

# Home Gardening using IoT

Internet of Things

EVD520

B. Tech ICT

Semester – 7

Group – 8

Group Members

Falgun Prajapati – 1401080

Kedar Acharya – 1401081

Shloak Agarwal – 1401105

## **Introduction & Motivation**

By the year 2050, nearly 80% of the Earth's population will live in urban centres. Applying the most conservative estimates to current demographic trends, the human population will increase by about three billion people by then. An estimated 109 hectares of new land (about 20% larger than Brazil) will be needed to grow enough food to feed them, if traditional farming methods continue as they are practised today. At present, throughout the world, over 80% of the land that is suitable for raising crops is in use. Historically, some 15% of that has been laid waste by poor management practices. What can be done to ensure enough food for the world's population to live on?

The concept of indoor farming is not new, since hothouse production of tomatoes and other produce has been in vogue for some time. What is new is the urgent need to scale up this technology to accommodate another three billion people. Many believe an entirely new approach to indoor farming is required, employing cutting-edge technologies. One such proposal is for the "Vertical Farm". The concept is of multi-storey buildings in which food crops are grown in environmentally controlled conditions. Situated in the heart of urban centres, they would drastically reduce the amount of transportation required to bring food to consumers. Vertical farms would need to be efficient, cheap to construct and safe to operate. If successfully implemented, proponents claim, vertical farms offer the promise of urban renewal, sustainable production of a safe and varied food supply (through year-round production of all crops), and the eventual repair of ecosystems that have been sacrificed for horizontal farming.

## **Literature Review**

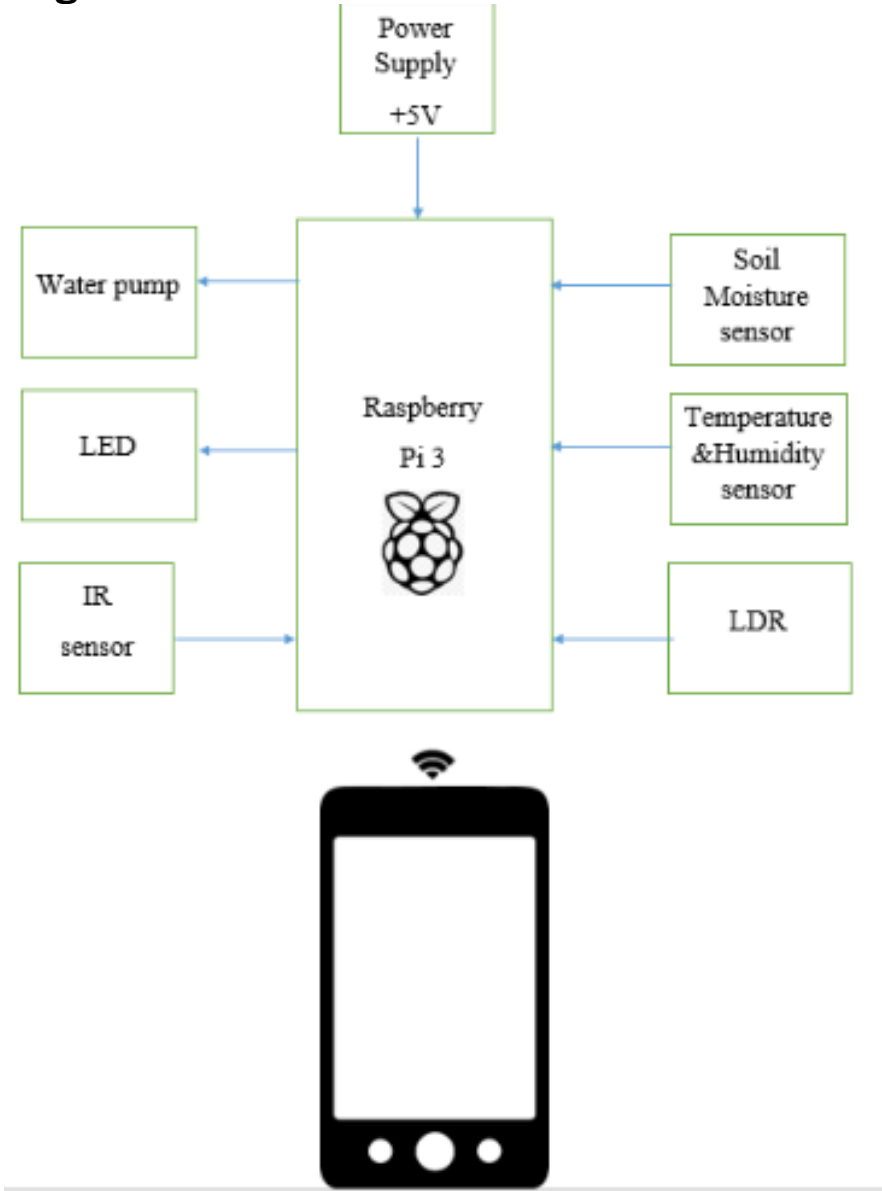
As of now, there are no product available which is similar to our framework in the market. Though framework for the same has already been published in paper Agri-IoT: A Semantic Framework for Internet of Things-enabled Smart Farming Applications (by Andreas Kamlaris, Feng Gao, Francesc X. Prenafeta-Boldu and Muhammad Intisar Ali, GIRO Joint Research Unit IRTA-UPC, Barcelona, Spain).

In the paper, they have described Agri-IoT, an IoT-based framework applying real-time stream processing, analysis and reasoning in the domain of agriculture, based on semantic web technologies, facilitating more informed and accurate decision making by farmers and event detection. They have investigated the introduction of IoT in smart farming and its opportunities, through the seamless combination of heterogeneous technologies, as well as the semantic integration of information from various sources (sensors, social media, connected farms, governmental alerts, regulations etc.), ensuring increase of production and productivity, better products 'quality, protection of the environment, less use of resources (e.g. water, fertilizers), faster reaction to unpredictable events and more transparency to the consumer. Agri-IoT achieves this by offering interoperability between sensors, processes, data streams, farms as entities and web-based services, exploiting open data, making use of semantic technologies and linked web data. Their evaluation efforts focusing on two realistic and demanding farming scenarios indicate the good performance of the proposed framework in medium-to-large farms, while their discussion reveals the large opportunities arising in farming by introducing open standards and semantics based on IoT.

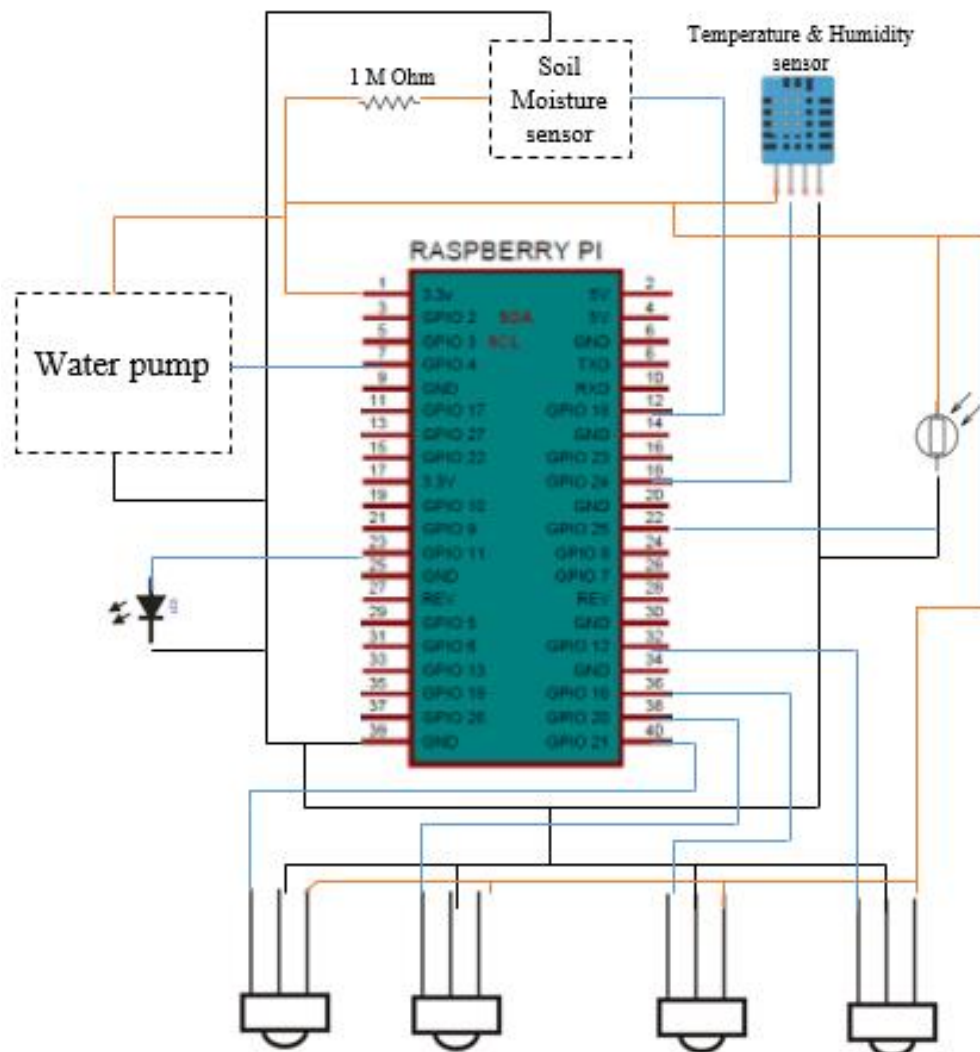
What they have proposed above is a framework that automates the irrigation process in farming thus by helping farmers for their farming. Problem with current process is farmer always has to stay on the farm for keeping an eye on water supply to each and every plant which no longer be any problem as this IOT based farming framework takes care of that just with the help of handful of sensors which detects humidity and temperature of surrounding atmosphere, water level of tank, and moisture of soil and then calculates the right amount of water to be supplied to the whole farm.

What we are planning in our project is implementing the same framework for a bit conservative system (i.e. house farming) in which our system will work solely for a particular plant as each plant's requirements of water supply are different and thus the system will also include a mobile application (or a webpage) that will keep notifying as well as updating the owner about water left in tank, modes through which he/she can control water in water tank.

**Block Diagram**



## Circuit Diagram







## List of Sensors/Actuators: -

Sr. No.	Sensor	Total no.	Detected Matter	Selection Criteria
1	Infrared	4	Obstacle	To detect the growth of the plant
2	DHT11	1	Humidity, temperature	To check the atmospheric conditions around the plant
3	KG003	1	Soil Moisture	To decide whether plant
4	LDR	1	Sunlight	To decide whether to turn on the lamp or not

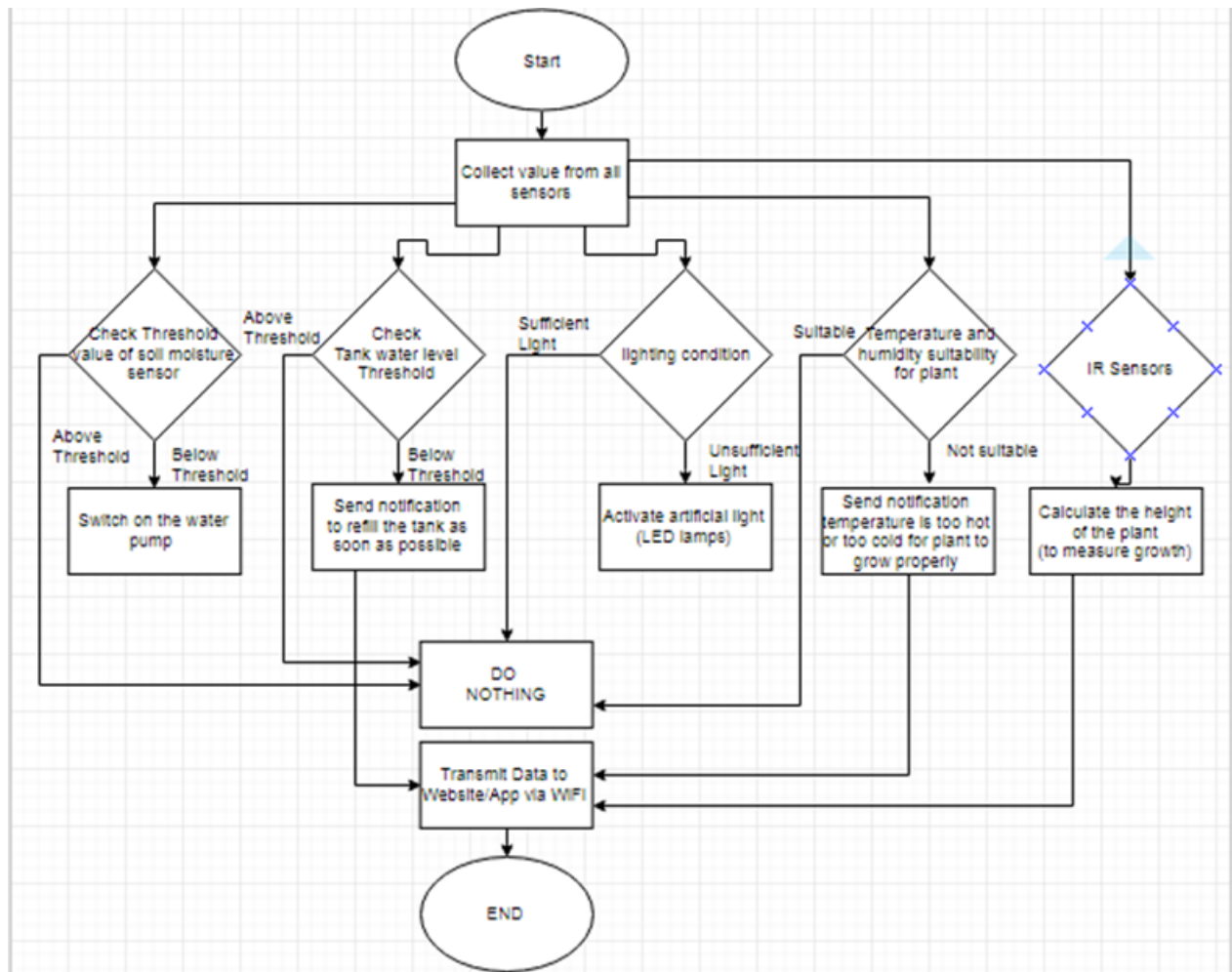
## Raspberry Pi Features

- If your projects need network, Beagle Bone Black (BBB) and Raspberry PI (RPI) are relatively suitable. In terms of Ethernet functionality, BBB is stronger than RPI because it supports internally by the processor AM3358, but the Ethernet from RPI is converted by extended USB-to-Ethernet chip set. What's more, the USB interface on RPI and BBB can be connected with a wireless device like WiFi modules to realize the network function also RPI comes with built WiFi and Bluetooth.
- In terms of User Interface, if your projects require HD Video output, RPI will be better as Raspberry pi has video decoding and encoding capabilities and its HDMI resolution is up to 1280\*1024, but BBB can only support to 720P, not to consider Edison and Link It ONE. There have been various inches of LCD module and display screen for BBB and RPI, which you can easily build a mini ARM Computer with screen.
- In terms of performance of Processor, BBB has a faster processor than RPI and Link It ONE, but all of them are single ARM core, so if your projects need higher data processing capability, then consider BBB first and RPI next.
- In terms of the whole open hardware ecosystem, RPI and BBB are well-known and there are thousands of PI module and CAPES, projects, prototypes around them. You can download almost any resources for reference when you want to build one prototype, same as Arduino. However, Link It ONE and Edison are younger, they still need time to build up their own world.

	Arduino Yun	Beaglebone Black	Intel Galileo	Raspberry Pi
Picture				
SoC	Atheros AR9331	Texas Instruments AM3358	Intel Quark X1000	Broadcom BCM2835
CPU	MIPS32 24K and ATmega32U4	ARM Cortex-A8	Intel X1000	ARM1176
Architecture	MIPS and AVR	ARMv7	i586	ARMv6
Speed	400mhz (AR9331) and 16mhz (ATmega)	1ghz	400mhz	700mhz

<b>Memory</b>	64MB (AR9331) and 2.5KB (ATmega)	512MB	256MB	256MB (model A) or 512MB (model B)
<b>FPU</b>	None (Software)	Hardware	Hardware	Hardware
<b>GPU</b>	None	PowerVR SGX530	None	Broadcom VideoCore IV
<b>Internal Storage</b>	16MB (AR9331) and 32KB (ATmega)	2GB (rev B) or 4GB (rev C)	8MB	None
<b>External Storage</b>	MicroSD (AR9331)	MicroSD	MicroSD	SD card
<b>Networking</b>	10/100Mbit ethernet and 802.11b/g/n WiFi	10/100Mbit ethernet	10/100Mbit ethernet	None (model A) or 10/100Mbit ethernet (model B)
<b>Power Source</b>	5V from USB micro B connector, or header pin.	5V from USB mini B connector, 2.1mm jack, or header pin.	5V from 2.1mm jack, or header pin.	5V from USB micro B connector, or header pin.
<b>Dimensions</b>	2.7in x 2.1in (68.6mm x 53.3mm)	3.4in x 2.1in (86.4mm x 53.3mm)	4.2in x 2.8in (106.7mm x 71.1mm)	3.4in x 2.2in (85.6mm x 56mm)
<b>Weight</b>	1.4oz (41g)	1.4oz (40g)	1.8oz (50g)	1.6oz (45g)
<b>Approximate Price</b>	\$75	\$55 (rev C), \$45 (rev B)	\$80	\$25 (model A), \$35 (model B)

## Flow Chart of Program



## **Sensors**

### **1. Soil Moisture Sensor – KG003**

#### Working

Soil moisture sensors measure the water content in soil. A soil moisture probe is made up of multiple soil moisture sensors. One common type of soil moisture sensors in commercial use is a Frequency domain sensor such as a capacitance sensor. Another sensor, the neutron moisture gauge, utilize the moderator properties of water for neutrons. Soil moisture content may be determined via its effect on dielectric constant by measuring the capacitance between two electrodes implanted in the soil. Where soil moisture is predominantly in the form of free water (e.g., in sandy soils), the dielectric constant is directly proportional to the moisture content. The probe is normally given a frequency excitation to permit measurement of the dielectric constant. The readout from the probe is not linear with water content and is influenced by soil type and soil temperature. Therefore, careful calibration is required and long-term stability of the calibration is questionable.

- In This Sensor We are using 2 Probes to be dipped into the Soil
- As per Moisture We will get Analog Output variations from 0.60volts - 5volts
- Input Voltage 5V DC

Sensing Probe Dimensions: 60x30mm

Panel PCB Dimensions: 30x60mm

Operating Voltage: 3.3V ~ 5V

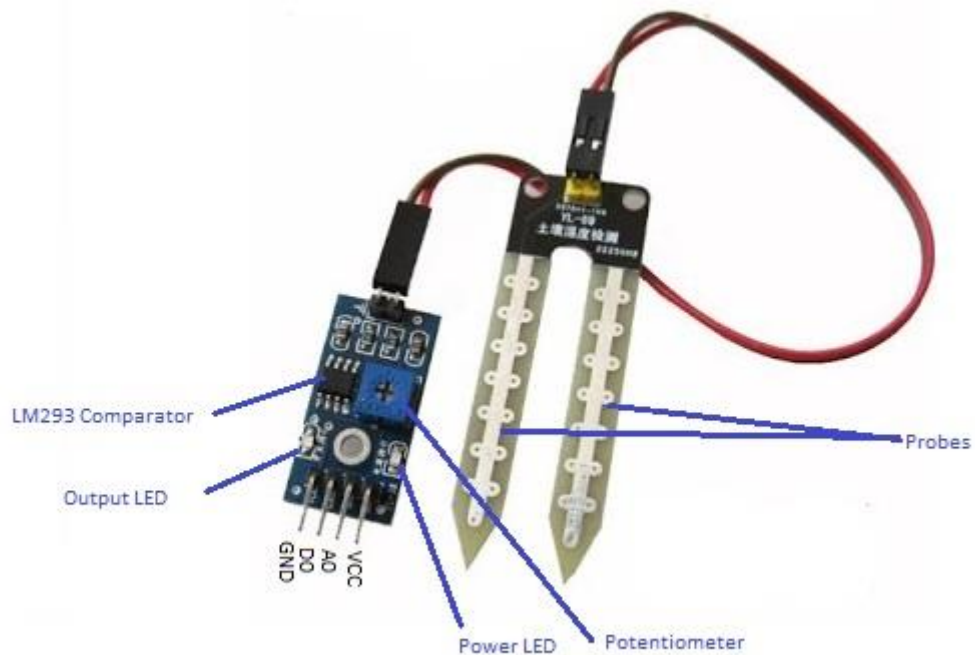
Digital Outputs

On-board LM393 comparator

On-board power indicator LED

On-board digital switching indicator LED



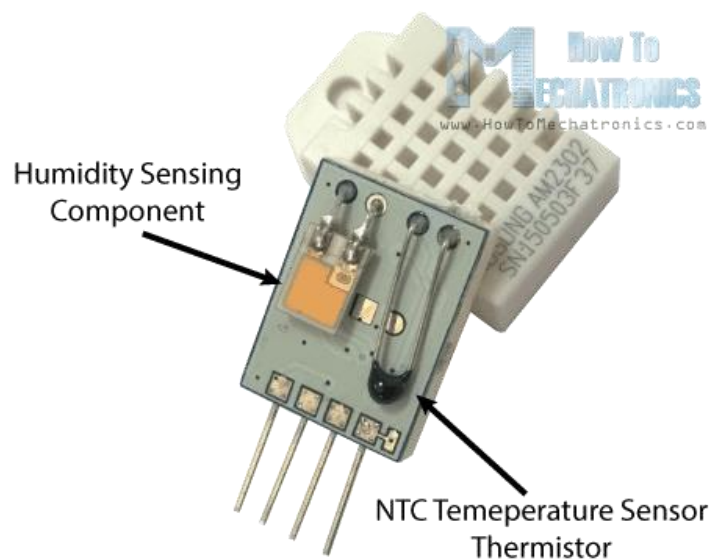


Type of interface with raspberry pi: Indirect

## 2. Temperature and humidity module - DHT11

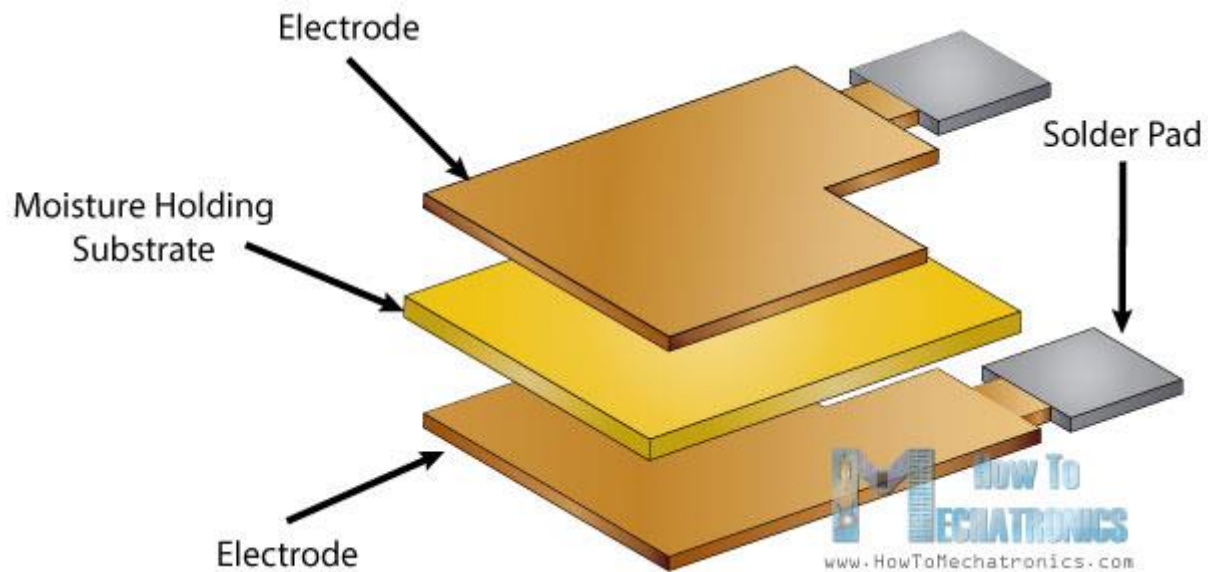
### Working:

Dht-11 consist of a humidity sensing component, a NTC temperature sensor (or thermistor) and an IC on the back side of the sensor.

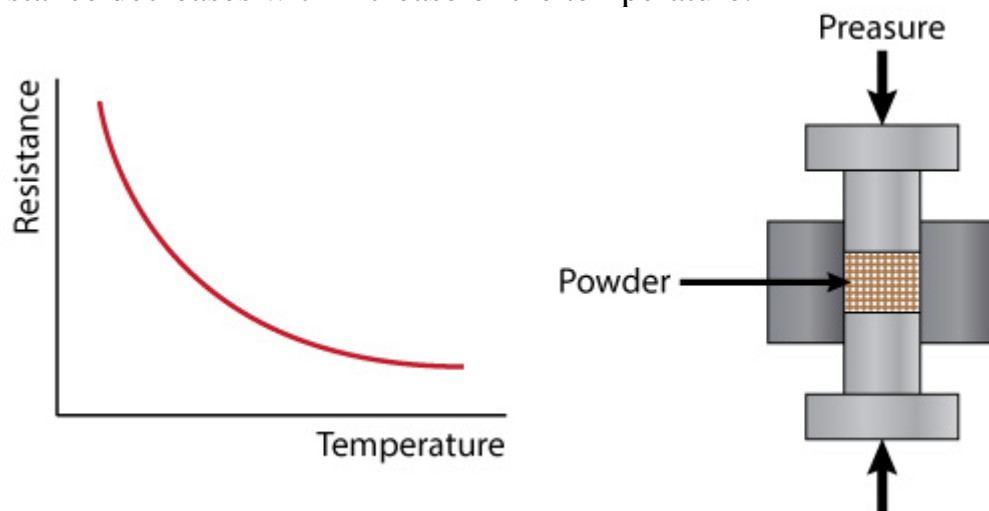


For measuring humidity, they use the humidity sensing component which has two electrodes with moisture holding substrate between them. So as the humidity changes, the conductivity of the substrate changes or the resistance between these

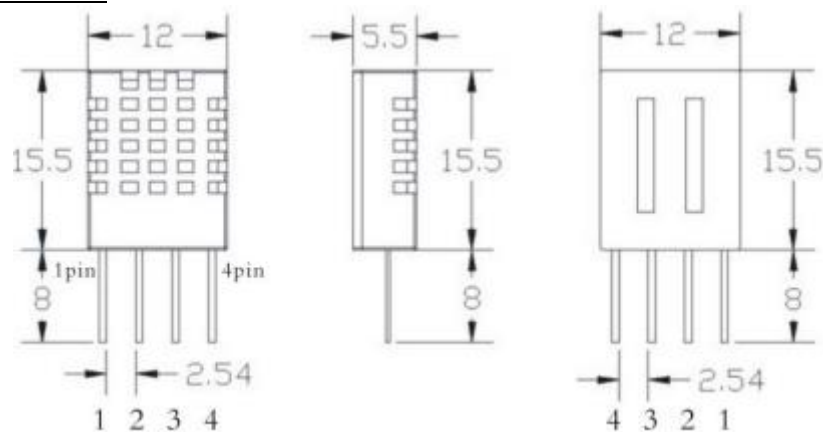
electrodes changes. This change in resistance is measured and processed by the IC which makes it ready to be read by a microcontroller.



On the other hand, for measuring temperature these sensors use a NTC temperature sensor or a thermistor. A thermistor is actually a variable resistor that changes its resistance with change of the temperature. These sensors are made by sintering of semi-conductive materials such as ceramics or polymers in order to provide larger changes in the resistance with just small changes in temperature. The term “NTC” means “Negative Temperature Coefficient”, which means that the resistance decreases with increase of the temperature.



Dimension: In mm



### Relative humidity

Resolution: 16Bit

Repeatability:  $\pm 1\%$  RH

Accuracy: At 25°C ±5% RH

Interchangeability: fully interchangeable

Response time: 1 / e (63%) of 25°C 6s 1m / s air 6s

## Temperature

Resolution: 16Bit

Repeatability:  $\pm 0.2^{\circ}\text{C}$ 

Range: At 25°C ±2°C

Response time:  $1 / e$  (63%) 10S

## Electrical Characteristics

---

Power supply: DC 3.5~5.5V

Supply Current: measurement 0.3mA standby 60μ A

Sampling period: more than 2 seconds

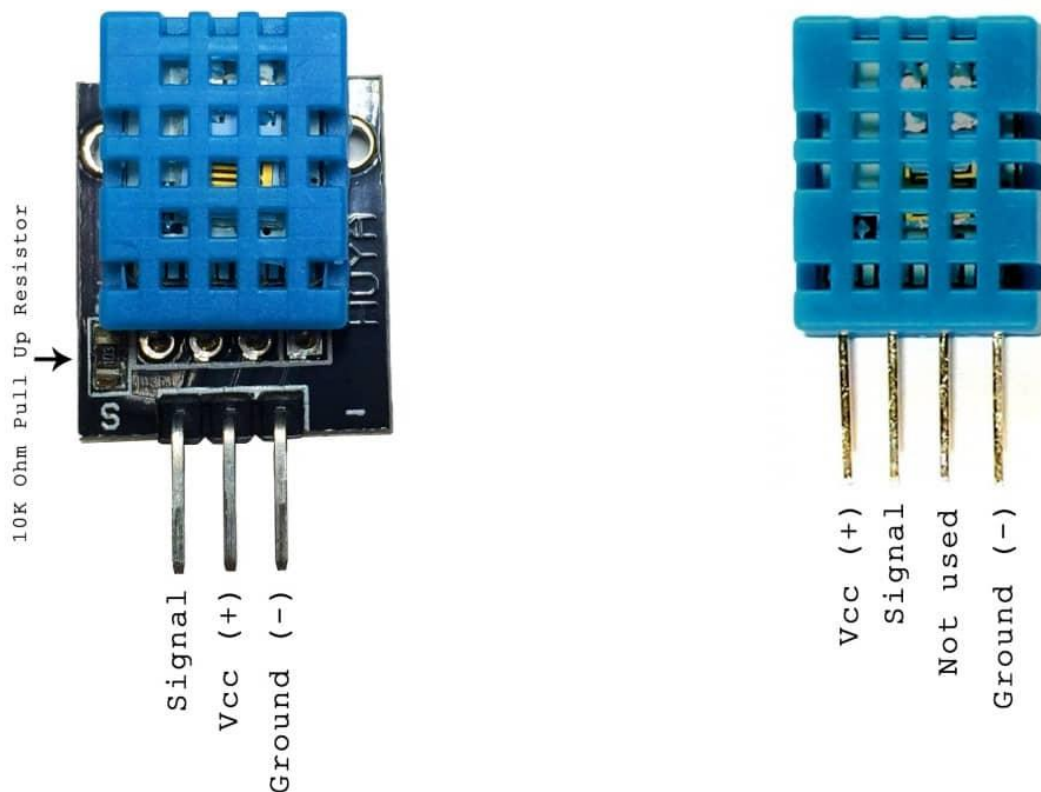
### Pin Description

1, the VDD power supply 3.5~5.5V DC

2 DATA serial data, a single bus

3, NC, empty pin

4, GND ground, the negative power



Type of interface with raspberry pi: Indirect

### 3. IR Sensors

#### Working:

In this project, the transmitter section includes an IR sensor, which transmits continuous IR rays to be received by an IR receiver module. An IR output terminal of the receiver varies depending upon its receiving of IR rays. Since this variation cannot be analyzed as such, therefore this output can be fed to a comparator circuit. Here an operational amplifier (op-amp) of LM 339 is used as comparator circuit.

When the IR receiver does not receive a signal, the potential at the inverting input goes higher than that non-inverting input of the comparator IC (LM339). Thus, the output of the comparator goes low, but the LED does not glow. When the IR receiver module receives signal to the potential at the inverting input goes low. Thus, the output of the comparator (LM 339) goes high and the LED starts glowing. Resistor R1 (100), R2 (10k) and R3 (330) are used to ensure that minimum 10 mA current passes through the IR LED Devices like Photodiode and normal LEDs respectively. Resistor VR2 (preset=5k) is used to adjust the output terminals. Resistor VR1 (preset=10k) is used to set the sensitivity of the circuit Diagram. Read more about IR sensors.

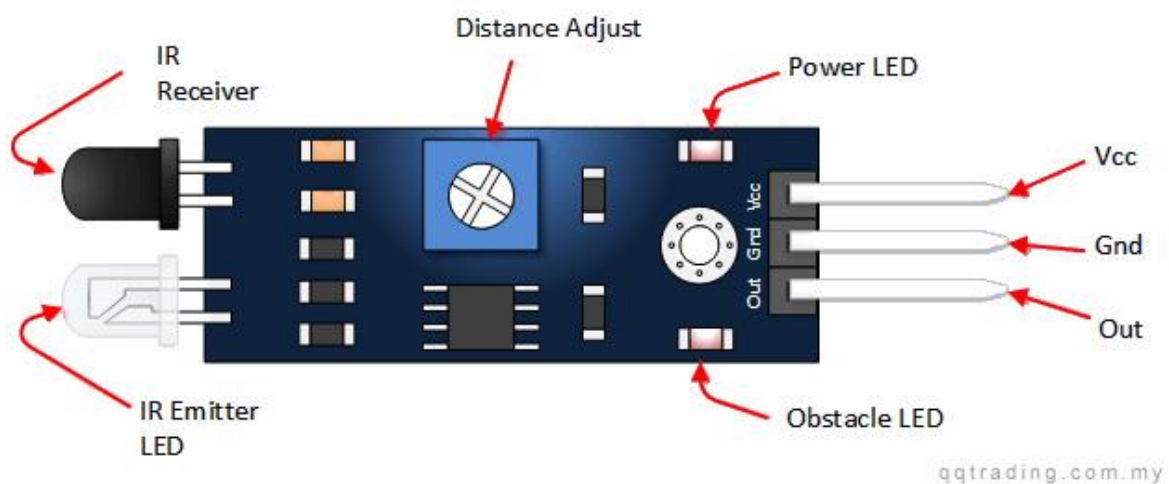
#### Features:

- There is an obstacle, the indicator light on the circuit board

- Digital output signal
- Detection distance: 2 ~ 30cm
- Detection angle: 35 ° Degree
- Comparator chip: LM393
- Adjustable detection distance range via potentiometer:
  - Clockwise: Increase detection distance
  - Counter-clockwise: Reduce detection distance

#### Specifications:

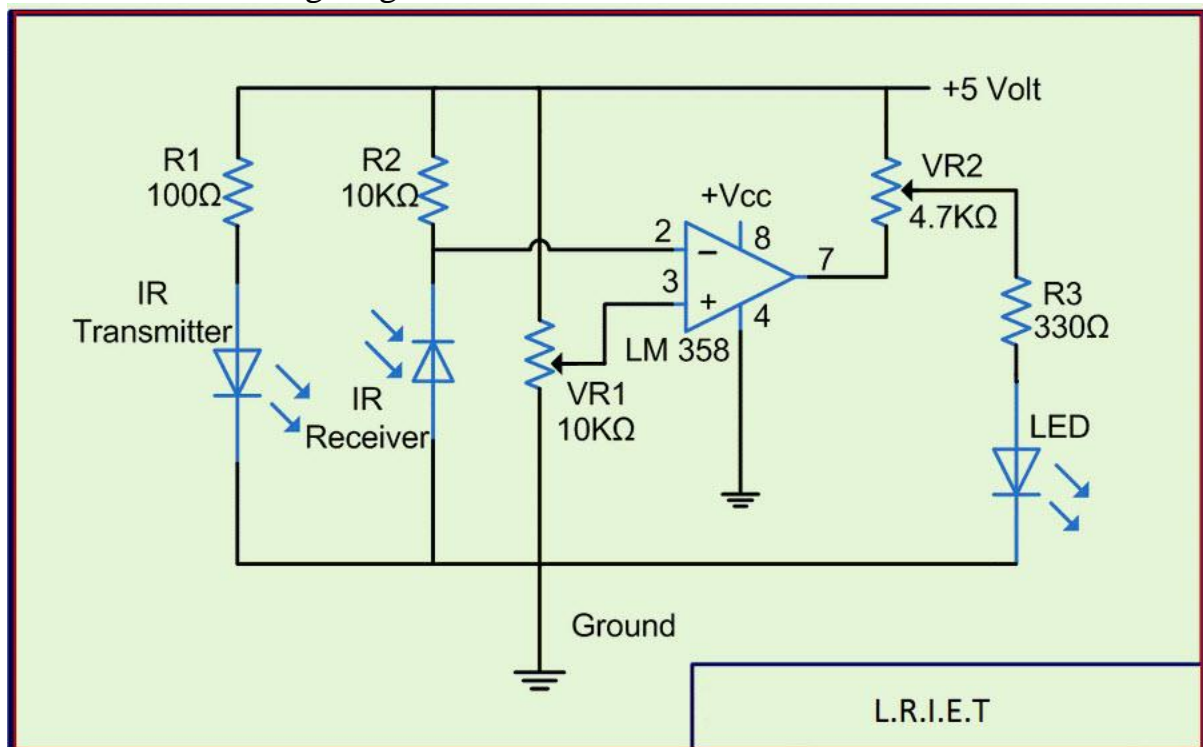
- Working voltage: 3 - 5V DC
- Output type: Digital switching output (0 and 1)
- 3mm screw holes for easy mounting
- Board size: 3.2 x 1.4cm



Pin Control Indicator	Description
Vcc	3.3 to 5 Vdc Supply Input
Gnd	Ground Input
Out	Output that goes low when obstacle is in range
Power LED	Illuminates when power is applied
Obstacle LED	Illuminates when obstacle is detected
Distance Adjust	Adjust detection distance. CCW decreases distance. CW increases distance.
IR Emitter	Infrared emitter LED
IR Receiver	Infrared receiver that receives signal transmitted by Infrared emitter.

Type of interface with raspberry pi: Indirect

Details of interfacing diagram:



### Webpage Details: -

As we brainstormed about from where to send the sensor values to user and where he can interact with Pi; we decided on creating a webpage that will let user to interact with the program; where he can select the different modes for watering the plants and he can also ask for weekly analysis of the growth of the plant; as well as he can do this remotely while on the Wide Area Network. To host this webpage, we take help of apache service and by thus making our raspberry-pi to work as an independent server. To create webpage, we used PHP language. The code of the same is as follows:

```
<html>
<head>

<meta name="viewport" content="width=device-width" />
<style>
p {font-family: "Times New Roman", Times, serif; font-size: 20px; margin-left: 10px}

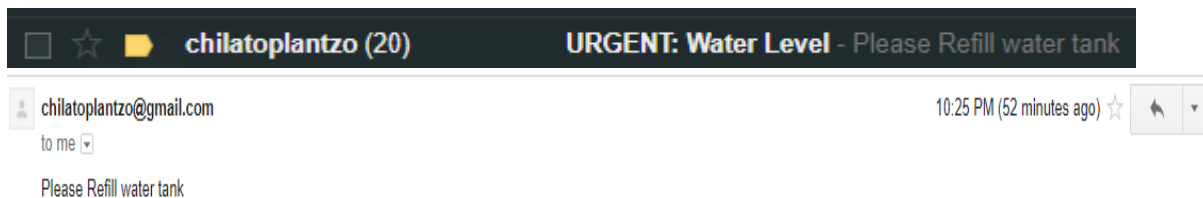
h1 {color:green; text-align: center;font-size: 35px;}
</style>
<title>Plantzo</title>
</head>
<meta http-equiv="refresh" content="2" >
    <body background="PAGE1.jpg">
```

```
<h1>CHILATO PLANTZO</h1>
<p>Temperature: ($_GET['on']) C</p>
<p>Humidity: ($_GET['on'])%</p>
<p>Plant Height: ($_GET['on'])</p>
<p>Lamp Control:</p>
    <form method="get" action="plantzo.php">
        <input type="submit" value="ON" name="on">
        <input type="submit" value="OFF" name="off">
    </form>
    <?php
    $setmode24 = shell_exec("gpio -g mode 24 out");
    if(isset($_GET['on'])){
        $gpio_on = shell_exec("gpio -g write 24 1");
        echo "LED is on";
    }
    else if(isset($_GET['off'])){
        $gpio_off = shell_exec("gpio -g write 24 0");
        echo "LED is off";
    }
    ?>
<p>Watering Mode:</p>
    <form method="get" action="plantzo.php">
        <input type="submit" value="ECONOMICAL"
name="economical">
        <input type="submit" value="NORMAL" name="normal">
    </form>
    <?php
    $setmode24 = shell_exec("gpio -g mode 24 out");
    if(isset($_GET['on'])){
        $gpio_on = shell_exec("gpio -g write 24 1");
    }
    else if(isset($_GET['off'])){
        $gpio_off = shell_exec("gpio -g write 24 0");
    }
    ?>
<p>Water level: Sufficient-above 70%</p>
</body>
</html>
```

Below is the screenshot on webpage



Notification send on email in case of emergency





## **Complete Python code: -**

```
import RPi.GPIO as GPIO
import time
import dht11
import datetime
import smtplib
from email.MIMEMultipart import MIMEMultipart
from email.MIMEText import MIMEText

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.cleanup()

ldrpin = 11
lamppin = 7
smpin = 12
thpin=18
IRpin1 = 29
IRpin2 = 31
IRpin3 = 32
IRpin4 = 33
wlpin1=13
wlpin2=15
motorpin=37

b=0 #notification water level
sumtemperature=0
sumhumidity=0

GPIO.setup(wlpin1, GPIO.IN)
GPIO.setup(wlpin2, GPIO.IN)
GPIO.setup(IRpin1, GPIO.IN)
GPIO.setup(IRpin2, GPIO.IN)
GPIO.setup(IRpin3, GPIO.IN)
GPIO.setup(IRpin4, GPIO.IN)
GPIO.setup(smpin, GPIO.IN)
GPIO.setup(ldrpin, GPIO.IN)
GPIO.setup(lamppin, GPIO.OUT)
GPIO.setup(motorpin, GPIO.OUT)
GPIO.setup(22, GPIO.IN)
instance = dht11.DHT11(thpin)

while True:
    input_event = GPIO.input(ldrpin)
    input_event_sm = GPIO.input(smpin)
    print str(input_event)
    if GPIO.input(22):
        if GPIO.input(ldrpin):
```

```

        print("Detected")
        GPIO.output(lamppin, 0)
    else:
        print("Low Light Condition")
        GPIO.output(lamppin, 1)
    else:
        GPIO.output(lamppin, 0)

    if GPIO.input(smpin):
        print("DRY DRY")
        GPIO.output(motorpin, 1)
    else:
        print("WET WET")
        print("Motor Turn OFF")
        GPIO.output(motorpin, 0)

    result = instance.read()
    #if result.is_valid():
        #print("Last valid input: " + str(datetime.datetime.now()))
        #print("Temperature: %d C" % result.temperature)
        #print("Humidity: %d %% " % result.humidity)

    if GPIO.input(IRpin1):
        if GPIO.input(IRpin2):
            if GPIO.input(IRpin3):
                if GPIO.input(IRpin4):
                    print("Fourth level")
                else:
                    print("Third level")
            else:
                print("Second level")
        else:
            print("First height")
    else:
        print("Ground level")

    if GPIO.input(wlpin1):
        if GPIO.input(wlpin2):
            print("Water level sufficient : above 70%")
        else:
            print("Water levelmoderate : above 20%")
    else:
        print("Water level low : below 20% plz refill asap")
        b=b+1
    if(b>=720):
        fromaddr = "chilatoplantzo@gmail.com"
        toaddr = "creativeknight258@gmail.com"
        msg = MIMEMultipart()
        msg['From'] = fromaddr

```

```
msg['To'] = toaddr
msg['Subject'] = "URGENT: Water Level"

body =str("Please Refill water tank " )
msg.attach(MIMEText(body, 'plain'))

server = smtplib.SMTP('smtp.gmail.com', 587)
server.starttls()
server.login(fromaddr, "onetwo345")
text = msg.as_string()
server.sendmail(fromaddr, toaddr, text)
server.quit()
b=0

time.sleep(0.5)

GPIO.cleanup()
```

### **Working of the project: -**

We have used both protocol that is, simple mail protocol and hyper text transfer protocol. Through soil moisture sensor we get the moisture value (in Binary) after that program decides that whether the plant requires water or not. From temperature and humidity sensor, we get the environmental conditions around the plant which will also decide water requirements of the plant. Whenever there is requirement of water to plant the pi will give output “1” to the pin where our motor is connected through which motor will get voltage which will result in pumping of water from miniature water tank to plant and will be on until soil moisture sensor detects moisture. IR sensors which will be installed vertically alongside of the plant will always be detecting the height of the plant which will be helpful to keep the track of the growth of the plant. Through VNC server and client application, we have made SSH connection to pi simpler as Pi OS can be accessed by any devices which are on Wide Area Network so now, no more worries of connecting confusing wires just go to your phone and you can access your pi. Furthermore, we have developed webpage which will also be accessible to each and every device which are on WAN. So, now you can access your pi as well as keep track of your plant’s growth; Isn’t it interesting?

What more can you think this pi could do? We have added an E-mail feature too which will send mail to destined e-mail id notifying about Lightings, critical water and temperature levels. It will also send mail every week informing about growth of the plant as well as detailed weekly analysis.

	24/10/2017	1/11/2017	5/11/2017	10/11/2017
Report 1 Submission	X			
Ckt Diagram	X	X		
uC to sensor connection		X		
Program completion		X	X	
Final Demo and Viva				X

### References:

<https://readwrite.com/2014/06/27/raspberry-pi-web-server-website-hosting/>  
<https://learn.adafruit.com/adafruits-raspberry-pi-lesson-3-network-setup/setting-up-wifi-with-occidentalis>  
<https://www.dexterindustries.com/howto/walk-through-and-tutorial-on-how-to-connect-the-raspberry-pi-and-mobile-phone-connect-raspberry-pi-mobiletablet/>  
<https://www.raspberrypi.org/documentation/linux/usage/rc-local.md>  
<http://www.makeuseof.com/tag/setup-wi-fi-bluetooth-raspberry-pi-3/>  
<http://www.raspberry-pi-geek.com/Archive/2014/07/PHP-on-Raspberry-Pi>  
<http://naelshiab.com/tutorial-send-email-python/>  
<https://projects.raspberrypi.org/en/projects/lamp-web-server-with-wordpress>  
[https://www.w3schools.com/css/css3\\_buttons.asp](https://www.w3schools.com/css/css3_buttons.asp)

## Appendix A: Data Sheets

- Soil Moisture Sensor



1 | Page

An ISO 9001-2008 Certified Company

SOIL MOISTURE SENSOR

Order Code RDL/SOM/13/001/V1.0

### Soil Moisture Sensor

This sensor can be used to test the moisture of soil, when the soil is having water shortage, the module output is at high level, else the output is at low level. By using this sensor one can automatically water the flower plant, or any other plants requiring automatic watering technique. Module triple output mode, digital output is simple, analog output more accurate, serial output with exact readings.



### Features

- Sensitivity adjustable.
- Has fixed bolt hole, convenient installation.
- Threshold level can be configured.
- Module triple output mode, digital output is simple, analog output more accurate, serial output with exact readings.

### Applications

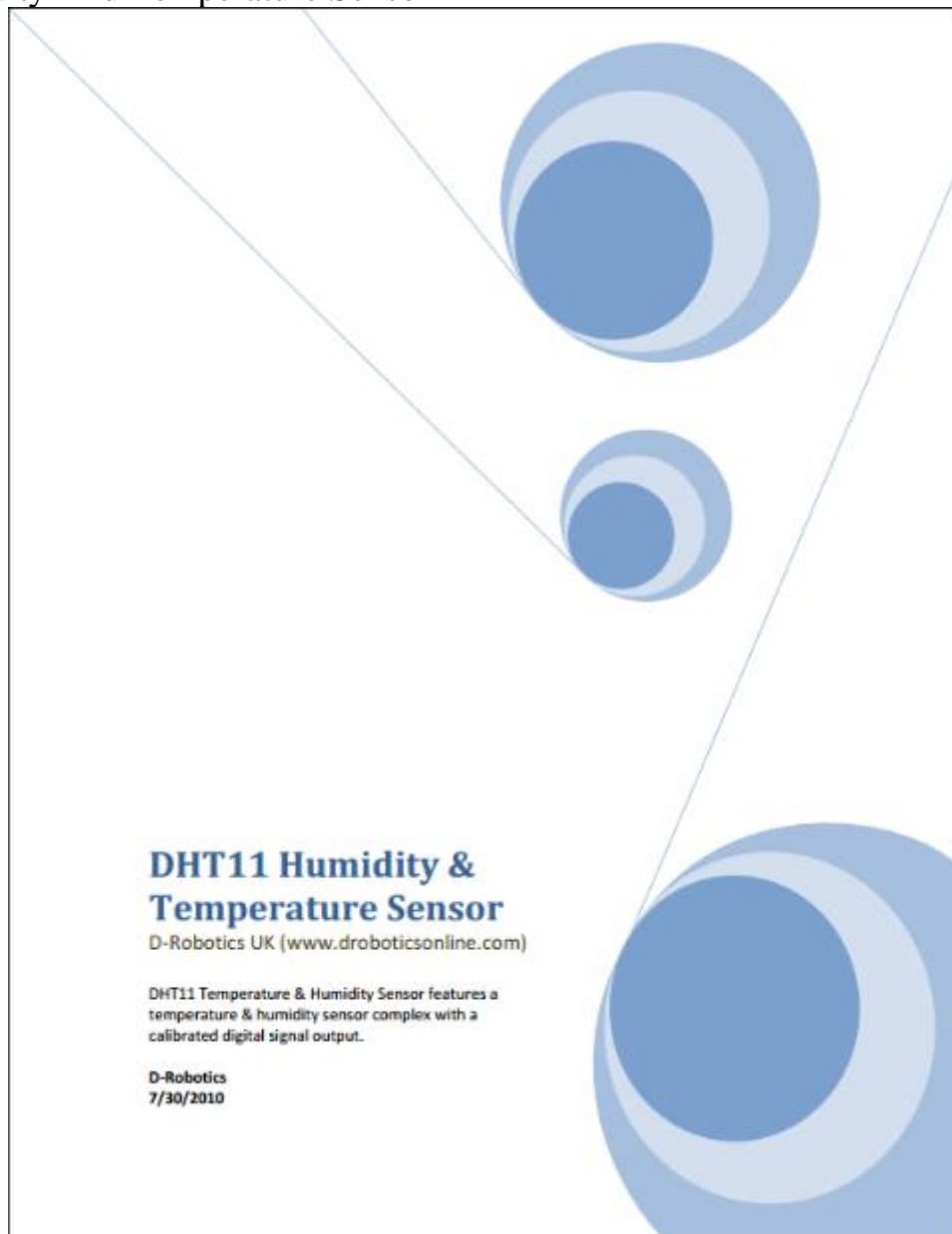
- Agriculture
- Landscape irrigation

### Specifications

Parameter	Value
Operating Voltage	+5v dc regulated
Soil moisture	Digital value is indicated by out pin

[www.researchdesignlab.com](http://www.researchdesignlab.com)

- Humidity And Temperature Sensor



## **Appendix B: Programming Review**

- **Java**

Python programs are generally expected to run slower than Java programs, but they also take much less time to develop. Python programs are typically 3-5 times shorter than equivalent Java programs. This difference can be attributed to Python's built-in high-level data types and its dynamic typing. For example, a Python programmer wastes no time declaring the types of arguments or variables, and Python's powerful polymorphic list and dictionary types, for which rich syntactic support is built straight into the language, find a use in almost every Python program. Because of the run-time typing, Python's run time must work harder than Java's. For example, when evaluating the expression,  $a + b$ , it must first inspect the objects  $a$  and  $b$  to find out their type, which is not known at compile time. It then invokes the appropriate addition operation, which may be an overloaded user-defined method. Java, on the other hand, can perform an efficient integer or floating-point addition, but requires variable declarations for  $a$  and  $b$ , and does not allow overloading of the  $+$  operator for instances of user-defined classes.

For these reasons, Python is much better suited as a "glue" language, while Java is better characterized as a low-level implementation language. In fact, the two together make an excellent combination. Components can be developed in Java and combined to form applications in Python; Python can also be used to prototype components until their design can be "hardened" in a Java implementation. To support this type of development, a Python implementation written in Java is under development, which allows calling Python code from Java and vice versa. In this implementation, Python source code is translated to Java bytecode (with help from a run-time library to support Python's dynamic semantics).

- **C++**

Almost everything said for Java also applies for C++, just more so: where Python code is typically 3-5 times shorter than equivalent Java code, it is often 5-10 times shorter than equivalent C++ code! Anecdotal evidence suggests that one Python programmer can finish in two months what two C++ programmers can't complete in a year. Python shines as a glue language, used to combine components written in C++.

## **Appendix C: Troubleshooting:**

1. The first ever problem we ever faced while making this project a great success was pi does not take analog inputs. To solve the same, we first thought of connecting Arduino with sensors with analog outputs and raspberry-pi which was the most appropriate solution on the internet but this added more device to the circuit also adding more values in the budget which we do not want; so thus, we came up with a different solution through which our work got done. The solution was importing a python library called DHT11\_PYTHON which was specially made for the sensor DHT11 which gave us analog output at the start.

2. Second problem which was also the biggest one which we faced through our journey of the project is finding solution to the confusing wires and giving wirelessly access to the user through which he can access pi as well as check the progress of the plant. For which, we first tried making a data base and connecting that database to PHP webpage or using WordPress to show data on website. This method didn't work for us and always created errors for us (there might be our mistake but never mind) then we arrived to a solution and told ourselves this might not work for us and decided to use just PHP to create webpage which will not store the past values instead will always show the latest values.

3. Rather calling it a problem, it was malfunction by pi as it restarted on itself and when we reconnected mostly all our progress was gone i.e. all the files were there but empty ones. Now, to overcome this we wrote codes all together from the trash.