TODD ROGERS
CHRISTOPHER JOHNSON
DARIO BOSNJAK
LEVI BALLING

SMART HOME

COMPUTER ENGINEERING FINAL PROJECT

2012

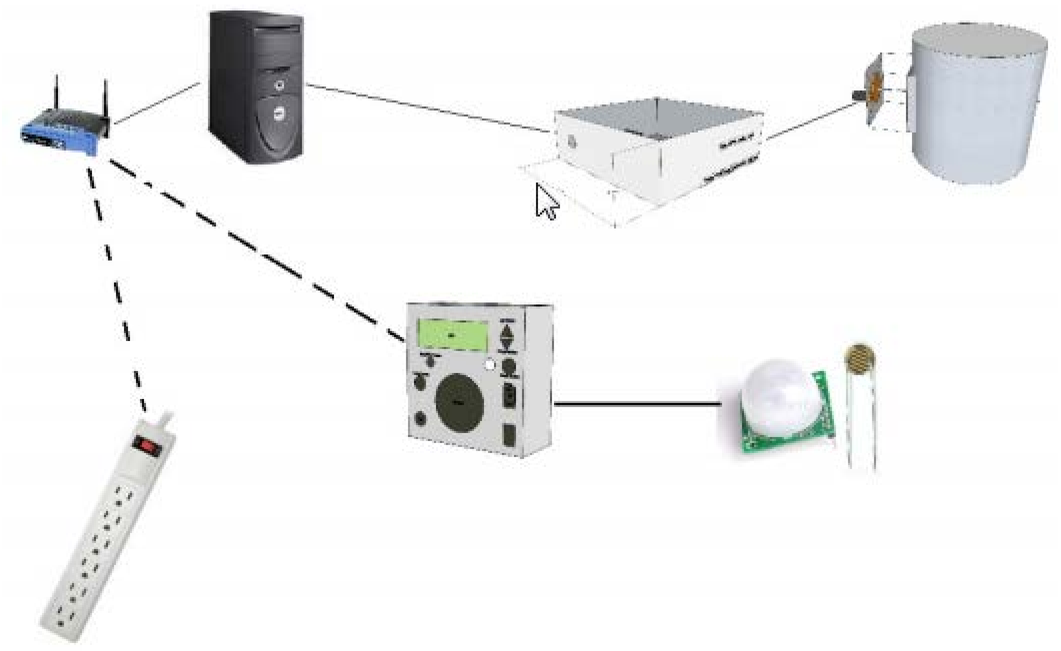

TABLE OF CONTENTS

INTRODUCTION

While the prices of living are going up, the main focus is to involve technology to assist us with maintaining those prices and still provide manageable living costs. With this in mind the Smart Home project will allow us to build and maintain a house that is smart enough to keep the energy levels down while providing the most automation possible. A smart home will take advantage of its environment and allow seamless control whether you are present or away. With a home that has this advantage, you can know that your home is performing at its best in energy performance.

By implementing this project we were able to explore a variety of different fields of engineering, including software programing, PCB design, Wi-Fi TCPIP protocols, Web Server logic design, and other aspects. This project turned out to be providing great insights to both sides of engineering hardware and software vise.

OVERVIEW

OVERALL SOFTWARE FUNCTION

(Everyone discusses the basics of the drivers and protocols created for each portion of the project)

OVERALL HARDWARE FUNCTION

(Levi includes info on PCB Fabrication)

(Todd includes info on hardware interface with outlet control)

(Dario includes info about soil sensor)

Chris includes info about ComPort design)

SERVER (Chris' update)

<u>Hardware</u>

<u>Software</u>

W i - F i F U N C T I O N

INTRODUCTION

In order to communicate over long distances without running wires, we came up with a more convenient way of communicating with our sensors and with controlling different I/O devices by using TCP/IP over Wi-Fi.802.11 g protocol. Data being gathered from sensors such as (heat sensors, light sensors, laser trip wire sensors) is being processed on an Arduino Micro-controller and then broad-casted with an attached WiShield v2.0 to a server over TCP/IP protocol. Arduino has a statically assigned IP address that corresponds to an individual room in the house. Each time a request is made to that IP address a HTML page is returned with implemented functionality. One of the perks of using HTML is that data can be viewed from all of the sensors in one location and also to control remotely Input/Output devices such as power strip plugs.

SOFTWARE FUNCTION

The main functionality of software is responsible to monitor the changes in attached hardware and also to to initiate controlling statements that depending on the log-in involved would trigger an invent based on that log-in.

- Monitor analog inputs go gather moisture change in ground and light intensity in order to turn on a digital output.
- Create software serial communication in order to communicate with another Arduino responsible for controlling power strip plugs
- Create a Arduino hosted web server responsible for keeping track of sensor information and current states of attached devices.
- Gather and store current sensor information and store it for clients to see.

Software is based on Arduino code that is based on C programming language. It consists of libraries that create Web Servers for Arduino MCU's and also libraries responsible of setting up software serial communication to another Arduino.

Arduino consists of code that initiates HTML page over the web server that is supported on Arduino. Once the page is established series of loops are ran on the arduino to constantly measure different sensor values that are hard wired in to the arduino over the analog inputs. Once those values are known to the code they are stored in variables that can be displayed over HTML. Once an request is made to the arduino hosting the web page the returned HTML page consists of fields populated by sensor values, and options to manually turn on power strip plugs. Software side also takes advantage of digital inputs on the arduino by utilizing them to monitor values coming from motion detector sensor and laser trip wire sensor. If these events are triggered they automatically get stored in a variable that is translated and displayed on the web page for clients to examine. Another feature that is implemented by software side is that we are able to do logic when it comes to manipulating I/O devices. One of the features involves ability to check the moisture of soil and also check the time of day by monitoring the light intensity and if both parameters meet the logic criteria a digital enable gets sent which triggers the hardware side and starts the process designed by hardware. Web Server will be able to listen on requests that are being sent over the assigned IP address in this case http://192.168.1.102 and upon receiving will display current state of the sensors and I/O devices. In case an I/O device needs to be triggered by logic already pre-determined an request of http://192.168.1.102/?LED0 … LED3 will be sent which will cause an event to be triggered and will execute a specified block of code responding to that URL page, and in most cases turning on an appliance connected to that power plug being controlled by that URL. Actual code used in the project is attached at the bottom of the document in *Appendix A*.

HARDWARE FUNCTION

So far software is responsible for doing most of the work when it comes to communication between the devices and TCP/IP protocol. In order to successfully allow the control and monitoring of different devices I am implementing Arduino Duemilanove MCU with a WiShield v2.0 Wi-Fi wirelles adapter network card that supports static IP address assignments. The power usage of the Wi-Fi Shield with Arduino is low. It requires 5-7 volts. Wi-Fi communication is done over 802.11b at 1Mbps throughput speeds with Netgear wireless N router. A 5V line is connected to the 5V pin on the arduino and the common ground is shared between the Arduino running the power strip outlets. This way noise over the software serial is dramatically limited and communication is enhanced between the two MCU controllers.

- Sleep mode: 250µA
- Transmit: 230mA
- Receive: 85mA
  - SPI
    - o Slave select (SS) : Arduino pin 10 (port B, pin 2)
    - o Clock (SCK) : Arduino pin 13 (port B, pin 5)
    - o Master in, slave out (MISO) : Arduino pin 12 (port B, pin 4)
    - o Master out, slave in (MOSI) : Arduino pin 11 (port B, pin 3)
  - Interrupt (Uses only one of the following, depending on jumper setting)
    - o INT0 : Arduino pin 2 (port D, pin 2)
    - o DIG8 : Arduino pin 8 (port B, pin 0)
  - LED : Arduino pin 9 (port B, pin 1)
    - o To regain use of this pin, remove the LED jumper cap
  - 5V power
  - GND

WiShield v2.0

Figure 1

WiShield will be placed inside a power strip box where it will be in close proximity with the MCU responsible for controlling the power strip outlets. Due to noise disturbance an Linksys Wi - Fi antenna has be soldered on the WiShield and placed outside the box to increase Wi-Fi strength signal between the router and the WiShield.

S P R I N K L E R   S Y S T E M F U N C T I O N A L I T Y

INTRODUCTION

In the attempt to make the house as automated as possible a sprinkler control system has been invented that is responsible on turning on the sprinkler based on some preset parameters. It is well known that it can be expensive to maintain green grass and the last thing you want to do is to waste water when it's not necessary to water the grass. In attempt to resolve this issue, we have implemented a soil moisture sensor that measures the soil conductivity and reports it back to the Arduino MCU controller and at the same time it does that, through photo resistive it gathers the information about light intensity to determine if its light or day. This approach will limit the sprinkler system only to be ran during the night and only if the soil moisture is less than a preset value.

SOFTWARE FUNCTION

Coded in Arduino code based on C language, through built in hardware by Arduino we are able to monitor the analog input values of pins 0 and 1. Values that are coming in through the pin 0 are values that are gathered from the soil moisture sensor. Sensor is connected with one wire to analog input 0 on the arduino board; values coming in through this input are stored in a integer variable that ranges between 0 to 700 conductivity based on the moisture in the soil. In order to come up with these values we connected the sensor to the arduino and coded it to monitor the change in the values when sensor was completely dry and then when it's placed in a cup of water. Once desired values were achieved and observed, they would be pre coded in the function to be compared against. Analog input 1 is responsible for monitoring the change in light intensity received from the photo cell. Once the light is off the value was about 100 and with light on the value went in to 900 ohm resistance. After determining these values a loop was created that would every 10 min check these values and based on them decide if it should turn on the sprinkler system. In case the value for light intensity of analog pin 1 was below 200 and moisture sensor was less than 300 a digital pin 6 would be turned on to send a High output to the relay to turn it on and to start the sprinkler system.

HARDWARE FUNCTION

The following layout describes how the Sprinkler System will be automated in order to function through the WiShield that is attached to the Arduino. *Figure 2* explains the layout of the implementation.

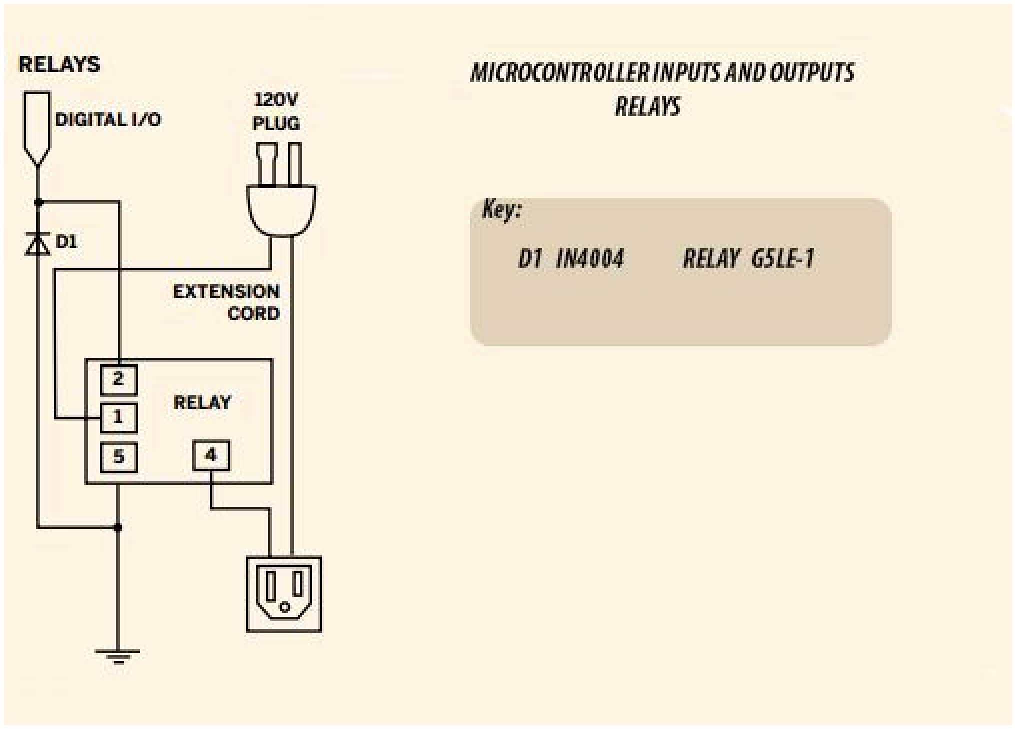

Figure 2.

Relays will be controlled with digital outputs sent from the Arduino once the webserver initiates a certain command, that command will be based upon two different factors. One of the factors is the exact based on soil moisture level in the ground and the second is based on the time of the day. If the soil moisture is under the acceptable level and its night turn on the sprinkler system. Else leave it off until one the right parameters are acquired. *Figure 3* illustrates the next two steps necessary in controlling the sprinkler system.

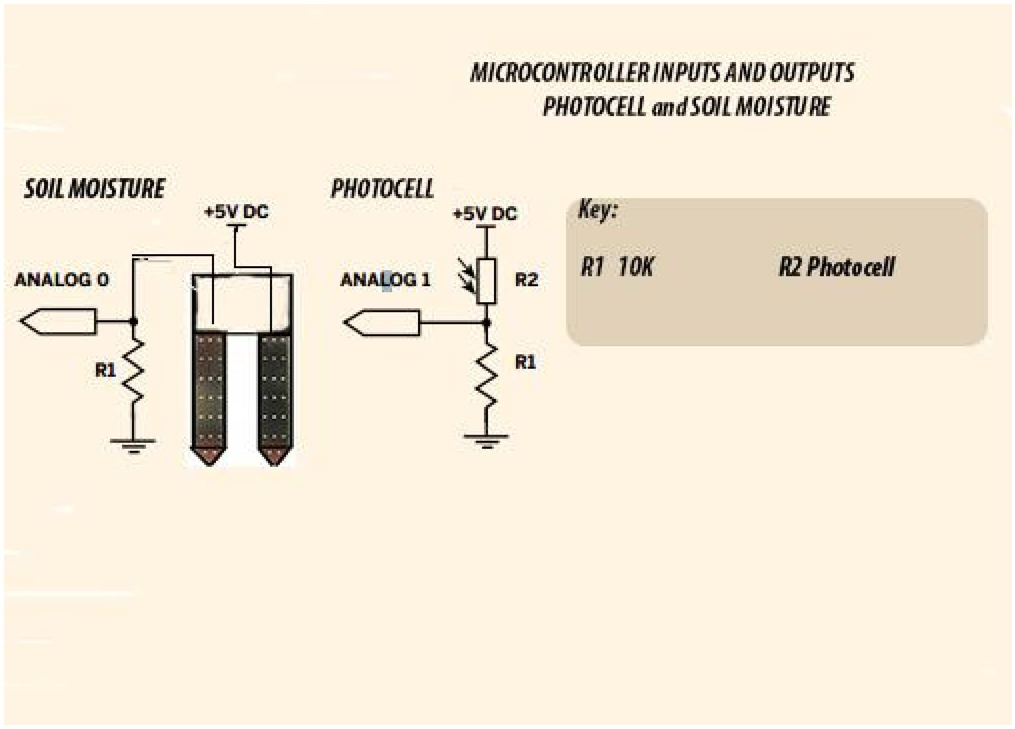

Figure 3.

OUTLET CONTROL (Todd's update)

<u>Software</u>

<u>Hardware</u>

MOTION SENSOR (Chris' update)

Software

Hardware

SOIL SENSOR (Dario's update)

Software

Hardware

OUTLET CONTROL (Todd's update)

Software

Hardware

TEMPERATURE SENSOR (Levi's update)

Software

Hardware

BLOWERS (Levi's update)

Software

Hardware

HVAC (Levi's update)

Software

Hardware

STEPPER MOTOR CONTROL (Levi's update)

Software

Hardware

DAMPER CONTROL (Levi's update)

Software

Hardware

SPRINKLER CONTROL (Levi's update)

Software

Hardware

GARAGE CONTROL (Levi's update)

Software

Hardware

CONCLUSION

REFERENCES AND ACKNOWLEDGEMENTS

APPENDIXES

Appendix A

```
/*
* A Web page used to control all of the information necessary for control of the whole wifi system
* The Wireless configuration parameters were borrowed from Arduino.cc in order to
* get the Wifi Board to comunicate with the router.
*/
/*
Libraries needed for the wifi implementation
*/

#include <WiServer.h>
#include <string.h>
#include <SoftwareSerial.h>

#define WIRELESS_MODE_INFRA   1
#define WIRELESS_MODE_ADHOC   2

//#define ledPin1 8 //changed this from 5 to 8
//#define ledPin2 6
//#define ledPin3 7
/*
Variable declaration for Software serial and also for the analog and
digital input values
*/
String str;
String txtLights;
boolean stringComplete;
int LaserValue = 0;
int MotionValue = 0;
int LaserPin = 6;
int TripPin = 7;

//Declaring software serial
SoftwareSerial mySerial(4, 5); //RX, TX

/*
This part of the code was borrowed from the library created by the Arduino.cc team
in order to get the WiShield v2.0 connected to the wifi router.
*/
// Wireless configuration parameters ----------------------------------------
unsigned char local_ip[] = {192,168,1,102};  // IP address of WiShield
unsigned char gateway_ip[] = {192,168,1,1};             // router or gateway IP address
unsigned char subnet_mask[] = {255,255,255,0};        // subnet mask for the local network
const prog_char ssid[] PROGMEM = {"SmartHome"};                          // max 32 bytes

unsigned char security_type = 0;     // 0 - open; 1 - WEP; 2 - WPA; 3 - WPA2

// WPA/WPA2 passphrase
const prog_char security_passphrase[] PROGMEM = {"DeadBeef"};                  // max 64 characters

// WEP 128-bit keys
// sample HEX keys
prog_uchar wep_keys[] PROGMEM = {       0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0a, 0x0b,
0x0c, 0x0d,          // Key 0
                                                                                         0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00,        0x00,   // Key 1
                                                                                         0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00,        0x00,   // Key 2
                                                                                         0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00,        0x00      // Key 3
                                                                                 };

// setup the wireless mode
// infrastructure - connect to AP
// adhoc - connect to another WiFi device
unsigned char wireless_mode = WIRELESS_MODE_INFRA;

unsigned char ssid_len;
unsigned char security_passphrase_len;

// End of wireless configuration parameters ----------------------------------------

boolean states[4]; //Led states
char stateCounter, tmpStrCat[64], stateBuff[4], numAsCharBuff[2], ledChange; //Variable declarations

/*
Check the conversion to display on the page so we can see if the url is on or off
*/
void boolToString (boolean tests, char returnBuff[4]){
  returnBuff[0] = '\0';
  if (tests){
    strcat(returnBuff, "on");}
  else{
    strcat(returnBuff, "off");}
}
/*
State print counter
*/
```

```
void printStates(){
            for (stateCounter = 0 ; stateCounter < 4; stateCounter++){
          boolToString(states[stateCounter], stateBuff);
          int count = stateCounter;

/*
Software serial implementation in order to controll the second arduino to
perform its desired controlls
*/
          //mySerial.begin(9600);
          mySerial.print("turn ");
          mySerial.print(stateBuff);
          mySerial.print(" ");
          mySerial.print(count);
          mySerial.print('\n');
          mySerial.print("allpower\n");
        //  if(analogRead(1)<810){
         //mySerial.print("turn ");
         //mySerial.print(stateBuff); 3\n");
         //Serial.print("turn on 3\n");
       //}else{
         //mySerial.print("turn off 3\n");
         //Serial.print("turn off 3\n");
            //}
/*
This code can be ignored as it is only there for troubleshooting purposes
*/
          Serial.print("turn ");
          Serial.print(stateBuff);
          Serial.print(" ");
          Serial.println(count);
            }
            }
/*
This part of the code can also be ignored as its purpose was to see
if the LED would function based on the desired parameters.
*/
//void writeStates()
//{
            //set led states
//        digitalWrite(ledPin1, states[0]);
//        digitalWrite(ledPin2, states[1]);
//        digitalWrite(ledPin3, states[2]);
//}
/*void readSerial()
{
  while(mySerial.available()){
            char inChar = (char)mySerial.read();
```

```
              str += (String)inChar;
   if(inChar == '\n'){
     stringComplete = true;
              break;
    }

  }
  if (stringComplete==true){
  Serial.print(str);
  }
  str = "";
}*/
// Here the page gets served
/*
This part of the code serves the webpage, its a boolean funciton
that is responsible to run the code and then be served on the server*/
boolean sendPage(char* URL) {

    printStates();
  //  writeStates();



  //Using to check if the URL needs to change acting like a refresh button
  //if the url has not changed the page will not change either
  if (URL[1] == '?' && URL[2] == 'L' && URL[3] == 'E' && URL[4] == 'D') //url has a leading /
  {
              ledChange = (int)(URL[5] - 48); //get the led to change.

              for (stateCounter = 0 ; stateCounter < 4; stateCounter++)
              {
              if (ledChange == stateCounter)
              {
        states[stateCounter] = !states[stateCounter];
          //  Serial.print("Have changed ");
            Serial.println(ledChange);
              }
              }

              //after having change state, return the user to the index page.
    WiServer.print("<HTML><HEAD><meta http-equiv='REFRESH' content='0;url=/'></HEAD></HTML>");
              //Send over software serial commands in order to display the current state of the current sensor and also the
current state of the power strip
    mySerial.print("allpower\n");
              return true;
  }

  if (strcmp(URL, "/") == false) //why is this not true?
```

```
 {
    while(mySerial.available()){
           char inChar = (char)mySerial.read();
           str += (String)inChar;
     if((str.length() > 8) || (inChar == '\n')){
    stringComplete = true;
           break;
           }


}
           if (stringComplete==true){
   Serial.print(str);
}

   WiServer.print("<html>");

           // WiServer.print("<body><center>Please select the led state:<center>\n<center>");
           for (stateCounter = 0; stateCounter < 4; stateCounter++) //for each led
           {
     numAsCharBuff[0] = (char)(stateCounter + 49); //as this is displayed use 1 - 3 rather than 0 - 2
     numAsCharBuff[1] = '\0'; //strcat expects a string (array of chars) rather than a single character.
                                     //This string is a character plus string terminator.

     tmpStrCat[0] = '\0'; //initialise string
     strcat(tmpStrCat, "<a href=?LED"); //start the string
     tmpStrCat[12] = (char)(stateCounter + 48); //add the led number
     tmpStrCat[13] = '\0'; //terminate the string properly for later.

     strcat(tmpStrCat, ">Led ");
     strcat(tmpStrCat, numAsCharBuff);
     strcat(tmpStrCat, ": ");

     boolToString(states[stateCounter], stateBuff);
     strcat(tmpStrCat, stateBuff);
     strcat(tmpStrCat, "</a> "); //we now have something in the range of <a href=?LED0>Led 0: Off</a>

     WiServer.print(tmpStrCat);
           }

           //Declare the variables to find the temperature
 float temp_in_celsius = 0;
 float temp_in_kelvin=0;
 float temp_in_fahrenheit=0;

//Reads the input and converts it to Kelvin degrees
 temp_in_kelvin = analogRead(0) * 0.004882812 * 100;
 //Converts Kelvin to Celsius minus 2.5 degrees error
```

```
temp_in_celsius = temp_in_kelvin - 2.5 - 273.15;
temp_in_fahrenheit = ((temp_in_kelvin - 2.5) * 9 / 5) - 459.67;
//Temperature
            float h = temp_in_kelvin;
            float t = temp_in_celsius;
              float f = temp_in_fahrenheit;
      //float laser = analogRead(1);
            String txt;

      //Check to see if the Laser is tripped or not
      if((analogRead(1)<810) && (analogRead(2)<300)){
            txt = "Sprinkler System is on";
      }else{
            txt = "Sprinkler System is off";
            }
      //Display if the lights are on or off
      if((digitalRead(LaserPin) == HIGH) && (digitalRead(TripPin) == HIGH)){
        txtLights = "Lights are on";
            }else{
            txtLights = "Lights are off";
            }

      LaserValue = digitalRead(LaserPin);
      MotionValue = digitalRead(TripPin);

            // check if returns are valid, if they are NaN (not a number) then something went wrong!
            if (isnan(t) || isnan(h)) {
                          WiServer.print("<html>");  //Here is the code for the html page
               WiServer.print("Failed to read from Sensor Input");
                          WiServer.println("             </html>");
            } else {
                          WiServer.print("<html>");
                          //WiServer.print("<center><H2>Room 1 Temperature Statistics: ID: 1</H2><br><br><br>");

      WiServer.print("Temperature Kelvin: ");
      WiServer.print(h);
      WiServer.print(" T\t");
      WiServer.print("Temperature Celsius ");
                WiServer.print(t);
      WiServer.println(" *C");
      WiServer.print("Temperature Fehrenhite ");
      WiServer.print(f);
      WiServer.println(" *F");
      WiServer.println("        ");
      WiServer.print("</center>");
            WiServer.print("<center>");
      WiServer.print("Sprinkler Detection: ");
      WiServer.print(txt);
```

```
        WiServer.println("        ");
              // WiServer.print(laser);
        WiServer.println("Testing Current: ");
        WiServer.println(str);
              str = "";
        WiServer.println("</center>");
        WiServer.print("<center>");
        WiServer.print("Light Status: ");
        WiServer.print(txtLights);
        WiServer.print("</center>");
        WiServer.print("<center>");
              WiServer.print("Trip Wire Status: ");
        WiServer.print(LaserValue);
        WiServer.print("</center>");
        WiServer.print("<center>");
         WiServer.print("Motion Sensor Status: ");
        WiServer.print(MotionValue);
        WiServer.print("</center>");
        WiServer.print("</html></center>");     // URL was recognized
              return true;
              }
              // URL not found
              return false;

        WiServer.print("</html> ");
              return true;
              //str = "";
   }
}

void setup() {
  // Initialize WiServer and have it use the sendMyPage function to serve pages
 // pinMode(ledPin1, OUTPUT);
  //pinMode(ledPin2, OUTPUT);
  //pinMode(ledPin3, OUTPUT);
  pinMode(3, OUTPUT);

  // Enable Serial output and ask WiServer to generate log messages (optional)
 // Serial.begin(57600);

  WiServer.enableVerboseMode(true);
  Serial.begin(9600);
              while (!Serial) {
              ; // wait for serial port to connect. Needed for Leonardo only
  }
  mySerial.begin(9600);
  WiServer.init(sendPage);
  states[0] = false;
  states[1] = false;
  states[2] = false;
}

void loop(){

 // if (mySerial.available())
   //   mySerial.print("turn on 1\n");
              //  if (mySerial.available())
              // Serial.write(mySerial.read());
      //mySerial.print("turn on 1\n");

  if((analogRead(1)<810)&&(analogRead(2)<200)){
          digitalWrite(3, HIGH);
          //mySerial.print("turn on 3");
              // mySerial.print('\n');
              }else{
          digitalWrite(3, LOW);
              // mySerial.print("turn on 3");
              // mySerial.print('\n');
              }
  // Run WiServer
  WiServer.server_task();

  delay(10);
}
```