Introduction to IoT and Its Industrial Applications (CS698T)
Indian Institute of Technology Kanpur
Assignment 2

Member Names: Divyansh Bisht, Manu Shukla
Member Emails: dbisht21@iitk.ac.in, manushukla21@iitk.ac.in
Member Roll Numbers: 21111027, 21111040
Date: November 3, 2021

REPORT

1 Introduction

Aravind is an ambitious farmer who wants to improve the yields from his farmland. While looking for ways to improve the yields, he came across tech buzzwords like the Internet of Things and Machine Learning. He wants to try them out to build an intelligent irrigation system for his field.

The irrigation system he wants to build contains servo motors that control the water supply to the farm, based on the present humidity, temperature levels in the farm. Using sensor data, Aravind wants to build a machine learning model that predicts how much water (in percentage) should be supplied to his farm. Aravind also does not want to water the plants during nighttime.

However, Aravind is good at farming and has little experience building Machine learning models and IoT systems. So, we created an irrigation system for Aravind using an IoT simulator(WOKWI) and designed a farm irrigation circuit using the Arduino Mega board. The irrigation system feedS the sensor readings to a machine learning model and produces the correct signal (percentage of water flow) to control the water supply. We use a servo motor to control the water supply.

The farm irrigation system contains four sensor units such that each unit senses temperature and humidity values. These values are provided to the ML model on the Arduino board to predict the water flow in percentage. Then the water flow percentage will be provided as input to the servo motor. The percentage values are mapped with servo motor rotation of 0◦ to 180◦. The sensor values from each sensor unit control a servo motor. Also, the percentage of water flow fed as input to servo motors is displayed on an LCD (Refer to Figure. 1). The block diagram for the circuit schematic is shown in Figure 2.

We trained our machine learning model separately and obtained the optimal parameter set. Then we built the ML model using the optimal parameter set on the simulator only for performing inference based on sensor readings. We also simulated day and night using photoresistor-sensor. The water flow is set to zero during the nighttime, irrespective of temperature and humidity values.

Figure 1: LCD Display

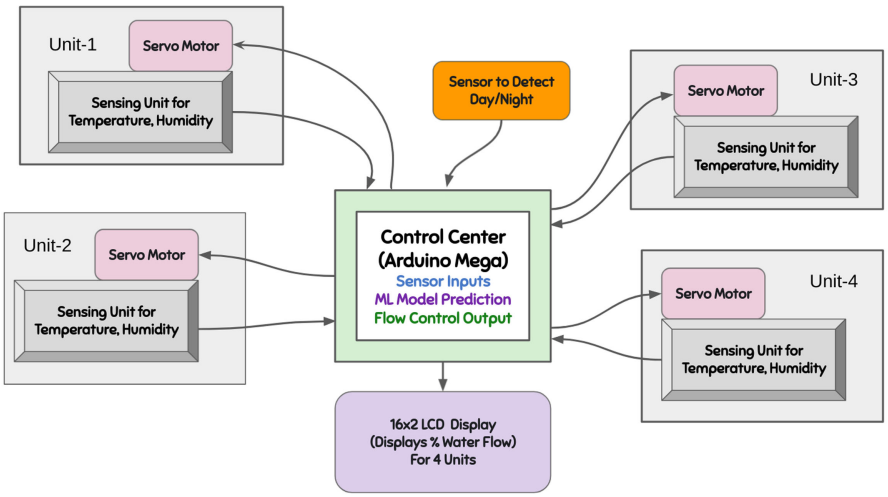

Figure 2: Block Diagram of Irrigation System

2 Dataset

The dataset that we are using consists of 200 entries and 3 features.Our dataset contains values of Temperaure and Humidity as input features and Percentage waterflow as output variable. It's basic structure is shown in Fig 3.

	Humidity(%)	Temperature(°C)	WaterFlow(%)
0	25.045045	76.856857	67.432607
1	8.880881	31.771772	0.000000
2	24.956957	75.735736	66.105904
3	40.328328	88.308308	0.000000
4	38.742743	57.157157	0.000000
...	...	...	...
195	14.474474	30.890891	0.000000
196	35.395395	51.551552	0.000000
197	23.855856	100.000000	96.088487
198	41.649650	76.856857	0.000000
199	35.967968	67.967968	0.000000

200 rows × 3 columns

Figure 3: Dataset

3 Equipments Used

We implemented this farm irrigation system on WOKWI simulator. The equipments we used on this simulator are listed as follows:

- wokwi-arduino-mega (Arduino Mega 2560)
- wokwi-servo (A standard Micro Servo Motor)
- wokwi-dht22 (Digital Humidity and Temperature sensor)
- wokwi-lcd1602 (An LCD with 2 lines, 16 characters per line)
- wokwi-photoresistor-sensor (Photoresistor (LDR) sensor module)

4 Methodology

4.1 ML model designing

We used a two-layer perceptron machine learning model (input layer + 2 hidden layers + output layer) to train our regression model. We imported the dataset file and saved it as a pandas dataframe in our .ipynb notebook. We created our two-layer perceptron ML model to train our input features(Temperature and Humidity) and produce the required output(Percentage water flow). The basic structure of our ML model is shown in figure 4. Our input layer consisted of 2 neurons that took the two inputs(Temperature and Humidity) from the sensors. We used the 'ReLu' activation function in all our layers. The input layer is followed by the first hidden layer, which consists of 3 neurons. The first hidden layer is followed by the second hidden layer, which consists of 3 neurons again. Finally, after the second hidden layer, we have our single neuron output layer. All our layers are dense. A dense layer is a regular but deeply connected neural network layer. The dense layer does the below operation on the input and returns the output.

$$output = activation(dot(inputs, weights) + bias)$$

After designing the architecture of our model, we compile our training data. We tried different learning rates and epochs to obtain maximum accuracy in the results. We found optimal values for learning rate and the number of epochs to be 0.01 and 5000, respectively. Adam is a replacement optimization algorithm for stochastic gradient descent for training deep learning models. We used it here during our compilation step. We also did a 0.2 validation split on our training data to make our model learn well. We specified the loss function to be 'mean squared error' during compilation. We use it to graphically compare the performance of our training(training loss)data and validation(validation loss) data based on mean squared errors.

After completing our compilation and obtaining values of epochs and learning rate, we proceed to the prediction step. We perform predictions on training data and compared it with their true values. We obtained the mean squared errors and R2 scores too on training data. Finally, we obtained the weights of our model and created a function that takes two inputs(temperature and humidity) and returns the water flow percentage as output. We used the weights we obtained from our model to operate upon the inputs and generate the required output inside this function.

We trained our model several times to obtain optimal weights, epochs, learning rate, etc. The results we got were pretty good considering the less amount of data we had. The results we obtained using our model are specified in the Result section of our report. To run our model, just run all cells of the .ipynb file. It will take time to execute as the number of epochs is high. Our .ipynb file is well commented and will be easy to understand.

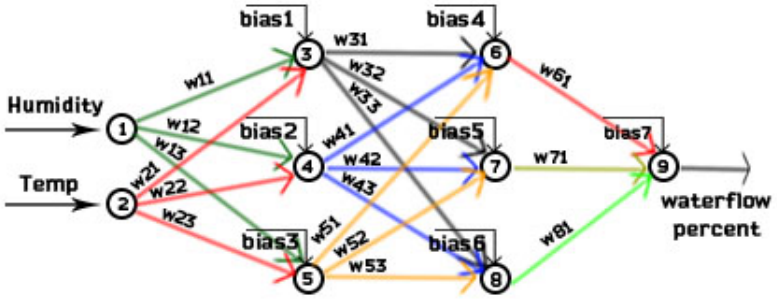

Figure 4: Basic architecture of our model

4.2 Circuit Designing on Simulator

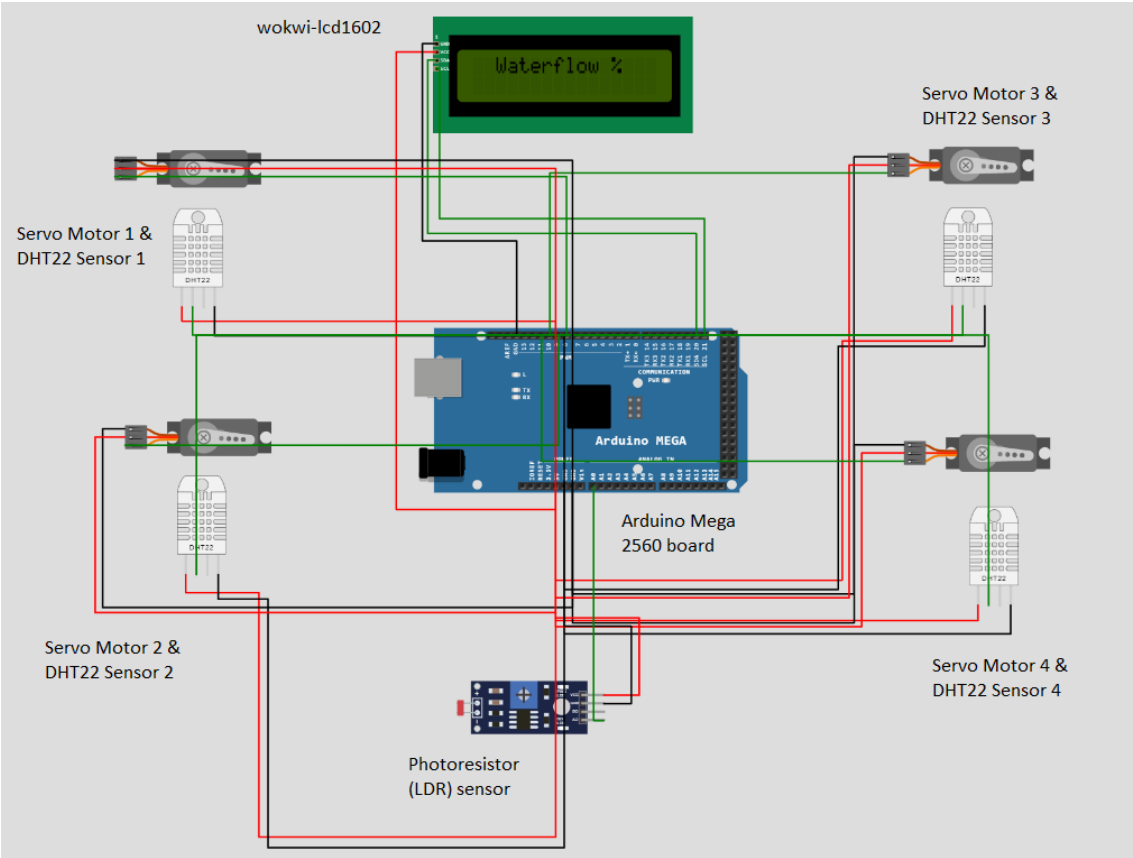

Figure 5: Circuit Diagram

Arduino Mega 2560 board: Powered by the ATmega2560 chip, which has 256K bytes of Flash program memory, 8k bytes of SRAM and 4K bytes of EEPROM. The board features 54 digital pins, 16 analog input pins, and 4 serial ports. It runs at 16MHz.This is the board that will hand all our connections and upon which the algorithm has been written. Let us now see all the sensors used and then the connections we have done for our simulation.

- **LCD 1602:** LCD1602, or 1602 character-type liquid crystal display, is a kind of dot matrix module to show letters, numbers, and characters and so on. The model 1602 means it displays 2 lines of 16 characters.

- **Photoresistor LDR Sensor:** The photoresistor sensor module includes a LDR (light-dependant resistor) in series with a 10K resistor. The AO pin is connected between the LDR and the 10K resistor. The voltage on the AO pin depends on the illumination - that is the amount of light that falls on the sensor.
- **DHT22 Sensor:** The DHT22 is a basic digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin, no analog input pins needed.
- **Servo Motor:** A servomotor is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration. It consists of a suitable motor coupled to a sensor for position feedback.

Sensors and their connections

Sensor	Voltage	Ground	Analog IN	SDA(Data)/Digital IN	SCL(Clock)
LCD 1602	5V	GND.1	-	20	21
Photoresistor(LDR)	5V	GND.2	A0	-	-
DHT22 Sensor 1	5V	GND.2	-	3	-
DHT22 Sensor 2	5V	GND.2	-	4	-
DHT22 Sensor 3	5V	GND.2	-	5	-
DHT22 Sensor 4	5V	GND.2	-	6	-
Servo Motor 1	5V	GND.3	-	8	-
Servo Motor 2	5V	GND.3	-	9	-
Servo Motor 3	5V	GND.3	-	10	-
Servo Motor 4	5V	GND.3	-	11	-

4.3 Simulator working

- First the Photoresistor(LDR) sensor senses for the current value in (lux). If the value is less than 50, it will be considered as night time and the LCD will show 0% for all 4 servo motors. This is handled by the rotate_servo(float,float,float,float) function which takes as input the water percent value for all the 4 motors. If the lux value ¿ 50, it will be considered as day time and we will continue our program.
- Then we will store the temperature and humidity values from all the 4 DHT22 sensors in arrays temperature[4] and humidity[4] respectively.
- Now for each pair of temperature and humidity value we will call our ml_predict_water(t,h) function which will run a set of equations that is obtained using the optimal weights got by separately training our ML model using the given dataset. The function will return us the water percentage for the sent temperature and humidity values.
- Upon receiving the water percentage values update the water_percent[4] array. We will now sent the values of this array to our rotate_servo() function mentioned in 1st point.
- The rotate_servo(float pos1, float pos2, float pos3, float pos4) is responsible for printing the percentage values on the LCD and rotating the servo motors by converting the percentage values into angle between 0 - 180 using water_percent_to_angle(float pos) function.
- All this runs in an infinite loop until interrupted externally.

4.4 Simulation

The link to our simulated project is given below:
`https://wokwi.com/arduino/projects/314001811667157570`

5 Results

After training the ML model, the results we obtained were as follows:

- The R2 score on the Train set is: 0.840
- Mean squared error on Train data is(percentage-wise): 13.948
- Epochs used on training: 5000
- Learning Rate: 0.01

6 Conclusions

We successfully trained and simulated our model that generates water flow percentage as output upon seeing Temperature and Humidity as inputs. In our simulation process, we can provide random values of temperature and humidity to the DHT22 sensor. Based on that, it will produce suitable output. The result produced by this regression model might not be exactly the same as the true value. But, the difference in true values and predicted values will be minimal, which would not affect Aravind much. For simulating Day/Night, the threshold sensor value is set at 50 units, i.e., if the sensor value is set to less than 50 units, our simulation will treat it as night, and if the sensor value is greater than 50 units, it will treat it as day time. The water flow at night is automatically turned to 0 irrespective of temperature and humidity values.