# Design Review 2



## Subsystem Working Tests

### LED Strip

Our LED Strip, which will be used as the main roads and structures/cities, will run between all of the individual hexagons and be illuminated with our 5V bus after receiving instruction from the Pi. To confirm that this subsystem is working as expected, we hooked up a single strip of LEDs to a Raspberry Pi and tested if we could illuminate all of the lights, change the brightness of the LEDs (so we're not using too much current), change the colors of the LEDs (for the different players), and just connect to the LED Strip in general. Our testing was successful. The Strip managed to fully illuminate at the different colors that we required. Also, the brightness was able to be reduced such that we are only using, at most, 3A of current in a strip, but that's assuming all LEDs on white, which should never occur in an actual game.

Figure 1. Example of the LED Strip, illuminated using command from Raspberry Pi

## Hexagons

It is difficult to demonstrate the functionality of this subsystem in a document; nevertheless, we will describe our prototyping process and what we learned from the progress we have made.

First, we separately modeled the input (buttons) and output (LEDs) sides of the hexagon to ensure their functionality and make any necessary adjustments to the circuitry and part selections. This was largely successful on the first attempt, with only minor adjustments to the respective circuits being necessary. The $I^2C$ communications and ICs functioned perfectly and were very easy to use.

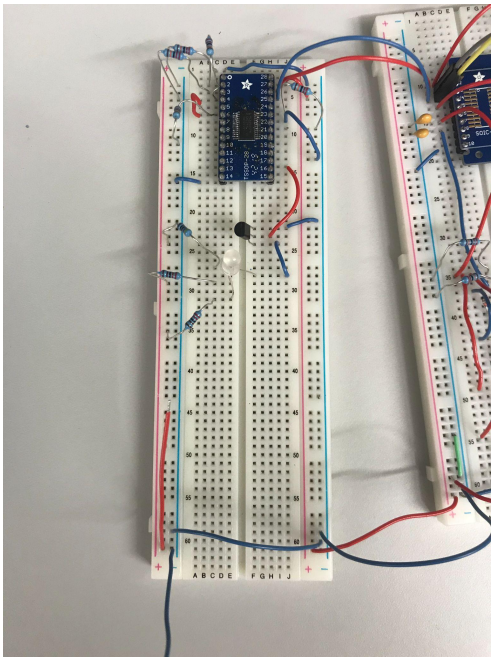

Figure 2. Showing the output side of the hexagon breadboard model: with RGB LED and PCA9685 LED driver as major components

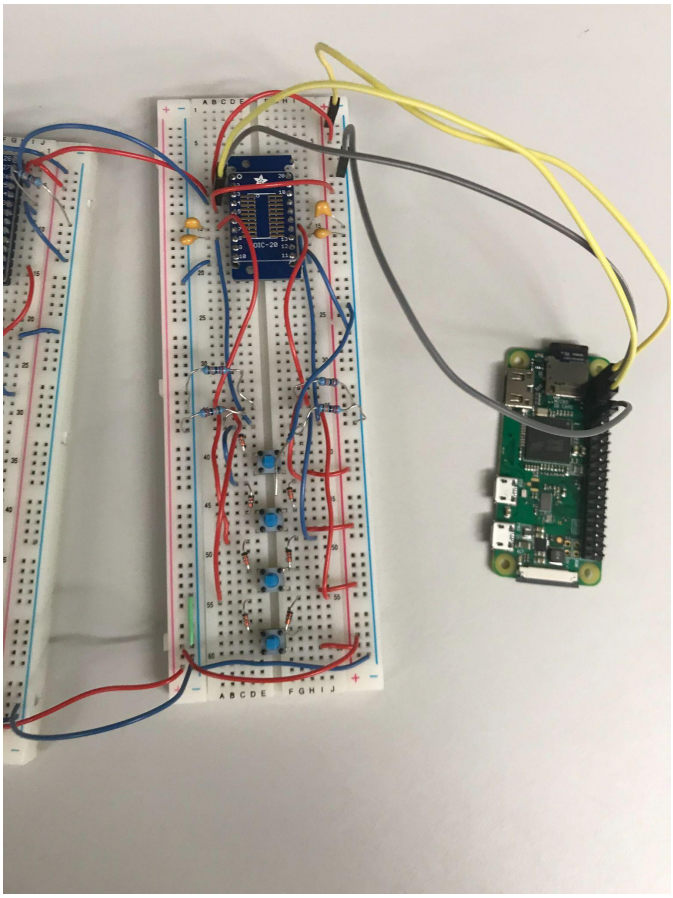

Figure 3. Showing the output side of the breadboard model with a reduced button matrix and PCA9501 I/O expander as major components. The Raspberry Pi is also visible.

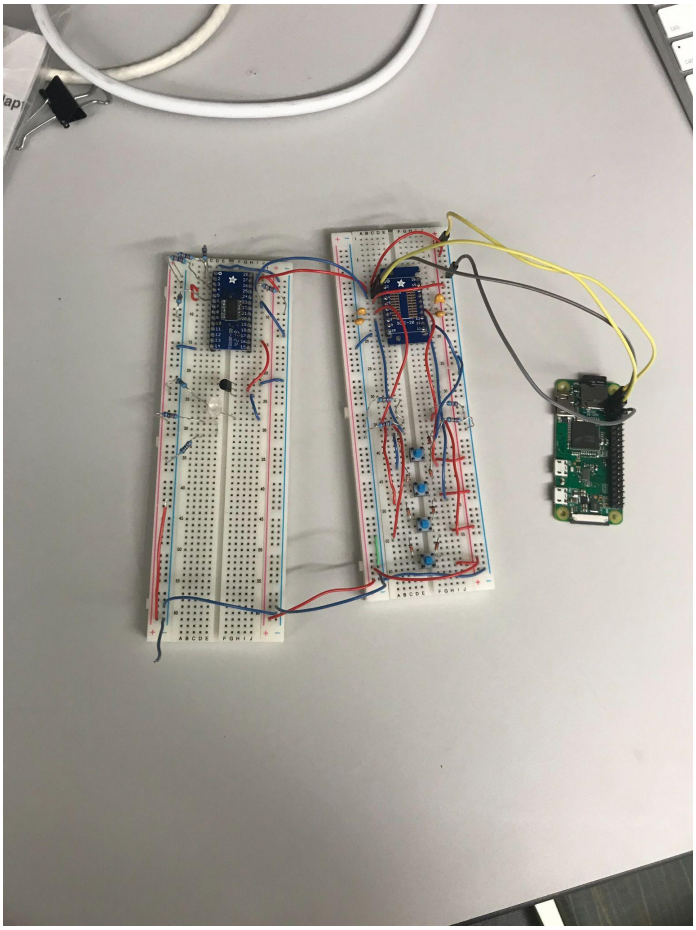

Figure 4. The complete hexagon prototype, after connecting the $I^2C$ and power buses common to all parts.

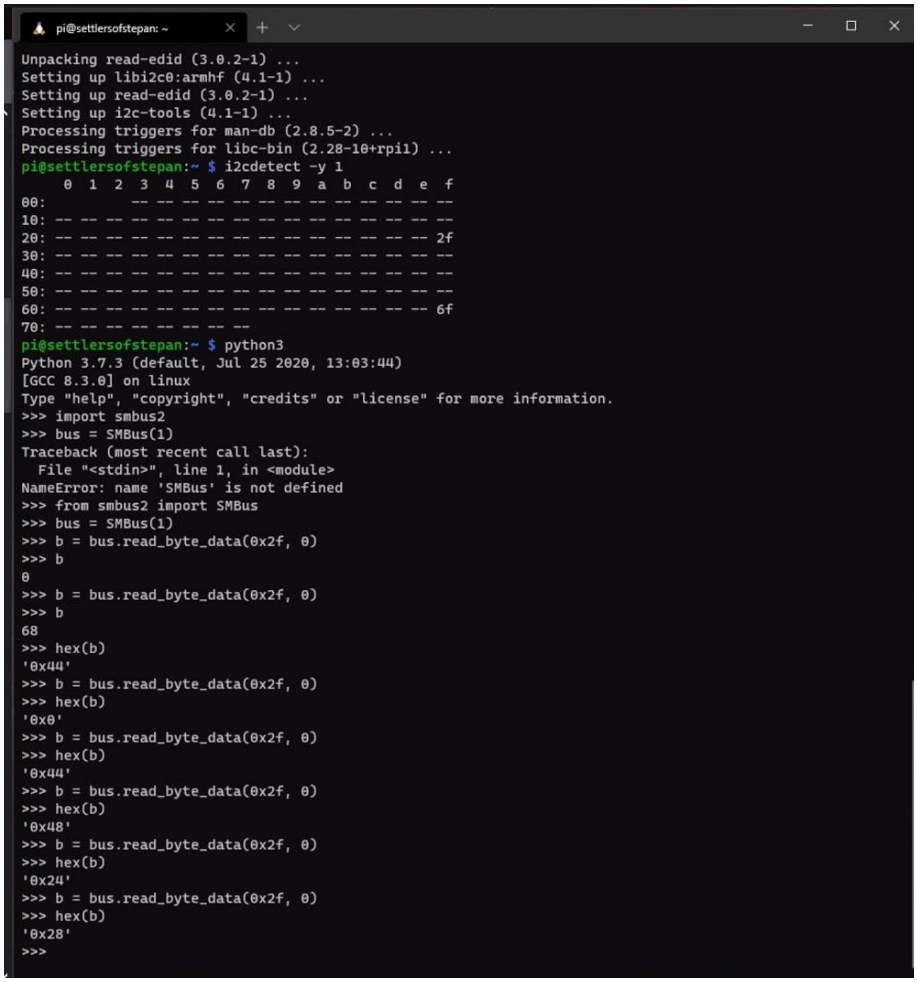


Figure 5. Console demonstrating the functionality of the input side of the hexagon prototype.

The figure above shows two things: at the top, the successful identification of the PCA9501 I/O expander on the $I^2C$ bus with its addresses--this demonstrates that the IC is connected properly and that both it and the Raspberry Pi can communicate well. The second thing shown is the data byte from the input pins of the PCA9501, resulting from various combinations of button presses. This shows that the input matrix logic is sound, and more importantly, that we can sense input from the PCA9501 successfully over $I^2C$.

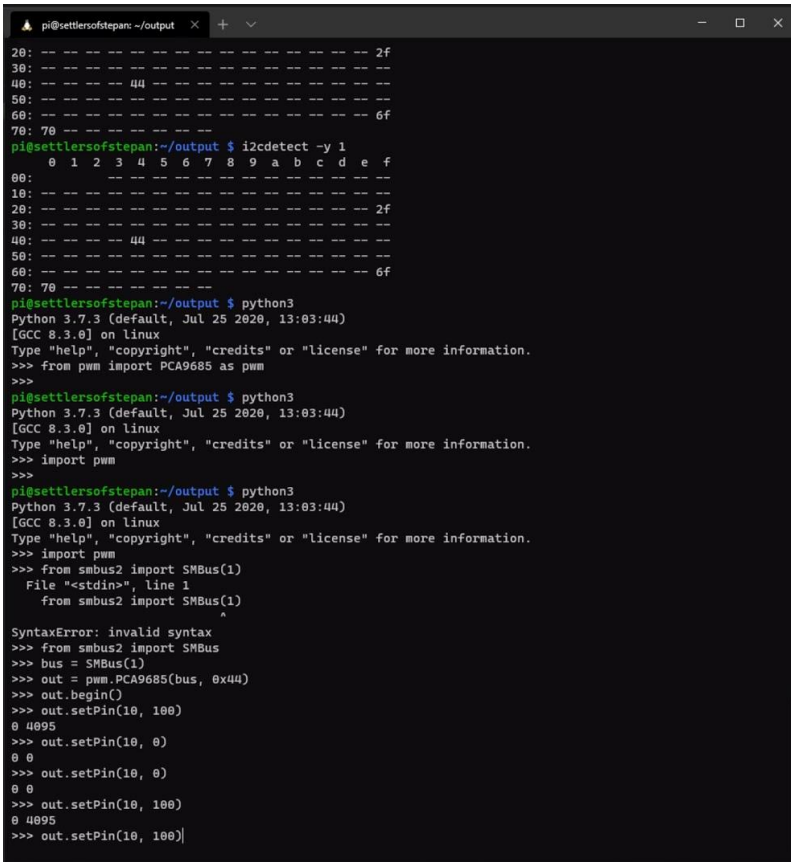


Figure 6. Console demonstrating the testing of the output side of the hexagon prototype

The above figure shows that the PCA9685 is present on the bus with its own respective addresses (you can see that the PCA9501 is also connected). At the bottom, we have set the output LED pin we are testing to various brightness levels, with successful results. Although we do not have a picture of the LED turned on, the functionality of using the LED driver over $I^2C$ to control LEDs worked just as expected with minimal need for improvements.

We could not demonstrate or prototype the 7-segment display portion of the hexagon because we did not order any for prototyping; however, we believe this is a relatively unproblematic part of the design and should not encounter any major roadblocks upon incorporating it into the real PCB hexagons.

## Game Website

The web-based GUI was completed and demonstrated as its own subsystem for Design Review 1. So, a demonstration of its functionality here would be redundant.

However, going beyond the base requirements of this Design Review, we have connected the web GUI and the game logic/server, and will demonstrate the connection between these subsystems using a simple example. This is one of the only inter-subsystem

connections we have left to demonstrate, well ahead of the nominal deadlines for the remaining Design Reviews.

A JavaScript library, Mithril, will be used to send data from the GUI to the game itself and to receive data from the game and display it on the GUI. This is ideal because both the GUI behavior and the game data are already implemented in JavaScript. It enables sending and receiving of data between a web server and any Python program (in this case, our game API) as JSON objects, which can be readily parsed and reflected in the GUI, or used to change game state information.

In this example, we use a button to send a small data payload to the game API server. It returns a different small payload of data to the web GUI. We can demonstrate the communication between these two subsystems by showing the data that the server received from the web GUI, and displaying the data the data received back from the API on the GUI itself, just like it would be in the real game.


Figure 7. Game API server log demonstrating receipt of data from the web GUI

This figure contains the log of the game API server which interacts with the GUI. The last line proves that the API has received a data object sent from the web GUI.

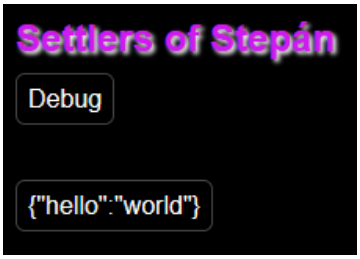


Figure 8. Test portion of GUI demonstrating receipt of data from the game API

This figure shows a test field on the web GUI itself displaying the contents of a data object that the game API sent back to it.

Already, we can demonstrate successful two-way communication between these two subsystems: the game API and the web-based GUI.

### Game API

Through breadboarding the Hexagon wiring, we verified that we can control our Hexagons with python3 and smbus2 on the Raspberry Pi (See ‘Hexagons’ section above).

We have written a server.py file that will function as a file server and run the game logic through a JavaScript API that the server.py provides. We have tested this application and verified that it can serve static files and an API well.

## Preliminary Board Design

We will order a prototype board run to test this design, and upon verifying and/or improving it, will use this design for our final PCB run.

### Layout

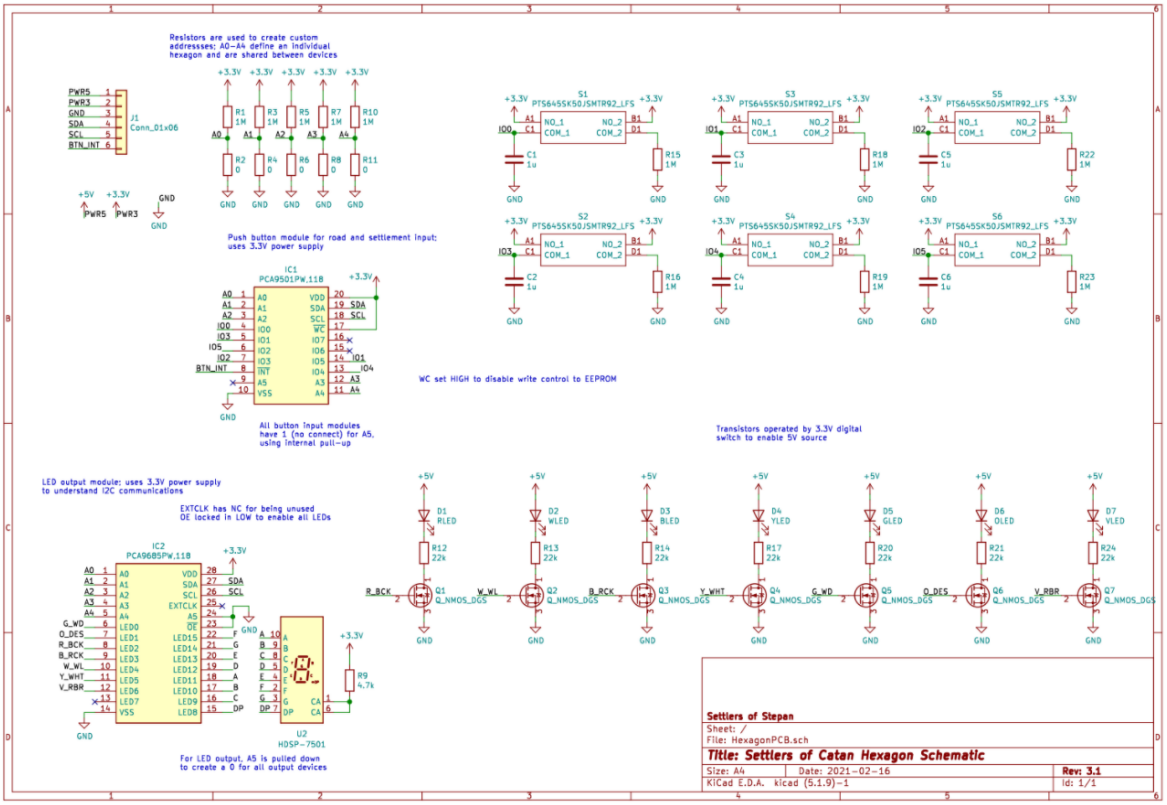


Figure 9. Board schematic, v3.1

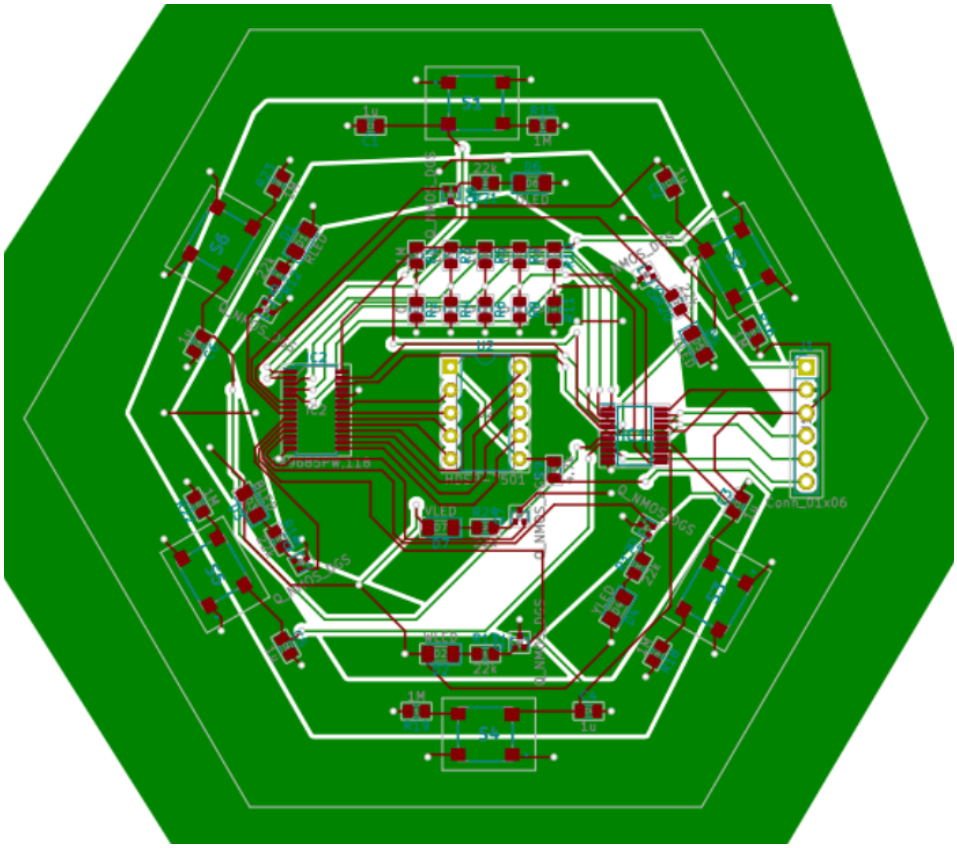

Figure 10. Board layout, v1.0

## Connections

The green area is the ground plane which is currently larger than the area of the board, denoted by the grey hexagon outline. This will obviously be trimmed before generating the board files.