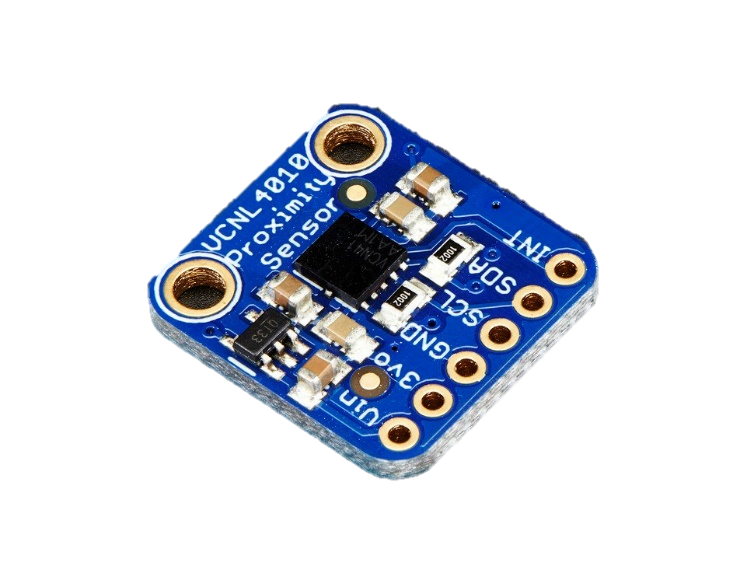
CENG 317: Hardware Production Tech – **Final Report**

**VCNL4010 Proximity/Light Sensor**

**Name:** Vikas Sharma

**Student ID:** N01160135

**Course Code:** CENG 317

**Instructor:** Austin Tian

**Section:** 0NC

**Date:** December 14, 2019

**Abstract:**

This report outlines the hardware development/process to integrate the proposed hardware with the software APP in CENG 319. The VCNL4010 is a Proximity/Ambient Light sensor that is used to detect ranges of distances far or nearby and provide proximity measurements for microcontrollers or devices that can make use of the sensor and its qualities. From a broader perspective, this report will go through the designing, prototyping, power – up and testing phases of the hardware production process. The project consisted of working with hardware sensors to communicate, record measurements to be able to read/write data. The main goal, being able to communicate with the software APP in CENG 319 and provide real-time measurements to the selected database interface. For our parking lot system model, the job of this hardware sensor is to be able to detect the status of a lot depending on how far, near a vehicle is from the sensor. In terms, of the project itself for the hardware portion, I only required to use the proximity feature of the sensor, excluding the ambient light results. This document will touch on various aspects of the hardware production process. These aspects include, initially providing an introduction to the project, budget/schedule process, software and hardware implementation, the revised proposal, introduction to CPU platform/components, schematic design, breadboard testing results, PCB design/testing, the final enclosure design and finally the troubleshooting procedures/conclusion. A table of contents has been provided for your convenience, please use if required.

**Table of Contents**

Abstract……………………………………………………………………………... iii

Introduction to Project…………………………………………………………………………….1

CPU Platform/Components……………………………………………………………………….1

Budget……………………………………………………………………………………………..

Schedule…………………………………………………………………………………………...2

Software + Hardware Implementation………………………………………………….................3

Proposal……………………………………………………………………………………………6

Schematic Design………………………………………………………………………………… 8

Breadboard Testing Results………………………………………………………..…...................9

PCB Design/Testing……………………………………………………………………………..11

Testing Results/Screenshots ……………………………………………………..........................10

Enclosure Design………………………………………………………………………………...13

Troubleshooting Procedures……………………………………………………………………..12

Conclusion……………………………………………………………………………………….14

**Introduction to Project:**

1

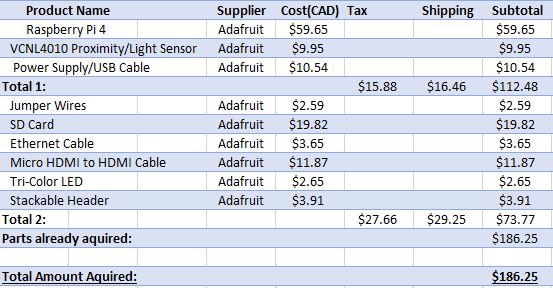
Proximity sensors help provide the outlet to detect the presence of an object without any physical contact, whether that is for measuring vehicle distance, making use of mobile devices, GPS coordinate systems/Uber related technology, or parking-assist functions. The VCNL4010 Proximity/Ambient Light sensor is a hardware device that is built on the idea of detecting movement from short distances of a range of up to 200mm, working best at a range of 1-15mm for exact, and the most efficient results. It is an efficient source of hardware that is capable of detecting objects from short/far distances and provide proximity readings to be able to analyze and monitor movement of a location. Some of these applications, may include detecting when a car enters/exits a parking lot, status of a given location, or hand/object movement. It is one of the most accurate real-time measurement devices utilized for reading proximity data. The hardware comes with a i2C interface equipped with a built-in IR LED light sensor to measure luminance values, attached to a 3.3V ultra low dropout regulator. For the ideal operating voltage range, the recommended use is between 3.3V to 5V. Making use of the full 5V of power, will provide the most use and obtains the best measurements from the device. In reference to the final project WatechPark, the main goal was to handle ticketing with a simpler, digital method without having the need for manual interference, and to help manage capacity levels. For this hardware project I will be using 5V from the GPIO pins of my microcontroller platform which will be explained in detail further on in this document.

**CPU Platform/Components**

The CPU Broadcom development platform chosen is the Raspberry Pi 4 Model B. The RPI 4 is the fastest, latest piece of technology that is capable to providing multi-support interfaces, at much higher speeds. For large data processing, and data heavy projects such as this hardware project it is the suggested microcontroller to do the job. Moving on to the hardware components needed to complete the hardware side, I made use of the following: 3 (220) ohm variable resistors to limit the current flow through the circuit, Common Anode RGB(red,green,blue) LED to detect and react to changing states in proximity readings/parking lot status, 20x2 stackable header, to mount the PCB board to the RPI, 6 pin stackable header, to mount the sensor to the soldered PCB board, and finally a set of jumper wires(female to male) for the breadboard testing portion.

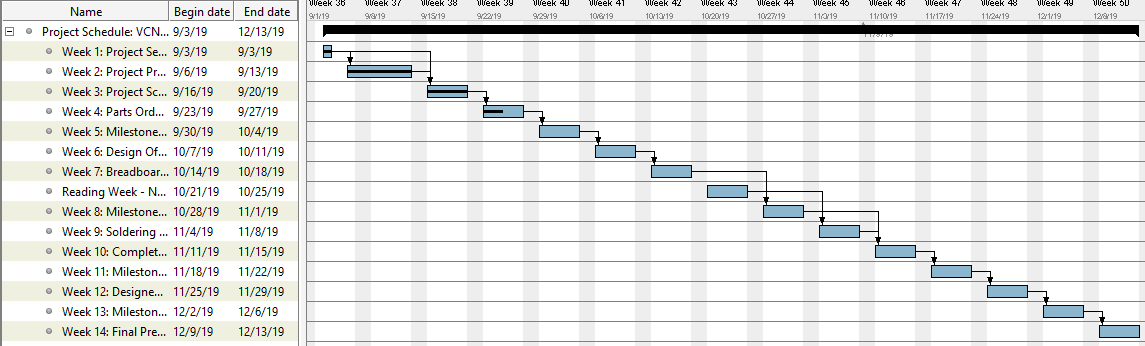
**Budget**

2

****

The total project budget did not exceed (CAD)$186.25 for the duration of the development stage throughout the semester. The budget stayed intact throughout completing the project for the most part. The parts I made use of included the presented ones above. I made my best effort to make sure the project is inexpensive in completing, and kept the budget under $200. Not included here are the parts that were already acquired. This included, the Parts Kit from semester 1(not included in the total accumulation).

**Schedule**

****

In terms of scheduling, for the most part I stayed on track and followed the planned set of objectives. This included, getting the proposal done, the parts selected and ordered in time for designing/starting the prototyping process. As well, as completing the milestone which I was successfully able to complete, whether that be for the breadboard testing, PCB design, soldering, making the full connection or the final enclosure. I took advantage of the reading week to familiarize my self with the correct methods to solder, and also took time out to make sure I was prepared for the upcoming milestone following the reading week for the PCB design. In total, the project required 12 weeks of development from the planned 14. I was able to complete all objectives in due time, a week ahead and perform/test the project, with having the final presentation/demo in week 13.

3

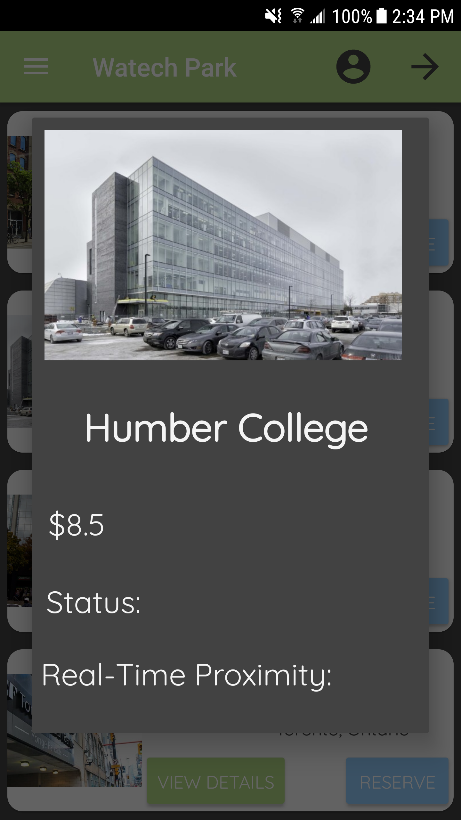
**Software + Hardware Implementation:**

To implement the hardware with the software APP, I decided to use python programming as the basis of testing if the data can be displayed on the APP’s UI. The VCNL4010 proximity sensor interacts with the app to relay back the parking lot data, displaying the readings on the interface. The software app takes the raw readings from the sensors and display those proximity values for each parking location under the main menu of the APP. Based on these values, it informs users about the status of each lot, such as which lot is available to use/which ones to avoid.

Going through the actual process (software APP wise) first the user would select a parking location from the list of parking lots available to display on the Home menu. From there, by clicking on a lot or the “View Details” button real-time data is read in proximity. This includes, the overall state/status of the lot at the time of accessing the data. The data would be changing each time depending on the actual data of the lot at the time of access.

The way this works is, the sensor detects the status of parking space, and sends the raw proximity data fetched from the sensor to the server (Firebase real-time database). The server updates the proximity level on the APP depending on the changing high/low values of the proximity. The status is updated in real-time depending on various situations. Such as, if the parking space is available, if the car is approaching, or the car is parked (space is full). The logic behind this, is the closer the object/vehicle is to the sensor the higher the proximity level, which means the lower chance of an opening. The other way is true, if the object/vehicle is farther away from the sensor then the proximity readings would be around a lower range. So, this means there is a high chance of an open spot.

**Below is a screenshot portraying how this works:**

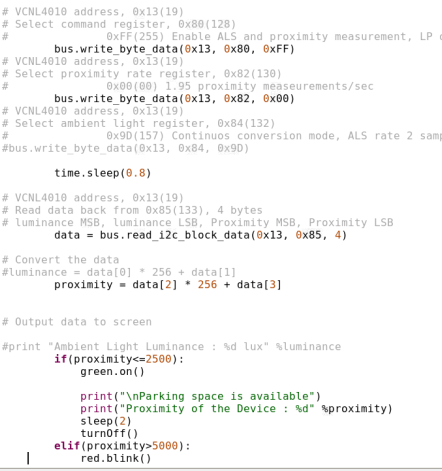
****

4

The code below shows the strategy that was used to test the hardware + software connection including sending the proximity data to the database and then reading from the data structure “ProximityData” and displaying the data on the app.



The first step required to initialize the firebase configuration, with all of the project details.



5

Here, I took the proximity value from the sensor and basically used a while True loop that would print the data while the condition is true. So, if the proximity is less then or equal to a range of 2500, then the green light of the LED will turn on and print a message to say “Parking space is available”, then print the current proximity value, and turn off the led after 2 seconds.



Then, using an if/else structure I checked if the proximity value is greater than 5000 then print out the lot is full, and corresponding data. If it is in the range of 3000 or greater than 4500, display a message saying the “Car is approaching the parking space”. Towards the end of the program, after pushing the data to Firebase I used GPIO.cleanup() to reset the GPIO pins and basically turn off the LED.

6

In the APP, how this would work is the data that is sent including the status of the lot would be displayed on the main menu once more details are viewed of the lot. So, while the proximity value changes each time the user chooses a new parking location, the status would also change. On the APP, it would be indicated by a “Status: Open/Closed/Parked” state.

**Proposal**

**CENG 317: IoT Hardware Project Proposal Vikas Sharma**

**N01160135**

**Name of Project:** Park Smart

**Team Members:** Vikas Sharma, George Alexandris

**Name of collaborator:** Mike Wrona

**Contact(email):** [mike@parkingboxx.com](mailto:mike@parkingboxx.com)

**Summary/Plan:**

This proposal is intended to outline the hardware side of the capstone project, which will be communicated with the software APP in CENG 319. The general purpose being to develop an IoT (Internet of Things) type capstone project. We will be working with the parking lot system model. I have proposed the hardware use of the VCNL4010 Proximity/Light sensor for this hardware project side. Our intention is to carry this group over to the CENG 355 Computer Systems Project course to complete the final cloud server integration.

**Background:**

In the parking industry today, there have been many occurrences where occupants feel the need to manually have to search, locate or reserve a spot in congested cities. This solution can focus on solving the problem of unnecessary travel, manage capacity levels, and provide more efficient ways to getting to a location.

In terms of the courses relating to this project, the following would be ideal candidates to consider:

* **CENG 215 Digital and Interface Systems**: working with sensors/actuators, designing of circuits
* **CENG 212 Programming Techniques in Java:** Java programming skills, methods
* **CENG 216 Intro to Software Engineering**: Software deployment, design, building, management/development process with use of Gantt charts
* **CENG 252 Embedded Systems:** Sensor control, use with microcontrollers (raspberry pi), reading real-time data
* **CENG 254 Database with Java:** Connecting to a database/networks, using SQL language

7

* **CENG 256 Internet Scripting:** Web design, server access to send data through web application

**Hardware Implementation/Time Estimation:**

**Checkmark 1: Hardware Design/Build**

**This phase would be completed in the Fall 2019 term by week 10.**

* Design the circuit, create a prototype that would work on a breadboard, build the circuit
* Use variable resistors to design the circuit, using the sensor datasheet to help
* Test the functions of the sensor, soldering process with PCB board
* I2C interface to connect the hardware built with sensors to the PI
* The power supply used would be ideally 1.7 – 5.0V for maximum power dissipation

**Checkmark 2: Hardware connection/coding**

**This phase would be completed in the Fall 2019 term by week 11.**

* Connection to PCB board made, soldered components
* Connection to the Raspberry Pi device made with the PCB Board
* Python programming to talk to the sensor and the software APP.

**Checkmark 3: Individual presentations/Project demo**

**This phase would be completed in the Fall 2019 term by week 14.**

* showcase 2 semesters of work to a group of potential employers, and demonstrate my knowledge of our IoT project

14 weeks will be needed to complete the hardware side with functioning parts/code.

**\*12 weeks was needed to complete the project at the end**

**\*Week 13 – Final Presentation/Demo**

**Database setup/connection:** Firebase Cloud Datastore.

If there are any problems along the way I will act accordingly and aim to solve the problem quickly. I will follow the agile methodology, to have patience and keep working diligently to get the end goal. To get back on track, I would have to review my design and re-assess which parts are causing the issue and work around from there.

Similar products out there include companies such as Indigo, and SpotHero. These platforms also use sensors to detect the parking space status, and updates the change to the database/ mobile app. What we would do differently is use:

* VCNL4010 proximity/light sensor to detect status of an area. (empty/full parking spaces and use a 3 color LED to indicate the 3 scenarios. (red – space is full, yellow – space is occupied, green – space is available)

8

* IR Beam sensor to detect the distance of an object from the intended target. (gate)
* Gate controlled by 2 stepper motors to control opening/closing lanes.
* Camera sensor used to take a snapshot of the license plate and send the data to the database if valid/if not the gate would not allow entrance.

**Conclusion:**

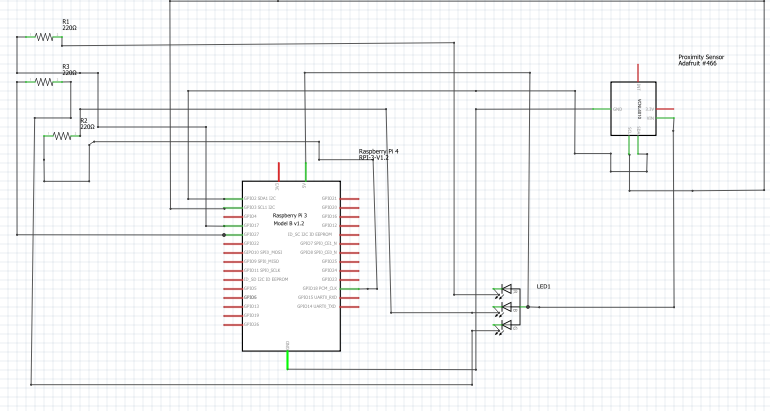
This IoT project focuses on reducing industry wide parking issues in terms of capacity, and congested outlooks and offers efficient ways to retrieve information. This capstone project would allow me to demonstrate the concepts I have learnt, and practice working as part of a team collaborating with real industry partners in the marketing process. I request the approval for this hardware project to begin**.**

**References:**

Park Indigo Canada Inc. (2019). *Indigo.* Retrieved from <https://ca.parkindigo.com/en>

Spot Hero. (2019). *Spot Hero*. Retrieved from <https://spothero.com/>

**Schematic Design**

****

This is the schematic design I created for the circuit using the Fritzing software. The design above shows the basic pin-outs used for the sensor. There are 6 pin connections and 4 of the 6 will be used (VIN, GND, SCL SDA).

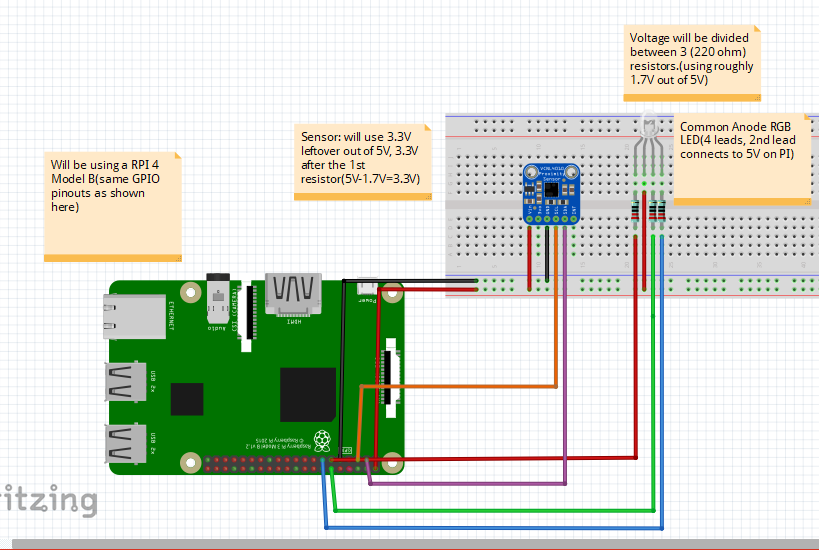
* VIN of the sensor connects to the 5V GPIO pin on PI
* GND of sensor connects to GND on PI

9

* SCL line of sensor to SCL of PI
* SDA line of sensor to SDA of PI

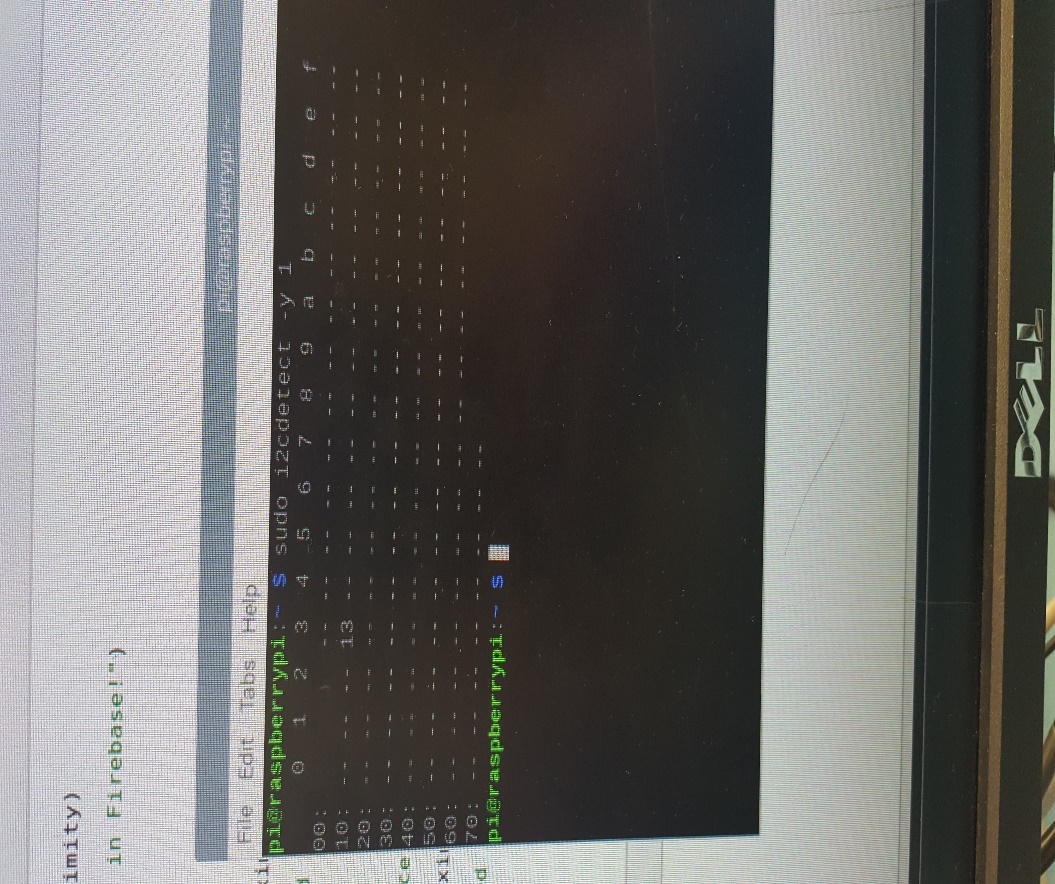
The 2nd lead of the RGB LED(Common Anode pin) connects to 5V GPIO pin on PI. R1(resistor 1) connects to GPIO pin 17 of PI, R2 to GPIO pin 18 of PI, R3 to GPIO pin 27 of PI.

**Breadboard Testing Results**



The breadboard testing portion required to test the circuitry if it responds to sensor and if the RPI can detect the sensor and its i2C interface. Using the Fritzing software, here I basically transferred the schematic design to the breadboard. I followed the breadboard design on Fritzing for the circuit, with use of 11 jumper wires. After analyzing the professor’s advice, and knowledge I decided to use the 5V GPIO pin, designing a voltage divider circuit in the process. This is to ensure there is no possible damage to the circuit, with the amount of current being supplied to the circuit. Based on the breadboard design the sensor used 3.3V out of the 5V which was leftover voltage after the 1st resistor (5V – 1.7V = 3.3V). The voltage was divided between the 3 220-ohm resistors, using roughly 1.7V out of the 5V.

To test the breadboard circuit, I first had to check if the sensor was working as needed with the design. The sensor uses a i2c interface, so I used the followed command to detect if it works:

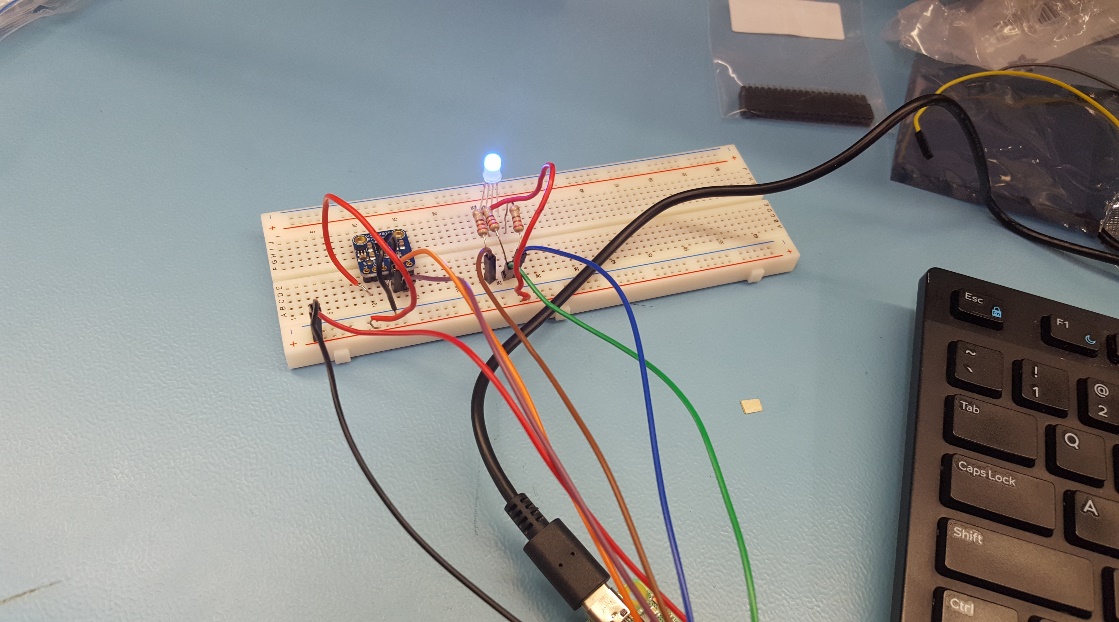


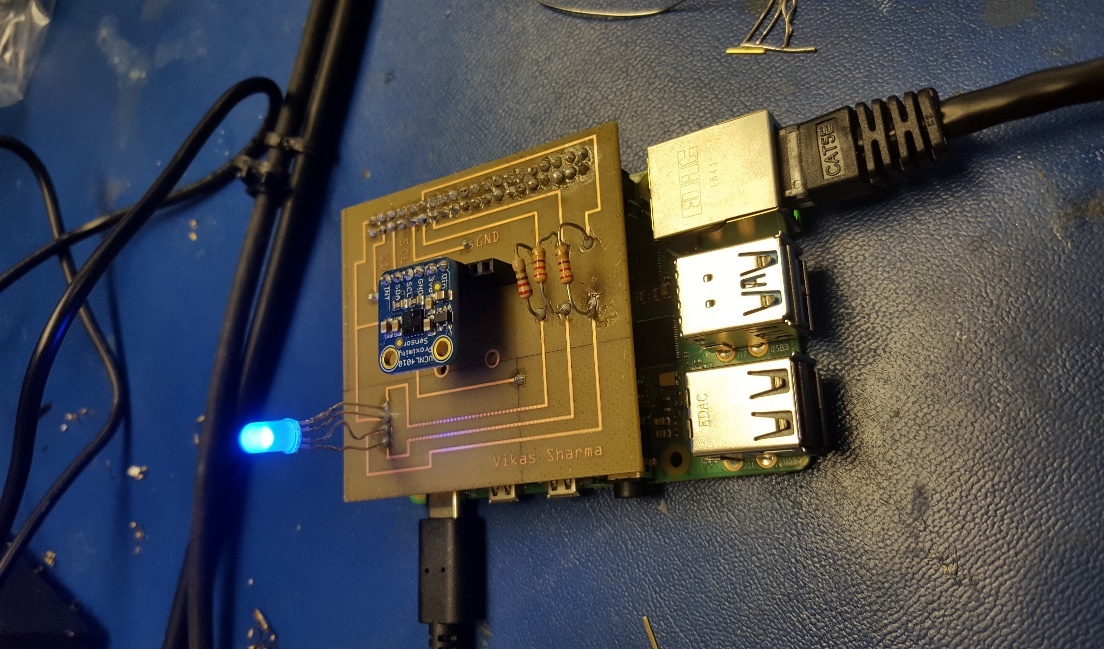
10

**Command: sudo i2cdetect -y 1**

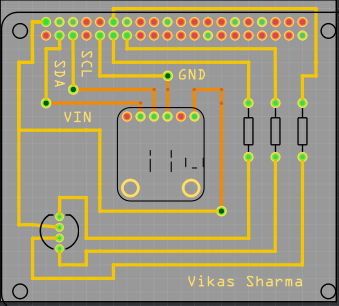
Using this command, you can see the sensor is reading the bytes of data from the i2c interface on address 0x13, showing successful connection of the sensor to the breadboard design.

Next, I tested if the LED functioned as designed. After setting up the circuit, and running the provided sample code for the LED.py program from the professor I was able to get the LED working. I played around with the sample code testing different colors to see if I can get all three values of red, green, blue to blink/flash which I was able to successfully.

****

**PCB Design/Testing**

11



The next step was to design the PCB board. During this process, I ran into many problems trying to get the right design down, as the board required many changes. The main point being the positioning of the top/bottom layers. After much consideration, and further advice from the professor I was able to successfully complete the PCB design. This is because, all the connections that go to the GPIO pins had to be in the top layer, and there must be no overlapping connections between the top/bottom layers. So, after redesigning the board, I had to rearrange the top/bottom layers and from there I was able to complete the PCB design as needed. After getting the PCB design finalized, I sent it into the prototype lab to be produced.

To solder the PCB board, you would require a lot of focus, and attention, because any accidental procedures/connections going wrong can cause you to redesign the whole board. I was able to luckily not have to go through that process, while soldering. For my design, there were 4 vias that needed to be soldered. You must thread a single thin wire through the holes, solder it and then cut off the remaining excess wires. The same process, was taken for the resistors where I set up the resistors in their place and had to solder the 2 sides. Then, cut off the excess wire. While soldering the LED, I had to be very cautious as the connections were designed really close to each other, so a lot of focus was needed here. Initially, there is a 6-pin header that comes with the sensor that needed to be soldered. This is because, you cannot connect the sensor directly to the PCB board as later on for the next semester it will become a difficult task to take apart the sensor when we would have to redesign the board with all 3 sensors. I also soldered the 40 pin GPIO pins on the PI, to keep the header stable while soldering. After soldering, the board should be ready to be tested, and should look like the screenshot shown on the right above.

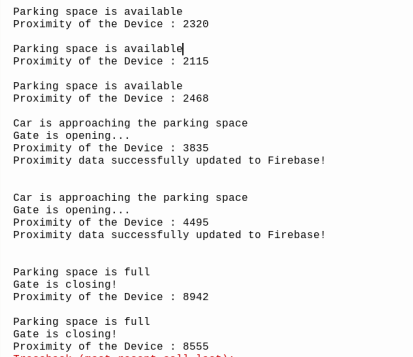
Testing the board, I first checked the sensor connection, by issuing the command:

**sudo i2cdetect -y 1**

12

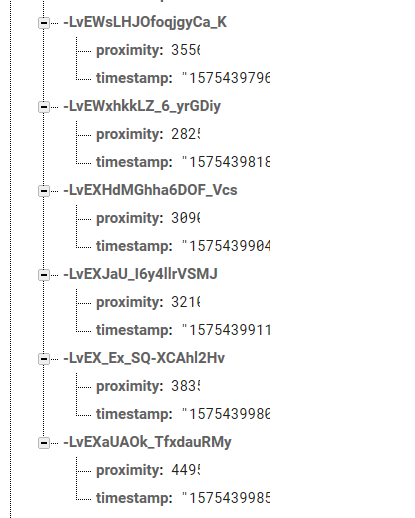
I was successfully able to identify my sensor, so from there now I had to integrate the final code in relation to my project. I used part of the sample code I had previously as well as added in the needed requirements to match the parking lot system design. I tested to see if the sensor works with the RGB LED. Using if/else statements to see if the LED can communicate with the PCB board and the sensor, which I was successfully able to do.

**Testing Results/Screenshots**



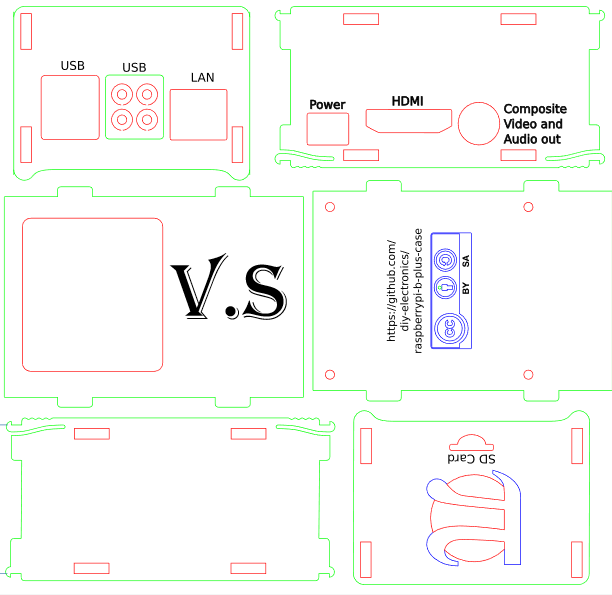
These were the results of testing the final product (with the PCB board attached to the PI). Based on the different proximity levels of the sensor, and how close an object is to the sensor the values would either change from high/low and show the corresponding message. Such as, if the parking space is available the value is less than 2500, if the car is approaching the space, then this means the gate is opening so the proximity value must be in the range between 3000 and greater than 4500 as set in the code. Then, the data was successfully able to be sent to the Firebase real-time database under the “ProximityData” data structure showing the proximity, and the timestamp. Also, if the parking space is full, the proximity value has exceeded the limit set of 5000, and the gate is closing.

The data structure below shows where the data was sent to on Firebase, reading the raw proximity value from the sensor. It would be ideal to try to implement/test the database connection with your sensor now rather than later and see that the sensor is not working, especially for the start of next semester.



13

**Enclosure Design**



Once the PCB board testing phase was successfully complete and I was retrieving/sending data to Firebase and the app, I moved on to the Enclosure design . For this, I decided to use the CorelDraw software as it seemed like the best choice to design the case the way you want. I went with a acrylic case design, as the case is to prevent/protect your raspberry pi/sensor from potentially being damaged. While making the case , you have to be aware of the dimensions of the raspberry pi(height,width from all sides). For the design, I had to make some changes due to using the RPI 4, some ports are arranged differently. Such as the LAN(ethernet port), and the 2 micro HDMI ports. For this, I decided to make a wider port/hole to fit both mini HDMI ports into one. I ran into a issue where the hole for the power port was cut too short, for this I had to shave off some access acrylic to get the hole to be wider to fit the power cable into the port. I was able to fit it perfectly in the outcome. For the design, I kept the top of the case open with a medium-sized hole, as there needed to be a object in the way of the sensor or far away for it to display the readings.

14

**Troubleshooting Procedures**

In the initial stages of the hardware testing, I ran into some issues that I was able to solve by debugging the software that is connected to the hardware sensor. Such as, initially I could not receive the sensor connection by issuing the i2c command. The reason being I hadn’t soldered the header pins to the sensor to get the physical connection. To troubleshoot, I fixed this by soldering and testing if it had worked and after I was able to get the sensor connected to the i2c interface. Another minor mistake, I made while testing the database portion was that for a little time, I was not getting the data from the sensor sent to Firebase. Along the help of the professor I was able to figure this out, because I was missing a function call in the main part of my python program, which was a silly mistake. Other troubleshooting methods, included making sure my sensor can send changing data to Firebase. For a while, I was getting the proximity value and pushed it into the data structure but was unable to keep the program running with changing values. To fix this, I used a while loop to run the conditions while the state is true and I was able to push the current value of the proximity to Firebase, instead of the same one throughout. I learned to stay patient, and step through the code that was not working and test the values of a variable that I was receiving an unexpected result on. I did not encounter many problems overall just minor ones that were resolved very quickly and manageable.

**Conclusion**

Overall, the project provided me to learn more about the sensor I was working with, and how the design process works for testing, PCB board design, working with the Fritzing software. As well as designing, prototyping, testing in each stage to ensure the hardware works as expected. I was able to grab the knowledge from other courses and use this here wisely to build working hardware, connect the sensor data to Firebase and read/write the data back to the software end. Next steps are to integrate all 3 sensors from each member, that being the IR beam sensor – handled by George Alexandris to detect gate opening/closing, 2 stepper motors/LSR camera sensor for controlling the gate operations, and checking for valid license plates in the next semester for the final Capstone Project integration. This includes, being able to talk between the hardware/software and connect the data to a cloud server.

* **GitHub was not used for the hardware project side.**