Monitoring Home Security Through License Plate Reading and Image Recognition on an Android Application Utilizing Wireless Sensor Networks*

Emily Lakic†
Thomas J. Watson School of Engineering & Computer Science
Binghamton University
New York, United States
elakic1@binghamton.edu

Troy Ballinger
Thomas J. Watson School of Engineering & Computer Science
Binghamton University
New York, United States
tballin1@binghamton.edu

Riddhi Zaveri
Thomas J. Watson School of Engineering & Computer Science
Binghamton University
New York, United States
rzaveri1@binghamton.edu

Tushar Sharma
Thomas J. Watson School of Engineering & Computer Science
Binghamton University
New York, United States
tsharma6@binghamton.edu

ABSTRACT

Computers have been integrated into many aspects of our everyday lives due to dropping costs and ever-expanding research. Home automation has become a convenience that homeowners enjoy and benefits society as a crucial tool in home security. We aim to design a low-cost, versatile automated garage door opener that capitalizes on both ease-of-use and home security. By using off-the-shelf components with no soldering involved, we demonstrate an easily implementable solution to garage automation, remotely controllable by any Android smartphone.

Our project design follows that of a simulation. Efficiency and effectiveness are ensured as a simulation supports repeated application and immediate, contextual feedback, let alone permits users to practice through trial and error and experience how the system will respond to their actions and decisions in a cost-efficient manner.

The Raspberry Pi, perhaps the flagship device of this new era, has erupted into a global phenomenon with over 12 million units sold - and for good reason, too. The device is a harbinger for prowess of the Internet of Things, from its effectiveness in use with a number of sensors, to its low-cost functionality serving the needs of users quite adequately.

Societal advancements in mobile applications feature a multitude of applications seeing success through usage, engagement, and retention. Unmistaken for their efficiency and reliability, such applications attribute their success to their ability to handle large amounts of data. The realm of technology in recent years has shifted quite prominently in the direction of storing data big or small on the cloud. When approaching the best means to building functionality by using the cloud, Mobile Backend as a Service (MBaaS) provides a medium that offers a linkage between the web and mobile applications to the backend cloud storage and backend APIs. MBaaS's features provide a safe and uniform transfer of big data between platforms while retaining proper functionality on all fronts.

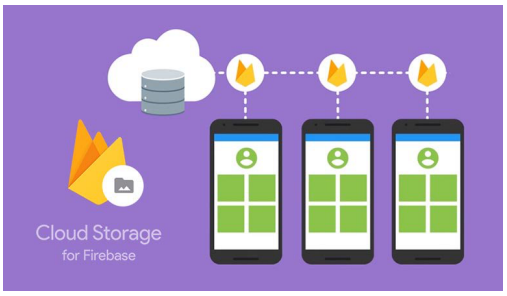

Figure 1: **Firebase Database by Google.**

Our License Plate Detection System utilizes Google's Firebase Database Console. Its realtime cloud-hosted database platform proved a strong match to our need for quick and reliable data transfers between the cloud and our android application. Data is synced across all clients in realtime, and remains available when our application goes offline. Such a system is beneficial as a user relying on an application for home security purposes will spend most of their time disconnected from the app while entrusting the app to provide a safety net in the meantime.

CCS CONCEPTS

• **Wireless Sensor Networks → Sensors & Modules;** *Servo Motor; PIR Motion Sensor; Raspberry Pi Camera v6 Module;*

KEYWORDS

License Plate Detection, Raspberry Pi, Android Application, Automatic Garage Door

1 Introduction

In the United States, there are roughly 2.5 million burglaries every year, with home break-ins honing in at 66%. With the influx of commercially-available home automation products, this number and associated percentage only stands to decrease in the coming decades. CCTVs and other home surveillance systems enable homeowners to view live footage of their house and who is at their front door, but most security implementations fall short of full automation. Our team recognized the gaps in the home security market and sought to build our own solution. Opening garage doors can be a hassle, with typing in security codes, a finite number of openers, and even getting off the couch to hit the button. A simple solution would be to have a video feed and a door opener controlled by an app, but video feeds unnecessarily chew up bandwidth. We sought to simplify this process, with a scanner that alerts you of the plate number, and car make, model, and color, approaching your home. Furthermore, our solution will open the door on your behalf just by pulling in front of it.

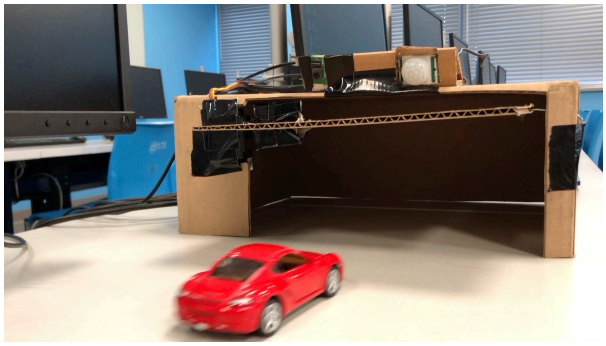

Figure 2: **Vehicle in Motion Entering the Garage Upon Sensor Recognizing License Plate Number.**

To go about providing an accurate and reliable simulation for our project, our team incorporated two modules and one sensor to aid in the system setup.

1. PIR Motion Sensor
2. Servo Motor Module
3. Raspberry Pi Camera v6 Module

Each sensor and module worked synonymously through system testing to communicate with the database and aid in accurate data transfer between devices. As a cost-efficient method given proper wiring to the Raspberry Pi, the three sensors/modules were a strong solution to the proposed simulation.

2 Design

2.1 The Server

We began by creating a server that could handle communications with a homeowners smartphone. This server is to be connected to a camera and the home Internet, to enable phone alerts even when nobody is in range of the setup. To save bandwidth, we decided to have the camera snap pictures only when movement is detected in front of the garage. So our system will be mounted directly above the garage, facing downwards towards where a car would pull in. This ensures minimal risk of tampering, by having the sensor in a hard-to-reach area, and minimizes the risk of weather impacting the camera's view.

2.2 License Plate Recognition

To save the homeowner from false positives bombarding their phone with alerts, as well as to make data more readily available, we analyze each image captured by the camera using an open-source license plate recognition software, called OpenALPR. OpenALPR scans the image and pulls out candidate cars that could be potential matches, as well as possible plate numbers and the confidence associated with each car property. If the car's plate matched a recognized car, one the homeowner has set to be automatically accepted, the garage will be opened immediately. If the car is unrecognized, it will ping the homeowner's smartphone with information about the car, allowing them to open or keep the door closed remotely. Our algorithm looked like this:

```
OnMotionDetection
image = camera.snap()
car_candidates = openALPR.processImage(image)
possible_cars = []
// Pull any possible car with properties higher than a threshold
forEach car in car_candidates:
        if car.confidence > 75%
                possible_cars.push(car)

// Check database for match
forEach car in possible_cars
        if database.contains(car)
                openDoor()
                return
// Upload unknown car to database LoadImage
database.push(possible_cars[0])
storage.upload(image)
return
```

For the purposes of our demonstration, we set the threshold to 75% confidence and utilized OpenALPR's Cloud API to process all images, because in our implementation we are not using real cars or garages.

2.3 Android Application

Our License Plate Detection mobile application centers entirely around the user and that user's safety, so much so that a

strong Android platform was necessary to provide a potent home security system both reliable and simple enough for any individual to operate on their own handheld device.

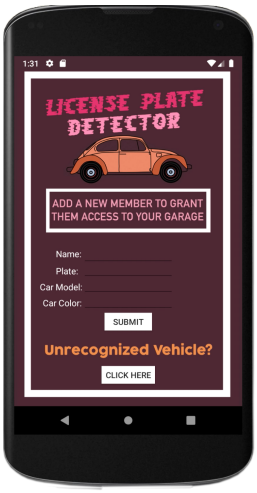

Figure 3: **License Plate Detector Application Main Screen.**

The sleek platform features several screens. Upon opening the application, users are met with four fields to fill in pertaining to a vehicle's information, as shown in Figure 3. Data entered into these four fields is sent through the application to the Firebase database by pressing the 'Submit' button. Users have the ability to navigate to the second screen in the case of an unrecognized vehicle at the garage by pressing the 'Click Here' button.

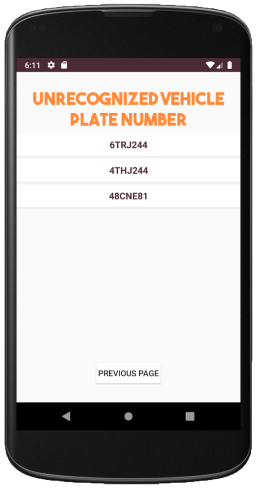

Figure 4: **License Plate Detector Application Second Screen.**

The second screen of our android application shows an array list of each unrecognized vehicle's plate number that has approached the garage, as shown in Figure 4. A 'Previous Page' button was implemented to allow users to navigate back to the main screen. Otherwise, users have the ability to choose any vehicle from the list and view its associated plate number, car name, car model, and car color, as shown in Figure 5. An image is displayed as well upon the license plate being sensed by the motion sensor and the pi camera taking a photo. Users have the ability to reject or accept the car, or may navigate back to the list of all unrecognized vehicles, as shown as in Figure 4.

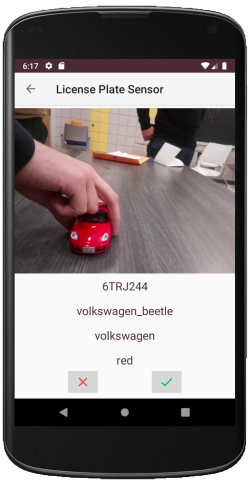

Figure 5: **License Plate Detector Application Third Screen.**

3 Implementation

3.1 Raspberry Pi

We implemented our design with a Raspberry Pi 3 Model B and wired together the components using a solderless breadboard, as shown in Figure 6. Schematics for each sensor and module were studied prior to ensure proper connections and dismiss the potential of shorting the 5V and 3V3 pins on the Raspberry Pi. Emily implemented a breadboard as well as our jumper wires required electrical connections in multiple ports used by a range of sensors and modules. We constructed a model garage for demonstration purposes, as shown in Figures 7 & 8. Alongside so, a small, diecast red Porsche contributed to system testing. For detection motion, we used a PIR sensor facing down from the top of the garage. Next to it, we installed our camera, also attached to the Pi. Finally, a small servo motor was built into the garage that allows the cardboard garage door to open and close at a rotational movement of 90 degrees.

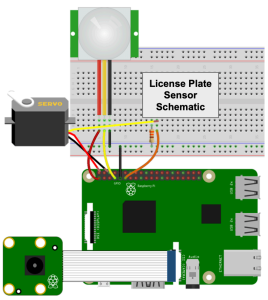

Figure 6: **License Plate Sensor Schematic & Circuitry**

Figure 7: **Garage Opening for Recognized Vehicle.**

Figure 8: **Servo Motor, Pi Camera, and PIR Motion Sensor.**

We ran a Node.js (v8.0.0) server on the Pi and awaited input from the PIR sensor, letting its subsequent trigger function run asynchronously. Because of the difficulties presented with connecting to our server's API when our demo project was switching networks and locations rapidly, we opted for a cloud-based solution for storage of images and databases.

3.2 Google Firebase Database

We utilized Google's Firebase, specifically its bucket storage and real-time database. This allowed reliable, efficient communication between the Pi and smartphone application from anywhere with an Internet connection. Furthermore, the Firebase API was compatible with Android/Android Studio. The Firebase Database is comprised of two nodes, 'LoadImage' and 'Member'. Each entry of the LoadImage node has five children: URL, color, make, model, and plate. This node associates with the data obtained by the server, sent to the database, and received by the Android device. Each entry of the Member node has four children: color, model, name, and plate. This node associates with the data sent by the Android device to the database, for use by the server when determining whether a car is recognized or not.

3.3 Android Application

The Android Studio application was responsible for both sending data about a known vehicle to the database, as well as receiving data from the database about an unrecognized vehicle, initially from the Node.js server.

3.3.1 Sending Data to Firebase Database

Upon opening the License Plate Detector application, the user is prompted to enter their information. The user may wish to grant another individual access to the garage as well, prompting said user to submit a multitude of entries to the application. This information includes an individual's name, plate number, car model, and car color. A submit button pushes each value entered into the five fields to the database. Within the Java file of the main screen, there is one type of object, *DatabaseReference*, and two instances of the object, with their respective references, *reff* and *reff2*. By getting a reference of an instance of the database for the Member node, the information entered into the Android application appears immediately in the realtime database, as shown in Figure 9.

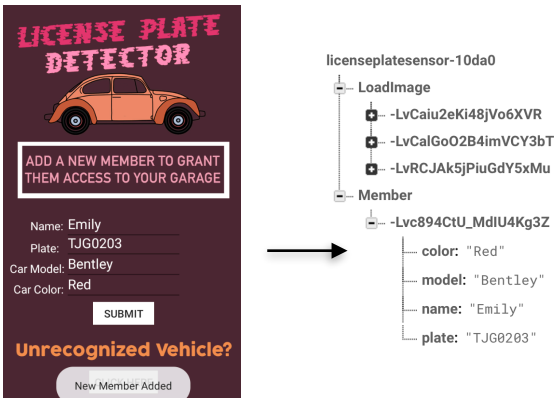

Figure 9: **New Member Information on App and Database.**

3.3.2 Receiving Data from Firebase Database

A car is let into the garage if and only if the garage door opens by movement of the servo motor. Whether the servo motor decides to open is entirely controlled by the user's decision when an unrecognized car approaches the motion sensor (given that any recognized car in the database system will automatically be allowed in). Thus, data received by the Android device from the database governs the users decision. In the case that a user decides an unrecognized car is allowed into the garage, the user will click the check mark button, and the database entry containing information about that specific car is deleted from LoadImage and moved into Member as it is now a recognized and known car that the user has granted permission to enter the garage. Below is Java code exhibiting the events occurring when the user accepts the unknown vehicle.

```
reff=FirebaseDatabase.getInstance().getReference().child("Member");
image = findViewById(R.id.image);
plate_number = findViewById(R.id.plate_number);
color = findViewById(R.id.color);
```

```
make = findViewById(R.id.make);
model = findViewById(R.id.model);
yes_button = findViewById(R.id.yes_button);
no_button = findViewById(R.id.no_button);

member = new PlateNumberModel();
yes_button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //member.setName("Emily");
member.setPlate(Objects.requireNonNull(getIntent().getExtras()).getString("plates"));
 member.setModel(getIntent().getExtras().getString("model"));
 member.setColor(getIntent().getExtras().getString("color"));
        reff.push().setValue(member);
                        DatabaseReference reff1 = FirebaseDatabase.getInstance().getReference();
                        Query carsQuery = reff1.child("LoadImage").orderByChild("plate").equalTo(Objects.requireNonNull(getIntent().getExtras()).getString("plates"));
                carsQuery.addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
                        public void onDataChange(DataSnapshot dataSnapshot) {
                            for (DataSnapshot appleSnapshot: dataSnapshot.getChildren()) {
                appleSnapshot.getRef().removeValue();
                }
            }
                });
                Toast.makeText(PermissionScreen.this, "Finish", Toast.LENGTH_LONG).show();
        }
    });
```

More likely than not, however, a user will not recognize a license plate number. In this case, the user will see an array list of unrecognized vehicles to select from. Below is Java code exhibiting the events occurring when the user rejects the unknown vehicle.

```
no_button.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        Toast.makeText(PermissionScreen.this, "Vehicle Denied Entry",
        Toast.LENGTH_LONG).show();
        Intent intent = new Intent(PermissionScreen.this, LoadImage.class);
        startActivity(intent);
    }
});

plate_number.setText(Objects.requireNonNull(getIntent().getExtras()).getString("plates"));
color.setText(getIntent().getExtras().getString("color"));
make.setText(getIntent().getExtras().getString("make"));
model.setText(getIntent().getExtras().getString("model"));
Glide.with(this).load(getIntent().getExtras().getString("URL")).into(image);
```

As shown in Figure 10, each field: the plate number, car color, car model, car make, and the image of the car, is received by the database and shown to the user to aid in their decision. In the case that an unknown vehicle is not accepted, the user is informed that the vehicle was denied entry, and is returned to the previous screen to select another unknown license plate in the case that there is more than one.

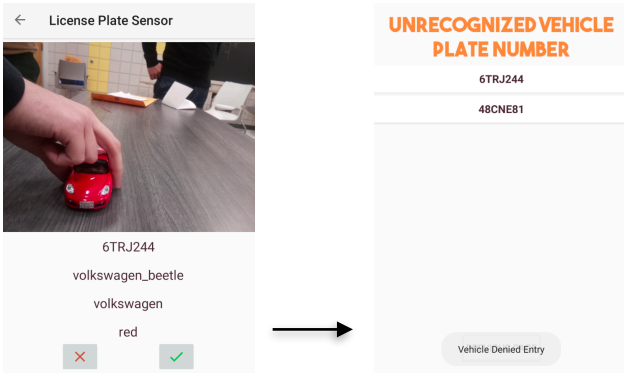

Figure 10: **User Rejects Unrecognized Vehicle.**

4 Evaluation

Our implementation was able to effectively deny entry to any vehicle not approved via the app or pre-approved by the database. Our algorithms ran efficiently -- most of the delay between motion and door access involved the seconds taken by the Raspberry Pi camera software, Raspistill, focusing and adjusting exposure before taking images. By using smaller image sizes, we were able to rapidly send image data to OpenALPR's Cloud and get results back, as shown in Figure 11.

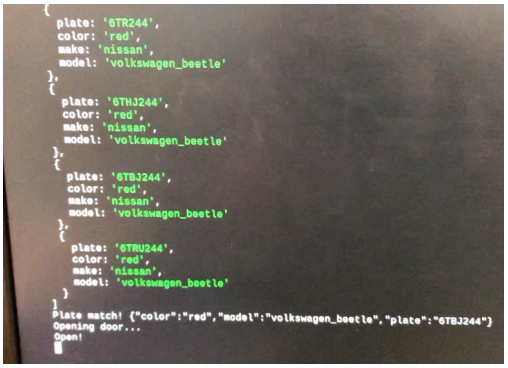

Figure 11: **New Member Information on App and Database.**

We experienced no false positives during testing, however they could be introduced with similar plate numbers if our threshold

remained as low as 75%. We believe that if the system were to be tested on a real car, the confidence threshold could be increased to 95%+, as the image recognition software is trained using real cars. The Android application was able to receive and accept unrecognized car information within seconds of the Pi uploading them, even on different wireless networks. Similarly, the Pi opened the door almost immediately upon being triggered by the application's acceptance of an unrecognized car. For these reasons, we believe the system was largely successful, with some room for improvement by tweaking its configuration.

5 Conclusion

While working on the project, we ceded it was challenging to host a server with changing locations and wireless networks as the project moved between houses and school. It became too complicated to keep the Pi functioning as a dedicated server, especially on Binghamton University wifi, which we do not manage. For these reasons, we switched to Firebase to host our database and images. Admittedly, the server aspect of the Pi became obsolete at this point, but was kept in for legacy and testing purposes. If we were to recreate the project with the knowledge learned, we would write our algorithms exclusively in Python. Utilizing OpenALPR's Cloud API was efficient for testing, but in production it would be necessary to install it locally, given its data limits and bandwidth usage. In terms of Google's Firebase Database Console and Android Studio, the two worked synonymously to deliver an accurate system of both sending and receiving data to advance home security through application and cloud implementation.

ACKNOWLEDGMENTS

This work was supported in part by the Thomas J. Watson School of Engineering at Binghamton University. The Android Device was lent to the team by Professor Mo Sha of Binghamton University's Computer Science Department. Sensors along with corresponding materials used for the development of the project were purchased through Amazon.

REFERENCES

[1] Sara Santos et al. 2019. Car Plate Recognition System with Raspberry Pi and Node-RED. (April 2019). Retrieved December 8, 2019 from https://randomnerdtutorials.com/car-plate-recognition-system-with-raspberry-pi-and-node-red/

[2] Simon Monk. Adafruit's Raspberry Pi Lesson 8. Using a Servo Motor. Retrieved December 8, 2019 from https://learn.adafruit.com/adafruits-raspberry-pi-lesson-8-using-a-servo-motor/overview

[3] The Pi Hut. 2017. Raspberry Pi GPIO Sensing: Motion Detection. (November 2017). Retrieved December 8, 2019 from https://thepihut.com/blogs/raspberry-pi-tutorials/raspberry-pi-gpio-sensing-motion-detection

[4] Pi Hut. 2014. How to install/use the Raspberry Pi Camera. (November 2014). Retrieved December 8, 2019 from https://thepihut.com/blogs/raspberry-pi-tutorials/16021420-how-to-install-use-the-raspberry-pi-camera