



INTERNET OF THINGS: PROJECT: 2 INTELLIGENT ELECTRICITY CONSUMPTION MONITORING SYSTEM

Abstract

This project utilizes data on current weather forecast and energy consumption within a particular area to predict when to turn your thermostat and other devices on/off. Along with the Raspberry Pi it uses a temperature sensor as a peripheral.

Janki Shah 131017
Prem Shah 131036
Rishabh Shah 131040

Index

1. Definition
2. Block Diagram
3. List of Components
 - Table: 1.1 List of components and needed Quantity
4. Selection Criteria of Sensors and Actuators
 - Table 1.2 DTH11 Temperature and Humidity Sensor
5. Working Principle of Sensors and Actuators
 - How the project Works
 - DTH11 Temperature and Humidity Sensor
6. Flow Chart
 - Logical Flow the “Intelligent Electricity Consumption Monitoring System”
 - Derivation of Equations For The Gradient Descent Algorithm
7. Schematic
8. Code
9. Screenshots
10. Difficulties
11. References

Figures:

1. Figure 1: Block diagram of Intelligent Electricity Consumption Monitoring System
2. Figure 2a: Integrated DHT11 Temperature and Humidity Sensor
3. Figure 2b: Generic DHT11 Sensor
4. Figure 3: Internal circuit of Integrated DHT11
5. Figure 4: Components integrated in DHT11
6. Figure 5: Logical Flow the “Intelligent Electricity Consumption Monitoring System”
7. Figure 6: Schematic of the System

1. Definition

Initially, the system will collect data on energy consumption from available datasets on the Internet called “Energy Consumption Dataset” from UCIRVINE assuming that the available dataset is of particular city or area and using a classification model will divide the data set into timeframes when the energy consumption is the most. According to this classification and the cost per unit of electricity calculated the system will generate the timeframes when the energy consumption is higher than needed and vice versa.

Along with consumption data, the device will also concurrently collect current and predicted data of weather. All this data can be stored in SD card of Raspberry Pi. Using a temperature sensor along with the Raspberry Pi, the current room temperature will be calculated. Depending on the data, two cases can be considered:

Case 1: When the outside temperature as taken from the web is low and the room temperature is also low (i.e. air-conditioning is on). In this case there will be two other cases. The first case is applied if it falls in the timeframe when consumption is very high. Then it will alert the user to stop consumption as he will be overcharged. In the other case, if it does not fall within a higher consumption timeframe, we will warn the user of higher energy consumption since the outside temperature is lower.

Case 2: When the outside temperature is high and the indoor temperature is not pleasant, we will suggest the best suitable room temperature to put the thermostat on in order to maximize energy savings and minimize costs.

The concept is to emphasize on reducing the carbon footprint to save the environment.

The system has an inbuilt multivariate linear regression module which takes two parameters i.e. days and hours and based on those inputs predicts what the power consumption will be for that particular day and hour. This will provide the company a broader view of what the power requirements will be in the future. It can help in proper management of resources, and a well-planned budget can be prepared for the same. It will help the company make decisions on which city needs expansion at their power plants and which city uses less than required electricity. To get better precision in prediction, neural networks can be implemented.

The project emphasises on collecting real time data from internet and comparing it to data/readings taken by the sensor. The project uses minimal hardware and still gives output that can be used in saving energy as well as money.

2. Block Diagram

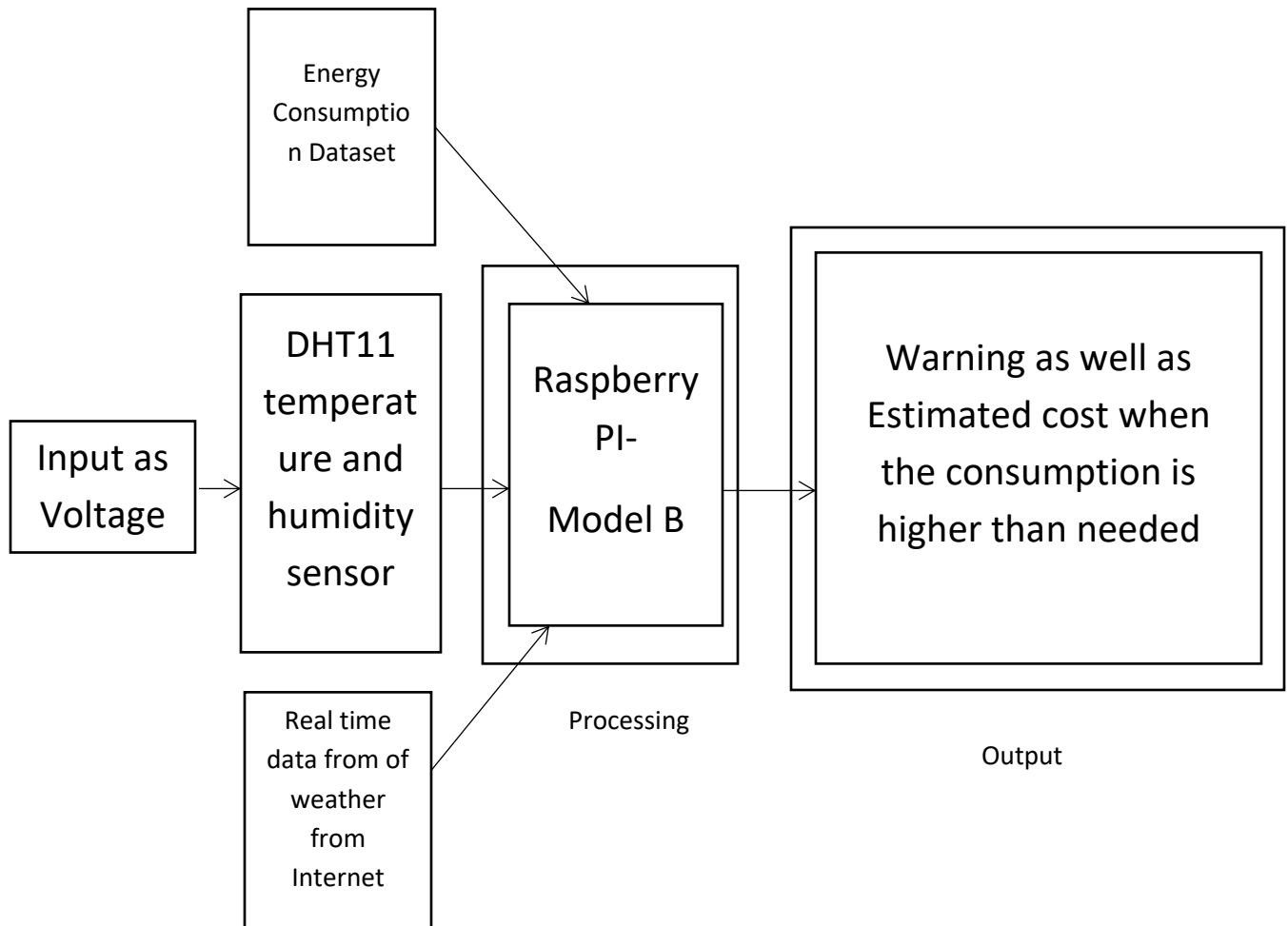


Figure 1: Block diagram of Intelligent Electricity Consumption Monitoring System

3. List of Components

Table: 1.1 List of components and needed Quantity

Component	Quantity
Raspberry Pi Model B	1
DHT11 – Temperature	1

4. Selection Criteria

Table: 1.2 DTH11 Temperature and Humidity Sensor

	Sensor 1
Company	Robokits
Price	300/- INR
Measurement Range	0 – 50C (Temperature) 20-80% RH (Humidity)
Interfacing Type	Digital
Power Supply	3 V – 5 V
Principle of Operation	Measure change in resistance of thermistor and humidity sensing component

5. Working Principle of Sensors and Actuators

How the project works:

The project incorporates, utilises and combines data from three sources and gives output according to that.

1. Live Weather Data (Ahmedabad India)
2. UC Irvine Household Energy Consumption Dataset
3. Data from DHT11 (Temperature and Humidity sensor)

It is divided into two major parts. We have built this project keeping in mind that we are on the side of the energy company. First we take the temperature and humidity data given by the DHT11 sensor who's working in explained later. We consider these readings as to be the room temperature and humidity. We calculate the heat index from it. The heat index is the relative temperature in terms of humidity and absolute temperature. It symbolises the actual temperature felt by a human due to the presence of humidity in the atmosphere.

Second, we access the Weather API through Python to collect current weather and humidity data from the internet. The data is obtained through an API key which grants access to the data on the internet. Once we have both the information on hand we implement an algorithm to figure out discrepancies between them. An average normal human can stay healthy between temperatures of 20 – 25 degrees Celsius and around 30 – 40% humidity. Keeping this in mind, we take two scenarios. One when the outside temperature is too cold (10 – 15 degrees Celsius) and room temperature is around 30 – 35 degrees Celsius. This scenario indicates that the heater in the thermostat is working unnecessarily. Another scenario taken into consideration is when the room temperature is low and the outdoor temperature is high. This indicates unusual use of the air conditioner. In both the cases, the electricity consumption is unnecessarily increased.

Simultaneously, using the consumption data, we have implemented our own algorithm to calculate which particular hours an average household has the maximum electricity consumption. Maximum limit set by us for this is 18 Amperes. Anything above that is considered excessive usage. If during those hours, the above explained scenarios occur, the energy charge per unit will be doubled. The estimate per hour of extra charge is calculated and given to the user along with the consumption data. The charge is calculated on the assumption that an average household will have 2 1.5 tonnage air conditioners running with some efficiency. According to that, they use 15.4 kWh (15.4 units of electricity per hour). Per unit charge currently in Ahmedabad is 490 paise. According to these parameters, the approximate increase in the electricity bill is calculated and displayed to the user.

Additionally, we have used machine learning to implement a multivariate linear regression algorithm to predict. This will provide the company a broader view of what the power requirements will be in the future. It can help in proper management of resources, and a well-planned budget can be prepared for

the same. It will help the company make decisions on which city needs expansion at their power plants and which city uses less than required electricity.

DHT11: Temperature and Humidity Sensor:

DHT11 Temperature & Humidity Sensor features a temperature & humidity sensor complex with a calibrated digital signal output. By using the exclusive digital-signal-acquisition technique and temperature & humidity sensing technology, it ensures high reliability and excellent long-term stability. This sensor includes a resistive-type humidity measurement component and an NTC temperature measurement component, and connects to a high performance 8-bit microcontroller, offering excellent quality, fast response, anti-interference ability and cost-effectiveness. [5]



Figure 2a: Integrated DHT11 Sensor [1]

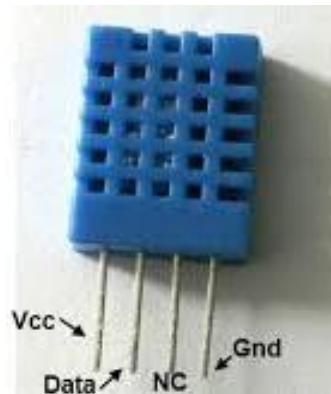


Figure 2b: Generic DHT11 Sensor [2]

The generic sensor has 4 pins namely VCC, Data Output, NC (Not connected) and Ground from left to right direction (See figure 2b). NC is used for the output of humidity. Here as shown in figure 2a integrated sensor there are only three pins VCC, DATA and GND as “not connected” pin is not there. According to circuit shown in figure 3 a pull-up resistor of 5-10K is needed between voltage pin and Data output pins. Capacitor is added between GND and VCC for power filtering.

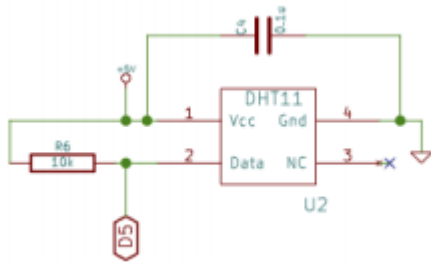


Figure 3: Internal circuit of Integrated DHT11 [3]

The sensor includes thermistor and along with thermistor a humidity sensing module and IC is connected. Which is shown below in figure: 4

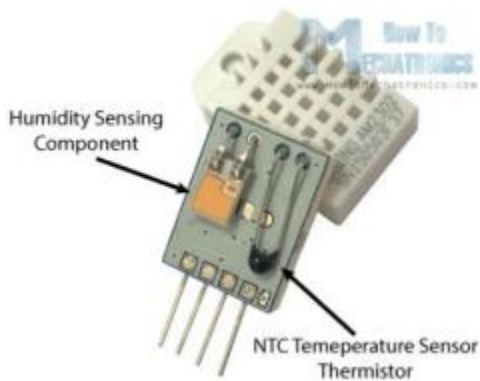


Figure 4: Components integrated in DHT11 [4]

Thermistor is the resistor that changes the value of the resistance as the temperature changes. Here NTC(Negative Temperature Coefficient) thermistor is used where the resistance decreases as the temperature increases or vice versa. These resistors are made with materials such as ceramics or polymers so that even smaller changes in temperature lead to significantly larger changes in resistance of thermistor.

Communication Process:

DHT11 uses single wire two way communication protocol which reduces the cost and extends the distance. Communication Format can be separated into three stages:

1. Request: To make the DHT11 to send sensor readings a request needs to be send which is to pull down the bus for more than 18ms and then pull it up for 40us.
2. Response: This is an automatic reply from DHT which indicates that DHT received your request. The response is 54uS low and 80uS high.
3. Data reading: The data sent by the sensor is packed in a packet of 5 segments each of 8 bits. First two segments are humidity read, integer and decimal respectively. Next two is temperature read, integer and decimal. And the last segment is of parity.
Bit '0': 54uS Low and 24uS High
Bit '1': 54uS Low and 70uS High
At the end of packet DHT sends a 54uS Low level, pulls the bus to High and goes to sleep mode.

[6]

6. Flow Chart

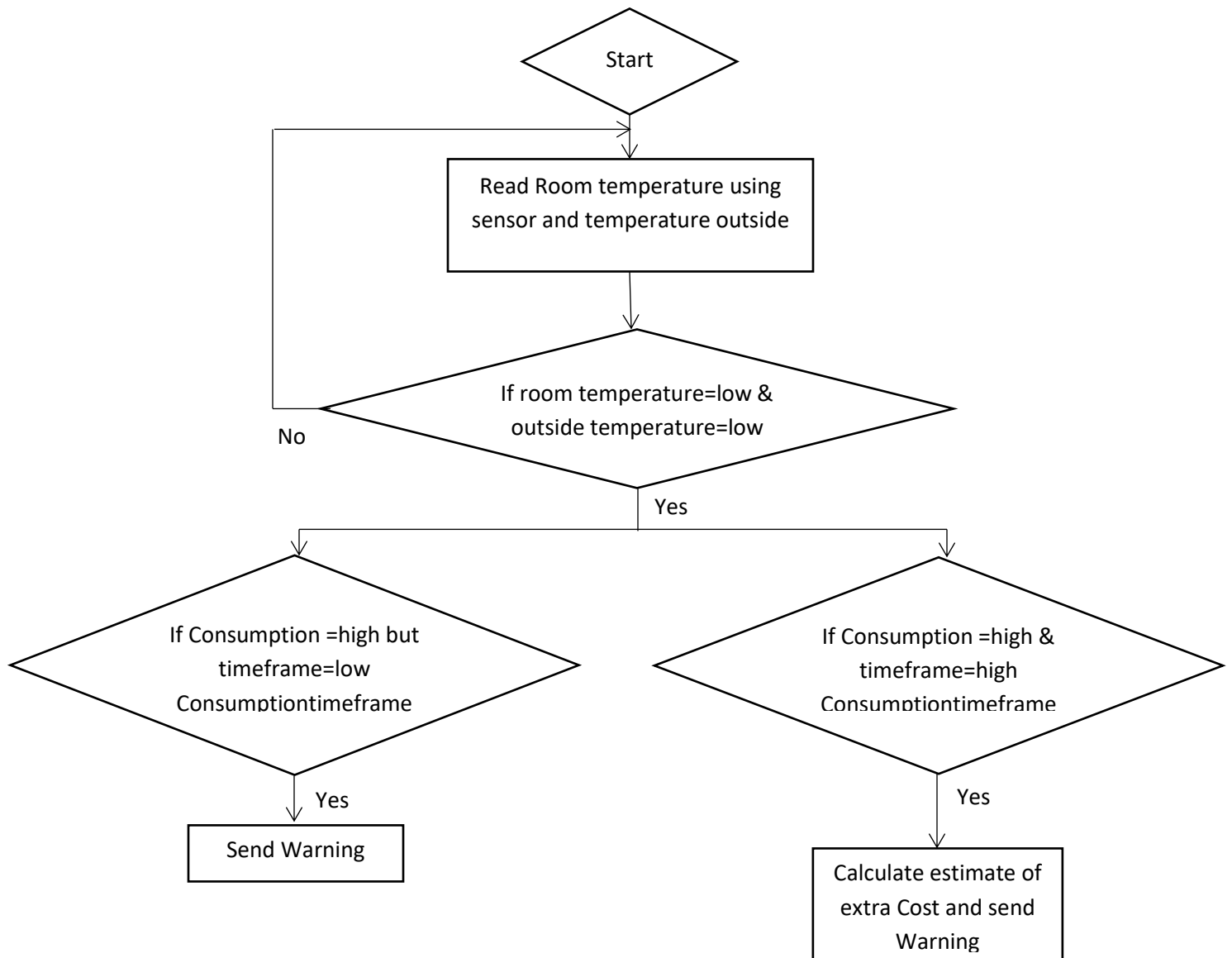


Figure 5: Logical Flow the "Intelligent Electricity Consumption Monitoring System"

DERIVATION OF EQUATIONS FOR THE GRADIENT DESCENT ALGORITHM

$$h_{\theta}(x_0, x_1) = \theta_0 x_0 + \theta_1 x_1 + \theta_2$$

$$J = \frac{1}{2m} \sum_{i=1}^{24} \sum_{j=1}^{31} \left\{ (\theta_0 x_0^{(i)} + \theta_1 x_1^{(i)} + \theta_2) - z^{(i)} \right\}^2$$

Here $\theta_0, \theta_1, \theta_2$ are the parameters of 2-D plane.
 z is the data from the dataset of "UCIRVI".

$$\frac{dJ}{d\theta_0} = \frac{1}{m} \sum_{i=1}^{24} \sum_{j=1}^{31} x_0^{(i)} \left(\theta_0 x_0^{(i)} + \theta_1 x_1^{(i)} + \theta_2 - z^{(i)} \right)$$

$$\frac{dJ}{d\theta_1} = \frac{1}{m} \sum_{i=1}^{24} \sum_{j=1}^{31} x_1^{(i)} \left(\theta_0 x_0^{(i)} + \theta_1 x_1^{(i)} + \theta_2 - z^{(i)} \right)$$

$$\frac{dJ}{d\theta_2} = \frac{1}{m} \sum_{i=1}^{24} \sum_{j=1}^{31} \left(\theta_0 x_0^{(i)} + \theta_1 x_1^{(i)} + \theta_2 - z^{(i)} \right)$$

$x_0^{(i)}$ represents normalized hour & $x_1^{(i)}$ represents normalized day.

$$\theta_0 = \theta_0 - \frac{dJ}{d\theta_0} \quad \theta_1 = \theta_1 - \frac{dJ}{d\theta_1} \quad \theta_2 = \theta_2 - \frac{dJ}{d\theta_2}$$

Through the above equations we get updated parameters on each iteration.

7. Schematic

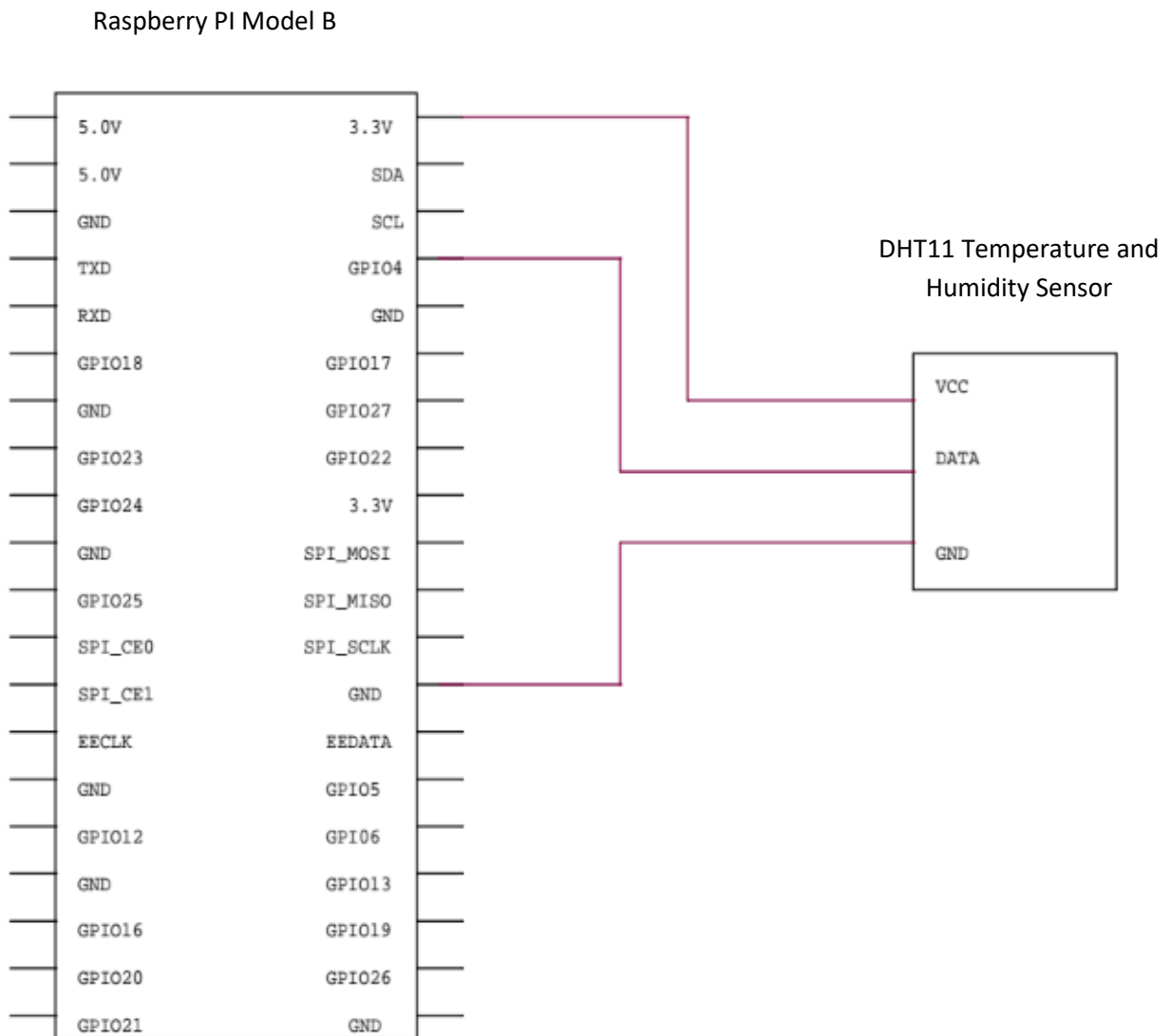


Figure 6: Schematic of the System

8. Code

Multivariate Linear Regression Code

```
1. from __future__ import division
2. import numpy as np
3. from matplotlib import pyplot as plt
4. import glob
5. import csv
6.
7. def calcJ( matrix, hour1, day1, theta0, theta1, theta2 ):
8.     J = 0;
9.     for i in xrange(24):
10.         for j in xrange(31):
11.
12.             J = J + ((theta0*hour1[i] + theta1*day1[j] + theta2) -
matrix[i][j]) * ((theta0*hour1[i] + theta1 * day1[j] + theta2) -
matrix[i][j]);
13.             #print image_list[i],distance[i],J,theta0,theta1
14.             J = 0.125 * J;
15.             #print J
16.             return J
17.
18.
19.
20. #for filename in glob.glob('[0-9]*ft.jpg'): #assuming gif
21.     ifile = open('data.csv', "rb")
22.     reader = csv.reader(ifile)
23.
24.     rownum = 0
25.
26.     w, h = 31 , 24
27.     matrix = [[0 for x in range(w)] for y in range(h)]
28.
29.     frequency = [0 for y in range(h)]
30.     hour1 = [0.0, 0.043478260869565216, 0.08695652173913043,
0.13043478260869565, 0.17391304347826086, 0.21739130434782608,
0.2608695652173913, 0.30434782608695654,
0.34782608695652173,0.391304347826087,0.43478260869565216,
0.4782608695652174, 0.5217391304347826, 0.5652173913043478,
0.6086956521739131, 0.6521739130434783, 0.6956521739130435,
0.7391304347826086, 0.782608695652174, 0.8260869565217391,
0.8695652173913043, 0.9130434782608695, 0.9565217391304348, 1.0]
31.     day1 = [0.03225806451612903, 0.06451612903225806, 0.0967741935483871,
0.12903225806451613, 0.16129032258064516, 0.1935483870967742,
0.22580645161290322, 0.25806451612903225, 0.2903225806451613,
0.3225806451612903, 0.3548387096774194, 0.3870967741935484,
0.41935483870967744, 0.45161290322580644, 0.4838709677419355,
0.5161290322580645, 0.5483870967741935, 0.5806451612903226,
0.6129032258064516, 0.6451612903225806, 0.6774193548387096,
0.7096774193548387, 0.7419354838709677, 0.7741935483870968,
0.8064516129032258, 0.8387096774193549, 0.8709677419354839,
0.9032258064516129, 0.9354838709677419, 0.967741935483871, 1.0]
32.
33.
```

```

34.     for row in reader:
35.
36.
37.         if rownum == 0:
38.             header = row
39.             summ = 0
40.             day = 0
41.             hour = 0
42.         else:
43.             colnum = 0
44.             for col in row:
45.                 if colnum % 4 == 3:
46.                     summ += int( col)
47.
48.             colnum += 1
49.
50.         if rownum % 60 == 1:
51.             #print '%s %-8s: %f %d' % (row[0],header[1],summ /60.0 ,rownum)
52.             matrix[hour][day] = summ / 60.0
53.             if hour == 9 and day == 0:
54.                 print hour,day
55.                 print matrix[hour][day]
56.                 print '%s %-8s: %f %d' % (row[0],header[1],summ /60.0
, rownum)
57.                 summ = 0
58.                 hour += 1
59.                 if hour % 24 == 0:
60.                     hour = 0
61.                     day += 1
62.
63.             rownum += 1
64.
65.     ifile.close()
66.
67.     for x in range(w):
68.         for y in range(h):
69.             if matrix[y][x] > 15:
70.                 frequency[y] += 1
71.
72.
73.     print frequency
74.
75.     #theta0 = 9
76.     #theta1 = -1/30
77.
78.     theta0 = 1
79.     theta1 = 0
80.     theta2 = 1
81.
82.     Jold = 0
83.     Jnew = -10
84.     diff_J0 = 0
85.     diff_J1 = 0
86.     diff_J2 = 0
87.     counter = 0
88.

```

```

89.     while ((Jold - Jnew > 0.0000000000000001) or (Jnew - Jold >
0.0000000000000001)):
90.         diff_J0 = 0;
91.         diff_J1 = 0;
92.         diff_J2 = 0;
93.         for i in xrange(24):
94.             for j in xrange(31):
95.                 temptheta = 1 * ((theta0*hour1[i] + theta1*day1[j] +
theta2) - matrix[i][j]) ;
96.                 diff_J0 = diff_J0 + 1 * hour1[i] * ((theta0*hour1[i] +
theta1*day1[j] + theta2) - matrix[i][j]) / 720.0;
97.
98.
99.                 temp0 = theta0 - diff_J0;
100.
101.                 diff_J1 = 0
102.                 for i in xrange(24):
103.                     for j in xrange(31):
104.                         temptheta = 1 * ((theta0*hour1[i] + theta1*day1[j] +
theta2) - matrix[i][j]) / 720.0;
105.                         diff_J1 = diff_J1 + 1 * day1[j] * ((theta0*hour1[i] +
theta1*day1[j] + theta2) - matrix[i][j]) / 720.0;
106.
107.
108.                 temp1 = theta1 - diff_J1;
109.
110.                 diff_J2 = 0
111.                 for i in xrange(24):
112.                     for j in xrange(31):
113.                         temptheta = 1 * ((theta0*hour1[i] + theta1*day1[j] +
theta2) - matrix[i][j]) / 720.0;
114.                         diff_J2 = diff_J2 + 1 * ((theta0*hour1[i] + theta1*day1[j]
+ theta2) - matrix[i][j]) / 720.0;
115.
116.
117.                 temp2 = theta2 - diff_J2;
118.
119.                 theta0 = temp0
120.                 theta1 = temp1
121.                 theta2 = temp2
122.                 counter = counter + 1;
123.                 Jold = Jnew
124.                 #print Jold
125.                 #print Jnew
126.                 Jnew = calcJ(matrix, hour1, day1, theta0, theta1, theta2)
127.                 counter += 1;
128.
129.     print theta0, theta1, theta2
130.     #print Jold, Jnew
131.     print counter
132.
133.
134.     print matrix[10][0]

```

Main Code:

```
1. import RPi.GPIO as GPIO      #import libraries and utilities for GPIO
2. import dht11
3. import time
4. import datetime
5. import pyowm                  #import Python GitHub API for weather data
6. import csv
7. import numpy as np
8. import socket                 #Import socket library for wireless communication

9. ifile = open('data.csv','rb')  #Open Energy Consumption Dataset
10. reader = csv.reader(ifile)    #Read Energy consumption dataset
11. now = datetime.datetime.now()  #Get current date and time

12. #initialize GPIO for input/output
13. GPIO.setwarnings(False)
14. GPIO.setmode(GPIO.BCM)
15. GPIO.cleanup()

16. #Defining constants for calculating heat index
17. c1 = -42.379
18. c2 = 2.04901523
19. c3 = 10.14333127
20. c4 = -0.22475541
21. c5 = -6.83783*pow(10,-3)
22. c6 = -5.481717*pow(10,-2)
23. c7 = 1.22874*pow(10,-3)
24. c8 = 8.5282*pow(10,-4)
25. c9 = -1.99*pow(10,-6)

26. #defining electricity rates and units for the average household
27. units = 15.4
28. noofac = 2
29. rate = 4.9
30. ratemax = rate*2
31. cost = rate*units*noofac
32. costmax = ratemax*units*noofac

33. #Read data using pin 4
34. instance = dht11.DHT11(pin=4)

35. #API Key to use Weather API to collect current data
36. owm = pyowm.OWM('a84022c6391609d95b57dabe30d9d809')

37. #Connect to host
38. host='10.20.32.156'
39. s = socket.socket()          #Create a socket object
40. port = 2345                  #Reserve a port for your service.
41. s.connect((host, port))

42. #Initialise matrix to zeros to store consumption data
43. rownum = 0
```



```

44. #31X24 matrix for each day and each hour of the month
45. w, h = 31 , 24
46. matrix = [[0 for x in range(w)] for y in range(h)]
47. frequency = [0 for y in range(h)]

48. #Read file and calculate the number of instances of greater intensity
    amperage
49. for row in reader:
50.     if rownum == 0:
        a. header = row          #header is zeroth row
        b. summ = 0              #Initialize all variables to zero
        c. day = 0
        d. hour = 0
51.     else:
        a. colnum = 0
        b. for col in row: #3rd column in dataset for intensity
            i. if colnum % 4 == 3:
                ii. summ += int( col)    #summation of all intensity per minute
                    values of particular hour

        c. colnum += 1            #increment column

52.     if rownum % 60 == 1:        #next hour
        a. matrix[hour][day] = summ / 60.0 #mean of intensity of that
            particular hour for the whole month

        b. summ = 0
        c. hour += 1              #increment hour
        d. if hour % 24 == 0:      #change of day
            i. hour = 0
            ii. day += 1

53.     rownum += 1              #increment row

54.     ifile.close()

55.     #if intensity average value is greater than 15, log the value
56.     for x in range(w):
57.         for y in range(h):
            a. if matrix[y][x] > 15:
                i. frequency[y] += 1

58.     #If there are more than 18 instances of higher intensity figure out the
        hour for that.
59.     for i in frequency:
60.         if frequency[i] > 18:
            a. print frequency[i], i
            b. if now.hour == frequency[i]:
            c. print 'true'
61.     else:
        a. print now.hour
        b. print 'false'

62.     while True:
63.         sensor = instance.read()

```

```

64.     #if sensor.is_valid():
65.     print("Last valid input: " + str(datetime.datetime.now()))
66.     print("Temperature: %d C" % sensor.temperature)
67.     print("Humidity: %d %" % sensor.humidity)
68.     indoorF = sensor.temperature*(9/5.0) + 32
69.     #Calculate the heat index
    a. heatindex_room_F = c1 + c2*indoorF + c3*sensor.humidity +
        c4*indoorF*sensor.humidity + c5*pow(indoorF,2) +
        c6*pow(sensor.humidity,2) + c7*pow(indoorF,2)*sensor.humidity +
        c8*indoorF*pow(sensor.humidity,2) +
        c9*pow(sensor.humidity,2)*pow(indoorF,2)
70.     heatindex_room_C = (heatindex_room_F-32)*(5/9.0)

71.     #Search for current weather in Ahmedabad, India
72.     observation = owm.weather_at_place('Ahmadabad, IN')
73.     w = observation.get_weather()
74.     print(w)
    a. temp = w.get_temperature('celsius')
    b. api_temp = temp["temp_max"]
    c. api_humidity = w.get_humidity()
75.     print api_temp

76.     #Send data through socket to server
77.     s.send(repr(now))
78.     s.send(repr(api_temp))
79.     s.send(repr(sensor.temperature))

80.     #difference = sensor - api_temp;
81.     #Two scenarios as described in the report of discrepancies in the
    temperature
82.     if (sensor.temperature > 25 and sensor.temperature < 38) and (api_temp
    < 35 and api_temp > 20):
        a. print "No need of heating its pleasant outside"
        b. print("You are increasing your bill by %d rupees per hour " %
            costmax)

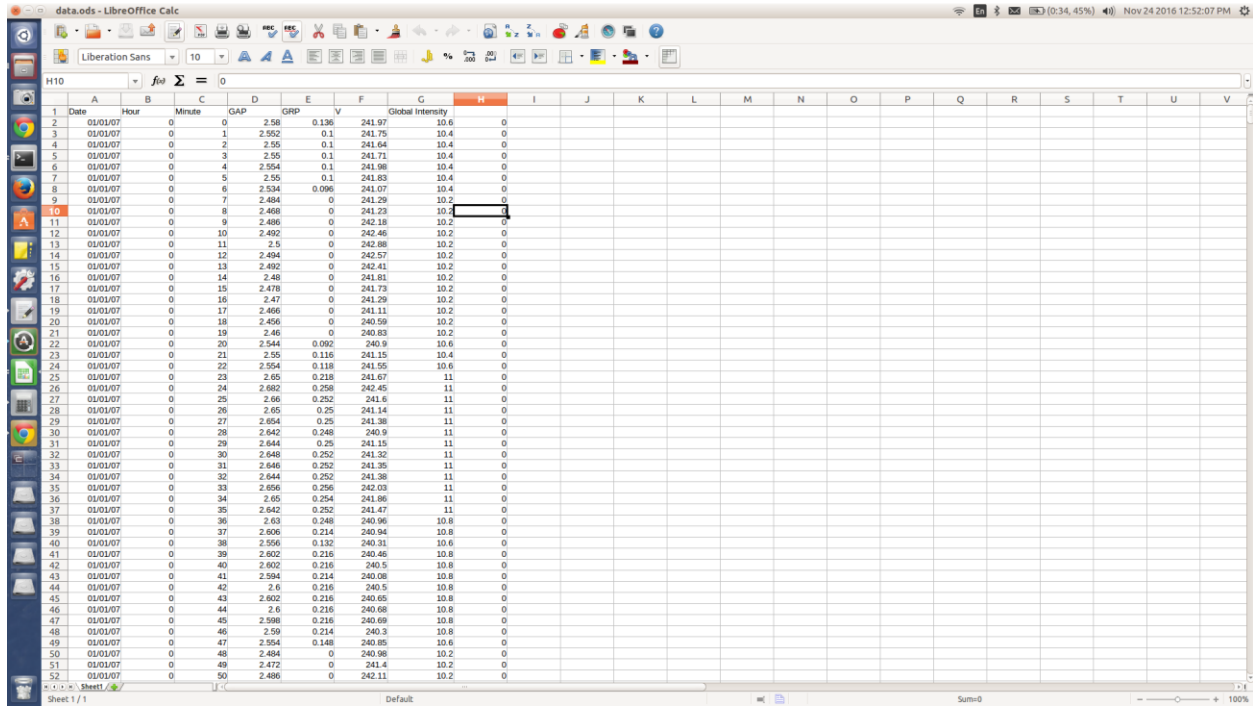
83.     if sensor.temperature > 10 and sensor.temperature < 18 and api_temp >
    30:
        a. print "No need of AC its pleasant outside"
        b. print("You are increasing your bill by %d rupees per hour " %
            costmax)
84.     else:
        a. #s.send(repr(cost))
        b. print("You are increasing your bill by %d rupees per hour " % cost)

85.     time.sleep(10)
86.     s.close
87. #Close the socket when done

```

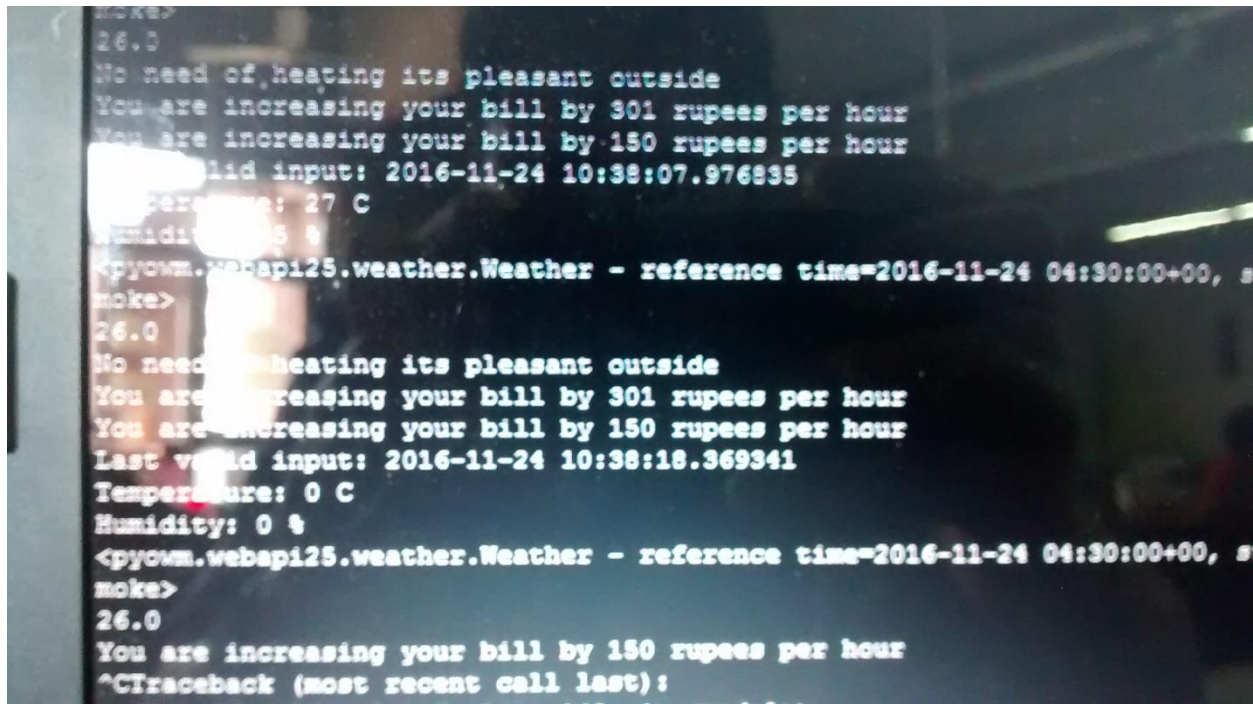
9. Screenshots

INPUT



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	Date	Hour	Minute	GAP	GRP	V	Global Intensity															
2	01/01/07	0	0	2.58	0.136	241.97	10.6	0														
3	01/01/07	0	1	2.552	0.1	241.75	10.4	0														
4	01/01/07	0	2	2.55	0.1	241.64	10.4	0														
5	01/01/07	0	3	2.55	0.1	241.71	10.4	0														
6	01/01/07	0	4	2.554	0.1	241.98	10.4	0														
7	01/01/07	0	5	2.55	0.1	241.83	10.4	0														
8	01/01/07	0	6	2.534	0.096	241.07	10.4	0														
9	01/01/07	0	7	2.484	0	241.29	10.2	0														
10	01/01/07	0	8	2.466	0	241.23	10.2	0														
11	01/01/07	0	9	2.486	0	242.18	10.2	0														
12	01/01/07	0	10	2.492	0	242.46	10.2	0														
13	01/01/07	0	11	2.5	0	242.88	10.2	0														
14	01/01/07	0	12	2.494	0	242.57	10.2	0														
15	01/01/07	0	13	2.492	0	242.41	10.2	0														
16	01/01/07	0	14	2.48	0	241.81	10.2	0														
17	01/01/07	0	15	2.478	0	241.73	10.2	0														
18	01/01/07	0	16	2.47	0	241.29	10.2	0														
19	01/01/07	0	17	2.466	0	241.11	10.2	0														
20	01/01/07	0	18	2.456	0	240.99	10.2	0														
21	01/01/07	0	19	2.46	0	240.83	10.2	0														
22	01/01/07	0	20	2.544	0.092	240.9	10.6	0														
23	01/01/07	0	21	2.55	0.116	241.15	10.4	0														
24	01/01/07	0	22	2.554	0.118	241.55	10.6	0														
25	01/01/07	0	23	2.65	0.218	241.67	11	0														
26	01/01/07	0	24	2.682	0.258	242.45	11	0														
27	01/01/07	0	25	2.66	0.252	241.6	11	0														
28	01/01/07	0	26	2.65	0.25	241.14	11	0														
29	01/01/07	0	27	2.654	0.25	241.38	11	0														
30	01/01/07	0	28	2.642	0.248	240.9	11	0														
31	01/01/07	0	29	2.644	0.25	241.15	11	0														
32	01/01/07	0	30	2.646	0.252	241.32	11	0														
33	01/01/07	0	31	2.646	0.252	241.35	11	0														
34	01/01/07	0	32	2.644	0.252	241.38	11	0														
35	01/01/07	0	33	2.656	0.256	242.03	11	0														
36	01/01/07	0	34	2.65	0.254	241.86	11	0														
37	01/01/07	0	35	2.642	0.252	241.47	11	0														
38	01/01/07	0	36	2.63	0.248	240.96	10.8	0														
39	01/01/07	0	37	2.606	0.214	240.94	10.8	0														
40	01/01/07	0	38	2.556	0.132	240.31	10.6	0														
41	01/01/07	0	39	2.602	0.216	240.46	10.8	0														
42	01/01/07	0	40	2.602	0.216	240.5	10.8	0														
43	01/01/07	0	41	2.594	0.214	240.08	10.8	0														
44	01/01/07	0	42	2.6	0.216	240.5	10.8	0														
45	01/01/07	0	43	2.602	0.216	240.65	10.8	0														
46	01/01/07	0	44	2.6	0.216	240.68	10.8	0														
47	01/01/07	0	45	2.598	0.216	240.69	10.8	0														
48	01/01/07	0	46	2.59	0.214	240.3	10.8	0														
49	01/01/07	0	47	2.554	0.148	240.85	10.6	0														
50	01/01/07	0	48	2.484	0	240.98	10.2	0														
51	01/01/07	0	49	2.472	0	241.4	10.2	0														
52	01/01/07	0	50	2.486	0	242.11	10.2	0														

OUTPUT



```
Smoke>
26.0
No need of heating its pleasant outside
You are increasing your bill by 301 rupees per hour
You are increasing your bill by 150 rupees per hour
Last valid input: 2016-11-24 10:38:07.976835
Temperature: 27 C
Humidity: 0 %
<pyowm.webapi25.weather.Weather - reference time=2016-11-24 04:30:00+00, s
Smoke>
26.0
No need of heating its pleasant outside
You are increasing your bill by 301 rupees per hour
You are increasing your bill by 150 rupees per hour
Last valid input: 2016-11-24 10:38:18.369341
Temperature: 0 C
Humidity: 0 %
<pyowm.webapi25.weather.Weather - reference time=2016-11-24 04:30:00+00, s
Smoke>
26.0
You are increasing your bill by 150 rupees per hour
^CTraceback (most recent call last):
```

```

You are increasing your bill by 301 rupees per hour
Last valid input: 2016-11-24 10:38:07.976835
Temperature: 27 C
Humidity: 35 %
<pyowm.webapi25.weather.Weather - reference time=2016-11-24 04:30:00+00, s
make>
26.0
No need of heating its pleasant outside
You are increasing your bill by 301 rupees per hour
You are increasing your bill by 150 rupees per hour
Last valid input: 2016-11-24 10:38:18.369341
Temperature: 0 C
Humidity: 0 %
<pyowm.webapi25.weather.Weather - reference time=2016-11-24 04:30:00+00, st
make>
26.0
You are increasing your bill by 150 rupees per hour
You are increasing your bill by 150 rupees per hour
^CTraceback (most recent call last):
  File "finalfinal.py", line 142, in <module>

```

```

You are increasing your bill by 150 rupees per hour
^C[Traceback (most recent call last):
  File "finalfinal.py", line 142, in <module>
    time.sleep(10)
KeyboardInterrupt
^Cpi@raspberrypi:~/Desktop/iotproject $ sudo nano reg
pi@raspberrypi:~/Desktop/iotproject $ sudo python reg
[4, 2, 1, 1, 1, 0, 0, 10, 16, 15, 13, 13, 13, 13, 16,
7.66006357093 2.55555431938 2.2348434702
490
The calculated value is 2
The predicted value is 5
pi@raspberrypi:~/Desktop/iotproject $

```

10. Difficulties

We faced the following difficulties in the project

1. **Entering missing values:** In the UC Irvine data set there were a lot of values. We faced a lot of difficulties while reading the values in Python. Initially we didn't know what the error was and why it occurred. Upon a lot, of researching we found that the error was due to missing values in the CSV file. So we performed data cleaning on the entire data set.
2. **Implementing Multivariate Linear Regression:** We implemented linear regression without using any inbuilt libraries. We had to derive the gradient descent equations on our own.

11. References

1. UC Irvine Machine Learning Repository Dataset
<http://archive.ics.uci.edu/ml/datasets/Individual+household+electric+power+consumption>
2. Image on E-bay website for integrated DHT11:
<http://www.ebay.com/itm/DHT11-Temperature-and-Humidity-Sensor-Module-for-Arduino-/271096647277>
3. Image on Embedded Lab for generic DHT11:
<http://embedded-lab.com/blog/measurement-of-temperature-and-relative-humidity-using-dht11-sensor-and-pic-microcontroller/>
4. Image from cdn.hackaday.io for internal circuit diagram of DHT11:
<https://hackaday.io/project/3475/logs>
5. Nedelkovski D. (2016), DHT11 and DHT22 Sensors Temperature and Humidity Tutorial Using Arduino:
<http://howtomechatronics.com/tutorials/arduino/dht11-dht22-sensors-temperature-and-humidity-tutorial-using-arduino/>
6. Working of DHT11:
<http://www.micropik.com/PDF/dht11.pdf>
7. Communication process in DHT11:
<http://robocraft.ru/files/datasheet/DHT11.pdf>