Lund University
Faculty of Engineering, LTH

3D Temperature Distribution of an Orchid Greenhouse

by

Rayaan CHOUDRI

Yew Meng KHAW

Supervisors:
Anders ARDO
Bertil LINDVALL

2015

Table of Contents

EXECUTIVE SUMMARY

This report provides a detailed summary of the process and results of the temperature measurement system of the orchid greenhouse. A 3D temperature distribution is needed for revamping the temperature regulating system of the greenhouse. Major electronic components used can be found under *Technical Description*. The installation of hardware, programs used for data collections, and the method used for interpolation and extrapolation of data can be found under *Implementation*. The 3D temperature distribution of the greenhouse can be found under *Result.* The temperature is observed to clearly fall outside the desired temperature range of 24-30 °C. The range of temperature measured lies between 16°C and 32°C, indicating a clear need for a better heating system of the greenhouse. A video showing the temperature distribution with time of the greenhouse is attached with this report. The report however did not investigate the external and internal conditions of the greenhouse which have clear implications on the internal temperature distribution of the greenhouse.

TECHNICAL DESCRIPTION

Raspberry Pi (RPi)

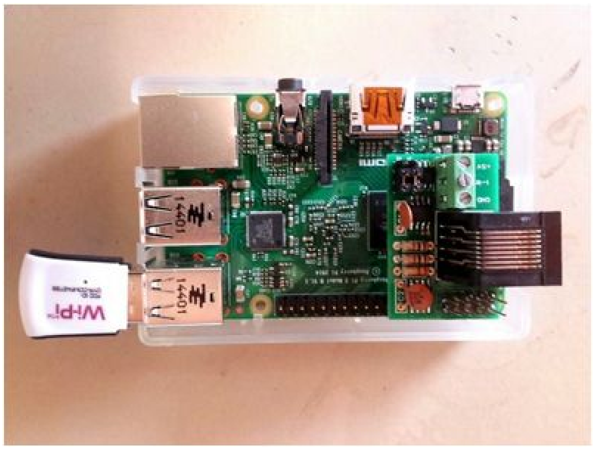

Figure 1: Raspberry Pi attached to the RPI2 1-Wire Host Adapter and a USB WIFI dongle

The raspberry pi serves as the central processing unit of the measurement system. It communicates with the sensors and pushes the sensor readings onto a server. A USB WIFI dongle is used with the Raspberry Pi to allow wireless connection with the server from an external machine.

DS18B20 Sensor

Figure 2: DS18B20 Sensor[1]

The DS18B20 sensor measures temperature. It uses the 1-wire device communications bus system. Each DS18B20 sensor has a 64bit unique serial code[2] allowing one Raspberry Pi to control multiple DS18B20 sensors.

[1] "DS18B20 - Digital Temperature Sensor - Rs.145.00 : Ventor ..." 2013. 24 Jun. 2015 <http://www.ventor.co.in/index.php?main_page=product_info&products_id=258>
[2] "DS18B20 Programmable Resolution 1-Wire Digital ... - Maxim." 2012. 24 Jun. 2015 <http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf>

RPI2 1-Wire Host Adapter

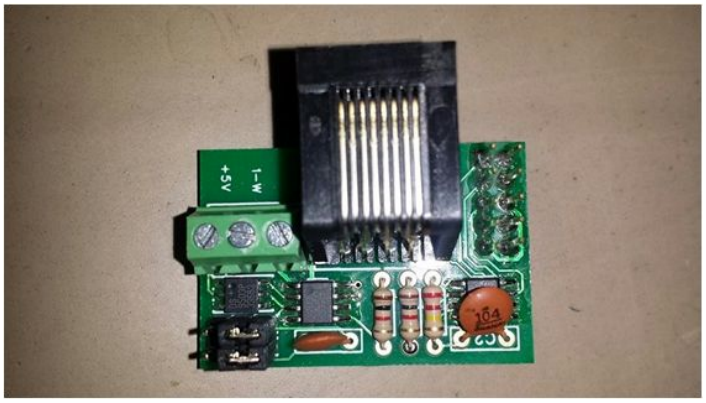

Figure 3: RPI2 1-Wire Host Adapter

The RPI2 1-Wire Host Adapter allows the Raspberry Pi to communicate with 1-Wire devices such as the DS18B20 sensors. Connection with the 1-Wire device network is achieved via the screw terminals.

IMPLEMENTATION

Overall Architecture of the Measurement System

A linux machine hosts a server. The Raspberry Pi pushes data to the server via Hypertext Transfer Protocol (HTTP). Temperature is measured every 5 minutes and the sensor readings are stored locally in the RPi and also the linux machine. Data can be accessed online in realtime with a web browser. The firewall settings are set such that any machine could access the server. This data can be collected from any machine with a web browser and wifi capability. A brief outline is shown in Figure 4.

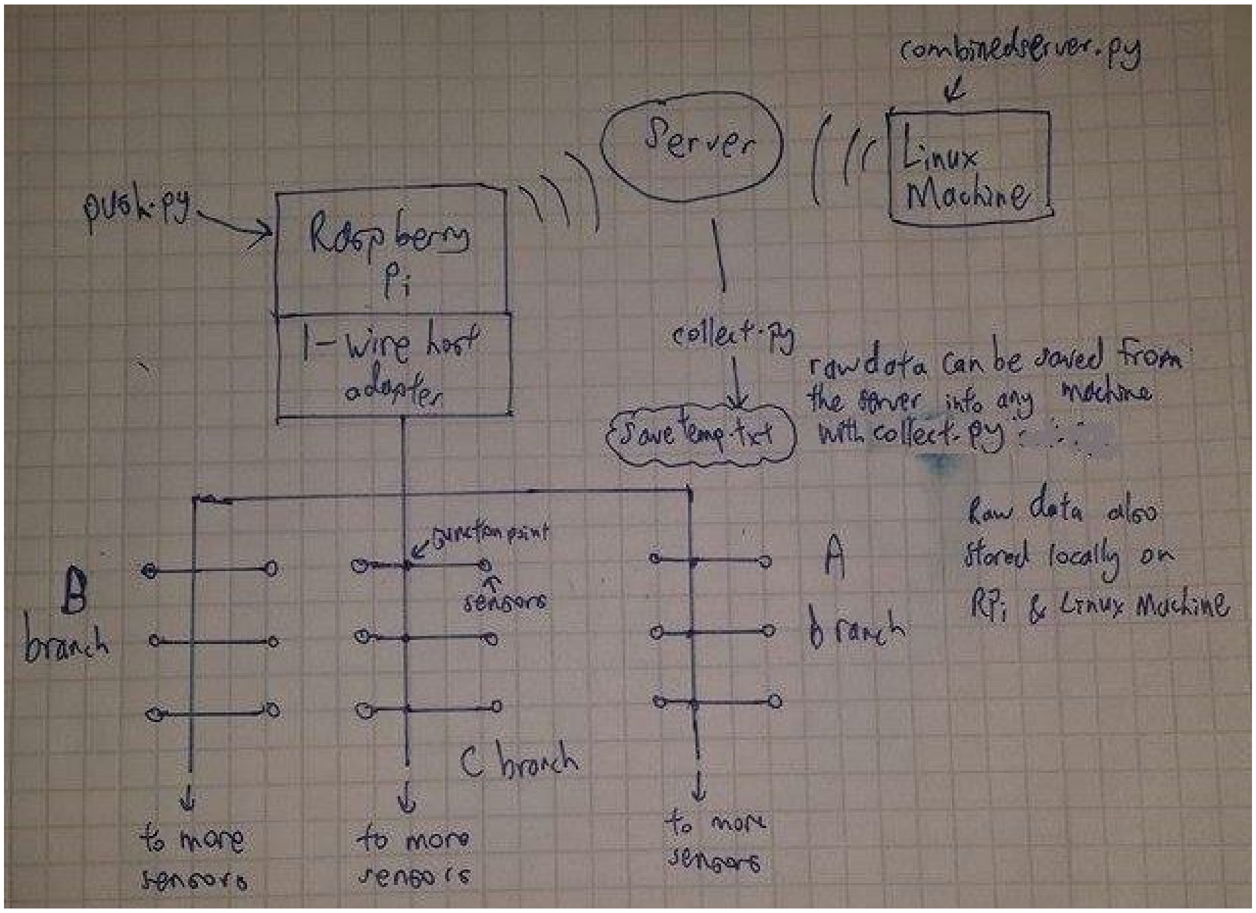

Figure 4: System Architecture

Placement of Sensors

30 DS18B20 sensors are used. One RPI and one RPI2 1-Wire host adapter are sufficient to control all 30 sensors. The sensors are split into 3 branches: branch A, branch B and brach C as shown in Figure 5. Branch A goes through the top and right section of the greenhouse and has 9 sensors. Branch B goes through the bottom and left section of the greenhouse and has 15 sensors. Branch C goes through the center of the greenhouse and has 6 sensors. The ID number and the corresponding serial number can be found in push.py under *Appendix*.

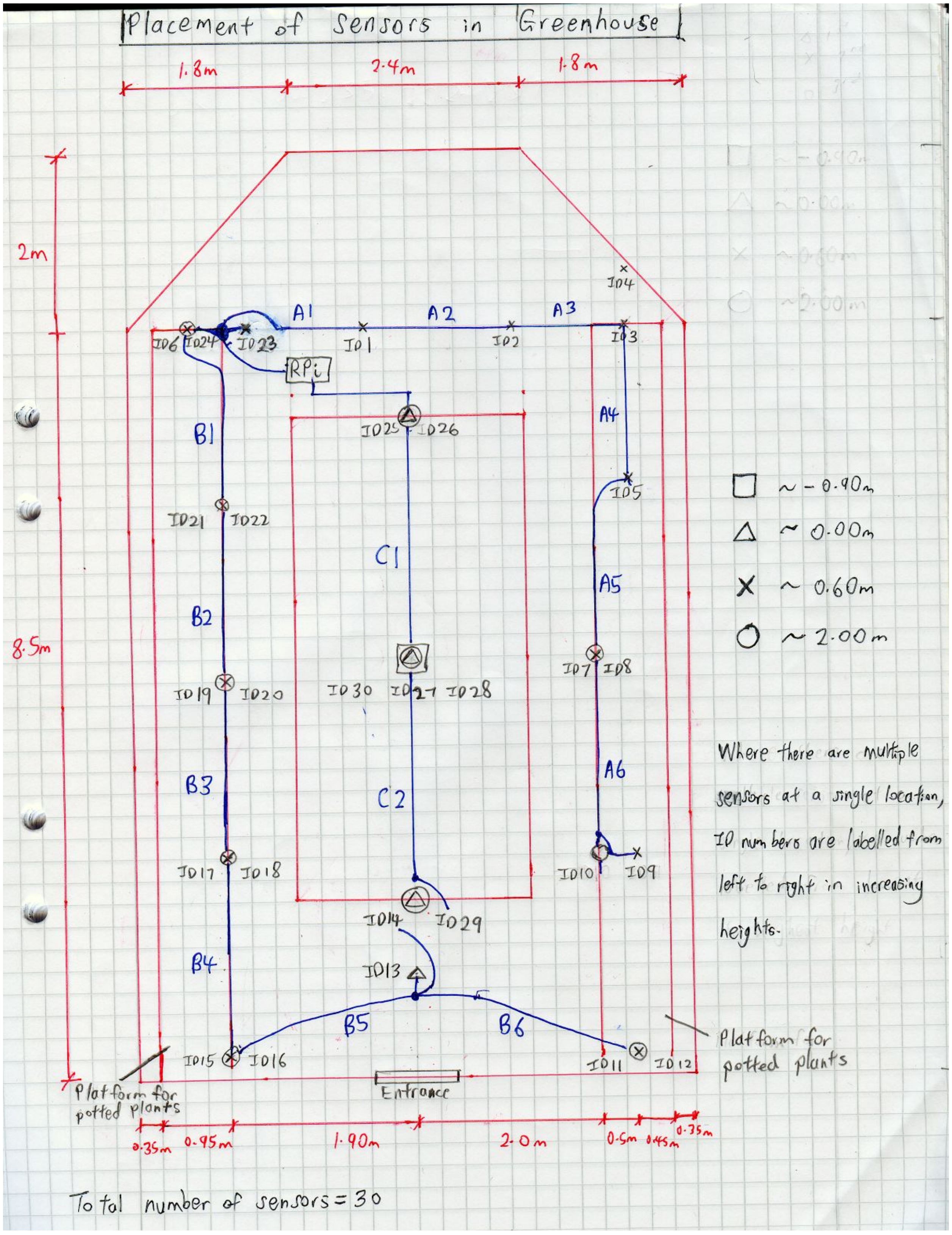

Figure 5: Placement of Sensors in the Greenhouse

Shielding of Electronic Components

Raspberry Pi
A water resistant housing shields the RPi from water exposure such as the automatic sprinkler system. 3 holes are drilled into the sides of the housing to allow entry of cables.

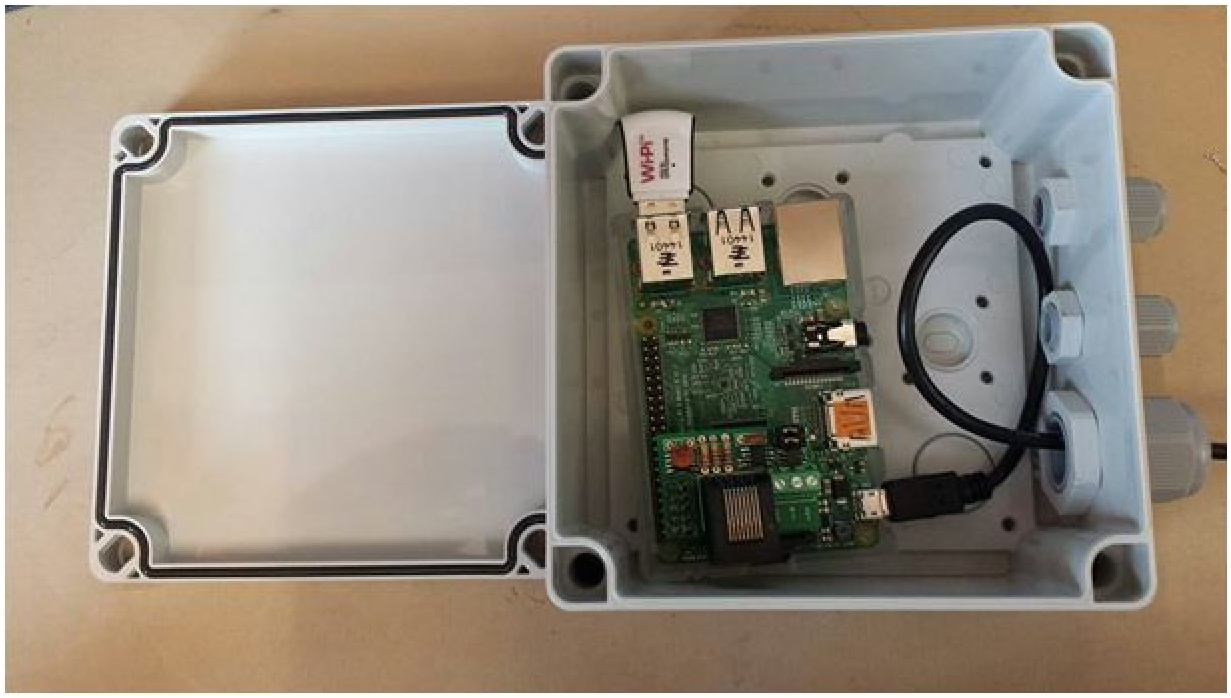

Figure 6: Water resistant housing of the Raspberry Pi

Sensors
Paper cups wrapped in aluminum foil shield the sensors from water exposure. The aluminum foil shields the sensor from radiation that heats up the sensor, giving false readings. Holes at the top of the paper cup enhances ventilation. Trapped hot air in the paper cup is allowed to escape. The cable is taped into a U shape with the sensor facing upwards as shown in FIgure 7. This prevents water flowing down the cable from contacting the sensor. Water on the sensor may also lead to false reading as evaporation of water from the sensor's surface leads to cooling.

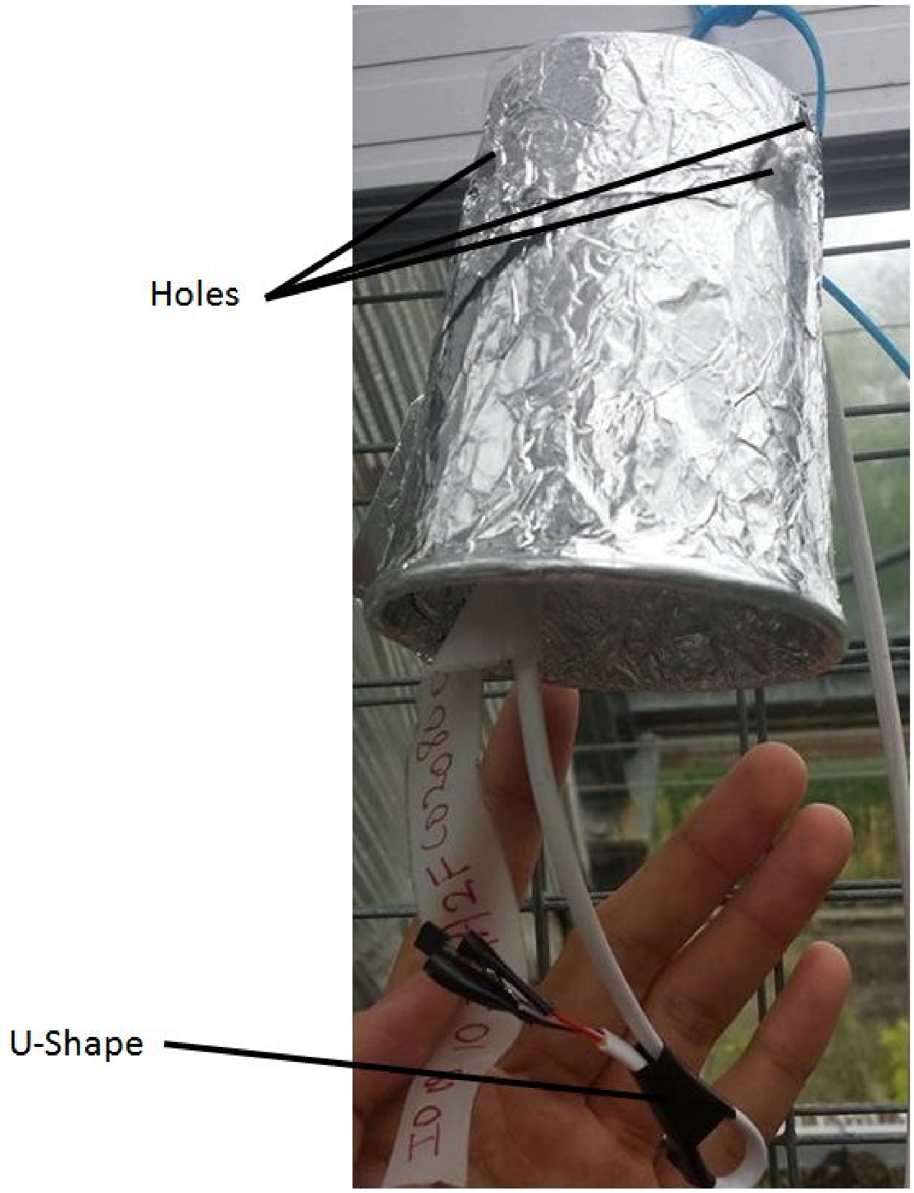

Figure 7: Shielding of the sensor showing the holes and U-shape

Shielding of Junction Points

The junction points are sealed with wiring tape to prevent exposure to water. The wires are also secured as shown in Figure 8 to reduce tension on wires at the soldering point.

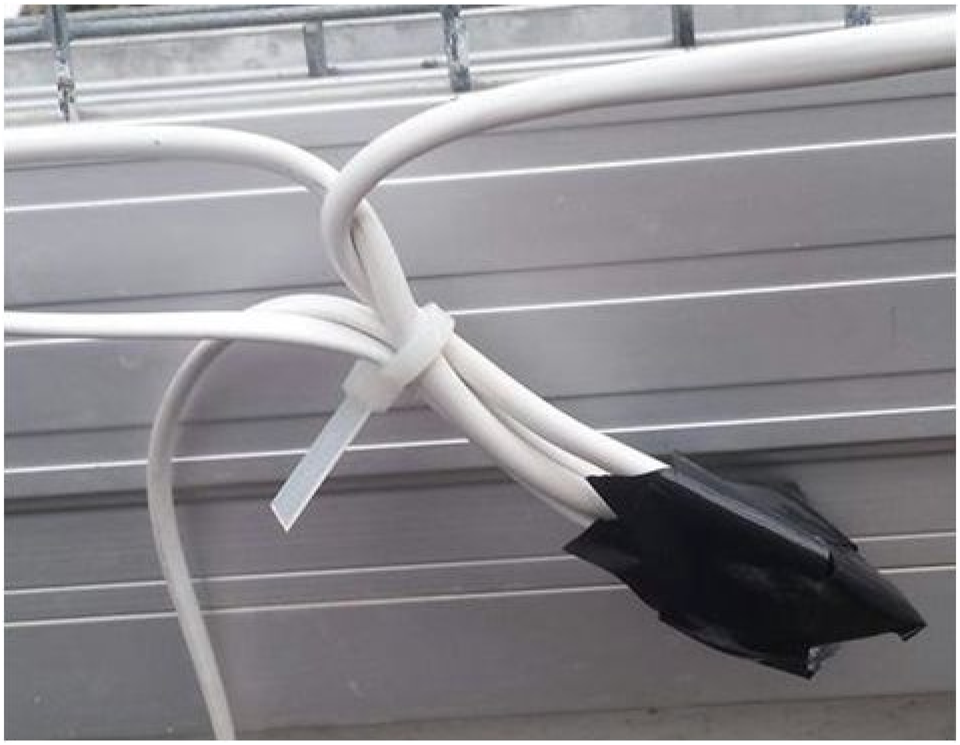

Figure 8: Shielding of the junction point

Measuring the Water Temperature
The DS18B20 was tested for robustness underwater. Prior to implementing the measurement system, sensor ID30 was submerged in water for 72 hours and has been observed to function properly. ID30 was utilized in the measurement system to measure the water temperature at the center of the greenhouse. After an additional 72 hours being submerged in water, ID30 maintains proper function. No casing or waterproofing was done on ID30.

Programs for Data Collection
All programs are written in Python. An overview of the program can also be seen in Figure 4.

combinedserver.py
An external linux machine (other than the RPi) runs combinedserver.py. Webpy allows the user to create a website with Python. The code can be found under *Appendix*.

push.py
The RPi pushes the sensor reading onto the server with push.py. The parameters of the URL are updated with the most current temperature readings. The RPi opens the URL, calling the GET function which takes the values of the parameters and write it into savetemp.txt which can be seen in combinedserver.py. The code can be found under *Appendix*.

collect.py

The data in the server can be collected from any machine with a web browser and with wifi capability. The code can be found under *Appendix*.

Visualization

For the visualization of the data we collected, we decided to use Matlab as it was the simplest software available to us. The visualization needed to account for 5 dimensions: space(3D), temperature, and time.

To visualize the dimensions of space and temperature, the data needed to be interpolated and extrapolated in order to visualize the entire volume of the room. This accounted for the random sensor locations in the room. We utilized Matlab's *scatteredInterpolant*() function that takes scattered data and performs a *natural neighbour interpolation* and *linear* extrapolation to do this.

For information about the *natural neighbor interpolation* see:
http://dilbert.engr.ucdavis.edu/~suku/nem/nem_intro/node3.html

For *linear extrapolation* Matlab uses the following algorithm to approximate data in blind spots:

$$y(x_*) = y_{k-1} + \frac{x_* - x_{k-1}}{x_k - x_{k-1}}(y_k - y_{k-1}).$$

After generating the 3d temperature plot, we took 2D slices of the plot at heights of 60, 120, and 180 centimeters since our sensors were congregated most around these areas. This made it easier to analyze the distribution of temperature in the room. The differences in temperature are represented by color ranging from a 16.5 - 32°C scale. These values were chosen based on the lowest and highest temperatures found in the data.

To account for the 5th dimension of time, we took the 5 minute interval temperature readings of the hour and computed the hourly average. Using this technique we arrived at 72 hours worth of visuals with each hour showing the average at the 3 heights.

DataCollect.m can be found under *Appendix*.

Results

For the spacial distribution, we noticed that the temperatures spiked along the edges of the room. We could attribute this partially due to the radiators along the edges and to the sunlight hitting those areas more. We found that the centre of the room was the coolest part. This could be in part due to the fan located in the centre. The maximum difference between two points at any given time was approximately 4°C. However, this difference was not so often the case. In fact, we found that the variation in temperature of the spatial distribution was not too significant. Figures 9-11 below show the spatial temperature distribution at a given time at 60cm,120cm and 180 respectively.

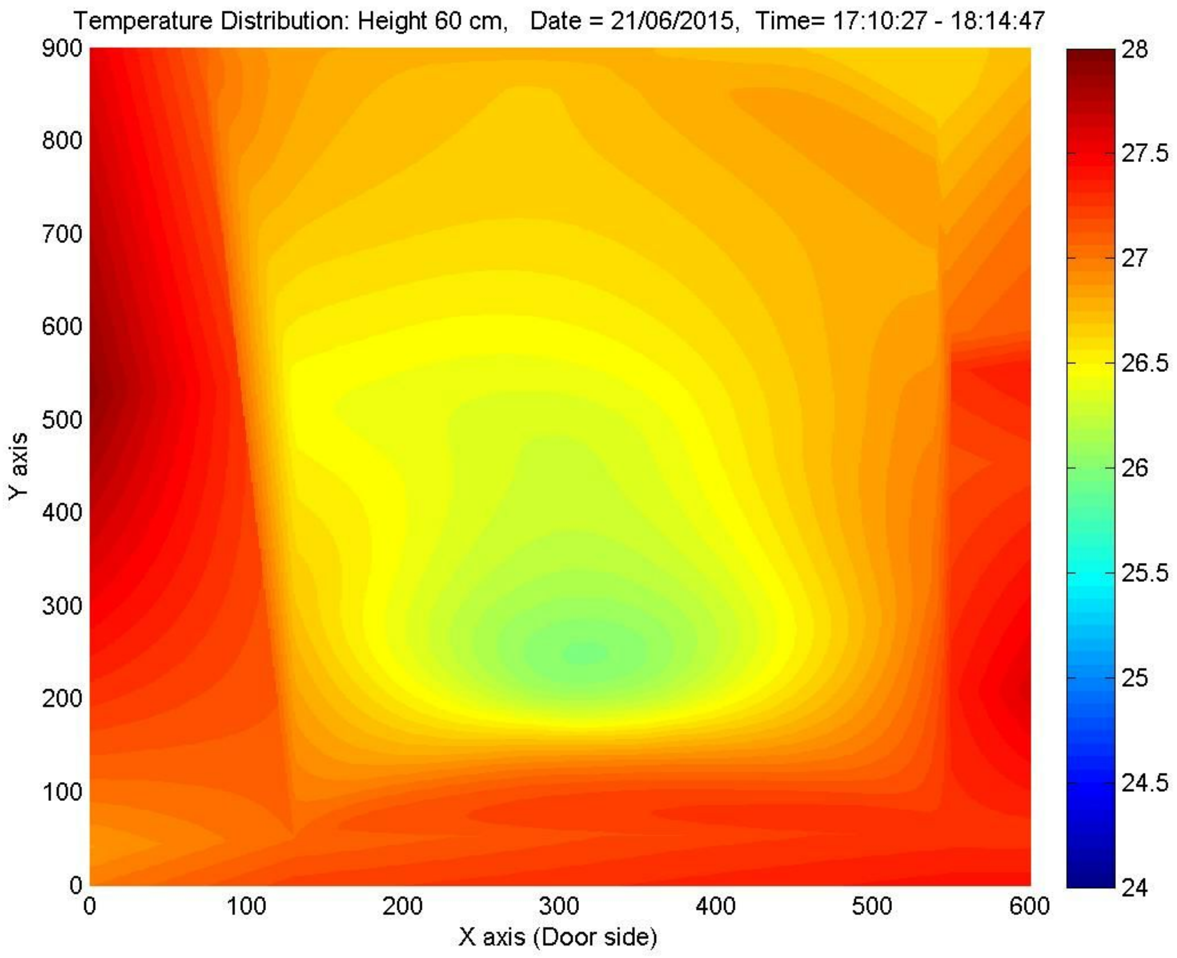

Figure 9 - Spatial Dist at Height = 60cm

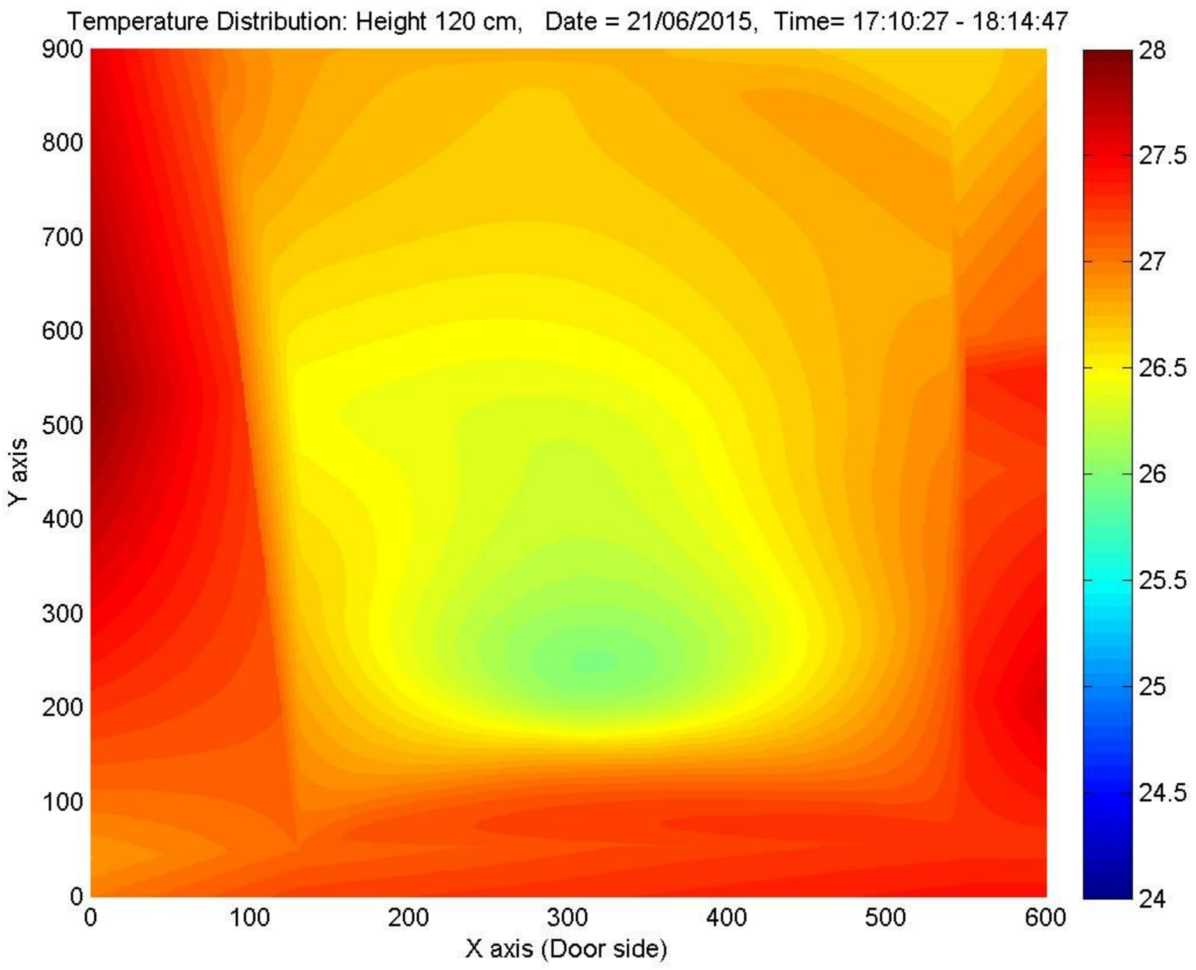

Figure 10 - Spatial Dist at Height = 120cm

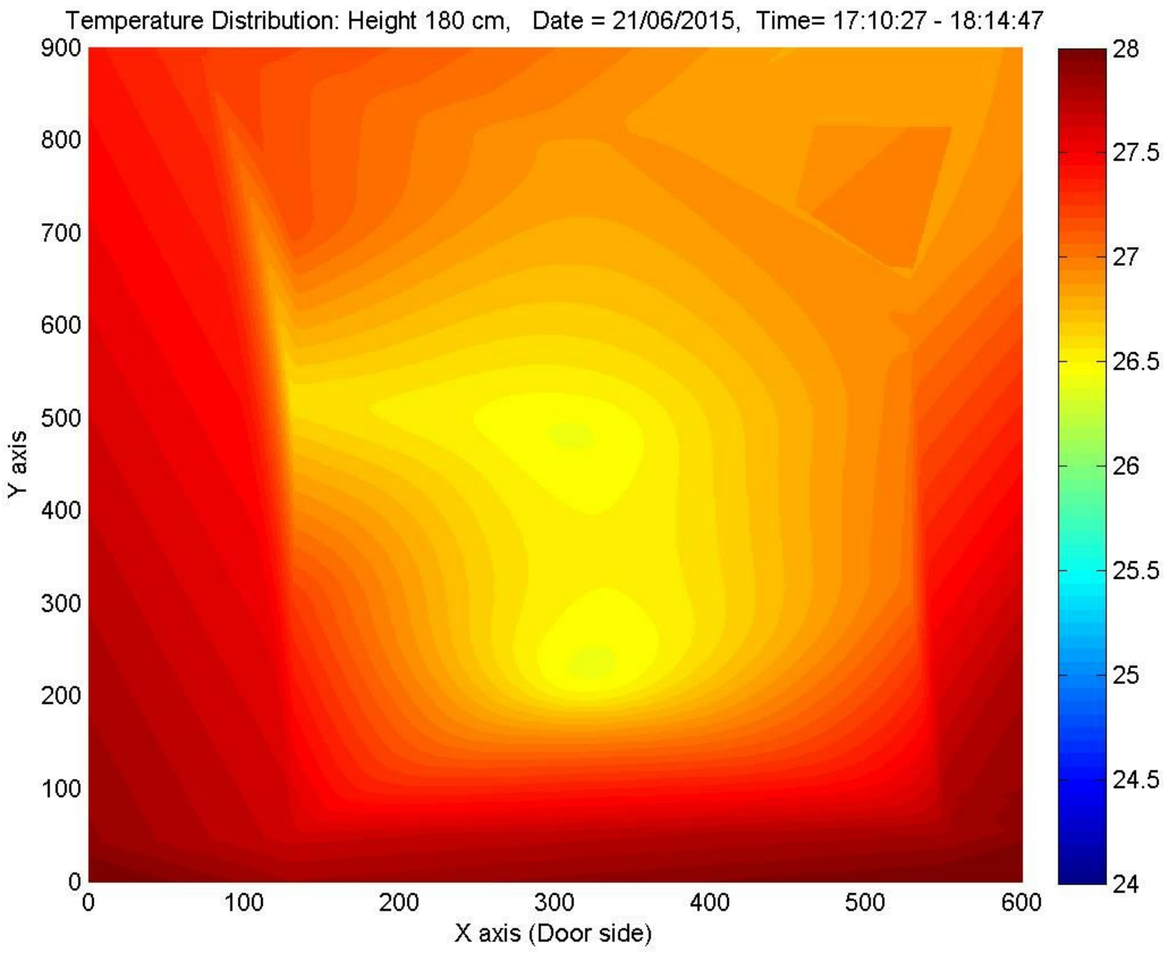

Figure 11 - Spatial Dist at Height = 180cm

The largest spatial variation of temperature is approximately 3ºC. For temperature variation with time, we noticed a trend of high temperatures during the day and low temperatures during the night. We found that the temperatures peaked between 12pm-2pm and dipped the most between 12am-4am. The highest temperature was between 31-32ºC while the lowest temperature was between 16-17ºC. Temperature variation with time is observed to be more significant than variation with space. Figures 12 and 13 show the distribution at height = 120cm with a 12 hour difference between both figures.

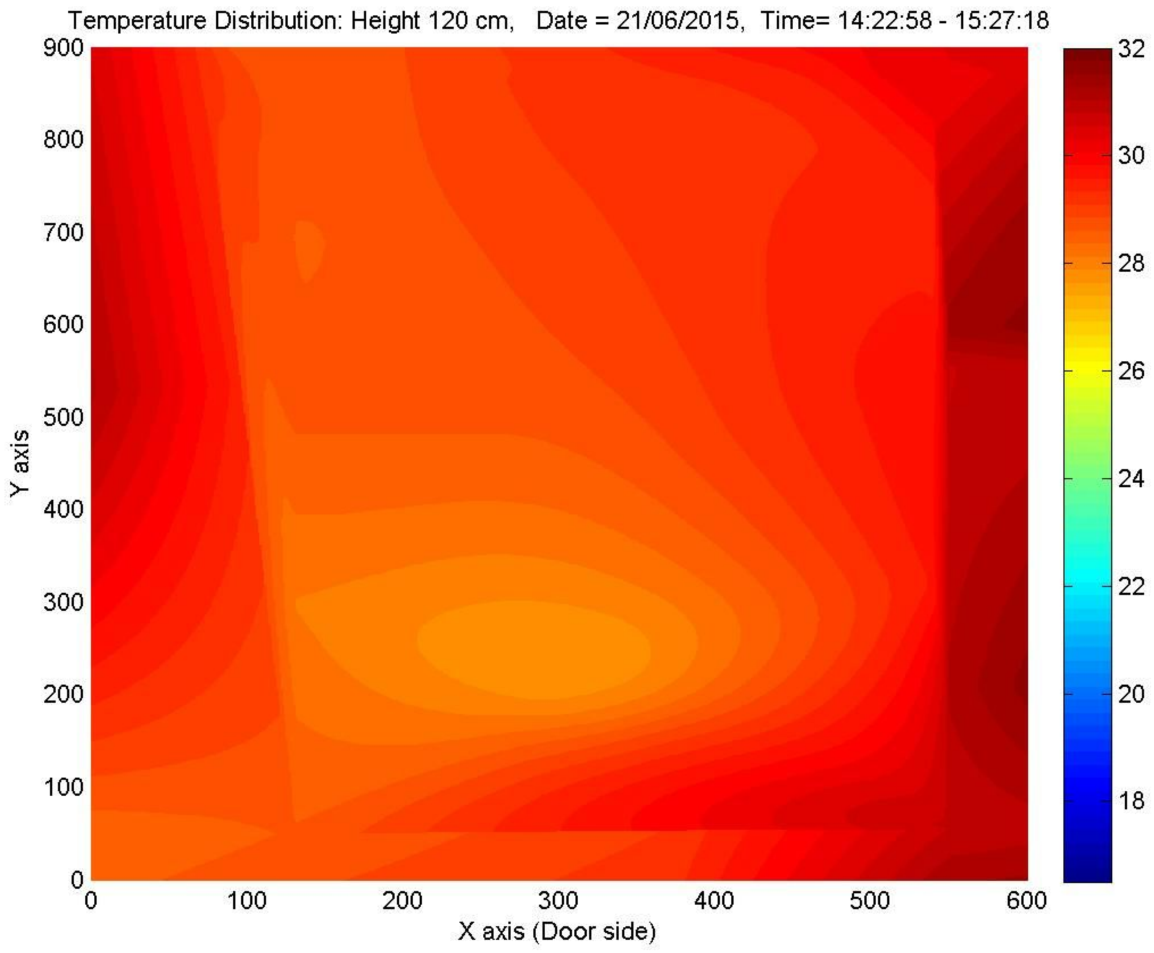

Figure 12 - Temp Dist at Time 14:22 - 15:27

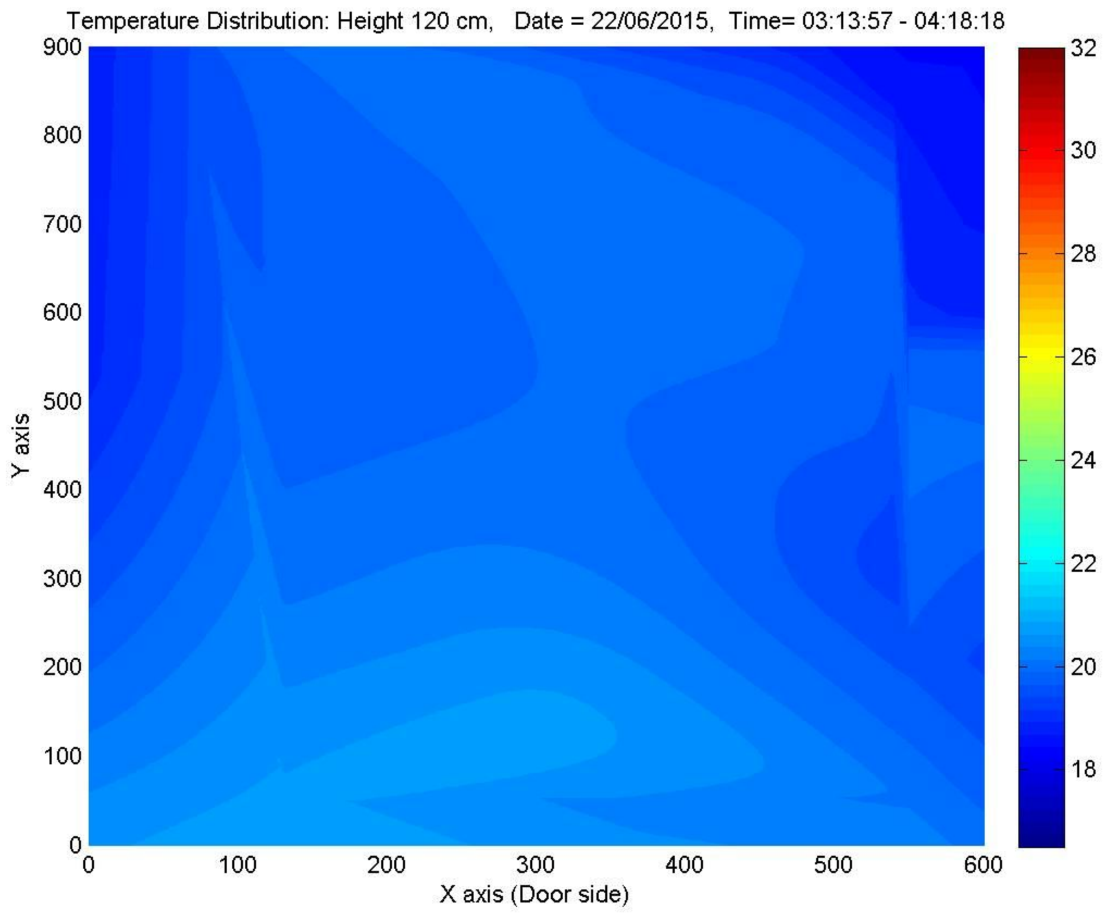

Figure 13 - Temp Dist at Time 03:13 - 04:18

The difference in temperature can be as great as 15^{o}C.

ID30 which was submerged in water measure an insignificant variation with time as shown in Figure 14.

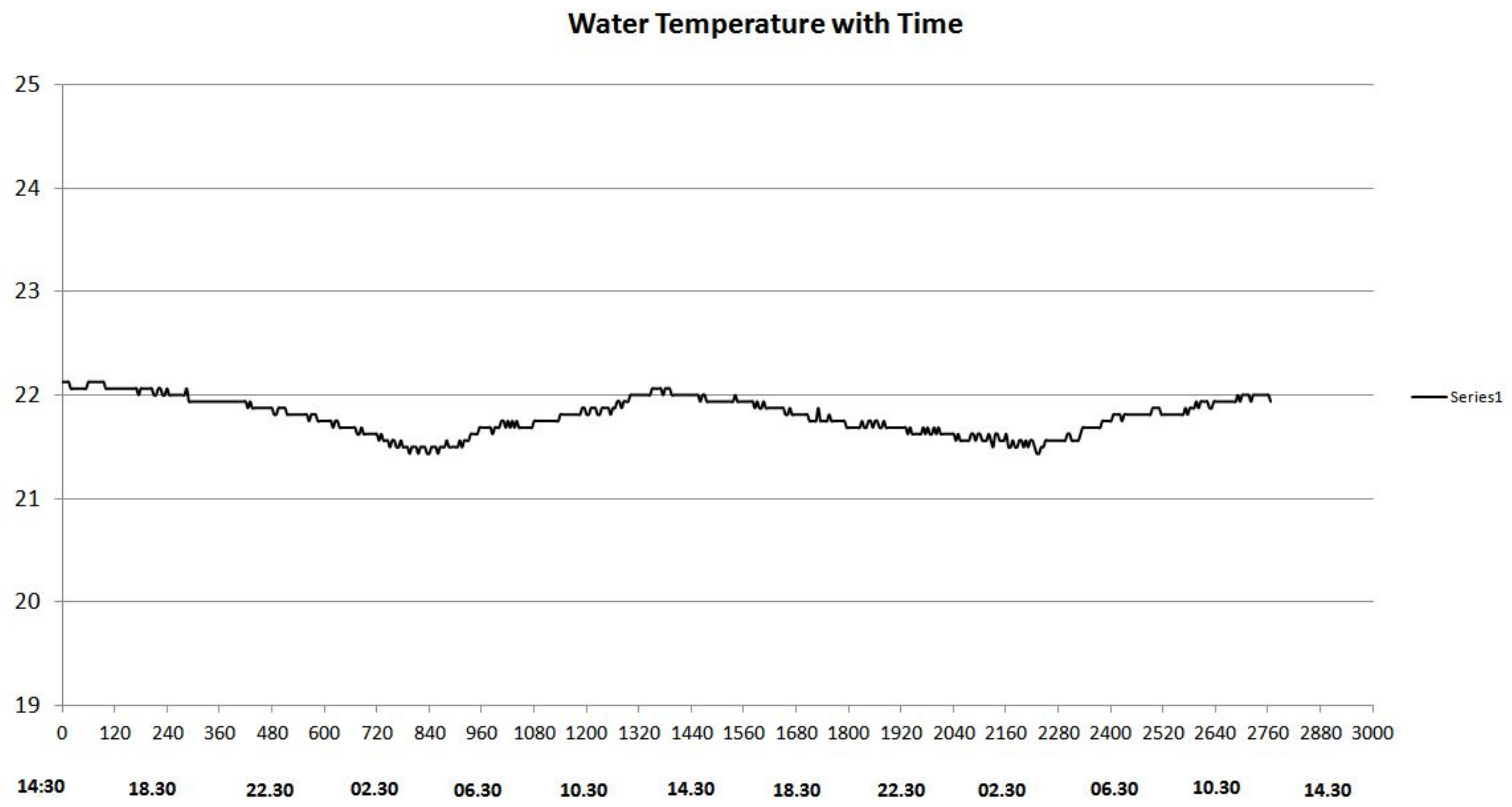

Figure 14 - Water temperature Graph over 24 hours

Future Work and Improvements

Additional 1-wire sensors such as the DS18B20 can be installed to test the limit on the number of sensors and length of wire of the RPI2 1-wire host adapter. In future measurement where ore sensors are used, the sensors can be better organized with the RPI3 1-Wire host adapter[3]. The RPI3 allows one raspberry pi to utilize multiple 1-Wire busses, allowing the RPi to potentially control a lot more 1-Wire devices. Partitioning the sensors into multiple busses allows better detection of problems in the hardware including broken junction points and malfunctioning sensors. The distribution of both the humidity and temperature should also be measured simultaneously. The SHT21 is able to measure both temperature and humidity[4]. External conditions outside the greenhouse should also be measured along with internal conditions such

[3] 2014. RPI3 v1 1-Wire Host Adapter for Raspberry Pi Models A and B. http://www.sheepwalkelectronics.co.uk/product_info.php?products_id=33.
[4] 2014. Datasheet SHT21 - Sensirion. http://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/Humidity/Sensirion_Humidity_SHT21_Datasheet_V4.pdf.

as position of curtains throughout the day. The radiation shield also needs to be further tested. There is a possibility that certain sensors particularly ID10 and ID21 are heating up, giving false observation that the greenhouse is hotter on the top left section and right bottom section at the height of 60cm as shown in Figure 15. The light intensity inside the shield can be measured to test the effectiveness of the radiation shield. The DS18B20 sensor can also be used as a parasitic load. Only the ground pin and data pin could be used, eliminating the need for an external power source.

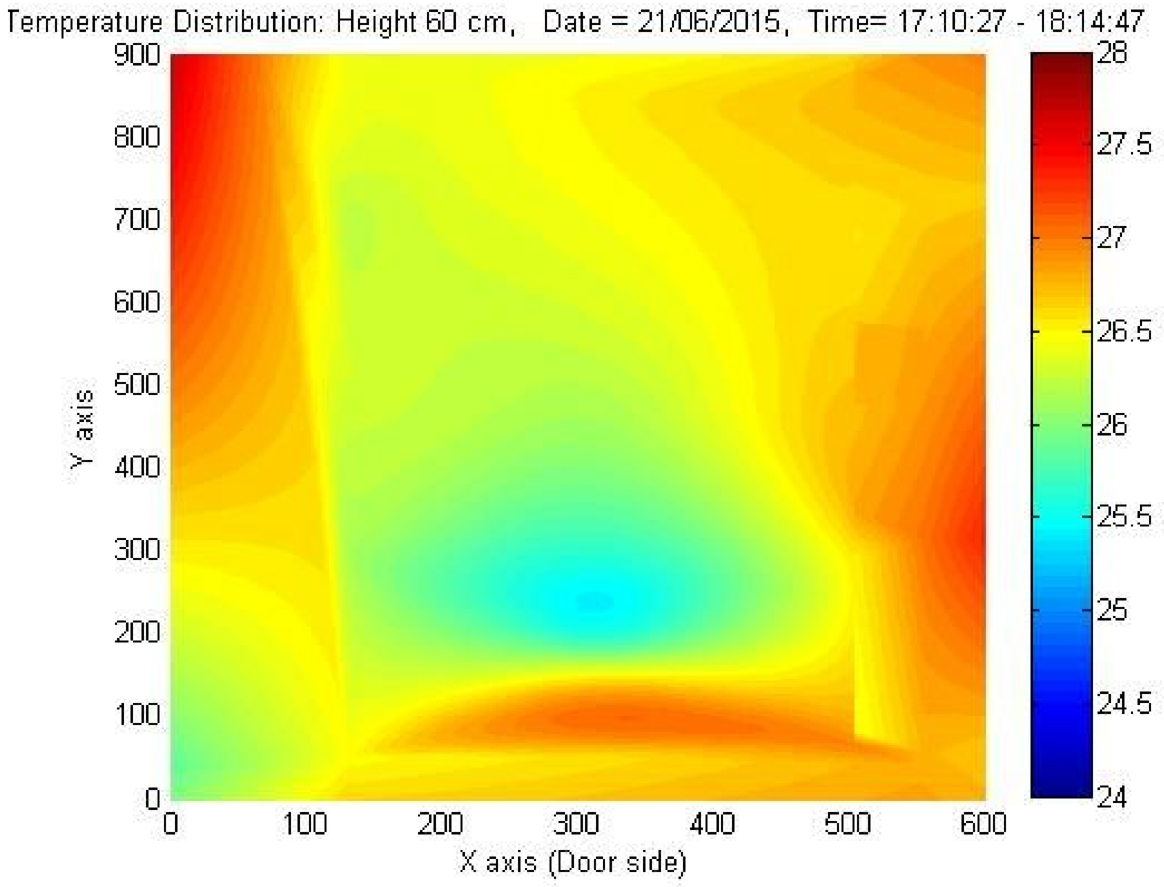

Figure 15: Temperature distribution at height = 60cm.

Appendix

push.py

```
# count: no of time to push data
# count1: used in while loop to push data to the server by giving values to parameters in the
URL
# count2: used in while loop to obtain values of all sensors
# count3: used in while loop to create the URL link with the updated sensor values
# count4: used in while loop to save values of sensor locally in RPi
# id: list that stores id of all sensors
# temp: list that stores temperature of all sensors

# start OWFS
from subprocess import call
call("sudo owfs -C -uall -m/mnt/1wire --allow_other", shell=True)
import urllib2
import time
no_of_sensor = 30
total_duration = 4320  # in minutes
interval = 5    # in minutes
count = total_duration/interval
wait_time  = interval*60
baseURL = 'http://dbkit13.eit.lth.se:8085/temp?'
#baseURL = 'http://130.235.201.156:1234/temp?'
id = [0]
id.append('10.9974F3020800')    #id1
id.append('10.3ACCF3020800')    #id2
id.append('10.39A3FC020800')    #id3
id.append('10.C23DF4020800')    #id4
id.append('10.69F0FC020800')    #id5
id.append('10.7CFEFC020800')    #id6
id.append('10.E4A2FC020800')    #id7
id.append('10.A3FBFC020800')    #id8
id.append('10.10C9FC020800')    #id9
id.append('10.10C9FC020800')    #id10
id.append('10.EF62F3020800')    #id11
id.append('10.41A0F3020800')    #id12
id.append('10.0267F3020800')    #id13
id.append('10.99DFF3020800')    #id14
id.append('10.FEC6FC020800')    #id15
id.append('10.26C7F3020800')    #id16
id.append('10.4776F3020800')    #id17
id.append('10.7AA2FC020800')    #id18
id.append('10.AE85F3020800')    #id19
id.append('10.BBC3F3020800')    #id20
id.append('10.2BF8F3020800')    #id21
id.append('10.F5DBF3020800')    #id22
id.append('10.7FFCF3020800')    #id23
id.append('10.7AA2FC020800')    #id24
id.append('10.7597FC020800')    #id25
id.append('10.78EDFC020800')    #id26
id.append('10.D1B6FC020800')    #id27
id.append('10.89DDFC020800')    #id28
id.append('10.13EDFC020800')    #id29
id.append('10.ACD7FC020800')    #id30 submerged in water

count1 = 0

# push data
while (count1 <= count):
        count2 = 1
        temp = [0]

        # obtain values of sensors
        while (count2<=no_of_sensor):
                id_no = 'id'+str(count2)
                try:
                        file = '/mnt/1wire/' + id[count2] + '/temperature'
                        file = open(file,'r')
                except IOError:
                        print ('%s is not working' %id_no)
                        temp.append(0)
                else:
                        temp.append( float(file.read()) )
                        file.close()
                count2 = count2 + 1

        # creating the URL and updating values of its parameter
        URL = baseURL + 'temp1=' + str(temp[1]) + '&id1=' + id[1]
        count3 = 2
        while (count3<=no_of_sensor):
                temp_no = 'temp'+ str(count3)
                id_no = 'id' + str(count3)
                URL =  URL + '&' + temp_no + '=' + str(temp[count3]) + '&' + id_no + '=' +  id[count3]
                count3=count3+1
        print(URL)
        # checks if RPi can establish connection with server.If fail, print 'unable to connect to
server' and move on
        try:
                f = urllib2.urlopen(URL)
        except urllib2.URLError:
                print 'unable to connect to server'
        else:
                print 'connected to server'

        # save data on RPi
        save = open('/home/pi/webpy/savetemp.txt','a')
        save.write(time.strftime('%d/%m/%Y') + '\t' + time.strftime('%H:%M:%S') + '\t')
        count4 = 1
        while (count4 <= no_of_sensor):
                save.write(id[count4] + '\t' + str(temp[count4]) + '\t')
                count4 = count4 + 1
        save.write('\n')
        save.close()
        time.sleep(wait_time)
        count1 = count1 + 1
```

combinedserver.py

```
import web
import time
no_of_sensors = 30
urls = (
        '/', 'index',
        '/result', 'result',
        '/temp?','temperature'
)

class index:
        def GET(self):
                return "What's Good"

class temperature:
        def GET(self):
                page = web.input()
                save = open ('/home/temp3d/savetemp.txt','a')
                save.write(time.strftime("%d/%m/%Y") + '\t' + time.strftime("%H:%M:%S") + '\t')
                count = 1
                while (count <= no_of_sensors):
                    temp = 'temp' + str(count)
                    id = 'id' + str(count)
                    save.write(page[id] + '\t' + page[temp] + '\t')
                    count = count + 1
                save.write('\n')
                save.close()
                return 'asd'

class result:
        def GET(self):
                file = open('/home/temp3d/savetemp.txt','r')
                data = file.read()
                file.close()
                return data

if __name__ == "__main__":
        app = web.application(urls, globals())
        app.run()
```

collect.py

```
import time
import urllib2
link = "http://dbkit13.eit.lth.se:8085/result"
count = 0
f=urllib2.urlopen(link)
temp = f.read()
save = open ('//loke.eit.lth.se/UW/int15ykh/Desktop/Temp/savetemp.txt','w')
save.write(str(temp))
save.close()
```

DataCollect.m

```
%Temperature Data File Parse
delim='\t';
A=tdfread('savetemp (4).txt','\t');


%Sensor coordinates
DataCopy =importdata('data.txt', '\t');
C=tdfread('coordinates.txt','\t');
%Makes Array From vectors
```

```
T=cat(2,A.Temp1,A.Temp2,A.Temp3,A.Temp4,A.Temp5,A.Temp6,A.Temp7,A.Temp8,A.Temp9
,A.Temp10,A.Temp11,A.Temp12,A.Temp13,A.Temp14,A.Temp15,A.Temp16,A.Temp17,A.Tem
p18,A.Temp19,A.Temp20,A.Temp21,A.Temp22,A.Temp23,A.Temp24,A.Temp25,A.Temp26,A.
Temp27,A.Temp28,A.Temp29);
Tavg=zeros(29,1);

%Loops through all 5 minute intervals
for i = 0:67
    for n = 1:29
        sum = 0;
        k=12*i+1;
        j=k+12;
        for m =k:j %(10i):(10i+10) %%%WRONGGGG
            sum=sum +T(m,n);
        end
        %Computes the Average
        Tavg(n)=sum/12;
    end

    %Setting coordinates
    X=C.X_Y_Z(:,1);
    Y=C.X_Y_Z(:,2);
    Z=C.X_Y_Z(:,3);

    %Interpolating coordinates and values
    interpolation_fnc = scatteredInterpolant(X,Y,Z,Tavg,'natural','linear');
    [xn,yn,zn] = meshgrid(0:1:600,0:1:900,60);
    [x1,y1,z1] = meshgrid(0:1:600,0:1:900,120);
    [x2,y2,z2] = meshgrid(0:1:600,0:1:900,180);

    temp_level_0= interpolation_fnc(xn,yn,zn);
    temp_level_1= interpolation_fnc(x1,y1,z1);
    temp_level_2= interpolation_fnc(x2,y2,z2);


    %plotting figures
    MinTemp=16.5; %setting scale for plot
    MaxTemp=32;

    h=figure; %first depth
    set(gcf, 'renderer', 'zbuffer');
    mesh(xn,yn,zn,temp_level_0);
    hold on;
    view(0,90);
    %setting variables for graph title
    d=char(DataCopy.textdata(k,1));
    e=char(DataCopy.textdata(k,2));
    f=char(DataCopy.textdata(j,2));
    title(['Temperature Distribution: Height 60 cm,   Date = ' d, ',  Time= ' e,' - ' f])
    xlabel('X axis (Door side)') % x-axis label
    ylabel('Y axis') % y-axis label
    colormap(jet)
    caxis manual
    caxis([MinTemp MaxTemp]);% set color scale manually, requires knowledge of min and max
temp of data set
    colorbar;
    hold off;
    w=num2str(i);
    h1=strcat(w,'-1');
    saveas(h, h1, 'jpg'); % saves the file as jpeg

    % Repeat process for the separate heights

    l=figure(2); % second depth
    set(gcf, 'renderer', 'zbuffer');
    mesh(x1,y1,z1,temp_level_1);
    hold on;
    view(0,90);
    title(['Temperature Distribution: Height 120 cm,   Date = ' d, ',  Time= ' e,' - ' f])
    xlabel('X axis (Door side)') % x-axis label
    ylabel('Y axis') % y-axis label
    colormap(jet)
    caxis manual
    caxis([MinTemp MaxTemp]);
    colorbar;
    hold off;
    h2=strcat(w,'-2');
    saveas(l, h2, 'jpg');

    o=figure(3); %third depth
    set(gcf, 'renderer', 'zbuffer');
    mesh(x2,y2,z2,temp_level_2);
    hold on;
    view(0,90);
    title(['Temperature Distribution: Height 180 cm,   Date = ' d, ',  Time= ' e,' - ' f])
    xlabel('X axis (Door side)') % x-axis label
    ylabel('Y axis') % y-axis label
    colormap(jet)
    caxis manual
    caxis([MinTemp MaxTemp]);
    colorbar
    hold off;
    h3=strcat(w,'-3');
    saveas(o, h3, 'jpg');

end
```