Greenhouse Control System

Murad Qasaimeh

Mamoun Al-Mardini

Introduction:

Agriculture is essential in our life since early civilizations and it is the source for our food. Greenhouse plays vital part in agriculture sector which can be used to grow plants under controlled climatic conditions for optimum production.

The objective of this project is to develop a multiple automatic greenhouses irrigation system with sublamentary functionality such as cooling and providing light in the absense of sun. These greenhouses are all connected via a wireless communication, and send the sensor readings to a centralized station.

Functionality of the project:

The goal of this project is to develop a distributed greenhouse control system, with the following functionalities:

1. Monitoring the actual measurements (temp , humidity , Light) in GUI.
2. Remotely turn ON/OFF the cooler, artificial light, and water valve.
3. Control the water valves ,cooler, and light based on time schedules.
4. Control the water valves ,cooler, and light based on given thresholds.

Hardware and Software components:

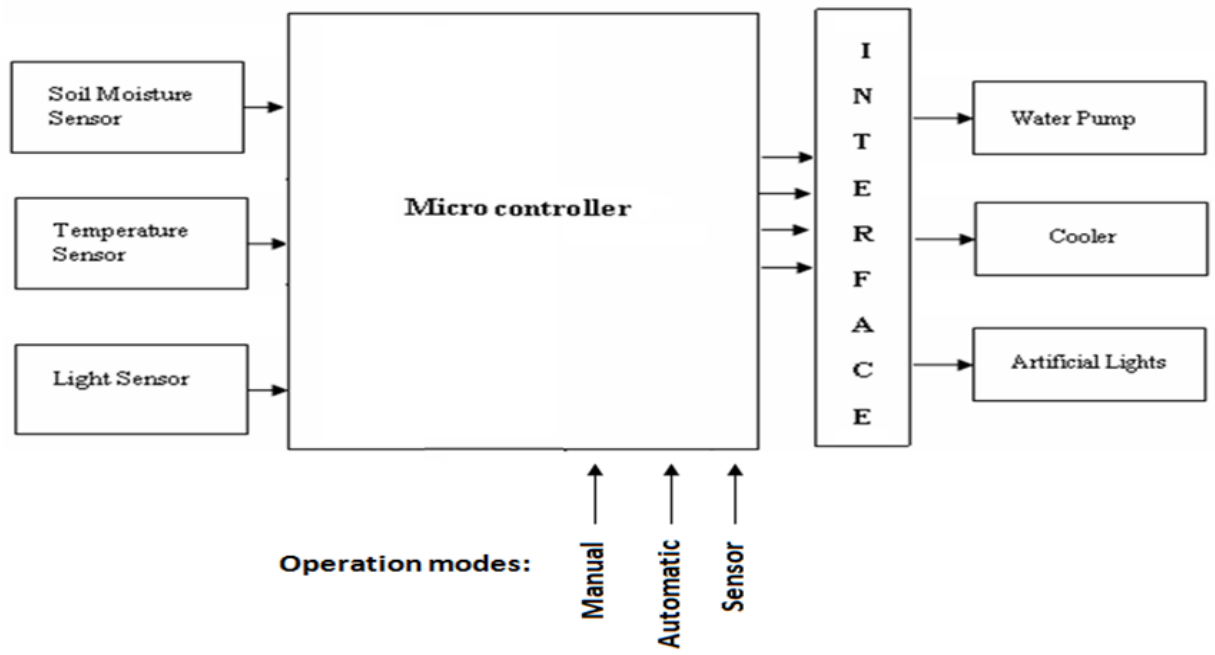
Hardware component:

1. Arduino Micro http://www.sparkfun.com/products/9266
2. Temperature sensor http://www.sparkfun.com/products/8777
3. Lilypad Light sensor http://www.sparkfun.com/products/8464)
4. XBee breakout board https://www.sparkfun.com/products/8937
5. XBee Explorer USB http://www.sparkfun.com/products/8687
6. XBee Pro Chip Antenna http://www.sparkfun.com/products/8690
7. LilyPad LED White http://www.sparkfun.com/products/8735
8. LilyPad Button Board https://www.sparkfun.com/products/8776
9. Humidity Sensor http://www.digikey.com/us/en/ph/Honeywell/hih.html

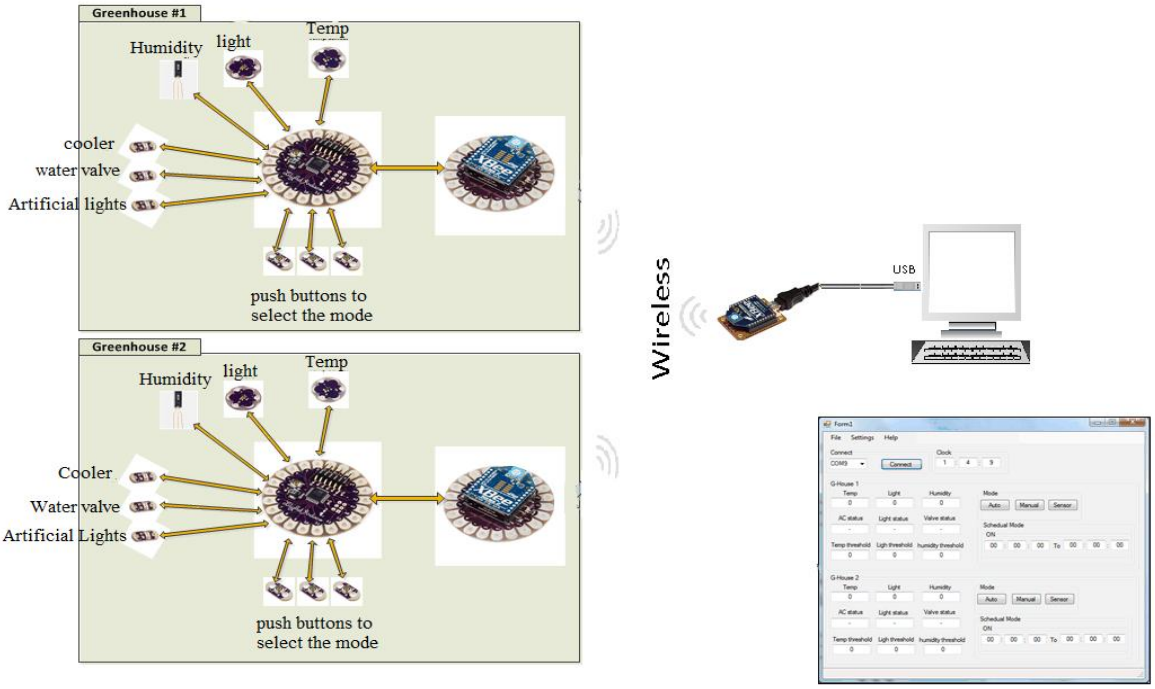
Software component:

1. Microsoft visual Studio – C#.NET
2. Arduino 1.0 environment.

Block Diagram:



System Architecture:



As seen in the above figure, we have two greenhouses, each one consist of 1 Arduino microcontroller, transmitter, three sensors, three LEDS as indicators for the status of the devices, and three push buttons to select the mood.

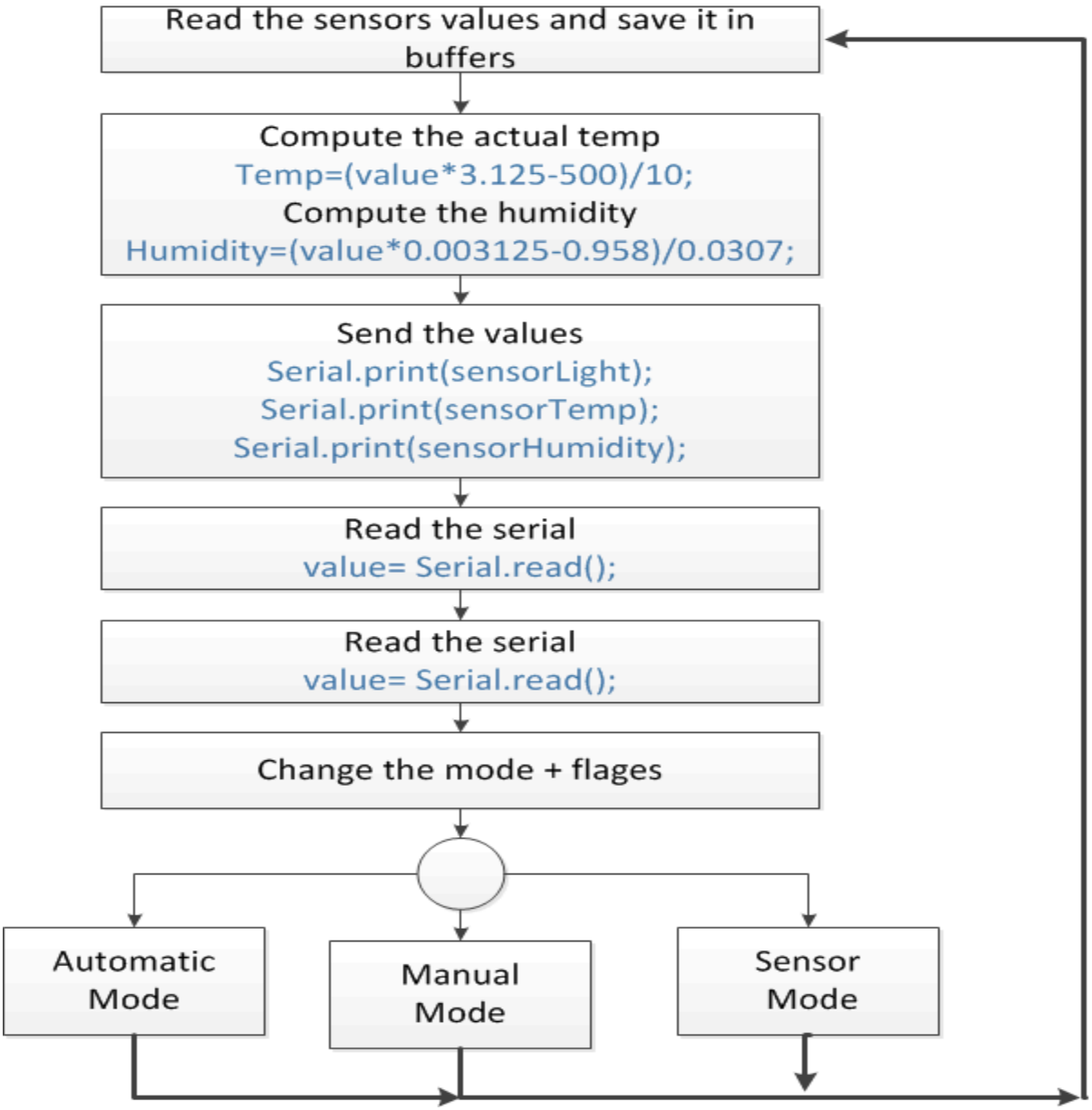
All data are sent wirelessly between the end station using Zigbee devices, and at the end data are displayed on the main station.

Real Circuit

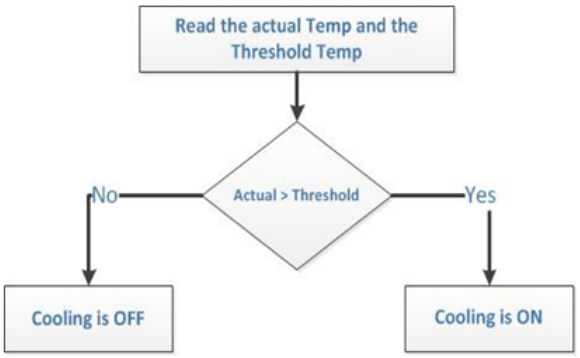
The following picture shows the real circuit, where each plant represents a greenhouse

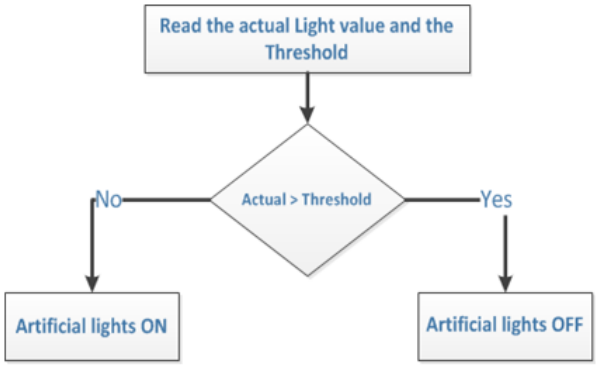
Flowcharts:

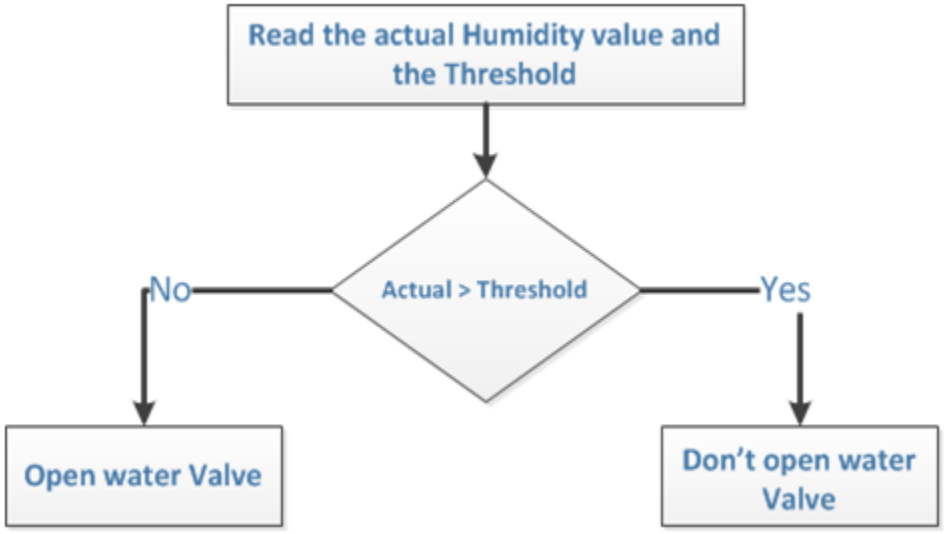
Following flowchart explains how the system works, and how the sensor values are calculated and sent to the main station:



Additional flow charts:



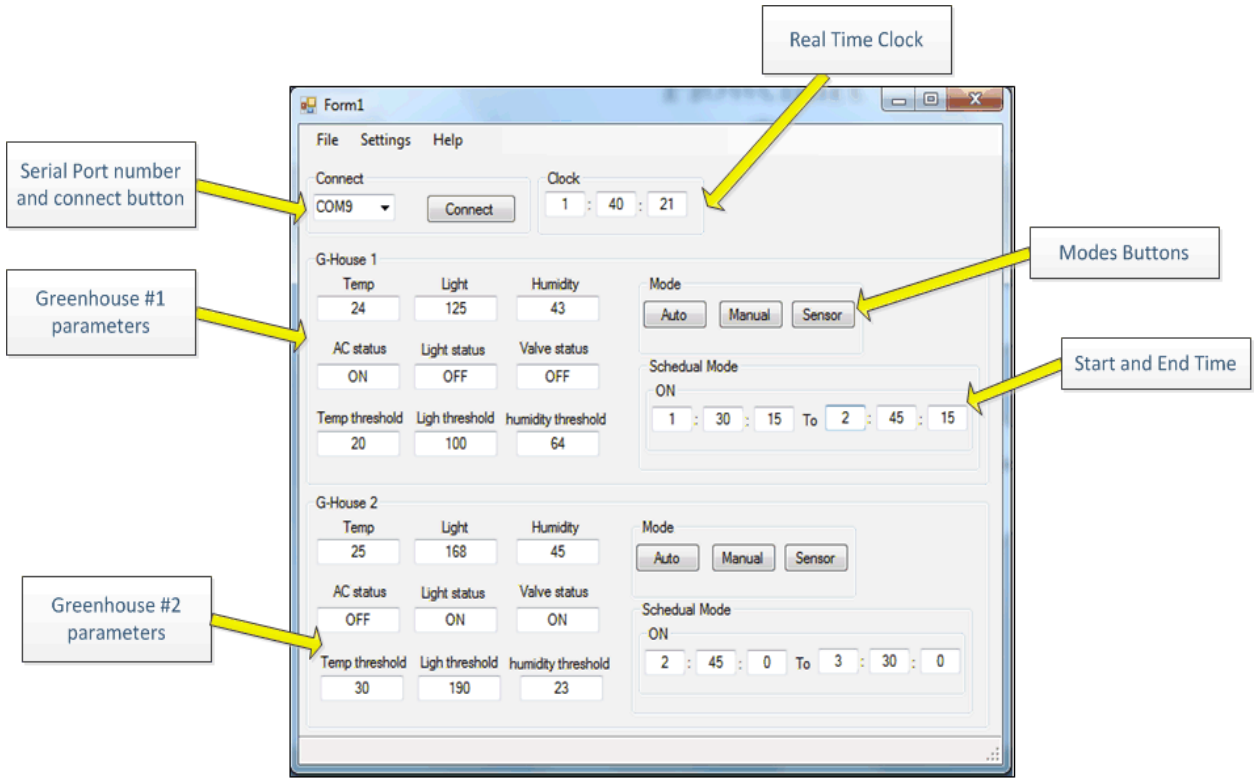




Graphical user interface:

Graphical user interface contains textboxes and buttons for two greenhouses, and real time clock to compare the time in the scheduled mode.

The first set of textboxes contain the sensor readings, and the second set of boxes display the status of the AC, Light and water valve. The last set includes the threshold values.



Conclusion:

In this project, we were able to design a multiple greenhouses irrigation system based on three essential parameters for plant growth, which are temperature, soil moisture, and light intensity.

This system can maintain the proper temperature inside the greenhouse, irrigate the plants when the humidity is less than the threshold value, and turn on artificial lights when there isn't enough light. All that could be done using sensors to read humidity, temperature, and light, and send these readings wirelessly to a base station.

Appendix:

Arduino code:

Code for the first end device:

```
/*
  AnalogReadSerial
 Reads an analog input on pin 0, prints the result to the serial monitor

 This example code is in the public domain.
 */

/*
Light sensor is connected to port A0
Temperature sensor is connected to port A1
Humidity sensor is connected to port A2


*/

int mode=7;
int serialIn;
int flag=0;//used for scheduled(Automatic) irrigation...set by default to 1....when the schedule ends, the main station will set the flag to 0 via serial port.
int serial2;
int flagLight=0;
int flagTemp=0;
int flagHum=0;
int sensorType=0;
int send_flage=0;


void setup() {
  Serial.begin(9600);
  pinMode(11, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(13, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);
   pinMode(5, INPUT);
  pinMode(6, INPUT);
  pinMode(7, INPUT);
```

```
}

void loop() {

  int sensorLight = analogRead(A0);//to get light sensor readings
  int sensorTemp = analogRead(A1); //to get Temp sensor readings
  int sensorHumidity = analogRead(A2);
  char packet[]="-";//to recognize the packet
sensorTemp=sensorTemp/7.5;
//  sensorTemp=(sensorTemp*3.125-500)/10;// temperature conversion equation : Ta=(Vout -V@0c)/Tc
   sensorHumidity=(sensorHumidity*0.003125-0.958)/0.0307;




  /*each packet contains the following parts: [-,(0 , 1 or 2),sensor reading] */


  /*
  following lines are to get the mode from the main station.

  possible modes are :
  0 for Automatic----->when selected LED connected to port 11 will be ON
  1 for Manual   ----->when selected LED connected to port 12 will be ON
  2 for Sensor   ----->when selected LED connected to port 13 will be ON

  */


serial2= Serial.read();

if(serial2==115)
{
  send_flage=serial2;
}

 if(send_flage==115)
  {
  //sending light readings
  Serial.print(packet);

  Serial.print('N');
  Serial.print(sensorLight);
  delay(50);

  //sending temp readings
  Serial.print(packet);
```

```
 Serial.print('K');
 Serial.print(sensorTemp);

 //sending Humidity readings
 Serial.print(packet);

 Serial.print('J');
 Serial.print(sensorHumidity);
 send_flage=0;
 }


//flag value used in Automatic(scheduled) mode    52(4):start of the schedule 53(5) end of the schedule
if(serial2==52 || serial2==53)
{
 flag=serial2;
}
//mode values: 48(0): Automatic mode   49(1): Manual mode    50(2): Sensor mode
if(serial2==48 || serial2==49 || serial2==50)
{
 mode=serial2;
}
/*flagLight is used in the sensor mode to know when the light is above or lower than the predefined
threshold.
if the flagLight is 65(A): ligh is insufficient and artificial light is ON*/
if(serial2==65 || serial2==66)
{
 flagLight=serial2;
}
/*flagTemp is used in the sensor mode to know when the Temp is above or lower than the predefined
threshold.
if the flagTemp is 67(C): Temp is high and cooling is ON*/

if(serial2==67 || serial2==68)
{
 flagTemp=serial2;
}

/*flagHum is used in the sensor mode to know when the Humidity is above or lower than the predefined
threshold.
if the flagHum is 69(E): Humidity is low and cooling is ON*/

if(serial2==69 || serial2==70)
{
 flagHum=serial2;
}
```

```
//sensorType is used in the sensor mode to know which threshold are we receiving from the serial port
// light 54(6)   Temp  55(7)     Humidity  56(8)
if(serial2==54 || serial2==55 || serial2==56)
{
 sensorType=serial2;
}

 //Mode 0 Automatic, 48 represent the ASCII for number
 if (mode==48)
 {
   digitalWrite(8, LOW);
   digitalWrite(9, LOW);
   digitalWrite(10, LOW);


    if(flag==53)
    {
      digitalWrite(8, LOW); //to turn the LED connected to port 10 ON
      digitalWrite(9, LOW);
      digitalWrite(10, LOW);
    }

    if(flag==52)
    {
      digitalWrite(8, HIGH); //to turn the LED connected to port 10 ON
      digitalWrite(9, HIGH);
      digitalWrite(10, HIGH);
    }


    }

//Mode 1 Manual, 49 represent the ASCII for number 1
 else if(mode==49)
 {
   digitalWrite(8, LOW);
   digitalWrite(9, LOW);
   digitalWrite(10, LOW);

   if(digitalRead(5)==LOW)
   {
   digitalWrite(8, HIGH); //to turn the LED connected to port 11 ON
   }

   if(digitalRead(6)==LOW)
   {
   digitalWrite(9, HIGH);
   }
```

```
  if(digitalRead(7)==LOW)
  {
  digitalWrite(10, HIGH);
  }
}

//Mode 2 Sensor, 50 represent the ASCII for number 2
 else if (mode==50)
{

  if (sensorType==54)//light sensor
{


    if(flagLight==66)
    {
      digitalWrite(8, LOW);
    }

    if(flagLight==65)
    {
      digitalWrite(8, HIGH);
    }


    }

    if (sensorType==55)//Temp sensor
{


    if(flagTemp==68)
    {
      digitalWrite(9, LOW);
    }

    if(flagTemp==67)
    {
      digitalWrite(9, HIGH);
    }


    }

    if (sensorType==56)//Hum sensor
{
```

```
    if(flagHum==70)
    {
      digitalWrite(10, LOW);
    }

    if(flagHum==69)
    {
      digitalWrite(10, HIGH);
    }


    }




  }

}//End of the loop
```

Code for the second end device:

```
/*
  AnalogReadSerial
 Reads an analog input on pin 0, prints the result to the serial monitor

 This example code is in the public domain.
 */

/*
Light sensor is connected to port A0
Temperature sensor is connected to port A1
Humidity sensor is connected to port A2


*/

int mode=7;
int serialIn;
int flag=0;//used for scheduled(Automatic) irrigation...set by default to 1....when the schedule ends, the
main

//station will set the flag to 0 via serial port.
int serial2;
int flagLight=0;
int flagTemp=0;
```

```
int flagHum=0;
int sensorType=0;
int send_flag=0;


void setup() {
  Serial.begin(9600);
  pinMode(11, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(13, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);
   /*pinMode(5, INPUT);
   pinMode(6, INPUT);
     pinMode(7, INPUT);*/


}

void loop() {

  int sensorLight = analogRead(A3);//to get light sensor readings
  int sensorTemp = analogRead(A4); //to get Temp sensor readings
  int sensorHumidity = analogRead(A5);
  char packet[]="-";//to recognize the packet



sensorTemp=sensorTemp/7.5;
    //sensorTemp=(sensorTemp*3.125-500);
   // sensorTemp=sensorTemp/10;// temperature conversion equation : Ta=(Vout -V@0c)/Tc
   sensorHumidity=(sensorHumidity*0.003125-0.958)/0.0307;




  /*each packet contains the following parts: [-,(0 , 1 or 2),sensor reading] */

  //sending light readings


  /*
  following lines are to get the mode from the main station.

  possible modes are :
  G for Automatic----->when selected LED connected to port 11 will be ON
  P for Manual   ----->when selected LED connected to port 12 will be ON
```

```
 I for Sensor   ----->when selected LED connected to port 13 will be ON

 */


serial2= Serial.read();

//flag value used in Automatic(scheduled) mode    52(4):start of the schedule 53(5) end of the schedule
if(serial2==89 || serial2==85)
{
 flag=serial2;
}
//mode values: 48(0): Automatic mode   49(1): Manual mode    50(2): Sensor mode
if(serial2==71 || serial2==80 || serial2==73)
{
 mode=serial2;
}
/*flagLight is used in the sensor mode to know when the light is above or lower than the predefined

threshold.
if the flagLight is 65(A): ligh is insufficient and artificial light is ON*/
if(serial2==97 || serial2==98)
{
 flagLight=serial2;
}
/*flagTemp is used in the sensor mode to know when the Temp is above or lower than the predefined

threshold.
if the flagTemp is 67(C): Temp is high and cooling is ON*/

if(serial2==99 || serial2==100)
{
 flagTemp=serial2;
}

if(serial2==114)
{
 send_flag=serial2;
}

/*flagHum is used in the sensor mode to know when the Humidity is above or lower than the predefined

threshold.
if the flagHum is 69(E): Humidity is low and cooling is ON*/

if(serial2==81 || serial2==79)
{
 flagHum=serial2;
```

```
}

//sensorType is used in the sensor mode to know which threshold are we receiving from the serial port
// light 54(6)   Temp  55(7)     Humidity  56(8)
if(serial2==82 || serial2==86 || serial2==77)
{
 sensorType=serial2;
}


 if(send_flag==114)
 {

   Serial.print(packet);
 Serial.print('X');
 Serial.print(sensorLight);

 //sending temp readings
 Serial.print(packet);
 Serial.print('Y');
 Serial.print(sensorTemp);
 //sending Humidity readings
 Serial.print(packet);

 Serial.print('Z');
 Serial.print(sensorHumidity);
 send_flag=0;
 }




 //Mode 0 Automatic, 48 represent the ASCII for number
 if (mode==71)
 {
    if(flag==85)
    {
       digitalWrite(8, LOW); //to turn the LED connected to port 10 ON
        digitalWrite(9, LOW);
         digitalWrite(10, LOW);

    }

    if(flag==89)
    {
       digitalWrite(8, HIGH); //to turn the LED connected to port 10 ON
       digitalWrite(9, HIGH);
```

```
      digitalWrite(10, HIGH);
    }


    }

 //Mode 1 Manual, 49 represent the ASCII for number 1
  else if(mode==80)
  {
/*    digitalWrite(8, LOW);
    digitalWrite(9, LOW);
    digitalWrite(10, LOW);*/
    /*
    if(digitalRead(5)==HIGH)
    {
    digitalWrite(8, HIGH); //to turn the LED connected to port 11 ON
    }

    if(digitalRead(6)==HIGH)
    {
    digitalWrite(9, HIGH);
    }

    if(digitalRead(7)==HIGH)
    {
    digitalWrite(10, HIGH);
    }*/
    digitalWrite(10, HIGH); //to turn the LED connected to port 11 ON

  }

 //Mode 2 Sensor, 50 represent the ASCII for number 2
  else if (mode==73)
  {
    if (sensorType==82)//light sensor
  {


    if(flagLight==98)
    {
      digitalWrite(8, LOW);
    }

    if(flagLight==97)
    {
      digitalWrite(8, HIGH);
    }
```

```
    }

    if (sensorType==86)//Temp sensor
  {


    if(flagTemp==100)
    {
      digitalWrite(9, LOW);
    }

    if(flagTemp==99)
    {
      digitalWrite(9, HIGH);
    }


    }

    if (sensorType==77)//Hum sensor
  {


    if(flagHum==81)
    {
      digitalWrite(10, LOW);
    }

    if(flagHum==79)
    {
      digitalWrite(10, HIGH);
    }


    }




  }

}//End of the loop
```

Graphical user interface code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Diagnostics;
using System.IO;
using System.Text.RegularExpressions;
using System.Globalization;
namespace GUI
{
    delegate void updateLabelTextDelegate(string newText);

    public partial class Form1 : Form
    {
        DateTime time;
        int mode = -1;
        int mode2 = -1;

        public Form1()
        {
            InitializeComponent();
            time = DateTime.Now;
            timer1.Enabled = true;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            try
            {
                if (serialPort1.IsOpen != true)
                {
                    serialPort1.PortName = comboBox1.Text.ToString();
                    serialPort1.Open();
                    button1.Text = "Disconnect";
                    timer3.Enabled = true;
                }
                else
                {
                    serialPort1.Close();
                    button1.Text = "Connect";
                }
            }
            catch (Exception t)
            { MessageBox.Show(t.Message.ToString()); }
        }
        string Separator = "-";
        int counter = 0;
        private void serialPort1_DataReceived(object sender,
System.IO.Ports.SerialDataReceivedEventArgs e)
        {
            try
            {
```

```
                counter++;
                //string data = serialPort1.ReadLine();

              string data=serialPort1.ReadExisting();
            // update();
                String[] dtParsing = Regex.Split(data, Separator);

                foreach (string strpacket in dtParsing)
                {
                    //update("P= "+strpacket+"\n");
                    string id = strpacket.Substring(0, 1);
                    //update("id= " + id.ToString() + "\n");
                    if (id == "N")
                    {
                        int light_val =
Convert.ToInt32(strpacket.Substring(1, strpacket.Length - 1));
                       // update("light= " + light_val.ToString() + "\n");
                        update_Light(light_val.ToString());
                    }
                    else if (id == "K")
                    {
                        int Temp_val = Convert.ToInt32(strpacket.Substring(1,
strpacket.Length - 1));
                        Temp_val = Convert.ToInt32((Temp_val /7.75));
                        //if (Temp_val < 30)
                        {
                            //Debug.Print("x= "+x.ToString()+"\n");
                            // double x1 = (Convert.ToDouble(Temp_val * 5) /
1023.0);
                            //Debug.Print("x1= " + x1.ToString() + "\n");
                            //  double Temp_val1 =
(Convert.ToDouble(Temp_val) - 0.5) * 100;
                            update_temp(Temp_val.ToString());
                           // update_temp(Temp_val.ToString());

                        }

                    }
                    else if (id == "J")
                    {
                        int Hum_val = Convert.ToInt32(strpacket.Substring(1,
strpacket.Length - 1));

                        Hum_val = Convert.ToInt32((Hum_val * 0.003125 -
0.958) / 0.0307);

                        update_Humidity(Hum_val.ToString());

                    }else if (id == "X")
                    {
                        int light_val =
Convert.ToInt32(strpacket.Substring(1, strpacket.Length - 1));
                        // update("light= " + light_val.ToString() + "\n");
                        update_Light2(light_val.ToString());
                    }
                    else if (id == "Y")
                    {
```

```
                        int Temp_val = Convert.ToInt32(strpacket.Substring(1,
strpacket.Length - 1));
                        Temp_val = Convert.ToInt32((Temp_val/7.75));
                        //  if (Temp_val < 1023)
                        {
                            //Debug.Print("x= "+x.ToString()+"\n");
                            // double x1 = (Convert.ToDouble(Temp_val * 5) /
1023.0);
                            //Debug.Print("x1= " + x1.ToString() + "\n");
                            //  double Temp_val1 =
(Convert.ToDouble(Temp_val) - 0.5) * 100;
                            if (Temp_val > 17)
                            {
                                update_temp2(Temp_val.ToString());
                            }

                        }

                    }
                    else if (id == "Z")
                    {
                        int Hum_val = Convert.ToInt32(strpacket.Substring(1,
strpacket.Length - 1));
                        Hum_val = Convert.ToInt32((Hum_val * 0.003125 -
0.958) / 0.0307);
                        update_Humidity2(Hum_val.ToString());

                    }
                }
            }
            catch (Exception f) {
               // MessageBox.Show(f.Message.ToString());
            }
        }


        private void update_temp(string newText)
        {

           // Debug.Print("x11= " +x1.ToString() + "\n");
           // newText = Convert.ToInt32(x1.ToString()).ToString();
            if (textBox1.InvokeRequired)
            {
                updateLabelTextDelegate del = new
updateLabelTextDelegate(update_temp);
               textBox1.Invoke(del, new object[] { newText });
            }
            else
            {
                textBox1.Text = newText;
            }

        }

        private void update_Light(string newText)
        {
            if (textBox2.InvokeRequired)
```

```
            {
                updateLabelTextDelegate del = new
updateLabelTextDelegate(update_Light);
                textBox2.Invoke(del, new object[] { newText });
            }
            else
            {
                textBox2.Text = newText;
            }

        }

        private void update_Humidity(string newText)
        {
            if (textBox3.InvokeRequired)
            {
                updateLabelTextDelegate del = new
updateLabelTextDelegate(update_Humidity);
                textBox3.Invoke(del, new object[] { newText });
            }
            else
            {
                textBox3.Text = newText;
            }

        }

        private void update_temp2(string newText)
        {

            if (textBox20.InvokeRequired)
            {
                updateLabelTextDelegate del = new
updateLabelTextDelegate(update_temp2);
                textBox20.Invoke(del, new object[] { newText });
            }
            else
            {
                textBox20.Text = newText;
            }

        }

        private void update_Light2(string newText)
        {
            if (textBox21.InvokeRequired)
            {
                updateLabelTextDelegate del = new
updateLabelTextDelegate(update_Light2);
                textBox21.Invoke(del, new object[] { newText });
            }
            else
            {
                textBox21.Text = newText;
            }

        }
```

```
        private void update_Humidity2(string newText)
        {
            if (textBox19.InvokeRequired)
            {
                updateLabelTextDelegate del = new
updateLabelTextDelegate(update_Humidity2);
                textBox19.Invoke(del, new object[] { newText });
            }
            else
            {
                textBox19.Text = newText;
            }

        }


        private void button4_Click(object sender, EventArgs e)
        {
            if (serialPort1.IsOpen)
            {
                serialPort1.Write("2");//50
            }
            mode = 2;
            timer2.Enabled = true;
        }

        private void button3_Click(object sender, EventArgs e)
        {
            if (serialPort1.IsOpen)
            {
                serialPort1.Write("1");//49
            }
            mode = 1;
        }

        private void button2_Click(object sender, EventArgs e)
        {
            if (serialPort1.IsOpen)
            {
               // serialPort1.Write("0");//48
                serialPort1.Write("0");
            }
            mode = 0;
        }

        int flage = 0;
        private void timer1_Tick(object sender, EventArgs e)
        {
            textBox13.Text = DateTime.Now.Hour.ToString();
            textBox14.Text = DateTime.Now.Minute.ToString();
            textBox15.Text = DateTime.Now.Second.ToString();

            if (mode == 0)
            {
                if ((textBox7.Text == textBox13.Text) && (textBox9.Text ==
textBox14.Text) && (textBox8.Text == textBox15.Text))//ON
```

```
                {
                    flage = 4;
                    if (serialPort1.IsOpen)
                    {
                        serialPort1.Write("4");
                    }

                }
                if ((textBox13.Text == textBox10.Text) && (textBox14.Text ==
textBox12.Text) && (textBox15.Text == textBox11.Text))//OFF
                {
                    flage = 5;
                    if (serialPort1.IsOpen)
                    {
                        serialPort1.Write("5");
                    }
                }
            }
            if (mode == 71)
            {
                if ((textBox31.Text == textBox13.Text) && (textBox33.Text ==
textBox14.Text) && (textBox32.Text == textBox15.Text))//ON
                {
                    flage = 89;
                    if (serialPort1.IsOpen)
                    {
                        serialPort1.Write("Y");
                    }

                }
                if ((textBox13.Text == textBox28.Text) && (textBox14.Text ==
textBox30.Text) && (textBox15.Text == textBox29.Text))//OFF
                {
                    flage = 85;
                    if (serialPort1.IsOpen)
                    {
                        serialPort1.Write("U");
                    }
                }

            }
        }

        private void button5_Click(object sender, EventArgs e)
        {

        }

        private void timer2_Tick(object sender, EventArgs e)
        {
            try
            {
                if (Convert.ToInt32(textBox2.Text) <=
Convert.ToInt32(textBox22.Text))
                {
                    serialPort1.Write("6");
                    serialPort1.Write("A");//turn the artificial lights ON
```

```
                    textBox5.Text = "ON";

                }
                else
                {
                    serialPort1.Write("6");
                    serialPort1.Write("B");//Turn the artificial lights OFF
                    textBox5.Text = "OFF";
                }

                if (Convert.ToInt32(textBox1.Text) >=
Convert.ToInt32(textBox23.Text))
                {
                    serialPort1.Write("7");
                    serialPort1.Write("C");//turn the coller ON
                    textBox6.Text = "ON";
                }
                else
                {
                    serialPort1.Write("7");
                    serialPort1.Write("D");//turn the cooler OFF
                    textBox6.Text = "OFF";
                }
                if (Convert.ToInt32(textBox3.Text) <=
Convert.ToInt32(textBox24.Text))
                {
                    serialPort1.Write("8");
                    serialPort1.Write("E");//turn the irrigation ON
                    textBox4.Text = "ON";
                }
                else
                {
                    serialPort1.Write("8");
                    serialPort1.Write("F");//Turn the irrigation OFF
                    textBox4.Text = "OFF";
                }

                ////////////////////////////
                if (Convert.ToInt32(textBox21.Text) <=
Convert.ToInt32(textBox26.Text))
                {
                    serialPort1.Write("R");
                    serialPort1.Write("a");//turn the artificial lights ON
                    textBox17.Text = "ON";

                }
                else
                {
                    serialPort1.Write("R");
                    serialPort1.Write("b");//Turn the artificial lights OFF
                    textBox17.Text = "OFF";
                }

                if (Convert.ToInt32(textBox20.Text) >=
Convert.ToInt32(textBox27.Text))
                {
                    serialPort1.Write("V");
```

```
                    serialPort1.Write("c");//turn the coller ON
                    textBox16.Text = "ON";
                }
                else
                {
                    serialPort1.Write("V");
                    serialPort1.Write("d");//turn the cooler OFF
                    textBox16.Text = "OFF";
                }
                if (Convert.ToInt32(textBox19.Text) <=
Convert.ToInt32(textBox25.Text))
                {
                    serialPort1.Write("M");
                    serialPort1.Write("O");//turn the irrigation ON
                    textBox18.Text = "ON";
                }
                else
                {
                    serialPort1.Write("M");
                    serialPort1.Write("Q");//Turn the irrigation OFF
                    textBox18.Text = "OFF";
                }
            }
            catch (Exception t) { }
        }

        private void groupBox3_Enter(object sender, EventArgs e)
        {

        }

        private void groupBox7_Enter(object sender, EventArgs e)
        {

        }

        private void button5_Click_1(object sender, EventArgs e)
        {
            if (serialPort1.IsOpen)
            {
                // serialPort1.Write("0");//48
                serialPort1.Write("G");
            }
            mode = 71;
        }

        private void button7_Click(object sender, EventArgs e)
        {
            if (serialPort1.IsOpen)
            {
                serialPort1.Write("P");//49
            }
            mode = 80;
        }

        private void button6_Click(object sender, EventArgs e)
        {
```

```
            if (serialPort1.IsOpen)
            {
                serialPort1.Write("I");//50
            }
            mode = 73;
            timer2.Enabled = true;
        }

        bool switch_ = false;
        private void timer3_Tick(object sender, EventArgs e)
        {
            if (serialPort1.IsOpen == true)
            {
                switch_ = !switch_;
                if (switch_ == true)
                {
                    serialPort1.Write("s");
                }
                if (switch_ == false)
                {
                    serialPort1.Write("r");
                }
            }

        }
    }
}
```