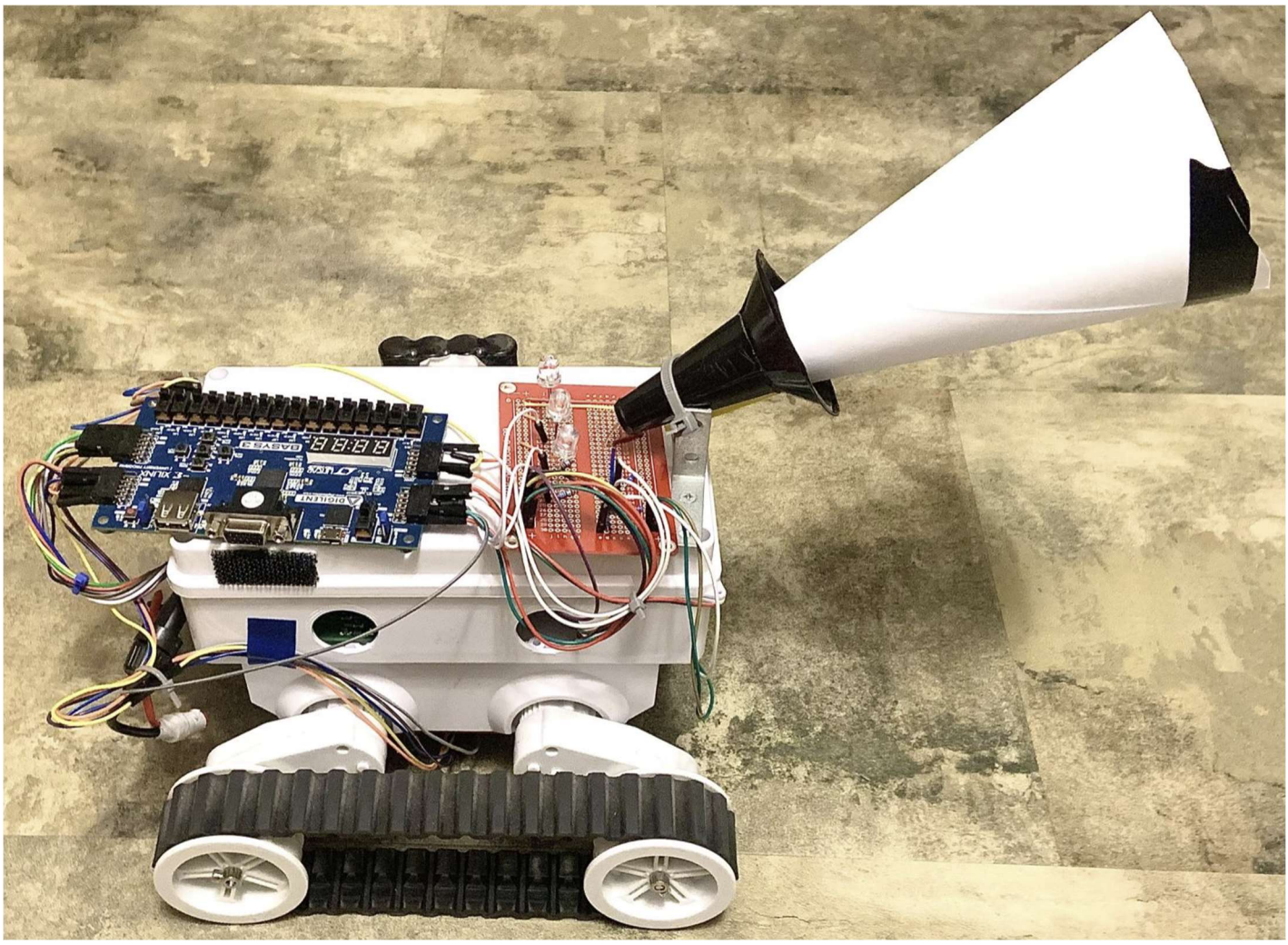


Figure 2: Junction Box Rover Design – Main-Project

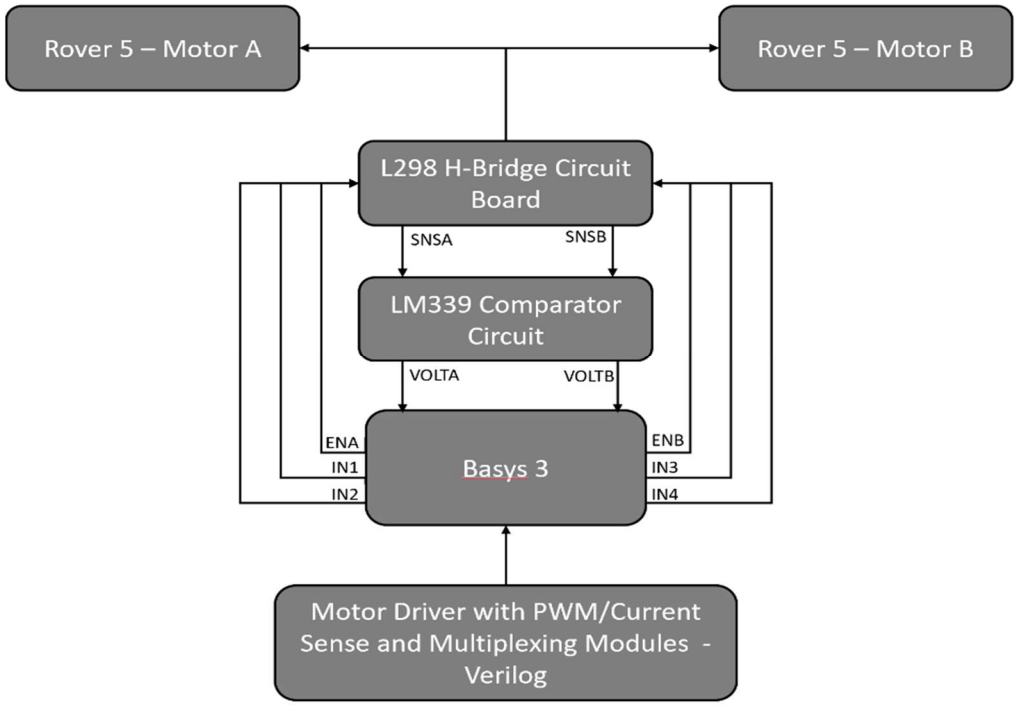


Figure 3: Diagram of Mini-Project System

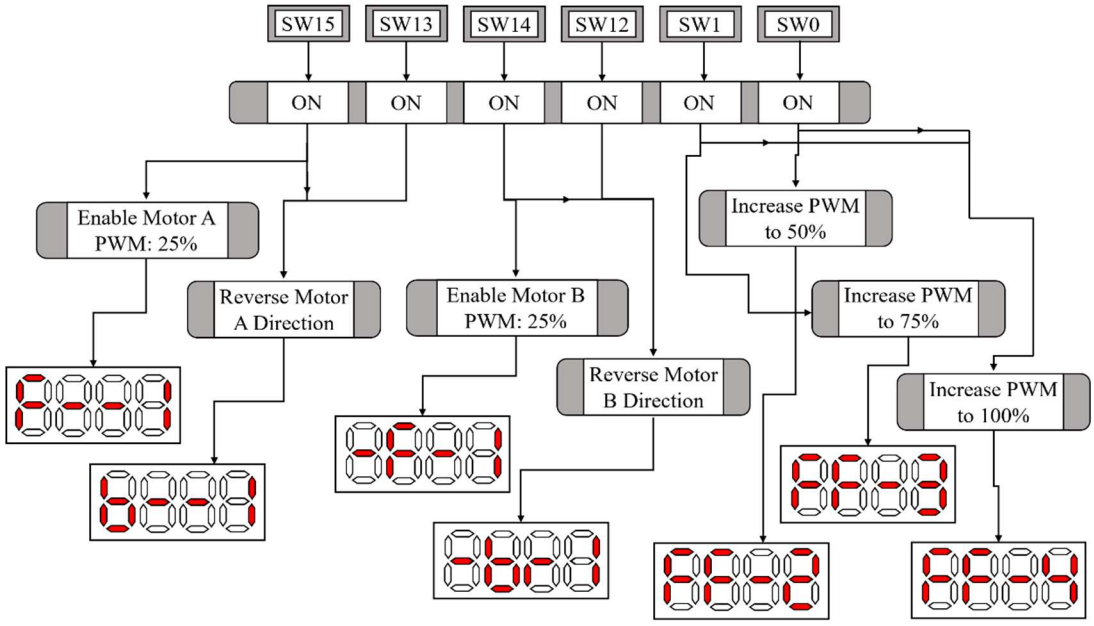


Figure 4: System I/O Diagram – Mini-Project

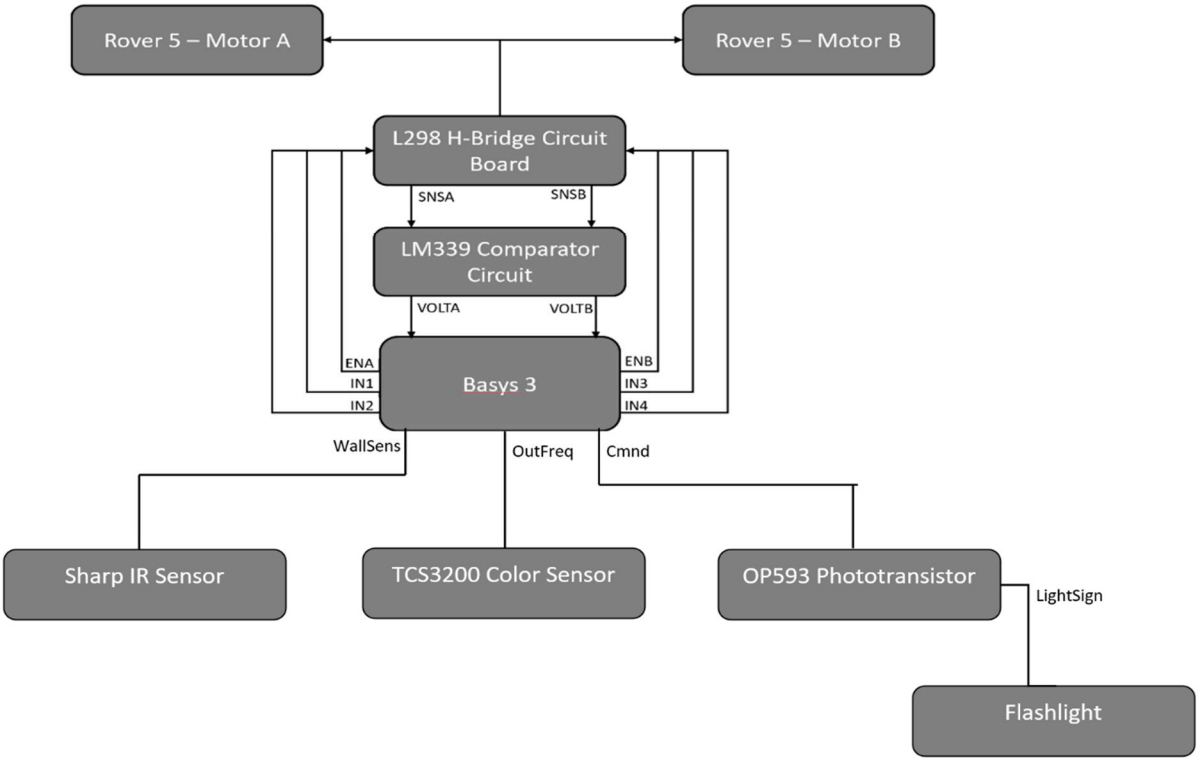


Figure 5: Diagram of Main-Project System

B. Datasheets

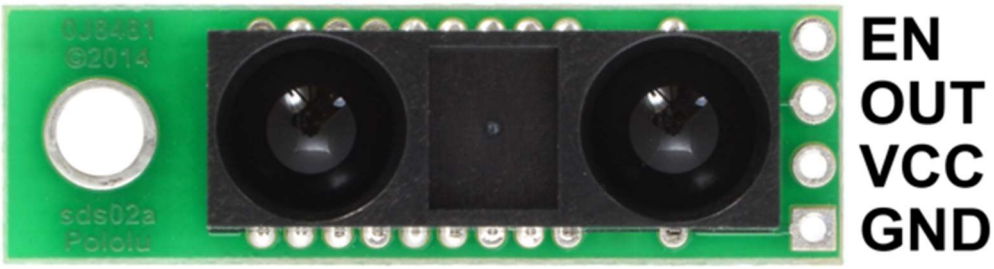


Figure 6: SHARP GP2Y0A60SZ0F Ports [5]

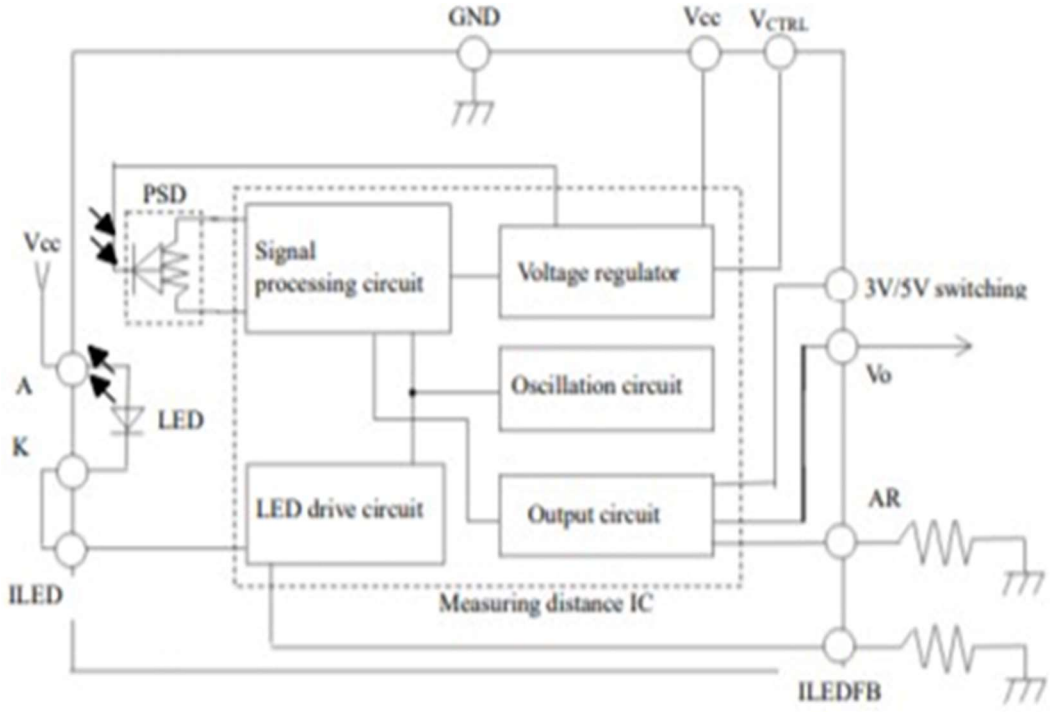


Figure 7: SHARP GP2Y0A60SZ0F Schematic [5]

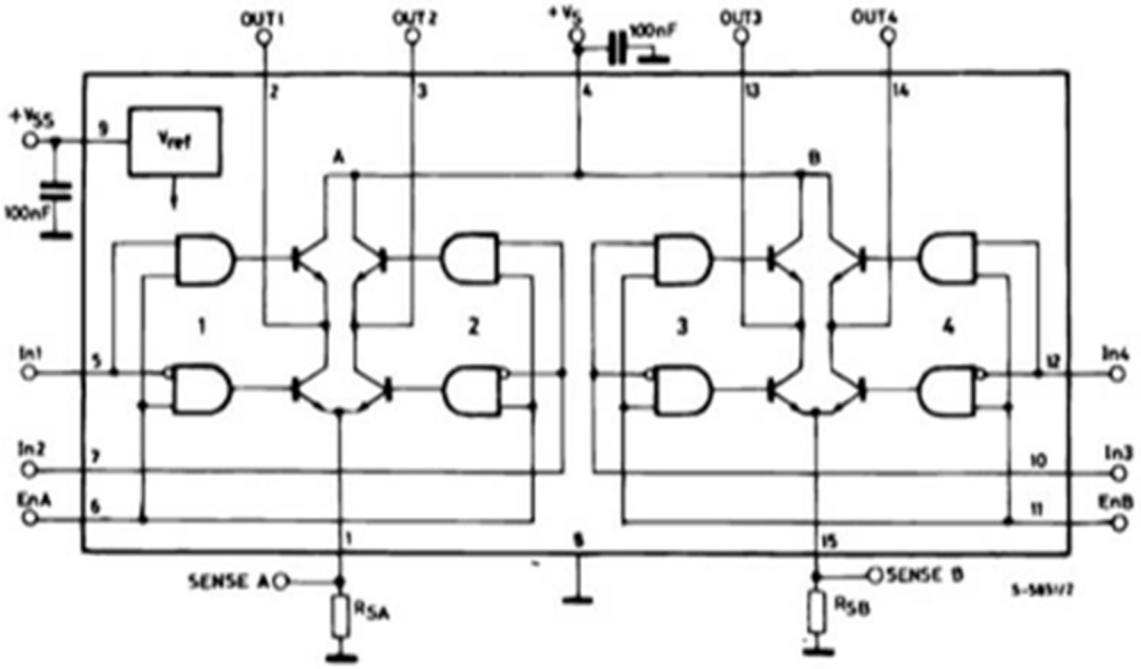


Figure 8: L298 H-Bridge Inputs [2]

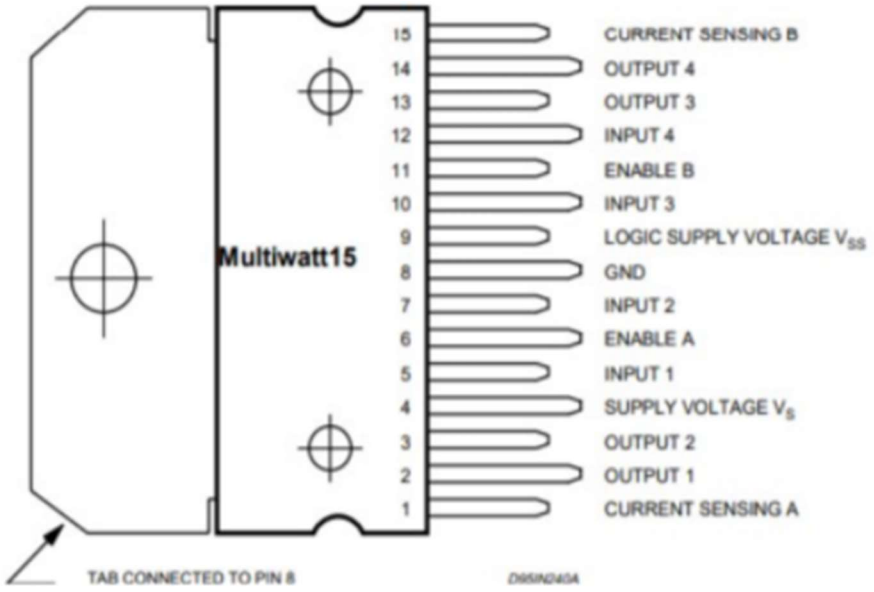


Figure 9: L298 H-Bridge Schematic [2]

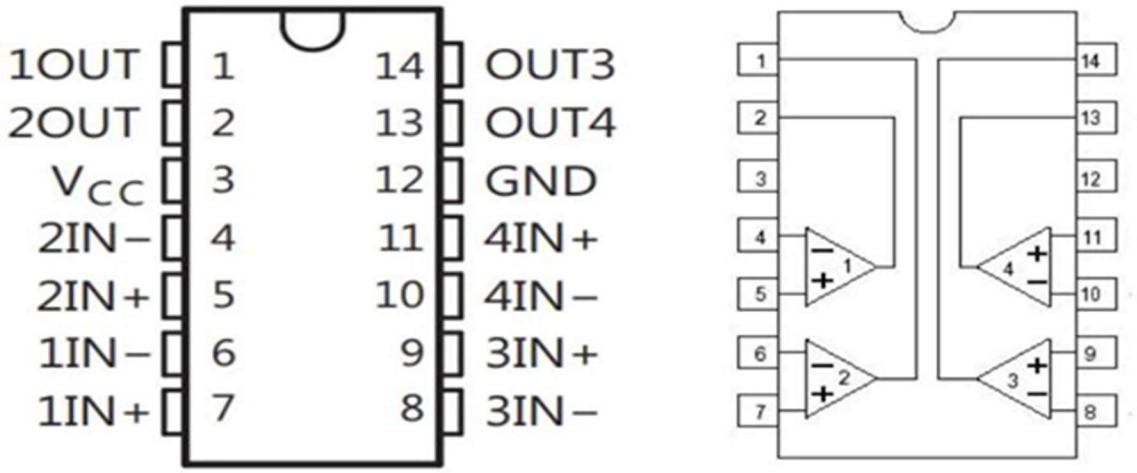


Figure 10: LM339N Comparator Pins [3]

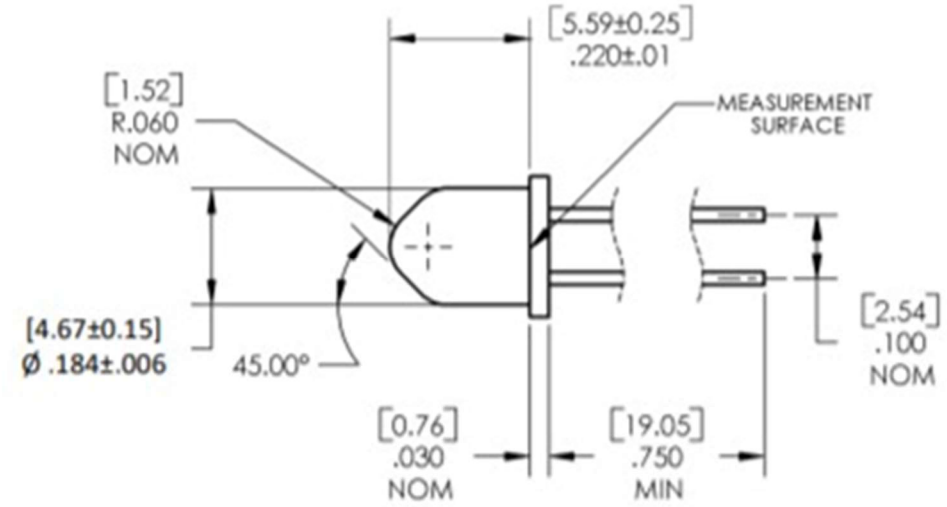


Figure 11: OP593 Viewing Angle [4]

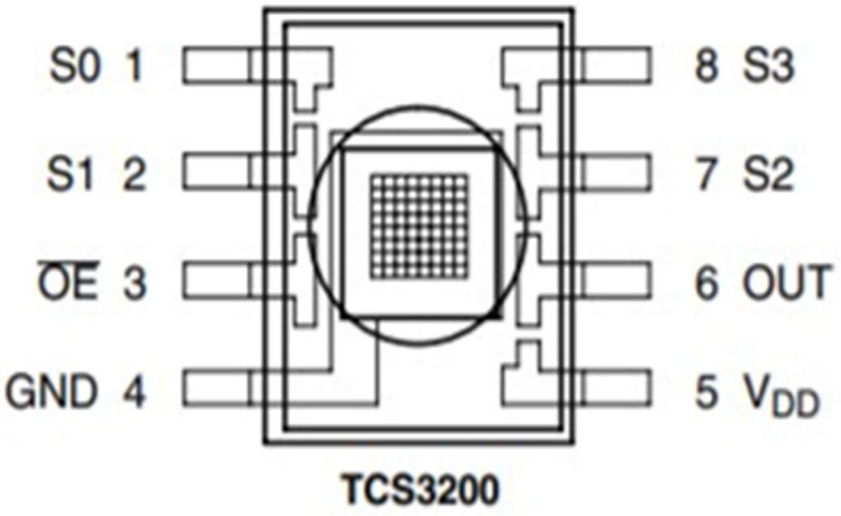


Figure 12: TCS3200 Schematic [6]

S2	S3	PHOTODIODE TYPE
L	L	Red
L	H	Blue
H	L	Clear (no filter)
H	H	Green

Figure 13: TCS3200 Photodiode Selector [6]

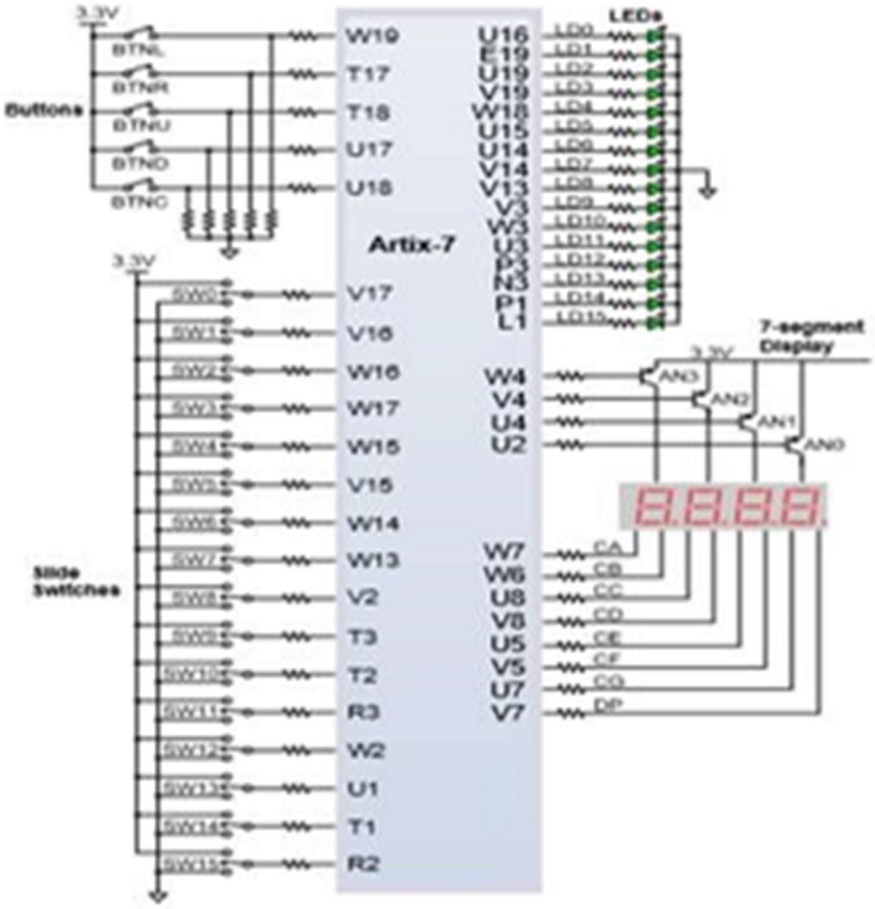


Figure 14: Basys3 Schematic- Component Layout [1]

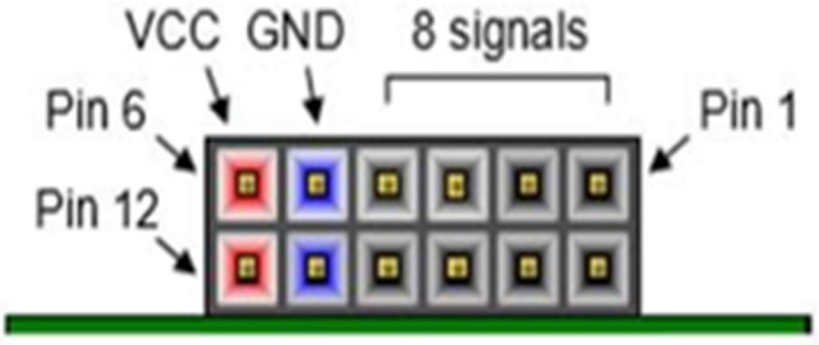


Figure 15: Basys3 Standard PMOD Port [1]

C. References

[1] Basys 3 FPGA Board Reference Manual. Digilent, Pullman, WA, United States, 2019, pp. 2. [Online]. Available: basys3_rm.pdf (digilentinc.com)

[2] L298 Dual Full-Bridge Driver Datasheet. STMicroelectronics, Geneva, Switzerland, 2000. [Online]. Available: Dual full-bridge driver (st.com)

[3] LM339, LM239, LM139, LM2901 Quad Differential Comparators Datasheet. Texas Instruments, Dallas, TX, United States, 2020. [Online]. Available: LM339, LM239, LM139, LM2901 Quad Differential Comparators datasheet (Rev. U)

[4] NPN Plastic Silicon Phototransistor Datasheet. TT Electronics, Woking, United Kingdom, 2016. [Online]: Available: OP593-598-798.pdf (ttelectronics.com)

[5] Sharp GP2Y0A60SZ0F/GP2Y0A60SZLF Datasheet. Sharp, Montvale, NJ, United States, 2020. [Online]: Available: pc123xnnsz_j (socle-tech.com).

[6] TCS3200, TCS3210 PROGRAMMABLE COLOR LIGHT-TO-FREQUENCY CONVERTER. TAOS, Plano, TX, United States, 2009. [Online]: Available: Document: (mouser.com)

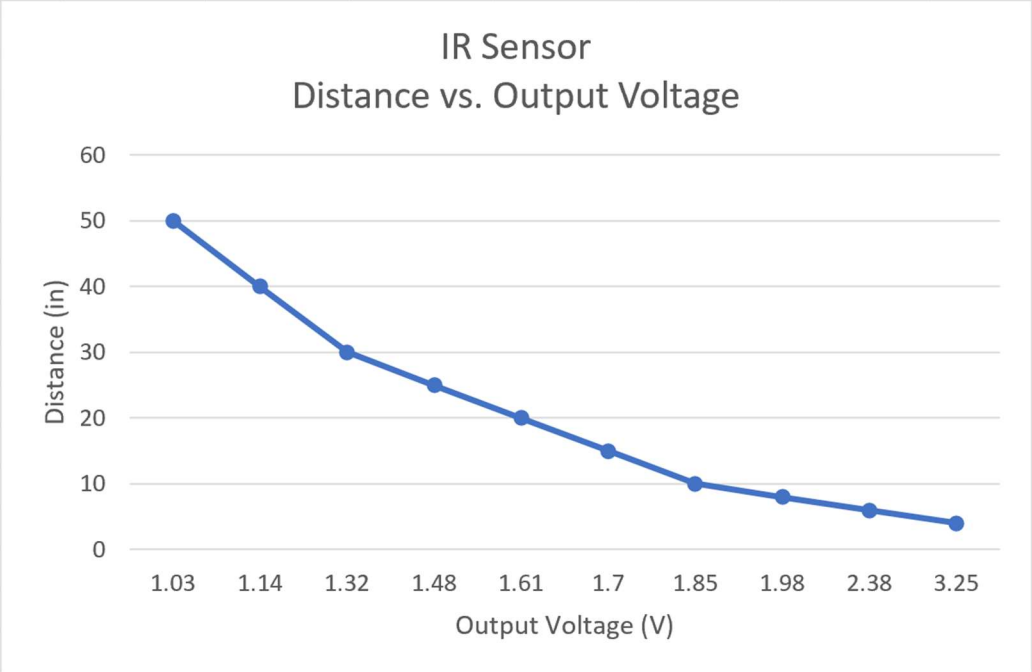
Division III

A. Sharp IR Sensor Experimental Data

Note. The IR sensor was tested by supplying the sensor with 3.3 V and moving a notebook closer and further away from the sensor. The distance was measured using a yard stick. These values were used in determining the JXADC value to indicate to the rover to turn.

Table i: IR Sensor Output Voltages

Distance (in)	50	40	30	25	20	15	10	8	6	4
Output Voltage (V)	1.03	1.14	1.32	1.48	1.61	1.7	1.85	1.98	2.38	3.25



Graph i: Distance vs. Output Voltage – IR Sensor

B. Python Vector Simulation

Note. A vector simulation script was developed to test 3, 90° turns off three walls and 1 random degree turn of the last wall. The simulation proved a fatal pattern and did not reach the center of the playing field as originally thought. Ideas were discussed, but eventually a 160° turn was decided on.

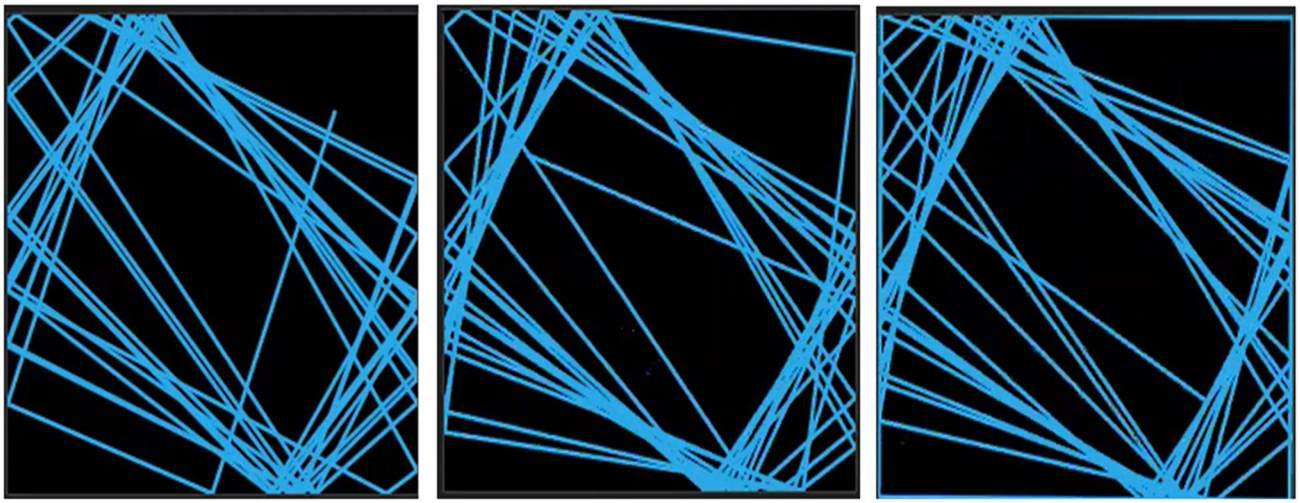


Figure 16: Python Vector Simulation from Different Starting Points

```
startX = .2
startY = .3
currentX = 1
currentY = .6

ctx.move_to(startX, startY)

prevX = startX
prevY = startY
x = 0
counter = 0
counterAngle = 0
angle = .1
choseAngle = 0

def paramCheck(foundCoord):
    if foundCoord < 0:
        return 1 - abs(foundCoord)
    elif foundCoord > 1:
        print("HERE: ", foundCoord)
        return abs(1 - foundCoord)
    else:
        return foundCoord


for x in range(50):
    print(x, ":", prevX, prevY, currentX, currentY)
```

Figure 17: Python Vector Simulation Code Snippet

C. TCS3200 Color Sensor Experimental Data

Table ii: Color Sensor Frequency Findings

Arduino Color Sensor Frequency Readings in Microseconds			
Color	Photodiode Type	Upper Bound	Lower Bound
Red	Red	58	46
	Blue	144	129
	Green	191	174
Blue	Red	227	192
	Blue	151	125
	Green	108	80
Green	Red	223	174
	Blue	53	34
	Green	110	80

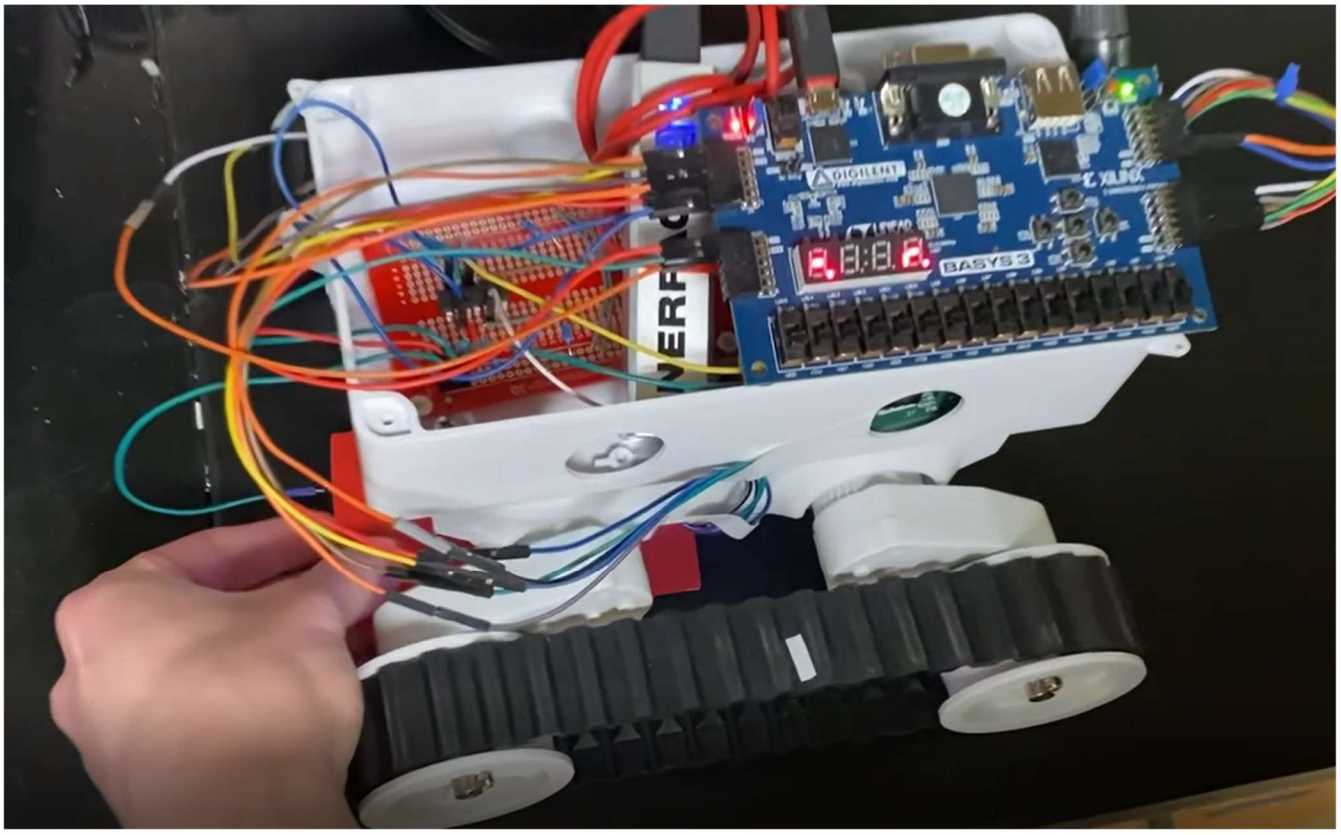


Figure 18: Color Sensor Reading and Displaying Red

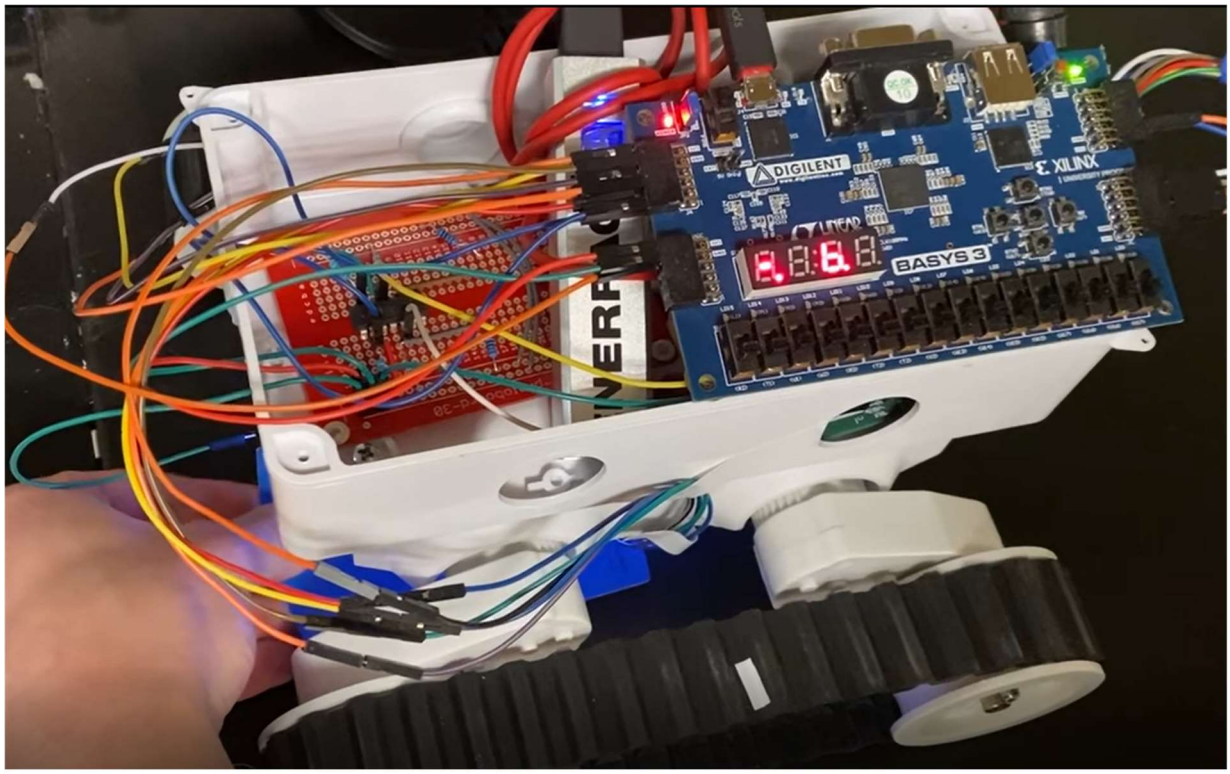


Figure 19: Color Sensor Reading and Displaying Blue

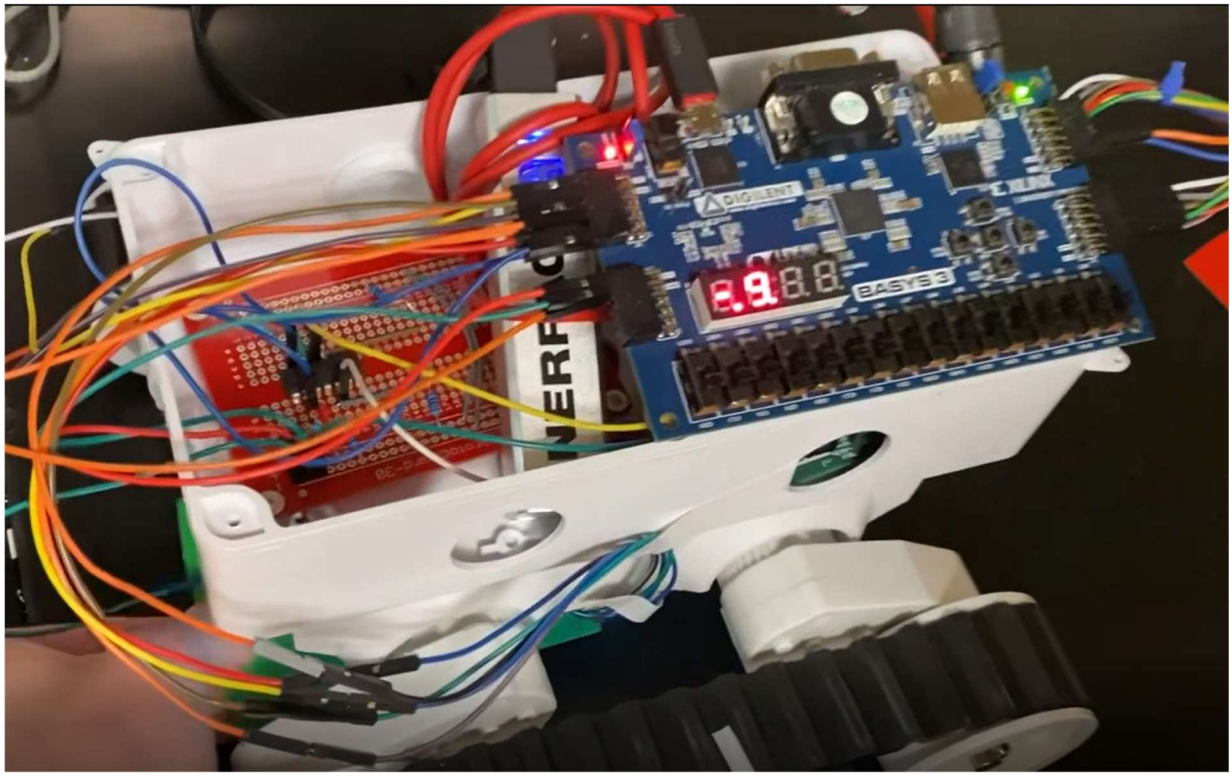


Figure 20: Color Sensor Reading and Displaying Green

D. OP593 Phototransistor Experimental Data

Table iii: OP593 Phototransistor Emitter Voltage

Distance (ft)	>6	<6
Output Voltage (V)	0	3.3

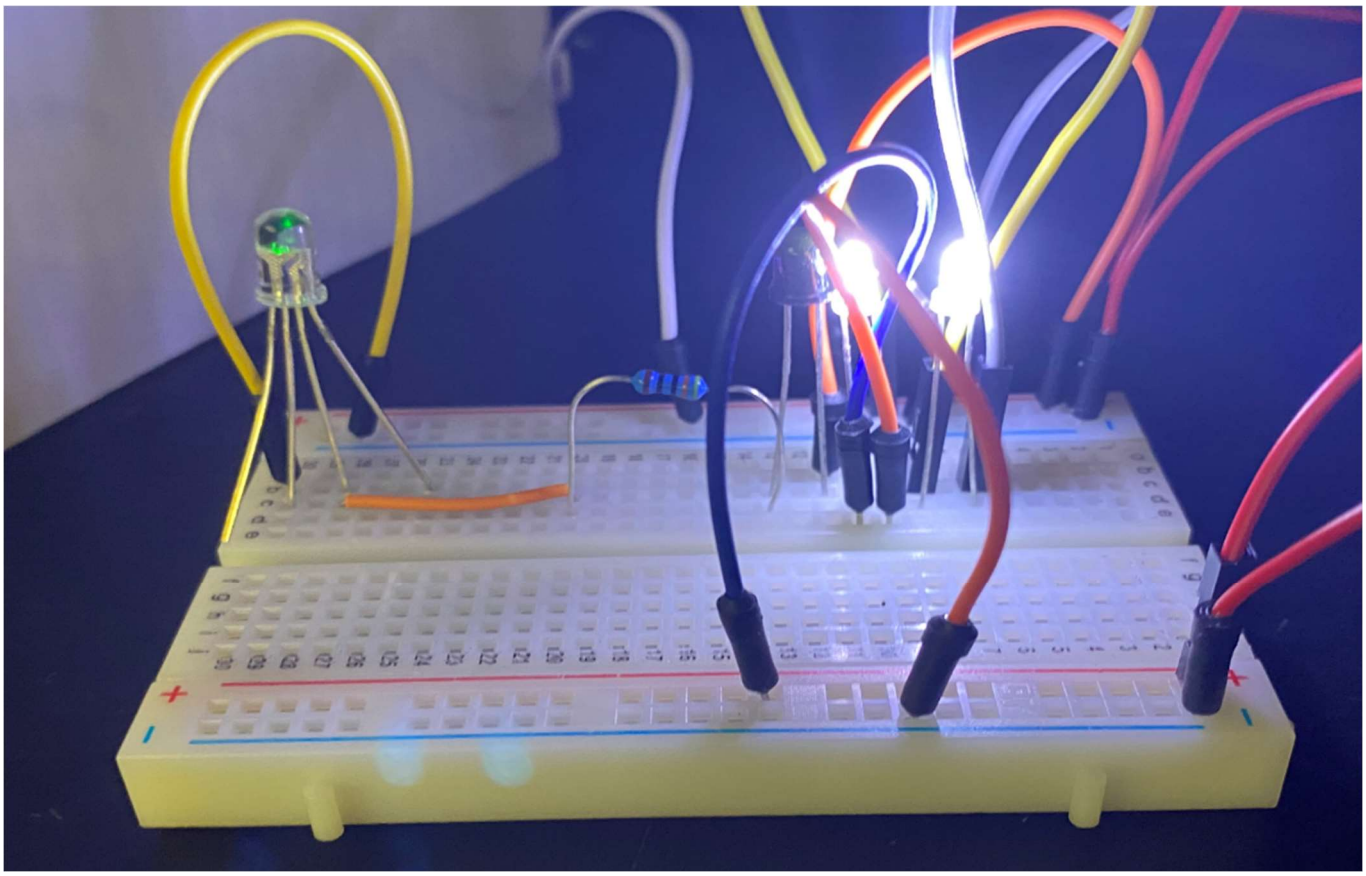


Figure 21: Inducing Current in Phototransistor with LEDs for Testing

E. PWM Oscilloscope Images

Note. The PWM of the Rover 5 motors was tested using the oscilloscope. A final PWM of 50% was decided upon when the rover moved forward and 75% when turning because of a faulty motor on the left track.

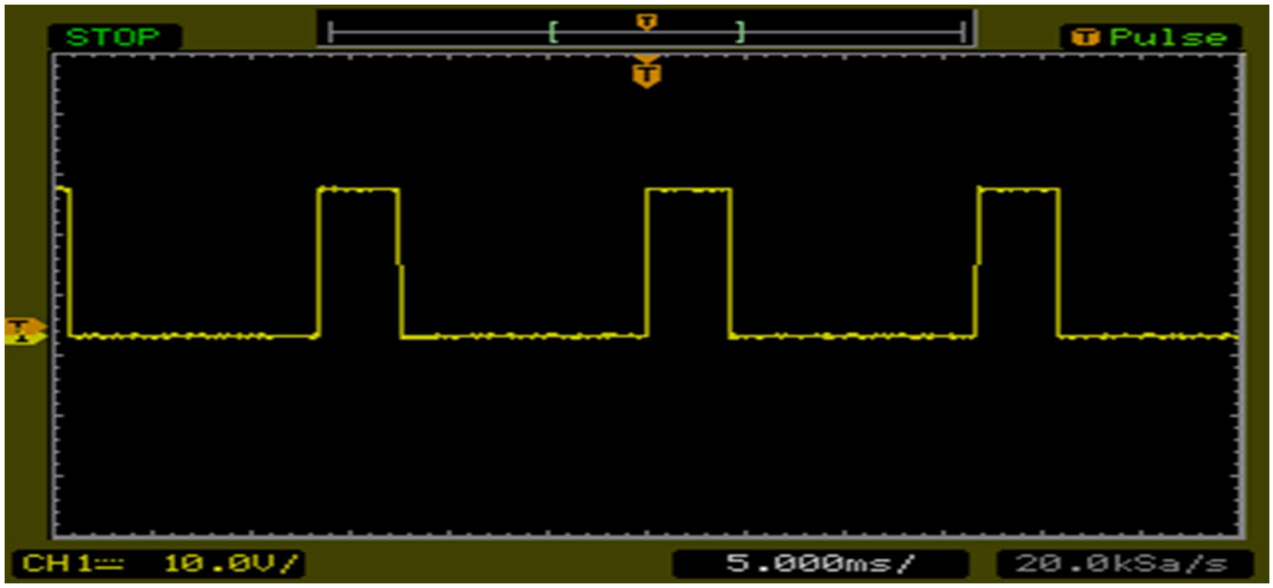


Figure 22: 25% PWM Oscilloscope Testing

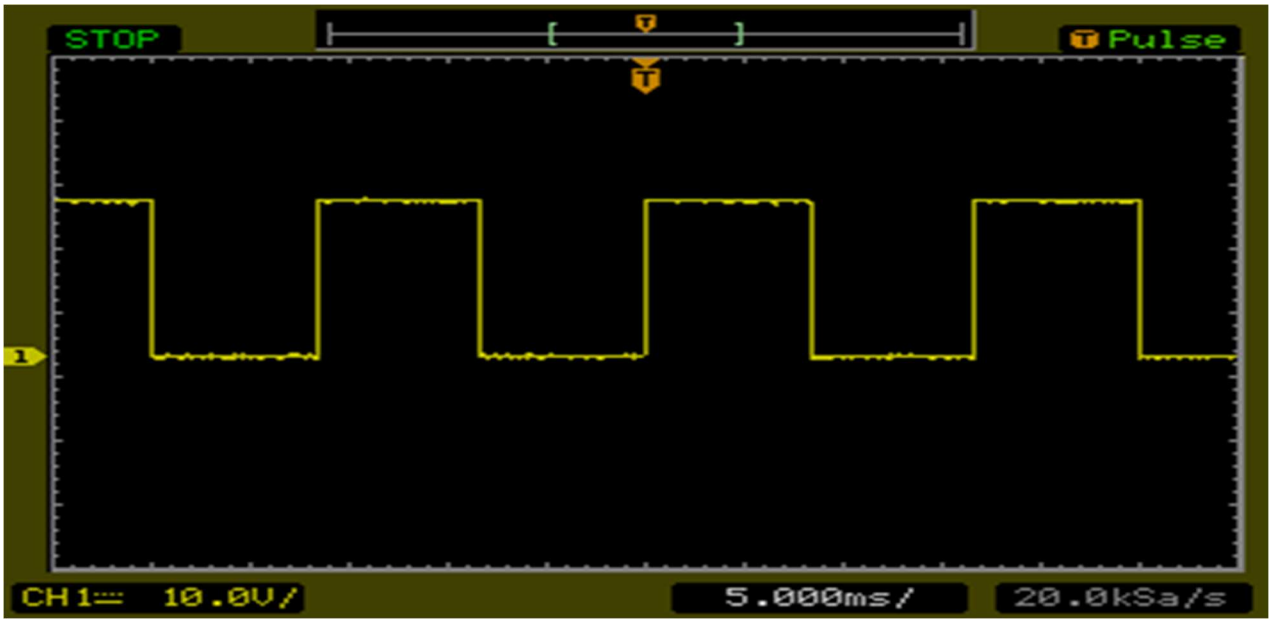


Figure 23: 50% PWM Oscilloscope Testing

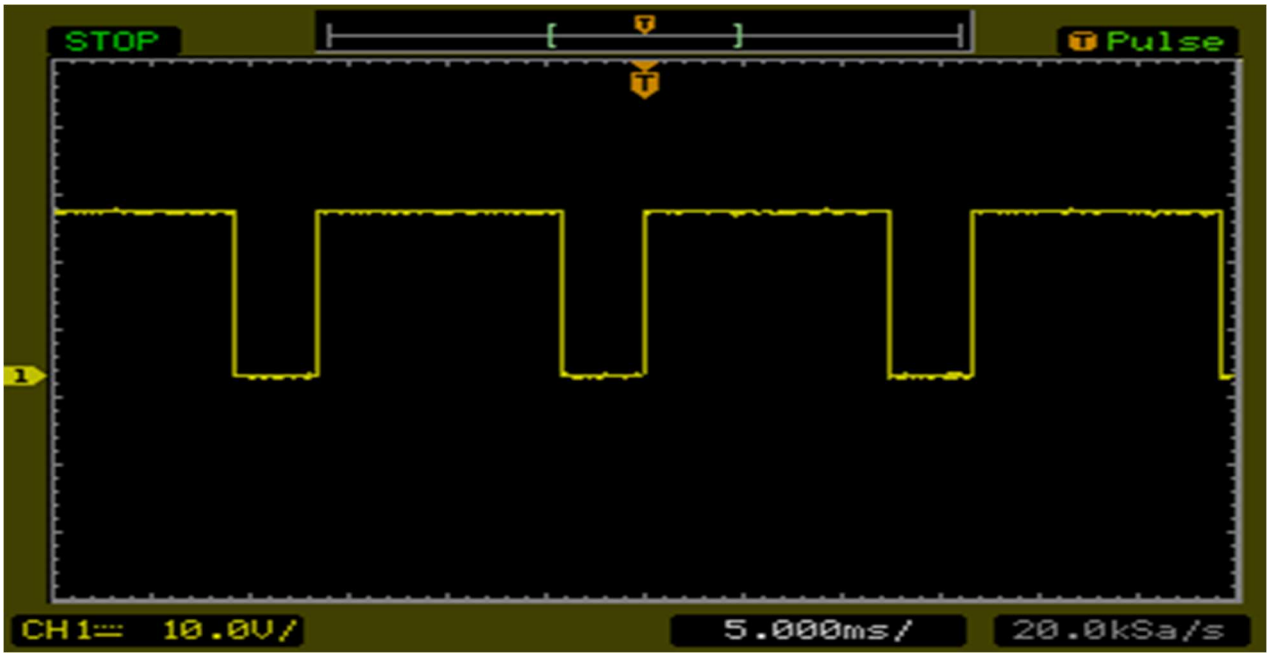


Figure 24: 75% PWM Oscilloscope Testing

F. Overcurrent Protection Oscilloscope Images

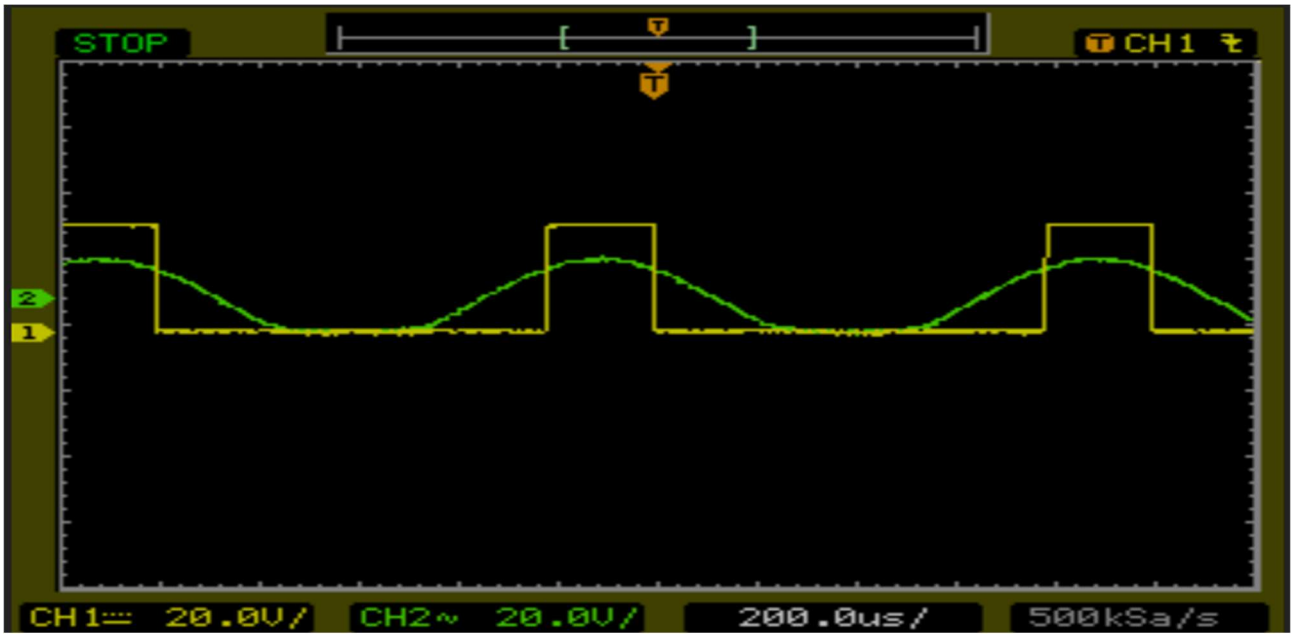


Figure 25: PCB Design Functionality – Oscilloscope Testing

Division IV

A. Parts List

Product Number	Cost	Quantity	Supplier	Purchase Date	Total	Description	Material Cost Total
424-BASYS3	$ 149.00	1	stockroom/mouser	SUPPLIED 2/1/21	$ 149.00	Basys 3 FPGA board	$ 290.61
594-K104K15X7RF53H5	$ 0.11	4	stockroom/mouser	SUPPLIED 1/27/21	$ 0.44	.1uF Filter Capacitor	
652-4607X-1LF-1K	$ 0.63	2	stockroom/mouser	SUPPLIED 1/27/21	$ 1.26	1Kohm bussed resistor	
696-SLX-LX3044HC		6	stockroom/mouser	SUPPLIED 1/27/21		3 mm led	
551-PS2501-4-A	$ 2.11	1	stockroom/mouser	SUPPLIED 1/27/21	$ 2.11	4 channel octocoupler	
551-PS2501A-1-A	$ 0.50	2	stockroom/mouser	SUPPLIED 1/27/21	$ 1.00	1 channel octocoupler	
660-RK73H2BTTD3000F	$ 0.10	6	stockroom/mouser	SUPPLIED 1/27/21	$ 0.60	300 ohm resistor	
571-5103308-1	$ 1.16	1	stockroom/mouser	SUPPLIED 1/27/21	$ 1.16	10 pin header	
538-42375-1923	$ 0.20	1	stockroom/mouser	SUPPLIED 1/27/21	$ 0.20	3 pin header	
511-L298	$ 4.86	1	stockroom/mouser	SUPPLIED 1/27/21	$ 4.86	L298 h bridge	
625-SB2H100-E3/73	$ 0.42	8	stockroom/mouser	SUPPLIED 1/27/21	$ 3.36	high vf schottky diodes	
750-SB540E-G	$ 0.44	1	stockroom/mouser	SUPPLIED 1/27/21	$ 0.44	low VF schottky Rectifier/Diodes	
594-AC05W1R000J	$ 0.51	2	stockroom/mouser	SUPPLIED 1/27/21	$ 1.02	schunt resistor	
647-UVZ1E102MPD	$ 0.58	1	stockroom/mouser	SUPPLIED 1/27/21	$ 0.58	25v 1000uF Filter Capacitor	
926-LM7805CT/NOPB	$ 1.54	1	stockroom/mouser	SUPPLIED 1/27/21	$ 1.54	Linear Voltage Regulators 5 Volt Reg	
532-566010B03400G	$ 1.31	1	stockroom/mouser	SUPPLIED 1/27/21	$ 1.31	multiwatt-15 heatsink	
532-574102B00	$ 0.38	1	stockroom/mouser	SUPPLIED 1/27/21	$ 0.38	T0-220 heatsink	
651-1935161	$ 0.41	1	stockroom/mouser	SUPPLIED 1/27/21	$ 0.41	5mm screw terminal block	
LM339N	$ 0.46	2	stockroom/mouser	SUPPLIED 2/5/21	$ 0.92	LM339 Comparator	
ROB0055-E	$ 51.25	1	stockroom/mouser	SUPPLIED 2/4/21	$ 51.25	Rover 5 Chassis	
Lm201709071387	$ 14.06	1	amazon	SUPPLIED 2/12/21	$ 14.06	LeMotech Junction Box	
GMA-2A	$ 4.36	1	home depot	SUPPLIED 2/12/21	$ 4.36	2 A Fuse	
Sharp GPZY0A60SZLF	$ 10.00	2	Stockroom/pololu	SUPPLIED 2/27/21	$ 20.00	ANALOG DISTANCE FINDER	
YL-64 TCS3200	$ 5.00	1	Stockroom/mouser	SUPPLIED 2/27/21	$ 5.00	COLOR SENSOR	
828-OP593A	$ 0.71	1	Stockroom/mouser	SUPPLIED 2/27/21	$ 0.71	OP593 PHOTOTRANSISTOR	
593-VAOL-5LWY4	$ 0.78	2	Stockroom/mouser	SUPPLIED 3/4/21	$ 1.56	White LED	
997-L1HX507020	$ 1.04	2	Stockroom/mouser	SUPPLIED 3/22/21	$ 2.08	White-White Lumileds High-Power LED	
PRT-12702	$ 3.00	3	Stockroom/sparkfun	SUPPLIED 4/10/21	$ 9.00	Mini Solderable Breadboard	
PRT-12070	$ 6.00	2	Stockroom	SUPPLIED 4/10/21	$ 12.00	Medium Solderable Breadboard	
	$ 0.40	5	JLCPCB	SUPPLIED 4/8/21	$ 2.00	PCB Design	

Figure 26: Material Costs Sheet

B. Complete Circuit Schematics

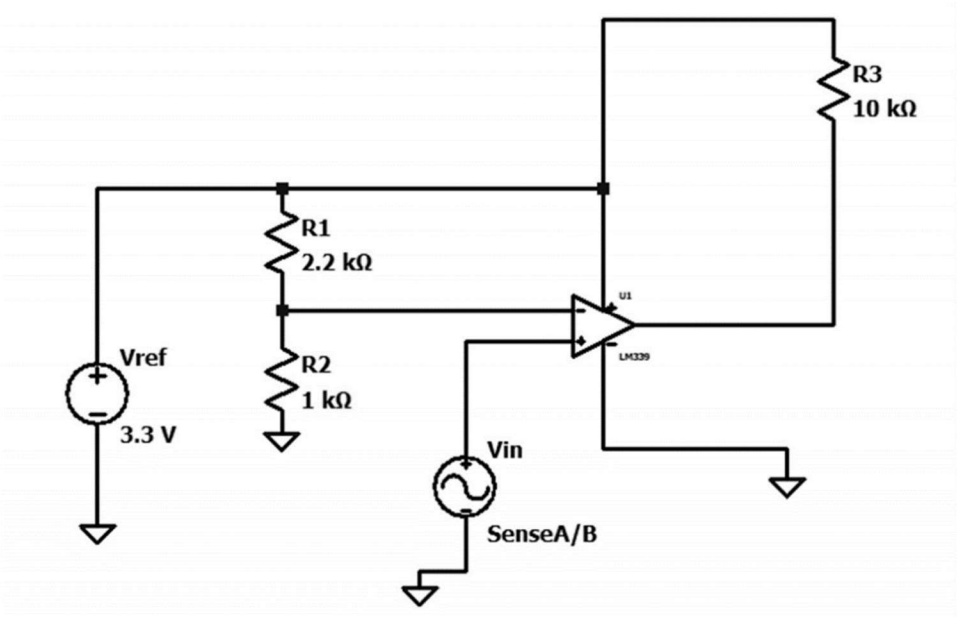


Figure 27: LM339 Comparator – Overcurrent Protection Circuit

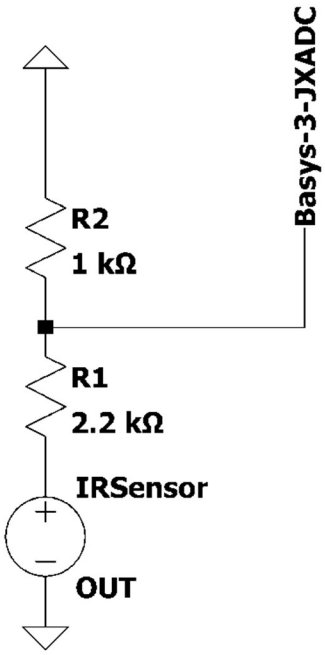


Figure 28: Wall Avoidance – Voltage Divider

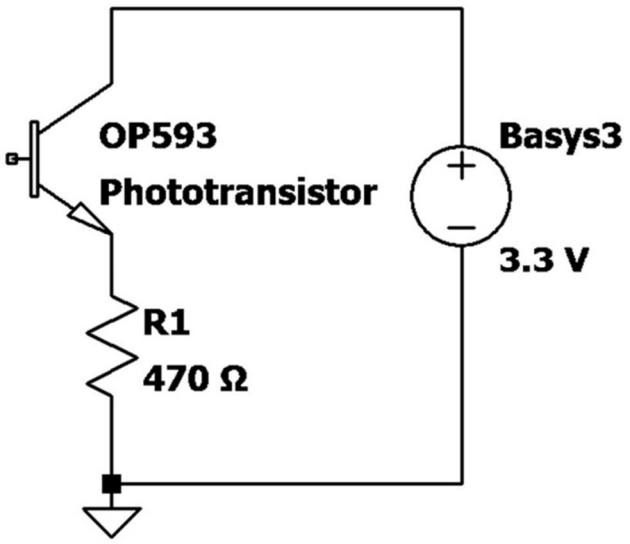


Figure 29: OP593 Phototransistor – Morse Code Receiver Circuit

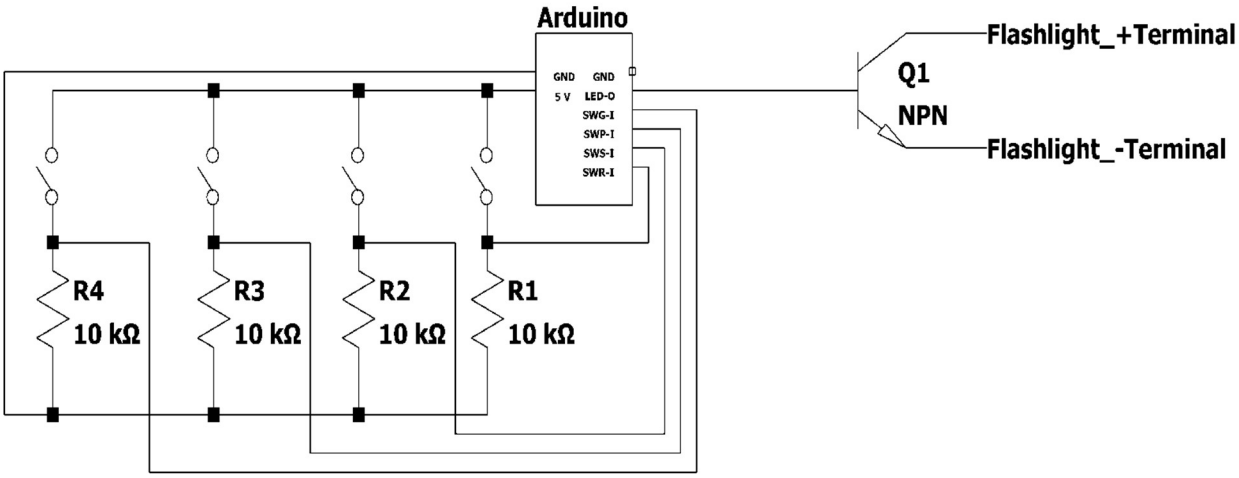


Figure 30: Strobe Beacon Circuit

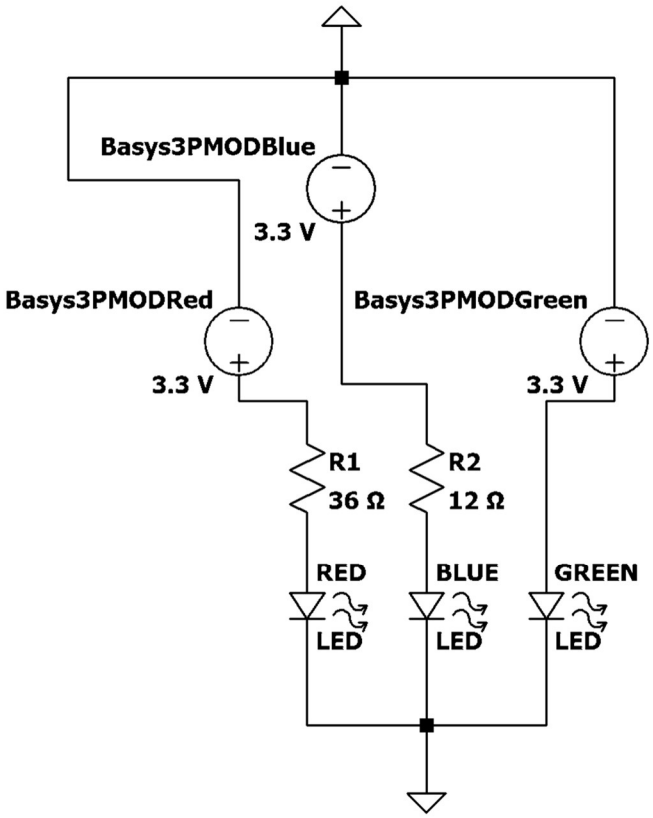


Figure 31: Visual Indicator LEDs – Color Sensing

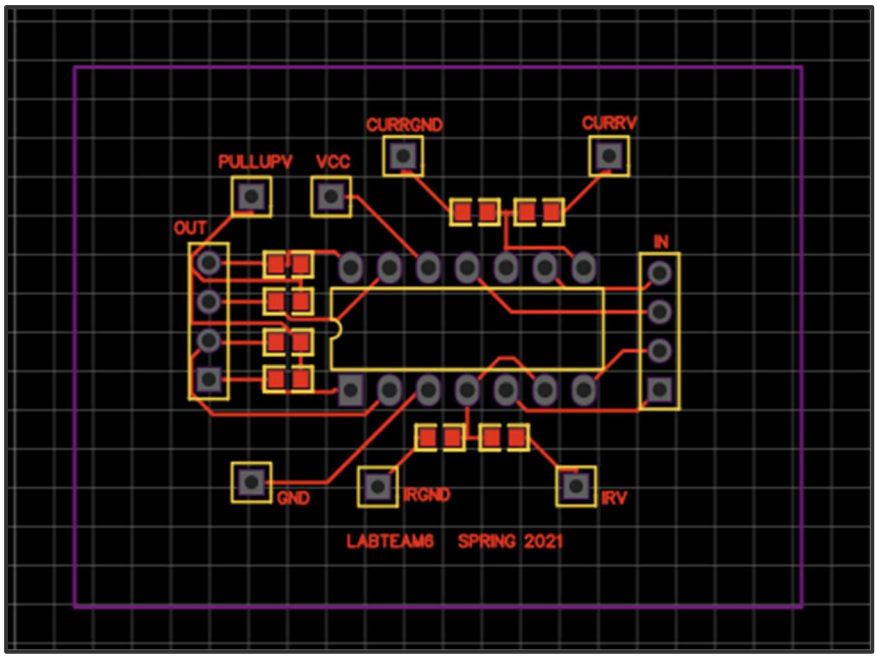


Figure 32: PCB Design

C. Finalized Hardware and PCB

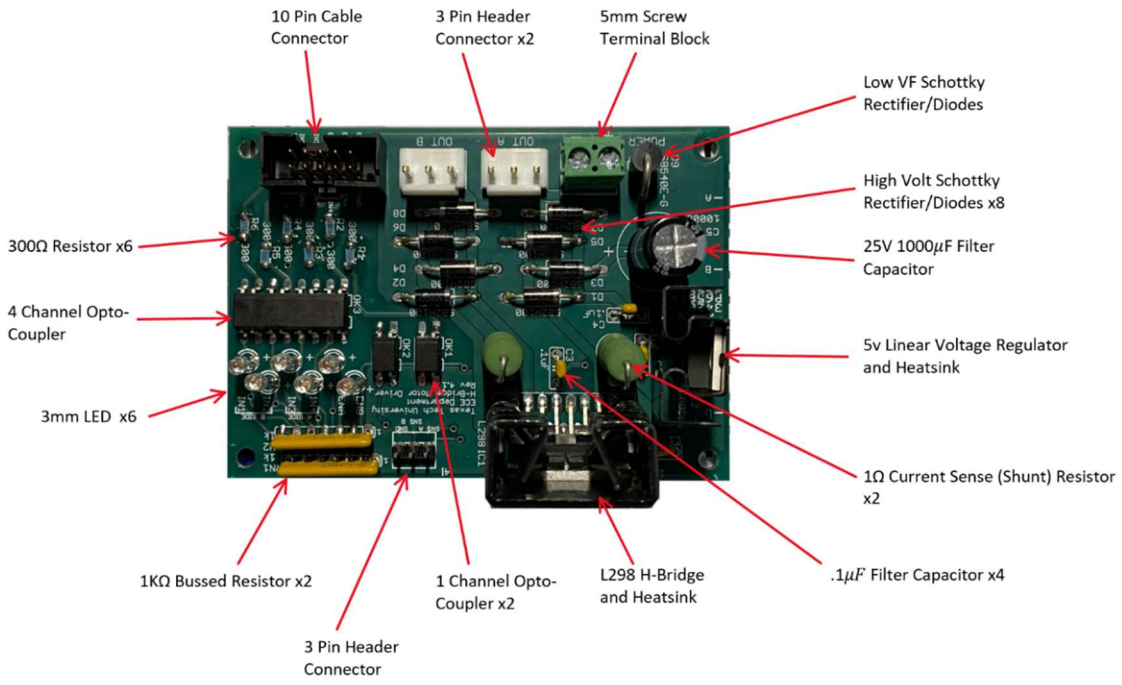


Figure 33: H-Bridge Circuit Board and Components

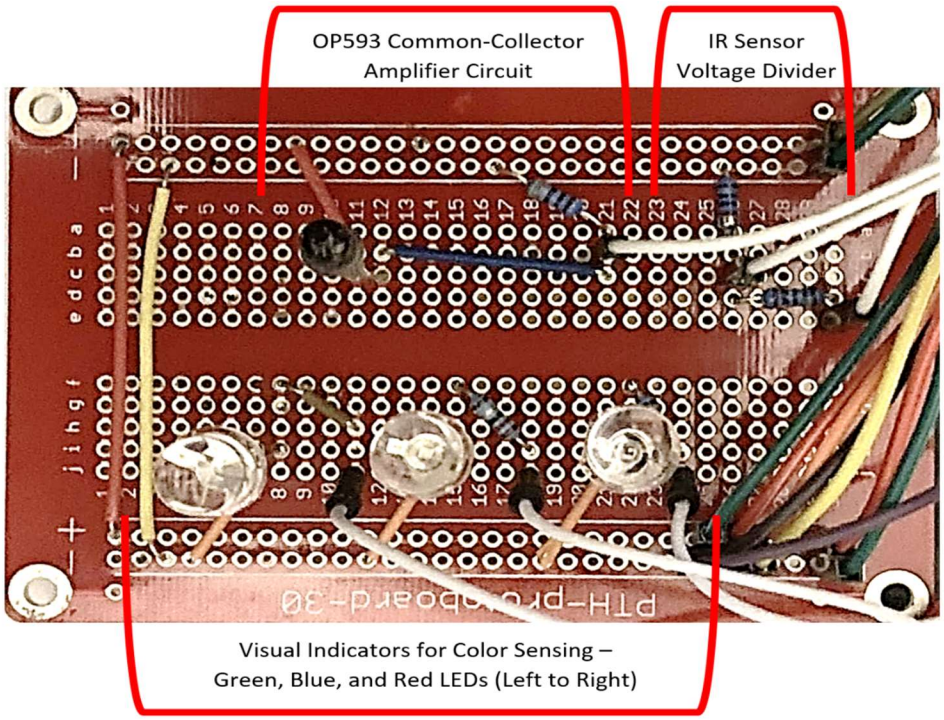


Figure 34: Wall Avoidance, Morse Code, and Visual Indicator Circuit Board

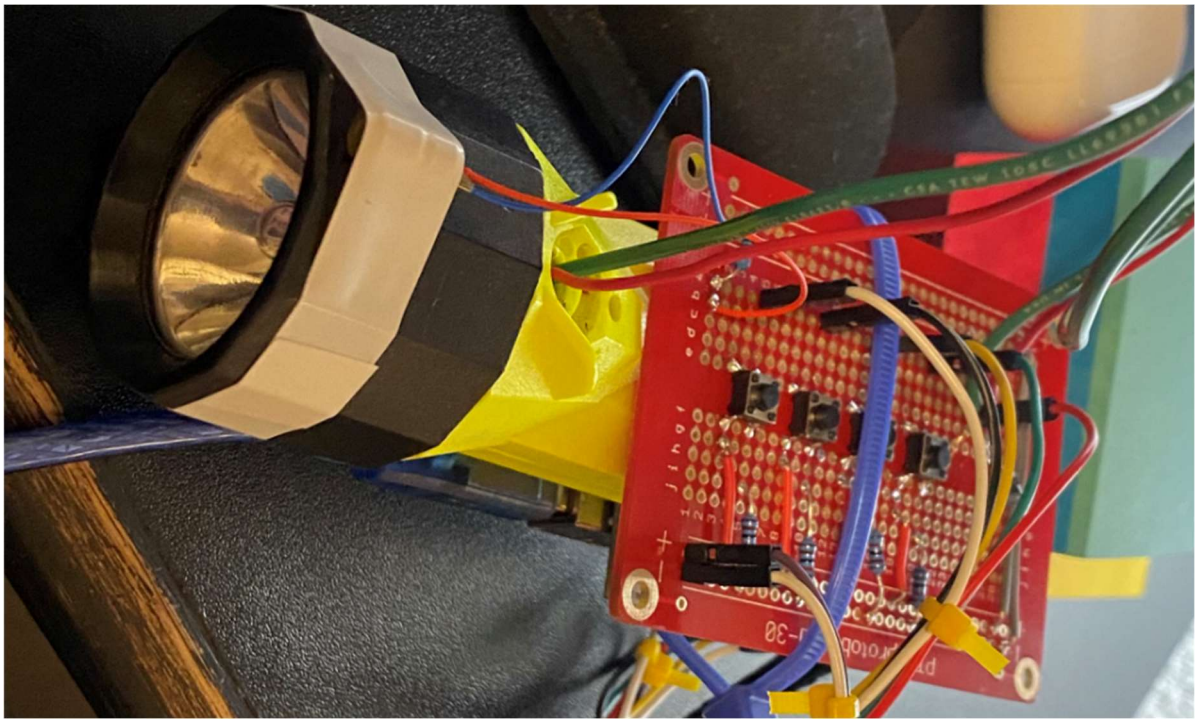


Figure 35: Strobe Beacon - Flashlight Circuit

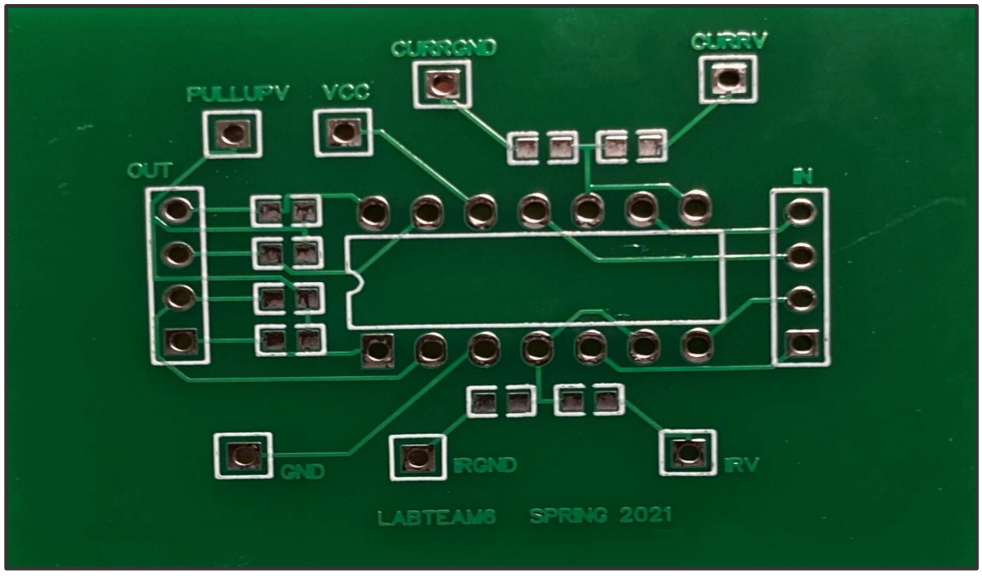


Figure 36: Finalized PCB

D. Software State-Machine Diagrams and Flowcharts

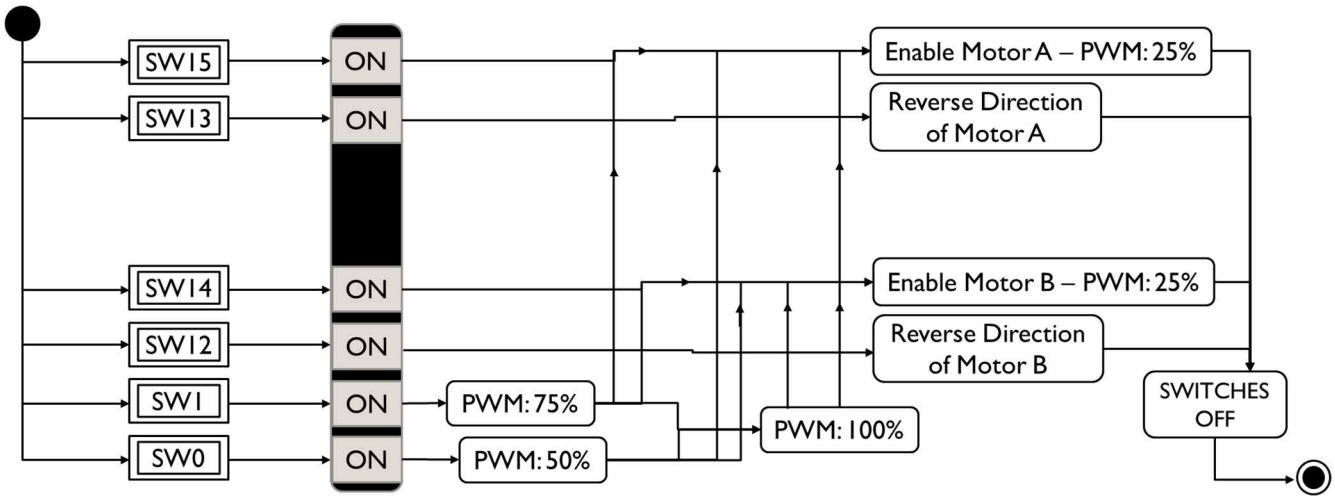


Chart 1: Software Flowchart – Mini-Project

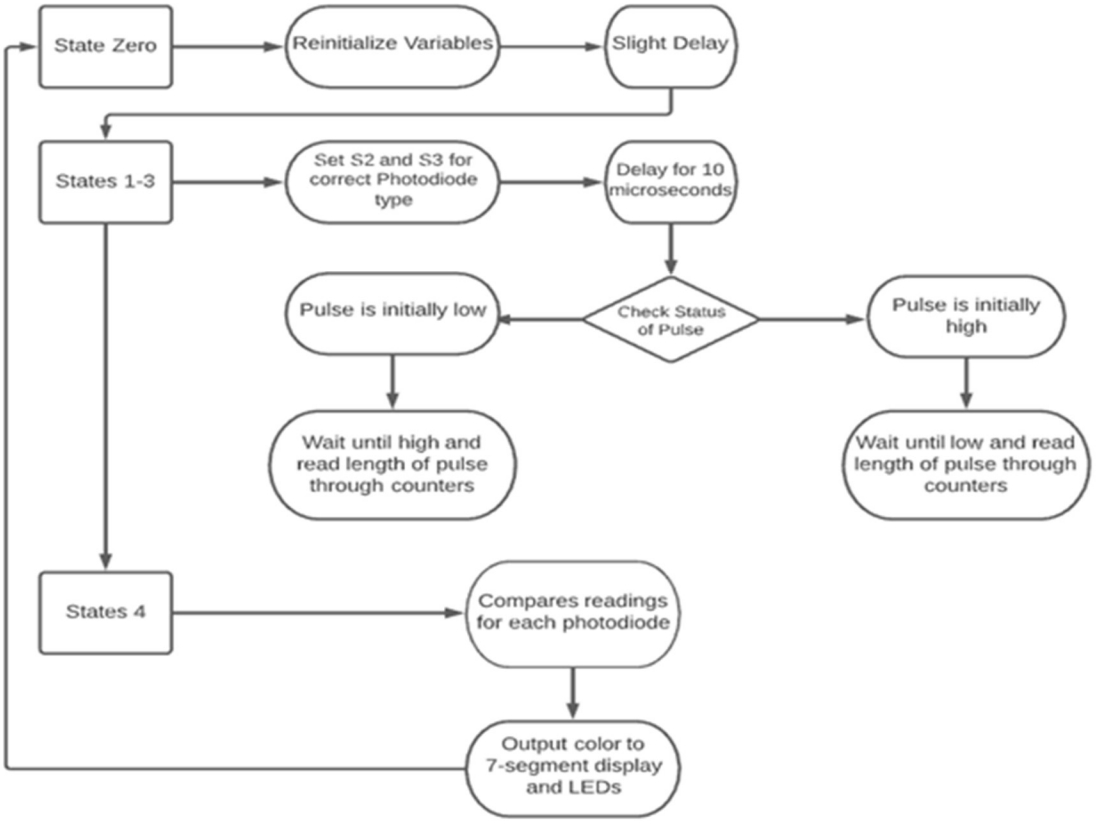


Chart 2: Color Sensor State-Machine Diagram

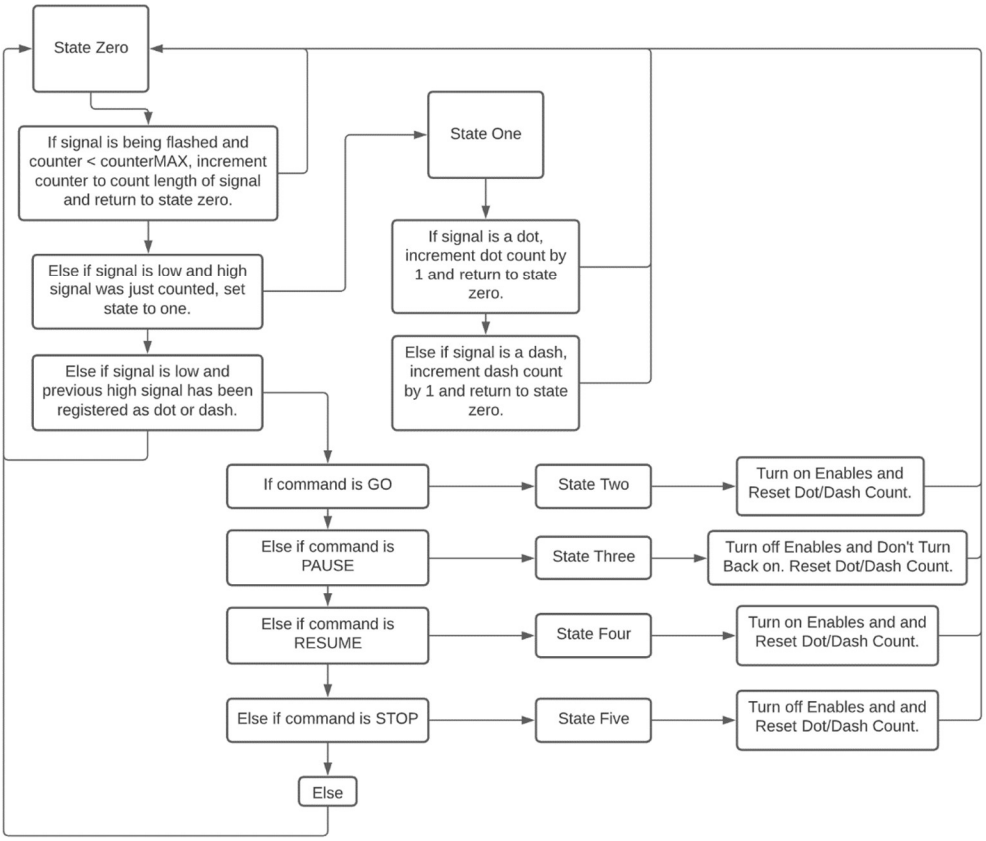


Chart 3: Morse Code State-Machine Diagram

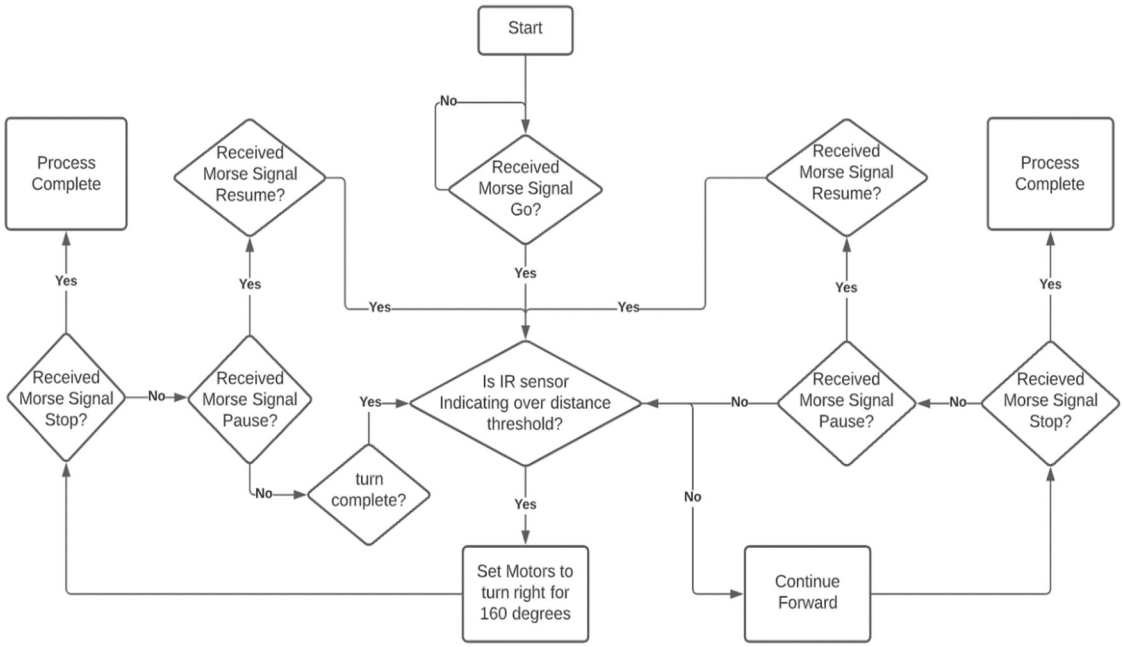


Chart 4: Wall Avoidance State-Machine Diagram

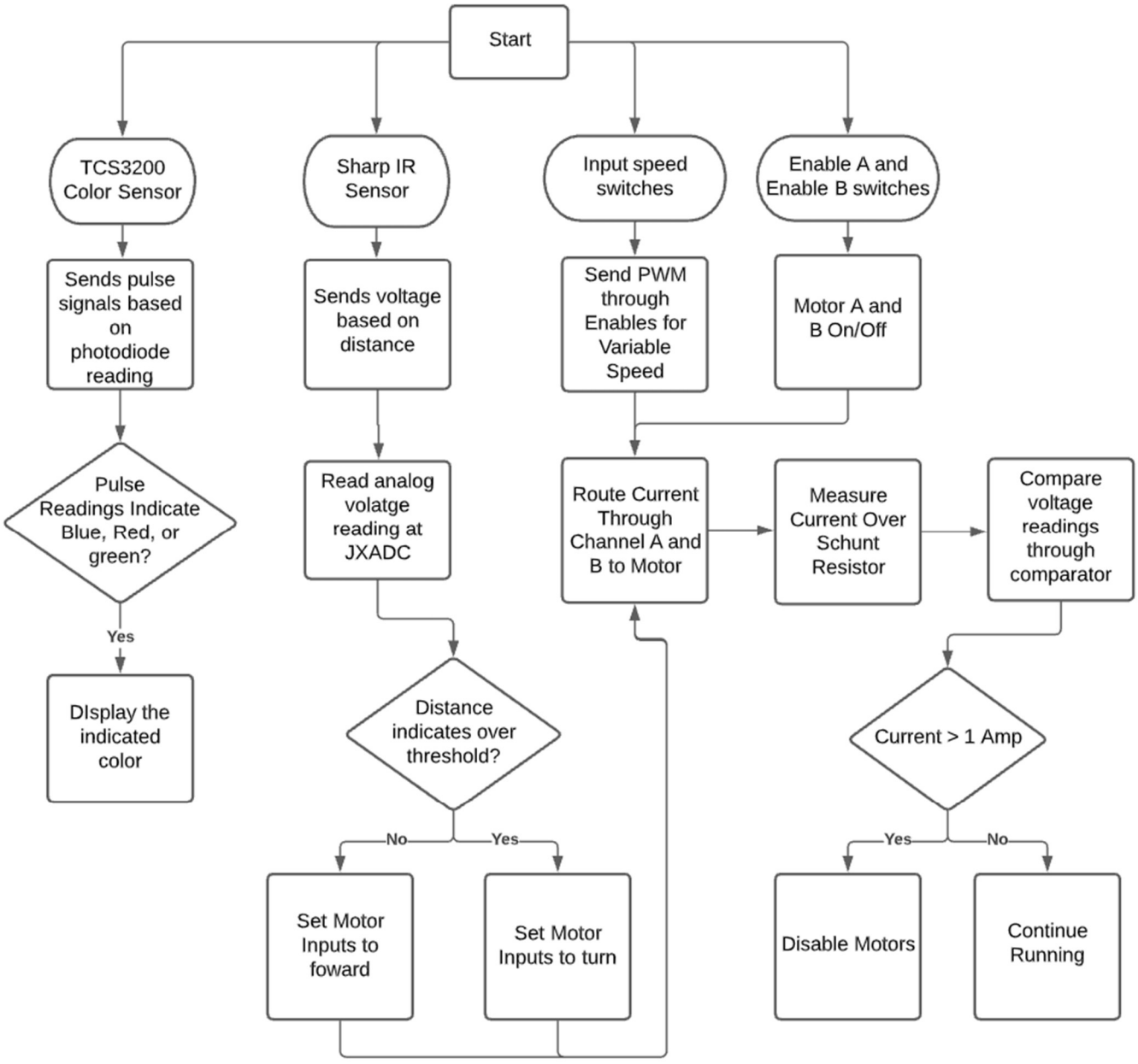


Chart 5: State-Machine Diagram for Entire System

E. Verilog Modules

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 02/06/2021 02:17:11 PM
// Design Name:
// Module Name: MainProject
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
module Main(
input clock, reset,sw14A,sw15B,currentSenseA,
currentSenseB,IRsensorP,IRsensorN,colorsensor_input,vp_in,vn_in,lightSignal,//IRsensorP2,IRsensorN2,
input [1:0] speed,
output [3:0] inputs,an,
output [15:0] led,
output a, b, c, d, e, f, g, dp,enable1, enable2,S0,S1,S2,S3,RedLED,GreenLED,BlueLED
 );
 wire Red_wire, Blue_wire,Green_wire;
```

```
 wire turnOffWire;

 wire PWM;

 wire [11:0] IRsensorWire, IRsensorWire2;

 wire enableStatusWire;

 wire [3:0] inputs_pwmwire;

 wire color_enable_wire;


Overcurrent
instanceOvercurrent(.clock1(clock),.reset(reset),.currentSenseA(currentSenseA),.currentSenseB(currentSenseB),.turnOffSet
(turnOffWire));

IRSensor
instanceSensor(.clock1(clock),.reset(reset),.IRsensorP(IRsensorP),.IRsensorN(IRsensorN),.IRsensor_output(IRsensorWire),.le
d(led),.vp_in(vp_in),.vn_in(vn_in),.IRsensor_output2(IRsensorWire2));//,.IRsensorP2(IRsensorP2),.IRsensorN2(IRsensorN2));

MotorCode
instanceMotor(.clock1(clock),.reset(reset),.PWM(PWM),.sw14A(sw14A),.sw15B(sw15B),.IRsensor(IRsensorWire)

,.enable1(enable1),.enable2(enable2),.inputs(inputs),.turnOff(turnOffWire),.IRsensor2(IRsensorWire2),.enableStatusTemp(
enableStatusWire),.inputs_pwm(inputs_pwmwire),.color_enable(color_enable_wire));

Morse instanceMorse(.clock1(clock),.reset(reset),.enableStatus(enableStatusWire),.lightSignal(lightSignal));

PWM_Code instancePWM(.speed(speed),.clock1(clock),.reset(reset),.PWM_output(PWM),.inputs(inputs_pwmwire));

ColorSensor(.clock1(clock),.colorsensor_input(colorsensor_input),.S0(S0),.S1(S1),.S2(S2),.S3(S3),.Red(Red_wire),.Blue(Blue_
wire),.Green(Green_wire),.RedLED(RedLED),.BlueLED(BlueLED),.GreenLED(GreenLED),.color_enable(color_enable_wire));

SevenSegment
instanceMulti(.clock(clock),.reset(reset),.Red(Red_wire),.Blue(Blue_wire),.Green(Green_wire),.a(a),.b(b),.c(c),.d(d),.e(e),.f(f)
,.g(g),.dp(dp),.an(an),.turnOff(turnOffWire));

endmodule


module SevenSegment(

 input clock, reset,Red,Blue,Green,turnOff,

 output a, b, c, d, e, f, g, dp, //the individual LED output for the seven segment along with the digital point

 output [3:0] an   // the 4 bit enable signal

 );

reg [17:0]count; //the 18 bit counter which allows us to multiplex at 1000Hz

always @ (posedge clock or posedge reset)
```

```
 begin
  if (reset)
   count <= 0;
  else
   count <= count + 1;
 end

reg [6:0]sseg,sseg_temp; //the 7 bit register to hold the data to output
reg [3:0]an_temp; //register for the 4 bit enable
reg enable1Temp, enable2Temp;
always @ (*)
begin
case (count[17:16])
	2'b00:
	begin

	an_temp <= 4'b1110;
	if (Red == 1)
	begin
	sseg <= 4'd0;
	end
	else
	sseg <= 4'd3;
	end
	2'b01:
	begin
	an_temp <= 4'b1101;
	if (Blue == 1)
	begin
```

```
        sseg <= 4'd1;
        end
        else
        sseg <= 4'd3;
        end
        2'b10:
        begin
        an_temp <= 4'b1011;
        if (Green == 1)
        begin
        sseg <= 4'd2;
        end
        else
        sseg <= 4'd3;
        end
        2'b11:
        begin
        an_temp <= 4'b0111;
        if (turnOff == 1)
        sseg <= 4'd3;
        else if (turnOff == 0)
        sseg <= 4'd4;
        end
        endcase
        end

assign an = an_temp;
always @ (*)
 begin
```

```
 case(sseg)
  4'd0 : sseg_temp = 7'b0101111; // display r
  4'd1 : sseg_temp = 7'b0000011; //display b
  4'd2 : sseg_temp = 7'b0010000; //display g
  4'd3 : sseg_temp = 7'b1111111; //to display nothing
  4'd4 : sseg_temp = 7'b1000000; //to display O
  4'd5 : sseg_temp = 7'b0010010; //to display 5
  4'd6 : sseg_temp = 7'b0111111; //dash
  4'd7 : sseg_temp = 7'b1111000; //to display 7
  4'd8 : sseg_temp = 7'b0000000; //to display 8
  4'd9 : sseg_temp = 7'b0010000; //to display 9
  4'd10 : sseg_temp = 7'b0010010; //to display S
  4'd11 : sseg_temp = 7'b0000111; //to display t
  4'd12 : sseg_temp = 7'b1000000; //to display O
  4'd13 : sseg_temp = 7'b0001100; //to display P
  4'd14 : sseg_temp = 7'b0000011; //to display b
  4'd15 : sseg_temp = 7'b1111111; //to display F
   default : sseg_temp = 7'b0111111; //dash
 endcase
end
assign {g, f, e, d, c, b, a} = sseg_temp;
assign dp = 1'b1;
endmodule

module PWM_Code(
input clock1,reset,
input [3:0] inputs,
input [1:0] speed,
output PWM_output
```

```
);
reg PWM;
reg [20:0] counter,count_limit,pwm_limit;
initial begin
counter = 0;
count_limit = 21'd1666666; // 60 Hz - 1,666,666 counts of 100 MHz clock - fits with 21 bit counter
pwm_limit =   21'd0; // 0% duty cycle - 0 counts of 100 MHz clock - fits with 21 bit counter
PWM = 1;
end
always @ (posedge clock1) begin
if (reset || counter == count_limit)
   counter <= 0;
else
   counter <= counter + 1;
if (counter < pwm_limit)
   PWM <= 1;
else
   PWM <= 0;
end
always @ (*) begin
case(speed)
   2'd0 : begin
         if (inputs == 4'b0110)
         pwm_limit <= 21'd0833333;  // speed bits 01 - counter will be reset every 416,667 pulses of 100 MHz clock
         else if (inputs == 4'b1010)
         pwm_limit <= 21'd1333332;
         end
   2'd1 : pwm_limit = 21'd1666666; // speed bits 10 - counter will be reset every 813,333 pulses of 100 MHz clock
   default: pwm_limit = 21'd0416667; // Not really used since all states accounted for with for assignments
```

```
   endcase
end
assign PWM_output = PWM;
endmodule

module Overcurrent(
input currentSenseA,currentSenseB,clock1,reset,
output turnOffSet
);
reg turnOff;
reg [20:0] counterCurrentA,counterCurrentB,counterCurrent_limit;
initial begin
counterCurrentA = 21'd0;
counterCurrentB = 21'd0;
counterCurrent_limit = 21'd1666666;
turnOff = 1;
end
always @ (posedge clock1)
 begin
  if (currentSenseA == 1)
      counterCurrentA <= counterCurrentA + 1;
  else if (currentSenseA == 0)
  begin
      counterCurrentA <= 0;
  end
  if (counterCurrentA == counterCurrent_limit)
   turnOff <= 0;
  if (currentSenseB == 1)
      counterCurrentB <= counterCurrentB + 1;
```

```
  else if (currentSenseB == 0)
  begin
      counterCurrentB <= 0;
  end
  if (counterCurrentB == counterCurrent_limit)
   turnOff <= 0;
end
 assign turnOffSet = turnOff;
endmodule
module Morse(
input lightSignal,clock1,reset,
output enableStatus
);
reg [26:0] morseCounter, morseCounterMAX, morseCounterMAX2, morseCounterDIFF, morseCounterDIFF2,
morseCounterInitial, clearCommand;
reg dotTrue, dashTrue, goTrue, enableStatusTemp;
reg [1:0] prevCommand;
reg [1:0] dotCount, dashCount;
reg [2:0] outMorse;
reg [2:0] stateMorse;
parameter zero=0, one=1, two=2, three=3, four =4, five = 5, six = 6;
initial begin
morseCounter = 27'd0;
morseCounterMAX = 27'd90000000;
morseCounterDIFF = 27'd50000000;
morseCounterDIFF2 = 27'd10000000;
morseCounterInitial = 27'd1000000;
clearCommand = 27'd0;
prevCommand = 2'd0;
dotTrue = 0;
```

```
dashTrue = 0;
dotCount = 0;
dashCount = 0;
goTrue = 0;
enableStatusTemp = 0;
stateMorse = zero;
end
always @(stateMorse)
    begin
        case (stateMorse)
            zero:
                outMorse <= 3'b000;
            one:
                outMorse <= 3'b001;
            two:
                outMorse <= 3'b010;
            three:
                outMorse <= 3'b011;
            four:
                outMorse <= 3'b100;
            five:
                outMorse <= 3'b101;
            default:
                outMorse <= 3'b000;
        endcase
    end
always @(posedge clock1)
begin
 if (reset)
```

```
  stateMorse <= zero;
else
  case (stateMorse)
   zero:
   begin
     if (lightSignal == 1 && (morseCounter < morseCounterMAX))
     begin
        morseCounter <= morseCounter + 1;
        stateMorse <= zero;
     end
     else if (lightSignal == 0 && (morseCounter > morseCounterInitial))
     begin
        stateMorse <= one;
        clearCommand <= 0;
     end
     else if (lightSignal == 0 && morseCounter == 0)
     begin
        if (dotCount == 1 && dashCount == 2 && goTrue == 1 && prevCommand == 0)  //go
        begin
           stateMorse <= two;
        end
        else if (dotCount == 2 && dashCount == 2 && goTrue == 0 && (prevCommand == 1 || prevCommand == 3))  //pause
        begin
           stateMorse <= three;
        end
        else if (dotCount == 2 && dashCount == 1 && prevCommand == 2)  //resume
        begin
           stateMorse <= four;
        end
```

```
        else if (dotCount == 3 && dashCount == 0 && (prevCommand == 1 || prevCommand == 3))  //stop
         begin
            stateMorse <= five;
         end
         else if (clearCommand < morseCounterMAX)
         begin
            clearCommand <= clearCommand + 1;
            stateMorse <= zero;
         end
         else if (clearCommand == morseCounterMAX)
         begin
            clearCommand <= 0;
            dotCount <= 0;
            dashCount <= 0;
            stateMorse <= zero;
         end
       end
    end
one:
    begin
      if (morseCounter < morseCounterDIFF && morseCounter > morseCounterDIFF2)
      begin
         dotTrue <= 1;
         morseCounter <= 0;
      end
      else if (morseCounter >= morseCounterDIFF)
      begin
         dashTrue <= 1;
         morseCounter <= 0;
```

```
        end
if (dotTrue == 1)
        begin
            dotTrue <= 0;
            dotCount <= dotCount + 1;
            stateMorse <= zero;
        end
        else if (dashTrue == 1)
        begin
            dashTrue <= 0;
            dashCount <= dashCount + 1;
            if (dotCount == 1)
                goTrue <= 0;
            else
                goTrue <= 1;
            stateMorse <= zero;
        end
    end
two:
    begin
        prevCommand <= 1;
        enableStatusTemp <= 1;
        //inputsTemp <= 4'b0110;
        dotCount <= 0;
        dashCount <= 0;
        stateMorse <= zero;
    end
three:
    begin
```

```
        prevCommand <= 2;
        enableStatusTemp <= 0;
        //inputs_temp <= 4'b0110;
        dotCount <= 0;
        dashCount <= 0;
        stateMorse <= zero;
    end
four:
    begin
        prevCommand <= 3;
        enableStatusTemp <= 1;
        //inputsTemp <= 4'b0110;
        dotCount <= 0;
        dashCount <= 0;
        stateMorse <= zero;
    end
five:
    begin
        enableStatusTemp <= 0;
        //inputsTemp <= 4'b0110;
        dotCount <= 0;
        dashCount <= 0;
        stateMorse <= zero;
    end
endcase
end
assign enableStatus = enableStatusTemp;
endmodule
```

```
module MotorCode (
input sw14A, sw15B,clock1,reset,turnOff,PWM,enableStatusTemp,color_enable,
input [11:0] IRsensor,IRsensor2,
input [1:0] speed,
output enable1, enable2,turnOffSet,
output [3:0] inputs,inputs_pwm
);
reg enable1_temp,enable2_temp;
reg [3:0] inputs_temp;
reg [11:0] distance, distance_read;
//reg [7:0] delay, delay_max;
reg [3:0] out;
reg [2:0] state, previous_state;
reg [32:0] counter1, counter1_max;
reg [32:0] Acounter,Acounter_max,Acounter2; //90 degree turn-figure out the time to turn 90 degrees

parameter zero=0, one=1, two=2, three=3, four =4, five = 5, six = 6;
initial begin

state = zero;
enable1_temp = 0;enable2_temp = 0;inputs_temp = 4'b0000; Acounter = 0; Acounter2 = 0; counter1 = 0;
counter1_max = 245000000;//525000000
Acounter_max = 252000000;  //35% duty cycle lab floor
end
always @(state)
    begin
        case (state)
            zero:
                out <= 4'b0000;
```

```
            one:
                out <= 4'b0001;
            two:
                out <= 4'b0010;
            default:
                out <= 4'b0000;
        endcase
    end
always @ (posedge clock1)
begin
        if (reset)
        begin
            state <= zero;
            inputs_temp <= 4'b0000;
        end
        else
            case (state)
                zero:
                 begin
                   counter1 <= 0;
                   state <= one;
                 end
                one:
                    if (IRsensor > 1900)
                        state <= two;
                    else
                    begin
                        inputs_temp <= 4'b0110;
                    end
```

```
                two:
                if (counter1 < counter1_max)
                begin
                   inputs_temp <= 4'b1010;
                   counter1 <= counter1 + 1;
                end
                else if (counter1 == counter1_max)
                state <= zero;
            endcase
end
always @ (posedge clock1)
 begin
  case(sw14A)
   1'b1 :
    enable1_temp <= 1;
   1'b0:
    enable1_temp <= 0;
  endcase
end

 always @ (posedge clock1)
 begin
  case(sw15B)
   1'b1 :
    enable2_temp <= 1;
   1'b0:
    enable2_temp <= 0;
  endcase
end
```

```
  assign enable1 = enable1_temp && enableStatusTemp && PWM && turnOff && color_enable;
  assign enable2 = enable2_temp && enableStatusTemp  && PWM && turnOff && color_enable;
  assign inputs = inputs_temp;
  assign inputs_pwm = inputs_temp;
endmodule
module IRSensor(
    input clock1,reset,
    input IRsensorP,
    input IRsensorN,
    input vp_in,
    input vn_in,

    output reg [15:0] led,
    output [11:0] IRsensor_output,IRsensor_output2
);
    wire enable,enable2;
    wire ready,ready2;
    wire [15:0] data,data2;
    reg [6:0] Address_in,Adress_in2;
    reg [11:0] output1;
//secen segment controller signals
    reg [32:0] count;
    localparam S_IDLE = 0;
    localparam S_FRAME_WAIT = 1;
    localparam S_CONVERSION = 2;
    reg [1:0] state = S_IDLE;

    //xadc instantiation connect the eoc_out .den_in to get continuous conversion
    xadc_wiz_0  XLXI_7 (
```

```
    .daddr_in(8'h16), //addresses can be found in the artix 7 XADC user guide DRP register space
    .dclk_in(clock1),
    .den_in(enable),
    .di_in(0),
    .dwe_in(0),
    .busy_out(),
    .vauxp6(IRsensorP),
    .vauxn6(IRsensorN),
    .vn_in(vp_in),
    .vp_in(vn_in),
    .alarm_out(),
    .do_out(data),
    .eoc_out(enable),
    .channel_out(),
    .drdy_out(ready)
);
//led visual dmm
always @(posedge(clock1)) begin
    if(ready == 1'b1) begin
        case (data[15:12])
        1:  led <= 16'b11;
        2:  led <= 16'b111;
        3:  led <= 16'b1111;
        4:  led <= 16'b11111;
        5:  led <= 16'b111111;
        6:  led <= 16'b1111111;
        7:  led <= 16'b11111111;
        8:  led <= 16'b111111111;
        9:  led <= 16'b1111111111;
```

```
          10: led <= 16'b11111111111;
          11: led <= 16'b111111111111;
          12: led <= 16'b1111111111111;
          13: led <= 16'b11111111111111;
          14: led <= 16'b111111111111111;
          15: led <= 16'b1111111111111111;
          default: led <= 16'b1;
          endcase
       End
end
   always @ (ready)
   begin
   output1 <= data [15:4];
   end
   assign IRsensor_output = output1;
endmodule
module ColorSensor (
input clock1, colorsensor_input, reset,
output Red,Blue,Green,S0,S1,S2,S3,RedLED,BlueLED,GreenLED,color_enable
);
reg [2:0] out;
reg [2:0] state;
reg [30:0] stop, stop_max,stop_delay,stop_delay_max;
reg [20:0] delay, delay_max;
reg detect_bool;
reg status, status_check;
reg led_red,led_blue,led_green;
reg [29:0] led_delay_red,led_delay_blue,led_delay_green, led_delay_max;
reg [6:0] microsecond_counter, microsecond_max;
```

```
reg [19:0] pulselengthRed,pulselengthBlue,pulselengthGreen;
reg boolRed,boolGreen,boolBlue,color_enable_reg;
reg S0_reg,S1_reg,S2_reg,S3_reg, Red_reg,Blue_reg,Green_reg;
parameter zero=0, one=1, two=2, three=3, four = 4;
initial begin
S0_reg = 1; S1_reg = 0; S2_reg = 0; S3_reg = 0;
pulselengthRed = 0;pulselengthBlue = 0;pulselengthGreen = 0;
boolRed = 0;boolBlue = 0; boolGreen = 0;
detect_bool = 0;
Red_reg = 0;Blue_reg = 0;Green_reg = 0; stop_max = 100000000; stop = 0;stop_delay = 0; stop_delay_max = 100000000;
status_check = 0;
delay = 0; delay_max = 32767;
led_delay_red = 0;led_delay_blue = 0;led_delay_blue = 0; led_delay_max = 170000000; color_enable_reg = 1;
end
always @(state)
    begin
        case (state)
            zero:
                out <= 3'b000;
            one:
                out <= 3'b001;
            two:
                out <= 3'b010;
            three:
                out <= 3'b011;
            four:
                out <= 3'b100;
            default:
                out <= 3'b000;
```

```
        endcase
    end

always @(posedge clock1)
    begin
        if (reset)
            state <= zero;
        else
            case (state)
     zero:
                    begin

                     pulselengthRed <= 0;
                     pulselengthBlue <= 0;
                     pulselengthGreen <= 0;

                     status_check <= 0;
                     boolRed <= 0;
                     boolGreen <= 0;
                     boolBlue <= 0;
                     if (delay < 4*delay_max)
                     delay <= delay + 1;
                     else
                     state <= one;
                     end


     one: //read red photodiode
     begin
```

```
    S2_reg <= 0;
    S3_reg <= 0;
  if(delay == delay_max || delay > delay_max)
        begin
  if (colorsensor_input == 0 && status_check == 0)
  begin
status_check <= 1;
status <= 0;
end
else if (colorsensor_input == 1 && status_check == 0)
begin
status <= 1;
status_check <= 1;
end

  if (status == 1)
  begin
   if(colorsensor_input == 0 && boolRed == 0)
                begin
                boolRed <= 1;
                pulselengthRed <= pulselengthRed+ 1; //pulselengthRed <= pulselengthBlue+ 1;
                end

                   else if (colorsensor_input == 0 && boolRed == 1)
                begin
                pulselengthRed <= pulselengthRed + 1;
                end

                   else if (colorsensor_input == 1 && boolRed == 1)
```

```
                begin


                  delay <= 0;
                status_check <= 0;
                boolRed <= 0;
                state <= two;
                end
        end

        if (status == 0)
        begin
         if(colorsensor_input == 1 && boolRed == 0)
                begin
                boolRed <= 1;
                pulselengthRed <= pulselengthRed + 1;
                  end
else if (colorsensor_input == 1 && boolRed == 1)
                  begin
                  pulselengthRed <= pulselengthRed + 1;
                  end
else if (colorsensor_input == 0 && boolRed == 1)
                 begin


                delay <= 0;
                status_check <=0;
                boolRed <= 0;
                state <= two;
                end
        end
```

```
    end
            else if (delay < delay_max)
            begin
            delay <=delay + 1;
            end
            end


    two: // read blue


begin
            S2_reg <= 0;
            S3_reg <= 1;
   if (delay == delay_max || delay > delay_max)
   begin
           if (colorsensor_input == 0 && status_check == 0)
   begin
   status_check <= 1;
   status <= 0;
   end
      else if (colorsensor_input == 1 && status_check == 0)
      begin
   status <= 1;
   status_check <= 1;
   end

  if (status == 1)
  begin
                if(colorsensor_input == 0 && boolBlue == 0)
```

```
                begin
                boolBlue <= 1;
                pulselengthBlue<= pulselengthBlue+ 1;
                end

                  else if (colorsensor_input == 0 && boolBlue == 1)
                begin
                pulselengthBlue <= pulselengthBlue + 1;
                end

                else if (colorsensor_input == 1 && boolBlue == 1)
                  begin

                delay <= 0;
                status_check <= 0;
                boolBlue <= 0;
                state <= three;
                end
end

else if (status == 0)
begin
    if(colorsensor_input == 1 && boolBlue == 0)
                begin
                boolBlue <= 1;
                pulselengthBlue<= pulselengthBlue+ 1;
                  end

                  else if (colorsensor_input == 1 && boolBlue == 1)
```

```
                    begin
                    pulselengthBlue <= pulselengthBlue + 1;
                    end

                      else if (colorsensor_input == 0 && boolBlue == 1)
                begin

                delay <= 0;
                status_check <=0;
                boolBlue <=0;
                state <= three;
                end
      end



    end
            else if (delay < delay_max)
            begin
            delay <= delay + 1;
            end
end

 three: //read green
 begin
            S2_reg <= 1;
            S3_reg <= 1;
 if (delay == delay_max || delay > delay_max)
            begin
```

```
        if (colorsensor_input == 0 && status_check == 0)
        begin
        status <= 0;
        status_check <= 1;

                end

        else if (colorsensor_input == 1 && status_check == 0)
        begin
        status <= 1;
        status_check <= 1;
        end

if (status == 1)
begin
      if(colorsensor_input == 0 && boolGreen == 0)
                  begin
                  boolGreen <= 1;
                  pulselengthGreen <= pulselengthGreen + 1;
                  end

                  else if (colorsensor_input == 0 && boolGreen == 1)
                  begin
                  pulselengthGreen <= pulselengthGreen + 1;
                  end

                  else if (colorsensor_input == 1 && boolGreen == 1)
                  begin
```

```
                    delay <= 0;
                    status_check <= 0;
                    boolGreen <= 0;
                    state <= four;
                end
end
else if (status == 0)
begin
    if(colorsensor_input == 1 && boolGreen == 0)
                    begin
                    boolGreen <= 1;
                    pulselengthGreen <= pulselengthGreen + 1;
                    end
                    else if (colorsensor_input == 1 && boolGreen == 1)
  begin
                    pulselengthGreen <= pulselengthGreen + 1;
                      end
                     else if (colorsensor_input == 0 && boolGreen == 1)
                    begin
                    delay <= 0;
                    status_check <= 0;
      boolGreen <= 0;
      state <= four;
                    end
    end
    end
            else if(delay < delay_max)
              begin
              delay <= delay + 1;
```

```
                end
    end
    four: // compare
                begin
                Red_reg <= 0;
                Blue_reg <=0;
                Green_reg <= 0;

                 if(pulselengthRed < 30200 && pulselengthRed > 10100 && pulselengthBlue > 30400 && pulselengthBlue <
65000 && pulselengthGreen > 35000 && pulselengthGreen < 65000)
                begin
                Red_reg <=1;
                state <= zero;
                end
                 if(pulselengthRed < 72000 && pulselengthRed > 33000 && pulselengthBlue > 5100 && pulselengthBlue < 27900
&& pulselengthGreen > 6400 && pulselengthGreen < 37300)
                begin
                Blue_reg <=1;
                state <= zero;
                end
                else if(pulselengthRed < 62900 && pulselengthRed > 31900 && pulselengthBlue > 10300 && pulselengthBlue <
41100 && pulselengthGreen > 8400 && pulselengthGreen < 34300)
                begin
                Green_reg <=1;
                state <= zero;
                end
                else
                begin
                state <= zero;
                end
```

```
            end
        endcase
end
always @(posedge clock1)
begin
if (Red_reg == 1)
led_red <=1;
if (led_red == 1 && led_delay_red < led_delay_max)
led_delay_red <=led_delay_red + 1;
else if (led_delay_red == led_delay_max)
led_red <= 0;
led_delay_red = 0;

if (Blue_reg == 1)
led_blue <=1;
if (led_blue == 1 && led_delay_blue < led_delay_max)
led_delay_blue <=led_delay_blue + 1;
else if (led_delay_blue == led_delay_max)
led_blue <= 0;
led_delay_blue = 0;

if (Green_reg == 1)
led_green <=1;
if (led_green == 1 && led_delay_green < led_delay_max)
led_delay_green <=led_delay_green + 1;
else if (led_delay_green == led_delay_max)
led_green <= 0;
led_delay_green = 0;
end
```

```
always @ (posedge clock1)
begin
if((led_green == 1 || led_blue == 1 || led_red == 1) && stop != stop_max)
begin
detect_bool <= 1;
color_enable_reg <= 0;
stop <= stop + 1;
end
else if(stop == stop_max && detect_bool == 1)
begin
color_enable_reg <= 1;
stop_delay <= stop_delay + 1;
if(stop_delay == stop_delay_max)
begin
detect_bool <= 0;
stop <= 0;
stop_delay <= 0;
end
end
end

assign S0 = S0_reg;
assign S1 = S1_reg;
assign S2 = S2_reg;
assign S3 = S3_reg;
assign Red = led_red;
assign Blue = led_blue;
assign Green = led_green;
assign RedLED = led_red;
```

```
    assign BlueLED = led_blue;
    assign GreenLED = led_green;
    assign color_enable = color_enable_reg;
endmodule
```

F. Xilinx Vivado Simulation Waveforms

Figure 37: Mini-Project – Simulation Waveform

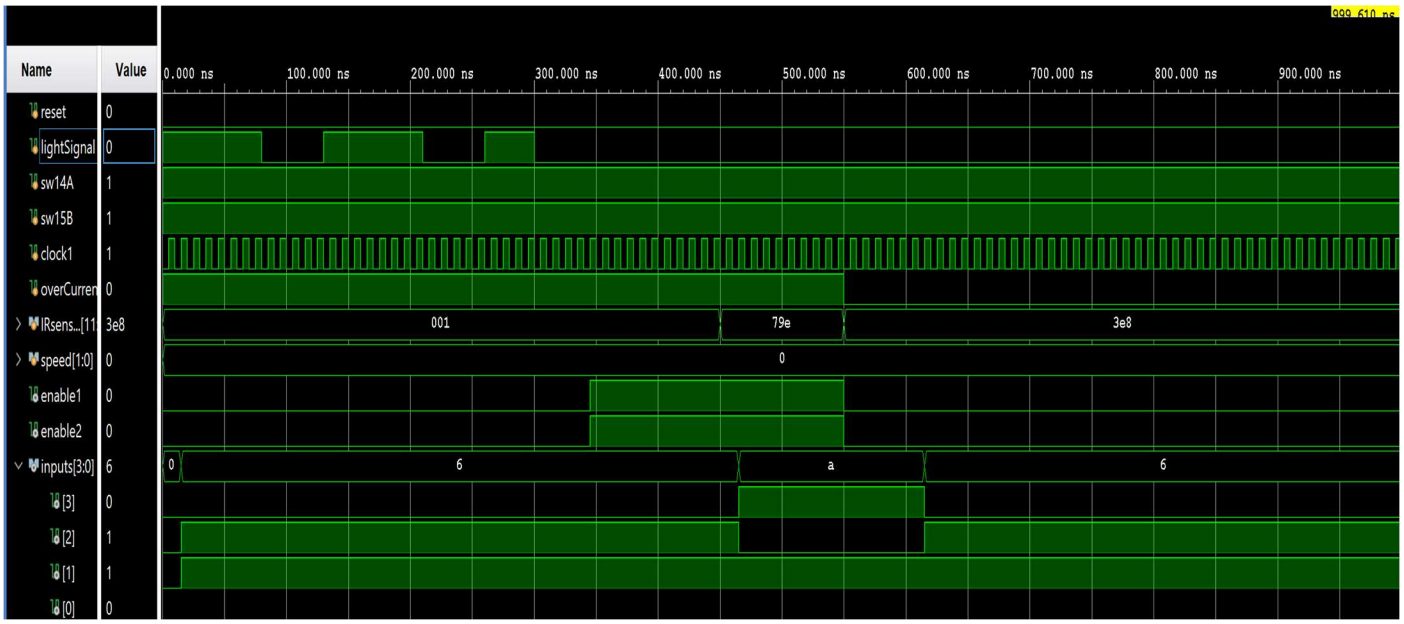


Figure 38: Wall Avoidance and Overcurrent – Simulation Waveform

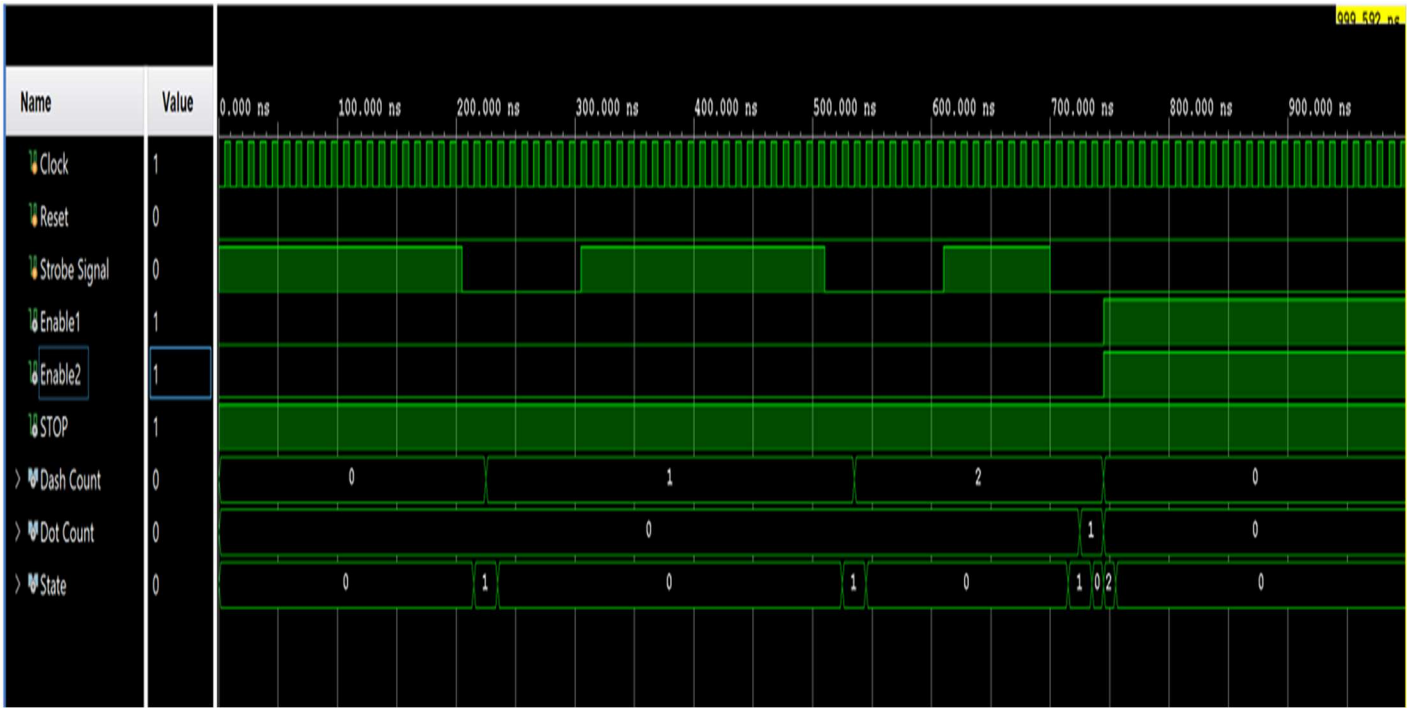


Figure 39: Morse Code GO – Simulation Waveform

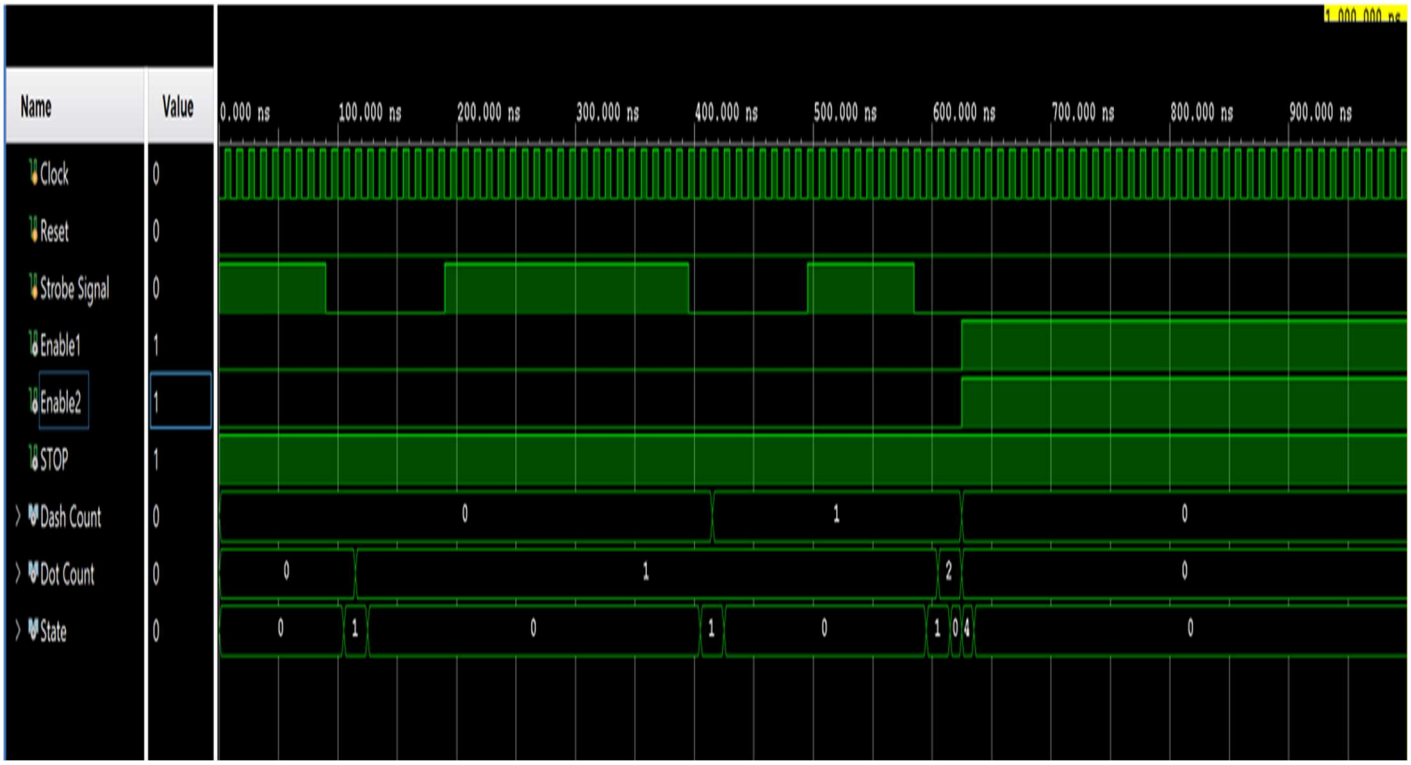


Figure 40: Morse Code RESUME – Simulation Waveform

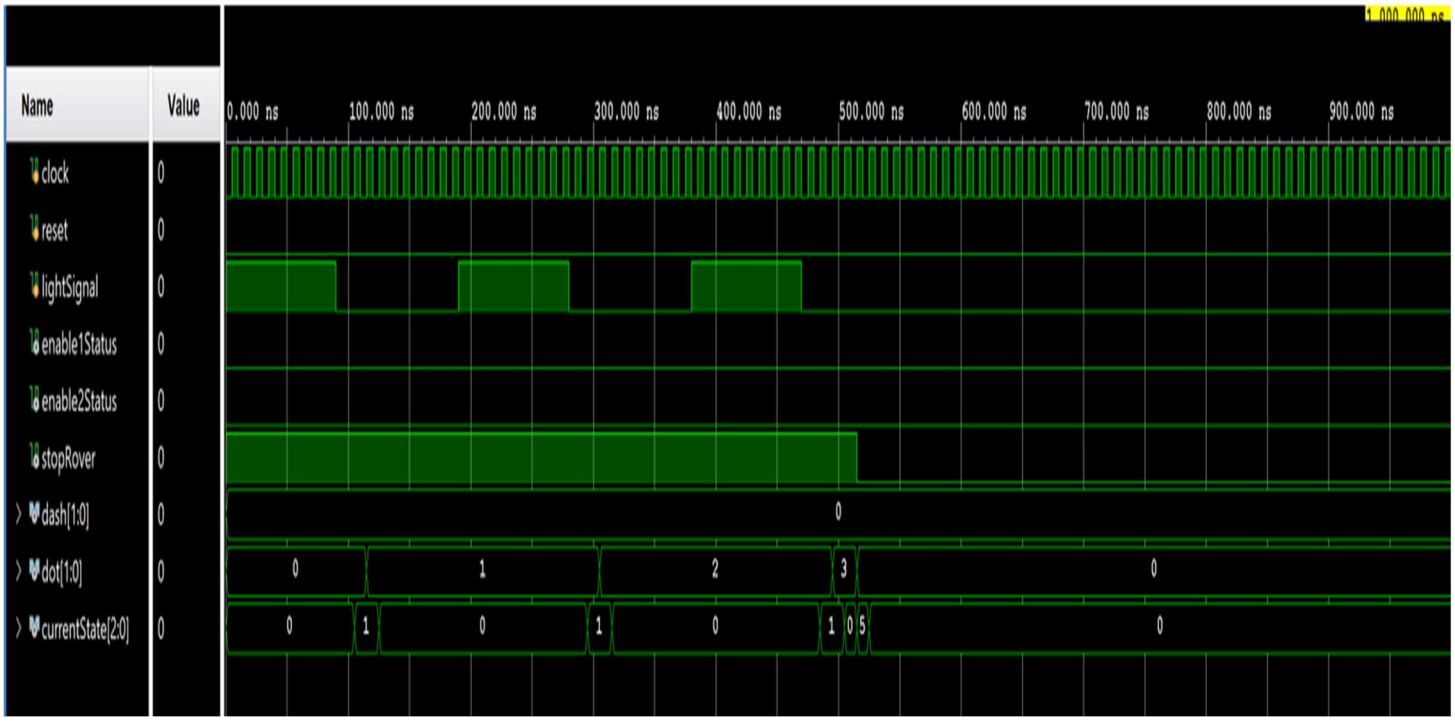


Figure 41: Morse Code STOP – Simulation Waveform

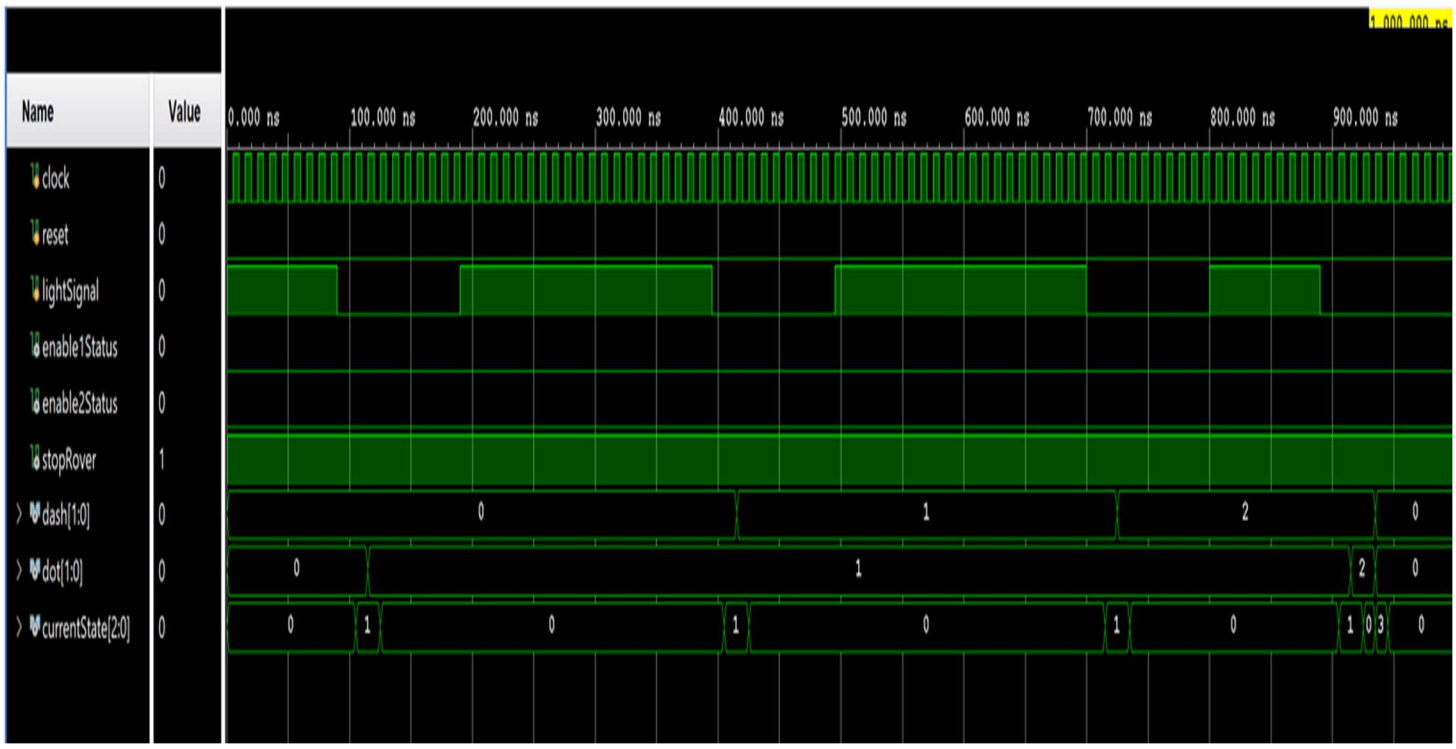


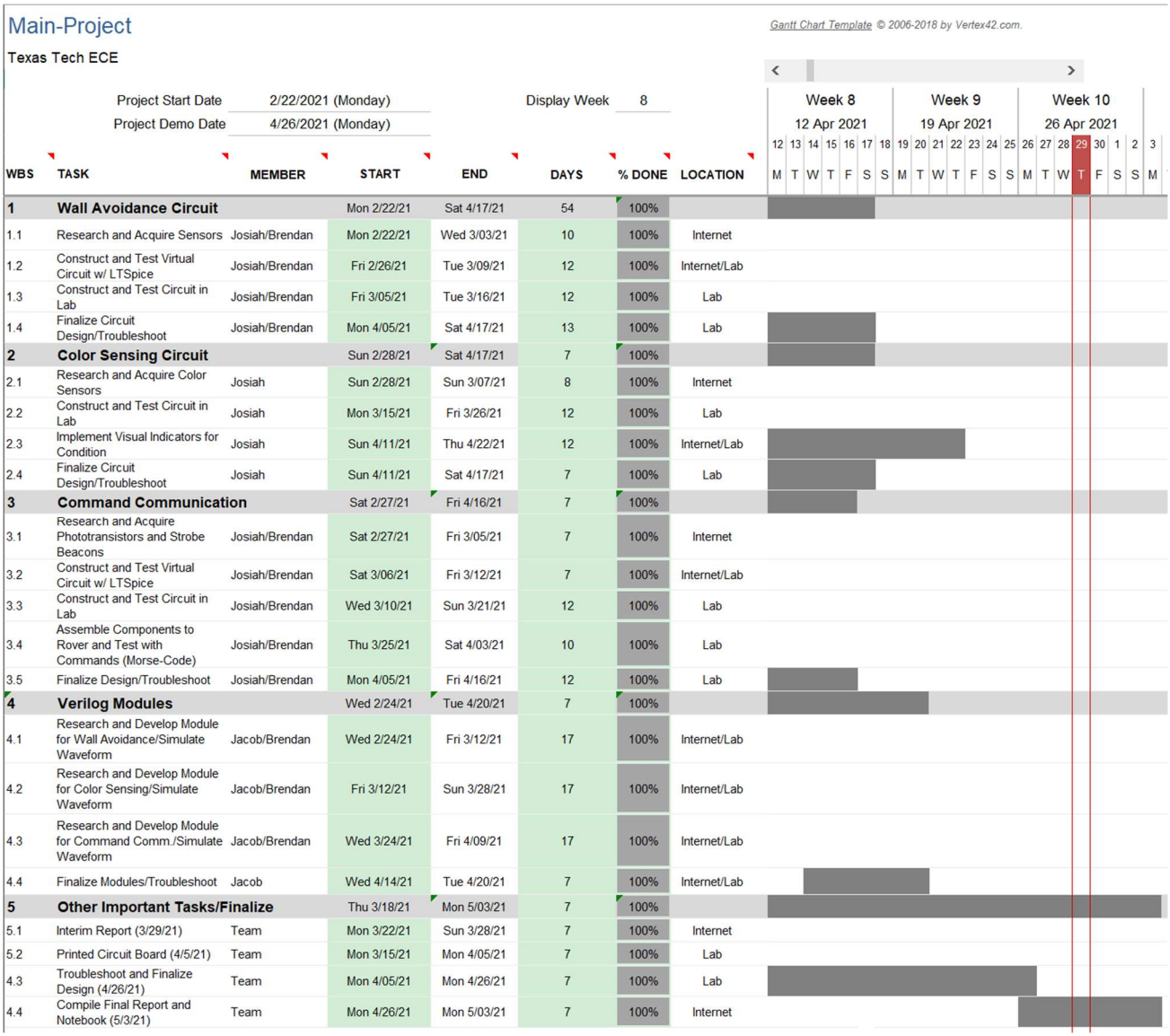
Figure 42: Morse Code PAUSE – Simulation Waveform

G. Gantt Chart

WBS	TASK	MEMBER	START	END	DAYS	% DONE	LOCATION
1	**Lab Pre-Reqs**		Mon 1/25/21	Sat 1/30/21	5	100%	
1.1	Lab Safety Videos and Quiz	Team	Tue 1/26/21	Tue 1/26/21	1	100%	Internet
1.2	Lab Safety Training	Team	Tue 1/26/21	Tue 1/26/21	1	100%	Internet
1.3	Soldering H-Bridge Board	Team	Wed 1/27/21	Thu 1/28/21	2	100%	Lab
1.4	Downloading/Learning Appropriate Software	Team	Fri 1/29/21	Sat 1/30/21	2	100%	Internet
2	**DC Motor Circuit**		Sat 1/30/21	Thu 2/18/21	19	100%	
2.1	Research Circuit Design	Josiah	Sat 1/30/21	Wed 2/03/21	5	100%	Internet
2.2	Construct and Test Circuit with Ltspice	Josiah	Wed 2/03/21	Fri 2/05/21	3	100%	Internet/Lab
2.3	Construct and Test Circuit in Lab/Get Components	Josiah	Fri 2/05/21	Tue 2/09/21	5	100%	Lab
2.4	Troubleshoot and Finalize Circuit	Team	Tue 2/09/21	Thu 2/18/21	10	100%	Lab
3	**DC Motor Code**		Sun 1/31/21	Thu 2/18/21	18	100%	
3.1	Research Verilog/FPGA	Brendan/Jacob	Sun 1/31/21	Wed 2/03/21	4	100%	Internet
3.2	Construct Flowchart	Brendan/Jacob	Sun 1/31/21	Wed 2/03/21	4	100%	Internet
3.3	Write Code/Testbench and Run Simulations	Brendan/Jacob	Wed 2/03/21	Tue 2/09/21	7	100%	Lab
3.4	Debug Code	Brendan/Jacob	Wed 2/03/21	Tue 2/09/21	7	100%	Lab
3.5	Troubleshoot and Finalize Code	Team	Tue 2/09/21	Thu 2/18/21	10	100%	Lab
4	**Finalize/Protection**		Sun 2/07/21	Sun 2/21/21	14	100%	
4.1	Overcurrent Protection - Software	Jacob/Brendan	Sun 2/07/21	Tue 2/16/21	10	100%	Lab
4.2	Overcurrent Protection - Hardware	Josiah	Sun 2/07/21	Tue 2/16/21	10	100%	Lab
4.3	Compile Report	Team	Fri 2/12/21	Sun 2/21/21	10	100%	Internet
4.4	Troubleshoot Problems and Finalize	Team	Tue 2/09/21	Thu 2/18/21	10	100%	Internet/Lab

Chart 6: Mini-Project Final Gantt Chart

Chart 7: Main-Project Interim Gantt Chart



Main-Project

Texas Tech ECE

Gantt Chart Template © 2006-2018 by Vertex42.com.

Project Start Date 2/22/2021 (Monday)

Project Demo Date 4/26/2021 (Monday)

Display Week 8

WBS	TASK	MEMBER	START	END	DAYS	% DONE	LOCATION
1	**Wall Avoidance Circuit**		Mon 2/22/21	Sat 4/17/21	54	100%	
1.1	Research and Acquire Sensors	Josiah/Brendan	Mon 2/22/21	Wed 3/03/21	10	100%	Internet
1.2	Construct and Test Virtual Circuit w/ LTSpice	Josiah/Brendan	Fri 2/26/21	Tue 3/09/21	12	100%	Internet/Lab
1.3	Construct and Test Circuit in Lab	Josiah/Brendan	Fri 3/05/21	Tue 3/16/21	12	100%	Lab
1.4	Finalize Circuit Design/Troubleshoot	Josiah/Brendan	Mon 4/05/21	Sat 4/17/21	13	100%	Lab
2	**Color Sensing Circuit**		Sun 2/28/21	Sat 4/17/21	7	100%	
2.1	Research and Acquire Color Sensors	Josiah	Sun 2/28/21	Sun 3/07/21	8	100%	Internet
2.2	Construct and Test Circuit in Lab	Josiah	Mon 3/15/21	Fri 3/26/21	12	100%	Lab
2.3	Implement Visual Indicators for Condition	Josiah	Sun 4/11/21	Thu 4/22/21	12	100%	Internet/Lab
2.4	Finalize Circuit Design/Troubleshoot	Josiah	Sun 4/11/21	Sat 4/17/21	7	100%	Lab
3	**Command Communication**		Sat 2/27/21	Fri 4/16/21	7	100%	
3.1	Research and Acquire Phototransistors and Strobe Beacons	Josiah/Brendan	Sat 2/27/21	Fri 3/05/21	7	100%	Internet
3.2	Construct and Test Virtual Circuit w/ LTSpice	Josiah/Brendan	Sat 3/06/21	Fri 3/12/21	7	100%	Internet/Lab
3.3	Construct and Test Circuit in Lab	Josiah/Brendan	Wed 3/10/21	Sun 3/21/21	12	100%	Lab
3.4	Assemble Components to Rover and Test with Commands (Morse-Code)	Josiah/Brendan	Thu 3/25/21	Sat 4/03/21	10	100%	Lab
3.5	Finalize Design/Troubleshoot	Josiah/Brendan	Mon 4/05/21	Fri 4/16/21	12	100%	Lab
4	**Verilog Modules**		Wed 2/24/21	Tue 4/20/21	7	100%	
4.1	Research and Develop Module for Wall Avoidance/Simulate Waveform	Jacob/Brendan	Wed 2/24/21	Fri 3/12/21	17	100%	Internet/Lab
4.2	Research and Develop Module for Color Sensing/Simulate Waveform	Jacob/Brendan	Fri 3/12/21	Sun 3/28/21	17	100%	Internet/Lab
4.3	Research and Develop Module for Command Comm./Simulate Waveform	Jacob/Brendan	Wed 3/24/21	Fri 4/09/21	17	100%	Internet/Lab
4.4	Finalize Modules/Troubleshoot	Jacob	Wed 4/14/21	Tue 4/20/21	7	100%	Internet/Lab
5	**Other Important Tasks/Finalize**		Thu 3/18/21	Mon 5/03/21	7	100%	
5.1	Interim Report (3/29/21)	Team	Mon 3/22/21	Sun 3/28/21	7	100%	Internet
5.2	Printed Circuit Board (4/5/21)	Team	Mon 3/15/21	Mon 4/05/21	7	100%	Lab
4.3	Troubleshoot and Finalize Design (4/26/21)	Team	Mon 4/05/21	Mon 4/26/21	7	100%	Lab
4.4	Compile Final Report and Notebook (5/3/21)	Team	Mon 4/26/21	Mon 5/03/21	7	100%	Internet

Week 8 — 12 Apr 2021: 12 13 14 15 16 17 18 (M T W T F S S)

Week 9 — 19 Apr 2021: 19 20 21 22 23 24 25 (M T W T F S S)

Week 10 — 26 Apr 2021: 26 27 28 29 30 1 2 3 (M T W T F S S M)

Chart 8: Main-Project Final Gantt Chart

H. Budget

Mini-Project Budget	Running Cost			Cost Estimate			Start Date	1/25/2021
Direct Labor:	**Rate/Hr**	**Hrs**	**Total**	**Rate/Hr**	**Hrs**	**Total**	**Today**	2/22/2021
Brendan Blacklock	$ 15.00	60	$ 900.00	$ 15.00	50	$ 750.00		
Josiah Ramirez	$ 15.00	62	$ 930.00	$ 15.00	50	$ 750.00	**End Date**	2/22/2021
Jacob Holyoak	$ 15.00	53	$ 795.00	$ 15.00	50	$ 750.00		
Direct Labor Subtotal			$ 2,625.00			$ 2,250.00		
Labor Overhead (Rate: 100%)			$ 2,625.00			$ 2,250.00		
Direct Labor Total			$ 5,250.00			$ 4,500.00		
Contract Labor:	**Rate/Hr**	**Hrs**	**Total**	**Rate/Hr**	**Hrs**	**Total**		
Lab Tutor	$ 40.00	1	$ 40.00	$ 40.00	2	$ 80.00		
Dr. Clark	$ 200.00	0	$ -	$ 200.00	1	$ 200.00		
Contract Labor Total			$ 40.00			$ 280.00		
Direct Material Costs:			**Total**			**Total**		
(Refer to Material Costs Sheet)								
Direct Material Total			$ 239.80			$ 300.00		
Equipment Rental Costs:	**Value**	**Rental Rate**	**Total**	**Value**	**Rental Rate**	**Total**		
Oscilloscope	$ 5,300.00	0.20%	$ 296.80	$ 5,300.00	0.20%	$ 296.80		
Function Generator	$ 1,600.00	0.20%	$ 89.60	$ 1,600.00	0.20%	$ 89.60		
Power Supply	$ 958.00	0.20%	$ 53.65	$ 958.00	0.20%	$ 53.65		
DMM	$ 1,700.00	0.20%	$ 95.20	$ 1,700.00	0.20%	$ 95.20		
Equipment Rental Total			$ 535.25			$ 535.25		
Subtotal Cost:			$ 6,065.05			$ 5,615.25		
Business Overhead (Rate: 55%)			$ 3,335.78			$ 3,088.39		
Total Cost:			$ 9,400.82			$ 8,703.63		

Figure 43: Mini-Project Budget

Main-Project Budget	Running Cost			Cost Estimate			Start Date	2/22/2021
Direct Labor:	**Rate/Hr**	**Hrs**	**Total**	**Rate/Hr**	**Hrs**	**Total**	**Today**	4/30/2021
Brendan Blacklock	$ 15.00	118	$ 1,770.00	$ 15.00	200	$ 3,000.00		
Josiah Ramirez	$ 15.00	100	$ 1,500.00	$ 15.00	200	$ 3,000.00	**End Date**	5/3/2021
Jacob Holyoak	$ 15.00	111	$ 1,665.00	$ 15.00	200	$ 3,000.00		
Direct Labor Subtotal			$ 4,935.00			$ 9,000.00		
Labor Overhead (Rate: 100%)			$ 4,935.00			$ 9,000.00		
Direct Labor Total			$ 9,870.00			$ 18,000.00		
Contract Labor:	**Rate/Hr**	**Hrs**	**Total**	**Rate/Hr**	**Hrs**	**Total**		
Lab Tutor	$ 40.00	0	$ -	$ 40.00	5	$ 200.00		
Dr. Clark	$ 200.00	0	$ -	$ 200.00	2	$ 400.00		
Contract Labor Total			$ -			$ 600.00		
Direct Material Costs:			**Total**			**Total**		
(Refer to Material Costs Sheet)								
Direct Material Total			$ 290.61			$ 450.00		
Equipment Rental Costs:	**Value**	**Rental Rate**	**Total**	**Value**	**Rental Rate**	**Total**		
Oscilloscope	$ 5,300.00	0.20%	$ 710.20	$ 5,300.00	0.20%	$ 742.00		
Function Generator	$ 1,600.00	0.20%	$ 214.40	$ 1,600.00	0.20%	$ 224.00		
Power Supply	$ 958.00	0.20%	$ 128.37	$ 958.00	0.20%	$ 134.12		
DMM	$ 1,700.00	0.20%	$ 227.80	$ 1,700.00	0.20%	$ 238.00		
Equipment Rental Total			$ 1,280.77			$ 1,338.12		
Subtotal Cost:			$ 11,441.38			$ 20,388.12		
Business Overhead (Rate: 55%)			$ 6,292.76			$ 11,213.47		
Total Cost:			$ 17,734.14			$ 31,601.59		

Figure 44: Main-Project Budget