Smart Campus

- Student Name:

  Cameron Cross

- Advisor Name:

  Dr. Lin

- Expected Graduation Date:

  May 2021

1. Statement of Purpose:

With the world facing the Covid-19 pandemic, the main problem the campuses are facing currently is, knowing how many students are in an area and if that area is over occupied based on the University Covid-19 guidelines. With the University being able to see if too many people are in a certain area, they can break up those groups and help fight the spread of Covid-19. Students also face the issue of not knowing when and where to eat, study, or even simply relax due to the limited number of seating and the strict regulations on occupation sizes. This would allow them to view the occupancy in buildings and allow them to determine when the best time to eat, study, or relax would be. Campuses are not the only population effected by Covid-19 that could benefit from Smart Campus. Other businesses could use Smart Campus to determine when their busiest hours are so they can edit their schedules or the number of products they order based on how many customers

they are expecting. They may also want to monitor how busy they are to project the number of Covid-19 cases they may see.

The solution I am proposing is a monitoring software that keeps track of how many people are entering and leaving through doors on various buildings around campus. Even after Covid-19 is no longer a virus that needs to be monitored the University can still use the system to see when students are going places, allowing them to allocate resources based on these trends. Students can also benefit from the system. They will be able to see when a good time to go to lunch is or determine which spot on campus would be the emptiest, quiet place to study is. In light of Covid-19, the proposed problem and solution that is being introduced will help both students and the University monitor students' activities and determine hotspots on campus.

I believe this will help me get an idea of what designing, and programming real-world application looks like and allow me to further my knowledge of IoT devices, Python, and simple web design with database connection. I expect to see satisfaction from students and staff members that are surveyed in this project. I am also expecting some issues that I will have to solve in order to get the project working.

2. Research & Background:

In this section I am going to present my findings from my research. I found that there is a plethora of libraries in many different languages that are built to handle sensors and database connections. Along with the different languages I found that there are many IoT devices and different types of sensors, with the two most popular being IR (Inferred) and ultra-sonic to choose from.

There are plenty of sensors to choose from. I decided to research two different types, IR and ultrasonic. IR sensors use a small bulb and sensor to send and detect inferred light. I determined that IR sensors would not be the best choice to detect when someone is passing by due to their limited range and need for a mirror to reflect the light. So, I decided to use ultra-sonic sensors because they have better range and no need to use mirror since the signal will bounce off of any object. Below is the link to a website which explains the library used in python to connect to the sensors and the math needed to find distance in centimeters.

https://tutorials-raspberrypi.com/raspberry-pi-ultrasonic-sensor-hc-sr04/

Based on the link above I found that the speed the ultrasonic signal travels at is 34300cm/sec. It also explained that to find the distance the signal traveled I needed to multiply the speed by the time to receive a signal back and divide the whole thing by 2. The equation looks something like this, (time * 34300) / 2.

In order to set up Smart Campus I needed some sort of IoT Devices. I found that Arduino, and Raspberry Pi were some of the most popular IoT devices for people new to these types of devices. I was unsure with the multiple different IoT devices on the market so, I went to Dr. Lin for suggestions. Based on Dr. Lin's recommendation of IoT device, I did research on Raspberry Pi and the OS used on them by default which is Raspbian. I found that Raspbian is a streamlined OS that has a built-in terminal with Linux commands making it an easy transition into development for a student like me that has used Ubuntu and other Linux based OS. Dr. Lin also explained that Raspberry Pi makes multiple

devices, and he informed me that the Raspberry Pi 4 would be the best choice as it has 4GB of RAM and a better CPU compared to other IoT devices on the market.

For the database and web development side of the project, while I have skills I learned in Dr. Lin's Applied Networking covered the web design, and the database side from Dr. Sessions' Database Management class, I had to confirm a few things like: which database servers I wanted to use. MariaDB and MySQL are two of the leading database servers that I considered. MariaDB offers Perl-compatible regular expressions while MySQL has its own standard for regular expressions which are not as precise and powerful as the ones MariaDB supports. After looking at the functionality and features of both servers I chose MariaDB because it is reliable, great compatibility, free, and easy to manage all while being very secure with correct notation.

While looking at the many languages available that would work for this project, I chose Python as the main language for handling the ultrasonic sensors because, Python is one of the most versatile languages that also offered support for SQL. PHP the language I first thought of when thinking, which language would be good for pulling information from a database while allowing me to create webpages to display the data. This is because PHP works well with the most popular web design language which is HTML. HTML is a simple yet effective language when paired with CSS which would be used for adding extra cosmetic enhancements to the web design aspect of Smart Campus.

In this project I did thorough research on new topics and topics that were covered in the classes I have taken. I learned a lot about languages I have used and why I should be

using them as well as the software I needed to complete this project. In the next section I will be discussing the requirements for Smart Campus.

3. Project Requirements:

This is a list of the programming languages needed for this project: Python, SQL, PHP, and HTML with CSS. The software needed is: SQL server called MariaDB and a webserver called Apache. The hardware I needed is: IoT devices (Raspberry Pi 4), wires, breadboards, resistors, and ultrasonic sensors.

I used the languages for multiple aspects of the project, for the web-based development I used PHP with embedded HTML on the webpages and CSS to style the webpages. I used PHP as the main web development language because it is used to communicate with the database, and I used HTML with CSS because they make for a better web interface while working well with PHP. For the backend of the code communicating with the sensors and sending information to the database I used Python because it has built-in libraries for sensors and database connection. Python was used on the scripts to add and subtract from the population in the database. I chose to use SQL for the database because it is the one that I am most familiar with and if I did run into issues, I could not fix there is plenty of support online.

The particular software used in this project are two servers that are easy to setup and reliable. MariaDB is a useful database server that has a minimal installation process which is supported on Linux based distributions like Raspbian. I chose Apache as the web server because it is the software that is used in Applied Networking and it has an easy setup as well and a stable connection to the host.

As for the hardware I used Raspberry Pi 4s because of Dr. Lin's recommendation and the upgraded RAM with higher specs across the board compared to older models. I used breadboards so I would not have to solder any wires or resistors to the IoT devices. With the breadboard I needed an array of wires ranging from "male to male", "male to female", and "male to female" connectors. The resistors I used were the 330 OHMS and 470 OHMS. I had to use these particular resistors because of the voltage rating on the HC-SR04 ultrasonic sensor. If I were to use different resistors, say one that allowed higher voltage it may fry the sensor since it is only able to handle low voltage. I used the HC-SR04 sensor because of the price and efficiency they are about $10 for 10 of them that are rated for a range up to 4 meters or 13 feet.

Here I have discussed the different components that needed for Smart Campus and why they are important. Using these components would ensure that the project could be repeated. Next, I will be going over the steps I took to implement the languages, hardware, and software all together to make Smart Campus possible.

4. Project Implementation Description & Explanation:

This section is to discuss the methods and code needed to implement this project. I will include screenshots, snippets of code, and any other diagrams that may be useful to recreate this project. Each of these will be labeled figures that I will describe and explain why they are needed for implementation.

The implementation of the project requires the wires to be in a certain configuration on the breadboards and to the Raspberry Pi as seen in Fig. 1. The diagram shows VCC (red wire) to Pin 2 (VCC), GND (black wire) to Pin 6 (GND), TRIG (green wire) to Pin 12

(GPIO18), the 330 OHM resistor to ECHO (purple wire), ECHO to Pin 18 (GPIO24), and through the 470 OHM resistor to Pin6 (GND). This is done because the GPIO pins can only tolerate a maximum of 3.3V.

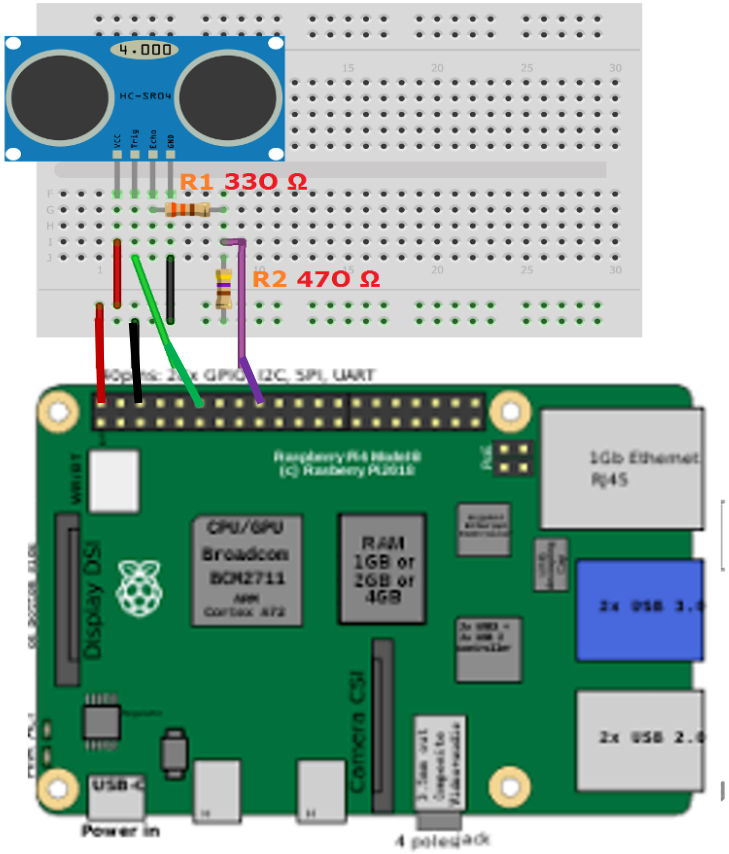


Fig. 1 Wiring Configuration

The home page is shown in Fig. 2 which shows the map of CSU which has clickable links to all of the buildings shown. This was made using an outline of the buildings from google maps. This was done using adobe photoshop to trace the outline.

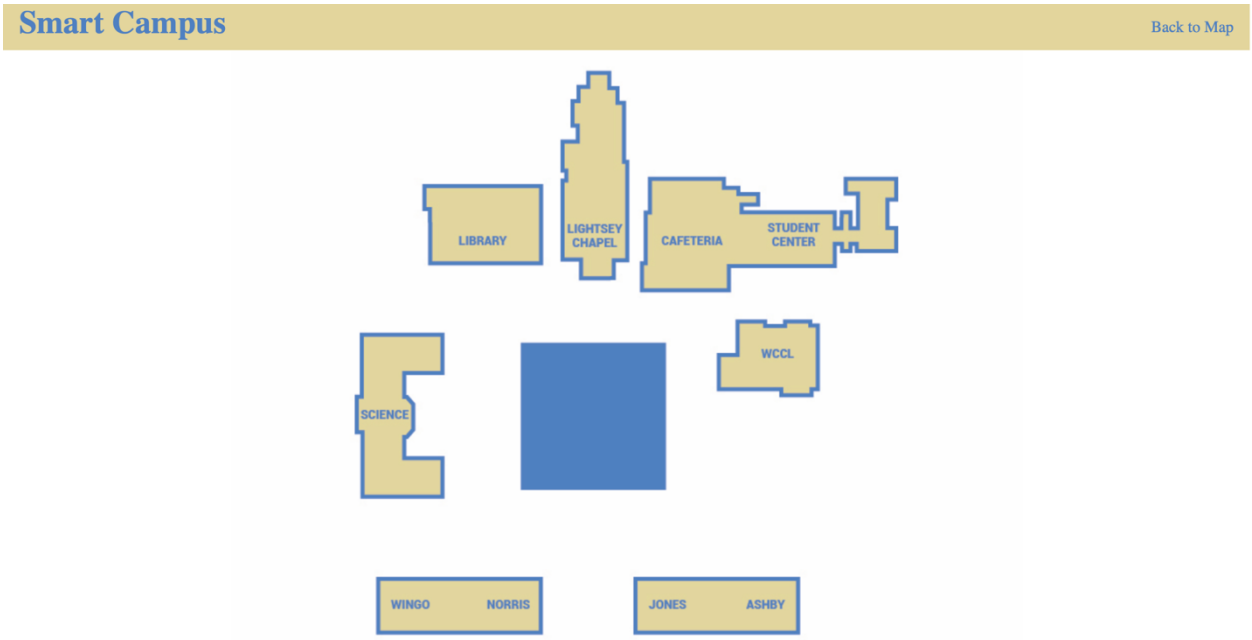


Fig. 2 Map of CSU Main Buildings

The links to these buildings are made though HTML image maps which are shown in Fig. 3 which takes the user to a page specific to the building with information about the population.

```
<img class="map" src="CSUMAP.jpg" alt= "Map" usemap="#map" hidefocus="true" />

<map name = "map">
    <area shape="poly" coords="411, 241, 411, 214, 415, 214, 415, 163, 419, 163, 419,
130, 494, 130, 494, 138, 498, 241" alt="Cafeteria" href="cafeteria.php">
```

Fig. 3 Snippet of code for link to Cafeteria

Fig. 3 is just a sample of the linking code to the buildings on campus, I only included the code for the cafeteria as an example. The links are made by tracing the buildings using coordinates which are shown. Since the buildings are not a specific shape I had to use “poly” which allows me to input multiple points to make for an accurate linking area. All of the buildings’ pages are setup the with the same design which can be seen in Fig. 4.

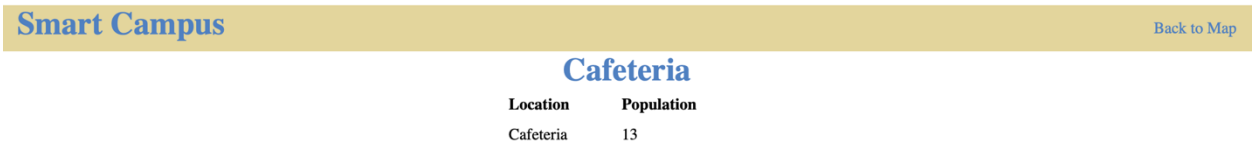


Fig. 4 Building page design

These pages use PHP with embedded HTML to get information from the database. The code for this can is done by making a SQL query that asks the database for the location and population from a specific area on campus as seen in Fig. 5.

```
$sql = "SELECT Location, Population FROM Campus WHERE Location = 'Cafeteria'";
$result = $conn->query($sql);
```

Fig. 5 SQL query for the population data

The query command is stored in "$sql" and then the results of the query are stored in "$result" after the connection query is ran using the "$conn->query()" method which send the query to the database. Each query is unique to the building page and it allows the page to select the unique data for that area. The code that displays the data for these pages is the same because the unique code comes from the lines in Fig. 5. The generic code to display the data in tables can be seen in Fig. 6.

```
if ($result->num_rows > 0){
        while($row = $result->fetch_assoc()){
                echo "
                    <tr>
                        <td>" . $row["Location"] . " </td>
                        <td>" . $row["Population"] . " </td>
                    </tr>";
        }
    }else{
        echo "0 results";
    }
```

Fig. 6 Display code for query results

To display the results the data from the query is cycled through by row and then output to the webpage in table form which can be done using HTML table, tr, and td tags. The display code is also setup to display “0” if no results were returned from the query.

The python scripts to handle the ultrasonic sensors is made possible by two libraries that are well documented which are RPi.GPIO as GPIO and time. GPIO is needed for the script to communicate with the sensors and have them output a signal, while the time library is used to make the signal output in bursts and record the time it takes for them to receive the signal once it bounces off of an object this can be seen in Fig. 7.

```
while GPIO.input(GPIO_ECHO) == 0:
       StartTime = time.time()
while GPIO.input(GPIO_ECHO) == 1:
       StopTime = time.time()
TimeElapsed = StopTime - StartTime
distance = (TimeElapsed * 34300) / 2
```

Fig. 7 GIPO and time library in use

The code starts a timer which waits for the signal to be bounced back and then calculates the distance based on the time it took to receive the signal and the speed of the signal which is 34300ms. This is then divided by two to account for both the sending and the receiving of the signal. The next section of the python script in Fig. 8 shows how it connects to the SQL database and how it authenticates the device to connect.

```
mydb = SQL.connector.connect(
    host = "10.127.98.154",
    user = "cam2",
    password = "XSW@2wsx",
    database = "SmartCampus"
)
```

Fig. 8 Code connecting the database to python

This section of code uses the SQL.connector library which is useful when working with databases in python. The code requires the host device's IP address as well as a username and password that was granted privileges on the host prior to connecting. This allows the script to make changes to the Smart Campus database. The code for these changes is shown in Fig. 9.

```
if(alreadyDetected == False):
        alreadyDetected = True
        print("Motion Detected")
        sql = "UPDATE Campus SET Population = Population + 1 WHERE Location =
'Cafeteria'"
        mycursor.execute(sql)
        mydb.commit()
```

Fig. 9 Code to update the database

Before the code updates the database, it has to verify that the motion detected is not the same motion, so it checks with an if statement. After passing the check, code prints "Motion Detected" as a debugging feature which users will not see, it is only displayed in the terminal for the admin. It then loads the update statement into a sql variable and uses the SQL.connector library to execute and commit the changes to the database. The code shown in Fig. 9 is for the add.py script which adds new people to the population while the subtract script does not look much different. It is also specific to the cafeteria since that is where my main testing will be conducted.

This is the link to my Github repository with the source code:

https://github.com/clcross/SmartCampus

5. Test Plan:

These are the test plans that I plan implemented to test the functionality and efficiency of the project. I will have two test that are made to test first the functionality of the devices and sensors, the second is to test the real-world application of this project which I plan to test in the campus cafeteria.

The first test will be conducted in my dorm room where I will have people such as my roommate, suitemates, and anyone I can get that lives near my room to come in on one side of the room. They will pass the device running the python script that adds to the population checking to make sure the device only adds one person when they walk by one at a time. They will then walk by the device that is running the python script that subtracts from the population. This test will allow me to determine the accuracy of the sensors and ensure that the connection to the host device it with the SQL server is stable. They will then view the webpage and make sure the count is correct and complete the survey for the project.

The second test will be conducted in the cafeteria during lunch. I will setup the host device in the cafeteria. I will then setup the device running the python script that adds to the population at the register where students scan their IDs and other people have to pay for entry to the cafeteria. The other devices running the python script that subtracts from the population will be setup at exits around the cafeteria. These exits are in the back near the Charleston Room, and at the front near the coffee machines. This test is necessary to test the real-world application the Smart Campus on a small scale.

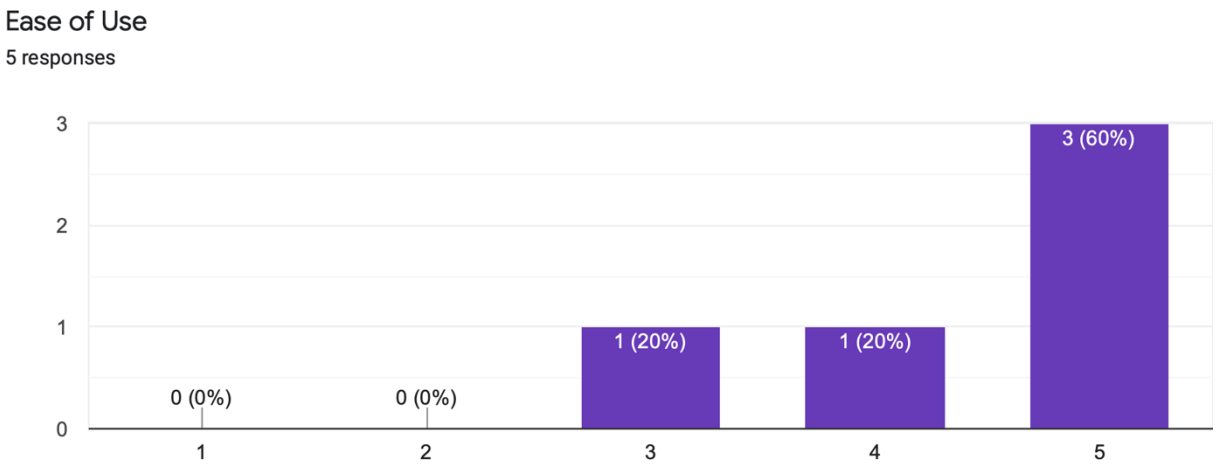
Both tests are required, the first is required to make small adjustments if needed to verify that the project works as intended and the second is required the show the

effectiveness and accuracy of the project when running on a small scale. The next section is where I will go over the results from both tests and discuss their comments.

6. Test Results:

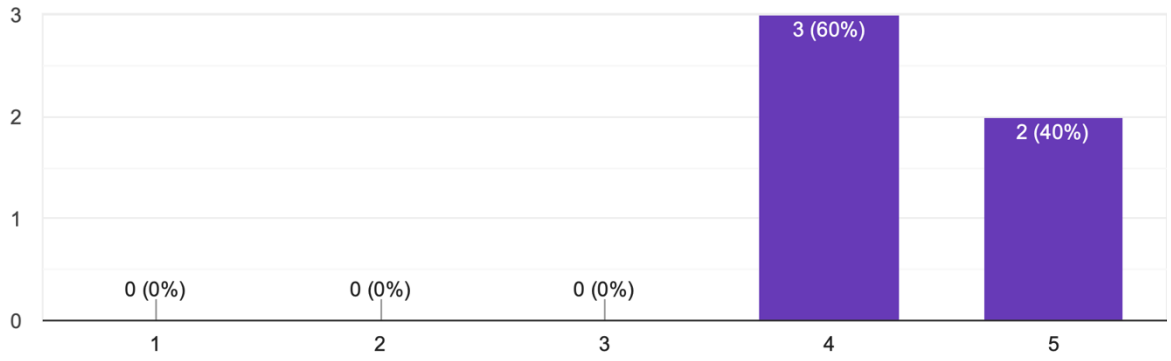
In this section I will show graphs with user scores based on different elements in the project and the users comments they added to the survey. I will explain the data from these graphs and any comments made. I hope to see an improvement of user satisfaction once I work out any issues that may be found in the first test.

Below will be all of the data and comments from the participants from the first test that was conducted in my room.



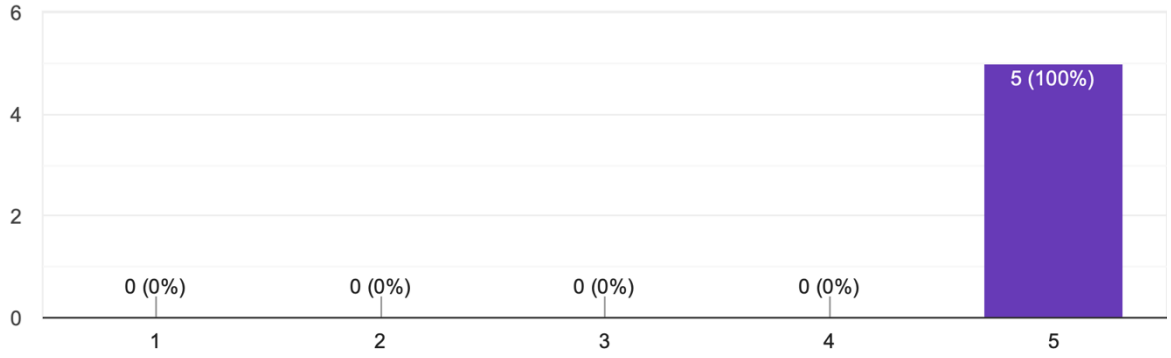
Design

5 responses



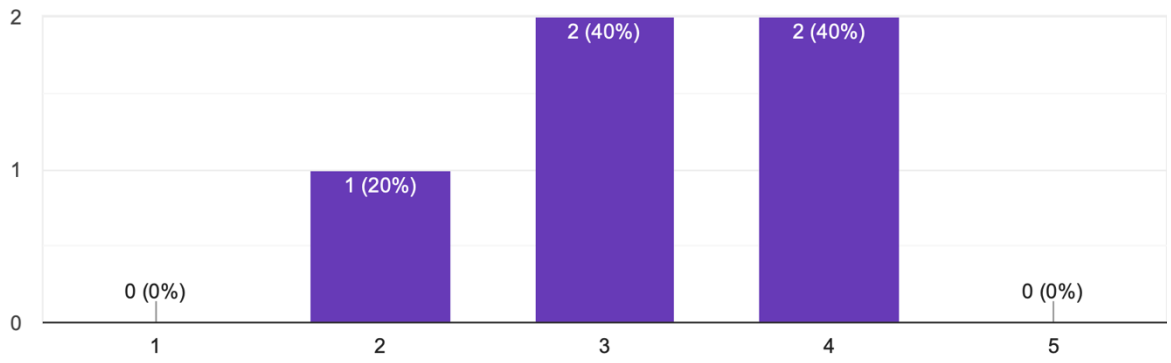
Usefulness

5 responses



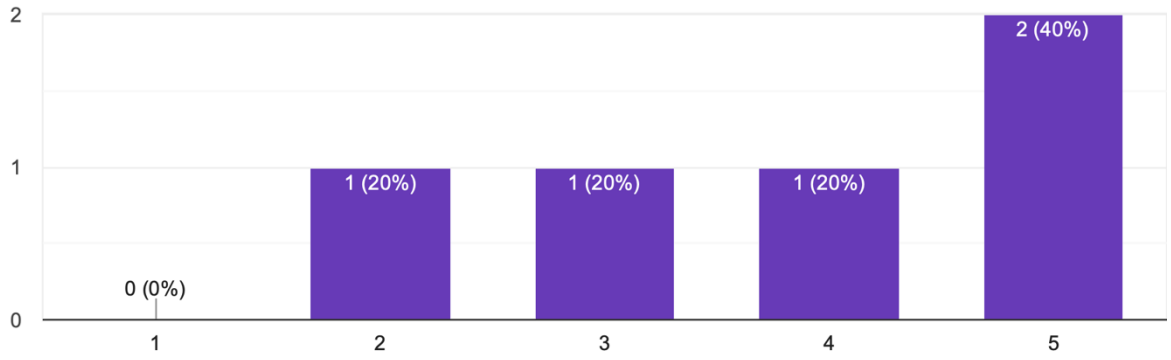
Sensor Responsiveness

5 responses



Website Accuracy

5 responses



Listed below are the comments left for the first test conducted in my dorm room.

sensor to leave won't work sometimes and the one to enter is too sensitive.

the website looks good and but the sensors seem buggy when walking fast

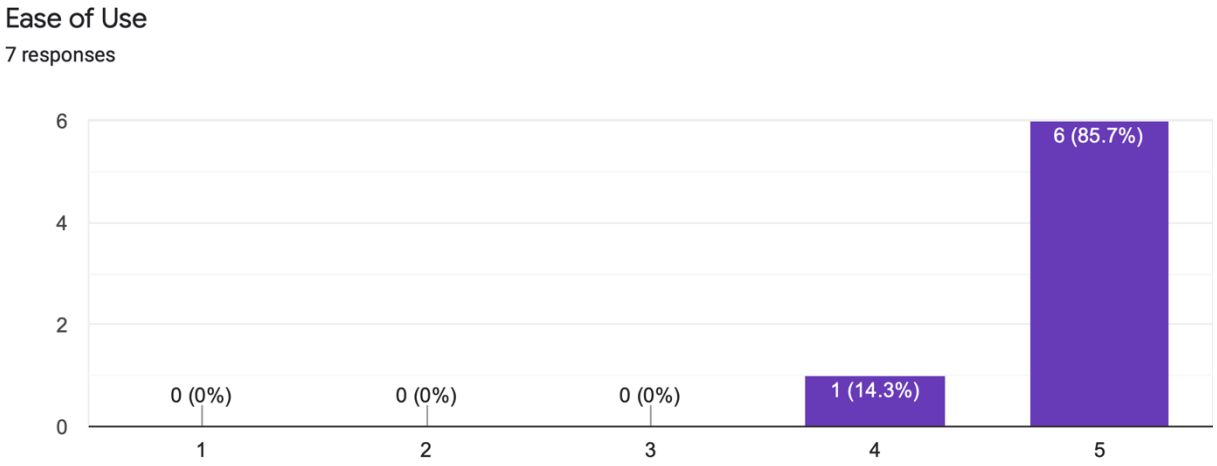
the website is hard to get to. I had to type numbers that I was told was the IP address and then the website looked very simple but its is a good idea if the sensor works right.

I'd like to see this at Charleston Southern in the future. it is cool and I think if the sensors were better the website would be more accurate.

for me it worked well but it seemed the wifi is a problem and the sensors probably need some work.

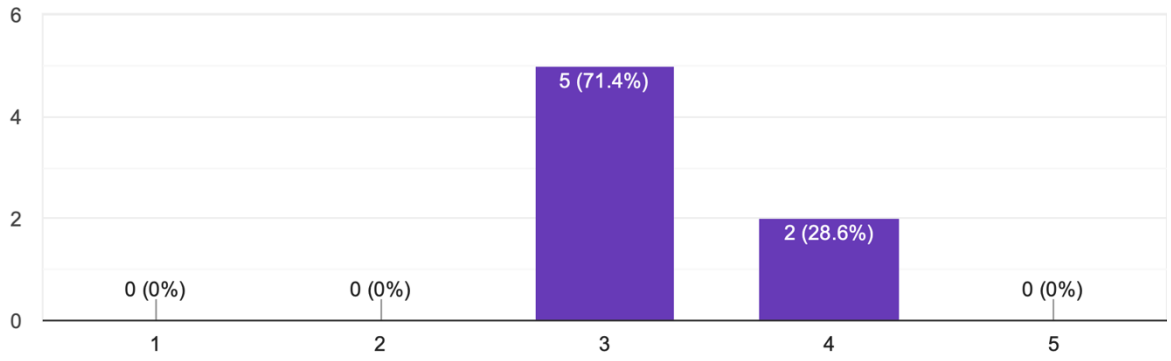
For the data above, Smart Campus was rated with an average of 4.1. These initial ratings were a little higher than was expected. The comments suggest changing the address for the webpage so the user would not have to type the IP address of the host device and edit the sensitivity of the sensors. I am not sure how to change the domain name for the project without paying for one or having CSU add a webserver for my project. As for the sensor sensitivity issue I have changed the distance required to pick up someone walking by to leave and I have reduced the number of times the sensor pulses for the devices that adds to the population.

This is the section for the second test that was conducted in the cafeteria where I selected random students and staff members to rate the project while it was set up.



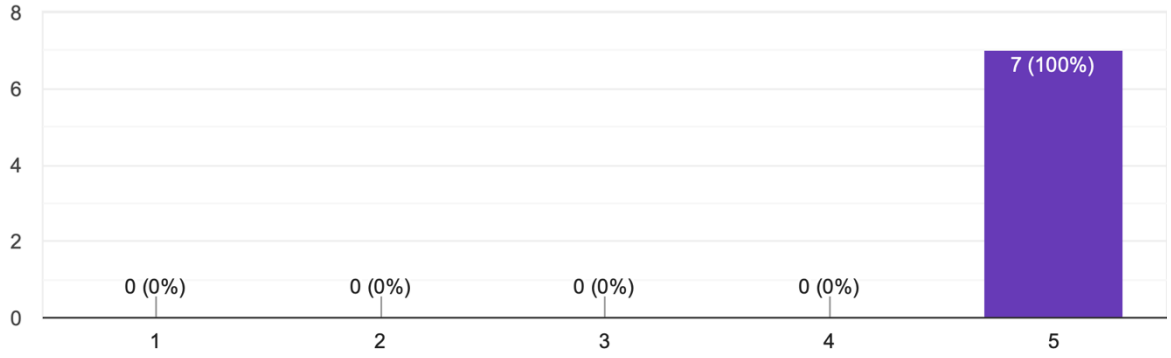
Design

7 responses



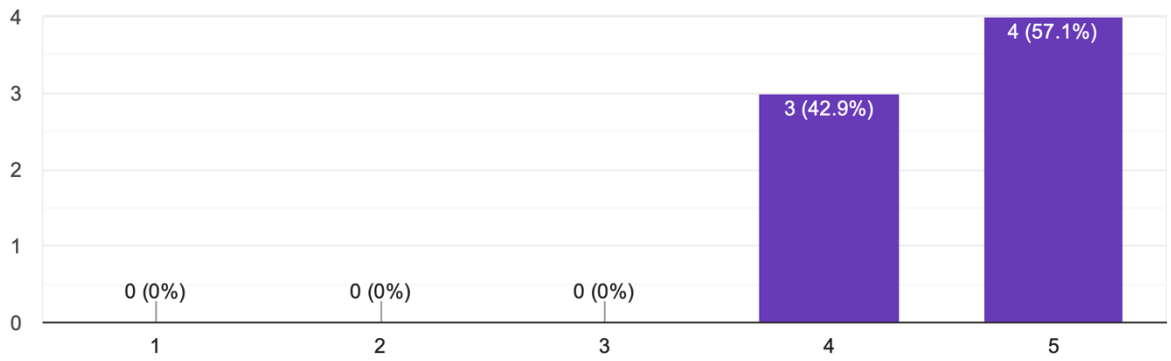
Usefulness

7 responses



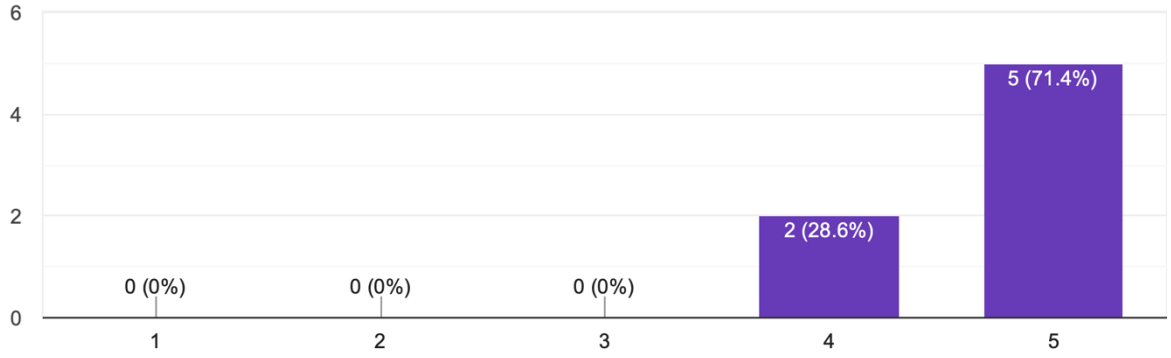
Sensor Responsiveness

7 responses



Website Accuracy

7 responses



Listed below are the comments left for the second test conducted in the cafeteria.

seems useful and works well

Very cool everything worked great

Good design and logic. Smart. Good for early development. The look could be better but understandable at this stage

This is a cool idea. Need to focus on hiding the motion sensors and the cables. Website design looks good but is very plain.

I think this is a great way to keep track of the number of people in a building.

Sensor is too sensitive and the design of the sensor is bulky.

Based on the data above, Smart Campus was rated with an average of 4.5, and the comments that were submitted it is clear that the ratings for the project did get higher. However, there are still issues that need to be worked out. A few people stated that the devices are bulky or stick out too much and the cables are an issue. These issues could be solved by making professional cable routes and laying the devices flush to a surface rather than just sticking them to an area. I still had reports that devices are too sensitive which I will discuss later.

Conclusion/bridge:

7. Challenges Overcome:

In this section I am going over some of the issues I had to fix and other problems that did not come from the project itself. I will also explain how I fixed these issues or how I would solve them if I had proper materials and better conditions.

I had to do a lot of “trial and error” work to figure out how to use the sensors with the correct distance to accuracy ratio. I had to work out a few bugs I saw while testing in my dorm room such as: the subtract script was not sensitive enough to pick up someone walking by at a fast pace, the add script was too sensitive when the users stood in front of the sensor simulating scanning their IDs. Another challenge I had was figuring out how to display the data on the webpage in a way that it updates when there are changes made to the database. I fixed this issue by using PHP to refresh the page every two seconds so it wouldn’t refresh too often but often enough that the data displayed was accurate.

The main issue I faced that I could not solve was the WIFI problem. This problem stems from the lack of WIFI signal around campus. In my dorm room the WIFI was stable enough to host the tests that I conducted for an hour, but the small-scale test in the cafeteria was a little trickier. The WIFI would not stay connected long enough for me to leave the devices running for the entire lunch rush.

Those are the challenges that I was able to overcome and some that I wasn't able to fix completely. With these issues in mind, I will discuss how they may be solved with some ideas for future enhancements that can be made.

8. Future Enhancements:

Here I will discuss the enhancements that would fix the issues from the last section and some other enhancements that would be beneficial to implement Smart Campus on a larger scale. The entire campus or public buildings around highly populated areas such as the city of Charleston.

One future enhancement that can be made to this project is if the devices had a LAN connection, an external wireless adapter with a little more range, or stronger signal from the routers. This may have also fixed the connectivity issues faced in the cafeteria. Another enhancement would be increasing the number of devices so they can be put at all access points making them more accurate. I would also recommend changing the way the data is displayed with "live updates" on the web pages. This can be done with some added Java Script or another language that interacts with HTML. The most reported issue is still the sensors sensitivity, I believe could be solved with a few statements of code to limit the number of times the device sends the ultrasonic signal.

These enhancements could make Smart Campus a very useful tool for administrators to track the occupancy of buildings and help determine the spread of Covid-19 based on student activity. This could also allow students to track traffic flow for building to determine busy hours and when a good quiet place is to study.

9. Defense Presentation Slides:

Presentation link

https://youtu.be/c7XSsw4244I

10. References:

https://tutorials-raspberrypi.com/raspberry-pi-ultrasonic-sensor-hc-sr04/

https://mariadb.com/resources/blog/why-should-you-migrate-from-mysql-to-mariadb/

https://www.tomshardware.com/features/raspberry-pi-vs-arduino