Boston University
Electrical & Computer Engineering
EC464 Capstone Senior Design Project

Final Prototype Testing Plan

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

Required Materials

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

Set Up

For this testing, we plan on running the game locally on a laptop and having it communicate with the raspberry pi to read the file that is being updated with the active sensor data matrix. On the laptop, the game is run on a local web browser and, to access the file information, uses a web server that shows the contents of the data.txt file and an additional proxy server that attaches CORS headers to every request that is made. This is necessary as the web server does not allow GET requests from other unconfirmed browsers, so these headers need to be added to get access to the data. The raspberry pi continually sends the data.txt file with the multiplexing data to the laptop and overwrites the file that is being read from in the web server.

The holds use AC power to transmit wirelessly through the wall. This power is then converted into DC through a full bridge rectifier and brought up to 0V or 3.3V with a comparator. The current hold is selected with a system of multiplexers that is designed to support up to 64 holds in total. This selection and reading post comparison is all controlled by the raspberry pi.

Pre-Testing Setup Procedure

1. Assemble the hold hardware by bolting the enclosure onto the test board with necessary buttons and electronics connected in between
2. Test with function generator and multimeter that wireless transfer with the hold is working correctly

3. Connect output of backside wireless power coils to full bridge rectifier and measure DC voltage
4. Connect output of full bridge rectifier to multiplexer, ground all non used inputs, and test functionality via multimeter
5. Connect output of multiplexer to comparator and test via Raspberry Pi
6. On the laptop, start Unity Engine and build and run the project for the climbing wall game
7. Begin the web server that has access to the data.txt file
8. Start the proxy server that adds the CORS Header to every request sent to it

Testing Procedure

1. Demonstrate depression of hold results in increased voltage at the output of full bridge rectifier
2. Demonstrate this increased voltage is successfully read by the software on the raspberry pi
3. Play through a game cycle demonstrate that the data is successfully passed from the raspberry pi and read by the game

Measurable Criteria

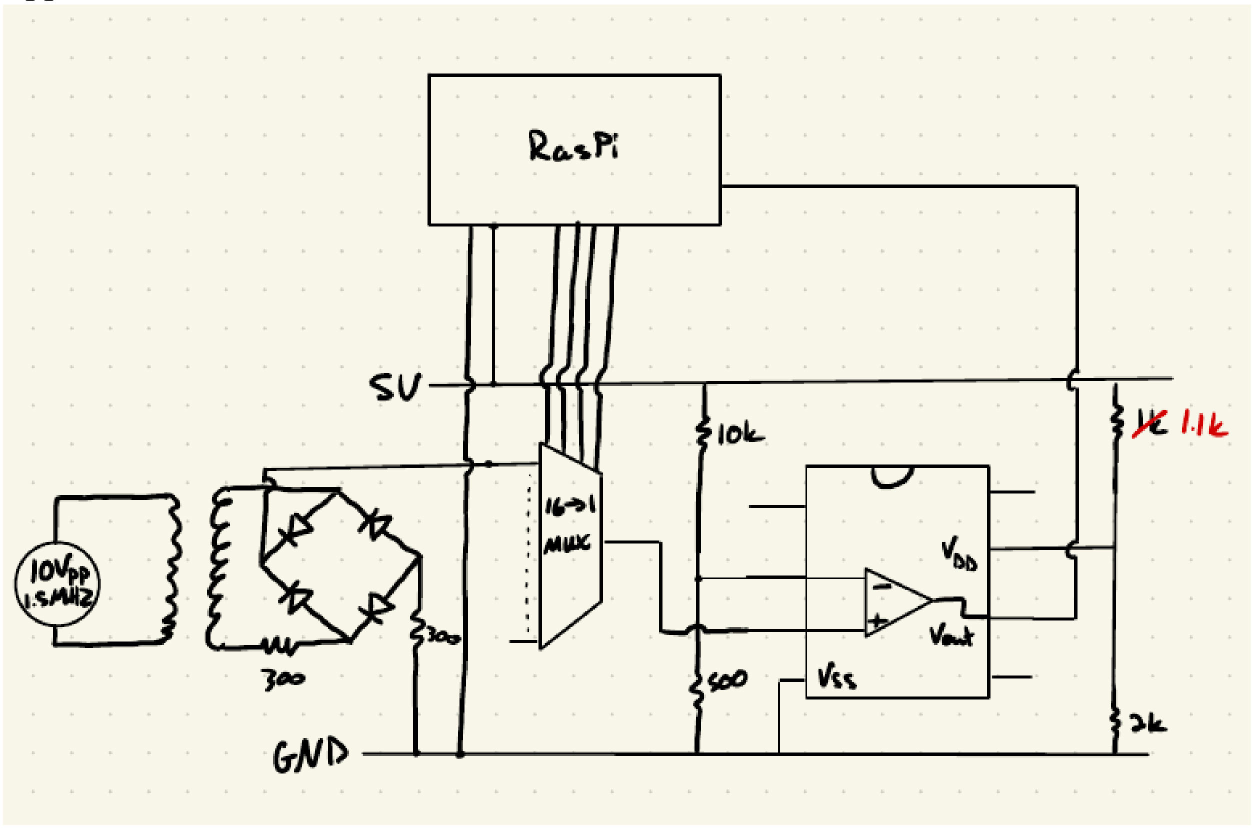
The criteria for successfully testing with the correct result is as follows:

I. During the testing procedure, all of the hold data bits are unset while all holds are unpressed.
II. When a hold is depressed, only its corresponding hold data bit is set and remains set while the hold is held.
III. When a hold is released, only its corresponding hold data bit is unset and remains unset until the hold is pressed again.

Score Sheet

Action	Correct? (1/0)
Hold data console displays all 0s when no holds pressed	
Hold data console correctly displays 1 in the row/column that the depressed hold is assigned to.	
Hold data console correctly displays 1s for simultaneously depressed holds	
Hold data console correctly displays 1s for simultaneously depressed holds attached to the same multiplexer index but different GPIO ports.	
Game plays without error and within expectations for the gameplay	
Game successfully calibrates hold position and activates the depressed holds	
Game reads change in depressed holds and reacts successfully	
Total	/7

Appendix


RasPi
5V
10k
~~1k~~ 1.1k
16→1 MUX
V_{DD}
V_{out}
V_{SS}
10Vpp 1.5MHz
300
300
500
2k
GND