BerrySafe

Final report

Team Pikachu

Tim Tran

Jason Tran

Patrick Cramer

Peter McAughan

Austin Akerley

Department of Computer Science and Engineering

Texas A&M University

Date: November 30, 2017

Table of Contents

1 Executive summary

With everything turning automated, cybersecurity now poses a larger issue needing to protect more sensitive user data, such as home addresses, more than ever. This means that home security and home surveillance technology should be implemented in every household in order to avoid potential break-ins and unwanted guests.

In addition to home surveillance systems, it also needs to be affordable. The technology to monitor a person's home should be affordable enough so that nearly every household is able to have as well as be simple enough for anyone to use. One of the biggest goals of BerrySafe is to implement an intuitive effective design that any user can enjoy.

The main goal of BerrySafe is to have home surveillance at the touch of the fingertips. This means that a live picture feed of a person's home will be sent from the Raspberry to an Android phone through a client server implementation (TCP, UDP, Socketgram connection). In addition, the user will also be able to manually sound the alarm which will be implemented using some sort of a buzzer and switch between cameras through the android phone application. Because information could be requested at any time on a person's home, the Raspberry will require a constant internet connection either directly via an ethernet cable or home wifi.

We expect that this will proactively prevent break-ins from going further. For example, intruders are more likely to abandon their plans at the sound of a loud alarm noise going off. While we can't prevent intruders from coming in, a home surveillance system has the potential to prevent break-ins and robberies from going too far.

Our final design implementation included an Android application that was able to arm and disarm the system via a button on an Android phone. When armed, a motion sensor is active and if movement is detected, a loud buzzer noise will go off to indicate that the alarm has been tripped. To disarm it, a simple disarm is signal is sent back to the Raspberry Pi which will turn off the alarm. In addition a view picture option was also implemented to see a capture from the camera in a room. The picture is sent back to the Android application after some processing. The image is first encoded into a string and then once the Android phone receives the string image, it decodes it back to the image (a PNG, JPG, etc.).

We were able to consistently replicate a break in or trip in the alarm as well as take photos of the intruder and environment on command. The motion sensor was able to pick up movement in the room as long as it was within the scope of the sensor. The door sensor is connected via bluetooth from the Arduino and triggers once the door has been opened. The sensor is flexible as it can be used for windows and virtually anything that hinges. The camera is able to responsively take pictures upon activation. Overall the system worked as planned.

To complete the project, we divided the testing of the sensors to different group members. The goal was to test all the sensors to make sure they were functional and fit our use and then integrate all the sensors into an Android application at the end. From a team management perspective, this worked pretty well but the integration process is a lot more difficult than expected. Problems such as thread timing and socket synchronization had to be taken into account. Communication between group members went relatively smoothly. Everyone chipped in when something needed help and when setbacks happened, they were taken care of on the day of or by the next day.

2 Project background

Home security has always been an important issue for society, and now with your information on the internet it's easier than ever for an invader to find a target. In particular, we're going to focus on the monitoring of a room and entrance for this scope. When considering a security system, one should hold two objectives: keep the assets safe, and maintain peace of mind for the owner. Thus, a security system should hold a dual ability to maintain security in the home, while offering a way for the owner to access status and updates easily. We aim to achieve the following by creating an android application that allows the user to monitor their stationary camera with a single input command. Visual feedback by image allows the user to witness an invasion immediately when the home security system is armed.

2.1 Needs statement

Now more than ever, there is a need for affordable home surveillance without the assistance of a third party. The current market for home surveillance requires that home-owners sign a contract with a surveillance company who will take care of all the security aspects in a person's home. This is costly as alarm companies require a person to pay monthly but we're proposing a one time buy that is able to monitor a person's home through security cameras controllable through an android application.

2.2 Goal and objectives

The goal of BerrySafe is to make home surveillance as easy as simple as opening an app on a person's smartphone. There shouldn't be any need to phone a security company to see what is going on in a person's home. A user should have full control of when they need to view their homes.

Our objectives are to create a home surveillance system that is able monitor a whole room using a Raspberry camera unit. In addition to the camera, the information received from the camera unit will be able to capture screenshots as well as sound an alarm in case of unwanted guests using motion sensors.

The information received from the cameras and sensors will be communicated to the Raspberry Pi using a client server architecture. This would mean that the Raspberry Pi must be connected to the internet in order to send information as well as receive information.

The Android app will be the staple of the project. It will be able to remotely control what the user is able to view through their smartphone. It will be able to view different cameras throughout the home as well as manually trigger the alarm as opposed to waiting for the alarm to motion sense.

UI Design will also be kept in mind as we want the interface to be as simple and intuitive to use as possible. One of our goals is to have users be able to access surveillance information as quick as possible and then move on with their day.

2.3 Design constraints and feasibility

The first constraint that comes to mind is an upkeep constraint. We wanted to take the "set it and forget it" approach to this system, which means that users shouldn't have to worry about replacing batteries or similar tasks. This implies that we shouldn't use wireless devices, and thus we have to figure out how to wire all the components in an effective way. Wire extensions and configurations are an important part of this project. However, over the course of the project we determined that some components of our design should be wireless because there isn't much appeal to having cords and wires scattered across a home.

The scope of this project is another constraint. With security in mind, there are countless paths you can take with how many assets you want monitored at once, what control the user has over the monitoring methods, and different alarm systems you can set. With this project, we only have a limited amount of time and money, so we can't take security to an extreme and have to prioritize what we want to monitor and how we want to do so. We ended up using two cameras stationed at different angles to capture the entire environment. A motion sensor is set up at one door because if we were successful in monitoring with one entry, we can produce the same effect with multiple entries in a household.

A technical constraint we came across was figuring out how to properly integrate the Java socket properly to communicate with the Raspberry pi. While having the android application serve as the client connecting to the Raspberry pi, we needed to figure out how to send an image from the Raspberry pi to the android application. The inability of the android application to use Java's awt library prevented us from implementing a possible solution to sending an image, so we had to figure out another way of doing so. Also, the Raspberry pi is not able to fully utilize the sarxos webcam capture libraries to control the usb

camera connected. While the sarxos webcam capture library worked on a machine, we had to figure out another method that did not use the above library in order to functionalize our camera.

2.4 Literature and technical survey

Simplisafe is an existing product that's related to our project. This is a security system that's connected to your mobile device and allows for updates upon request, as well as alarm systems that go off upon intruder detection. This is different from our project in design, however. Simplisafe systems are wireless, and require a premium monthly subscription for continued functionality. This is the opposite of our approach, where users won't have to alter the system unless they specifically want to. Batteries or other upkeep is nonexistent to minimal in our design, which we feel is more dependable and robust. This product gives great examples of user UI in a security system, and the lightweight feel of only having cameras and motion sensors in a house.

Another similar security system is canary.is. This is another system that offers a package of cameras, temperature sensors, and motion sensors to be placed around the house. The same wireless issue presents itself in this product, where users will have to worry about charging their components eventually. Another issue, however, is that canary.is allows for live HD video streaming to your mobile device. This inherently requires a lot of bandwidth and so for users with a weak signal or a device without a lot of computing power, this functionality clogs up their device. Our design offers the same principle of updates-upon-request, but offers a screenshot instead of live video to avoid this bandwidth bottleneck issue.

VISTA-15PSIA is another existing product that is related to our project. This security system is an integrated alarm control system that is capable of adjusting the minimal exit delay. The main focus of this product is to reduce false alarms while providing the necessary and essential benefits of a regular home security system. This home security system, while effective at producing less false alarms, doesn't include an android application to monitor the home or provide visual surveillance. Our approach provides both functionality while also allowing the user to disarm and arm whenever they please. The price for the VISTA-15PSIA is approximately $170 or $11 monthly, but our product provides more functionality.

The Fortress Alarm security system is another existing product that provides similar functionalities and more. This home security system includes multiples ways of connecting to the home security system. Connection by GSM cellular SIM card, existing Wi-Fi, landline, or VoIP phone are some examples of how it can connect. Along with easy installation, this security system provides many other detectors, sensors, and alarms that fits into various lifestyles. Similar to our project, the security system includes an application available for download on iOS or Android. What makes our product better than the Fortress Alarm system is that we made it more simple by only including the necessary functionalities. BerrySafe

also includes a camera to send a capture of the home at a click of a button through our android application.

The LiveWatch home security system provides a product that includes motion sensor detection, a mobile app to arm or disarm, and an HD camera to monitor the home. Besides burglary, this home security system can protect users from fire, carbon monoxide, extreme temperatures, and more. The Plug&Protect package comes with additional accessories that complement with the touchscreen and system control included. However, our product has an android application that allow users to arm/disarm their home on-demand, but the LiveWatch home security system only allow users to arm/disarm their system within 175 feet from the control. Our product simplifies the life of the user by complementing with users who already uses their phone daily.

2.5 Evaluation of alternative solutions

The first alternative solution is to use Simplisafe systems for your home. Simplisafe systems is great and provides customer support along with frequent software updates for your mobile application. This solution is based upon frequent upkeep. Keeping track of updates and the battery life of wireless components, if maintained correctly, results in a state-of-the-art system that can keep a home secure and safe. The particular pros of this solution is reliability given effort. The con of this system is if the user forgets to charge batteries or update their software. Given this, it's possible for an intruder to exploit these by targeting known software bugs in deprecated systems or time their breakin to coincide with a battery outage.

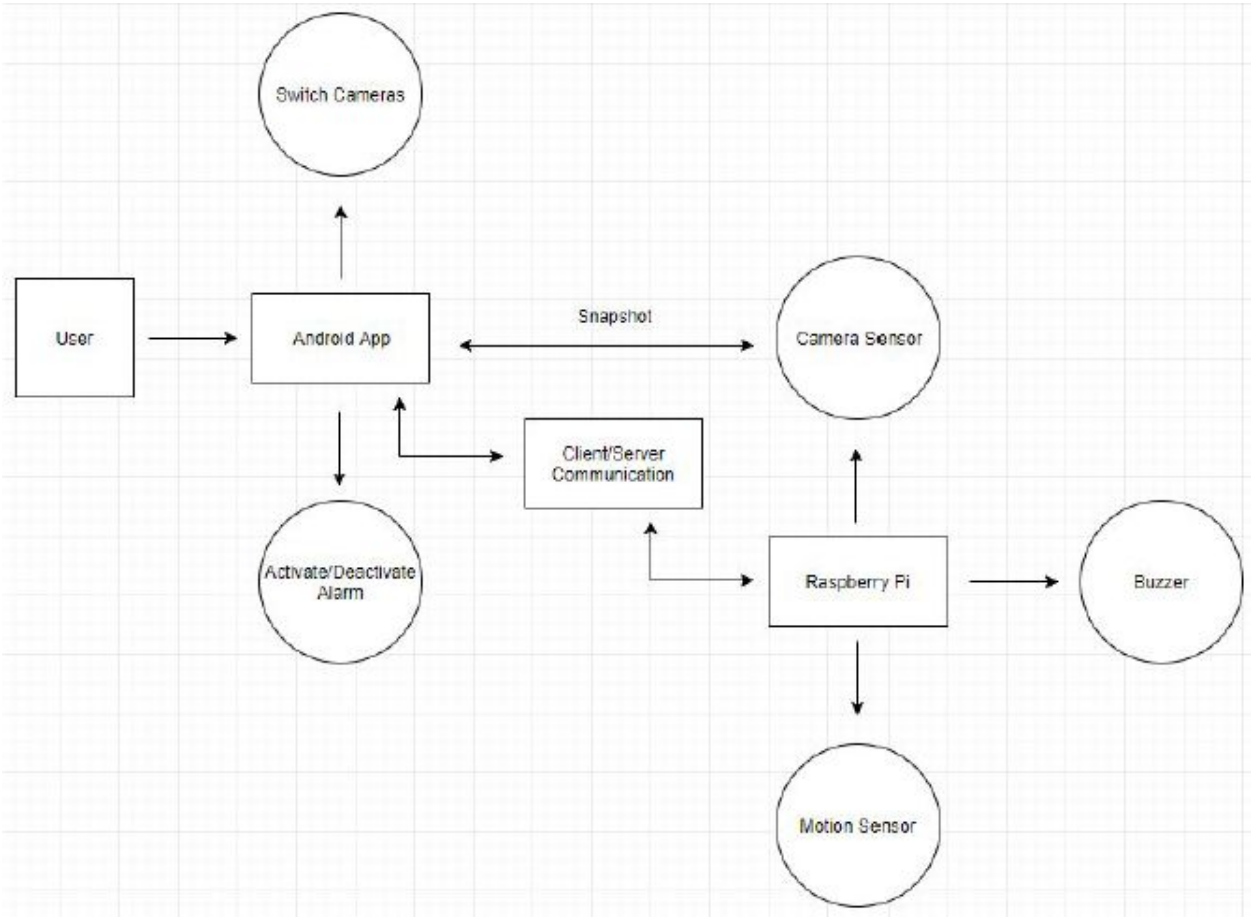
The second solution is to use a more comprehensive home security system such as ADT, a home automation and surveillance company. ADT provides sensors on doors, temperature sensors, cameras, intercom systems, and 24/7 monitoring support by ADt staff. This approach is very integrated into the home environment and can successfully ensure safety. However, this can also be much more than needed. Temperature sensors are not so useful to users who aren't prone to fires, and there are definitely users who would rather not spend the extra money to integrate an intercom system. Monthly installments are a pain too, and can drain users financially. Another con is that the control of these services are through ADT. They are the in-between for communications to ensure safety, but they don't offer as extensive control over systems as our proposed plan. Some users may have aversion to that, and would prefer to control their setup and know that all information transfer is happening between them and their system, with no in-between.

The third alternative solution is to use the Fortress alarm security system for your home. The Fortress security system is literally a fortress in that it provides an alarm system for many various things. Along with a 24/7 online support team, automated calls, portable systems, the Fortress alarm system constantly

monitors your home for any motion when armed. The pro of this solution is that it includes an application that's downloadable on both Android and iOS. The con of this solution is that there is no camera to display a visual image onto the application on-demand.

3 Final design

3.1 System description



User

The user would have full control of the Android app and be able to operate the three main functionalities within the application.

Android App

The three main functionalities of the Android application is switching between two cameras, activating/deactivating the alarm, and taking a snapshot of the cameras.

Switch Cameras
Through the app, the user can switch between two cameras in order to view the environment's surrounding.

Activate/Deactivate Alarm
While being able to activate and deactivate the alarm through the Android app, the user can properly determine what action they should take next before leaving their home or after the alarm sounds.

Client/Server Communication
Being the bulk of the system, the client/server communication is key to having communication between the raspberry pi and the Android app. The android application would act as the client sending a signal to the raspberry pi, the server. The server would acquire the connection signal and send a captured image and display it back to the android application.

Raspberry Pi
The pi controls the buzzer, camera sensor, and motion sensor. The data that would be acquired will be used to determine if the alarm should sound or not.

Buzzer
The buzzer will sound if an alarm is activated.

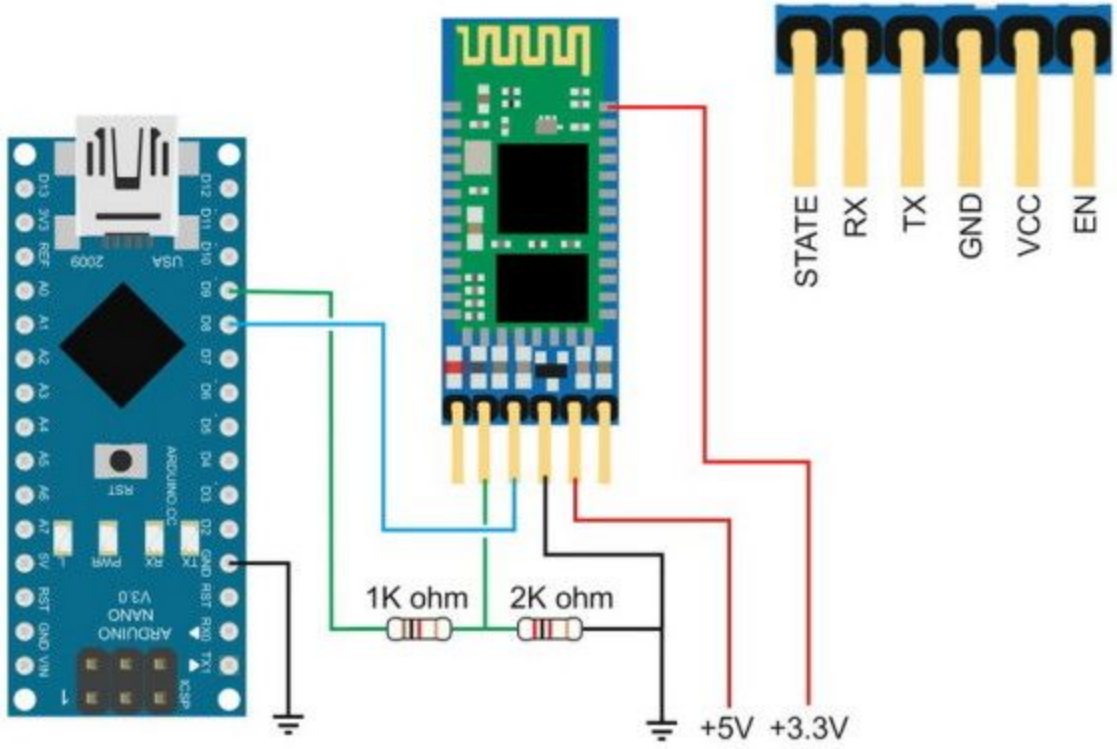
Motion Sensor
The motion sensor is used to determine if there's movement at the door if the security system is on. It would use an arduino bluetooth motion sensor to detect for any movement and report back to the raspberry pi.

Camera Sensor
Display a captured image of the environment that's viewable on the Android app.

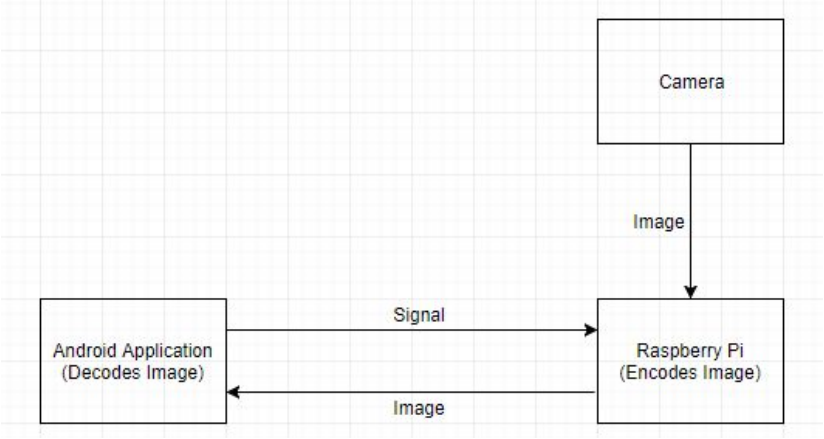
Snapshot/View
Be able to take a snapshot of the camera anytime through the Android app.

3.2 Complete module-wise specifications

Arduino Bluetooth Module Pinout



Client/Server



Parts

- ELP fisheye cameras
- Camera mounts
- USB Extension cables
- Raspberry pi motion sensor
- Window/Door motion sensors
- Alarm Buzzer
- Arduino bluetooth
- Android phone

3.3 Approach for design validation

Our system is tested by components to make sure the entire system works as a whole. The validation tests are:

- Ensuring that the cameras and sensors are positioned in a way to capture essential perspectives for monitoring a house.
- Ensuring that the cameras can relay correct data to the Pi effectively.
- Ensuring that the Pi can form a connection to an Android application and transmit this camera data correctly and effectively.
- Ensuring that the data is presented effectively in the Android mobile application to the user.

These validation tests are going to have sub-tests that follow logically. Each sensor is going to be tested when it is mailed to make sure each component works correctly, and the Pi should be giving updates on what sensors are attached and what isn't. One of the most important is the user feedback test for the application. A big part of this project is ensuring the user can interact with their security system, and thus this mobile application must be intuitive and easy to use, so a user can efficiently use it.

Demonstrating our project on demo day will consist of setting up our system across the 462 computer lab or anywhere with a stable Wifi connection and hooking it up to an android emulator on a laptop. Since no one in our group owns a physical Android device, an emulator must be used to run the mobile application. If an Android device is provided before demo day, we will have a GSM cellular card with the Android device to demonstrate the application. The demonstration will involve arming and disarming the system, requesting and receiving camera images, and setting off the security alarm by intruding on an armed system, or, opening the door.

4 Implementation notes

The hardware connection used by the BerrySafe system relies on the exchange of data between the raspberry pi, android device, and camera using wired connections and bluetooth connections.

The motion sensor is connected via an arduino nano using a hc-05 bluetooth module to communicate with each other. From following the interface and pinout specifications, LEDs are used to determine if the motion sensor works. If the LED lights up when the motion sensor are separated, then that means the motion sensor functions properly. By trial and error, the motion sensor ended up working as should.

We ran into the problem of the door sensors not reading the high inputs and low inputs correctly. After hours of trying to find a solution, we realized that the grounds of the bluetooth Arduino and the Pi need to be connected to the same grounds since it is relative.

One of the most vital part of the BerrySafe system is the communication between the Raspberry Pi and the Android phone. This was done so using socket programming. It's important to note that when sending a large request such as an image request, the threads and sockets will become unsynchronized. This means that certain delays are needed to be added so that it takes into account the time it takes to send an encoded image over the network. If not done so, some socket connections might be refused because the port might still be in use. We ran into this problem and thus added thread.sleep commands after sockets to avoid synchronization errors.

Sending an image to an Android phone was made a lot more difficult when we found out that Android studio was not able to recognize java.awt packages. This means that the BufferedImage class was unusable. The BufferedImage class is usually the default way of sending image over a socket but to avoid the problem, we encoded the image and then decoded it on the Android phone. This was done so by encoding it to a base64 string. The string can be very long depending on the image size thus sending the string over the socket took a while. Once the string was sent over, the Android phone would need to decode the string and return it back to the image so the user would be able to see.

For the webcam, the command to take a picture is done so on the command prompt window. The Java code for the camera portion essentially writes to the command prompt to take a picture and save it in a specified directory. In addition, to change the resolution of the image, it is just an extra argument before the image name. It can go up to 1920x1080 but we scaled it down so that it would be easier to send over the socket.

5 Experimental results

Tests

- Client/Server
    - We tested the client/server by making sure the android application connected to the raspberry pi and the raspberry pi sent an encoded image for the android application to decode. Since this is the bulk of the project, we made sure that the client/server worked properly to ensure that an image can be sent to the android application multiple times.
- Camera

    - We tested the camera by calling a function in the raspberry pi that takes a screenshot. We did this multiple times to ensure that a screenshot is taken on-demand based on a button pushed in the android device.
- Motion Sensor
    - We tested the motion sensor by determining if the LED lights lit up when the magnetic wave of the motion sensor is pulled away from each other. We did this by testing on a door. We then tested the raspberry pi to ensure that it read the data correctly.
- Arm/Disarm
    - We tested this functionality by testing various instances of arming and disarming in the android application. During this process, we also tested out the buzzer to determine if an alarm would sound if the motion sensor was triggered. If the application is armed, a buzzer should sound when the motion sensor is triggered, else do nothing when the application is disarmed.

Numeral Results

- Client/Server
    - During the initial test of the client/server the raspberry pi would send at most two images before going into an error. The following error consisted of not being able to read an input file. The android application would connect to the raspberry pi flawlessly though. After fixing the initial errors, the client/server works 100% of the time to ensure an image is properly requested and sent between the raspberry pi and the android application.
- Camera
    - The camera successfully captured an image and sent it to the android application on-demand 100% of the time.
- Motion Sensor
    - The motion sensor successfully detected movement when someone or an object passes through it 100% of the time.
- Arm/Disarm
    - The arm functionality successfully triggered the buzzer 100% of the time when the motion sensor detected movement. If the security is disarmed, the buzzer stops beeping as should 100% of the time.

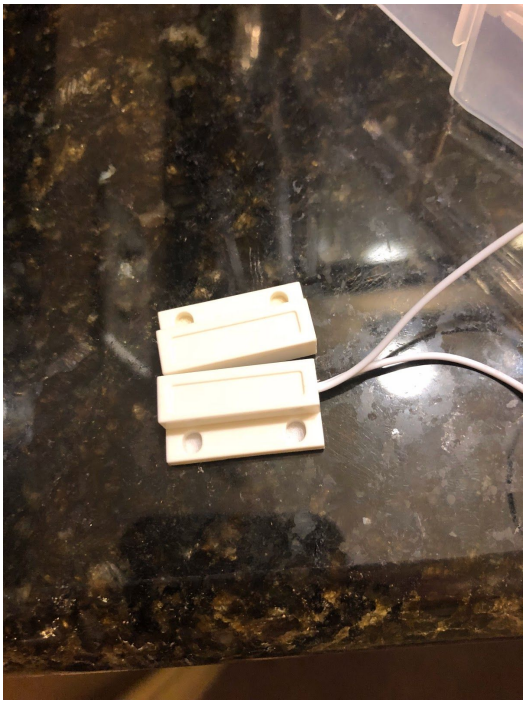
Analysis and Discussion

Based on these results, our finished product successfully and properly functions together to provide a security system that simplifies the average user's life. The client/server is the main bulk of the project because the android application and raspberry pi depended on the communication with each other to properly call for the other functionalities of the product. By making sure that the client/server properly worked, the product was able to arm/disarm, send an image from the camera, and work with the motion sensor to detect movement. During the whole process, we extensively tested each main functionality to determine whether or not they properly functioned accordingly to the desired result. Although it took many trial and errors, we were able to successfully put each function together to communicate with one another. Although it was suggested to use a sim card, we found that the sim card would not be using a local IP address, which is crucial for the socket communication between the raspberry pi and the android.

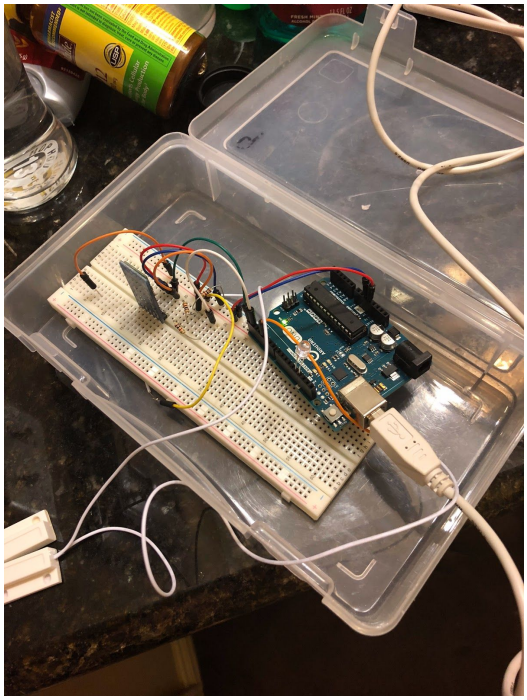
From studying further into other possible solutions, it was determined that the sim card would be infeasible in our product and decided to remove the sim card from our product.

Various Pictures of the Final Product

Motion Sensor

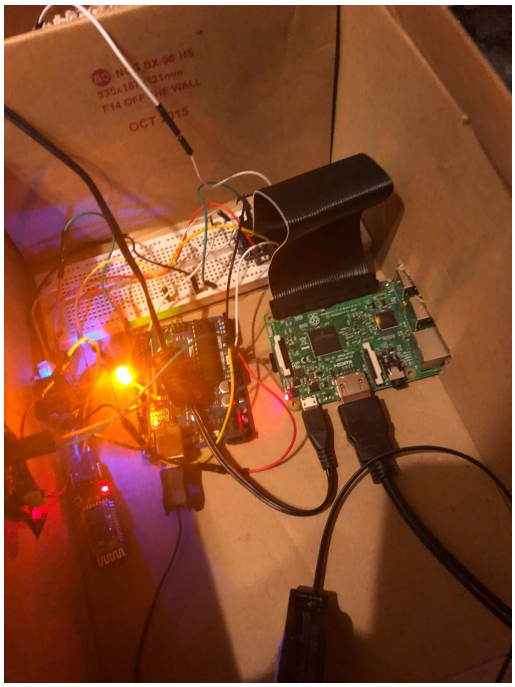


Bluetooth Door Sensor



Mounted Camera

Motion Sensor Detection



6 User's Manuals

User Hardware Installation: Begin by opening the package you receive in the mail with your custom number of sensors and cameras.(SENSORS) Next install and mount your door and window sensors by removing the adhesive paper tape cover off the sensor box. Place the box near the window or door you plan to install the sensor at. Remove adhesive tape from sensor back connected with wires to the box and place it on the top of the door frame. Mount the other half of the sensor on the door in alignment with the piece attached to the door frame. When the door is closed the two parts of the sensors should be touching. Use this process to mount all the door and window sensors. (CAMERA) Place the camera in a central location inside your home this will be the camera you can access in the Application when you click the view button. This will need to be close enough to your Central Hub box that a 10 meter usb cable will reach the camera. Once you've found your spot, mount the camera using the provided suction cup mount. Once you've mounted the camera place the 10 meter usb cord into any usb slot on your Central Hub's Raspberry Pi.(CENTRAL HUB) Place the box labeled central hub near any outlet in your house that is the most convenient for you. This is the device that will alert you of an intruder via WiFi/SMS. Plug in the power cord leaving this box into the outlet and that completes the hardware setup instructions.

User Software Installation: Begin by downloading the android application onto your android device. Next connect the application with the central hub to begin setting the desired security. At this point, the key functionalities of the home security should be connected with the central hub and should exchange data amongst each other. No other program is needed to operate BerrySafe. In addition to compile the code on the pi, use the command:

```
sudo javac -classpath .:classes:/opt/pi4j/lib/'*' -d . BerrySafe.java
```

And to run the program on the Pi use the command:

```
sudo java -classpath .:classes:/opt/pi4j/lib/'*' BerrySafe
```

The program can be found here:

https://github.com/pocramer14/BerrySecure.git

User Operation Instruction: The main functionalities available in the android application are arm, disarm, or view. When Arm is set, the buzzer will sound if motion is detected by the motion sensor. When Disarm

is set, no buzzer will sound even when motion is detected by the motion sensor. When clicking view, an image will be captured by the camera and display on the android application.

7 Budgets

In total, our project was relatively inexpensive compared to other existing products ranging past $200. The following table is the list of all the parts or support tools used to make the finished product a success.

Component	Individual Price ($)	Quantity	Individual Price * Quantity ($)
ELP fisheye camera	54.99	1	54.99
Camera Mount	10.89	1	10.89
USB Extension Cables	5.79	1	5.79
Raspberry Pi Motion Sensor	7.99	1	7.99
Window/Door Motion Sensor	5.92	1	5.92
Alarm Buzzer	4.99	1	4.99
Arduino Nano	8.29	2	16.58
Raspberry Pi	35.00	1	35.00
HC-05 Bluetooth Module	$9.99	2	19.98
Nova Global 3G/2G Cellular Modem	$49.99	1	49.99
Anker 20000mAh Portable Charger PowerCore 20100	$31.69	1	$31.69
Total Price			$243.81

Overall, the cost of each component was reasonable because it was the cheapest we can find. The components we bought also met the exact specification to properly implement each sub-module and configure them together as one whole unit.

Compared to the proposed budget, the final product costed significantly less than what we needed to demo. We believe that if we were able to properly set up and monitor one door, then we can replicate the same product for multiple entries.

If we were to mass produce the finished product, the price to make the whole product would be relatively cheaper because buying the parts in bulk would significantly reduce the overall price to make the product. Based on estimation, the price of the product would be approximately $120, which is less than buying the parts individually.