ECE 3331-303

Analysis of an Autonomous Rover

Jason Bissias

Sam Labarre

Lucas Bratton

Austin Ladd

Texas Tech University

Electrical and Computer Engineering Department

December 2023

Abstract

This report outlines the design, implementation, and results of the software and hardware for a Rover that can direct itself autonomously and the mechanisms involved in the arm it uses to pick up items.

Acknowledgements

My appreciations to our Professor, Dylan Tarter for his advice, patience, and understanding throughout this project.

Table of Contents

List of Figures

List of Tables

1. Introduction

The main objectives of this project were to create a Rover that can autonomously move visually along a track and pick up ferrous washers on temperature-identifiable stations.

2. Hardware

Much of the hardware of this project was custom designed and built, however some notable components involved the Digilent Basys3 FPGA board, an L-298 H-Bridge circuit, a uxcell 12V Electromagnet, an SG90 Servo, ZTP-135SR Thermopile, and a triad of Ximimark li18a3-8-z/bx 8mm IP Sensors.

2.1 Implementation

The root of this Rover is the Digilent Basys3 FPGA board, from which the entire system is controlled. The first outside connection is with the L-298 H-Bridge circuit via PWM signals out of its digital I/O ports to drive the motors. The H-Bridge circuit will be described briefly here, but for more details on this process refer to the prior report, *Robotics Mini Project* [1]. From there, the H-Bridge circuit is connected to a comparator circuit for overcurrent protection, which also has a connection to the Basys3 board. The input hardware for the Basys3 are an optocoupler circuit connected to the IP Sensors, a temperature sensing circuit using the Thermopile connected to a custom ADC, and a triad of IR sensors. The final outside connections the Basys3 has are to an electromagnet and servo upon the arm.

The H-Bridge circuit's core is the L-298 chip, which enables the Rover's motors' speed and direction based on the signals from the Basys3 board. The H-Bridge circuit additionally ensures that there are no faults by sending signals of both directions to the motors, which could burn them out. For further safety the H-Bridge circuit can interface with a comparator circuit.

This circuit uses an LM339N IC to send out a signal to the Basys3 board to stop the motors should the current drawn exceed ~1 A.

The next most crucial hardware elements are the IP Sensors and the optocoupler circuit that accompanies them. IPS stands for Inductive Proximity Sensor, which essentially allows us to detect metal. This is crucial to the objective, as the track is just a strip of metal stuck to the floor. The interesting thing about them, however, is that they generate a signal of at least 9 V when they are not detecting any metal, and they generate a 0 V signal when they do detect metal. This is interesting for two reasons: For one, the Basys3 board can only receive up to 3.3 V for logic, so we need to bring those 9 V signals down. Secondly, we must remember when programming the logic that the logic of the IPS is inverse to how we typically think. The latter consideration is simple enough, albeit confusing when initially developing the logic.

As per the challenge of bringing down the voltage, the solution lies with the optocoupler circuit. An optocoupler essentially allows for transmission of signals while keeping the source and the output isolated, thus rendering for a safe method of stepping down voltage. Figure 1 shows the design of our optocoupler circuit, which uses a PS2501-1 optocoupler chip for each IPS. The implemented design has the appropriate robustness for our current and voltage requirements. Therefore, we have three of these on one circuit board isolated from one another.

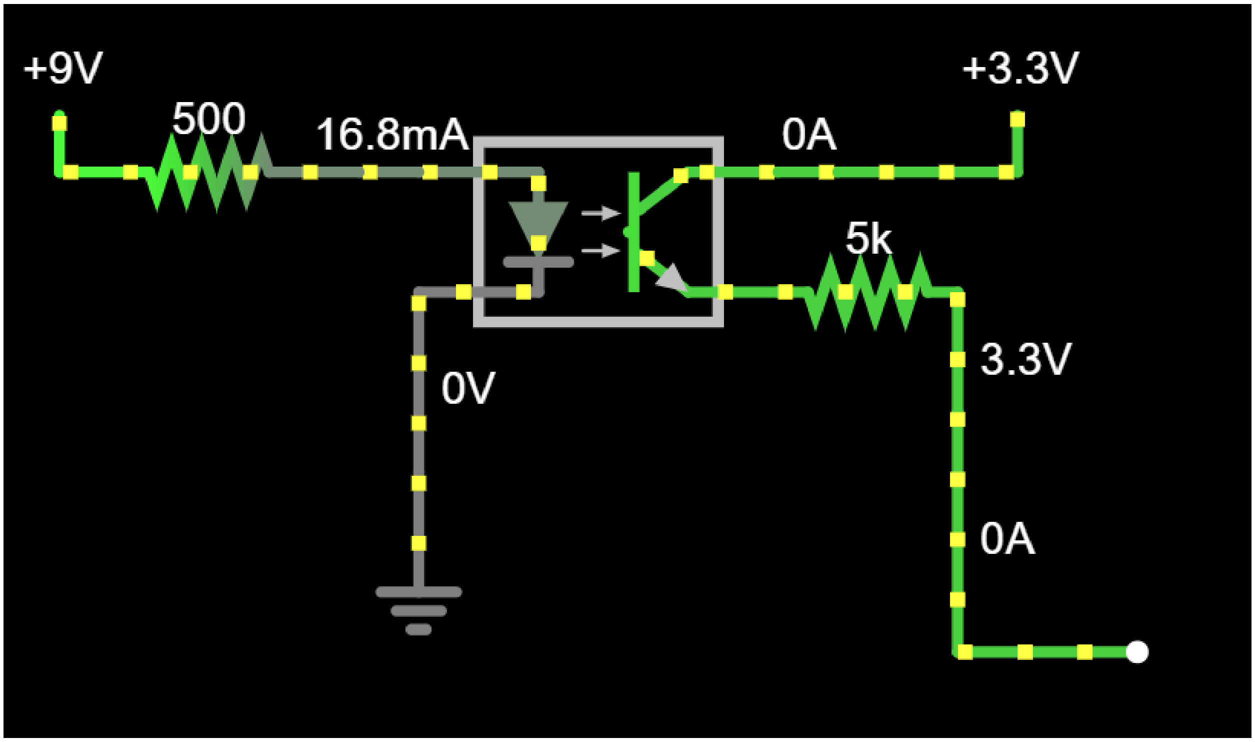


Figure 1: Optocoupler Circuit

The servo hardware is decently simple, meanwhile. The servo being used is an SG90, which is connected to a control circuit consisting of a flyback diode and an NMOS FET to prevent flyback voltage and to control the servo, respectively. The circuit diagram is in Figure 2. The gate pin is what connects to the Basys3's digital I/O ports. The servo will connect to the basket below the electromagnet shown in Table I to simply catch and provide the washers to the electromagnet as needed, depending on if it needs to pick up or drop off the payload respectively.

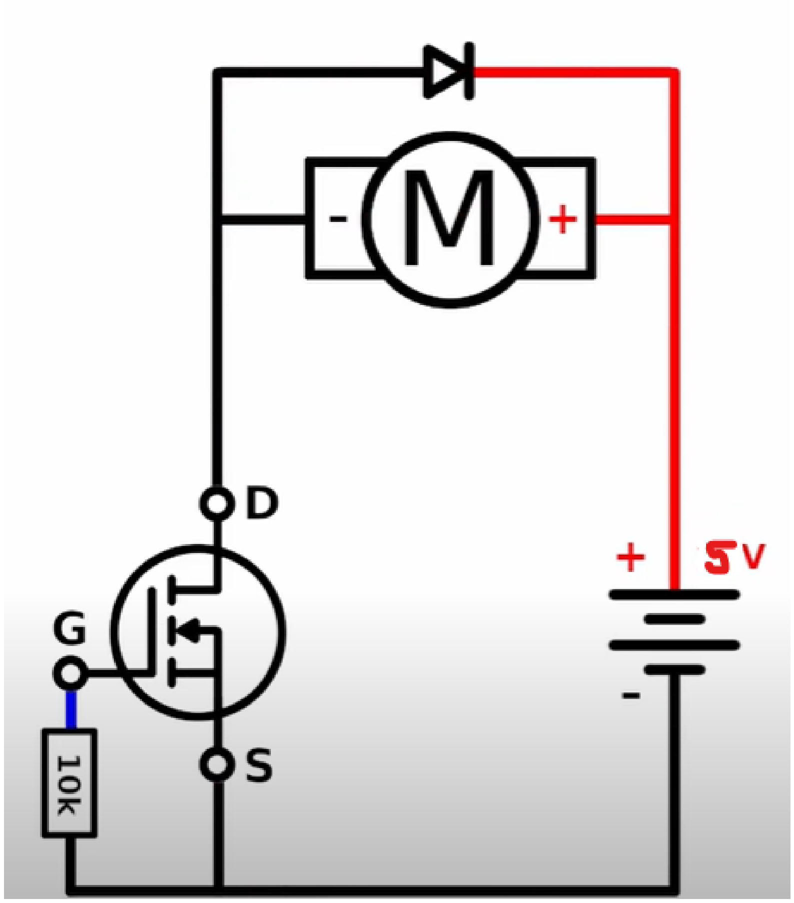


Figure 2: Servo Control Circuit

Temperature sensing was done via a ZTP-135SR Thermopile connected to a custom-built AD converter, as shown in Figure 3. The essential function of the circuit is to convert the granular values from the Thermopile to simple hot/ambient/cold digital values using two comparators. Displayed in Table I is the values necessary for each digital value.

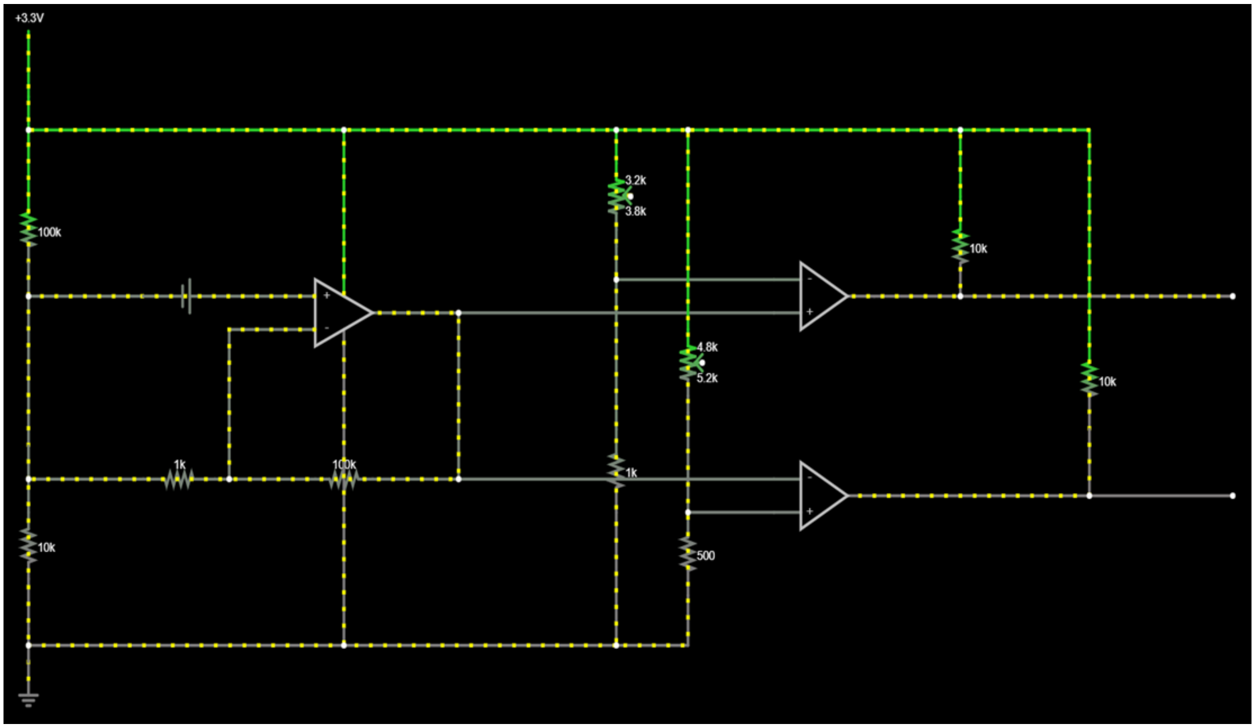


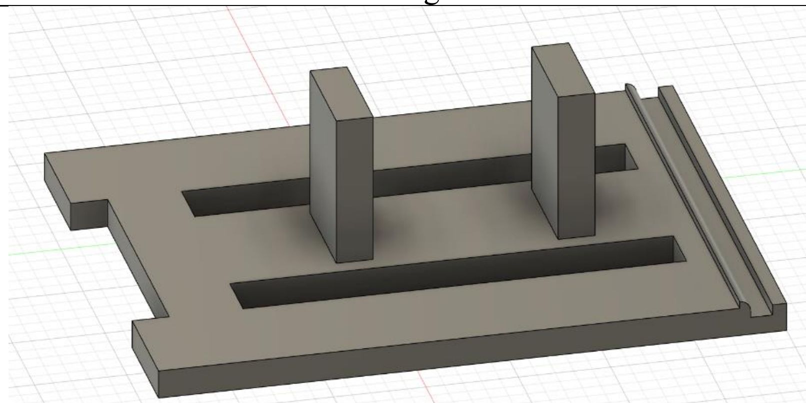
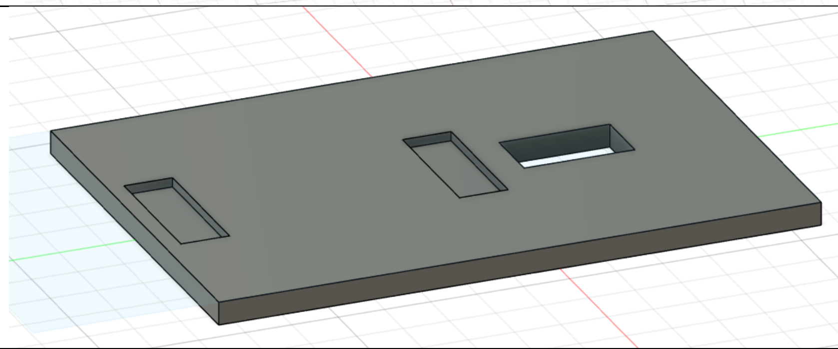
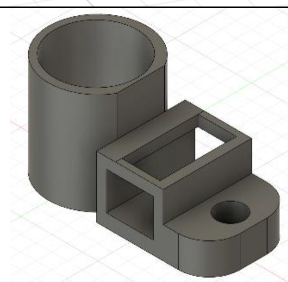
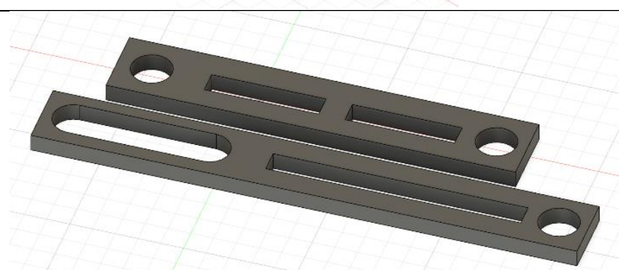
Figure 3: Thermopile and ADC Circuit Displaying Ambient Temperature

Temperature (°C)	Output Voltage (mV)	Top Comparator Logic	Bottom Comparator Logic	Basys Interpretation
15	-0.6175	0	1	Cold
16	-0.494	0	0	Ambient
34	1.014	0	0	Ambient
35	1.1375	1	0	Hot

Table I: Thermopile ADC Truth Table

To handle the washer, a 12 V electromagnet is used in conjunction with an aforementioned basket to hold onto the payload. Due to the behavior of the electromagnet, a flyback protection circuit was necessary to prevent current from flowing back and damaging other components as the magnet is turned off. The circuit is identical as the one in Figure 2, except that the input voltage is 9 V, the minimum needed to drive the electromagnet.

The final thing to consider is how everything fits on the chassis. Using CAD, we designed some structures to 3D print so that everything can fit and be mounted on our Rover. The 3D printed components include: two platforms that stack, an arm, a magnet holder, an IPS holder, and a few pieces of hardware to fasten everything together. Models of each of these are shown below in Table II and the total layout is shown in Figures 4 and 5.

Part	Design
Base platform	
Top platform (upside down to display joints)	
Magnet holder	
Magnet arm	

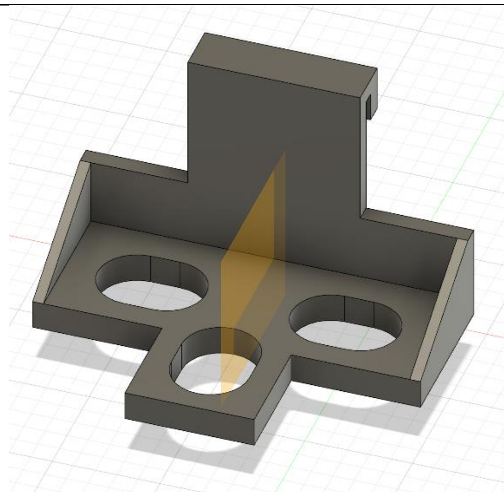
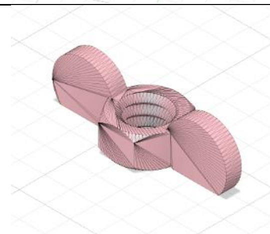
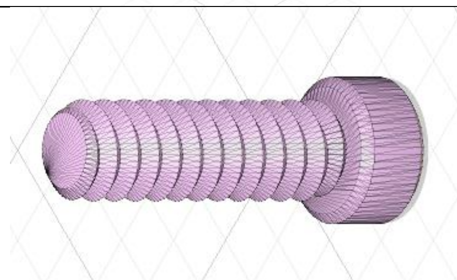
IPS holder	
Hardware: nut	
Hardware: bolt	

Table II: 3D Printed Structural Components

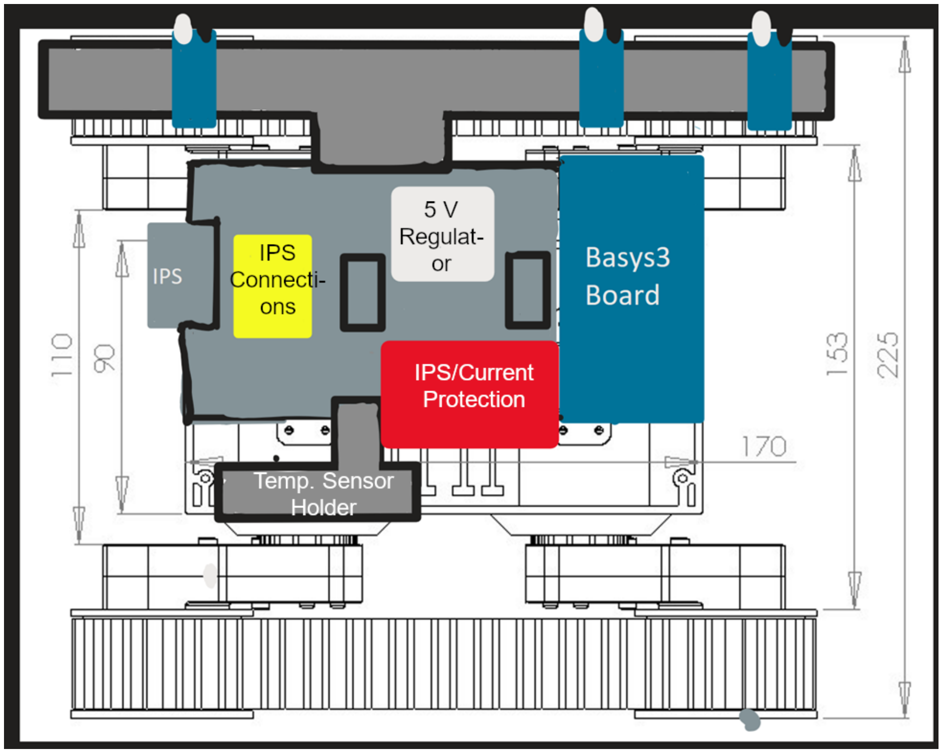


Figure 4: Layout of Hardware on Base Platform

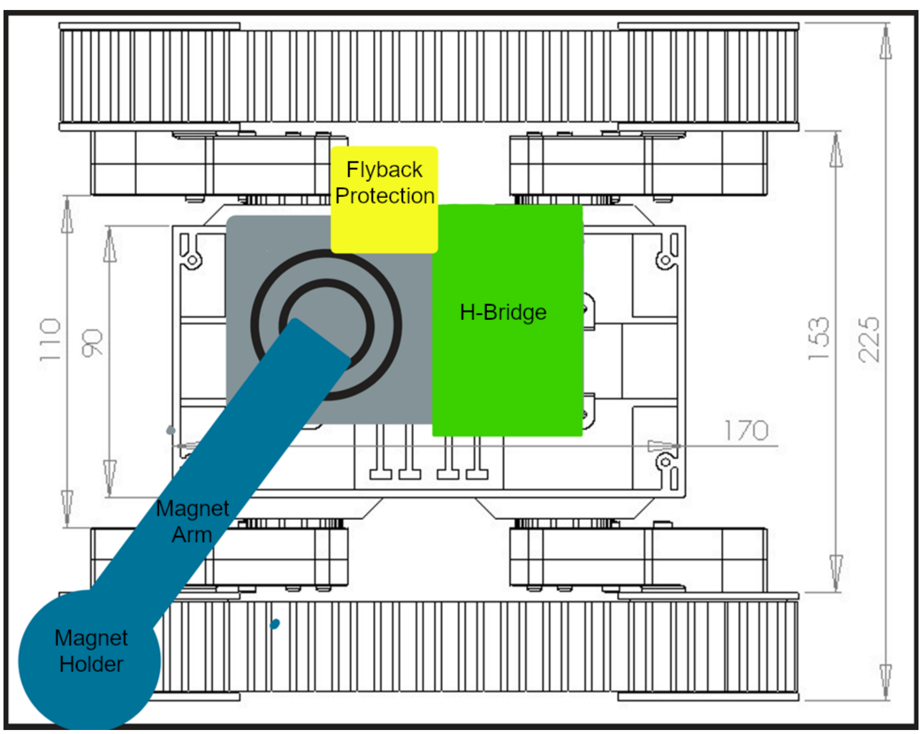


Figure 5: Layout of Hardware on Top Platform

2.2 Results

The hardware components work just as expected. The optocoupler outputs 0 or 3.3 V as needed as noted in Figure 6. In there, the red line is the output, which goes low when the yellow input goes high.

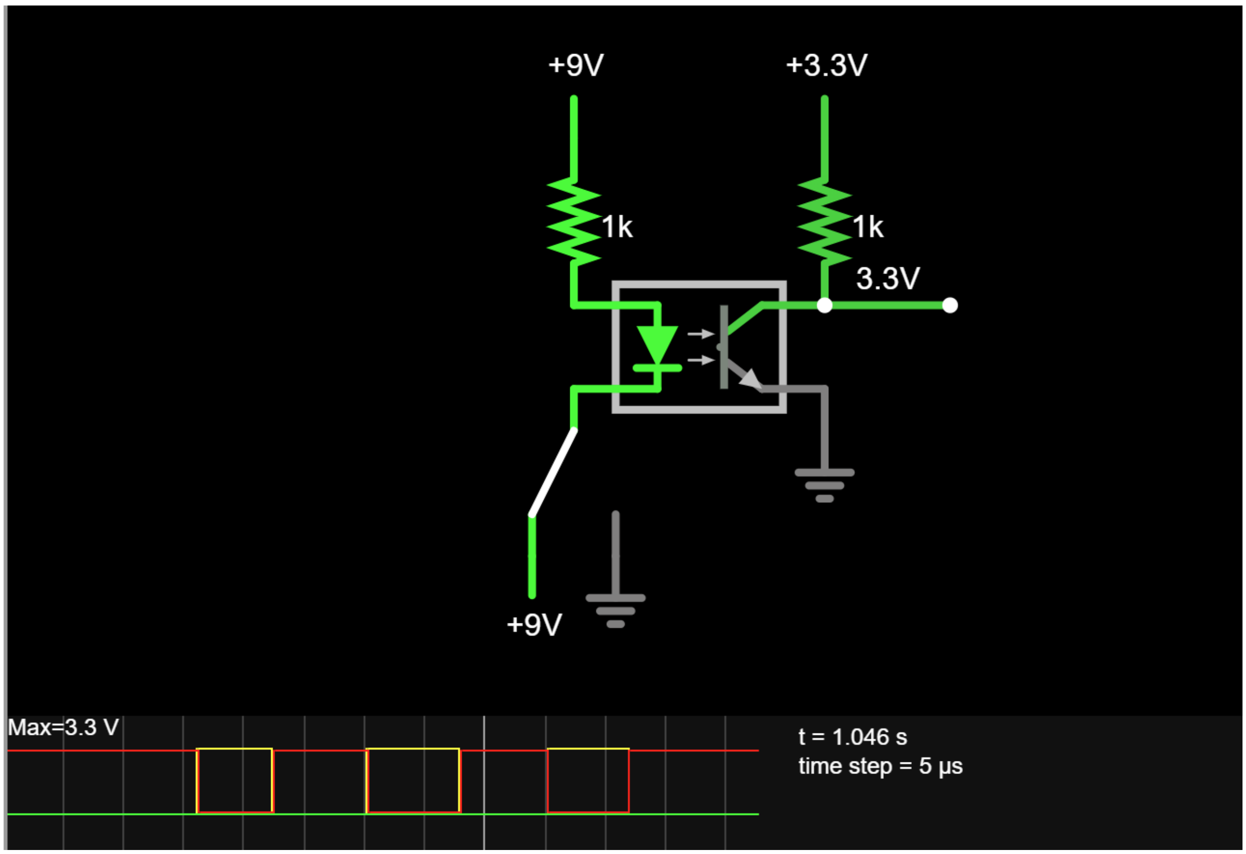


Figure 6: Optocoupler Circuit and Simulation Results

The comparator sends a signal when the current of the H-Bridge circuit exceeds 1 A, as visible in Figure 7; when the red line exceeds 1 A, the output (green) goes high, telling the Basys3 to stop driving the motors.

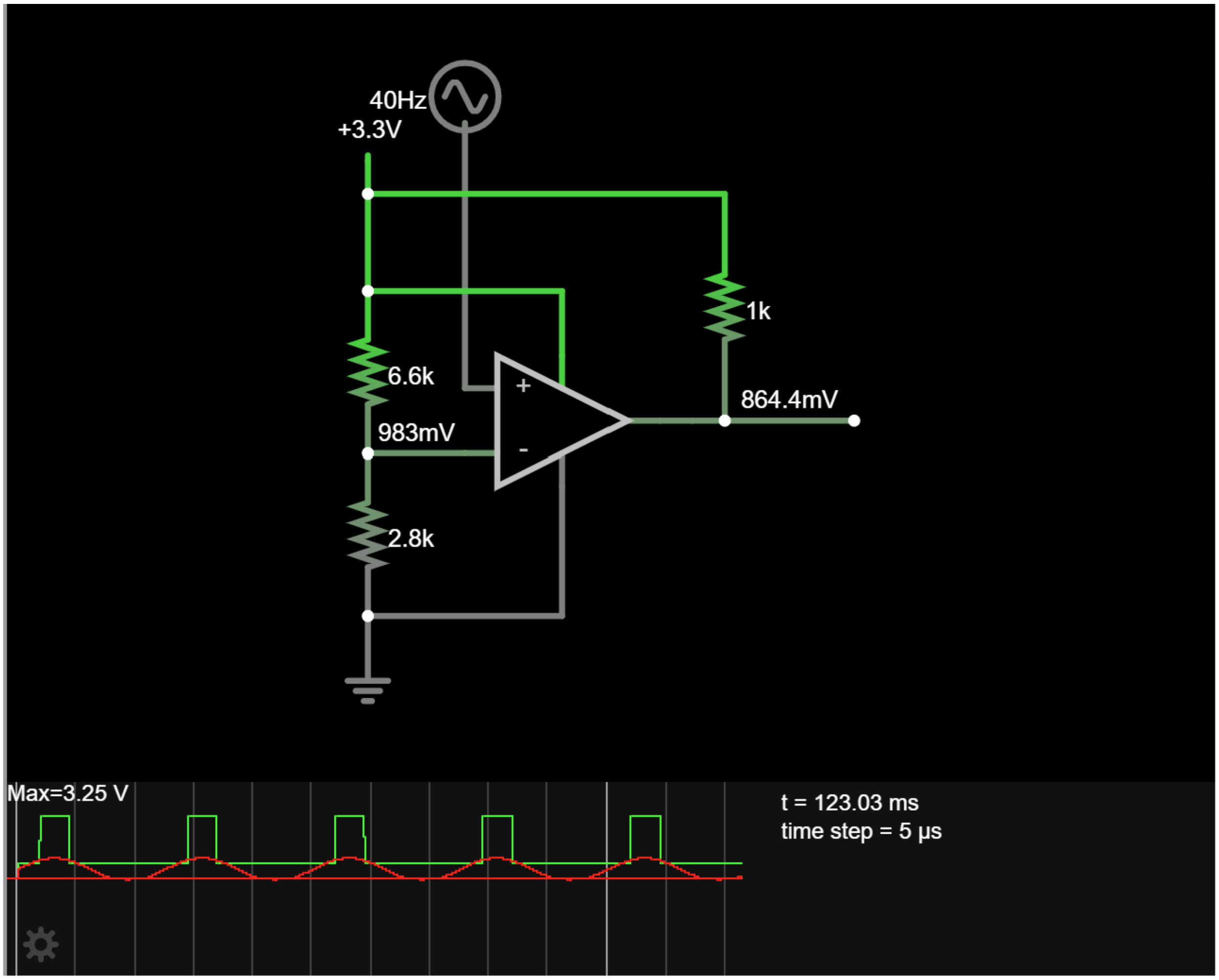


Figure 7: Comparator Circuit and Simulation

The electromagnet and servo's flyback diode circuit are shown working in Figure 8, wherein the red line is the input, yellow represents the current in the diode loop, and green shows the voltage in the diode loop. When the circuit is receiving an "on" signal, the voltage of the loop remains high after a moment of adjustment. When the circuit's input is off, the current ramps up through the diode but dissipate linearly, thus allowing the voltage to remain stable.

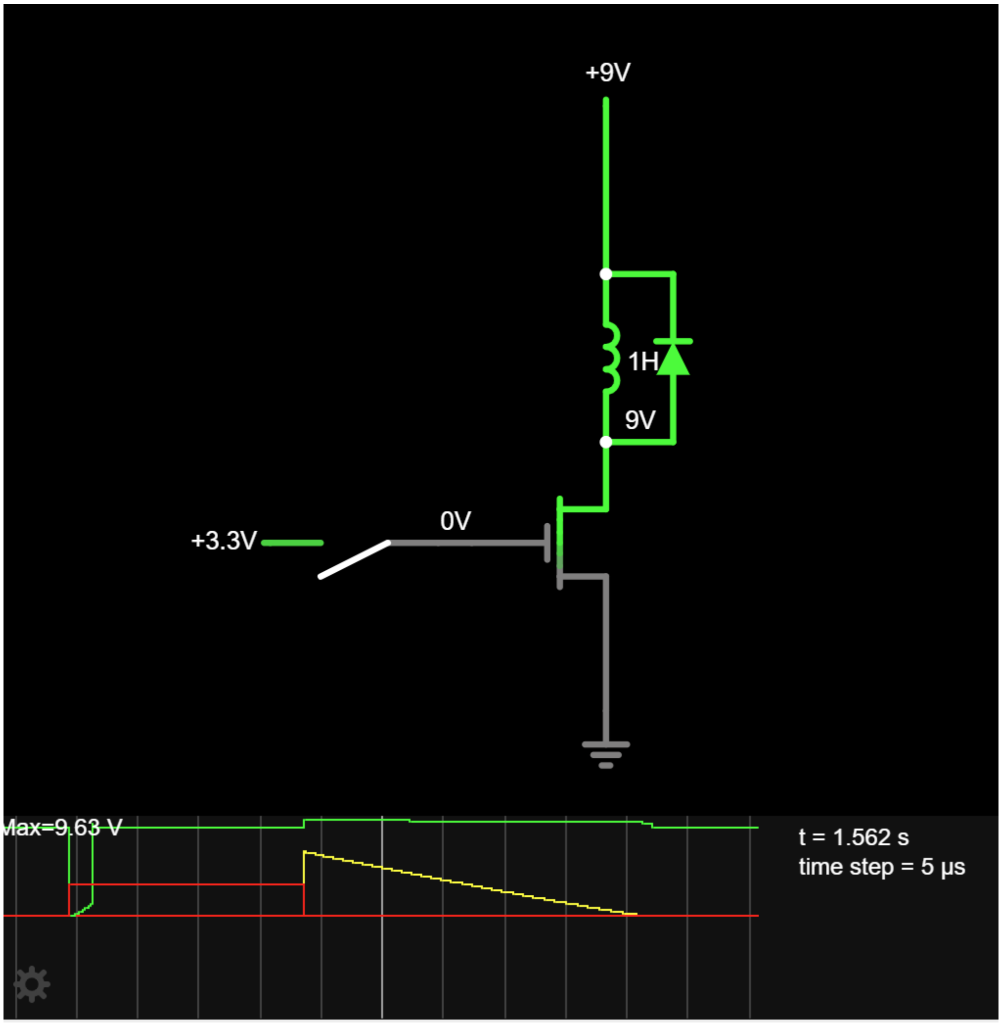


Figure 8: Flyback Diode Circuit and Simulation

3. Software

The software portion of the project was written entirely in Xilinx Vivado 2022.2 using the Verilog programming language.

3.1 Implementation

The final code of the project is comprised of five main modules, six total including the top module. For the movement of the rover, a “Movement”, “PWM”, “IR Sensing”, and “Temperature Sensing” module is present. Then, for the arm, there is a module that controls the basket's servo PWM signal and magnet actions in conjunction. Finally, a "Seven-Segment Display" allows us to see more of what the Rover is thinking.

The Movement module receives the signals from the optocoupler circuit and the Arm module, determines what action needs to be taken depending on the signals, and outputs a "Movement Code" for the PWM module to read. It can additionally maintain actions depending on if there is a feedback loop enabled. The loops have a numerical code so that the module can determine what action it is supposed to maintain. Another capability is that it can determine at tricky situations, for example an intersection, what the appropriate course of action is. A more concise explanation of these processes can be found in Figure 9 below. The Movement Code output is 12-bits long, 8 for the speed and 4 for the direction of each motor.

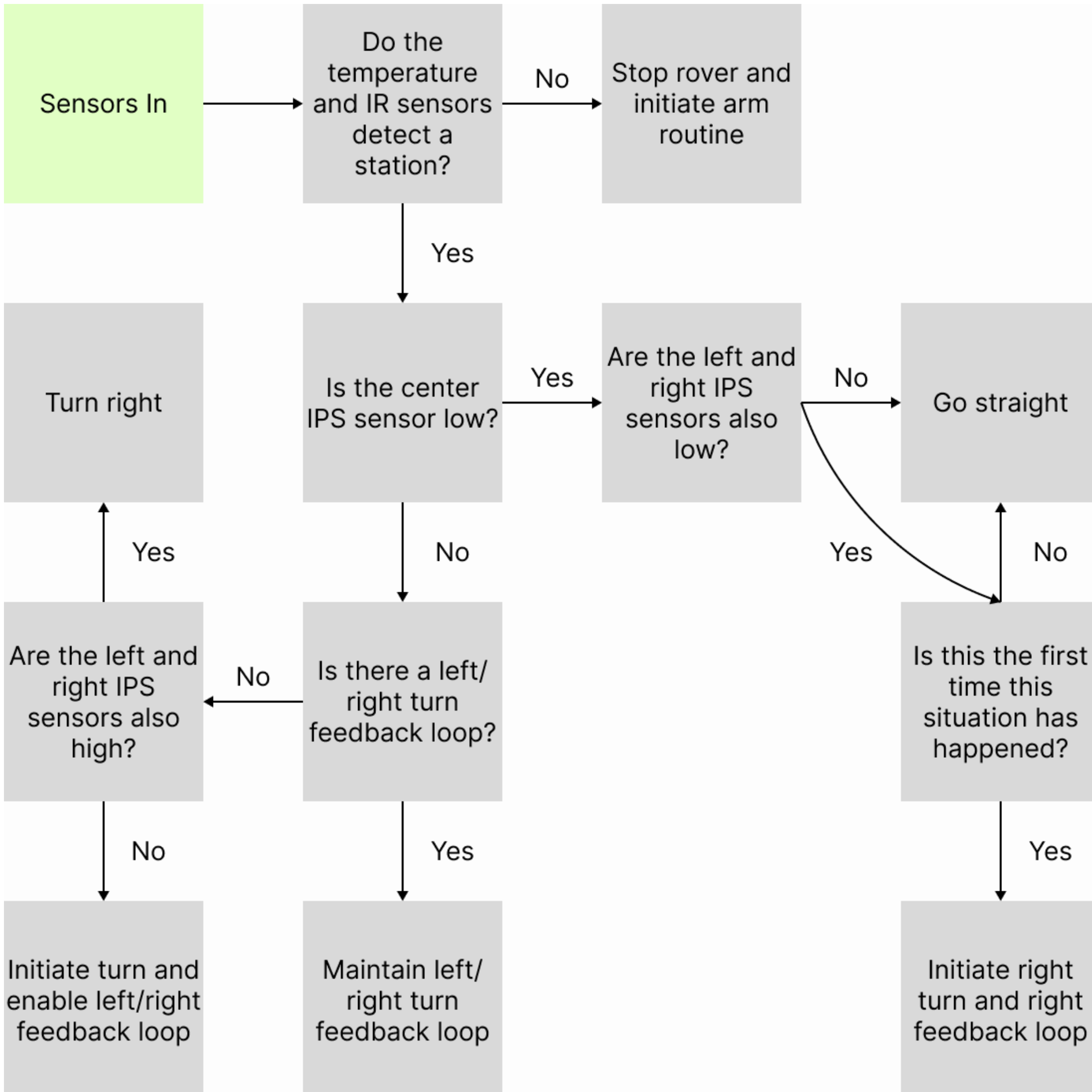


Figure 9: Flowchart of Movement Module

The Pulse Width Modulation, or PWM, module controls the signals to the motor using PWM signals. The signals are generated at various speeds based on the input bits from Movement Code 0 through 7. The speeds are processed using an internal clock of 480 Hz generated stepping down the main board's 100 MHz clock via a counter within the module.

Then, the control of which motor and which direction each motor achieves is based upon bits 8 through 11, two bits for each motor. The truth table in Table III shows the logic of the motor enabling and direction. Figure 10 shows the connections between the PWM module and the outside hardware.

Bit 11	Bit 10	Left Motor	Bit 9	Bit 8	Right Motor
0	0	Z	0	0	Z
0	1	Positive	0	1	Positive
1	0	Negative	1	0	Negative
1	1	Z	1	1	Z

Table III: Truth Table of Switches Controlling Motor Enabling and Direction

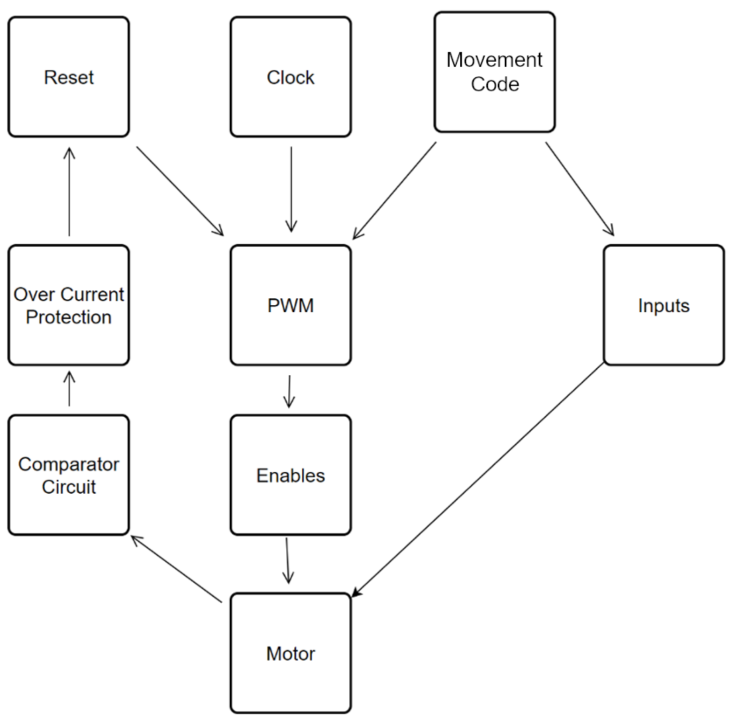


Figure 10: Block Diagram of Pulse Width Modulation Module with the Hardware

For the Arm module, it pulls double duty of controlling the basket servo's movement and the magnet's action. The servo's portion of the script are similar to the PWM module shown in the next paragraph and Figure 12, except there is only one output. For the magnet, the script also works by PWM signals so that the magnet gradually ramps up to full power. When "sw5" is high, the magnet is turning on; when "sw6" is on, the magnet is turning off. When either action is complete, the "done" variable turns on so that the Rover knows when it can begin moving again.

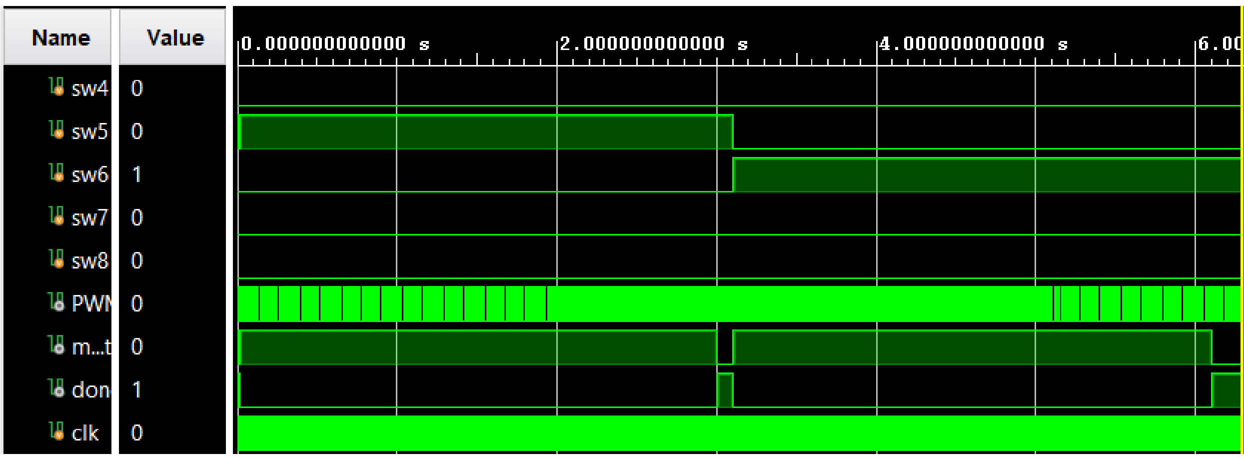


Figure 11: Electromagnet's Control Test Bench

The final piece of software to consider in this report is that of the servo for the electromagnet arm. The essence of that software is another PWM module, though this one only generates a PWM signal (unlike the module in the prior paragraph, which also generates direction via separate output signals). Servos determine what position they are turned to based upon the width of the pulse as compared to the total period. In our case, the total period is 50 Hz. The pulse widths necessary for neutral is 1.5 ms (which is 7.5% duty cycle, as in 7.5% of the total period), and 90° is 2 ms (10% duty cycle). An example of this is shown in Figure 12, wherein initially the servo is meant to be at 5% duty cycle before it turns to 2.5%. The actual PWM signal is in the "enable" signals.

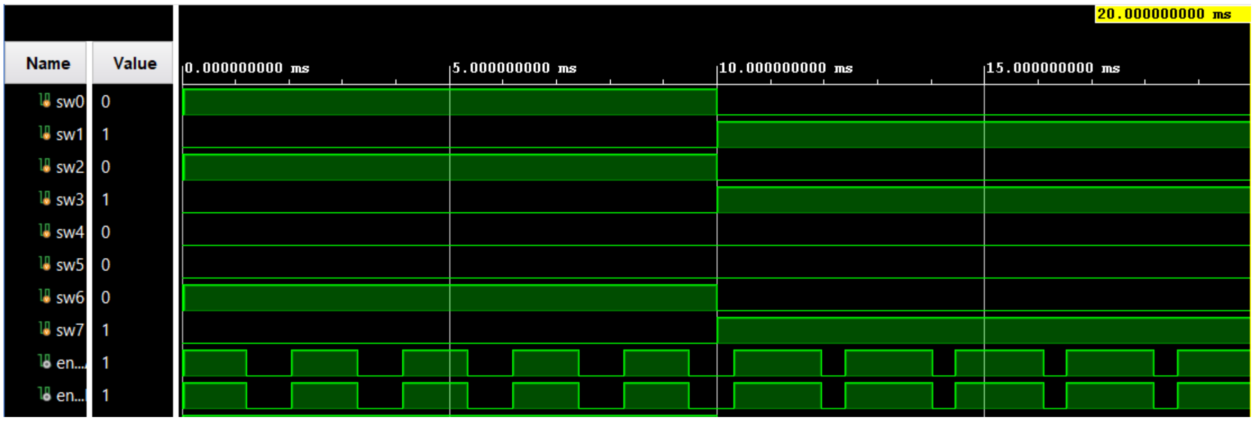


Figure 12: PWM for Servo Test Bench

3.2 Results

The software, as to be proven by the following test benches, works as designed. To start, the station sensing, which is comprised of the Temperature Sensing and IR Sensing modules, is shown in Figure 13. In it, the Rover only stops when the IR module outputs a 1 and when the desired & read temperatures match. The Rover begins moving again as the "stopFlag" variable goes low, which is triggered by the Temperature Sensing module but maintained by the Arm module. Lastly, at every valid stop, the module increments to the next desired temperature.

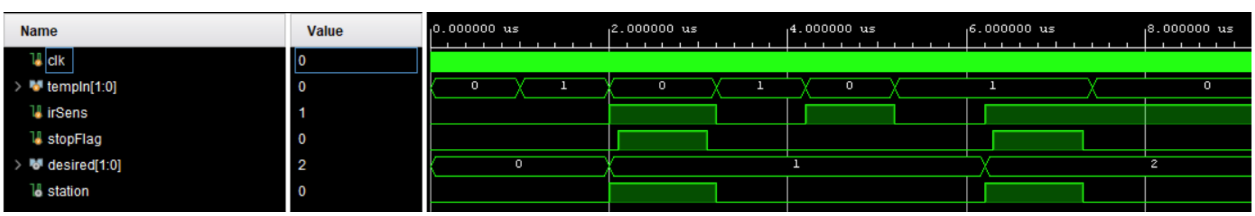


Figure 13: Station Sensing Test Bench

Shown below in Figure 14 is a simulated test bench of the Movement module, wherein a variety of situations occur. Of note, until 2 µs, the Rover maintains a straight path even though there is interference from the right IPS since the center IPS still senses a metal track. At 3 µs, the center IPS no longer sees a signal, therefore allowing the Rover to turn right. It maintains this motion until the center IPS turns low again at 5 µs, causing it to head straight. At 630 ns, the movement module receives a stop command from the magnet arm module while it is working. When the stop flag disengages, the Rover continues with its prior path. At about 8 µs, there is a small moment that all three IPS are low, hence it begins to turn right and “ledx”- which is supposed to signify if a crossover has occurred or not- goes high. The right turn is maintained until the center IPS is reading the track again, at which point it continues on straight at 920 ns.

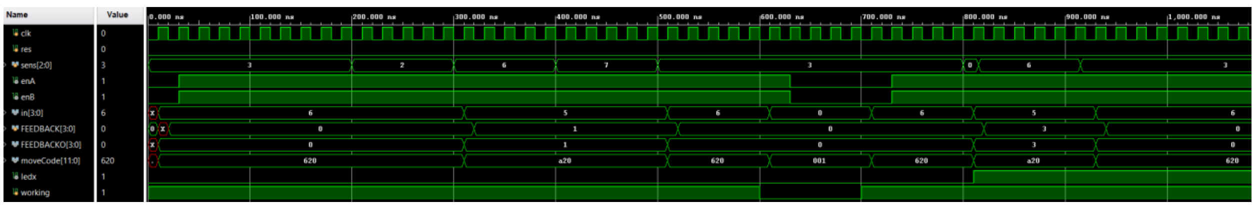


Figure 14: Movement and PWM Modules’ Test Bench

4. Combining the Hardware with the Software

The combining of the hardware and software components of our project is a seamless process thanks to the efficient and frequent communication between the two sides of the team. The main part of the integration of the two components is in a script entitled *final.xdc*. A *.xdc* file tells the Basys3 board what variables in the script(s) correlate to what physical I/O. A snippet of this file is presented in Figure 15. This example shows which I/O ports correspond to the “sens” register, which correspond to the IP Sensors.

```
##Pmod Header JA
#Sch name = JA1
set_property PACKAGE_PIN J1 [get_ports {sens[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sens[0]}]
#Sch name = JA2
set_property PACKAGE_PIN L2 [get_ports {sens[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sens[1]}]
#Sch name = JA3
set_property PACKAGE_PIN J2 [get_ports {sens[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sens[2]}]
```

Figure 15: A Brief Look Into *final.xdc*

5. Conclusion

In total, the Rover can complete the tasks assigned to it. All the hardware components are functional and safe while the software can handle both predicted and unpredicted situations. The only insufficiency of the Rover is in the action of picking up the payloads. The magnet needs to be extremely close to the washers to pick them up, which posses the risk of being thrown away by the swinging basket if too low, or that the magnet just does not have the power to pick it up if it is too high compared to the platform. Otherwise, the systems work according to the designs, which manage to complete the course and stop as needed at the stations.

References

1. Jason Bissias, Sam Labarre, Lucas Bratton, Austin Ladd. “Mini Project Report,” https://jbissias.xyz/pdf/JasonBissias_Mini_Project_Report.pdf, September 27, 2023

Appendix A

Considerations

Though our project is a one-off production of a low-power and low-footprint device, there are still some considerations we as a group must keep in mind for the safety and wellbeing of ourselves and others.

1. Safety, Public Health, and Welfare Considerations

When building our project, we had to solder some components together. Although a relatively low-danger process, proper precautions must still be taken. With any soldering, appropriate PPE was always worn, and we aimed to keep our stations well ventilated. Additionally, we made sure our work area was clean to avoid any trips, cuts, or other bodily damage. We were always careful to keep exposed wires away from one another so as to avoid problematic short circuiting, or completely cover them in insulating material, whenever possible. We stored batteries safely and kept mind of our settings on power supplies. Even with our precautions, however, errors were unfortunately made. When errors were made, we made sure to quickly remedy the situation with the appropriate methods. For example, when a set of our batteries began leaking, they were promptly disposed of in the appropriate manner with the leakage cleaned off of any place it had touched.

2. Global, Cultural, Social, Environmental, and Economic Factor Considerations

Our project faces a miniscule impact in global, cultural, social, environmental, and economic issues. The inefficiencies of our project are minimal, thanks to our usage of rechargeable batteries and usage of as few components as possible.