SMART IRRIGATION MODEL

Abstract

The ability to grow massive amounts of food in recent times has only been possible with the advent of pesticides, mechanisation, and using a *lot* of water. However, these practices are often unsustainable, thus ruining the environment and depleting valuable resources in the process. This project is meant to tackle water usage, as being able to pinpoint exactly where water is needed can prevent over-irrigation.

The Smart Agriculture System based on IoT devices, which was previously employed in this field and supported this model, is able to monitor soil moisture and climate conditions in order to grow and yield a decent crop. from which we developed the smart irrigation concept and dataset.

This project focuses on the development of a Smart Irrigation System leveraging TinyML (TinyML is a very efficient and cost effective way to deploy ML models without SOTA architectures, this way machine learning can penetrate into different domains) technology for efficient and automated field irrigation. The goal is to predict the need for irrigation based on environmental factors such as soil moisture and temperature.This system is intended to create an automated irrigation mechanism that turns the pumping motor ON and OFF on detecting the dampness content of the earth. In this system, we are interfacing the Arduino board through, soil moisture sensor and temperature sensor.

Timeline

1.Data Collection

The Dataset extracted has variables as Soil Moisture & Temperature and predicts whether the crop needs irrigation or not. As a binary classification in 0 or 1.

0s [21]

```
read_data=read_data.drop('crop',axis='columns')
read_data
```

	moisture	temp	pump
0	638	16	1
1	522	18	1
2	741	22	1
3	798	32	1
4	690	28	1
...	...	...	...
195	941	13	1
196	902	45	1
197	894	42	1
198	1022	45	1
199	979	10	1

200 rows × 3 columns

2.Train Model

We used Google Colab to Train and Test the model using Tenserflow library and logistic regression . Then we extract the coefficient and the intercept.

Python Code

```
[3] #importing Required Libraries
    import numpy as np
    import pandas as pd
    from sklearn.model_selection import train_test_split
    from sklearn.linear_model import LogisticRegression
    from sklearn.metrics import accuracy_score
```

```
[20] #Reading the Dataset
     read_data = pd.read_csv("data.csv")
```

```
[21] #Dropping "Crop" Variable
     read_data=read_data.drop('crop',axis='columns')
     read_data
```

	moisture	temp	pump
0	638	16	1
1	522	18	1
2	741	22	1
3	798	32	1
4	690	28	1
...	...	...	...
195	941	13	1
196	902	45	1

```
[5] #Labelling Input Data
    X = read_data[["moisture","temp"]]
    X
```

	moisture	temp
0	638	16
1	522	18
2	741	22
3	798	32
4	690	28
...	...	...
195	941	13
196	902	45
197	894	42
198	1022	45
199	979	10

200 rows × 2 columns

```
[6] #Labelling Output Data
    y=read_data[["pump"]]
    y
```

	pump
0	1
1	1

```
[7] #Creating Input Numpy Array
    X=np.array(X)
    X[12]

    array([507,  45])
```

```
[8] #Creating Output Numpy Array
    y=np.array(y)
    y[56]

    array([1])
```

```
[10] #Train_Test Spltting the Data
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
     print(X.shape)
     print(y.shape)

     (200, 2)
     (200, 1)
```

```
[11] #Applying Logistic Regression
     model = LogisticRegression()
```

```
[12] model.fit(X_train, y_train)

     /usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was pas
       y = column_or_1d(y, warn=True)
     ▾ LogisticRegression
     LogisticRegression()
```



```
[11] #Applying Logistic Regression
     model = LogisticRegression()
```

```
[12] model.fit(X_train, y_train)

     /usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was pas
       y = column_or_1d(y, warn=True)
     ▾ LogisticRegression
     LogisticRegression()
```



```
[13] #Predicting Output
     y_pred = model.predict(X_test)
```

```
[14] #Finding the Accuracy of Output
     accuracy = accuracy_score(y_test, y_pred)
     print(f"Model Accuracy: {accuracy}")

     Model Accuracy: 1.0
```

```
#Finding the Coefficiens & Intercept of Regression
coefficients = model.coef_
intercept = model.intercept_

print("Coefficients:", coefficients)
print("Intercept:", intercept)

Coefficients: [[0.31252223 0.18626435]]
Intercept: [-160.56781552]
```

3. Arduino Model

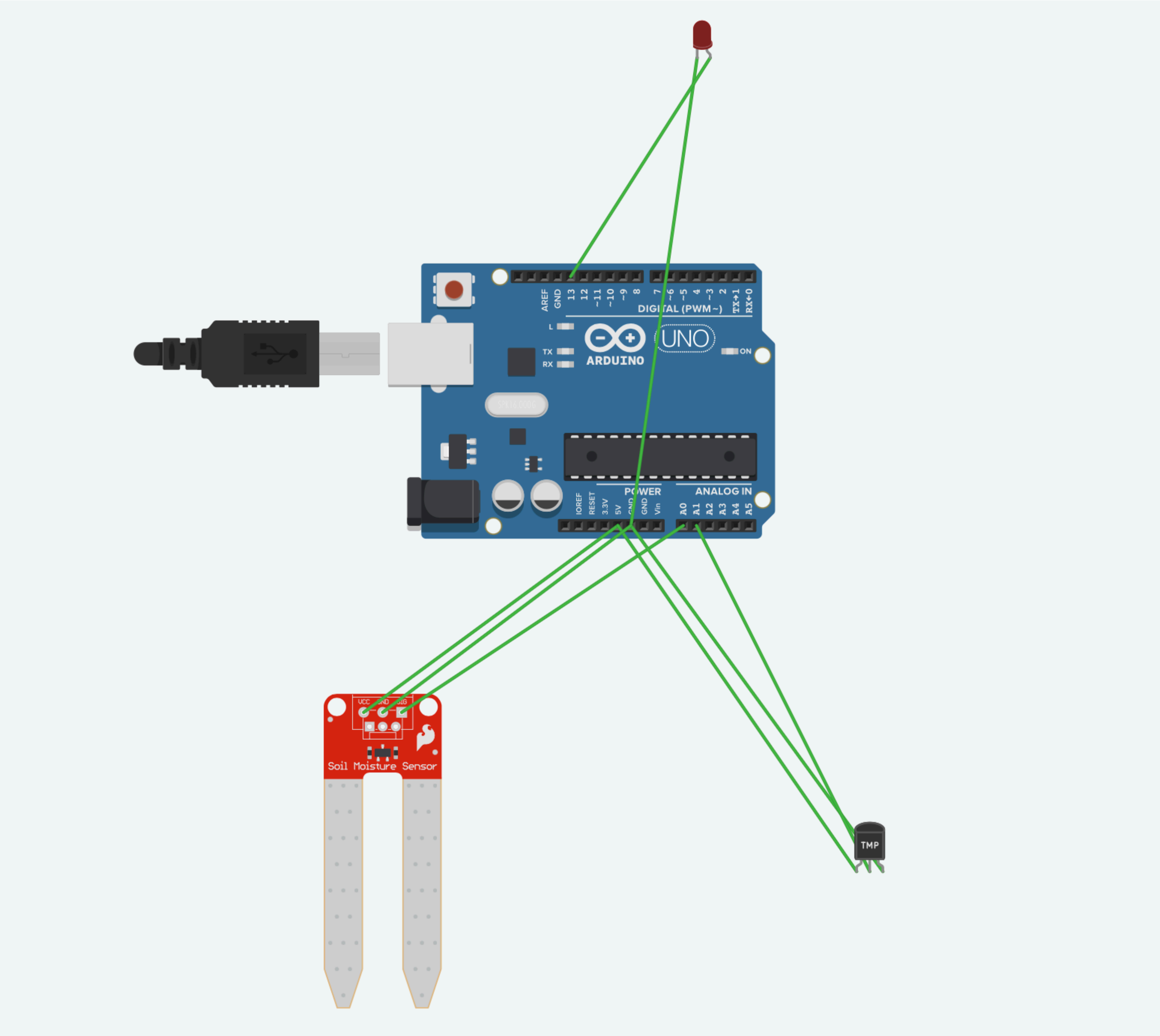
TinkerCAD online Arduino simulator which provides numerous number of objects for running stable environments. Since the online simulator does not support .tf file, we inputed the values of coefficient and intercept manually in cpp code. Then we used that code on TinkerCAD to classify new data wether pump will be ON/ OFF.

CPP Code

```cpp
float model_coef[2] = {0.31252223, 0.18626435};  //Coeeficients calculated
float model_intercept = -160.56781552;

void setup() {
  Serial.begin(9600);

  pinMode(A1, INPUT);  // Analog input for TMP36 temperature sensor
  pinMode(A0, INPUT);  // Analog input for soil moisture sensor
  pinMode(13, OUTPUT);  // Single-color LED
}

void loop() {
  // Read sensor values
  int moistureValue = analogRead(A0);  // Soil moisture sensor
  float tempValue = analogRead(A1);  // TMP36 temperature sensor

  // Apply the logistic regression model

    float prediction = (model_coef[0] * moistureValue) + (model_coef[1] * tempValue) + model_intercept;


  // Display results with single-color LED
  if (prediction < 0.0) {
    // Irrigation needed (Display HIGH)
    digitalWrite(13, HIGH);
  } else {
    // No irrigation needed (Display LOW)
    digitalWrite(13, LOW);
  }

  Serial.print("Moisture: ");
  Serial.print(moistureValue);
  Serial.print(", Temp: ");
  Serial.print(tempValue);
  Serial.print(", Prediction: ");
  Serial.println(prediction);

  delay(1000);
}
```

Show / hide serial monitor

ARDUINO CIRCUIT



Requirements

- TinyML Model
- Soil Moisture Sensor
- Temperature Sensor
- LED

4. Result

After building the circuit and programming it we can predict the need of irrigation using an led light by changing soil moisture and temperature. A red light gives us that yes irrigation is needed and a no light indicates no irrigation is needed.