Section C – Software Documentation

Block Diagrams

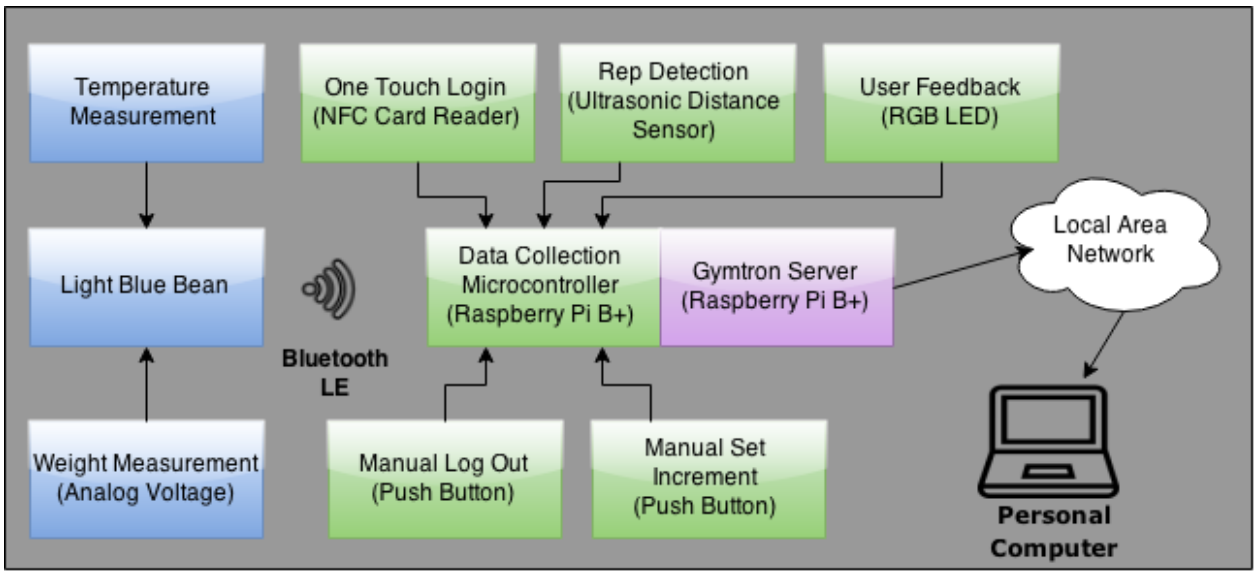
Gymtron Flowchart

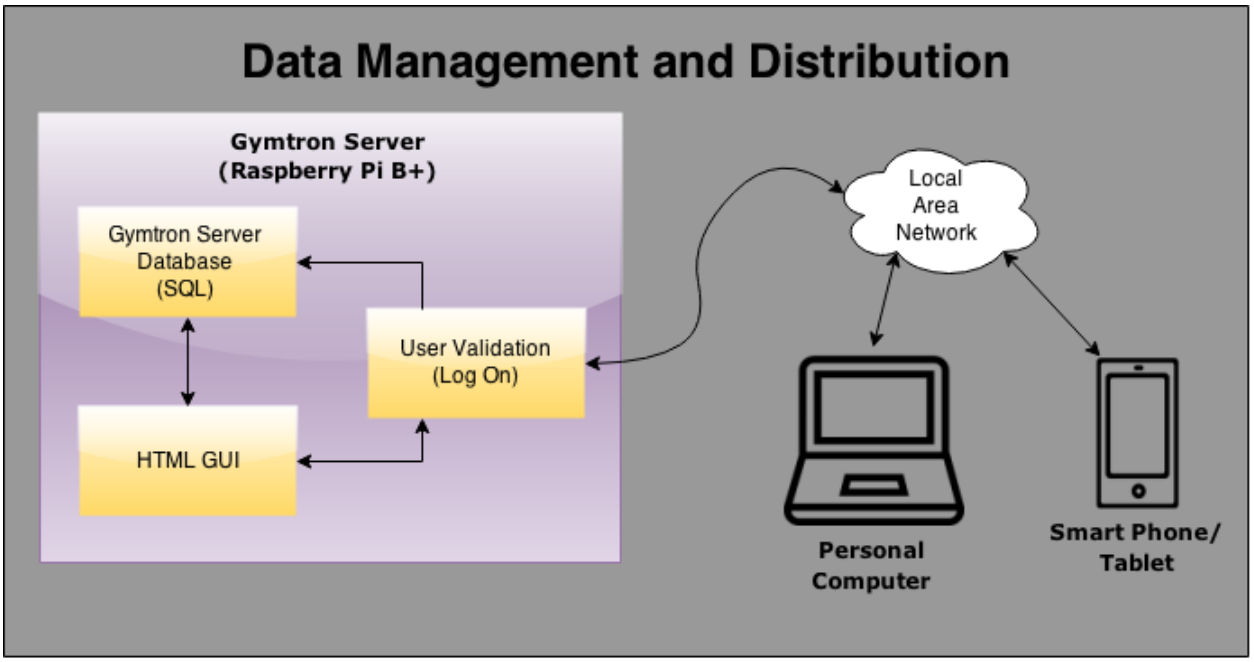
Gymtron State Diagram

Website Flow Charts

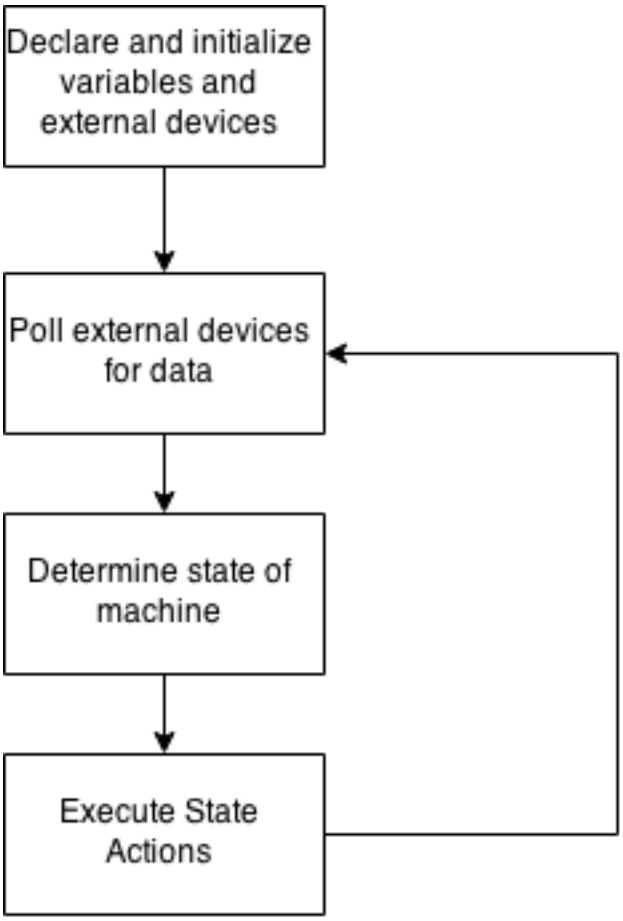
Code

Block Diagrams

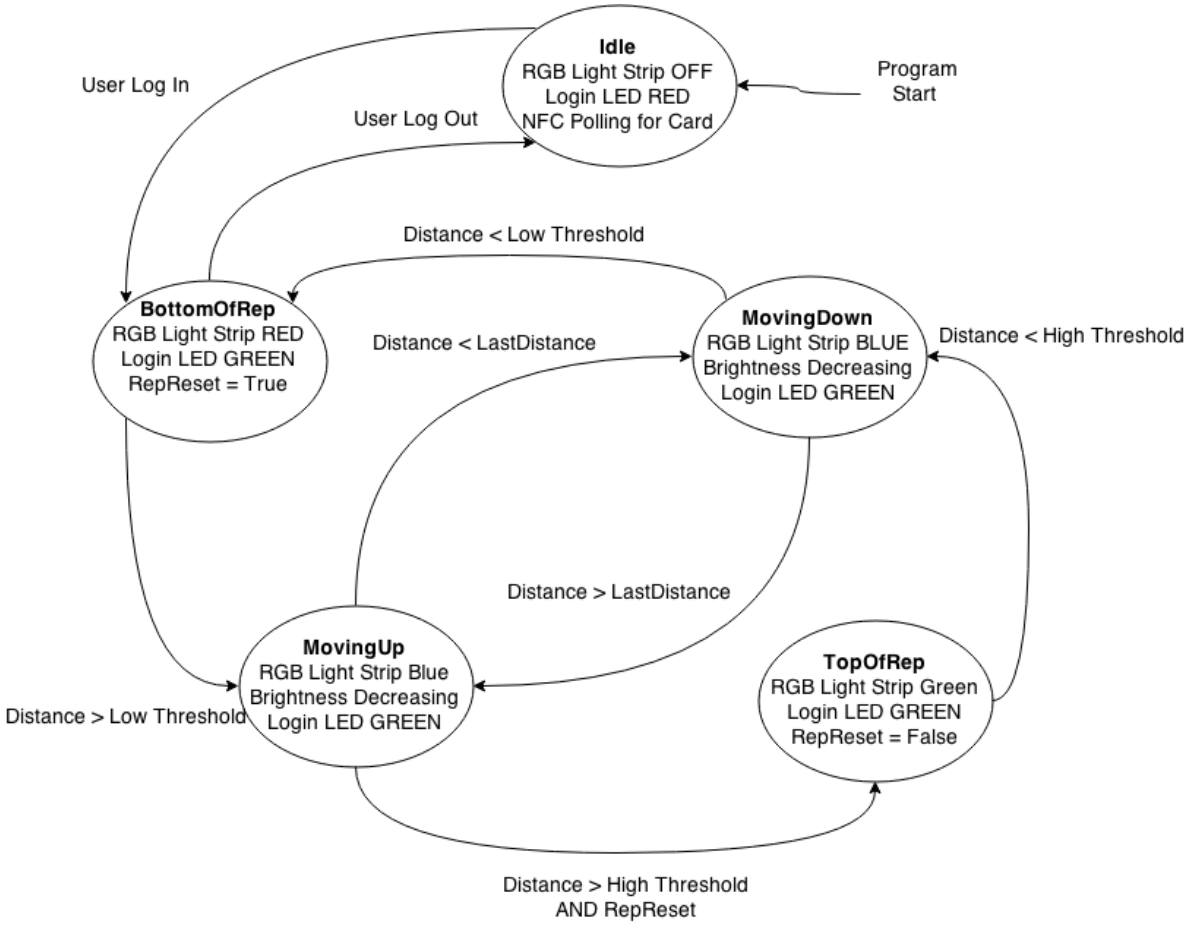




Gymtron Flow Chart

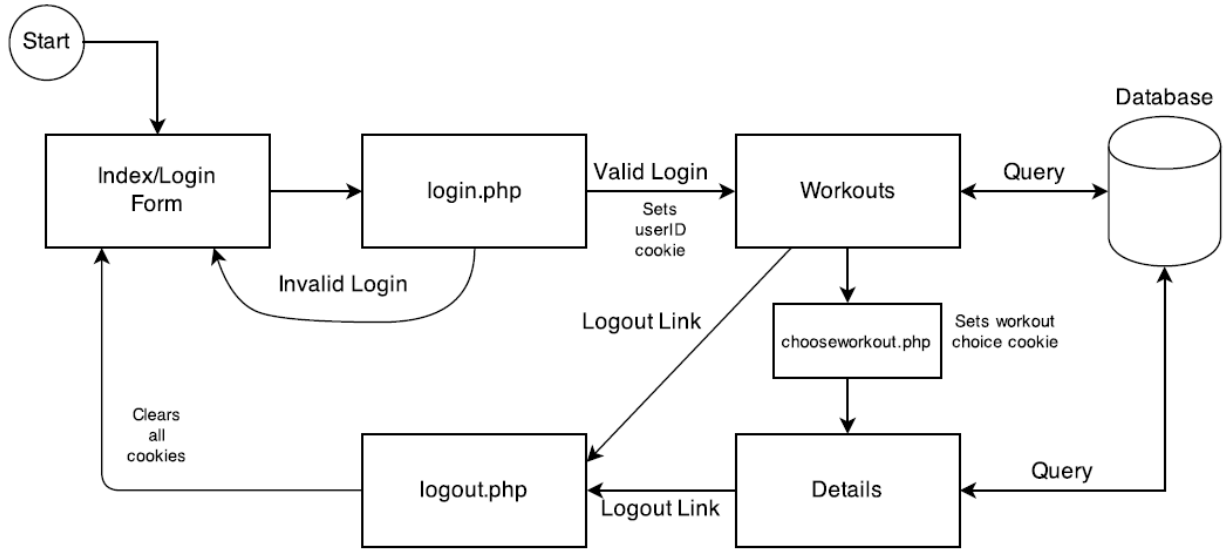


Gymtron State Diagram



Inputs	Outputs
Distance	RGB Light Strip (OFF, RED, GREEN, BLUE)
Last Distance	Login LED (RED, GREEN)
User Log In	
User Log Out	

Website Flow Chart



Code

php Code:

chooseworkout.php

details.php

index.php

login.php

logout.php

mystyle.css

workouts.php

C Code:

gymtronClean.c

Light Blue Bean Code:

Init_bean.sh

bean_talk.sh

GymtronData.ino

```php
<?php
//*****************************
//chooseworkout.php
//Authors: Mark Rumpel & Kyle Shaw
//Winter 2015
//Version 1.6

//this page is used to retrieve the data from the form on the workouts.php page
//and store the contents in a cookie that can be used later in the details.php page
//*****************************

//retrieves data from form
$wchoice = $_POST['wochoice'];

echo "whoice = " . $wchoice;

//inserts data from form into a cookie
setcookie('wochoice', $wchoice, time() + 120, "/");
echo "cookie set";
//redirects to details page upon completion
header( "Location: details.php" );

?>
```

```
<!--
*****************************
details.php
Authors: Mark Rumpel & Kyle Shaw
Winter 2015
Version 1.6

Adapted upon example code provided by W3Schools

this page will displays the details of a workout to the user
this page will query the database using the data stored in the cookies
and display the results back to the user in a table

*****************************
-->

<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>

<body>
<h1>Gymtron</h1>
<center><a id="toplinks" href="workouts.php">Back to Workouts</a></center>
<br/>
<center><a id="toplinks" href="logout.php">logout</a></center>
<br/>


<?php
//checks to ensure a user is logged in, otherwise redirects to the login page for security
reasons
if(!(isset($_COOKIE['usernum']))){
header( "Location: index.php" );
}

//gets data for variables from the cookies
//stores the workout # the user choose to see details for in the workouts page
$wchoice = $_COOKIE['wochoice'];
//stores the users gymtron id from login page
$userid = $_COOKIE['usernum'];

//echo "Userid = " . $userid;
//echo "WOChoice = " . $wchoice;

echo "<h2>User " . $userid . "'s details for workout " . $wchoice . ": </h2>";

//database connection info contained in variables
$servername = "localhost";
$username = "root";
$password = "gymtron";
$dbname = "Gymtron_DB";
```

```
//create connection
$conn = mysqli_connect($servername, $username, $password);

//check connection and output error message if unsuccessful
if (!$conn) {
    die("Connection Failed: " . mysqli_connect_error());
}

//echo "Connected Successfully";

//telling sql which db to use
$sqlusedb = "USE Gymtron_DB";
if (mysqli_query($conn, $sqlusedb)) {
    //echo "Using Gymtron_DB";
} else {
    echo "Not using Gymtron_DB";
}

//queries db for all columns for specific user and workout #
//while loop displays data on page in tabular form
echo "<table>";
//query the db and take all records with matching user id and workout number
$sql = "SELECT * from GTUser_Data WHERE User_ID='" . $userid . "' AND Workouts='" . $wchoice .
"'";
//result of query stored
$result = mysqli_query($conn, $sql);
//if records found, display the data contained in $result
if (mysqli_num_rows($result) > 0) {
    //setting up the table header
    echo "<tr><th>User ID</th><th>Workout</th><th>Set</th><th>Reps</th><th>Weight</th><th>Set
    Time</th><th>Date</th></tr>";
    //output data of each row by stepping through the array row by row
    while($row = mysqli_fetch_assoc($result)) {
        echo "<tr>";
        echo "<td>" . $row["User_ID"]. "</td><td>" . $row["Workouts"]. "</td><td>" . $row["Sets"
        ]. "</td><td>" . $row["Reps"]. "</td><td>" . $row["Weight"]. "</td><td>" . $row["WOtime"
        ]. "</td><td>" . $row["WOdate"]. "</td>";
        echo "</tr>";
}
} else {
    echo "0 Results";
}

//close connection to db
mysqli_close($conn);

?>

</table>
</body>
</html>
```

```
<!--
*****************************
index.php
Authors: Mark Rumpel & Kyle Shaw
Winter 2015
Version 1.6

this is the home page, displays a login form and prompts user for credentials
this page will then send the input from the form to login.php

*****************************
-->


<!DOCTYPE html>
<head>
<title>Login Form</title>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>

<body id="loginbody">
  <section class="container">
    <div class="login">
      <h1>Login to Gymtron Web App</h1>
        <br/>
        <br/>
        <br/>
      <form method="post" action="login.php">
        <p><input type="text" name="login" value="" placeholder="Enter Gymtron ID"></p>
       <!-- <p><input type="password" name="password" value="" placeholder="Password"></p> -->
        <p class="submit"><input type="submit" name="commit" value="Login"></p>
      </form>
    </div>

  </section>
</body>
</html>
```

```php
<?php
//******************************
//login.php
//Authors: Mark Rumpel & Kyle Shaw
//Winter 2015
//Version 1.6

//this page handles checking the users credentials with the database and redirecting once
status is known
//credentials are stored in a cookie once verified as valid

//******************************

//storing the connection details for the database in variables
$servername = "localhost";
$username = "root";
$password = "gymtron";
$dbname = "Gymtron_DB";

//retrieves the entered id from the form on index.php
$userid = $_POST['login'];



//create connection
$conn = mysqli_connect($servername, $username, $password);

//check connection and error out if no connection
if (!$conn) {
    die("Connection Failed: " . mysqli_connect_error());
}

//tell sql which db to use
$sqlusedb = "USE Gymtron_DB";
if (mysqli_query($conn, $sqlusedb)) {
    echo "Using Gymtron_DB";
} else {
    echo "Not using Gymtron_DB";
}
echo "<br>";

//queries the db with the entered credentials
$sql = "SELECT * from GTUser_Data WHERE User_ID='" . $userid . "'";
$result = mysqli_query($conn, $sql);

//checking to see if credentials are contained in the db
//if nothing entered, return to login page
if (empty($_POST['login'])){
    echo "please input data";
    header( "Location: index.php" );

}else{
//if entered id is a user, set a cookie containing the user id for use later and redirect to
```

```php
the workouts page
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (mysqli_num_rows($result) > 0) {

    setcookie('usernum', $userid, time() + 120, "/");
    echo "cookie set";
    header( "Location: workouts.php" );


  } else {
    echo "cookie not set";
    header( "Location: index.php" );

    }
    }
}

//close the connection to the db
mysqli_close($conn);

?>
```

```php
<?php

//******************************
//logout.php
//Authors: Mark Rumpel & Kyle Shaw
//Winter 2015
//Version 1.6

//this page will clear all cookies and redirect to the login page

//******************************

//this page deletes the data contained in the cookies and redirects to the login page
setcookie('usernum', '', time()-60*60*24*365, '/');
setcookie('wochoice', '', time()-60*60*24*365, '/');

echo "Redirecting...";

//redirects to the login page
header( "Location: login.php" );

?>
```

```
html {
        height: 120%
}
h1 {
        text-align: center;
        font-size: 40px;
        font-style: bold;
        font-style: italic;
}

#toplinks {
        font-size: 25px;
        text-decoration: none;
}

#loginbody{
        text-align: center;
}

body {
        font-size: 20px
        background-color: #A9F5F2;
        background: -webkit-linear-gradient(#A9F5F2, white); /* For Safari 5.1 to
6.0 */
        background: -o-linear-gradient(#A9F5F2, white); /* For Opera 11.1 to 12.0
*/
        background: -moz-linear-gradient(#A9F5F2, white); /* For Firefox 3.6 to 15
*/
        background: linear-gradient(#A9F5F2, white); /* Standard syntax */
}

p {
    font-size: 20px;
}
table, th, td {
        font-size: 20px;
        border: 2px solid black;
        padding: 20px;
        border-collapse: collapse;
}
```

```
<!--
*****************************
workouts.php
Authors: Mark Rumpel & Kyle Shaw
Winter 2015
Version 1.6

Adapted upon example code provided by W3Schools

this page will displays a summary of the details for each workout to the user
this page will query the database using the data stored in the cookies to gather workout data
and display the results back to the user in a table

*****************************
-->

<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>

<body>
<h1>Gymtron</h1>
<center><a id="toplinks" href="logout.php">logout</a></center>


<h2>Welcome!</h2>
<h3>View Details for Past Workouts</h3>

<form method="post" action="chooseworkout.php">
        <p><input type="text" name="wochoice" value="" placeholder="Enter Workout Choice"></p>
        <p class="submit"><input type="submit" name="commit" value="Enter"></p>
      </form>



<?php

//checks to ensure a user is logged in, if not redirects to login page
//for security reasons
if(!(isset($_COOKIE['usernum']))){
header( "Location: index.php" );
}

//initializing and setting up the info about the database and storage variables
$servername = "localhost";
$username = "root";
$password = "gymtron";
$dbname = "Gymtron_DB";
$userid = $_COOKIE['usernum'];
$count = "0";
$worknum = "";
```

```
$totsets = "";
$workdate = "";

if(isset($_COOKIE['usernum'])){
    //echo "cookie set";
    //echo "welcome back " . $_COOKIE['usernum'];
}else {echo "cookie not set";}

//echo "UserID= " . $userid;
//echo "<br>";

//create connection
$conn = mysqli_connect($servername, $username, $password);

//check connection, display error if not connected properly
if (!$conn) {
    die("Connection Failed: " . mysqli_connect_error());
}

//telling sql which db to use
$sqlusedb = "USE Gymtron_DB";
if (mysqli_query($conn, $sqlusedb)) {
    //echo "Using Gymtron_DB";
} else {
    echo "Not using Gymtron_DB";
}

//find the largest workout number for display purposes
$sqlmax = "SELECT MAX(Workouts) as WONum FROM GTUser_Data WHERE User_ID='" . $userid . "'";
$resultmax = mysqli_query($conn, $sqlmax);
$row2 = mysqli_fetch_assoc($resultmax);
$worknum = $row2["WONum"];
//echo "MaxWO = " . $worknum;


echo "<h3>Past Workouts</h3>";

//displaying the workout, the number of sets in that workout to the user in a table
$count = $worknum;
echo "<table>";
echo "<tr><th>Workout</th><th>Sets</th><th>Date</th></tr>";
//counts down through each set
while($count > 0){
    echo "<tr>";
    //finds the maximum set number with an sql query and storess the result
    $sqlsets = "SELECT MAX(Sets) as Totsets FROM GTUser_Data WHERE Workouts='" . $count . "'
    AND User_ID='" . $userid . "'";
    $resultsets = mysqli_query($conn, $sqlsets);
    $row3 = mysqli_fetch_assoc($resultsets);
    $totsets = $row3["Totsets"];

    //finds the date of the first set of the given user and workout and stores the result
    $sqldate = "SELECT WOdate FROM GTUser_Data WHERE User_ID='" . $userid . "' AND Sets='1' AND
```

```
    Workouts='" . $count . "'";
    $resultdate = mysqli_query($conn, $sqldate);
    $row4 = mysqli_fetch_assoc($resultdate);
    $workdate = $row4["WOdate"];

    //prints the workout, number of sets in the workout, and the date of each workout
    echo "<td>" . $count . "</td><td>" . $totsets ."</td><td>" . $workdate . "</td>";
    $count = $count - 1;
    echo "</tr>";
}
//closes connection to database
mysqli_close($conn);

?>

</table>
</body>
</html>
```

```
/*
 * gymtron.c
 *
 * Purpose: Main control logic for monitoring and recordeing the state of
 * the exercise machine and the user's workout on that exericse machine.
 *
 * Authors: Kyle Shaw and Mark Rumpel
 * *
 */



#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
#include <string.h>
#include <wiringPi.h>   //For GPIO Control
#include <nfc/nfc.h>
#include <mysql/mysql.h>
#include <time.h>

//GPIO Pin Constants (using BCM pin numbers)
#define RG_PWM 13       //PWM1 Channel to control red and green brightness
#define BLUE 18         //PWM0 channel to control blue brightness
#define RED 17          //Enable Red LEDS (HIGH is ON)
#define GREEN 27        //Enable Green LEDs (HIGH is ON)
#define SONIC_TRIGGER 23//trigger pin for HC-SR04
#define SONIC_ECHO 22   //Echo pulseinput from HC-SR04
#define LOGOUT_BTN 6    //button press triggers manual logout
#define SET_BTN 5       //pressed when set is completed
#define LOGGED_IN 4     //Indicates a user is currently logged in(green)
#define IDLE_LED 25     //Indicates that a user is not logged in (red)

//System Constants
#define DEFAULT_TEMP 23                 //temperture used for distance calc if data is
unavailable from the bean
#define MAX_SETS 4                      //max sets a user can have per workout
#define MAX_TIME_BETWEEN_REPS 30000     //Max time between two reps in the same set (in
milliseconds) before auto set count++
#define AUTO_LOGOUT_TIME 60000          //Logout if no activity after this amount of time (ms)
#define LOW_THRESH 70                   //Positions greater than this define the stack in a
resting position
#define HIGH_THRESH 19                  //Position of stack that defines the top of a rep
#define MAX_DISTANCE 150                //max distance that the machine is allowed to measure
(cm) (ultrasonic timeout)
#define RG_BRIGHTNESS 700               //PWM setting for red and green LEDs
#define BLUE_MAX 800                    //PWM max brightness for blue LEDS
#define BLUE_MIN 0                      //PWM min brightness for blue LEDS
#define FILTER_SIZE 3                   //Filter length for distance measurment (# samples per
measurement)
#define BEAN_BUFFER 128                 //Buffer size for reading bean values from script


//Data structures
```

```
struct Set{
    int numReps;
    int weight;
    int timeOfFirstRep;
    int timeOfLastRep;
};

struct Workout{
    struct Set userSets[MAX_SETS];
    int startTime;
    int endTime;
    int workoutID;          //unique ID for a particular user's workouts
};

//Global Variables for ISRs
int manualLogout = 0;
int manualSet = 0;

//Function Declarations (see definitions for more detailed comments)
void GpioSetup();
void CleanUp();
int GetMedianDistance(int temp);
int GetDistance(int temp);
void GetWeightAndTemp(int *weight, int *temp, int defaultTemp);
int DistanceToBrightness(int distance);
void SubmitToDataBase(struct Workout *userWorkout, int currentSet, int userID);
int UserLogout(int userID);
void InitWorkout(struct Workout *userWorkout, int *currentSet);
int Mode(int array[], int size);
int login(int currentid);
int GetMaxWorkout(int userID);
void ManualLogout();
void ManualSet();
void SetIncrementFlash();

int main(void)
{
//Debugging varaibles
    int a;
    int length;
    int cycleTime;

//General Variable Declaration and Init
    bool run = true;        //true means program will run.
    int userID = 0;
    int prevDistance = 0;
    int newDistance = 0;
    int weight = 0;
    int temp = 0;
    int currentSet = 0;
    int currentTime = 0;
    bool repReset = false; //ensures full range of movement for each rep
```

```
    struct Workout userWorkout; //must be initialized after GPIOSetup() so that millis() clock
    is set

//State Variable Declarations
    bool idle = true;               //default state
    bool stackMovingUp = false;
    bool stackMovingDown = false;
    bool topOfRep = false;
    bool bottomOfRep =  false;

//Setup GPIO
    GpioSetup();

//Setup ISR functions
    wiringPiISR(LOGOUT_BTN, INT_EDGE_RISING, &ManualLogout);
    wiringPiISR(SET_BTN, INT_EDGE_RISING, &ManualSet);

//Initialize Bluetooth using script
    system ("./init_bean.sh");
    delay(100);
    GetWeightAndTemp(&weight, &temp, DEFAULT_TEMP); //initialize weight and temp from bean

//Main Program Loop
    while(run){
        delay(100);
        cycleTime = micros();

//Take Measurements from system
        prevDistance = newDistance;
        newDistance = GetMedianDistance(temp);

//Set State Variables
        if(userID == 0){
            //Machine is in idle mode
            idle = true;

            //Ensure all other states are set to false
            stackMovingUp = false;
            stackMovingDown = false;
            topOfRep = false;
            bottomOfRep = false;

        }
        else{
            idle = false;

            if(newDistance != prevDistance){
                stackMovingUp = (newDistance < LOW_THRESH) && (newDistance > HIGH_THRESH) && (
                newDistance < prevDistance);
                stackMovingDown = (newDistance < LOW_THRESH) && (newDistance > HIGH_THRESH) && (
                newDistance > prevDistance);
                topOfRep = (newDistance <= HIGH_THRESH);
                bottomOfRep = (newDistance >= LOW_THRESH);
```

```
            }
        }


//Execute State Actions (priority decreases as line number increases)

        if(idle){
                printf("IDLE\n");
            //while no one is logged into machine, poll for a new user and set status lights
                digitalWrite(LOGGED_IN,LOW);
                digitalWrite(IDLE_LED, HIGH);

                userID = login(userID); //sets the new users id from their card as the current id

                //make sure flag is cleared for new user
                manualLogout = 0;
                manualSet = 0;

                digitalWrite(IDLE_LED, LOW);
                digitalWrite(LOGGED_IN, HIGH);

                InitWorkout(&userWorkout, &currentSet);//initialize workout once a user has
                logged in
                userWorkout.workoutID = GetMaxWorkout(userID) + 1 ;//set the current workoutID
                to one greater than the user's largest workoutID in database
        }
//*******
        else if(stackMovingUp){
            printf("Stack Moving Up\n");

            //Turn off red and green lights
            digitalWrite(GREEN, LOW);
            digitalWrite(RED, LOW);
            pwmWrite(RG_PWM, 0);

            //Update user lights
            pwmWrite(BLUE, DistanceToBrightness(newDistance));
        }
//*******
        else if(stackMovingDown){
            printf("Stack Moving Down\n");

            //Turn off red and green lights
            digitalWrite(GREEN, LOW);
            digitalWrite(RED, LOW);
            pwmWrite(RG_PWM, 0);

            //Update user lights
            pwmWrite(BLUE, DistanceToBrightness(newDistance));
        }
//*******
        else if(topOfRep){
```

```
        printf("Top of Rep\n");

        //check to make sure rep started from bottomOfRep
        if(repReset){
            //Turn off red and blue lights
            digitalWrite(RED, LOW);
            pwmWrite(BLUE, 0);

            //Turn on green lights
            digitalWrite(GREEN, HIGH);
            pwmWrite(RG_PWM, RG_BRIGHTNESS);

            //Hold current time for comparisons
            currentTime = millis();

            //check to see if new set has been inidcated by user
            if(manualSet == 1){
                //don't allow set to be incremented with 0 reps or above the set limit
                if((userWorkout.userSets[currentSet].numReps != 0) && (currentSet < (
                MAX_SETS -1))){
                    currentSet = currentSet + 1;
                }
                manualSet = 0;//reset flag
            }

            //check to see if set has timed out (if manual ++ alrady occured then
            timeofLastRep = 0)
            if((userWorkout.userSets[currentSet].timeOfLastRep != 0) && (currentTime -
            userWorkout.userSets[currentSet].timeOfLastRep) > MAX_TIME_BETWEEN_REPS){
                if((userWorkout.userSets[currentSet].numReps != 0) && (currentSet < (
                MAX_SETS -1))){
                    currentSet = currentSet + 1;
                    SetIncrementFlash();
                }
            }

            //if the first rep, update the necessary information
            if(userWorkout.userSets[currentSet].numReps == 0){
                //no reps have taken place so record time of first rep
                userWorkout.userSets[currentSet].timeOfFirstRep = currentTime;

                //Check to see if weight data is available (Is GymtronBean communication
                working)
                //if so update the weight for this set, else weight remains undefined
                if(weight != -1){
                    GetWeightAndTemp(&weight, &temp, DEFAULT_TEMP);//communication working
                    so update weight
                    userWorkout.userSets[currentSet].weight = weight;//record the weight
                    for this set
                }
            }

            //Record new information
```

```
                userWorkout.userSets[currentSet].numReps++;                    //record the
                current rep
                userWorkout.userSets[currentSet].timeOfLastRep = currentTime;  //update the
                last rep time
                repReset = false;                                              //Make sure
                bottomOfRep happens before another rep is counted
            }

        }
//*******
        else if(bottomOfRep){
            printf("Bottom of Rep\n\n");

            //Turn off green and blue lights. turn on red lights
            digitalWrite(GREEN, LOW);
            digitalWrite(RED, HIGH);
            pwmWrite(RG_PWM, 500);
            pwmWrite(BLUE, 0);

            //Check for user inactivity timeout
            currentTime = millis();
            if((((currentTime - userWorkout.userSets[currentSet].timeOfLastRep) >
            AUTO_LOGOUT_TIME) && ((currentTime - userWorkout.startTime) > AUTO_LOGOUT_TIME)) ||
            (manualLogout == 1)){ //if no reps then timeOfLastRep is 0
                //Add user data to the database, reset workout stats, and logout the user
                printf("currentTime: %d\n", currentTime);
                printf("timeofLastRep: %d\n",userWorkout.userSets[currentSet].timeOfLastRep);
                printf("startTime: %d\n",userWorkout.startTime);
                userWorkout.endTime = millis();
                SubmitToDataBase(&userWorkout, currentSet, userID);
                userID = UserLogout(userID);
                //printf("userID: %d\n",userID);
            }

            //bottom of rep has occured so allow another rep to be counted
            repReset = true;

        }

//Print out varable values for dubugging
        if(userID != 0){
            //printf("prevDistance: %d \n",prevDistance);
            printf("userID: %d\n", userID);
            printf("newDistance: %d \n", newDistance);
            printf("weight: %d \n", weight);
            printf("temp: %d \n\n", temp);
            printf("Iteration Time: %d\n", (micros()-cycleTime));
            //print out current workout
            for(a = (currentSet-2); a<=(currentSet); a++){
                printf("Set %d First Time: %d\n", a, userWorkout.userSets[a].timeOfFirstRep);
                printf("Set %d Prev Time: %d\n", a, userWorkout.userSets[a].timeOfLastRep);

                length = userWorkout.userSets[a].timeOfLastRep - userWorkout.userSets[a].
```

```
                timeOfFirstRep;
                printf("Set %d duration: %d\n", a, length);

                printf("Set %d Reps: %d @ %d lbs", a, userWorkout.userSets[a].numReps,
                userWorkout.userSets[a].weight );

                printf("\n\n");
            }
        }
    }//end main loop

    CleanUp();
    return 0;
}


// *************** Function Definitions ******************************

void GpioSetup()
// Configure all I/O
{
    printf("Gymtron\n");

    wiringPiSetupGpio(); //initialize wiringPi library

    pinMode(RED, OUTPUT); //RED led enable output
    digitalWrite(RED, LOW);

    pinMode(GREEN, OUTPUT); //GREEN led enable output
    digitalWrite(GREEN, LOW);

    pinMode(RG_PWM, PWM_OUTPUT); //RED and GREEN LED brightness control
    pwmWrite(RG_PWM,0);

    pinMode(BLUE, PWM_OUTPUT); // BLUE LED brightness control (and enable)
    pwmWrite(BLUE, 0);

    pinMode(LOGGED_IN, OUTPUT); //User logged in status led  enable output
    digitalWrite(LOGGED_IN, LOW);

    pinMode(IDLE_LED, OUTPUT); //idle state led enable output
    digitalWrite(IDLE_LED, LOW);

    pinMode(SONIC_TRIGGER, OUTPUT); //ultrasonic sensor trigger pin
    digitalWrite(SONIC_TRIGGER, LOW);
    //delay(5);make sure it settles to low

    pinMode(SONIC_ECHO, INPUT); //ultrsonic return pulse pin

    pinMode(LOGOUT_BTN, INPUT); //triggers manual user logout
    pullUpDnControl(LOGOUT_BTN, PUD_DOWN);

    pinMode(SET_BTN, INPUT); //tells user that set has been incremented
```

```
    pullUpDnControl(SET_BTN, PUD_DOWN);


    printf("Setup Complete!\n");
}

// ***********************************************************************

void InitWorkout(struct Workout *userWorkout, int *currentSet)
{
    int i;
    struct Workout zeroWorkout;

    *currentSet = 0;

    wiringPiSetupGpio(); //initialize wiringPi library to reset clocks

    zeroWorkout.startTime = millis();
    zeroWorkout.endTime = 0;
    zeroWorkout.workoutID = 0;

    for(i = 0; i<MAX_SETS; i++){
        zeroWorkout.userSets[i].timeOfLastRep = 0;
        zeroWorkout.userSets[i].timeOfFirstRep = 0;
        zeroWorkout.userSets[i].numReps = 0;
        zeroWorkout.userSets[i].weight = 0;
    }

    *userWorkout = zeroWorkout;

    return;
}

// ***********************************************************************

void CleanUp()
//Writes all used I/O to low
{
    //make bluetooth down?
    digitalWrite(RED, LOW);
    digitalWrite(GREEN, LOW);
    pwmWrite(BLUE, 0);
    pwmWrite(RG_PWM, 0);
    digitalWrite(SONIC_TRIGGER, LOW);
    digitalWrite(LOGGED_IN, LOW);
    digitalWrite(IDLE_LED, LOW);
    printf("Clean Up Complete!\n");
}

// ***********************************************************************

int GetDistance(int temp)
//gives distance to stack in cm
```

```
{
    int start, end, echoTime, maxEchoTime;
    float v, distance;

    //calculate current velocity of sound
    v = (331.4 + (0.6*temp))/10000;

    //Calculate timeout for pulse
    maxEchoTime = 2*MAX_DISTANCE/v;

    //Send trigger pulse
    digitalWrite(SONIC_TRIGGER, HIGH);
    delayMicroseconds(10);
    digitalWrite(SONIC_TRIGGER, LOW);

    //wait for echo to start
    while(digitalRead(SONIC_ECHO) == 0)
        start = micros();

    //find end time of echo
    while(digitalRead(SONIC_ECHO) == 1){
        end = micros();
        if((end-start) > maxEchoTime)
            break;//end loop early if timeout occurs so program cannot get hung up waiting on
            lost signal
    }

    //find total time of echo
    echoTime = end - start;
    if(echoTime > maxEchoTime)
        echoTime = maxEchoTime;

    ////calculate distance measured (cm)
    distance = v*((float)echoTime/2);

    return ((int)(distance +0.5));//return distance rounded to the nearest cm
}

// ****************************************************************

int GetMedianDistance(int temp)
//measures distance FILTER_SIZE times and returns the median
{
    int pos, t, i;
    int buffer[FILTER_SIZE];


    for(i=0; i<FILTER_SIZE; i++){
        buffer[i] = GetDistance(temp);
        pos = i;
        //insert new distance into buffer going from smallest to largest
        while(pos != 0 && buffer[pos] < buffer[pos-1]){
            t = buffer[pos];
```

```
            buffer[pos] = buffer[pos-1];
            buffer[pos-1] = t;
            pos--;
        }

        delay(25);
    }

    return buffer[FILTER_SIZE/2]; //return median value
}
// ****************************************************************

void GetWeightAndTemp(int *weight, int *temp, int defaultTemp)
//gets weight and temperature from GymtronBean
//If communication to GymtronBean fails weight is set to -1
//and temp is set to defaultTemp
{
    char line[BEAN_BUFFER];
    int linenr = 1;
    FILE *pipe;

    //Setup pipe to be where output of script is ("r" means read)
    pipe = popen("./bean_talk.sh", "r");

    //Check for errors
    if(pipe == NULL){
        printf("Pipe to bean_talk.sh failed.\n");
    }
    else{
        //Read in temperature from pipe
        if(fgets(line, BEAN_BUFFER, pipe) != NULL){
            *temp = strtol(line, NULL, 16);//convert value from bean from hex to dec and store
            in temp

            if(*temp == 107){//check for error (107 is connection error code)
                printf("Connection to GymtronBean failed while fetching temperature.\n");
                *temp = defaultTemp; //assume a default temperature
            }
            else{//connection was successful
                ++linenr;//increment to next line in buffer
            }
        }
        else{
            printf("Temperature data unavailable.\n");
            *temp = defaultTemp; //assume a default temperature
        }

        //Read in weight from pipe
        if(fgets(line, BEAN_BUFFER, pipe) != NULL){
            *weight = strtol(line, NULL, 16); //convert value from bean from hex to dec and
            store in weight

            if(*weight == 107){//check for error (107 is connection error code)
```

```
                printf("Connection to GymtronBean failed while fetching weight.\n");
                *weight = -1; //set weight to -1 to flag that weight is not available
            }
            else{//connection was successful
                ++linenr;//increment to next line in buffer
            }
        }
        else{
            printf("Weight data unavailable.\n");
            *weight = -1; //set weight to -1 to flag that weight is not available
        }
    }

    //Script has ended so close pipe
    pclose(pipe);

    return;
}

// ********************************************************************

int DistanceToBrightness(int distance)
//convert distance to pwm brightness
// assumes a linear relationship between voltage and LED brightness
//(Faster way than using floats like this???)
{
    int brightness;
    float fBright;

    //typecast ints here so calculation code is easier to read
    float fDis = (float)distance;
    float fMaxDis = (float)LOW_THRESH;
    float fMinDis = (float)HIGH_THRESH;
    float fBlueMax = (float)BLUE_MAX;
    float fBlueMin = (float)BLUE_MIN;

    //Complete calculation with typecasted ints because multiplying by a percentage
    fBright = (((fMaxDis - fDis)/(fMaxDis - fMinDis))*(fBlueMax - fBlueMin)) +fBlueMin;

    //Convert back to an int before return
    brightness = (int)fBright;

    return brightness;
}

// ********************************************************************

void SubmitToDataBase(struct Workout *userWorkout, int currentSet, int userID)
{
    //printf("User %d entered database function.\n\n", userID);

    //gives info about database to login
    #define dbhost "localhost"
```

```
#define dbuser "root"
#define dbpass "gymtron"
#define dbdatabase "Gymtron_DB"

//gives specifications for the data to be inserted
#define MAX_DATA_SIZE 150
#define SQL_IN_SIZE 200
#define SQL_INSERT "INSERT into GTUser_Data (User_ID, Workouts, Sets, Reps, Weight, WOtime,
WOdate) VALUES "

//pointers used to connect to the database
MYSQL *conn;
MYSQL_ROW row;
MYSQL_FIELD *field;

//initializing variables to store the sql query
int i;
char data[MAX_DATA_SIZE];
char sql[SQL_IN_SIZE];

//copy workout passed into function into a local workout struct for easy syntax
struct Workout submitWorkout = *userWorkout;

//variables needed to get workout date and length
time_t rawtime;
struct tm * timeInfo;
char date[9];
//int woLength;
int setLength;

//Get current date in MM/DD/YY format
time(&rawtime);
timeInfo = localtime(&rawtime);
strftime(date, 9, "%D", timeInfo);

//Calcualte workout length in minutes from mirco seconds
//woLength = (submitWorkout.endTime - submitWorkout.startTime)/60000;

////intializing connection pointer
conn = mysql_init(NULL);

////connecting to database using login data
mysql_real_connect(conn, dbhost, dbuser, dbpass, dbdatabase, 0, NULL, 0);


//Fill values into userWorkout
//userWorkout.workoutID = 222;
//for(i=0; i<MAX_SETS; i++){
    //userWorkout.userSets[i].numReps = i+1;
    //userWorkout.userSets[i].weight = i+100;
//}

for(i=0; i <= currentSet; i++){
```

```
        //build sql query
        setLength = (submitWorkout.userSets[i].timeOfLastRep - submitWorkout.userSets[i].
        timeOfFirstRep)/1000;
        strcpy(sql, SQL_INSERT);
        snprintf(data, MAX_DATA_SIZE, "(%d, %d, %d, %d, %d, %d, ", userID, submitWorkout.
        workoutID, i, submitWorkout.userSets[i].numReps, submitWorkout.userSets[i].weight,
        setLength);
        strcat(data, "'");
        strcat(data, date);
        strcat(data, "')");
        //add the data unique to each set to the query
        strcat(sql,data);
        mysql_query(conn, sql); //submit the query to the database
        strcpy(sql, "");//clear the query string for the next iteration
    }

    //printf("Test MySQL client version: %s\n", mysql_get_client_info());
    mysql_close(conn);

    printf("User %d submited to the database.\n\n", userID);

    return;
}

// ********************************************************************

int UserLogout(int userID)
{
    manualLogout = 0;//reset logout button flag
    CleanUp();
    printf("User %d logout sucessful.\n\n", userID);

    return 0; //return 0 if user has been logged out successfully
}

// ********************************************************************

int login(int currentid)
{
    //initializing pointers for the nfc device
    nfc_device *pnd;
    nfc_target nt;
    nfc_context *context;
    nfc_init(&context);
    long scanid;

//check to ensure libnfc is installed
  if (context == NULL) {
    printf("Unable to init libnfc (malloc)\n");
    exit(EXIT_FAILURE);
  }
  //open the nfc device
  pnd = nfc_open(context, NULL);
```

```
    //if unable to find nfc device, print error and exit
  if (pnd == NULL) {
    printf("ERROR: %s\n", "Unable to open NFC device.");
    exit(EXIT_FAILURE);
  }
  if (nfc_initiator_init(pnd) < 0) {
    nfc_perror(pnd, "nfc_initiator_init");
    exit(EXIT_FAILURE);
  }
    //output successful opening
  printf("NFC reader: %s opened\n", nfc_device_get_name(pnd));

//setting parameters for the nfc cards used
  const nfc_modulation nmMifare = {
    .nmt = NMT_ISO14443A,
    .nbr = NBR_106,
  };
  //poll for an nfc card, and when detected, output info
  if (nfc_initiator_select_passive_target(pnd, nmMifare, NULL, 0, &nt) > 0) {
    printf("The following (NFC) ISO14443A tag was found:\n");
    printf("       UID (NFCID%c): ", (nt.nti.nai.abtUid[0] == 0x08 ? '3' : '1'));

  }
  //set scanid to the cards given unique id field
    scanid = nt.nti.nai.abtUid[1];

    printf("\n\ncurrentid = ");
    printf("%d\n", currentid);
    printf("\n");

    printf("scanid = ");
    printf("%ld\n", scanid);
    printf("\n");

//if the same user scans in, log them out and set user to 0
    if(scanid == currentid){
    currentid = 0;
    printf("logout and flash red LED");
    printf("\n");
    }else{
        //if a new user is logging in, set their id to be the current user
    currentid = scanid;
    printf("login and flash green LED");
    printf("\n");
    }

    //close the nfc device and free the pointers
      nfc_close(pnd);
      nfc_exit(context);

    //return the current user
    return currentid;
```

```
}

// ********************************************************************

int GetMaxWorkout(int userID)
{
    //gives info about database to login
    #define dbhost "localhost"
    #define dbuser "root"
    #define dbpass "gymtron"
    #define dbdatabase "Gymtron_DB"
    //#define MAX_DATA_SIZE 23
    //#define SQL_IN_SIZE 94

    //initializing pointers for the database query
    MYSQL *conn;
    MYSQL_RES *result;
    MYSQL_ROW row;
    MYSQL_FIELD *field;

     //intializing variables for storage
    int maxWO;
    char data[MAX_DATA_SIZE];
    char sql[SQL_IN_SIZE] = "SELECT Workouts FROM GTUser_Data WHERE User_ID=";

    //intializing connection pointer
    conn = mysql_init(NULL);
    //connecting to database using login data
    mysql_real_connect(conn, dbhost, dbuser, dbpass, dbdatabase, 0, NULL, 0);

    //these lines of code construct the query to be executed using c string functions
    snprintf(data, MAX_DATA_SIZE, "%d", userID);
    strcat(sql,data);
    strcat(sql," ORDER BY Workouts DESC");
    //printf("%s", sql);
    //printf("\n");

    //executing the query and storing the records returned
    mysql_query(conn, sql);
    result = mysql_store_result(conn);
    row = mysql_fetch_row(result);
    //maxWO = row[0];
    //printf("%s", row[0] ? row[0] : "NULL");
    //finding the largest workout number
    maxWO = strtol(row[0] ? row[0] : "NULL", NULL, 10);
    /*printf("\n");
    printf("%d", maxWO);
    printf("\n");
    maxWO++;
    printf("%d", maxWO);
    printf("\n");
    */
```

```
    //freeing the result pointer and closing the connection
    mysql_free_result(result);
    mysql_close(conn);

//passing the max workout back to the main
return maxWO;
}

// ********************************************************************

void ManualLogout()
//ISR triggered when logout button is pressed
{
    delay(100);
    if(digitalRead(LOGOUT_BTN) == 1)
        manualLogout = 1;
}

// ********************************************************************

void ManualSet()
//ISR Routine triggered when set complete button is pressed
{

    int start;
    int debounceTime = 200;
    int curState = digitalRead(LOGGED_IN);//reads register so OK for o/p

    //only do something if currently logged in
    if(curState == 1){
        start = millis();
        while(digitalRead(SET_BTN) == 1){
            if((millis() - start) > debounceTime){
                manualSet = 1;
                SetIncrementFlash();
            }
        }
        digitalWrite(LOGGED_IN, curState);
    }


}

// ********************************************************************

void SetIncrementFlash()
//Indicates to the user that they have successfully pressed the
//LOGGED_IN button
{
    int i;

    for(i=0; i<3; i++){
        digitalWrite(LOGGED_IN, HIGH);
```

```
        delay(300);
        digitalWrite(LOGGED_IN, LOW);
        delay(200);
    }

    digitalWrite(LOGGED_IN, HIGH); //leave HIGH b/c still logged in
}
```

```
#!/bin/bash
#this script uses BlueZ to startup the bluetooth radio that
#is connected to the Raspberry Pi via USB

sudo hciconfig hci0 up
```

```
#!/bin/bash

#This script connects to the bean and reads the temperature
#from two scratch characteristics and then prints them
#to the output stream. c code will call this script and
#collect the values written to the output stream

# "$()" store result of cmd in brackets into variable temp
# temp is stored in scratch 1 (0x0033) and weight is stored in scratch 2 (0x0037)
#D0:39:72:D3:4D:83 is the MAC address of the LightBlue Bean that we used
#gatttool is provided by the BlueZ library to access properties that are advertised
#   by a bluetooth LE device

temp="$(sudo gatttool -b D0:39:72:D3:4D:83 --char-read -a 0x0033)"

if [ "$?" -ne "0" ]
then
    #connection to bean has failed return error code
    echo "6B"
    exit 1
fi

weight="$(sudo gatttool -b D0:39:72:D3:4D:83 --char-read -a 0x0037)"

if [ "$?" -ne "0" ]
then
    #connection to bean has failed return error code
    echo "6B"
    exit 1;
fi

#Start at 33 character and take two chars to get only temperature
temp=${temp:33:2}
weight=${weight:33:3}

echo $temp
echo $weight
```

```
/*
  This sketch reads the temperature and analog weigh input and
  writes them to scratch characteristics 1 and 2 respectively.

  The arduino is only woken fom sleep when a conection to the
  LightBlue Bean is established. After reading the weight and
  temperature the arduino microcontroller is put to sleep to
  conserve battey power.

  Writen By: Kyle Shaw

  Code is based off of example code from the LightBlue Bean
  website

*/

int controlPin = 3;//Pin for weight measurement

void setup() {
  Bean.enableWakeOnConnect( true );//wake the arduino up when a BT connection is made
 //use serial and windows loder application to debug
 // Serial.begin();
  pinMode(controlPin, OUTPUT);
}

void loop() {
  bool connected = Bean.getConnectionState();
  uint8_t buffer[1];
  uint16_t weight;
  ScratchData thisScratch;

  if(connected){
   //Measure weight (AI has 10 bits resolution)
   digitalWrite(controlPin,HIGH);
   //delay(3);
   //Each weigt is 10lbs. Each weight is given range of 51/1024 bits
   weight = (20-(analogRead(A0)/51))*10;
   digitalWrite(controlPin,LOW);

  //adjust calculation for open circuit measurement
  if(weight < 15)
    weight = 10;

  // Serial.println(weight);

   buffer[0]= Bean.getTemperature(); // Returns the temperature in the format "24" in degrees
   celcius)
   Bean.setScratchData(1, buffer, 1);//write the temperature to scratch 1

   buffer[0]= weight;
   Bean.setScratchData(2, buffer, 1);//write the weight scratch 2

   Bean.sleep(100);//turn the arduino off for 100 ms
```

```
  }
  else{
    Bean.sleep(0xFFFFFFFF);//sleep unless woken
  }
}
```