# Assignment 2(CS667A)

# Farm Irrigation System

## TheToaster

Abhay Kumar Dwivedi (22111001)
Hrithik Lal (22111027)
Sanket Kale (22111052)
Saqeeb (22111053)
Shubham Rathore (22111054)

**Report Overview** :

The objective of this report is to build IOT based system for controlling water supply using a Machine Learning model that takes data from Temperature and Humidity sensors to predict the amount of water supplied to farm. And used Respberry Pi 3 Model B,DHT11(temparature and humidity sensor),LCD to design the circuit.We used MQTT protocol for communcation between Raspberry Pi and a system on which we are running machine learning model.

## Steps to setup Raspberry PI

- First we installed RaspberryPI OS in the sd card.
- Connected RaspberryPI with the laptop using LAN and gave it power supply.
- And then, we installed Putty And Connected Raspberry PI with Putty by entering username and password.
- And Installed VNC Viewer to control RaspberryPI OS.

## Steps to connect the Sensor

- Sensor which is used is DHT11 which measures temperature and Humidity.
- The sensor has three pins Vcc, Data and Ground.
- We have connected Vcc of sensor to Raspberry PI 5V, Data of sensor to GPIO4 of Raspberry PI and Ground of sensor to Ground of Raspberry PI.

## Steps for Getting Readings from the Sensor

- We have installed Adafruit_DHT library.

- Then we wrote a program which uses read_retry function from the Adafruit_DHT module to read the inputs given by sensor.

## Connection of LCD to Respberry Pi

First we have connected the pins of LCD to Respberry Pi And then, installed and import the library "ADAFRUIT_CharLCD" and run the code to display on LCD.
Pins connection as follows :

- LCD pin 1, 3, 5, 16 is connected to the ground of Respberry Pi.
- LCD pin 2 is connected to the 5V pin of Respberry Pi.
- LCD pin 4 is the Register Select and it is connected to GPIO 26 of the Raspberry pi.
- LCD pin 6 is connected to GPIO 19 which makes LCD to run instructions.
- LCD pin 11 to 14 is connected to GPIO 13, GPIO 06, GPIO 05, GPIO 11 to present the output in the lcd screen.
- LCD pin 15 is for the LCD backlight and it is connected to the 3.3V pin of Raspberry PI for brightness.

## Communication using MQTT

We used 'broker.emqx.io' MQTT broker and used port '1883' for communication between Raspberry Pi and a system.
Firstly,Raspberry Pi takes data from sensor and sends to MQTT broker which then sends to system containing ML model for prediction of water flow percentage.Similarly,after computing prediction,our system sends output to MQTT broker which then sends to Raspberry Pi.
While sending sensor data,Raspberry Pi is acting as publisher and system is acting as subscriber.

While taking predicted output from ML model from system,the system acts as publisher and Raspberry Pi acts as subscriber.
We used python's 'paho-mqtt' library for establishing and executing communication using mqtt protocol.

## ML Model Architecture

We have created a Sequential model using the Tensorflow library which is the linear model and can be used to find the weights of linear equation.
The linear equation used in finding the water flow from humidity and temperature is as follows :

$$\mathbf{W_0X_0+W_1X_1+b}$$

Where,

$X_0$ = Humidity in %

$X_1$ = Temperature in $^o$C

**Data analysis :**

We have observed that for higher value of $X_1$ water flow should be more and for lower value of $X_0$ water flow should be less.

**Perceptron Layers Description :**

Input layers : Two features $X_0$ and $X_1$ have been used as input layer.
Hidden layers : Two hidden layers have been used in this model, first hidden layer uses 4 nodes, these nodes will take features and their corresponding weights and give the updated weights according to output to next hidden layer.
Second hidden layer is also uses 4 nodes and take the features and weights from previous layer and provide updated weights to Output layer.
Each hidden layer uses Relu activation function to avoid negative output which works as follows,

$$f(W.X+b) = \max(0,W.X+b)$$

Output Layer : This layer will provide final weights for our linear equation.

**Optimizer used :**

We have used Adam optimizer for better accuracy. This optimizer uses stochastic gradient descent but with different learning rates for each network weights which can provide more accuracy.
In our model Adam optimizer performed better than stochastic optimizer.

**Initial Learning Rate taken :** 0.001
**Number of epochs used :** 400
**Weights provided by each node of Second Hidden layer to Output layers :** 0.36102438, 0.5336438 ,-0.96419895, 0.9594198 .
**Evaluation metrics :**
R2 score is used as evaluation metrics which uses variance between the predicted and actual data to find the accuracy.
In our model R2 score is around 0.70 .

## Contribution of Team members

Abhay Kumar Dwivedi (22111001) - 20%
Hrithik Lal (22111027) - 20%
Sanket Kale (22111052) - 20%
Saqeeb (22111053) - 20%
Shubham Rathore (22111054) - 20%

**Code to train model and send water flow percentage**

```
import random
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras import activations
import matplotlib.pyplot as plt
from paho.mqtt import client as mqtt
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from keras.models import load_model
import time



i = -1

# Reading CSV file

data = pd.read_csv('IOT_Assignment_2_data_regression
        _sensor_range.csv')
X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values

# Splitting data
X_train, X_test, y_train, y_test = train_test_split(X, y,
        test_size = 0.3, random_state = 0)

# Loading model
ann = load_model('Final_IOT_Model.h5')

# ann = tf.keras.models.Sequential()
# ann.add(tf.keras.layers.Dense(units=4,
        activation='relu'))
# ann.add(tf.keras.layers.Dense(units=4,
        activation='relu'))
```

```
# ann.add(tf.keras.layers.Dense(units=1))
# custom_optimizer=tf.keras.optimizers
        .Adam(learning_rate=0.001)
# ann.compile(optimizer = custom_optimizer,
        loss = 'mean_squared_error')
# ann.fit(X_train, y_train, epochs = 100)
y_pred = ann.predict(X_test)
R2 = r2_score(y_test, y_pred,
        multioutput='variance_weighted')
print(R2)

# Function to execute on connection
def on_connect(client, userdata, flags, rc):
    print(f"Connected with result code {rc}")

    # Subscribe to topic "raspberry/topic"
    client.subscribe("raspberry/topic")

def on_message(client, userdata, msg):
    global i
    if(i == -1):
        i = 0
        return

    # Taking out temp and humidity in list
    li = msg.payload.decode().split()

    # Initialising temp and humid with there
        respective values
    temp = int(li[0])
    humid = int(li[1])

    # Testing output
    print('Temp : ', temp, ' Humid : ', humid)

    # Predicting the water flow percentage
    x = str(ann.predict([[temp, humid]])[0][0])

    # Testing output
    print('Prediction : ', x)

    # Publishing the waterflow
    client.publish('raspberry/model', payload = x,
        qos=0, retain=True)

    i += 1
    print("sending water flow")
    print()

# setting client
client = mqtt.Client()

# setting callback methods
client.on_connect = on_connect
client.on_message = on_message


# connecting to broker
client.connect("broker.emqx.io", 1883, 60)

client.loop_forever()
```

**Code to send temp,humidity data and show water flow percentage on LCD**

```
import paho.mqtt.client as mqtt
import time
from time import sleep
from Adafruit_CharLCD import Adafruit_CharLCD
import Adafruit_DHT
import random
import math
DHT11=Adafruit_DHT.DHT11



# Initialising Lcd and giving out the pin number
    in the arguements
lcd = Adafruit_CharLCD(rs = 26, en = 19, d4 = 13,
        d5 = 6, d6 = 5, d7 = 11, cols = 20, lines=3)

# Clearing out the lcd screen
lcd.clear()

loop = -1

# On connection with broker execute the
    following function
def on_connect(client, u, f, r):
    print(f"Connected with result code {r}")

    #Subscribing to topic "raspberry/model"
    client.subscribe("raspberry/model")

# On getting a message execute the following function
def on_message(client, userdata, msg):
    global loop
    if loop == -1:
        loop = 0
        return
    # Decoding payload from the received message
    flow = float(msg.payload.decode())

    # Converting flow to 2 decimal precision and
        then string
    flow = '{0:.2f}'.format(flow)
    flow = str("Flow : " + flow + " %")

    # Printing flow to lcd
    lcd.message(flow)

    # Printing for testing purpose
    print('Flow : ', flow)

    # Waiting
    sleep(1)

    # Setting cursor to next row, 0th column
    lcd.show_cursor(True)
    lcd.set_cursor(0, loop+1)
    loop += 1

# setting client
client = mqtt.Client()

# setting callback methods
client.on_connect = on_connect
client.on_message = on_message

# connecting to broker
client.connect("broker.emqx.io", 1883, 60)


for i in range(3):
    # Taking sensor reading
    temp,humid=Adafruit_DHT.read_retry(DHT11,21)

    #temp = random.randint(20, 30)
    #humid = random.randint(40, 70)

    # Forming payload
    p = str(temp)+ ' ' +str(humid)

    # Publishing message
    client.publish('raspberry/topic', payload=p,
        qos=0, retain=False)

    # Testing output
    print(f"send {i} to raspberry/topic")


client.loop_forever()
```