Technical Paper for Texas Tech University

Search and Rescue Rover

Mario Martinez

R11434349

Texas Tech University

May 9, 2017

Abstract

This technical paper describes the process of how a BASYS3 board is used to control a rescue rover. The rover can identify and locate specific targets on a playing field. Once the target is located, the robot will then identify the target to the rescuers. The BASYS board is coded using Verilog language in Vivado which is later integrated to the hardware that is designed using circuit simulation software like LTSpice. The hardware includes multiple Microphone sensors (BOB-12758) to detect the various frequencies on a “field” that represent the targets, an inductive sensor to detect reflective tape to prevent running over the target, and an H bridge board to control the speed and direction of the motors. The rover will take on the appearance of Disney & Pixar’s Wall-E. The paper elaborates the process of how the system is designed and how the hardware communicates with the software. The paper also includes images, schematics and code snippets with detailed explanations for each section for reference.

Table of Contents

1. List of Figures……...…………………………………….……………….……….…….3
2. Introduction……..…….……………………………………...……………...……….….4
3. Current Detector……….……………….……………………………….……...…….…5
4. Current Detector Code….……………….…………………….…………………..…...6
5. Sound Detection..……………………………………………………...…………...…..7
6. Sound Detector Code…..……………………………………………...………...……..8
7. Motor Driver…………….…………………………....……………...…………...…...…9
8. Motor Driver Code …..…………………………...……………………..….............….9
9. Inductive Sensor………………………………………………………………..……..10
10. Inductive Sensor Code………………………………………………………………..11
11. LED Circuit……………………………………………………………………………..11
12. Rover Design…………………………………………………………………………..12
13. Acknowledgment Statement ………………………………………………………...13
14. Conclusion……………………………………………………………………………...13
15. Appendix A: Gantt Chart.……………………………..…………………..…….…….15
16. Appendix B: Budget …….………………………….…………………...……….……18
17. Appendix C: Code……….…………………………………….….………………..…19

List of Figures

Figure 1: Rover 5………………………...………...………………………….……..…………4 Figure 2: BASYS 3……………………...………………………...….………………………...4 Figure 3: Current Detector Circuit...…………………………………………..………………5

Figure 4: Final Current Detector Circuit…………………………………………...…….………6

Figure 5: Example of Current Detected………………………………………...…….………6

Figure 6: Amp Detector Code………....…………………………………...……………..…..6

Figure 7: Amp Detector Flowchart………………....………………………..………………..6 Figure 8: BOB-12758……………………....…………………………………………………...7 Figure 9: LM339 IC chip.……………………….………………………….…………………...7 Figure 10: Square Wave Example of Sound Detected...………………………..…...……8 Figure 11: H-Bridge…………………..…………………………..…………………………..…9 Figure 12 : Motor Driver Code Example…………….…………………………………….9-10 Figure 13: Inductive sensor circuit…………...…………..…………………………….……10

Figure 14: LED Circuit…………….……………………...……...……………………………11 Figure 15: Rover Body…………….……………………...…..……………………………12 Figure 16: Body Insides………………………………………..…………....…………..…12

Figure 17: Final Design………………………………………..…………....…………..…12

Introduction

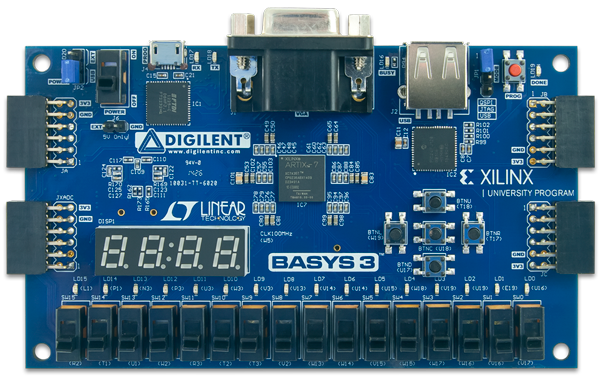
The main objective for the system is to use a BASYS board controlled Rover 5 (Figure 1) with sensors that can detect a specific frequency, 9Khz, 10Khz, 11Khz, and an alternating 12 Khz - 13Khz, played in a certain pattern. These frequencies are labeled as human, gas leak, mayday sound, and home respectively. The rover will travel towards its target frequency, provide a signal indicating the targets location in the form of red, green, and blue LEDs, then travel back to its original destination, home base. To achieve this, two microphones, one in the front and the other in the back, will identify the frequency and location. Also, the rover motors are connected to an H bridge controlled by the BASYS board (Figure 2). To stop the rover from running over the frequency sources, a reflecting tape will be placed around them allowing the inductive sensor to detect the tape and stopping the rover. To prevent the rover motors from burning out, an amp detector circuit will measure the current of the rover motors. If the current rises above 1 amp for more than a certain amount of time, the rover motors stop moving until ready again.



Figure 2: BASYS 3

Figure 1: Rover 5

Current Detector

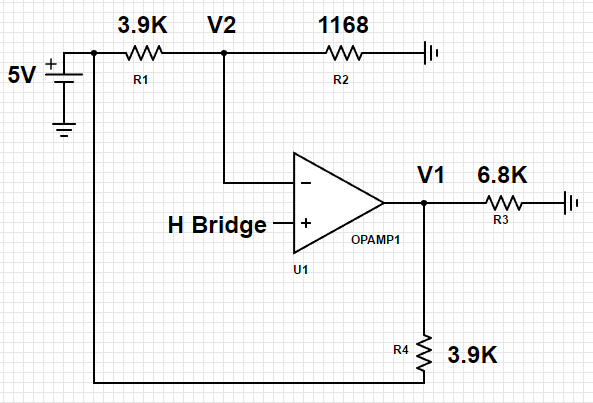
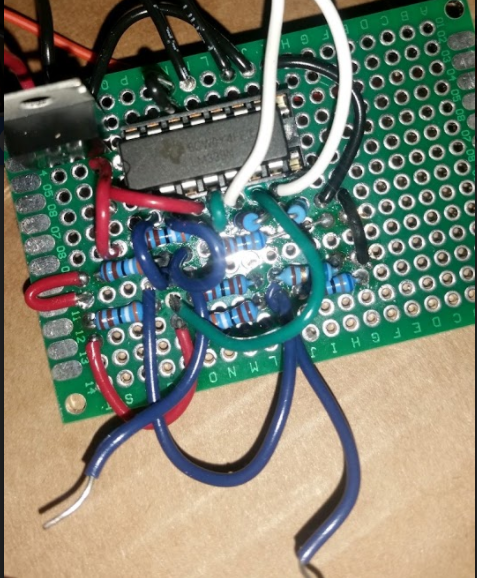
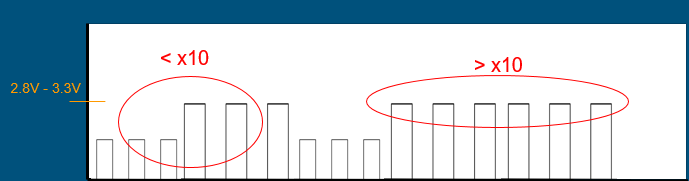
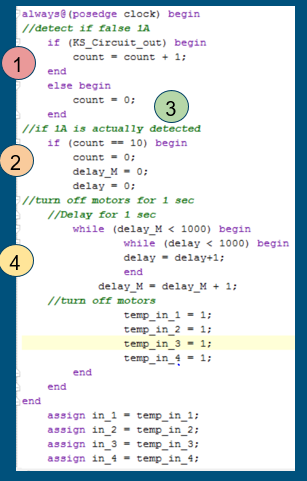
 The current detector will measure the current in the motor. If the motor exceeds 1 amp, the motor will stop for a few seconds, and then continue. The circuit (Figure 3) utilizes a LM358N op amp with 1.1V (V2) going into the inverting terminal and a range between 2.8V – 3.3V, V1, going into the BASYS board. When less than 1A goes into the inverting terminal, the op amp will output a high, or a 1 to the BASYS board. In the final circuit, two current circuits were soldered onto the same breadboard.

Figure 4: Final Current Detector Circuit

Figure 3: Current Detector Circuit

In Figure 4, the blue wires, denoted by the arrows, are to be connected into the BASYS board while the white wires, denoted but white arrows, were to be connected to the H bridge.

The current detector circuit will output either a zero or a one into the BASYS board. Zero if the current falls below 1A and a one if the current is 1A or greater. The idea is to do a check to see if 1A is actually detected and if so, stop the motor(Figure 5). The current may sometimes rise above 1A for a brief moment then return to below 1A. This is not enough time to do damage to the board. Because of this, the rover will stop whenever 1A is detected, even if for a brief moment, hindering movement. To fix this, code will be written to actually detect if 1A is being detected for a certain amount of time (Figure 7).The code will detect whether or not the board can be damages, false 1A for no threat. If 1A is detected, then the motors will be turned off for about one second (Figure 6).

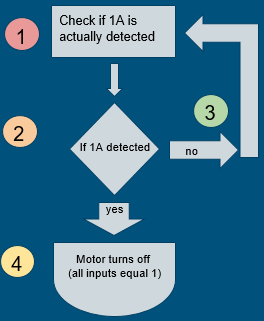


Figure 6: Amp Detector Code

Figure 5: Example of Current Detected

Figure 7: Amp Detector Flowchart

Sound Detection

The sound detection circuit will identify the frequency picked up and determines the location of the frequency source. The target frequencies are 9 Khz - 13Khz. When the target frequency is detected, the rover will turn towards the frequency location and head towards it. Two BOB-12758 microphones (Figure 7) are used to pick up the sinusoidal sound wave. Because the BASYS board only reads square waves, an LM339 IC chip (Figure 8) is used to convert the sinusoidal waves into a square wave. In order to receive a clean signal with the desired frequency, a bandpass filter was designed by cascading a low pass filter with a high pass filter.

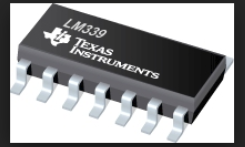


Figure 8: BOB-12758

Figure 9: LM339 IC chip

Sound Detection Code

Once a square wave can be picked up by the BASYS board, code can be written to distinguish between human, gas leak, mayday call, and home. The idea is to count the peaks of the square wave (Figure 9), and determine whether it matches the pattern for a human, gas leak, mayday call, or home. The pattern for a human is, at 9 kHz, 3 short 100 ms burst followed by 3 long 500 ms bursts and finally 3 more short 100 ms bursts repeated with a one second delay. The pattern for a gas leak is a repetitive single frequency hiss at 10 kHz bursting for one second and off for 500 ms. The pattern for a mayday sound is at 11 kHz, five 25 ms bursts of on then off (on 25mn then off 25mn) every 500 ms. The pattern for home base is an alternating 12 kHz and 13 kHz frequency, 200 ms each (200mn 12 Khz, then 200ms 13 Khz).

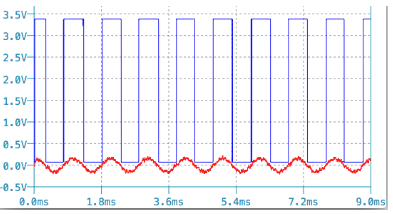


Figure 10: Square Wave Example of Sound Detected

Motor Driver

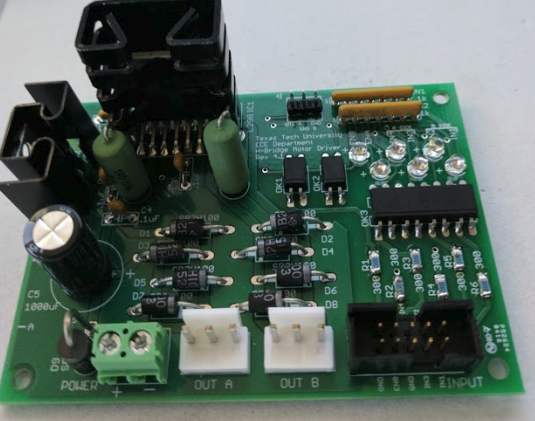
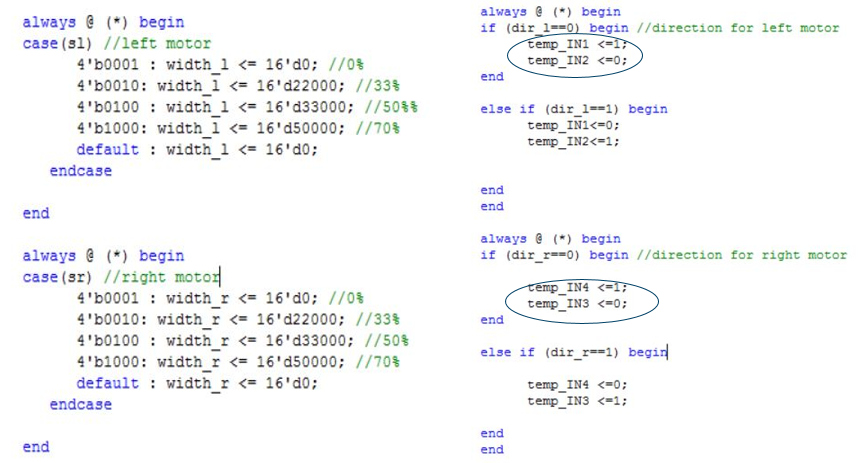
The motor driver circuit controls the speed and direction of the two motors on the tank chassis. Controlling the speed and direction allows for turning. The H bridge (Figure 10) connects to the chassis and controls the left and right motors. The BASYS board connects to the H bridge and allows movement in the direction of the frequency sound.

Figure 11: H-Bridge

Motor Driver Code

The motor code is divided into four parts (Figure 11), on/off of the left and right motors, and the direction of the left and right motors. The motors have five speeds, off, 33%, 50%, 70%, and max speed. The direction of the motor can be either forward or backward.



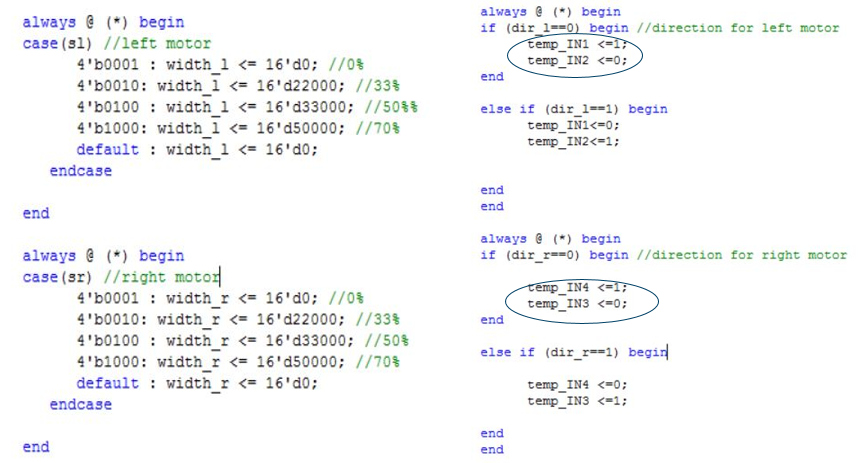
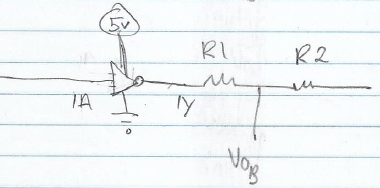


Figure 12: Motor Driver Code Example



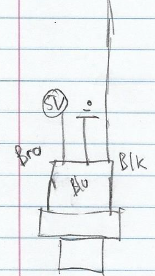
Inductive Sensor

Figure 13: Inductive Sensor

The inductive sensor will detect the metallic tape around the field and stops the rover from moving. The sensor will hang below the base in order to detect the reflective tape. Figure 12 is the sensor circuit. The 5V will supply the IPS. The sensor’s brown wire is powered by the 5V source while the Blue wire goes to ground. The resistors divide the voltage for the BASYS board and the black wire is connected to an inverter. Because the IPS gives an active low, an inverter is used to give an active high.

Figure 12: Inductive sensor circuit

Inductive Sensor Code

When the IPS gives an active high, a flag in the code stops the motors. The motors stay paused until the appropriate push button is pressed.

LED Circuit

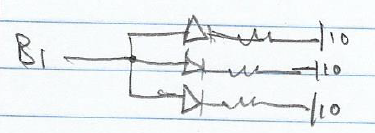


Figure 14: LED Circuit

An led circuit is used to signal the user that the rover has landed at its destination. The circuit displays a red light for the 10Khz signal, green light for the 11Khz signal, and a blue light for the 9Khz signal. The full circuit consists of three individual smaller circuits. Each circuit consists of three LEDs of the same color, each connected to a 100-ohm resistor, and connected to ground. The smaller circuits are each powered by an output on the BASYS board. Then the IPS gives an active high, the BASYS board will deliver power to the appropriate LEDs, depending on the sound signal.

Rover Design

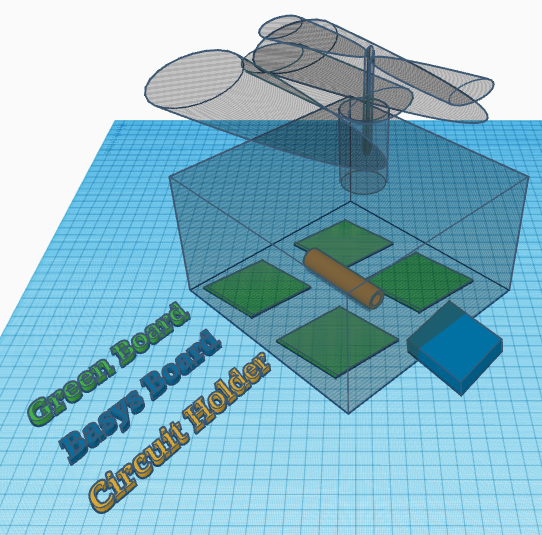
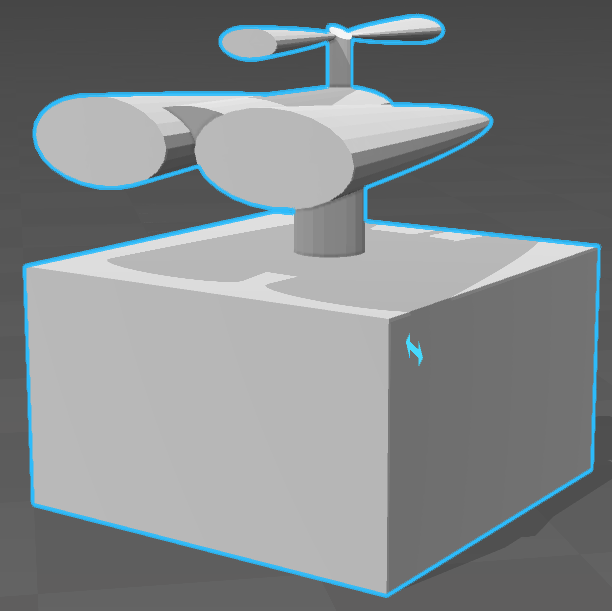
The rover body will be 3D printed. It will take the appearance of Disney and Pixar’s Wall-E (Figure 13). An antenna like object will sprout from the eyes/head. The mics will be placed there and its wires will run down through the antenna, and into the head. Inside the body (Figure 14) will be four green boards, holding the four different circuits, the BASYS board, and a circuit/wire holder. To make removal of the body easy, the body and antenna will be two separate pieces. Wires from the antenna will connect to the head. Wires will then run down the neck into the body and into four separate circuit boards. To keep wires neat inside, wires will run through a circuit holder inside

Figure 15: Rover Body

Figure 16: Body Insides

the body. At any time, the antenna can be detached from the body, and the body removed from the tank chassis.

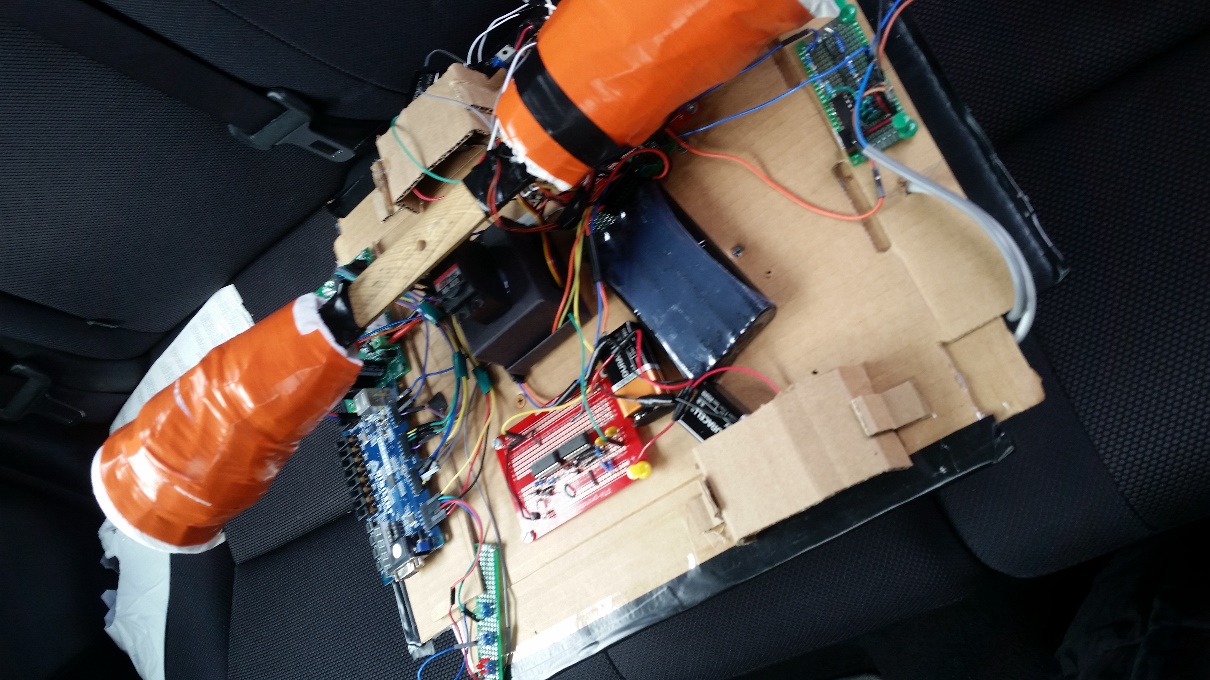
This design was scrapped due to time constraints. The final design consisted of a flat surface that housed all the circuits openly. The IPS was mounted below the base. The microphones were each housed in a cone, which rotate 180 degrees with a servo motor.

Figure 17: Final Design

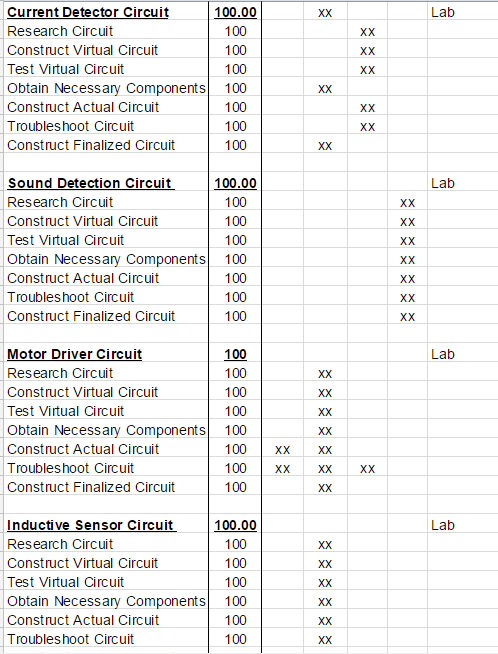
Acknowledge Statement

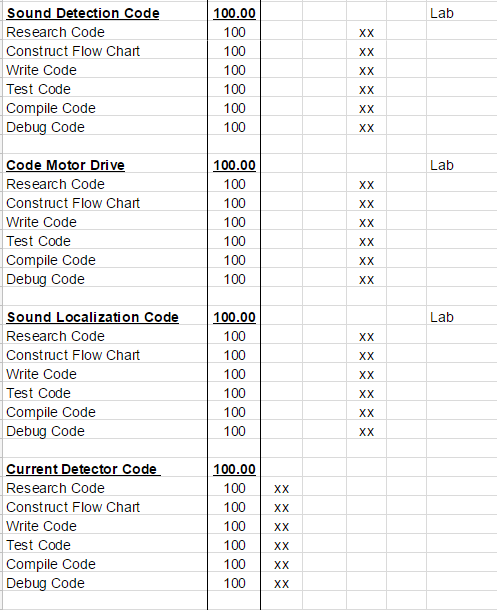
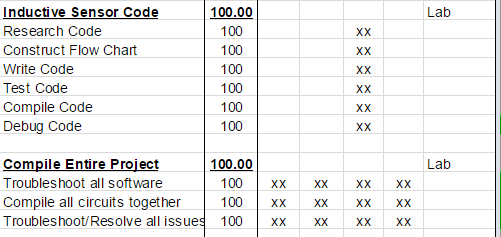
Thanks to the efforts of Daniel, the lab tutor, and Alan, Zarka, Derek, and Tony, progress was made when there was a problem that could not be solved.

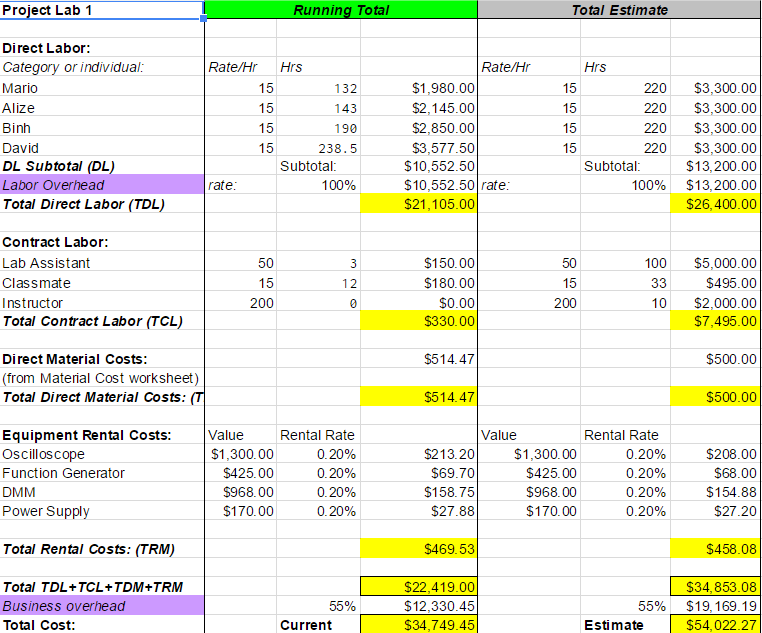
Conclusion

The entire system was mostly finished. With a little more time, it would have been possible to use the full WALL-E design and have it 3D printed. The inductive sensor and circuit worked perfectly. The motor and movement worked as well, with the exception of having a long response time due to the final design. Although the sound detection worked, the rover had trouble detecting sound when it was very close to the sound signal. All code module worked as expected except for the sound and movement.

Appendix A: Gantt Chart



Everything on the Gantt chart was finished. However, a lot of trouble shooting needed to be done. The sound detection was worked on the most and gave the most problems.

Appendix B: Budget 

The project remained underbudget. No one part of the budget has been exceeded. The estimate is still under $55K and the total amount was just under $35K

Appendix C: Code (Frequency Counter)

module Frequency\_Counter(

input clk4,

input freqcount,

output LED1,

output LED2,

output LED3,

output LED4

);

reg tempLED1;

reg tempLED2;

reg tempLED3;

reg tempLED4;

reg currentvalue;

reg prevalue;

reg [15:0]count;

reg flag;

initial begin

tempLED1 <=0;

tempLED2 <=0;

tempLED3 <=0;

tempLED4 <=0;

currentvalue<=0;

prevalue <=0;

count <=0;

end

//Assignments

assign LED1=tempLED1;

assign LED2=tempLED2;

assign LED3=tempLED3;

assign LED4=tempLED4;

always @ (posedge clk4)

begin

if (freqcount==1) //begins count

currentvalue<=1;

else if (freqcount==0)

currentvalue<=0;

if ((currentvalue==0)&&(prevalue==0)) begin //keeps count from starting

//if there is no rise

count<=count+0;

prevalue<=currentvalue;

flag<=0;

end

else if ((currentvalue==1)&&(prevalue==1)) begin //count continues if input stays high

count<=count+1;

prevalue<=currentvalue;

flag<=1;

end

else if ((prevalue==0)&&(currentvalue==1)) begin //starts count if input rises

count<=count+1;

prevalue<=currentvalue;

flag<=1;

end

else if ((prevalue==1)&&(currentvalue==0)) begin //stops count if input falls

count<=count+0;

prevalue<=currentvalue;

flag<=0;

end

if (flag==0) begin

if ((count>5618)) begin //below 9kHz

tempLED1 <=1;

tempLED2 <=1;

tempLED3 <=1;

tempLED4 <=1;

count<=0;

end

else if((count<=5618)&&(count>5435))begin //9kHz range

tempLED1 <=1;

tempLED2 <=0;

tempLED3 <=0;

tempLED4 <=0;

count<=0;

end

else if((count<=5051)&&(count>4902)) begin //10kHz range

tempLED1 <=0;

tempLED2 <=1;

tempLED3 <=0;

tempLED4 <=0;

count<=0;

end

else if((count<=4587)&&(count>4464)) begin //11 kHz range

tempLED1 <=0;

tempLED2 <=0;

tempLED3 <=1;

tempLED4 <=0;

count<=0;

end

else if((count<=4202)&&(count>=4098)) begin // 12kHz range

tempLED1 <=0;

tempLED2 <=0;

tempLED3 <=0;

tempLED4 <=1;

count<=0;

end

else if (count>4098) begin //Above 12kHz range (Sample ranges for Basys Board input)

tempLED1 <=1;

tempLED2 <=0;

tempLED3 <=0;

tempLED4 <=1;

count<=0;

end

end

end

endmodule