**Wi-Fi & Automated Sprinkler System**

## INTRODUCITON

Implementing an Arduino Microcontroller will allow us to gather the data from different sensors throughout the house (heat sensors, light sensors, laser trip wire) and broadcast the data to a server that will be set up using Wi-Fi Interface. A Wi-Fi shield for Arduino will be connected to the server using the TCP/IP protocol with an assigned IP address that the server will use to listen to the information on that IP. We are planning on running an Apache server where we could store the data that comes in from the micro controllers. This data will be formatted and displayed on a secure website which is accessible from any internet connections. Using the Wi-Fi we will be able to send and receive data. Once we get the information from a temperature sensor in one of the rooms, and we decide to turn on the fans in the room to make it cooler, we would send a command to that micro controller that will be in charge of running the fans, to start to pump the cold air inside that room. Below is a diagram to illustrate Wi-Fi operations in the house. *Figure 1* – Shows how the overall look of the Wi-Fi will look like.


***Figure 1***

## FUNCTIONAL SPECIFICATION

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.

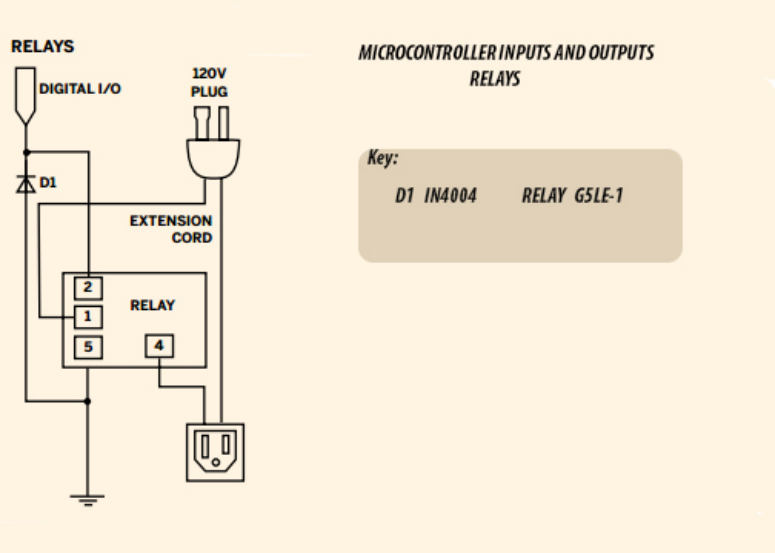


***Figure 2***

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.

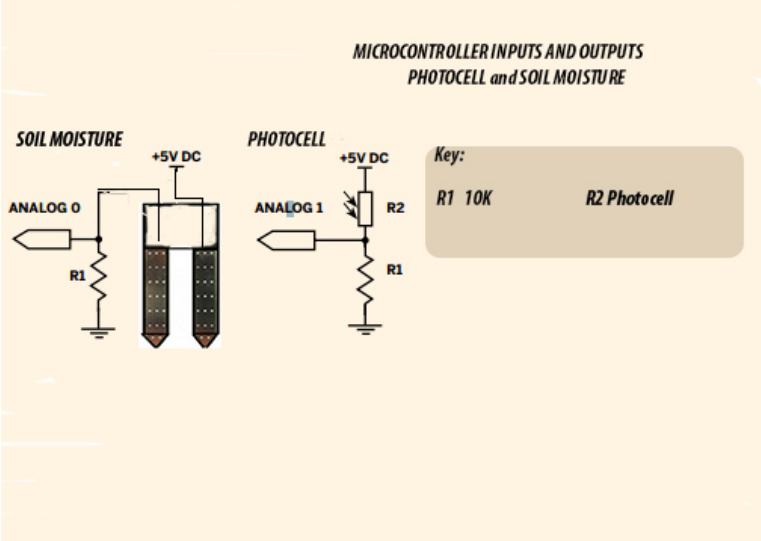


***Figure 3***

The Wi-Fi portion of the project consists of a number of functions. Its main purposes is to be in a a location that can benefit to all of the sensors and control outputs but still available to the end user at their finger prints. For that to work a number of coded functions have to be implemented in order to take the full potential of the MCU in the account. *Appendix A* consists of the code written in Arduino code which is based on C language which is responsible for controlling some aspects of the house over Wi-Fi.

## REFERENCES AND ACKNOWLEDGEMENTS

## 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);
}
```