IoT Based Irrigation System Using ML

TABLE OF CONTENTS -

1. Problem Statement

This project focuses on a Smart Irrigation system that makes use of an IoT network of sensors that collect data from the environment, specifically humidity and temperature values, and a Machine Learning (ML) model, which is trained with this data to make predictions on the percentage of water required for farming. This whole idea helps in automating the irrigation process with minimal human intervention.

2. Our Approach

The approach used in the project is creating an IoT network of sensors that collects the data from the environment, performs pre-processing using Raspberry Pi to calculate the percentage of water to be given to the plants and transfers this information to the LCD screen via MQTT communication protocol.

Now, to make this automation more intelligent, an ML model is used to take decisions from the data provided. The data collected from the environment i.e., humidity and temperature values are fed to the training model. This training model architecture consists of a two layer perceptron model using activation function, ReLu. Now, this trained model displays the percentage of water that needs to be pumped out on the LCD screen.

3. Hardware and Software requirements

3.1 Hardwares used

- Raspberry Pi
- Jump wires
- Breadboard
- 20x4 LCD
- Humidity and temperature sensor(DHT 11)

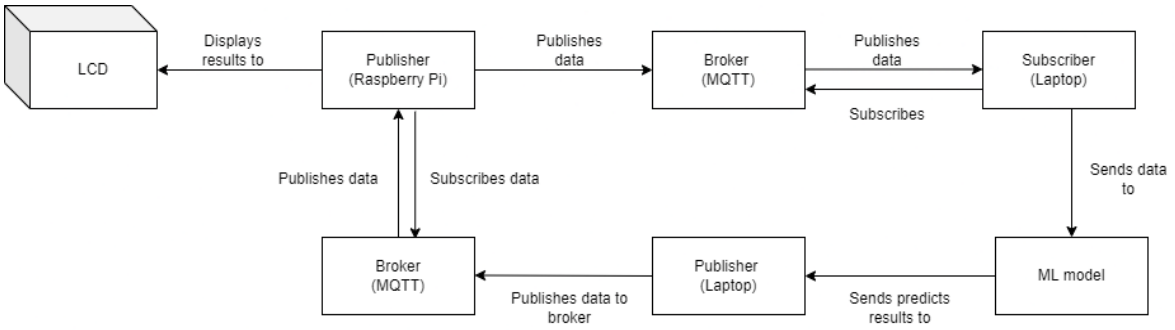
3.2 Softwares used

- Python
- Paho MQTT
- Google colaboratory

4. Hardware model using IoT

4.1 MQTT Communication protocol

The protocol MQTT(MQ Telemetry Transport) is a light weight and publish/subscribe based communication protocol. In this protocol, the sender acts as a publisher and the receiver acts as a subscriber. This communication happens through broker. The publisher publishes the data as a topic (here topic name should be unique) and receiver/subscriber subscribes to this topic to receive the data. In the project, since the humidity & temperature sensor sends data from time to time, this protocol is best suited.



Flow of MQTT communication protocol

4.2 LCD connection

The LCD to Raspberry Pi connections are as follows:

Raspberry PI	LCD display	
Ground	Ground	1
5V	VCC	2
Ground	Contrast	3
GPIO 26	RS	4
Ground	R/W	5
GPIO 19	Enable	6

GPIO 13	LCD D4	11
GPIO 6	LCD D5	12
GPIO 5	LCD D6	13
GPIO 11	LCD D7	14
5V	LED +	15
Ground	LED -	16

4.3 Implementation

In the project, the humidity and temperature sensors on the Raspberry Pi board senses the values from the environment. Here, the Raspberry Pi acts as a publisher and publishes the data to the broker via MQTT protocol. Now, the laptop subscribes to the broker and the broker sends the data to the laptop. The data collected by the laptop is fed to the two layer perceptron model and the model uses ReLu activation function to predict the results of the amount of water required based on humidity and temperature values in the data set.

These results are sent to the laptop and the laptop publishes these results to the broker using MQTT protocol. The Raspberry Pi then subscribes to the broker and gets the prediction results from the broker so as to display on the LCD display.

Below the hardware implementation of the irrigation system is included.

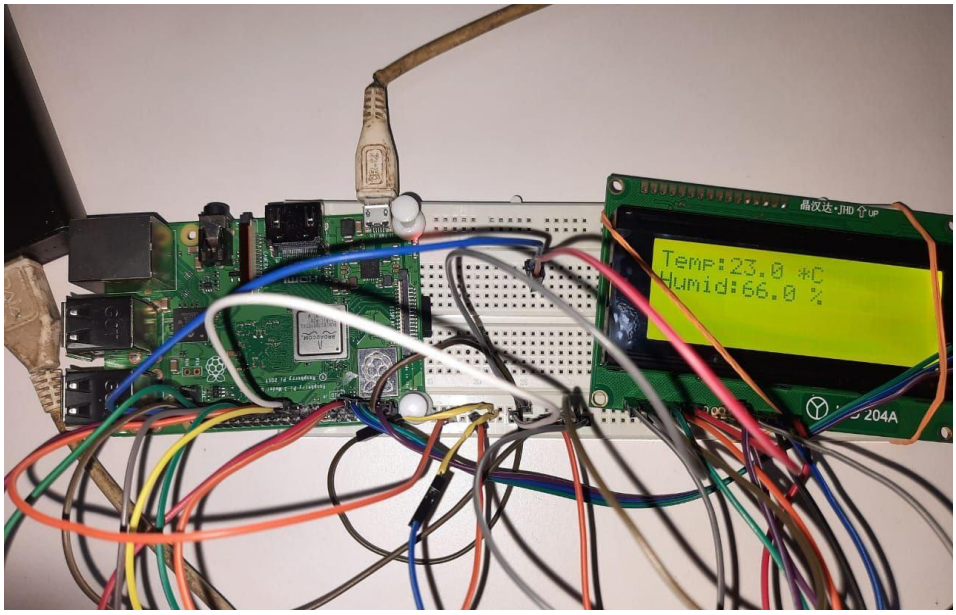


Fig.1 Breadboard connection with Raspberry Pi and LCD with output.

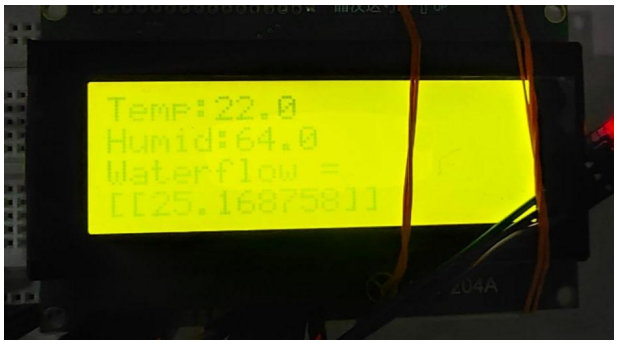


Fig. 2 Final waterflow output in the LCD.

5. ML model

5.1 Architecture

A multilayer perceptron Model (MCP) is a fully connected feedforward artificial neural network (ANN). It consists of at least three layers, one input layer, one hidden layer and one output layer. The learning happens in each perceptron by changing connection weights using activation function, after each piece of data is processed based on the amount of error in the output compared to expected result.

ReLu (Rectified linear Activation function) is a piecewise linear function which outputs input directly, if input is greater than 0, and output is zero otherwise. This activation function is used

most commonly because it is easier to train, gives better performance and also overcomes vanishing gradient problems.

5.2 Epochs used for training

The term epoch in machine learning refers to the number of passes the entire training data set takes to complete a machine learning algorithm. In ML, a very large data set is divided into batches and these can be trained using an ML model without any issues on memory overflow. Training all these batches using the model exactly once becomes an epoch. This increases the accuracy and convergence of data.

The number of epochs used in this ML model is 500 and the batch size is 16.

5.3 Evaluation metrics

The optimization algorithm used is RMSprop. RMSprop is calculated by finding a discounted average of the square of gradients and dividing the gradient by the root of the average. The arguments to this function are rho, learning_rate, epsilon, momentum, name and **kwargs. The values used in this ML model are learning_rate = 0.03, rho = 0.9, epsilon = $1e^{-7}$.

The loss is calculated using MSE (Mean Squared Error). It takes the distances from the data points to the regression line and squares them, thereby indicating the closeness of the regression line to the points. The result of the ML model is:

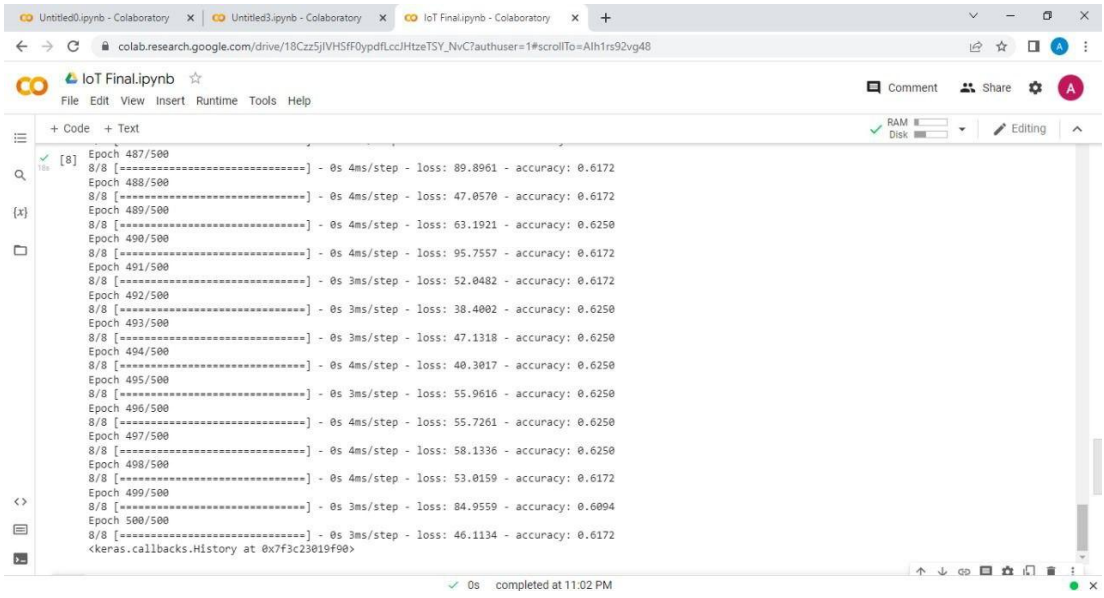


Fig.3 Training accuracy and loss

6. References

- https://keras.io/api/optimizers/rmsprop/
- https://machinelearninggeek.com/multi-layer-perceptron-neural-network-using-python/
- https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
- https://medium.com/coinmonks/how-to-connect-lcd-display-to-your-raspberry-pi-3-iot-68d990843e0d
- https://pypi.org/project/paho-mqtt/

7. Members' Contribution

All the members of the group contributed equally in the project. Below is the percentage contribution of each member considering 100% as the total work done -

- Anjali Manoj - 20%
- Ashee Jain - 20%
- Atanu Kar - 20%
- Deepak Mathur - 20%
- Srujana Sabbani - 20%