INDUSTRIAL IOT

AN INTERNSHIP REPORT

Submitted by

**TAMILSELVAN G** **731121105044**

**PAVISH S** **731121105027**

### in partial fulfilment for the award of the degree

of

**BACHELOR DEGREE IN ENGINEERING**

in

# ELECTRICAL AND ELECTRONICS ENGINEERING

**GOVERNMENT COLLEGE OF ENGINEERING, ERODE-638 316**

#### (Formerly known as Institute of Road and Transport Technology)

**ANNA UNIVERSITY: CHENNAI, 600 025**

26/06/2023 - 09/07/2023

# ANNA UNIVERSITY: CHENNAI, 600 025 BONAFIDE CERTIFICATE

Certificate that this internship report “**INDUSTRIAL INTERNET OF THINGS**” is the bonafide work of “**TAMIL SELVAN G & PAVISH S**” who have carried out the training in “**QUANTANICS TECHSERV PVT LTD**”.

# SIGNATURE SIGNATURE

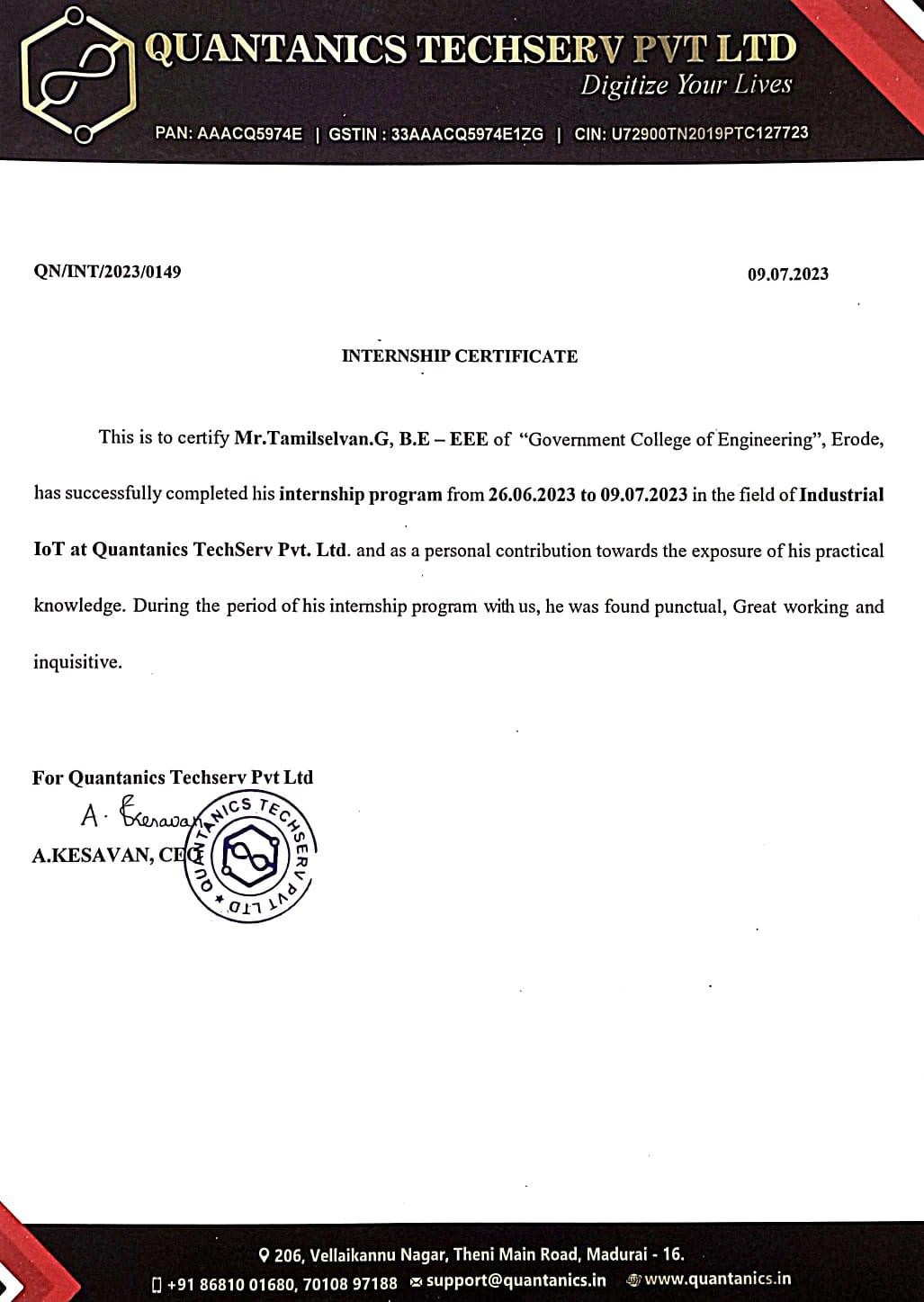
#### Dr. M. MOHAMMADHA Dr. S. GOMATHI, M.E., Ph.D.,

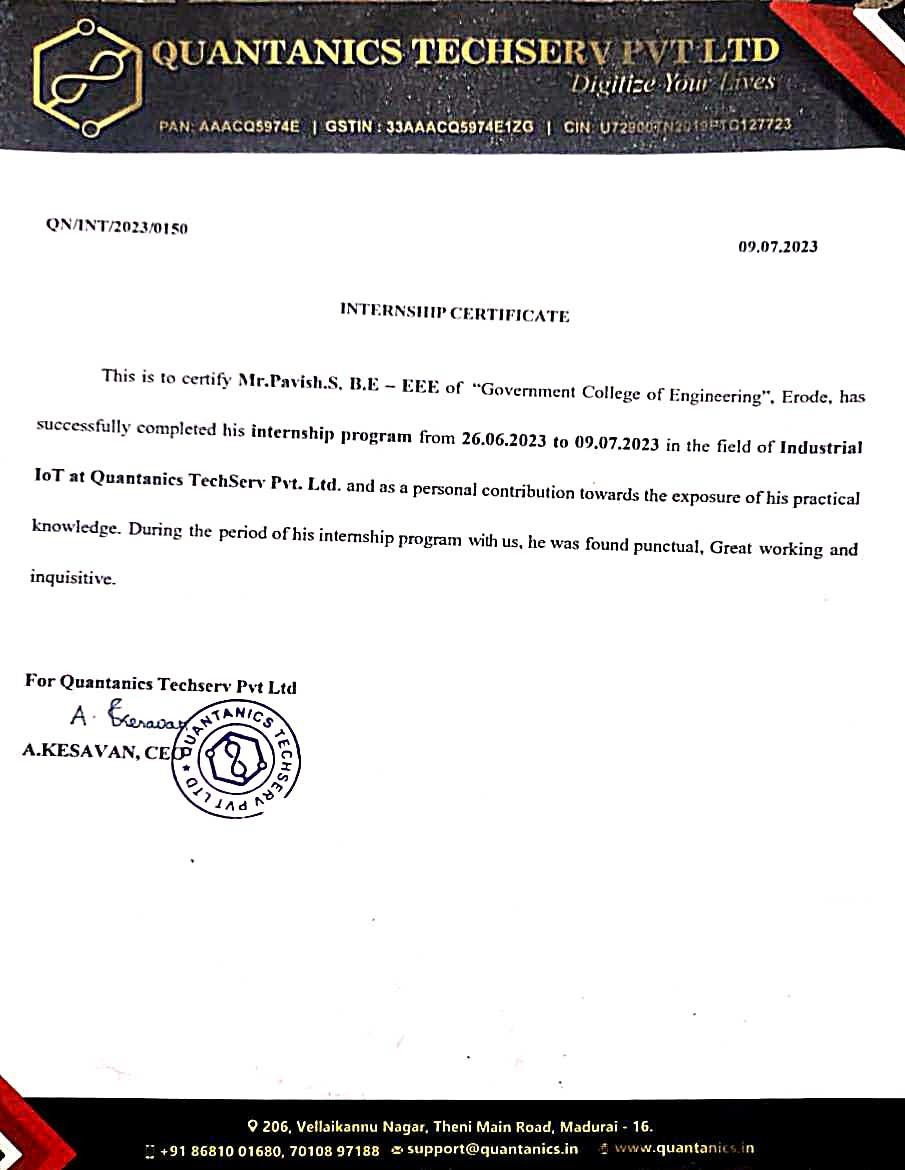
**HUSSAINI, M.E., Ph.D.,** CLASS ADVISIOR HEAD OF THE DEPARTMENT ASSISTANT PROFESSOR

ELECTRICAL & ELECTRONICS GOVERNMENT COLLEGE OF ENGINEERING, ENGINEERING,

GOVERNMENT COLLEGE OF ERODE-638316. ENGINEERING,

ERODE-632316.





# ACKNOWLEDGEMENT

Support on demand, encouragement at the needed moment and guidance in the right direction are indispensable for the success of any project. We have received these in excess from all corners from various people and we are glad to submit our gratitude to them.

We thank **Dr. R. MURUGESAN, M.E., Ph.D.,** the esteemed Head of our Institution who has been a source of motivation to all the staffs and students of our college. We are so much thankful to him.

Our sincere thanks to **Dr. M. MOHAMMADHA HUSSAINI, M.E., Ph.D., Associate Professor and Head of the Department** for her continuous support and motivation to undergo this internship.

We extend our profound gratitude to our class advisor **Dr. S. GOMATHI, M.E., Ph.D., Assistant Professor**, Department of Electrical and Electronics Engineering and **all teaching and nonteaching staff members**, for their valuable guidance and support.

Last, but not the least, we take this opportunity to thank our family, classmates and friends who offered an unconditional moral support during this internship.

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO** | **TABLE OF CONTENT**  **TITLE**  **LIST OF FIGURES** | **PAGE NO**  **vii** |
|  | **ABSTRACT** | **x** |
| **1** | **IOT BASED ENERGY METER** | **11** |
|  | **1.1 ENERGY METER** | **11** |
|  | **1.2 MULTIFUNCTION METER** | **12** |
|  | **1.3 SMART METERS** | **12** |
|  | 1.3.1 COMPONENTS | 12 |
|  | 1.3.2 ELITE 440 | 13 |
|  | 1.3.3 ARCHITECTURE OF IOT | 13 |
|  | **1.4 PROTOCOLS** | **14** |
|  | 1.4.1 UART | 14 |
|  | 1.4.2 I2C | 15 |
|  | 1.4.3 SPI | 16 |
|  | **1.5 MODBUS** | **18** |
|  | **1.6 MQTT** | **25** |
| **2** | **HACKATHON** | **27** |
|  | 2.1 INSTALLATION OF RASPBERRY | 36 |
|  | 2.2 ADVANCED IP SCANNER | 37 |
|  | 2.3 PUTTY | 38 |
|  | 2.4 XAMPP | 40 |
| **3** | **CONCLUSION** | **44** |
| **4** | **FEEDBACK** | **45** |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIG. NO** | **FIGURE NAME** | **PAGE NO** |
| 1.1 | OUTLINE OF THE INTERNET OF THINGS | 11 |
| 1.3.1.1 | ELITE 440 | 13 |
| 1.3.3.1 | ARCHITECTURE OF IOT | 13 |
| 1.5.1 | MODBUS POLL | 18 |
| 1.5.2 | MODBUS INSTALLATION | 19 |
| 1.5.3 | MODBUS APPLICATION | 19 |
| 1.5.4 | MODBUS RETRIVAL MODBUS | 20 |
| 1.5.5 | MODBUS CONNECTIONS | 20 |
| 1.5.6 | MODBUS DATA SETUP | 21 |
| 1.5.7 | MODBUS DATA TRAFFIC | 21 |
| 1.5.8 | MODBUS FUNCTION CODE | 22 |
| 1.5.9 | MODBUS PARAMETERS | 23 |
| 1.5.10 | MODBUS ERRORS | 24 |
| 1.6.1 | MQTT SETUP | 25 |
| 1.6.2 | MQTT CLIENT | 25 |
| 1.6.3 | MQTT CONNECTION | 26 |
| 2.1 | ARDUINO | 28 |
| 2.2 | DHT11 SENSOR | 29 |
| 2.3 | CONNECTIONS DIAGRAM | 30 |
| 2.4 | ARDUINO SETUP | 31 |
| 2.1.1 | RASPBBERRY PI SETUP | 36 |
| 2.1.2 | RASPBERRY PI OS | 36 |
| 2.2.1 | ADVANCE IP SETUP | 37 |
| 2.2.2 | IP ADDRESS | 38 |
| 2.3.1 | PUTTY SETUP | 38 |
| 2.3.2 | PUTTY SCANNING | 39 |
| 2.4.1 | XAMPP DATA SHEET | 40 |

* + 1. XAMPP TABLE 40
    2. XAMPP OUTPUT 41
    3. GRAFANA DATA FEED 42
    4. GRAFANA DATABASE SELECTION 42
    5. GRAFANA DATABASE SELECTION-2 43
    6. GRAFANA OUTPUT GRAPH 43

# ABSTRACT

Over the past few years, IoT has become one of the most important technologies of the 21st century. Now it connects everyday objects—kitchen appliances, cars, thermostats, baby monitors—to the internet via embedded devices, seamless communication is possible between people, processes, and things. By means of low-cost computing, the cloud, big data, analytics, and mobile technologies, physical things can share and collect data with minimal human intervention. In this hyperconnected world, digital systems can record, monitor, and adjust each interaction between connected things. The physical world meets the digital world—and they cooperate. IoT applications use machine learning algorithms to analyse massive amounts of connected sensor data in the cloud. Using real-time IoT dashboards and alerts, it gains visibility into key performance indicators, statistics for mean time between failures, and other information. Machine learning–based algorithms can identify equipment anomalies and send alerts to users and even trigger automated fixes or proactive counter measures. Hence the internet of thing is useful to monitoring the live data in wireless from all over the world.

**CHAPTER 1**

**WEEK 1-ACTIVITIES**

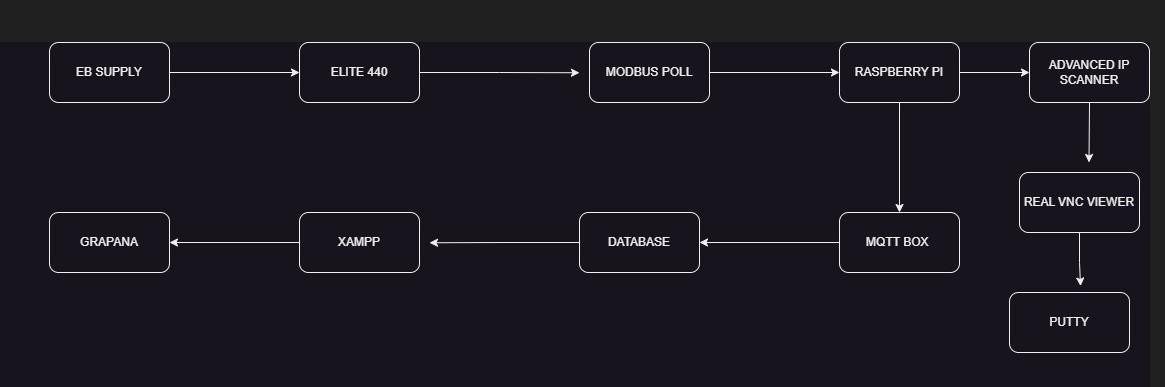
**IOT BASED ENERGY METER**

* 1. **ENERGY METER**

An energy consumption meter or power meter is a device used to measure and monitor the energy consumption of electrical application in a building (home or industry)

* It can measure active power (W), apparent power (VAR), voltage (V), current(A) and power factor.
* It is ideal to say that a normal EB meter does not consist of COMPORT (communication port).

A normal EB meter can only read lesser detailing compared to a multifunction meter. A normal meter requires manual support to read data.



**FIG 1.1: OUTLINE OF THE INTERNET OF THINGS**

# MULTIFUNCTION METER

A multifunction meter is an advanced version of the available power meter which can give great detailing compared to the older one. It can measure Active power (Watts), Reactive power (VAR), Apparent Power (VA), Voltage (Volts), Current (Amperes), power Factor, Frequency (Hertz). In addition to reading data, multifunction meters consist of COM ports which can be used to send data to a client using different protocols and hence doesn’t need manual support for data measurement.

Hence compared to a normal EB meter a multifunction meter can do more data detailing and can transmit data to an external software.

# SMART METERS

Smart meters are an extended version of the MFM, which facilitates two-way data transmission it can both send and receive data through the presence of COM Port. For example, when there is high power consumption the external software sends an alert signal to the meter. Here the data transmission is done through MODBUS PROTOCOL which transmits data from the smart meter to the raspberry for further processing.

# COMPONENTS

A smart meter can be both AC or DC based on the requirements. ELITE 440 is the smart meter used here.

The schematic diagram of the smart meter is given below:

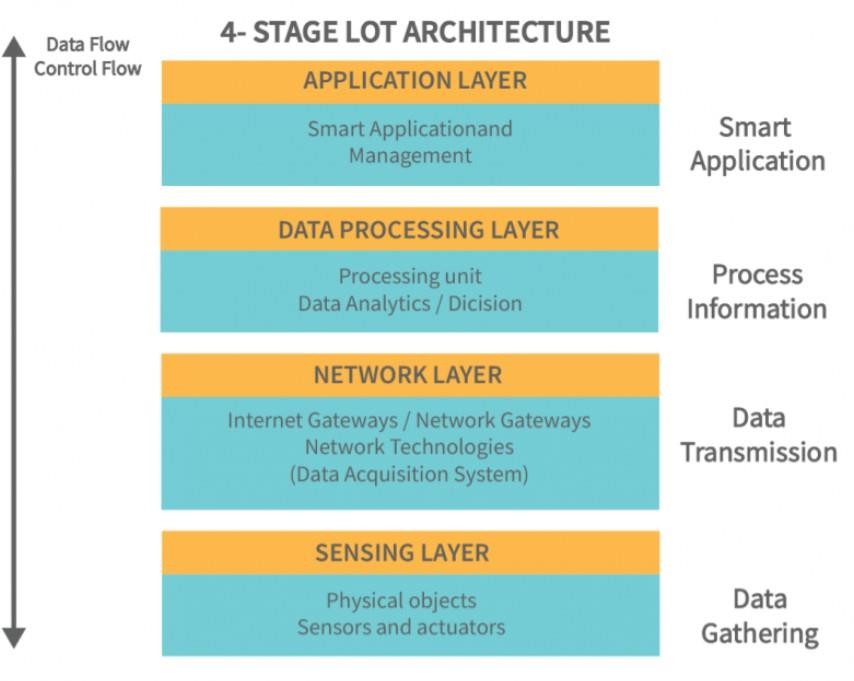


**FIG 1.3.1.1: ELITE 440**

# ELITE 440

The above image is elite 440 smart energy meter it will measure the voltage, current and power, kWH etc... The above reading directly sent to MODBUS protocol to see the live data using raspberry pi. This energy meter is generally used in the industrial application. The above-mentioned elite 440 is a 3-phase meter. By using VNC viewer and putty we will get the IP address of the raspberry pi connected to the elite 440.

# ARCHITECTURE OF IOT

****

**FIG 1.3.3.1: ARCHITECTURE OF IOT**

# PROTOCOLS

Peripheral device refers to an external hardware input or output device. These are the major types of serial bus protocols.

* UART
* I2C
* SPI

# UART

UART stands for universal asynchronous receiver transmitter protocol. It is used for serial communication between two devices.

UART can have 4 ways namely:

* + - * Rx
      * Tx
      * 5 V
      * Ground

UART can be used for two types of applications:

* + - * General purpose.
      * Industrial purpose.

When a protocol is used for general purposes like household purpose data loss is not given much importance, it is not a closed loop (i.e.) there is no feedback to ensure there is successful transmission of data. And the distance of transmission is comparatively low.

But an industrial application protocol cannot encourage data loss and has to operate over long distances. Hence, it has to be closed loop to ensure feedback message has been received.

# MAJOR INFORMATIONS FOR A UART:

1. Baud rate:

The number of bits transmitted or received per second. It is usually 9600.

1. Start bit:

It is usually 7 or 8. Out of these 7 or 8 bits the last bit acts as the Cyclic redundancy check or error bit. Which shows the end of data.

1. Parity bit:

Parity bit is used for error checking purposes by detecting error in the transmitted data. It can be of three types:

* + Even parity.
  + Odd parity.
  + None parity.

1. Stop bit:

It can be 1.5 or 1 or 2.

It is usually a 1 stop bit for RS-485 and 2 stop bits for RS-232.

RS-232 is also a type of MODBUS protocol in which the slave sends data without the master requesting for data. Hence, there is much data loss in a RS-232 Modbus protocol.

# I2C

I2C communication stands for inter integrated circuit. It is a bus interface communication protocol which has 2 lines for data communication called SDA and SCL.

Here SDA stand for serial data which transfers data and SCL stands for serial clock which carries out the clock signal. These kinds of I2C compatible devices are found in embedded systems that include EEPROMS.

I2C is a serial bus protocol it can transmit ‘n’ amount of data. It has minimal data loss as it times the data transmission using a clock. The clock decides the amount of time through

which data has to be transmitted hence the receiver waits until the complete transmission of data which helps in minimizing data loss.

# SPI PROTOCOL

SPI stands for serial peripheral interface. SPI is a synchronous serial communication interface commonly used for short distance communication between microcontrollers, sensors, and other peripheral devices. The SPI communication protocol allows data to be simultaneously transmitted and received.

The basic SPI configuration involves four signal lines:

1. MISO (Master Input, Slave Output)
2. MOSI (Master Output, Slave Output)
3. SCL (Serial Clock)
4. CS (Chip Select)

## UART (Universal Asynchronous Receiver – Transmitter)

* + UART is a simple and widely used serial communication protocol.
  + It uses two wires, commonly referred to as TX (transmit) and RX (receive), for data transfer.
  + It operates in an asynchronous mode, where data is sent as a stream of bits without a clock signal.
  + UART is often used for simple point-to-point communication between two devices, such as between a microcontroller and a computer or between two microcontrollers.
  + It has relatively low data transfer rates compared to SPI and I2C.
  + UART does not support multi-master communication or device addressing.

## SPI (Serial Peripheral Interface)

* + SPI is a synchronous serial communication protocol.
  + It used four wires: SCK, MOSI, MISO, and SS.
  + SPI typically involves a master device that controls communication with one or more slave devices.
  + It does not have a standardized addressing scheme for multiple devices on the same bus.

## I2C (Inter-Integrated Circuit)

* + I2C is a synchronous serial communication protocol.
  + It used two: SCL (serial clock) and SDA (serial data).
  + I2C employs a multi-master architecture, allowing multiple devices to share the same bus.
  + It supports device addressing, enabling communication with a large number of devices connected to the same bus.
  + It operates in half-duplex mode, where data is transferred in one direction at a time.
  + It offers lower data transfer rates compared to SPI but is suitable for communication with devices That don’t require high speed data transfer
  + It is commonly used for communication with various peripherals, such as sensors, EEPROMs, real time clocks, and other integrated circuits.

In summer, UART is a simple asynchronous protocol for point-to-point communication. SPI is a fast synchronous protocol, primarily used for communication with peripherals, and I2C is a multi-master synchronous protocol that supports device addressing for communication with multiple devices on same bus.

# MODBUS

MODBUS is a communication protocol commonly used in industrial automation systems to establish communication between electronic devices. MODBUS is a master slave protocol, meaning that there is a master device that initiates communication and one or more slave devices that responds to the Master’s request. MODBUS Protocol is the most preferred communication protocol for industrial purpose, as it gives back feedback on the data received. Hence resulting in lesser data losses.

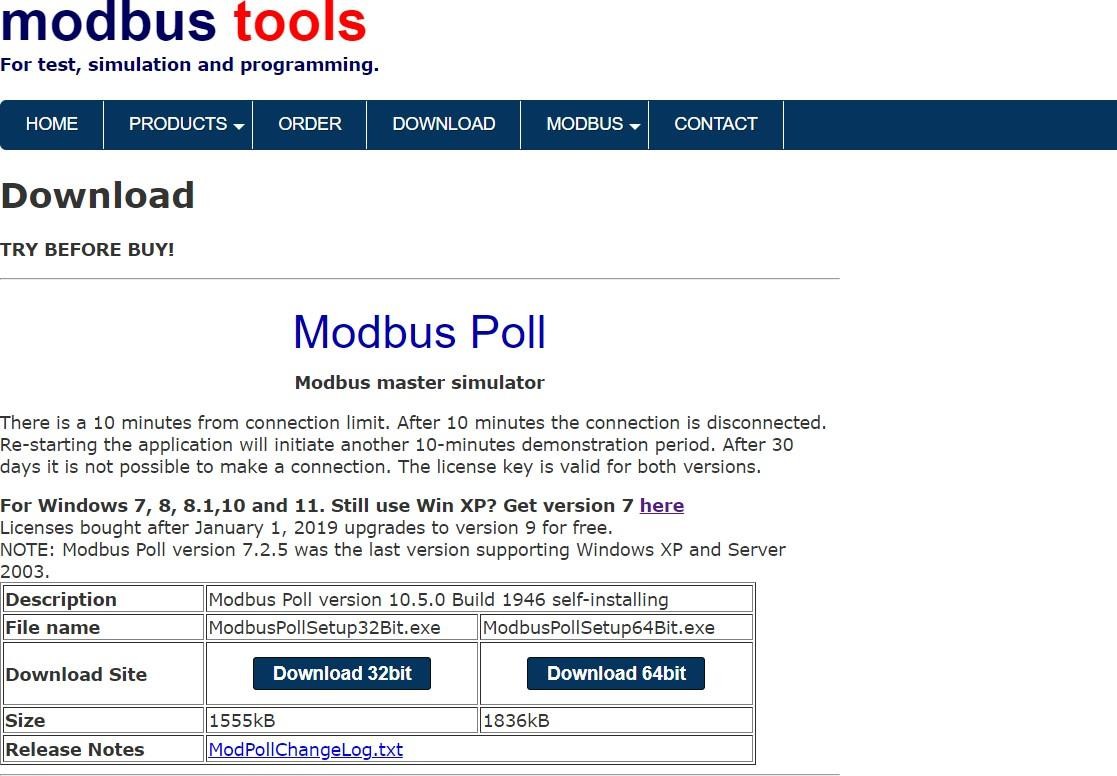
There are different variants of Modbus protocol:

* MODBUS RTU
* MODBUS TCP/IP

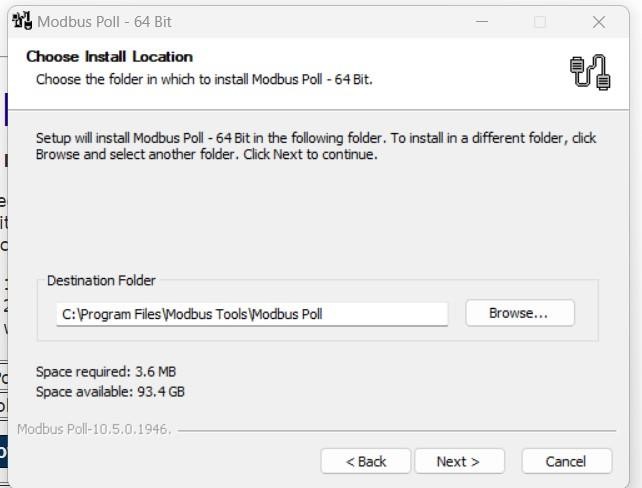
# MODBUS POLL:

## Installation process:

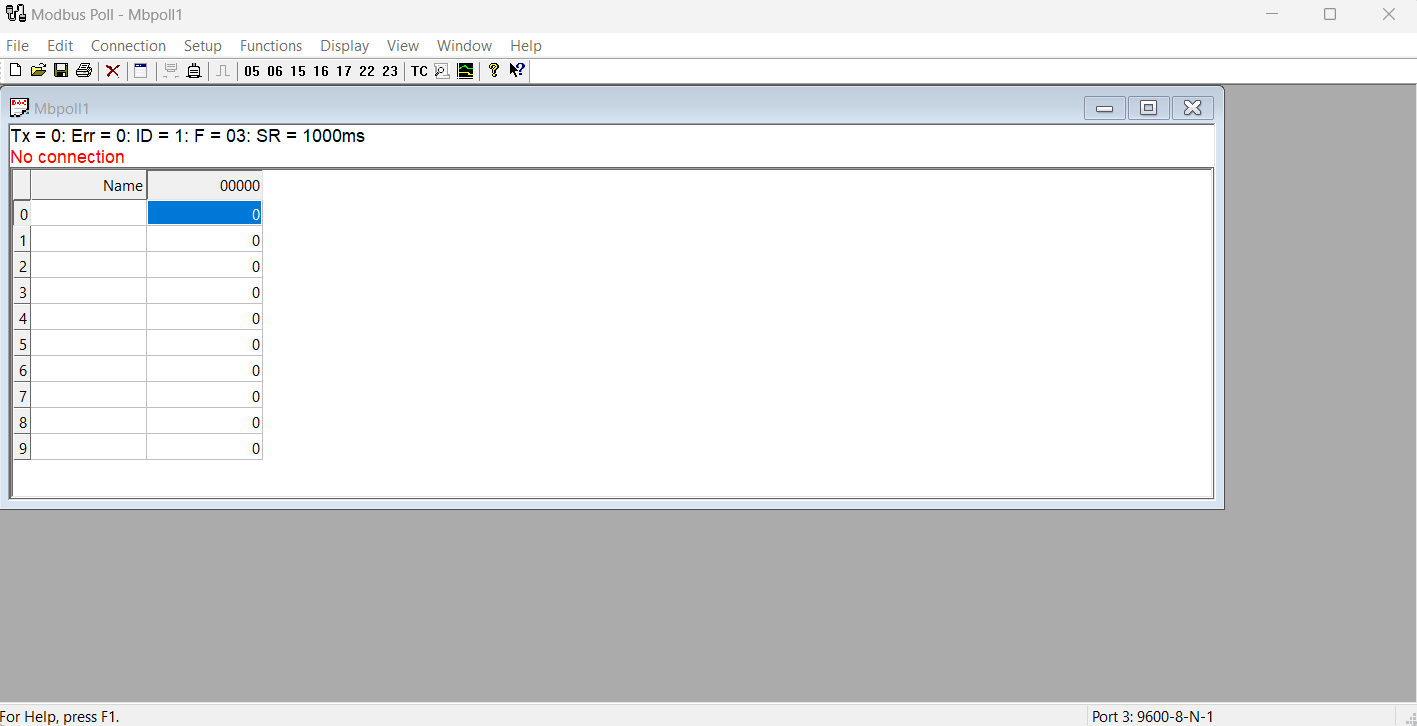
Go to MODBUS poll download.



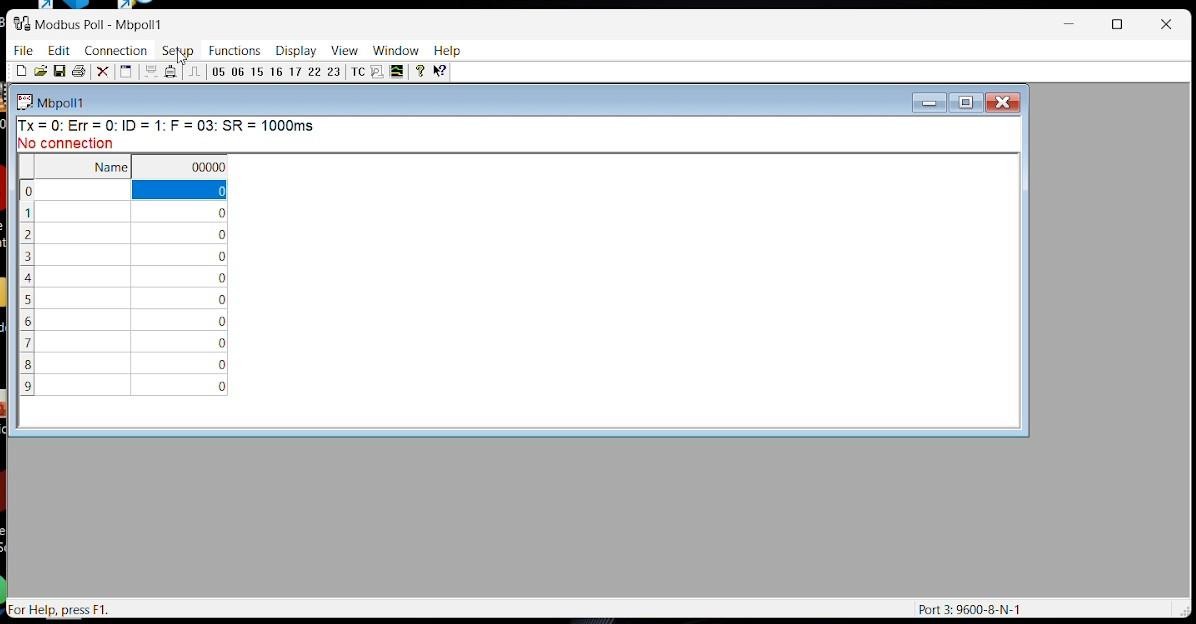
**FIG 1.5.1: MODBUS POLL**



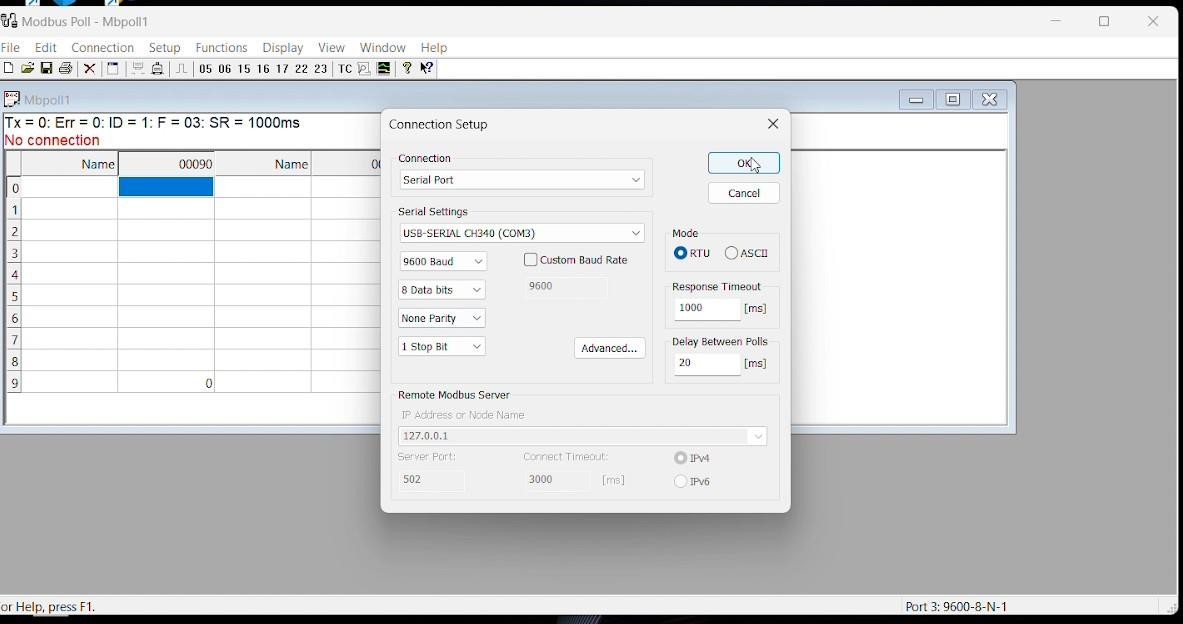
**FIG 1.5.2: MODBUS INSTALLATION**

****

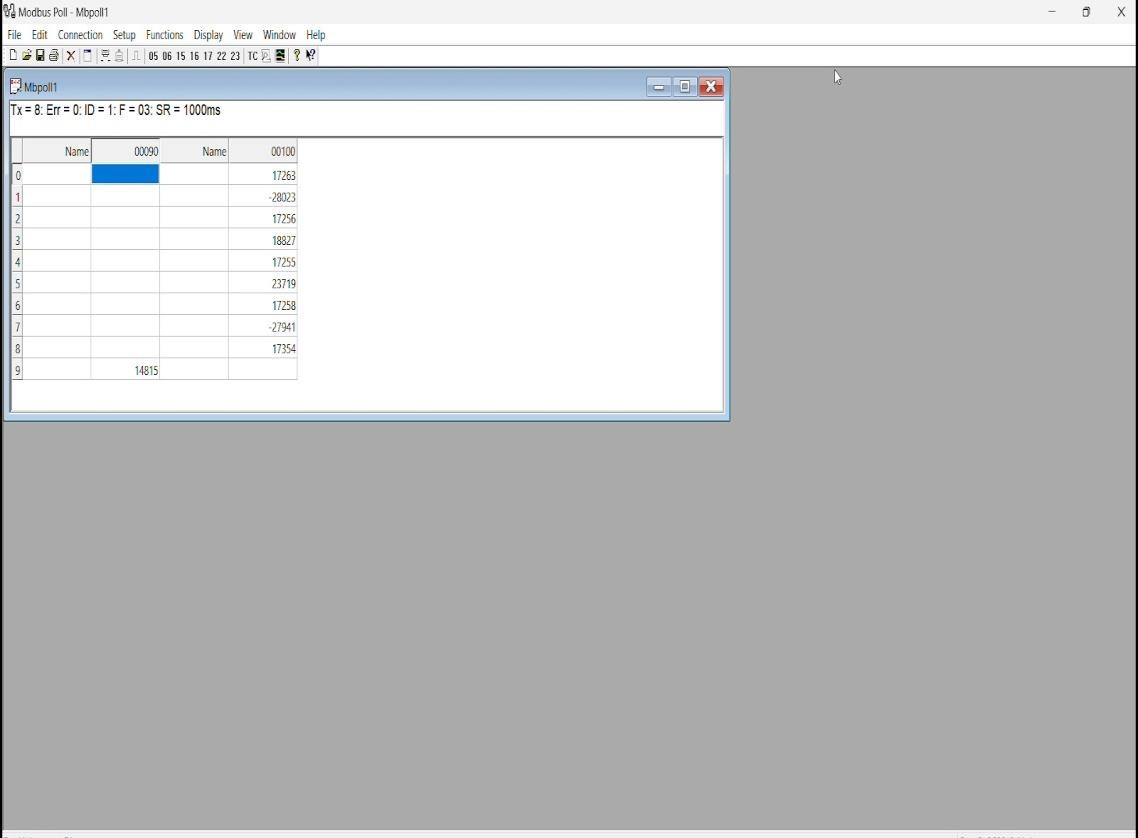
**FIG 1.5.3: MODBUS APPLICATION**



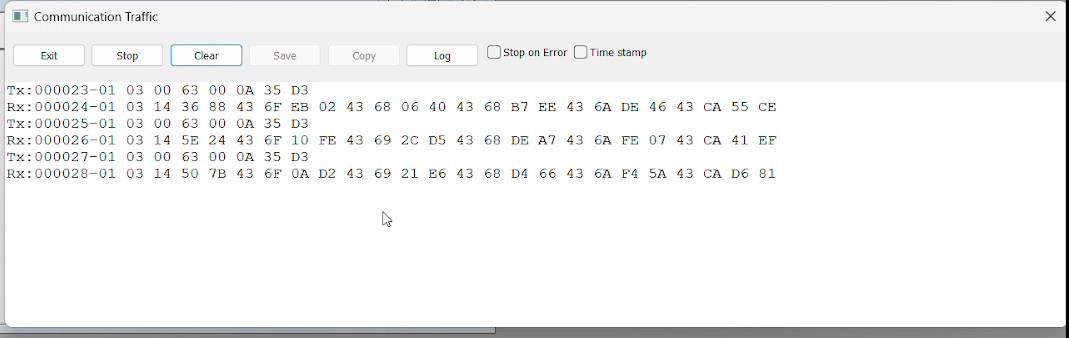
**FIG 1.5.4: MODBUS RETRIVAL MODBUS**

****

**FIG 1.5.5: MODBUS CONNECTION**

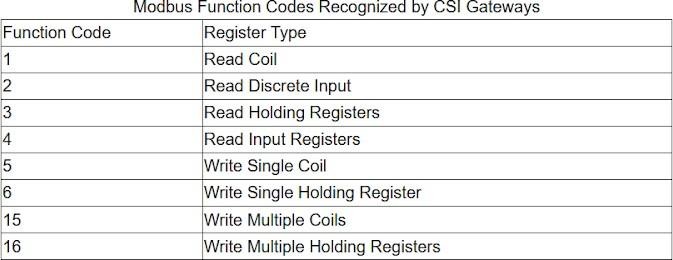


**FIG 1.5.6: MODNUS DATA SETUP**

****

**FIG 1.5.7: MODBUS DATA TRAFFIC**

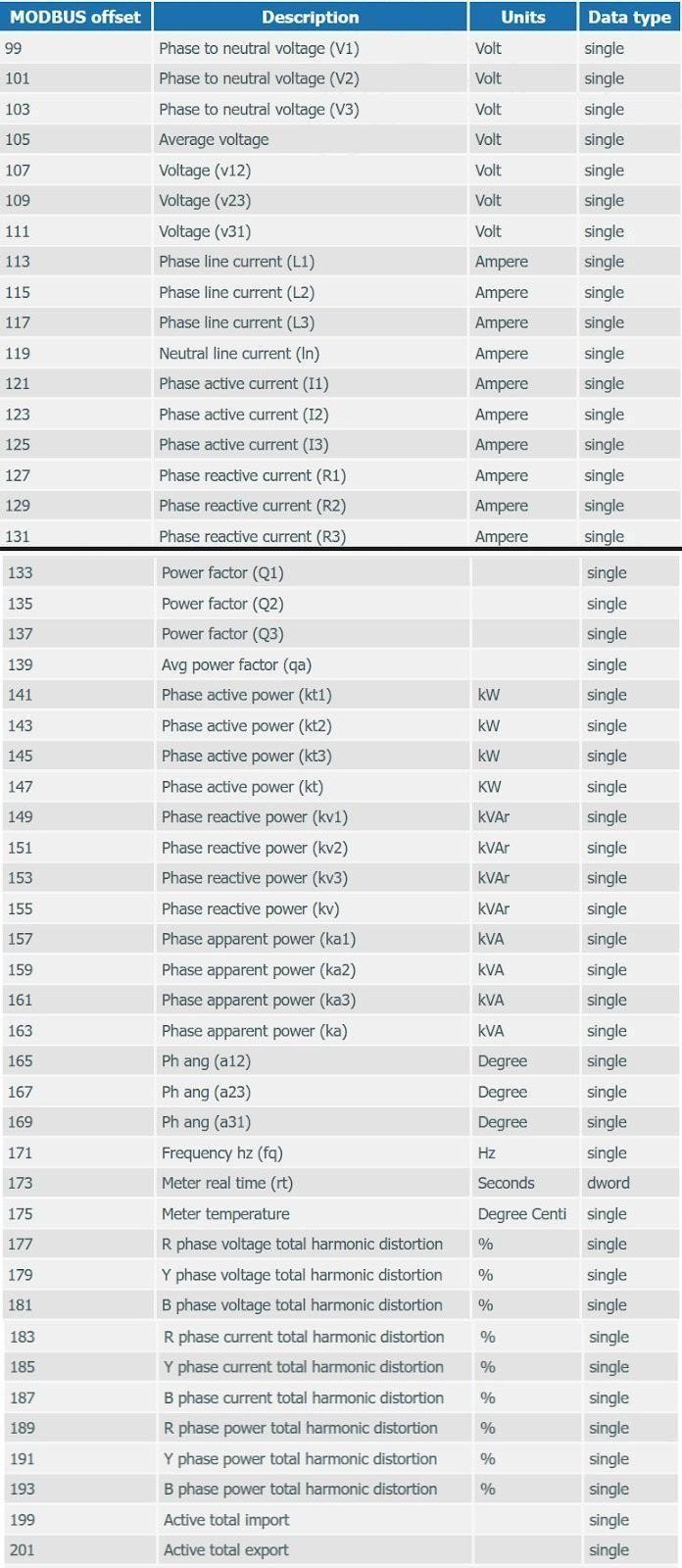
In the above data of “Tx”-03 represents the register used to retrieve data. From the specification of ELITE 440 here read holding register is used. The below table gives reference to the different register in RS-485 MODBUS and their representative number.



**FIG 1.5.8: MODBUS FUNCTION CODE**

In the next set of data in “Tx”-00 0A represents the number of data requested and the last set of data represent error checking pair. In the response line of the slave:

The first data “03” represent the register type, the next pair of data represents the number of bits to be reported and then each pair of consecutive data represents the output of each corresponding description for the given offset MODBUS register number.

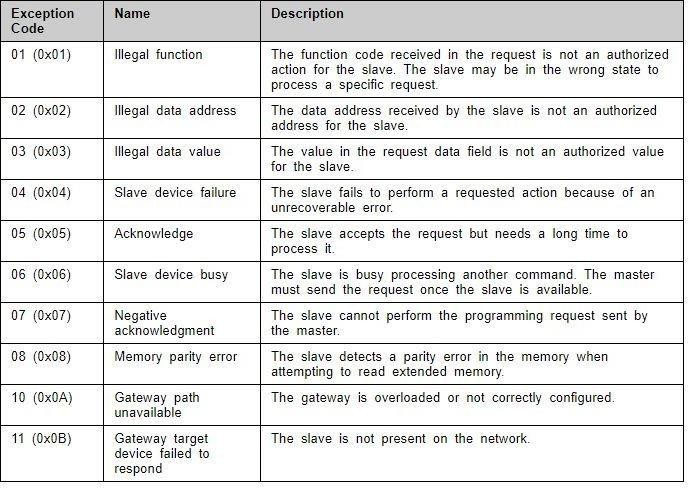


**FIG 1.5.9: MODBUS PARAMETERS**

In the next set of data in “Tx”-00 0A represents the number of data requested and the last set of data represent error checking pair. In the response line of the slave:

The first data “03” represent the register type, the next pair of data represents the number of bits to be reported and then each pair of consecutive data represents the output of each corresponding description for the given offset MODBUS register number.

Different errors that can be deducted in a MODBUS poll:



**FIG 1.5.10: MODBUS ERRORS**

## JSON data type:

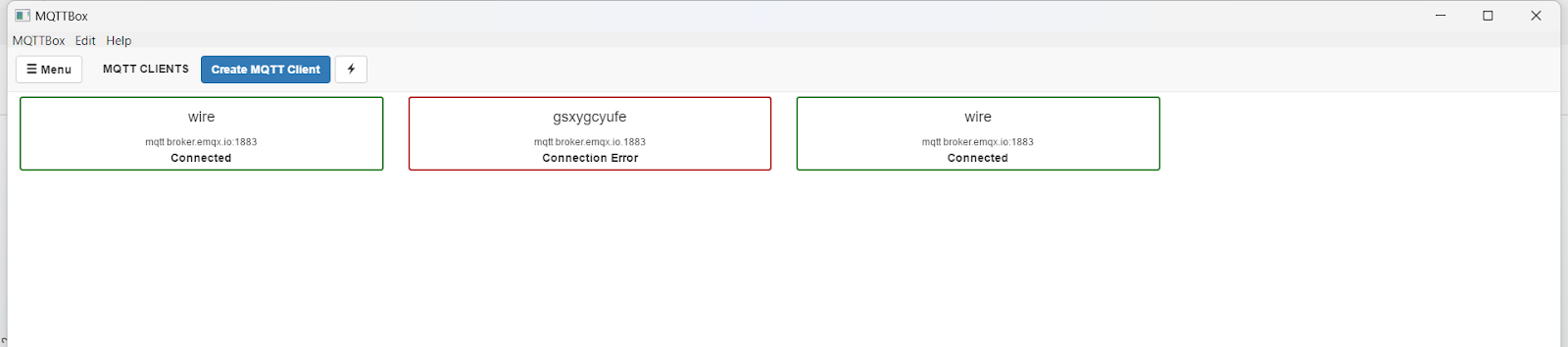
JSON stands for JavaScript Object Notation. It is a standard text-based and lightweight format. It is mainly used for human-readable date interchange. It supports almost all kinds of languages. It has structured data. It is used when the data is transferred form client to the server.

## MQTT (Message Queueing Telemetry Transmission)

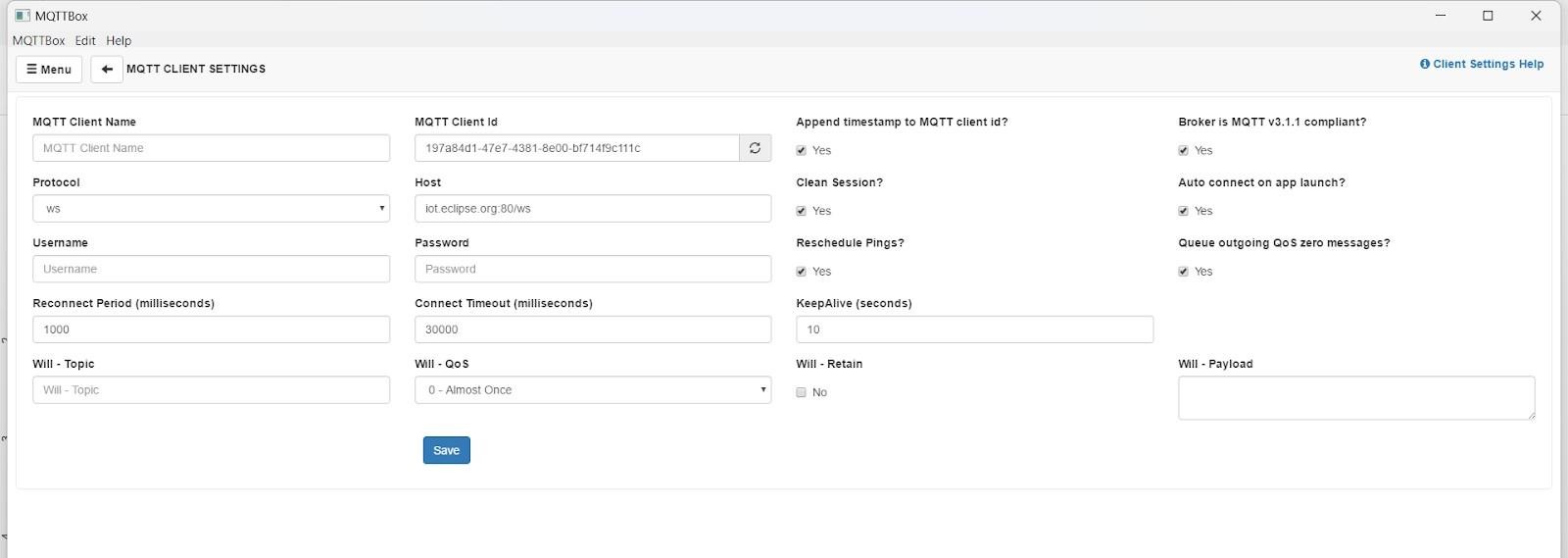
MQTT follows a publish-subscribe model, where devices or applications can publish messages to topics, and other devices or application can subscribe to those topics to receive the messages. The sender publishes data and the receiver subscribes to the chosen topic to receive data. The devices connect to the broker through a unique client id. Here, the raspberry connects to the broker id as a client to publish its data.

# STEPS TO RETRIEVE DATA THROUGH MQTTBOX:

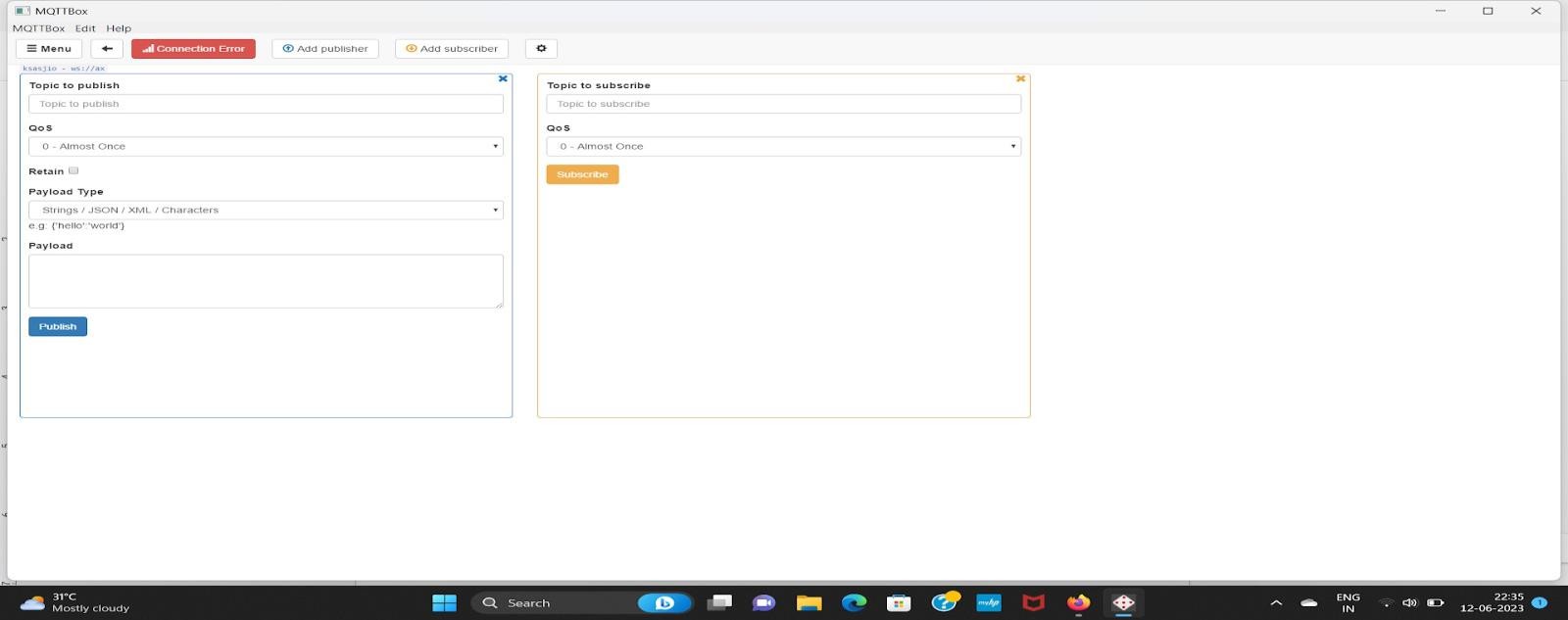
### Open MQTT box



**FIG 1.6.1: MQTT SETUP**

****

**FIG 1.6.2: MQTT CLIENT**



**FIG 1.6.3: MQTT CONNECTION**

# CHAPTER 2

**WEEK 2-ACTIVITIES HACKATHON**

## Visualising and prediction of Temperature and Humidity

#### Components used:

* DHT11 sensor
* Arduino UNO

#### Software’s used:

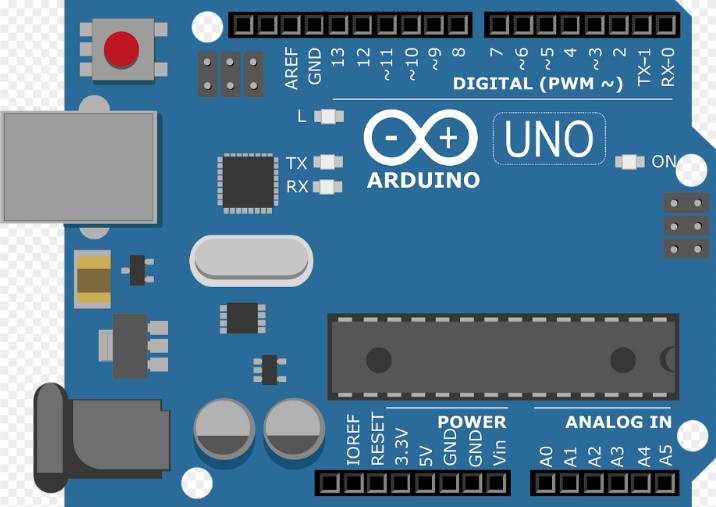
* Arduino IDE
* Xampp server
* Grafana

#### Arduino UNO:

The Arduino Uno is an open-source microcontroller board based on the Microchip ATmega328P microcontroller (MCU) and developed by Arduino.cc and initially released in 2010. The microcontroller board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board has 14 digital I/O pins (six capable of PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable.

It can be powered by a USB cable or a barrel connector that accepts voltages between 7 and 20 volts, such as a rectangular 9-volt battery. It has the same microcontroller as the Arduino Nano board, and the same headers as the Leonardo board. The hardware reference design is distributed under a Creative Commons Attribution Share-Alike 2.5 licence and is available on the Arduino website.

Layout and production files for some versions of the hardware are also available.



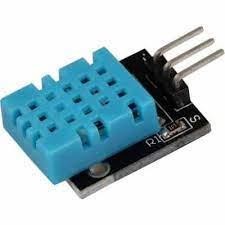
**FIG 2.1: ARDUINO**

The word "uno" means "one" in Italian and was chosen to mark a major redesign of the Arduino hardware and software. The Uno board was the successor of the Duemilanove release and was the 9th version in a series of USB-based Arduino boards. Version 1.0 of the Arduino IDE for the Arduino Uno board has now evolved to newer releases. The ATmega328 on the board comes pre-programmed with a bootloader that allows uploading new code to it without the use of an external hardware programmer.

While the Uno communicates using the original STK500 protocol, it differs from all preceding boards in that it does not use a FTDI USB-to-UART serial chip. Instead, it uses the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

#### DHT11 sensor:

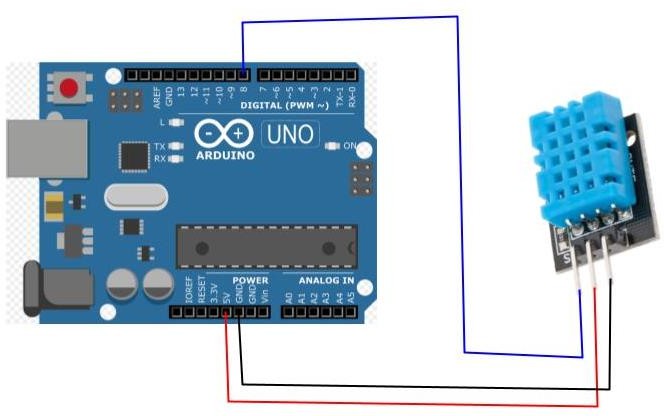
This is a calibrated digital temperature and humidity module with onboard sensor DHT11. It can be used for detecting ambient temperature and humidity, through the standard single-wire interface.



**FIG 2.2: DHT11 SENSOR**

* Sensor: DHT11
* Working Voltage: 3.3V-5.5V
* Temperature
  + Resolution: 1°C
  + Accuracy: ±2℃
  + Measuring range: 0°C ~ 50°C
* Humidity
  + Resolution: 1%RH
  + Accuracy: ±5%RH (0~50°C)
  + Measuring range: 20%RH ~ 90%RH (25°C)
* Operating voltage: 3.3V ~ 5.5 V
  + Recommended storage condition
  + Temperature: 10°C ~40°C
  + Humidity: 60%RH or below
* Fixed Hole Size: 2.0mm

**Connection:**

****

**FIG 2.3: CONNECTION DIAGRAM**

**Arduino IDE:**

The Arduino IDE is an open-source software, which is used to write and upload code to the Arduino boards. The IDE application is suitable for different operating systems such as Windows, Mac OS X, and Linux. It supports the programming languages C and C++. Here, IDE stands for Integrated Development Environment.

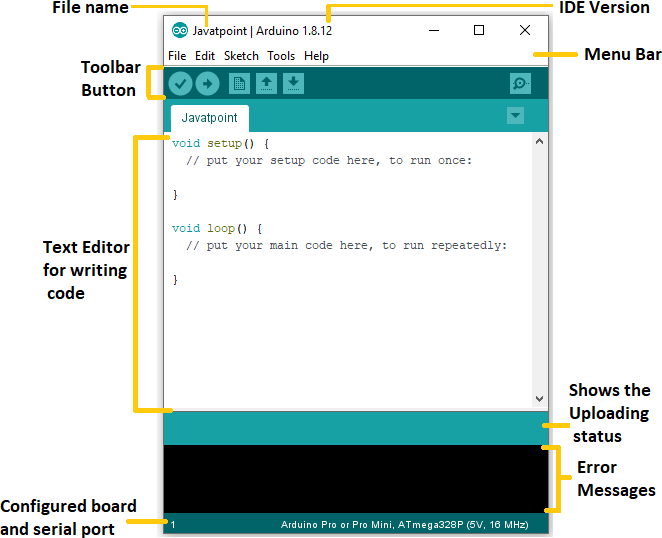
The program or code written in the Arduino IDE is often called sketching. We need to connect the Genuino and Arduino board with the IDE to upload the sketch written in the Arduino IDE software. The sketch is saved with the extension '.ino.'

The Upload button compiles and runs our code written on the screen. It further uploads the code to the connected board. Before uploading the sketch, we need to make sure that the correct board and ports are selected.

We also need a USB connection to connect the board and the computer. Once all the above measures are done, click on the Upload button present on the toolbar. The latest Arduino boards can be reset automatically before beginning with Upload. In the older

boards, we needed to press the Reset button present on it. As soon as the uploading is done successfully, we can notice the blink of the Tx and Rx LED.

The Arduino IDE will appear as:



**FIG 2.4: ARDUINO SETUP**

**Program code in Arduino:**

#include <dht11.h> #define DHTPIN 8 dht11 DHT11;

int x; int t;

void setup() { Serial.begin(115200);

}

void loop() {

x = DHT11.read(DHTPIN);

t = millis();

String data = String(DHT11.temperature) + "," + String(DHT11.humidity); Serial.println(data);

delay(1000);

}

\*After upload this program open the serial monitor and select the baud rate it shows the output.

**Xampp server:**

XAMPP is an abbreviation where X stands for Cross-Platform, A stands for Apache, M stands for MYSQL, and the Ps stand for PHP and Perl, respectively. It is an open-source package of web solutions that includes Apache distribution for many servers and command- line executables along with modules such as Apache server, MariaDB, PHP, and Perl. XAMPP helps a local host or server to test its website and clients via computers and laptops before releasing it to the main server. It is a platform that furnishes a suitable environment to test and verify the working of projects based on Apache, Perl, MySQL database, and PHP through the system of the host itself. Among these technologies, Perl is a programming language used for web development, PHP is a backend scripting language, and MariaDB is the most vividly used database developed by MySQL. The detailed description of these components is given below. XAMPP is one of the widely used cross-platform web servers, which helps developers to create and test their programs on a local webserver. It was developed by the Apache Friends, and its native source code can be revised or modified by the audience. It consists of Apache HTTP Server, MariaDB, and interpreters for the different programming languages like PHP and Perl.

**PROCEDURE:**

\*Open the Xampp application, turn on the Apache and MySQL then click Admin.

\*Enter the Database name (it same as in python code)

\*Enter the Table name and select the no. of table then click create.

\*Enter the name of the parameter you will need.

\*After entering all the parameters click GO and the Xampp file is created.

**PYTHON:**

Install the python depends on your windows and install the

\*pip install serial

\*pip install pyserial on command prompt Python Program for DHT11:

import serial

import MySQL.connector

def insert\_data(mydata):

mydb = mysql.connector.connect( host="localhost",

user="root", password="", database="mq6"

)

my\_arr = mydata.split(',')

mycursor = mydb.cursor()

sql='INSERT INTO `sensor\_data`(`gasData`) VALUES ('+my\_arr[0]+')' mycursor.execute(sql)

mydb.commit() print("insertion success")

while True:

myser = serial.Serial("COM3" ,115200,

parity=serial.PARITY\_NONE, stopbits=serial.STOPBITS\_ONE, bytesize=serial.EIGHTBITS)

line = (myser.readline()) data = line.decode('utf-8') print(data) insert\_data(data) myser.close()

\*Run this code after creating your Xampp file…...

**Grafana:**

Grafana is an open-source interactive data-visualisation platform, developed by Grafana Labs, which allows users to see their data via charts and graphs that are unified into one dashboard (or multiple dashboards!) for easier interpretation and understanding. You can also query and set alerts on your information and metrics from wherever that information

is stored, whether that’s traditional server environments, Kubernetes clusters, or various cloud services, etc.

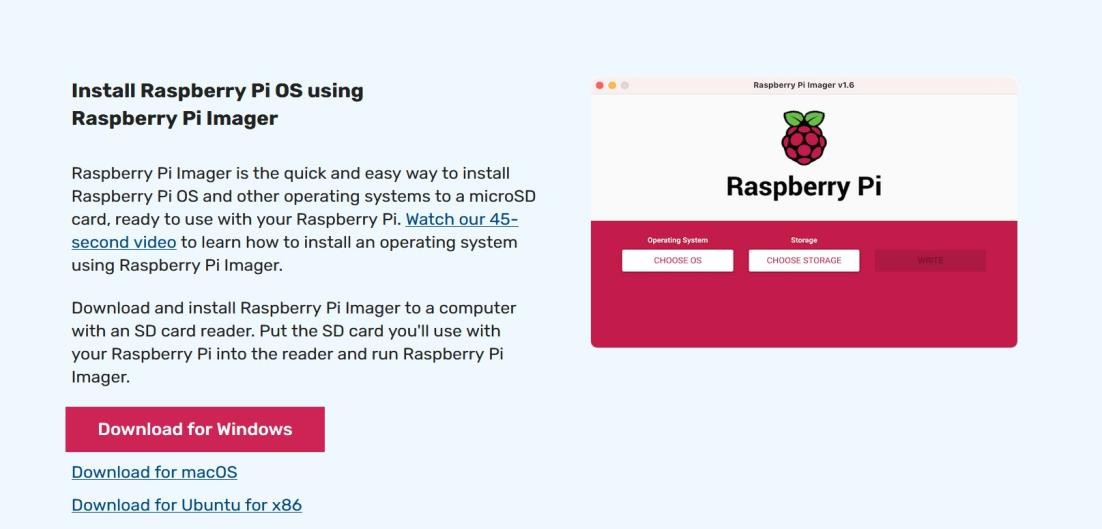
You’re then more easily able to analyse the data, identify trends and inconsistencies, and ultimately make your processes more efficient. Grafana was built on open principles and the belief that data should be accessible throughout an organisation, not just to a small handful of people. This fosters a culture where data can be easily found and used by anyone who needs it, empowering teams to be more open, innovative, and collaborative.

Grafana’s dashboards give a new level of meaning to data collected from various sources. Those dashboards can then be shared with other team members and other teams, allowing collaboration and more extensive exploration of the data and its implications. Build dashboards specifically for you and your team and customise your panels to create the visualisations you want, using advanced querying and transformation capabilities.

**PROCEDURE:**

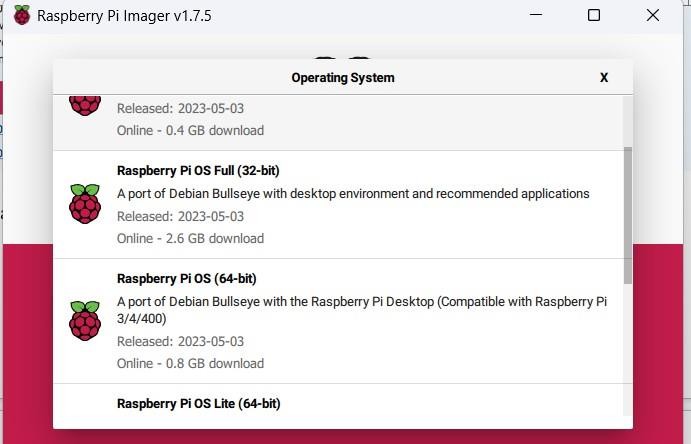
* search “localhost:3000” in chrome
* Open the Grafana application,
* Open new data source and select the MySQL
* Enter the database, host name, username
* Click save and test, give build new dashboard
* In new dashboard, select MySQL and select the temperature and humidity and timestamp
  1. **INSTALLATION OF RASPBERRY PI**

Search raspberry pi imager in browser



**FIG 2.1.1 RASPBERRY PI SETUP**

Download the software according to system requirement and install it.



#### Fig 2.1.2: RASPBERRY PI OS

Choose an OS with desktop environment so that viewing it becomes easier. Choose the settings options below to get the advanced options:

* + 1. Enable SSH.
    2. Enter password.
    3. Enter host name.

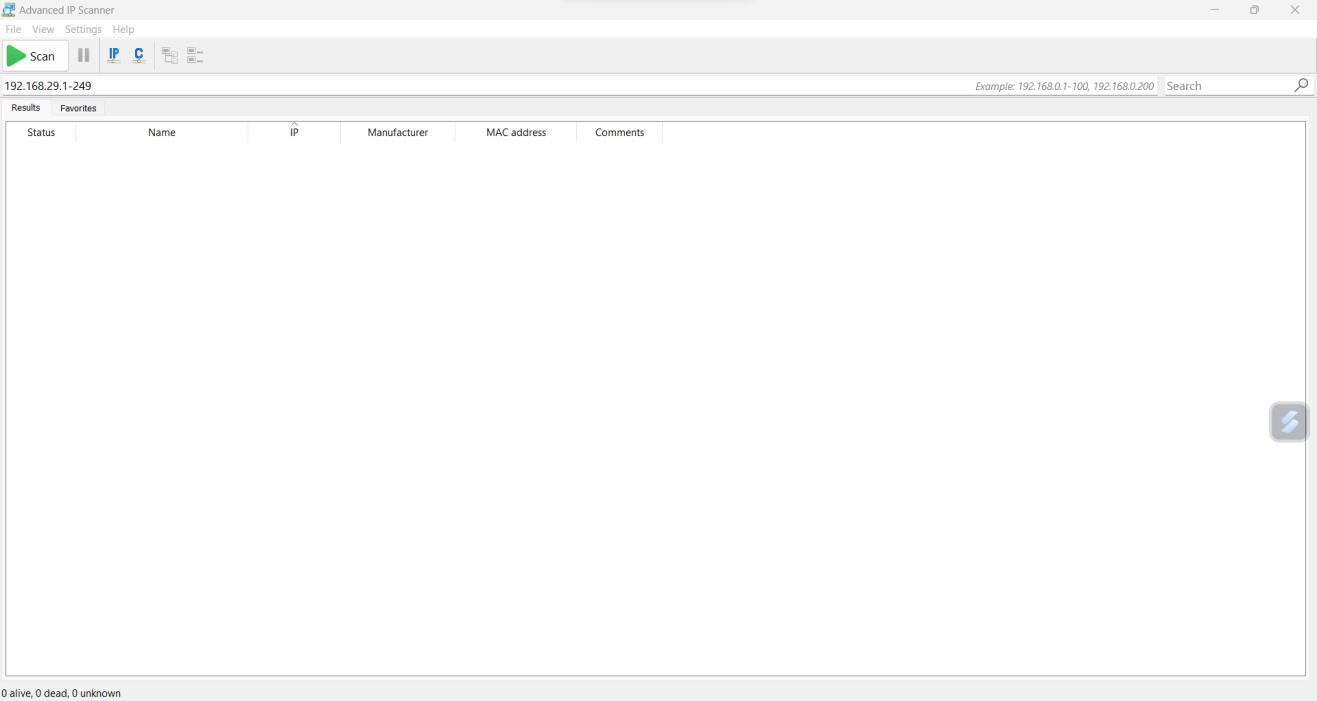
Choose the storage where the SD card has been fitted in the system and click write data. This writes all the data to SD card.

* 1. **ADVANCED IP SCANNER**

Advanced IP scanner is used to find the IP address of the raspberry we are using. It scans all the devices connected to a network.

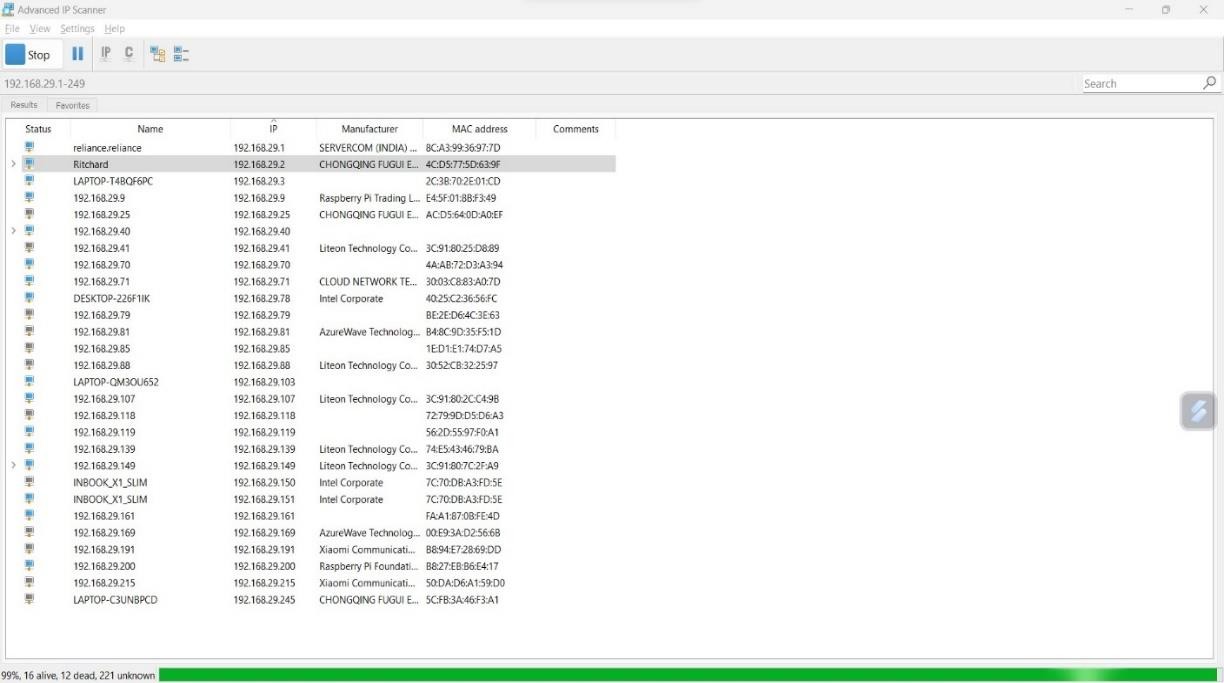
#### Steps:

Install the application from the browser and open it



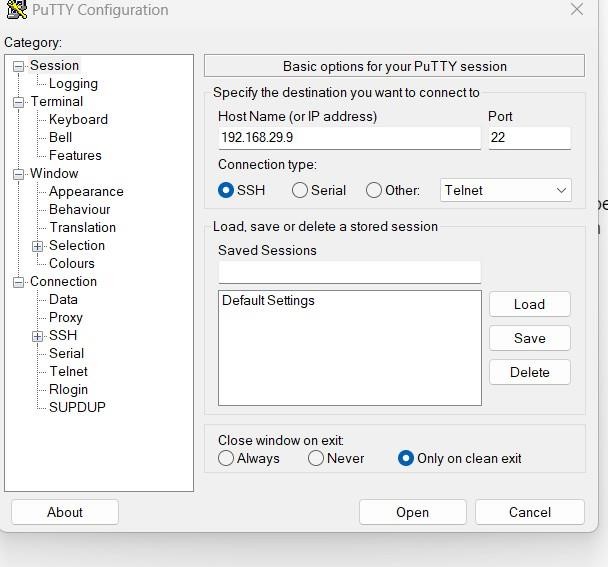
#### Fig 2.2.1: ADVANCE IP SETUP

Enter the IP address range 192.168.29.1-247 and enter scan. The list of all IP address connected to the particular network will list down below. In the select and copy the IP address of our raspberry pi. Find your IP address by using command prompt by entering IP config.



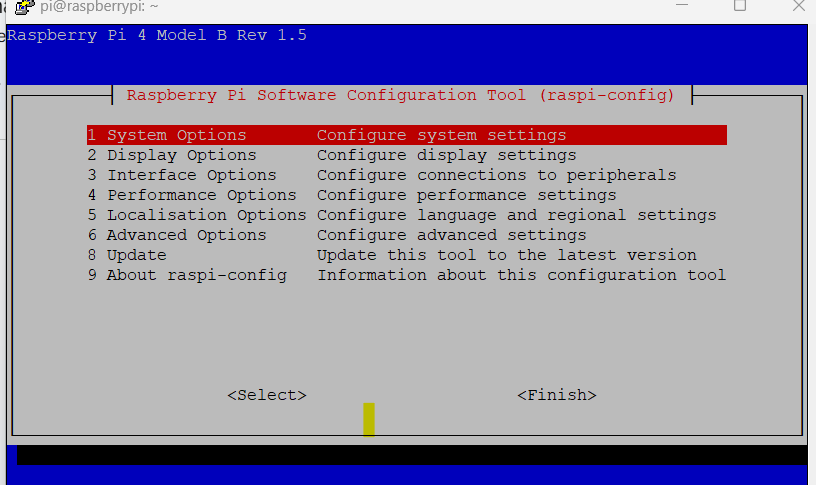
**FIG 2.2.2: IP ADDRESS**

* 1. **PUTTY**

Putty is a free open-source application. It used file transfer protocol. PUTTY can connect to a remote machine through SSH. SSH (secure shell) is a protocol that allows a secure connection. In this way, a PC can securely send and receive data from a remote.

**FIG 2.3.1 PUTTY SETUP**

Install the application from the web browser and open it. After that enter the host name enter the IP address of our raspberry pi obtain from the “Advanced IP Scanner” app.

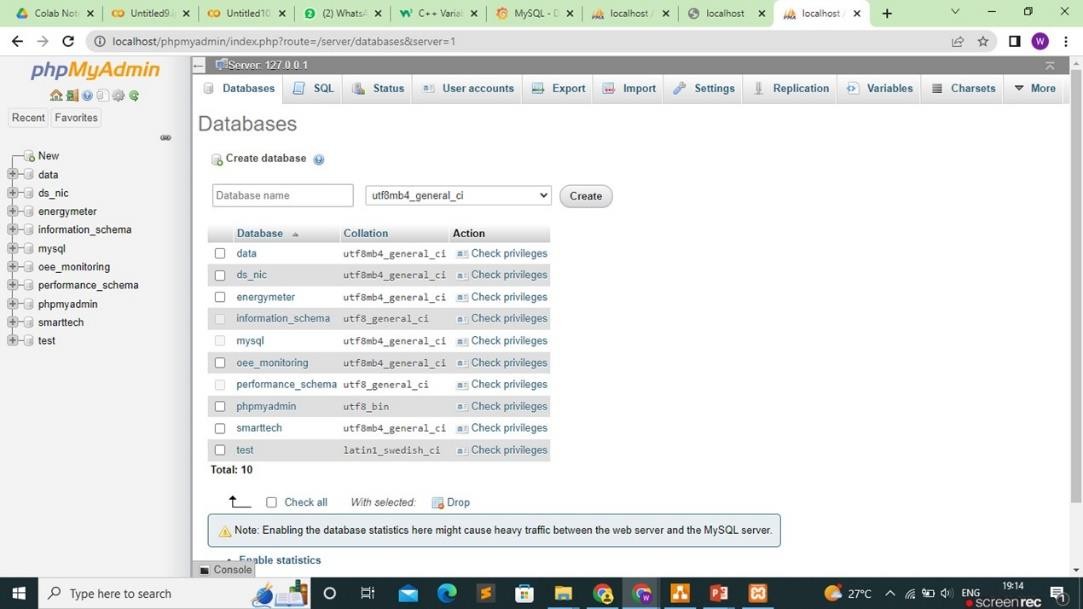


**FIG 2.3.2: PUTTY SCANNING**

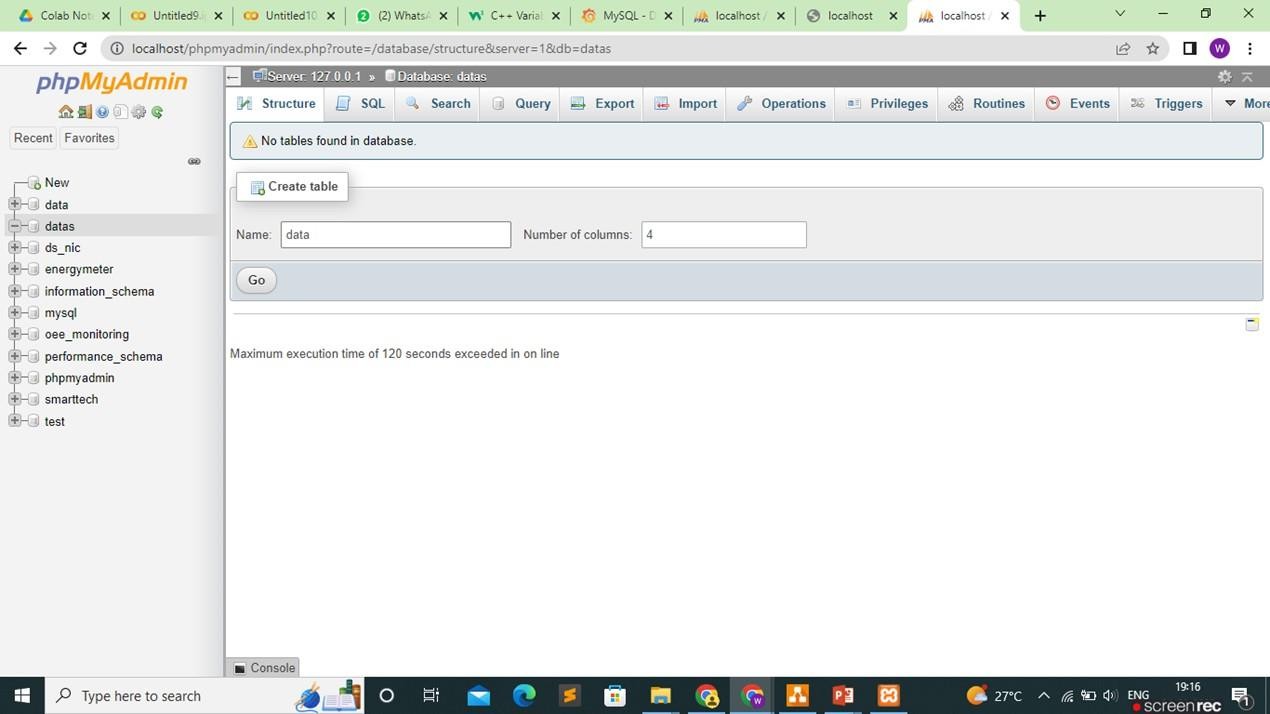
**DATABASE:**

A database is a structured collection of data where the data can be modified and stored in way which is suitable for visualization, retrieval, updating and analyzation. A database system consists of 2 main components: a DBMS and actual database. They are organized into table which are rows and columns. We have MySQL database to store data in the form of tables, mangrove DB database to store in file format and JSON or CSV database to store data in JSON format.

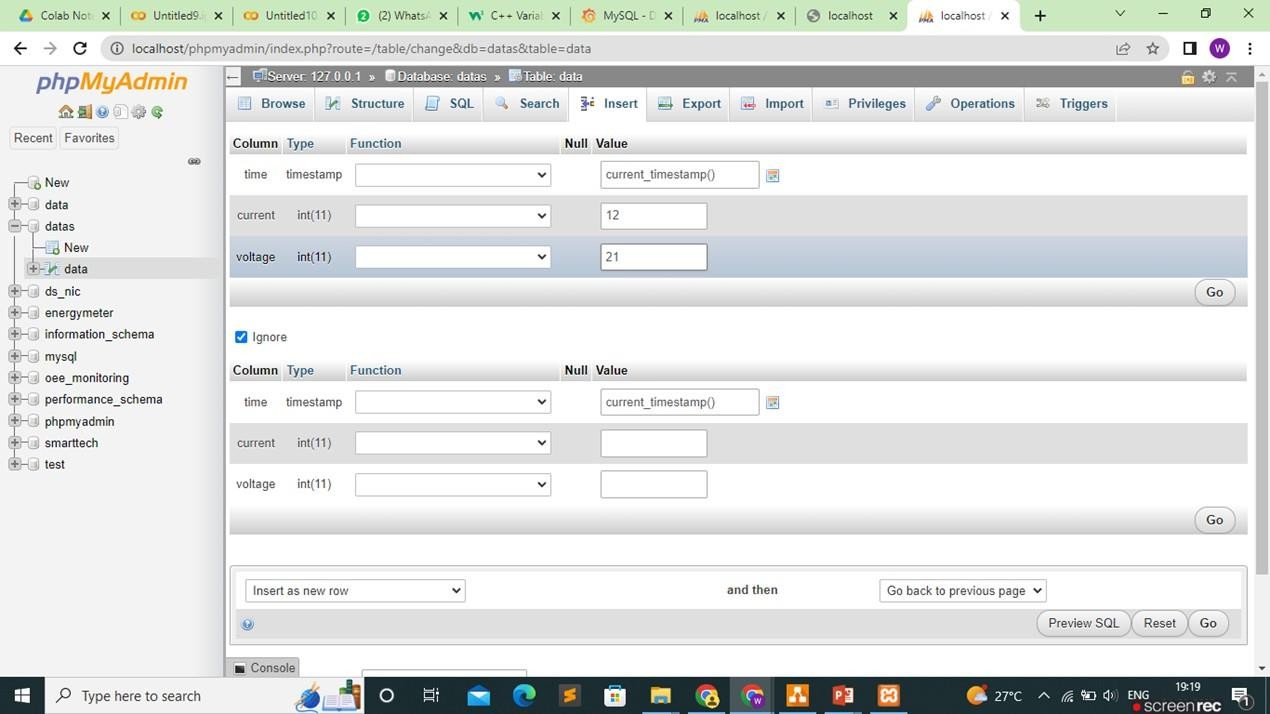
* 1. **Xampp**

****

**FIG 2.4.1: XAMPP DATA SHEET**

****

**FIG 2.4.2: XAMPP TABLE**



**FIG 2.4.3: XAMPP OUTPUT**

**PYTHON IDLE:**

#### Installation of python idle:

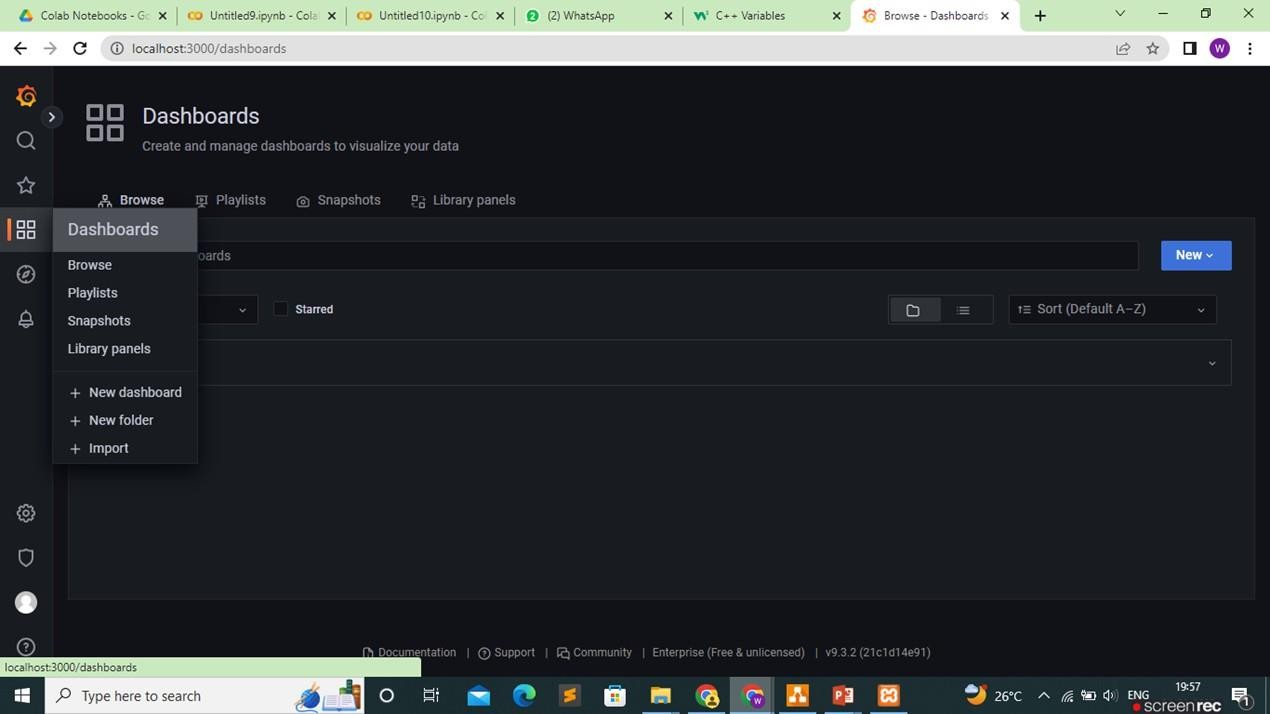
* + 1. Install the python idle software
    2. Download the necessary extensions to execute the program to sync the database.
    3. Execute the program for syncing purpose.

**GRAFANA:**

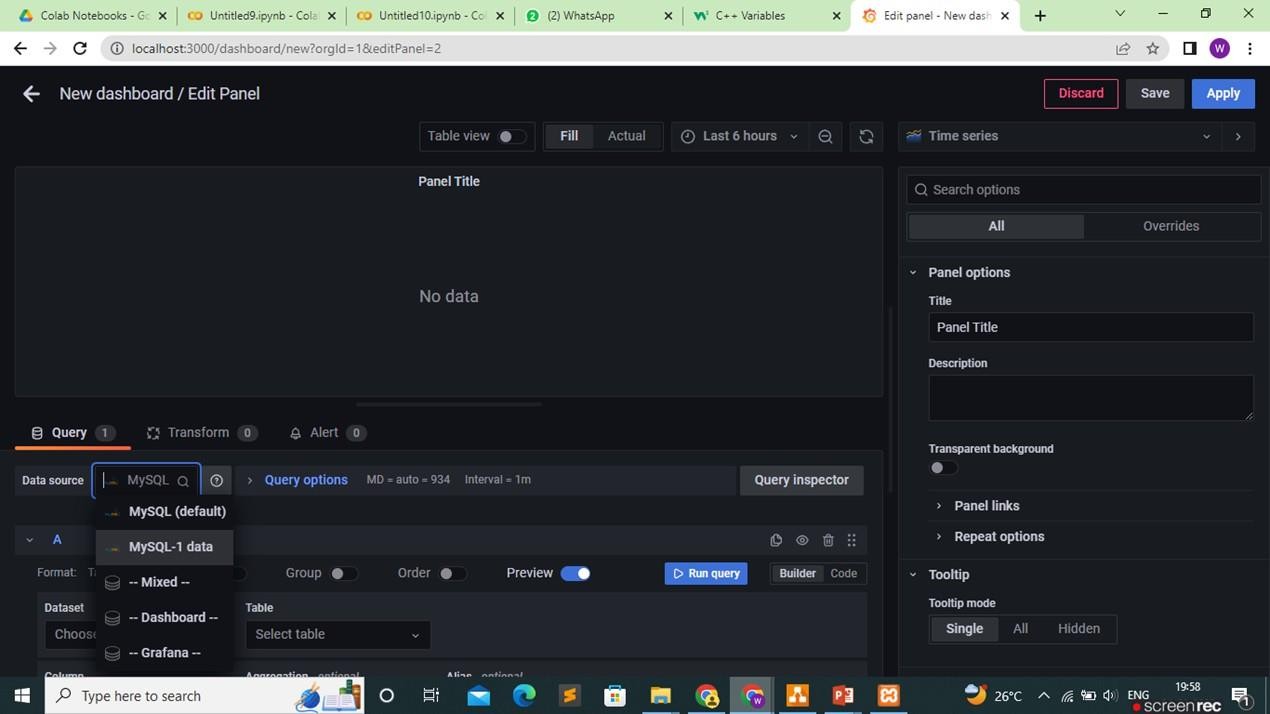
Grafana is an online open-source data visualization tool used to visualize data from various source data visualization tool used to visualize data from various source in a graphical format. We Grafana here to graphically represent the data from the smart meter which is stored in the MySQL database. By creating a new panel and importing the database and the table name and by specifying the value that we need to represent in our visualized chart we can visualize it.

**VISUALIZATION:**

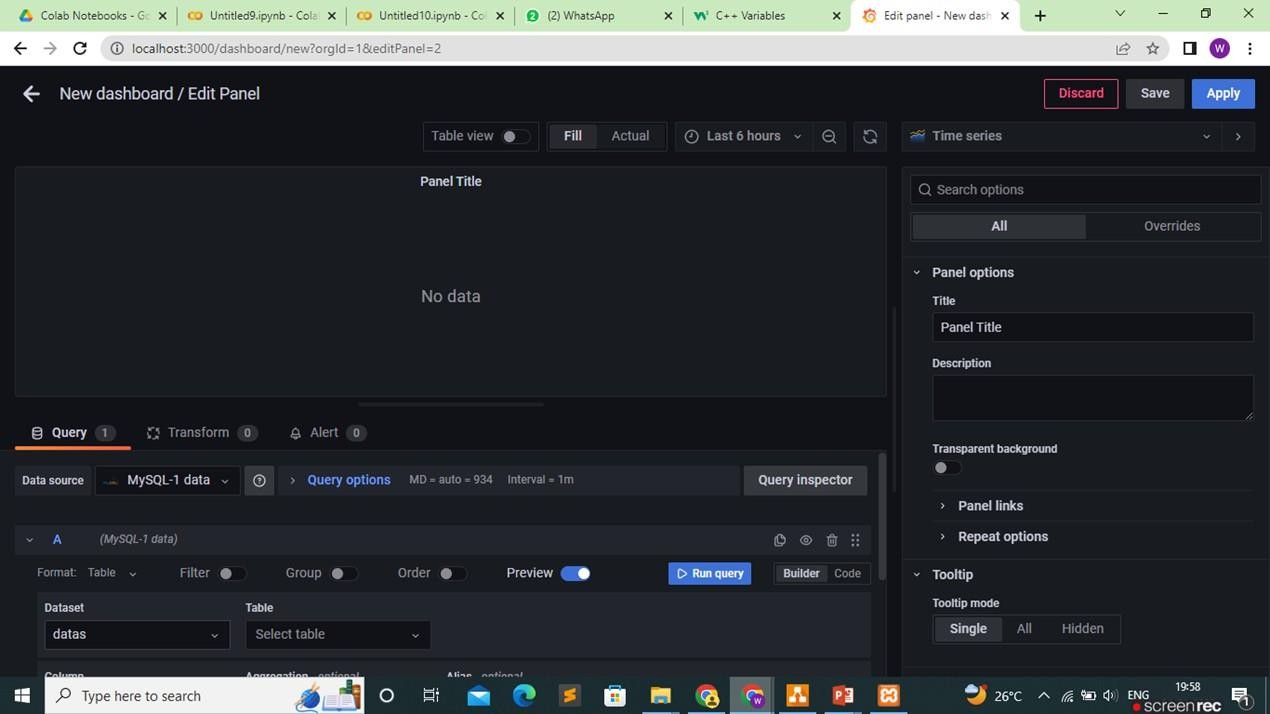
* + - * Open Grafana.
      * Open the new dashboard and select the new panel.
      * Select the database and table and columns.
      * All setups finished to click to run the query.
      * The panel click to zoom in data visual the data.



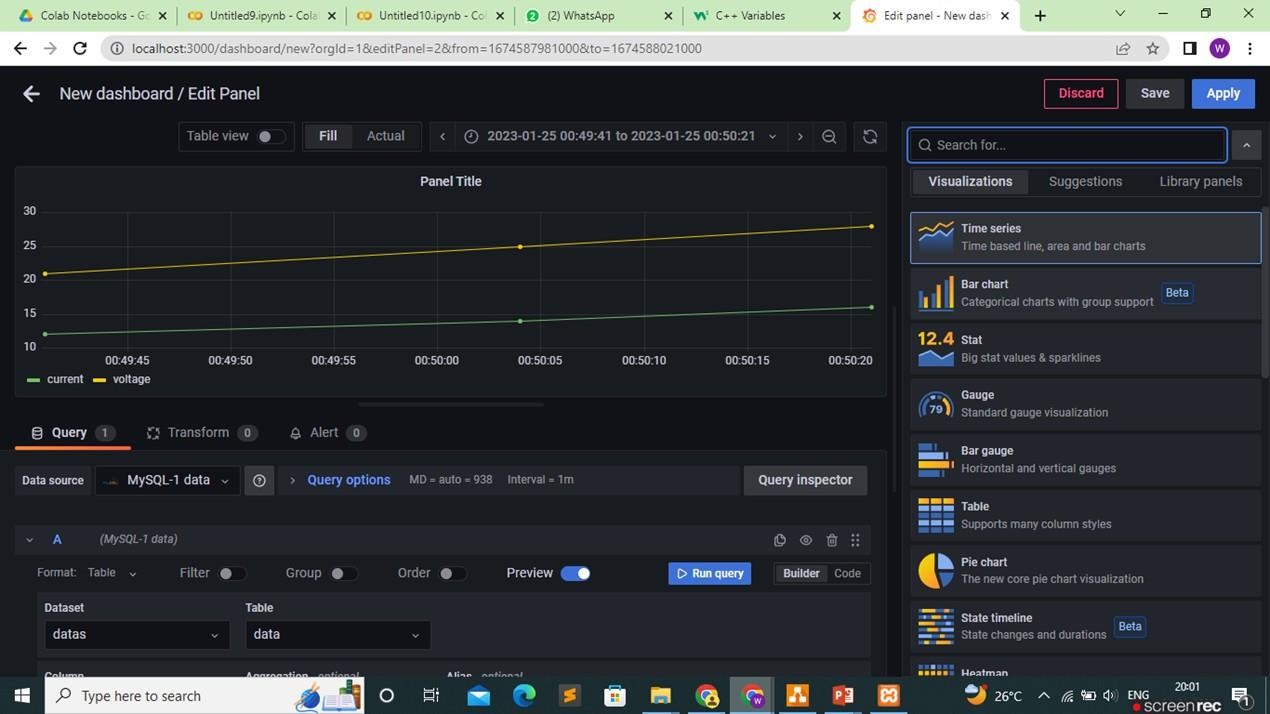
**FIG 2.4.4: GRAFANA DATA FEED**

****

**FIG 2.4.5: GRAFANA DATABASE SELECTION**



**FIG 2.4.6: GRAFANA DATABASE SELECTION-2**

****

**FIG 2.4.7: GRAFANA OUTPUT GRAPH**

# CHAPTER 3 CONCLUSION

This internship experience in the real time of Industrial Internet of Things (IoT) has been a profound journey of professional and personal growth. Throughout the duration of this internship, privilege of working alongside a dedicated team of experts in the field, allowing to gain invaluable insights into the intricate workings of IoT technologies and their applications within the industrial sector.

It has provided a unique opportunity to apply the knowledge acquired during academic pursuits in a real-world context. Good experience in participating in the design, implementation, and maintenance of IoT systems, witnessing firsthand their transformative potential in optimizing industrial operations, enhancing efficiency, and ensuring the highest levels of reliability and security.

# CHAPTER 4 FEEDBACK

#### What were your overall impressions of the internship?

It’s very useful, industrial oriented concept of Internet of Things were thought and hardware work with Arduino microcontroller was being taught.

1. **What did you learn and how did it help you develop your skills and knowledge?** The core concepts of the IoT and the hardware implementation was taught. It helps improve in both hardware and software in the domain of IoT.

#### What were your biggest task and how did you overcome that?

The hackathons were the biggest task and challenge to face and complete the program. Most of the time was spent in the learning process but still the hackathon part was the craziest part of the internship.

#### How did the internship prepare you for your future career?

Apart from the skill and core knowledge, to collaborate with the people who are new has given immense inspirations. Single heartedly the hackathons were solved. This will help to connect and develop the network in our future.

#### What do you feel about your mentors in the internship and how the internship differs from your college procedure of learning?

The amount of knowledge and skills that gained from the mentor in the internship was good. This internship helped a lot to gain new experience and gave the clear-cut idea for the future career.