2016

Remotely Monitor and Manage a Garage with IoT

Brandon Ridge
Grand Valley State University

Follow this and additional works at: http://scholarworks.gvsu.edu/cistechlib

Recommended Citation
http://scholarworks.gvsu.edu/cistechlib/236

This Project is brought to you for free and open access by the School of Computing and Information Systems at ScholarWorks@GVSU. It has been accepted for inclusion in Technical Library by an authorized administrator of ScholarWorks@GVSU. For more information, please contact scholarworks@gvsu.edu.
Remotely Monitor and Manage a Garage with IoT

By
Brandon Ridge
April, 2016
Remotely Monitor and Manage a Garage with IoT

By
Brandon Ridge

A project submitted in partial fulfillment of the requirements for the degree of
Master of Science in
Computer Information Systems

at
Grand Valley State University

April, 2016
Table of Contents

Abstract .................................................................................................................................................. 2
Introduction ........................................................................................................................................... 2
Background and Related Work ............................................................................................................. 2
Program Requirements ......................................................................................................................... 3
Implementation .................................................................................................................................... 3
Results, Evaluation, and Reflection ...................................................................................................... 11
Conclusions and Future Work ............................................................................................................. 13
Bibliography ......................................................................................................................................... 15
Abstract
With the explosion in the Internet of Things (IoT) more and more people are trying to integrate various pieces and parts of their lives with the internet. This project builds on this and attempts to create a fully integrated garage security system. The Garage Eye gives the ability to see the status of a garage from any device connected to the internet. It gives the ability to remotely manage (open/close) the garage as well as see the current garage state. A camera from within the garage also gives the ability to see in the garage and receive notifications of detected movement while the garage is closed.

Introduction
Knowing the status of your home when you're away is a very important thing. Before the age of the internet this was very difficult; however, in today's age it is easier than ever to connect pieces and parts of your home to the internet so that they can be monitored from afar. Everything from complex, professionally installed systems, to simple, do it yourself (DIY) projects exists to achieve this goal. All of these systems aim to provide information about the status of your home via the internet so that when the home isn't occupied, important information can still be monitored and reported.

The Garage Eye focuses on one specific part of the house, the garage. The garage is the main entry in and out of most people's home so monitoring this while the house is vacant can provide some peace of mind.

Is the garage closed? Did someone break into the garage? Need to be able to let someone into the garage while the house is empty? The Garage Eye is a multi-part system that aims to solve these problems by allowing the user to get information about the garage via the internet while they are not home.

Background and Related Work
The IoT craze has released an explosion of do it yourself home automation projects. There currently are many tutorials online of people making internet connected garage door
openers, security cameras and even status indicators. However, there are none that exist to combine all three systems into one cohesive and easy to use system.

**Program Requirements**

The user interface consists of the following:

- Secured system via a login.
- Viewer window to see the garage via the camera.
- Remotely open/close the garage.
- Show the status of the garage (open/closed and secured/unsecured).
- Easily viewable log of the most recent actions recorded by the system.
- Historically see photo/video from logged actions.
- Ability to notify the user of changes to their system.

The hardware interface consists of the following:

- Self-contained device housing all components.
- Low cost and easily accessible parts.
- Internet connected.
- Low power usage.

**Implementation**

The application was developed in Python[1]. Flask[2], a micro web development framework for Python was used to develop the front end GUI (Graphical User Interface). The front end GUI uses the Flask microframework as well as the Foundation Framework[3] to make the entire interface styled and responsive. Pushbullet[4] is used to send notifications from the system to the user’s devices.

All of the software interacts to form a user interface that is accessible from any internet connected device.

**Software Implementation**

**Python**

Python is an open-source, cross-platform that is widely used and can serve many functions. Python can be written in object-oriented, imperative, functional, or procedural styles. Due
to Python’s popularity, a vast amount of third party library support available for integrating with other software packages and hardware devices.

**SQLite**

SQLite[5] is a relational database management system (RDBMS), similar to MySQL or Oracle. The major difference in SQLite is that it’s embedded into the program itself instead of relying on an actual server to connect. SQLite is known as a zero configuration because there is no server to setup or connect to and no daemon process that must be running in the background. The entire SQLite database is contained inside of one *.db file, which includes the definitions, tables, indexes and the data itself. These characteristics make it ideal for use with IoT projects.

**SQLAlchemy**

SQLAlchemy[6] is an object-relational mapper (ORM) and a SQL toolkit for the Python programming language. The ORM allows a link to be made between the SQLite database and an object class in Python.

**Flask**

Flask is a web development microframework for Python. It is considered a micro framework because it doesn’t have the bloat of other popular Python frameworks such as Django. Instead it allows the user to install specific plugins to meet their needs for the given application. Flask-Login, Flask-Principal, and Flask-SQLAlchemy are examples of these plugins that were used in the development of this application. Flask-Login provides session management and user authentication, while Flask-Principal provides user roles. Together these things allow for user creation, authentication and permission management. Flask-SQLAlchemy adds support to flask to connect to SQLAlchemy.

**Foundation**

Foundation is a front end responsive framework that uses CSS, HTML and JS. Using Foundation allows for a fully responsive user interface that works great on anything from desktops to mobile phones. It also adds consistent styling to all the UI elements across the front end.
**Pushbullet**

Pushbullet is an application that can be installed on any mobile device or as a browser extension onto a computer. Pushbullet’s goal is to bridge the gap between your devices and make them feel like one by synchronizing notifications and allowing links, files, pictures, etc. to be transferred between them. Pushbullet provides an API that allows notifications to be created and pushed to one or all of the user’s devices. The API was used to send notifications of motion and garage intrusions to the user’s phone.

**Hardware Implementation**

All of the hardware interacts together through the Raspberry Pi computer and the Python application running on it. Figure 1 shows how the hardware interacts with the Raspberry Pi and the garage.

![Figure 1: Diagram showing how hardware interacts](image)

**Raspberry Pi 3**

![Figure 2: Raspberry Pi Computer](image)

Raspberry Pi 3[7] is the most recent release of the credit-card sized computer from the Raspberry Pi Foundation. The Raspberry Pi is a low cost, low powered computer that is popular with DIY and IoT projects. Costing only $35 and including the necessary
components such as wireless, Bluetooth and GPIO pins made it a great candidate for the base of the project.

**Raspberry Pi NoIR Camera**

![Raspberry Pi NoIR Camera Module](image)

Raspberry Pi NoIR camera is a webcam that integrates directly with the Raspberry Pi itself. The camera has software built in to the default Raspberry Pi OS (Raspbian) that allows easy interfacing. The NoIR version of the camera was chosen specifically to help with low light conditions and provides the ability to add Inferred lights in the future to see objects in the dark.

**PIR Motion Sensor**

![PIR Motion Sensor](image)

Passive Infrared (PIR) motion sensor measures infrared radiation and detects changes in the levels. This makes it ideal for detecting the presence of humans because they emit heat and when they pass by a PIR sensor it can detect the difference between them and the surrounding area. The PIR is connected to one of the GPIO pins on the Raspberry Pi. When motion is detected, the PIR sensor flips to a high state and the program, which is monitoring the pin, is notified and can execute a series of commands.
Magnetic Reed Switch

Figure 5: Magnetic Reed Switch

Magnetic reed switch is a simple switch that is grounded when closed (connected) and the circuit is broken when the magnet half is removed. These switches are commonly used in home security on doors and windows to signal when it is open or closed. One half of the switch is attached to the garage door and the other half is attached to the inside of the garage. When the garage door is closed the two halves are close enough together to close the switch, but when the garage door is opened the switch is opened signaling to the Raspberry Pi that the door is open.

SPST Relay

Figure 6: SPST Relay

The single pole single throw (SPST) relay is used to physically open the garage door. Tapping into the same two wires used for the physical button, normally placed next to the door by the entrance house, and connecting them to the relay you can toggle the garage door. A signal is sent from one of the Raspberry Pi’s GPIO pins to quickly switch the relay on and off. When the relay is switched on and off it mimics the pressing of the button on the wall.

Hardware Summary

All of the hardware connects to a custom built HAT (Hardware Attached on Top). This HAT, see figure 7, connects to the Raspberry Pi GPIO (General Purpose IO) pins. The software interfaces with the GPIO pins to determine if the garage is open or closed, to detect motion, and to open/close the garage door remotely.
The hardware is contained within a small case and attached to the bottom of the garage door opener, see figure 8. This location provides easy access to power, the wires needed to open/close the garage, and the magnetic reed switch mounted on the garage door itself. From this location there is also a good view of the entrance points of the garage so that the camera can record any activity of people in the garage.
User Interface

**Dashboard**

The dashboard is the landing page once the user is logged into the system. From here the user can see all the important information the system provides. Figure 9 shows the layout of the dashboard. On the left it shows the status of the garage as well as the ability to open/close it remotely. A list of devices is shown beneath the button. The devices are used to secure the garage when the house is vacant and sets if the users are notified of changes in the garage. Finally, below that is a list of the most recent status updates logged in the system. On the right hand side there is a photo of the garage with a timestamp of when it was last updated.

![Garage Eye dashboard](image)

Figure 9: Garage Eye dashboard

In figure 10 the dashboard is shown again, only from a mobile device. This shows the responsive design made possible by the Foundation framework. The dashboard shows all the same information as the full size dashboard, the sections are simply rearranged and resized for better functionality on a smaller screen.
Log

The ability to see a log of all the activities recorded by the system and easily download the photos and videos taken is an important part of this project. The logs page, shown in figure 11, displays all the logs, most recent entry first, along with links to download the picture and video. The pictures/videos can only be downloaded if the activity recorded was ‘motion detected’ or ‘door opened’. 
User Admin

Admin users have the ability to add/remove other users access to the system, see figure 12. By default, these users have access to the dashboard and the log screens. If a Bluetooth mac address is added to the user, it will associate that device with the user and automatically use it as one of the devices that are monitored to secure the garage when the house is vacant.

Results, Evaluation, and Reflection

Overall the project met the objectives. The Garage Eye is a small, self-contained hardware device that allows the user to interface with it from any internet connected device. From the interface the user can see the status of the garage, remotely manage it and see a log of the most recent activity recorded by the system.
When I started this project I was planning on two separate devices. One for the interfacing with the garage door and the other for the motion sensing and camera. As I got further along I realized that having two separate units in the garage had a few problems that could be avoided if there was only one unit. Finding a place to mount both devices where they would have access to power and within a range to complete their tasks lead to both of the devices being mounted on the garage door opener. Also coordinating the communication between the devices add unnecessary overhead if they were going to be mounted so close together.

A few months into the project the Raspberry Pi Foundation released the Raspberry Pi 3. After its release the specifications for it would be the perfect sole device to use. The new Raspberry Pi 3 added some very important pieces of hardware that made it perfect for the Garage Eye. Integrated Wireless and Bluetooth provided the ability to connect to the internet and search for Bluetooth devices without having to buy two separate components that would have to connect via usb. The performance was also increased by almost 10x making it great to record and process 1080p video from the security camera while still being able to send notifications and serve up web pages.

Both the motion sensor and the magnetic door sensor occasionally trigger a false positive. The PIR motion sensor has a physical adjustment knob to adjust the sensitivity of the motion sensor. However, even with the sensitivity at the lowest setting, it still would trigger some false positives, as in sending an alert that there was motion in the garage when there was indeed no motion in the garage.

While checking for Bluetooth devices seems to be working decent there are many times that while the device is still in the house, just a bit out of range that the Garage Eye enables the security. To help alleviate false positives in the system when it checks (every five minutes) for any devices if it finds none in one five-minute increment instead of turning on the security right away, it waits another cycle and checks again to see if the devices are still all out of range. This helps to eliminate the quick, device out of range problem, but will still securing the garage within 10 minutes of leaving the house. Using the same concept of pinging device’s Bluetooth MAC address only using wireless should be able to give much more accurate results since wireless has a much longer range than Bluetooth does.
Conclusions and Future Work

Even though the system meets all the original objectives there are still many areas for improvement. Some of the possible improvements to the system are outlined below.

The motion sensor is placed in the center of the garage facing the floor which makes it hard to sense motion in all areas of the garage. Also due to the size of the garage it’s placed in, 28’x28’, a PIR motion sensor would never be able to reach all areas of the garage anyway. I think a better alternative would be to add a magnetic reed sensor on the side entry door, the same one that is on the garage door itself would be a better permanent solution. Both the side door and the garage door are the only two external entrances to the garage. If they were both set up with a magnetic reed sensor the user would always know when someone entered the garage.

Currently the garage secures itself and un-secures itself based on the presence of any Bluetooth devices registered through the admin. When the garage is ‘secured’ it sends the notifications to a cell phone via Pushbullet, however when a user just gets home the garage is still secured and when the garage is opened it automatically sends a notification. It would be great to have the program check immediately after the door was opened to see if any of the devices are in rage before sending a notification. In other words, if it was an authorized person that is entering the house, there is no need to send a notification.

Being able to set a schedule and/or override the security status would be a nice feature to add. Even if the user(s) are at home, there might be times that they would want to receive notifications of a breach in the garage. An example of this would be when you are sleeping. Most home security systems are armed at night so if there is a break in it will alert the homeowners.

Adding an audible alarm to the garage that would sound when an intruder enters the secured garage would make it function much more like an actual home alarm system and act as a deterrent if someone was attempting to break in.
Using Pushbullet as the channel for communication between the Garage Eye and the admin’s devices is a practical and easy way to communicate changes in the garage to the internet connected devices. However, it would be nice to add the ability to add more than one person’s device(s) to receive the alerts. Say, for example, if there are multiple people in the home that need to receive alerts or if the user is on vacation and would like to be able to send the alerts to a friend or family member that are close by. Pushbullet allows you to add multiple user accounts and/or email addresses to send the notifications, but the Garage Eye admin would have to be altered to allow for these contacts to be saved.
Bibliography

   https://www.python.org/
   http://flask.pocoo.org/
3. “Foundation | The most advanced responsive front-end framework in the world.” 
4. “Pushbullet - Your devices working better together.” Pushbullet. Accessed April 3, 
   2016. https://docs.pushbullet.com/
   http://www.sqlalchemy.org/
   https://learn.adafruit.com/
   https://learn.sparkfun.com/