Advanced Automatic Water Irrigation System Using Sensor Data

DONE BY :

DWARAKANATH M

SANTOSHRAM M B

VISHWA AMRUTH D

Irrigation plays a crucial role in agriculture and gardening, ensuring that plants receive the right amount of water for healthy growth. However, managing watering schedules, especially for large areas or during periods of absence, can be a labor-intensive and time-consuming task. Traditional methods often rely on manual intervention, which can result in inconsistent watering, over-watering, or under-watering, negatively impacting plant health.

Requirement Specifications

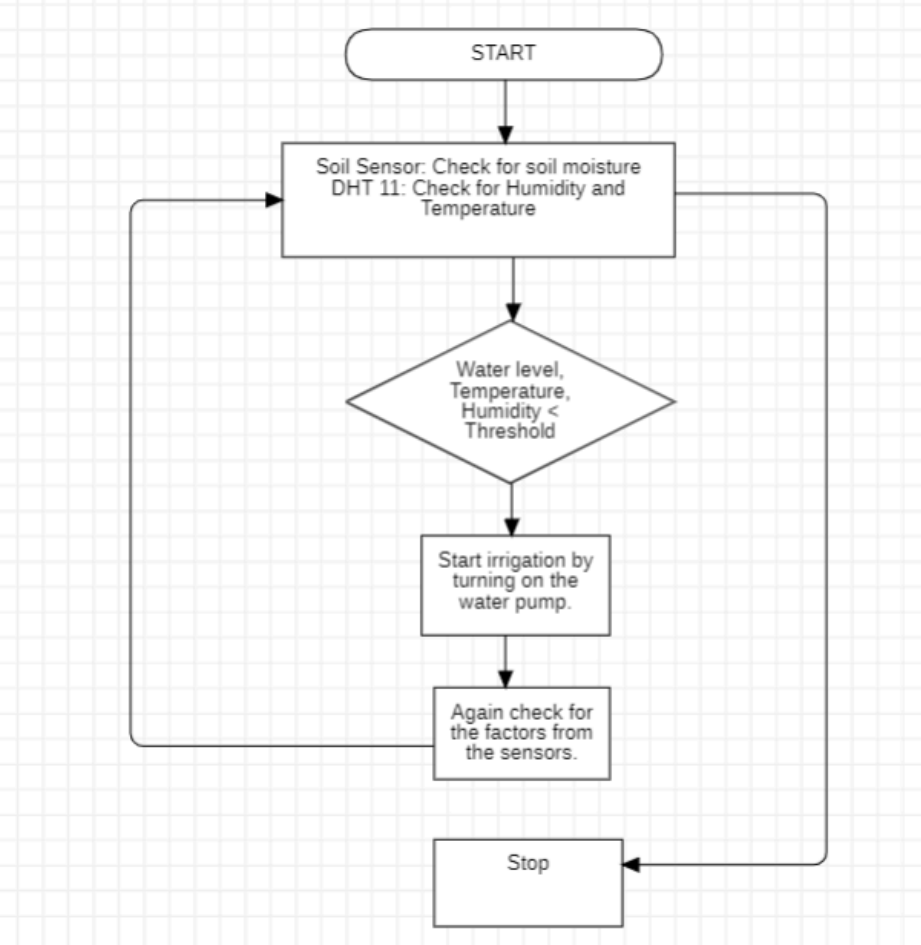
Hardware Requirements:

- Arduino UNO
- Soil Moisture Sensor (YL-69)
- DHT11 Temperature and Humidity Sensor
- 6V Submersible Water Pump
- Relay Module (Single Channel, 5V)
- Battery
- D-Type USB Cable
- Breadboard & Jumper Wires

Software Requirements:

- Arduino IDE (Version 1.8.16 or later)
- Python (Version 3.10 or later)
- Jupyter Notebook

System Design and Implementation



System Design and Hardware Setup

The first step is to design the overall system architecture, which consists of the following components:

Arduino Uno: Acts as the main controller to process sensor data and control the relay that activates the water pump.

Soil Moisture Sensor (YL-69): Used to measure the moisture content in the soil. When the soil is too dry, the sensor will trigger the system to start watering.

DHT11 Temperature and Humidity Sensor: Measures the ambient temperature and humidity. These values are used to understand the environmental conditions and make more accurate irrigation decisions.

Relay Module: Used to control the 6V water pump, switching it on or off based on the conditions set by the sensors.

6V Submersible Water Pump: Provides the water needed for irrigation when the system determines that the soil moisture is below the threshold.

Power Supply: The Arduino is powered by a or USB connection, while the water pump is powered separately by a 6V battery.

Sensor Data Acquisition

Soil Moisture Sensor: The sensor is placed in the soil to measure its moisture content. When the soil moisture level is below a predefined threshold (i.e., the soil is dry), the Arduino will receive a signal to activate the pump.

DHT11 Sensor: The temperature and humidity readings from the DHT11 sensor are used as additional features for the machine learning model. These environmental factors play a role in plant watering needs, especially in varying weather conditions.

Data Collection: Data from the sensors (soil moisture, temperature, and humidity) are read by the Arduino, and this data is sent to the connected computer for processing and analysis.

Data Preprocessing and Machine Learning Model

Once the data is collected, it is processed using **Python** and **machine learning algorithms** to predict when and how much water should be applied to the plants.

Dataset Creation: Historical data from the sensors is used to create a dataset with features such as **soil moisture**, **temperature**, and **humidity**. This dataset will be used to train the machine learning model.

Model Training: Several machine learning algorithms are tested, including **Random Forest**, **Support Vector Machine (SVM)**, and **Decision Tree**. The models are trained on the dataset to learn the patterns in the data and predict watering needs.

- The **Random Forest algorithm** was chosen for its ability to handle non-linear relationships between features and its robustness in handling noisy data.

Model Validation: After training, the model is validated using a test dataset to evaluate its accuracy and performance in predicting watering events. The model's accuracy is assessed based on how well it predicts when watering is needed based on the test data.

Model Integration with the Arduino System

After training the machine learning model and achieving satisfactory results, the next step was to integrate the model with the Arduino system. The trained model, implemented in Python, uses real-time sensor data to predict the watering needs of the plants. The Python script, running on a computer, communicates with the Arduino via a serial connection, sending control signals based on the model's predictions. When the model determines that watering is necessary—based on soil moisture levels and environmental factors like temperature and humidity—the Python script instructs the Arduino to trigger the relay module. The relay, in turn, activates or deactivates the 6V water pump, ensuring that the pump is only turned on when the soil is dry, and the conditions indicate that the plants need water. This integration enables the system to automate the watering process efficiently and autonomously.

Results and Discussions

	A	B	C	D
1	Soil Moisture	Air Humidity (	Temperature	Status
2	39.96320951	29.82557594	21.54264209	OFF
3	86.05714451	65.65901063	21.17446998	OFF
4	68.55951534	56.7211191	37.65636451	OFF
5	57.89267874	47.84712398	21.238655	OFF
6	22.48149124	23.80265534	21.79874315	ON
7	22.47956163	22.32352681	33.98495656	ON
8	14.64668897	17.17544793	26.24349606	ON
9	79.29409166	63.99782068	34.41776392	OFF
10	58.08920094	43.15912075	16.63415394	OFF

Classification reports for all the models trained:

Random Forest:

Accuracy: 0.92

	precision	recall	f1-score	support
OFF	0.91	0.90	0.91	2173
ON	0.93	0.94	0.93	2626
accuracy			0.92	4799
macro avg	0.92	0.92	0.92	4799
weighted avg	0.92	0.92	0.92	4799

Support Vector Machine:

Accuracy: 0.88

	precision	recall	f1-score	support
OFF	0.87	0.86	0.87	2173
ON	0.89	0.90	0.89	2626
accuracy			0.88	4799
macro avg	0.88	0.88	0.88	4799
weighted avg	0.88	0.88	0.88	4799

Logistic Regression:

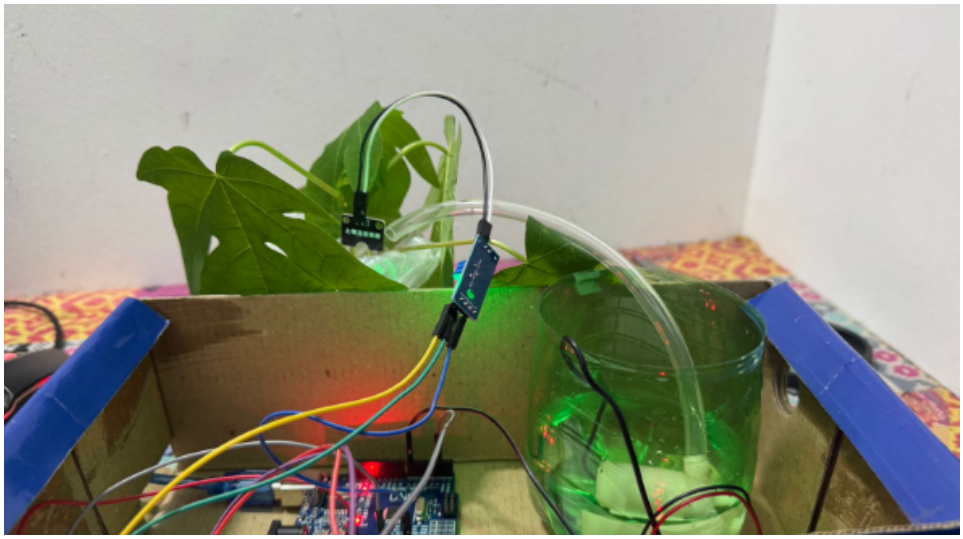
Accuracy: 0.85

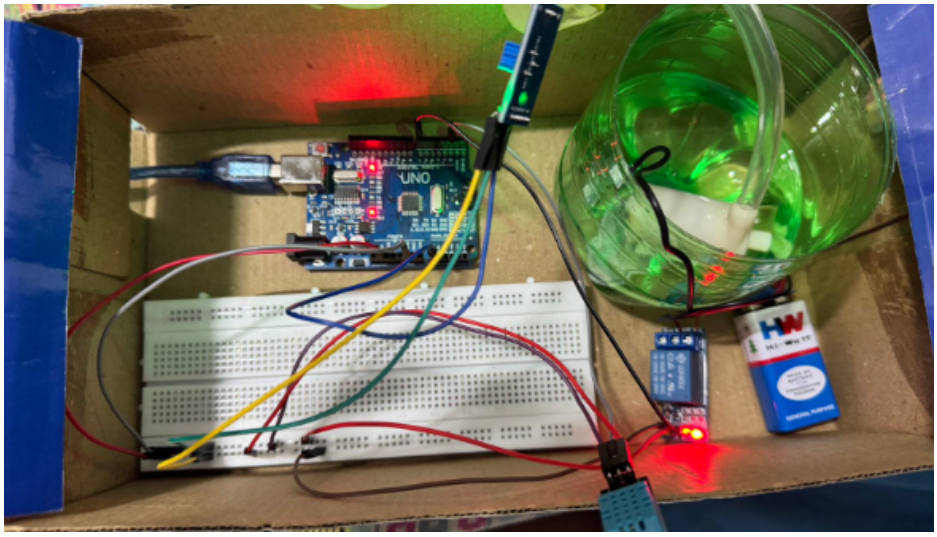
	precision	recall	f1-score	support
OFF	0.83	0.82	0.83	2173
ON	0.86	0.87	0.86	2626
accuracy			0.85	4799
macro avg	0.85	0.85	0.85	4799
weighted avg	0.85	0.85	0.85	4799

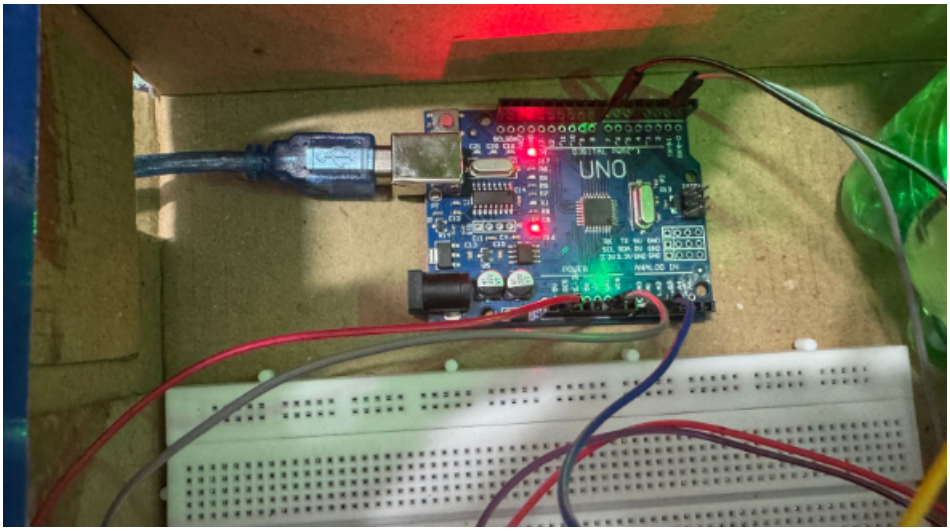
Out of all the models evaluated, **Random Forest (RF)** has emerged as the best performing model. It achieved the highest accuracy of **92%**, along with superior **precision**, **recall**, and **F1-scores** for both classes, **OFF** and **ON**. Compared to the **Support Vector Machine (SVM)** and **Logistic Regression (LR)** models, which achieved accuracies of **88%** and **85%** respectively, the **Random Forest model** demonstrated more consistent and reliable performance.

This makes **Random Forest (RF)** the optimal choice for this task, as it provides a better balance between **precision** and **recall**, ultimately leading to a higher **F1-score** and more robust generalization on unseen data. Therefore, based on these results, **Random Forest** is recommended for deployment in this scenario.

Appendix








```
Soil Moisture: 67.80, Air Humidity %: 81.00,Temperature: 34.70
Motor Status: ON
```

```
Soil Moisture: 18.23, Air Humidity %: 79.10,Temperature: 34.70
Motor Status: OFF
```