Boston University
Electrical & Computer Engineering
EC464 Capstone Senior Design Project

Second Prototype Test Report

Augmented Reality Climbing Wall

by

Team 14
Augmented Reality Climbing Wall

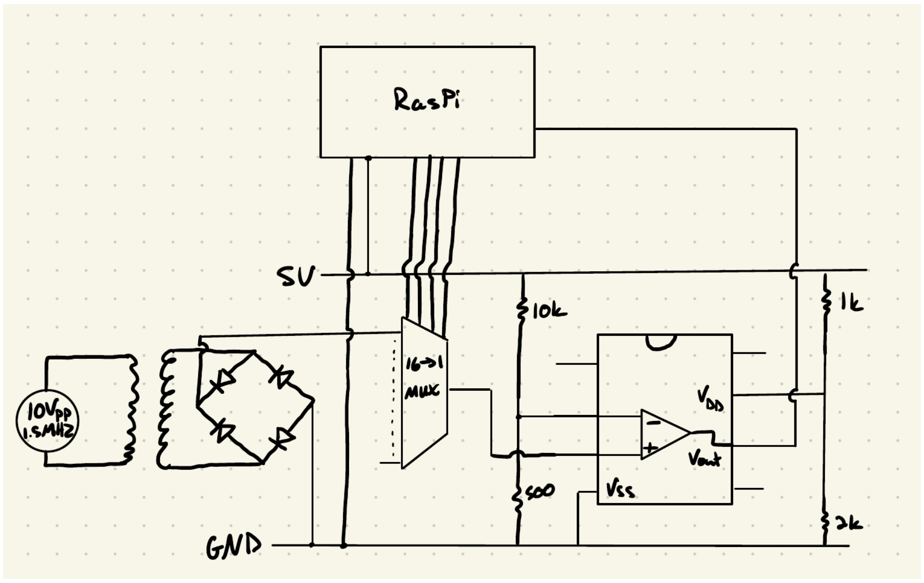
Team Members

Tom Panenko tompan@bu.edu
Taylor Hartman hartmant@bu.edu
Michael Igwe migwe@bu.edu
Ryan Smith rsmith66@bu.edu

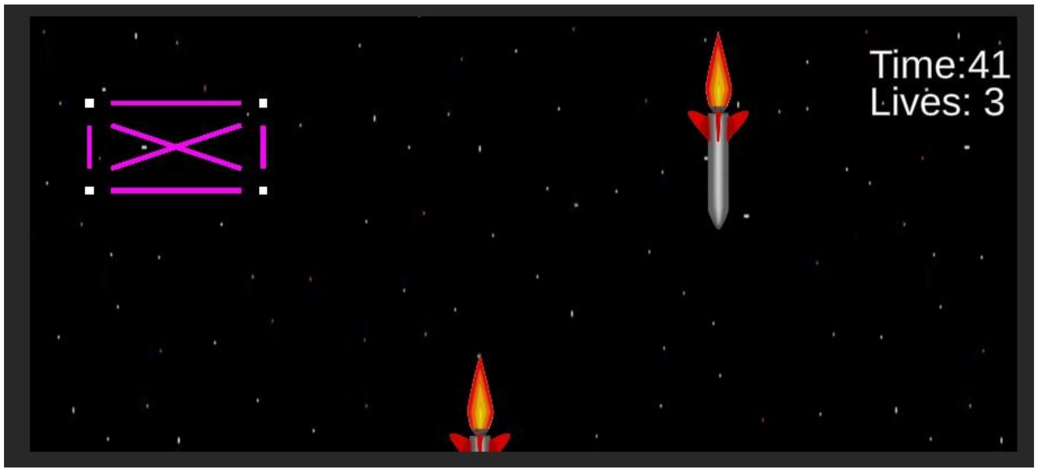
Equipment

- Circuitry
  - Full Bridge Rectifier (x2)
    - Diode (x4)
  - Comparator
    - MCP6241 Op Amp
    - Resistors (x4)
  - Multiplexor
  - Coils (x8)
  - Push Buttons (x4)
  - Function Generator
  - Power Supply
  - Jumper Wires
- Hardware
  - Enclosure (x2)
  - Button (x2)
  - Push Button Holder (x2)
  - Raspberry Pi 4 Model B
- Software
  - Laptop
  - Unity game engine
  - Game .cs scripts

Setup



We set up the multiplexing capability by connecting multiple buttons and holds to the multiplexer and grounding the unused inputs. We used the buttons to simulate additional holds, as we have not yet begun the process of mass producing them. After connecting everything to the multiplexer we connected the output to a comparator, which translated the signal to a constant 3.3V if the input was greater than 0.5V and a constant 0V if it was below. We then read this output from the raspberry pi and displayed the matrix of selected holds on our monitor, demonstrating that multiple holds could be depressed at once and the delay between depressing/releasing a button is negligible.



One of the features we planned on testing for this prototype testing is the game that will be played on the climbing wall. The game is played by the user holding onto the wall and dodging missiles that slowly fall down overhead. Each time a user is hit, he or she loses a life, and the game ends once a user has survived for 45 seconds or has lost all three lives. This game was coded in C# on the Unity Game Engine. The game is calibrated to work with the climbing wall by first taking in a calibration.txt file that contains a matrix used to set the location for each of the holds and takes in a data.txt file that notifies the game which holds are currently being pressed by the user using a matrix filled with 0s and 1s. Lines are drawn between the activated holds to simulate the body of the user, and the data.txt file is repeatedly checked to see if the activated holds have changed, and if so, the old holds are deactivated and the new ones are activated.

Measurements Taken

Here is the completed score sheet that was used to track results based off the test procedure in our Test Plan:

Action	Correct? (1/0)
Hold data console displays all 0s when no holds pressed	1
Hold data console correctly displays 1 in the row/column that the depressed hold is assigned to.	1
Hold data console correctly displays 1s for simultaneously depressed holds attached to the same multiplexer	1

Hold data console correctly displays 1s for simultaneously depressed holds attached to the same multiplexer index but different GPIO ports.	1
Game plays without error and within expectations for the gameplay	1
Game successfully calibrates hold position and activates the depressed holds	1
Game reads change in depressed holds and reacts successfully	1
Total	7/7

For the measurements taken during the testing, we were able to accurately meet all the Measurement Criteria we set in the Test Plan, specifically the multiplexing ability of our input reading system and the functionality of the game we made for the climbing wall. When the hold for the corresponding data port on the multiplexer was pressed, the data console correctly detected that the hold was pressed by outputting the number “1” on the console both for all tested conditions. Meanwhile, the game was able to read the text file of 1s and 0s and correctly activate the holds that were pressurized and adjust for changes in the holds. We also tested that the game we created runs properly, especially the “game over” condition and then takes in the proper information to update the game accordingly.

Conclusions

Our testing went as hoped, and we successfully scored a 7/7 on our measured objectives. This indicates that the software for our game works successfully and can take in inputs for the wall and our system can read which hold in specific is being pressed by the user. Professor Hirsch and the TAs seemed to be very happy with our testing and said that we were in a good place with the features we have shown. Some takeaways for next steps that we got from our testing was that the multiplexing works well for the individual holds that we tested, but now, we must test it with the four holds. Another issue we aim to work on is the efficiency of the power usage of the holds when the holds are all active. We saw that after a certain amount of usage the accuracy of the holds detection would decline so we want to reduce the voltage leakage of the holds when the system is both on and off. Overall, we were very pleased with the results of our testing and how our project performed and will continue working on the next steps we took away.