

# **INTERNET OF THINGS (CS667A)**

## **Assignment-2**

**FARMER IRRIGATION SYSTEM**

**GROUP\_NAME = AUTOMATION**

### **PARTICIPANTS:**

*MOHIT SINGH (22111402) -20%(Contribution)*

*KOMAL YADAV (22111031) -20%(Contribution)*

*AKASH SHRIWAS (22111006) -20%(Contribution)*

*RAHUL AGARWAL ( 22111403) -20%(Contribution)*

*ABHINAV KURUMA ( 22111401) -20%(Contribution)*

## **1. GETTING STARTED WITH RASPBERRY PI**

- Download the Raspbian operating system from the official website link <https://www.raspberrypi.com/software/> and install it on the memory card.
- Connect the Raspberry Pi via ethernet to the PC to enable the interface of the Pi. Here we will make a bridge from ethernet/Wi-Fi to enable the display of the Raspberry Pi.
- Install putty software for enabling an interface to the localhost to the VNC viewer.
- Now open/install VNC Viewer to get the graphical user interface of the Raspberry Pi.

## **2. CONNECTING SENSOR TO THE RASPBERRY PI**

### **DHT11:**

The **DHT11** is a commonly used **Temperature and humidity sensor** that comes with a dedicated NTC to measure temperature and an 8-bit microcontroller to output the values of temperature and humidity as serial data.

DHT11 Pinout Configuration:

No:	Pin Name	Description
<b>For DHT11 Sensor</b>		
1	Vcc	Power supply 3.5V to 5.5V

2	Data	Outputs both Temperature and Humidity through serial Data
3	NC	No Connection and hence not used
4	Ground	Connected to the ground of the circuit

#### For DHT11 Sensor module

1	Vcc	Power supply 3.5V to 5.5V
2	Data	Outputs both Temperature and Humidity through serial Data
3	Ground	Connected to the ground of the circuit

## 2.1: Attaching a DHT11 sensor to a Raspberry Pi

Physical Connection-->>

We have connected the data pin from GPIO pin 17 to get the output, vcc to the 3.3v pin, and gnd to the ground pin.

Software-->

Installing the dht11 library from git clone repository link = <https://github.com/adafruit/DHT-sensor-library.git>

The programme to display the value of the sensors, which is mentioned below:

```

import RPi.GPIO as GPIO
import dht11
# initialize GPIO
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.cleanup()
# read data using pin 14
instance = dht11.DHT11(pin = 17)
result = instance.read()
if result.is_valid():
    print("Temperature: %-3.1f C" % result.temperature)
    print("Humidity: %-3.1f %% " % result.humidity)
else:
    print("Error: %d" % result.error_code)

```

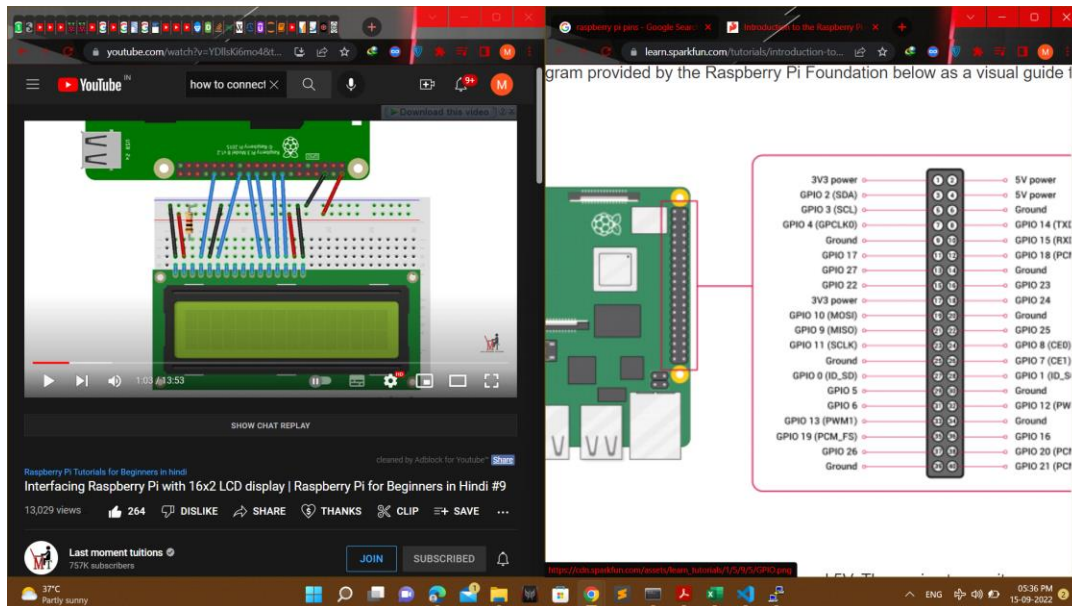
## ***2.2 Connecting a 20 X 4 LCD display to the Raspi***

Physical Connection-->>

The LCD display has 16 pins, out of which we have not used 7,8,9,10 data pins.

Here we have used 4,6,11,12,13,14 pins as a data pin of the LCD display and mapped those pins to the GPIO\_PINS ==12,7,8,25,17,23 of the Raspberry Pi.

We have made a connection as shown in the fig below.



Software Connection--->

To print the data on the LCD display, we have used the Adafruit char library. The GitHub repository for the following is linked at [https://github.com/adafruit/Adafruit\\_Python\\_CharLCD.git](https://github.com/adafruit/Adafruit_Python_CharLCD.git)

### 3. **ML MODEL: TRAINING & TESTING INFORMATION**

#### 3.1 *The libraries used in the models are*

*import numpy as np*

*from sklearn.model\_selection import train\_test\_split*

*import sklearn*

*from sklearn.svm import LinearSVC*

```
from pandas import read_csv  
import pandas as pd
```

### **3.2 WORKING OF THE MODEL**

First, we have collected the data given using the read\_csv option.

The data has 3 columns: Humidity, Temperature, and Water Flow. With this data, we have to calculate water flow percentage by giving the parameters of humidity and temperature.

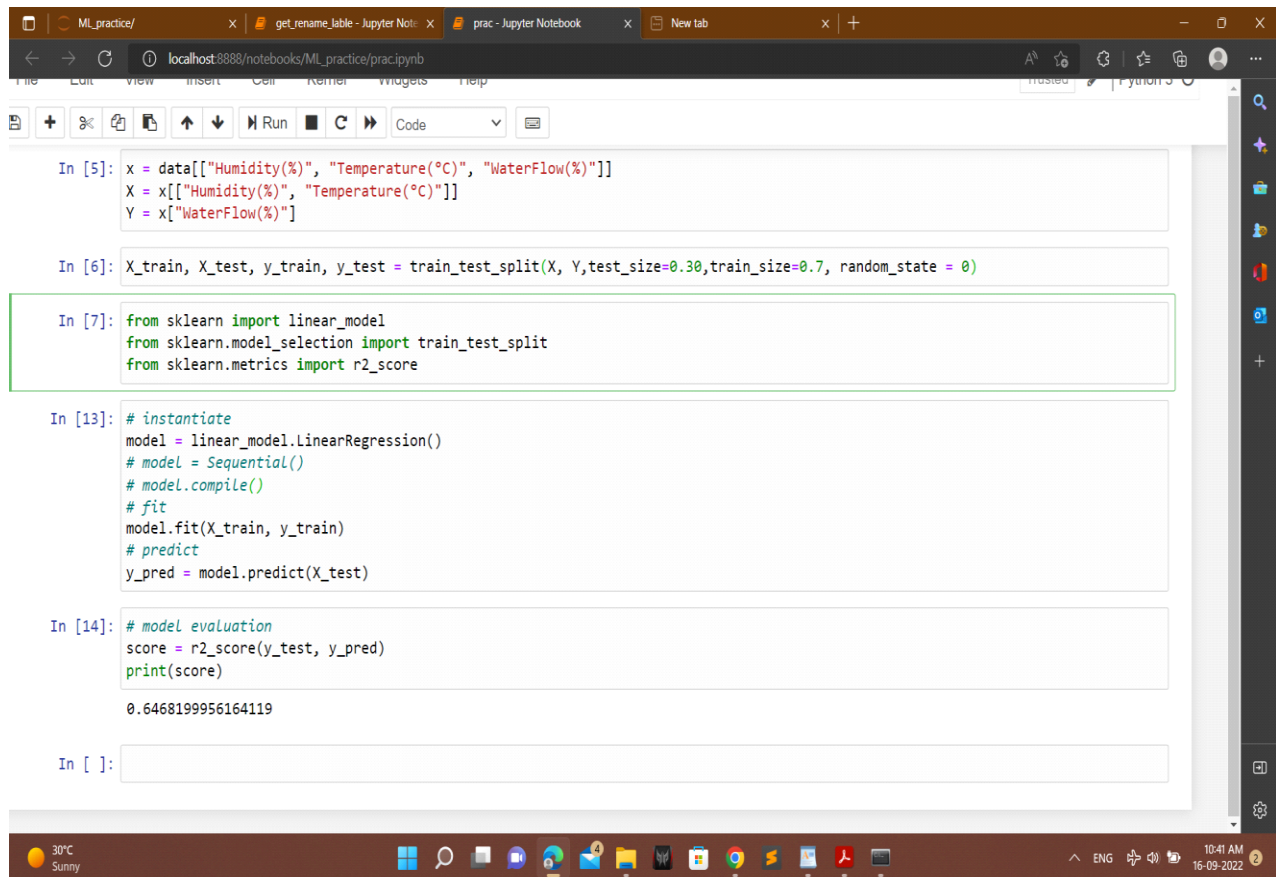
We have separated [humidity, temperature] and [waterflow%] columns.

Then we used test train data from the sklearn.model\_selection library.

To test, train, and predict the data, we used three libraries, namely

```
from sklearn import linear_model  
from sklearn.model_selection import train_test_split  
from sklearn.metrics import r2_score
```

The below code is implemented to find the accuracy of the model.



```
In [5]: x = data[["Humidity(%)", "Temperature(°C)", "WaterFlow(%)"]]
X = x[["Humidity(%)", "Temperature(°C)"]]
Y = x["WaterFlow(%)"]

In [6]: X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.30, train_size=0.7, random_state = 0)

In [7]: from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score

In [13]: # instantiate
model = linear_model.LinearRegression()
# model = Sequential()
# model.compile()
# fit
model.fit(X_train, y_train)
# predict
y_pred = model.predict(X_test)

In [14]: # model evaluation
score = r2_score(y_test, y_pred)
print(score)

0.6468199956164119

In [ ]:
```

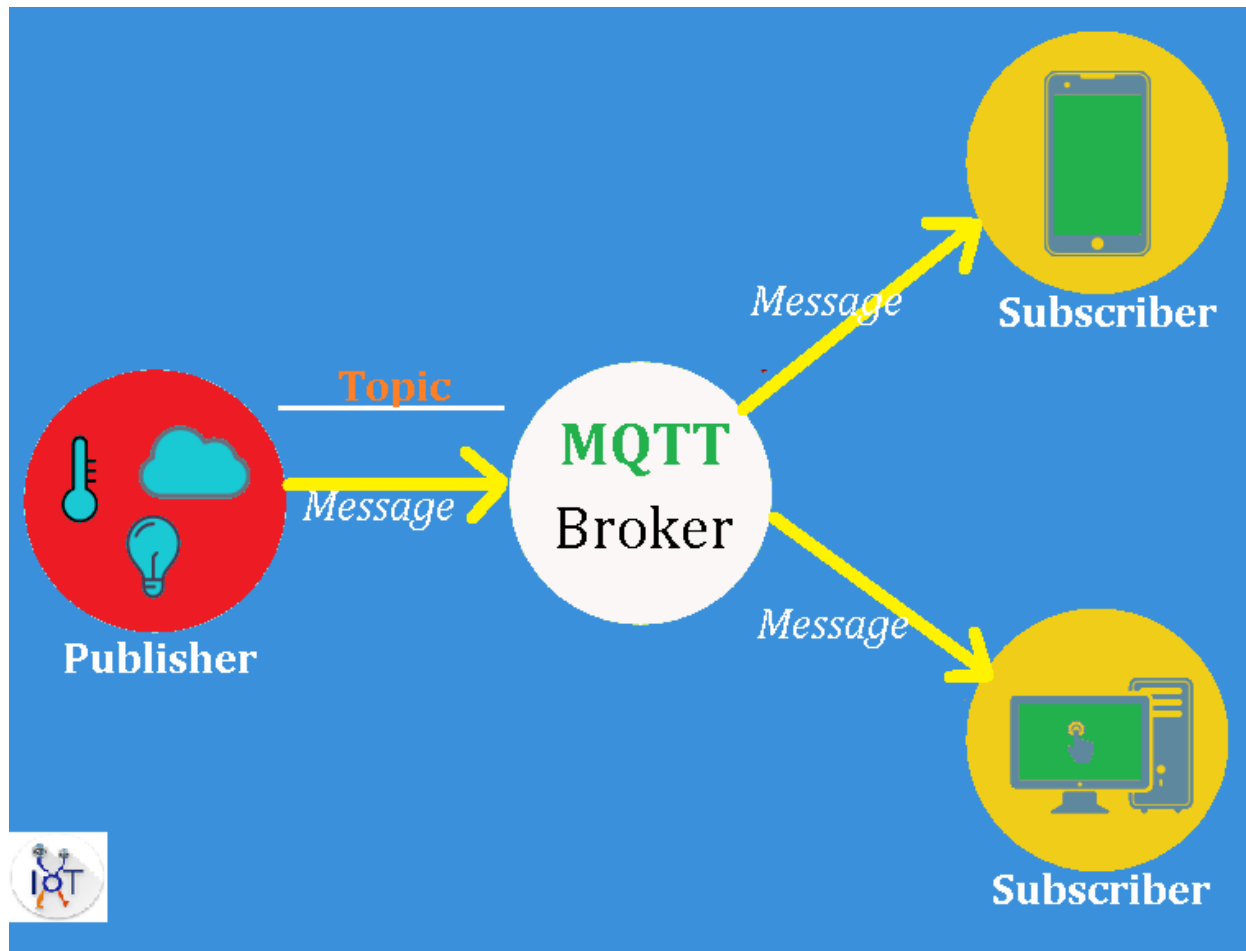
We have finally received an accuracy of 64.6%.

We also tried to implement the layers in the model, but in the end, we were unable to achieve an accuracy of more than 66%.

### **MQTT (Message Queuing Telemetry Transport) Protocol:**

It is a very simple protocol based on the publish-subscribe model. It allows us to send messages on a topic via broker. It is a machine-to-machine IOT communication protocol. In MQTT, communication involves MQTT clients and MQTT brokers, where MQTT clients (publishers) publish a message to an MQTT broker, and other MQTT clients (subscribers) subscribe to messages they want to receive. In this implemented irrigation system, we are using "test.mosquitto.org" as a broker, the Raspberry pi as a publisher, and our system as a subscriber, in which we are receiving the temperature and humidity values in our system from the Raspberry pi, and eventually to display the predicted water percentage on the LCD, we are using our system as a publisher, the Raspberry pi as a subscriber, and the same "test.mosquitto.org" as a broker. We are using two different topics for these two cases. The architecture of the MQTT protocol can be illustrated with the help of the below mentioned diagram:





**Fig: MQTT Architecture**

### **Required installations:**

1. Mosquitto: As a Message broker
2. On the Raspberry Pi terminal, we executed the below mentioned commands:
  - git clone  
<https://github.com/eclipse/paho.mqtt.python>
  - cd paho.mqtt.python
  - python3 setup.py install
  - pip3 install paho-mqtt

As per our implementation, we have created three python files to establish the required communication between the system and the Raspberry Pi. The use of three different python files is illustrated below:

- subscriber.py: This file is present on our system (laptop), where in the first case we are acting as a subscriber, and in the second case we are acting as a publisher. This file contains the below mentioned code:

```
1. import paho.mqtt.client as mqtt
2. def on_connect(client, userdata, flags, rc):
3.
4.     print(f'Connected with result code {rc}')
5.     client.subscribe("raspberrypi/topic")
6.
7. def on_message(client, userdata, msg):
8.     print(f'{msg.payload}')
9.
10.    publish(client,msg)
11.
12. def publish(client,msg):
13.
14.     client.publish("topic",payload=f'{msg.payload}', qos=0, retain=False)
15.
16. client_s = mqtt.Client()
17. client_s.on_connect = on_connect
18. client_s.on_message = on_message
19. client_s.enable_bridge_mode()
20.
21. client_s.will_set('raspberrypi/status', b'{"status": "Off"}')
22.
23. client_s.connect("test.mosquitto.org", 1883, 60)
24.
25. client_s.loop_forever()
26.
```

- publisher.py: This file is present on our Raspberry Pi, where we are publishing the temperature and humidity values. This file contains the below mentioned code:

```
import paho.mqtt.client as mqtt
import time
import RPi.GPIO as GPIO
import dht11

# initialize GPIO
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.cleanup()

# read data using pin 14
instance = dht11.DHT11(pin = 17)

def on_connect(client, userdata, flags, rc):
    print(f"Connected with result code {rc}")

client = mqtt.Client()
client.on_connect = on_connect
client.connect("test.mosquitto.org", 1883, 60)

while True:
    result = instance.read()
    client.publish('raspberrypi/topic', payload=f" temp:{result.temperature} humidity:{result.humidity} ", qos=0,
retain=False)
    print(f"send temp:{result.temperature} humidity:{result.humidity} to raspberrypi/topic")
    time.sleep(5)

client.loop_forever()
```

- subscriber.py: This file is also present on our Raspberry Pi where we are acting as a subscriber to receive the predicted value of water percentage from the publisher with the help of an ML model. This file contains the below mentioned code:

```
import paho.mqtt.client as mqtt

def on_connect(client, userdata, flags, rc):
    print(f'Connected with result code {rc}')
    client.subscribe("topic")

def on_message(client, userdata, msg):
    print(f'{msg.topic} {msg.payload}')

client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message

client.will_set('raspberrypi/status', b'{"status": "Off"}')
client.connect("test.mosquitto.org", 1883, 60)
client.loop_forever()
```

Eventually, to display the predicted water percentage in LCD, which is calculated via the ML Model, we have designed the code as shown below with the required configuration.

```
import time
import Adafruit_CharLCD as LCD

# Raspberry Pi pin configuration:
lcd_rs      = 12
lcd_en      = 7
lcd_d4      = 8
lcd_d5      = 25
lcd_d6      = 17
```

```
lcd_d7      = 23
lcd_backlight = 4

lcd_columns = 20
lcd_rows    = 4

# Initialize the LCD using the pins above.
lcd = LCD.Adafruit_CharLCD(lcd_rs, lcd_en, lcd_d4, lcd_d5, lcd_d6, lcd_d7,
                           lcd_columns, lcd_rows, lcd_backlight)

lcd.message('Hello\nworld!')

time.sleep(5.0)

# Demo showing the cursor.
lcd.clear()
lcd.show_cursor(True)
lcd.message('Show cursor')

time.sleep(5.0)

# Demo showing the blinking cursor.
lcd.clear()
lcd.blink(True)
lcd.message('Blink cursor')

time.sleep(5.0)

# Stop blinking and showing cursor.
lcd.show_cursor(False)
lcd.blink(False)

# Demo scrolling message right/left.
lcd.clear()
message = 'Scroll'
lcd.message(message)
for i in range(lcd_columns-len(message)):
    time.sleep(0.5)
    lcd.move_right()
for i in range(lcd_columns-len(message)):
    time.sleep(0.5)
    lcd.move_left()

# Demo turning backlight off and on.
lcd.clear()
```

```
lcd.message('Flash backlight\nin 5 seconds...')
time.sleep(5.0)
# Turn backlight off.
lcd.set_backlight(0)
time.sleep(2.0)
# Change message.
lcd.clear()
lcd.message('Goodbye!')
# Turn backlight on.
lcd.set_backlight(1)
```

## **Conclusion:**

Aravind, as mentioned in the problem statement, is good at farming but has little experience building machine learning models or IoT systems. So, we have built an irrigation system for Aravind with the help of which he gets to know how much water (in percentage) should be supplied to his farm in different environmental conditions.