DESIGN AND DEVELOPMENT OF MONITORING SYSTEM USING ARDUINO FOR PALM OIL MILL EFFLUENT (POME) IN DARK ENVIRONMENT OF ANAEROBIC DIGESTER

MUHAMMAD AMIRUL NAQIL BIN MOHD NOHAN

UNIVERSITI TEKNOLOGI MALAYSIA

PSZ 19:16 (Pind. 1/23)

UNIVERSITI TEKNOLOGI MALAYSIA
DECLARATION OF UNDERGRADUATE REPORT

Author's full name	:	Muhammad Amirul Naqil bin Mohd Nohan			
Student's Matric No.	:	A21EE0127	Academic Session	:	20242025-2
Date of Birth	:	6 February	UTM Email	:	muhammadamirulnaqil@graduate.utm.my
Report Title	:	DESIGN AND DEVELOPMENT OF MONITORING SYSTEM USING ARDUINO FOR PALM OIL MILL EFFLUENT (POME) IN DARK ENVIRONMENT OF ANAEROBIC DIGESTER			

I declare that this thesis is classified as:

☒	**OPEN ACCESS**	I agree that my report to be published as a hard copy or made available through online open access.
☐	**RESTRICTED**	Contains restricted information as specified by the organization/institution where research was done. *(The library will block access for up to three (3) years)*
☐	**CONFIDENTIAL**	Contains confidential information as specified in the Official Secret Act 1972)

(If none of the options are selected, the first option will be chosen by default)

I acknowledged the intellectual property in the report belongs to Universiti Teknologi Malaysia, and I agree to allow this to be placed in the library under the following terms :

1. This is the property of Universiti Teknologi Malaysia
2. The Library of Universiti Teknologi Malaysia has the right to make copies for the purpose of only.
3. The Library of Universiti Teknologi Malaysia is allowed to make copies of this report for academic exchange.

Signature of Student:

Signature :

Full Name : MUHAMMAD AMIRUL NAQIL BIN MOHD NOHAN
Date : 2 JULY 2025

Approved by Supervisor(s)

Signature of Supervisor I:	Signature of Supervisor II
Full Name of Supervisor I DR. KHAIRUL HAMIMAH BINTI ABAS	Full Name of Supervisor II
Date : 2 JULY 2025	Date :

NOTES : If the thesis is CONFIDENTIAL or RESTRICTED, please attach with the letter from the organization with period and reasons for confidentiality or restriction

This letter should be written by a supervisor and addressed to Perpustakaan UTM. A copy of this letter should be attached to the thesis.

Date:
Librarian
Jabatan Perpustakaan UTM,
Universiti Teknologi Malaysia,
Johor Bahru, Johor

Sir,

CLASSIFICATION OF THESIS AS RESTRICTED/CONFIDENTIAL
TITLE: Click or tap here to enter text.
AUTHOR'S FULL NAME: Click or tap here to enter text.

Please be informed that the above-mentioned thesis titled _________ should be classified as RESTRICTED/CONFIDENTIAL for a period of three (3) years from the date of this letter. The reasons for this classification are
(i)
(ii)
(iii)

Thank you.

Yours sincerely,

SIGNATURE:
NAME:
ADDRESS OF SUPERVISOR:

"I hereby declare that I have read this report and in my
opinion this report is sufficient in term of scope and quality for the
award of the degree of Bachelor of Engineering (Electrical-Mechatronics) with
Honours"

Signature	:	___________________________
Name of Supervisor I	:	DR. KHAIRUL HAMIMAH BINTI ABAS
Date	:	2 JULY 2025

Declaration of Cooperation

This is to confirm that this research has been conducted through a collaboration Click or tap here to enter text. and Click or tap here to enter text.

Certified by:

Signature	:
Name	:
Position	:
Official Stamp	
Date	

* This section is to be filled up for theses with industrial collaboration

DESIGN AND DEVELOPMENT OF MONITORING SYSTEM USING ARDUINO FOR PALM OIL MILL EFFLUENT (POME) IN DARK ENVIRONMENT OF ANAEROBIC DIGESTER

MUHAMMAD AMIRUL NAQIL BIN MOHD NOHAN

A report submitted in partial fulfilment of the
requirements for the award of the degree of
Bachelor of Engineering (Electrical-Mechatronics) with Honours

Faculty of Electrical Engineering
Universiti Teknologi Malaysia

JULY 2025

DECLARATION

I declare that this report entitled *"Design and Development of Monitoring System for Palm Oil Mill Effluent (POME) in Dark Environment of Anaerobic Digester"* is the result of my own research except as cited in the references. The report has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature	:	....
Name	:	MUHAMMAD AMIRUL NAQIL BIN MOHD NOHAN
Date	:	2 JULY 2025

ACKNOWLEDGEMENT

I would like to begin by expressing my sincere gratitude to my supervisor, Ir. Dr. Khairul Hamimah binti Abas, for her continuous support, valuable guidance, and insightful advice throughout my final year project 1. Her patience and dedication, especially when I faced difficulties in conducting programming and testing my machine learning model, were essential in helping me stay on track and achieve my project goals. Her encouragement has not only guided me academically but also motivated me to keep going despite the challenges.

I would also like to extend my heartfelt thanks to my family for their unwavering support and belief in my abilities. Their constant encouragement has been a great source of motivation, pushing me to give my best throughout this journey.

I am also extremely grateful to Dr. Safia for her guidance in setting up the experiments for the preliminary results. Her assistance in this part of my project was crucial and much appreciated.

Lastly, I want to express my gratitude to my friends, whose support and advice have been invaluable. They were always there to lend a hand whenever I faced challenges, making this journey more enjoyable and manageable.

ABSTRACT

Malaysia is the second-largest producer of palm oil globally, yet the processing of Palm Oil Mill Effluent (POME) remains heavily dependent on manual labour, which can be inefficient and time-consuming. POME is a highly polluting byproduct, with elevated levels of Chemical Oxygen Demand (COD) and Biological Oxygen Demand (BOD), which pose significant risks to ecosystems if untreated. This project aims to automate and optimize the POME treatment process in an anaerobic digester by developing a sensor-based monitoring system for sludge detection. The system integrates turbidity, temperature, and methane sensors to analyze and correlate these parameters with sludge concentration, providing monitoring and reducing human supervision. A machine learning model, trained on data collected from lab-scale experiments, was used to predict sludge concentration with improved precision. The project employs a structured methodology, including sample preparation, sensor calibration, and the implementation of a linear regression model to demonstrate the feasibility of correlating turbidity levels with sludge concentration. Preliminary results showed a strong correlation between sludge concentration and turbidity, enabling accurate predictions of sludge levels. This automation could streamline POME treatment, ensuring better environmental compliance and operational efficiency. By reducing reliance on manual monitoring and minimizing environmental hazards, this system provides a scalable and sustainable solution for the palm oil industry, highlighting the potential for broader applications in wastewater management.

ABSTRAK

Malaysia merupakan pengeluar kedua terbesar minyak sawit di dunia, namun pemprosesan sisa buangan kilang kelapa sawit (POME) masih banyak bergantung kepada tenaga manusia, yang tidak cekap dan memakan masa. POME merupakan bahan buangan yang sangat mencemarkan, dengan tahap Keperluan Oksigen Kimia (COD) dan Keperluan Oksigen Biologi (BOD) yang tinggi, yang boleh memberi kesan negatif kepada ekosistem jika tidak dirawat dengan betul. Projek ini bertujuan untuk mengautomasi dan mengoptimumkan proses rawatan POME dalam pencerna anaerobik dengan membangunkan sistem pemantauan berasaskan sensor untuk mengesan enap cemar. Sistem ini mengintegrasikan sensor kekeruhan, suhu, dan metana untuk menganalisis dan menghubungkan parameter ini dengan kepekatan enap cemar, menyediakan pemantauan masa nyata dan mengurangkan keperluan pengawasan manusia. Model pembelajaran mesin telah dilatih berdasarkan data eksperimen berskala makmal untuk meramal kepekatan enap cemar dengan ketepatan yang lebih tinggi. Projek ini menggunakan pendekatan metodologi yang terancang, termasuk penyediaan sampel, penentukuran sensor, dan pelaksanaan model regresi linear untuk menunjukkan kebolehlaksanaan mengaitkan tahap kekeruhan dengan kepekatan enap cemar. Hasil awal menunjukkan hubungan yang kuat antara kepekatan enap cemar dan kekeruhan, membolehkan ramalan tahap enap cemar yang tepat. Automasi ini dapat menyelaraskan rawatan POME, memastikan pematuhan alam sekitar yang lebih baik dan meningkatkan kecekapan operasi. Dengan mengurangkan kebergantungan pada pemantauan manual dan meminimumkan bahaya alam sekitar, sistem ini menyediakan penyelesaian yang berskala dan mampan untuk industri minyak sawit, serta menunjukkan potensi untuk aplikasi yang lebih luas dalam pengurusan air sisa.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

POME	-	Palm Oil Mill Effluent
BOD	-	Biochemical Oxygen Demand
COD	-	Chemical Oxygen Demand
CO2	-	Carbon Dioxide
AD	-	Anaerobic Digester
IoT	-	Internet of Things
UTM	-	Universiti Teknologi Malaysia
DO	-	Dissolved Oxygen
TDS	-	Total Dissolved Solids
SPM	-	Suspended Particulate Matter
SSC	-	Suspended Sediment Concentration
CSTR	-	Continuous Stirred Tank Reactor
ML	-	Machine Learning
MLP	-	Multilayer Perceptron
GFFR	-	Generalized Feedforward Regression
RBF	-	Radial Basis Function
NRMSE	-	Normalized Root Mean Square Error
NMAE	-	Normalized Mean Absolute Error
BNN	-	Bilateral Neural Networks
RMSE	-	Root Mean Square Error Learning
MAE	-	Mean Absolute Error
NTU	-	Nephelometric Turbidity Unit
GUI	-	Graphical User Interface
TCP/IP	-	Terminal Communication Protocol/ Internet Protocol
ADC	-	Analog to Digital
CSV	-	Comma Separated Values

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Problem Background

Palm Oil Mill Effluent (POME) is a significant byproduct of the palm oil processing industry, especially in Malaysia, which is the second-largest palm oil producer globally. One of the main causes of pollution is POME which consists of high amounts of several pollutants such as Biochemical Oxygen Demand (BOD), Chemical Oxygen Demand (COD), total suspended solids, and oil content, that can significantly threaten water bodies and ecosystems if untreated [2]. Traditional treatment processes such as open ponding systems are recognized as less efficient and not environmentally sustainable because of the high methane emissions they emit and their inability to meet stringent discharge standards [2].

The recent progress made in POME treatment technologies including anaerobic digestion and membrane filtration, as well as photocatalysis, has demonstrated its ability to overcome these challenges. Anaerobic digester tank systems, for example, have reduced COD by 85% and BOD by 88%, with low CO2 releases [3]. In addition, integrated systems that combine different methods have been reported to successfully remove greater than 90% of colour, COD and total suspended solids [2]. Innovative approaches, such as phycoremediation using microalgae species like Chlorella Sorokiniana UKM2 and Chlorella Pyrenoidosa UKM7, offer dual benefits of POME treatment and CO2 fixation, enhancing the sustainability of the palm oil industry [4].

Monitoring sludge concentration during anaerobic digestion (AD) is crucial for optimizing biogas production and wastewater treatment. Accurate sludge measurement helps in enhancing biogas yields, as seen in studies where optimising solid content boosted fermentation efficiency [5]. Furthermore, continuous monitoring enables process optimization by adjusting organic loading rates based on volatile fatty acids [6].

In this project, the primary aim is to develop a sensor-based monitoring system to predict sludge levels in Palm Oil Mill Effluent (POME) during its treatment process in an anaerobic digestor. Multiple sensors will be employed to measure turbidity, temperature, and methane levels in POME samples to associate turbidity with the sludge concentration in POME. The system will then correlate turbidity values with temperature and methane levels and test to estimate the sludge levels in the sample. This data will be used to train a machine-learning model and predict the level of sludge present, based on new readings of turbidity, temperature, and methane.

This project aims to automate the sludge detection process to reduce the reliance on human supervision. Operators will be able to use these reliable predictions from the machine learning model to decide if the treatment process stage is achieved and manage POME treatment systems more effectively. This project will support the development of more efficient and sustainable wastewater treatment within the palm oil industry.

1.2 Problem Statement

How can we monitor turbidity, temperature and methane levels of Palm Oil Mill Effluent (POME) in the dark environment of an anaerobic digester to effectively track sludge formation, ensure proper sludge management, and prevent system inefficiencies and environmental hazards?

1.3 Research Objectives

The objectives of this project are:

(a) To study the correlation between fermentation parameters (turbidity, temperature, methane levels) and sludge concentration.

(b) To develop a monitoring system for POME in the dark environment of anaerobic digester.

(c) To implement machine learning models for predicting sludge concentration.

1.4 Scope of Project

The scopes of this project are:

(a) Constrained sample size and experimental setup under laboratory conditions.

(b) Parameters to be measured are only turbidity and methane levels.

(c) Stirring samples at a fixed rpm when measuring the parameters to ensure consistency.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This section reviews previous studies and methodologies related to the monitoring in the wastewater industry. The primary goal of this review is to analyse techniques and technologies employed in sludge detection, turbidity measurement and machine learning applications for wastewater quality prediction. POME treatment presents unique challenges due to its variability in composition, high turbidity, and the need for accurate sludge-level detection. This chapter focuses on understanding the methodologies used in these studies, identifying gaps, and highlighting their relevance to developing an automated system for POME sludge-level prediction.

2.2 Wastewater Treatment Monitoring

Wastewater treatment monitoring is a critical component in ensuring the quality and safety of water released into the environment. Recent advancements in technology have significantly enhanced the ability to monitor various parameters of wastewater treatment processes. These advancements include the use of low-cost sensors, IoT-based systems, multivariate statistical process control, and innovative sensor technologies. Each of these approaches offers unique benefits and challenges, contributing to a comprehensive understanding of wastewater treatment monitoring.

Literature [7-9] discusses the integration of Internet of Things (IoT) technology into wastewater treatment monitoring, highlighting its impact on real-time water quality management. IoT systems enable the continuous collection, analysis, and visualization of wastewater parameters such as dissolved oxygen (DO), total dissolved solids (TDS), pH, temperature, and turbidity. These parameters are essential for assessing water quality and determining necessary treatment interventions. Literature [10-12] emphasizes the use of low-cost sensors and microcontrollers, such as ESP32 and NodeMCU, to collect data, which is transmitted to cloud servers for storage and analysis. The system allows for remote access and management of wastewater treatment processes. These systems also improve operational efficiency by reducing the time and cost associated with manual data collection and analysis [13].

Literature [7, 14] focuses on the role of data analytics and visualization in IoT-based wastewater treatment monitoring. The research highlights the use of analytical dashboards that offer meaningful insights into wastewater quality, assisting in the selection of suitable treatment interventions. The study also explores the application of machine learning techniques, such as Random Forest and LightGBM, to enhance the prediction capabilities of IoT systems. This approach improves the accuracy of water quality assessments and reduces the need for frequent manual data collection, making the wastewater treatment process more efficient.

Literature [12, 15] emphasize the integration of IoT enables real-time data collection, analysis, and communication, facilitating decentralized monitoring and decision-making support. Such systems can improve the efficiency of wastewater treatment facilities, ensure compliance with regulatory standards, and enhance overall water quality. Additionally, IoT-based monitoring solutions can be applied to urban water systems, agriculture, and drinking water quality assessment.

2.3 Turbidity and Sludge Concentration Correlation

As industries focus on enhancing environmental management, understanding the link between turbidity and sludge concentration is crucial. Turbidity, a measure of how clear water is, is influenced by suspended particles like sediments and organic matter. Recognising this connection helps predict sediment movement and manage water quality. This section reviews studies that explore the relationship between turbidity and sludge concentration in different environments and the methods used to monitor this correlation.

Literature [16] examined the relationship between turbidity and suspended particulate matter (SPM) in the Southern Baltic. They found a moderate linear relationship, with a correlation coefficient (R²) of 0.69, especially noticeable in spring as shown in Figure 2.1, suggesting that seasonal changes play a role in sediment dynamics. These findings highlight the importance of considering seasonal variations when studying turbidity and suspended particles in water.

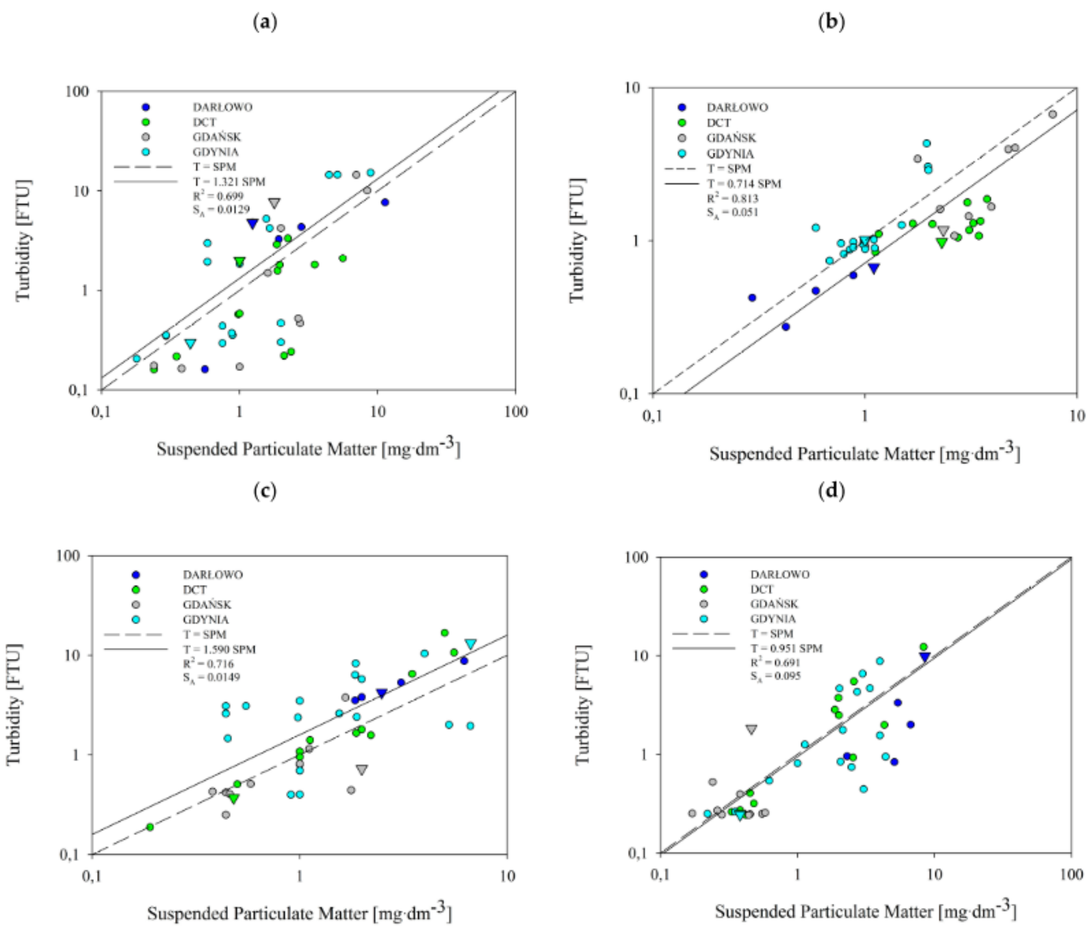


Figure 2.1 Correlations between in situ turbidity and the concentration of SPM in the near-surface layer for the spring (a), summer (b), autumn (c), and winter (d) [16].

Literature [17] studied the relationship between suspended sediment concentration (SSC) and turbidity in the Baisha River, China. The study showed a strong positive correlation, with the regression slopes varying depending on rainfall conditions. The correlation was strongest during the dry season and under non-erosive rainfall. This suggests that environmental factors like rainfall intensity can significantly influence the relationship between turbidity and sludge concentration.

Literature [18] introduced underwater image analysis to predict SSC based on turbidity. The method proved highly accurate, with an adjusted R^2 of over 0.91, making it a promising alternative to traditional turbidity measurements. While the research uses a different technique, it emphasizes the potential of advanced monitoring methods, such as turbidity sensors, for continuous monitoring of suspended sediment concentrations in environments like anaerobic digesters.

Literature [19] explored coal mine water and found a strong correlation between turbidity and suspended solids. They applied different formulas based on concentration gradients, achieving errors within 4-5%. This study emphasizes the importance of precise measurements and tailored methods, which can be adapted for monitoring systems with varying particulate concentrations, such as Palm Oil Mill Effluent (POME).

The correlation between turbidity and sludge concentration is generally positive but can vary due to factors like seasonal changes and rainfall intensity. These studies show the importance of localized research for accurate water quality assessment.

2.4 Stirring in Particle Distribution

Stirring and homogenization play a vital role in sludge treatment and management. These processes ensure the particles are evenly distributed, which improves the efficiency of treatment and disposal methods. The reviewed studies highlight different mechanisms and outcomes related to these processes.

Literature [20] discusses the impact of stirring on sludge particle size and structure. Stirring influences the morphology of sludge particles, causing a more homogeneous distribution. According to [20], this process can lead to flocs becoming looser and smaller in diameter, while still allowing for efficient settling. This is particularly important in processes such as anaerobic digestion, where the structure of the flocs affects microbial activity and overall digestion efficiency. Maintaining a uniform particle distribution is crucial for obtaining reliable turbidity readings in wastewater treatment processes, such as Palm Oil Mill Effluent (POME) treatment, as inconsistencies could result in inaccurate data.

Literature [21] examines how different stirring strategies influence sludge granulation and mixing. Research by [21] indicates that axial continuous agitation produces more uniform granules compared to magnetic stirring, which leads to flocculent sludge. The type of stirring used can have a significant effect on the physical characteristics of sludge particles, which is critical for enhancing machine learning models used to predict sludge concentration. In the context of wastewater treatment, stirring in a continuous stirred tank reactor (CSTR) enhances the mixing efficiency, which is crucial for the uniform distribution of sludge particles [22].

2.5 Machine Learning in Wastewater Monitoring

As machine learning (ML) continues to gain traction in wastewater monitoring, it proves to be a valuable tool for improving accuracy, efficiency, and real-time data analysis. By integrating ML with technologies such as IoT and sensors, wastewater treatment facilities are better able to predict and manage water quality, leading to enhanced environmental protection and resource management. Below, we explore several machine learning models and their applications in wastewater monitoring.

Literature [23] presents various machine learning models that have been used to predict water quality parameters, such as the Multilayer Perceptron (MLP), Generalized Feedforward Regression (GFFR), and Radial Basis Function (RBF) models. The GFFR model demonstrated superior performance in predicting conductivity in the second modelling scenario, achieving a high correlation accuracy of $R = 0.893$ and lower prediction deviations with $NRMSE = 0.091$ and $NMAE = 0.071$. Recent studies have also explored ML approaches for predicting sludge production and characteristics in wastewater treatment plants. Ensemble methods like

XGBoost and Random Forest have demonstrated superior performance in modeling the complex, nonlinear relationships in sludge production [24, 25]. These models outperform traditional empirical estimations and accurately predict sludge yield using operational data and water quality parameters [26].

Literature [27] delves into the use of Neural Networks, specifically Bilateral Neural Networks (BNN), which have proven to be highly accurate in achieving a Root Mean Square Error (RMSE) of 0.0008 and a Mean Absolute Error (MAE) of 0.0003 in predicting resistivity, an important water quality indicator. These models are also known for their efficiency in computation. Additionally, ensemble methods like XGBoost and Random Forest have been successful in predicting nutrient removal and effluent quality, achieving notable precision and accuracy [28].

Literature [29] discusses the integration of IoT and machine learning, which allows for continuous, real-time monitoring of wastewater quality. This combination of sensors measuring parameters like temperature, turbidity, and pH enables data to be transmitted to cloud-based platforms for analysis. Such systems can trigger immediate alerts if contamination occurs, enhancing the responsiveness of wastewater management.

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Introduction

This chapter outlines the methodology adopted in Final Year Project 2 (FYP2) to enhance the sludge monitoring system introduced in FYP1. It describes the procedures for sample preparation, sensor data collection, data visualization, machine learning model training, and the deployment of the best model into a monitoring interface. These procedures are directly aligned with the project objectives, particularly in studying the correlation between fermentation parameters which is turbidity and methane levels, and sludge concentration.

3.2 System Overview

The overall process flow of the system is illustrated in Figure 3.1. The monitoring system aims to estimate sludge concentration in raw Palm Oil Mill Effluent (POME) under simulated anaerobic digester conditions. The system begins with preparing diluted raw POME samples of varying sludge concentrations. These samples are stirred to evenly distribute suspended particles, and measurements are taken using turbidity and methane sensors to study the correlation with sludge concentrations.

The sensor readings are sent via a TCP/IP connection from an Arduino Uno R4 Wi-Fi (client) to an ESP32 (server). The ESP32 transmits the data to a MATLAB App Designer GUI for data collection, visualization, monitoring, and sludge concentration prediction using a trained regression model.

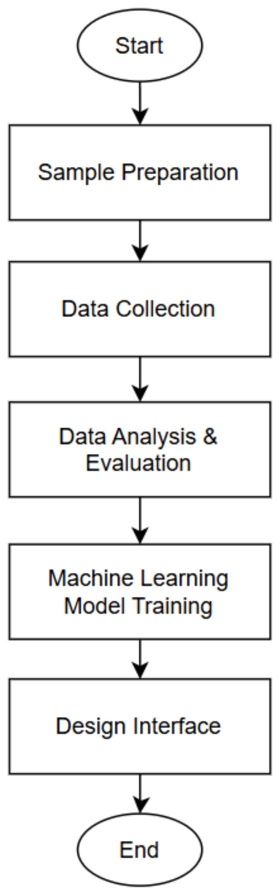


Figure 3.1 Flowchart of the overall system.

3.3 System Architecture

The system architecture consists of three core components: sensor hardware, embedded microcontrollers, and software for data visualization and prediction. Sensors gather data and send it to the Arduino Uno R4 Wi-Fi, which forwards the data to an ESP32 acting as a TCP server. This data is then received by the GUI and can be stored in a CSV file. In MATLAB, the collected data is processed using the Data Cleaner to remove outliers, and a regression model is developed using the Regression Learner. The trained model can be integrated into the GUI to visualize and predict sludge concentration for new readings. Figure 3.2 presents the complete system architecture.:

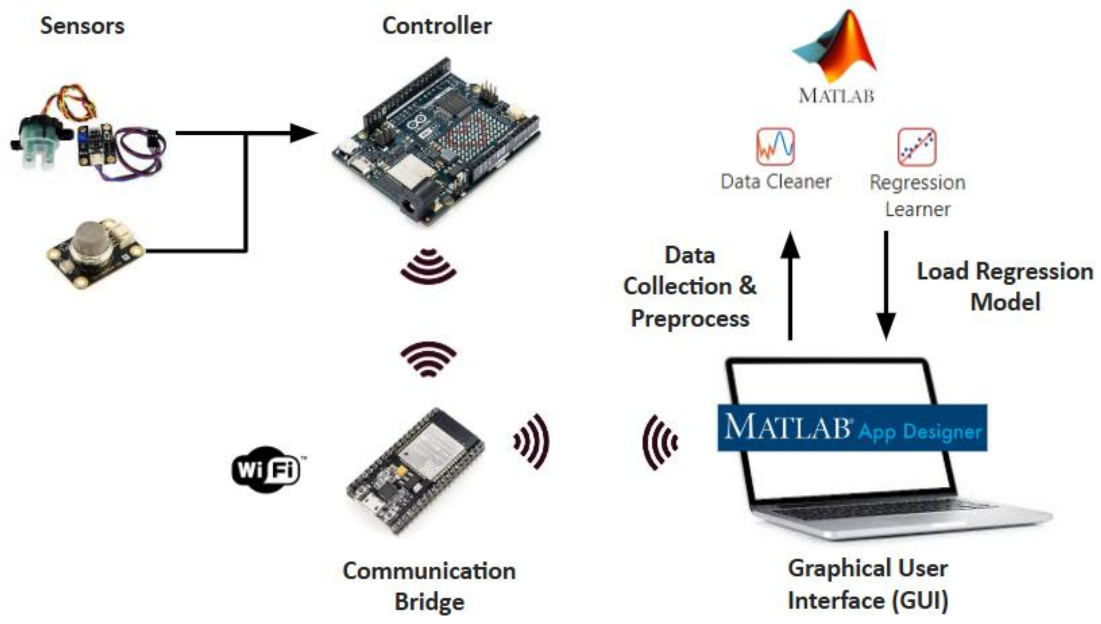


Figure 3.2 System architecture of the monitoring system.

3.4 Sample Preparation

Raw POME with 100% sludge concentration was used as the base material. Samples were diluted using distilled water to produce different sludge concentrations. Each sample maintained a constant volume of 1000 ml, calculated using the dilution formula as shown in Equation 3.1:

$$M_1V_1 = M_2V_2 \tag{3.1}$$

$M_1 = Initial\ concentration$
$V_1 = Volume\ of\ raw\ POME$
$M_2 = Desired\ concentration$
$V_2 = Final\ volume\ (fixed\ at\ 1000ml)$

For example, to prepare a 20% sludge concentration sample from 80% sludge concentration sample:

$$80\% \times V_1 = 20\% \times 1000ml \rightarrow V_1 = 250ml$$

Thus, 250 ml from 80% sample was mixed with 750 ml of distilled water to produce a 20% sample.

Samples were prepared at concentrations of 80%, 60%, 50%, 40%, 30%, 20%, 10% and 0%. The 70% sample was intentionally omitted due to the negligible difference observed between 60% and 80% samples during initial testing. Additionally, each percentage of sludge concentrations only capable of one sample as there was shortage of raw POME supplies in the lab.

3.5 Experimental Setup

Each prepared sample was poured into a 1000 ml beaker and stirred using a magnetic stirrer set to 130 RPM for 5 minutes to ensure uniform distribution of suspended particles. To simulate the dark environment of an anaerobic digester and prevent ambient light interference with turbidity readings, the beaker was enclosed in a black box. A custom-designed sensor casing, functioning as a lid, was securely placed over the beaker to hold the turbidity and methane sensors in a fixed position. This setup ensured consistent sensor placement and minimized any vibration or movement during data collection. Figure 3.3 illustrates the complete experimental setup, while Figure 3.4 shows the designed casing used to mount the sensors.

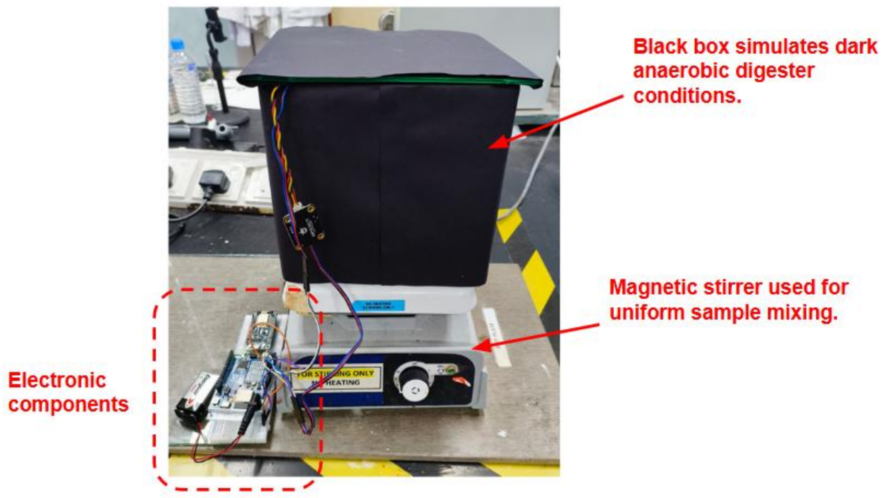

Black box simulates dark anaerobic digester conditions.
Magnetic stirrer used for uniform sample mixing.
Electronic components

Figure 3.3 Experimental setup of the system.

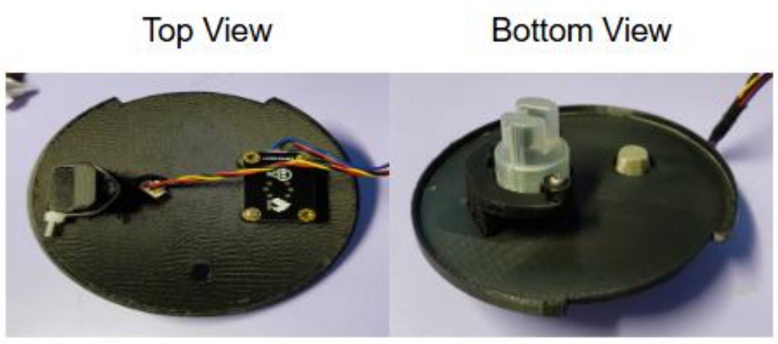


Figure 3.4 Designed casing for the sensors.

3.6 Electronic Circuit Development

This section outlines the development of the electronic circuitry used to acquire sensor data and enable communication between the hardware and software components of the system. The circuit integrates analog sensors with microcontrollers to collect and transmit data efficiently. The design prioritizes accuracy, stability, and compatibility with MATLAB for real-time monitoring and prediction.

3.6.1 Controller

The primary controller used in this system is the Arduino Uno R4 Wi-Fi, as shown in Figure 3.5. This board was selected due to its upgraded 14-bit analog-to-digital converter (ADC), which offers significantly higher resolution compared to the standard 10-bit ADC on traditional Arduino Uno boards. This increased resolution allows for more precise readings from analog sensors, such as turbidity and methane gas sensors.

In addition to its high ADC resolution, the Arduino Uno R4 Wi-Fi supports wireless communication. Within this system, it functions as the data acquisition unit, collecting sensor data and transmitting it to the ESP32 via TCP/IP communication. The board's ability to read analog voltages in the 0–5 V range, combined with built-in Wi-Fi support, makes it well-suited for this application by reducing hardware complexity.

Features of the Arduino Uno R4 Wi-Fi:

(a) High ADC Resolution: 14-bit ADC for enhanced analog signal accuracy.

(b) Wi-Fi Capability: Enables wireless data transmission via TCP/IP.

(c) Arduino Ecosystem Compatibility: Seamless integration with existing libraries and tools.

(d) Flexible Power Supply: Operates on 5 V logic and supports USB-C or external 7–12 V input.

(e) Programmability: Can be programmed via Arduino IDE or PlatformIO.

Figure 3.5 Arduino Uno R4 Wi-Fi Microcontroller.

Table 3.1 Specifications of Arduino Uno R4 Wi-Fi

Specification	Description
Model Name	Arduino Uno R4 Wi-Fi
ADC Resolution	14-bit (0 to 16383)
Input Voltage	7V to 12V (external), 5V via USB
Operating Voltage	5V logic
Connectivity	Wi-Fi (via ESP32-S3 chip)
USB Interface	USB-C

3.6.2 Communication Bridge

The ESP32 microcontroller module, as shown in Figure 3.6, serves as the communication bridge between the Arduino and the MATLAB interface. It operates as a TCP server, receiving data packets from the Arduino Uno R4 Wi-Fi, which functions as the TCP client. Once received, the ESP32 transmits the data to the MATLAB App Designer GUI via a socket connection.

This setup allows monitoring, live graphing, alert generation, and data export features, all accessible through the MATLAB-based user interface. The ESP32 was selected for its built-in Wi-Fi capabilities, dual-core processor, and low power consumption, making it a reliable and efficient intermediary for wireless data transmission.

Figure 3.6 ESP32

3.6.3 Sensor Modules

This system utilizes two key analog sensors: a turbidity sensor for detecting suspended solids in the fluid, and a methane gas sensor for monitoring gas levels produced by the sludge. Both sensors are mounted securely on a custom-designed lid to ensure stable positioning over the beaker during data collection and to minimize external disturbances.

3.6.3.1 Turbidity Sensor (DFRobot SEN0189)

The turbidity sensor used in this project is the DFRobot SEN0189, as shown in Figure 3.7. It operates at 5V and provides an analog voltage output that corresponds to the turbidity level of the fluid, measured in Nephelometric Turbidity Units (NTU). The sensor supports a measurement range of 0 to 3000 NTU, making it suitable for detecting varying levels of suspended particles in raw Palm Oil Mill Effluent (POME).

To establish a reference point for clean water, the sensor was calibrated using distilled water, representing 0 NTU. The voltage output from the sensor is then converted to NTU using an empirical equation derived from the sensor's datasheet, as illustrated in Figure 3.8.

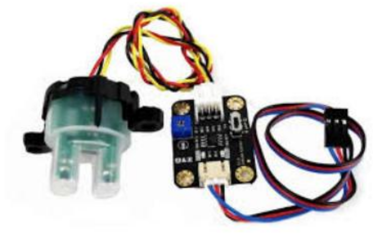


Figure 3.7 DFRobot SEN0189 Turbidity Sensor.

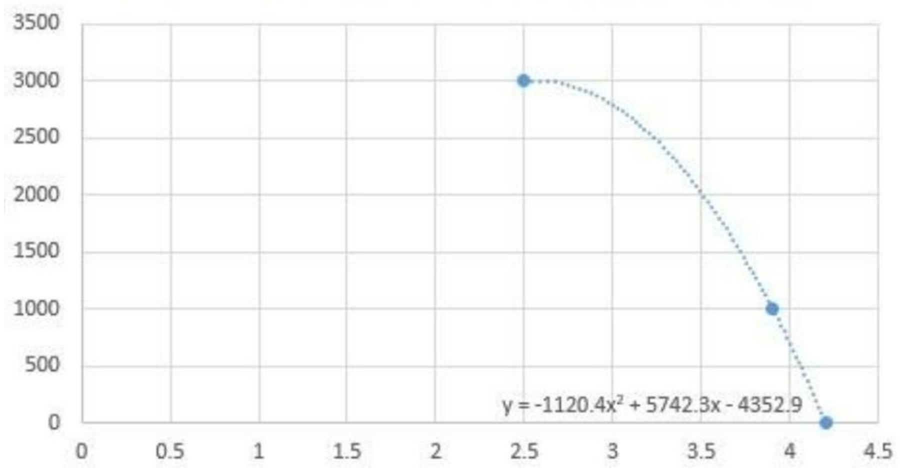


Figure 3.8 Relationship between sensor output voltage and turbidity (NTU).

3.6.3.2 Methane Gas Sensor (MQ-4)

The methane gas sensor used is the MQ-4, as shown in Figure 3.9. It detects methane concentrations in the surrounding air and outputs a corresponding analog voltage signal. The sensor typically operates in the range of 200 to 10,000 ppm of methane gas. In this project, the MQ-4 was not calibrated using known reference gas concentrations due to laboratory limitations. However, its raw analog values (ADC readings) were still utilized in the regression model to identify relative gas trends.

Despite the lack of absolute calibration, the sensor effectively contributed to capturing fluctuations in gas emission levels, which were relevant for pattern analysis and sludge concentration prediction.

Figure 3.9 MQ-4 Methane Gas Sensor.

3.7 Sensor Calibration

To ensure accurate and consistent readings from the sensors, calibration procedures were considered as part of the system development. Sensor calibration is essential for translating raw voltage or ADC values into meaningful physical quantities, especially when working with analog sensors. This section explains the calibration approach taken for both the turbidity and methane sensors used in the project.

3.7.1 Turbidity Sensor Calibration

To ensure reliable sensor output, the turbidity sensor was calibrated prior to use. The DFRobot SEN0189 sensor provides analog voltage readings, which are converted to NTU (Nephelometric Turbidity Units) using the calibration equation, as shown in Equation 3.2 from its datasheet.

$$\text{Turbidiy (NTU)} = -1120.4V^2 + 5742.3V - 4352.9 \qquad (3.2)$$

Where V is the sensor output voltage in volts.

For this project, 0 NTU was used as the reference point, using distilled water as the calibration sample. The sensor was immersed in distilled water to determine the baseline voltage representing clear water. Due to project constraints, multi-point calibration typically using Formazin standards was not performed. However, the 0 NTU baseline improves accuracy across diluted raw POME samples. Calibration yield Equation 3.3 which was used to convert the output voltage readings to NTU.

$$\text{Turbidiy (NTU)} = -1120.4V^2 + 5133.73V - 2839.35 \tag{3.3}$$

3.7.2 Methane Sensor Calibration

The MQ-4 methane sensor was not calibrated against known methane concentrations due to the lack of gas calibration kits. Instead, raw ADC readings were used to identify relative trends across samples. This method is acceptable for data-driven machine learning models, but further calibration is recommended in future work.

3.8 Data Collection and Preprocessing

After each sample was prepared, it was stirred for 5 minutes at 130 RPM to ensure uniform suspension of particles. The sensor readings which are turbidity in NTU, analog voltage from the turbidity sensor, and methane gas sensor ADC values were continuously streamed from the Arduino Uno R4 Wi-Fi to the MATLAB App Designer interface via a TCP connection. These live readings were displayed in real time within the GUI and could be optionally saved as a CSV file for further analysis.

If the user initiated the data collection process in the GUI, the application would buffer three consecutive sensor readings and automatically compute their average. This averaged data, together with the user-input sludge concentration, was stored as a single batch data point. Once the number of desired samples was reached, the application prompted the user to save the batch data in CSV format. This batch dataset was later cleaned using MATLAB’s Data Cleaner tool to remove outliers and prepare it for regression model training.

3.9 Machine Learning Implementation

The cleaned dataset obtained from the experimental results was used to train regression models in MATLAB’s Regression Learner App. Three input features were selected as predictors which are turbidity in NTU, the ADC value from the methane gas sensor, and the output voltage from the turbidity sensor. The target output was the sludge concentration, expressed in percentage.

To improve the reliability of the model evaluation, a 5-fold cross-validation strategy was applied to improve generalization and minimize overfitting. Additionally, 10% of the dataset was set aside as test data to evaluate how well the model would perform on unseen data Various regression algorithms provided in the Regression Learner App were tested, which are Linear Regression, Support Vector Machines (SVM), Regression Trees, Gaussian Process Regression, Efficiently Trained Linear, Neural Networks, Kernel Approximation Regression, and Ensemble of Trees.

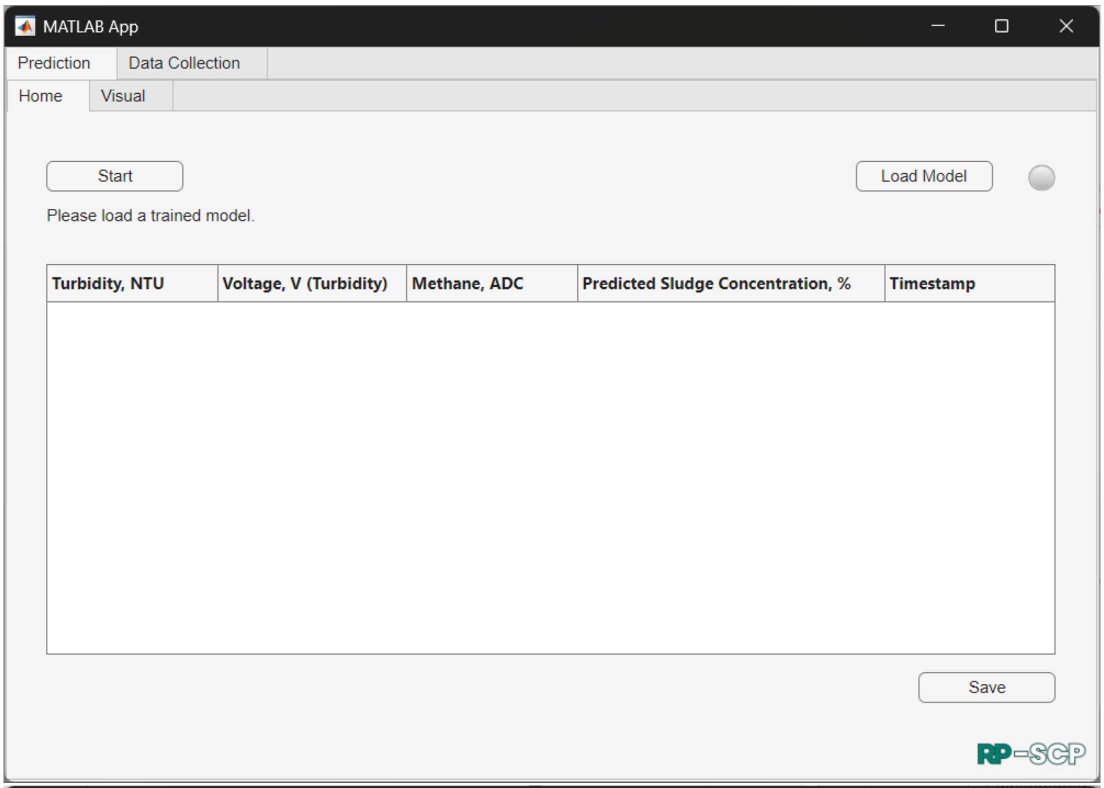
Model performance was evaluated based on Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and the coefficient of determination (R^2). The model with the highest accuracy and lowest RMSE was selected for deployment. This optimal model was then exported and loaded into the MATLAB App Designer interface for sludge concentration prediction.

3.10 GUI Deployment

The best-performing regression model was exported from MATLAB’s Regression Learner App and loaded into GUI developed using MATLAB App Designer. This graphical user interface (GUI) serves as a monitoring platform for predicting sludge concentration based on live sensor input.

The application features several key functionalities. First, it continuously receives incoming data from the turbidity and methane sensors through a communication bridge established between the Arduino Uno R4 Wi-Fi and ESP32. As new sensor values are received, the GUI uses the loaded trained model to generate sludge concentration predictions. The interface also provides live graphical visualization of both sensor readings and predicted concentration trends.

For data logging purposes, users can export the collected data into CSV format for further analysis and model training. Additionally, a built-in safety alert system triggers a warning notification if the predicted sludge concentration exceeds 60%, which is considered the critical threshold for potential system overload or inefficiency. Figure 3.10 illustrates the layout of the monitoring interface for prediction and Figure 3.11 illustrates the layout of the monitoring interface for data collection.



Turbidity, NTU	Voltage, V (Turbidity)	Methane, ADC	Predicted Sludge Concentration, %	Timestamp

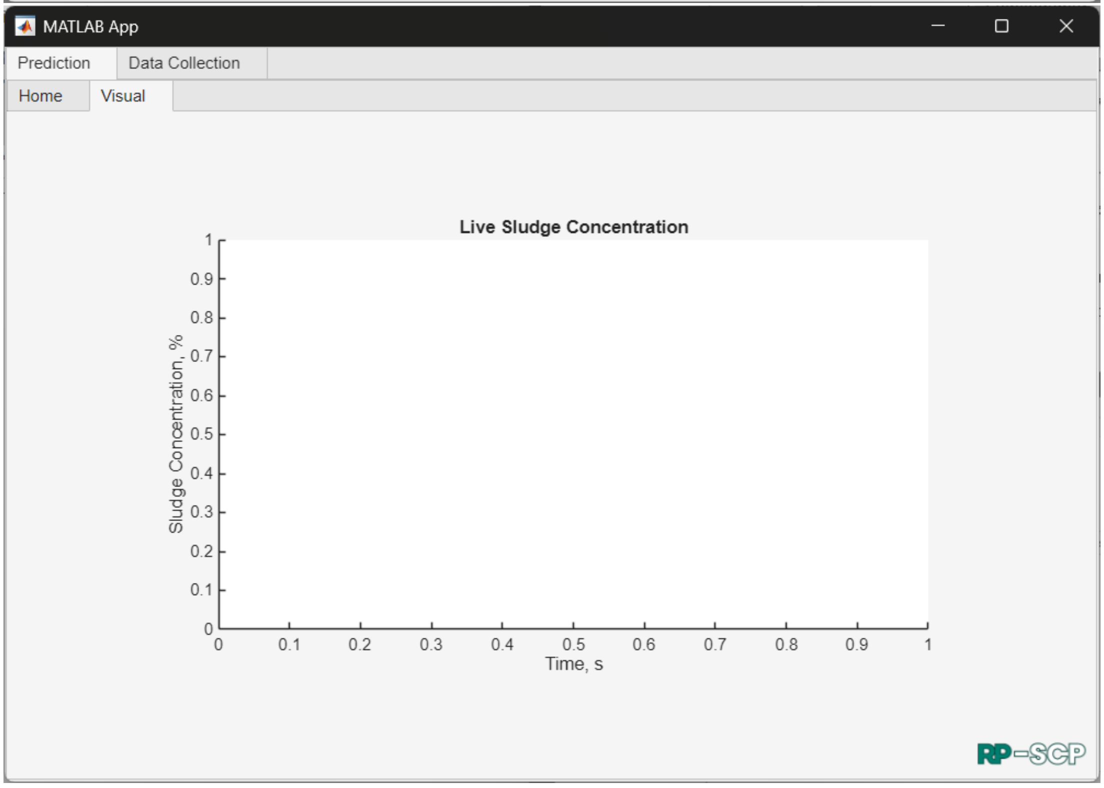


Figure 3.10 GUI Layout for Prediction.

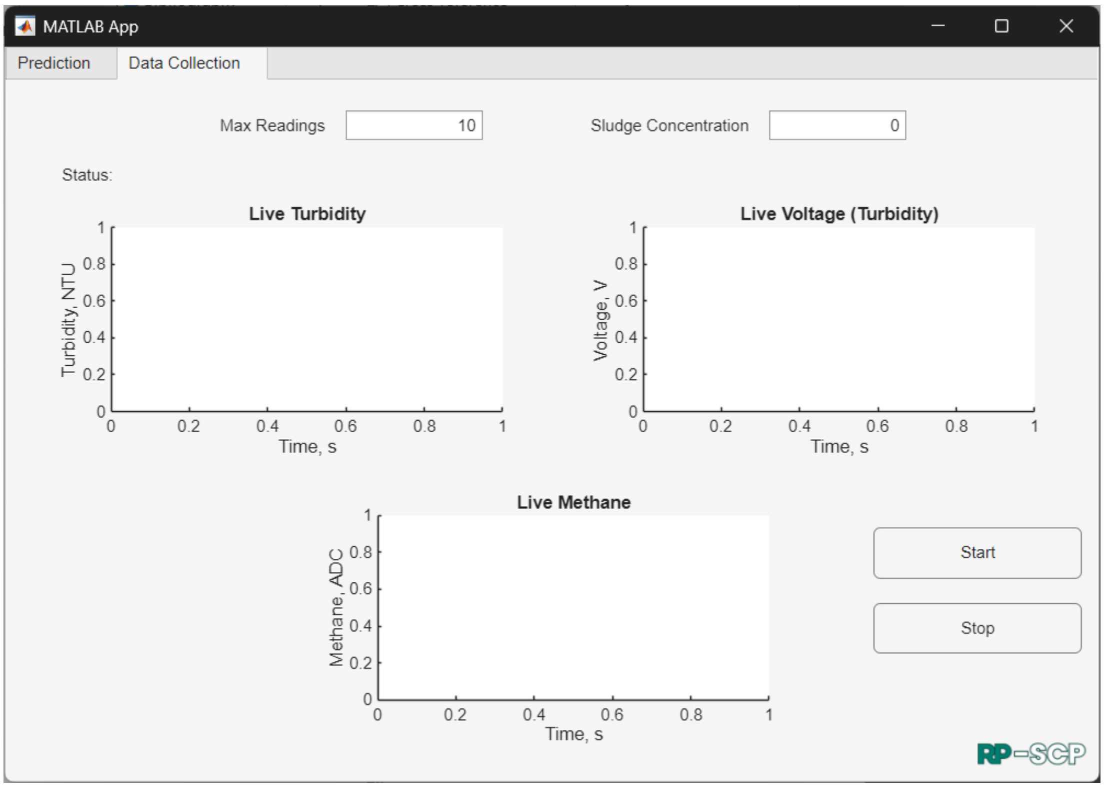


Figure 3.11 GUI Layout for Data Collection.

3.11 Costing

The total cost of developing the sludge monitoring system was approximately RM 326.15, with the biggest spent on the Arduino Uno R4 Wi-Fi, which handled both sensor data collection and wireless communication. The ESP32 module acted as a bridge to the MATLAB interface for data transfer. Sensors which are DFRobot SEN0189 turbidity sensor and MQ-4 methane sensor were essential in capturing the parameters needed for sludge concentration prediction. To keep the sensors stable, a 3D-printed casing was made for accurate and repeatable placement. Additional basic components such as jumper wires and a stripboard were used to complete the circuit at minimal cost. Overall, the components were selected carefully based on performance, reliability, and cost, making the system affordable yet effective for research and learning purposes. Table 3.2 shows the price of the components.

Table 3.2 Cost of project.

Tools/Software/Equipment	Price (RM)
Arduino UNO R4 WiFi	151.25
DFRobot MQ-4 Methane Gas Sensor	66.00
DFRobot SEN0189 Turbidity Sensor	35.00
NodeMCU ESP32	23.90
3D Printed Parts (PLA+)	50.00

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

This chapter presents the findings from the experiments outlined in previous chapter. The results focus on the relationship between turbidity, methane levels, and sludge concentration in raw Palm Oil Mill Effluent (POME) samples. Key outcomes include sensor response trends, model performance metrics, and the system's ability to predict sludge concentration. The discussion interprets these results with reference to the project's objectives and highlights the system's strengths and limitations.

4.2 Sensor Readings and Trends

This section highlights the trends observed from the turbidity and methane sensor readings across different sludge concentrations. Both sensors showed measurable and predictable responses that align with the expected behaviour of raw Palm Oil Mill Effluent (POME) under increasing solids and fermentation activity.

4.2.1 Turbidity Sensor Response

The DFRobot SEN0189 turbidity sensor produced voltage outputs that decreased as sludge concentration increased as shown in Figure 4.1. This behaviour aligns with its internal infrared light scattering mechanism, higher particle content results in more light scattering and less light reaching the photodiode, hence producing a lower voltage. After converting the voltage to NTU using the sensor's calibration equation, a strong positive correlation between NTU and sludge concentration was observed. As seen in Figure 4.2, the NTU values rose sharply from 0% to 20% sludge and dropped from 30% onwards, indicating the sensor was operating beyond its optimal range which is 3000 NTU. As a result, readings in this upper range may be less reliable and should be interpreted with caution. Because of this limitation, the raw voltage output from the turbidity sensor was included as an additional supporting parameter in the machine learning model. While the converted NTU values captured the trend well at lower concentrations, the direct voltage readings provided more continuous resolution across the full sludge concentration spectrum, including higher levels where NTU values were capped.

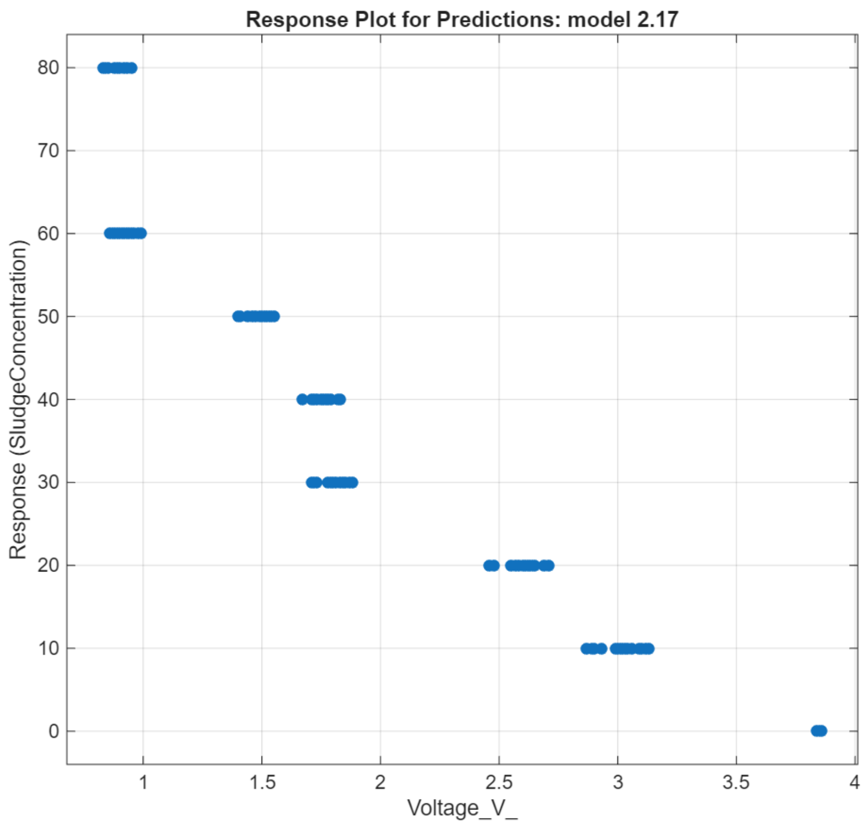


Figure 4.1 Relationship between turbidity sensor voltage reading and sludge concentration.

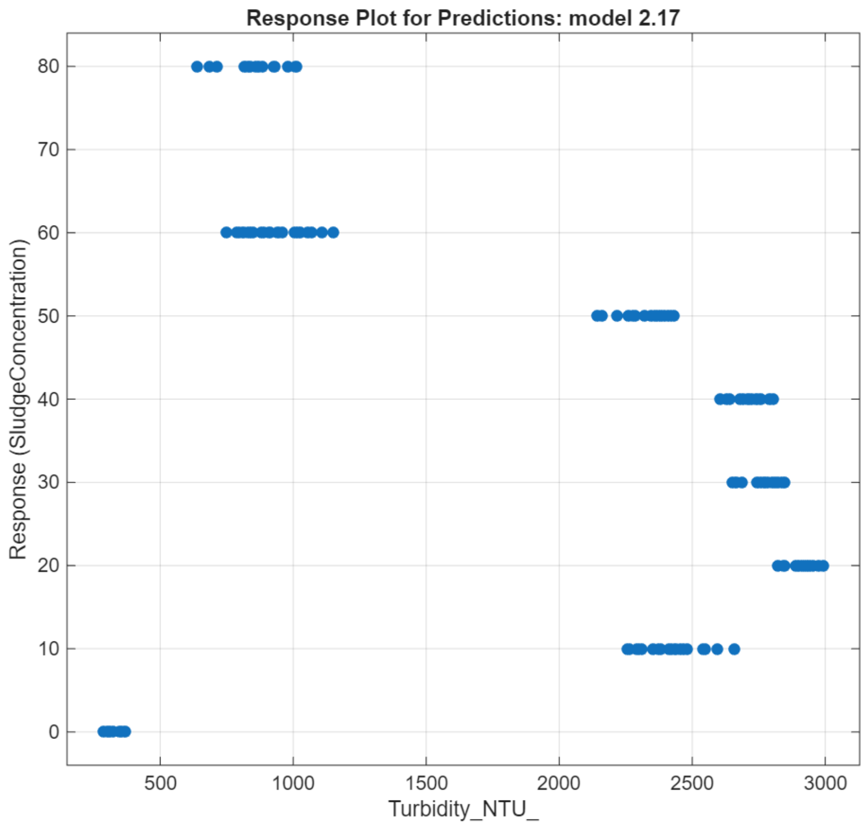
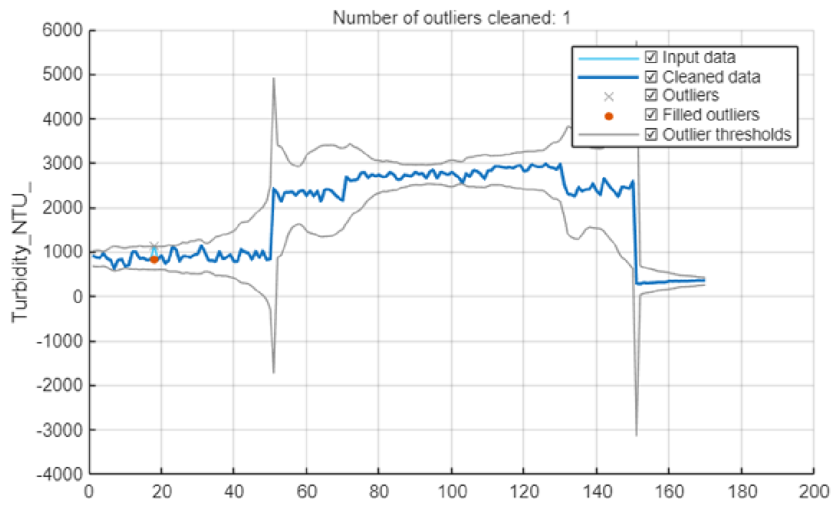


Figure 4.2 Relationship between turbidity (NTU) and sludge concentrations.

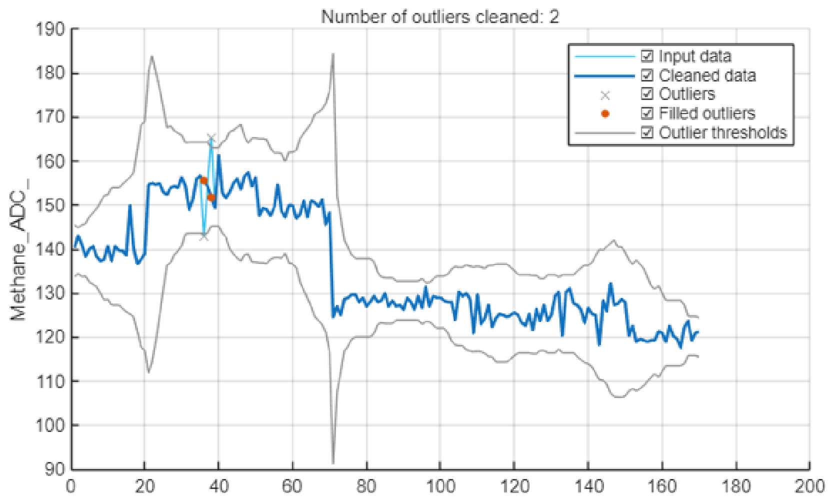
4.3 Data Cleaning

To ensure the quality of training data, MATLAB's Data Cleaner tool was used to fill outliers and inconsistent readings caused by sensor noise. The cleaning process involved visual inspection and automatic filtering using the app’s built-in algorithms. Figure 4.3 highlights how it helped refine the dataset before it was used for model training using moving median with a threshold of 3 to detect the outliers.

(a)



(b)



(c)

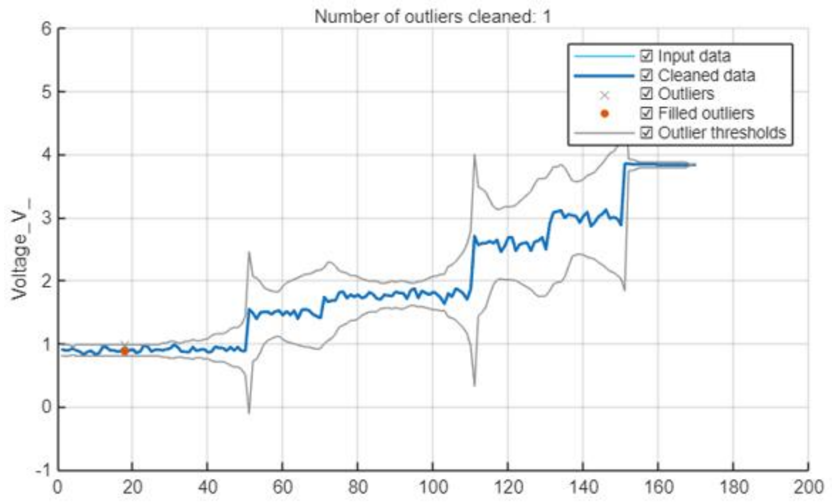


Figure 4.3 Graph of the parameters trend with filled in outliers for turbidity in NTU (a), methane ADC value (b), and turbidity sensor voltage reading (c).

4.4 Machine Learning Model Performance

This section outlines the evaluation of regression models trained to predict sludge concentration based on sensor inputs. The goal was to develop a reliable prediction model using features collected during the experimental phase.

4.4.1 Model Comparison

Several regression models were trained using MATLAB's Regression Learner App. The predictors used were turbidity (NTU), methane sensor ADC values, and turbidity sensor voltage, with sludge concentration as the target output. The training used 5-fold cross-validation and reserved 10% of the dataset for testing. Models used were Linear Regression, Support Vector Machines (SVM), Regression Trees, Gaussian Process Regression, Efficiently Trained Linear, Neural Networks, Kernel Approximation Regression, and Ensemble of Trees and evaluated based on Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and the coefficient of determination (R^2). Figure 4.4 visualize the performance of the trained models based on evaluation parameters that was stated.

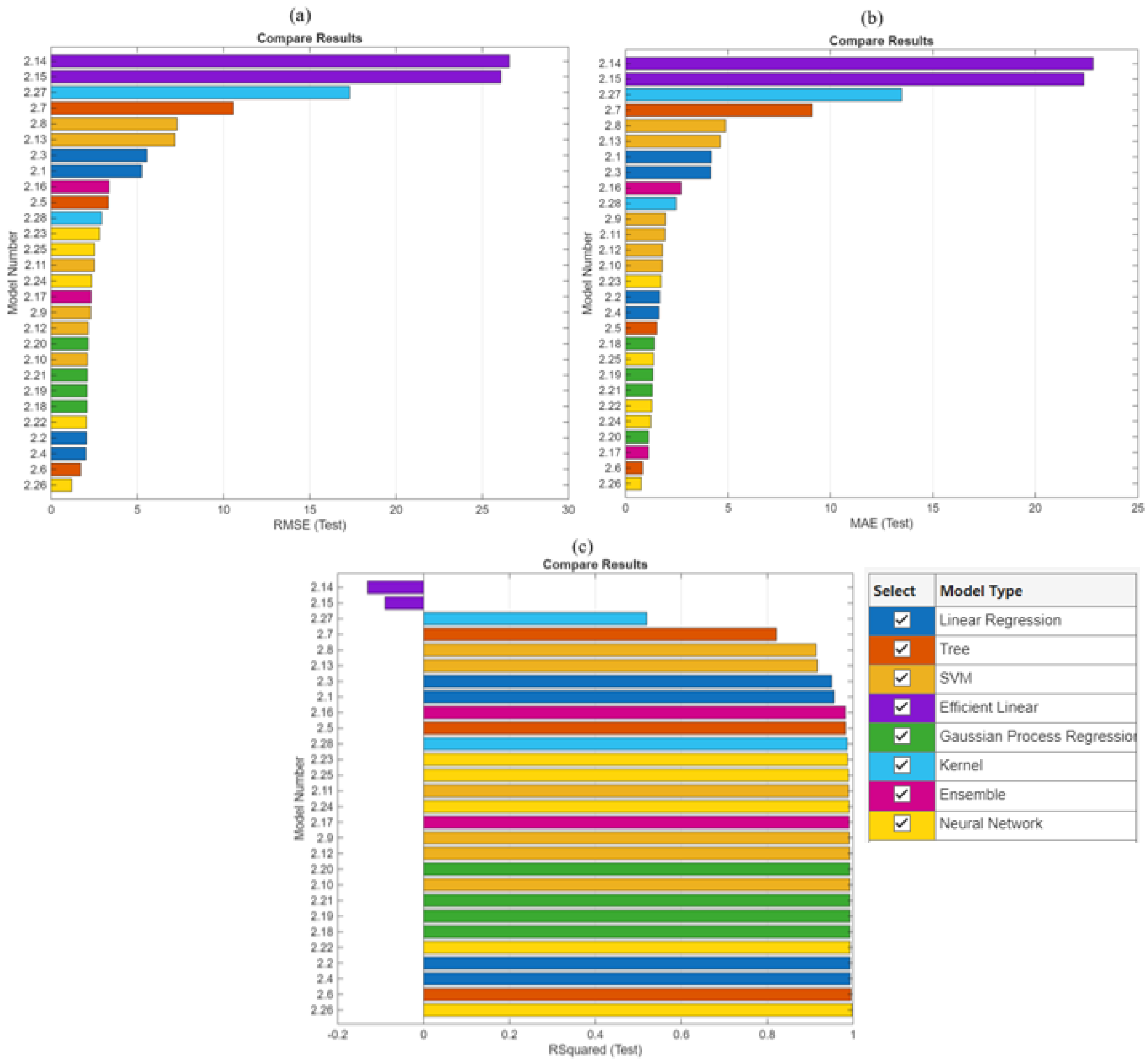


Figure 4.4 Comparison results of trained models for RMSE (Test) (a), MAE (TEST) (b), and R² (Test) (c).

From the results shown in the figure, models based on neural networks, some types of linear regression, SVM, and Gaussian process performed quite well. These models showed low RMSE and MAE values along with R² scores that were close to 1, indicating strong accuracy. On the other hand, models using kernel and efficient linear methods have poor performance, showing higher errors and weaker predictive power. To narrow down the most suitable model for deployment, the top five most accurate models were shortlisted and compared in detail. These included variations of neural networks, decision trees, and linear regressions. As shown in Table 4.1, the Tri-layered Neural Network achieved the best performance, with the lowest RMSE (1.1951), lowest MAE (0.7683), and a very high R² value (0.9977), making it the most reliable choice for sludge concentration prediction.

Table 4.1 Evaluation parameters value of the five most accurate models.

Model	RMSE (Test)	MAE (Test)	R^2 (Tesr)
Neural Network (Trilayered)	**1.1951**	**0.7683**	**0.9977**
Tree (Medium)	1.7261	0.8291	0.9952
Stepwise Linear Regression	2.0168	1.6380	0.9935
Linear Regression (Interactions Linear)	2.0512	1.6746	0.9932
Neural Network (Narrow)	2.0531	1.2882	0.9932

To further assess the performance of the tri-layered neural network, its test residual plot as shown in Figure 4.5 was analyzed. In this plot, positive residuals indicate that the model under-predicted the sludge concentration while negative residuals indicate over-prediction. The average residuals were found to be below 1, suggesting that for a true sludge concentration of 15%, the model typically predicted values around 16% or 17%, demonstrating relatively low prediction error.

Additionally, the predicted vs. actual plot in Figure 4.6 shows that the model's predictions align closely with the observed test data. The data points lie very near the ideal diagonal line, indicating high prediction accuracy and minimal deviation across the tested concentration range.

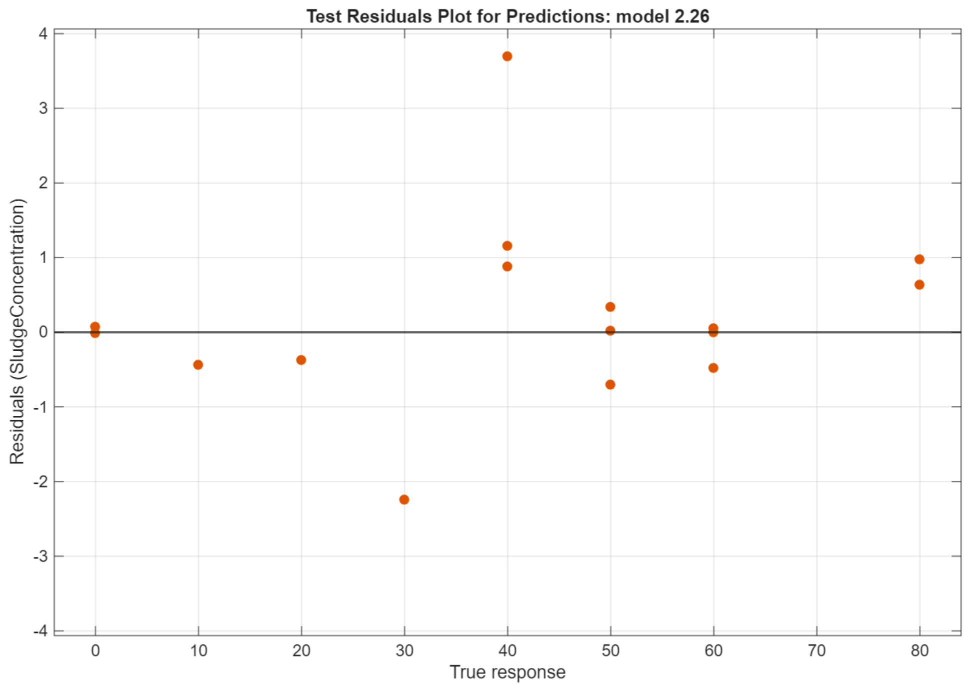


Figure 4.5 Test Residual Plot for Neural Network (Trilayered).

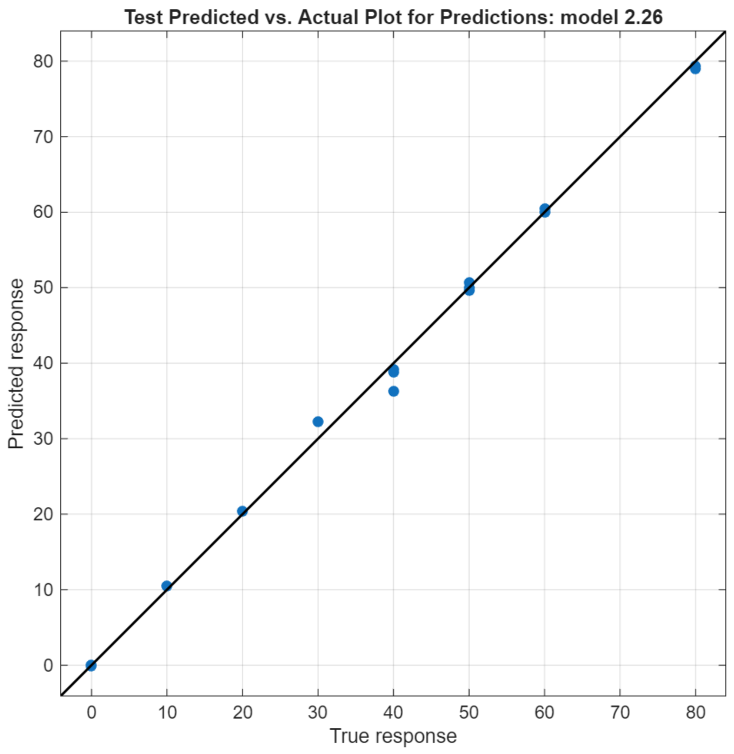
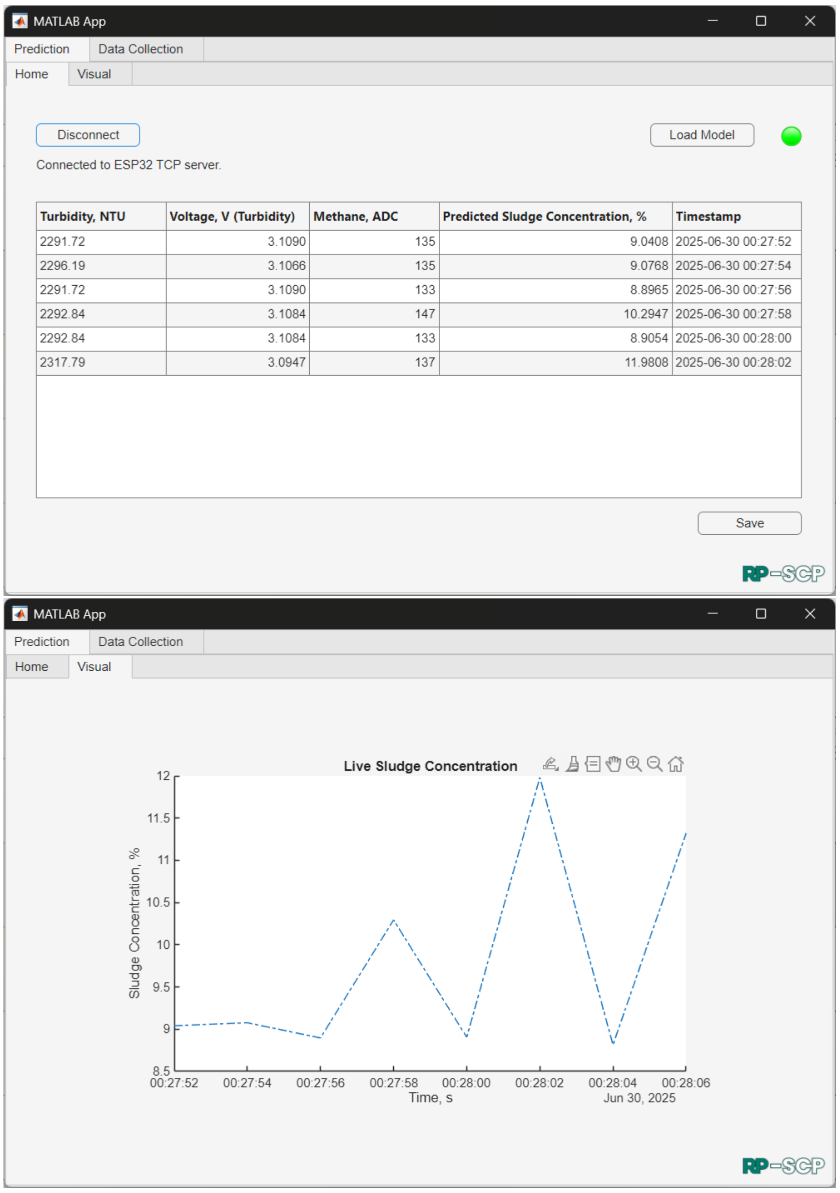


Figure 4.6 Test Predicted vs. Actual Plot for Neural Network (Trilayered).

4.5 GUI Deployment

The selected model was loaded into the MATLAB App Designer GUI. The interface was designed to receive sensor data via TCP/IP communication from the Arduino and ESP32. Predictions and visualization were successfully generated based on incoming data, offering users immediate insight into sludge concentration as shown in Figure 4.7.

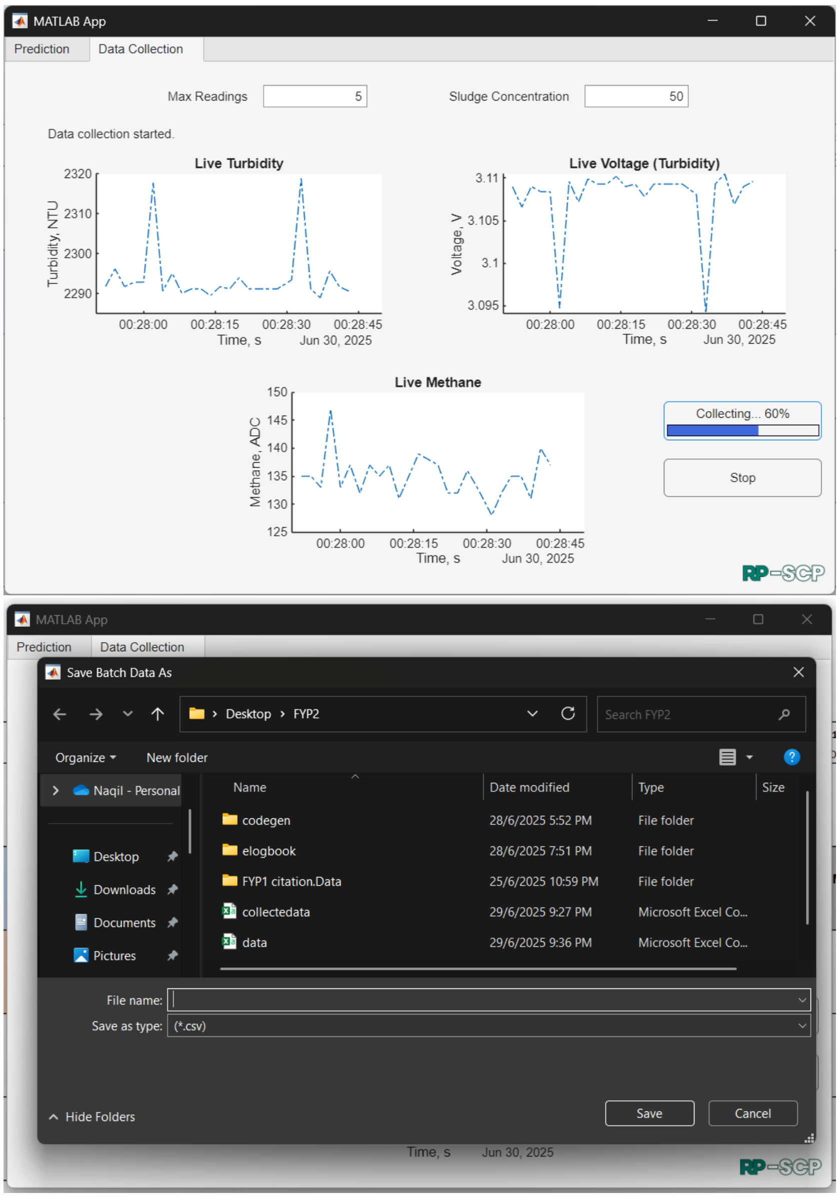


Connected to ESP32 TCP server.

Turbidity, NTU	Voltage, V (Turbidity)	Methane, ADC	Predicted Sludge Concentration, %	Timestamp
2291.72	3.1090	135	9.0408	2025-06-30 00:27:52
2296.19	3.1066	135	9.0768	2025-06-30 00:27:54
2291.72	3.1090	133	8.8965	2025-06-30 00:27:56
2292.84	3.1084	147	10.2947	2025-06-30 00:27:58
2292.84	3.1084	133	8.9054	2025-06-30 00:28:00
2317.79	3.0947	137	11.9808	2025-06-30 00:28:02

Figure 4.7 Prediction and Visualization of Sludge Concentration in the GUI.

In addition to monitoring, the GUI also supports batch data collection for future model training. Users can specify the number of readings and the known sludge concentration percentage, then begin data collection using the Start button as shown in Figure 4.8. The system averages three readings per sample to ensure data consistency and saves the results in CSV file.



Max Readings 5

Sludge Concentration 50

Data collection started.

Name	Date modified	Type	Size
codegen	28/6/2025 5:52 PM	File folder	
elogbook	28/6/2025 7:51 PM	File folder	
FYP1 citation.Data	25/6/2025 10:59 PM	File folder	
collectedata	29/6/2025 9:27 PM	Microsoft Excel Co...	
data	29/6/2025 9:36 PM	Microsoft Excel Co...	

File name:

Save as type: (*.csv)

Figure 4.8 Data Collection GUI.

4.6 Discussion

The results from this project showed that it is possible to develop a functional sludge monitoring system using simple and affordable components such as the Arduino Uno R4 Wi-Fi, ESP32, and analog sensors. Even though the system did not rely on expensive or advanced equipment, it was still able to produce useful insights regarding sludge concentration in raw Palm Oil Mill Effluent (POME). The integration of machine learning helped improve prediction accuracy by recognizing patterns in the sensor data.

One of the key findings was the significance of the turbidity sensor's raw voltage readings, especially at higher sludge concentrations. The sensor could only measure up to 3000 NTU, and this limit was reached around the 20 percent sludge concentration mark. Beyond this point, the NTU readings were no longer reliable. To overcome this issue, the raw voltage output was included as an extra input for the machine learning model. This allowed the model to continue identifying trends where NTU data had already saturated. In the case of the methane sensor, calibration was not performed due to the lack of a gas calibration kit. However, the raw ADC readings still followed consistent patterns that aligned with microbial activity in the POME samples, making it a helpful secondary indicator.

Among all the trained models, the tri-layered neural network showed the best overall performance. It produced the lowest RMSE and MAE values and achieved an R^2 value that was very close to one. The residual plots and predicted versus actual plots confirmed that the model could estimate sludge concentration accurately, particularly at medium and high concentration levels.

The MATLAB App Designer interface played an important role in making the system user-friendly and complete. It was able to display live predictions, graph sensor readings in real time, and collect new data that could be saved for future training. Features such as alert notifications and CSV export made it convenient for users, especially for those without a strong technical background.

Despite the successful results, this project did have some limitations. The number of POME samples was limited, mainly due to supply constraints in the laboratory. Only one sample was prepared for each sludge concentration level, which may have reduced the variability of the dataset. Moreover, raw POME is biologically active and contains bacteria and microbes that continue to ferment over time. This means that samples taken at different times may behave differently during testing, which could affect the accuracy of the readings. In addition, the uncalibrated methane sensor limited the precision of the gas measurements. Lastly, even though a black enclosure was used to control lighting, small disturbances such as bubbles, foam, and lighting inconsistencies may still have introduced some noise into the sensor data.

In summary, this project successfully demonstrated that a practical sludge monitoring system could be built using basic sensors and machine learning. While there are several areas that can be improved in the future, the system has laid a strong foundation for continued development and potential real-world application.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusions

5.2 Contributions to Knowledge

Describe the potential impact of this proposed work on societal and environmental contexts (university-industry-government-public-environment).

5.3 Future Works

REFERENCES

[1] R. A. Kristanti, T. Hadibarata, A. Yuniarto, and A. Muslim, "Palm Oil Industries in Malaysia and Possible Treatment Technologies for Palm Oil Mill Effluent: A Review," Environmental Research, Engineering and Management, 2021.

[2] M. S. Saad, M. D. H. Wirzal, and Z. A. Putra, "Review on current approach for treatment of palm oil mill effluent: Integrated system," Journal of environmental management, vol. 286, p. 112209, 2021.

[3] S. Pallikodathan, H. C. Man, T. Idaty, A. Sulaiman, G. Nagarajoo, and M. F. Shukery, "Minimizing CO<sub>2</sub> Emission of POME Treatment System Using MILP Model," in Preprints, ed: Preprints, 2024.

[4] G. T. Ding et al., "Phycoremediation of palm oil mill effluent (POME) and CO2 fixation by locally isolated microalgae: Chlorella sorokiniana UKM2, Coelastrella sp. UKM4 and Chlorella pyrenoidosa UKM7," Journal of water process engineering, vol. 35, p. 101202, 2020.

[5] Z. Jákói, C. Hodúr, and S. Beszédes, "Monitoring the Process of Anaerobic Digestion of Native and Microwave Pre-Treated Sludge by Dielectric and Rheological Measurements," Water, 2022-04-15 2022, doi: 10.3390/w14081294.

[6] H. Falk, P. Reichling, C. Andersen, and R. Benz, "Online monitoring of concentration and dynamics of volatile fatty acids in anaerobic digestion processes with mid-infrared spectroscopy," Bioprocess and biosystems engineering, vol. 38, 08/21 2014, doi: 10.1007/s00449-014-1263-9.

[7] L. Y. K. Wang, J. W. Bong, Y. X. Eng, and W. S. Wong, "IoT Based Wastewater Dissolved Oxygen and Total Dissolved Solids Monitoring with Data Analytics," presented at the 2024 11th International Conference on Future Internet of Things and Cloud (FiCloud), 2024. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/FiCloud62933.2024.00023.

[8] A. Soetedjo, M. R. Abdilah, F. Y. Limpraptono, and E. Hendriarianti, "Implementation of Wastewater Monitoring System using Low-Cost Water

Sensor and IoT-based SCADA," ALINIER: Journal of Artificial Intelligence & Applications, vol. 5, no. 2, pp. 165-172, 2024.

[9] R. K. Nishan, S. Akter, R. I. Sony, M. M. Hoque, M. J. Anee, and A. Hossain, "Development of an IoT-based multi-level system for real-time water quality monitoring in industrial wastewater," Discover Water, vol. 4, no. 1, p. 43, 2024/07/09 2024, doi: 10.1007/s43832-024-00092-y.

[10] R. Martínez Carreras, N. Vela, E. Murray, P. Roche, and J. M. Navarro, "On the Use of an IoT Integrated System for Water Quality Monitoring and Management in Wastewater Treatment Plants," Water, vol. 12, p. 1096, 04/12 2020, doi: 10.3390/w12041096.

[11] S. T, "A Smart Monitoring System for Textile Wastewater Treatment," INTERANTIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT, vol. 08, pp. 1-5, 04/03 2024, doi: 10.55041/IJSREM30032.

[12] S. Pasika and S. T. Gandla, "Smart water quality monitoring system with cost-effective using IoT," Heliyon, vol. 6, no. 7, p. e04096, 2020/07/01/ 2020, doi: https://doi.org/10.1016/j.heliyon.2020.e04096.

[13] I. A, T. I. S, N. D, K. V. R, and A. K. S, "Streamlining Industrial Wastewater Management Through Real Time IoT Cloud Monitoring," in 2023 2nd International Conference on Edge Computing and Applications (ICECAA), 19-21 July 2023 2023, pp. 23-27, doi: 10.1109/ICECAA58104.2023.10212374.

[14] C. C. Chang, C. H. Wei, M. T. Lin, and S. C. J. Hwang, "Machine Learning Approach to IoT- Based Water Quality Monitoring," in 2023 IEEE 5th Eurasia Conference on Biomedical Engineering, Healthcare and Sustainability (ECBIOS), 2-4 June 2023 2023, pp. 182-186, doi: 10.1109/ECBIOS57802.2023.10218420.

[15] P. M. Kumar and C. S. Hong, "Internet of things for secure surveillance for sewage wastewater treatment systems," Environmental Research, vol. 203, p. 111899, 2022/01/01/ 2022, doi: https://doi.org/10.1016/j.envres.2021.111899.

[16] B. Lednicka et al., "Water Turbidity and Suspended Particulate Matter Concentration at Dredged Material Dumping Sites in the Southern Baltic," Sensors, vol. 22, no. 20, p. 8049, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/20/8049.

[17] H. Li, J. Lv, X. He, Y. Bao, and G. Nsabimana, "Precipitation-dependent

sensitivity of suspended sediment concentration to turbidity in a mountainous river in southwestern China," Ecological Indicators, vol. 159, p. 111644, 2024/02/01/ 2024, doi: https://doi.org/10.1016/j.ecolind.2024.111644.

[18] W. Kang, K. Lee, and J. Kim, "Prediction of Suspended Sediment Concentration Based on the Turbidity–Concentration Relationship Determined via Underwater Image Analysis," Applied Sciences, vol. 12, p. 6125, 06/16 2022, doi: 10.3390/app12126125.

[19] S. Du, Y. Shi, H. Chen, S. Song, Y. Fan, and J. Zhao, A prediction model for suspended solids based on turbidity in coal mine water (Second International Conference on Energy, Power, and Electrical Technology (ICEPET 2023)). SPIE, 2023.

[20] Z. Yatin, "Effect of stirring on structure of anaerobic flocs and space distribution of microbial activity," Environmental Pollution & Control, 2012.

[21] Y.-p. Hou, D.-c. Peng, B.-b. Wang, X.-y. Zhang, and X.-d. Xue, "Effects of stirring strategies on the sludge granulation in anaerobic CSTR reactor," Desalination and Water Treatment, vol. 52, no. 34, pp. 6348-6355, 2014/10/01/ 2014, doi: https://doi.org/10.1080/19443994.2013.841102.

[22] P. T R et al., "Impact of sludge density and viscosity on continuous stirred tank reactor performance in wastewater treatment by numerical modelling," Journal of the Taiwan Institute of Chemical Engineers, vol. 166, p. 105368, 2025/01/01/ 2025, doi: https://doi.org/10.1016/j.jtice.2024.105368.

[23] H. A. Afan et al., "Data-driven water quality prediction for wastewater treatment plants," Heliyon, vol. 10, no. 18, p. e36940, 2024/09/30/ 2024, doi: https://doi.org/10.1016/j.heliyon.2024.e36940.

[24] S. Shao, D. Fu, T. Yang, H. Mu, Q. Gao, and Y. Zhang, "Analysis of Machine Learning Models for Wastewater Treatment Plant Sludge Output Prediction," Sustainability, 2023.

[25] E. Ekinci, B. Özbay, S. I. Omurca, F. E. Sayın, and İ. Özbay, "Application of machine learning algorithms and feature selection methods for better prediction of sludge production in a real advanced biological wastewater treatment plant," Journal of environmental management, vol. 348, p. 119448, 2023.

[26] Y. Hu et al., "Exploring sludge yield patterns through interpretable machine learning models in China's municipal wastewater treatment plants," Resources, Conservation and Recycling, vol. 204, p. 107467, 2024/05/01/ 2024, doi:

https://doi.org/10.1016/j.resconrec.2024.107467.

[27] S. I. Abba, M. A. Yassin, S. M. H. Shah, J. Usman, A. Bashir, and I. H. Aljundi, "Tracking Health Risk-Based Resistivity in Treated Wastewater-Based on Pilot-System using Sensors, IoT, and 2nd-order Ensemble Machine Learning Algorithms," presented at the 2024 International Conference on Emerging Innovations and Advanced Computing (INNOCOMP), 2024. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/INNOCOMP63224.2024.00096.

[28] O. Inbar and D. Avisar, "Enhancing wastewater treatment through artificial intelligence: A comprehensive study on nutrient removal and effluent quality prediction," Journal of Water Process Engineering, vol. 61, p. 105212, 2024/05/01/ 2024, doi: https://doi.org/10.1016/j.jwpe.2024.105212.

[29] P. Vilasrao, "Water Quality Monitoring Using Machine Learning Model," Journal of Electrical Systems, vol. 20, pp. 5686-5694, 07/10 2024, doi: 10.52783/jes.6416.

[30] S. Kumari. "Beer Lambert Law | Equation, Units & Examples - Lesson." Study. https://study.com/academy/lesson/how-to-use-spectrophotometry-to-understand-beers-law.html#:~:text=in%20the%20tea.-,Lesson%20Summary,the%20concentration%20of%20the%20sample. (accessed.

[31] Turbidity_sensor_SKU__SEN0189-DFRobot. (n.d.). Wiki.dfrobot.com. https://wiki.dfrobot.com/Turbidity_sensor_SKU__SEN0189

Appendix A Gantt Chart

FYP 1

No	Activity	Week													
		2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	Contact Supervisor	🟨													
2	FYP Briefing & Research Methodology Workshop		🟨	🟨											
3	Search Topic		🟨	🟨	🟨										
4	Literature Review				🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	
5	**FYP 1-0a** Topic Proposal Submission				🟨										
6	Logbook Progress Evaluation 1, 2, 3					🟨			🟨			🟨			
7	Research Methodology					🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	
8	Prepare Preliminary Result									🟨	🟨	🟨	🟨	🟨	
9	**FYP 1** Presentation Slide Submission												🟨		
10	**FYP 1** Seminar													🟨	
11	**FYP** Report Draft											🟨	🟨	🟨	🟨

FYP 2

No	Activity	Week														
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	Data Collection & Analysis	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨					
2	Train Regression Model						🟨	🟨	🟨	🟨	🟨	🟨	🟨			
3	Making Interface and Database		🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨			
4	Logbook Progress Evaluation 1, 2, 3					🟨				🟨			🟨			
5	**FYP 2** Journal Paper				🟨									🟨		
6	Project Report Writing	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨			
7	**FYP 2** Seminar													🟨	🟨	
8	FYP 2 Report Writing									🟨	🟨	🟨	🟨	🟨	🟨	
9	FYP 2 Report Submission															🟨

Appendix B ARDUINO PSEUDOCODE

```
#include <WiFiS3.h>

char ssid[] = "ESP32_AP";
char pass[] = "12345678";
char server[] = "192.168.4.1"; // ESP32 AP IP
uint16_t port = 1234;
WiFiClient client;

void setup() {
  Serial.begin(115200);
  delay(5000);
  analogReadResolution(14);
  while (WiFi.begin(ssid, pass) != WL_CONNECTED) {
    delay(500);
  }
  Serial.println("Connected to ESP32 AP");
}

void loop() {
  if (!client.connected()) {
    client.connect(server, port);
    delay(100); // wait for connection
  }
  if (client.connected()) {
    // Example sensor data
    int sensorValue = analogRead(A0);
    float voltage = sensorValue * (5.0 / 16383.0);
    float turbidity = -1120.4 * pow(voltage, 2) + 5133.7264 * voltage - 2839.3503721;
    int methane = analogRead(A1);
    char payload[64];
    snprintf(payload, sizeof(payload), "%.2f,%d,%.4f\n", turbidity, methane, voltage);
    client.print(payload);
    Serial.print("Sent: ");
    Serial.print(payload);
  }
  delay(2000);
}
```

Appendix C ESP32 PSEUDOCODE

```
#include <WiFi.h>

const char* ssid = "ESP32_AP";
const char* password = "12345678";
WiFiServer server(1234);

WiFiClient clients[5]; // Allow up to 5 clients
const int maxClients = 5;

void setup() {
  Serial.begin(115200);
  delay(250);
  WiFi.softAP(ssid, password);
  server.begin();
  Serial.print("ESP32 AP IP: ");
  Serial.println(WiFi.softAPIP());
  Serial.println("TCP server started");
}

void loop() {
  // Accept new clients
  if (server.hasClient()) {
    for (int i = 0; i < maxClients; i++) {
      if (!clients[i] || !clients[i].connected()) {
        if (clients[i]) clients[i].stop();
        clients[i] = server.available();
        Serial.print("New client connected, slot: ");
        Serial.println(i);
        break;
      }
    }
    // If max clients reached, reject
    WiFiClient newClient = server.available();
    newClient.stop();
  }

  // Loop through clients to read data
  for (int i = 0; i < maxClients; i++) {
    if (clients[i] && clients[i].connected() && clients[i].available()) {
      String data = clients[i].readStringUntil('\n');
      Serial.print("Received: ");
      Serial.println(data);

      // Forward to all other clients (except sender)
      for (int j = 0; j < maxClients; j++) {
        if (j != i && clients[j] && clients[j].connected()) {
          clients[j].print(data + "\n");
        }
      }
    }
  }
}
```

Appendix D MATLAB APP DESIGNER PSEUDOCODE

```
classdef FinalGUI < matlab.apps.AppBase

  % Properties that correspond to app components
  properties (Access = public)
    UIFigure                  matlab.ui.Figure
    TabGroup                  matlab.ui.container.TabGroup
    PredictionTab             matlab.ui.container.Tab
    Image_2                   matlab.ui.control.Image
    TabGroup2                 matlab.ui.container.TabGroup
    HomeTab                   matlab.ui.container.Tab
    SaveButton                matlab.ui.control.Button
    StatusLamp                matlab.ui.control.Lamp
    UITable                   matlab.ui.control.Table
    LoadModelButton           matlab.ui.control.Button
    StatusLabel               matlab.ui.control.Label
    StartButton               matlab.ui.control.Button
    VisualTab                 matlab.ui.container.Tab
    UIAxes                    matlab.ui.control.UIAxes
    DataCollectionTab         matlab.ui.container.Tab
    Image                     matlab.ui.control.Image
    SludgeConcentrationEditField  matlab.ui.control.NumericEditField
    SludgeConcentrationEditFieldLabel  matlab.ui.control.Label
    MaxReadingsEditField      matlab.ui.control.NumericEditField
    MaxReadingsEditFieldLabel matlab.ui.control.Label
    StatusLabel_2             matlab.ui.control.Label
    StopButton                matlab.ui.control.Button
    StartButton_2             matlab.ui.control.Button
    UIAxes2                   matlab.ui.control.UIAxes
    UIAxes3                   matlab.ui.control.UIAxes
    UIAxes4                   matlab.ui.control.UIAxes
  end
```

```
properties (Access = private)
    tcpClient          % tcpserver object
    dataMatrix = []    % Matrix to store sensor data
    pollTimer              % Timer for polling data
    isConnected = false     % Client connection flag
    trainedModel           % Regression model
    turbidityBuffer = []
    methaneBuffer = []
    voltageBuffer = []
    readingCount = 0
    csvFilename = 'sensor_data_log.csv'
    DataCollectionActive = false
    batchData = []  % Add this to hold the averaged batch rows
end

methods (Access = private)

    function connectClient(app)
        try
            ip = "192.168.4.1"; % ESP32 AP IP
            port = 1234;        % ESP32 TCP server port

            % Close any previous client
            if ~isempty(app.tcpClient) && isvalid(app.tcpClient)
                clear app.tcpClient;
                app.tcpClient = [];
            end

            app.tcpClient = tcpclient(ip, port);

            % Start polling timer
            app.pollTimer = timer( ...
                'ExecutionMode', 'fixedSpacing', ...
                'Period', 0.2, ...
                'TimerFcn', @(~,~) app.readIncomingData());
            start(app.pollTimer);

            app.StatusLabel.Text = "Connected to ESP32 TCP server.";
            app.StatusLamp.Color = [0 1 0]; % Green
            app.StatusLamp.Tooltip = 'Connected';
            app.isConnected = true;
            app.StartButton.Text = "Disconnect";
        catch ME
            app.StatusLabel.Text = "Failed to connect.";
            app.StatusLamp.Color = [1 0 0];
            app.StatusLamp.Tooltip = 'Connection error!';
            uialert(app.UIFigure, ME.message, 'Client Error');
        end
    end

    function disconnectClient(app)
        try
            if ~isempty(app.pollTimer) && isvalid(app.pollTimer)
                stop(app.pollTimer);
                delete(app.pollTimer);
                app.pollTimer = [];
            end
            if ~isempty(app.tcpClient) && isvalid(app.tcpClient)
                clear app.tcpClient;
                app.tcpClient = [];
            end
        catch
        end
        app.StatusLabel.Text = "Disconnected.";
        app.StatusLamp.Color = [0.8 0.8 0.8];
        app.StatusLamp.Tooltip = 'Disconnected';
        app.isConnected = false;
```

```
            app.StartButton.Text = "Connect";
        end

        function readIncomingData(app)
            try
                while app.tcpClient.NumBytesAvailable > 0
                    dataLine = readline(app.tcpClient);
                    values = sscanf(dataLine, '%f,%f,%f');
                    if numel(values) == 3
                        % Predict using trained model
                        inputTable = array2table(values', ...
                            'VariableNames', {'Turbidity_NTU_',
'Methane_ADC_','Voltage_V_'});
                        if ~isempty(app.trainedModel)
                            predictedValue = app.trainedModel.predictFcn(inputTable);
                        else
                            predictedValue = NaN; % or '' or 0 depending on your preference
                        end

                        % Get timestamp
                        timestamp = datestr(now, 'yyyy-mm-dd HH:MM:SS');
                        % Append all data (sensor, prediction, timestamp)
                        app.dataMatrix = [app.dataMatrix; ...
                         {values(1), values(3), values(2), predictedValue, timestamp}];

                        % Update UI

                        if predictedValue >= 60
                        uialert(app.UIFigure, ...
                            sprintf('Warning: Predicted Value is high (%.2f%%).',
predictedValue), ...
                            'High Value Warning', ...
                            'Icon', 'warning');
                        end
                        tableData = app.dataMatrix;
                        for i = 1:size(tableData,1)
                            tableData{i,1} = sprintf('%.2f', tableData{i,1});
                        end
                        app.UITable.Data = cell2table(tableData, ...
                            "VariableNames", {'Turbidity_NTU_', 'Voltage_V_',
'Methane_ADC_', 'Predicted Sludge Concentration', 'Timestamp'});
                        if size(app.dataMatrix,1) > 1
                            timestamps = datetime(app.dataMatrix(:,5), 'InputFormat', 'yyyy-
MM-dd HH:mm:ss');
                   % Extract all predicted values from dataMatrix (assuming column
4)
                            % Defensive conversion: ensure all data is numeric
                            try
                                predictedVec = cellfun(@double, app.dataMatrix(:,4));
                                tbd          = cellfun(@double, app.dataMatrix(:,1));
                                vtbd         = cellfun(@double, app.dataMatrix(:,2));
                                mthn         = cellfun(@double, app.dataMatrix(:,3));
                            catch
                                % If conversion fails, display error and skip plotting
                                uialert(app.UIFigure, 'Non-numeric value detected in data
matrix. Skipping plot update.', 'Plot Error');
                                return;
                            end

                            % Clear axes to avoid overplotting
                            cla(app.UIAxes); cla(app.UIAxes2); cla(app.UIAxes3);
cla(app.UIAxes4);

                            plot(app.UIAxes, timestamps, predictedVec, '-.');
                            title(app.UIAxes, 'Live Sludge Concentration');
                            xlabel(app.UIAxes, 'Time, s');
                            ylabel(app.UIAxes, 'Sludge Concentration, %');
```

```
    plot(app.UIAxes2, timestamps, tbd, '-.');
    title(app.UIAxes2, 'Live Turbidity');
    xlabel(app.UIAxes2, 'Time, s');
    ylabel(app.UIAxes2, 'Turbidity, NTU');

    plot(app.UIAxes3, timestamps, vtbd, '-.');
    title(app.UIAxes3, 'Live Voltage (Turbidity)');
    xlabel(app.UIAxes3, 'Time, s');
    ylabel(app.UIAxes3, 'Voltage, V');

    plot(app.UIAxes4, timestamps, mthn, '-.');
    title(app.UIAxes4, 'Live Methane');
    xlabel(app.UIAxes4, 'Time, s');
    ylabel(app.UIAxes4, 'Methane, ADC');
end
if app.DataCollectionActive
% Buffer for averaging
app.turbidityBuffer(end+1) = values(1);
app.methaneBuffer(end+1)   = values(3);
app.voltageBuffer(end+1)   = values(2);

if length(app.turbidityBuffer) == 3
    avgTurbidity = mean(app.turbidityBuffer);
    avgMethane   = mean(app.methaneBuffer);
    avgVoltage   = mean(app.voltageBuffer);

    app.readingCount = app.readingCount + 1;
    maxReadings = app.MaxReadingsEditField.Value; % <-- moved up
here

    % --- Progress bar update for StartButton_2 ---
    btn = app.StartButton_2;
    barHeight = 15; barWidth = 200;
    progress = app.readingCount / maxReadings;
```

```
                currentProg = min(round((barWidth-2)*(progress)), barWidth-2);

                if isempty(btn.Icon)
                    bg = btn.BackgroundColor;
                    wbar = permute(repmat(bg, barHeight, 1, barWidth), [1, 3, 2]);
                    wbar([1,end],:,:) = 0;
                    wbar(:,[1,end],:) = 0;
                    btn.Icon = wbar;
                end

                RGB = btn.Icon;
                RGB(2:end-1, 2:currentProg+1, 1) = 0.25391;
                RGB(2:end-1, 2:currentProg+1, 2) = 0.41016;
                RGB(2:end-1, 2:currentProg+1, 3) = 0.87891;
                btn.Icon = RGB;
                btn.Text = sprintf('Collecting... %d%%', round(progress*100));
                drawnow;
                % --- End of progress bar update ---
                timestamp = datestr(now, 'yyyy-mm-dd HH:MM:SS');
                sludgeConc = app.SludgeConcentrationEditField.Value;

                % Store batch row in memory
                app.batchData = [app.batchData; {timestamp, sludgeConc,
avgTurbidity, avgMethane, avgVoltage}];

                % Clear buffers for next batch
                app.turbidityBuffer = [];
                app.methaneBuffer = [];
                app.voltageBuffer = [];

                if app.readingCount >= maxReadings
                    app.DataCollectionActive = false;
                    app.StatusLabel.Text = "Batch collection complete.";
                    app.StatusLabel_2.Text = "Batch collection complete.";
```

```
                        % Prompt for save
                        [file, path] = uiputfile('*.csv', 'Save Batch Data As');
                        if isequal(file,0)
                            uialert(app.UIFigure, 'User cancelled save. Data is not saved.',
'Save Cancelled');
                        else
                            fullPath = fullfile(path, file);
                            T = cell2table(app.batchData, ...
                                'VariableNames', {'Timestamp', 'SludgeConcentration',
'Turbidity_NTU', 'Voltage_V', 'Methane_ADC'});
                            writetable(T, fullPath);
                            uialert(app.UIFigure, 'Batch data saved successfully.',
'Success');
                        end
                        app.batchData = [];
                     end
                   end
                   end
                end
             end
          catch ME
             uialert(app.UIFigure, ME.message, 'Data Read Error');
          end
       end
    end


    % Callbacks that handle component events
    methods (Access = private)

       % Code that executes after component creation
       function startupFcn(app)
          % app.StartButton.Enable = 'off';
    app.StatusLabel.Text = 'Please load a trained model.';
    app.StatusLamp.Color = [0.8 0.8 0.8];
    app.StatusLamp.Tooltip = 'No model loaded';
end

% Button pushed function: StartButton
function StartButtonPushed(app, event)
    if app.isConnected
        app.disconnectClient();
    else
        app.connectClient();
    end
end

% Close request function: UIFigure
function UIFigureCloseRequest(app, event)
    if app.isConnected
        app.disconnectClient();
    end
    delete(app)
end

% Button pushed function: SaveButton
function SaveButtonPushed(app, event)
    if isempty(app.UITable.Data)
        uialert(app.UIFigure, 'No data to save.', 'Warning');
        return;
    end

    [file, path] = uiputfile('*.csv', 'Save Table Data');
    if isequal(file, 0)
        return;
    end
```

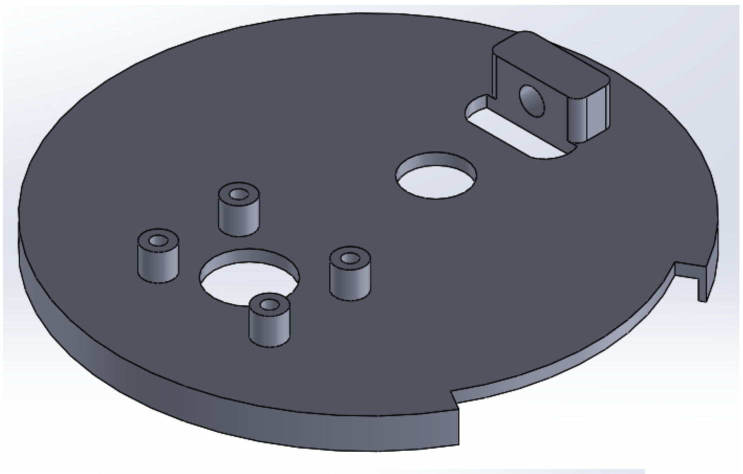
```
            fullPath = fullfile(path, file);
            try
                writetable(app.UITable.Data, fullPath);
                uialert(app.UIFigure, 'Data saved successfully.', 'Success');
            catch err
                uialert(app.UIFigure, ['Failed to save: ' err.message], 'Error');
            end
        end

        % Button pushed function: LoadModelButton
        function LoadModelButtonPushed(app, event)
            [file, path] = uigetfile('*.mat', 'Select Trained Model File');
            if isequal(file, 0)
                app.StatusLabel.Text = 'Model load cancelled by user.';
                return;
            end

            try
                modelPath = fullfile(path, file);
                s = load(modelPath);
                % Try to automatically find a regression model field
                fnames = fieldnames(s);
                modelVar = [];
                for i = 1:numel(fnames)
                    if isa(s.(fnames{i}), 'RegressionModel') || isfield(s.(fnames{i}),
'predictFcn')
                        modelVar = s.(fnames{i});
                        break;
                    end
                end
                if isempty(modelVar)
                    error('No regression model found in mat file.');
                end
                app.trainedModel = modelVar;
        app.StatusLabel.Text = 'Model loaded successfully.';
        app.StatusLamp.Color = [0 1 0];
        app.StatusLamp.Tooltip = 'Model loaded!';
        app.StartButton.Enable = 'on';   % <---- ADD THIS LINE HERE
    catch ME
        app.StatusLabel.Text = 'Failed to load model.';
        app.StatusLamp.Color = [1 0 0];
        app.StatusLamp.Tooltip = 'Model load error!';
        uialert(app.UIFigure, ME.message, 'Load Error');
    end
end

% Button pushed function: StartButton_2
function StartButton_2Pushed(app, event)
    btn = app.StartButton_2;
    originalText = btn.Text;
    originalIcon = btn.Icon;

    % Automatically restore button after completion/error
    cleanup = onCleanup(@()set(btn, 'Text', originalText, 'Icon', originalIcon));

    btn.Text = 'Collecting...';
    btn.IconAlignment = 'bottom';

    % Progress bar image setup
    barHeight = 15;
    barWidth = 200;
    bg = btn.BackgroundColor;
    wbar = permute(repmat(bg, barHeight, 1, barWidth), [1, 3, 2]);
    wbar([1,end],:,:) = 0; % Black border
    wbar(:,[1,end],:) = 0;
    btn.Icon = wbar;

    app.DataCollectionActive = true;
```

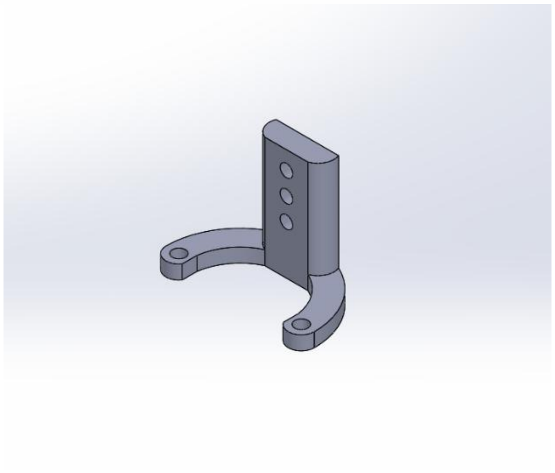
```
            app.turbidityBuffer = [];
            app.methaneBuffer = [];
            app.voltageBuffer = [];
            app.readingCount = 0;
            app.batchData = [];
            app.StatusLabel_2.Text = "Data collection started.";
            n = app.MaxReadingsEditField.Value;

            % This loop only simulates data collection progress!
            % The actual data collection is event-driven in your readIncomingData.
            % So, update the progress bar based on readingCount after each batch
            while app.DataCollectionActive && app.readingCount < n
                % Wait for a batch to be collected (simulate or poll)
                pause(0.1); % Small delay to allow UI update

                % Update progress bar (after each batch in readIncomingData, you should
also update)
                progress = app.readingCount / n;
                currentProg = min(round((barWidth-2)*(progress)), barWidth-2);
                RGB = btn.Icon;
                RGB(2:end-1, 2:currentProg+1, 1) = 0.25391; % R (royalblue)
                RGB(2:end-1, 2:currentProg+1, 2) = 0.41016;
                RGB(2:end-1, 2:currentProg+1, 3) = 0.87891;
                btn.Icon = RGB;
                btn.Text = sprintf('Collecting... %d%%', round(progress*100));
                drawnow;
            end
        end

        % Button pushed function: StopButton
        function StopButtonPushed(app, event)
            app.DataCollectionActive = false;
            app.StatusLabel_2.Text = "Data collection stopped.";
        end
```

```
    end

    % Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Get the file path for locating images
            pathToMLAPP = fileparts(mfilename('fullpath'));

            % Create UIFigure and hide until all components are created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.Position = [100 100 798 536];
            app.UIFigure.Name = 'MATLAB App';
            app.UIFigure.CloseRequestFcn = createCallbackFcn(app,
@UIFigureCloseRequest, true);

            % Create TabGroup
            app.TabGroup = uitabgroup(app.UIFigure);
            app.TabGroup.Position = [1 1 798 536];

            % Create PredictionTab
            app.PredictionTab = uitab(app.TabGroup);
            app.PredictionTab.Title = 'Prediction';

            % Create TabGroup2
            app.TabGroup2 = uitabgroup(app.PredictionTab);
            app.TabGroup2.Position = [1 1 796 511];

            % Create HomeTab
            app.HomeTab = uitab(app.TabGroup2);
            app.HomeTab.Title = 'Home';
            % Create StartButton
            app.StartButton = uibutton(app.HomeTab, 'push');
            app.StartButton.ButtonPushedFcn = createCallbackFcn(app,
@StartButtonPushed, true);
            app.StartButton.Position = [30 429 100 22];
            app.StartButton.Text = 'Start';

            % Create StatusLabel
            app.StatusLabel = uilabel(app.HomeTab);
            app.StatusLabel.Position = [30 392 736 38];
            app.StatusLabel.Text = 'Status';

            % Create LoadModelButton
            app.LoadModelButton = uibutton(app.HomeTab, 'push');
            app.LoadModelButton.ButtonPushedFcn = createCallbackFcn(app,
@LoadModelButtonPushed, true);
            app.LoadModelButton.Position = [620 429 100 22];
            app.LoadModelButton.Text = 'Load Model';

            % Create UITable
            app.UITable = uitable(app.HomeTab);
            app.UITable.ColumnName = {'Turbidity, NTU'; 'Voltage, V (Turbidity)';
'Methane, ADC'; 'Predicted Sludge Concentration, %'; 'Timestamp'};
            app.UITable.RowName = {};
            app.UITable.Position = [30 91 736 285];

            % Create StatusLamp
            app.StatusLamp = uilamp(app.HomeTab);
            app.StatusLamp.Position = [746 429 20 20];
            app.StatusLamp.Color = [0.8 0.8 0.8];

            % Create SaveButton
            app.SaveButton = uibutton(app.HomeTab, 'push');
```

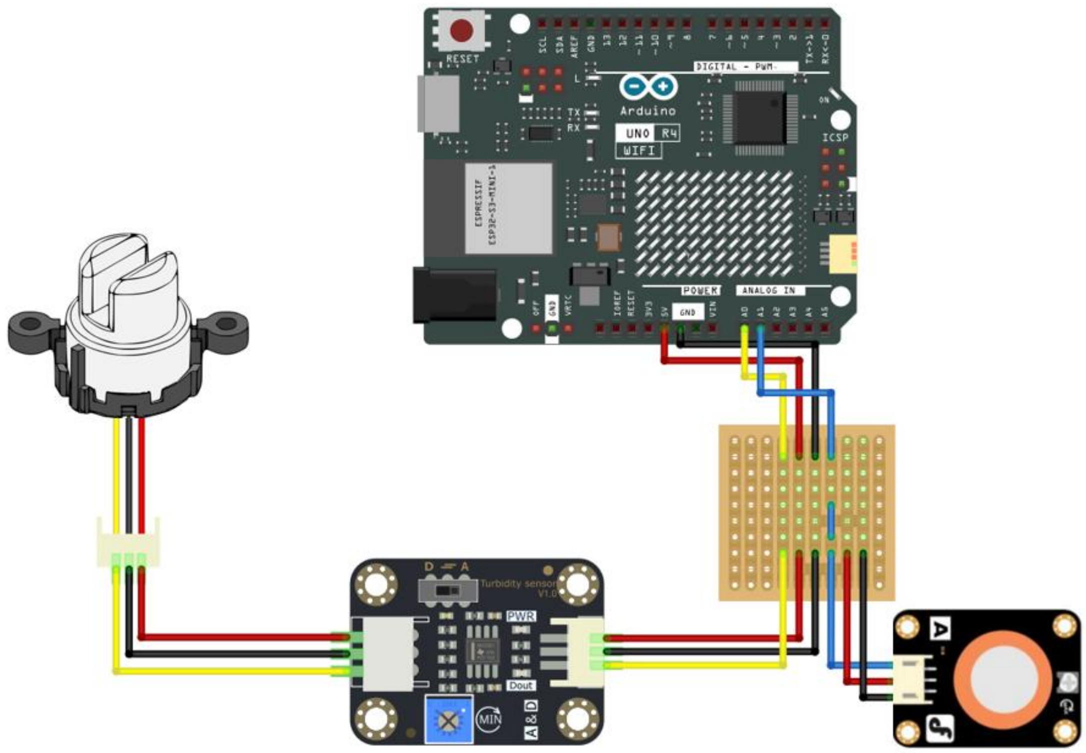
```
            app.SaveButton.ButtonPushedFcn = createCallbackFcn(app,
@SaveButtonPushed, true);
            app.SaveButton.Position = [666 56 100 22];
            app.SaveButton.Text = 'Save';

            % Create VisualTab
            app.VisualTab = uitab(app.TabGroup2);
            app.VisualTab.Title = 'Visual';

            % Create UIAxes
            app.UIAxes = uiaxes(app.VisualTab);
            title(app.UIAxes, 'Live Sludge Concentration')
            xlabel(app.UIAxes, 'Time, s')
            ylabel(app.UIAxes, 'Sludge Concentration, %')
            zlabel(app.UIAxes, 'Z')
            app.UIAxes.Position = [118 77 560 335];

            % Create Image_2
            app.Image_2 = uiimage(app.PredictionTab);
            app.Image_2.Position = [696 1 100 39];
            app.Image_2.ImageSource = fullfile(pathToMLAPP, 'Screenshot 2025-06-25
164123.png');

            % Create DataCollectionTab
            app.DataCollectionTab = uitab(app.TabGroup);
            app.DataCollectionTab.Title = 'Data Collection';

            % Create UIAxes4
            app.UIAxes4 = uiaxes(app.DataCollectionTab);
            title(app.UIAxes4, 'Live Methane')
            xlabel(app.UIAxes4, 'Time, s')
            ylabel(app.UIAxes4, 'Methane, ADC')
            zlabel(app.UIAxes4, 'Z')
            app.UIAxes4.Position = [235 27 328 185];
            % Create UIAxes3
            app.UIAxes3 = uiaxes(app.DataCollectionTab);
            title(app.UIAxes3, 'Live Voltage (Turbidity)')
            xlabel(app.UIAxes3, 'Time, s')
            ylabel(app.UIAxes3, 'Voltage, V')
            zlabel(app.UIAxes3, 'Z')
            app.UIAxes3.Position = [428 237 328 185];

            % Create UIAxes2
            app.UIAxes2 = uiaxes(app.DataCollectionTab);
            title(app.UIAxes2, 'Live Turbidity')
            xlabel(app.UIAxes2, 'Time, s')
            ylabel(app.UIAxes2, 'Turbidity, NTU')
            zlabel(app.UIAxes2, 'Z')
            app.UIAxes2.Position = [40 237 328 185];

            % Create StartButton_2
            app.StartButton_2 = uibutton(app.DataCollectionTab, 'push');
            app.StartButton_2.ButtonPushedFcn = createCallbackFcn(app,
@StartButton_2Pushed, true);
            app.StartButton_2.Position = [634 148 152 37];
            app.StartButton_2.Text = 'Start';

            % Create StopButton
            app.StopButton = uibutton(app.DataCollectionTab, 'push');
            app.StopButton.ButtonPushedFcn = createCallbackFcn(app,
@StopButtonPushed, true);
            app.StopButton.Position = [634 93 152 37];
            app.StopButton.Text = 'Stop';

            % Create StatusLabel_2
            app.StatusLabel_2 = uilabel(app.DataCollectionTab);
            app.StatusLabel_2.Position = [42 431 714 22];
            app.StatusLabel_2.Text = 'Status:';

            % Create MaxReadingsEditFieldLabel
            app.MaxReadingsEditFieldLabel = uilabel(app.DataCollectionTab);
            app.MaxReadingsEditFieldLabel.HorizontalAlignment = 'right';
            app.MaxReadingsEditFieldLabel.Position = [152 467 82 22];
            app.MaxReadingsEditFieldLabel.Text = 'Max Readings';

            % Create MaxReadingsEditField
            app.MaxReadingsEditField = uieditfield(app.DataCollectionTab, 'numeric');
            app.MaxReadingsEditField.Position = [249 467 100 22];
            app.MaxReadingsEditField.Value = 10;

            % Create SludgeConcentrationEditFieldLabel
            app.SludgeConcentrationEditFieldLabel = uilabel(app.DataCollectionTab);
            app.SludgeConcentrationEditFieldLabel.HorizontalAlignment = 'right';
            app.SludgeConcentrationEditFieldLabel.Position = [423 467 120 22];
            app.SludgeConcentrationEditFieldLabel.Text = 'Sludge Concentration';

            % Create SludgeConcentrationEditField
            app.SludgeConcentrationEditField = uieditfield(app.DataCollectionTab,
'numeric');
            app.SludgeConcentrationEditField.Position = [558 467 100 22];

            % Create Image
            app.Image = uiimage(app.DataCollectionTab);
            app.Image.Position = [696 1 100 39];
            app.Image.ImageSource = fullfile(pathToMLAPP, 'Screenshot 2025-06-25
164123.png');

            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end
    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = FinalGUI

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.UIFigure)

            % Execute the startup function
            runStartupFcn(app, @startupFcn)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```

Appendix E DESIGNED 3D PRINTED PARTS





Appendix F HARDWARE CONNECTION


RESET
SCL
SDA
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX->1
RX<-0
DIGITAL - PWM~
L
TX
RX
Arduino
UNO
R4
WIFI
ON
ICSP
ESPRESSIF
ESP32-S3-MINI-1
OFF
GND
VRTC
IOREF
RESET
3V3
5V
GND
VIN
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5
D
A
Turbidity sensor
V1.0
PWR
Dout
MIN
A&D
A