Smart Greenhouse

ES-42422: Embedded System Architecture

Ma Khin Pyae Phyo San
CST-114

Outline

- Introduction
- Objective
- Implementation
- Result and Discussion

Introduction

- For ambitious farmers!
- Always eager improve the yields from his Greenhouse by using intelligent irrigation system.
- Controlling the irrigation system based on the present humidity, temperature levels.
- ML model that predicts water supply ratio by using sensor data.
- System will turnoff during nighttime.

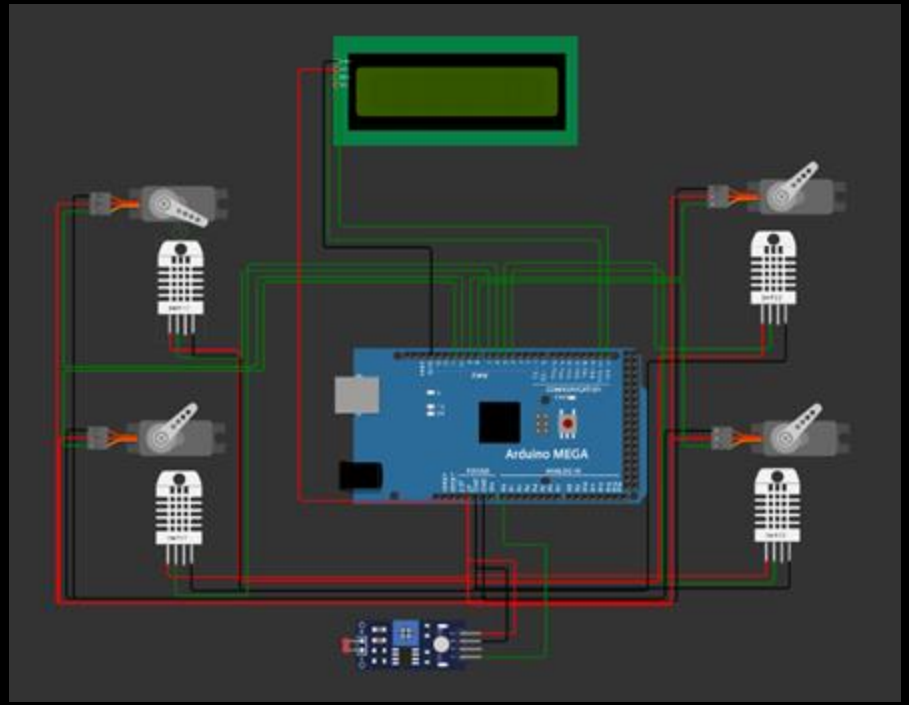
Objective

- The sensor data from the irrigation system needs to be transmitted to a ML model, which will then generate the accurate signal (representing the percentage of water flow) to manage the supply of water.

Implementation

- Components:
  - Arduino Mega Board
  - a single Multi
  - Layer Perceptron (MLP) Model deployed, which has been trained on a dataset a priori to the simulation.
  - DHT22 sensor to detect temperature and humidity.
  - Servo Motors to control the water flow.
  - LCD to display the percentage of each water supply unit.
  - Photoresistor sensor to detects time of day as either day or night.

Implementation

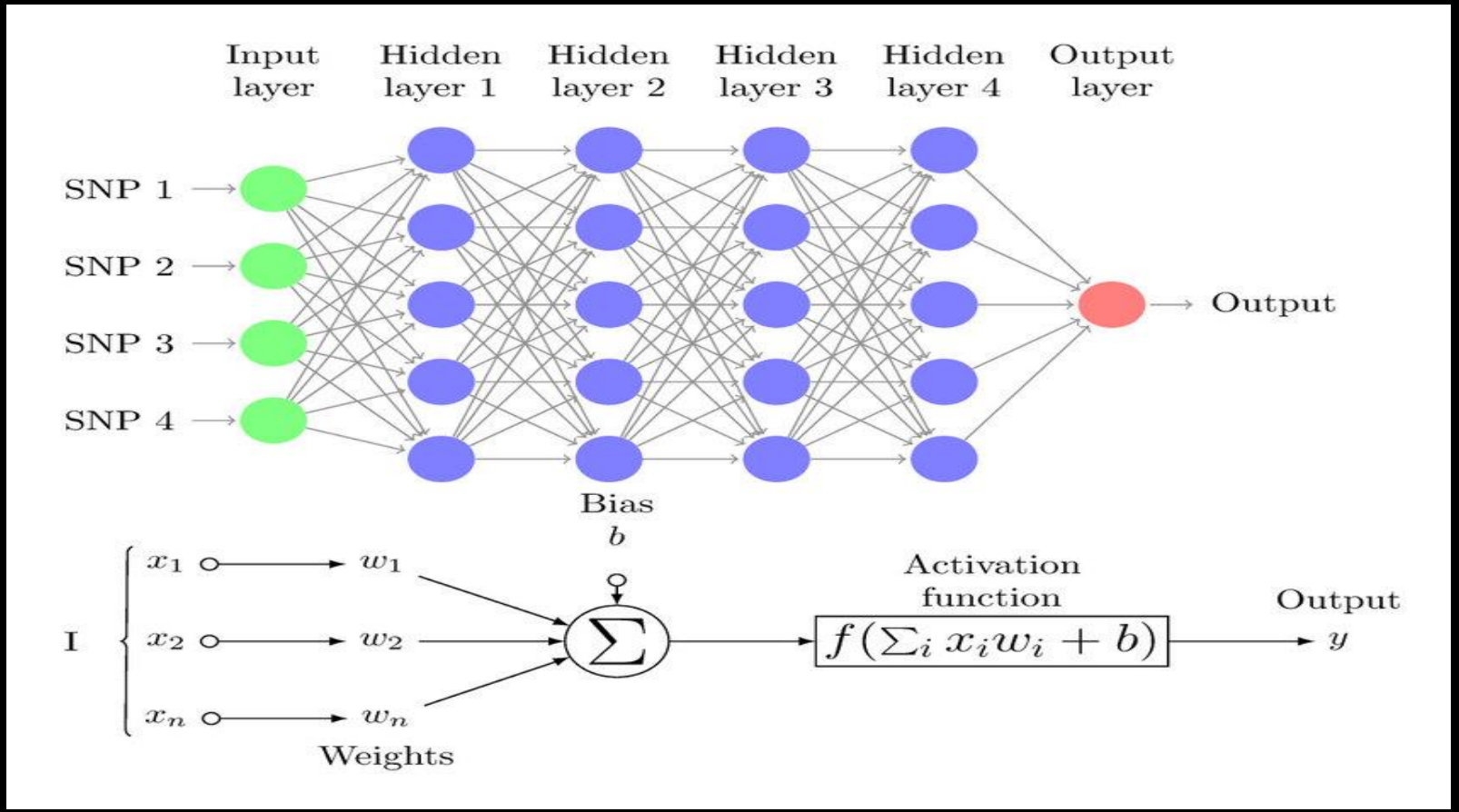


Architecture Diagram

Implementation

- The arduino process the data sent by photoresistor sensor (AO) and forecast day and night.
- If Day, value is above 50.
- DHT22 sensor values sent to arduino board and apply min max scaling.
- t = 50, min = 20, max = 100. Then new t = (t-min)/(max−min)
- Servo motor angle = (water percentage ∗ 180)/100

Implementation



Implementation

- A Multi-layer Perceptron (MLP) regression model is a type of artificial neural network used for regression tasks.
- Used to predict continuous numerical values.
- Layers include an input layer, one or more hidden layers, and an output layer.

Implementation

```
from sklearn.metrics import mean_squared_error
model = MLPRegressor(activation='relu', hidden_layer_sizes=(16,8),random_state = 56,
max_iter=8000).fit(x_train, y_train)
y_pred = model.predict(x_test)
rms = mean_squared_error(y_test, y_pred, squared=False)
print("r2_score: ", (r2_score(y_pred, y_test)), ",accuracy",model.score(x_test, y_test),
",rmse:",rms)
```

This line creates an instance of the MLPRegressor class (a type of neural network for regression), sets its activation function to 'relu', specifies two hidden layers with 16 and 8 neurons respectively, sets a random seed (random_state) for reproducibility, and sets the maximum number of iterations to 8000. It then fits (trains) the model using the `x_train` input data and corresponding `y_train` target data.

Result and Discussion

Demo Here

Thank You!