

Automated Greenhouse monitoring and controlling system

Shivam Batra
Computer Science with
Cyber Physical Systems
Vellore Institute of
Technology
Chennai, India
shivam.batra2019@vitstu-
dent.ac.in

Viresh Bhurke
Computer Science with
Cyber Physical Systems
Vellore Institute of
Technology
Chennai, India
viresh.bhurke2019@vitst-
udent.ac.in

Tanmay Deshpande
Computer Science with
Cyber Physical Systems
Vellore Institute of
Technology
Chennai, India
tanmay.deshpande2019-
@vitstudent.ac.in

Abstract— With technology evolving quickly enough for science fiction to become an overnight reality, it's not surprising that news outlets seem to feature vertical agriculture, indoor farming, or hydroponic growing every few days. Futuristic urban buildings that contain lettuce grown in plastic pipes and farmers dressed in lab-coats do make for a great story, after all.

In our project we plan to create an environment which can control the temperature, grow lights, humidity and check the pH level of soil automatically. The system can also automatically water the plants. The system uses Arduino to configure all the sensors and activate fan, water or different mechanisms to modulate the temperature, humidity and pH of the soil. The project aims to improve the yield in quality and quantity largely. It also makes it possible to grow rare plants(exotic flowers) anywhere in the world. The portability also makes the system very convenient.

In this an automated greenhouse was built with the purpose of investigating the watering system's reliability and if a desired range of temperatures can be maintained. The microcontroller used to create the automated greenhouse was an Arduino UNO. This project utilizes two different sensors, a photosensor and a temperature sensor. The sensors are controlling the two actuators which are a heating fan and a pump. The heating fan is used to change the temperature and the pump is used to water the plant. The watering system and the temperature control system were tested both separately and together. The result showed that the temperature could be maintained in the desired range.

I. INTRODUCTION

Background :

In modern society, the consumption of fruits and vegetables has become the norm. A variety of fresh fruits and vegetables should be accessible at all times. However, the northern

climate prevents the growth of certain fruits and vegetables, especially during winter. This results in import from southern countries, which in turn has some drawbacks. Not only does the shipping of imported goods affect the environment negatively but imported food or vegetables are also less flavourful and sold to a higher price. The crops must be harvested prematurely when importing food. This is to delay the ripening process so that it is possible for the fruits or vegetables to reach their destination before they are considered inedible. The ripening process is later resumed by spraying the food with ethylene gas, a gas that is deemed to promote ripening in certain fruits and vegetables. However, this postharvest ripening can lead to poor taste. In the past couple of years, an increased interest in organic and locally produced food has become a growing trend. Locally grown foods are picked at their peak of ripeness and are therefore full of flavor. Furthermore, growing food at home assures truly organic food. On the other hand it takes a lot of time and effort, and time is something that most people lack today.

Purpose :

The purpose of this project was therefore to make it easier to grow food at home. This can be achieved with the use of an automated greenhouse. A greenhouse makes it possible to replicate a different climate and consequently grow food that would not typically grow in the

area. Additionally, making the greenhouse automated enables people to grow their own food or plants at home without having to constantly look after them. It can be reassuring to know that the plants are taken care of while one is on vacation or not around the house for a longer period of time. The research question of this study was to analyze if it is possible to maintain the greenhouse temperature in a desired range for optimal plant growth using a temperature control system. Another objective was to investigate if the watering system is reliable, that is whether or not it can obtain a perfect soil moisture level for the chosen plant.

Method :

Automated Grow Lab is an artificially system environment designed to grow plants. It is designed to create an ideal environment for any type of plant. The main goal is to get a greater yield and higher quality.

Various natural aspects are controlled by the system which largely affect the growth of the plant. The temperature is managed by controlling the speed of the fan in the designed vent. The temperature determines the rate of plant development and consequently affects length of the total growing period of the plant. Relative humidity levels affect when and how plants open the stomata on the undersides of their leaves. When relative humidity levels are too high or there is a lack of air circulation, a plant cannot make water evaporate (part of

the transpiration process) or draw nutrients from the soil. The growlights help us manipulate the UV rays on the plants.

The whole system is automated and is an open loop system. The readings are taken in order to change the temperature, humidity and sunlight. It is configured by Arduino which has a temperature sensor , and humidity sensor.

The formatter will need to create these components, incorporating the applicable criteria that follow.



Fig. 1

II. EXISTING SYSTEM

Automated Greenhouse Projects:

Plenty of projects involving gardening and Arduinos have been done. An Arduino is a prototyping platform which is well suited for interactive projects since it can sense its surroundings with the help of sensors and subsequently affect the same surroundings by controlling actuators [Arduino, 2015]. There is in fact a term for gardening Arduinos,

“Garduinos“. Some previous projects are simple and others are complex. What can count as Garduino is not defined but most commonly they all have a watering system. Other features can be carbon dioxide and light regulation. This project differs from other “Garduinos“ since it focuses more on temperature control.

Sensors :

This section presents the two sensors needed to build the automated greenhouse and to answer the research question. DHT11 was used to sense temperature and humidity. Photoelectric sensor is used to sense if light falls under bright/dark/dim category.



Fig. 2

III. PROBLEM STATEMENT

There are various problems faced in these systems when maximum efficiency needs to be achieved.

1) Uncontrollable temperature.

High temperature, even for short period, affects crop growth especially in temperate crops like wheat. High air temperature reduces the growth of shoots and in turn reduces root growth. High soil temperature is more crucial as damage to the roots is severe resulting in substantial reduction in shoot growth.

2) Uncontrollable humidity.

When conditions are too humid, it may promote the growth of mold and bacteria that cause plants to die and crops to fail, as well as conditions like

root or crown rot. Humid conditions also invite the presence of pests, such as fungus gnats, whose larvae feed on plant roots and thrive in moist soil.

3) Risk of insects.

Insects are responsible for two major kinds of damage to growing crops. First is direct injury done to the plant by the feeding insect, which eats leaves or burrows in stems, fruit, or roots. These swarms may completely destroy crops in an invaded region.

4) Can get affected by the outer environment.

IV. INTRODUCTION TO PROPOSED MODEL

Structure :

The proposed model has a huge advantage over other systems as it is designed to be portable and sturdy. The structure involves a potted planting system with the use of soil required by the plant. The different requirements of the plants are satisfied with the automated system programmed in the Arduino UNO along with ESP system to connect with the internet.

Material :

Lightweight polythene plastic, with plastic sturdy pots and glass covered box.

System :

In our project we plan to create an environment which can control the temperature, grow lights,

humidity and check the pH level of soil automatically. The system can also automatically water the plants. The system uses Arduino to configure all the sensors and activate fan, water or different mechanisms to modulate the temperature, humidity and pH of the soil. The project aims to improve the yield in quality and quantity largely. It also makes it possible to grow rare plants(exotic flowers) anywhere in the world. The portability also makes the system very convenient. The system is connected to the AWS S2 service where it uploads all the data generated.

V. COMPONENTS OF THE MODEL

1. Arduino Uno
2. Linux system (Blinky)
3. AWS S2 Service
4. Hadoop
5. Breadboard
6. Photoresistor
7. LED
8. Temperature sensor
9. Structural material
 - 1)Glass box
 - 2)DC Fan
 - 3)Roof
 - 4)Polythene plastic pipes

BLOCK DIAGRAM

The system is built with multiple processes and needs to prioritize its tasks based on the inputs from the environment. The system consists of UV lights(represented by normal LED), temperature sensor and photoresistor. These sensors give inputs continuously and need to be processed.

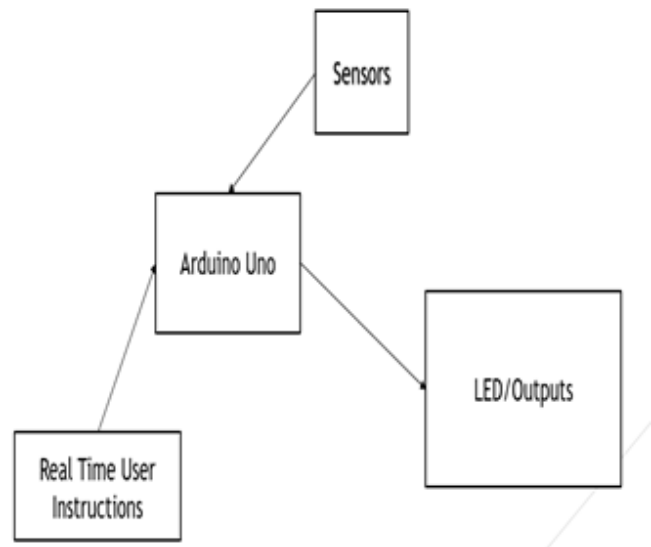


Fig. 3

VI. FLOW CHART

The following process occurs in the microcontroller connected to the Arduino Uno system :

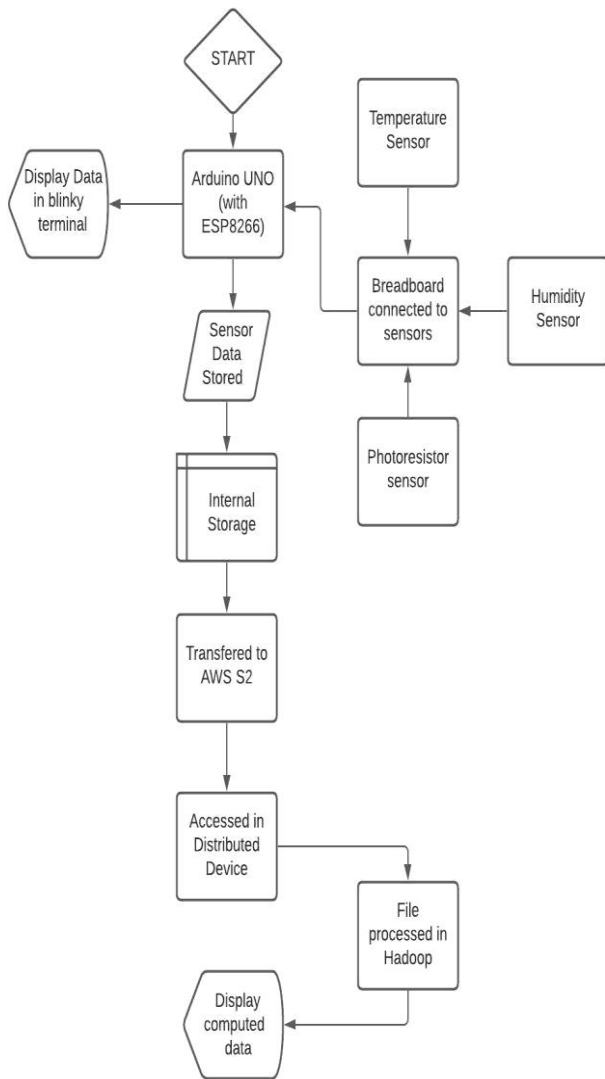


Fig. 4

VII. CODE

Arduino Blink :

```

int photocellPin = 0;    // the cell and 10K
                          // pulldown are connected to a0
int photocellReading;    // the analog reading
                          // from the sensor divider
int LEDpin = 9;          // connect Red LED to
                          // pin 9 (PWM pin)
int LEDbrightness;
//
int baselineTemp = 0;
int celsius = 0;
int fahrenheit = 0;
void setup(){
    // We'll send debugging information via the
    // Serial monitor
    pinMode(A1, INPUT);
    Serial.begin(9600);
    pinMode(2, OUTPUT);
    pinMode(3, OUTPUT);
    pinMode(4, OUTPUT);
}

```

```

void loop(void) {
    photocellReading
    analogRead(photocellPin);
    =

```

```

    // LED gets brighter the darker it is at the
    // sensor
    // that means we have to -invert- the reading
    // from 0-1023 back to 1023-0
    photocellReading = 1023 - photocellReading;
    //now we have to map 0-1023 to 0-255 since
    //that's the range analogWrite uses
    LEDbrightness = map(photocellReading, 0,
    1023, 0, 255);
    analogWrite(LEDpin, LEDbrightness);

```

```

    Serial.print("Analog reading = ");
    Serial.print(photocellReading);    // the raw
    // analog reading

```

```
// We'll have a few thresholds, qualitatively
determined
```

```
if (photocellReading < 10) {
  Serial.println(" - Dark");
} else if (photocellReading < 200) {
  Serial.println(" - Dim");
} else if (photocellReading < 500) {
  Serial.println(" - Light");
} else if (photocellReading < 800) {
  Serial.println(" - Bright");
} else {
  Serial.println(" - Very bright");
}
//delay(300);
baselineTemp = 40;

celsius = map(((analogRead(A1) - 20) *
3.04), 0, 1023, -40, 125);

fahrenheit = ((celsius * 9) / 5 + 32);
Serial.print(celsius);
Serial.print(" C, ");
```

```
Serial.print(fahrenheit);
Serial.println(" F");
int h =(celsius*2+5);
Serial.print("Humidity: ");
Serial.print(h);
Serial.print(" %");
if (celsius < baselineTemp) {
  digitalWrite(2, LOW);
  digitalWrite(3, LOW);
  digitalWrite(4, LOW);
}
if (celsius >= baselineTemp && celsius <
baselineTemp + 15) {
  digitalWrite(2, HIGH);
  digitalWrite(3, LOW);
  digitalWrite(4, LOW);
}
if (celsius >= baselineTemp + 20 && celsius
< baselineTemp + 40) {
  digitalWrite(2, HIGH);
```

```
digitalWrite(3, HIGH);
digitalWrite(4, LOW);
}
if (celsius >= baselineTemp + 40 && celsius
< baselineTemp + 60) {
  digitalWrite(2, HIGH);
  digitalWrite(3, HIGH);
  digitalWrite(4, HIGH);
}
if (celsius >= baselineTemp + 30) {
  digitalWrite(2, HIGH);
  digitalWrite(3, HIGH);
  digitalWrite(4, HIGH);
}
digitalWrite(8,HIGH); //HIGH is set to about
5V PIN8
delay(1000);          //Set the delay time,
1000 = 1S
digitalWrite(8,LOW); //LOW is set to about
5V PIN8
delay(1000);
//delay(1000);
}
```

Output :

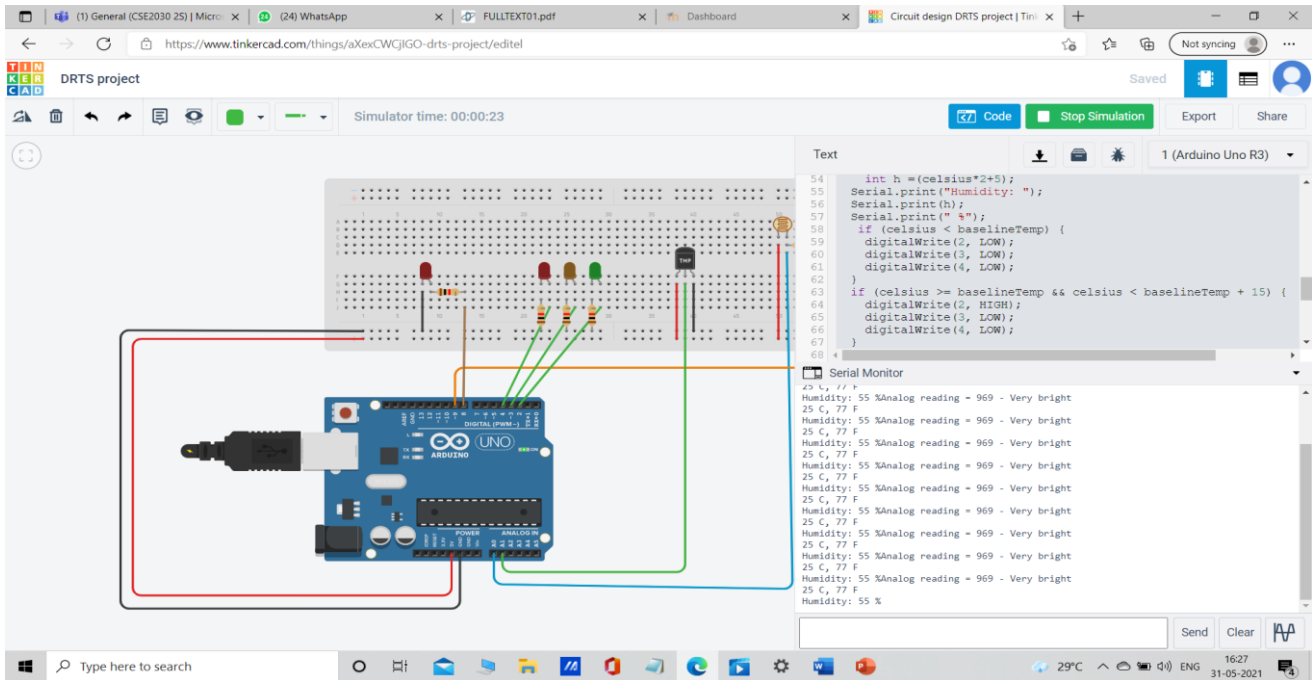


Fig. 5

Representation of Physical Demonstration :

The circuit of system :

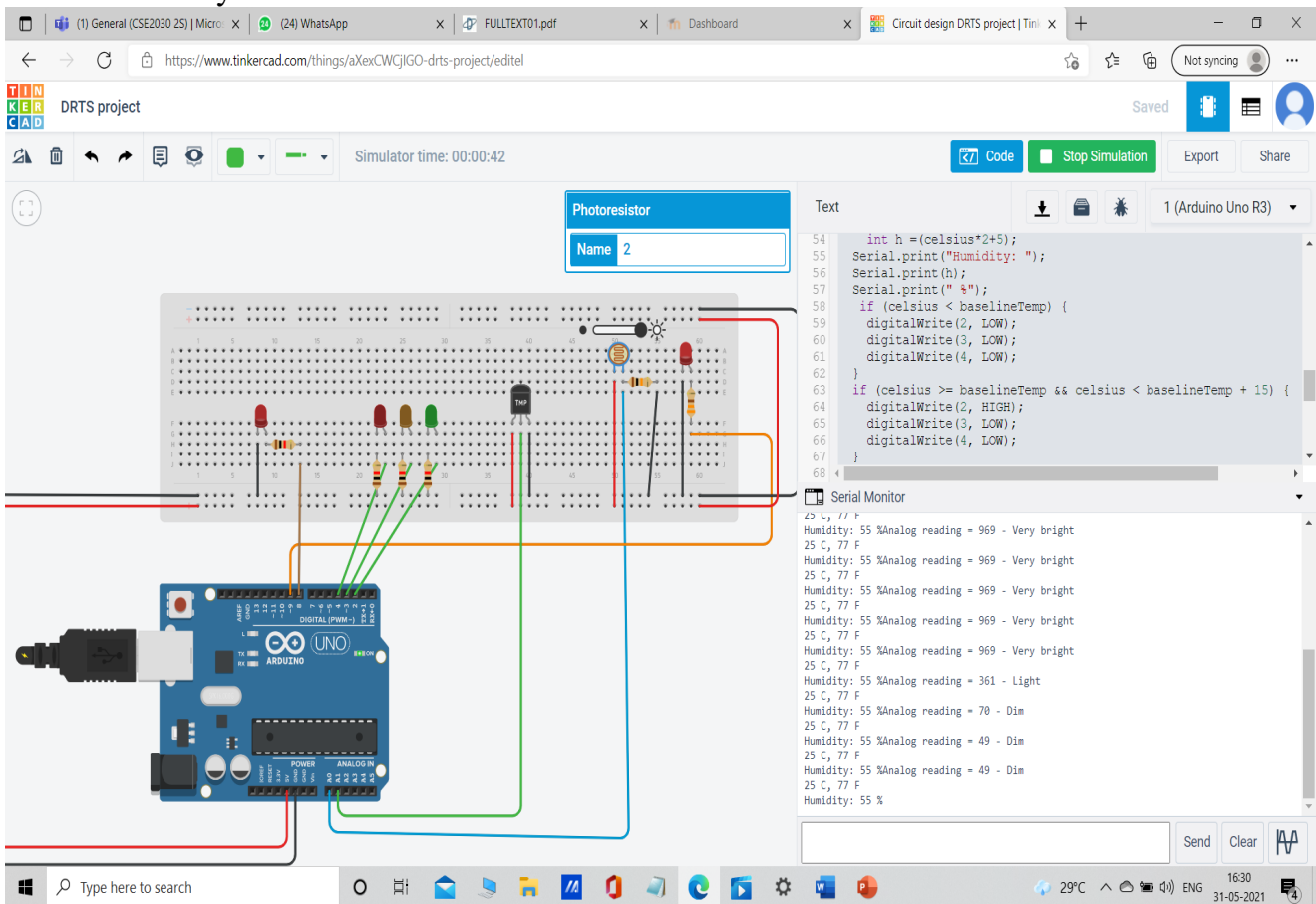


Fig. 6

Photosensor control loop :

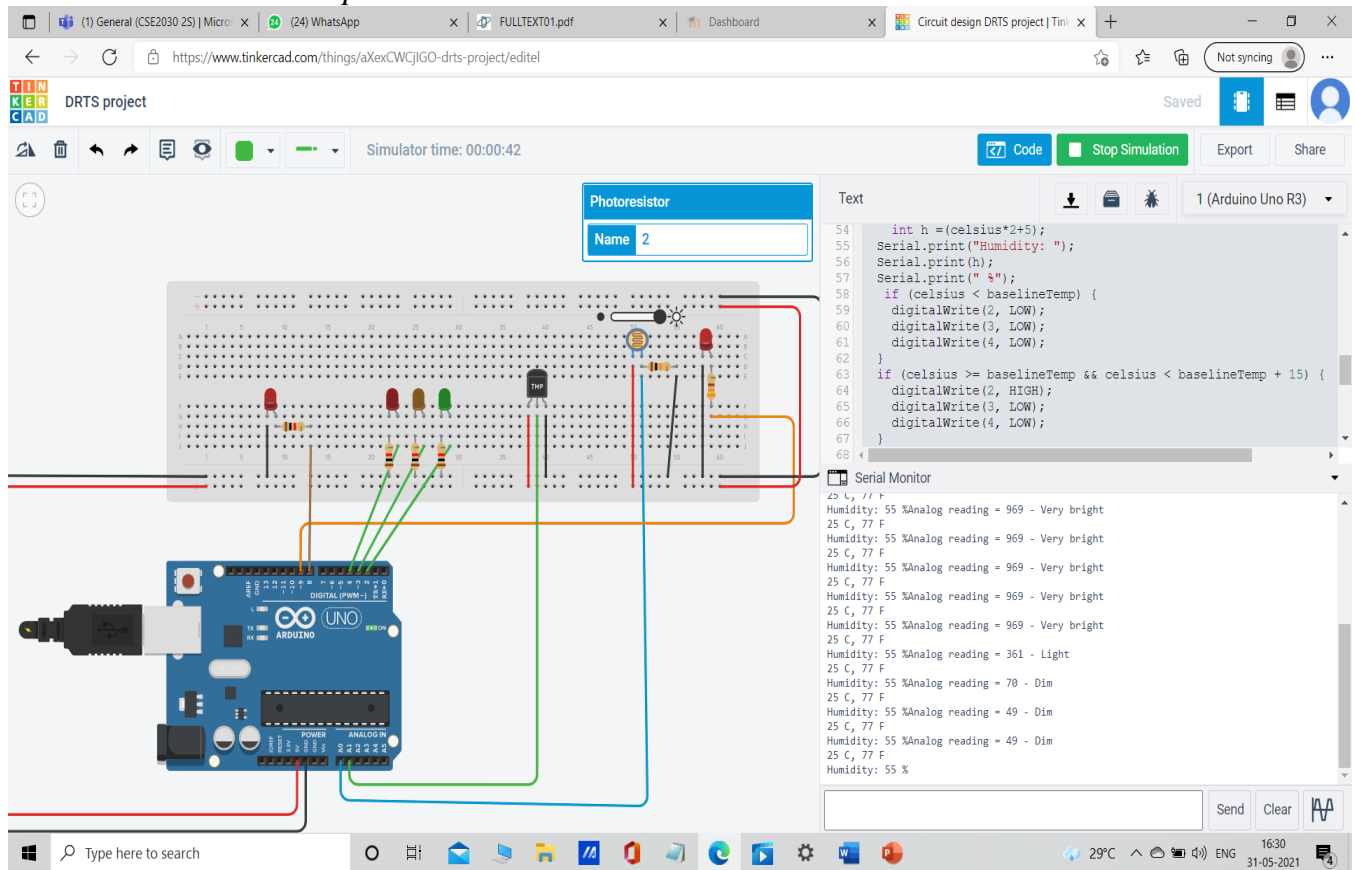


Fig. 7

Photoresistor signals task to execute (denoted by LED)

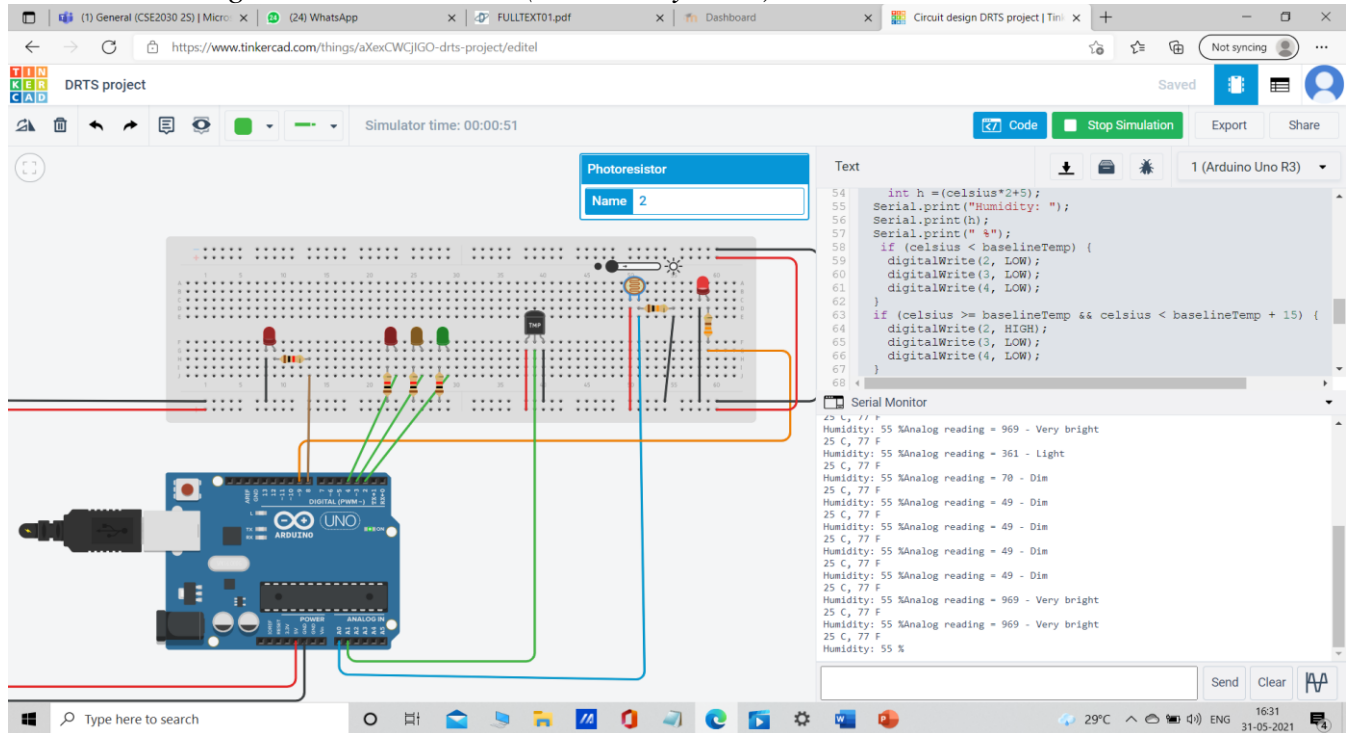


Fig. 8

Temperature control loop :

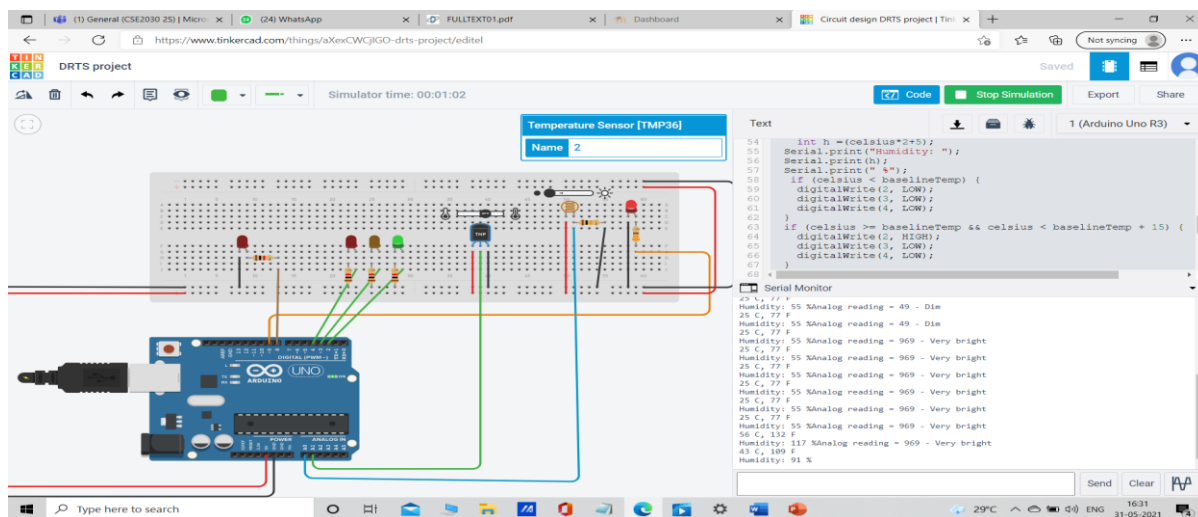


Fig. 9

The three LEDs indicate the speed of Fan. As the temperature increases the cooling system uses more energy by increasing the speed of the fan.

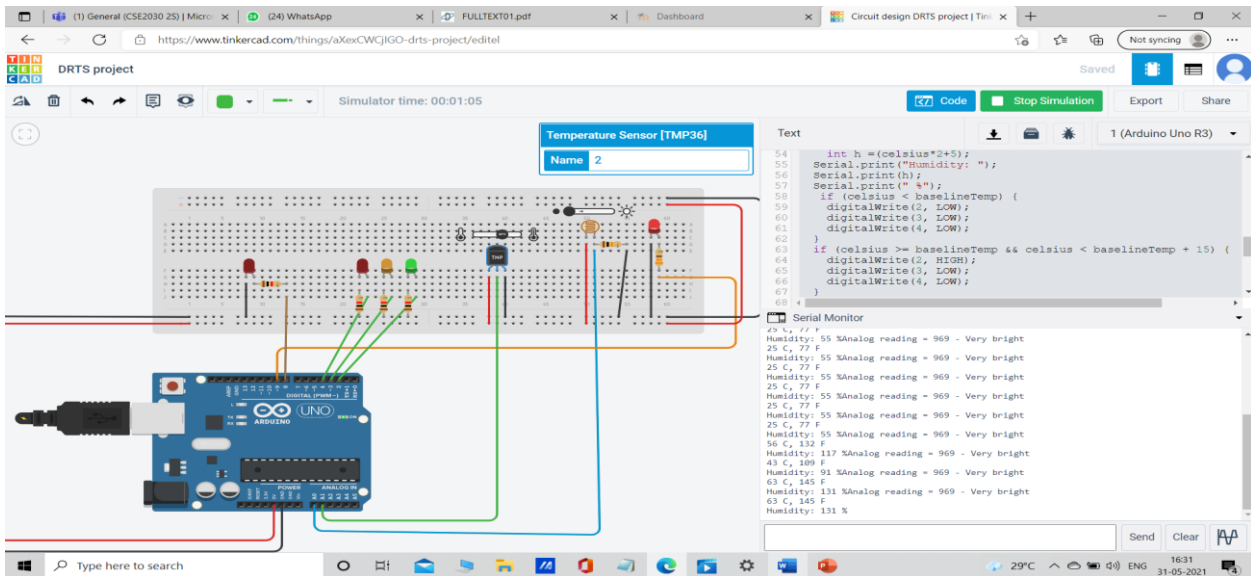


Fig. 10

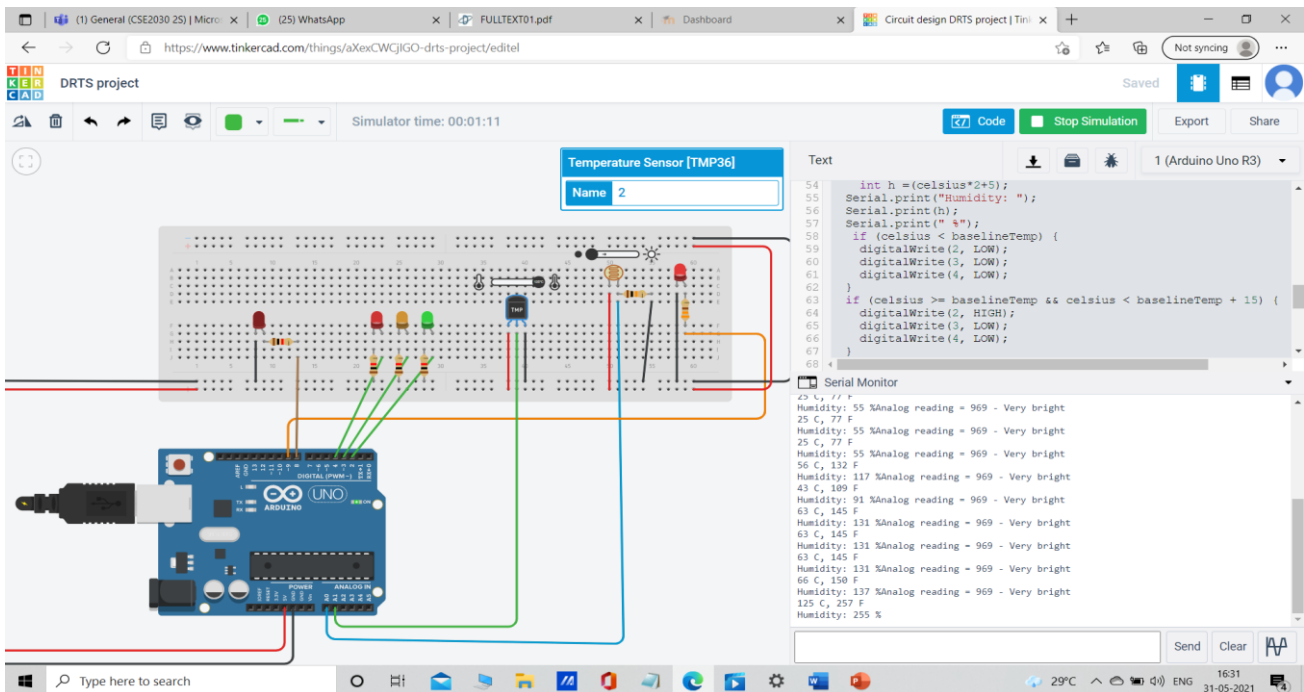


Fig. 11

Timed Growlight :

The timed Growlights are set on a timer based on the requirement of the plants.

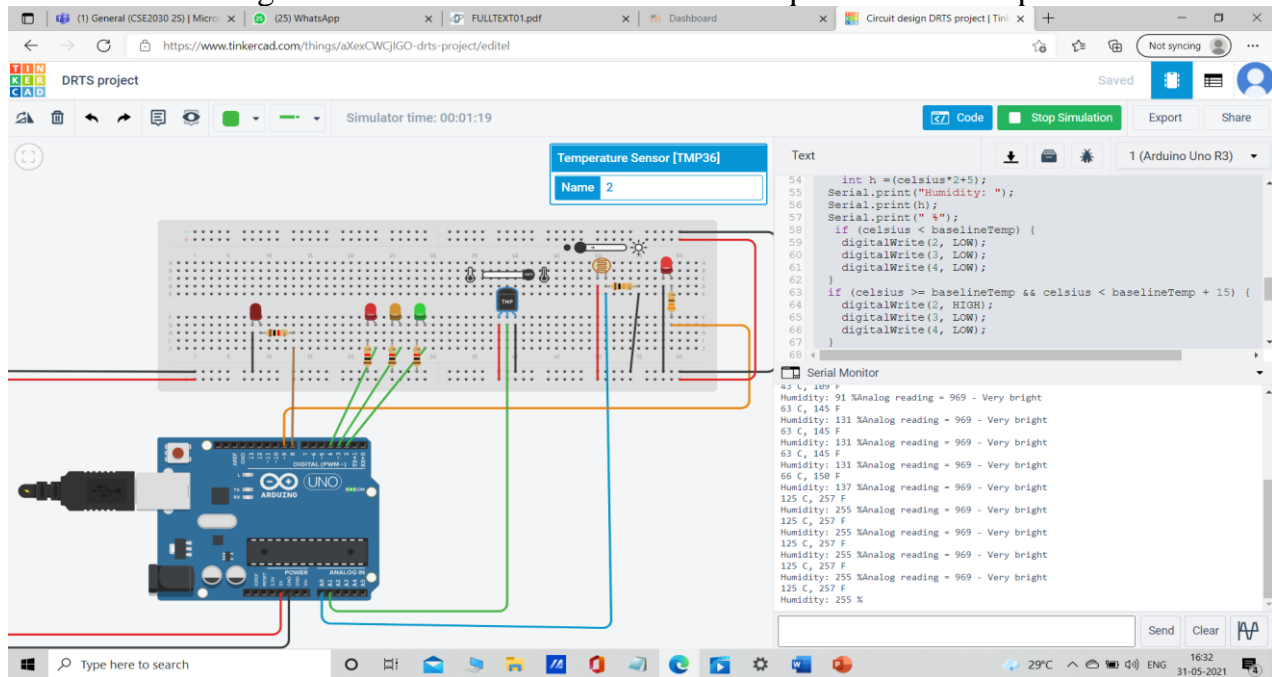


Fig.12

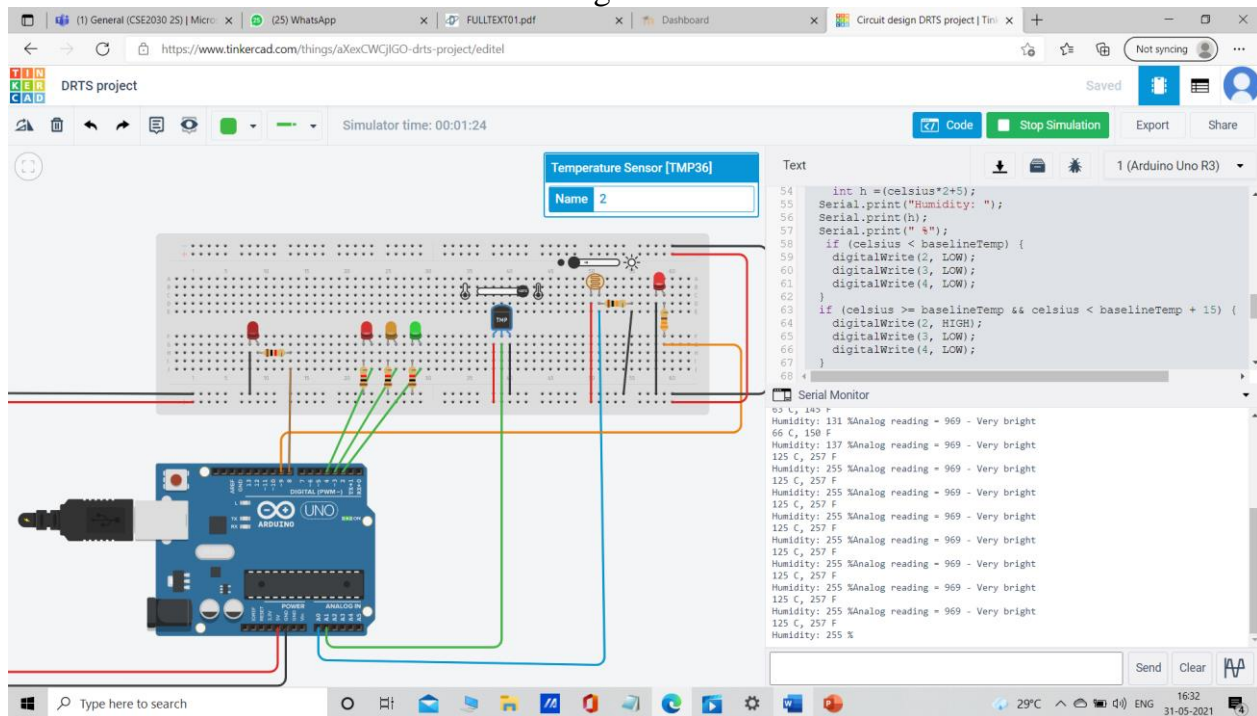


Fig. 13

AWS S2 code :

```
import boto3
import pandas
# Creating the low level functional client
client = boto3.client(
    's3',
```

```
aws_access_key_id =
'AKIAV5NIF2BJGDDF3BNS',
aws_secret_access_key =
'ZsXGzZr02s+/EV7DHdTtMmlaQusvDLgnV
e1lnXiw',
region_name = 'Asia Pacific (Mumbai) ap-
south-1'
```

```

)
# Creating the high level object oriented
interface
resource = boto3.resource(
    's3',
    aws_access_key_id =
'AKIAV5NIF2BJGDDDF3BNS',
    aws_secret_access_key =
'ZsXGzZr02s+/EV7DHdTtMmIaQusvDLgnV
e1lnXiw',
    region_name = 'Asia Pacific (Mumbai) ap-
south-1'
)
# Print the bucket names one by one

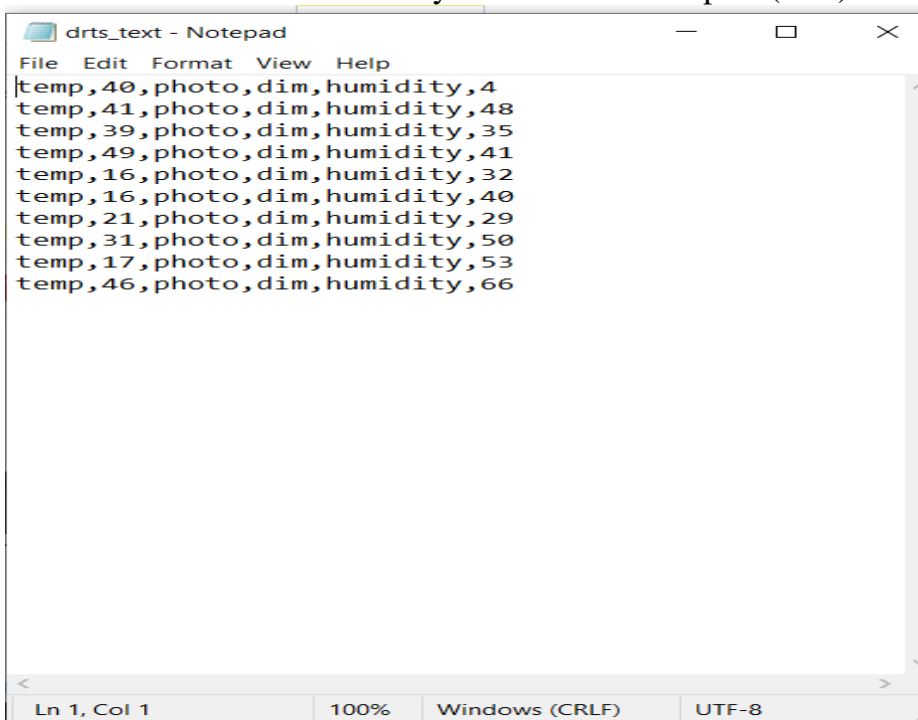
```

```

print('Printing bucket names...')
for bucket in clientResponse['Buckets']:
    print(f'Bucket Name: {bucket["Name"]}')
    # Create the S3 object
obj = client.get_object(
    Bucket = 'drts-proj1',
    Key = 'drts_proj.txt'
)
# Read data from the S3 object
data = pandas.read_txt(obj['Body'])

# Print the data frame
print('Printing the data frame...')
print(data)

```



```

File Edit Format View Help
temp,40,photo,dim,humidity,4
temp,41,photo,dim,humidity,48
temp,39,photo,dim,humidity,35
temp,49,photo,dim,humidity,41
temp,16,photo,dim,humidity,32
temp,16,photo,dim,humidity,40
temp,21,photo,dim,humidity,29
temp,31,photo,dim,humidity,50
temp,17,photo,dim,humidity,53
temp,46,photo,dim,humidity,66

```

Ln 1, Col 1 100% Windows (CRLF) UTF-8

Fig. 14

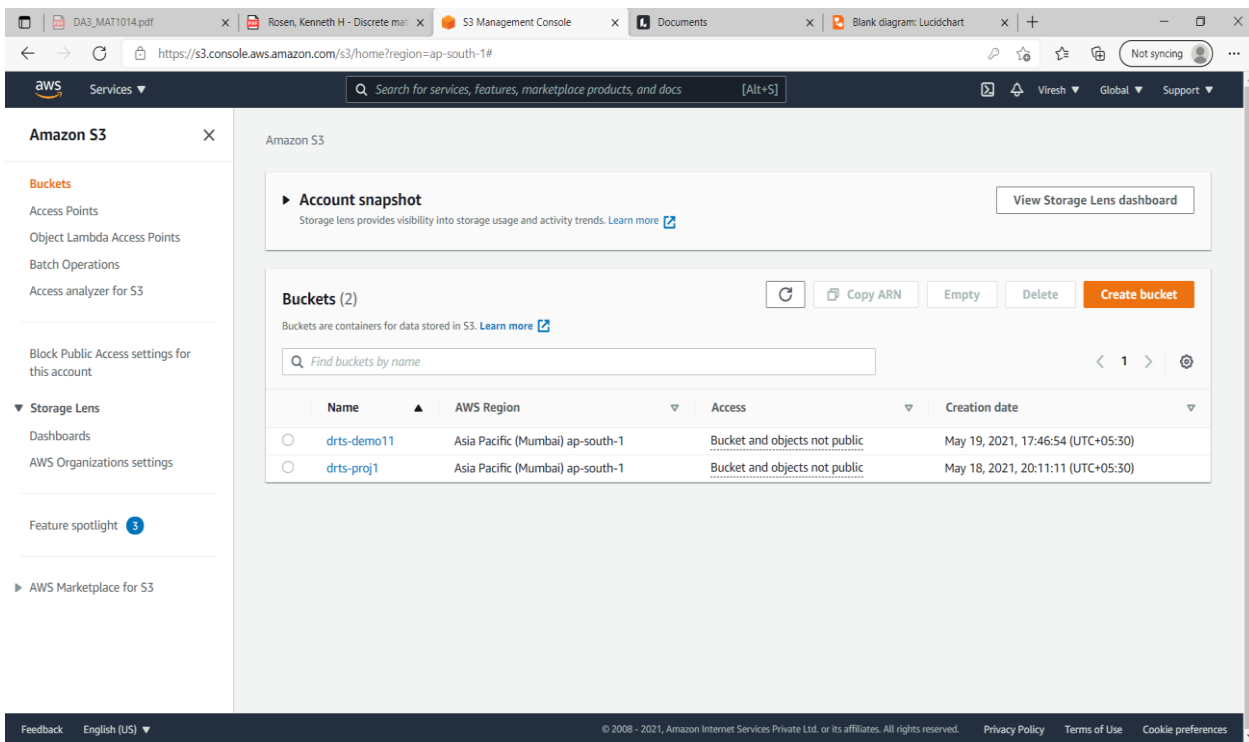


Fig. 15

AWS S2 service using python 3.9 :

The code calls the service and accesses the data file using python.

```
Suggestion [3,General]: The command drtsproj.py was not found, but does exist in the current location. Windows PowerShell does not load commands from the current location by default. If you trust this command, instead type: ".\drtsproj.py". See "get-help about_Command_Precedence" for more details.
PS C:\Users\Viresh> py drtsproj.py
Printing bucket names...
Bucket Name: drts-demo11
Bucket Name: drts-proj1
Printing the data frame...
temp,40,photo,dim,humidity,4
temp,41,photo,dim,humidity,48
temp,39,photo,dim,humidity,35
temp,49,photo,dim,humidity,41
temp,16,photo,dim,humidity,32
temp,16,photo,dim,humidity,40
temp,21,photo,dim,humidity,29
temp,31,photo,dim,humidity,50
temp,17,photo,dim,humidity,53
temp,46,photo,dim,humidity,66
PS C:\Users\Viresh>
```

Fig. 16

Hadoop

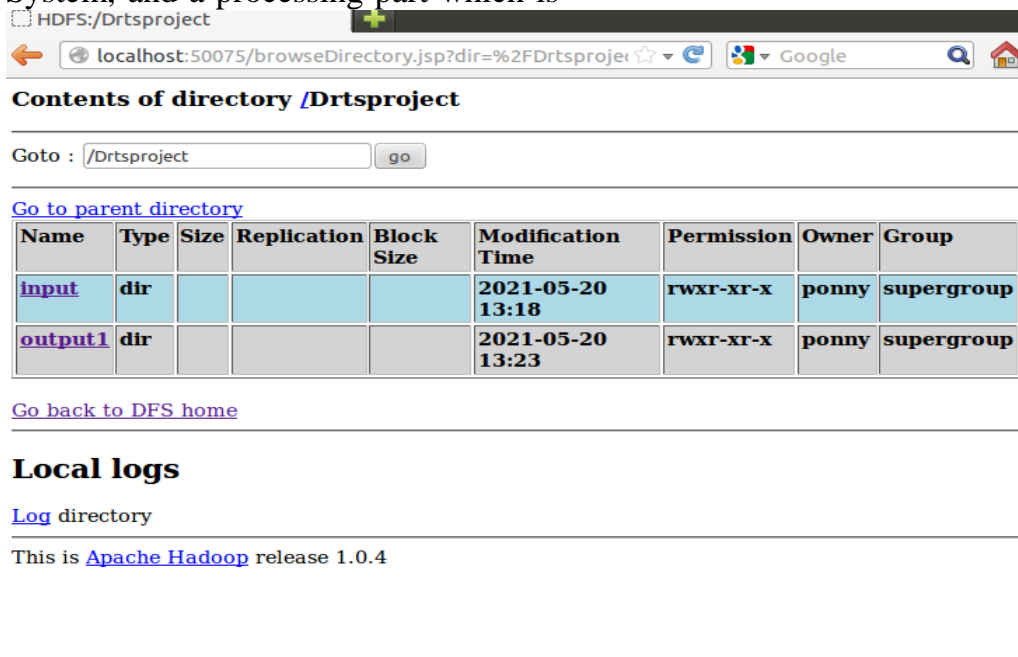
1. Introduction

Apache Hadoop is a collection of open-source software utilities which provides a software framework for distributed storage and processing of big data using the MapReduce programming model.

Hadoop consists of a storage part, known as Hadoop Distributed File System, and a processing part which is

a MapReduce programming model. Hadoop splits files into large blocks and distributes them across nodes in a cluster. It then transfers packaged code into nodes to process the data in parallel.

ScreenShot of HDFS:



2. Code:

```
import java.io.IOException;
import java.util.*;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import
org.apache.hadoop.mapreduce.lib.input.*;
import
org.apache.hadoop.mapreduce.lib.output.*;
```

```
public class Drts{
    public static class Map extends
Mapper<LongWritable, Text, Text, Text>
    {
        //private final static
IntWritable one = new IntWritable(1);
        IntWritable one = new
IntWritable(1);
        //private Text word = new
Text();
```

```

        public void
map(LongWritable key, Text value, Context
context) throws IOException,
InterruptedException {
        String[] line =
value.toString().split(",");

        String light =
line[3];

        context.write(new
Text(light), new Text(value.toString()));

    }
}

//light scope

count++;

}

public static class Reduce
extends Reducer<Text, Text, Text, Text> {
    int M = 0;

    Text wrd1 = new Text();

    IntWritable res1 = new
IntWritable();

    double averageHumidity
= 0d, averageTemp = 0d;

    int totalCount = 0;

    public void reduce(Text
key, Iterable<Text> values, Context context)
throws IOException, InterruptedException
    {
        int count = 0;

        for (Text val :
values) {
            input = val.toString().split(",");

            //global
            scope

            averageTemp+=
Integer.parseInt(input[1]);

            averageHumidity+=Integer.parseInt(in
put[5]);

            totalCount++;

            //light scope

            count++;

        }

        String output =
"The number of count is: "+ count;

        context.write(key,
new Text(output));

    }

    public void
cleanup(Context context)throws IOException,
InterruptedException
    {
        context.write(new
Text("Average of
Temperature:"+(averageTemp/totalCount)),n
ew Text("Average of
Humidity:"+(averageHumidity/totalCount)));

    }

}

```



```
public static void main(String[]
args) throws Exception {
```

```
Configuration conf = new
Configuration();
```

```
Job job = new Job(conf,
"Drts");
```

```
job.setJarByClass(Drts.class);
```

```
job.setOutputKeyClass(Text.class);
```

```
job.setOutputValueClass(Text.class);
```

```
job.setMapperClass(Map.class);
```

```
job.setReducerClass(Reduce.class);
```

```
job.setInputFormatClass(TextInputFor
mat.class);
```

```
job.setOutputFormatClass(TextOutput
Format.class);
```

```
FileInputFormat.addInputPath(job,
new Path(args[0]));
```

```
FileOutputFormat.setOutputPath(job,
new Path(args[1]));
```

```
job.waitForCompletion(true);
```

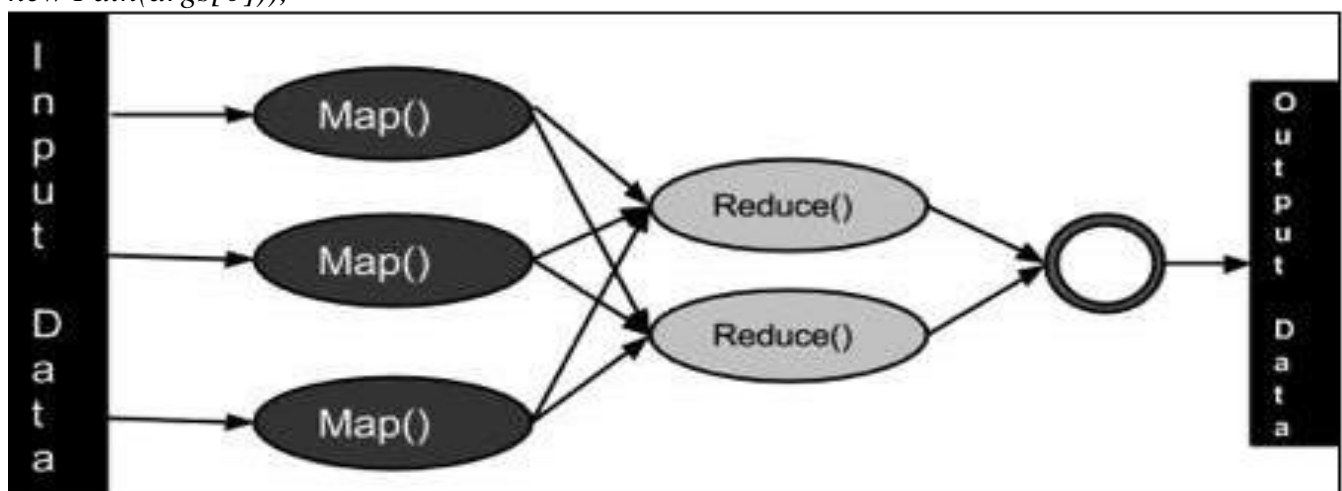
```
}
```

```
}
```

Working Of Mapper- Reducer :

1. **Map stage**— The map or mapper's job is to process the input data. Generally the input data is in the form of a file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.

2. **Reduce stage**— This stage is the combination of the **Shuffle** stage and the **Reduce** stage. The Reducer's job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.



	Input	Output
Map	<k1, v1>	list (<k2, v2>)
Reduce	<k2, list(v2)>	list (<k3, v3>)

Loading Mapper Reducer program in Hadoop

```
ponny@ubuntu:~$ start-all.sh
Warning: $HADOOP_HOME is deprecated.

starting namenode, logging to /home/ponny/hadoop/libexec/../logs/hadoop-ponny-namenode-ubuntu.out
localhost: starting datanode, logging to /home/ponny/hadoop/libexec/../logs/hadoop-ponny-datanode-ubuntu.out
localhost: starting secondarynamenode, logging to /home/ponny/hadoop/libexec/../logs/hadoop-ponny-secondarynamenode-ubuntu.out
starting jobtracker, logging to /home/ponny/hadoop/libexec/../logs/hadoop-ponny-jobtracker-ubuntu.out
localhost: starting tasktracker, logging to /home/ponny/hadoop/libexec/../logs/hadoop-ponny-tasktracker-ubuntu.out
```

```
ponny@ubuntu:~$ hadoop fs -mkdir /Drtsproject/input
Warning: $HADOOP_HOME is deprecated.
```

```
ponny@ubuntu:~$ hadoop fs -put /home/ponny/Desktop/drts_text.txt /Drtsproject/input
Warning: $HADOOP_HOME is deprecated.

ponny@ubuntu:~$ export HADOOP_CLASSPATH=$(hadoop classpath)
Warning: $HADOOP_HOME is deprecated.
```

Hadoop server is started on the local host and the input data file is uploaded to Hadoop File System.

```
ponny@ubuntu:~/Desktop/Drtsproject$ jar -cvf finaljar.jar -C /home/ponny/Desktop/Drtsproject/class_files/ .
added manifest
adding: Drts.class(in = 1420) (out= 717)(deflated 49%)
adding: Drts$Reduce.class(in = 2813) (out= 1185)(deflated 57%)
adding: Drts$Map.class(in = 1617) (out= 643)(deflated 60%)
ponny@ubuntu:~/Desktop/Drtsproject$ hadoop jar /home/ponny/Desktop/Drtsproject/finaljar.jar Drts /Drtsproject/input /Drtsproject/output1
Warning: $HADOOP_HOME is deprecated.
```

Java file is compiled and then all the files are compressed into a jar file.

```

21/05/20 13:22:54 WARN mapred.JobClient: Use GenericOptionsParser for parsing the
arguments. Applications should implement Tool for the same.
21/05/20 13:22:55 INFO input.FileInputFormat: Total input paths to process : 1
21/05/20 13:22:55 INFO util.NativeCodeLoader: Loaded the native-hadoop library
21/05/20 13:22:55 WARN snappy.LoadSnappy: Snappy native library not loaded
21/05/20 13:22:56 INFO mapred.JobClient: Running job: job_202105201315_0001
21/05/20 13:22:57 INFO mapred.JobClient: map 0% reduce 0%
21/05/20 13:23:13 INFO mapred.JobClient: map 100% reduce 0%
21/05/20 13:23:25 INFO mapred.JobClient: map 100% reduce 100%
21/05/20 13:23:30 INFO mapred.JobClient: Job complete: job_202105201315_0001
21/05/20 13:23:30 INFO mapred.JobClient: Counters: 29
21/05/20 13:23:30 INFO mapred.JobClient:   Job Counters
21/05/20 13:23:30 INFO mapred.JobClient:     Launched reduce tasks=1
21/05/20 13:23:30 INFO mapred.JobClient:     SLOTS_MILLIS_MAPS=14073
21/05/20 13:23:30 INFO mapred.JobClient:     Total time spent by all reduces waitin
ting after reserving slots (ms)=0
21/05/20 13:23:30 INFO mapred.JobClient:     Total time spent by all maps waitin
g after reserving slots (ms)=0
21/05/20 13:23:30 INFO mapred.JobClient:     Launched map tasks=1

```

Input:

A text file consisting of daily reports of temperature , humidity and light status (bright,dim,dark). This text file contains 1000 entries , generated randomly (in our project) using python program:

```

import random
import os
import string

```

```

file1 = open("drts_text.txt","w+")
temp = random.choices(range(50), k=1000)

```

```

photo=random.choices(["dim",
"dark","bright"], weights=[1,1,1.5]
,k=1000)
humidity = random.choices(range(70),
k=1000)

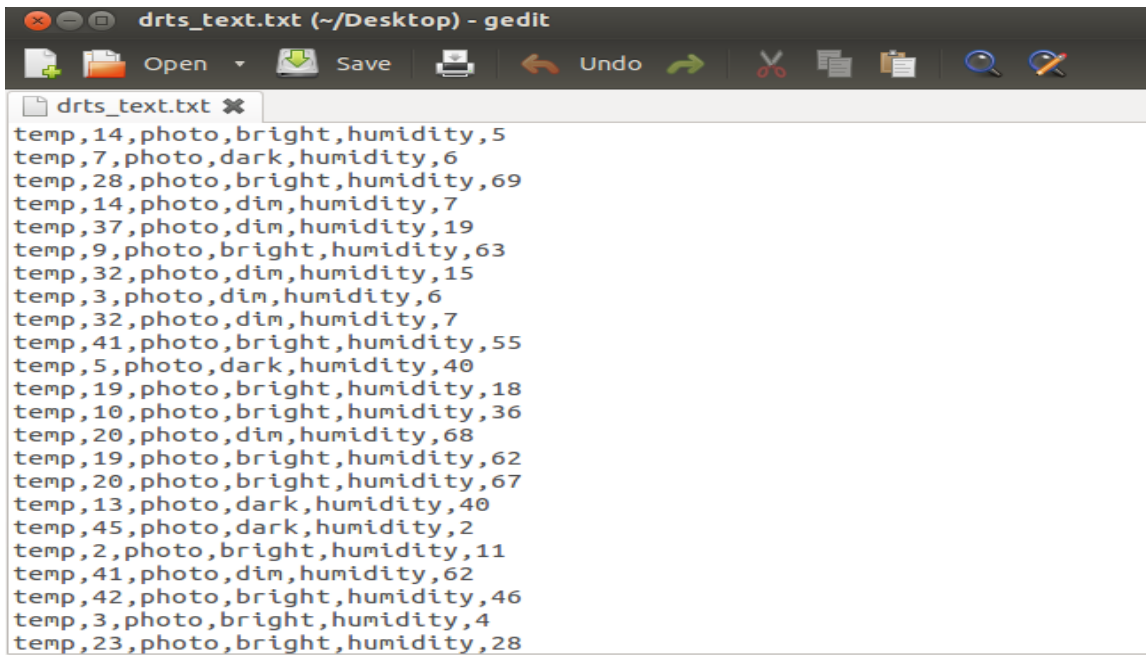
```

```

for i in range(1000):
    file1.write("temp,")
    file1.write(str(temp[i]))
    file1.write(",photo,")
    file1.write(photo[i])
    file1.write(",humidity,")
    file1.write(str(humidity[i]))
    file1.write("\n")

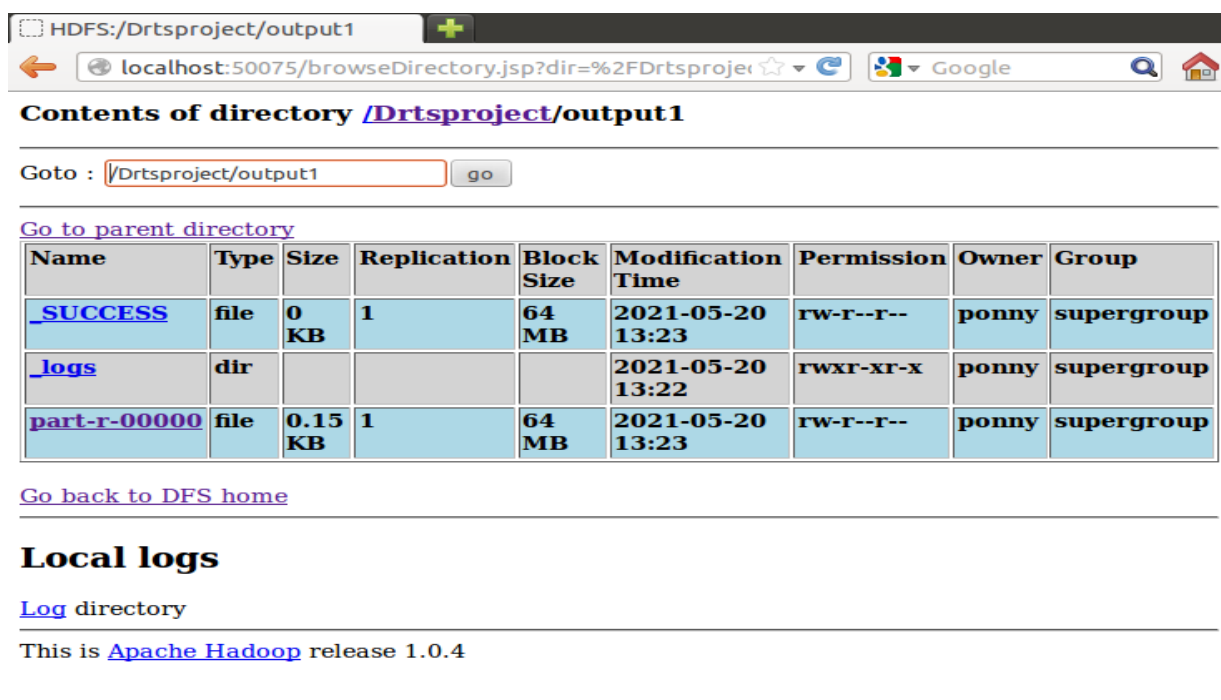
```

Sample Screenshot of File generated :



```
drts_text.txt
temp,14,photo,bright,humidity,5
temp,7,photo,dark,humidity,6
temp,28,photo,bright,humidity,69
temp,14,photo,dim,humidity,7
temp,37,photo,dim,humidity,19
temp,9,photo,bright,humidity,63
temp,32,photo,dim,humidity,15
temp,3,photo,dim,humidity,6
temp,32,photo,dim,humidity,7
temp,41,photo,bright,humidity,55
temp,5,photo,dark,humidity,40
temp,19,photo,bright,humidity,18
temp,10,photo,bright,humidity,36
temp,20,photo,dim,humidity,68
temp,19,photo,bright,humidity,62
temp,20,photo,bright,humidity,67
temp,13,photo,dark,humidity,40
temp,45,photo,dark,humidity,2
temp,2,photo,bright,humidity,11
temp,41,photo,dim,humidity,62
temp,42,photo,bright,humidity,46
temp,3,photo,bright,humidity,4
temp,23,photo,bright,humidity,28
```

Output :



HDFS:/Drtsproject/output1

localhost:50075/browseDirectory.jsp?dir=%2FDrtsproject/output1

Contents of directory /Drtsproject/output1

Goto : go

[Go to parent directory](#)

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
_SUCCESS	file	0 KB	1	64 MB	2021-05-20 13:23	rw-r--r--	ponny	supergroup
_logs	dir				2021-05-20 13:22	rwxr-xr-x	ponny	supergroup
part-r-00000	file	0.15 KB	1	64 MB	2021-05-20 13:23	rw-r--r--	ponny	supergroup

[Go back to DFS home](#)

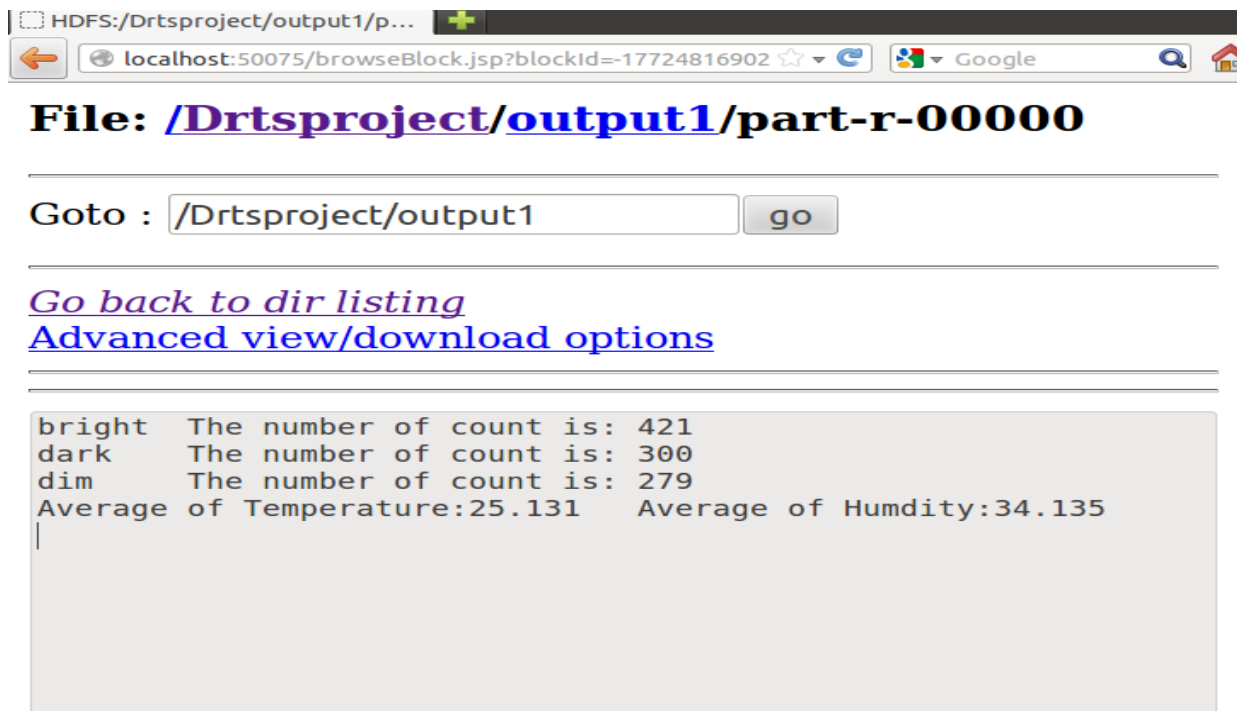
Local logs

[Log](#) directory

This is [Apache Hadoop](#) release 1.0.4

The output shows following entries:

- Number of bright days out of 1000 days
- Number of dark days out of 1000 days
- Number of dim days out of 1000 days
- Average temperature of 1000 days
- Average Humidity of 1000 days



CONCLUSION

The developed cost effective greenhouse model can be used to monitor and control temperature, light intensity, humidity and soil moisture of a greenhouse in order to increase productivity in farming especially in countries like India where there is ample risk of insect infestation, harsh climate and increasing demand of food with the decrease of fertile land. The model is fully automatic and so, does not require human interaction to smoothly monitor the plant. Moreover, the efficiency and accuracy of the system is more accurate than manual systems. Humans can observe whether the soil is wet or not but the proposed system is able to measure the actual amount of moisture that is present in the soil. Again, it is very tough for humans to measure actual light intensity, temperature and humidity while this proposed system can do them all very accurately. It eliminates the risk of

human errors to maintain a greenhouse at a specific environmental condition. Moreover, it is also eco-friendly. With the addition of wireless technologies between the sensors the system can be more reliable, cost effective for larger area and easy to implement.

ACKNOWLEDGMENT

The authors sincerely thank Prof. Harini S for her impeccable guidance and expert support throughout the project. The authors would like to thank VIT Chennai for providing opportunity to work on this project and give us a platform to expand our knowledge and vision

REFERENCES

- [1] M. N. Hassan, A. S. Noor and S. I. Abdullah, "An automatic monitoring and control system inside greenhouse", Green Energy and Technology, 2015.
- [2] J. Xiao, B. Jiang and K. J. Ming, "Design for wireless temperature and humidity

monitoring system of the intelligent greenhouse”, International Conference on Computer Engineering and Technology, Volume 3, pp. 59-63, 2010.

- [3] M. H. Khan and M. M. Alom, “Greenhouse effect in Bangladesh-Environmental rules and regulations perspective”, Journal of Multidisciplinary Engineering Science and Technology, Volume 2, Issue 1, pp. 283-285, January-2015
- [4] L. Liu and Y. Zhang, “Design of greenhouse environment monitoring system based on wireless sensor network”, 3rd International Conference on Control, Automation and Robotics, pp. 463-466, 2017.
- [5] K. Rangan and T. Vigneswaran, “An embedded systems approach to monitor greenhouse”, Recent Advances in Space Technology Services and Climate Change, Pages: 61-65, 2010.
- [6] Sensors Magazine. (2004). Editorial: This changes everything—market observers quantify the rapid escalation of wireless sensing and explain its effects. *Wireless for Industry, Supplement to Sensors Magazine, Summer*, pp. S6–S8.
- [7] Alves-Serodio, C. M. J., Monteiro, J. L., & Couto, C. A. C. (1998). An integrated network for agricultural management applications. *IEEE International Symposium on Electromagnetic Compatibility. America: Denver*, pp. 679–683.