Spring 2019

# Smart Garage Opener

Jacob Wasson
jtw59@zips.uakron.edu

# Smart Garage Opener

## Senior Project Final Report

DT05

Andrey Kadoutchek

Jacob Wasson

Teddy Helton

Dr. Lee

April 28 2019

## Table of Contents

## List of Figures

## List of Tables

**Table No.** **Page No.**

## Abstract

This design report describes a security device that can work with residential garage door systems to ensure safer at home package delivery. Research has shown that package theft is a growing problem directly related to the growth of the e-commerce market. By utilizing a barcode scanner, a system can be built that opens and closes a garage door when an expected package is scanned at the time of delivery. The system will be convenient, safe, and secure.

## 1. Problem Statement

### Project Need (AK)

Consumers are online shopping now more than ever. However, many Americans are coming home with their packages nowhere to be found. Package theft has become a big problem for online shoppers as well as retailers. More than half of Americans say they know someone who's had a package stolen from outside their home, and 30% say they've experienced package theft themselves [1]. Currently, there are only a few workarounds to this problem. For example, Amazon offers lock boxes where users can go to pick up their orders. Also, Shipping services like UPS and FedEx have programs that allow customers to reroute packages to their offices or other safe locations for pickup. These methods help reduce the risk of letting expensive goods sit out in the open for anyone to steal but they defeat the purpose of home delivery and convenience.

## Objective (AK)

The goal of Smart Garage Opener is to allow convenient, safe, and secure delivery of online orders and eliminate the chances of them being stolen. The owner will need to set up the Smart Garage Opener on the outside of their garage below their garage door opener. Once it is secured and plugged in the Smart Garage Opener will guarantee protection and delivery of the package. Smart Garage Opener will only unlock and open a garage door with the scan of an acceptable tracking number barcode. If Smart Garage Opener is unlocked with the scan of an accepted barcode it will keep the garage door open long enough for a package to be slid into the garage and then automatically close as well as notifying the owner via SMS. The Smart Garage Opener system will be quick and easy to use which is important for delivery persons who work very quickly.

## Research Survey (AK, JW, TH)

Barcode scanners have revolutionized the supermarket and retail store checkout process along with their inventory control. The pattern of black lines seen on virtually all products today is the UPC code (Universal Product Code). The UPC is just one type of barcode. There are actually many other barcodes besides the UPC that are used for other diverse applications. Package routing and tracking is one of those applications. These barcodes have been carefully designed to be easily decoded when scanned in either direction, at any arbitrary angle, and with variable speed [2].

The basic architecture of barcode scanners tends to be very similar. The basic principle is to use a collimated laser beam, rotating multi faceted mirror, several stationary mirrors, and other optics to generate a scan pattern above or beside the scanner that will intercept the barcode printed on the item to be scanned. While the scan may appear to consist of multiple lines or a continuous pattern, it is in reality a single rapidly moving spot. Currently, the electro-mechanical laser scanner is still the most common. Some of the newest barcode technology does away with the laser scanner altogether and uses a 2-D video-camera (CMOS or CCD)-based imaging system and high-speed DSP (Digital Signal Processor), instead. This technology eliminates most of the complex and costly optical and mechanical components making for a compact robust system [2] This is an alternative barcode scanning system the design group is considering.

There is no risk to the user in proximity to a barcode scanner. The laser beam is moving rapidly and is low power. A rough estimate of the maximum possible eye exposure to a properly functioning scanner is about 10 microwatts or less. The only possible risk would be if the scanner motor failed for some reason, and the laser beam was stationary. However, most if not all scanners have a safety device to shut off the laser, should the return beam not behave properly [2].

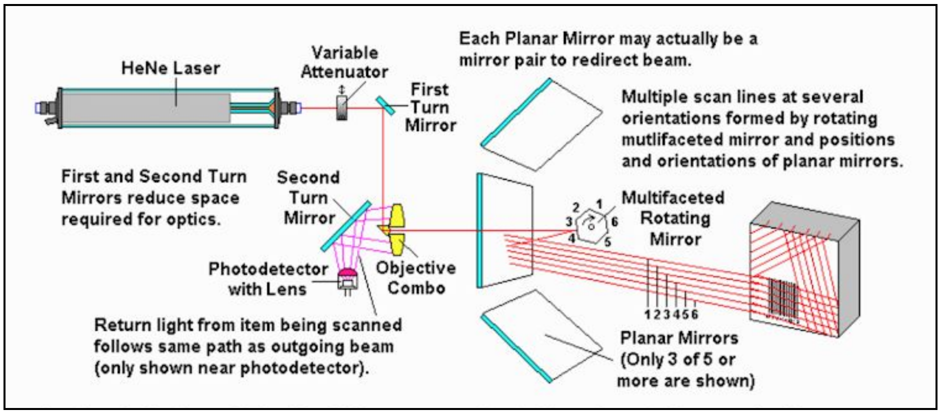

HeNe Laser
Variable Attenuator
First Turn Mirror
Each Planar Mirror may actually be a mirror pair to redirect beam.
Multiple scan lines at several orientations formed by rotating mutlifaceted mirror and positions and orientations of planar mirrors.
First and Second Turn Mirrors reduce space required for optics.
Second Turn Mirror
2 1
3 6
4 5
Multifaceted Rotating Mirror
Photodetector with Lens
Objective Combo
123456
Return light from item being scanned follows same path as outgoing beam (only shown near photodetector).
Planar Mirrors (Only 3 of 5 or more are shown)

*Figure 1: Diagram that shows the optical path of a typical barcode-scanner*

Alerting the owner of package will be necessary if the garage door is not properly closed after the delivery person once has delivered a package. Leaving the garage door open creates the possibility of the package or other materials being stolen from inside the garage. Mobile phone text messaging, also known as the short message service (SMS), provides an asynchronous means of communication [4]. By using the SMS messaging the owner of the package can be alerted when the package has been delivered safely or unsafely, depending on whether the garage door has been closed properly at the end of the delivery. Doing this can be useful in preventing theft and will help minimize the number of stolen packages. If the garage is not shut after the package has been slid into the garage, the owner will know almost immediately which will give the owner time to take appropriate actions to prevent theft of the package.

The Smart Garage Opener locking system will unlock when the barcode of an expected package is scanned, but if multiple packages are expected, Smart Garage Opener will be able to identify multiple barcodes by accessing a database of user-expected tracking numbers. Similar systems are already in place throughout the world in the form of electronic card scanning door locks. These locks scan an employee's key card and record the entry and exit of each individual as they enter or leave a building [6]. This same ideology could apply and be implemented for packages entering or leaving a garage. To open the garage door for package delivery, the scanner should accept multiple input barcodes as long as they are stored the list of acceptable entries.

Most garage door systems rely on a radio signal from a remote control near the home to be opened or closed.  Garage opener remotes are simple transmitters that send a signal to the receiver which controls the garage motor that lifts the door.  Some of the newer garage door systems on the market are able to connect to wifi and can be controlled through an app. With the app, users are able to open and close their garage from anywhere and at anytime. These Wi-Fi enabled garage systems are also able to connect to Amazon Alexa or Google Home devices and can be controlled with voice commands. An example of one of these newer garage door systems is the NEXX Garage NXG-100 NXG [9]. Currently there are no garage door systems that can be automatically controlled with a barcode scanner to be opened and closed for package delivery.

## Marketing Requirements (AK, JW, TH)

1. The product will provide a secure method of package delivery and safekeeping.
2. The product will notify the owner when scans/deliveries are made.
3. The product will work with any garage door system.
4. The product will have a indicator light that will show if barcode is accepted or not.
5. The product will be user programmable.
6. The product will operate automatically.
7. The product will accept packages from multiple delivery services.

## Objective Tree (JW)

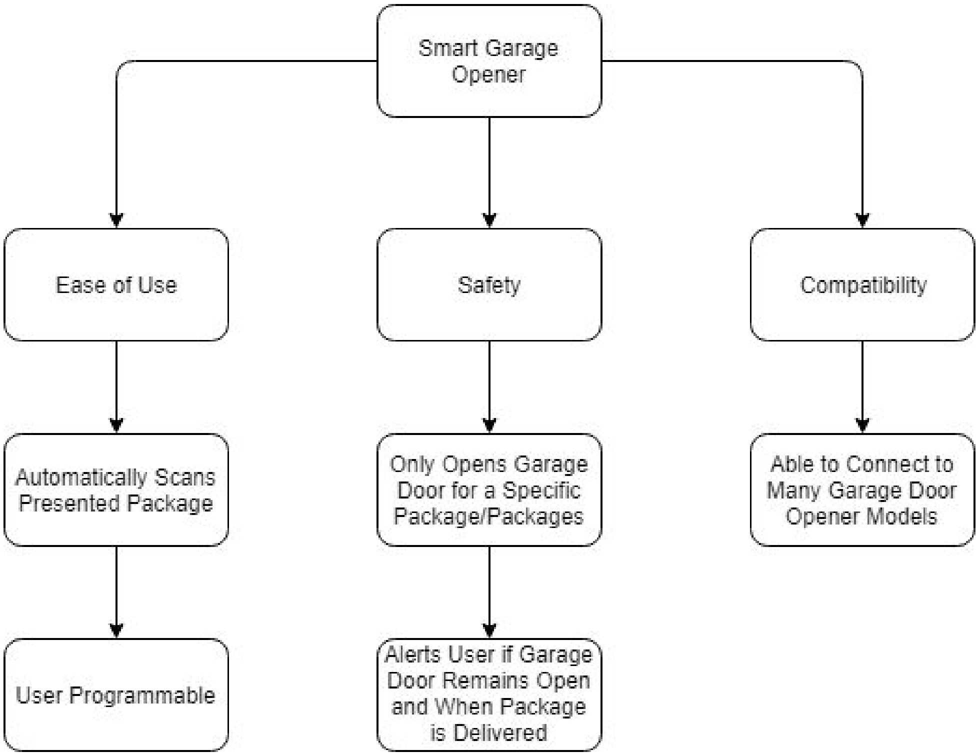

Smart Garage Opener
Ease of Use
Safety
Compatibility
Automatically Scans Presented Package
Only Opens Garage Door for a Specific Package/Packages
Able to Connect to Many Garage Door Opener Models
User Programmable
Alerts User if Garage Door Remains Open and When Package is Delivered

*Figure 2- Objective Tree*

## 2. Design Requirements Specification

| Marketing Requirements | Engineering Requirements | Justification |
|---|---|---|
| 1, 5 | The product will open the garage door to a user defined height. | The garage door should not open up all thy way once the barcode has been scanned in order to limit the entrance of people or other objects into the garage. |
| 3 | The product will be powered by 120 VAC. | The power supply will be sourced via wall plug making it easily installed in any garage. |
| 1, 2, 6 | The product will notify the owner if there is a malfunction with the garage door opening/closing within one minute. | The user should know if the garage door malfunctions and is left open. |
| 1, 2, 6 | The product will notify the owner when a package is scanned within one minute.. | The owner should know when the package has been delivered. |
| 1, 4, 6 | The product will automatically scan a presented package within 1 second. | The garage door should open within a reasonable amount of time to allow the delivery person to continue with their other deliveries quickly and efficiently. |
| 1, 6 | The product will have a two second delay before the garage door begins to close after package has been delivered into the garage. | The product needs to account for the time it takes the delivery person to deliver the package into the garage. |
| 6 | The product will be able to detect if the garage door is opened or closed. | The garage door should be closed unless told otherwise to be open. |
| 7 | The product will read barcodes from USPS, UPS, FedEx. | The use of multiple delivery carriers must be included to accommodate different delivery services. |
| 4 | The indicator light will light up green if a package is accepted and red if a package is not accepted. | The indicator light will let the delivery person know if they should hold on to the package. |
| 5 | The product will only be on during the user specified hours of operation. | The product should only be on during the hours when the homeowner is not home. If the homeowner is home there is no need for the package to be delivered using the garage. |
| 1 | The system will remove package information from the database once it has been delivered. | To increase security, barcode information should be discarded once a package has been delivered. |

# 3. Accepted Technical Design

## Timing calculations (AK)

The average speed of a residential garage door is around seven inches per second. That translates into roughly 12-15 seconds of operation time to open or close the garage door. When a package is scanned the software will activate the radio transmitter. When the opener hears a signal from the transmitter, it activates a relay that starts the motor running. After 4 seconds the software will activate the radio transmitter again to stop the garage motor. The ground clearance will be approximately 28 inches. After two seconds the software will activate the radio transmitter again to start the garage motor in the reverse direction and the door will close. A successful package delivery should take no longer than 10 seconds.

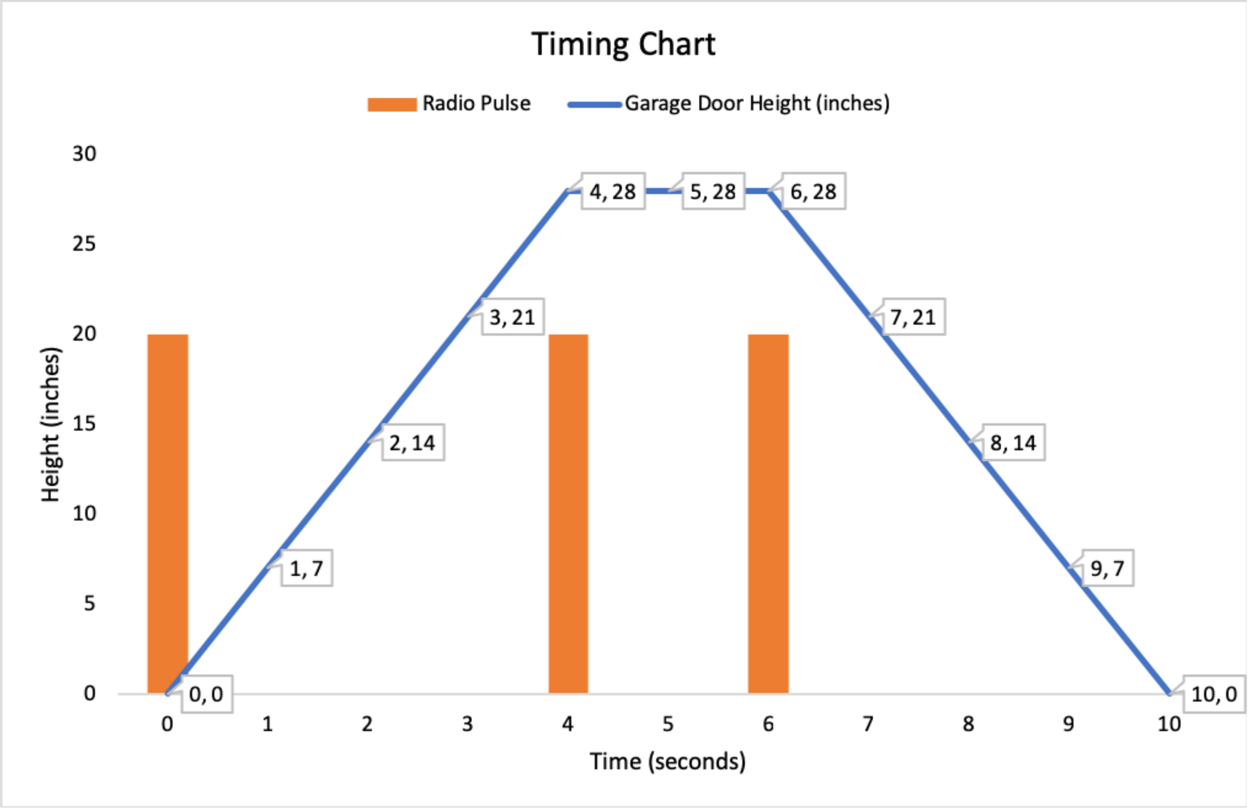


*Figure 3 - Timing Chart*

## Barcode Decoding (TH)

Barcodes are used to keep track of all things that are sold either on the internet or in stores. They are essential in keeping track of all the items that are available. The breakdown of a barcode is much more simple than people tend to think. The barcode is actually just a printed number that a barcode scanner is able to detect and read using and LED or laser light. When the barcode is being scanned light is reflecting into a photoelectric cell. While the scanner is moving the photoelectric cell is generating patterns that correlate with the black an white stripes in each unit. The pulses are then converted to binary code and sent to a computer which can then detect the code.  Each barcode is broken up into units consisting of seven black and white stripes varying in thickness and pattern. A barcode scanner will scan the thickness and the pattern of the lines giving all of the seven units a number 0-9. The tracking number from companies such as amazon are barcode driven and the products tracking number will be put into a barcode for a delivery driver to scan.

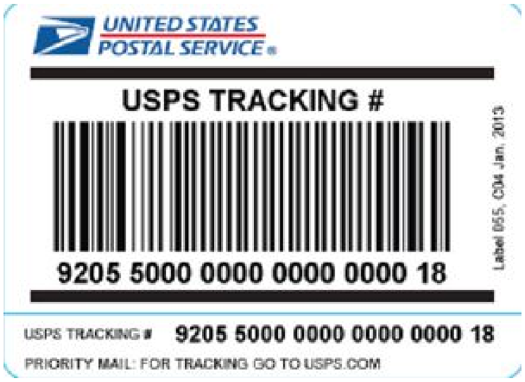


*Figure 4 - Tracking Number Example*

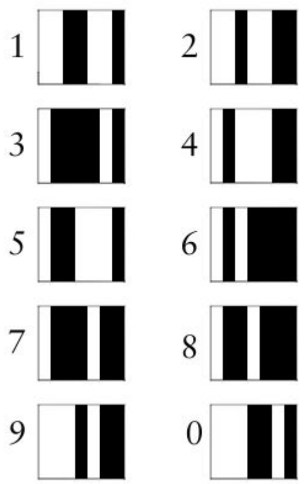


*Figure 5 - Decoded Bits of Barcode*

**Radio Communication (JW)**

The communication between the radio module and the garage door is done through a rolling code operating on either a 310, 315, or 390 MHz frequency. A rolling code is a form of security procedure that generates a new sequence or control code every time a garage door opener is activated. The previous code that was used to open or close the garage door is discarded after a single use and a new code is generated using an algorithm. This new code is unique to the next opening of the garage door and is only useable once. With the number of rolling code combinations being in the billions it means the previous garage door codes cannot be used to hack into the opener or open the garage door in the future. Both devices contain the same algorithm that calculates a string of possible new codes extending from dozens to hundreds of “activations” of the garage door in advance. This means that if a single or multiple activation signals from the transceiver are not received correctly it doesn’t permanently break the sequence and render the wireless communication inoperable.

### PIR Sensor (TH)

The PIR sensor also known as the Passive Infrared or the Pyroelectric Infrared sensor is a sensor that detects levels of infrared radiation. Infrared radiation is the energy or heat an object gives off that is not visible to the human eye. By using an infrared sensor, the smart garage opener will be able to detect if an unwanted person has entered the garage when the home owners are away. When motion is detected the PIR sensor detects a change in infrared levels, the voltage levels change and the PIR sensor sends a high output signal on its output pin to the microcontroller.

### Height Sensor (TH)

The height sensor or the laser sensor is used to determine the distance or height of an object. The laser will be set to detect the garage door height. To do this the laser will be fixed on the garage door where the laser will be focused through its emitting lens. The laser is then able to detect the light that is reflected back from the garage door. When the garage door moves the laser will detect the movement therefore detecting the displacement of the garage door. By using the displacement sensor the smart garage opener will be able to send a sms message to the owner if the garage door is left open.

### Hardware Level 0

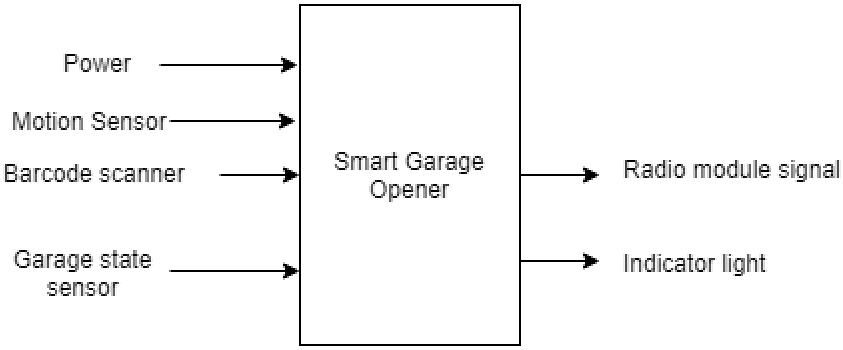


*Figure 6- Hardware Level 0 Block Diagram*

### Hardware Level 0 Theory of Operation

The hardware level 0 shows the primary inputs and outputs of the smart garage opener. Power, the state of the garage door, and a scanned barcode will be input into the system. The system will process these inputs in order to respond by activating or deactivating a radio module and triggering the indicator light appropriately.

| | |
|---|---|
| Module | Smart Garage Opener |
| Designer | Team |
| Inputs | - Barcode scanner<br>- Garage door sensor<br>- Power (120V AC)<br>- Motion Sensor |
| Outputs | - Radio signal for garage door receiver<br>- Indicator light |
| Functionality | The garage door will be opened partially to allow package delivery before closing again after a preset duration. The indicator light will indicate if the barcode is accepted or not. |

*Table 1 - Functional Requirement of Smart Garage Opener*

## Hardware Level 1

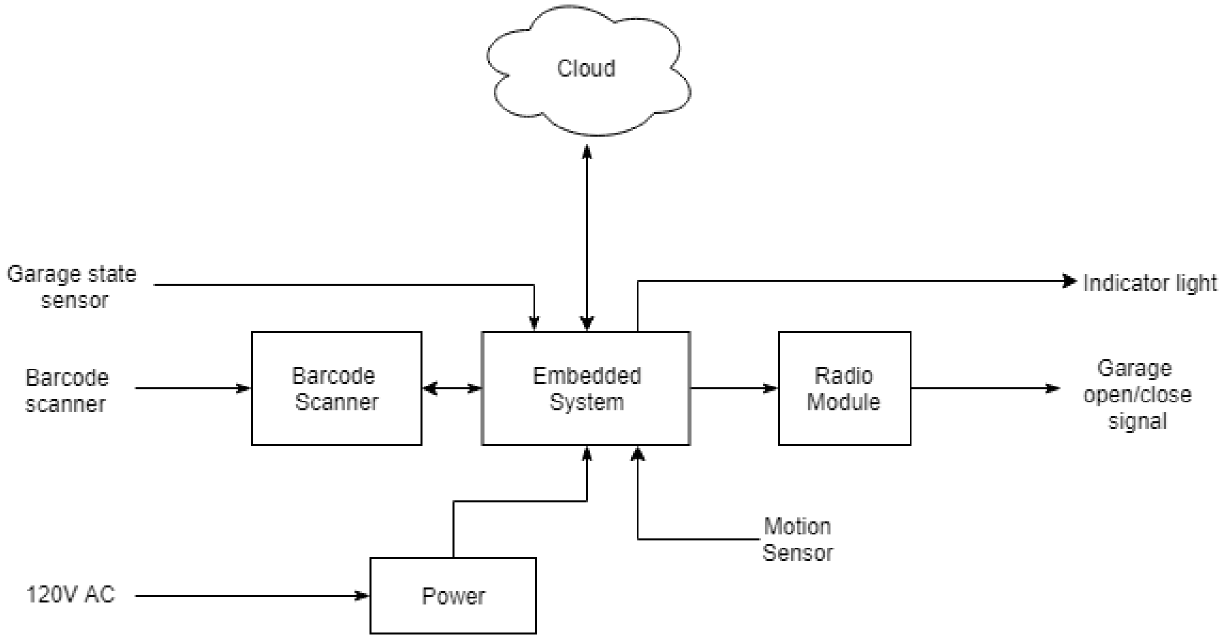


*Figure 7- Hardware Level 1 Block Diagram*

## Hardware Level 1 Theory of Operation

The hardware level 1 diagram expands upon the basic concepts of the level 0 diagram. A barcode is scanned by the barcode scanner and sent to the embedded system. The embedded system connects to the cloud to compare the scanned barcode with stored barcodes. A garage state sensor determines the current open/close state of the garage. If the garage is closed and a barcode is accepted the embedded system signals the radio module to open and close the garage door. An indicator light will indicate either red or green with respect to a rejected or accepted barcode. If the motion sensor detects movement it will alert the user.

| | |
|---|---|
| Module | Power System |
| Designer | Andrey |
| Inputs | - 120V AC |
| Outputs | - 5V DC |
| Functionality | Power system will supply power to the microcontroller. |

*Table 2- Functional Requirement of Power System*

| | |
|---|---|
| Module | Barcode Scanner |
| Designer | Andrey |
| Inputs | - Package barcode from scan<br>- 5V DC from microcontroller |
| Outputs | - Tracking number |
| Functionality | The barcode scanner will scan packages and send package tracking number to the microcontroller for processing. |

*Table 3- Functional Requirement of Software*

| | |
|---|---|
| Module | Embedded System |
| Designer | Andrey |
| Inputs | - 5V DC<br>- Package tracking number<br>- Current state of garage door<br>- List of acceptable tracking numbers from cloud<br>- Detection of Motion from the Motion Sensor |
| Outputs | - Radio signal for garage door receiver<br>- Signal to send owner notification<br>- Power to the radio module<br>- Power to the barcode scanner<br>- Indicator light |

| | |
|---|---|
| Functionality | The microcontroller will take a tracking number and verify it is stored in the cloud and is expected by the owner. When tracking number is verified the microcontroller will power the radio module and the indicator light. |

*Table 4- Functional Requirement of Embedded System*

| | |
|---|---|
| Module | Radio Module |
| Designer | Andrey |
| Inputs | - Power from microcontroller |
| Outputs | - Radio signal for garage door receiver |
| Functionality | The radio module will broadcast a signal for the garage door receiver to open and close. |

*Table 5- Functional Requirement of Radio Module*

| | |
|---|---|
| Module | Cloud |
| Designer | Andrey |
| Inputs | - Tracking number<br>- Additional data from microcontroller |
| Outputs | - Verification information<br>- SMS Notification |
| Functionality | The cloud will store a list of acceptable tracking numbers entered by the user. When a matching tracking number is uploaded from the microcontroller the cloud will send an acknowledgement to the microcontroller which will allow the system to proceed and open the garage door. A notification to the owner will be sent. |

*Table 6- Functional Requirement of Cloud Block*

## Hardware Level 2

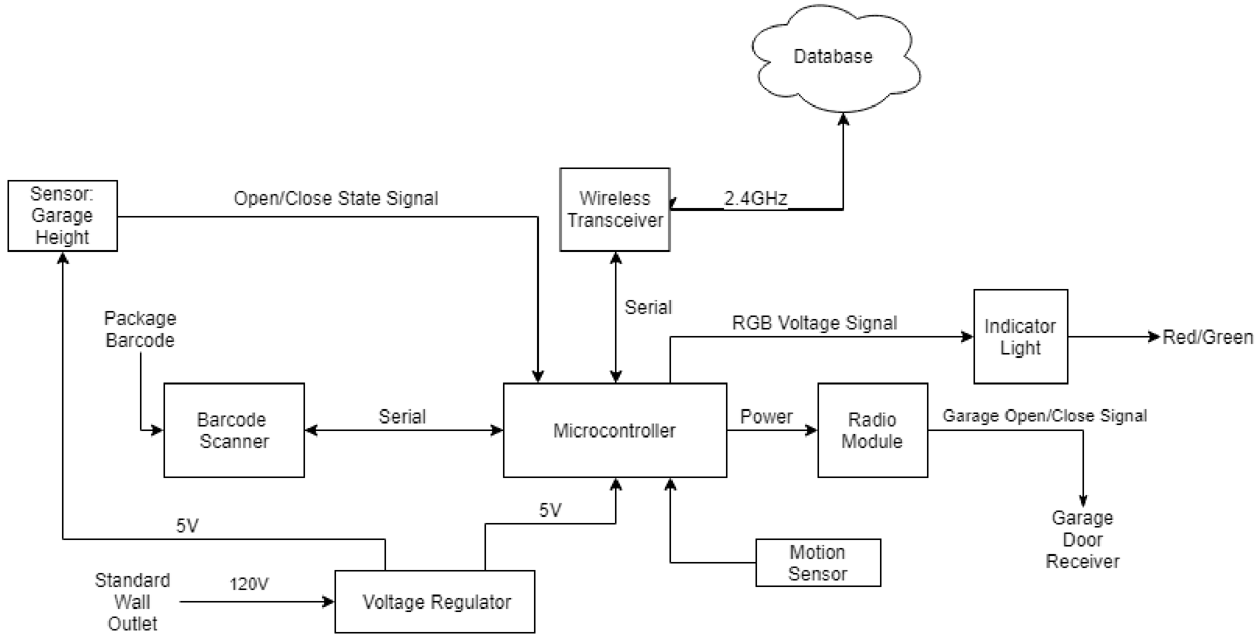


*Figure 8- Hardware Level 2 Block Diagram*

## Hardware Level 2 Theory of Operation

The hardware level 2 diagram outlines the hardware functionality of the system in greater detail. When a barcode is scanned using the barcode scanner it will send a serial signal to the microcontroller. This serial barcode signal will be sent through a serial connection to the wireless transceiver via 2.4GHz to be compared to the acceptable barcodes in the database. If the barcode scanned matches the microcontroller will send power to the radio module and in turn open and close the garage door for the preprogrammed 10 seconds and send an RGB voltage signal to the indicator light to turn green. If the barcode is not accepted the indicator light will receive a red RGB signal. The microcontroller will receive a constant open/close state signal from the garage height sensor and if the garage door is already open will stop the radio module from activating. If the motion sensor detects motion once the garage door has closed then the user will be alerted of detected motion.

| Module | Voltage Regulator |
|---|---|
| Designer | Jacob |
| Inputs | - 120V AC |
| Outputs | - 5V DC |
| Functionality | Power system will supply power to the microcontroller as well as the sensor for the garage door height. |

*Table 7- Functional Requirement of Voltage Regulator*

| Module | Microcontroller |
|---|---|
| Designer | Jacob |
| Inputs | - 5V DC<br>- Serial Data with Package tracking number<br>- Current state of garage door<br>- Serial Data with list of acceptable tracking numbers from cloud<br>- Signal from Motion Sensor |
| Outputs | - Serial signal to send owner notification<br>- Power to the radio module<br>- Power to the barcode scanner<br>- RGB Voltage Signal to indicator light<br>- Power to wireless transceiver |
| Functionality | The microcontroller will take a scanned barcodes serial data and send a wireless signal to the database to verify that barcode is in the database. The wireless transmitter will send either a “yes” or “no” serial signal in response. The microcontroller will respond to the signal by sending an RGB voltage to the indicator light displaying the appropriate colored light and powering the radio module to open the garage door if the barcode was verified. |

*Table 8- Functional Requirement of Microcontroller*

| | |
|---|---|
| Module | Barcode Scanner |
| Designer | Jacob |
| Inputs | - 5V DC<br>- Package Barcode |
| Outputs | - Serial signal containing barcode information |
| Functionality | Barcode scanner will scan and transmit the barcode serial data to the microcontroller. |

*Table 9- Functional Requirement of Barcode Scanner*

| | |
|---|---|
| Module | Sensor: Garage Height |
| Designer | Jacob |
| Inputs | - 5V DC |
| Outputs | - Open/Close State Signal |
| Functionality | Informs the microcontroller of the current state of the garage door. |

*Table 10- Functional Requirement of Garage Height Sensor*

| | |
|---|---|
| Module | Wireless Transceiver |
| Designer | Jacob |
| Inputs | - 5V DC<br>- Serial Signal from Microcontroller<br>- 2.4GHz Wireless signal from Database |
| Outputs | - Serial Signal to Microcontroller<br>- 2.4GHz Wireless signal to Database |
| Functionality | Exchanges serial data wirelessly between the database and the microcontroller for the purpose of verifying scanned barcodes against the database. |

*Table 11- Functional Requirement of Wireless Transceiver*

| | |
|---|---|
| Module | Radio Module |
| Designer | Jacob |
| Inputs | - 3V DC |
| Outputs | - Rolling code on 310, 315, or 390 MHz |
| Functionality | Sends a pulse signal when powered that opens or closes the garage door. |

*Table 12- Functional Requirement of Radio Module*

| | |
|---|---|
| Module | Indicator Light |
| Designer | Jacob |
| Inputs | - RGB Voltage Signal |
| Outputs | - Red/Green Light |
| Functionality | Provides a visual representation of whether or not the package scanned has been accepted. |

*Table 13- Functional Requirement of Indicator Light*

| | |
|---|---|
| Module | Database |
| Designer | Jacob |
| Inputs | - 2.4GHz Wireless Signal |
| Outputs | - 2.4GHz Wireless Signal |
| Functionality | Compares the scanned barcode with stored tracking information and sends the confirmation signal wirelessly back to the microcontroller. |

*Table 14- Functional Requirement of Database*

| | |
|---|---|
| Module | Motion Sensor |
| Designer | Jacob |
| Inputs | - 5V DC<br>- Motion |
| Outputs | - Signal Voltage |
| Functionality | Detects motion within the garage and sends a signal to the microcontroller if motion is detected. |

*Table 15- Functional Requirement of Motion Sensor*

### Software Level 0

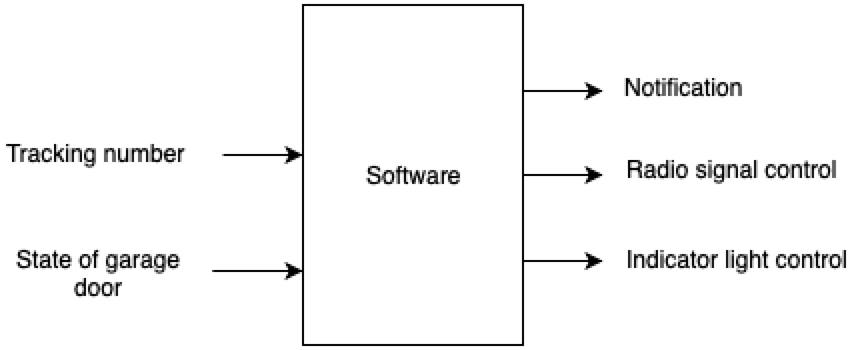


*Figure 9- Software Level 0 Block Diagram*

### Software Level 0 Theory of Operation

The software level 0 block diagram shows what inputs the software will receive and what output the software will control. In theory, if the system is provided with an registered barcode and the garage door is closed then the software will control the indicator light and radio module as well as trigger an SMS notification to be sent.

| | |
|---|---|
| Module | Software |
| Designer | Andrey |
| Inputs | - Tracking number<br>- Garage door state |
| Outputs | - Radio signal control<br>- Indicator light control<br>- Notification |
| Functionality | The software will control the radio signal module, indicator light, and notifications. |

*Table 16- Functional Requirement of Software*

## Software Level 1

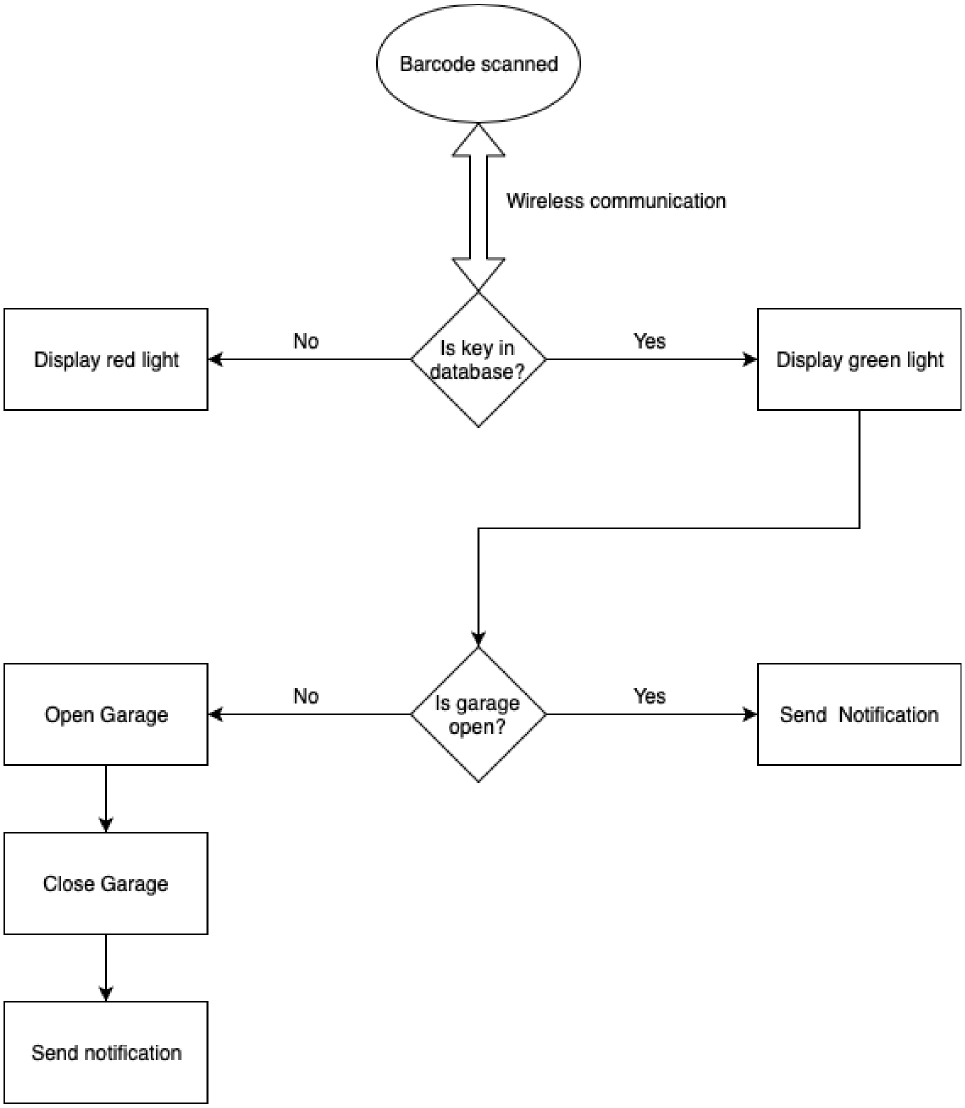


*Figure 10- Software Level 1 Flow Chart*

## Software Level 1 Theory of Operation

This level of the software depicts the main event - the scan of a barcode. The software will check if the barcode is registered, control the indicator light, check the state of the garage door, open the garage door if it is closed, and send the owner a notification.

## Software Level 2

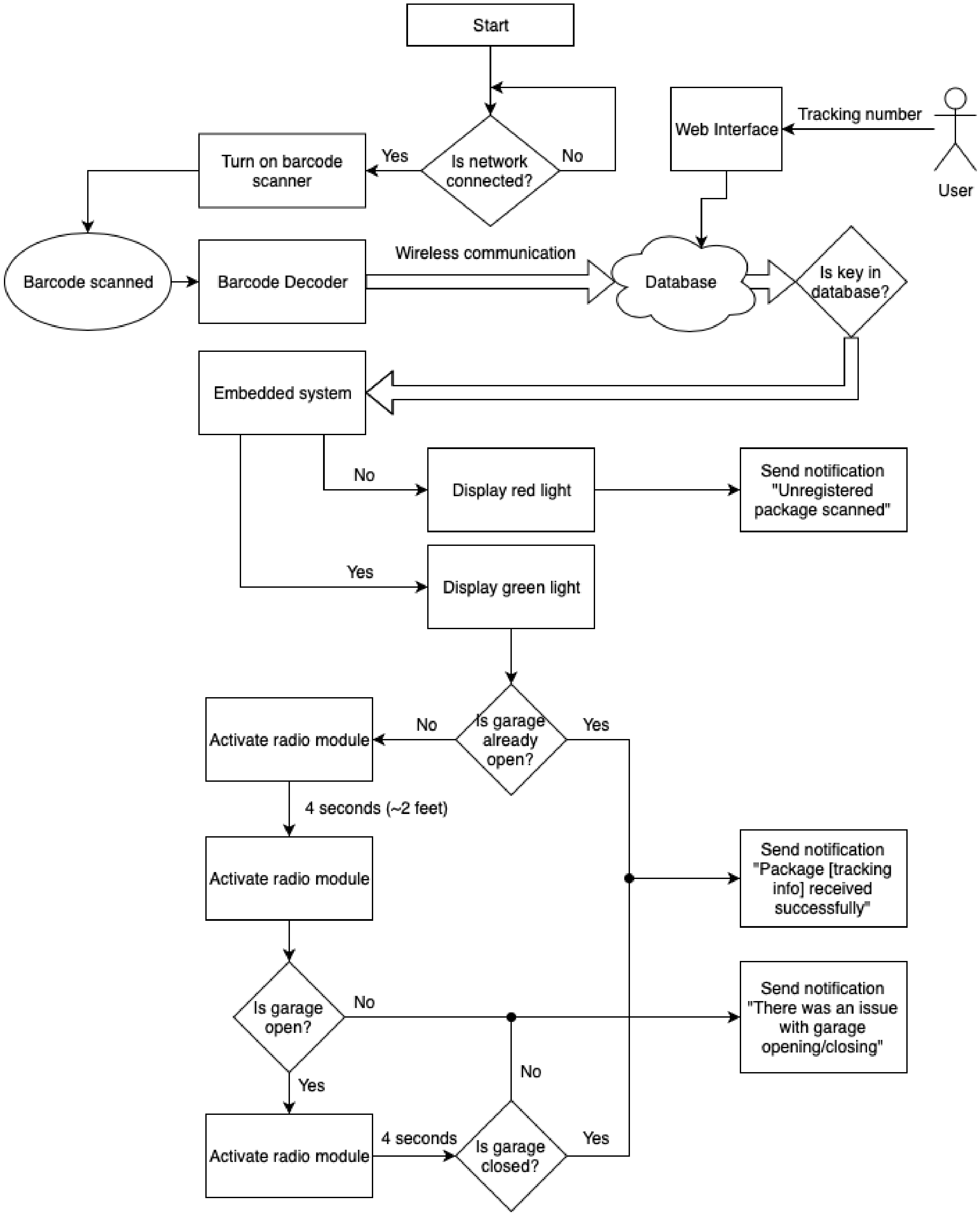


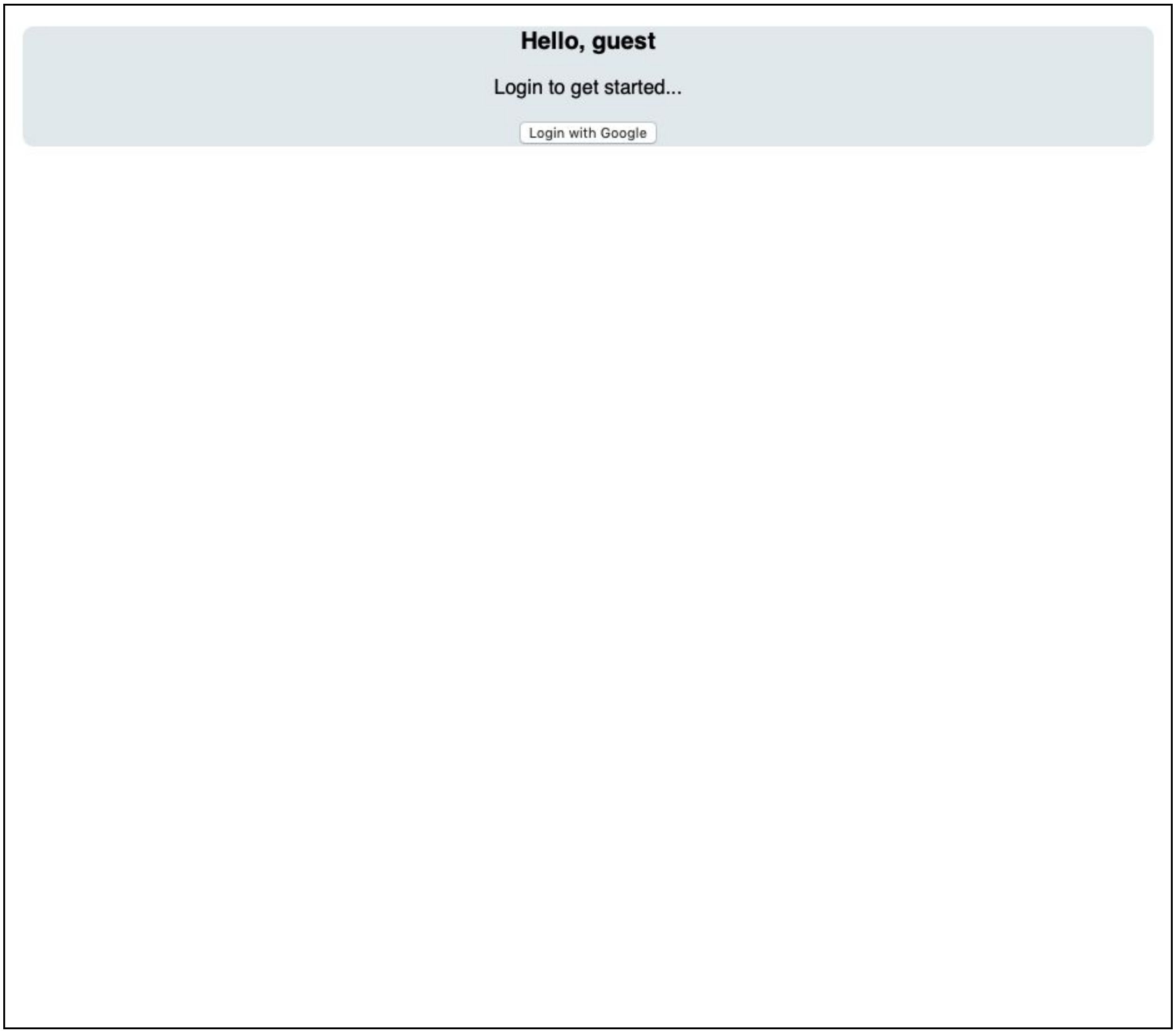
*Figure 11- Software Level 2 Flow Chart*

### Software Level 2 Theory of Operation

This level shows all concepts of software control and operation. The software will include a web interface that the owner uses to save tracking numbers for expected packages. When a barcode is scanned it is then decoded and the tracking number is sent to the database to be checked. If the tracking number is not found in the database then the software will turn on the red LED, trigger an SMS notification to be sent, and the operation ends. If the tracking number has been registered, the software will turn on the green LED and the operation moves to the next stage. The state of the garage door is checked. If the garage is already open then the operation ends and an SMS notification is triggered. If the garage door is closed then the software will power the radio module and a radio signal will be sent to the receiver. The timing sequence can be found in figure 2. The operation ends and an SMS notification is triggered.

### Web Application (AK)

The Smart Garage Opener web application is used to store tracking numbers of packages the are expected for delivery. The web application was built using Angular and Firebase which are both platforms for building mobile and desktop web applications. Firebase was used for its user authentication, real-time database, and cloud function services. This web application also integrates a communication platform called Twilio for it’s text message updates feature. The web app consists of 2 pages - The login page and the home page which are both pictured below.

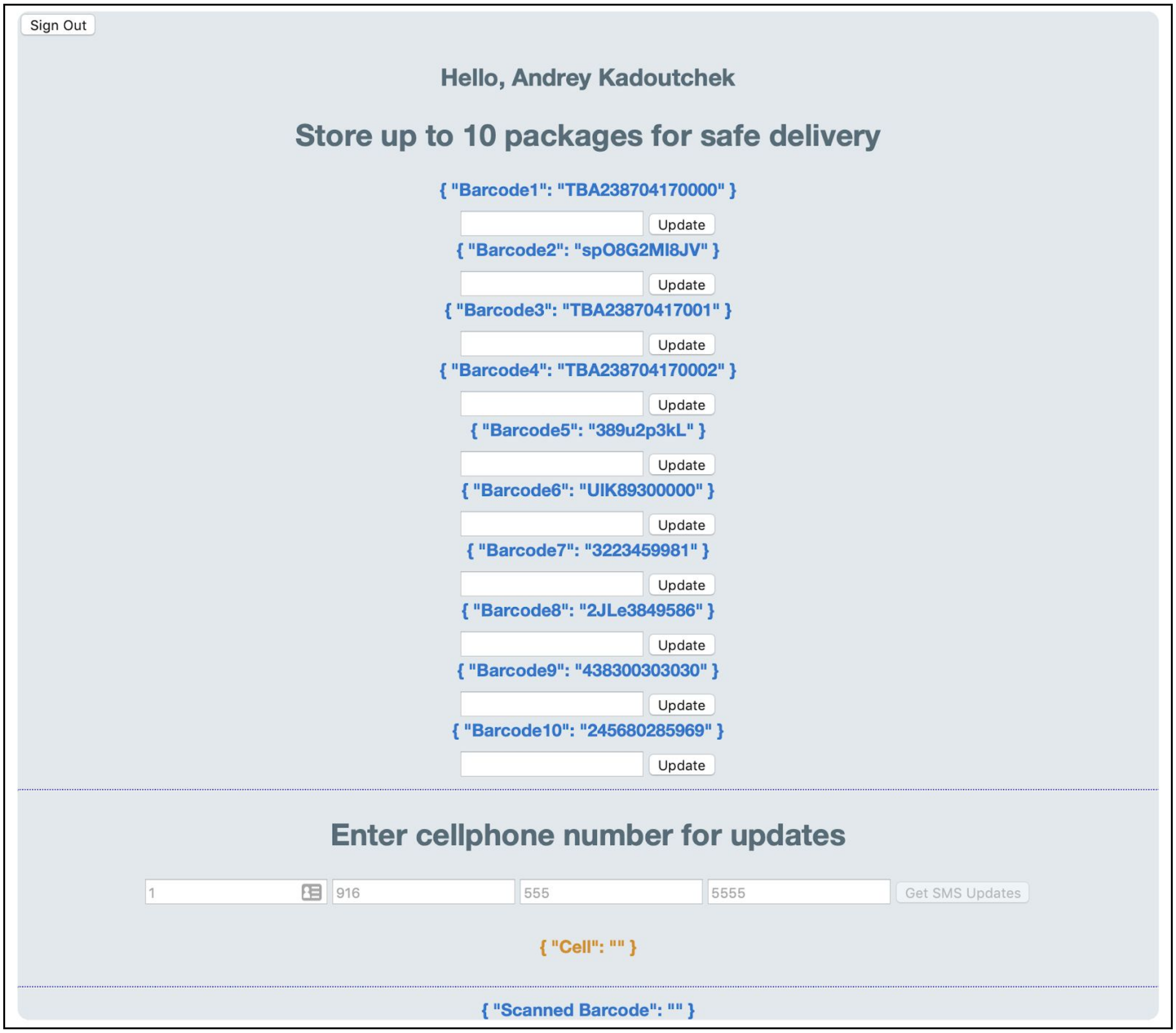


**Hello, guest**

Login to get started...

*Figure 12 - Login Page*

The user must have a Google account in order to login and store tracking numbers.



Hello, Andrey Kadoutchek

Store up to 10 packages for safe delivery

{ "Barcode1": "TBA238704170000" }

{ "Barcode2": "spO8G2MI8JV" }

{ "Barcode3": "TBA23870417001" }

{ "Barcode4": "TBA238704170002" }

{ "Barcode5": "389u2p3kL" }

{ "Barcode6": "UIK89300000" }

{ "Barcode7": "3223459981" }

{ "Barcode8": "2JLe3849586" }

{ "Barcode9": "438300303030" }

{ "Barcode10": "245680285969" }

Enter cellphone number for updates

1 916 555 5555

{ "Cell": "" }

{ "Scanned Barcode": "" }

*Figure 13 - Home Page*

Once the user logs in with their Google account information they will be redirected to the home page pictured here. This page is divided into 3 sections. The first section shows 10 stored tracking numbers and text fields to update them. The second section shows the stored phone number for text message updates. The third section shows the decoded barcode as it is scanned in real time.

## Web Application Code Files (AK)

```
const functions = require('firebase-functions');
const admin = require('firebase-admin');
admin.initializeApp(functions.config().firebase);
const twilio = require('twilio');
var db = admin.firestore()
const accountSid = 'AC431772cffc77a0e4ae40d4be739a17fb'    //firebaseConfig.twilio.sid;
const authToken = 'eddda11615e5746876bd274ca1627475'  //firebaseConfig.twilio.token;
const client = new twilio(accountSid, authToken);
const twilioNumber = '+12162084160'
var cellphone = ''
var espBarcode;

function validE164(num) {
   return /^\+?[1-9]\d{1,14}$/.test(num)
}
function compareB1(num) {                         //functions to compare each database entry
   return new Promise((resolve, reject) => {
       db.doc("barcode1/barcode").get().then(item => {
           console.log(item.data());
           const data = item.data();
           resolve(data.Barcode1 == num);
       });
   })
}
function compareB2(num) {
   return new Promise((resolve, reject) => {
       db.doc("barcode2/barcode").get().then(item => {
           console.log(item.data());
           const data = item.data();
           resolve(data.Barcode2 == num);
       });
   })
}
function compareB3(num) {
   return new Promise((resolve, reject) => {
       db.doc("barcode3/barcode").get().then(item => {
           console.log(item.data());
           const data = item.data();
           resolve(data.Barcode3 == num);
       });
   })
}
function compareB4(num) {
   return new Promise((resolve, reject) => {
       db.doc("barcode4/barcode").get().then(item => {
           console.log(item.data());
           const data = item.data();
           resolve(data.Barcode4 == num);
       });
   })
}
function compareB5(num) {
   return new Promise((resolve, reject) => {
       db.doc("barcode5/barcode").get().then(item => {
           console.log(item.data());
           const data = item.data();
           resolve(data.Barcode5 == num);
       });
   })
}
function compareB6(num) {
   return new Promise((resolve, reject) => {
       db.doc("barcode6/barcode").get().then(item => {
           console.log(item.data());
           const data = item.data();
           resolve(data.Barcode6 == num);
       });
   })
}
function compareB7(num) {
   return new Promise((resolve, reject) => {
       db.doc("barcode7/barcode").get().then(item => {
           console.log(item.data());
           const data = item.data();
           resolve(data.Barcode7 == num);
       });
   })
}
function compareB8(num) {
   return new Promise((resolve, reject) => {
       db.doc("barcode8/barcode").get().then(item => {
           console.log(item.data());
           const data = item.data();
           resolve(data.Barcode8 == num);
       });
   })
}
function compareB9(num) {
   return new Promise((resolve, reject) => {
       db.doc("barcode9/barcode").get().then(item => {
           console.log(item.data());
           const data = item.data();
           resolve(data.Barcode9 == num);
       });
   })
}
function compareB10(num) {
   return new Promise((resolve, reject) => {
       db.doc("barcode10/barcode").get().then(item => {
           console.log(item.data());
           const data = item.data();
           resolve(data.Barcode10 == num);
       });
   })
}
function getNumber() {
   return new Promise((resolve, reject) => {
       db.doc("cell/cellNum").get().then(item => {
           console.log(item.data());
           const data = item.data();
           resolve(data.Cell);
       });
   })
}
exports.textStatus = functions.firestore                //phone number update message
   .document('cell/{cellNum}')
   .onUpdate((change, context) => {
       const newValue = change.after.data();
       const previousValue = change.before.data();
       const phoneNumber = newValue.Cell;
       cellphone = phoneNumber;
       if (!validE164(phoneNumber)) {
           throw new Error('number must be E164 format!')
       }
       const textMessage = {
           body: `Your phone number has been set as:  ${phoneNumber}`,
           to: phoneNumber,
           from: twilioNumber
       }
       return client.messages.create(textMessage)
   })
exports.espfunc = functions.database                    //references realtime database and
   .ref('/esp/{id}')                                   //updates firestore with scanned
barcode
   .onCreate((barcodefield, context) => {
       const barcode = barcodefield.val()
       console.log(`barcode: ${barcode}`)
       espBarcode = barcode;
       var scanDocRef = db.collection("scan").doc("scan");
       scanDocRef.update({
           "Scanned Barcode": espBarcode
       })
   });
function setFound(bool) {                               //paramter for YES/NO signal
   console.log(`parameter: ${bool}`)
   return admin.database().ref('found').set({
       match: bool
   });
}
exports.compareScan = functions.firestore               //generates text message update
depending on if
   .document('scan/{scan}')                            //barcode was found in database or not
   .onUpdate((change, context) => {
       const newValue = change.after.data();
       const scannnedBarcode = newValue["Scanned Barcode"];
       console.log("scannnedBarcode value is: ", scannnedBarcode);
       var acceptedScan = false

       return new Promise(resolve => {
           getNumber().then(item => {
               cellphone = item
               console.log("set new number as ", cellphone);
           });
           Promise.all([
               compareB1(scannnedBarcode),
               compareB2(scannnedBarcode),
               compareB3(scannnedBarcode),
               compareB4(scannnedBarcode),
               compareB5(scannnedBarcode),
               compareB6(scannnedBarcode),
               compareB7(scannnedBarcode),
               compareB8(scannnedBarcode),
               compareB9(scannnedBarcode),
               compareB10(scannnedBarcode)
           ]).then(results => {
               const barcode1result = results[0];
               const barcode2result = results[1];
               const barcode3result = results[2];
               const barcode4result = results[3];
               const barcode5result = results[4];
               const barcode6result = results[5];
               const barcode7result = results[6];
               const barcode8result = results[7];
               const barcode9result = results[8];
               const barcode10result = results[9];

               let foundRes = 0

               if (barcode1result || barcode2result || barcode3result || barcode4result ||
barcode5result || barcode6result || barcode7result || barcode8result || barcode9result ||
barcode10result) {
                   acceptedScan = true
                   const textMessage = {
                       body: `Barcode scan accepted:  ${scannnedBarcode}`,
                       to: cellphone,
                       from: twilioNumber
                   }
                   client.messages.create(textMessage);
                   foundRes = 1;
               }
               else {
                   const textMessage = {
                       body: `Barcode scan not accepted:  ${scannnedBarcode}`,
                       to: cellphone,
                       from: twilioNumber
                   }
                   client.messages.create(textMessage);
               }
               setFound(foundRes).then(x => {
                   resolve(true);
               }).catch(err => {
                   console.log(err);
               });
           });
       })
   })
```

*Figure 14 - Index.js File*

The *Index.js* file contains all cloud functions and database references along with the Twilio API credentials for SMS updates.

```
import { AngularFirestore, AngularFirestoreCollection, AngularFirestoreDocument } from
'@angular/fire/firestore';
import { Observable } from 'rxjs';
import { AngularFireDatabase } from '@angular/fire/database';
import { Validators, FormGroup, FormBuilder } from '@angular/forms';
import { Component, OnInit } from '@angular/core';
import { AuthService } from '../services/auth.service';

interface Note {
 content: string;
}
interface Note2 {
 content: number;
}
@Component({
 selector: 'app-home',
 templateUrl: './home.component.html',
 styleUrls: ['./home.component.css']
})
export class HomeComponent implements OnInit {

 notesCollection1: AngularFirestoreCollection<Note>;
 notes1: Observable<Note[]>;
 notesCollection2: AngularFirestoreCollection<Note>;
 notes2: Observable<Note[]>;
 notesCollection3: AngularFirestoreCollection<Note>;
 notes3: Observable<Note[]>;
 notesCollection4: AngularFirestoreCollection<Note>;
 notes4: Observable<Note[]>;
 notesCollection5: AngularFirestoreCollection<Note>;
 notes5: Observable<Note[]>;
 notesCollection6: AngularFirestoreCollection<Note>;
 notes6: Observable<Note[]>;
 notesCollection7: AngularFirestoreCollection<Note>;
 notes7: Observable<Note[]>;
 notesCollection8: AngularFirestoreCollection<Note>;
 notes8: Observable<Note[]>;
 notesCollection9: AngularFirestoreCollection<Note>;
 notes9: Observable<Note[]>;
 notesCollection10: AngularFirestoreCollection<Note>;
 notes10: Observable<Note[]>;

 notesCollection11: AngularFirestoreCollection<Note2>; // for cell #
 notes11: Observable<Note2[]>;

 notesCollection12: AngularFirestoreCollection<Note>; // for scanned barcode to display
 notes12: Observable<Note[]>;

 newContent1: string;
 newContent2: string;
 newContent3: string;
 newContent4: string;
 newContent5: string;
 newContent6: string;
 newContent7: string;
 newContent8: string;
 newContent9: string;
 newContent10: string;
 newContent11: number; // for cell #
 newContent12: string; // for scanned barcode to display

 constructor(private afs: AngularFirestore, private db: AngularFireDatabase, private fb:
FormBuilder, public auth: AuthService) { }

 ngOnInit() {
   this.notesCollection1 = this.afs.collection('barcode1') //this uses collection
   this.notesCollection2 = this.afs.collection('barcode2')
   this.notesCollection3 = this.afs.collection('barcode3')
   this.notesCollection4 = this.afs.collection('barcode4')
   this.notesCollection5 = this.afs.collection('barcode5')
   this.notesCollection6 = this.afs.collection('barcode6')
   this.notesCollection7 = this.afs.collection('barcode7')
   this.notesCollection8 = this.afs.collection('barcode8')
   this.notesCollection9 = this.afs.collection('barcode9')
   this.notesCollection10 = this.afs.collection('barcode10')


   this.notes1 = this.notesCollection1.valueChanges()
   this.notes2 = this.notesCollection2.valueChanges()
   this.notes3 = this.notesCollection3.valueChanges()
   this.notes4 = this.notesCollection4.valueChanges()
   this.notes5 = this.notesCollection5.valueChanges()
   this.notes6 = this.notesCollection6.valueChanges()
   this.notes7 = this.notesCollection7.valueChanges()
   this.notes8 = this.notesCollection8.valueChanges()
   this.notes9 = this.notesCollection9.valueChanges()
   this.notes10 = this.notesCollection10.valueChanges()

   this.notesCollection11 = this.afs.collection('cell')
   this.notes11 = this.notesCollection11.valueChanges()

   this.notesCollection12 = this.afs.collection('scan')
   this.notes12 = this.notesCollection12.valueChanges()
   this.notes12.subscribe(data => { console.log(data) })
   console.log("hello");
   this.buildForm()
 }
 updateContent1() {
   this.notesCollection1.doc('barcode').update({ Barcode1: this.newContent1 })
 }
 updateContent2() {
   this.notesCollection2.doc('barcode').update({ Barcode2: this.newContent2 })
 }
 updateContent3() {
   this.notesCollection3.doc('barcode').update({ Barcode3: this.newContent3 })
 }
 updateContent4() {
   this.notesCollection4.doc('barcode').update({ Barcode4: this.newContent4 })
 }
 updateContent5() {
   this.notesCollection5.doc('barcode').update({ Barcode5: this.newContent5 })
 }
 updateContent6() {
   this.notesCollection6.doc('barcode').update({ Barcode6: this.newContent6 })
 }
 updateContent7() {
   this.notesCollection7.doc('barcode').update({ Barcode7: this.newContent7 })
 }
 updateContent8() {
   this.notesCollection8.doc('barcode').update({ Barcode8: this.newContent8 })
 }
 updateContent9() {
   this.notesCollection9.doc('barcode').update({ Barcode9: this.newContent9 })
 }
 updateContent10() {
   this.notesCollection10.doc('barcode').update({ Barcode10: this.newContent10 })
 }
 numberForm: FormGroup;
 order: any;
 validateMinMax(min, max) {
   return ['', [
     Validators.required,
     Validators.minLength(min),
     Validators.maxLength(max),
     Validators.pattern('[0-9]+')
   ]]
 }
 buildForm() {
   this.numberForm = this.fb.group({
     country: this.validateMinMax(1, 2),
     area: this.validateMinMax(3, 3),
     prefix: this.validateMinMax(3, 3),
     line: this.validateMinMax(4, 4)
   });
 }
 get e164() {
   const form = this.numberForm.value
   const num = form.country + form.area + form.prefix + form.line
   return `+${num}`
 }
 updatePhoneNumber() {
   this.notesCollection11.doc('cellNum').update({ Cell: this.e164 })
 }
}
```

*Figure 15 - Home.component.ts File*

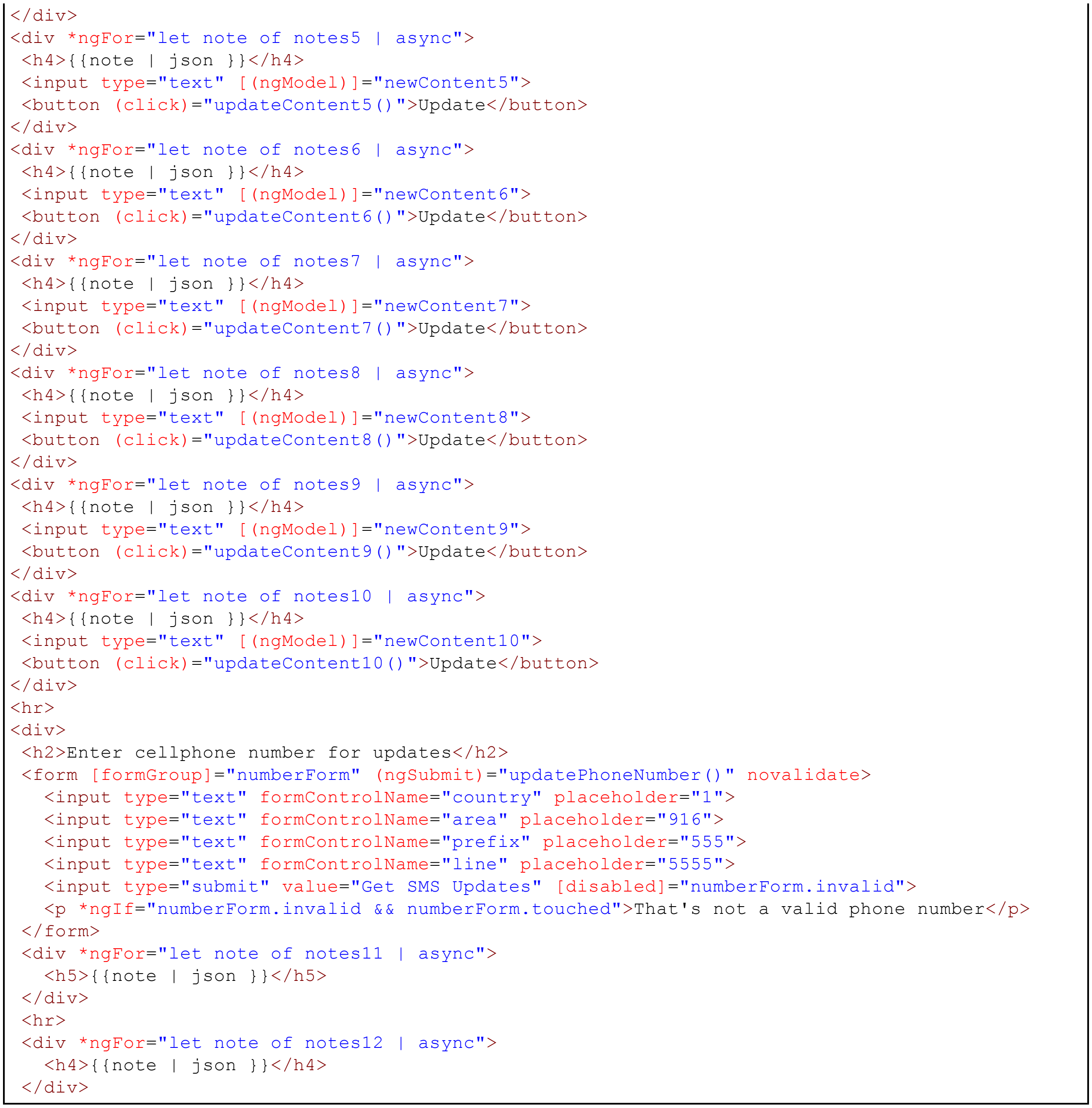
The *home.component.ts* file links the home page to the Firestore database to display the data stored as well as update the data when the user clicks on the ‘update’ buttons. This file also validates that the phone number is entered in the correct format for Twilio to use.

```
<html>
<div *ngIf="auth.user$ | async; then authenticated else guest">
</div>

   <!-- User NOT logged in -->
<ng-template #guest>
   <h3>Hello, guest</h3>
   <p>Login to get started...</p>
   <button (click)="auth.googleSignin()">
       <i class="fa fa-google"></i> Login with Google
   </button>
</ng-template>

   <!-- User logged in -->
<ng-template #authenticated>
</ng-template>
</html>
```

*Figure 16 - Loginpage.component.html File*

This is the html file used by the internet browser to display the login page.

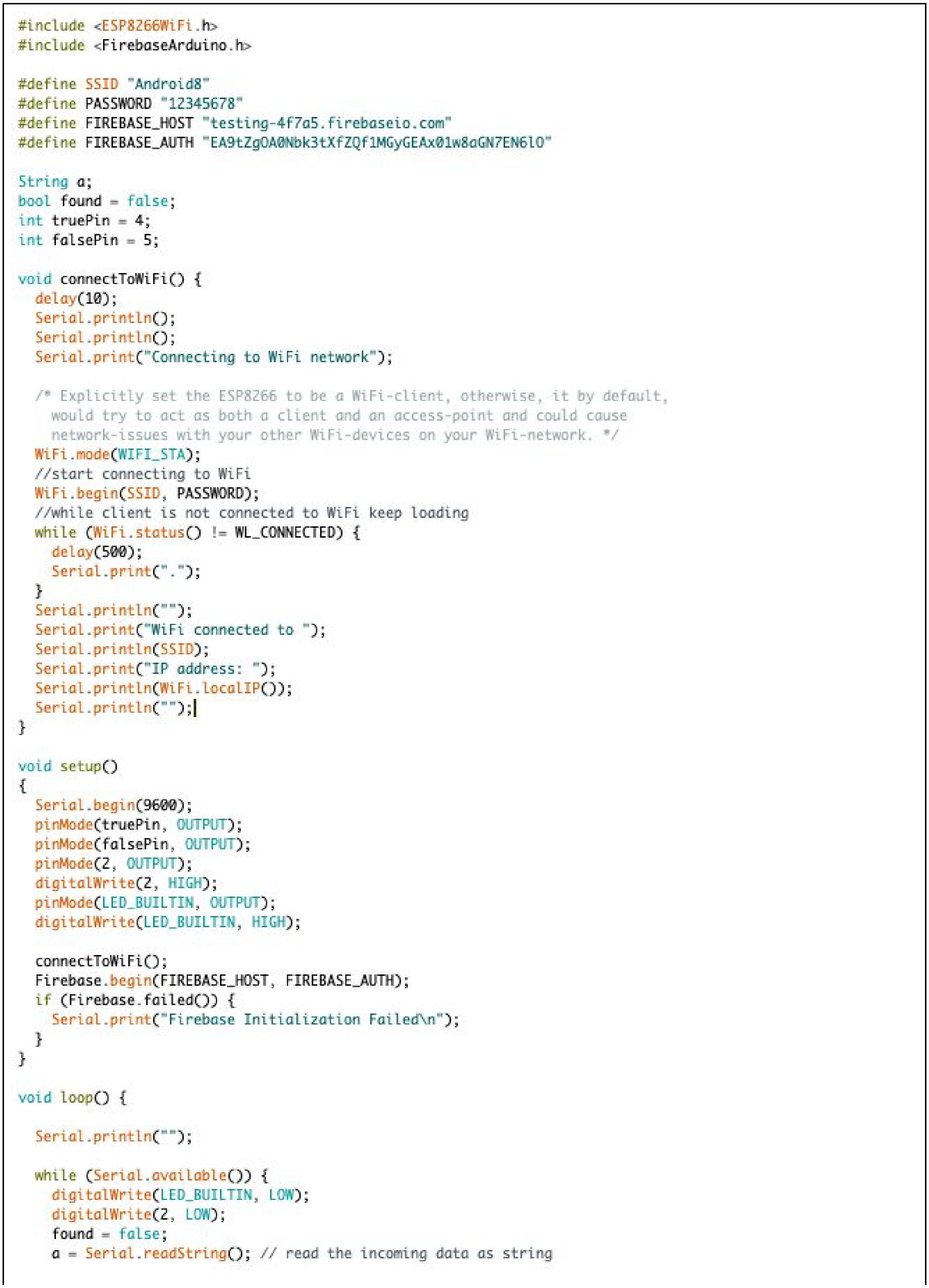
```
<nav>
 <button (click)="auth.signOut()">Sign Out</button>
</nav>
<div *ngIf="auth.user$ | async as user">
 <h3>Hello, {{ user.displayName }}</h3>
</div>
<h2>Store up to 10 packages for safe delivery</h2>
<div *ngFor="let note of notes1 | async">
 <h4>{{note | json }}</h4>
 <input type="text" [(ngModel)]="newContent1">
 <button (click)="updateContent1()">Update</button>
</div>
<div *ngFor="let note of notes2 | async">
 <h4>{{note | json }}</h4>
 <input type="text" [(ngModel)]="newContent2">
 <button (click)="updateContent2()">Update</button>
</div>
<div *ngFor="let note of notes3 | async">
 <h4>{{note | json }}</h4>
 <input type="text" [(ngModel)]="newContent3">
 <button (click)="updateContent3()">Update</button>
</div>
<div *ngFor="let note of notes4 | async">
 <h4>{{note | json }}</h4>
 <input type="text" [(ngModel)]="newContent4">
 <button (click)="updateContent4()">Update</button>
```



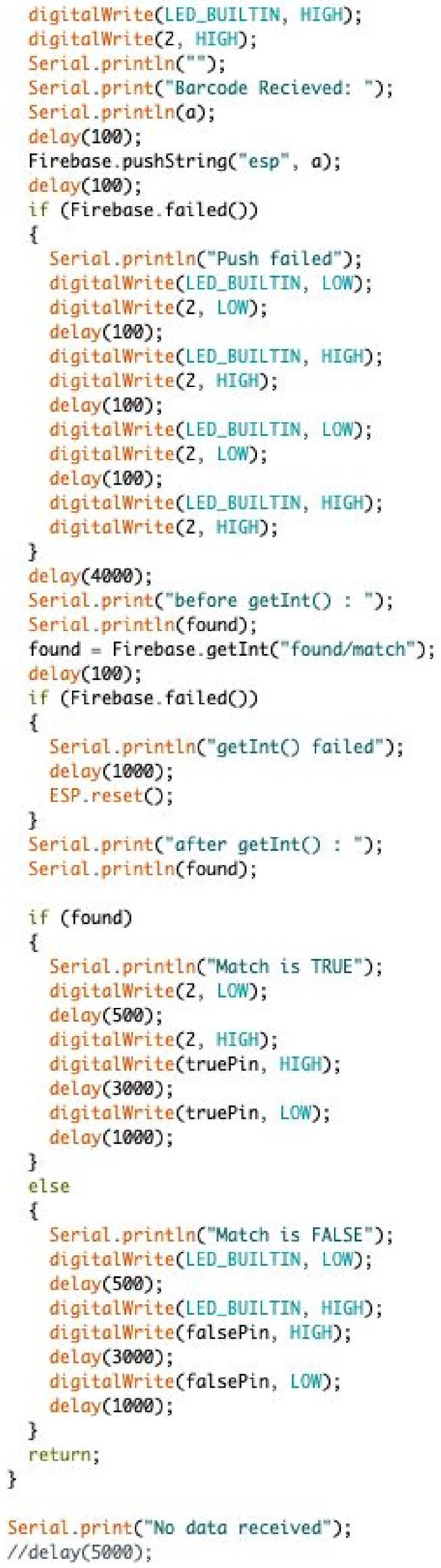
```html
</div>
<div *ngFor="let note of notes5 | async">
 <h4>{{note | json }}</h4>
 <input type="text" [(ngModel)]="newContent5">
 <button (click)="updateContent5()">Update</button>
</div>
<div *ngFor="let note of notes6 | async">
 <h4>{{note | json }}</h4>
 <input type="text" [(ngModel)]="newContent6">
 <button (click)="updateContent6()">Update</button>
</div>
<div *ngFor="let note of notes7 | async">
 <h4>{{note | json }}</h4>
 <input type="text" [(ngModel)]="newContent7">
 <button (click)="updateContent7()">Update</button>
</div>
<div *ngFor="let note of notes8 | async">
 <h4>{{note | json }}</h4>
 <input type="text" [(ngModel)]="newContent8">
 <button (click)="updateContent8()">Update</button>
</div>
<div *ngFor="let note of notes9 | async">
 <h4>{{note | json }}</h4>
 <input type="text" [(ngModel)]="newContent9">
 <button (click)="updateContent9()">Update</button>
</div>
<div *ngFor="let note of notes10 | async">
 <h4>{{note | json }}</h4>
 <input type="text" [(ngModel)]="newContent10">
 <button (click)="updateContent10()">Update</button>
</div>
<hr>
<div>
 <h2>Enter cellphone number for updates</h2>
 <form [formGroup]="numberForm" (ngSubmit)="updatePhoneNumber()" novalidate>
   <input type="text" formControlName="country" placeholder="1">
   <input type="text" formControlName="area" placeholder="916">
   <input type="text" formControlName="prefix" placeholder="555">
   <input type="text" formControlName="line" placeholder="5555">
   <input type="submit" value="Get SMS Updates" [disabled]="numberForm.invalid">
   <p *ngIf="numberForm.invalid && numberForm.touched">That's not a valid phone number</p>
 </form>
 <div *ngFor="let note of notes11 | async">
   <h5>{{note | json }}</h5>
 </div>
 <hr>
 <div *ngFor="let note of notes12 | async">
   <h4>{{note | json }}</h4>
 </div>
```

*Figure 17 - Home.component.html File*

This is the html file used by the internet browser to display the home page.



```
#include <ESP8266WiFi.h>
#include <FirebaseArduino.h>

#define SSID "Android8"
#define PASSWORD "12345678"
#define FIREBASE_HOST "testing-4f7a5.firebaseio.com"
#define FIREBASE_AUTH "EA9tZgOA0Nbk3tXfZQf1MGyGEAx01w8aGN7EN6lO"

String a;
bool found = false;
int truePin = 4;
int falsePin = 5;

void connectToWiFi() {
  delay(10);
  Serial.println();
  Serial.println();
  Serial.print("Connecting to WiFi network");

  /* Explicitly set the ESP8266 to be a WiFi-client, otherwise, it by default,
    would try to act as both a client and an access-point and could cause
    network-issues with your other WiFi-devices on your WiFi-network. */
  WiFi.mode(WIFI_STA);
  //start connecting to WiFi
  WiFi.begin(SSID, PASSWORD);
  //while client is not connected to WiFi keep loading
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("WiFi connected to ");
  Serial.println(SSID);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
  Serial.println("");
}

void setup()
{
  Serial.begin(9600);
  pinMode(truePin, OUTPUT);
  pinMode(falsePin, OUTPUT);
  pinMode(2, OUTPUT);
  digitalWrite(2, HIGH);
  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(LED_BUILTIN, HIGH);

  connectToWiFi();
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
  if (Firebase.failed()) {
    Serial.print("Firebase Initialization Failed\n");
  }
}

void loop() {

  Serial.println("");

  while (Serial.available()) {
    digitalWrite(LED_BUILTIN, LOW);
    digitalWrite(2, LOW);
    found = false;
    a = Serial.readString(); // read the incoming data as string
```



```
    digitalWrite(LED_BUILTIN, HIGH);
    digitalWrite(2, HIGH);
    Serial.println("");
    Serial.print("Barcode Recieved: ");
    Serial.println(a);
    delay(100);
    Firebase.pushString("esp", a);
    delay(100);
    if (Firebase.failed())
    {
      Serial.println("Push failed");
      digitalWrite(LED_BUILTIN, LOW);
      digitalWrite(2, LOW);
      delay(100);
      digitalWrite(LED_BUILTIN, HIGH);
      digitalWrite(2, HIGH);
      delay(100);
      digitalWrite(LED_BUILTIN, LOW);
      digitalWrite(2, LOW);
      delay(100);
      digitalWrite(LED_BUILTIN, HIGH);
      digitalWrite(2, HIGH);
    }
    delay(4000);
    Serial.print("before getInt() : ");
    Serial.println(found);
    found = Firebase.getInt("found/match");
    delay(100);
    if (Firebase.failed())
    {
      Serial.println("getInt() failed");
      delay(1000);
      ESP.reset();
    }
    Serial.print("after getInt() : ");
    Serial.println(found);

    if (found)
    {
      Serial.println("Match is TRUE");
      digitalWrite(2, LOW);
      delay(500);
      digitalWrite(2, HIGH);
      digitalWrite(truePin, HIGH);
      delay(3000);
      digitalWrite(truePin, LOW);
      delay(1000);
    }
    else
    {
      Serial.println("Match is FALSE");
      digitalWrite(LED_BUILTIN, LOW);
      delay(500);
      digitalWrite(LED_BUILTIN, HIGH);
      digitalWrite(falsePin, HIGH);
      delay(3000);
      digitalWrite(falsePin, LOW);
      delay(1000);
    }
    return;
  }

  Serial.print("No data received");
  //delay(5000);
```

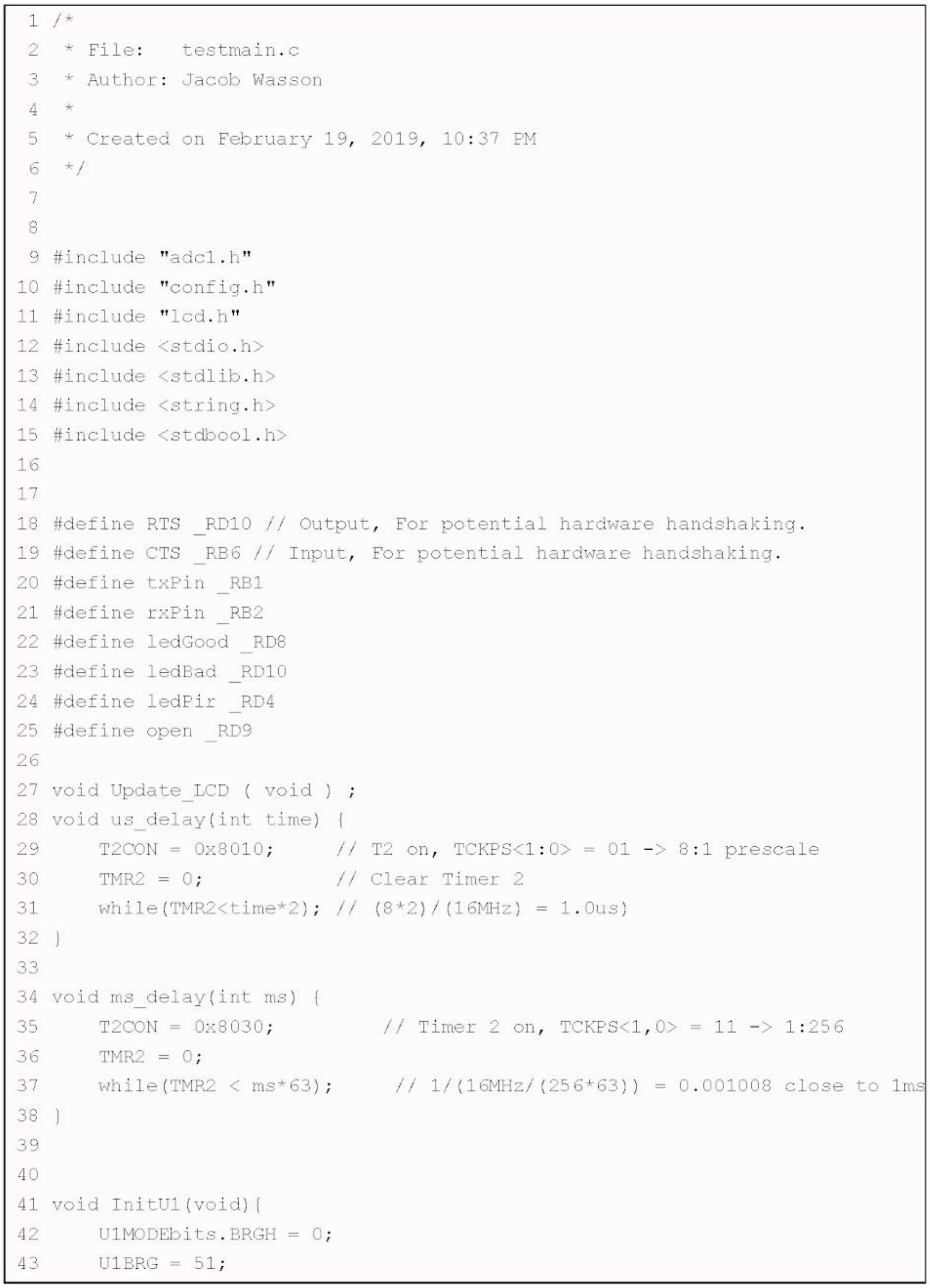
```
  Serial.print(".");
  delay(1000);
  Serial.print(".");
  delay(1000);
  Serial.print(".");
  delay(1000);
  Serial.print(".");
  delay(1000);
  Serial.print(".");
}
```

*Figure 18 - Wifi Module Code File*

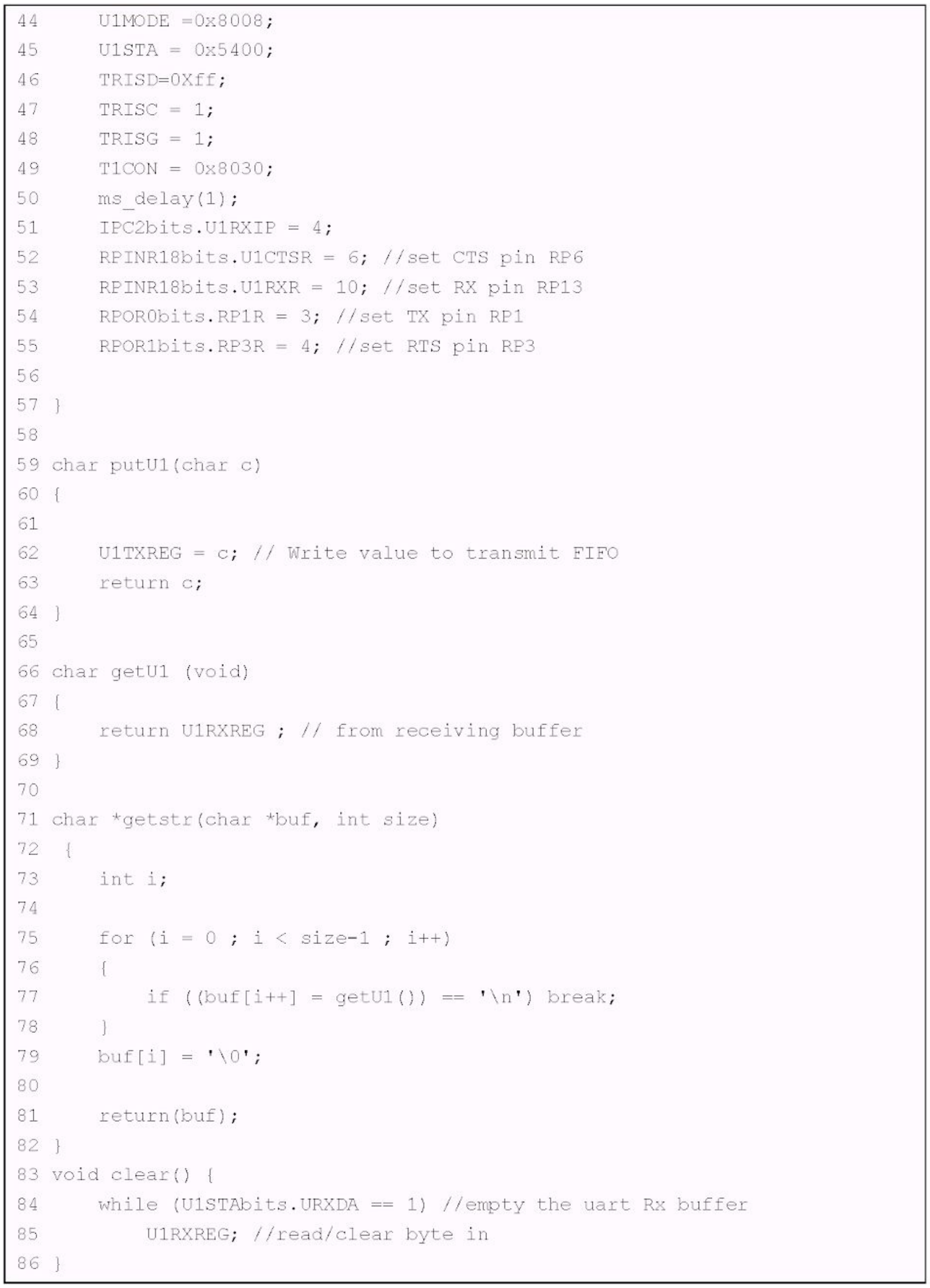
This code contains the commands to connect the Wifi module to a Wifi network and send and receive data to the Firestore database.

### Microcontroller Embedded System Code (JW)

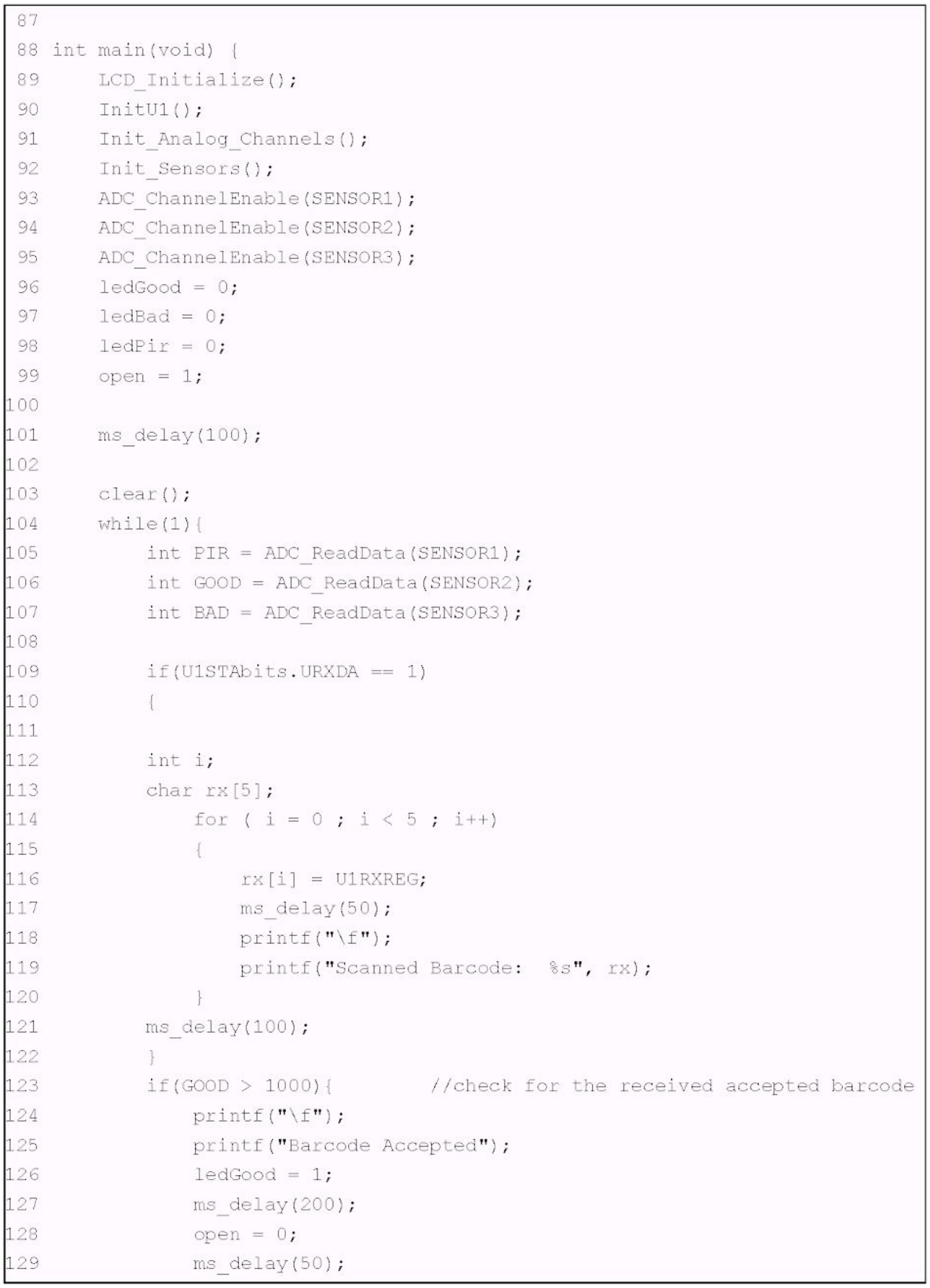
This code is used to communicate the barcode scanner to the microcontroller as well as receiving the response back from the wireless module in order for the system to respond accordingly. The main file program is displayed below:



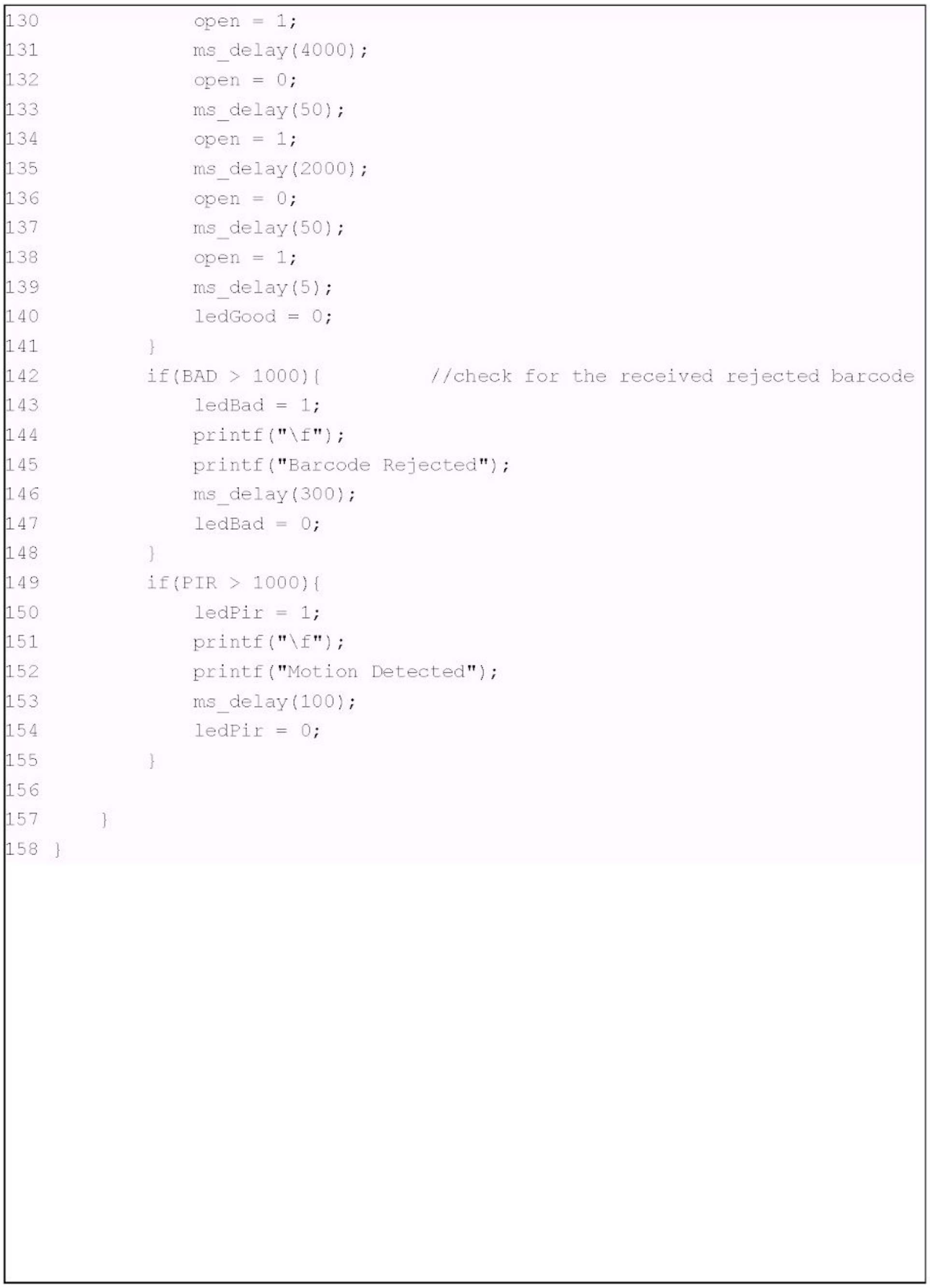
```c
/*
 * File:   testmain.c
 * Author: Jacob Wasson
 *
 * Created on February 19, 2019, 10:37 PM
 */


#include "adc1.h"
#include "config.h"
#include "lcd.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>


#define RTS _RD10 // Output, For potential hardware handshaking.
#define CTS _RB6 // Input, For potential hardware handshaking.
#define txPin _RB1
#define rxPin _RB2
#define ledGood _RD8
#define ledBad _RD10
#define ledPir _RD4
#define open _RD9

void Update_LCD ( void ) ;
void us_delay(int time) {
    T2CON = 0x8010;     // T2 on, TCKPS<1:0> = 01 -> 8:1 prescale
    TMR2 = 0;           // Clear Timer 2
    while(TMR2<time*2); // (8*2)/(16MHz) = 1.0us)
}

void ms_delay(int ms) {
    T2CON = 0x8030;         // Timer 2 on, TCKPS<1,0> = 11 -> 1:256
    TMR2 = 0;
    while(TMR2 < ms*63);     // 1/(16MHz/(256*63)) = 0.001008 close to 1ms
}


void InitU1(void){
    U1MODEbits.BRGH = 0;
    U1BRG = 51;
```



```
    U1MODE =0x8008;
    U1STA = 0x5400;
    TRISD=0Xff;
    TRISC = 1;
    TRISG = 1;
    T1CON = 0x8030;
    ms_delay(1);
    IPC2bits.U1RXIP = 4;
    RPINR18bits.U1CTSR = 6; //set CTS pin RP6
    RPINR18bits.U1RXR = 10; //set RX pin RP13
    RPOR0bits.RP1R = 3; //set TX pin RP1
    RPOR1bits.RP3R = 4; //set RTS pin RP3

}

char putU1(char c)
{

    U1TXREG = c; // Write value to transmit FIFO
    return c;
}

char getU1 (void)
{
    return U1RXREG ; // from receiving buffer
}

char *getstr(char *buf, int size)
 {
    int i;

    for (i = 0 ; i < size-1 ; i++)
    {
        if ((buf[i++] = getU1()) == '\n') break;
    }
    buf[i] = '\0';

    return(buf);
}
void clear() {
    while (U1STAbits.URXDA == 1) //empty the uart Rx buffer
        U1RXREG; //read/clear byte in
}
```



```c

int main(void) {
    LCD_Initialize();
    InitU1();
    Init_Analog_Channels();
    Init_Sensors();
    ADC_ChannelEnable(SENSOR1);
    ADC_ChannelEnable(SENSOR2);
    ADC_ChannelEnable(SENSOR3);
    ledGood = 0;
    ledBad = 0;
    ledPir = 0;
    open = 1;

    ms_delay(100);

    clear();
    while(1){
        int PIR = ADC_ReadData(SENSOR1);
        int GOOD = ADC_ReadData(SENSOR2);
        int BAD = ADC_ReadData(SENSOR3);

        if(U1STAbits.URXDA == 1)
        {

        int i;
        char rx[5];
            for ( i = 0 ; i < 5 ; i++)
            {
                rx[i] = U1RXREG;
                ms_delay(50);
                printf("\f");
                printf("Scanned Barcode:  %s", rx);
            }
        ms_delay(100);
        }
        if(GOOD > 1000){        //check for the received accepted barcode
            printf("\f");
            printf("Barcode Accepted");
            ledGood = 1;
            ms_delay(200);
            open = 0;
            ms_delay(50);
```



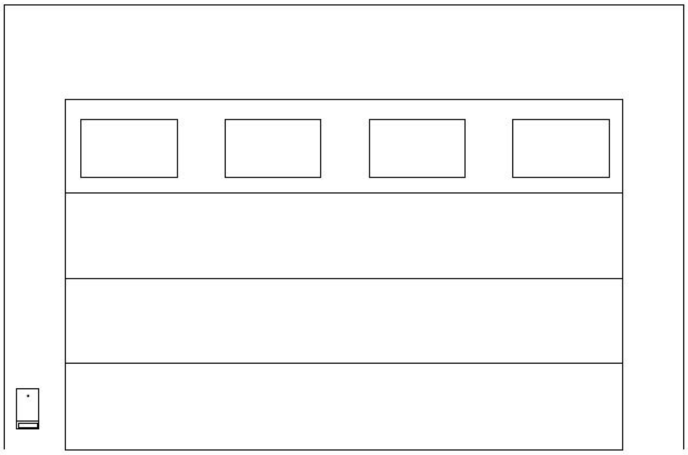
```
            open = 1;
            ms_delay(4000);
            open = 0;
            ms_delay(50);
            open = 1;
            ms_delay(2000);
            open = 0;
            ms_delay(50);
            open = 1;
            ms_delay(5);
            ledGood = 0;
        }
        if(BAD > 1000){         //check for the received rejected barcode
            ledBad = 1;
            printf("\f");
            printf("Barcode Rejected");
            ms_delay(300);
            ledBad = 0;
        }
        if(PIR > 1000){
            ledPir = 1;
            printf("\f");
            printf("Motion Detected");
            ms_delay(100);
            ledPir = 0;
        }

    }
}
```

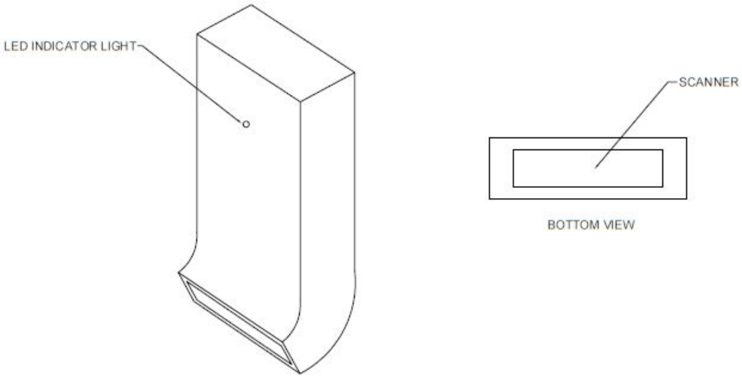
*Figure 19 - Microcontroller Code File*

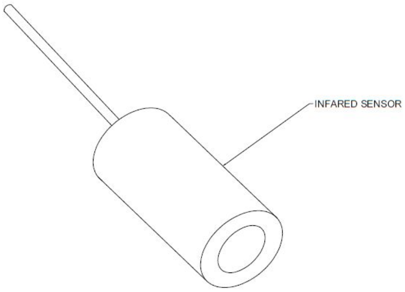
The logic of the code is as follows. The UART settings are set via the InitU1 function such that it operates in standard speed mode with a 9600 baud rate, no parity, 1 stop bit, 8 bits per transfer, least significant bit first, and a non inverted signal. These parameters for the UART were determined from the analysing the output of the barcode scanner once it had gone through the MAX232 device. After declaring the correct signal parameters the pin designations for each input and output are determined. It should be noted that the pins chosen for the received GOOD, BAD, and PIR signals were selected to be in analog mode. By choosing analog mode we were able to more easily trigger the desired response to an input in case the voltage level dropped below the 3.3V needed for a digital high. After declaring all pins the microcontroller waits for one of four conditions to be met. The first condition is if a barcode is detected. If detected the barcode will be read and displayed on the LCD of the explorer 16/32 board as it is received by the wifi module separately. The second condition is whether or not a signal is received back from the wifi module indicating a barcode has been accepted by the database and the garage door should be allowed to open. In this case a signal is sent to a green LED to inform the user that the barcode is accepted. After lighting the LED the microcontroller activates the radio module at three predetermined intervals to represent opening, stopping, and then closing of the garage door at the predescribed times and corresponding heights. It is noted that the radio module is unlike the indicator LED’s in that a constant high signal must be provided during an idle state while activation is achieved by providing a low ground voltage. The third condition is similar to the second in that a signal from the wifi module indicating a rejected barcode from the database will result in a red LED flashing on to indicate to the user that the barcode was not in the system. The final condition is whether or not the PIR sensor has detected motion and if so the LCD will display a warning the motion has been detected.

In addition to the main microcontroller code a number of source and header files were referenced that were taken from the microcontroller website. These files are open for distribution as sample code on the microcontroller website and are included in the references. The primary function of these code segments were too easily output to the explorer board LCD display, set pins to an analog function, and alter the printf command to go from the computer console to the LCD. The additional files used were the adc.h, adc.c, lcd.h, lcd.c, and lcd_printf.c found in the explorer16_demo_pic24fj1024gb610_pim sample code that is open for download by anyone on the microcontroller website.

**Mechanical Sketch of System (TH)**




LED INDICATOR LIGHT
SCANNER
BOTTOM VIEW


INFARED SENSOR

**Schematics (JW, TH)**

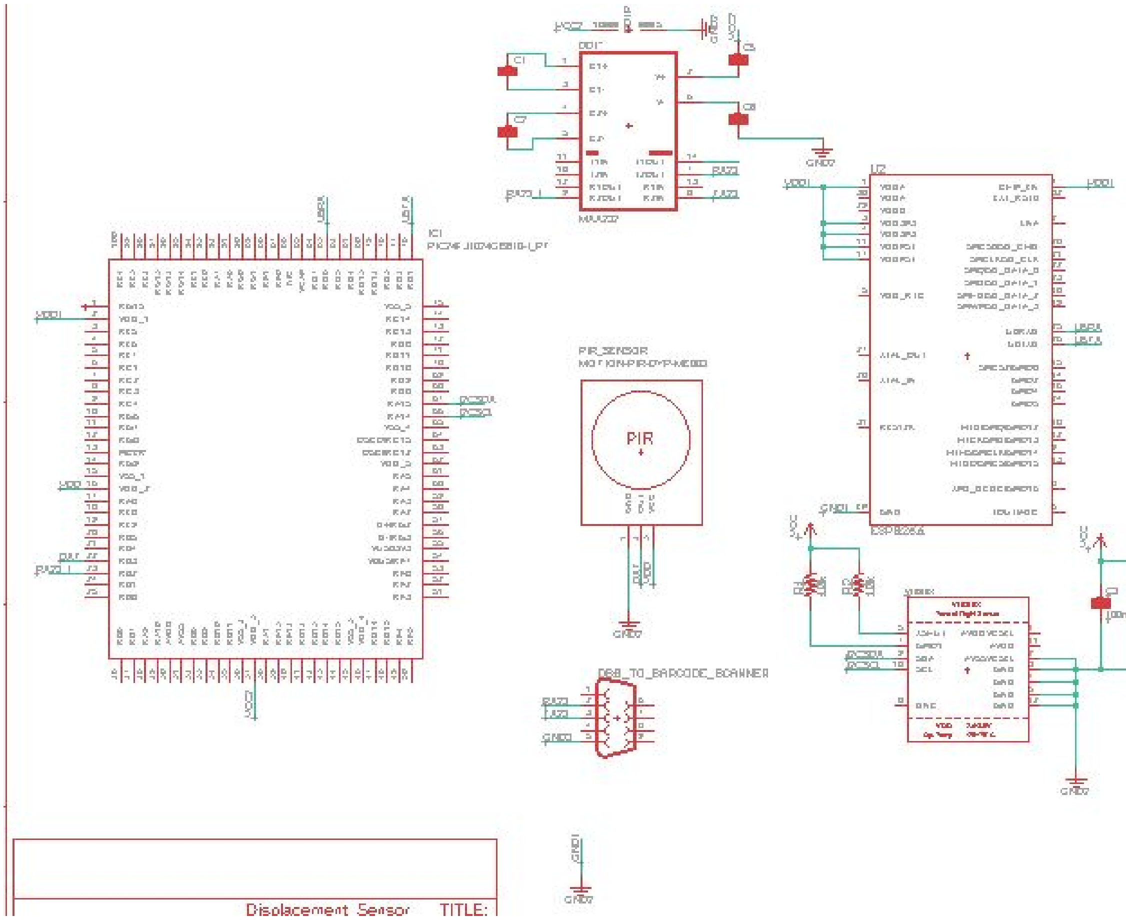


*Figure 20- Control Board Circuit Design*

The control board is built around a single PIC24FJ1024GB610 Microcontroller designated as such on the schematic but will be referred to as PIC. This Microcontroller is powered by a single 5.5mm 5V input power jack,which is not shown on the schematic. The power jack provides the VCC voltage to PIC, BARCODE_SCANNER, GARAGE_HEIGHT_SENSOR, PIR_SENSOR and the VDD pins on the PIC microcontroller. The barcode scanner is connected using a DB-9 connector and is designated as BARCODE-SCANNER in the schematic. The BARCODE_SCANNER . The barcode scanner also uses a module in order to decode the barcode. The module which is not labeled in the schematic is an  RS232 module which connects to the barcode scanner using a DB-9 connector. The RS232 module requires pin connections to RX, GND and VCC pins on the pic microcontroller. The Wifi module, ESP8266, consists of eight through hole pin connections that attach the module to the control board directly. In this case only five of the pins are needed and due to CHIP_EN and VDD pins being shorted as they both require a 3V input voltage. The other three pins on the WIRELESS-MODULE correspond to RX, TX, and a gnd connection. The garage opener designated by RADIO_MODULE requires only a single pin to transmit a voltage to the wireless garage opener from the PIC microcontroller. The sensor that determines the height of the garage door, VL53L0X, is an optical range finding laser that requires the SDA and the SCL pins t be connected to I2C compatible pins on the pic, a VCC connection, and a connection to ground. PIR_SENSOR is a passive infrared sensor that sends a voltage signal to the microcontroller from one pin and has a second pin connecting to ground.

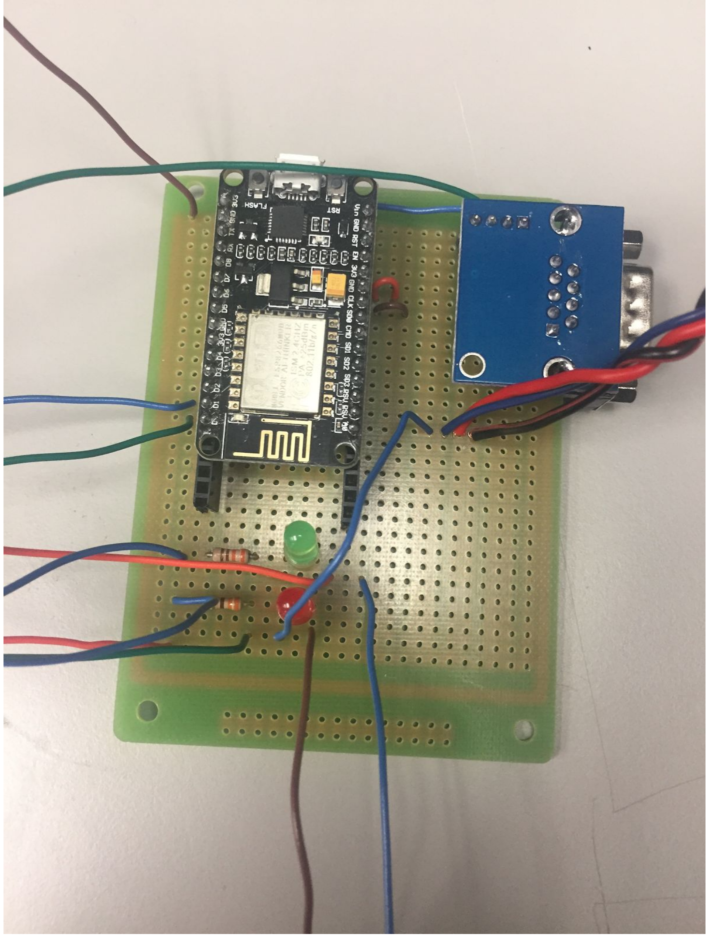


*Figure 21- Control Board Circuit Photo*

**Radio Module (JW)**

For the radio module a standard wireless garage door opener was modified to represent the ease with integration of standard devices. The three button module was disassembled and a button was removed and replaced with a two wire connection. One wire was held at a constant high voltage and by shorting the two wires the device would activate as intended. Due to the need to activate the device using the microcontroller it was established that the input wire had to be kept at a constant high (blue wire in figure 22) while the other must be grounded (brown wire in figure 22). By setting the input wire low the device activated and only required an activation time of 50ms for the proper functionality. As explained earlier this triggering was timed at three separate intervals to achieve the desired garage height and timings.

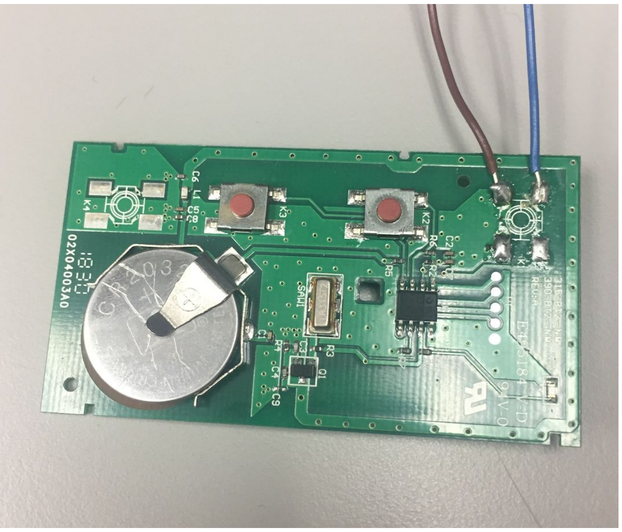
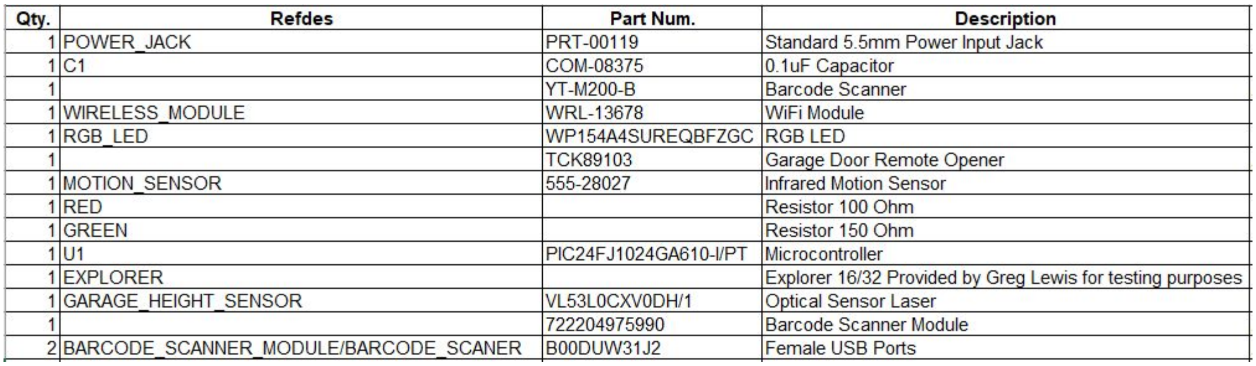


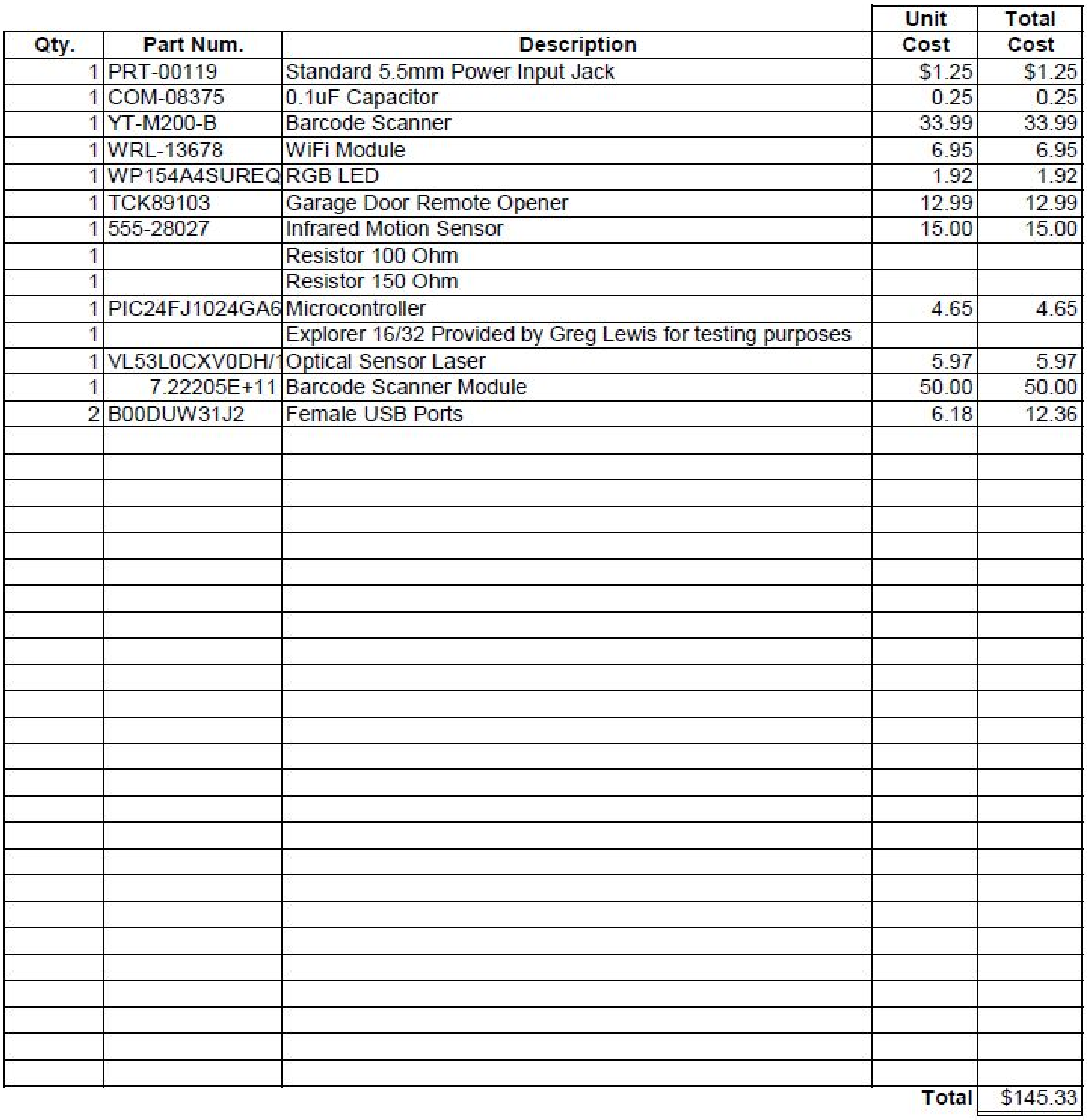
*Figure 22- Radio Module Circuit Photo*

## 4. Parts List



| Qty. | Refdes | Part Num. | Description |
|---|---|---|---|
| 1 | POWER_JACK | PRT-00119 | Standard 5.5mm Power Input Jack |
| 1 | C1 | COM-08375 | 0.1uF Capacitor |
| 1 |  | YT-M200-B | Barcode Scanner |
| 1 | WIRELESS_MODULE | WRL-13678 | WiFi Module |
| 1 | RGB_LED | WP154A4SUREQBFZGC | RGB LED |
| 1 |  | TCK89103 | Garage Door Remote Opener |
| 1 | MOTION_SENSOR | 555-28027 | Infrared Motion Sensor |
| 1 | RED |  | Resistor 100 Ohm |
| 1 | GREEN |  | Resistor 150 Ohm |
| 1 | U1 | PIC24FJ1024GA610-I/PT | Microcontroller |
| 1 | EXPLORER |  | Explorer 16/32 Provided by Greg Lewis for testing purposes |
| 1 | GARAGE_HEIGHT_SENSOR | VL53L0CXV0DH/1 | Optical Sensor Laser |
| 1 |  | 722204975990 | Barcode Scanner Module |
| 2 | BARCODE_SCANNER_MODULE/BARCODE_SCANER | B00DUW31J2 | Female USB Ports |

*Table 17 - Parts List*

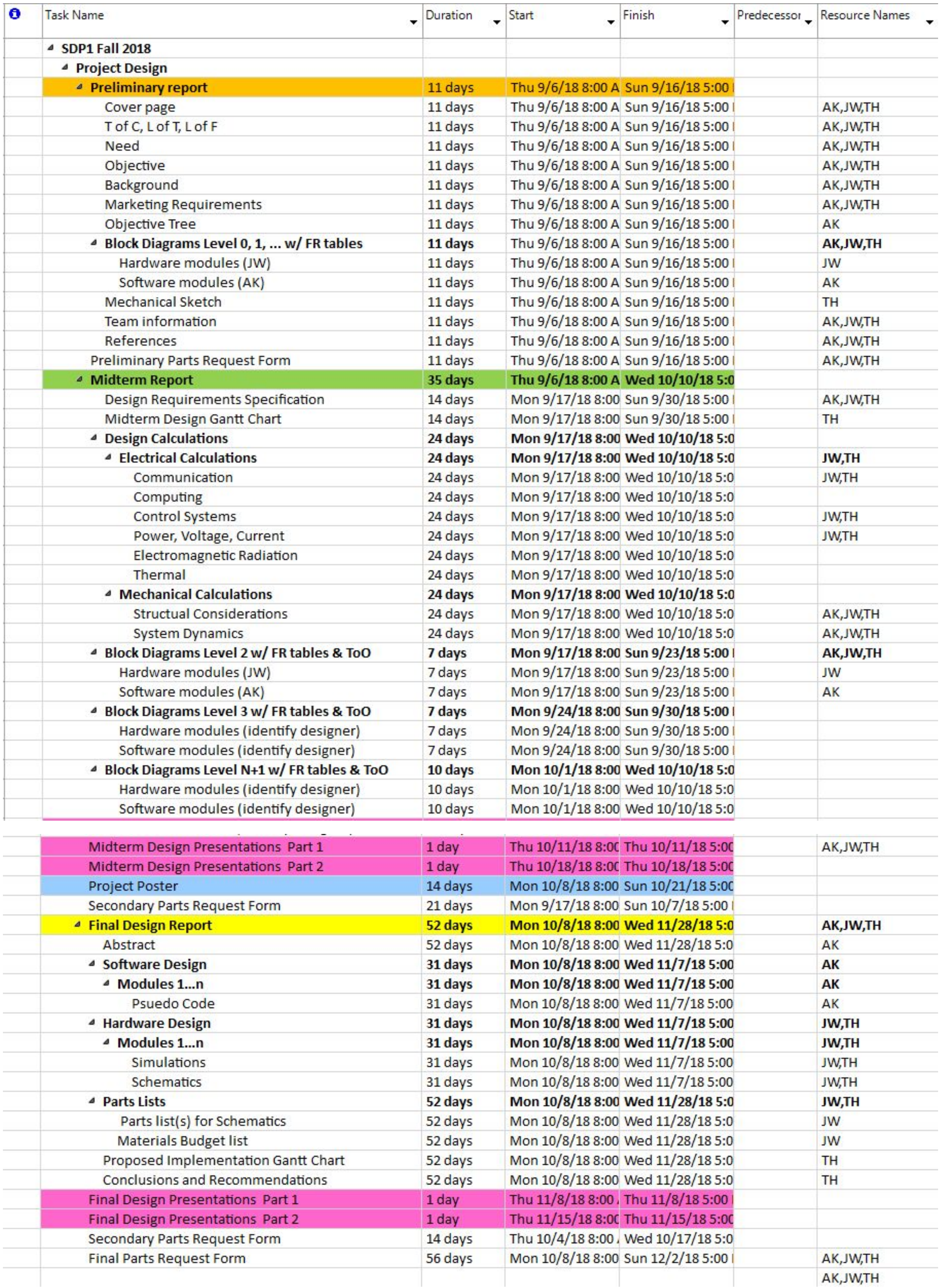


| | | | Unit | Total |
|---|---|---|---|---|
| Qty. | Part Num. | Description | Cost | Cost |
| 1 | PRT-00119 | Standard 5.5mm Power Input Jack | $1.25 | $1.25 |
| 1 | COM-08375 | 0.1uF Capacitor | 0.25 | 0.25 |
| 1 | YT-M200-B | Barcode Scanner | 33.99 | 33.99 |
| 1 | WRL-13678 | WiFi Module | 6.95 | 6.95 |
| 1 | WP154A4SUREQ | RGB LED | 1.92 | 1.92 |
| 1 | TCK89103 | Garage Door Remote Opener | 12.99 | 12.99 |
| 1 | 555-28027 | Infrared Motion Sensor | 15.00 | 15.00 |
| 1 | | Resistor 100 Ohm | | |
| 1 | | Resistor 150 Ohm | | |
| 1 | PIC24FJ1024GA6 | Microcontroller | 4.65 | 4.65 |
| 1 | | Explorer 16/32 Provided by Greg Lewis for testing purposes | | |
| 1 | VL53L0CXV0DH/1 | Optical Sensor Laser | 5.97 | 5.97 |
| 1 | 7.22205E+11 | Barcode Scanner Module | 50.00 | 50.00 |
| 2 | B00DUW31J2 | Female USB Ports | 6.18 | 12.36 |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | Total | $145.33 |

*Table 18 - Budget*

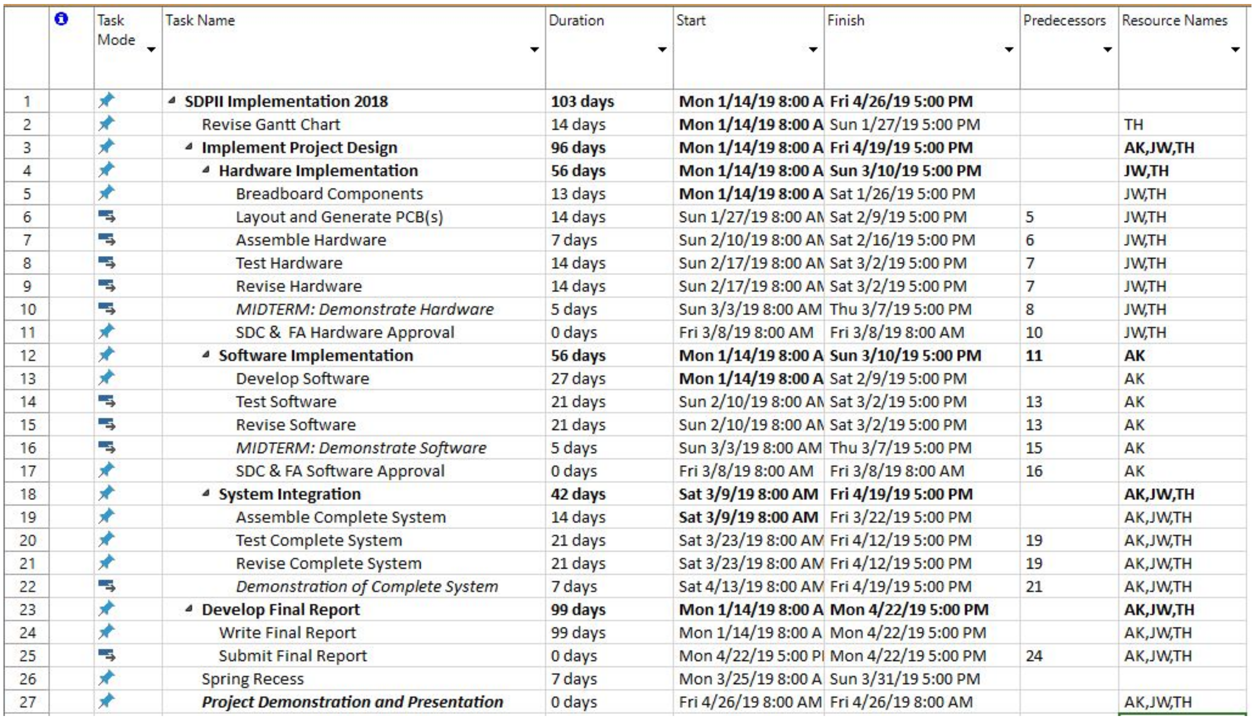
# 5. Project Schedules

## Fall 2018



| | Task Name | Duration | Start | Finish | Predecessor | Resource Names |
|---|---|---|---|---|---|---|
| | **SDP1 Fall 2018** | | | | | |
| | **Project Design** | | | | | |
| | **Preliminary report** | 11 days | Thu 9/6/18 8:00 A | Sun 9/16/18 5:00 | | |
| | Cover page | 11 days | Thu 9/6/18 8:00 A | Sun 9/16/18 5:00 | | AK,JW,TH |
| | T of C, L of T, L of F | 11 days | Thu 9/6/18 8:00 A | Sun 9/16/18 5:00 | | AK,JW,TH |
| | Need | 11 days | Thu 9/6/18 8:00 A | Sun 9/16/18 5:00 | | AK,JW,TH |
| | Objective | 11 days | Thu 9/6/18 8:00 A | Sun 9/16/18 5:00 | | AK,JW,TH |
| | Background | 11 days | Thu 9/6/18 8:00 A | Sun 9/16/18 5:00 | | AK,JW,TH |
| | Marketing Requirements | 11 days | Thu 9/6/18 8:00 A | Sun 9/16/18 5:00 | | AK,JW,TH |
| | Objective Tree | 11 days | Thu 9/6/18 8:00 A | Sun 9/16/18 5:00 | | AK |
| | **Block Diagrams Level 0, 1, ... w/ FR tables** | **11 days** | Thu 9/6/18 8:00 A | Sun 9/16/18 5:00 | | **AK,JW,TH** |
| | Hardware modules (JW) | 11 days | Thu 9/6/18 8:00 A | Sun 9/16/18 5:00 | | JW |
| | Software modules (AK) | 11 days | Thu 9/6/18 8:00 A | Sun 9/16/18 5:00 | | AK |
| | Mechanical Sketch | 11 days | Thu 9/6/18 8:00 A | Sun 9/16/18 5:00 | | TH |
| | Team information | 11 days | Thu 9/6/18 8:00 A | Sun 9/16/18 5:00 | | AK,JW,TH |
| | References | 11 days | Thu 9/6/18 8:00 A | Sun 9/16/18 5:00 | | AK,JW,TH |
| | Preliminary Parts Request Form | 11 days | Thu 9/6/18 8:00 A | Sun 9/16/18 5:00 | | AK,JW,TH |
| | **Midterm Report** | **35 days** | **Thu 9/6/18 8:00 A** | **Wed 10/10/18 5:0** | | |
| | Design Requirements Specification | 14 days | Mon 9/17/18 8:00 | Sun 9/30/18 5:00 | | AK,JW,TH |
| | Midterm Design Gantt Chart | 14 days | Mon 9/17/18 8:00 | Sun 9/30/18 5:00 | | TH |
| | **Design Calculations** | **24 days** | **Mon 9/17/18 8:00** | **Wed 10/10/18 5:0** | | |
| | **Electrical Calculations** | **24 days** | **Mon 9/17/18 8:00** | **Wed 10/10/18 5:0** | | **JW,TH** |
| | Communication | 24 days | Mon 9/17/18 8:00 | Wed 10/10/18 5:0 | | JW,TH |
| | Computing | 24 days | Mon 9/17/18 8:00 | Wed 10/10/18 5:0 | | |
| | Control Systems | 24 days | Mon 9/17/18 8:00 | Wed 10/10/18 5:0 | | JW,TH |
| | Power, Voltage, Current | 24 days | Mon 9/17/18 8:00 | Wed 10/10/18 5:0 | | JW,TH |
| | Electromagnetic Radiation | 24 days | Mon 9/17/18 8:00 | Wed 10/10/18 5:0 | | |
| | Thermal | 24 days | Mon 9/17/18 8:00 | Wed 10/10/18 5:0 | | |
| | **Mechanical Calculations** | **24 days** | **Mon 9/17/18 8:00** | **Wed 10/10/18 5:0** | | |
| | Structual Considerations | 24 days | Mon 9/17/18 8:00 | Wed 10/10/18 5:0 | | AK,JW,TH |
| | System Dynamics | 24 days | Mon 9/17/18 8:00 | Wed 10/10/18 5:0 | | AK,JW,TH |
| | **Block Diagrams Level 2 w/ FR tables & ToO** | **7 days** | **Mon 9/17/18 8:00** | **Sun 9/23/18 5:00** | | **AK,JW,TH** |
| | Hardware modules (JW) | 7 days | Mon 9/17/18 8:00 | Sun 9/23/18 5:00 | | JW |
| | Software modules (AK) | 7 days | Mon 9/17/18 8:00 | Sun 9/23/18 5:00 | | AK |
| | **Block Diagrams Level 3 w/ FR tables & ToO** | **7 days** | **Mon 9/24/18 8:00** | **Sun 9/30/18 5:00** | | |
| | Hardware modules (identify designer) | 7 days | Mon 9/24/18 8:00 | Sun 9/30/18 5:00 | | |
| | Software modules (identify designer) | 7 days | Mon 9/24/18 8:00 | Sun 9/30/18 5:00 | | |
| | **Block Diagrams Level N+1 w/ FR tables & ToO** | **10 days** | **Mon 10/1/18 8:00** | **Wed 10/10/18 5:0** | | |
| | Hardware modules (identify designer) | 10 days | Mon 10/1/18 8:00 | Wed 10/10/18 5:0 | | |
| | Software modules (identify designer) | 10 days | Mon 10/1/18 8:00 | Wed 10/10/18 5:0 | | |
| | Midterm Design Presentations Part 1 | 1 day | Thu 10/11/18 8:00 | Thu 10/11/18 5:00 | | AK,JW,TH |
| | Midterm Design Presentations Part 2 | 1 day | Thu 10/18/18 8:00 | Thu 10/18/18 5:00 | | |
| | Project Poster | 14 days | Mon 10/8/18 8:00 | Sun 10/21/18 5:00 | | |
| | Secondary Parts Request Form | 21 days | Mon 9/17/18 8:00 | Sun 10/7/18 5:00 | | |
| | **Final Design Report** | **52 days** | **Mon 10/8/18 8:00** | **Wed 11/28/18 5:0** | | **AK,JW,TH** |
| | Abstract | 52 days | Mon 10/8/18 8:00 | Wed 11/28/18 5:0 | | AK |
| | **Software Design** | **31 days** | **Mon 10/8/18 8:00** | **Wed 11/7/18 5:00** | | **AK** |
| | **Modules 1...n** | **31 days** | **Mon 10/8/18 8:00** | **Wed 11/7/18 5:00** | | **AK** |
| | Psuedo Code | 31 days | Mon 10/8/18 8:00 | Wed 11/7/18 5:00 | | AK |
| | **Hardware Design** | **31 days** | **Mon 10/8/18 8:00** | **Wed 11/7/18 5:00** | | **JW,TH** |
| | **Modules 1...n** | **31 days** | **Mon 10/8/18 8:00** | **Wed 11/7/18 5:00** | | **JW,TH** |
| | Simulations | 31 days | Mon 10/8/18 8:00 | Wed 11/7/18 5:00 | | JW,TH |
| | Schematics | 31 days | Mon 10/8/18 8:00 | Wed 11/7/18 5:00 | | JW,TH |
| | **Parts Lists** | **52 days** | **Mon 10/8/18 8:00** | **Wed 11/28/18 5:0** | | **JW,TH** |
| | Parts list(s) for Schematics | 52 days | Mon 10/8/18 8:00 | Wed 11/28/18 5:0 | | JW |
| | Materials Budget list | 52 days | Mon 10/8/18 8:00 | Wed 11/28/18 5:0 | | JW |
| | Proposed Implementation Gantt Chart | 52 days | Mon 10/8/18 8:00 | Wed 11/28/18 5:0 | | TH |
| | Conclusions and Recommendations | 52 days | Mon 10/8/18 8:00 | Wed 11/28/18 5:0 | | TH |
| | Final Design Presentations Part 1 | 1 day | Thu 11/8/18 8:00 | Thu 11/8/18 5:00 | | |
| | Final Design Presentations Part 2 | 1 day | Thu 11/15/18 8:00 | Thu 11/15/18 5:00 | | |
| | Secondary Parts Request Form | 14 days | Thu 10/4/18 8:00 | Wed 10/17/18 5:0 | | |
| | Final Parts Request Form | 56 days | Mon 10/8/18 8:00 | Sun 12/2/18 5:00 | | AK,JW,TH |
| | | | | | | AK,JW,TH |

## Spring 2018



| | ℹ | Task Mode | Task Name | Duration | Start | Finish | Predecessors | Resource Names |
|---|---|---|---|---|---|---|---|---|
| 1 | | 📌 | **◢ SDPII Implementation 2018** | **103 days** | **Mon 1/14/19 8:00 A** | **Fri 4/26/19 5:00 PM** | | |
| 2 | | 📌 | Revise Gantt Chart | 14 days | **Mon 1/14/19 8:00 A** | Sun 1/27/19 5:00 PM | | TH |
| 3 | | 📌 | **◢ Implement Project Design** | **96 days** | **Mon 1/14/19 8:00 A** | **Fri 4/19/19 5:00 PM** | | **AK,JW,TH** |
| 4 | | 📌 | **◢ Hardware Implementation** | **56 days** | **Mon 1/14/19 8:00 A** | **Sun 3/10/19 5:00 PM** | | **JW,TH** |
| 5 | | 📌 | Breadboard Components | 13 days | **Mon 1/14/19 8:00 A** | Sat 1/26/19 5:00 PM | | JW,TH |
| 6 | | ⇒ | Layout and Generate PCB(s) | 14 days | Sun 1/27/19 8:00 AI | Sat 2/9/19 5:00 PM | 5 | JW,TH |
| 7 | | ⇒ | Assemble Hardware | 7 days | Sun 2/10/19 8:00 AI | Sat 2/16/19 5:00 PM | 6 | JW,TH |
| 8 | | ⇒ | Test Hardware | 14 days | Sun 2/17/19 8:00 AI | Sat 3/2/19 5:00 PM | 7 | JW,TH |
| 9 | | ⇒ | Revise Hardware | 14 days | Sun 2/17/19 8:00 AI | Sat 3/2/19 5:00 PM | 7 | JW,TH |
| 10 | | ⇒ | *MIDTERM: Demonstrate Hardware* | 5 days | Sun 3/3/19 8:00 AM | Thu 3/7/19 5:00 PM | 8 | JW,TH |
| 11 | | 📌 | SDC & FA Hardware Approval | 0 days | Fri 3/8/19 8:00 AM | Fri 3/8/19 8:00 AM | 10 | JW,TH |
| 12 | | 📌 | **◢ Software Implementation** | **56 days** | **Mon 1/14/19 8:00 A** | **Sun 3/10/19 5:00 PM** | **11** | **AK** |
| 13 | | 📌 | Develop Software | 27 days | **Mon 1/14/19 8:00 A** | Sat 2/9/19 5:00 PM | | AK |
| 14 | | ⇒ | Test Software | 21 days | Sun 2/10/19 8:00 AI | Sat 3/2/19 5:00 PM | 13 | AK |
| 15 | | ⇒ | Revise Software | 21 days | Sun 2/10/19 8:00 AI | Sat 3/2/19 5:00 PM | 13 | AK |
| 16 | | ⇒ | *MIDTERM: Demonstrate Software* | 5 days | Sun 3/3/19 8:00 AM | Thu 3/7/19 5:00 PM | 15 | AK |
| 17 | | 📌 | SDC & FA Software Approval | 0 days | Fri 3/8/19 8:00 AM | Fri 3/8/19 8:00 AM | 16 | AK |
| 18 | | 📌 | **◢ System Integration** | **42 days** | **Sat 3/9/19 8:00 AM** | **Fri 4/19/19 5:00 PM** | | **AK,JW,TH** |
| 19 | | 📌 | Assemble Complete System | 14 days | **Sat 3/9/19 8:00 AM** | Fri 3/22/19 5:00 PM | | AK,JW,TH |
| 20 | | 📌 | Test Complete System | 21 days | Sat 3/23/19 8:00 AM | Fri 4/12/19 5:00 PM | 19 | AK,JW,TH |
| 21 | | 📌 | Revise Complete System | 21 days | Sat 3/23/19 8:00 AM | Fri 4/12/19 5:00 PM | 19 | AK,JW,TH |
| 22 | | ⇒ | *Demonstration of Complete System* | 7 days | Sat 4/13/19 8:00 AM | Fri 4/19/19 5:00 PM | 21 | AK,JW,TH |
| 23 | | 📌 | **◢ Develop Final Report** | **99 days** | **Mon 1/14/19 8:00 A** | **Mon 4/22/19 5:00 PM** | | **AK,JW,TH** |
| 24 | | 📌 | Write Final Report | 99 days | Mon 1/14/19 8:00 A | Mon 4/22/19 5:00 PM | | AK,JW,TH |
| 25 | | ⇒ | Submit Final Report | 0 days | Mon 4/22/19 5:00 PI | Mon 4/22/19 5:00 PM | 24 | AK,JW,TH |
| 26 | | 📌 | Spring Recess | 7 days | Mon 3/25/19 8:00 A | Sun 3/31/19 5:00 PM | | |
| 27 | | 📌 | ***Project Demonstration and Presentation*** | 0 days | Fri 4/26/19 8:00 AM | Fri 4/26/19 8:00 AM | | AK,JW,TH |

## 6. Design Team Information

Andrey Kadoutchek, Computer Engineering, Project Manager, Software Lead

Teddy Helton, Electrical Engineering, Archivist

Jacob Wasson, Electrical Engineering, Hardware Lead

## 7. Conclusions and Recommendations

The project as a whole came with a decent degree of success. A barcode that was entered into the database was able to be read through the barcode scanner and delivered to a database to be checked against ordered packages previously entered. The database was able to send a signal through wifi and communicate with the wifi chip to trigger the microcontroller to send various outputs depending on the databases assessment of the scanned barcode. PIR integration into the system was achieved as well so that the user would know if movement was detected in the garage at any point during the process.

One challenge faced during the development of our subsystems was utilizing a sensor to be used for detecting the state of the garage door. Our system is currently unable to detect if the garage door is open or closed. Also we were unable to implement a method of notifying the user if a malfunction occurred or if the garage door was already open at the time of delivery. This directly correlates to the detection of the garage door and whether it was open or closed.

One recommendation for a future implementation of the Smart Garage Opener system would be to remove the barcode scanner all together and develop a mobile application that delivery personnel would use at the time of delivery to scan the tracking label with. The barcode data could be decoded in the cloud (similar to the current implementation) and the customers garage door could be controlled over the air as long as the garage door is connected to a secure network.

From a team dynamic perspective it is recommended that each individual focus on their respective subsystem but not be afraid to assist one another when someone is falling behind schedule. As a team it is important to shift work efforts when required and not feel like one person alone is responsible for the failure or success of the project. It is recommended that future students work together loosely on each subsystem instead of fully depending on one person for each. Another recommendation is the attention to original project requirements and focusing on those without being sidetracked by features that were not originally intended or designed for.

## 8. References

[1] Elizabeth, Weise and TODAY USA. "Beware of Porch Pirates." *USA Today*, n.d. EBSCO*host*, ezproxy.uakron.edu:2048/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=J0E313238219317&site=ehost-live.

[2] Goldwasser, Sam. "Barcode (Upc) Scanners." *Poptronics*, vol. 2, no. 12, Dec. 2001, p. 55.EBSCO*host*, ezproxy.uakron.edu:2048/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=5475405&site=ehost-live.

[3] Churi, Advait, Anirudh Bhat, Ruchir Mohite, and Prathamesh P. Churi. "E-zip: An Electronic Lock for Secured System." *2016 IEEE International Conference on Advances in Electronics, Communication and Computer Technology (ICAECCT)*, 2016. doi:10.1109/icaecct.2016.7942553.

[4] Wei, Jin, Ilene Hollin, and Stan Kachnowski. "A Review of the Use of Mobile Phone Text Messaging in Clinical and Healthy Behaviour Interventions." *Journal of Telemedicine and Telecare*17, no. 1 (2011): 41-48. doi:10.1258/jtt.2010.100322.

[5] "Cam locks go modular." *Machine Design* 79, no. 10 (May 24, 2007): 23. *Academic Search Complete*, EBSCO*host* (accessed April 29, 2018).

[6] Koprda, Štefan and Martin Magdin. "Implementation of Innovative Technologies in the Fields of Electronic Locks." *Telkomnika*, vol. 14, no. 4, Dec. 2016, pp. 1329-1337. EBSCO*host*, doi:10.12928/TELKOMNIKA.v14i4.4184.

[7] Jiang, Shuai. Package Receiving Systems and Methods. Jiang; Shuai (San Mateo, CA), assignee. Patent. N.d. Print.

[8] Benini, David. Biometric Identification and Verification. AWARE, INC. (Bedford, MA), assignee. Patent 9646197. N.d. Print.

## 9. Appendices

PIR Sensor
https://www.parallax.com/sites/default/files/downloads/555-28027-PIR-Sensor-Prodcut-Doc-v2.2.pdf

Distance Sensor
https://www.st.com/content/ccc/resource/technical/document/datasheet/group3/b2/1e/33/77/c6/92/47/6b/DM00279086/files/DM00279086.pdf/jcr:content/translations/en.DM00279086.pdf

RGB LED
http://www.kingbrightusa.com/images/catalog/SPEC/WP154A4SUREQBFZGC.pdf

Connectors
https://www.molex.com/pdm_docs/sd/022232061_sd.pdf

https://www.molex.com/pdm_docs/ps/PS-10-07-001.pdf

Power Jack Connector
https://www.sparkfun.com/datasheets/Prototyping/Barrel-Connector-PJ-202A.pdf

Microcontroller
http://ww1.microchip.com/downloads/en/DeviceDoc/PIC24FJ1024GA610-GB610-Family-Data-Sheet-DS30010074F.pdf

Barcode Scanner Module

https://www.amazon.com/Barcode-Scanner-Module-Codes-Reader/dp/B07K2C9JP6/ref=sr_1_3?ie=UTF8&qid=1544208392&sr=8-3&keywords=nouii+barcode+scanner+module+1D%2F2D+codes+reader

Barcode Scanner
https://www.amazon.com/Yumite-Portable-Barcode-Scanner-Equipment/dp/B01IP3XICA

Wifi Module
https://nurdspace.nl/images/e/e0/ESP8266_Specifications_English.pdf