Abstract

The M.O.T.O.R-T initiative delves deep into the world of autonomous robotics, featuring a self-guided rover that traverses a metallic route, shifting metal items among distinct processing zones. This documentation delves into the nuanced decisions regarding hardware and the complexities of automation, inclusive of 90-degree maneuvers and intersectional pathways. The expedition of the rover commences with the acquisition of a green washer and encompasses visits to a molding facility, a hardening outpost, and a concluding product station. The document elucidates the rover's deployment of a thermopile sensor for discerning stations and obtaining instantaneous temperature data. The harmonization of detectors and automated operations exemplifies cross-functional innovation within the bustling foundry landscape.

Contents

List of Figures

List of Tables

1. Introduction

The M.O.T.O.R – T initiative seeks to transform material transportation and processing in foundries by harnessing the power of automation and robotics. Conventional manual techniques, burdened by labor intensiveness, inefficiencies, delays, and safety hazards, are reimagined with this venture. As a solution, an autonomous rover is designed to streamline transportation within the foundry. By embracing robotic technology and automation, the venture augments operational proficiency, curtails overhead costs, and lessens potential dangers, establishing itself as a trailblazing effort in updating foundry procedures. This completed endeavor employs a strategic framework, integrating a rover framework, Basys-3 development platform, H-Bridge, ZTP-135SR Thermopile for temperature feedback, and sensors for precise navigation. The system also integrates a magnetic arm for adept material management and fortified hardware safeguards for optimal motor and battery preservation. This initiative merges pioneering engineering with state-of-the-art automation, challenging conventional manufacturing norms, and underscoring the transformative role of robotics in confronting industrial hurdles. In doing so, it sets the stage for a more secure, efficient, and forward-thinking manufacturing paradigm.

2. Hardware

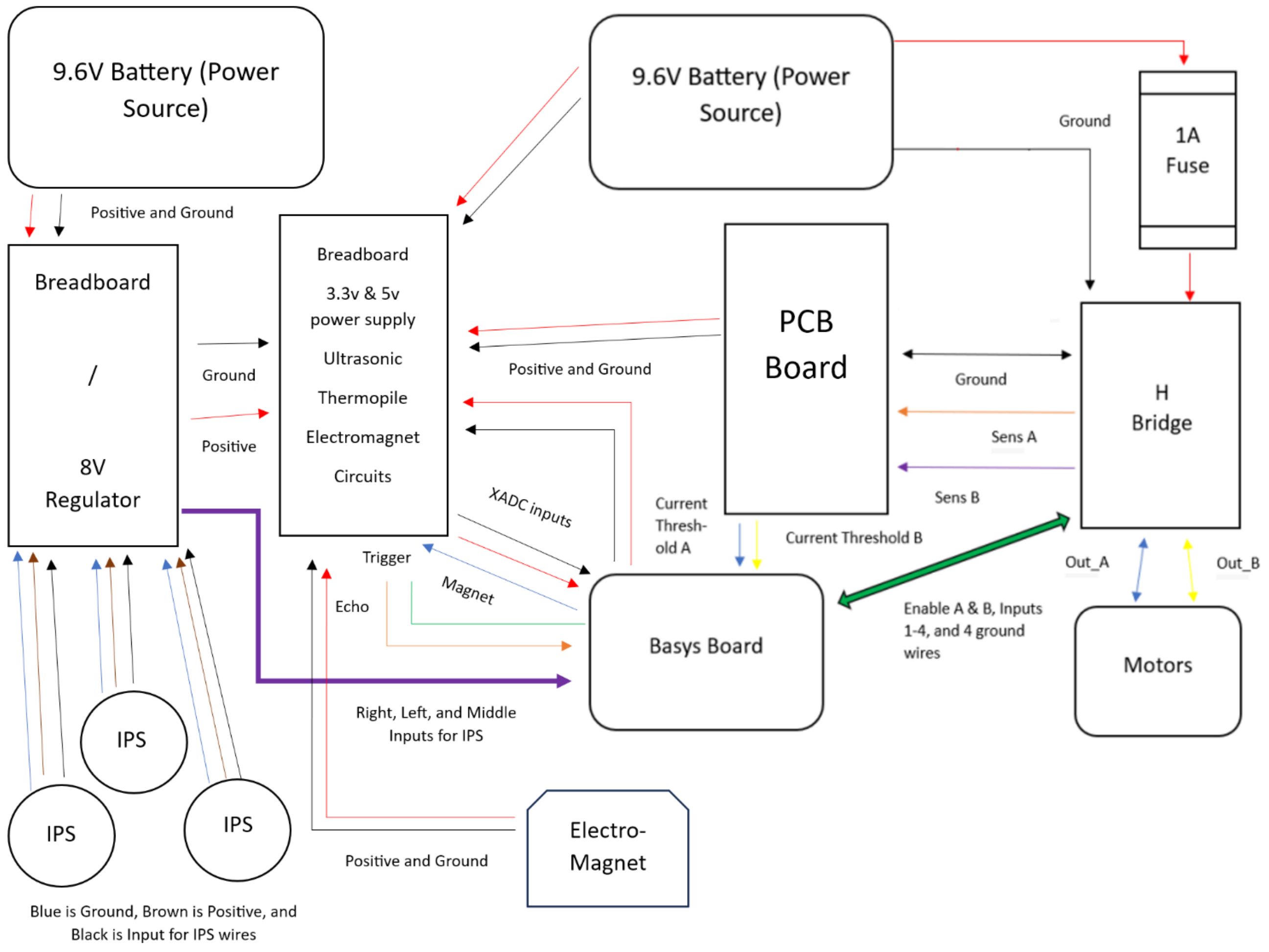


Figure 1: Hardware Design Schematic

2.1 Hardware Overview

The hardware architecture integrates vital elements, such as two comparator circuits, dual power sources, voltage stabilization mechanisms, overcurrent safeguards, an H-bridge command module, motor assembly, three inductive proximity detectors, and an ultrasonic detector. Starting with one of the two 9.6V batteries, the power is split between two sets of cables to cater to different components. A breadboard-fitted voltage regulator offers both 3.3V and 5V outputs tailored for the Basys board, guaranteeing the right voltage gradients for data relay and overcurrent shield. The inductive proximity detectors draw power from a distinct battery connected to an 8V stabilizer.

Simultaneously, both the ultrasonic detector and the thermopile circuit are energized by the same breadboard, assuring smooth functioning and dependable data communication across all units.

2.2 Power Source & Regulation

The system's power distribution is pivotal and utilizes two 9.6V batteries. The primary battery is linked to an H Bridge and a specialized breadboard furnished with a voltage regulator set at 5V and 3.3V. The 5V energizes the Basys board, while the 3.3V is channeled to the output of the comparator circuit connected to the PMOD ports of the Basys Board. This bifurcated regulation caters to the 5V demand of the Basys board and the 3.3V specification of the PMOD ports. Additionally, it powers both the ultrasonic sensor and the thermopile circuit, which mandate a consistent 5V supply. The secondary battery is solely reserved for the trio of inductive proximity (IP) sensors, operating within a 6-36 volt range. An 8-volt regulator maintains a uniform voltage. These regulators guarantee operations within specific voltage boundaries, bolstering the system's dependability and efficiency.

2.3 Motor Control & Fuse Implementation

The H-Bridge's primary purpose is to enable precise bidirectional spinning of the DC motors while providing variable speed control. The H-Bridge accomplishes this by allowing the current to flow through the motors in either direction, determining the direction and speed of the motor's rotation. By manipulating the Basys switch signals, it creates a path for current to flow through the motors. When current flows in one direction, the motor rotates in one direction, and when the current is reversed, the motor's rotation changes accordingly.

As the H-Bridge is a high-power component, protecting it from overcurrent conditions is paramount to ensure its longevity and prevent damage. One of the primary safeguards is the 1A fuse, strategically placed in the circuit before the H-Bridge. This fuse acts as a critical barrier against excessive current. If the current surpasses the 1A threshold, the fuse promptly interrupts the circuit, preventing any damage to the H-Bridge and the associated components. Additionally, the comparator circuit, as previously described, is instrumental in monitoring the current flow in real-time. The voltage output from the HBridge is continuously compared to a reference voltage. Any deviations from the expected current values are detected, allowing for swift corrective action.

With robust overcurrent protection mechanisms in place, including the 1A fuse and real-time current monitoring, the H-Bridge is shielded from potentially damaging current surges. This ensures its reliability and longevity while providing the motors with a controlled voltage for precise motor control, making it a critical component for achieving the system's objectives in motor manipulation and automation.

2.4 Comparator Circuit & Overcurrent Protection PCB

The hardware configuration features a comparator circuit from Texas Instrument, specifically the LM339N model. This circuit contains four distinct voltage comparators, but only two are utilized, each dedicated to a motor. The comparator assesses an input voltage against a reference voltage, producing an output when the input surpasses the reference. The LM339N was segmented into two comparator circuits, and their performance was emulated using LTspice software. Following the successful implementation of the LTSpice schematic, it was then recreated in EasyEDA software, a step that facilitated the transition from a virtual design to a physical printable circuit board (PCB).

A power source of 3.3V is partitioned with the assistance of resistors (R1 and R2) to generate a 1V voltage benchmark. This benchmark plays an essential role in contrasting sensor readings from the H Bridge. Connections, labeled as Sens A and Sens B, tie the H Bridge to the breadboard, providing 'V' with input values. There's also a 1 Ohm sensing resistor placed between Sens B and the ground on the H Bridge. This setup ensures the comparator output activates when the current surpasses 1A.

The comparator's resultant output is connected to the 3.3V power supply through a 1.1k Ohm resistor. This output then links to the PMOD ports on the Basys board, signaling when the current exceeds the 1A limit. The circuit's operation is verified by LTSpice simulations, displaying an output shift from 3.3V to slightly over 0V when the input voltage is less than 1V.

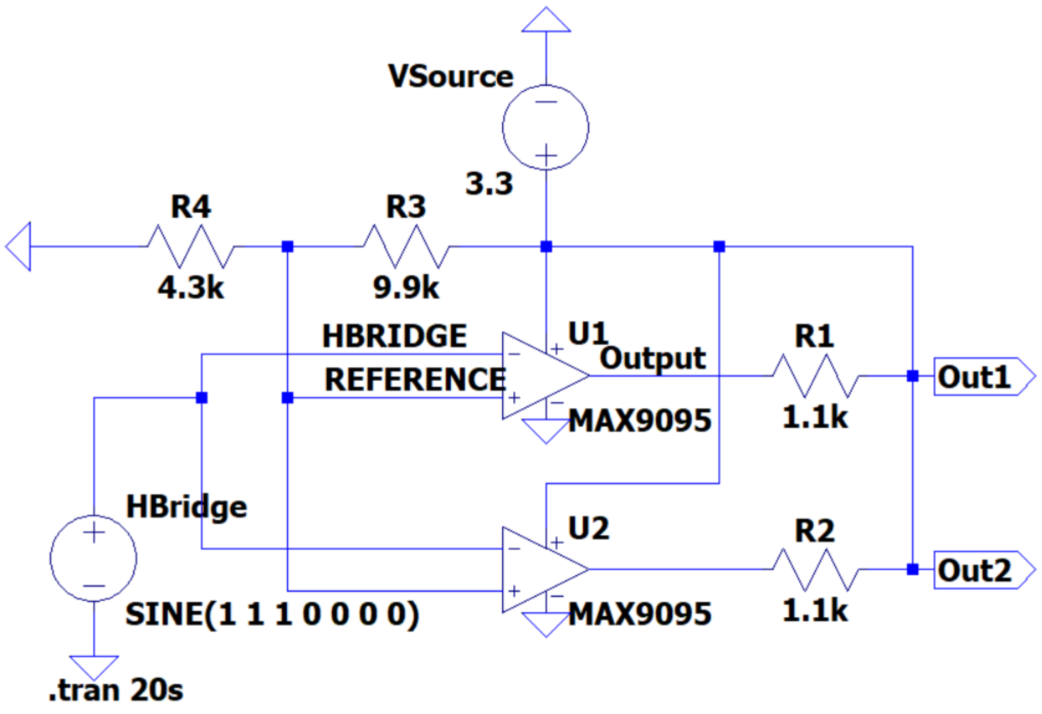


Figure 2: Comparator Circuit Schematic

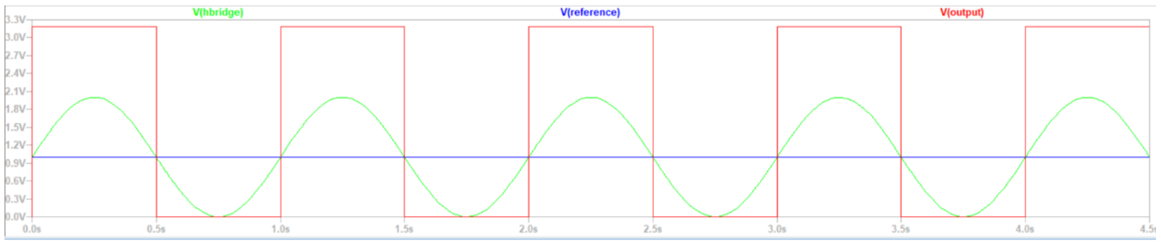


Figure 3: LTSpice Simulation of Comparator Circuit

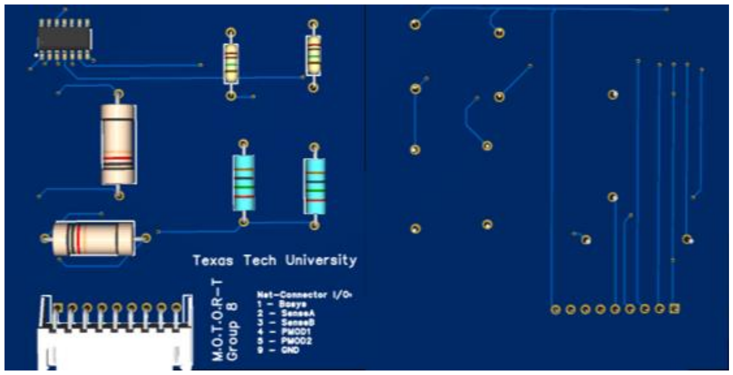


Figure 4: PCB Front/Back Preview

2.5 Inductive Proximity Sensor

In the heart of the rover's capability to navigate within its metallic track confines lies the Inductive Proximity Sensor (IPS) system. These sensors play an instrumental role in guiding the rover, ensuring it aligns seamlessly and recognizes its relative positioning to the track's boundaries.

The provided LTSpice schematic illustrates the precise electronic design underpinning the function of the IPS. At the outset, the IC-T808T 8V regulator effectively steps down the power from the 9.6V battery to serve as the voltage source for the entire circuit. This 8V feeds into a series of capacitors, namely C1 and C2, designed for noise suppression and ensuring a smoother voltage feed into subsequent stages.

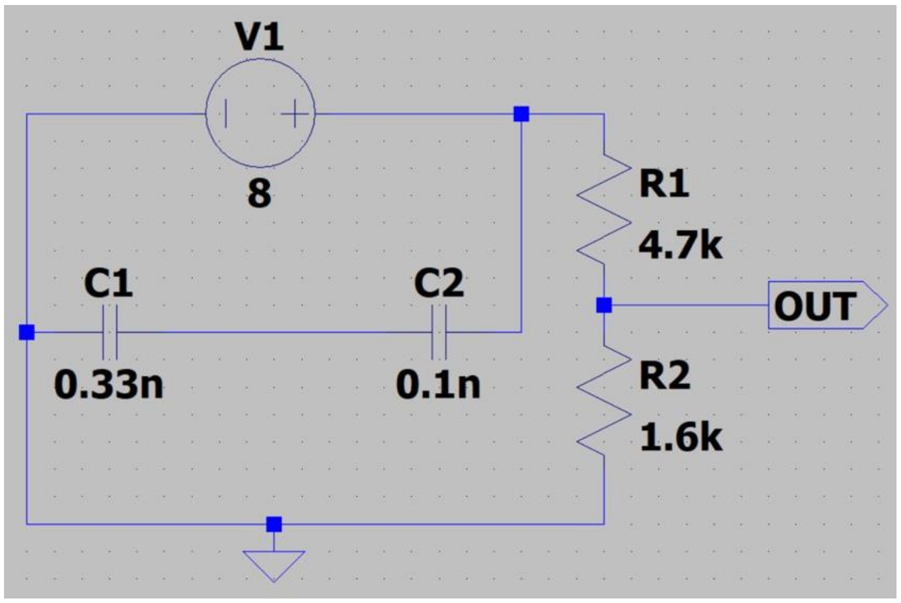


Figure 5: LTSpice Inductive Proximity Sensor Schematic

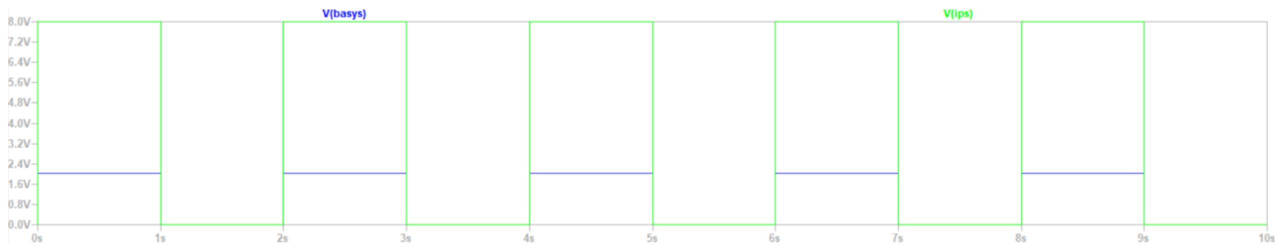


Figure 6: LTSpice Simulation of a Single IP Sensor

Subsequently, the circuit involves a pivotal voltage division operation. Using resistors R1 and R2, the circuit achieves a near-exact output value of 2V. This precise voltage division ensures that the IPS operates within its optimal voltage range, thereby guaranteeing accuracy in its detections. What enhances the design's versatility is its scalability. The same voltage division mechanism has been replicated twice more, enabling the entire breadboard system to output three identical near-exact 2V values for each IPS unit. This approach is instrumental in powering the three distinct IPS units used in the rover's design.

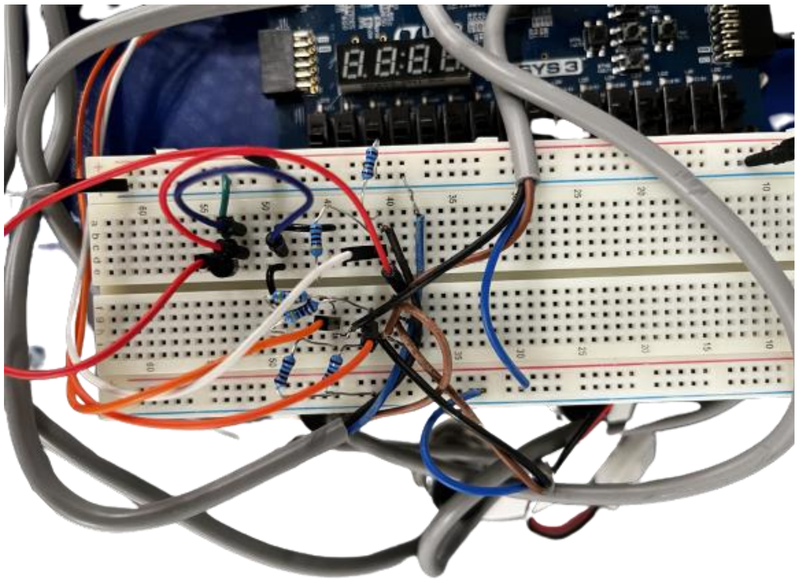


Figure 7: Inductive Proximity Sensor Circuit

The three IPS sensors are positioned ingeniously on a custom-designed 3D printed holster. Configured in a triangular shape, the left and right sensors are suspended slightly below the central top sensor. This arrangement is strategic, offering a broader detection range and improved sensitivity to any alignment deviations.

The thoughtfully crafted IPS circuit, combined with its strategic placement on the rover, ensures the autonomous vehicle remains attuned to its environment. Through noise suppression, precise voltage regulation, and an efficient voltage division mechanism, the hardware offers consistent, reliable input about the rover's position relative to the metallic track. It is a prime example of how meticulous hardware design, when combined with creative engineering solutions, can lead to groundbreaking advancements in industrial automation.

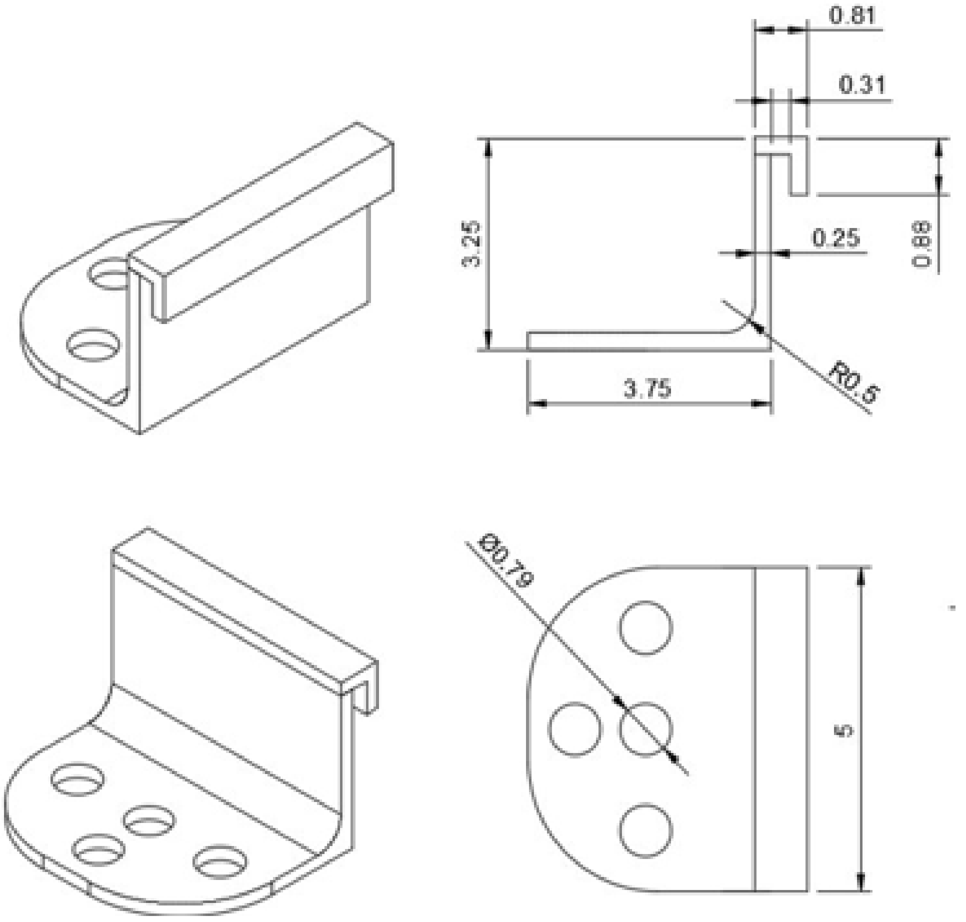


Figure 8: IPS Holster Preview

2.6 Ultrasonic Sensor

Ultrasonic sensors have emerged as indispensable tools in various fields due to their ability to detect and measure distance without direct contact. In the context of your project, the HC-SR04 Ultrasonic sensors calculate on the principle of sound wave reflection to detect when a base station is present. These sensors emit ultrasonic sound waves, which, when obstructed by an object or obstacle, bounce back to the sensor. By calculating the time taken for these waves to return, the sensor determines the distance between itself and the object.

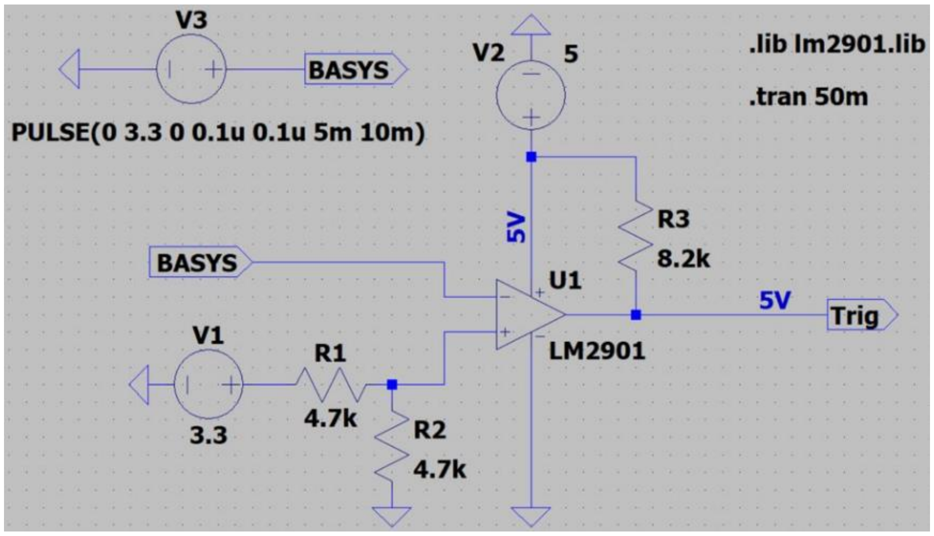


Figure 9: Ultrasonic Circuit Schematic

The provided LTSpice schematic illustrates the circuitry behind this operation. At the core of the schematic is the LM2901 comparator chip. A comparator essentially compares two voltage inputs and switches its output based on which input is higher. A constant 3.3V is fed into a voltage divider composed of two 4.7kΩ resistors which ensures a precise voltage reference for the comparator's positive input. The Basys board provides a 3.3V square pulse, which is given to the negative input of the comparator. Whenever the pulse from the Basys exceeds the reference voltage from the divider, the comparator toggles its output. The comparator's output is then conditioned using an 8.2kΩ resistor connected to a 5V source. This signal is fed into the 'Trig' pin of the HC-SR04 sensor. Upon triggering, the sensor emits ultrasonic waves from both the echo and trigger pins, shown from the simulation reading.

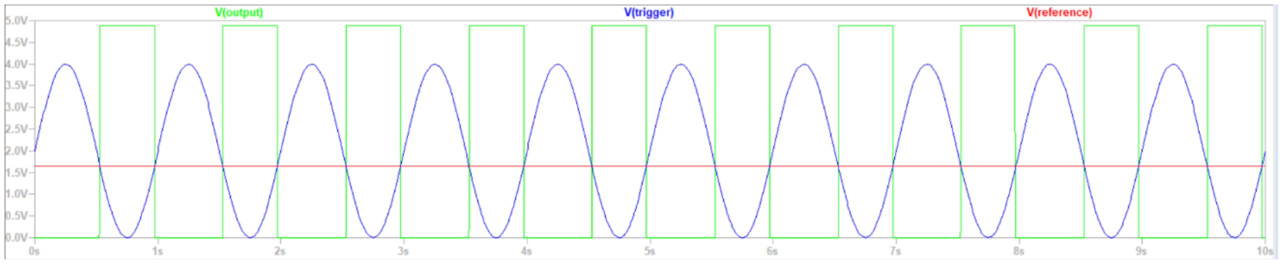


Figure 10: LTSpice Simulation of "Trigger" Comparator

The HC-SR04 is mounted on a specially designed 3D printed base, shared along with the thermopile circuit, which allows for seamless integration with the rover's station. This base ensures that the sensor has an unhindered view of the surroundings, optimizing distance measurement. The 3D printed base for the ultrasonic sensor not only provides structural support but also ensures optimal positioning for the sensor. Given the sensor's reliance on sound wave reflection and the thermopile's readability from range, its positioning and angle can significantly impact its accuracy. The bespoke design of this base ensures that the HC-SR04 can function at its best, enhancing the rover's navigational capabilities.

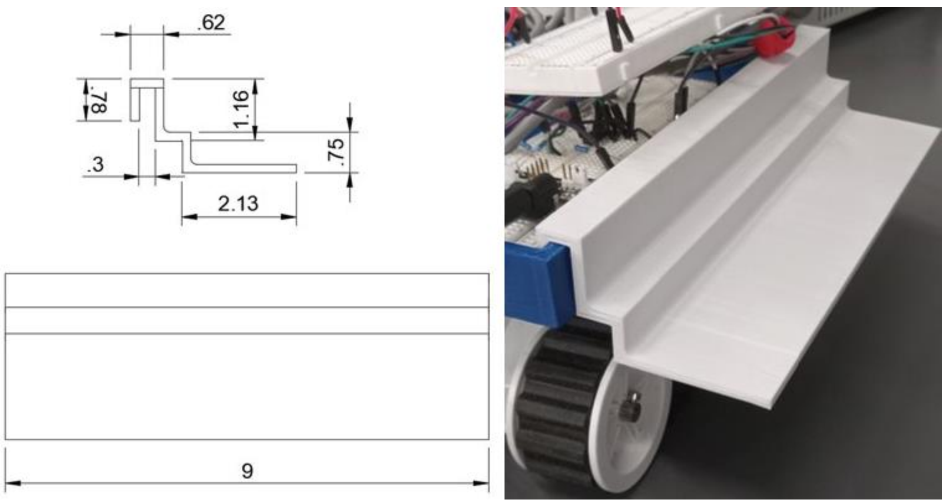


Figure 11: Ultrasonic/Thermopile 3D Printed Base Sample

The physical implementation on the breadboard mirrors the LTSpice schematic, as tested from the Tektronix TDS 2002C oscilloscope reading. It's crucial to ensure that connections on the breadboard are accurate to avoid any discrepancies in real-world performance. Given the intricacies of such circuits, careful attention to detail during the breadboarding phase can prevent potential issues during field tests. Once the ultrasonic waves are reflected back and received by the sensor, the time taken for this round trip is processed by the Basys logic. This logic then converts this time into a meaningful distance, which is used to determine whether the rover should halt or continue its journey.

Ultrasonic sensors, particularly the HC-SR04, are invaluable in applications requiring non-contact distance measurements. Through the careful design of the comparator circuit and the strategic use of the 3D printed base, this system promises reliable performance in the rover's navigation. The seamless integration of hardware (breadboard circuitry) and software (Basys logic) underscores the importance of interdisciplinary collaboration in modern engineering solutions.

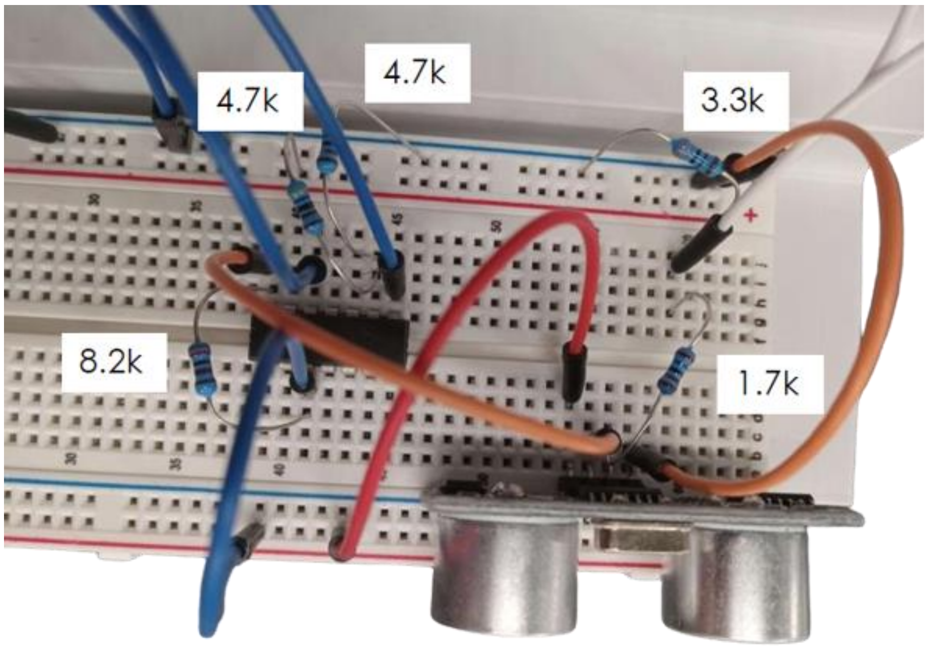


Figure 12: Ultrasonic Breadboard Circuit

2.7 Thermopile Circuit

The Thermopile Circuit is a pivotal component of the autonomous rover's sensory apparatus. Sharing the breadboard circuit with the Ultrasonic Sensor Circuit, it is meticulously engineered to ensure synergy between temperature detection and distance measurement functionalities. The Thermopile sensor capitalizes on the existing breadboard setup, minimizing space and promoting an efficient design.

Upon detecting a station with the Ultrasonic Sensor, the Thermopile sensor is activated to take an accurate temperature reading. The data acquired from the Thermopile is then converted into a readable format displayed on the Basys board's 7-segment display. This real-time temperature display is crucial for the operational protocol of the rover, determining the handling of the metallic washers based on their corresponding thermal stations.

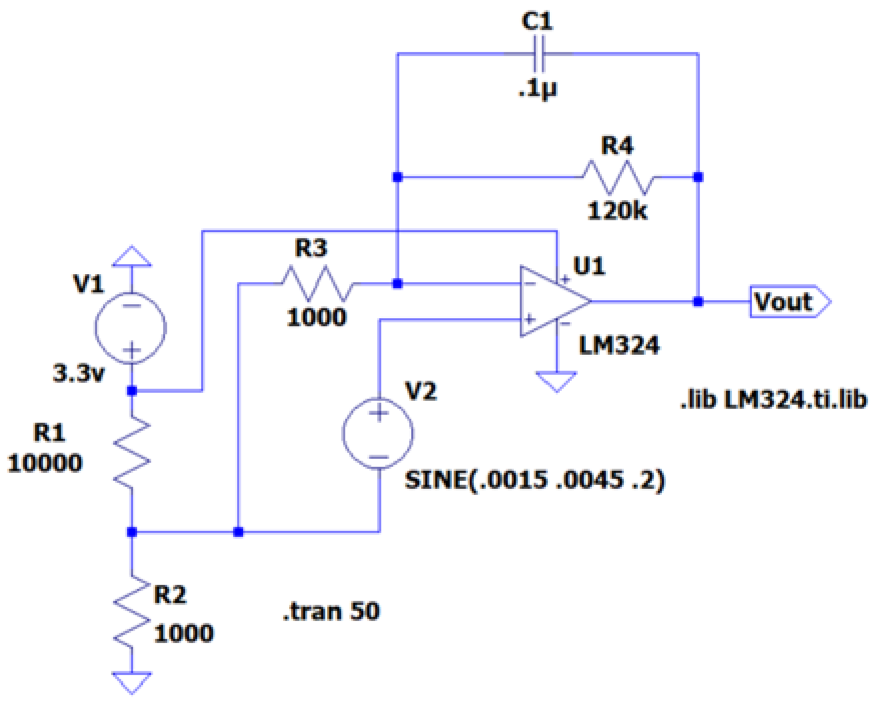


Figure 13: LTSpice Thermopile Schematic

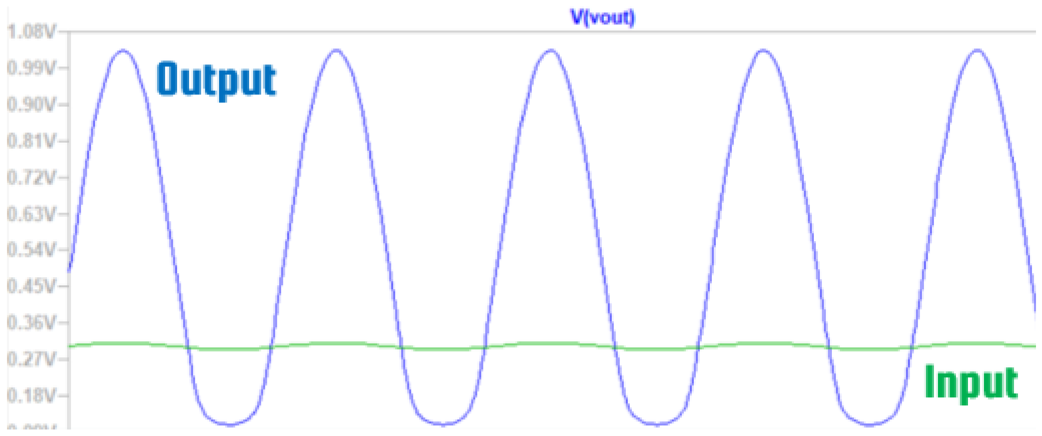


Figure 14: LTSpice Simulation for Thermopile Input/Output

Functionally, the Thermopile sensor influences the magnetic arm's action—either depositing or collecting the washer—based on the temperature data. If the temperature reading corresponds to a hot station, for instance, the current-driven magnet is activated to pick up a red washer. Conversely, for a cold station, the magnet releases the washer. This process is directly tied to the temperature readings, ensuring the rover performs its tasks with precision.

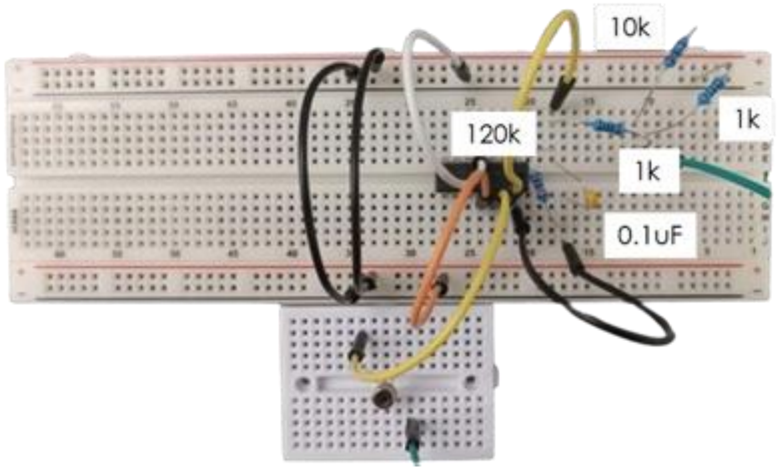


Figure 15: Thermopile Breadboard Circuit

This integration of temperature sensing with the mechanical action of the magnetic arm underlines the sophistication of the rover's automation capabilities. By leveraging these sensors in unison, the rover exhibits a high degree of autonomy, efficiency, and responsiveness to the dynamic conditions of the foundry floor.

2.8 Electromagnet

The electromagnet's operational dynamics are a cornerstone of the rover's material handling capability. This is orchestrated through a circuit involving a MOSFET transistor, specifically the IRF530, which serves as the switching mechanism for the electromagnet. The MOSFET is chosen for its efficiency and reliability in handling the current necessary to power the electromagnet.

The circuit schematic depicts the electromagnet connected in series with a flyback diode and powered by a 10V supply. The Basys board sends a signal to the gate of the MOSFET, which in turn controls the electromagnet's activation. When the Basys board output is high, the MOSFET conducts, and the electromagnet is energized, creating a magnetic field strong enough to pick up metallic washers. Conversely, when the output is low, the MOSFET ceases to conduct, effectively turning off the electromagnet and releasing the washer.

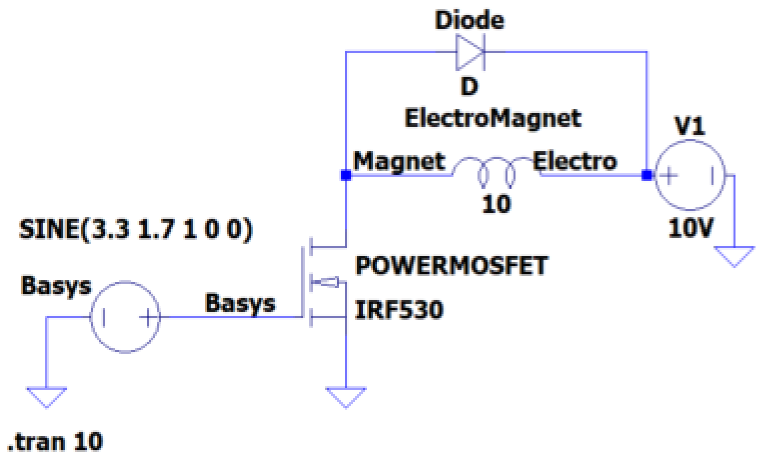


Figure 16: LTSpice Electromagnet Schematic

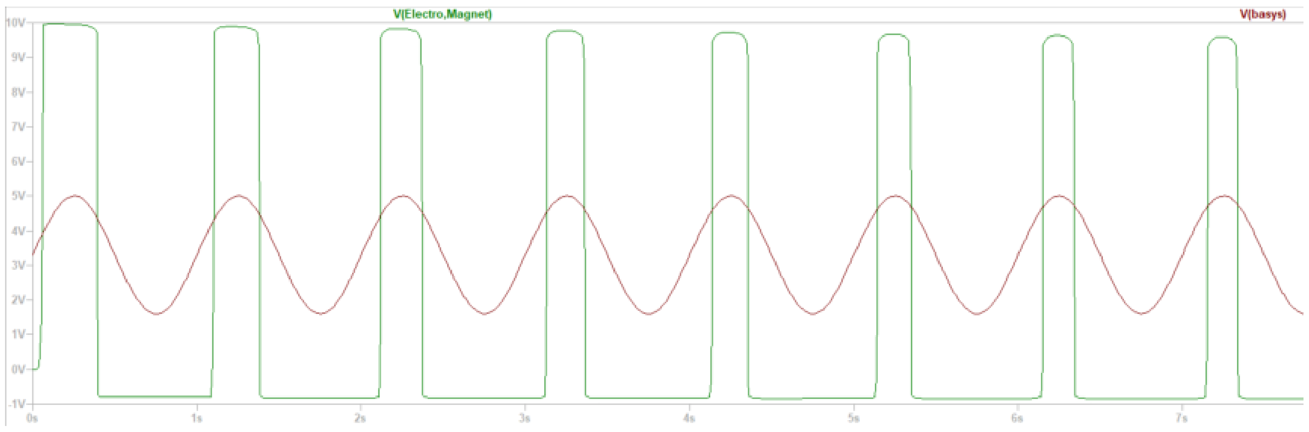


Figure 17: LTSpice Simulation of Electromagnet/Basys Output

Strategically attached to the end of a 3D-printed arm, the electromagnet is both versatile and precise in its function. The arm, mounted on the base of the rover, allows for a wide range of movement and is crucial for the accurate placement and retrieval of washers from the stations. The positioning of the electromagnet, at the extremity of this arm, ensures optimal interaction with the metallic washers, allowing the rover to fulfill its material transport objectives efficiently.

In completion, the electromagnet and magnet arm assembly is responsive to the data received from the Thermopile and Ultrasonic sensors. Once a station is identified, and the temperature is ascertained, the rover's logic system commands the electromagnet to act, either collecting or depositing washers, thereby streamlining the process and enhancing the rover's autonomous capabilities.

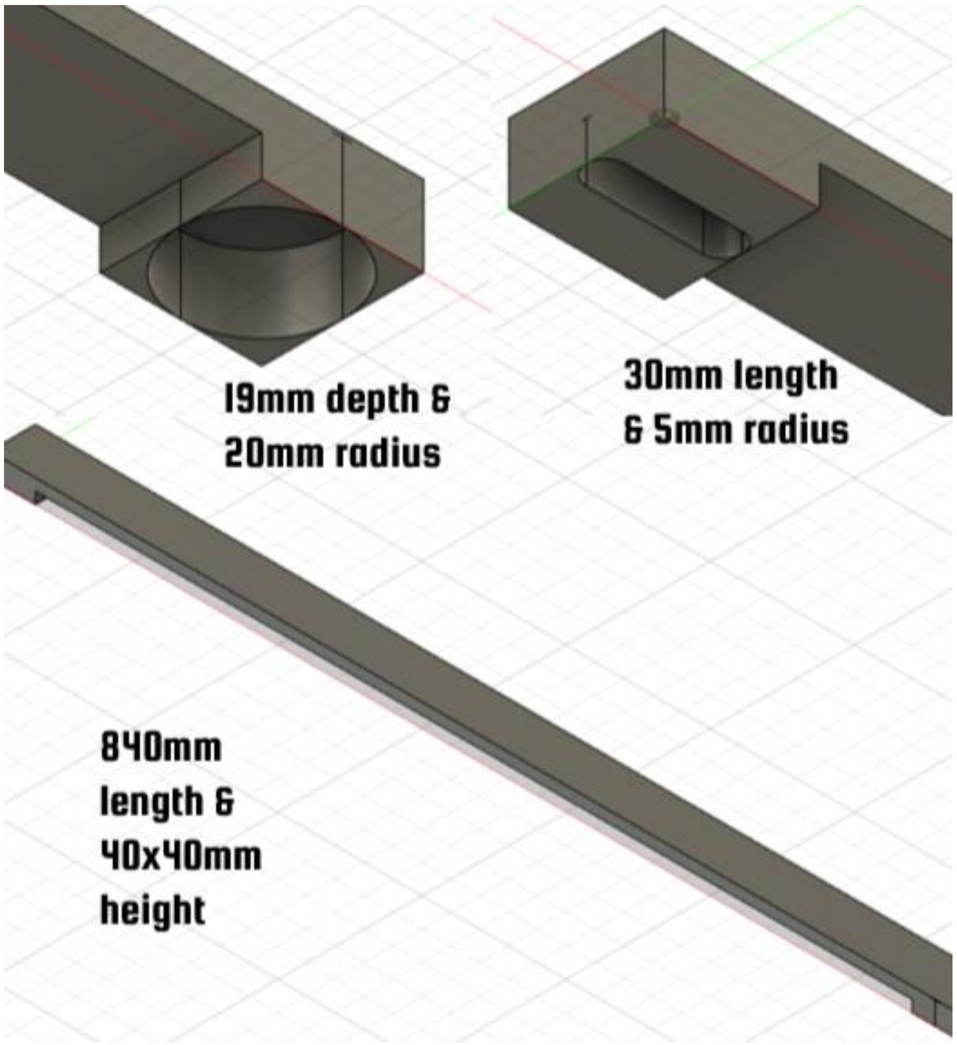


Figure 18: 3D Printed Electromagnet Arm

2.9 3D Printed Main Platform

The hardware setup incorporates custom 3D-printed components, designed using Autodesk Fusion 360 software for improved organization and system cohesion. These components were designed based on measurements obtained from two sources: physical measurements of the rover and reference data from the Rover 5 Chassis manual available on the Stark Fun website. This approach ensured accuracy and alignment with the rover's characteristics. Model A, replicating the central rover region, measures precisely at 90mm by 170mm. It features a base height of 10mm and a 7.50mm radius central hole, facilitating organized wire routing. Model B, measuring 50mm by 50mm, serves as a mounting platform for overcurrent protection mechanisms. With a 10mm base height, it accommodates these protective components while maintaining a low-profile design.

These 3D-printed models enhance hardware organization and functionality, with Model B providing essential protection mechanisms, contributing to a robust and safe electrical integration within the rover's framework.

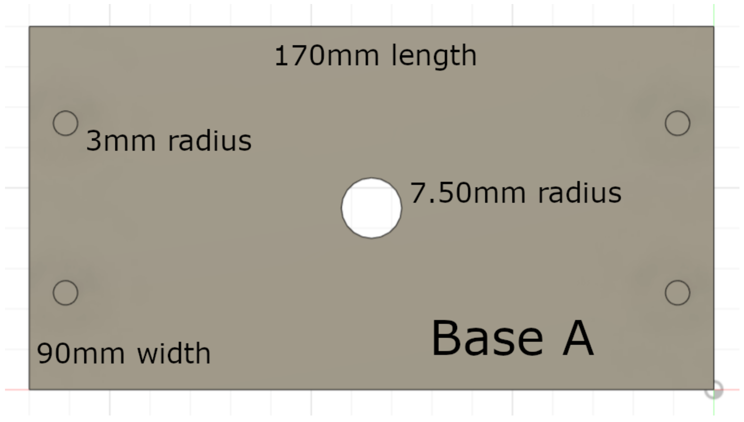


Figure 19: Model A - 2D (XY Axis)

The underside of Model A features four evenly spaced cylindrical structures, each with a height of 10mm and a 3mm radius, as shown in Fig. 8: Model A – 2D (XY Axis). These structures are strategically positioned to align with measurements from opposing corners of the rover's structure, ensuring a secure fit. This design serves a dual purpose. Firstly, it prevents unintended displacement of Model A during operation, ensuring stable attachment to the rover's framework. Secondly, it enhances the hardware setup's structural integrity, improving robustness and reliability. The detailed design of Model A, both on its upper and lower surfaces, emphasizes precision and functionality in the hardware setup. These design elements allow all components to be integrated and enhance overall system performance and safety.

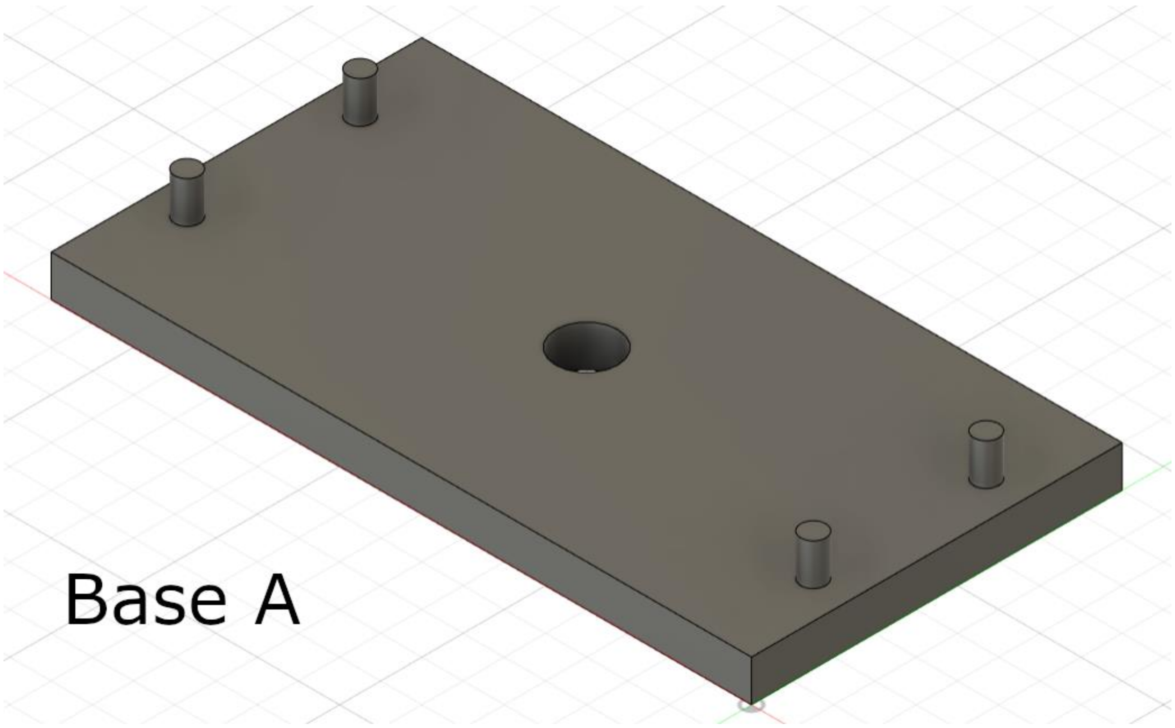


Figure 20: Model A - 3D (Bottom Corner Orbit Axis)

The second custom component (Model B) measures 245mm by 225mm with 6mm thick walls reaching a height of 25mm. A 7.50mm radius hole, consistent with Model A, facilitates alignment and connection between the two components, ensuring a secure assembly, as shown in Fig. 10: Model B – 2D (XY Axis).

Model B's design complements Model A, promoting seamless integration and enhancing overall hardware functionality and cohesion. These 3D-printed components aid in organizing and optimizing the hardware, aligning with the project's objectives of precise motor control and real-time current monitoring.

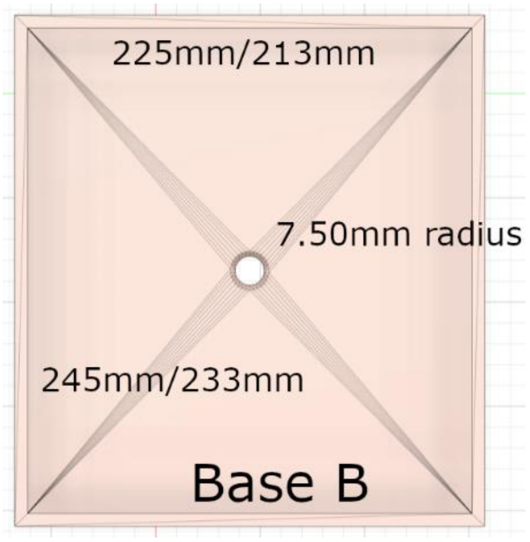


Figure 21: Model B – 2D (XY Axis)

Model B's 6mm thick walls encompass an interior space measuring 233mm by 213mm with a 19mm depth. This area, featured in Fig. 11: Model B – 3D (Top Corner Orbit), serves as a secure platform for housing hardware components, offering benefits such as secure housing, accessibility, optimal configuration, protection, and cohesive integration.

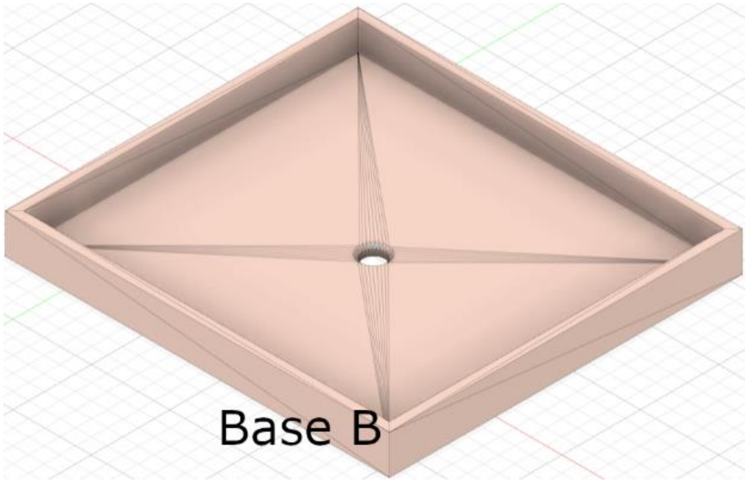


Figure 22: Model B - 3D (Top Corner Orbit Axis)

3. Software

Verilog is the programming language used to write code for the BASYS 3 FPGA board. This code is then uploaded to the board using the Vivado Design Suite. Additionally, Vivado allows for the simulation of the code through a Verilog testbench file. The inputs and outputs for the BASYS 3 board are defined within the Verilog code and a constraints file, which is part of the Vivado project files. This setup enables the BASYS 3 board to transmit and receive signals to various system components, facilitating the control of motor speed and the output of the seven-segment display.

Table I: Connections Between Elements in System

Hardware Connection	Ports	Purpose
BAYSYS3 -> H-Bridge	JB1 -> ENA JB2 -> ENB	Power/Speed Control
	JB3 -> IN1 JB4 -> IN2 JB7 -> IN3 JB8 -> IN4	Direction Control
H-Bridge -> Comparator	SNS A -> 2IN+ SNS B-> 4IN+	Overcurrent Detection
Comparator -> BASYS3	2OUT -> JA1 4OUT -> JA2	
BASYS3 -> Ultrasonic	JC2 -> TRIG	Station Detection
Ultrasonic -> BASYS3	ECHO -> JC1	
IPS -> BASYS3	Left -> JA1 Middle -> JA2 Right -> JA3	Track Detection
Thermopile -> BASYS3	-> JXADC1	Temperature Sensing
Electromagnet -> BASYS3	-> JC4	Object Retrieval

3.1 Software Overview

The BASYS 3 board controls the motors on the Rover V chassis through switches that interface with an H-Bridge. This H-Bridge not only drives the motors but also delivers voltage to the

LM339N comparator and its circuit, which provides software-based overcurrent protection for the BASYS3. The system also incorporates an ultrasonic sensor for station detection, where the BASYS3 sends a trigger pulse (defined in the software) and receives data from the sensor's echo pin to determine the rover's arrival at a station. Additionally, three inductive proximity sensors (IPSs) are utilized for track navigation in the foundry, feeding data as inputs to the BASYS3 board. The rover is equipped with a ZTP-135SR thermopile for temperature sensing. Since this thermopile generates an analog signal, the BASYS3 uses its JXADC ports along with the XADC wizard to convert it into a digital format for processing. The BASYS3 displays both temperature readings and overcurrent states on its built-in seven-segment display. Finally, an electromagnet is employed to pick up and deposit washers at designated stations based on their temperature.

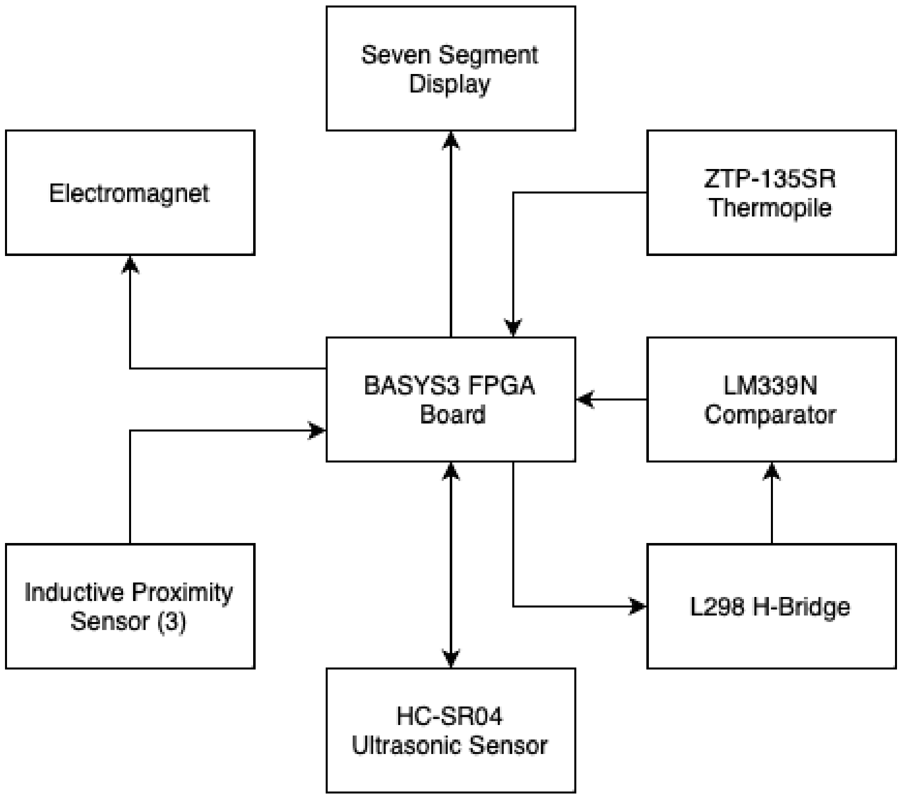
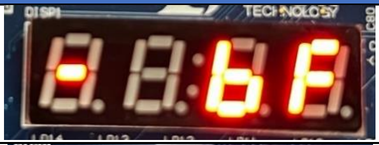
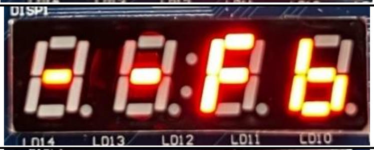
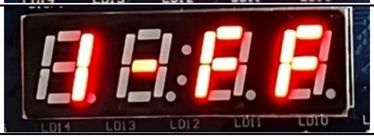


Figure 23: Software System Flowchart

3.2 Seven Segment Display

The BASYS 3 board employs its built-in seven-segment display for two key purposes: to show the current temperature as read by the thermopile and to indicate whether the software's overcurrent protection has been activated. This display features several unique screens. The content shown on each segment or anode of the display is governed by a case statement within the module. The real-time temperature from the thermopile is displayed on the first two anodes on the left. The thermopile module is tasked with dividing the temperature into tens and ones digits for proper display. These digits are then relayed from the thermopile module to the seven-segment display module. Within this module, a nested case statement, part of the main display case statement, decides which digit to show. The display's third anode consistently presents the letter 'c', signifying that the temperature is measured in Celsius. Should overcurrent be detected, the fourth anode will display a zero. In the event that the reset switch is engaged, all anodes will exhibit a dash. The extensive use of local parameters in this module enhances its readability and ease of maintenance.

Table II: Examples of Seven-Segment Display I/O

Overcurrent State	**Motor Direction**	**7 Segment Output**
0	Left Off Right Forwards	
0	Left Forwards Right Off	
1	Both Forwards	

3.3 PWM & Direction

The BASYS 3 board manages the motor speed through a PWM (Pulse Width Modulation) signal, which is created by software. To produce this signal, the clock frequency within the PWM module is first reduced to 48Hz. Each motor is equipped with its own separate speed control mechanism, allowing for three distinct speeds corresponding to duty cycles of 33%, 66%, and 99%. For each motor, three switches on the board are utilized to adjust these speeds, with the activation of each additional switch incrementally increasing the duty cycle.

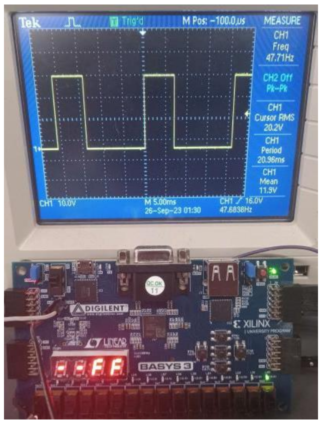


Figure 24: 33% Duty Cycle

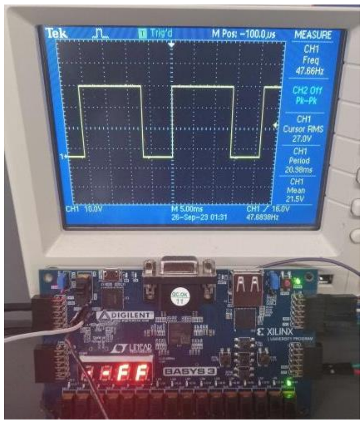


Figure 25: 66% Duty Cycle

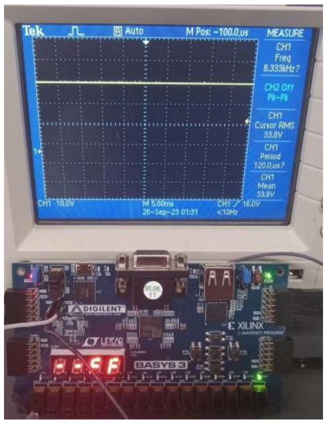


Figure 26: 99% Duty Cycle

Each motor on the rover has its own dedicated direction control, which is governed by the status of the three inductive proximity sensors (IPSs). The rover's direction is determined through a sequence of if statements in the software that process data from the IPSs. For instance, if both the left and middle IPSs are active, the rover will initiate a left turn by deactivating the left motor. The track design in the foundry where the rover operates includes a unique feature: an intersection. On the rover's first approach to this intersection, it is programmed to turn right, but on subsequent passes, it should continue straight. This behavior is managed by a register in the Verilog code. Initially set to 0, this register changes to 1 after the rover's first passage through the intersection, signaling the rover to turn right upon encountering the intersection again.

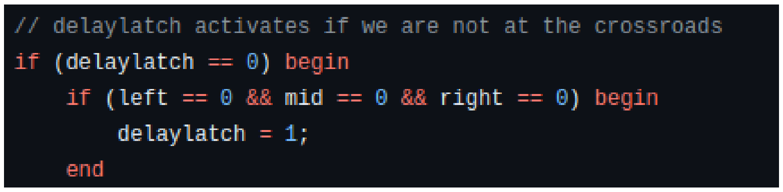

```
// delaylatch activates if we are not at the crossroads
if (delaylatch == 0) begin
    if (left == 0 && mid == 0 && right == 0) begin
        delaylatch = 1;
    end
```

Figure 27: Intersection Checking

3.4 Station Detection

A key part of this project is the rover being able to detect stations and interact with them in the correct order. To ensure this functionality, the M.O.T.O.R – T uses an ultrasonic sensor, thermopile, and electromagnet. All these parts are able to talk with each other and are used for various toggles in the entire system.

3.4.1 Ultrasonic Sensor

The rover is equipped with an ultrasonic sensor to determine when it reaches a station. This sensor is distinctive because it both sends and receives signals to and from the BASYS3 board. The BASYS3 board emits a 10us pulse to the ultrasonic sensor every 83ms, functioning as the trigger signal. This signal is projected towards an object, which the ultrasonic sensor then reads and reflects back to the BASYS3 board. The returned signal, known in this project as the echo signal, is received by the BASYS3 board through the echo pin. The echo signal stays high as long as an object is within a certain distance, defined in the program by a 33-bit binary value.

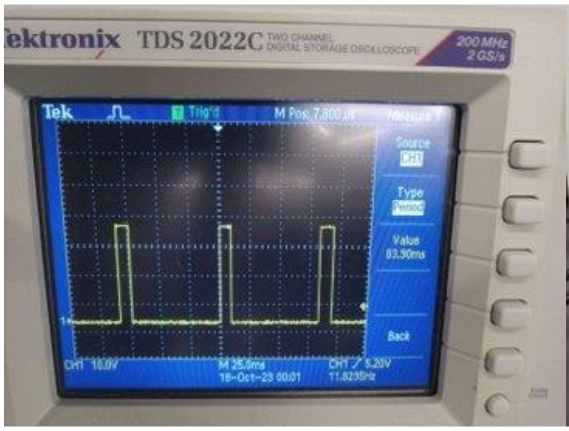


Figure 28: Ultrasonic Trigger Signal

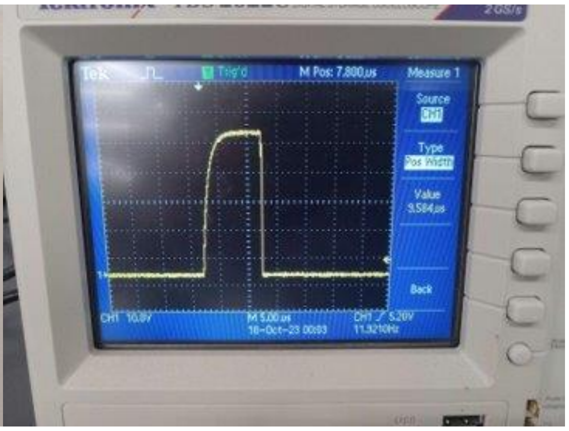
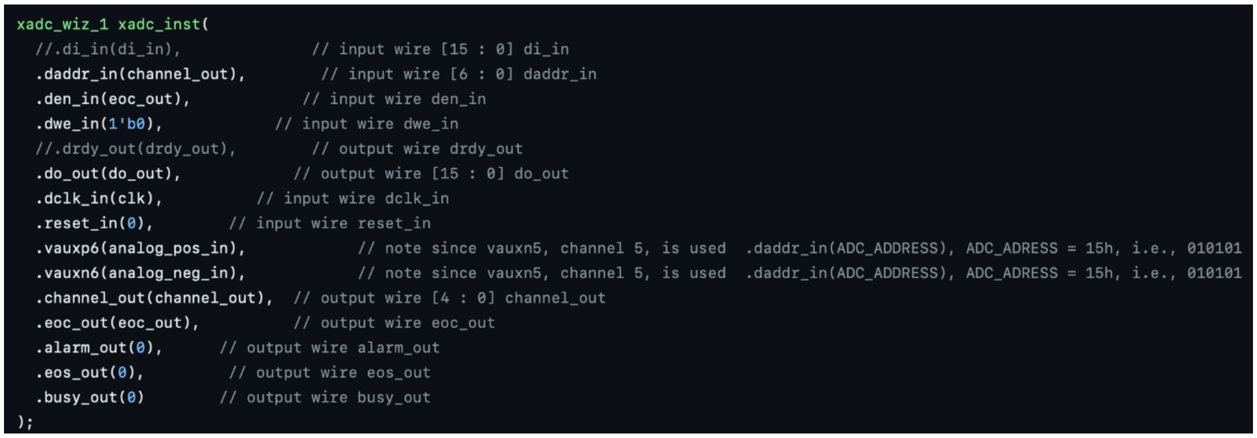


Figure 29: Ultrasonic Echo Signal

3.4.2 Thermopile & XADC Configuration

The ZTP-135SR thermopile on the rover serves to gauge the temperature at each station it visits, providing real-time and precise temperature readings of its environment. This thermopile stands out because it produces an analog signal, whereas other components in the system operate with digital signals. The BASYS3 FPGA board, to which this thermopile is connected, cannot directly interpret data from the thermopile in its default state. As a result, the XADC Pmod ports are necessary to facilitate the use of this thermopile, effectively serving as an analog-to-digital converter. Additionally, the built-in XADC wizard in the Vivado design suite is required for further setup. This wizard simplifies the process by automatically configuring various settings for the BASYS3, eliminating the need for manual setup. However, it is important to note that any module using the XADC ports must incorporate the XADC wizard within the module itself.



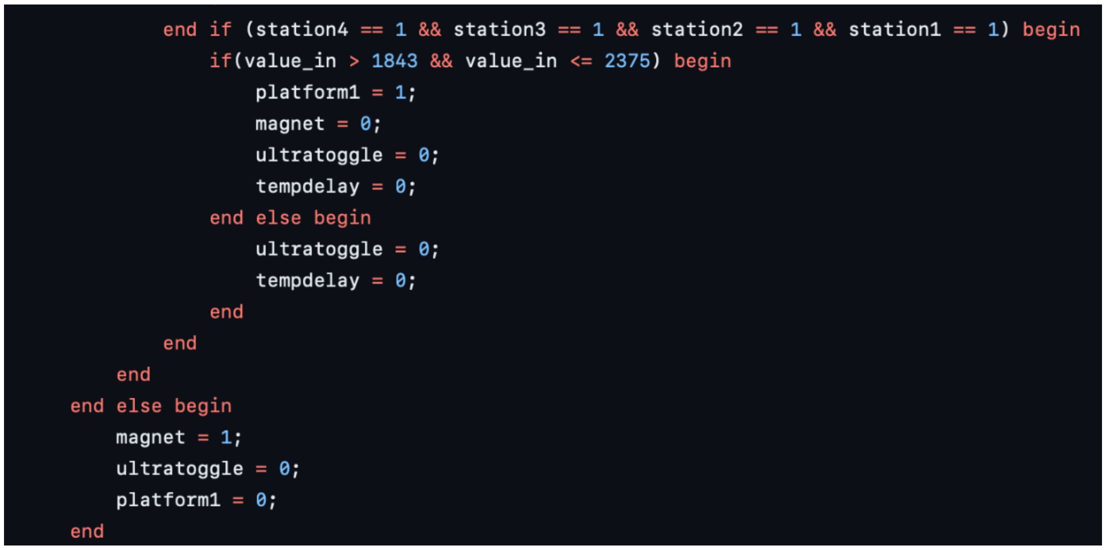
```
xadc_wiz_1 xadc_inst(
  //.di_in(di_in),              // input wire [15 : 0] di_in
  .daddr_in(channel_out),        // input wire [6 : 0] daddr_in
  .den_in(eoc_out),            // input wire den_in
  .dwe_in(1'b0),            // input wire dwe_in
  //.drdy_out(drdy_out),        // output wire drdy_out
  .do_out(do_out),            // output wire [15 : 0] do_out
  .dclk_in(clk),          // input wire dclk_in
  .reset_in(0),        // input wire reset_in
  .vauxp6(analog_pos_in),            // note since vauxn5, channel 5, is used  .daddr_in(ADC_ADDRESS), ADC_ADRESS = 15h, i.e., 010101
  .vauxn6(analog_neg_in),            // note since vauxn5, channel 5, is used  .daddr_in(ADC_ADDRESS), ADC_ADRESS = 15h, i.e., 010101
  .channel_out(channel_out),  // output wire [4 : 0] channel_out
  .eoc_out(eoc_out),          // output wire eoc_out
  .alarm_out(0),      // output wire alarm_out
  .eos_out(0),         // output wire eos_out
  .busy_out(0)        // output wire busy_out
);
```

Figure 30: XADC Instantiation to Main Code

The temperature, measured in Celsius, is read every 0.5 seconds to ensure accurate readings. Within the system, a register named 'value_in' is established, which is assigned the value of the 'do_out' register generated by the XADC instantiation. To focus on the relevant data, 'value_in' is shifted four bits to the right, capturing only the twelve most significant bits from 'do_out', as the thermopile does not utilize the remaining four bits. This value is then processed to determine the tens and ones digits of the temperature. This calculation involves linear interpolation, a necessary step since the thermopile measures an analog signal. Specific voltages are chosen to formulate a linear equation, which is then applied to derive a practical temperature reading.

3.4.3 Electromagnet

The rover employs an electromagnet to handle the task of picking up and depositing washers, a process determined by the temperature at the station it reaches. To control the electromagnet's operation, a register named "magnet" is used, which switches the electromagnet between its 'on' and 'off' states. This switch is governed by a series of if statements within the program. When the rover arrives at the designated station, the electromagnet is turned off, allowing it to release its current load at the station's drop-off pillar. Subsequently, the rover moves to the next pillar at the station, where the electromagnet is reactivated to acquire a new load. In contrast, if the rover arrives at an incorrect station, the electromagnet remains active, ensuring that the current load is not inadvertently dropped.



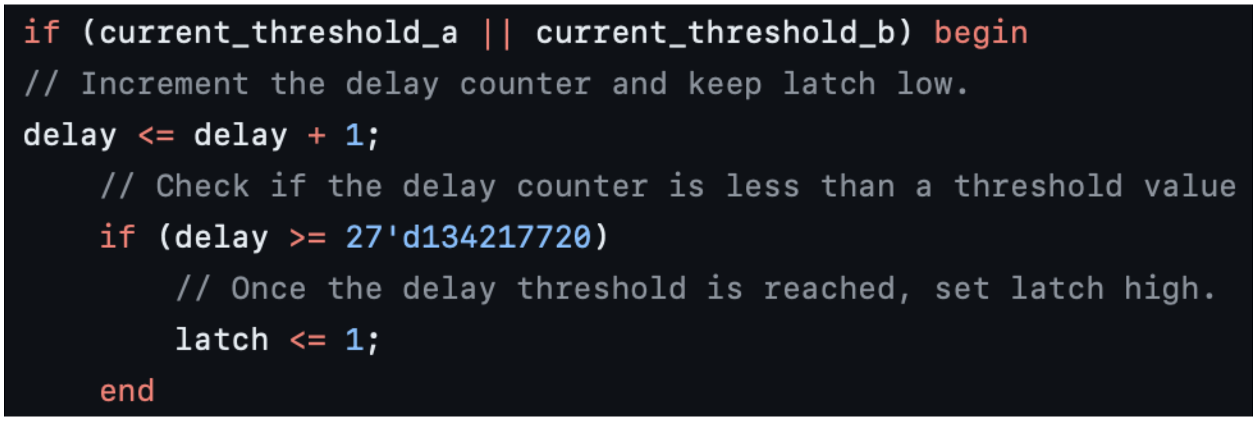
```
            end if (station4 == 1 && station3 == 1 && station2 == 1 && station1 == 1) begin
                if(value_in > 1843 && value_in <= 2375) begin
                    platform1 = 1;
                    magnet = 0;
                    ultratoggle = 0;
                    tempdelay = 0;
                end else begin
                    ultratoggle = 0;
                    tempdelay = 0;
                end
            end
        end
    end else begin
        magnet = 1;
        ultratoggle = 0;
        platform1 = 0;
    end
```

Figure 31: Electromagnet Toggles in Station Detection

3.5 Software Overcurrent Protection

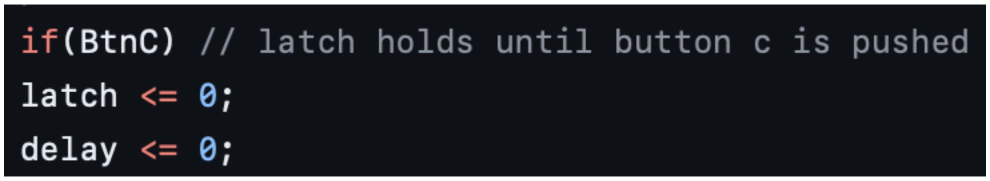
Primarily functioning as a controller for the motor speed and direction of the Rover 5, the H-Bridge also plays a crucial role in safeguarding the system against dangerous overcurrent situations. It accomplishes this by transmitting a voltage signal to the LM339N comparator. The comparator, by contrasting this signal with a pre-set reference voltage, can ascertain the presence of overcurrent in the system. Upon detecting overcurrent, the comparator emits a high signal to the BASYS3 board. This signal is then utilized in both the seven-segment display module and the PWM module. In response to overcurrent detection, the display shows a zero, and the motors are shut down after approximately 1.3

seconds of continuous overcurrent. This shutdown activates a latch mechanism. The system can only be reset and the motors returned to normal operation when a specific button on the BASYS3 is pressed, releasing the latch.



```
if (current_threshold_a || current_threshold_b) begin
// Increment the delay counter and keep latch low.
delay <= delay + 1;
    // Check if the delay counter is less than a threshold value
    if (delay >= 27'd134217720)
        // Once the delay threshold is reached, set latch high.
        latch <= 1;
    end
```

Figure 32: Enabling the Latch



```
if(BtnC) // latch holds until button c is pushed
latch <= 0;
delay <= 0;
```

Figure 33: Disabling the Latch

4. Results

Throughout its lifecycle, the M.O.T.O.R - T project has adeptly achieved the integration of software and hardware elements, culminating in a sophisticated and cohesive system. This complete phase of development not only showcases the project's comprehensive achievements but also marks the successful completion of its objectives.

A critical foundation of the project was its robust power management system, underpinned by 9.6V batteries and reinforced with a 1A fuse for safety. Power regulators played a pivotal role in ensuring consistent voltage across all circuit components, with the Basys Board efficiently utilizing the 5V & 3.3V regulators. The Ultrasonic module, essential for distance measurement and object detection, functioned seamlessly with its outputs precisely processed by the Basys Board. The IPS system effectively managed directional inputs, while a color-coded wiring scheme facilitated error-free assembly. At the heart of the system, the Basys Board controlled motor movement and direction with precision, supported by the H-Bridge. Additionally, the integration of the Thermopile chip was crucial, accurately detecting different stations to determine the activation of the electromagnet, while concurrently displaying information on the Basys Board's 7-segment display.

Reflecting on the project's complete journey, it stands as a testament to the successful and harmonious blending of software and hardware components. The results not only highlight the team's dedication and expertise but also represent the efforts, showcasing a complete and fully functional autonomous system.

5. Summary & Conclusions

In summary, the M.O.T.O.R - T project is poised to revolutionize material handling and processing in foundries. The integration of an autonomous rover with advanced sensors has significantly improved operational efficiency, precision, and safety. Inductive proximity sensors accurately detect metallic paths, while the ultrasonic sensor measures distances for station identification. The temperature sensor ensures proper washer handling. The project showcases technology's potential to enhance manufacturing processes by reducing costs and improving safety. It represents an inspiring example of engineering innovation in traditional industries, ushering in a more efficient and technologically advanced foundry landscape.

6. Acknowledgement

The successful completion of this project is indebted to the invaluable support and collaboration of key individuals who played vital roles throughout its execution. First and foremost, we extend our sincere gratitude to Landon Mehta, a dedicated Computer Engineering student at Texas Tech. Landon's extensive knowledge and expertise greatly assisted us in understanding the intricate circuit layout. His insightful recommendations, particularly regarding the incorporation of an overcurrent protection fuse, significantly enhanced the project's robustness and safety.

We also express our heartfelt thanks to Silas Rodriguez, another accomplished Computer Engineering student at Texas Tech. Silas played a pivotal role in deepening our comprehension of PWM integration and providing essential insights into Verilog programming. His guidance served as a crucial reference point, helping us navigate the complexities of programming and seamlessly integrate vital components into the project.

The contributions of Landon Mehta and Silas Rodriguez exemplify the collaborative spirit and knowledge-sharing ethos that defined this project. Their support was instrumental in overcoming challenges and ultimately achieving our project objectives.

7. Safety, Public Health, & Welfare Considerations

The development and deployment of any engineering project, especially one that involves autonomous movement and operations within a human-centric environment such as the Texas Tech Foundry, necessitates a thorough examination of the implications on safety, public health, and welfare.

In conclusion, while the M.O.T.O.R - T is a significant stride in enhancing efficiency and automation within the foundry, it brings forth a set of responsibilities. Prioritizing safety, public health, and welfare in its design and operation ensures a balanced approach, aligning technological advancements with human-centric considerations.

7.1 Safety Considerations

Physical Safety: The movement of the autonomous rover poses a potential hazard, especially if it veers off its set path. However, the use of shiny metallic line sensing sensors ensures that the rover adheres strictly to its intended course. In the rare case of deviations, a manual override or an emergency stop mechanism should be integrated.

Electrical Safety: The inclusion of a hardware current protection system is commendable, as it prevents potential overcurrent situations that could lead to electrical fires or damage. Furthermore, the rover's design should ensure that all electrical components are well insulated to prevent any inadvertent electrical shocks to the foundry workers.

Overheating: The interaction of the rover with hot stations (35-45°C) may expose it to heat levels that could affect its performance. Proper insulation and heat-resistant materials are vital to ensure the rover's operational longevity and prevent any fire hazards.

7.2 Public Health Conditions

Air Quality: Continuous movement of the rover, especially if powered by non-electric means, could lead to emissions. While this rover is electrically powered, ensuring minimal emissions from any associated operations is crucial.

Noise Pollution: The continuous operation of the rover, especially during pickup and drop-off of washers, could generate noise. It's essential to maintain noise levels within permissible limits to ensure a conducive working environment in the foundry.

Ergonomics: While the rover is designed to reduce the manual labor of transporting metals between stations, consideration of how workers interact with the rover (e.g., loading/unloading or maintenance) is essential. The design should prioritize ergonomic principles to prevent any strain or injuries.

7.3 Welfare Considerations

Job Implications: The introduction of an autonomous system invariably raises questions about its impact on jobs. While the rover automates certain transport tasks, it should be positioned as a tool to assist workers rather than replace them, potentially offering opportunities for workers to upskill and manage more sophisticated tasks.

Training: With the deployment of the rover, specialized training should be provided to foundry workers. This training should cover safe interaction with the rover, basic troubleshooting, and understanding its indications, especially from the 7-segment display.

Accessibility: Provisions should be made to ensure that the foundry floor remains accessible and navigable for all workers, including those with disabilities, even with the rover's movement.

8. Global, Environmental & Economic Factors

The M.O.T.O.R – T implementation of the autonomous rover is not just a technical feat; it stands at the intersection of global implications, environmental impacts, and economic ramifications. This section dissects these intertwined factors, shedding light on the broader repercussions of the project.

8.1 Global Considerations

Standardization: As technology becomes more intertwined with daily operations globally, adhering to international standards for autonomous machines ensures that the rover can be replicated or integrated into foundries worldwide. This encourages global collaboration and ensures consistency in operations across borders.

Interconnected Supply Chains: The components used for the rover, such as the Basys-3 development board, H-Bridge, and the ZTP-135SR Thermopile, might have a global supply chain. This interdependence underscores the importance of maintaining robust, disruption-free supply chains to ensure the timely execution of such projects.

8.2 Environmental Considerations

Energy Consumption: While the rover is designed to operate efficiently, its energy consumption patterns should be examined. Prioritizing energy-efficient operations and possibly integrating renewable energy sources can mitigate its carbon footprint.

Material Sustainability: The use of aluminum for temperature measurement blocks and other metal components necessitates a look into the sourcing of these materials. Using recycled or sustainably sourced metals can significantly reduce the environmental impact of the project.

E-Waste Management: At the end of its lifecycle, the rover and its components should be disposed of responsibly. Implementing a strategy for recycling and reusing parts can curb the proliferation of electronic waste.

8.3 Economic Factors

Budgeting and Expenditure: Tracking economic factors is pivotal for the success and feasibility of the project. As noted, economic factors were diligently tracked daily into an excel budget sheet, as seen in Appendix A. This transparency in expenditure ensures the project remains within its financial constraints while highlighting areas for potential savings or overruns.

Operational Savings: The rover, by automating transport tasks within the foundry, offers potential savings. Reducing manual labor costs and enhancing efficiency can lead to significant economic benefits in the long run.

Maintenance and Upkeep: While initial investments in the rover might be substantial, considerations should be made regarding its maintenance costs. Ensuring that the rover is built with durable components can minimize frequent replacements and repairs, leading to economic savings.

Job Reorientation: As the rover takes on transport tasks, there might be concerns about job losses. However, reorienting the workforce towards more skilled tasks, maintenance, or overseeing the rover's operations can lead to better job roles and potentially higher economic returns for the workers.

References

1. CourseHero. “This is the pulse width length for Hi-TEC HS-422 servo motor” <https://www.coursehero.com/tutors-problems/Electrical-Engineering/31473209-This-is-the-pulse-width-length-for-Hi-TEC-HS-422-servo-motor-This-ser/%E2%80%8B>
2. Digikey. “HS 422 Servo Datasheet” <https://media.digikey.com/pdf/data%20sheets/dfrobot%20pdfs/ser0002_web.pdf%E2%80%8B> (accessed October 15, 2023) >
3. Sparkfun Electronics. “Rover 5 Robot Platform,” 30 January 2013 <https://www.sparkfun.com/datasheets/Robotics/Rover%205%20Introduction.pdf> (accessed September 18, 2023).
4. Texas Instruments. “Chapter 3 Description” <https://www.ti.com/lit/ds/symlink/ina250.pdf> (accessed September 12, 2023)
5. Texas Instruments. “Chapter 7.3.1 Integrated Resistor” <https://www.ti.com/lit/ds/symlink/ina250.pdf> (accessed September 12, 2023)
6. YouTube. GEEK3502. “How to code verilog for ultrasonic distance sensing” <https://youtu.be/FcG-xNruEZE?si=Rq4GEU4s7Xwk3ieR>

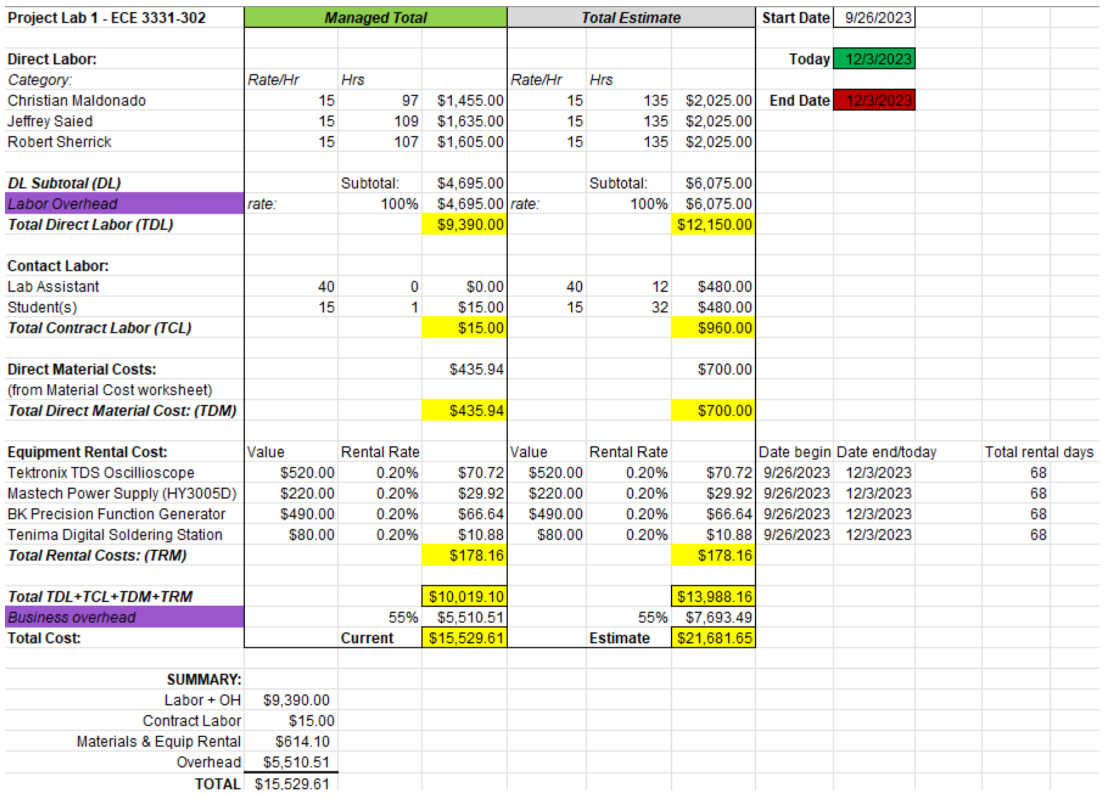
Appendix A

Excel Budget

This section provides an overview of the comprehensive budgeting process employed from the project's initiation on September 26, 2023, to its conclusion on December 3, 2023.

The total of TDL, TCL, TDM, and TRM, incorporating business overhead costs, resulted in the Managed Total Cost. This figure, totaling $15,529.61, encompasses all direct and indirect expenses associated with the project.

This managed total was compared against the initial estimate of $21,681.65. The deviation of $6,152.04 underscores the project's efficiency in cost management, resulting in over $5,000 of expenditure compared to the estimated projection cost.

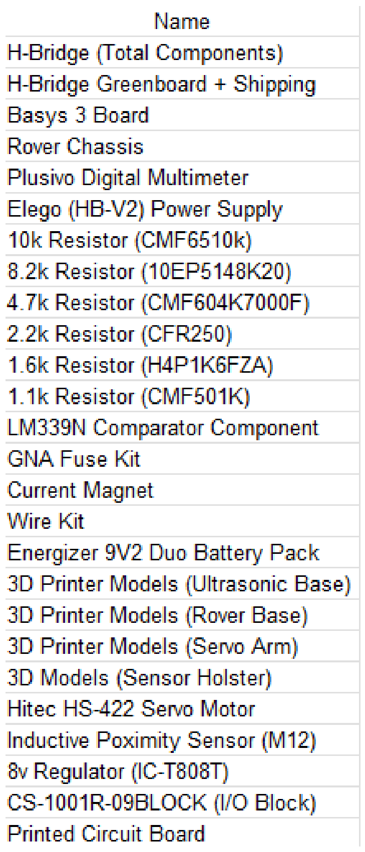


Project Lab 1 - ECE 3331-302	***Managed Total***			***Total Estimate***			**Start Date**	9/26/2023		
Direct Labor:							**Today**	12/3/2023		
Category:	*Rate/Hr*	*Hrs*		*Rate/Hr*	*Hrs*					
Christian Maldonado	15	97	$1,455.00	15	135	$2,025.00	**End Date**	12/3/2023		
Jeffrey Saied	15	109	$1,635.00	15	135	$2,025.00				
Robert Sherrick	15	107	$1,605.00	15	135	$2,025.00				
DL Subtotal (DL)		Subtotal:	$4,695.00		Subtotal:	$6,075.00				
Labor Overhead	*rate:*	100%	$4,695.00	*rate:*	100%	$6,075.00				
Total Direct Labor (TDL)			$9,390.00			$12,150.00				
Contact Labor:										
Lab Assistant	40	0	$0.00	40	12	$480.00				
Student(s)	15	1	$15.00	15	32	$480.00				
Total Contract Labor (TCL)			$15.00			$960.00				
Direct Material Costs:			$435.94			$700.00				
(from Material Cost worksheet)										
Total Direct Material Cost: (TDM)			$435.94			$700.00				
Equipment Rental Cost:	Value	Rental Rate		Value	Rental Rate		Date begin	Date end/today		Total rental days
Tektronix TDS Oscilloscope	$520.00	0.20%	$70.72	$520.00	0.20%	$70.72	9/26/2023	12/3/2023		68
Mastech Power Supply (HY3005D)	$220.00	0.20%	$29.92	$220.00	0.20%	$29.92	9/26/2023	12/3/2023		68
BK Precision Function Generator	$490.00	0.20%	$66.64	$490.00	0.20%	$66.64	9/26/2023	12/3/2023		68
Tenima Digital Soldering Station	$80.00	0.20%	$10.88	$80.00	0.20%	$10.88	9/26/2023	12/3/2023		68
Total Rental Costs: (TRM)			$178.16			$178.16				
Total TDL+TCL+TDM+TRM			$10,019.10			$13,988.16				
Business overhead		55%	$5,510.51		55%	$7,693.49				
Total Cost:		**Current**	$15,529.61		**Estimate**	$21,681.65				
SUMMARY:										
Labor + OH	$9,390.00									
Contract Labor	$15.00									
Materials & Equip Rental	$614.10									
Overhead	$5,510.51									
TOTAL	$15,529.61									

Appendix B

Excel Bill of Materials

The Bill of Materials (BOM) reflects a detailed inventory of components necessary for the successful execution of the project. Notably, the BOM total expense stands at $435.94, a figure that aligns with the meticulous planning and procurement strategies implemented. This cost is inclusive of all the hardware components listed, such as the H-Bridge Greenboard, Baysys 3 Board, and various electronic components and sensors, alongside their respective quantities and unit costs. The careful selection of parts and suppliers has ensured that the project remains within the economic scope, demonstrating prudence and resourcefulness in the allocation of resources. This BOM is integral to the Managed Total Cost, contributing directly to the project's overarching financial framework.

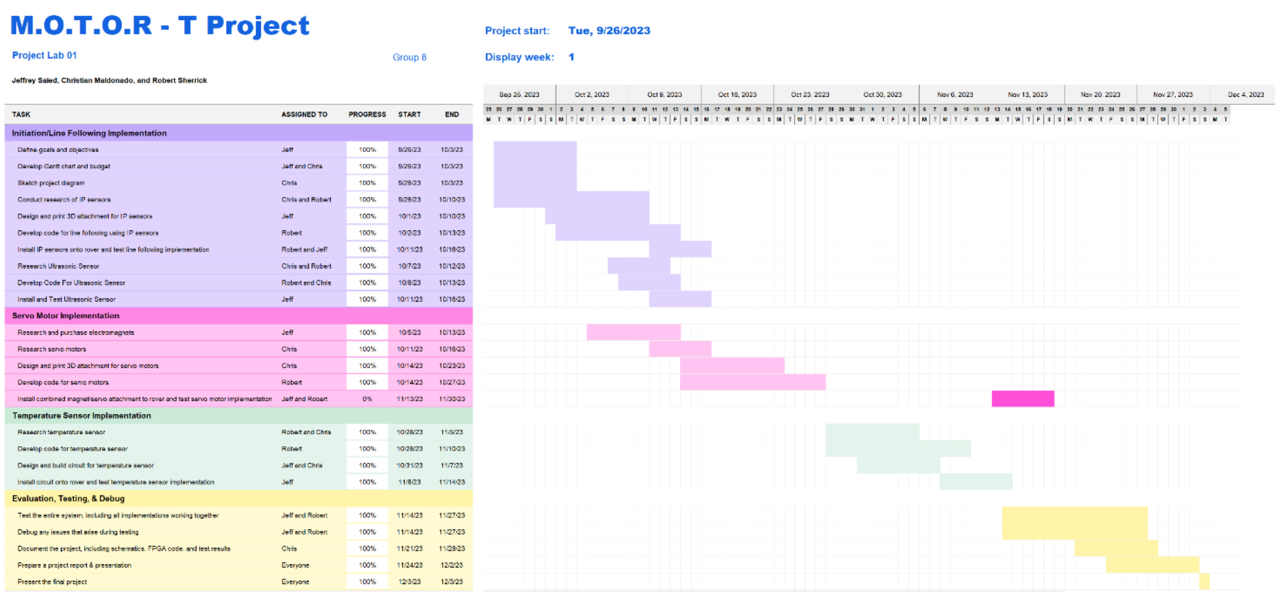


Name
H-Bridge (Total Components)
H-Bridge Greenboard + Shipping
Basys 3 Board
Rover Chassis
Plusivo Digital Multimeter
Elego (HB-V2) Power Supply
10k Resistor (CMF6510k)
8.2k Resistor (10EP5148K20)
4.7k Resistor (CMF604K7000F)
2.2k Resistor (CFR250)
1.6k Resistor (H4P1K6FZA)
1.1k Resistor (CMF501K)
LM339N Comparator Component
GNA Fuse Kit
Current Magnet
Wire Kit
Energizer 9V2 Duo Battery Pack
3D Printer Models (Ultrasonic Base)
3D Printer Models (Rover Base)
3D Printer Models (Servo Arm)
3D Models (Sensor Holster)
Hitec HS-422 Servo Motor
Inductive Poximity Sensor (M12)
8v Regulator (IC-T808T)
CS-1001R-09BLOCK (I/O Block)
Printed Circuit Board

Appendix C

Gantt Chart

The dynamic Gantt chart, initiated on September 26, 2023, and set to conclude on December 3, 2023, proved invaluable for flexible project management. This continuously updated chart allowed real-time progress tracking, efficient resource allocation, and bottleneck mitigation. The project was divided into four main tasks, each with subtasks. The team has largely adhered to the schedule, exceeding expectations for temperature sensing. However, there's a slight delay in the servo motor implementation, particularly in designing the arm for washer manipulation. To address this, the team may consider adjusting the timeline to prioritize temperature sensing without a significant impact on the project's overall success.



M.O.T.O.R - T Project

Project Lab 01 — Group 8

Jeffrey Saied, Christian Maldonado, and Robert Sherrick

Project start: Tue, 9/26/2023

Display week: 1

TASK	ASSIGNED TO	PROGRESS	START	END
Initiation/Line Following Implementation				
Define goals and objectives	Jeff	100%	9/26/23	10/3/23
Develop Gantt chart and budget	Jeff and Chris	100%	9/26/23	10/3/23
Sketch project diagram	Chris	100%	9/26/23	10/3/23
Conduct research of IP sensors	Chris and Robert	100%	9/26/23	10/10/23
Design and print 3D attachment for IP sensors	Jeff	100%	10/1/23	10/10/23
Develop code for line following using IP sensors	Robert	100%	10/2/23	10/13/23
Install IP sensors onto rover and test line following implementation	Robert and Jeff	100%	10/11/23	10/16/23
Research Ultrasonic Sensor	Chris and Robert	100%	10/7/23	10/12/23
Develop Code For Ultrasonic Sensor	Robert and Chris	100%	10/8/23	10/13/23
Install and Test Ultrasonic Sensor	Jeff	100%	10/11/23	10/16/23
Servo Motor Implementation				
Research and purchase electromagnets	Jeff	100%	10/5/23	10/13/23
Research servo motors	Chris	100%	10/11/23	10/16/23
Design and print 3D attachment for servo motors	Chris	100%	10/14/23	10/23/23
Develop code for servo motors	Robert	100%	10/14/23	10/27/23
Install combined magnet/servo attachment to rover and test servo motor implementation	Jeff and Robert	0%	11/13/23	11/30/23
Temperature Sensor Implementation				
Research temperature sensor	Robert and Chris	100%	10/28/23	11/5/23
Develop code for temperature sensor	Robert	100%	10/28/23	11/10/23
Design and build circuit for temperature sensor	Jeff and Chris	100%	10/31/23	11/7/23
Install circuit onto rover and test temperature sensor implementation	Jeff	100%	11/8/23	11/14/23
Evaluation, Testing, & Debug				
Test the entire system, including all implementations working together	Jeff and Robert	100%	11/14/23	11/27/23
Debug any issues that arise during testing	Jeff and Robert	100%	11/14/23	11/27/23
Document the project, including schematics, FPGA code, and test results	Chris	100%	11/21/23	11/28/23
Prepare a project report & presentation	Everyone	100%	11/24/23	12/2/23
Present the final project	Everyone	100%	12/3/23	12/3/23

Appendix D

Code

```
`timescale 1ns / 1ps

module MainProject (
    input wire clk,           // Clock signal
    input wire rst,           // Reset signal
    input wire [1:0] speed_a, // Speed control for motor A (2 bits)
    input wire [1:0] speed_b, // Speed control for motor B (2 bits)
    input current_threshold_a, // current threshold input
    input current_threshold_b, // current threshold input
    input wire ip_a, ip_b, ip_c,
    input BtnC,
    output wire a, b, c, d, e, f, g, dp, // Individual LED output
    output [3:0] an,             // 4-bit enable signal
    output reg pwm_a,         // PWM signal for motor A
    output reg pwm_b,         // PWM signal for motor B
    output reg input1, input2, input3, input4,        // Direction control for motor A & motor B (0: forward, 1: reverse)
    output reg led_a,        // LED control for motor A (connect to an LED)
    output reg led_b,         // LED control for motor B (connect to a different LED)
    output reg led_c,
    output wire trigger,
    input wire pulse,
    output wire state,
    input wire analog_pos_in, analog_neg_in,
    output reg led1, led2, led3, led4, led5, led6, led7, led8,
    output magnettoggle
);

    localparam N = 21;

    localparam ZERO = 7'b1000000;
    localparam ONE = 7'b1111001;
    localparam TWO = 7'b0100100;
    localparam THREE = 7'b0110000;
    localparam FOUR = 7'b0011001;
    localparam FIVE = 7'b0010010;
    localparam SIX = 7'b0000010;
    localparam SEVEN = 7'b1111000;
    localparam EIGHT = 7'b0000000;
    localparam NINE = 7'b0010000;
    localparam DASH = 7'b0111111;
    localparam CELSIUS = 7'b0100111;
    localparam NOTHING = 7'b1111111;

    localparam EN1 = 4'b0111;
    localparam EN2 = 4'b1011;
    localparam EN3 = 4'b1101;
    localparam EN4 = 4'b1110;

    // Internal signals
    reg [N-1:0] counter = 0;    // 21-bit counter for PWM generation
    reg [N-1:0] count = 0;    // 21-bit count for 7 seg display
    reg [N+4:0] delay = 0;    // 26-bit count for current threshold delay
    reg [N-1:0] duty_cycle_a = 0; // Duty cycle control for motor A (4 bits)
    reg [N-1:0] duty_cycle_b = 0; // Duty cycle control for motor B (4 bits)
    reg [6:0] s_temp; // 7-bit register to hold data for display
    reg [3:0] an_temp; // 4-bit enable register
    reg latch = 0;
    reg [N+6:0] turndelay = 0;
    reg delaylatch = 0;
    reg [N+4:0] controldelay = 0;
    reg delay_done = 0;
    reg delay_trigger = 0;
    reg [32:0] tempcount = 0;

//Ultrasonic
    reg [22:0] pcount;
    reg n_trigger;
    reg old_pulse;
    reg [32:0] count_out;
    reg [32:0] pulse_count;
    reg [32:0] status;

////ServoMotor
//    parameter PWM_PERIOD = 2000000; // 20ms in clock cycles (100MHz)
//    parameter MIN_WIDTH = 90000;   // 0.9ms in clock cycles
//    parameter MAX_WIDTH = 210000;  // 2.1ms in clock cycles
//    parameter TRANSITION_TIME = 100000000; // 1 seconds in clock cycles

//    reg [20:0] scounter = 0;
//    reg [27:0] tcounter = 0;
//    reg [20:0] temp_servo = 0;

// Delay Counts
    reg [31:0] ultracount = 0;
    reg [31:0] tempdelay = 0;

// Toggles
    reg temptoggle = 0;
    reg platform1;
    reg ultratoggle = 1;
    reg station1 = 1;
    reg station2 = 0;
    reg station3 = 0;
    reg station4 = 0;
    reg magnet = 0;

wire [15:0] do_out;  // ADC value; useful part are only [15:4] bits

wire [4 : 0] channel_out;

wire eoc_out;

xadc_wiz_1 xadc_inst(
  //.di_in(di_in),              // input wire [15 : 0] di_in
  .daddr_in(channel_out),        // input wire [6 : 0] daddr_in
  .den_in(eoc_out),            // input wire den_in
  .dwe_in(1'b0),            // input wire dwe_in
  //.drdy_out(drdy_out),        // output wire drdy_out
  .do_out(do_out),            // output wire [15 : 0] do_out
  .dclk_in(clk),          // input wire dclk_in
  .reset_in(0),        // input wire reset_in
  .vauxp6(analog_pos_in),            // note since vauxn5, channel 5, is used  .daddr_in(ADC_ADDRESS), ADC_ADRESS = 15h, i.e., 010101
  .vauxn6(analog_neg_in),            // note since vauxn5, channel 5, is used  .daddr_in(ADC_ADDRESS), ADC_ADRESS = 15h, i.e., 010101
  .channel_out(channel_out),  // output wire [4 : 0] channel_out
  .eoc_out(eoc_out),          // output wire eoc_out
  .alarm_out(0),      // output wire alarm_out
  .eos_out(0),         // output wire eos_out
  .busy_out(0)        // output wire busy_out
);


//assign value_in =  do_out[15:4];

    initial begin
        pcount = 23'b0;
        platform1 = 1'b0;
        n_trigger = 1'b0;
        old_pulse = 1'b0;
        count_out = 33'b0;
        pulse_count = 33'b0;
        status = 33'b0;
    end

// Define an integer parameter for the scaling factor

reg [15:0] value_in; // reg to hold XADC output
reg [3:0] tens;  // two 4-bit decimal outputs
reg [3:0] ones;

//Temp Sensor
always @ (posedge clk) begin
    if(temptoggle == 1) begin

        if (tempcount == 50000000) begin // checks temp. reading every .5 seconds

            value_in = do_out >> 4; // sets value_in to 12 MSBs of do_out

            // Calculate the ones and tens digits
            tens = (value_in - 1305) / 310;
            ones = ((value_in - 1305) / 31) % 10;

            tempcount = 0;
        end

        tempcount = tempcount + 1;
    end
end


//Ultrasonic Sensor Code
always @(posedge clk) begin
    if (ultratoggle == 1) begin
        // Always add one to count, this is the overall counter for the script
        pcount = pcount + 1;
        // This is used to pick a very specific timeout for trigger control
        n_trigger = ~&(pcount[22:10]);
            if (n_trigger) begin
                if (pulse == 1) begin
                    pulse_count = pulse_count + 1;
                end
                if ((old_pulse == 1) && (pulse == 0)) begin
                    count_out = pulse_count;
                    pulse_count = 0;
                end
            end
        // This number is the distance control; the status is used as a buffer for filtering (OR filter below)
        if (count_out < 33'b11111100010000000) begin
            status = status << 1;
            status[0] = 1;
        end else begin
            status = status << 1;
            status[0] = 0;
        end
        old_pulse = pulse;
    end
end

assign trigger = n_trigger;
// Logic OR filter status here
assign state = |status;


    // Speed control for motor A
    always @(*) begin
    case ({speed_a[1], speed_a[0]})
        2'b00: duty_cycle_a = 21'd0;              // 0%
        2'b01: duty_cycle_a = 21'd1153433;         // 55%
        2'b10: duty_cycle_a = 21'd1468006;        // 70%
        2'b11: duty_cycle_a = 21'd2097151;        // 100%
        default: duty_cycle_a = 21'd0;            // Default to 0% if neither case matches
    endcase
    end

    // Speed control for motor B
    always @(*) begin
    case ({speed_b[1], speed_b[0]})
        2'b00: duty_cycle_b = 21'd0;              // 0%
        2'b01: duty_cycle_b = 21'd1153433;         // 55%
        2'b10: duty_cycle_b = 21'd1468006;        // 70%
        2'b11: duty_cycle_b = 21'd2097151;        // 100%
        default: duty_cycle_b = 21'd0;            // Default to 0% if neither case matches
    endcase
    end


    // Direction control for motors
    always @(posedge clk) begin
        if(delaylatch == 0) begin // this triggers if it hasn't reached the crossroads yet
            if (ip_a == 0 && ip_c == 0 && ip_b == 0) begin
                delaylatch = 1;
            end

            else if (ip_b == 0 && ip_a == 1 && ip_c == 1) begin
                if (delay_trigger == 0)
                    delay_trigger = 1;
                else if (delay_done == 1) begin
                    delay_trigger = 0;
                    input2 <= 0;
                    input1 <= 1;
                    input4 <= 0;
                    input3 <= 1;
                end
            end

            else if(ip_c == 0 && ip_a == 1 && ip_b == 1) begin
                input2 <= 0;
                input1 <= 1;
                input4 <= 1;
                input3 <= 0;
            end

            else if(ip_a == 0 && ip_b == 1 && ip_c == 1) begin
                input1 <= 0;
                input2 <= 1;
                input3 <= 1;
                input4 <= 0;

            end else begin
                input2 <= 0;
                input1 <= 1;
                input4 <= 0;
                input3 <= 1;
            end
        end else begin // this triggers when and after it reaches the crossroads the first time
            if(turndelay < 27'd90217700) begin
                turndelay = turndelay + 1;
                input2 <= 1;
                input1 <= 0;
                input4 <= 0;
                input3 <= 1;
            end else begin
                if (ip_b == 0 && ip_a == 1 && ip_c == 1) begin
                if (delay_trigger == 0)
                    delay_trigger = 1;
                else if (delay_done == 1) begin
                    delay_trigger = 0;
                    input2 <= 0;
                    input1 <= 1;
                    input4 <= 0;
                    input3 <= 1;
                end
            end

                else if (ip_a == 0 && ip_b == 1 && ip_c == 1) begin
                    input1 <= 0;
                    input2 <= 1;
                    input3 <= 1;
                    input4 <= 0;
                end else if (ip_a == 1 && ip_b == 1 && ip_c == 0) begin
                    input2 <= 0;
                    input1 <= 1;
                    input4 <= 1;
                    input3 <= 0;
                end else if (ip_a == 1 && ip_b == 0 && ip_c == 0) begin
                    input2 <= 0;
                    input1 <= 1;
                    input4 <= 1;
                    input3 <= 0;
                end else begin
                    input2 <= 0;
                    input1 <= 1;
                    input4 <= 0;
                    input3 <= 1;
                end
            end
        end
    end
    always @(posedge clk) begin
        if(delay_trigger == 1) begin
            if (controldelay < 26'd50108860) begin
```

```
                controldelay <= controldelay + 1;
                delay_done <= 0;
            end else begin
                controldelay <= 0;
                delay_done <= 1;
            end
        end
    end



// PWM generation
always @(posedge clk) begin
    count <= count + 1;
    if (rst) begin
        counter <= 0;
        delay <= 0;
        pwm_a <= 0;
        pwm_b <= 0;

    end else begin
        // Increment counter on each clock cycle
        counter <= counter + 1;

        if (current_threshold_a || current_threshold_b) begin
        // Increment the delay counter and keep latch low.
        delay <= delay + 1;
            // Check if the delay counter is less than a threshold value
            if (delay >= 27'd134217720)
                // Once the delay threshold is reached, set latch high.
                latch <= 1;
            end

        if (latch==0) begin
        // Update PWM output for motor A based on duty cycle
            if (counter < duty_cycle_a)
                pwm_a <= 1;
            else
                pwm_a <= 0;

            // Update PWM output for motor B based on duty cycle
            if (counter < duty_cycle_b)
                pwm_b <= 1;
            else
                pwm_b <= 0;
        end else begin
            pwm_a <= 0;
            pwm_b <= 0;
            if(BtnC) // latch holds until button c is pushed
            latch <= 0;
            delay <= 0;
        end

        if(state == 1 && ultratoggle == 1) begin
            pwm_a <= 0;
            pwm_b <= 0;

        if(platform1 == 0) begin
            temptoggle = 1;
            tempdelay = tempdelay + 1;
            if(tempdelay == 200000000) begin // 2 second delay to read temperature
                if(station1 == 1 && station2 == 0 && station3 == 0 && station4 == 0) begin
                    if(value_in > 1843 && value_in <= 2375) begin
                        platform1 = 0;
                        magnet = 1;
                        ultratoggle = 0;
                        tempdelay = 0;
                        station2 = 1;
                    end else begin
                        ultratoggle = 0;
                        tempdelay = 0;
                    end
                    end if (station2 == 1 && station1 == 1 && station3 == 0 && station4 == 0) begin
                        if(value_in > 2375) begin
                            platform1 = 1;
                            magnet = 0;
                            ultratoggle = 0;
                            tempdelay = 0;
                            station3 = 1;
                        end else begin
                            ultratoggle = 0;
                            tempdelay = 0;
                        end
                    end if (station3 == 1 && station2 == 1 && station1 == 1 && station4 == 0) begin
                        if(value_in <= 1843) begin
                            platform1 = 1;
                            magnet = 0;
                            ultratoggle = 0;
                            tempdelay = 0;
                            station4 = 1;
                        end else begin
                            ultratoggle = 0;
                            tempdelay = 0;
                        end
                    end if (station4 == 1 && station3 == 1 && station2 == 1 && station1 == 1) begin
                        if(value_in > 1843 && value_in <= 2375) begin
                            platform1 = 1;
                            magnet = 0;
                            ultratoggle = 0;
                            tempdelay = 0;
                        end else begin
                            ultratoggle = 0;
                            tempdelay = 0;
                        end
                    end
                end
            end else begin
                magnet = 1;
                ultratoggle = 0;
                platform1 = 0;
            end

        end else begin
            temptoggle = 0;
        end
        if(ultratoggle == 0) begin
            ultracount = ultracount + 1;
            if(platform1 == 0) begin
                if(ultracount == 250000000) begin // 2.5 second delay
                    ultratoggle = 1;
                    ultracount = 0;
                end
            end else begin
                if(ultracount == 80000000) begin // .8 second delay
                    ultratoggle = 1;
                    ultracount = 0;
                    end
                end
            end
    end
end



    // LED control signals (for testing, connect to LEDs on the board)
    always @(*) begin
        // Control LED_A based on the state of motor A
        led_a = pwm_a;

        // Control LED_B based on the state of motor B
        led_b = pwm_b;

        led_c = latch; //shows when latch is enabled

        led1 = magnet;
        led2 = temptoggle;
        led3 = platform1;
        led4 = ultratoggle;
        led5 = station1;
        led6 = station2;
        led7 = station3;
        led8 = station4;

        end

    always @ (*)
    begin
      case(count[N-1:N-2]) //using only the 2 MSB's of the counter

       2'b00 :  //When the 2 MSB's are 00 enable the fourth display
                begin
                    if (res)
                        s_temp = DASH; // Display '-'
                    else if (oc_a || oc_b)
                        s_temp = ZERO; // Display '0'
                    else
                        s_temp = NOTHING; // Display ' '
                    an_temp = EN4;
                 end

            2'b01:  //When the 2 MSB's are 01 enable the third display
                begin
                    if (res)
                        s_temp = DASH; // Display '-'
                    else
                        s_temp = CELSIUS; // Display 'c'
                    an_temp = EN3;
                end

            2'b10:  //When the 2 MSB's are 10 enable the second display
                begin
                    if (res)
                        s_temp = DASH; // Display '-'
                    else
                        case (ones)
                            0 : s_temp = ZERO;
                            1 : s_temp = ONE;
                            2 : s_temp = TWO;
                            3 : s_temp = THREE;
                            4 : s_temp = FOUR;
                            5 : s_temp = FIVE;
                            6 : s_temp = SIX;
                            7 : s_temp = SEVEN;
                            8 : s_temp = EIGHT;
                            9 : s_temp = NINE;
                            default : s_temp = DASH;
                        endcase
                    an_temp = EN2;
                end

            2'b11:  //When the 2 MSB's are 11 enable the first display
                begin
                    if (res)
                        s_temp = DASH; // Display '-'
                    else
                        case (tens)
                            0 : s_temp = ZERO;
                            1 : s_temp = ONE;
                            2 : s_temp = TWO;
                            3 : s_temp = THREE;
                            4 : s_temp = FOUR;
                            5 : s_temp = FIVE;
                            6 : s_temp = SIX;
                            7 : s_temp = SEVEN;
                            8 : s_temp = EIGHT;
                            9 : s_temp = NINE;
                            default : s_temp = DASH;
                        endcase
                    an_temp = EN1;
                end
      endcase
     end

    assign an = an_temp; //because you can't manipulate outputs

    assign {g, f, e, d, c, b, a} = sseg_temp; //concatenate the outputs to the register

    assign dp = 1'b1; //since the decimal point is not needed, all 4 of them are turned off

    assign magnettoggle = magnet;

endmodule
```

Appendix E

WRITTEN LAB REPORT EVALUATION FORM

Student Name:

Course Number:

Instructor:

Date:

Please score the student by circling one of the responses following each of the statements.

1) The student's writing style (clarity, directness, grammar, spelling, style, format, etc)

A	B	C	D	F	Zero

2) The quality and level of technical content of the student's report

A	B	C	D	F	Zero

3) The quality of results and conclusions

A	B	C	D	F	Zero

4) Quality of measurements planned / taken

A	B	C	D	F	Zero

5) Appropriate engineering standards employed

A	B	C	D	F	Zero

6) Multiple realistic constraints considered

A	B	C	D	F	Zero

7) Properly utilized knowledge and skills acquired in earlier course work

A	B	C	D	F	Zero

Grade:

Appendix F

ORAL PRESENTATION EVALUATION FORM

Student Name:

Course Number:

Instructor:

Date:

Please score the student by circling one of the responses following each of the statements.

1) The student's apparent theoretical preparation

A	B	C	D	F

2) Quality of the student's specific design

A	B	C	D	F

3) The quality of the student's presentation

A	B	C	D	F

4) The student's ability to answer questions

A	B	C	D	F

5) The student's attitude toward the lab (initiative, ability to self-direct, team work, etc.)

A	B	C	D	F

Grade: