Wireless Cookie Jar
Milestone IV Report

Khanh Nguyen, 311253865 and Kevin Nguyen, 311254039

"Have you ever gotten off the computer and walked all the way to the kitchen for a cookie only to find out that the cookie jar is empty? If only there was a way to check the jar without having to get off the computer..."

Aim

The aim of the "Wireless Cookie Jar" project is to develop a cookie jar that can detect how many cookies are left inside it and send the data to a web application would display the information to the user. Ideally it would provide the user with the number of cookies left in the jar using a load cell and calculations based on the average serving size information.

Objectives

Our goal is to be able to determine if the cookie jar is empty or full. We would then like to be able to determine how many cookies are left in the jar.

With connectivity side of the project, we aim to be able to maintain a constant connection to the internet so that the web application can monitor the cookie jar and its contents.

Our goal for the web based side of the project is to build a simple web applications using Ruby on Rails 3.1. The application will ideally be able receive data from the cookie jar, process it and display it on a website.

Initial Design

The design of the Wireless Cookie Jar system will consist of the following:

1. A cookie jar that is light in weight so that it would not affect the load cell output too much. The lid of the cookie jar will also have its own circuit which will determine when data should be transmitted or not.
2. A load cell circuit to measure the weight of the cookie jar and its contents.
3. An Arduino DUE as the microcontroller that will process and send the measurements between different modules of the cookie jar system.
4. A WiFi Shield that will receive data from the Arduino and transmit the data to the web server via a HTTP request to the Internet. The WiFi shield will connect to a wireless modem with a preset SSID and WPA2 key which is editable in the .ino file.
5. A web server that will run a web application that will be implemented using Ruby on Rails.
6. A PostgreSQL database that will store the data sent from the physical components of the system. The data from this database will be displayed on the web application.

Since the system is connected to the Internet both local and external users will be able to access the web application online.

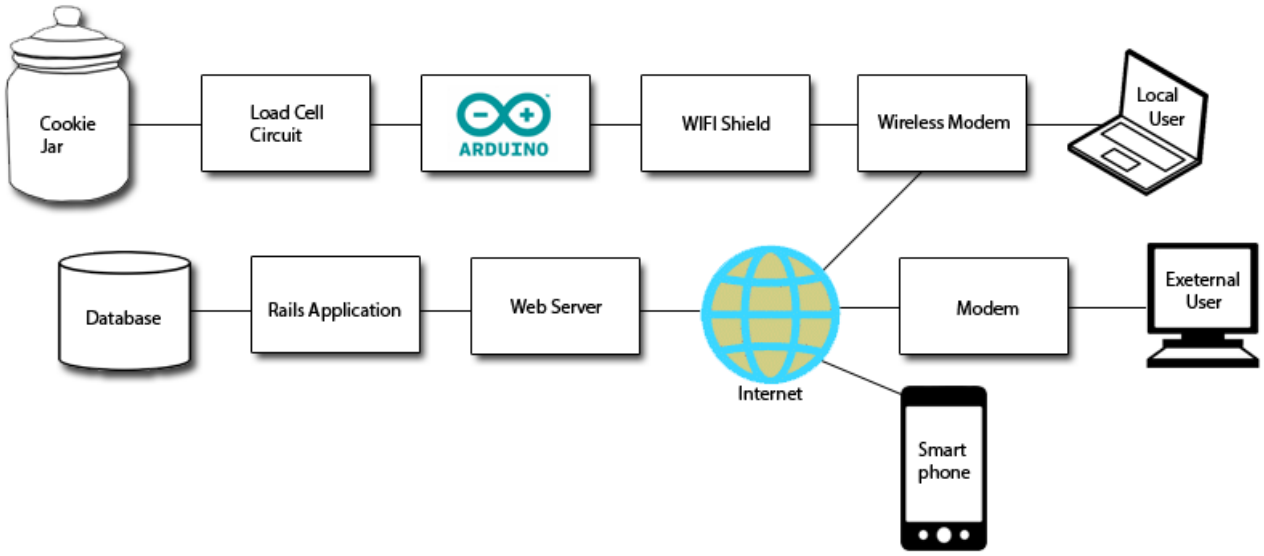

Figure 1: Initial Design Diagram of the Wireless Cookie Jar System

Schematic

The load cell and the instrumentation amplifier(INA126PA) are both connected to the 5v source of the Arduino. The output of the load cell is fed into the instrumentation amplifier which has a gain of Loading..., where Rg is the chosen 68Loading... resistor. This sets G = 1176.47 V/V. The amplified sig nal (pin 6 of INA126PA) is then connected to A0(analog input) of the Arduino for processing. Pin 6 of the INA126PA is also connected to a zener diode with Vz = 3.3v to limit the signal, this will prevent damage to the Arduino A0 pin which can only tolerate a maximum voltage of 3.3v.The Arduino then sends the information over the internet through the WiFi module which is connected to RX1 and TX1 of the Arduino.

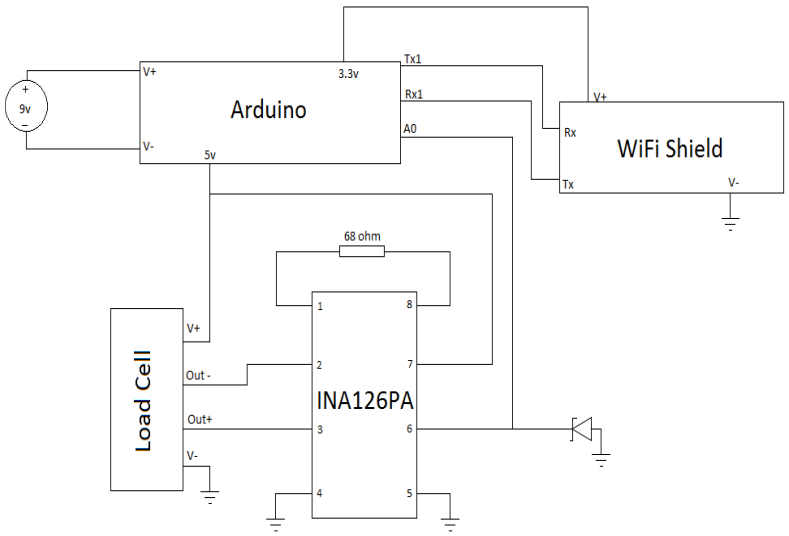

Figure 2: Schematic of the Wireless Cookie Jar

Problems Encountered with the Initial Design

- WiFi shield did not work, we are unsure if this was due the compatibility of the Arduino DUE or if the shield was just faulty.
- Arduino DUE had many compatibility issues and did not have many required libraries available yet.
- We could not deploy a Ruby on Rails application since we did not have the required server and domain to host it on.

So ultimately, we decided to refactor our design and change our approach.

Final Design

The final design of the Wireless Cookie Jar system:

1. A cookie jar that is light in weight so that it would not affect the load cell output too much.
2. The load cell circuit to measure the weight of the cookie jar.
3. The load cell circuit sends an analog to the Arduino UNO to process.
4. The Seeedstudio Bluetooth Shield stacked on the Arduino UNO allows users to connect their Android device.
5. With an Android application developed by the group, users can communicate with the Arduino and tell it to send data about the current weight of the cookie jar.
6. That information is then send to a PHP script via a HTTP POST response with the cookie weight as JSON (Javascript Object Notation).
7. The JSON is caught by the post_level.php and the cookie weight is recorded in a PostgreSQL database.
8. The main page (index.php) then, takes information from the database and displays it in a form of a graphic and dynamic chart.

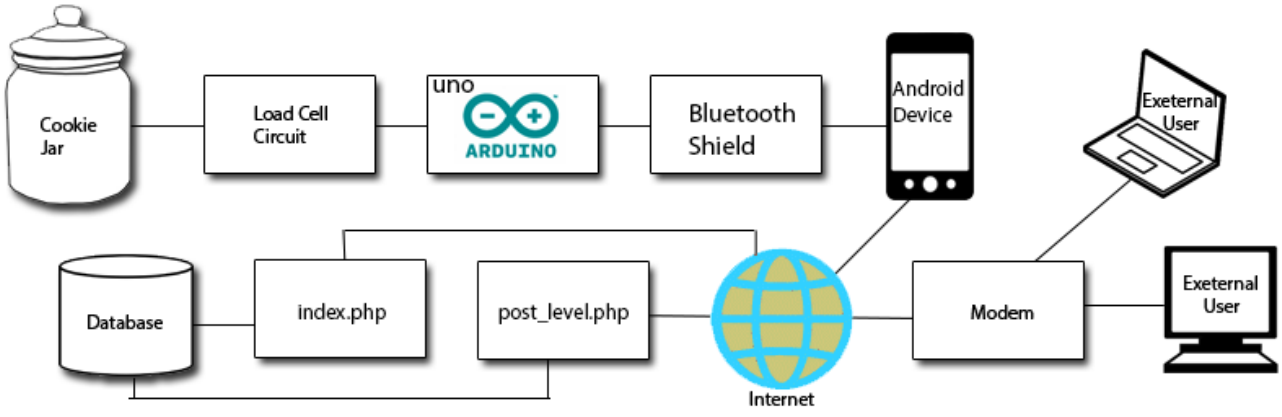

Figure 3: Final Design Diagram of the Wireless Cookie Jar System

Extension - The Cookie Dispenser

The design of the cookie dispenser was made for simplicity and reliability. The cookies are stacked up on top of one another with an opening at one end for the cookies to be pushed out by the servo arm and with a small hole directly on the other side of the jar for the arm to be inserted into. The basic mechanics behind the dispensing unit is that the arm pushes out only the bottom cookie. Once the arm retracts to its original position, the cookie stack falls down and another cookie is in position and ready to be dispensed again.

The arm comprises of two pieces of dowels, with a loose mortise and tenon joint to allow it act as an elbow(*figure 5*). This allows the bottom arm to almost stay parallel with the bottom of the jar while the servo rotates. The arm turns the rotational movement of the servo into a linear motion that is able to push the cookies out.

The servo is controlled via the Arduino Uno, once given the command to dispense, it will rotate a certain about of degrees clockwise then rotate back the same amount of degrees in the counterclockwise direction. This will cause the arm to push forward and back.

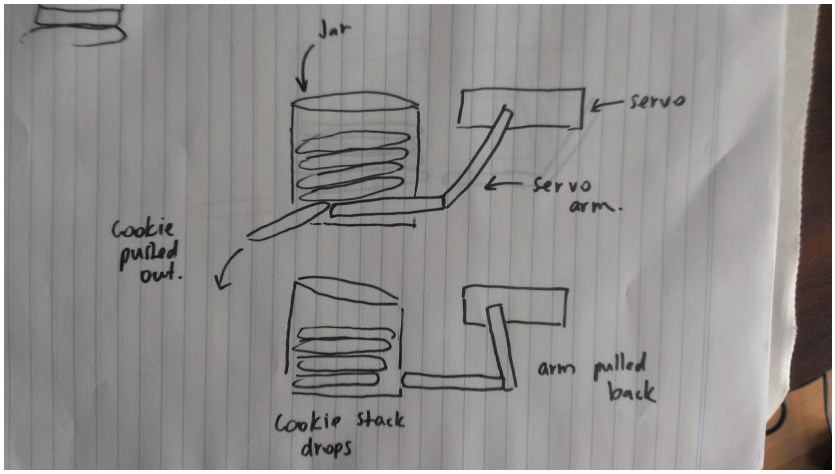

Figure 4: Cookie Jar Dispensing Unit

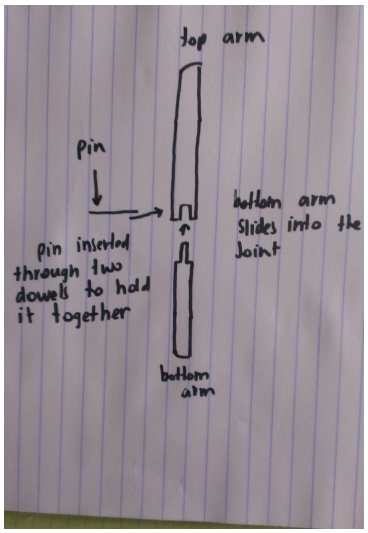

Figure 5: The arm of the servo

Final Schematic

The changes we made to the schematic was changing the WiFi Shield for a Bluetooth shield and adding in a servo. The servo runs of the 5v supply of the Arduino and the signal line is plugged into digital pin 9. We also removed the zener diode since we no longer needed to limit the input voltage to 3.3v because we changed the Due to an Uno board instead, which can handle a 5v input.

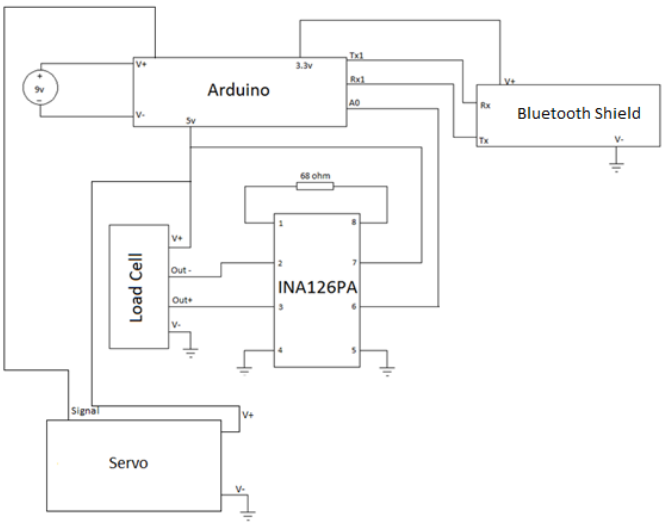

Figure 6: Final schematic of the Wireless Cookie Jar

Hardware Design

Load Cell

The load cell we used was a commercially available digital scale that we took apart. We figured this would be easier because the load cell would already have a proper housing for it. The signal from the load cell is then fed into the amplifier (INA126PA). Because the amplifier is being powered by a 5v supply and the Arduino Uno is capable of handling 5v inputs, we did not need to limit the output of the amplifier since the output should not exceed the 5v supply.

Code Design

Arduino Code

Load Cell

After reading in the analog signal from the load cell, we had to map the values of the analog signal to correspond to the percentage of the jars capacity. Slowly applying the load, we recorded each reading to obtain a table of values corresponding to load applied. We then made a function that would return integers between 0-9 based on the results to be interpreted as percentage of jar filled.

Dispenser

We used the Servo library to control the dispensing unit for the cookie jar. The servo library allows us to control precisely the degree of movement of the servo either in milliseconds or degrees. We chose to control the servo using degrees as this would be easier to calibrate with.

Bluetooth

For the bluetooth shield, we used the example Slave.ino file from the SeeedStudio wiki page.

When Bluetooth serial is available it will check the character that was received that if it was correct and then call the dispense function. After the dispensing of the cookie is completed, it will then immediately invoke the load cell function and send the current cookie level to the connected Bluetooth device.

Android Application Code

Bluetooth Connectivity

For the Wireless Cookie Application for Android, we used the sample Bluetooth Chat app that is provided in the Android SDK. We replaced the chatting functionality with a button that would send a specific character to the Arduino which would then respond accordingly.

Sending data to the web server

After the Arduino sends information about the current level of the cookie jar, the Android application will then proceed to send that information to the web server. This is done by putting the data into a JSON (Javascript Object Notation) object and then sending it as a parameter in a HTTP POST request to the post_level.php script that is hosted on the university's Apache server.

PHP Code

Receiving and storing Data from Android Device (post_level.php)

Upon loading the script will check for any POST parameters called 'cookie_level'. If the successfully finds such a parameter it will then validate it and invoke a prepared SQL statement to insert this data into the PostgreSQL table along with the current time stamp.

PostgreSQL

For the PostgreSQL database, we created a table called CookieJar which hold the current level of the cookie (cookie_level) and the timestamp of when the information is inputted (time_posted).

```
CREATE TABLE CookieJar (
  cookie_id SERIAL PRIMARY KEY,
  cookie_level INT,
  time_posted TIMESTAMP
);
```

Figure n: PostgreSQL function to create the CookieJar table

Displaying data (index.php)

When the user arrives on the main page of the application, the script will retrieve information from the database table and display it in the form of a cookie and percentage bar. If the cookie jar is not full then the picture of the cookie would have bites taken out of it.

An extension that was done for this webapp was allow the user to somehow monitor how many cookies they have eaten. This was implemented with some simple Javascript (Highcharts.js) using the data of the cookie level and the timestamp.

Testing Methodology

Load Cell Testing

We tested the load cell circuit by recording the analog output to the number of cookies. Using the nutrition information from the packaging of the cookies, we found that each cookie weigh approximately 15g.

Dispenser Testing

We tested the dispenser initially with a push button until the Android application was ready. When the application was ready, we made sure that it persisted with the same functionality as the push button.

Bluetooth Testing

We tested the bluetooth connectivity using a variety of applications that were available on the Play Store. Essentially those apps were all Bluetooth Chat so we just send characters to see the Arduino would be able to receive it and read it.

Software Testing

Testing the Android application and web application simply consisted of sending dummy data from the phone to the web server using POST requests and curl on a terminal. We checked and confirmed that the data was accurately being transferred between the devices.

Results

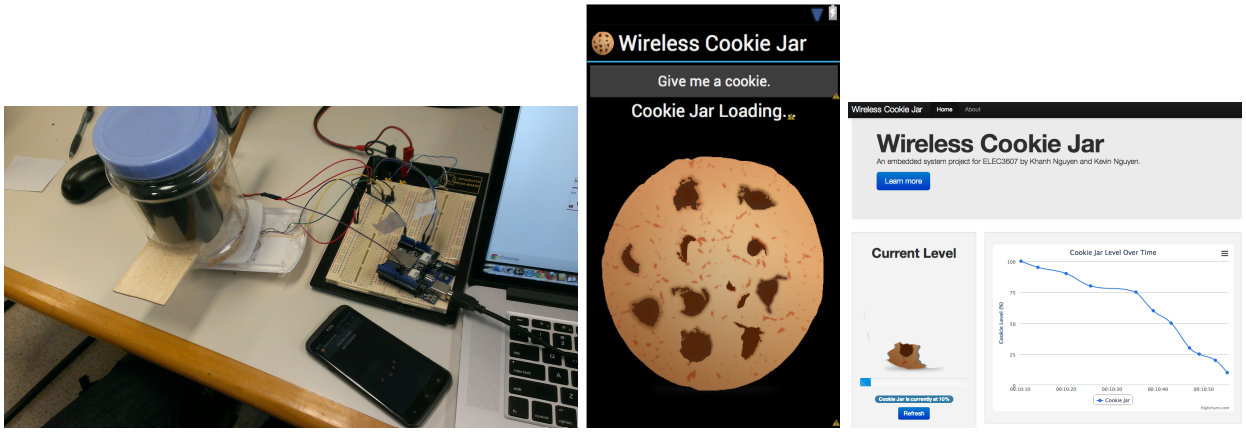

Figure 7: Final products - Wireless Cookie Jar, Android App, Web app

We were able to complete the assembly and development of the Wireless Cookie Jar in time, however there were a few limitations and bugs leftover from the rushed development.

Limitations:

- Accuracy of load cell is poor due since the calibration is incorrect.
- Bluetooth Shield tends to stop working randomly and requires the board to be restarted.

- Servo occasionally stopped working and caused the Arduino to restart since it drawing too much current.

Possible Improvements and Extensions

Improvements

Load Cell

One thing we failed to do is to properly calibrate the load cell circuit to accurately measure the amount of cookies that are present in the jar. We could have implemented this feature in a different way such as using sensors to detect the height of the cookie level rather than analysing the weight of jar, this would have given a much more accurate reading.

We should have polled for a change in weight and then called an interrupt, rather than just checking manually when the user requested it.

Dispenser

The servo was clearly drawing too much current from the Arduino board and caused it restart after a few presses of the dispense button. An improvement for this would be using a transistor or relay to control the current flow, or use a separate power source to ensure that it does not affect the microcontroller.

Research

We came across several issues that could have been easily avoided if we had did thorough research before he commenced building the prototype.

For example, the WiFi was not compatible with the Arduino DUE which required us to move to the Arduino UNO. Even then, the WiFi refused to work due to the lack of knowledge we had about how it worked and the poor documentation. If we had done proper research on the product we would have not ordered it and saved some of the budget.

Extensions

After completing this project we discovered that we could extend the skill sets and knowledge that we have gain from this project to implement other ideas such a scale that would record and analyse a person's weight and body fat online, or a locking mechanism that is considered via Android application or web application.

Conclusion

In summary, we believe that we have achieved our primary objective that is being able to check if the cookie jar is empty or not from a fair distance so that the user would not waste their time walking all the way to the kitchen for nothing and can alert someone to fill the jar with more cookies before leaving their seat.

Appendix - Source Code

Source code for Arduino, Android app and Webapp is available at:
https://github.com/khanh2907/wireless_cookie_jar

Arduino Code:

```
#include <SoftwareSerial.h>   //Software Serial Port
#include <Servo.h>
#define RxD 6
#define TxD 7

#define DEBUG_ENABLED  1

SoftwareSerial blueToothSerial(RxD,TxD);

float analogValueAverage = 0;
float noLoad = 160;
float increment = (888-noLoad)/9;
long t = 0; //
int timeBetweenReadings = 200; // We want a reading every 200 ms;

int led = 13;

Servo myservo;  // create servo object to control a servo
                // a maximum of eight servo objects can be created
void setup()
{
  Serial.begin(9600);
  pinMode(RxD, INPUT);
  pinMode(TxD, OUTPUT);
  setupBlueToothConnection();
  pinMode(led, OUTPUT);

  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
  myservo.write(68);

}

void dispense()
{
 myservo.write(40);
```

```
delay(500);
myservo.write(68);
}

void loop()
{
  char recvChar;
  while(1){
    Serial.print("Ready\n");
    if(blueToothSerial.available()){//check if there's any data sent from the remote bluetooth shield
      recvChar = blueToothSerial.read();
      Serial.print(recvChar);
      Serial.print("\nSignal Received\n");
      checkInstr(recvChar);
      delay(1000);
    }
//    if(Serial.available()){//check if there's any data sent from the local serial terminal, you can add
the other applications here
//      recvChar  = Serial.read();
//      blueToothSerial.print(recvChar);
//    }

  }
}

void checkInstr(char instr){
  //first instr
          if (instr == “2”){
               Serial.print("Dispensing\n");
               dispense();
               Serial.print("Dispensed - Done\n");
               scale();
          }
}

void setupBlueToothConnection()
{
  blueToothSerial.begin(38400); //Set BluetoothBee BaudRate to default baud rate 38400
  blueToothSerial.print("\r\n+STWMOD=0\r\n"); //set the bluetooth work in slave mode
  blueToothSerial.print("\r\n+STNA=CookieTEST\r\n"); //set the bluetooth name as
"SeeedBTSlave"
  blueToothSerial.print("\r\n+STOAUT=1\r\n"); // Permit Paired device to connect me
  blueToothSerial.print("\r\n+STAUTO=0\r\n"); // Auto-connection should be forbidden here
```

```
  delay(2000); // This delay is required.
  blueToothSerial.print("\r\n+INQ=1\r\n"); //make the slave bluetooth inquirable
  Serial.println("The slave bluetooth is inquirable!");
  delay(2000); // This delay is required.
  blueToothSerial.flush();
}

void scale(){
  float analogValue = analogRead(4);

  //analogValueAverage = 0.99*analogValueAverage + 0.01*analogValue;

  char load = mapfloat(analogValue);
  Serial.print("analogValue: ");Serial.println(analogValue);
  Serial.print(" load: ");Serial.println(load);

  blueToothSerial.print(load);

}

char mapfloat(float x)
{
    if(x <195)
      return 'z';


    if (x > 195 && x <= 210)
    return '0';


    if (x > 210 && x <= 225)
    return '1';

    if (x > 225 && x <= 237)
    return '2';

    if (x > 237 && x <= 252)
    return '3';

    if (x > 252 && x <= 265)
    return '4';

    if (x > 265 && x <= 280)
```

```
    return '5';

    if (x > 280 && x <= 300)
    return '6';

    if (x > 301 && x <= 315)
    return '7';

    if (x > 315 && x <= 330)
    return '8';

    if (x > 330 && x < 350)
    return '9';

    if (x > 350)
      return '9';
}
```

PHP Script:

post_level.php:

```
<?php
require_once('include/common.php');
require_once('include/database.php');

ini_set('display_errors', 'On');
        error_reporting(E_ALL);

if (isset($_POST['cookie_level'])){
        echo "WE GOT SOMETHING!";
        print_r($_POST['cookie_level']);

        $db = connect();
    try {
        $stmt = $db->prepare('INSERT INTO CookieJar(cookie_level, time_posted) VALUES
(:input, CURRENT_TIMESTAMP)');
        $stmt->bindValue(':input', $_POST['cookie_level'], PDO::PARAM_INT);
        $stmt->execute();
        $result = $stmt->fetchColumn();
        $stmt->closeCursor();
    } catch (PDOException $e) {
        print "Error checking login: " . $e->getMessage();
```

```
   }
}
else{
        echo "NO DATA BRO";
}

?>
```

index.php:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Wireless Cookie Jar</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <!-- Bootstrap -->
    <link href="css/bootstrap.min.css" rel="stylesheet" media="screen">
  </head>
  <body>

    <script src="js/bootstrap.min.js"></script>
    <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.js"
type="text/javascript"></script>
<script src="http://code.highcharts.com/highcharts.js"></script>
<script src="http://code.highcharts.com/modules/exporting.js"></script>

    <script type="text/javascript">
        $(function () {
      $('#container').highcharts({
          chart: {
              type: 'spline'
          },
          title: {
              text: 'Cookie Jar Level Over Time'
          },
          xAxis: {
              type: 'datetime'

          },
          yAxis: {
              title: {
                  text: 'Cookie Level (%)'
              },
```

```
            min: 0,
            max: 100
        },

        series: [{
            name: 'Cookie Jar',
            // Define the data points. All series have a dummy year
            // of 1970/71 in order to be compared on the same x axis. Note
            // that in JavaScript, months start at 0 for January, 1 for February etc.
            data: [
                <?php
                  require_once('include/database.php');
                  $data = getDataForHC();

                                                    for($i=0; $i < sizeof($data); $i++){
                                                            if ($data[$i]['cookie_level'] != ""){
                                                                    if ($i != sizeof($data)-1){
                                                                            echo
"[",$data[$i]['time_posted'], ", ", $data[$i]['cookie_level'], "], ";
                                                                    }
                                                                    else{
                                                                            echo
"[",$data[$i]['time_posted'], ", ", $data[$i]['cookie_level'], "]";
                                                                    }
                                                            }
                                                    }
                ?>
            ]
        }]
    });
  });


                </script>

  <div class="navbar navbar-inverse navbar-fixed-top">
    <div class="navbar-inner">
      <div class="container">
        <button type="button" class="btn btn-navbar" data-toggle="collapse"
data-target=".nav-collapse">
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
```

```
      </button>
      <a class="brand" href="index.php">Wireless Cookie Jar</a>
      <div class="nav-collapse collapse">
        <ul class="nav">
          <li class="active"><a href="#">Home</a></li>
          <li><a href="about.php">About</a></li>
        </ul>
      </div><!--/.nav-collapse -->
    </div>
  </div>
</div>

<div class="container">

  <!-- Main hero unit for a primary marketing message or call to action -->
  <div class="hero-unit">
    <h1>Wireless Cookie Jar</h1>
    <p>An embedded system project for ELEC3607 by Khanh Nguyen and Kevin Nguyen.</p>
    <p><a href="about.php" class="btn btn-primary btn-large">Learn more </a></p>
  </div>


  <div class="row">
        <div class="span4">
          <div class="mycontent-left well">
                <center><h2>Current Level</h2></center>
     <p><div id = "cookiepic" data-toggle="tooltip" title="first tooltip">

                        <?php
                                require_once('include/database.php');

                                ini_set('display_errors', 'On');
                                error_reporting(E_ALL);

                                $latest = getLatestLevel();
                                if($latest['cookie_level'] <= 5){
                                        echo "<img src = \"img/cookie0.png\"></img>";
                                }
                                elseif($latest['cookie_level'] <= 10){
                                        echo "<img src = \"img/cookie1.png\"></img>";
                                }

                                elseif($latest['cookie_level'] <= 20){
```

```
                    echo "<img src = \"img/cookie2.png\"></img>";
                }

                elseif($latest['cookie_level'] <= 30){
                    echo "<img src = \"img/cookie3.png\"></img>";
                }

                elseif($latest['cookie_level'] <= 40){
                    echo "<img src = \"img/cookie4.png\"></img>";
                }

                elseif($latest['cookie_level'] <= 50){
                    echo "<img src = \"img/cookie5.png\"></img>";
                }

                elseif($latest['cookie_level'] <= 60){
                    echo "<img src = \"img/cookie6.png\"></img>";
                }

                elseif($latest['cookie_level'] <= 70){
                    echo "<img src = \"img/cookie7.png\"></img>";
                }

                elseif($latest['cookie_level'] <= 80){
                    echo "<img src = \"img/cookie8.png\"></img>";
                }

                elseif($latest['cookie_level'] <= 90){
                    echo "<img src = \"img/cookie9.png\"></img>";
                }

                elseif($latest['cookie_level'] <= 100){
                    echo "<img src = \"img/cookie10.png\"></img>";
                }
                echo "</div></p>";

                echo "<p>
                <div class=\"progress progress-striped active\">
                            <div class=\"bar\"
style=\"width:",$latest['cookie_level'],"%;\"></div>
                            </div>
                    </p>
            <center><p><span class=\"badge badge-info\">Cookie Jar is currently at
```

```
",$latest['cookie_level'],"%</span></p></center>";

                        ?>
        <center><p><a class="btn btn-primary" href="index.php">Refresh </a></p></center>
            </div>

          </div>
          <div class="span8">
            <div class="mycontent-right well">
        <p><div id="container" style="min-width: 400px; height: 400px; margin: 0 auto"></div></p>
            </div>
          </div>
         </div>

      <hr>

      <footer>
        <center><p>Khanh Nguyen and Kevin Nguyen | ELEC3607 | Semester 1 |
2013</p></center>
      </footer>

    </div> <!-- /container -->
  </body>
</html>
```