

Pi Car Security Monitoring System with Cloud Storage and Mobile Application

1



Friday December 12, 2014

Final Presentation
CPET/ECET 491

Trent Barcus
Nolan Gilvin
Chris Vance

Presentation Outline

2

- Introduction
- Problem Statement/Solution
- Top Level Description
- System Design
- Hardware Design
- Software Design
- Integration
- Conclusion

Introduction

3

- Vehicle owners need to monitor their vehicles
- Security and current state of vehicle
- PCSMS gathers and displays this data
- Hardware device, cloud storage, and mobile application
- Locked/unlocked state of the car
- Open/closed state of a car's doors
- Time stamp

Problem/Solution

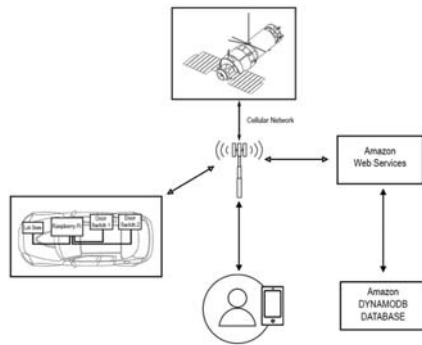
4

- 1550 vehicle related crimes involving vehicle or vehicular property theft*
- January 1st of 2014 Fort Wayne, IN
- Enhancing a car's security
- Android smartphone app
- Monitor the condition and security of automobiles

*<http://www.fortwayne.com/section/CRIME>. [Accessed 6 December 2014]

System Design (Top-Level Description)

Pi Car System Top Level Architecture



5

- Main Modules
 - Hardware System: Raspberry Pi and analog circuit
 - Web Service: Amazon Web Services DynamoDB database
 - Mobile App: Native Android Mobile Application

System Requirements

6

- The device shall collect data from sensors installed on the car's doors
- The device will store the collected data locally
- The device shall use a GSM modem
- The GSM modem shall transmit collected data to web servers
- The collected data shall be stored to a database

System Requirements (Continued)

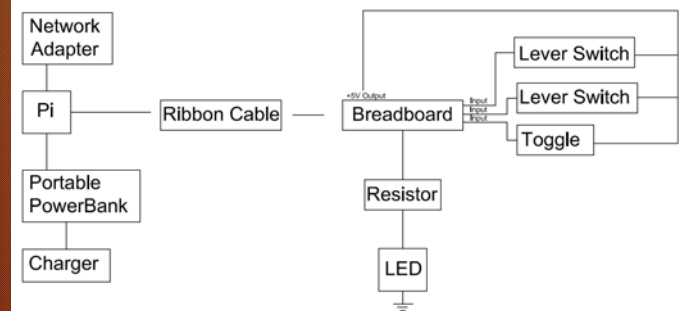
7

- The device shall be powered from the vehicle
- The device shall weigh less than 5 pounds
- The device shall be capable of being discretely mounted
- The device shall operate in -32 to 110 degrees Fahrenheit
- The device shall survive vibrations from driving on public roads

Hardware Design (System Overview)

8

- Raspberry Pi
- Network Adapter
- Portable Powerbank
- Breadboard
- Switches



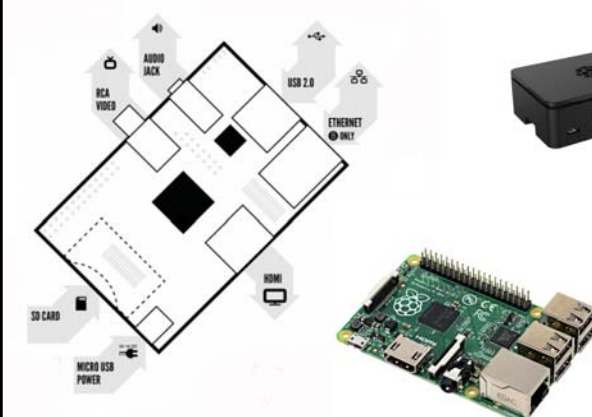
Hardware Design (Parts List)

9

- 1 x Raspberry Pi Model B+ (B Plus) 512 MB
- 1 x Raspberry Pi B+ Case
- 1 x SanDisk - Ultra 8GB SDHC UHS-I Class 10 Memory Card
- 1 x RadioShack 1250mAh Slim-Style Portable Power Bank
- 1 x CanaKit WiFi Adapter/Dongle (Ralink RT5370 chipset)
- 1 x Breadboard
- 1 x GPIO Ribbon Cable
- 1 x GPIO to Breadboard Interface Board
- 1 x Green LED
- 1 x 220 Ohm Resistor
- 1 x SPST Toggle Switch
- 2 x Micro Switch

Hardware Design (Raspberry Pi)

10



The diagram illustrates the hardware design for a Raspberry Pi project. It features a central schematic of the Raspberry Pi board with various components labeled: AUDIO JACK, RCA VIDEO, SD CARD, MICRO USB POWER, HDMI, USB 2.0, and ETHERNET. To the right, there are two images: a black Raspberry Pi B+ case and a Raspberry Pi B+ board. Below the images, a list of specifications is provided.

- 512MB System Memory
- 40 Pin Header for General-Purpose Input/Output (GPIO)
- 5V Power Via Micro USB Port
- 600 mA to 1.8A @ 5V Power Consumption
- 85mm X 56mm

Hardware Design (Pi Components)

11

RadioShack 1250mAh Slim-Style
Portable Power Bank



WiFi Adapter/Dongle
(Ralink RT5370 chipset)



SanDisk - Ultra 8GB SDHC
UHS-I Class 10 Memory Card



Hardware Design(Analog Components)

12

GPIO to Breadboard Interface
Board



GPIO Ribbon Cable



Breadboard



Hardware Design(Analog Components) 13

SPST Toggle Switch



Micro Switch



220 Ohm Resistor

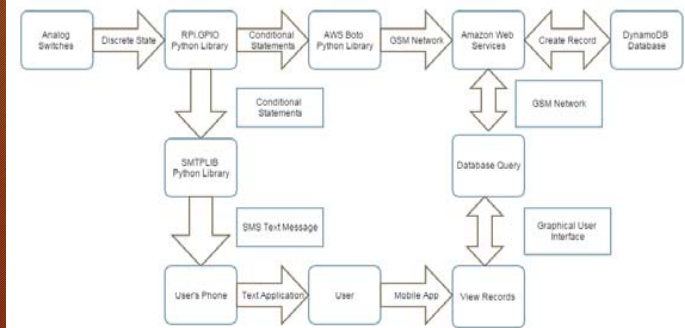


Green LED



Software Design (Overview) 14

- RPi.GPIO
- Boto
- SMTPLIB
- DynamoDB
- Mobile App



```

1 #import all boto libraries
2 import boto
3 from boto.dynamodb2.table import Table
4
5 #import smtp protocols to send email alerts
6 import smtplib
7
8 #import the Raspberry Pi GPIO library to work with GPIO
9 import RPi.GPIO as GPIO
10
11 #allows us to count the passage of time
12 import time
13 import datetime
14
15 #sets the GPIO numbering system to the board pin numbers
16 GPIO.setmode(GPIO.BOARD)
17
18 #sets the output pin to pin 11
19 GPIO.setup(11, GPIO.OUT)
20
21 #sets L input pin to pin 13 and internally sets it as a pull down resistor
22 GPIO.setup(13, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
23
24 #sets D1 input pin to pin 22 and internally sets it as a pull down resistor
25 GPIO.setup(22, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
26
27 #sets D2 input pin to pin 15 and internally sets it as a pull down resistor
28 GPIO.setup(15, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
29
30 #set all variables for messages
31 S1 = 'Car Locked / Door(s) Closed at '
32 S2 = 'Car Locked / Door(s) Open at '
33 S3 = 'Car Unlocked / Door(s) Closed at '
34 S4 = 'Car Unlocked / Door(s) Open at '

```

Code

15

- Python library imports
- GPIO pin configuration
- Message variables
- Comments

```

36 #set variables for car door and lock states
37 L = GPIO.input(13)
38 D1 = GPIO.input(22)
39 D2 = GPIO.input(15)
40
41 #set variables used in reporting time
42 reftime = datetime.time(0,0,0)
43 refdate = datetime.date(2014, 1, 23)
44 refdatetime = datetime.datetime.combine(refdate, reftime)
45
46 #set to and from fields for the email alert including the message
47 fromaddr = 'picarsystem@gmail.com'
48 toaddr = '2605643366@vtext.com'
49 msg = 'Your car alarm went off just now'
50
51 #credentials to log into the email account
52 username = 'picarsystem@gmail.com'
53 password = 'beepbeep!'
54
55 #specify the smtp mail server to send through
56 server = smtplib.SMTP('smtp.gmail.com:587')
57
58 #start the mail service and log in
59 server.starttls()
60 server.login(username, password)
61
62 #connect to the Amazon dynamoDB database using the access keys provided
63 conn = boto.connect_dynamodb(aws_access_key_id='AKIAIUKO05F035QX3LFG',aws_secret_access_key=
64
65 #specify table to work with
66 Messages = conn.get_table('Messages')
67

```

Code

16

- GPIO variables
- Time variables
- Text variables
- Email variables
- Boto connect
- Messages variable
- Comments


```

68 #returns message based on current state of doors and lock
69 def CreateMessage():
70     now = time.strftime("%s")
71     if(L == 1):
72         if(D1 == 1):
73             if(D2 == 1):
74                 result = S1 + now
75                 return str(result)
76             else:
77                 result = S2 + now
78                 return str(result)
79         else:
80             result = S2 + now
81             return str(result)
82     else:
83         if(D1 == 1):
84             if(D2 == 1):
85                 result = S3 + now
86                 return str(result)
87             else:
88                 result = S4 + now
89                 return str(result)
90         else:
91             result = S4 + now
92             return str(result)
93
94 #returns the RangeKey by subtracting our reference time from the current time.
95 def CreateID():
96     curdatetime = datetime.datetime.today()
97     result = curdatetime - refdatetime
98     return str(int(result.total_seconds()))
99

```

Code

17

- Create message method
- Create ID method
- Comments

```

100 #look for state changes
101 while True:
102     GPIO.output(11,GPIO.HIGH)
103     time.sleep(1)
104     #check lock state
105     if(L != GPIO.input(13)):
106         L = GPIO.input(13)
107         D1 = GPIO.input(22)
108         D2 = GPIO.input(15)
109         item_data = {
110             'Message': CreateMessage()
111         }
112         item = Messages.new_item(hash_key=CreateID(),
113                                 range_key=CreateID(),
114                                 attrs=item_data)
115         item.put()
116
117 #check to see if the condition is worthy of an alert
118 if(L == 1):
119     if(D1 == 0):
120         server.sendmail(fromaddr, toaddr, msg)
121     elif(D2 == 0):
122         server.sendmail(fromaddr, toaddr, msg)
123
124 #check door 1
125 elif(D1 != GPIO.input(22)):
126     L = GPIO.input(13)
127     D1 = GPIO.input(22)
128     D2 = GPIO.input(15)
129     item_data = {
130         'Message': CreateMessage()
131     }
132     item = Messages.new_item(hash_key=CreateID(),
133                             range_key=CreateID(),
134                             attrs=item_data)
135     item.put()
136

```

Code

18

- Constant while loop
- Set LED
- Monitor states
- Gather data
- Create database record
- Send message
- Comments

```

134 #check to see if the condition is worthy of an alert
135 if(L == 1):
136     if(D1 == 0):
137         server.sendmail(fromaddr, toaddr, emsg)
138     elif(D2 == 0):
139         server.sendmail(fromaddr, toaddr, emsg)
140
141 #check door 2
142 elif(D2 != GPIO.input(15)):
143     L = GPIO.input(13)
144     D1 = GPIO.input(22)
145     D2 = GPIO.input(15)
146     item_data = {
147         'Message': CreateMessage()
148     }
149     item = Messages.new_item(hash_key=CreateID(),
150                             range_key=CreateID(),
151                             attrs=item_data)
152     item.put()
153
154 #check to see if the condition is worthy of an alert
155 if(L == 1):
156     if(D1 == 0):
157         server.sendmail(fromaddr, toaddr, emsg)
158     elif(D2 == 0):
159         server.sendmail(fromaddr, toaddr, emsg)

```

Code

19

- Monitor states
- Gather data
- Create database record
- Send message
- Comments

Software (Database)

20

- HashID (String)
- RangeID (String)
- Message (String)

Amazon DynamoDB Explore Table: Messages

HashID	RangeID	Message
27626644	27626644	Car Locked / Door(s) Open at Mon Dec 8 18:04:04 2014
27116488	27116488	Car Unlocked / Door(s) Open at Tue Dec 2 20:21:29 2014
27620271	27620271	Car Unlocked / Door(s) Closed at Mon Dec 8 18:31:11 2014
27779430	27779430	Car Locked / Door(s) Open at Wed Dec 10 12:30:30 2014
27620054	27620054	Car Unlocked / Door(s) Closed at Mon Dec 8 18:04:14 2014
27788317	27788317	Car Locked / Door(s) Open at Wed Dec 10 14:58:37 2014
27787515	27787515	Car Unlocked / Door(s) Closed at Wed Dec 10 14:45:15 2014
27629216	27629216	Car Locked / Door(s) Closed at Mon Dec 8 18:46:56 2014
27629492	27629492	Car Locked / Door(s) Closed at Mon Dec 8 18:34:42 2014
27626766	27626766	Car Locked / Door(s) Open at Mon Dec 8 18:06:06 2014

```

1<?xml version="1.0" encoding="utf-8"?>
2<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3    android:layout_width="fill_parent"
4    android:layout_height="fill_parent"
5    android:orientation="vertical" >
6    <LinearLayout
7        xmlns:android="http://schemas.android.com/apk/res/android"
8        android:layout_width="fill_parent"
9        android:layout_height="fill_parent"
10       android:padding="10dp"
11       android:gravity="bottom"
12       android:orientation="vertical" >
13        <TextView
14            android:id="@+id/textview"
15            android:layout_width="fill_parent"
16            android:layout_height="fill_parent"
17            android:layout_weight="0.5"
18            android:ellipsize="none"
19            android:textSize="20sp"
20            android:drawablePadding="5dp"
21            android:includeFontPadding="true"
22            android:gravity="center_vertical"
23            android:text="The P1 Car Security Monitoring System (PCMS) contains
24                a combination of hardware and software technologies used for obtaining,
25                storing, and monitoring the physical state of your car. The current
26                open/closed state of the doors on your car as well as your car's
27                locked/unlocked state can be viewed from the convenience of your Android
28                smartphone. Everything is achieved through the implementation of a hardware
29                device, cloud storage, and this mobile application."
30        />
31        <Button
32            android:id="@+id/list_users_btn"
33            android:layout_width="fill_parent"
34            android:layout_height="wrap_content"
35            android:layout_gravity="center_horizontal"
36            android:text="@string/step" />
37    />
38</LinearLayout>

```

Software (Mobile App)

21

```

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    clientManager = new AmazonClientManager(this);

    final Button listUsersBtn = (Button) findViewById(R.id.list_users_btn);
    listUsersBtn.setOnClickListener(new View.OnClickListener() {

        public void onClick(View v) {
            Log.i(TAG, "listUsersBtn clicked.");

            new DynamoDBManagerTask()
                .execute(DynamoDBManagerType.LESS_USERS);
        }
    });
}

```

- Home Screen XML
- Home Screen Java
- PCSMS Explanation
- "View Records" button

```

private class GettersListTask extends AsyncTask<Void, Void, Void> {
    protected void doInBackground(Void... inputs) {
        labels = new ArrayList<String>();
        ArrayList<UserPreference> items = DynamoDBManager.getUsers(list);
        int total = items.size();
        int k;

        for (int n = total; n > 0; n--) {
            for (int i = 0; i < n - 1; i++) {
                k = i;
                int v = Integer.parseInt(items.get(i).getRangeID());
                int w = Integer.parseInt(items.get(i+1).getRangeID());
                if (v < w)
                    swapNumbers(i, k, items);
            }
        }

        for (UserPreference u : items) {
            labels.add(u.getMessage());
        }
        return null;
    }

    private void swapNumbers(int i, int j, ArrayList<UserPreference> items) {
        UserPreference temp;
        temp = items.get(i);
        items.set(i, items.get(j));
        items.set(j, temp);
    }

    protected void onPostExecute(Void result) {
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(UserListActivity.this,
            R.layout.user_list_item, labels);
        listView.setAdapter(adapter);

        listView.setOnItemClickListener(new OnItemClickListener() {
            @Override
            public boolean onItemClick(AdapterView<?> av, View v,
                int pos, long id) {
                return onItemClick(pos, id);
            }
        });

        Toast toast = Toast.makeText(UserListActivity.this,
            "Displaying Records", Toast.LENGTH_LONG);
        toast.show();
    }
}

```

Software (Mobile App)

22

```

1<?xml version="1.0" encoding="utf-8"?>
2<TextView xmlns:android="http://schemas.android.com/apk/res/android"
3    android:layout_width="fill_parent"
4    android:layout_height="fill_parent"
5    android:padding="10dp"
6    android:textSize="16sp" >
7</TextView>

```

- View Records XML
- View Records Java
- GetUserListTask
 - doInBackground
 - SwapNumbers
 - onPostExecute

Software (Mobile App)

23

```

/*
 * Scans the table and returns the records.
 */
public static ArrayList<UserPreference> getUserList() {
    AmazonDynamoDBClient ddb = PICarSecurityMonitoringSystem.clientManager
        .ddb();
    DynamoDBMapper mapper = new DynamoDBMapper(ddb);

    DynamoDBScanExpression scanExpression = new DynamoDBScanExpression();
    try {
        PaginatedScanList<UserPreference> result = mapper.scan(
            UserPreference.class, scanExpression);

        ArrayList<UserPreference> resultList = new ArrayList<UserPreference>();
        for (UserPreference up : result) {
            resultList.add(up);
        }
        return resultList;
    } catch (AmazonServiceException ex) {
        PICarSecurityMonitoringSystem.clientManager
            .wipeCredentialsOnError(ex);
    }

    return null;
}

```

- DynamoDBManager Java
 - getUserList
 - clientManager
 - Mapper
 - Scan table
 - Create list of results
 - wipeCredentials

Integration (Original Fixture)

24



- Original fixture built to contain components and simulate vehicle
- State: "Car Locked / Door(s) Closed"

Integration (Original Fixture)

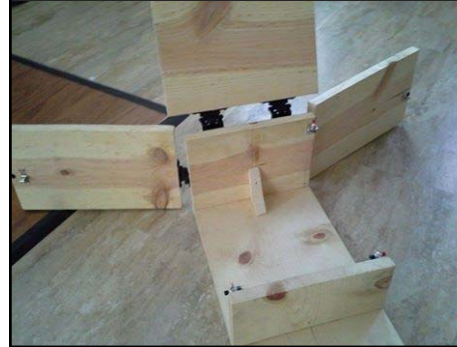
25



- Original fixture built to contain components and simulate vehicle
- State: "Car Locked / Door(s) Open"

Integration (Original Fixture)

26

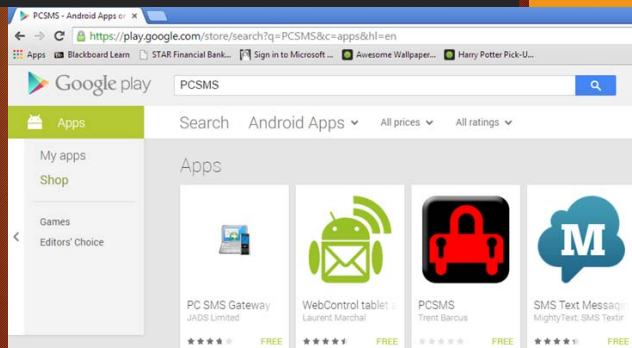


- Original fixture built to contain components and simulate vehicle
- State: "Car Unlocked / Door(s) Open"

Integration (Google Play Store)

27

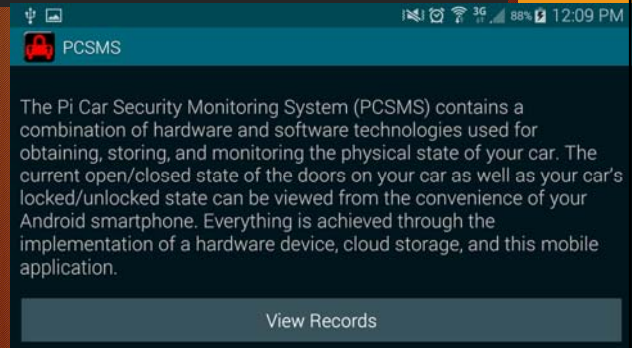
- Google Play Store
- Apps
- PCSMS
- Download



Integration (Mobile App Home)

28

- Logo
- App name
- Description
- "View Records"



Integration (Mobile App Records)

29

- Logo
- App name
- Current state
- Time stamp
- Descending order



The screenshot shows a mobile app interface with a status bar at the top displaying various icons and the time 12:10 PM. Below the status bar is a header with a red car icon and the text 'PCSMS'. The main content is a list of events, each with a description and a timestamp.

Event Description	Timestamp
Car Unlocked / Door(s) Closed	Tue Dec 2 20:21:59 2014
Car Unlocked / Door(s) Open	Tue Dec 2 20:21:28 2014
Car Unlocked / Door(s) Closed	Tue Dec 2 20:21:27 2014
Car Unlocked / Door(s) Open	Tue Dec 2 20:20:46 2014
Car Locked / Door(s) Open	Tue Dec 2 20:20:39 2014
Car Locked / Door(s) Open	Tue Dec 2 20:20:31 2014
Car Locked / Door(s) Closed	Tue Dec 2 20:20:12 2014

Integration (Demo Model)

30



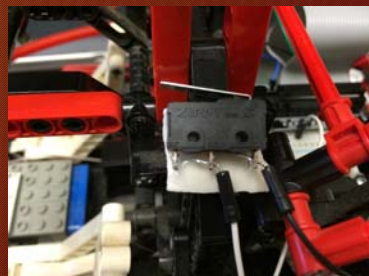
Integration (Switches)

31

SPST Toggle Switch



Micro Switch



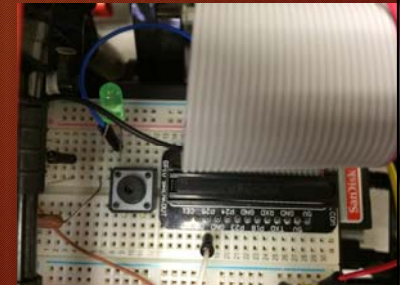
Integration (Finished Product)

32

Model Car with Hardware Device



Hardware Device in Model Car



Conclusion

33

- Project met all requirements
- Additional Feature: SMS text alert
- Additional Feature: Google Play integration
- Additional Feature: Batter backup
- Completed on time and under budget
- Future Enhancement: On Board Diagnostics(OBD-II) integration
- Future Enhancement: Wireless switches
- Future Enhancement: iOS mobile app

Acknowledgements

34

Kenneth Jaeger
Vaughn Novy
Blake Harrison
Caitlin Vance
Jacob Pitcher

