California State University of San Bernardino

iGarage Report

by
APPitizer Enthusiasts

Cynthia Milan
Brian Ayala
Miguel Jacuinde
Citlatly Garcia

CSE 5408
Spring 2021

Dr. F. Muheidat

Table of Contents

1 Introduction

1.1 Executive Summary

iGarage values every homeowner's accessibility and security. With garages containing one of our most prized possessions, security is essential. This system allows families to be aware of the status of their garage door. The APP-itizers made sure both are guaranteed by being able to open and close the garage through any mobile device with internet access, manually, and even through a keypad for emergency use. Homeowners will have their unique passcode for their keypad, passcode may only be shared with family members or anyone of trust allowing homeowners to have more control of who has access to open and close their home garage. Having access to the status of your garage door can give Forpeace of mind, especially for someone who may tend to be forgetful. iGarage will provide all customers with innovations that will lead to the security and safety of a homeowner's home along with easier accessibility to the current status of their garage.

1.2 Problem Description

The APP-itizers team focused on delivering a system that would provide homeowners with a sense of security for their homes. This system will allow security benefits to families by avoiding dangerous situations like break-ins or any unwanted visits from strangers. As a result, the team decided to move forward and use homeowners' valued security as our motivation. Our system consists of a sensor, to determine whether the garage door is opened or closed, a Raspberry Pi which wirelessly communicates with the server, a keypad for emergency use in case of any network connectivity issues, and lastly a server that could be accessed through any mobile device with internet access.

2 Requirements Specification

Unfortunately, the team had to communicate with the client regarding the change of scope for our project. Originally the group said that we would have a fully functional system with all the features both the client and the team agreed to. However, we were not able to accomplish what was expected of us and had a mandatory meeting with the client.

In the meeting, the client was informed that the team is more focused on getting the system to work and there would be certain features that might have to be removed to meet the deadline. The client and our group agreed that as long as the system was at least functional in the end, meeting its basic need then removing some features would be understandable. In the end, the security features like the timer, license plate scanner, and two-factor authentication were more of a want than need and were discarded from the project.

Another change of scope was the different sensors that we used. The sensor we thought of going with was more expensive and gave the system more features. Since some of those

features couldn't be met, we thought of going with the cheaper one with fewer features instead. That sensor was easier to work with because of its simplicity which allowed us to move on with other areas of the system. Switching sensors from one that didn't measure the distance of nearby objects to one that sensed whether or not the garage door was open was a difficult decision to make. Therefore, we spoke to the client once more about this other concern. The client understood that switching the sensors was more beneficial to the system because it made it more secure. The only way the previous sensor would be more secure would be if it was paired with the license plate scanner guaranteeing that the owner of the vehicle is the user.

There were many changes of scopes that our team and the client had to come together and resolve. Thankfully, the client was very understanding, and just like our team; we all prioritized a finished product with minimal features rather than an unfinished product with more features implemented within it.

3 System Alternatives and Alternative Selection

3.1 Alternative Designs

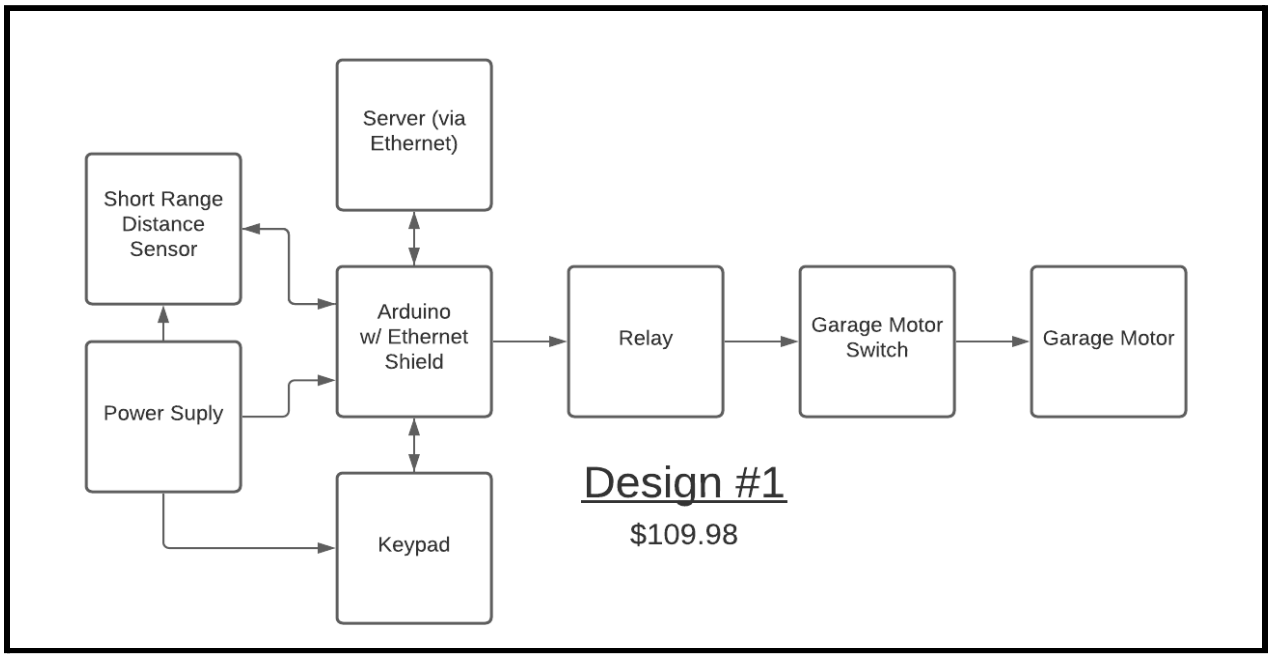


Figure 3.1.1: "Design 1"

The design must contain a small board that connects to a server or app for mobile access, a sensor to identify the state of the garage, and an emergency keypad in case the homeowner does not have access to a phone or experienced a loss of a phone. In order for the iGarage to work, the homeowner must have access to the internet and have a physical router within the property. The design can work with a wifi adapter for easier access to the program from the small board, however, an ethernet connection would be more efficient, faster, stable, reliable, and holds the most security. Since the team is most experienced in Arduino, this was one of the alternative

small boards. The most common and reliable small-board, is the Raspberry Pi 4 Model B, making it the other alternative small board.

The team APP-itizer Enthusiasts collaborated and created 3 alternatives designs, 2 of which conflicted with the appropriate milestones and budget. Although the team agreed in keeping the budget below $150, the idea was not to surpass it and attempt to maintain it as low as possible while still maintaining the feature functionality. The design from the first version of the system design revolved around figure 3.1.1, “Design 1”. Using the Arduino with an ethernet shield to directly connect it to the router for faster more efficient internet connection, with the short-range distance sensor, and the emergency keypad. However, the team reached a conflict with the server or app connection to the Arduino. The connection issue conflicted with the milestone the team was working on, ideally, the design could work in a matter of extra time, but the prototype must be completed by May, without the extra time the design was inefficient. Thus, the two designs below figures 3.1.2 and 3.1.3 were developed to complete the prototype within the time limit.

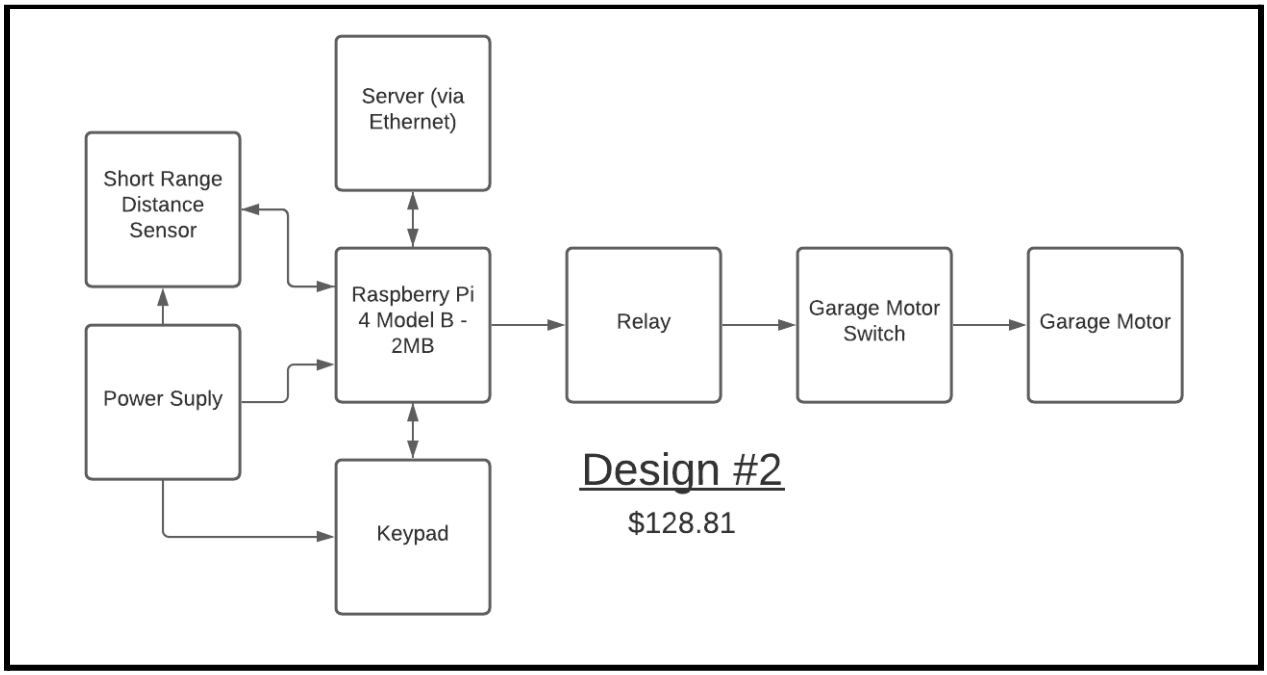


Figure 3.1.2: “Design 2”

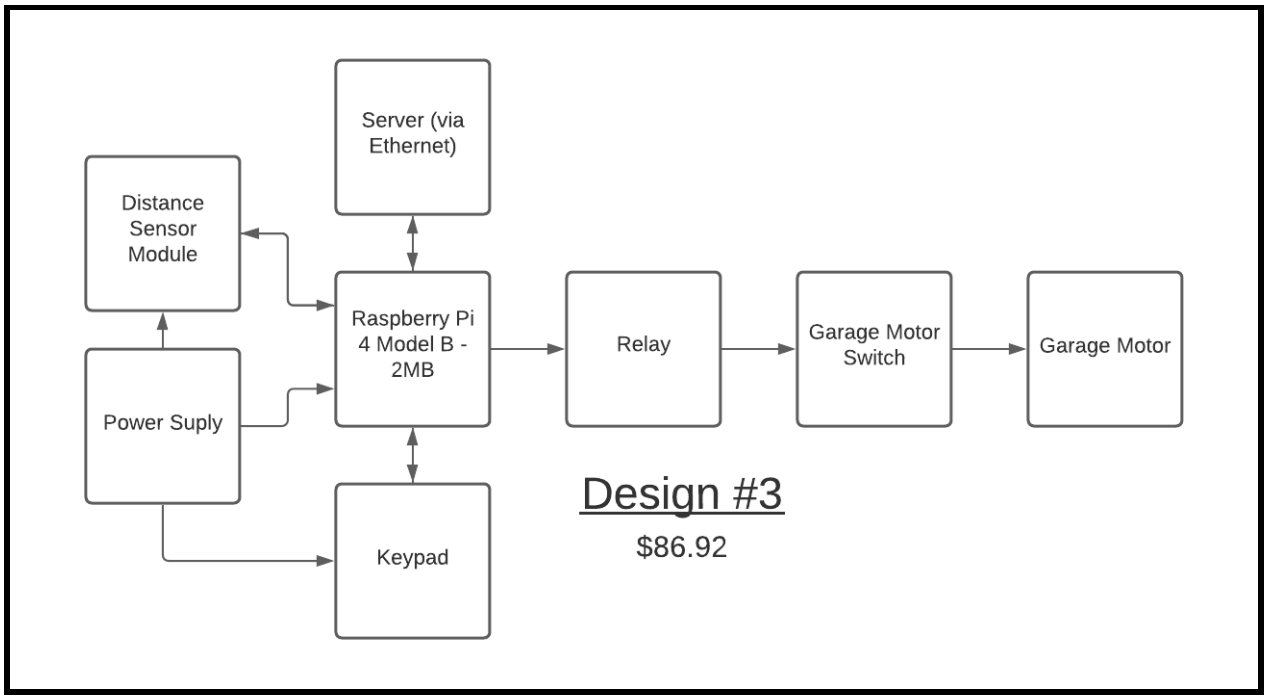


Figure 3.1.3: "Design 3"

Figure 3.1.2, "Design 2", describes the second alternative design, which replaces the small board with the raspberry pi. The board replacement provides a straightforward connection to the server using the IP address given. A possible conflict for the Arduino board was the use of I2C communication, the raspberry pi avoids the extra wiring completely, giving a clean circuit with fewer wiring. The short-range distance sensor affected the milestones, project timeline, and risked the project itself since there was no guarantee that the sensor would successfully work with the board. The team had to take into consideration that the sensor would have to be manually coded and there were no references to refer to, causing an issue with the project timeline. The sensor was advertised for the Arduino, so although there was a good possibility that it was compatible the fact we still needed to code to test it would take more time than expected. The last alternative design held the same components as figure 3.1.2, except it used the alternative sensor kept for backup.

3.2 Design Selection

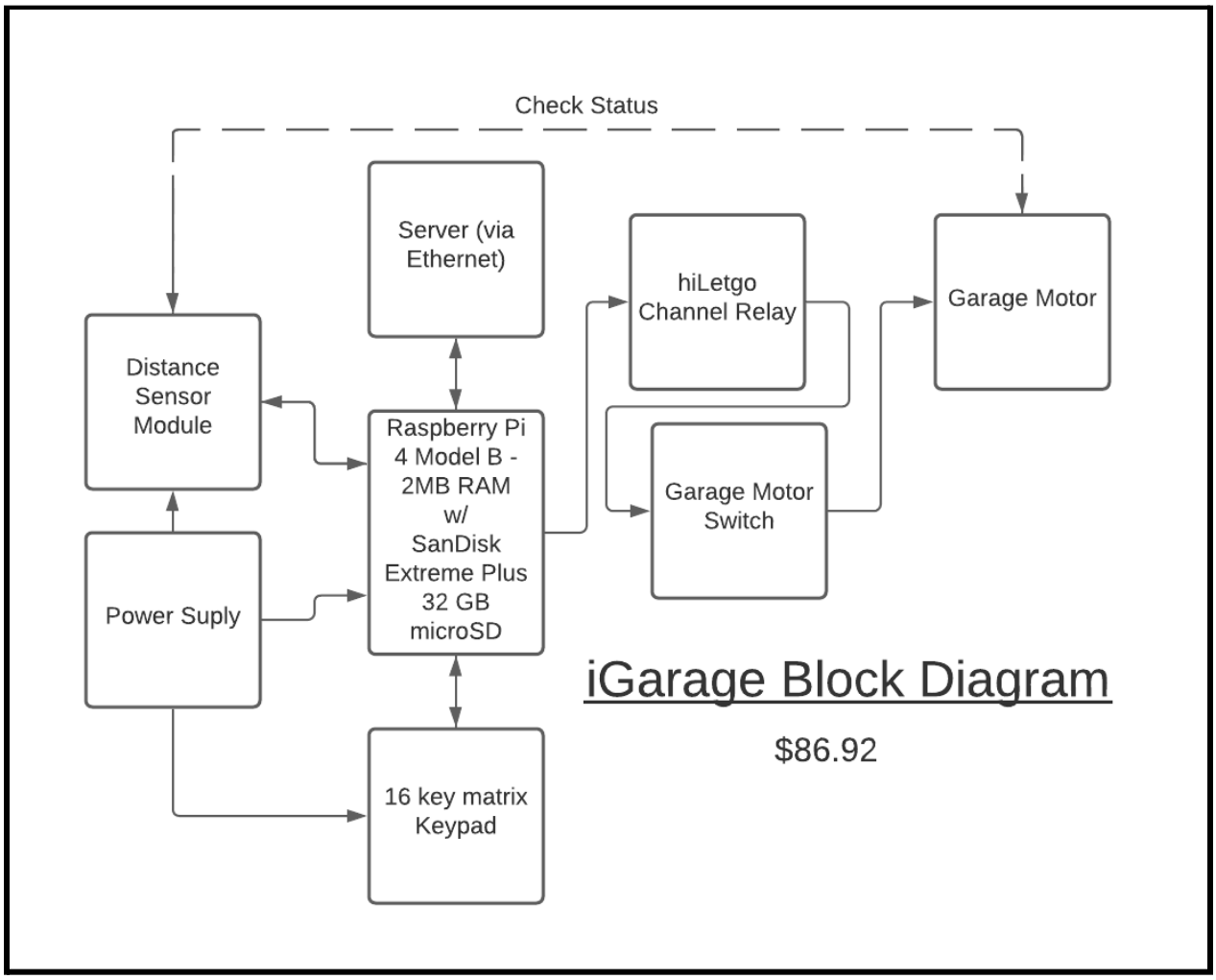


Figure 3.2.1, "iGarage Block-Diagram"

With proper analysis of all designs, the selected design was the block diagram above, as brainstormed in figure 3.1.3, "Design 3". This design was the simplest to build, held the cleanest circuit, and was the most reliable to work with. It can be developed within the budget, and saved the most amount, while still keeping the materials that were the most compatible and reliable. The references used were able to guide the team successfully through the coding process, providing the most assistance for the hardware since that gave the rest of the engineers time to assist in troubleshooting minimal errors before the final prototype. The raspberry pi avoided the I2C communication the Arduino would most likely need, which would have surpassed the project time limit. The board provides easier access to the server which can be reached with anyone holding internet access within the property radius. The backup Distance Sensor has guaranteed compatibility with multiple resources for the coding portion of the project.

4 Design

4.1 System Design

iGarage is very straightforward. The user enters the browser with the given IP address of the Raspberry Pi, followed by:1234/login. Once the user has accessed the browser, they will be prompted to enter the garage passcode which is pre-set within the code - it can be changed at any given time. If the passcode is correct, the garage opens and the user is prompted 'Garage Door Opening'; on the contrary, if the passcode is incorrect the user is prompted with 'Incorrect Passcode'. This method can be used to open/close the garage. The original garage buttons will also remain working therefore, they can still be opened/closed as before. The user has access to the garage state by entering the sensor state browser. This can be accessed by entering any browser and typing in the pi's IP address followed by:1234/sensor. It'll prompt the user to enter the garage's passcode as well, and once entered, if the garage is opened, it will display 'Garage Door is Opened' and vice-versa. Lastly, iGarage also comes with a manual keypad which can be used just like any other garage keypad. The user will manually enter the passcode and if correct, the garage will open, else it'll remain closed.

4.2 Detailed Design

Relay: The 5V relay module has three inputs (VCC, Ground, and IN) and three outputs (NO, NC, and COM). In this case, VCC was connected to a 3V pin on the raspberry pi (pin 17), IN was connected to GPIO 4 (pin 7), and Ground to a ground pin (pin 9). As for the outputs, NO and COM were connected to the garage motor, specifically the terminals that are wired to the garage door button, via 2 core wire. The relay is set to active low and when the code is run, the relay is set to high and then back to low imitating a switch causing the garage door to open.

Figure 4.2.1 "Relay connection to garage motor"

Sensor: The HCS 3V sensor has 4 inputs (VCC, Ground, Echo, and Trigger). The VCC is connected to a 3.3V pin (pin 1), the Ground is connected to ground (pin 25), the Echo input is connected to GPIO 10/SPIO MOSI (pin 19) and the Trigger input is connected to GPIO 11/SPIO SCLK (pin 23). The short-range sensor is placed in front of the garage motor so that when the garage is closed, nothing is disturbing its path, hence receiving zero detection implying that the garage is closed; vice-versa, if the garage is open, the door will be raised until it reaches the sensor's detection (detection means signal interruption implying the garage is open).

Figure 4.2.2 "Sensor location"

4x4 Matrix Keypad: The keypad does not need any sort of VCC connection other than that powering the actual Raspberry Pi. It has 8 inputs (4-row inputs and 4column inputs). The first 4 inputs are the row inputs that will get information for each row in the matrix, the last 4 inputs will get information for each column in the matrix. Using the row and column together, the pi will be able to pinpoint an exact button (location) on the matrix when pressed. The row connections are GPIO pins 5, 6, 13, and 19 (pins 29, 31, 33, and 35 respectively). The column connections are GPIO pins 12, 16, 20, and 21 (pins 32, 36, 38, and 40 respectively).

Raspberry Pi: After all the components are properly connected, the Raspberry Pi needs to be connected with a 5V power source. The Pi can be mounted on top of the garage motor if given enough space or on an enclosure that will best suit the garage motor space.

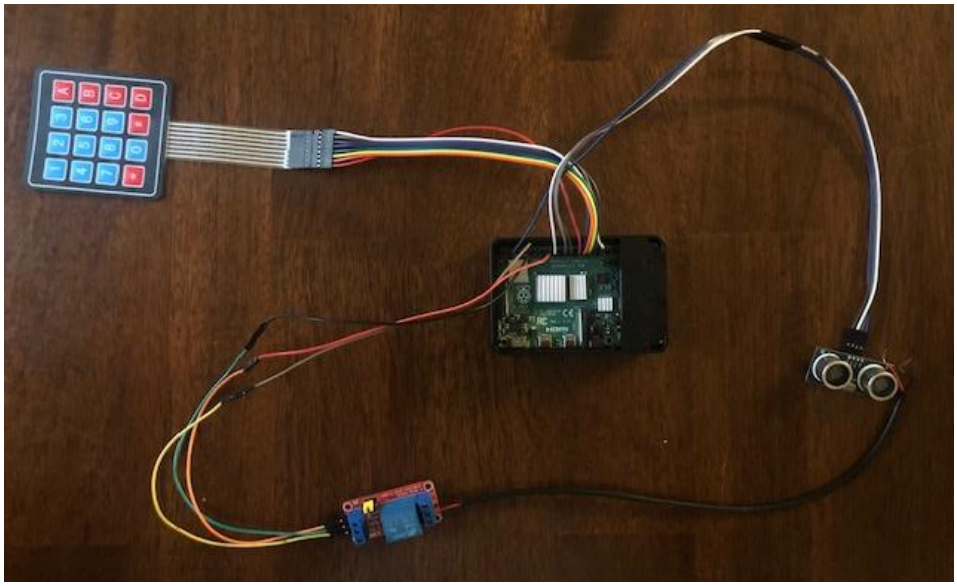


Figure 4.2.3 "Component connection overview"

Running the Program: The system is now ready to be activated. Turning on the Raspberry Pi and accessing it via VNC viewer, run the garage door program. Once the program is running, it is ready to be accessed through any web browser, whether via phone, laptop, or anything else with internet access. Enter the pi's IP address followed by:1234/login to open/close the garage door. The user will be prompted to enter the passcode to the garage door which is set within the coded program (can be updated at any time). If the user wants to check the garage status they can enter the pi's IP address followed by:1234/sensor. Again, the user will be prompted to enter the passcode to the garage door to check its current state. If the user wishes to open the garage door manually, they can do so via the 4x4 matrix keypad. The user simply inputs the same passcode manually followed by the 'A' button to accept the code. If the user messes up, they can push the 'C' button to clear all input and try again.

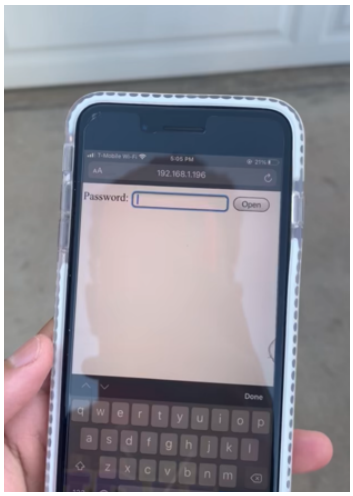


Figure 4.2.4 "Browser POV"

5 System Test Plan and Results

5.1 System Test Plan

5.1.1 Equipment Testing

Equipment Testing		
Equipment	Status	Notes:
Raspberry Pi 4 Model B 2MB RAM	90% passing rate	First board malfunctioned, replaced (-10%)
SanDisk Extreme Plus 32 GB microSD	100% passing rate	Passed test instantly
3.5A USB-C Power Supply, 5V	100% passing rate	Passed test instantly
HDMI	100% passing rate	Passed test instantly
Short Range Distance Sensor Module	0% passing rate	Failed to test
hiLetgo 5V Channel Relay	90% passing rate	Needed to be replaced, 12V to 5V to fit power supply and raspberry pi (-10%)
16-key matrix keypad	80% passing rate	Constant issues, needed to be replaced 2 times (-20%)
Distance Sensor Module	100% passing rate	Passed test instantly

Table 5.1.1 "Equipment Testing"

The equipment was tested individually for preparation. The team faced a few challenges during the testing of the materials. The raspberry pi malfunctioned the first time it was tested due to a built-in flaw and was sent to be replaced. Once replaced, it successfully ran as it was supposed to, the team deducted 10% for the first failure test, leaving a total of 90% passing rate. The microSD, power supply, HDMI, and the alternative distance sensor module passed successfully with a passing rate of 100%. Due to the power supply being low, the relay needed to be replaced, leaving it with a passing rate of 90% for having to replace it once. The one that caused the most issues was the keyboard with a passing rate of 80%, it failed twice when tested and remained unresponsive until it was replaced with a third one.

5.1.2 System Tests

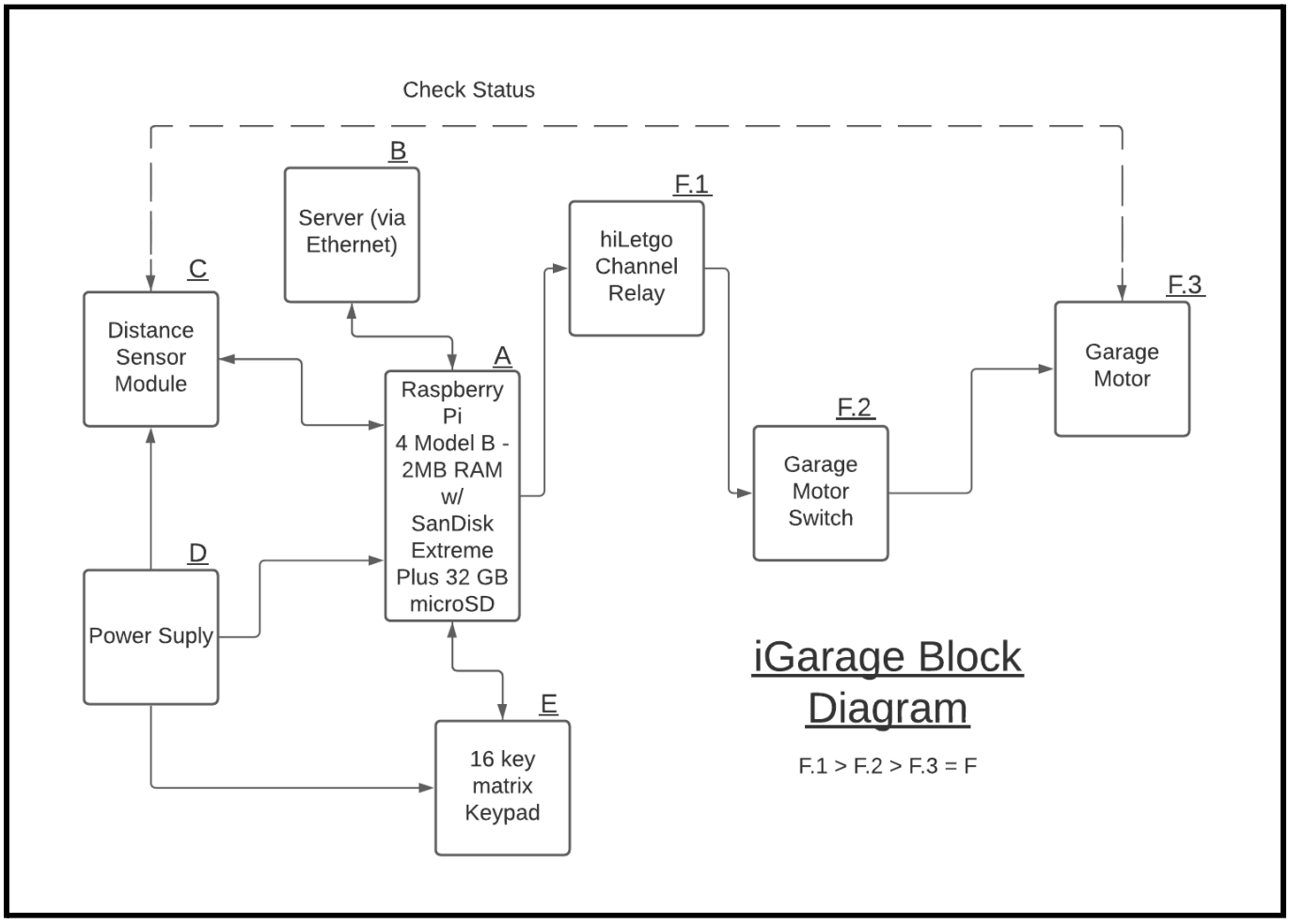


Figure 5.1.2 "Block Diagram Routes"

To ensure the system testing meets the requirements, each route from the figure above should work independently and together as one. Each route describes the required output when two stages cross paths, each stage is labeled from A to F. Each of the system features is described as a path between two stages. F.1, F.2, and F.3 are settled as one single stage since it belongs to the garage itself without any extra system features.

- ➔ A to B
  - ◆ This path is tested for access to the internet via an ethernet cable, which should be the most reliable and secure. A gives the received data to B and returns the command to A. A then sends the instruction to the next stage. The expected output for this is to give a success comment when the instruction is sent successfully, or an error message if it doesn't run.
- ➔ A to C

- This path is tested to ensure that the sensor is determining the status of the garage. This is very important since it is the main security feature. A sends the instruction received from B to C, after C gets the required data, A retrieves it and sends it back to B. The expected output depends on whether B was able to determine the status of the garage, whether it is open or closed.

➔ C to F.3(F)

- This path checks the status of the garage. C sends the new instruction data to F to check the distance between the sensor and the motor. The distance determines if it is closed or open, thus returning the status data to C.

➔ D to C,A,E

- This path provides the appropriate voltage for the equipment labeled as C, A, and E. The required output is to ensure that the equipment functions once the switch is on.

➔ A to F (F.1,F.2,F.3)

- This is the path that tests that the garage opens on command. A received the instruction data, in which it sends it to the parts of F, labeled as F.1, F.2, and F.3. The expected output is to either open or close the garage door.

➔ A to E

- This path is the emergency keypad, which still required the homeowner to have internet access. B sends the entered data to A, A then sends the instruction to B, which checks that the entered code was correct. At success, the instruction will continue to F, or the user has to input the code once again until they get it correct.

As described above, the expected outputs are what to be expected when tested in the system test plan. The expected result of the test plan for the Garage to successfully open through any mobile device, or manually using the keypad. The server must be able to print the status of the garage, which should be identified by the sensor.

5.2 Results

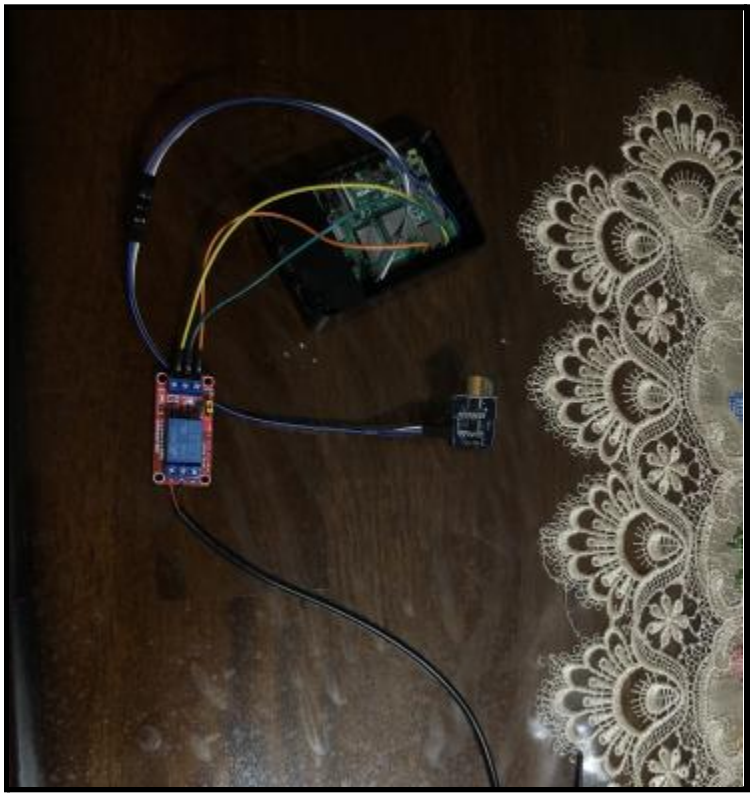


Figure 5.2.1 " First Prototype"

The first prototype had only the sensor and relay working, the keypad was not successfully working in the first few attempts. Only the distance sensor managed to work after it replaced the short-range distance sensor which was the first unsuccessful attempt. Figure 5.2.1, shows the first circuit with the relay and sensor; they worked successfully after replacing the relay to fit the 5V instead of 12V.

5.2.1 Prototype: Sensor

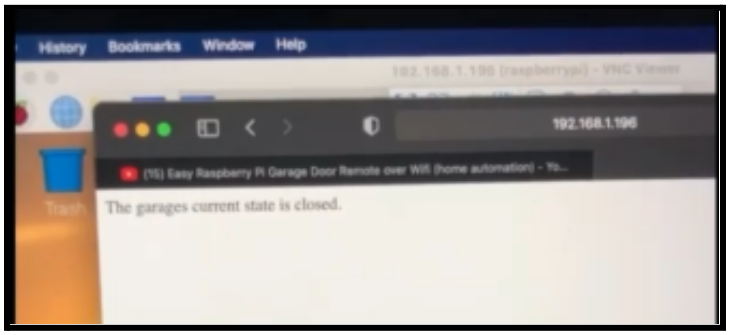


Figure 5.2.1.1 "Sensor Tested"

Figure 5.2.1.2 "Sensor with garage handle"

The sensor for the prototype was challenging to implement for the simple fact that the resources regarding any sensors for a raspberry pi were not compatible with the one we had. The sensor we had was the Short Range distance sensor, but the only ones that were successfully found through research were the regular distance sensor modules. After coming to terms that the sensor was not going to fit the prototype, the team decided to look into the alternative sensor recommended. Once implemented, the project sensed the garage door successfully and the door managed to open as well on command.

Figure 5.1.2.1 above shows the result of the sensor once it was implemented onto the device. The device managed to successfully determine the status of the garage door by sensing the manual pull rope, as shown in figure 5.2.1.2, within a few inches away from the sensor. If the handle is not within the radius of the sensor, then it recorded the status of the garage door is open, or else it remained closed.

5.2.2 Prototype: Keypad

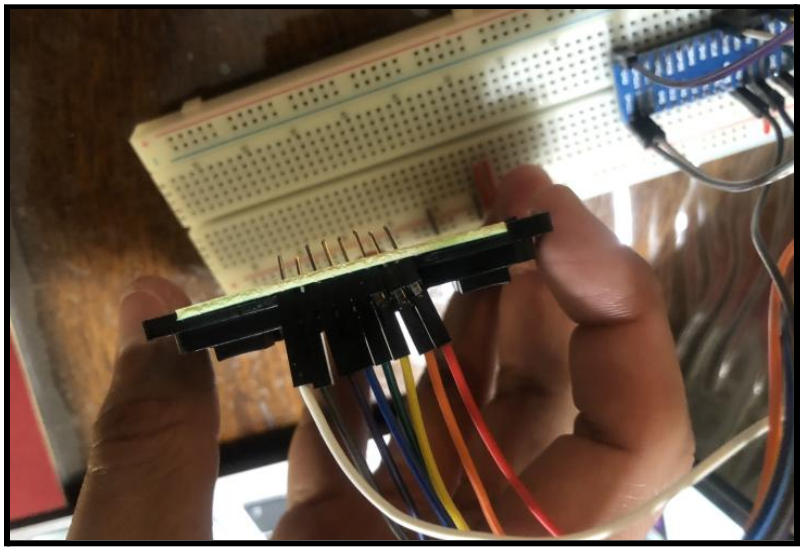


Figure 5.2.2.1 "Keypad 1"

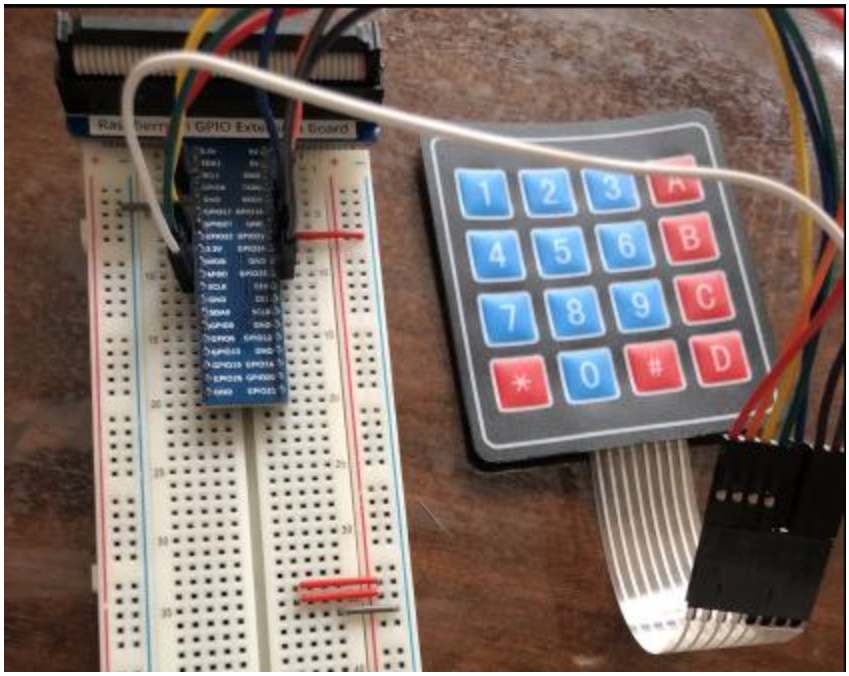


Figure 5.2.2.2 “Keypad 2”

The most failed attempts and biggest challenge were the keypads, this includes the device itself along with the codes used referenced. Figure 5.2.2.1 above shows how the first test failed. The pins were not soldered on which caused a lot of difficulties getting the keypad to run properly. Trying to avoid ruining the object that can be retired to keep the team within the budget, we replaced it with another set that doesn't require the soldering as shown in figure 5.2.2.2, “Keypad 2”. This set was a part of 5, which still managed to stay within the budget, but it failed the second attempt. Although it connected successfully it still failed the second test. After replacing it a third time, the keypad worked individually, however it still failed to work together with the actual prototype itself.

The code itself worked by itself but was challenging to work with the main code that had the sensor and relay instructions. After attempting to implement both codes together, we got a variety of errors. The keypad was not working properly, and the code that opened the garage was also having issues since the same logic applied to the keypad that was also supposed to open the garage door. After the code was closely troubleshot, the keypad was successfully running along with the sensor and relay.

5.2.3 Final Prototype

As shown in the figures from the sections above, “System Design” and “Detailed Design”, once the keypad was working successfully, the whole project came together easily. The garage opened from any smart device and through the keypad. The sensor also managed to easily detect the status of the garage.

The project works properly as long as the homeowner has access to home internet, without ethernet access, the smart garage prototype will not guarantee a successful connection. The garage would have to be opened manually, and the keypad will not work as it should. Since the server is connected to the internet via an ethernet cable, this lets the program receive and send instructions that allow the iGarage to work as expected. The instructions can be sent to the server through any smart device, that includes a smart tv, as long as the proper URL is entered. If any errors occur in the materials, nothing will happen. For example, if the user enters the incorrect code into the keypad, nothing will be opened. If the incorrect pin is entered into the server, it will deny access to the user, making it useful for others who try to access the pages. Therefore, unless the user had the proper server code, it will deny access and refuse to open the garage door. The team recommends that the user keep the given code in a safe place to avoid security issues.

6 Economic Analysis

Our team had major revisions to the system design as well as the hardware used for it. For that reason, the initial proposed budget was nowhere near what we had to invest in for the completion of the project.

In the beginning, our budget was originally about $230.00 that included mostly just hardware components for the system. That included the central hub that would control all the signals and commands, the Arduino and its wires, which the team was lucky enough that Miguel already owned. Based on the research we did, an Arduino on amazon would run us about $25.00 so we made the budget to $30.00. Following that, the next item on the list was the system's power supply, the heart of the system that would make it power on and move. Upon researching that item we saw that it would be about $8.00 so we made the budget to $30.00. Next, on the list was the case that would enclose the whole system, giving us a protective layer to avoid damage to the system. After researching that item we saw that it would be about $13.00 so we made the budget to $30.00. The last of the wires on our list was an ethernet cord that would enable the system to connect to the home's wi-fi. We figured since we had the internal system protected with the case we could invest in an ethernet shield as well. Research for that item showed us that it would be about $20.00 so we made the budget to $30.00. Once we had all the internal parts accounted for we began to look into the Arduino accessories that would give the system all the features we hoped for. The first accessory was the keypad that would allow for manual entry. Upon researching that item we saw that it would be about $9.00 so we made the budget to $30.00. The second accessory was the sensor that would be the indicator on whether or not the garage door opens based on the distance of the person's automobile. This was hands down the most expensive component of the project and the team spent the most time looking into it. After researching that item we saw that it would be about $49.00 so we made the budget to $80.00. We also purchased an inexpensive sensor just in case the other sensor was incompatible with our system. From this point on the team believed that we had all the needed hardware to have the

system fully functioning or at least turn on. However, we later found out that we needed a channel relay. Research for that item showed us that it would be about $5.00 so we made the budget to $10.00. Our team later discovered that this whole setup wasn't functional and we would have to turn to use a completely different board as the base. Fortunately, the professor was kind enough to have us borrow his for the sake of finishing the project on time. With that in mind, our team had to buy other pieces of hardware that were compatible with the Raspberry pi 4 not being part of the original budget. Still, we managed to find the few items we needed for a reasonable cost so late in the project time. The Raspberry pi micro SD ran us about $17.00 while the HDMI and Raspberry pi HDMI adapter were about $4.00 together.

We did not account for any cost dealing with software therefore we didn't make a budget for it. Hence, looking at the scope of things, even with all the different sudden hardware swaps, we were still able to be under our $230.00 budget and only spent approximately half of that.

7 Project Management

7.1 Schedule

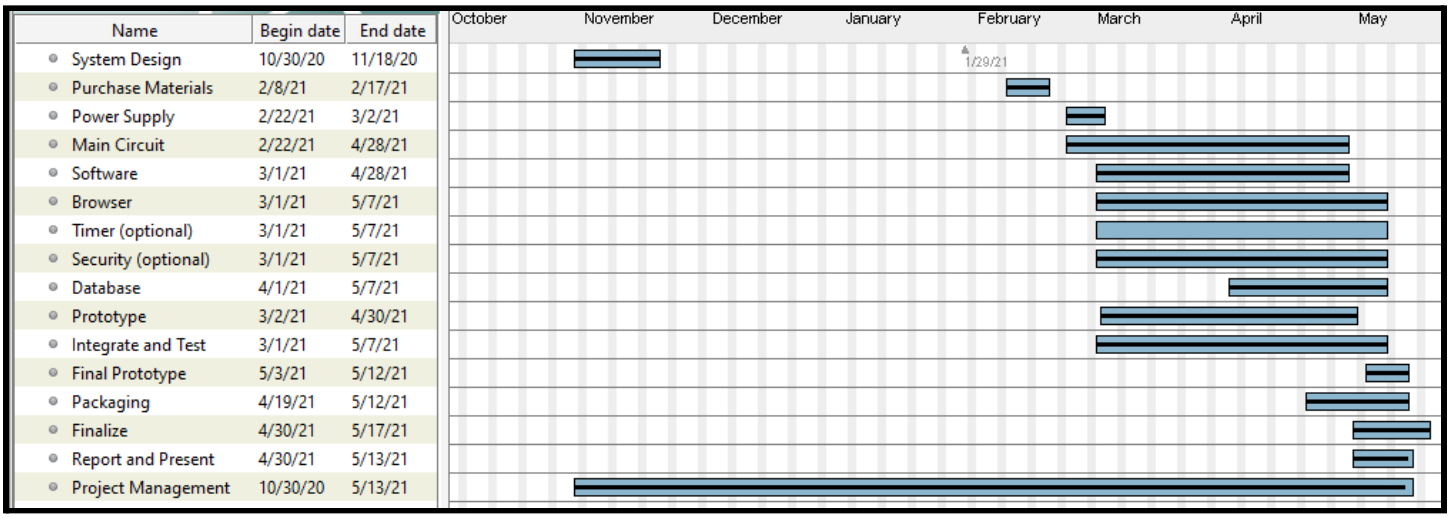


Figure 7.1.1: "Gantt Chart"

Due to minor challenges the team was unable to fulfill the chart from the first figure. Every team member had to input an equal amount of time to complete the project. This means that every team member is required to work every Monday and Wednesday on the project for the lab portion of the class plus the weekend or a total of at least 4 hours, making a total of about 6 hours a week per student. Miguel Jacuinde implemented most of his time with the hardware portion of the project requiring him to work every [day and time]. The rest of the team, Cynthia Milan, Brian Ayala, and Citlaly Garcia are in charge of finding the required codes, research, and troubleshooting the code as needed. Cynthia worked every day for at least 2 hours a day making sure that the team was on track, contributing and communicating with outside sources when

needed the extra assistance. Brian worked [days and times] a week contributing in research of links that can assist in troubleshooting the project. Citlaly researched codes that can help with the keypad and provide any links that showed tutorials using the lab time every week as needed and outside of class if necessary, giving an average of 2-3 hours a week.

7.2 Challenges

Change of Scope				
	Date	Description	Drawn	App
1	2/24/21	Changed small board from Arduino to Raspberry pi for simplicity	2/24/21	2/24/21
2	2/24/21	Worked with a server instead of an app using the raspberry pi's IP address to web host the project	2/24/21	2/24/21
3	4/05/21	Removed extra security features (i.e., login, profile, etc.)	4/05/21	4/05/21
4	4/14/21	Exchanged the Short-Range Distance Sensor to the alternative backup Distance Sensor Module	4/15/21	4/15/21
Drawn: 4		Approved: 4		Date:4/14/21
Module: iGarage (Smart Garage)				

Table 7.2.1: "Change of Scope"

The team faces a few challenges related to the project itself, since we changed from Arduino to raspberry pi, we had to return and repurchase some of the materials. We had to submit a change of scope late into the semester, changing minor things that would not be done on time due to the pandemic and lack of in-person meetings with the teams and professor. We modified the extra security features to simplify the home garage security. The changes are described in table 7.2.1 above. Due to difficulties with the server and app connection, and the possibility of I2C communication, the team decided to revise the first design and use the alternative board, the raspberry pi. The team chose to work with a server that was accessible through any smart device to avoid the timeline difficulties we would encounter while making the app, so we disregarded the app suggestion and continued with the server instead. Since the milestones needed to be completed on time, the extra security features that included a profile for the user were also disregarded. We simplified the security features with a given code that only the homeowner will be able to use to open and close the garage. The short-range distance sensor was also conflicting with the milestones, since there were no references regarding the coding process, and manually coding the program from nothing without guidance would affect the project timeline that needed to be completed by May. The alternate sensor managed to have a few resources which benefited the team to finish the project's final prototype on time.

8 Summary and Future Work

8.1 Project Summary

The APPitizer Enthusiasts had one goal in mind when we formed the group and that was to make the best smart garage possible. We agreed on choosing this project, later named iGarage, because we felt like it was a very useful product that could maybe even gain peer or industry recognition. For that to be a reality we had to make an iGarage unlike any other that we've heard of. We decided on building a design that would fulfill the basic requirement of opening a garage but also have the user have minimal to no interaction with the system while doing so. With this plan in mind, we began our journey to make the garage opener of our dreams.

Everything started with research and development. We all helped with looking into the smart garage market to see what components were the most common and which ones would be beneficial for our system. After the research was finished and our design was made, our team bought most of the components except for the Arduino board. Miguel already owned one and was kind enough to use it sorely for our project. Since he had the Arduino already at home, we decided to have Miguel set the project in his garage and have him primarily work on the hardware end while the rest of the team worked as software engineers. Next, we began to test the design we had first come up with foolishly thinking that it would work flawlessly. Unfortunately, the team soon found out that we missed a key component to our design, the channel relay, and without that crucial piece, communication between components was impossible.

Following up with the professor and other references to help solve our issue, we bought the correct channel relay. While Miguel worked on connecting and testing the newly bought channel relay onto the system, the rest of the team worked on developing all the software needed to have it accomplish everything we wanted. First, we attempted to develop a server through the browser to have it wirelessly communicate with the user's phone no matter the location. Next, we tried to develop the code needed to operate the sensor and keypad. Finally, we looked into developing timers and other features we wanted to add even though we figured it would be too late to implement.

After some time, Miguel notified us that connecting the channel relay was unsuccessful and the system still had a communication issue. The team reached out to the professor and he informed us that switching from an Arduino to a Raspberry pi would fix the communication issue. We took his advice and purchased all the necessary components hoping that things would finally go our way. Everything seemed fine and we were gaining progress until the Raspberry pi we bought fried on us while testing it. We reached out to the professor once more and he was generous enough to let us borrow his Raspberry pi

From that point on our team made significant progress while solving small hurdles that came our way. One of the problems was that our team did not have enough time to implement all of the features we had hoped for. This included switching from the expensive sensor with all the additional features to the less expensive sensor with basic features. More importantly, we were able to get the software working in a timely manner and have our project functional.

8.2 Future Project Investments

Due to major hardware issues and other non-project factors, we were unable to add all of the features we envisioned. One of those features was a timer that would enable the user to schedule when they would want the system to function on its own. We believed this would be a nice addition to our system if the user has a specific time that they arrive or leave home. Another feature we hoped to implement was a license plate scanner that would work alongside the sensor in determining when to open the garage. This would make our system more secure to use and it would give the user peace of mind that only authorized vehicles would prompt the garage to open. One other feature we would have liked to add was a notification setting where the user would receive a message through the app showing them the garage is open or closed. Speaking of an app, one of the team's original goals was to have an app accessing all these features instead of accessing it from a browser. We had other security features in mind like two-factor authentication, profile/master customization, and alerts. However, we understand that this is a work in progress and we are still extremely proud of what we have created.

9 References

[1] Hertz, Daniel. “How to Use a Keypad With a Raspberry Pi 4: Raspberry Pi.” *Maker Pro*, Maker Pro, 9 May 2021, maker.pro/raspberry-pi/tutorial/how-to-use-a-keypad-with-a-raspberry-pi-4.

[2]“How to Make a Raspberry Pi Smart Garage Door Opener.” *YouTube*, YouTube, 25 Apr. 2020, youtube.com/watch?v=An7KQbmUnhs&feature=share.

[3]Liftmaster Garage Door Opener Owner’s Manual http://embed.widencdn.net/pdf/plus/cgi/oix8sses0l/114a3083.pdf?u=gsn58x

[4]Shovon, Shahriar. “How to Measure Distance with Raspberry Pi.” *Linux Hint*, 1 Jan. 1968, linuxhint.com/measure_distance_raspberry_pi/#:~:text=You%20can%20measure%20distance%20using,)%20to%20400cm%20(4m).

[5]“Turn Your Raspberry Pi into a Web Server.” *YouTube*, YouTube, 29 May 2020, www.youtube.com/watch?v=9pn1KKhxwdM.

[6] upgrdman. “Setup a Raspberry Pi Web Server with Your Own .COM Using Google Domains.” *YouTube*, YouTube, 1 May 2016, www.youtube.com/watch?v=vzojwG7OB7c.

10 Appendix

Team Member Contribution	**Section(s)**	**Comment(s)**
Cynthia Milan	● System Alternatives & Alternative Selection (100%) ● System Test Plan & Results (100%) ● Project Management (100%)	● Proofread by Brian, Miguel, and Citlaly.
Brian Ayala	● Economic Analysis (100%) ● Summary and Future work (100%) ● Requirement Specification (100%)	● Proofread by Cynthia, and added comments.
Citlaly Garcia	● Executive Summary(100%) ● Problem Description(100%) ● References (100%)	● Looked over by Cynthia
Miguel Jacuinde	● System Design(100%) ● Detailed Design(100%)	● Given 2 sections for full contribution to hardware prototype. ● Proofread by Cynthia, requested pictures.

Table 10.1 "Team Contribution"

4x4 Matrix Membrane Keypad

This 16-button keypad provides a useful human interface component for microcontroller projects. Convenient adhesive backing provides a simple way to mount the keypad in a variety of applications.

Features

- Ultra-thin design
- Adhesive backing
- Excellent price/performance ratio
- Easy interface to any microcontroller
- Example programs provided for the BASIC Stamp 2 and Propeller P8X32A microcontrollers

Key Specifications

- Maximum Rating: 24 VDC, 30 mA
- Interface: 8-pin access to 4x4 matrix
- Operating temperature: 32 to 122 °F (0 to 50°C)
- Dimensions:
  Keypad, 2.7 x 3.0 in (6.9 x 7.6 cm)
  Cable: 0.78 x 3.5 in (2.0 x 8.8 cm)

Application Ideas

- Security systems
- Menu selection
- Data entry for embedded systems

Figure 10.1 "Keypad"

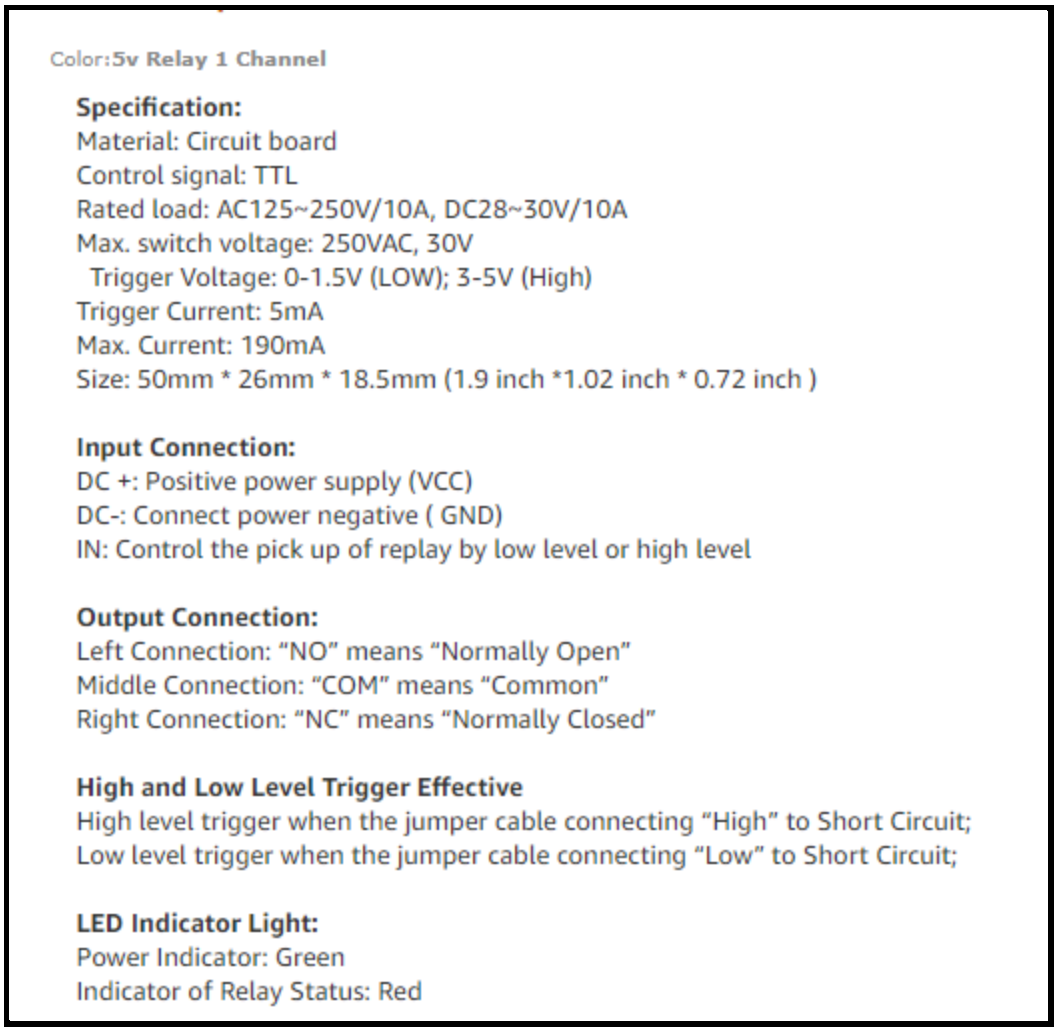


Figure 10.2 "Relay"

- Outside diameter of Jacket 3.5mm Typ. outside diameter of inner conductor insulation 1.1mm Typ.
- Jacket material: PVC (flexible flame retardant) Insulation: PVC Conductor: tinned stranded copper
- Conductors: 24 Awg (2 PVC insulated conductors red and black) Rated voltage 300V, rated temperature 80°C
- Round cord seals well into standard cord gland / cable grips.
- Applications: LED light power cord, street light power cord, indoor lighting cable, car light cable, electronic product internal cable, computer power cable

Product Specifications

Cable Length	25 feet feet
Color	25ft
Ean	0709327337270
Gauge	24
Material	Copper
Size	24AWG-2C
UPC	709327337270

Specification for this product family

Brand Name	CBAZY
UNSPSC Code	26121629

Figure 10.3 "Power Cable"

SainSmart Ultrasonic Ranging Detector Mod HC-SR04 Distance Sensor (Blue)

Providing 2cm to 400cm of non-contact measurement

- Stable performance, accurate measurement range.

- Using IO trigger for at least 10us high level signal.

More information please refer to sainsmart website.

Features:

- The ranging accuracy can reach to 3mm.

- The modules includes ultrasonic transmitters, receiver and control circuit.

The basic principle of work:

- The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.

- IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic to returning.

Distance = ((Duration of high level)*(Sonic :340m/s))/2.

Specifiction:

- Power supply :5V DC

- Quiescent current : <2mA

- Effectual angle: <15°

- Ranging distance : 2cm â€?500 cm

- Resolution : 0.3 cm

There are 4 pins out of the module : VCC , Trig, Echo, GND.

So it's a very easy interface for controller to use it ranging.

The all process is :

- Pull the Trig pin to high level for more than 10us impulse

- The module start ranging

- Finish ranging

- If you find an object in front

- Echo pin will be high level

- Based on the different distance,it will take the different duration of high level.

Distance = ((Duration of high level)*(Sonic :340m/s))/2

Figure 10.4 "Short Distance Module"

40pin dupont wire

Suitable for Arduino breadboard kit project, PCB project, PC motherboard, DIY experiment, 3D printer and etc.

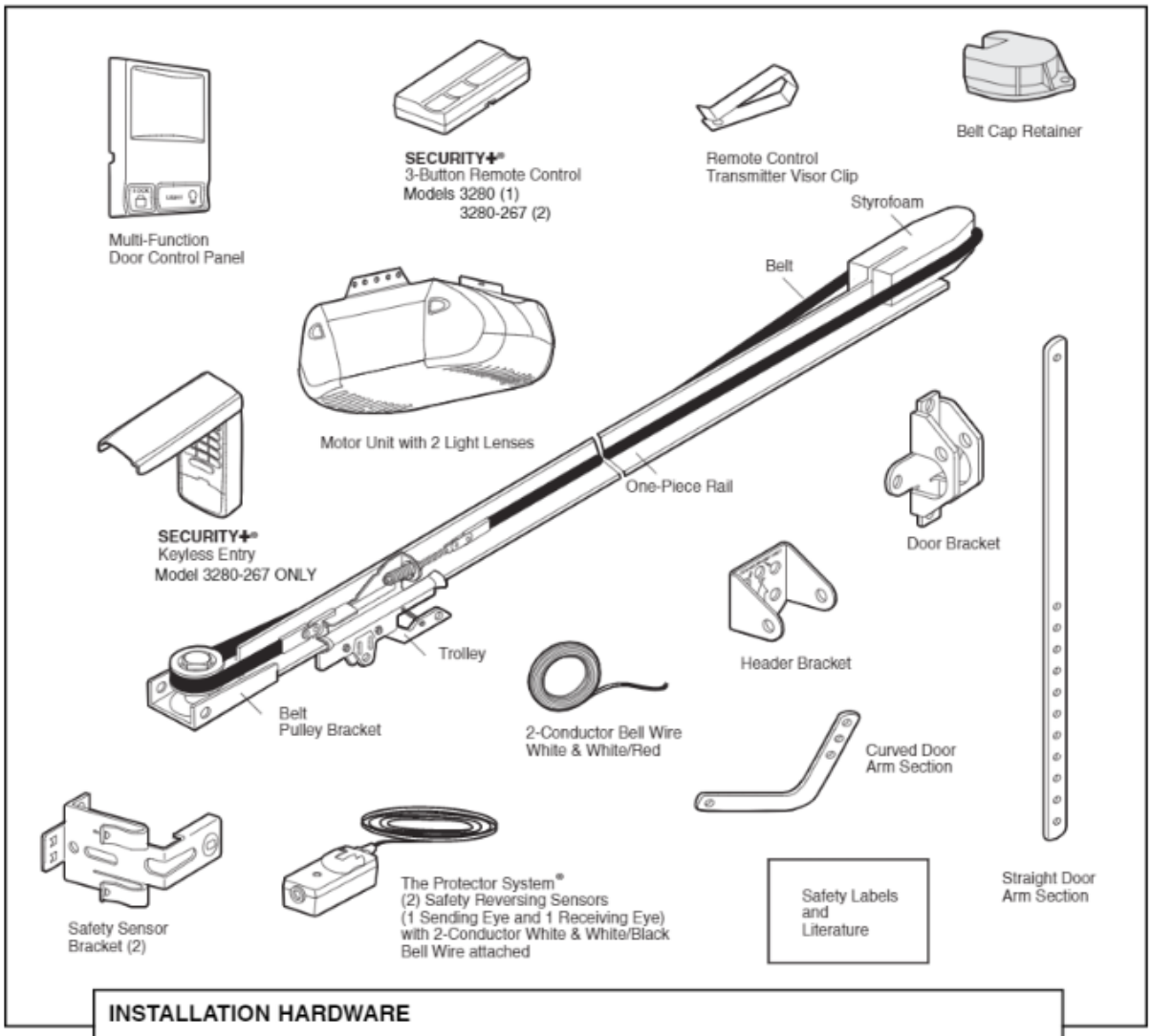
- Work on a Raspberry pi breadboard projects, Raspberry pi robot build, Breadboard prototyping, Arduino and ky-040 rotary encoders. In
- sert onto the SW LED pin on the typical desktop computer mother board. Tes
- t bread boarding projects or for any breadboard experimenter. Fit
- into a 3-pin JST.

- The male ends meant for insertion into standard 0.1 quot;(2.54mm) female sockets and the female ends are meant for insertion into standard 0.1 quot;(2.54mm) male headers.
- It can be used for the expansion of the experimental board pin and increase experimental projects.
- It can be reused if the terminal is not damaged, it can also be used for soldering if damaged.
- The cables can be separated into single root as you request to do multiple connection.

Figure 10.5 "Dupont Wire"

- 1.5GHz 64-bit quad-core ARM Cortex-A72 CPU (Broadcom 2711)
- **2GB RAM (LPDDR4 SDRAM) - also available in 1GB and 4GB versions!**
- On-board wireless LAN - dual-band 802.11 b/g/n/ac
- On-board Bluetooth 5.0 HS low-energy (BLE)
- 2 x USB 2.0 ports
- 2 x USB 3.0 ports
- True Gigabit Ethernet
- Extended 40-pin GPIO header
- 2x micro HDMI, 4k video
- 4 Pole stereo output and composite video port
- MIPI Camera port (CSI)
- MIPI Display port (DSI)
- microSD format for loading OS & data storage
- 5V/3A DC via USB type C connector
- 5V DC via GPIO
- PoE Enabled
- Multimedia H.265 decode (4kp60), H.254 decode (1080p60), H.254 encode (1080p30), OpenGL ES 1.1, 2.0, 3.0 graphics

Figure 10.6 "Raspberry Pi 4 Model B - 2MB RAM"



INSTALLATION HARDWARE

Hex Bolt 5/16"-18x7/8" (4)	Insulated Staples (30)
Lag Screw 5/16"-9x1-5/8" (2)	Ring Fastener (3)
Lag Screw 5/16"-18x1-7/8" (2)	Drywall Anchors (2)
Clevis Pin 5/16"x2-3/4" (1)	Rail Grease
Clevis Pin 5/16"x1-1/4" (1)	Carriage Bolt 1/4"-20x1/2" (2)
Clevis Pin 5/16"x1" (1)	Wing Nut 1/4"-20 (2)
Nut 5/16"-18 (4)	Rope
Lock Washer 5/16" (4)	Handle
Screw 6ABx1-1/4" (2)	
Screw 6-32x1" (2)	
Self-Threading Screw 1/4"-14x5/8" (2)	

Figure 10.6 "Liftmaster Chamberlain formula 3280"