IoT based smart water management system

Ankith Kumar (2101AI12)

Prakash Kumar (2101AI24)

Vinod (2101AI27)

Problem Statement

Design and implement a Smart Water Management System that combines sensor technology, data analysis, and user-friendly interfaces that provides real-time monitoring and management of the water supply

Motivation

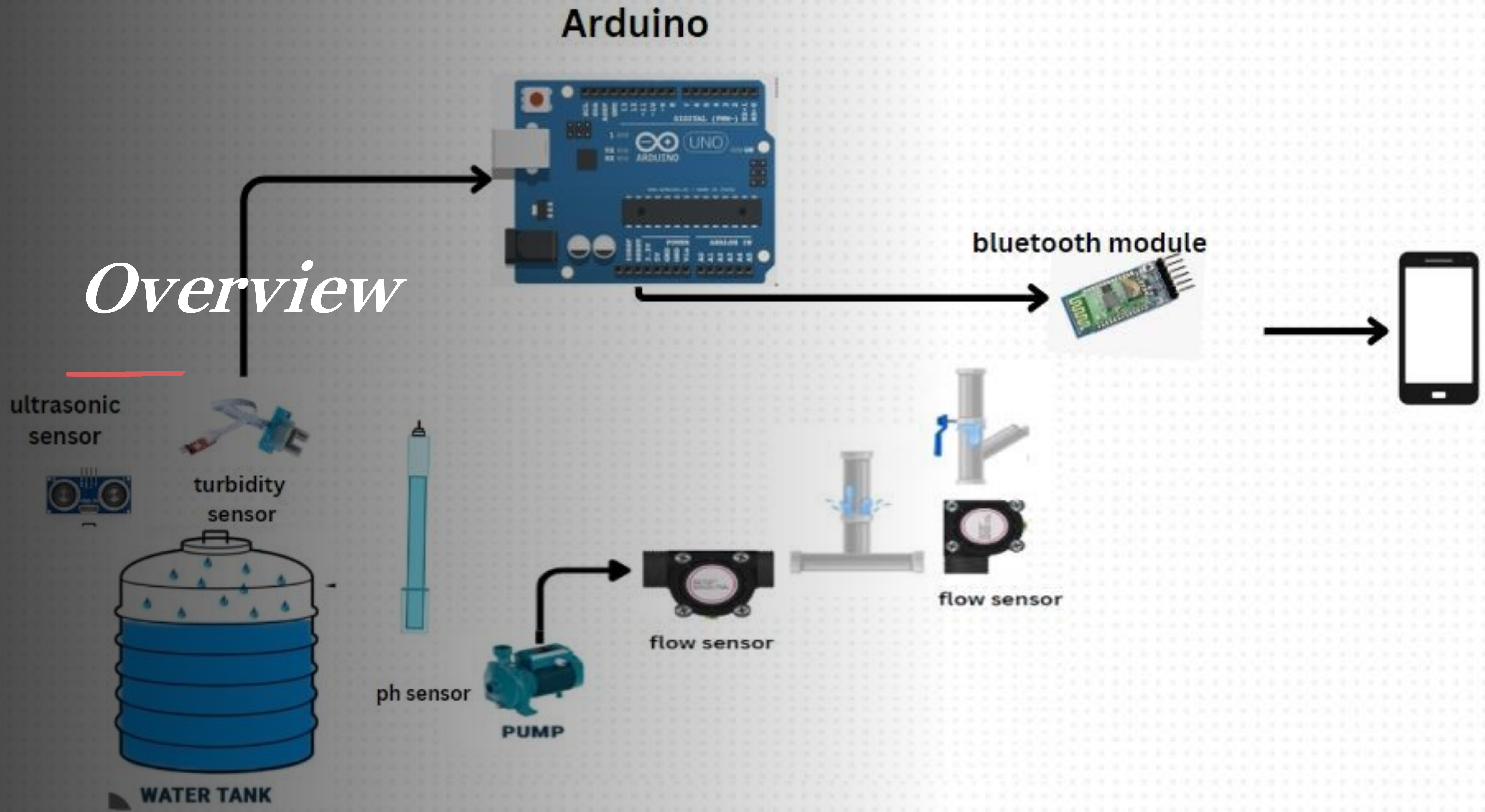
- **Water Scarcity:** Growing concerns about water scarcity and the need to manage this finite resource more effectively.
- **Water Waste and leakage:** According to the UN, between the primary location where pure water is released and the tap we use for pure water, more than 30% of water is wasted, and the main culprit for this problem is leakage.
- **Water Quality:** A significant portion of the population lacks awareness regarding the quality of the water they consume. Waterborne diseases have an economic burden of approximately USD 600 million a year in India.
- **Technological Advancements:** Leveraging advancements in IoT, sensor technology, and data analytics to address real-world problems.

Proposed System

This system will focus on three key aspects:

1. Water Level Monitoring
2. Leakage Detection
3. Water Quality Measurement

Overview


Arduino
bluetooth module
ultrasonic sensor
turbidity sensor
ph sensor
PUMP
flow sensor
flow sensor
WATER TANK

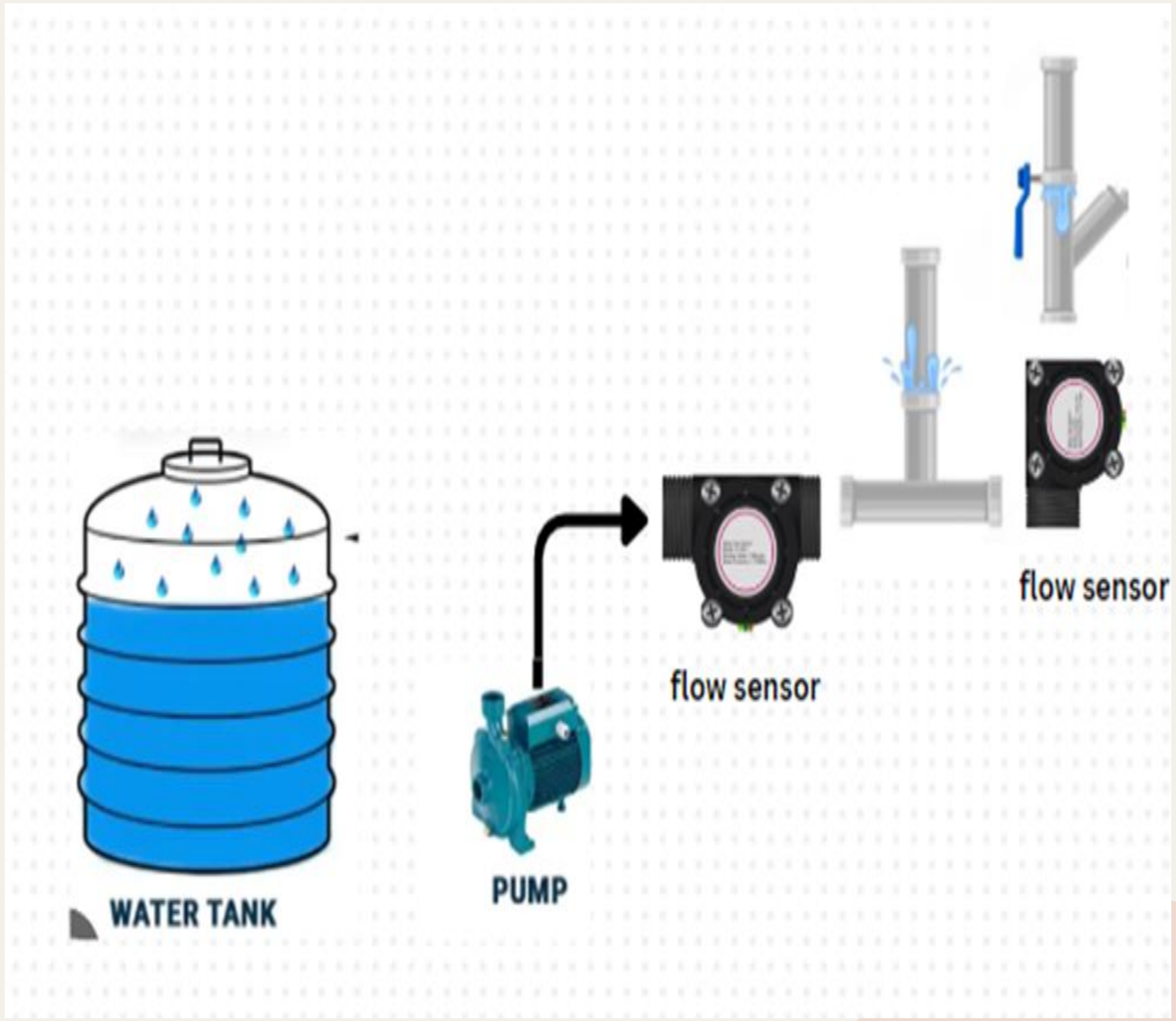
Sensors Used

- **Flow Sensor**:- Model – YF-S201, Rotating wheel inside that is turned by the flow of water, Magnetic sensor to detect rotation of wheel, Each rotation of wheel generates certain no of pulses and this is proportional to flow rate, 3.5-24V.

- **Turbidity Module**:- Sensor emits light and it travels through the liquid, Suspended particles in water scatter light, Degree of scattering proportional to turbidity of water, around 5V.

Leakage Detection

--We are connecting an Arduino board, flow sensor.
--Arduino programming involves reading flow sensor data and data processing.
--Flow Sensor Calibration ensures accurate flow measurements
--Data transmission to a mobile device is facilitated.
--Real-time monitoring enables immediate response to leaks.

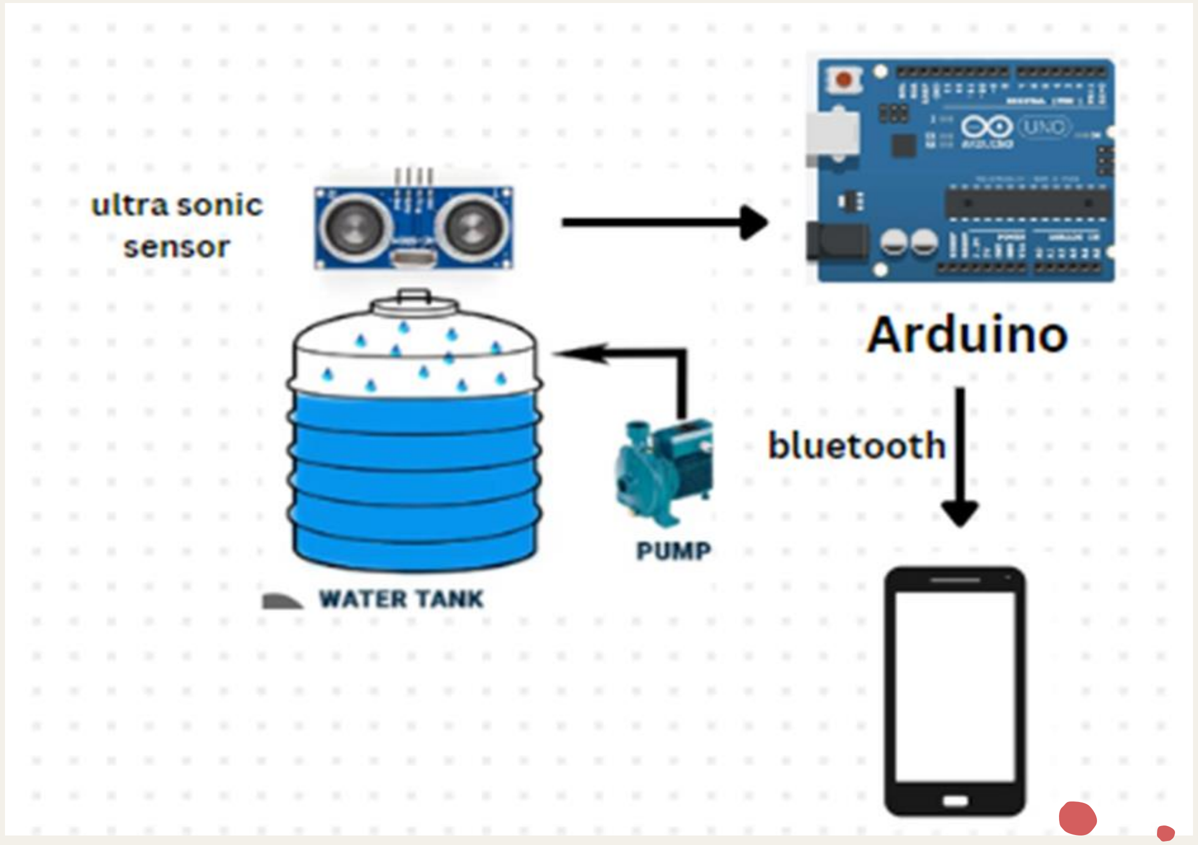


Water Quality Measurement

- **Measurement Techniques:** pH and turbidity measurements, is acquired through sensors connected to an Arduino Uno microcontroller, which interfaces with a Bluetooth module for data transmission to an Android-based platform.
- **ML Model:** Utilized a trained Decision Tree Classifier ML model, which is trained on a dataset, for accurate water potability prediction and a LSTM model to predict the amount of water used.
- **Water Quality Prediction:** The trained ML algorithm is integrated into the Android application, enabling real-time water potability prediction based on pH and turbidity parameters obtained from the connected sensors

Water Level Monitoring

- Ultrasonic sensor utilizing the principle of sound wave reflection to provide accurate distance measurements to the water's surface, placed in water tank.
- Connecting ultrasonic sensor with Arduino boards with built-in or add-on Bluetooth modules can establish a serial connection to the mobile device.
- Data is transmitted wirelessly over Bluetooth to a mobile app running on the paired device.
- Mobile will notify users when water levels reach specific thresholds or critical levels.

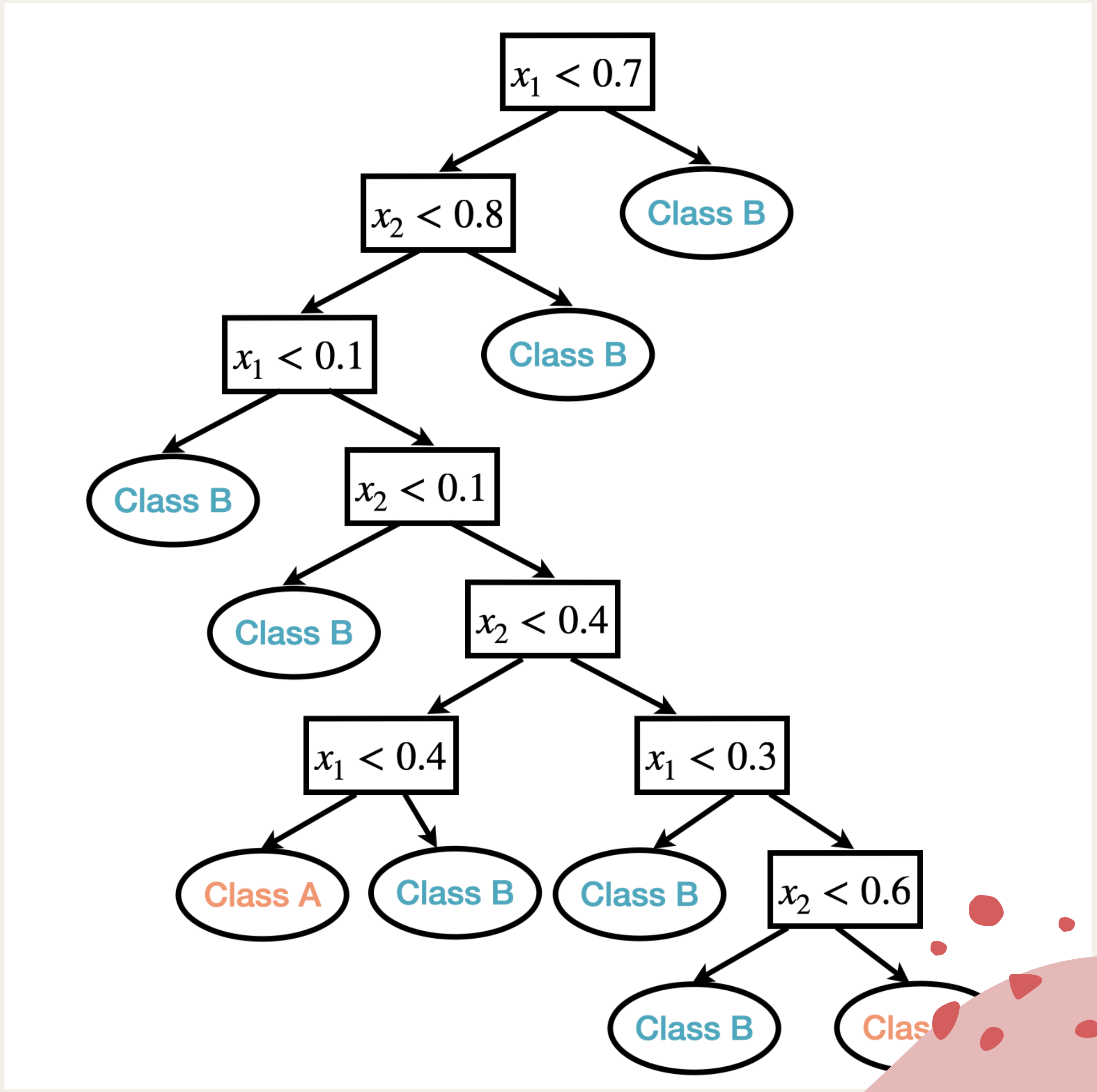


Machine Learning Model

- **Data collection:** The dataset used in this approach came from Kaggle’s Water Quality Dataset (https://www.kaggle.com/datasets/adityakadiwal/water-potability).
- **Data Preprocessing:** irrelevant parameters were excluded, resulting in the selection of turbidity and pH as the key features for predicting water potability.
- **Dealing with Missing Values:** Mean value imputation technique is used.
- **Model Training:** Trained the model employing a Decision Tree Classifier

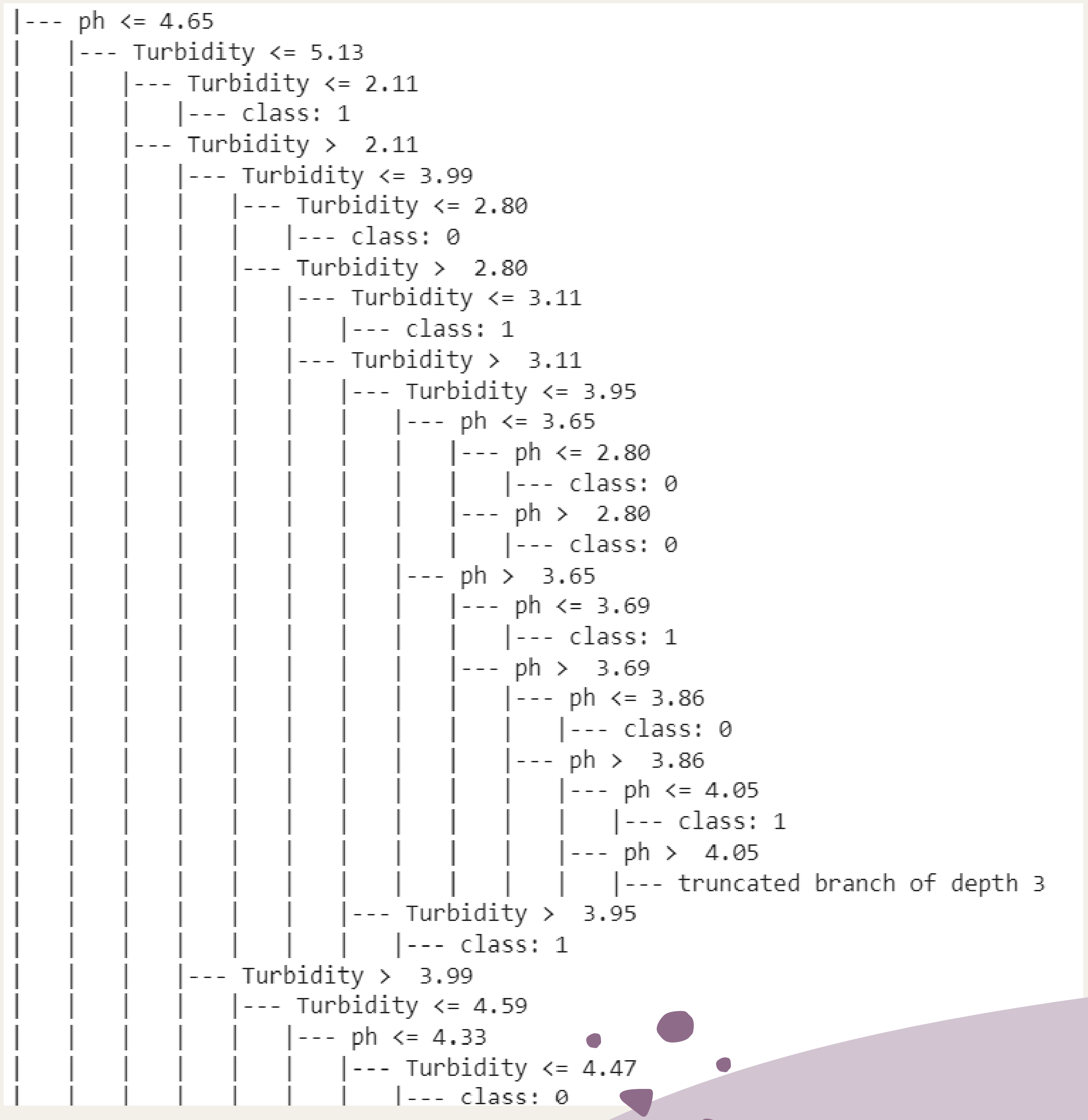
Decision Tree Classifier, Why?

- **Non-linearity Handling:** Decision trees are capable of modelling complex, non-linear relationships within the data, making them effective for tasks where the decision boundary is not linear.
- **Interpretable Structure:** Decision trees represent decision-making processes in a tree-like structure, which is highly interpretable
- **Scalability:** Decision trees can be readily implemented on resource-constrained platforms like Android applications, making them a practical choice for real-time predictions in your project.



Our model's tree rules

- This is the tree that ours model gives on training dataset.
- This if and else conditions is applied in android code to predict the potability of water.

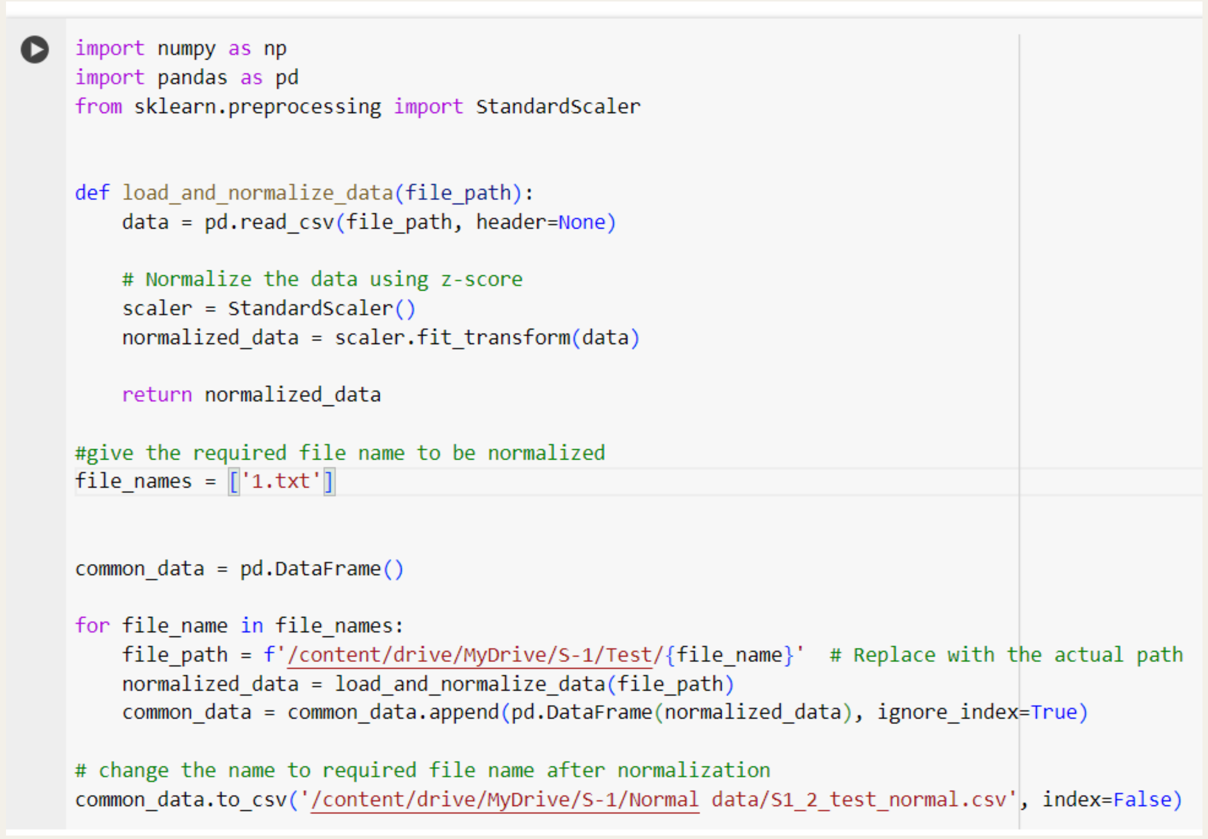


```
|--- ph <= 4.65
|   |--- Turbidity <= 5.13
|   |   |--- Turbidity <= 2.11
|   |   |   |--- class: 1
|   |   |--- Turbidity >  2.11
|   |   |   |--- Turbidity <= 3.99
|   |   |   |   |--- Turbidity <= 2.80
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- Turbidity >  2.80
|   |   |   |   |   |--- Turbidity <= 3.11
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- Turbidity >  3.11
|   |   |   |   |   |   |--- Turbidity <= 3.95
|   |   |   |   |   |   |   |--- ph <= 3.65
|   |   |   |   |   |   |   |   |--- ph <= 2.80
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- ph >  2.80
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- ph >  3.65
|   |   |   |   |   |   |   |   |--- ph <= 3.69
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- ph >  3.69
|   |   |   |   |   |   |   |   |   |--- ph <= 3.86
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- ph >  3.86
|   |   |   |   |   |   |   |   |   |   |--- ph <= 4.05
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |   |--- ph >  4.05
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 3
|   |   |   |   |   |   |--- Turbidity >  3.95
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |--- Turbidity >  3.99
|   |   |   |   |--- Turbidity <= 4.59
|   |   |   |   |   |--- ph <= 4.33
|   |   |   |   |   |   |--- Turbidity <= 4.47
|   |   |   |   |   |   |   |--- class: 0
```

LSTM model to predict the usage of water everyday

- Normalization is required to scale and standardize features in a dataset, ensuring that they are on a similar scale, preventing certain features from dominating others

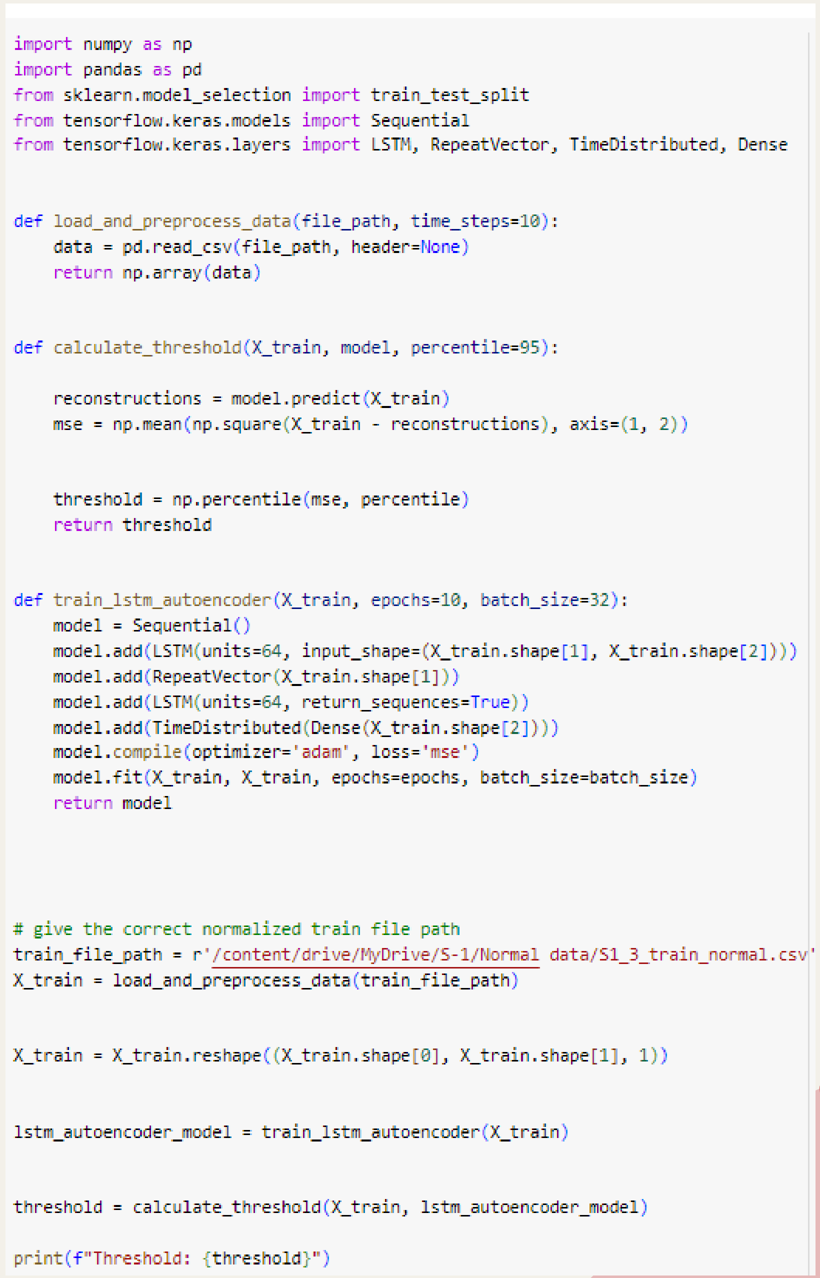
- The code uses the Z-score normalization method to normalize the data. This method transforms the data to have a mean of 0 and a standard deviation of 1



```python
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler


def load_and_normalize_data(file_path):
    data = pd.read_csv(file_path, header=None)

    # Normalize the data using z-score
    scaler = StandardScaler()
    normalized_data = scaler.fit_transform(data)

    return normalized_data

#give the required file name to be normalized
file_names = ['1.txt']


common_data = pd.DataFrame()

for file_name in file_names:
    file_path = f'/content/drive/MyDrive/S-1/Test/{file_name}'  # Replace with the actual path
    normalized_data = load_and_normalize_data(file_path)
    common_data = common_data.append(pd.DataFrame(normalized_data), ignore_index=True)

# change the name to required file name after normalization
common_data.to_csv('/content/drive/MyDrive/S-1/Normal data/S1_2_test_normal.csv', index=False)
```

LSTM – Threshold Prediction

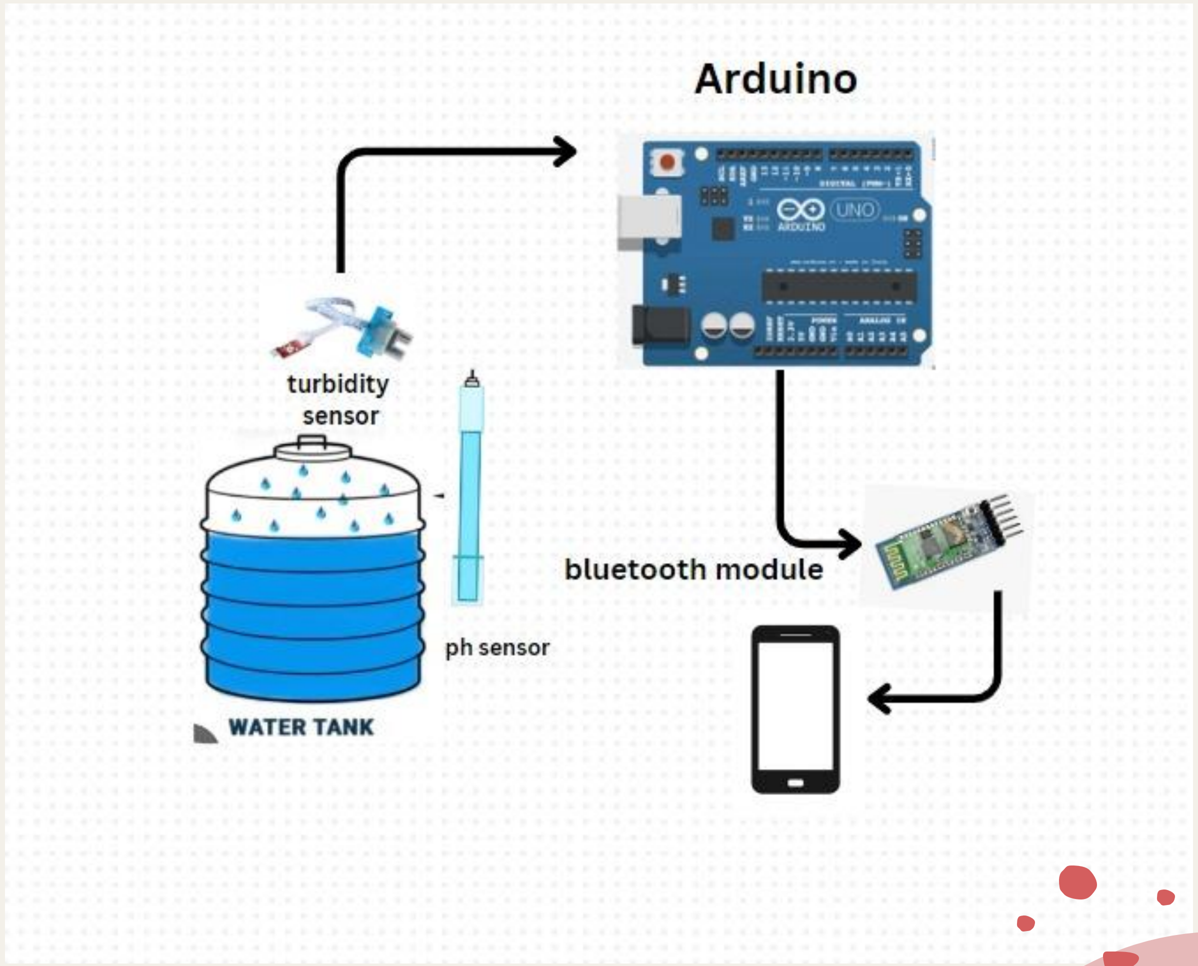
- We have implemented the LSTM autoencoder using keras library. We only consider a compressed version of input here
- As there would be different water requirements for each day so when ever the requirement has been higher than usual this would help in studying why this occurs



```python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, RepeatVector, TimeDistributed, Dense


def load_and_preprocess_data(file_path, time_steps=10):
    data = pd.read_csv(file_path, header=None)
    return np.array(data)


def calculate_threshold(X_train, model, percentile=95):

    reconstructions = model.predict(X_train)
    mse = np.mean(np.square(X_train - reconstructions), axis=(1, 2))


    threshold = np.percentile(mse, percentile)
    return threshold


def train_lstm_autoencoder(X_train, epochs=10, batch_size=32):
    model = Sequential()
    model.add(LSTM(units=64, input_shape=(X_train.shape[1], X_train.shape[2])))
    model.add(RepeatVector(X_train.shape[1]))
    model.add(LSTM(units=64, return_sequences=True))
    model.add(TimeDistributed(Dense(X_train.shape[2])))
    model.compile(optimizer='adam', loss='mse')
    model.fit(X_train, X_train, epochs=epochs, batch_size=batch_size)
    return model



# give the correct normalized train file path
train_file_path = r'/content/drive/MyDrive/S-1/Normal data/S1_3_train_normal.csv'
X_train = load_and_preprocess_data(train_file_path)


X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))


lstm_autoencoder_model = train_lstm_autoencoder(X_train)


threshold = calculate_threshold(X_train, lstm_autoencoder_model)

print(f"Threshold: {threshold}")
```

Progress

- Created the prototype that includes Turbidity sensor, Arduino uno and Bluetooth module.
- We have trained a basic decision model based on the training data, and then it makes predictions on the potability of water and usage of water.



Future Plans

- Develop a more concise app that can track many prototypes as we have shown in video to find easily at which junction has the leakage started.
- We plan to seamlessly integrate the trained machine learning algorithm into the Android application, thereby enabling real-time predictions of water potability and usability

Contributions

- Ankith Kumar 2101AI12
  Prototype Building and arduino code
  LSTM deep learning model
  APP
- Prakash Kumar 2101AI24
  Water quality prediction ML model
- Vinod Ramavath 2101AI27
  Help in APP

Thank You