

# **Optimizing Water Management Using LoRaWAN – Enabled IoT Framework and Behavioral Analysis**

*Report submitted to the SASTRA Deemed to be University  
As the requirement for the course*

## **CSE300 - MINI PROJECT**

*Submitted by*

**Harish T (126003101, CSE)**  
**Guru Prasath M (126003093, CSE)**  
**Nishanth M V (126003184, CSE)**

**May 2025**



**SASTRA**  
ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION  
**DEEMED TO BE UNIVERSITY**  
(U/S 3 of the UGC Act, 1956)



**THINK MERIT | THINK TRANSPARENCY | THINK SASTRA**

## Table of Contents

<b>Title</b>	<b>Page No.</b>
Bonafide Certificate	iii
Acknowledgements	iv
List of Figures	v
Abbreviations	vi
Abstract	vii
1. Introduction	1
2. Summary of the base paper	2
3. Merits and Demerits of the base paper	4
4. Implementation	5
5. Dataset Used	6
6. ML and DL Models Implemented	7
7. Source Code	8
8. Results	15
9. Efficiency	19
10. Inferences	20
11. Conclusion and Future Plans	21
12. References	22



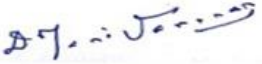
**SASTRA**  
ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION  
**DEEMED TO BE UNIVERSITY**  
(U/S 3 of the UGC Act, 1956)



THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

### **Bonafide Certificate**

This is to certify that the report titled “Optimizing Water Management Using LoRaWAN – Enabled IoT Framework and Behavioral Analysis” submitted as a requirement for the course, CSE300: MINI PROJECT for B.Tech is a Bonafide record of the work done by **T. Harish (126003101, CSE), M. Guru Prasath (126003093, CSE), M. V. Nishanth (126003184, CSE)** during the academic year 2024-25, in the School of Computing, under my supervision.

**Signature of Project Supervisor** :   
**Name with Affiliation** : Dr. D. Manivannan, Associate Dean, Infrastructure  
**Date** : 10.05.2025

Mini Project *Viva voce* held on\_\_\_\_\_

**Examiner 1**

**Examiner 2**

## Acknowledgements

We would like to thank our Honorable Chancellor Prof. **R. Sethuraman** for providing us with an opportunity and the necessary infrastructure for carrying out this project as a part of our curriculum.

We would like to thank our Honorable Vice-Chancellor **Dr. S. Vaidhyasubramaniam** and **Dr. S. Swaminathan**, Dean, Planning & Development, for the encouragement and strategic support at every step of our college life.

We extend our sincere thanks to **Dr. R. Chandramouli**, Registrar, SASTRA Deemed to be University for providing the opportunity to pursue this project.

We extend our heartfelt thanks to **Dr. V. S. Shankar Sriram**, Dean, School of Computing, **Dr. R. Muthaiah**, Associate Dean, Research, **Dr. K. Ramkumar**, Academics Associate Dean, **Dr. D. Manivannan**, Associate Dean, Infrastructure, **Dr. R. Alageswaran**, Associate Dean, Student Welfare.

Our guide **Dr. Manivannan D**, Associate Dean, Infrastructure, School of Computing was the driving force behind this whole idea from the start. His deep insight in the field and invaluable suggestions helped me/us in making progress throughout our project work. We also thank the project review panel members for their valuable comments and insights which made this project better.

We would like to extend our gratitude to all the teaching and non-teaching faculties of the School of Computing who have either directly or indirectly helped us in the completion of the project.

We gratefully acknowledge all the contributions and encouragement from my family and friends resulting in the successful completion of this project. We thank you all for providing us an opportunity to showcase our skills through project.

**List of Figures:**

<b>Figure No.</b>	<b>Title</b>	<b>Page No.</b>
5.a	Sample Dataset	6
7. a	Preprocessed Data Plot	9
7. b	1 <sup>st</sup> Differencing Plot	11
7. c	2 <sup>nd</sup> Differencing Plot	12
8. a	ARIMA Prediction v/s Actual Plot	15
8. b	LSTM Prediction v/s Actual Plot	16
8. c	GRU Prediction v/s Actual Plot	17
8. d	Hardware Setup Representation Diagram	18
8. e	Hardware Setup	18

**List of Tables:**

<b>Table No.</b>	<b>Title</b>	<b>Page No.</b>
8. a	Results from various optimizers (LSTM)	16
8. b	Results from various Optimizers (GRU)	17

## Abbreviations

<b>IoT</b>	Internet of Things
<b>CNN</b>	Convolutional Neural Network
<b>IWS</b>	Intermittent Water Supply
<b>CWS</b>	Continuous Water Supply
<b>OHT</b>	Overhead Tank
<b>RF</b>	Random Forest
<b>AMI</b>	Advanced Metering Infrastructure
<b>RoI</b>	Region of Interest
<b>MSE</b>	Mean Squared Error
<b>MAE</b>	Mean Absolute Error
<b>RMSE</b>	Root Mean Squared Error
<b>MSD</b>	Most Significant Digit
<b>VGG</b>	Visual Geometry Group (architecture)
<b>ResNet</b>	Residual Network (deep learning architecture)
<b>Pytorch</b>	Python-based Deep Learning Framework
<b>GRU</b>	Gated Recurrent Unit
<b>LSTM</b>	Long Short-Term Memory
<b>ARIMA</b>	AutoRegressive Integrated Moving Average
<b>DL</b>	Deep Learning

## Abstract

Sustainable water usage is an important factor in efficient resource management within extensive water supply networks. In this regard, the current study attempts to investigate water consumption behavior in the context of a university campus via the implementation of a cutting-edge IoT system driven by LoRaWAN technology for high-accuracy data collection. The system takes advantage of LoRaWAN-capable smart meters that are strategically placed throughout the water distribution network on campus, enabling secure real-time data transmission over long distances at low power. The low-power wide-area network architecture guarantees efficient and scalable data collection from points such as student hostels and faculty/staff quarters. The accumulated data from weekly and monthly usage is fed into deep learning algorithms that reveal important patterns of water use driven by events like the academic calendar, holidays, and occupancy variability. These data allow for nuanced insights into campus-wide water consumption patterns. Compared with conventional monitoring strategies, the addition of LoRaWAN further strengthens the system's scalability, cost-effectiveness, and robustness, especially for environments working under intermittent water supply systems (IWS). Furthermore, the study identifies the various campus areas have different patterns of usage, and this gives effective insights which can be utilized for water management plans and ensuring sustainable consumption patterns by the inhabitants in the campus.

**Keywords:** IoT-based Smart Retrofit Meter, Intermittent Water Supply, Hamming Distance Algorithm, Water Usage Patterns, Raspberry Pi

# CHAPTER 1

## 1.Introduction:

In the wake of increasing demand for efficient use of resources, particularly in big water distribution systems, there is a need for efficient monitoring of water usage. From this research, water usage behavior is assessed on a university campus following an enhanced IoT-based solution. The system utilizes intelligent retrofit meters, which are cost-effective devices mounted over installed analog water meters to be read digitally by taking pictures. The meters utilize deep learning-based algorithms to identify meter image digits and transmit processed data to a cloud server for real-time processing. To make sure that the readings are accurate, a Hamming distance-based data refining algorithm is used to recover errors due to environmental conditions such as dust, scratches, or darkness. The smart meters were installed in designated areas on campus, such as student hostels and staff and faculty residences, where they made high-frequency measurements for four months.

The analysis emerged with observable consumption patterns dependent on the academic calendar, holidays, and usage rates. Water usage in student hostels was characterized by extreme fluctuation, while usage in faculty/staff residences was more consistent. The system is under an Intermittent Water Supply (IWS) regime where water is supplied periodically and constant monitoring and forecasting more important. By employing deep learning models such as ResNet-18 and Raspberry Pi equipment, the solution provides a cost-effective, scalable, and trustworthy method for gathering and analyzing water usage data.

Unlike traditional systems, this IoT-based solution offers detailed behavioral data that can be used to form the basis of water management policy and encourage sustainable use. The research shows the versatility of the framework and proposes its applicability in other settings threatened by the same water distribution and resource optimization challenges



## CHAPTER 2

### 2. Summary of the base paper:

This work proposes an IoT-based solution for water consumption behavior analysis of a university campus. By equipping conventional analog water meters with smart devices that have cameras and Raspberry Pi microcontrollers, the system takes photographs of meter dials and employs deep learning (DL) techniques—namely a ResNet-18 model—for digit recognition. The readings are then corrected using a Hamming distance-based correction algorithm and transmitted to a cloud server for real-time analysis.

It took four months using data from strategically located smart meters in student hostels and staff/faculty quarters. It showed major variations in water use patterns, especially in student hostels, determined by academic schedules, holidays, and occupancy status. Staff/faculty quarters indicated more stable water use.

The system has an Intermittent Water Supply (IWS) framework, and in the research, the benefits of gathering high-frequency data to obtain a deeper perspective and manage the distribution of water are highlighted. The suggested technique enhances precision, is cost saving, and adaptable, thus, applicable for greater use in other institutions or municipal water systems. This research, apart from proof of technical innovativeness with smart metering, also addresses its capability in facilitating sustainable practices in water management.

### 2.1. Base Paper details:

**Title:** Behavioral Analysis of Water Consumption using IoT-based Smart Retrofit Meter

**Year:** 2024

**Journal Name:** IEEE Access (Volume:12), 2024

**ISSN NO:** 2169-3536

**Cite score:** 9.8 (2023)

**Impact Factor:** 3.4 (2023)

**Indexed in:** IEEE Xplore

## **2.2. Research addressed and solution proposed:**

### **Research Problem Addressed:**

- Inadequacy of high-frequency water consumption data in Intermittent Water Supply (IWS) systems, especially in developing regions.
- Analog meters still widely used, lacking digital real-time analytics.
- Inconsistent water usage data hampers sustainable water management.

### **Suggested solution:**

Low-cost IoT-based smart retrofit meter system for analog meters. Photographs analog dials. Deep learning ResNet-18 processes to detect digits. Hamming distance-based correction to correct misreadings. Digitized readings real-time sent to the cloud to be analyzed. Installed on an educational campus IIIT-H with 4 smart nodes over 4 months. 550,000 data points. Per week/per month behavioral trends weekly/monthly monitored for hostels vs. staff quarters under IWS.

## **2.3. Correctness of the system:**

Resolving errors corrected errors introduced by environment and individuals rain bugs orientation change outlier values were safely corrected by a hamming-based method useful implementation deployed on a real campus IWS network consistently monitored water consumption over a variety of time periods day week and month flexibility and scalability the retrofit model is low-cost and non-invasive scalable beyond campus settings and supports multiple analog meters

## **CHAPTER 3**

### **3. Merits and Demerits of the base paper:**

#### **3.1 Merits:**

1. Cost-Effective Retrofit Solution: Paves a low-cost route for retrofitting analog meters in place without complete replacement requirements.
2. Highly Accurate Digit Detection: ResNet-18 deep learning architecture attained ~99% accuracy; further improved using Hamming-distance-based correction.
3. Real-Time Water Monitoring: Allows uninterrupted, high-frequency monitoring of water consumption even in IWS systems.
4. Non-Invasive Retrofit Model: Simple-to-fit and non-invasive retrofit model on top of existing infrastructure.
5. Solid Error Handling: Fixes environmental and manual errors effectively through a refinement algorithm.
6. Scalability: Scalable to other residence locations or campuses with minimal modification.

#### **3.2 Demerits:**

1. Manual Maintenance Required: Devices needed to be manually brought in for maintenance during malfunctions or environmental damage.
2. No Output Flow Meters: The system only captures inflow; actual consumption within buildings is indirectly inferred.
3. Limited Instantaneous Consumption: IWS setup restricts ability to monitor real-time water usage at individual points.

## CHAPTER 4

### 4. Implementation:

#### Hardware Setup:

Retrofit Device: 3D-printed replica placed above standard analog water meters.

Camera Module: Raspberry Pi Camera (PiCam) takes photographs of meter dial.

Controller Raspberry Pi 3B+ handles the data locally.

Power Source: AC power with battery backup.

Lighting: LED ring to take good-quality photos at night.

#### Software Setup:

Image Processing

Digit Recognition

#### Data Correction:

Refinement Algorithm;

Hamming Distance-based to correct misread digits.

Dynamically adjusts according to time gap between readings.

Storage & Transmission:

Readings are digitized and sent to a cloud server for real-time monitoring and analysis.

#### Deployment Details:

Location: IIIT-Hyderabad campus.

Setup of Software

Image processing: OpenCV for calculating the Region of Interest (RoI).

Contour detection was utilized to recognize individual digits.

Digit Recognition: Segmented images are used to identify the digits using a deep learning model (ResNet-18).

## CHAPTER 5

### 5. Data Set Used:

- Real-time dataset of water flow from 4 different blocks in the university
- The dataset of each block consists of 7 attributes and 1865 instances.
- There are 2 datetime attributes, 1 categorical attribute and 4 numeric attributes.

	A	B	C	D	E	F	G	H	I
	Serial No.	Date	Time	Device Name	battery	raw_data	water_flow_in_cubic_meter	water_flow_in_litre	
	1	31-12-2024	23:32:03	SOC Outlet-01	100	4696.7	4696.7	4.70E+06	
	2	31-12-2024	17:32:03	SOC Outlet-01	100	4695.1	4695.1	4.70E+06	
	3	31-12-2024	11:32:03	SOC Outlet-01	100	4692.5	4692.5	4.69E+06	
	4	31-12-2024	05:32:03	SOC Outlet-01	100	4689.2	4689.2	4.69E+06	
	5	31-12-2024	00:16:38	SOC Outlet-01	100	4689.1	4689.1	4.69E+06	
	6	30-12-2024	23:32:03	SOC Outlet-01	100	4689.1	4689.1	4.69E+06	
	7	30-12-2024	17:32:02	SOC Outlet-01	100	4689	4689	4.69E+06	
	8	30-12-2024	11:32:02	SOC Outlet-01	100	4688.5	4688.5	4.69E+06	
0	9	30-12-2024	05:32:03	SOC Outlet-01	100	4688.2	4688.2	4.69E+06	
1	10	30-12-2024	00:16:38	SOC Outlet-01	100	4688	4688	4.69E+06	
2	11	29-12-2024	23:32:02	SOC Outlet-01	100	4688	4688	4.69E+06	

*Figure 5. a*

## CHAPTER 6

### 6. ML and DL Models Implemented:

#### ARIMA (Autoregressive Integrated Moving Average):

- The ARIMA model predicts a given time series based on its own past values.
- It can be used for any nonseasonal series of numbers that exhibits patterns and is not a series of random events.
- In ARIMA modeling, differencing transforms non-stationary time series data into a stationary form by removing trends and making the data more stable, allowing for better forecasting.

#### Long Short-Term Memory (LSTM):

- It is a sequential learning model which can establish temporal correlations between a previous instant  $t-1$  and a current instant  $t$ .
- Consequently, the LSTM seems the **most suitable model** for forecasting consumption processes, given its ability to deduce the intrinsic daily consumption resident routines.
- The LSTM is based on the **Back-Propagation Through Time** (BPTT) learning algorithm to calculate the weights. It is made up of units called memory blocks. Each memory block contains an “input gate”, an “output gate” and a “forget gate”.

#### GRU (Gated Recurrent Unit):

- GRU is designed to model sequential data by allowing information to be selectively remembered or forgotten over time. However, GRU has a simpler architecture than LSTM, with fewer parameters, which can make it easier to train and more computationally efficient.
- In GRU, the memory cell state is replaced with a “candidate activation vector,” which is updated using two gates: the reset gate and update gate.

## CHAPTER 7

### 7. Source Code:

#### Data Preprocessing:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.arima.model import ARIMA
# Load dataset
water_usage = pd.read_csv('Water Usage/SOC Outlet-1/Table(1).csv',
                          usecols=["Date", "Time", "Device Name", "water_flow_in_cubic_meter"])

# Reverse order if needed
data = water_usage[::-1]

# Set index and ensure DateTime format
data["Timestamp"] = pd.to_datetime(data["Date"] + " " + data["Time"],format="%d/%m/%Y
%H:%M:%S")
data["Rounded_Timestamp"] = data["Timestamp"].dt.round("6H")

# Set the new Timestamp as index
data.set_index("Rounded_Timestamp", inplace=True)

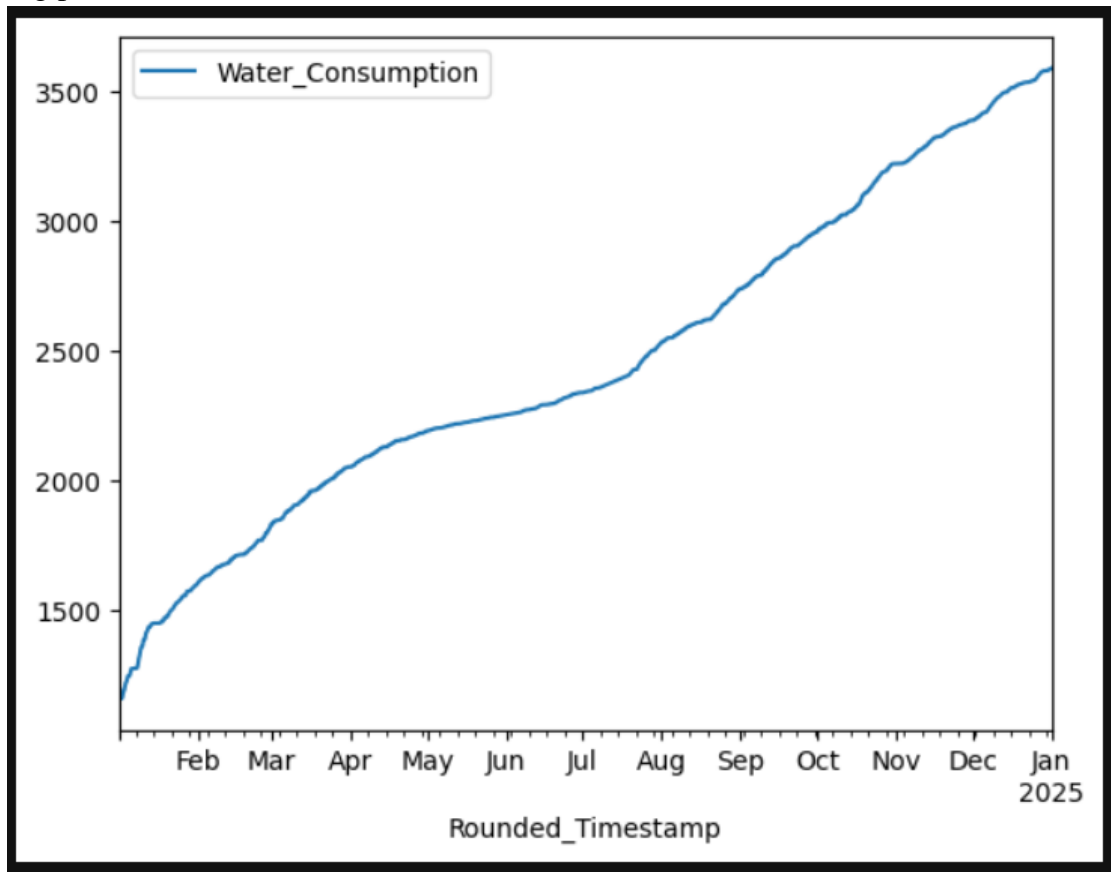
# Drop the original "Date" and "Time" columns if no longer needed

data.drop(columns=["Date", "Time", "Timestamp", "Device Name"], inplace=True)

# Ensure daily frequency
data = data.resample("6H").mean()
```

```
# Rename column
data.rename(columns={'water_flow_in_cubic_meter': 'Water_Consumption'}, inplace=True)
print(data.isna().sum())
data.interpolate(method="linear", inplace=True)

# Display first 20 rows
print(data.head(20))
data.plot()
data_log=np.log(data)
data_log.plot()
```



*Figure 7. a*

### Scaling and creating input sequence:

```
# Normalize the water consumption data
scaler = MinMaxScaler(feature_range=(0, 1))
```



```
data["Water_Consumption"] = scaler.fit_transform(data[["Water_Consumption"]])
original_timestamps = data.index # Save timestamps before preprocessing
```

```
# Converts data into sequences for LSTM
```

```
def create_sequences(data, time_steps=10):
```

```
    """ Create input-output pairs for LSTM training. """
```

```
    X, y = [], []
```

```
    for i in range(len(data) - time_steps):
```

```
        X.append(data[i : i + time_steps]) # Use past 'time_steps' values as input
```

```
        y.append(data[i + time_steps])    # Predict the next value
```

```
    X, y = np.array(X), np.array(y)
```

```
# Ensure 3D shape: (samples, time_steps, features=1)
```

```
X = X.reshape((X.shape[0], X.shape[1], 1))
```

```
return X, y
```

```
# Define time steps (e.g., past 7 readings)
```

```
time_steps = 7
```

```
data_values=data["Water_Consumption"].values
```

```
X, y = create_sequences(data_values, time_steps)
```

```
# Print shape (should be 3D: samples, time_steps, features)
```

```
print("X shape:", X.shape) # (num_samples, time_steps, 1)
```

```
print("y shape:", y.shape) # (num_samples,)
```

## Splitting the data:

```
# Define split index (80% training, 20% testing)
```

```
train_size = int(len(X) * 0.8)
```

```
# Split into training and testing sets
```

```
X_train, X_test = X[:train_size], X[train_size:]
```

```
y_train, y_test = y[:train_size], y[train_size:]
```

```
# Verify final shapes
```

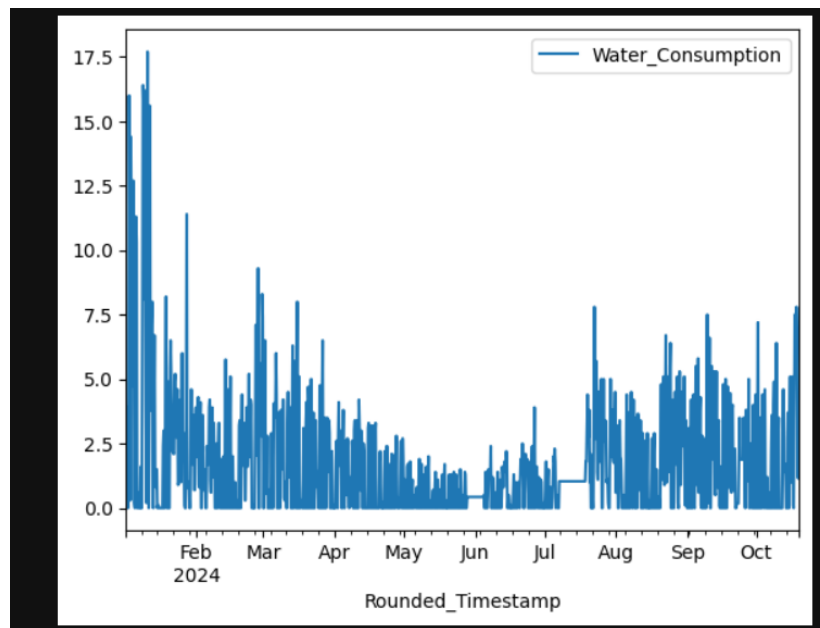
```
print(f"X_train shape: {X_train.shape}, y_train shape: {y_train.shape}")
```

```
print(f"X_test shape: {X_test.shape}, y_test shape: {y_test.shape}")
```

## ARIMA Model:

### Differencing:

```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
acf_original = plot_acf(train_data)
pacf_original = plot_pacf(train_data)
from statsmodels.tsa.stattools import adfuller
adf_test = adfuller(train_data)
print(f'p-value: {adf_test[1]}')
df_train_diff = train_data.diff().dropna()
df_train_diff.plot()
```

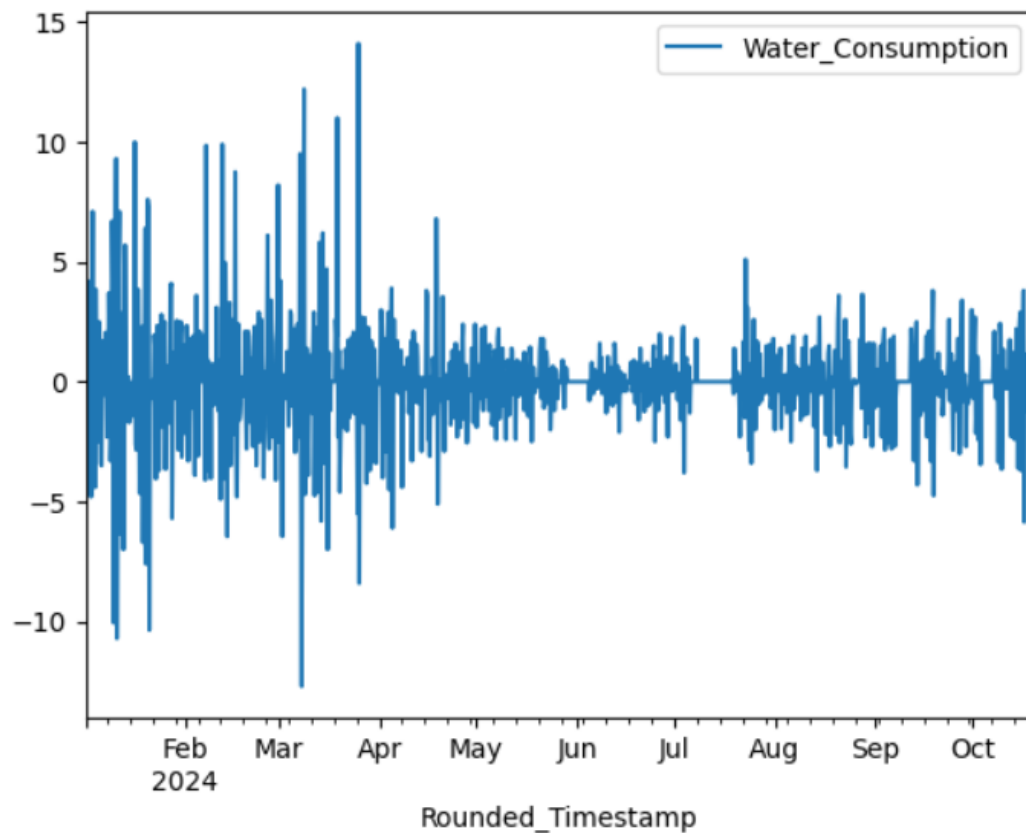


*Figure 7. b*

```
acf_diff = plot_acf(df_train_diff)
pacf_diff = plot_pacf(df_train_diff)
adf_test = adfuller(df_train_diff)
print(f'p-value: {adf_test[1]}')

df_train_diff2 = df_train_diff.diff().dropna()
df_train_diff2.plot()
```

```
[10]: <Axes: xlabel='Rounded_Timestamp'>
```



*Figure 7. c*

```
acf_diff2 = plot_acf(df_train_diff2)
pacf_diff2 = plot_pacf(df_train_diff2)
adf_test2 = adfuller(df_train_diff2)
print(f'p-value: {adf_test2[1]}')
```

### **Model Training:**

```
from statsmodels.tsa.arima.model import ARIMA

model = ARIMA(train_data, order=(4,2,3))
model_fit = model.fit()
print(model_fit.summary())
```

```

import matplotlib.pyplot as plt

residuals = model_fit.resid[1:]
plt.figure(figsize=(12,5))
fig, ax = plt.subplots(1,2)
residuals.plot(title='Residuals', ax=ax[0])
residuals.plot(title='Density', kind='kde', ax=ax[1])
plt.show()

acf_res = plot_acf(residuals)
pacf_res = plot_pacf(residuals)
forecast_test = model_fit.forecast(len(test_data))
data['forecast_manual'] = [None]*len(train_data) + list(forecast_test)
data.plot()

```

### **Optimizers:**

```

optimizers = {
    "Adam": Adam(learning_rate=0.001),
    "SGD": SGD(learning_rate=0.001, momentum=0.9),
    "RMSprop": RMSprop(learning_rate=0.0001),
    "Adagrad": Adagrad(learning_rate=0.001),
    "Adadelata": Adadelata(learning_rate=0.001)
}

```

### **Plotting:**

```

from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error,
mean_squared_error, r2_score
mae = mean_absolute_error(test_data, forecast_test)
mape = mean_absolute_percentage_error(test_data, forecast_test)
rmse = np.sqrt(mean_squared_error(test_data, forecast_test))
r2 = r2_score(test_data, forecast_test)
#acc=100*(1-(rmse/len(test_data)))
print(f'mae - manual: {mae}')
print(f'mape - manual: {mape}')
print(f'rmse - manual: {rmse}')
print(f'r2-score (closer to 1, more the model is preferred) : {r2}')

```

```

# Plot Only Test Data & Predictions
plt.figure(figsize=(10, 5))
plt.plot(range(len(train_data), len(data)), test_data, label="Actual Test Data", color="blue")
plt.plot(range(len(train_data), len(data)), forecast_test, label="Predictions", color="red",
linestyle="dashed")
plt.xlabel("Time")
plt.ylabel("Water Consumption") # Change based on your dataset
plt.title("ARIMA Model - Test Data & Predictions")
plt.legend()

# Display Metrics on Plot
metrics_text = f"MAE: {mae:.2f}\nRMSE: {rmse:.2f}\nR² Score: {r2:.2f}"
plt.text(0.05, 0.7, metrics_text, transform=plt.gca().transAxes, fontsize=12,
        bbox=dict(facecolor='white', alpha=0.8, edgecolor='black'))
plt.show()

```

### **LSTM Model:**

```

def build_lstm_model(optimizer):
    model = Sequential([
        LSTM(50, activation='relu', return_sequences=True, input_shape=(7, 1)),
        LSTM(50, activation='relu'),
        Dense(1)
    ])
    model.compile(optimizer=optimizer, loss='mse')
    return model

```

### **GRU Model:**

```

def build_gru_model(optimizer):
    model = Sequential([
        GRU(50, activation='linear', return_sequences=True, input_shape=(7, 1)),
        GRU(50, activation='linear'),
        Dense(1)
    ])
    model.compile(optimizer=optimizer, loss='mse')
    return model

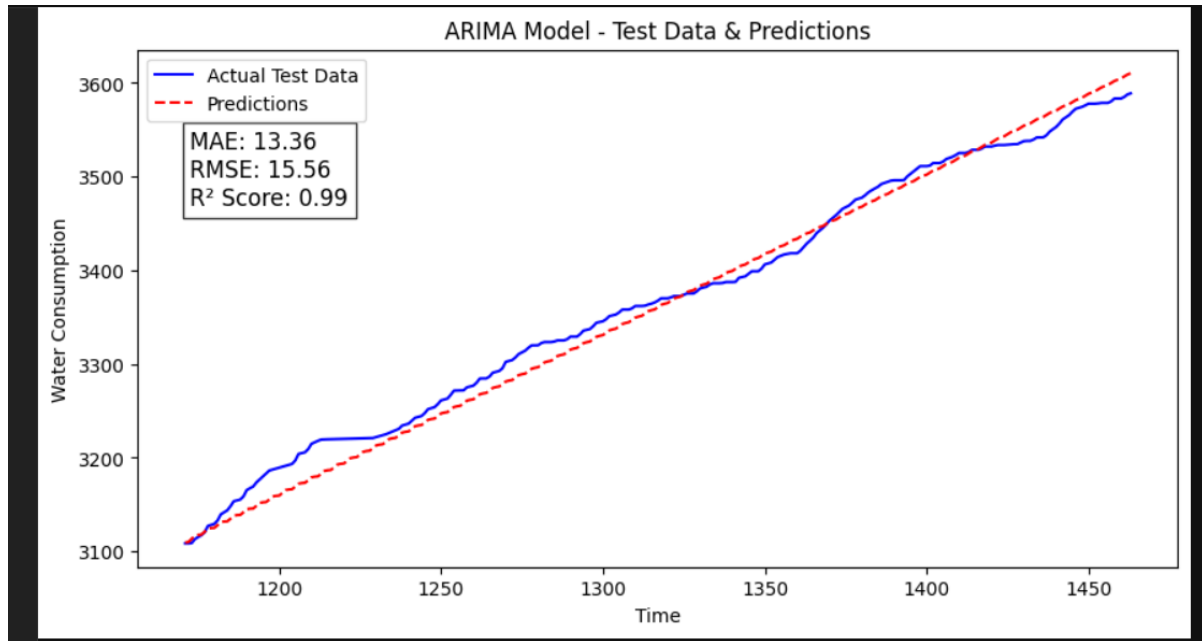
```

## CHAPTER 8

### 8.Results:

#### ARIMA Model:

##### Prediction v/s Actual Plot



*Figure 8. a*

## LSTM:

### Prediction v/s Actual Plot

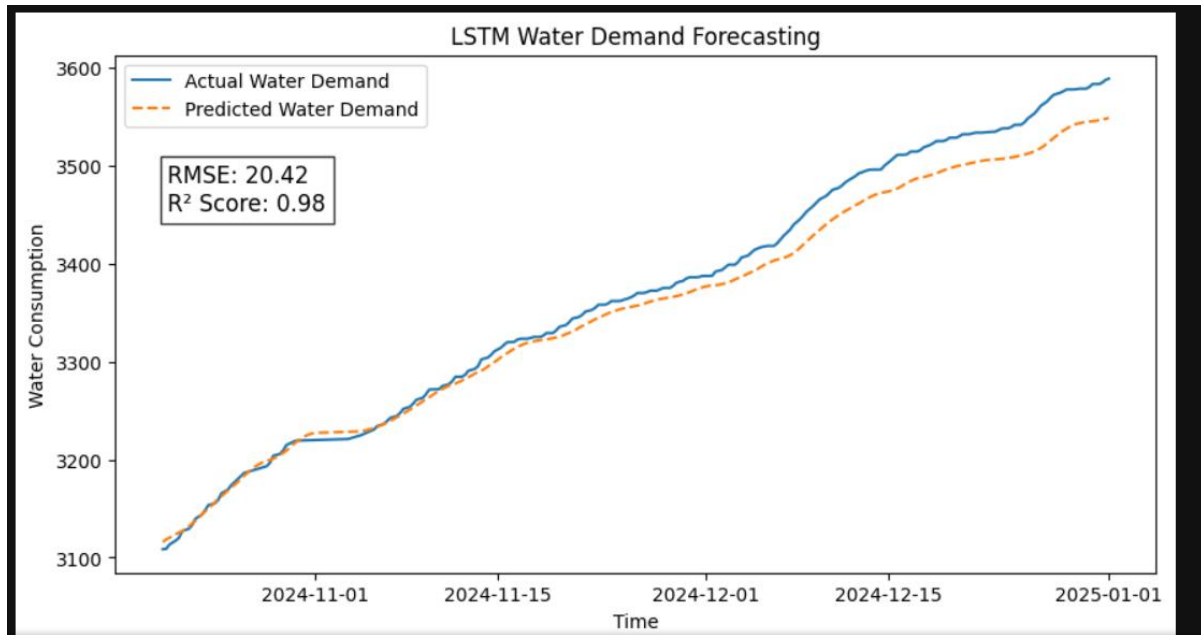


Figure 8. b

Optimizers	Parameters	RMSE / R <sup>2</sup> score
<b>Adam</b>	Learning Rate = 0.001	RMSE – 31.03 R <sup>2</sup> – 0.9464
<b>SGD</b>	Learning Rate = 0.001, Momentum = 0.9	RMSE – 210.67 R <sup>2</sup> – -1.46799
<b>RMSProp</b>	Learning Rate = 0.001	RMSE – 49.10 R <sup>2</sup> – 0.8659
<b>Adagrad</b>	Learning Rate = 0.001	RMSE – 826.84 R <sup>2</sup> – -37.0169
<b>Adadelata</b>	Learning Rate = 0.001	RMSE – 2122.12 R <sup>2</sup> – -249.419

Table 8. a

## GRU Model:

### Prediction v/s Actual Plot

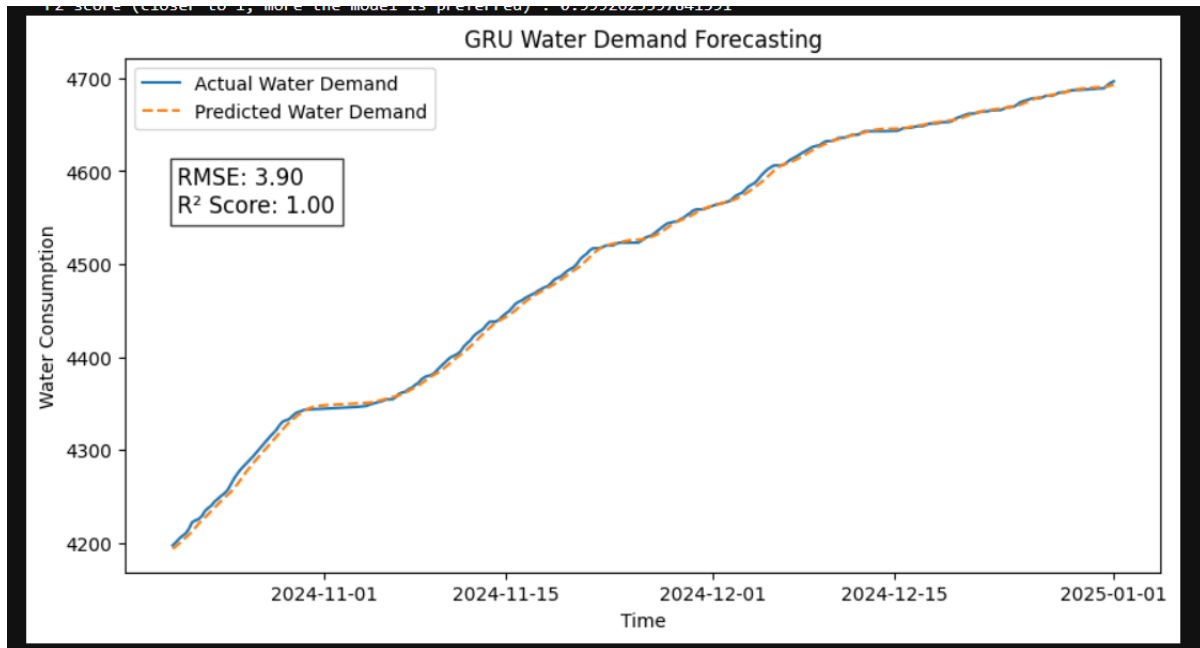


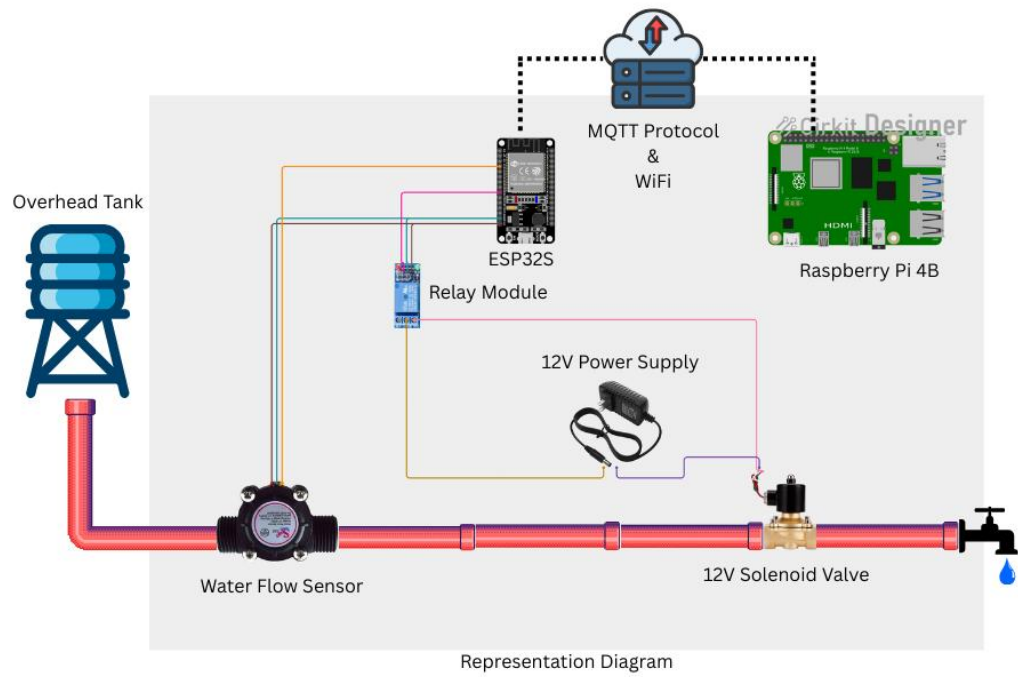
Figure 8. c

Optimizers	Parameters	RMSE / R <sup>2</sup> score
<b>Adam</b>	Learning Rate = 0.001	RMSE – 4.82 R <sup>2</sup> – 0.998
<b>SGD</b>	Learning Rate = 0.001, Momentum = 0.9	RMSE – 7.37 R <sup>2</sup> - 0.997
<b>RMSProp</b>	Learning Rate = 0.001	RMSE – 9.97 R <sup>2</sup> - 0.995
<b>Adagrad</b>	Learning Rate = 0.001	RMSE – 311.99 R <sup>2</sup> – -3.7130
<b>Adadelat</b>	Learning Rate = 0.001	RMSE – 1782.78 R <sup>2</sup> – -152.891

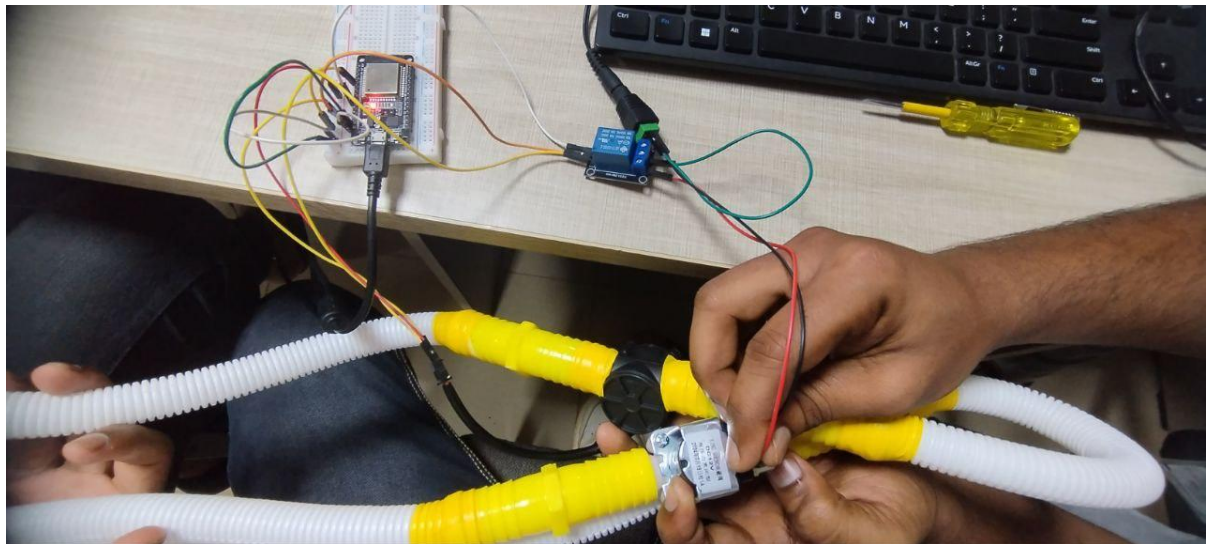
Table 8. b



## Hardware Setup:



*Figure 8. d*



*Figure 8. e*

## **CHAPTER 9**

### **9.Efficiency of Implementation over existing system (Base Paper):**

#### **Error Correction Mechanism:**

Integration of a Hamming distance-based optimization algorithm helps to correct errors caused by environmental contaminants like dust, light, or smudges—something legacy systems might not necessarily do.

#### **Real-Time, High-Frequency Data:**

Unlike legacy systems that are prone to having data collected at long intervals or by manual process, this system collects and sends data on one-minute intervals, making analysis comprehensive and timely.

#### **Behavioral Insights:**

It goes beyond simple raw data collection to providing behavioral insights of water consumption based on occupancy, holidays, and school schedules—facilitating smarter decision-making in water management.

#### **Flexibility in IWS Systems:**

Whereas many high-frequency monitoring systems are designed for Continuous Water Supply (CWS), the approach is more naturally adapted to Intermittent Water Supply (IWS), more common in developing nations.

## CHAPTER 10

### 10.Inferences:

**Smart Water Management is Achievable:** Real-time monitoring and predictive models allow dynamic, efficient water control in pipelines, tanks, or industrial systems.

**AI-Driven Forecasting Enhances Efficiency:** GRU-based predictions help pre-empt overflows or unusual usage patterns, enabling preventive actions.

**Seamless Integration of IoT and ML:** The combination of ESP32, Raspberry Pi, MQTT, and cloud platforms demonstrates how edge devices and intelligence can work together in real-time.

**Automation Reduces Operational Complexity:** Systems can automatically adjust outputs (like relay activation) without manual input, improving reliability.

**Scalable Solution for Urban & Industrial Applications:** The architecture supports extension to smart cities, buildings, or manufacturing systems with minimal modifications.

## **CHAPTER 11**

### **11.Conclusion & future work:**

#### **Conclusion:**

This paper introduces an efficient IoT-based monitoring and analysis of water consumption in a university campus. Through smart retrofit meters and deep learning models, the system reliably measures and processes water consumption data. It is demonstrated that water consumption patterns depend on issues such as holidays, academic calendars, and occupancy rates. The results facilitate an understanding of these patterns and enhance informed decisions regarding water management. Overall, the system is cost-effective, scalable, and appropriate for enhancing water use in intermittently supplied areas.

#### **Future Work:**

The system can be further augmented with predictive analytics to predict future water demand against past trends and external factors such as exams or seasonal fluctuations. Investigating the application of edge computing alongside fog and cloud layers has the potential to further decrease latency and enhance scalability. Lastly, incorporating the system into smart city infrastructures and municipal water supply systems can facilitate wider application of sustainable water management practices.

## CHAPTER 12

### 12.References:

- D. Li and Q. Fu, "Deep Learning Model-Based Demand Forecasting for Secondary Water Supply in Residential Communities: A Case Study of Shanghai City, China," in *IEEE Access*, vol. 12, pp. 38745-38757, 2024.
- A. Gil-Gamboa, P. Paneque, O. Trull, A. Troncoso, Medium-term water consumption forecasting based on deep neural networks, *Expert Systems with Applications*, Volume 247, 2024.
- Kim, J.; Lee, H.; Lee, M.; Han, H.; Kim, D.; Kim, H.S. Development of a Deep Learning-Based Prediction Model for Water Consumption at the Household Level. *Water* 2022.
- Bata, M., Carriveau, R. & Ting, D.SK. Short-term water demand forecasting using hybrid supervised and unsupervised machine learning model. *Smart Water* 5, 2 (2020).