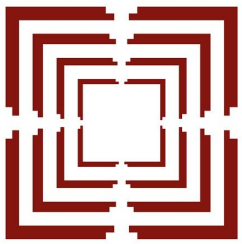


Ahmedabad University

IoT based Plant Health Monitoring using Image Processing and Machine Learning

Report - 3

Under Guidance of,

Prof. Anurag Lakhlani

By,

Group - 13

Saloni Chudgar - 1641013

Abhi Patel - 1641021

Nildeep Jadav - 1641024

Manav Chotalia - 1641036

INDEX

1. Motivation:

Most of the agricultural sector in the country is facing problems due to low economic resources. Moreover, seasonal changes also affect the agricultural growth of the plants. As a result it becomes extremely essential to monitor the health of the plant and the environment around it on a regular basis.

For the effective cultivation of plants, a particular level of sunlight, temperature, humidity and soil moisture becomes extremely essential. The temperature and humidity readings are critical to determining if heating/cooling curves around/in the environment around the plant are optimal for its growth or not. However, measuring these parameters becomes a tedious job. With the use of IoT devices this process can be automated or at least semi-automated with human monitoring. Also, factors like insects and diseases affect plant health. These can not be measured by any sensor. But the effect of most of these diseases can be easily seen on the plant leaf. So by attaching a camera, the plant health can be monitored by using image processing and machine learning algorithms.

2. Description:

Even though India is an agricultural country, numerous farmers fail to get good profits from the crops for the reason that they can't manage two essential factors, which determine plant growth as well as productivity. The temperature surrounding the plants must remain in a certain range. High humidity can result in crop transpiration, condensation of water vapor on various surfaces, and water evaporation from the humid soil. Also a disease in a single plant/crop can spread in all of them quite quickly and thus must be dealt with immediately. To overcome such challenges, this plant health monitoring system comes to the rescue. It can be used not only in farms, but also in homes, offices or any other place that you want to cultivate your plants in.

Devices like Moisture sensor, Temperature sensor, CO2 sensor, Light Sensor, Humidity Sensor can be attached to a Raspberry Pi Microprocessor. The data can be processed and insight can be obtained into the plant growth. The images taken by the camera can be processed using Image Processing Algorithms and Machine Learning algorithms can be used to make predictions about the health of the plant.

3. Final Outcome:

A plant health monitoring system created without human intervention. An alert will be generated and the user will be notified in case of any discrepancies found in the plant health. The live streaming of the data of the sensors will be sent to the user through the internet and the user can monitor it on his laptop/mobile device.

Also if the module(Raspberry Pi) permits i.e the processing is found to be feasible, we would like to do the time series analysis and prediction of the data obtained from sensors using ARIMA(Auto-Regressive Integrated Moving Average) method.

4. Market Survey:

1. Lakshmi, K., and S. Gayathri. "Implementation of IoT with image processing in plant growth monitoring system." *Jour. Sci. Inno. Res.*. Vol. 6. Tamilnadu, India, 2017. 80-83.

   This paper proposes a growth monitoring system using image processing and Internet of Things technique. They achieve the ideal values of temperature and humidity using sensors through IoT, while the disease is detected using image processing by canny edge detection.

2. Agremo - https://www.agremo.com/
   They enable drone operators to help farmers monitor and manage their crops per field, throughout the season. They also analyze and provide a full range of stats: plant counting, plant health tools and stress detectors that enable precise yield increase and increase of overall profit.

3. Kamruzzaman, S. M., et al. "Promoting Greenness with IoT-Based Plant Growth System." *Computational Intelligence and Sustainable Systems*. Springer, Cham, 2019. 235-253.

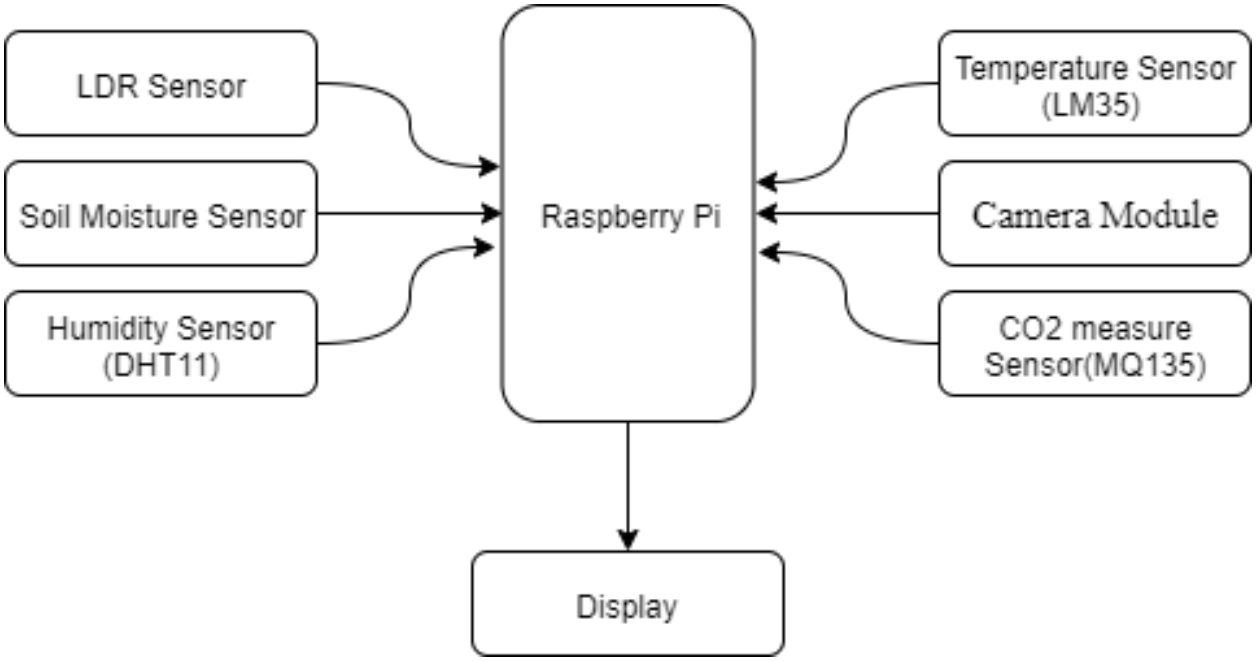
   They have built a plant monitoring system here.

   Pavel, Monirul Islam, et al. "An IoT Based Plant Health Monitoring System Implementing Image Processing." *2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS)*. IEEE, 2019.

   The same researchers have used the plant monitoring system discussed above and added image processing techniques to figure out whether the plants have diseases or not. They have also tried classifying these diseases into different types using k-means clustering.

   Observation : Most of the advancement in this field is in research work which has yet to be implemented. And the ones implemented are using drones which turn out to be costly on a long term basis.

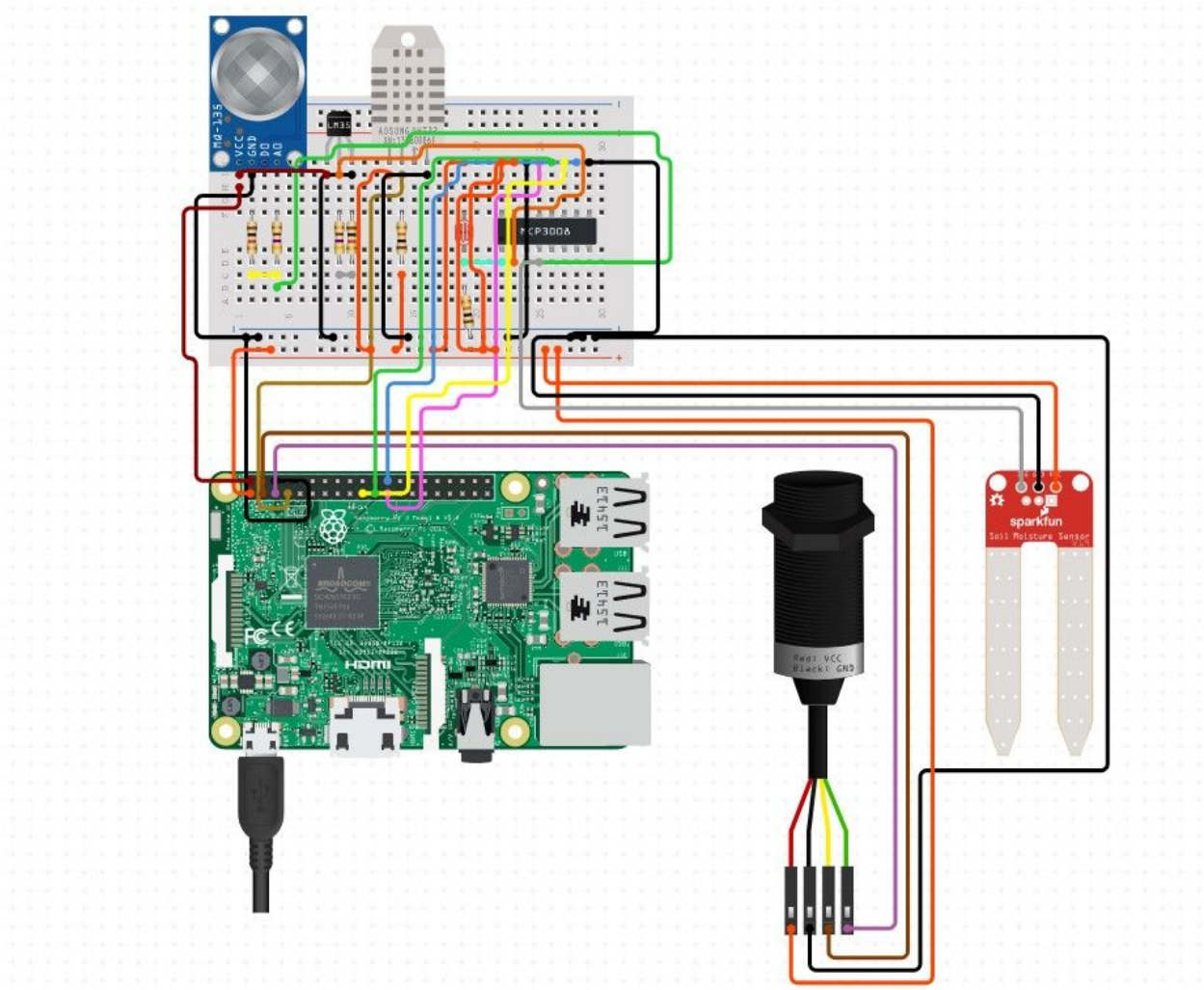
5. Block Diagram:



6. Components Needed:

- ➔ Raspberry Pie
- ➔ LDR Sensor
- ➔ Humidity Sensor
- ➔ CO2 Sensor
- ➔ Moisture Sensor
- ➔ Temperature Sensor
- ➔ Raspberry Pi Camera Module
- ➔ Breadboard
- ➔ Resistors

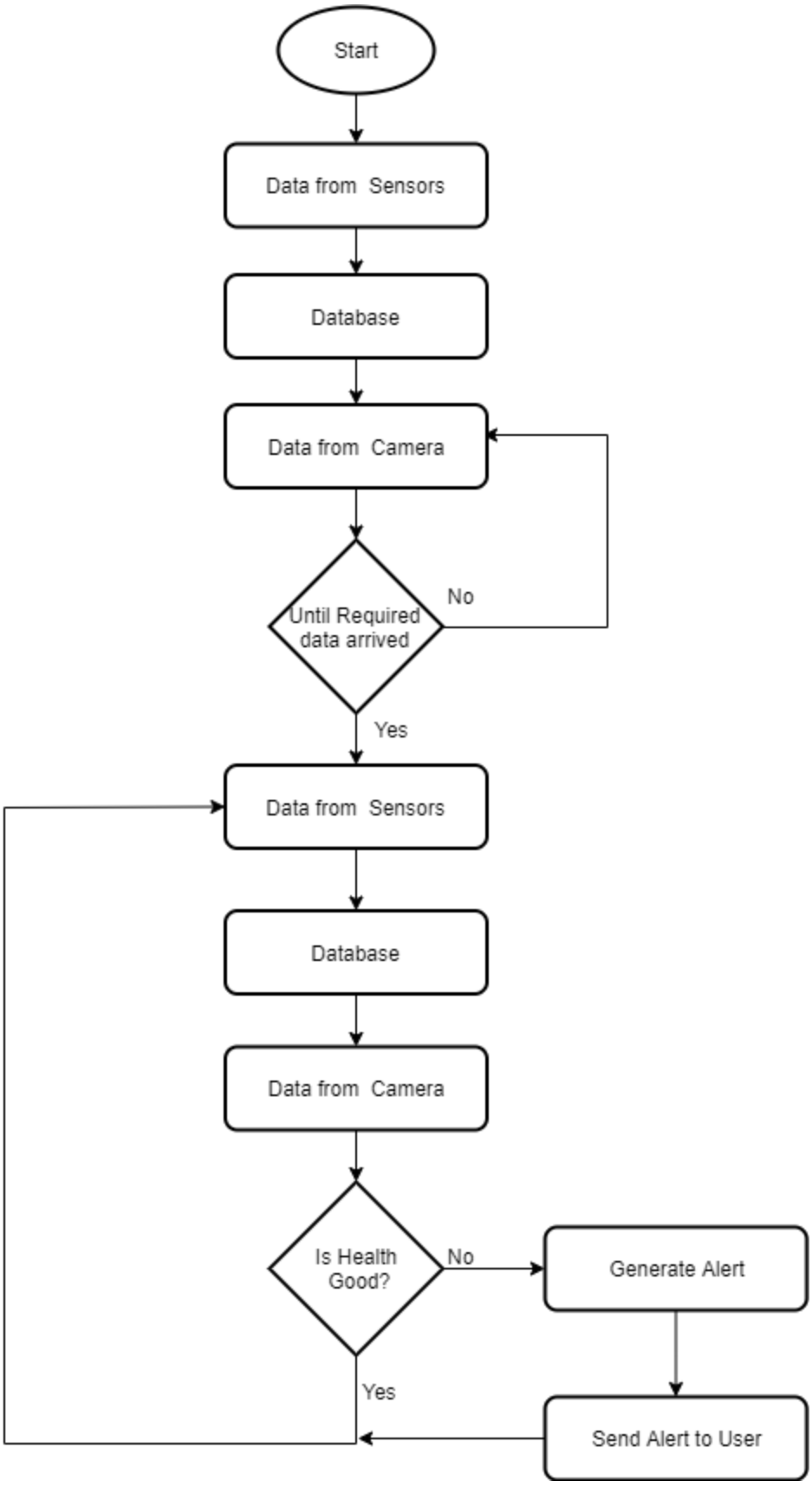
7. Circuit Diagram:



8. Comparison of Features:

	Raspberry Pi	**Intel Edison**	**Beagle Bone**	**Arduino**
Processor	ARM 11	Intel Atom CPU	ARM Cortex-A8	Atmega-328
CPU cores	x1	x2 Dual-core	x1	x1
CPU clocks	700MHz	700MHz	700MHz	16MHz
Memory	1GB/2GB/4GB	1GB	512MB	2KB
Flash	Micro SD Card	4GB(Micro SD)	4GB(Micro SD)	32KB
Expandable Memory	Available (via USB ports)	Not Available	Not Available	Not Available
Digital Input	8	40	66	14
Operating Voltage	5V	3.3V - 4.5V	5V	7V - 12V
Network Connections	Ethernet, Wi-Fi	Wi-Fi, Bluetooth	Ethernet	Ethernet(Externa l Connection)
USB Connections	4 - peripherals	2 - peripherals	1 - peripherals	1 - input only
Operating System	Linux	Yocto - Linux v1.6	BeagleBoard - Angstrom, Ubuntu, Android	None
Display	HDMI, composite	Console port	HDMI	HDMI
Camera Input	Yes	PRU input	No	No
Multitasking	Yes	Yes	Yes	No
Integrated Development Environment	IDLE, Scratch Squeak/LINUX	Arduino IDE, Eclipse, Intel XDK	Python, Scratch, Squeak, Cloud9/Linux	Arduino IDE

9. Flow Chart:


Start
Data from Sensors
Database
Data from Camera
Until Required data arrived
No
Yes
Data from Sensors
Database
Data from Camera
Is Health Good?
No
Generate Alert
Yes
Send Alert to User

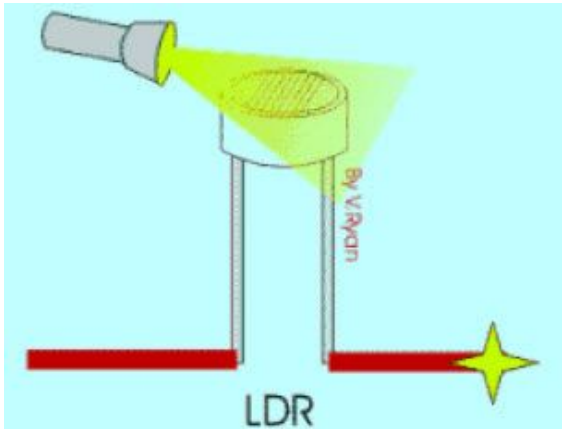
10. Sensors and Actuators:

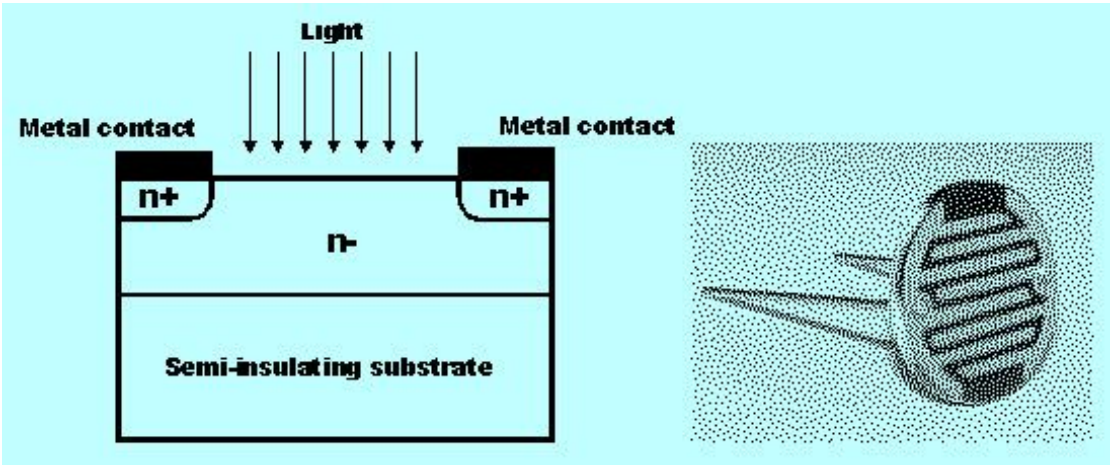
- **LDR Sensor:**

Operating principle (with diagrams):

The working principle of an LDR is photo conductivity, that is nothing but an optical phenomenon.When the light is absorbed by the material then the conductivity of the material reduces. When the light falls on the LDR, then the electrons in the valence band of the material are eager to the conduction band. But, the photons in the incident light must have energy superior than the band gap of the material to make the electrons jump from one band to another band (valance to conduction).

Hence, when light having ample energy, more electrons are excited to the conduction band which grades in a large number of charge carriers. When the effect of this process and the flow of current starts flowing more, the resistance of the device decreases.





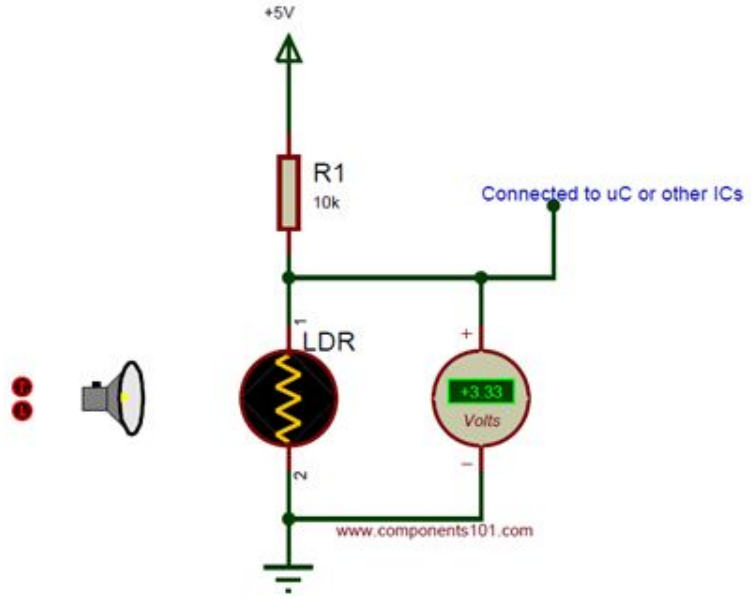
Physical dimensions:

3cm x 1.6 cm

Details of power ratings:

250mW

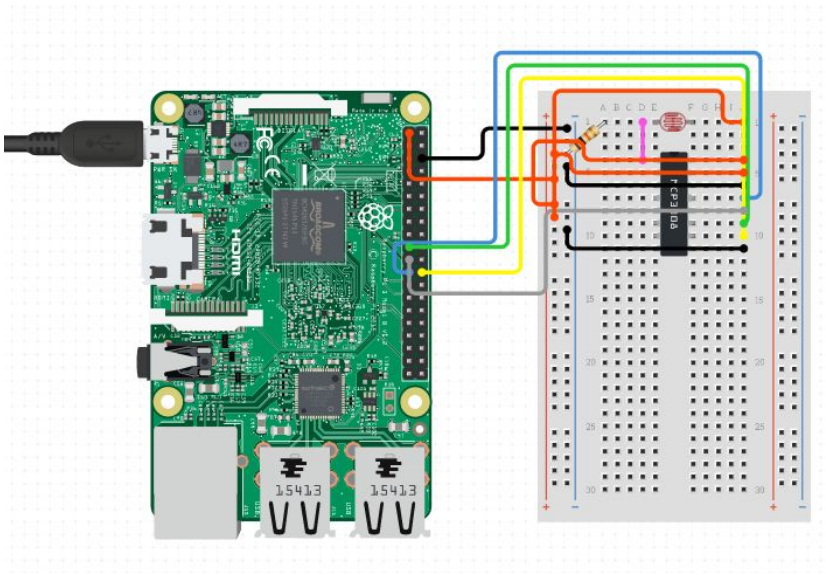
Pin diagram:



Type of interface with Raspberry Pi:

SPI interface

Details of interfacing diagram:



Python code:

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BOARD)                    #define the pin that goes to the circuit
pin_to_circuit = 22
def rc_time (pin_to_circuit):
    count = 0                               #Output on the pin for
```

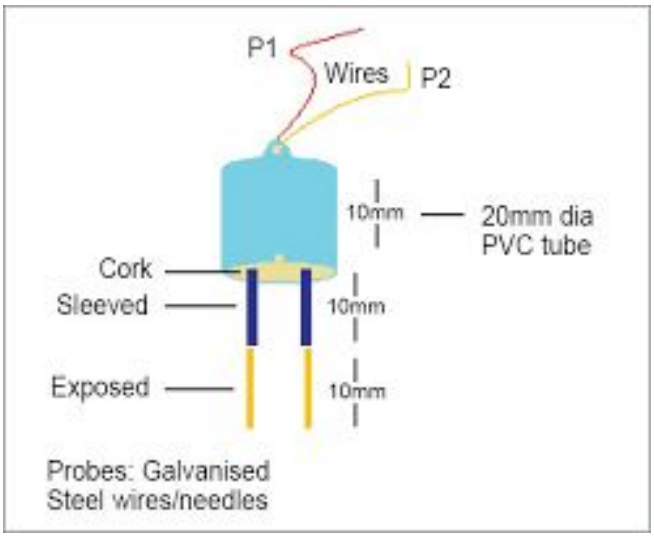
```
    GPIO.setup(pin_to_circuit, GPIO.OUT)
    GPIO.output(pin_to_circuit, GPIO.LOW)
    time.sleep(0.1)                    #Change the pin back to input
    GPIO.setup(pin_to_circuit, GPIO.IN)        #Count until the pin goes high
    while (GPIO.input(pin_to_circuit) == GPIO.LOW):
        count += 1
     return count                      #Catch when script is interupted, cleanup correctly
try                                    # Main loop
    while True:
    print ( rc_time(pin_to_circuit))
    except KeyboardInterrupt:
    pass
finally:
    GPIO.cleanup()
```

- **Soil Moisture Sensor:**

Operating principle (with diagrams):

This sensor mainly utilizes capacitance to gauge the water content of the soil (dielectric permittivity). The working of this sensor can be done by inserting this sensor into the earth and the status of the water content in the soil can be reported in the form of a percent.

This sensor makes it perfect to execute experiments within science courses like environmental science, agricultural science, biology, soil science, botany, and horticulture.



Physical dimensions:

4.6cm x 2.5cm

Details of power ratings:

Input Voltage: 3.3–5V. Output Voltage: 0–4.2V. Input Current: 35mA.

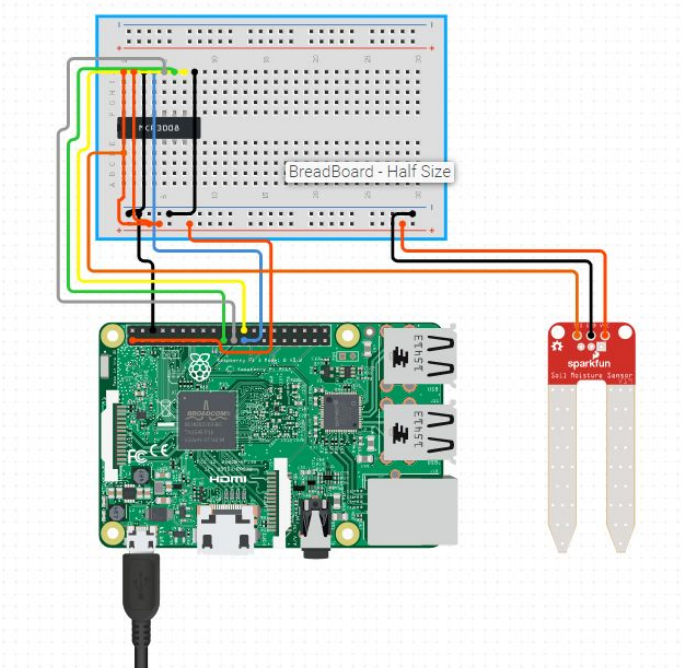
Pin diagram:

- The FC-28 soil moisture sensor includes 4-pins
- VCC pin is used for power
- A0 pin is an analog output
- D0 pin is a digital output
- GND pin is a Ground

Type of interface with Raspberry Pi:

SPI interface

Details of interfacing diagram:



Python code:

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time

#GPIO SETUP
channel = 21
GPIO.setmode(GPIO.BCM)
GPIO.setup(channel, GPIO.IN)

def callback(channel):
        if GPIO.input(channel):
                print "Water Detected!"
        else:
                print "Water Detected!"
```

```
GPIO.add_event_detect(channel, GPIO.BOTH, bouncetime=300)
GPIO.add_event_callback(channel, callback)
# infinite loop
while True:
        time.sleep(1)
```

- **DHT11 (Temperature and Humidity Sensor):**

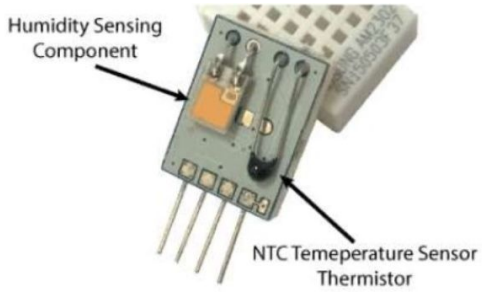
Operating principle (with diagrams):

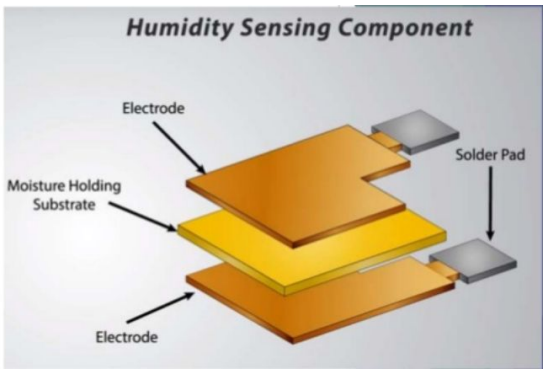
DHT11 sensor consists of a capacitive humidity sensing element and a thermistor for sensing temperature. The humidity sensing capacitor has two electrodes with a moisture-holding substrate as a dielectric between them. Change in the capacitance value occurs with the change in humidity levels. The IC measure, process changed resistance values and change them into digital form.

For measuring temperature this sensor uses a Negative Temperature coefficient thermistor, which causes a decrease in its resistance value with an increase in temperature. To get larger resistance value even for the smallest change in temperature, this sensor is usually made up of semiconductor ceramics or polymers.

Physical dimensions:

1.2cm x 0.5cm x 1.6cm





Details of power ratings: The temperature range of DHT11 is from 0 to 50 degree Celsius with a 2-degree accuracy. The humidity range of this sensor is from 20 to 80% with 5% accuracy. The sampling rate of this sensor is 1Hz .i.e. it gives one reading for every second. DHT11 is small in size with operating voltage from 3 to 5 volts. The maximum current used while measuring is 2.5mA.

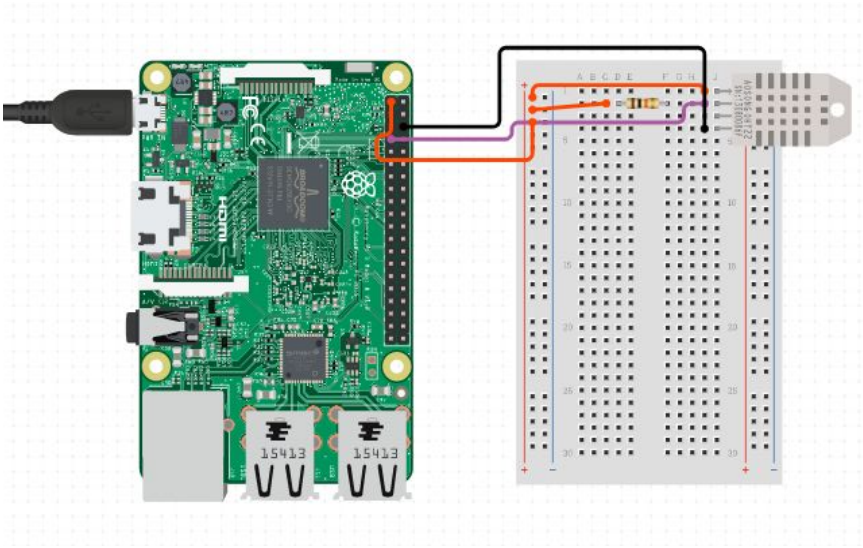
Pin diagram:

- VCC pin is used for power
- D1 pin is a digital output
- GND pin is for Ground

Type of interface with Raspberry Pi:

C GPIO libraries

Details of interfacing diagram:



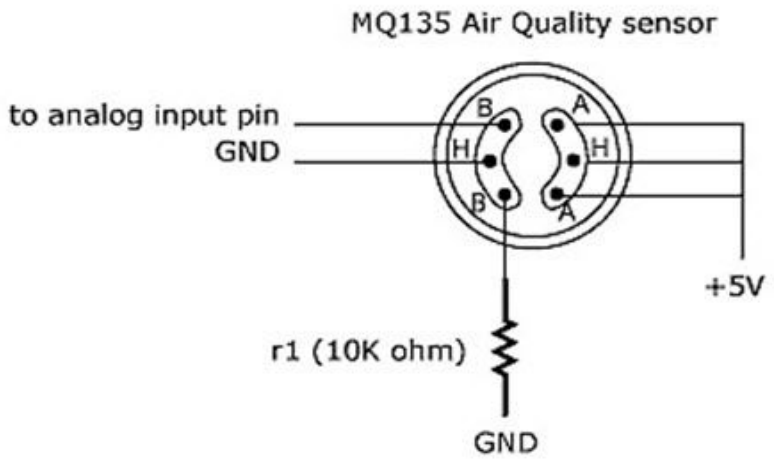
Python code:

```
import Adafruit_DHT
sensor=Adafruit_DHT.DHT11
gpio=17
humidity, temperature = Adafruit_DHT.read_retry(sensor, gpio)

if humidity is not None and temperature is not None:
  print('Temp={0:0.1f}*C  Humidity={1:0.1f}%'.format(temperature, humidity))
else:
  print('Failed to get reading. Please try again')
```

- **MQ135 (Air Quality Sensor):**

operating principle (with diagrams):

The MQ-135 alcohol sensor consists of a tin dioxide (SnO2), a perspective layer inside aluminum oxide microtubes (measuring electrodes) and a heating element inside a tubular casing. The end face of the sensor is enclosed by a stainless steel net and the backside holds the connection terminals.



Physical dimensions:
32 x 20 x 22 (l*w*h)mm

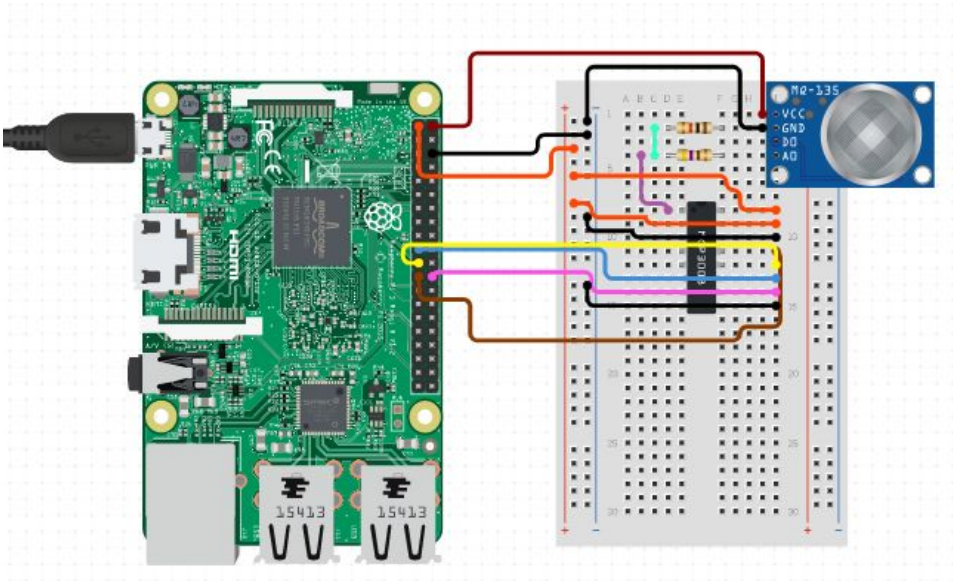
details of power ratings:
Power consumption (current): 150mA

Pin diagram:
The MQ135 soil moisture sensor includes 4-pins

- VCC pin is used for power
- A0 pin is an analog output
- D0 pin is a digital output
- GND is for Ground

Type of interface with Raspberry Pi:
SPI interface

Details of interfacing diagram:



Python code:

```
import spidev
import time

### CODE MQ-135 (CO2) ###

spi = spidev.SpiDev()
spi.open(0, 0)

def ReadChannel(channel):
  adc = spi.xfer2([1, (8 + channel) << 4, 0])
  data = ((adc[1] & 3) << 8) | adc[2]
  return data

while True:

  channel0 = ReadChannel(0)
  channel0_perc = format(channel0, '.2f')
  print("CO2: " + channel0_perc + "ppm")
  time.sleep(5)
```

- **Pi Camera:**

operating principle (with diagrams):

The camera consists of a small (25mm by 20mm by 9mm) circuit board, which connects to the Raspberry Pi's Camera Serial Interface (CSI) bus connector via a flexible ribbon cable. The camera's image sensor has a native resolution of five megapixels and has a fixed focus lens. The software for the camera supports full-resolution still images up to 2592x1944 and video resolutions of 1080p30, 720p60 and 640x480p60/90.

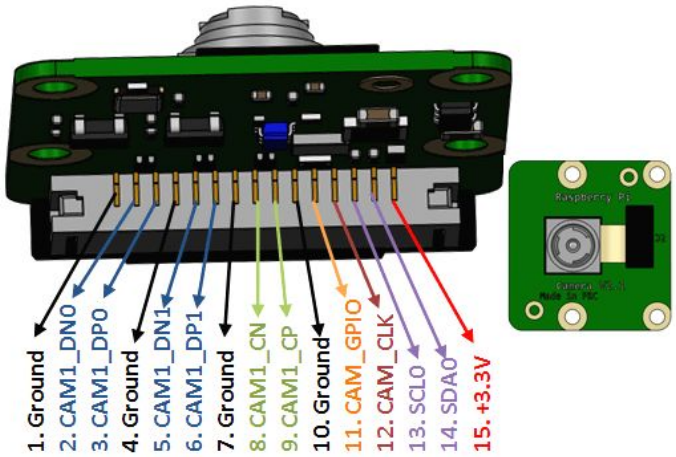
Physical dimensions:

2.3 x 2.5 x 0.9 cm

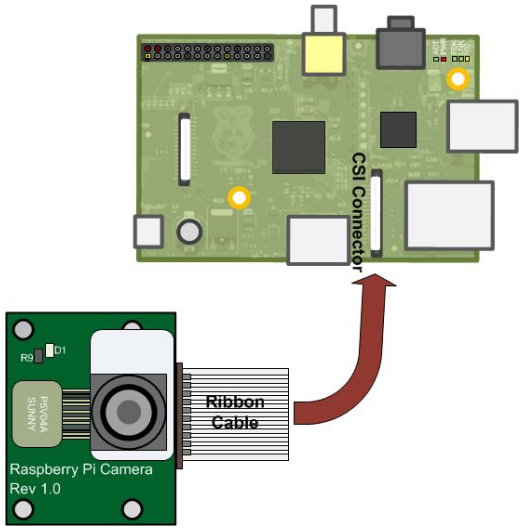
details of power ratings:

Power consumption(current): 200-250mA

Pin diagram:



Details of interfacing diagram:



Python code:

```
import picamera
import time

camera = picamera.PiCamera()
camera.capture('example.jpg')

camera.vflip = True

camera.capture('example2.jpg')

camera.start_recording('examplevid.h264')
time.sleep(5)
camera.stop_recording()
```

11. Details of Communication Protocols :

Wifi - IEEE 802.11 wireless LANs use a media access control protocol called Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). While the name is similar to Ethernet's Carrier Sense Multiple Access with Collision Detection (CSMA/CD), the operating concept is totally different.

WiFi systems are the half duplex shared media configurations, where all stations transmit and receive on the same radio channel. The fundamental problem of a radio system is that a station cannot hear while it is sending, and hence it is impossible to detect a collision. Because of this, the developers of the 802.11 specifications came up with a collision avoidance mechanism called the Distributed Control Function (DCF).

According to DCF, a WiFi station will transmit only when the channel is clear. All transmissions are acknowledged, so if a station does not receive an acknowledgement, it assumes a collision occurred and retries after a random waiting interval.

The incidence of collisions will increase as the traffic increases or in situations where mobile stations cannot hear each other.

Wi-Fi is a family of wireless networking technologies, based on the IEEE 802.11 family of standards, which are commonly used for local area networking of devices and Internet access. Wi‑Fi is a trademark of the non-profit Wi-Fi Alliance, which restricts the use of the term Wi-Fi Certified to products that successfully complete interoperability certification testing.As of 2010, the Wi-Fi Alliance consisted of more than 375 companies from around the world. As of 2009, Wi-Fi-integrated circuit chips shipped approximately 580 million units annually. Devices that can use Wi-Fi technologies include desktops and laptops, smartphones and tablets, smart TVs, printers, digital audio players, digital cameras, cars and drones.

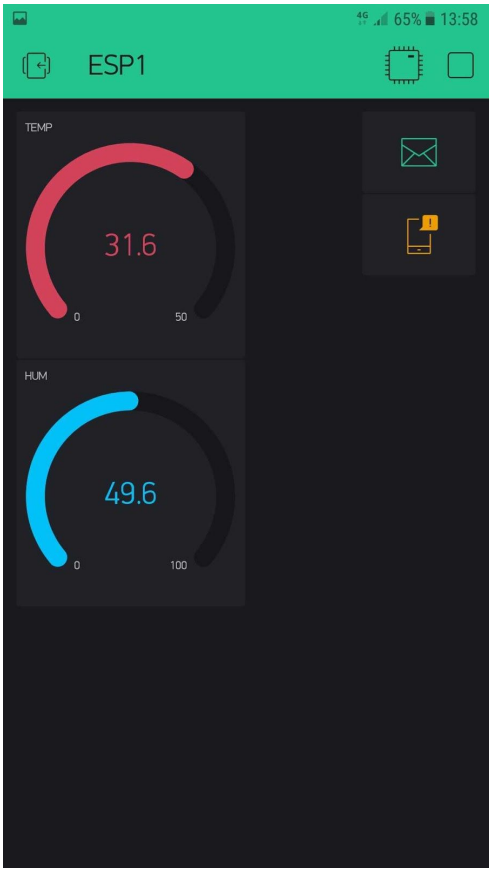
Wi-Fi uses multiple parts of the IEEE 802 protocol family and is designed to seamlessly interwork with its wired sibling Ethernet. Compatible devices can network through a wireless access point to each other as well as to wired devices and the Internet. The different versions of Wi-Fi are specified by various IEEE 802.11 protocol standards, with the different radio technologies determining radio bands, and the maximum ranges, and speeds that may be achieved. Wi-Fi most commonly uses the 2.4 gigahertz (120 mm) UHF and 5 gigahertz (60 mm) SHF ISM radio bands; these bands are subdivided into multiple channels. Channels can be shared between networks but only one transmitter can locally transmit on a channel at any moment in time.

Wi-Fi's wavebands have relatively high absorption and work best for line-of-sight use. Many common obstructions such as walls, pillars, home appliances etc. may greatly reduce range, but this also helps minimize interference between different networks in crowded environments. An access point (or hotspot) often has a range of about 20 metres (66 feet) indoors while some modern access points claim up to a 150-metre (490-foot) range outdoors. Hotspot coverage can be as small as a single room with walls that block radio waves, or as large as many square kilometres using many overlapping access points with roaming permitted between them. Over time the speed and spectral efficiency of Wi-Fi has increased. As

of 2019, at close range, some versions of Wi-Fi running on suitable hardware can achieve speeds of over 1 Gbit/s (gigabit per second).

To connect to a Wi-Fi network, a user typically needs the network name called the SSID and a password. Wi-Fi is potentially more vulnerable to attack than wired networks because anyone within range of a network with a wireless network interface controller can attempt access. Wi-Fi Protected Access (WPA) is a family of technologies created to protect information moving across Wi-Fi networks and includes solutions for personal and enterprise networks. As the security landscape has changed over time security features of WPA have included stronger protections and new security practices. WPA uses the password to encrypt Wi-Fi packets so as to block eavesdroppers.

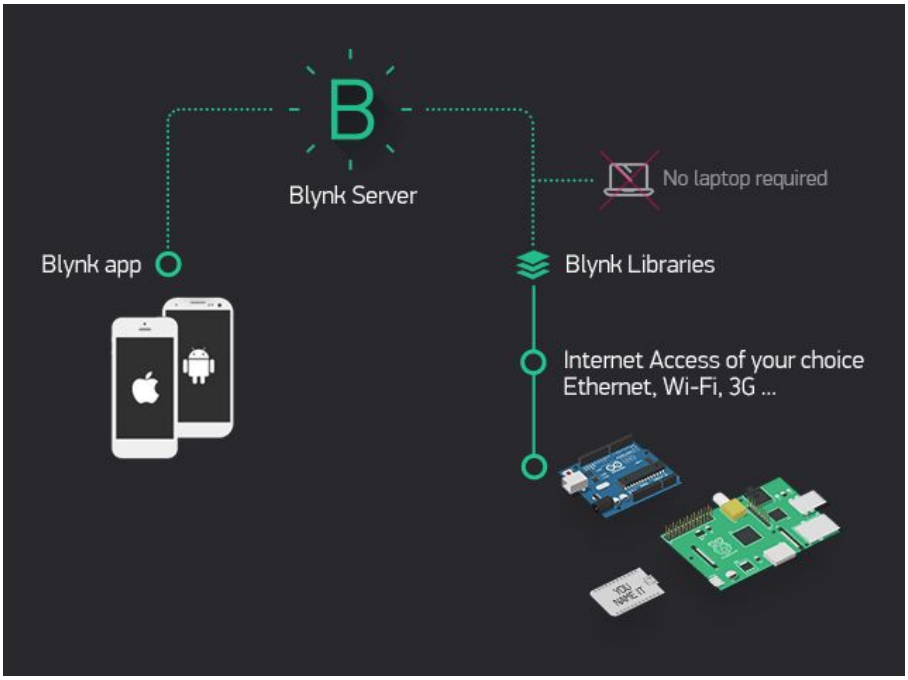
Blynk Notification Services - Blynk provides an app notification service as well as e-mail alert system. The email alert execution is done with 3 parameters: email address, subject and body The time limit between sending emails is minimum 15 seconds. In case you are using gmail you are limited with 500 mails per day (by google). Other providers may have similar limitations, so please be careful. The Blynk free cloud server gives you a maximum of 100 emails every day.
Blynk also offers Notification API which triggers Mobile OS specific notifications.

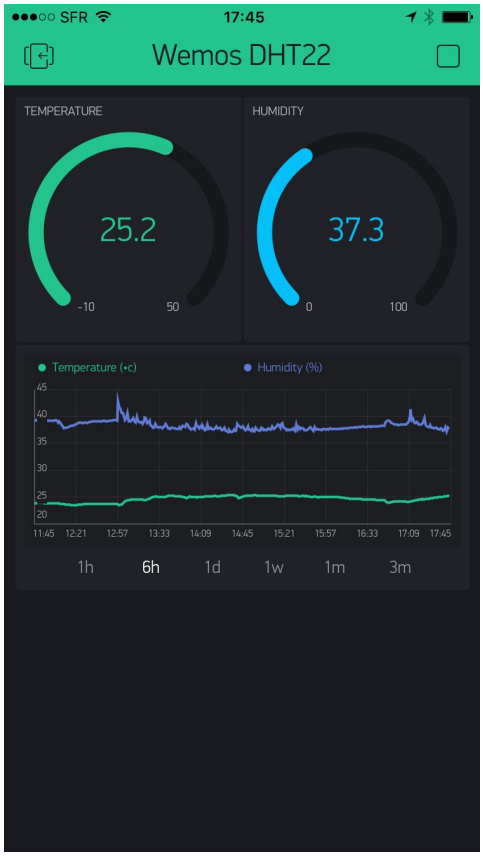


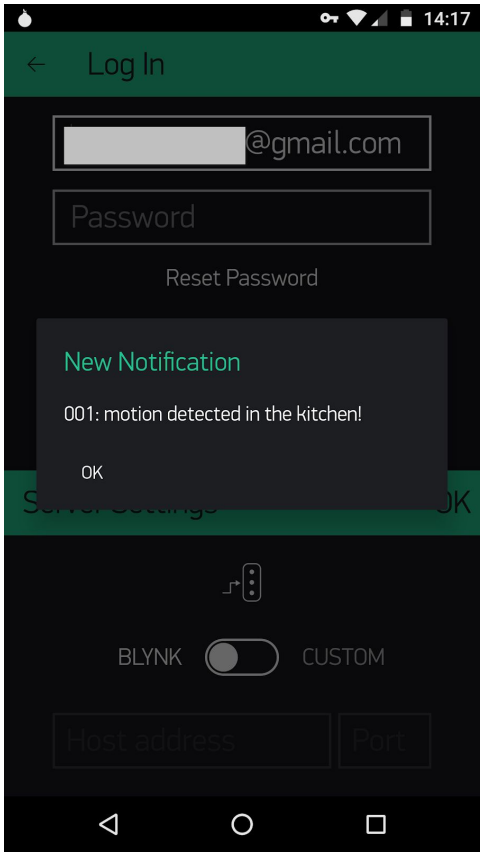
Blynk RESTful API - Blynk HTTP RESTful API allows to easily read and write values to/from Pins in Blynk apps and Hardware (microcontrollers and microcomputers like Arduino, Raspberry Pi, ESP8266, Particle, etc.).Every PUT request will update Pin's state both in apps and on the hardware. Every GET request will return current state/value on the given Pin.

12. Details of supporting tools:

1. **Blynk App** - It can control hardware remotely, it can display sensor data, it can store data, and help visualize it. We are using it for displaying and generating alert.







Code for interfacing between Blynk and Raspberry Pi

```
//#define BLYNK_DEBUG
#define BLYNK_PRINT stdout
#ifdef RASPBERRY
  #include <BlynkApiWiringPi.h>
#else
  #include <BlynkApiLinux.h>
#endif
#include <BlynkSocket.h>
#include <BlynkOptionsParser.h>

static BlynkTransportSocket _blynkTransport;
BlynkSocket Blynk(_blynkTransport);

static const char *auth, *serv;
static uint16_t port;

#include <BlynkWidgets.h>

BlynkTimer tmr;

BLYNK_WRITE(V1)
{
   printf("Got a value: %s\n", param[0].asStr());
}

void setup()
{
   Blynk.begin(auth, serv, port);
   tmr.setInterval(1000, [](){
     Blynk.virtualWrite(V0, BlynkMillis()/1000);
   });
}

void loop()
{
   Blynk.run();
   tmr.run();
}


int main(int argc, char* argv[])
{
```

```
    parse_options(argc, argv, auth, serv, port);

    setup();
    while(true) {
        loop();
    }

    return 0;
}
```

2. **phpMyAdmin** - phpMyAdmin is a free and open source administration tool for MySQL and MariaDB. As a portable web application written primarily in PHP, it has become one of the most popular MySQL administration tools, especially for web hosting services.
   We are using phpMyAdmin to store our image dataset.

13. Complete Code:

```
import Adafruit_DHT
from gpiozero import LightSensor
import time
import requests


dht=Adafruit_DHT.DHT11
gpio=17

ldr = LightSensor(25)
while True:
   humidity, temperature = Adafruit_DHT.read_retry(dht, gpio)
   if humidity is not None and temperature is not None:
     print('Temp={0:0.1f}*C  Humidity={1:0.1f}%'.format(temperature, humidity))
     time.sleep(3)
   else:
     print('Failed to get reading. Try again!')
     time.sleep(3)

   LDRValue = 1024 - (1000*ldr.value)
   #print(ldr.value)
   print(LDRValue)
   time.sleep(3)

   def req(url):
      r = requests.get(url)
      return r.text


req("http://10.1.36.241/senddata.php?temp="+str(temperature)+"&ldr="+str(LDRValue)+"&hu
mid="+str(humidity))
```

TrainModel.py

```
from keras.models import model_from_json
import cv2
import numpy as np
```

```
json_file = open('./model.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(loaded_model_json)

loaded_model.load_weights("./model.h5")
print("Loaded model from disk")

loaded_model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])

image = cv2.imread('./valid/good/1.JPG’)
image = cv2.resize(image, (50,50))
image = image.reshape(1, 50, 50, 3)

cv2.imshow("Input Image", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
result = loaded_model.predict_classes(image)
if(result[0][0] == 1):
   print("Diseased")
else:
   print("No Diseased")
```

Classifier.py

```
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense

model = Sequential()

model.add(Conv2D(32, (3, 3), input_shape = (50, 50, 3), activation = 'relu'))

model.add(MaxPooling2D(pool_size = (2, 2)))

model.add(Conv2D(32, (3, 3), activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
```

```
model.add(Flatten())

model.add(Dense(units = 128, activation = 'relu'))
model.add(Dense(units = 1, activation = 'sigmoid'))

model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])

from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)

training_set = train_datagen.flow_from_directory('./Dataset',
                                                 target_size = (50, 50),
                                                 batch_size = 32,
                                                 class_mode = 'binary')
model.fit_generator(training_set,
                    steps_per_epoch = 4000,
                    epochs = 25,
                    validation_steps = 2000)

model_json = model.to_json()
with open("./model.json","w") as json_file:
  json_file.write(model_json)

model.save_weights("./model.h5")

print("Classifier trained Successfully!")
```

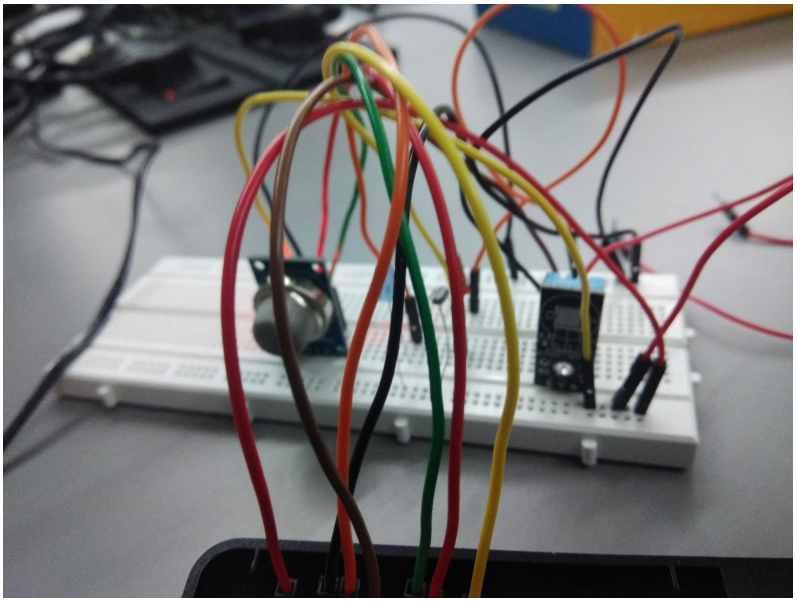
BlynkDisplay Code

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include "Adafruit_SSD1306.h"
#include <BlynkSimpleEsp8266.h>
#include <SimpleTimer.h>
```

```
#include <DHT.h>

#define OLED_RESET 0  // GPIO0
Adafruit_SSD1306 display(OLED_RESET);

#define DHTPIN D4
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

SimpleTimer timer;

char auth[] = "39RaC_T1irF9fbR1s_t2w0TaUpeGx5ck";

void sendSensor(){
  float t = dht.readTemperature();
  float h = dht.readHumidity();
  Blynk.virtualWrite(V0, t);
  Blynk.virtualWrite(V1, h);
  displaySensor(t,h);
}

void displaySensor(float t, float h){
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0,0);

  display.println("- Blynk -");
  display.print("T ");
  display.print(t);
  display.println(" *C");

  display.print("H ");
  display.print(h);
  display.println(" %");

  display.println("");
  display.println("projetsdiy");
  display.display();
}
```

```
void setup() {
  Serial.begin(115200);
  Blynk.begin(auth, ssid, pass, IPAddress(192,168,1,24), 8442);
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.display();
  timer.setInterval(1000L, sendSensor);
}
void loop() {
  Blynk.run();
  timer.run();
}
```

14. Summary:



The values from sensors like
Are sent to the server from Raspberry Pie. These values are displayed on the server and saved as data records.

+ Options

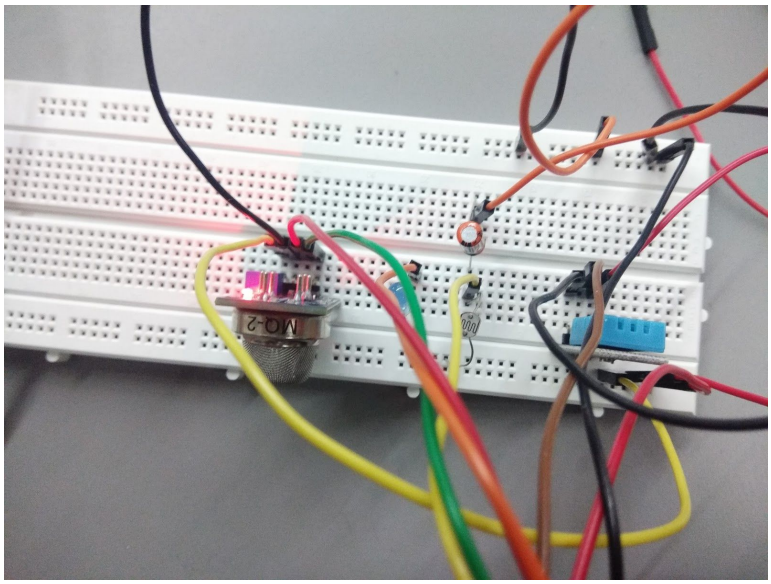
humidity	time
1	2019-11-16 15:14:26
55	2019-11-16 15:32:14
55	2019-11-16 15:32:21
55	2019-11-16 15:32:27
55	2019-11-16 15:32:42
55	2019-11-16 15:32:53

+ Options

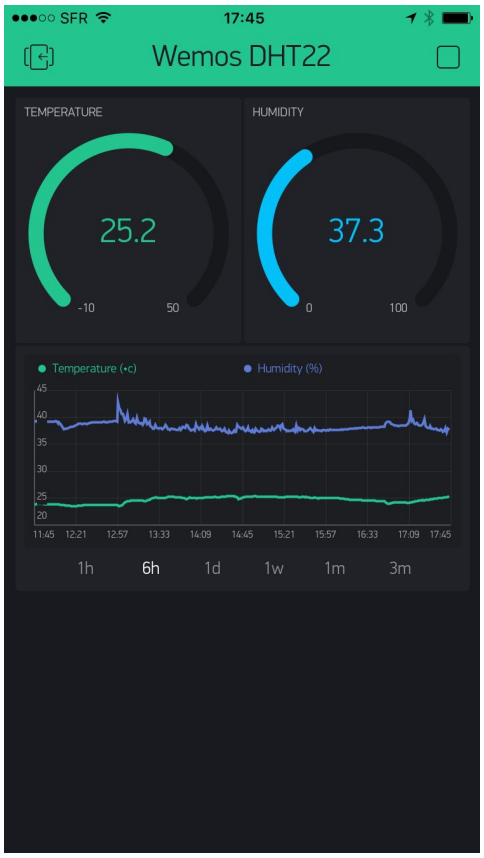
ldr	time
1	2019-11-16 15:14:26
241	2019-11-16 15:32:14
239	2019-11-16 15:32:21
240	2019-11-16 15:32:27
241	2019-11-16 15:32:42
241	2019-11-16 15:32:53

+ Options

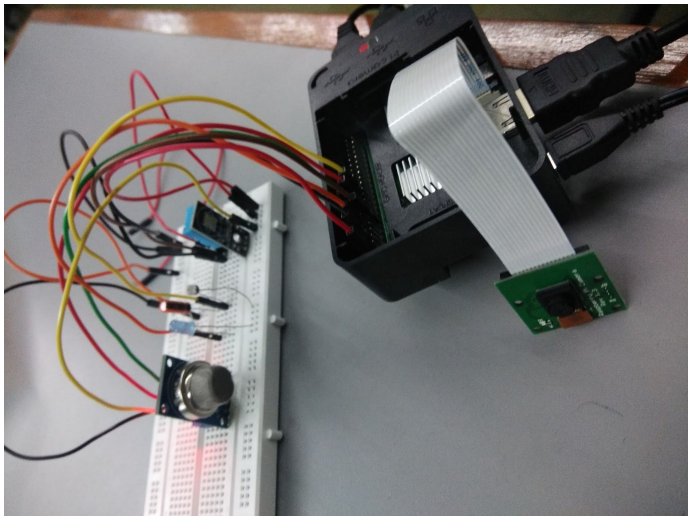
temp	time
1	2019-11-16 15:14:26
29	2019-11-16 15:32:14
29	2019-11-16 15:32:21
29	2019-11-16 15:32:27
29	2019-11-16 15:32:42
29	2019-11-16 15:32:53



Simultaneously these values are sent to the blynk app via WiFi. This app can be used by user for viewing purposes.



The images captured by PiCam are stored in a folder. On addition of every image, this image is then compared through a machine learning algorithm with the help of a pre-trained model. If the leaf(plant) is diseased, the user is alerted.



15. Timeline:

Tasks	30/9-2/10	3/10 - 15/10	16/10 -31 /10	1/11 - 15/11
Gather Components	Done			
Build circuit	Done			
Test individual Components		Done		
Connect via internet (Code for remote server)			Done	
Combine all the components			Done	
Testing and Debugging				Done
Project Report				Done

APPENDIX A - DATASHEETS

Raspberry Pi :

2 Features

2.1 Hardware

- Quad core 64-bit ARM-Cortex A72 running at 1.5GHz
- 1, 2 and 4 Gigabyte LPDDR4 RAM options
- H.265 (HEVC) hardware decode (up to 4Kp60)
- H.264 hardware decode (up to 1080p60)
- VideoCore VI 3D Graphics
- Supports dual HDMI display output up to 4Kp60

2.2 Interfaces

- 802.11 b/g/n/ac Wireless LAN
- Bluetooth 5.0 with BLE
- 1x SD Card
- 2x micro-HDMI ports supporting dual displays up to 4Kp60 resolution
- 2x USB2 ports
- 2x USB3 ports
- 1x Gigabit Ethernet port (supports PoE with add-on PoE HAT)
- 1x Raspberry Pi camera port (2-lane MIPI CSI)
- 1x Raspberry Pi display port (2-lane MIPI DSI)
- 28x user GPIO supporting various interface options:

  - Up to 6x UART
  - Up to 6x I2C
  - Up to 5x SPI
  - 1x SDIO interface
  - 1x DPI (Parallel RGB Display)
  - 1x PCM
  - Up to 2x PWM channels
  - Up to 3x GPCLK outputs

2.3 Software

- ARMv8 Instruction Set
- Mature Linux software stack
- Actively developed and maintained
  - Recent Linux kernel support
  - Many drivers upstreamed
  - Stable and well supported userland
  - Availability of GPU functions using standard APIs

4 Electrical Specification

Caution! Stresses above those listed in Table 2 may cause permanent damage to the device. This is a stress rating only; functional operation of the device under these or any other conditions above those listed in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Symbol	Parameter	Minimum	Maximum	Unit
VIN	5V Input Voltage	-0.5	6.0	V

Table 2: Absolute Maximum Ratings

Please note that VDD_IO is the GPIO bank voltage which is tied to the on-board 3.3V supply rail.

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Unit
V_{IL}	Input low voltage[a]	VDD_IO = 3.3V	-	-	TBD	V
V_{IH}	Input high voltage[a]	VDD_IO = 3.3V	TBD	-	-	V
I_{IL}	Input leakage current	TA = +85°C	-	-	TBD	μA
C_{IN}	Input capacitance	-	-	TBD	-	pF
V_{OL}	Output low voltage[b]	VDD_IO = 3.3V, IOL = -2mA	-	-	TBD	V
V_{OH}	Output high voltage[b]	VDD_IO = 3.3V, IOH = 2mA	TBD	-	-	V
I_{OL}	Output low current[c]	VDD_IO = 3.3V, VO = 0.4V	TBD	-	-	mA
I_{OH}	Output high current[c]	VDD_IO = 3.3V, VO = 2.3V	TBD	-	-	mA
R_{PU}	Pullup resistor	-	TBD	-	TBD	kΩ
R_{PD}	Pulldown resistor	-	TBD	-	TBD	kΩ

[a] Hysteresis enabled
[b] Default drive strength (8mA)
[c] Maximum drive strength (16mA)

Table 3: DC Characteristics

Pin Name	Symbol	Parameter	Minimum	Typical	Maximum	Unit
Digital outputs	t_{rise}	10-90% rise time[a]	-	TBD	-	ns
Digital outputs	t_{fall}	90-10% fall time[a]	-	TBD	-	ns

[a] Default drive strength, CL = 5pF, VDD_IO = 3.3V

Table 4: Digital I/O Pin AC Characteristics

5.1.1 GPIO Pin Assignments

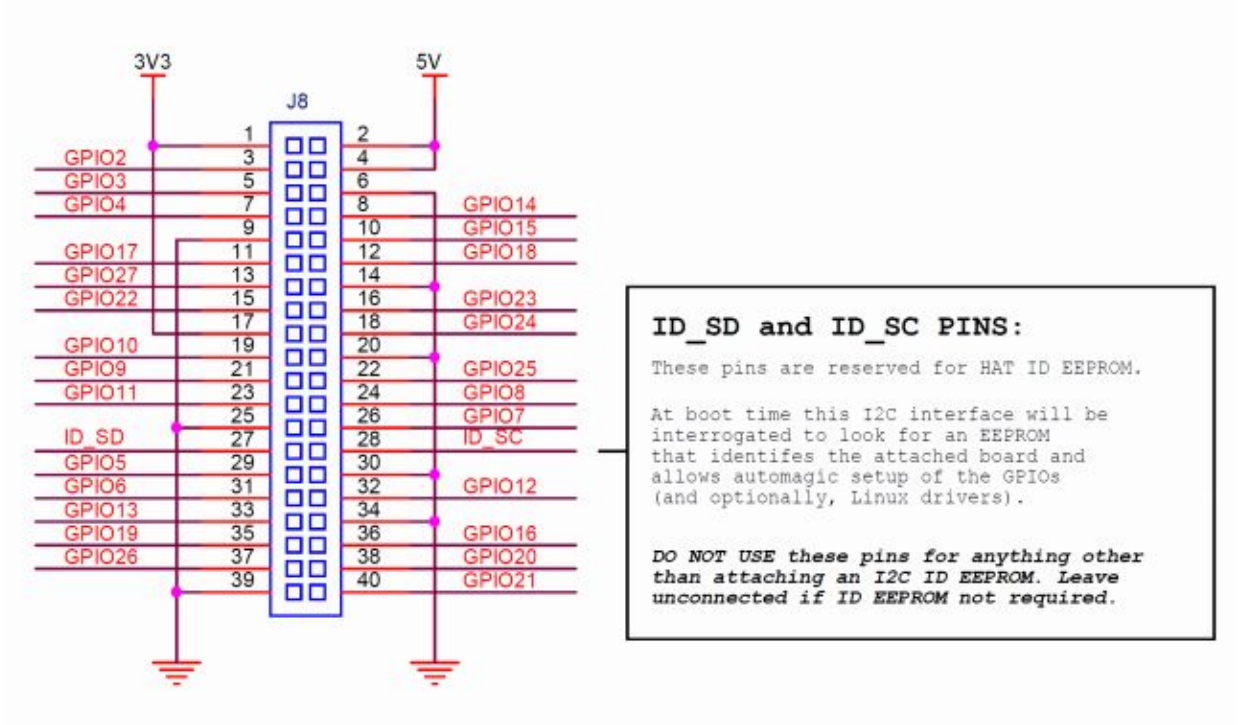
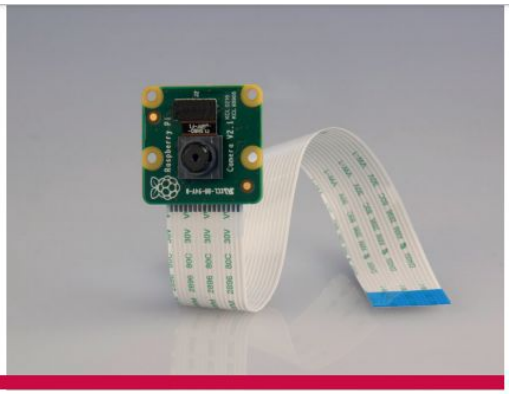


Figure 3: GPIO Connector Pinout

Camera Module :

Raspberry Pi



Camera Module

Product Name	**Raspberry Pi Camera Module**
Product Description	High Definition camera module compatible with all Raspberry Pi models. Provides high sensitivity, low crosstalk and low noise image capture in an ultra small and lightweight design. The camera module connects to the Raspberry Pi board via the CSI connector designed specifically for interfacing to cameras. The CSI bus is capable of extremely high data rates, and it exclusively carries pixel data to the processor.
RS Part Numer	**913-2664**

Specifications

Image Sensor	Sony IMX 219 PQ CMOS image sensor in a fixed-focus module.
Resolution	8-megapixel
Still picture resolution	3280 x 2464

Resolution	8-megapixel
Still picture resolution	3280 x 2464

Max image transfer rate	1080p: 30fps (encode and decode) 720p: 60fps
Connection to Raspberry Pi	15-pin ribbon cable, to the dedicated 15-pin MIPI Camera Serial Interface (CSI-2).
Image control functions	Automatic exposure control Automatic white balance Automatic band filter Automatic 50/60 Hz luminance detection Automatic black level calibration
Temp range	Operating: -20° to 60° Stable image: -20° to 60°
Lens size	1/4"
Dimensions	23.86 x 25 x 9mm
Weight	3g

LDR :

SUNROM TECHNOLOGIES
Your Source for Embedded Systems

Email: info@sunrom.com or sunrom@gmail.com
Visit us at http://www.sunrom.com

Document: Datasheet | Date: 28-Jul-08 | Model #: 3190 | Product's Page: www.sunrom.com/p-510.html

Light Dependent Resistor - LDR

Two cadmium sulphide(cds) photoconductive cells with spectral responses similar to that of the human eye. The cell resistance falls with increasing light intensity. Applications include smoke detection, automatic lighting control, batch counting and burglar alarm systems.

Applications

Photoconductive cells are used in many different types of circuits and applications.

Analog Applications

- Camera Exposure Control
- Auto Slide Focus - dual cell
- Photocopy Machines - density of toner
- Colorimetric Test Equipment
- Densitometer
- Electronic Scales - dual cell
- Automatic Gain Control – modulated light source
- Automated Rear View Mirror

Digital Applications

- Automatic Headlight Dimmer
- Night Light Control
- Oil Burner Flame Out
- Street Light Control
- Absence / Presence (beam breaker)
- Position Sensor

Electrical Characteristics

Parameter	Conditions	Min	Typ	Max	Unit
Cell resistance	1000 LUX 10 LUX	- -	400 9	- -	Ohm K Ohm
Dark Resistance	-	-	1	-	M Ohm
Dark Capacitance	-	-	3.5	-	pF
Rise Time	1000 LUX 10 LUX	- -	2.8 18	- -	ms ms
Fall Time	1000 LUX 10 LUX	- -	48 120	- -	ms ms
Voltage AC/DC Peak		-	-	320	V max
Current		-	-	75	mA max
Power Dissipation				100	mW max
Operating Temperature		-60	-	+75	Deg. C

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.

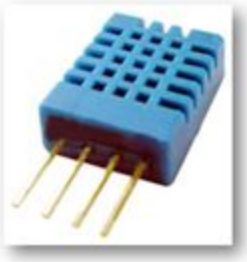
Applications

- Agriculture
- Landscape irrigation

Specifications

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

Humidity And Temperature Sensor (DHT11)



Each DHT11 element is strictly calibrated in the laboratory that is extremely accurate on humidity calibration. The calibration coefficients are stored as programmes in the OTP memory, which are used by the sensor's internal signal detecting process. The single-wire serial interface makes system integration quick and easy. Its small size, low power consumption and up-to-20 meter signal transmission making it the best choice for various applications, including those most demanding ones. The component is 4-pin single row pin package. It is convenient to connect and special packages can be provided according to users' request.

2. Technical Specifications:

Overview:

Item	Measurement Range	Humidity Accuracy	Temperature Accuracy	Resolution	Package
DHT11	20-90%RH 0-50 ℃	±5%RH	±2℃	1	4 Pin Single Row

Detailed Specifications:

Parameters	Conditions	Minimum	Typical	Maximum
Humidity				
Resolution		1%RH	1%RH	1%RH
			8 Bit	
Repeatability			±1%RH	
Accuracy	25℃		±4%RH	
	0-50℃			±5%RH
Interchangeability	Fully Interchangeable			
Measurement Range	0℃	30%RH		90%RH
	25℃	20%RH		90%RH
	50℃	20%RH		80%RH
Response Time (Seconds)	1/e(63%)25℃, 1m/s Air	6 S	10 S	15 S
Hysteresis			±1%RH	
Long-Term Stability	Typical		±1%RH/year	
Temperature				
Resolution		1℃	1℃	1℃
		8 Bit	8 Bit	8 Bit
Repeatability			±1℃	
Accuracy		±1℃		±2℃
Measurement Range		0℃		50℃
Response Time (Seconds)	1/e(63%)	6 S		30 S

TECHNICAL DATA MQ-135 GAS SENSOR

FEATURES

Wide detecting scope Fast response and High sensitivity
Stable and long life Simple drive circuit

APPLICATION

They are used in air quality control equipments for buildings/offices, are suitable for detecting of NH3,NOx, alcohol, Benzene, smoke,CO_2 ,etc.

SPECIFICATIONS

A. Standard work condition

Symbol	Parameter name	Technical condition	Remarks
Vc	Circuit voltage	5V±0.1	AC OR DC
V_H	Heating voltage	5V±0.1	ACOR DC
R_L	Load resistance	can adjust	
R_H	Heater resistance	33Ω±5%	Room Tem
P_H	Heating consumption	less than 800mw	

B. Environment condition

Symbol	Parameter name	Technical condition	Remarks
Tao	Using Tem	-10 -45	
Tas	Storage Tem	-20 -70	
R_H	Related humidity	less than 95%Rh	
O_2	Oxygen concentration	21%(standard condition)Oxygen concentration can affect sensitivity	minimum value is over 2%

C. Sensitivity characteristic

Symbol	Parameter name	Technical parameter	Ramark 2
Rs	Sensing Resistance	30KΩ-200KΩ (100ppm NH_3)	Detecting concentration scope 10ppm-300ppm NH_3 10ppm-1000ppm Benzene 10ppm-300ppm Alcohol
α (200/50) NH_3	Concentration Slope rate	≤0.65	
Standard Detecting Condition	Temp: 20 ±2 Humidity: 65%±5%	Vc:5V±0.1 Vh: 5V±0.1	
Preheat time	Over 24 hour		

D. Structure and configuration, basic measuring circuit

	Parts	Materials
1	Gas sensing layer	SnO_2
2	Electrode	Au
3	Electrode line	Pt
4	Heater coil	Ni-Cr alloy
5	Tubular ceramic	Al_2O_3
6	Anti-explosion network	Stainless steel gauze (SUS316 100-mesh)
7	Clamp ring	Copper plating Ni
8	Resin base	Bakelite
9	Tube Pin	Copper plating Ni

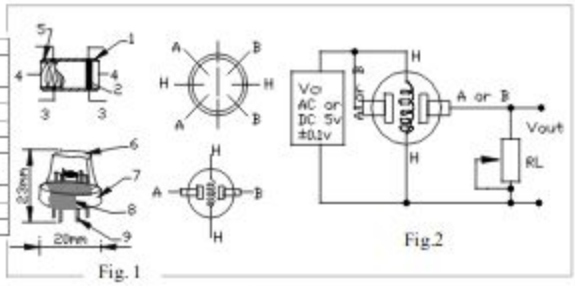
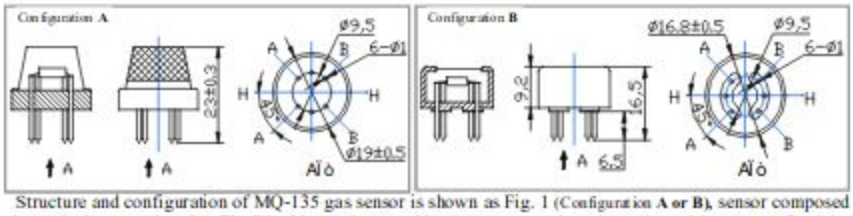


Fig. 1 Fig.2



Structure and configuration of MQ-135 gas sensor is shown as Fig. 1 (Configuration **A or B**), sensor composed by micro AL_2O_3 ceramic tube, Tin Dioxide (SnO_2) sensitive layer, measuring electrode and heater are fixed into a crust made by plastic and stainless steel net. The heater provides necessary work conditions for work of sensitive

MQ135 Semiconductor Sensor for Air Quality Control

Sensitive material of MQ135 gas sensor is SnO_2, which with lower conductivity in clean air. When the target combustible gas exist, The sensor's conductivity is more higher along with the gas concentration rising. Please use simple electrocircuit, Convert change of conductivity to correspond output signal of gas concentration.

MQ135 gas sensor has high sensitity to Ammonia, Sulfide and Benze steam, also sensitive to smoke and other harmful gases. It is with low cost and suitable for different application.

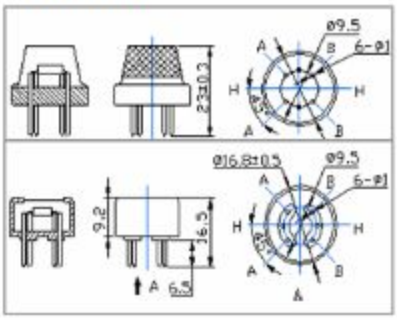
Character

* Good sensitivity to Harmful gases in wide range
* High sensitivity to Ammonia, Sulfide and Benze
* Long life and low cost
* Simple drive circuit

Application

* Domestic air pollution detector
* Industrial air pollution detector
* Portable air pollution detector

Configuration

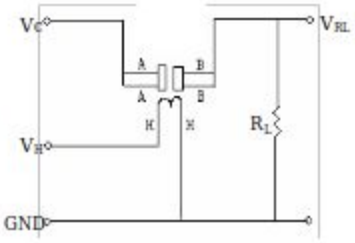


Technical Data

Model No.			MQ135
Sensor Type			Semiconductor
Standard Encapsulation			Bakelite (Black Bakelite)
Detection Gas			Ammonia, Sulfide, Benze steam
Concentration			10-10000ppm (Ammonia, Benze, Hydrogen)
Circuit	Loop Voltage	V_C	≤24V DC
	Heater Voltage	V_H	5.0V±0.2V AC or DC
	Load Resistance	R_L	Adjustable
Character	Heater Resistance	R_H	31Ω±3Ω（Room Tem.）
	Heater consumption	P_H	≤900mW
	Sensing Resistance	R_s	2KΩ-20KΩ(in 100ppm NH_3)
	Sensitivity	S	Rs(in air)/Rs(100ppm NH_3)≥5
	Slope	α	≤0.6(R_{300ppm}/R_{100ppm} NH_3)
Condition	Tem. Humidity		20℃±2℃; 65%±5%RH
	Standard test circuit		Vc:5.0V±0.1V; V_H: 5.0V±0.1V
	Preheat time		Over 48 hours

$Ps=Vc^2 \times Rs/(Rs+RL)^2$

Basic test loop



The above is basic test circuit of the sensor. The sensor need to be put 2 voltage, heater voltage(VH) and test voltage(VC). VH used to supply certified working temperature to the sensor, while VC used to detect voltage (VRL) on load resistance （RL） whom is in series with sensor. The sensor has light polarity, Vc need DC power. VC and VH could use same power circuit with precondition to assure performance of sensor. In order to make the sensor with better performance, suitable RL value is needed:

Power of Sensitivity body(Ps):

APPENDIX B - PROGRAMMING REVIEW

	C	Java	Python
Object Oriented	No	Yes	Yes(but optional)
Functional	No	No	Yes
Readability	Difficult	Difficult	Easy
Ease of Typing	Strongly Typed	Strongly Typed	Dynamic
Code Length	5-10 times greater than python	3-5 times greater than python	Small and manageable
Header Files	Yes	No (import libraries)	No(import libraries)
Syntax Complexity	Yes	Yes	None
Portability	Platform dependent	Platform independent	Platform independent
Multiple Inheritance	Not Supported	Partially Supported	Supported
Compilation Process	Compilation Language	Compilation Language	Interpretation Language
Support to pointers in memory management	Yes	No use	Not Supported