GOVERNMENT OF KARNATAKA

MAHARANI CLUSTER UNIVERSITY

SHESHADRI ROAD, BANGALORE – 560001

PROJECT REPORT ON

“REAL TIME GRAPH USING MATPLOTLIB”

Project report submitted in partial fulfilment for award of

BACHELORE OF COMPUTER APPLICATION

III Year VIth Semester BCA

Submitted by

PAAVANA M RAO (20US1084)

SUSHMITHA A(20US1134)

UNDER THE GUDIANCE OF

Dr Nagamani HS

Department of computer science

2022-2023

CERTIFICATE

This is to certify that the project entitled **“GENERATING REAL TIME GRAPH USING MATPLOTLIB”** submitted to Maharani Cluster University in Partial Fulfilment for the degree of bachelor of computer application in computer science is a bona fide original work carried out by **PAAVANA M (20US1084), SUSHMITHA A(20US1134)** under the guidance and Supervision during the year 2022-2023.

The project report as it satisfies in the academic requirement in respect of project work prescribed for BCA.

Project Guide	**Head of Department**
Dr Nagamani HS	**Prof Ashok BS**

Examiners **Date: -**

1…………………...

2………………….

DECLARATION

We PAAVANA M RAO**(20US1084),** SUSHMITHA A**(20US1134)** do here by declare that the project entitled **“GENERATING REAL TIME GRAPH USING MATPLOTLIB”** bona fide work carried out by us under the guidance of **Dr Nagamani HS**

This project, as presented in this report, is our original work and has not been presented for any other university award. This project has been submitted as Fulfilment of requirements for the degree of Bachelor of computer application of Bangalore central university.

Place: Bengaluru **Signature of the student**
Date:

ACKNOWLEDGEMENT

I take this opportunity to express our deep sense of gratitude to our honour our Principal Dr. C. USHA DEVI for their sincere co-operation and providing us facility to carry out project

My whole hearted admiration and deep sense of gratitude to our HOD of Computer Science Dr Ashok SB of Maharani Cluster university for his inspiration, valuable guidance encouragement, suggestions and overall help throughout for this successful completion of our project.

We take this opportunity to express our deepest sincere gratitude to our internal Project guide
I am thankful to our beloved guide **Dr Nagamani HS** of Computer Science who has been very kind and supportive during course of our project.

I would also like to express sincere and humble gratitude to our respectful lecturers and non-teaching staff of our Computer Science department.

I thank our beloved parents and all those who have directly or indirectly helped me.

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1 Introduction to Internet of Thing

IoT stands for Internet of Things. It refers to the interconnectedness of physical devices, such as appliances and vehicles, that are embedded with software, sensors, and connectivity which enables these objects to connect and exchange data. This technology allows for the collection and sharing of data from a vast network of devices, creating opportunities for more efficient and automated systems.

Internet of Things (IoT) is the networking of physical objects that contain electronics embedded within their architecture in order to communicate and sense interactions amongst each other or with respect to the external environment. In the upcoming years, IoT-based technology will offer advanced levels of services and practically change the way people lead their daily lives. Advancements in medicine, power, gene therapies, agriculture, smart cities, and smart homes are just a very few of the categorical examples where IoT is strongly established.

The reason IoT has become so huge depends partly on two things: Moore's law and Koomey's law. Moore's law states that the number of transistors on a chip doubles approximately every two years. This has enabled people to develop more powerful computers on the same sized chip. Intel, a well-known semiconductor chip maker had during 1971, 2300 transistors on a processor and by 2012 their current processors contained 1.4 billion transistors. This is an increase of approximately 610 000 % and it is expect that this trend will continue. Koomey's law explains that the number of computations per kilowatt-hour roughly doubles every one and a half years

Interconnecting the physical world with the virtual world and applying this concept to all things opens up new possibilities in the sense of being able to at any time access anything from any place.

Providing new possibilities will also generate new threats, security risks, and expose vulnerabilities in the unexplored world of interconnected everything. “Things” in the physical world are objects that physically exist and from the perspective of IoT we are able to sense, operate, and connect to these things, while in the virtual world “things” are objects that can be stored, accessed, and processed. IoT involves sensors in order to collect information. Sensors are already being used in daily life, however most people may not realise it. Smartphones contain different kind of sensors, such as accelerometers, cameras, and GPS receivers. Built-in sensors are nothing new in today’s society. Kevin Ashton said that IoT is already happening, but we might not see it compared to Smartphones which can both be seen and touched. RFID is such an IoT-technology that exists but is not necessarily seen; so the development of IoT might progress a long way before it is visible for everyone.

1.2 Characteristics of IoT

1. Intelligence

IoT comes with the combination of algorithms and computation, software & hardware that makes it smart. Ambient intelligence in IoT enhances its capabilities which facilitate the things to respond in an intelligent way to a particular situation and supports them in carrying out specific tasks. In spite of all the popularity of smart technologies, intelligence in IoT is only concerned as means of interaction between devices, while user and device interaction is achieved by standard input methods and graphical user interface.

2. Connectivity

Connectivity empowers Internet of Things by bringing together everyday objects. Connectivity of these objects is pivotal because simple object level interactions contribute towards collective intelligence in IoT network. It enables network accessibility and compatibility in the things. With this connectivity, new market opportunities for Internet of things can be created by the networking of smart things and applications.

3. Dynamic Nature

The primary activity of Internet of Things is to collect data from its environment, this is achieved with the dynamic changes that take place around the devices. The state of these devices changes dynamically, example sleeping and waking up, connected and/or disconnected as well as the context of devices including temperature, location and speed. In addition to the state of the device, the number of devices also changes dynamically with a person, place and time.

4. Enormous scale

The number of devices that need to be managed and that communicate with each other will be much larger than the devices connected to the current Internet. The management of data generated from these devices and their interpretation for application purposes becomes more critical. Gartner (2015) confirms the enormous scale of IoT in the estimated report where it stated that 5.5 million new things will get connected every day and 6.4 billion connected things will be in use worldwide in 2016, which is up by 30 percent from 2015. The report also forecasts that the number of connected devices will reach 20.8 billion by 2020.

5. Sensing

IoT wouldn't be possible without sensors which will detect or measure any changes in the environment to generate data that can report on their status or even interact with the environment. Sensing technologies provide the means to create capabilities that reflect a true awareness of the physical world and the people in it. The sensing information is simply the analogue input from the physical world, but it can provide the rich understanding of our complex world.

6. Heterogeneity

Heterogeneity in Internet of Things as one of the key characteristics. Devices in IoT are based on different hardware platforms and networks and can interact with other devices or service platforms through different networks. IoT architecture should support direct network connectivity between heterogeneous networks. The key design requirements for heterogeneous things and their environments in IoT are scalabilities, modularity, extensibility and interoperability.

7. Security

IoT devices are naturally vulnerable to security threats. As we gain efficiencies, novel experiences, and other benefits from the IoT, it would be a mistake to forget about security concerns associated with it. There is a high level of transparency and privacy issues with IoT. It is important to secure the endpoints, the networks, and the data that is transferred across all of it means creating a security paradigm.

There are a wide variety of technologies that are associated with Internet of Things that facilitate in its successful functioning. IoT technologies possess the above-mentioned characteristics which create value and support human activities; they further enhance the capabilities of the IoT network by mutual cooperation and becoming the part of the total system.

1. 3 Applications of IoT

1. Smart Home

Smart Home clearly stands out, ranking as highest Internet of Things application on all measured channels. More than 60,000 people currently search for the term "Smart Home" each month. This is not a surprise. The IoT Analytics company database for Smart Home includes 256 companies and startups. More companies are active in smart home than any other application in the field of IoT. The total amount of funding for Smart Home startups currently exceeds $2.5bn. This list includes prominent startup names such as Nest or AlertMe as well as a number of multinational corporations like Philips, Haier, or Belkin.

2. Wearables

Wearables remains a hot topic too. As consumers await the release of Apple's new smart watch in April 2015, there are plenty of other wearable innovations to be excited about: like the Sony Smart B Trainer, the Moy gesture control, or Look See bracelet. Of all the IoT start-ups, wearables maker Jawbone is probably the one with the biggest funding to date. It stands at more than half a billion dollars!

3. Smart City

Smart city spans a wide variety of use cases, from traffic management to water distribution, to waste management, urban security and environmental monitoring. Its popularity is fueled by the fact that many Smart City solutions promise to alleviate real pains of people living in cities these days. IoT solutions in the area of Smart City solve traffic congestion problems, reduce noise and pollution and help make cities safer.

4. Smart grids

Smart grids is a special one. A future smart grid promises to use information about the behaviors of electricity suppliers and consumers in an automated fashion to improve the efficiency, reliability, and economics of electricity. 41,000 monthly Google searches highlights the concept's popularity. However, the lack of tweets (Just 100 per month) shows that people don't have much to say about it.

5. Industrial internet

The industrial internet is also one of the special Internet of Things applications. While many market researches such as Gartner or Cisco see the industrial internet as the IoT concept with the highest overall potential, its popularity currently doesn't reach the masses like smart home or wearables do. The industrial internet however has a lot going for it. The industrial internet gets the biggest push of people on Twitter (~1,700 tweets per month) compared to other non-consumer-oriented IoT concepts.

6. Connected car

The connected car is coming up slowly. Owing to the fact that the development cycles in the automotive industry typically take 2-4 years, we haven't seen much buzz around the connected car yet. But it seems we are getting there. Most large auto makers as well as some brave startups are working on connected car solutions. And if the BMWs and Fords of this world don't present the next generation internet connected car soon, other well-known giants will: Google, Microsoft, and Apple have all announced connected car platforms.

7. Connected Health (Digital health/Telehealth/Telemedicine)

Connected health remains the sleeping giant of the Internet of Things applications. The concept of a connected health care system and smart medical devices bears enormous potential (see our analysis of market segments), not just for companies also for the well-being of people in general. Yet, Connected Health has not reached the masses yet. Prominent use cases and large-scale startup successes are still to be seen. Might 2015 bring the breakthrough?

8. Smart Retail

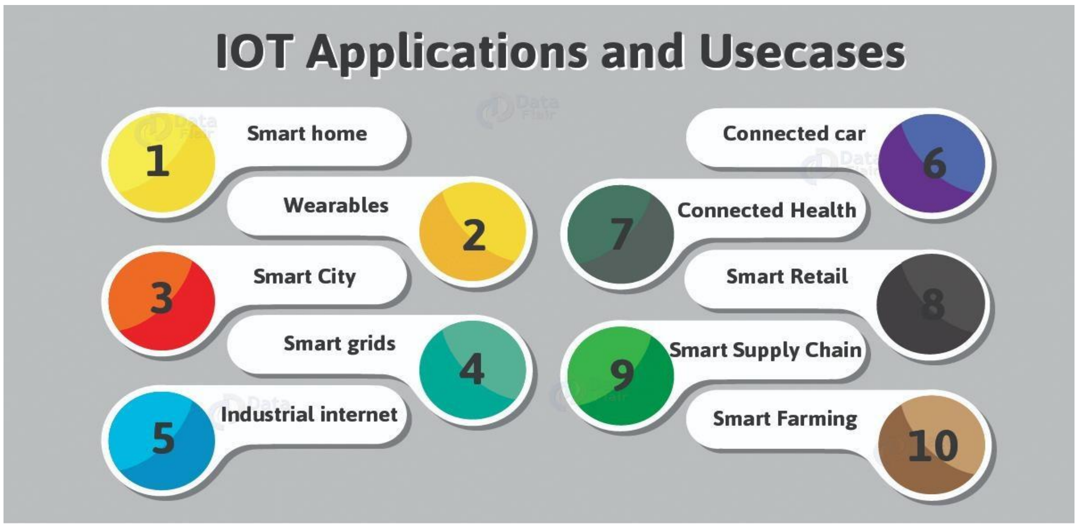
Proximity-based advertising as a subset of smart retail is starting to take off. But the popularity ranking shows that it is still a niche segment. One LinkedIn post per month is nothing compared to 430 for smart home.

9. Smart Supply Chain

Supply chains have been getting smarter for some years already. Solutions for tracking goods while they are on the road, or getting suppliers to exchange inventory information have been on the market for years. So while it is perfectly logic that the topic will get a new push with the Internet of Things, it seems that so far its popularity

10. Smart Farming

Smart farming is an often overlooked business-case for the internet of Things because it does not really fit into the well-known categories such as health, mobility, or industrial. However, due to the remoteness of farming operations and the large number of livestock that could be monitored the Internet of Things could revolutionize the way farmers work. But this idea has not yet reached large-scale



attention. Nevertheless, one of the Internet of Things applications that should not be underestimated. Smart farming will become the important application field in the predominantly agricultural-product exporting countries.

1.4 Technical Features

Arduino IDE:

Arduino is a physical computing platform based on a simple I/O board and a development environment that implements the Processing/Wiring language. Arduino can be used to develop stand-alone interactive objects or can be connected to software running on a computer (e.g., Macromedia Flash, Processing, Max/MSP, Pure Data, Supercollider). The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino and Genuine hardware to upload programs and communicate with them.

Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension. ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors.

The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom righthand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

1.6 Literature Survey

Real-time systems are often implemented by a number of concurrent tasks sharing

hardware resources, in particular the execution processors. The designer of such systems needs to construct workload models characterizing the resource requirements

of the tasks. With a formal description of the workload, a resource scheduler may

be designed and analysed. The fundamental analysis problem to solve in the design

process is to check (and thus to guarantee) the schedulability of the workload, i.e., whether the timing constraints on the workload can be met with the given scheduler.

In addition, the workload model may also be used to optimize the resource utilizations well as the average system performance

1.7 Problem Statement and Scope of the Project

In data visualization, real-time plotting can be a powerful tool to analyse data as it streams into the acquisition system. Whether temperature data, audio data, stock market data, or even social media data - it is often advantageous to monitor data in real-time to ensure that instrumentation and algorithms are functioning properly.

Real time data is used primarily to drive real time analytics and also can be used for navigating or tracking. By studying these plots we can make better decisions. Real time data are compared using previous data.

This is a technique for suggesting relevant or personalized content to users based on their interactions with other users or items. It can be used for enhancing user engagement, retention, or satisfaction in online platforms, such as news, entertainment, or education. A tool for this purpose is Graph Jet, an in-memory graph processing engine that maintains a real-time bipartite interaction graph between users and tweets.

1.8 Methodology

Identify requirements: Define the problem, the objectives, the target users and the expected outcomes of the project.

- Define the problem and the requirements. What kind of data do you want to visualize? What kind of chart do you want to use? How often do you want to update the graph? What are the performance and security constraints?

- **Choose a suitable data structure and algorithm.** Depending on the type and size of your data, you may need to use different data structures, such as arrays, lists, stacks, queues, trees, graphs, etc. You may also need to use different algorithms to update and render the graphs in real time, such as line charts, bar charts, pie charts, radar charts, etc.

- **Implement the data processing and graph generation** logic. You may need to use a programming language or a framework that supports real time data processing and graph generation. For example, you can use Python with Flask and Chart.js, Java with Swing and Plotly , or JavaScript with Google Chart API .

- **Test and debug the system**. You may need to test your system for functionality, accuracy, responsiveness, scalability, security, etc. You may also need to debug any errors or exceptions that may occur during the data processing and graph generation.

- **Deploy and monitor the system**. You may need to deploy your system on a web server or a cloud platform that can handle real time traffic. You may also need to monitor your system for performance issues or user feedback.

Chapter 2

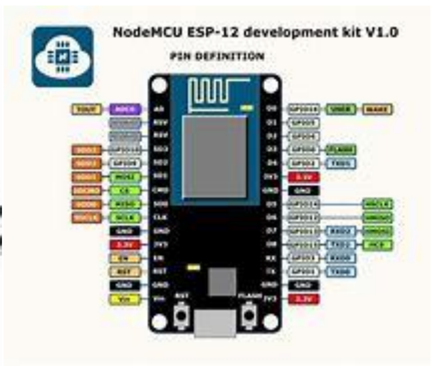
Hardware and Software Requirement Specification

2.1 Introduction

The design consists more on actual planning of software and hardware part than the code to be created. A number of software and hardware implementation techniques were used to design and develop the system.

2.2 Hardware Requirements

Node MCU**:NodeMCU** is an open source firmware for which open source prototyping board designs are available. The name "NodeMCU" combines "node" and "MCU" (micro-controller unit).



BREAD BOARD: A breadboard, solderless Breadboard, or protoboard is a construction Base used to build semi-permanent Prototypes of electronic circuits.

Jumper wires: wires that have connector pins at each end, allowing them to be used to connect two points to each other without soldering

Ultrasonic Sensor

This sensor is a high-performance ultrasonic range finder. It is compact and measures an amazingly wide range from 2cm to 4m.

This ranger is a perfect for any robotic application, or any other projects requiring accurate ranging information. This sensor can be connected directly to the digital I/O lines of your microcontroller and distance can be measured in time required for travelling of sound signal using simple formula as below. The module works on 5VDC input and also gives an output signal directly for detection of any obstacle up to 4M. As soon as the signals are transmitted the “Echo” pin goes to high level and remains in high level until the same sound waves are received by the receiver. If the received sound waves are same as what the same sensor transmitted then the Echo pin goes to low level. If no object is detected within 5M after 30ms the Echo signal will automatically go to low level.

2.3 Software requirements

Creating real-time graphs typically requires specialized software and libraries that can handle continuous data updates and display them in a dynamic manner. The specific software and requirements you need can vary depending on your project and the programming language you prefer.

- Windows operating system: Windows 10 is used in current and latest version of windows. Making UML diagrams, canvases using Photoshop (version: CC16), E draw, and paint as well. Power point presentation, sensor coning in python IDE, connecting sensors and software for trail we are using android mobile application

- Python scripting language**:** Python: Python is a basic language of processors and controllers such as NodeMCU's, Raspberry PI, etc. All the coding related to sensor would be done in this language. This language is very easy to implement because it's coding look like basic C language. Currently we are using 3.7 version of python IDE.

IDLE" stands for "Integrated Development and Learning Environment." It is the default interactive development environment that comes bundled with Python. IDLE provides a simple and convenient way to write, run, and test Python code.

"IDLE" stands for "Integrated Development and Learning Environment." It is the default interactive development environment that comes bundled with Python. IDLE provides a simple and convenient way to write, run, and test Python code.

- Here are the basic steps to use Python IDLE:
- Open IDLE:
  - On Windows, you can typically find IDLE in the Start menu under the Python folder.
  - On macOS, you can open it from the Applications folder.
  - On Linux, you can open it from the command line by typing `idle` and pressing Enter.
- Interactive Shell:
  - When IDLE opens, you'll see an interactive shell where you can enter Python code directly and see the results immediately.
- Create a New Python File:

  - You can also create and edit Python scripts by going to `File > New File`. This opens a new code editor window where you can write your code.
- Write and Run Code:
  - In the code editor, you can write your Python code. To execute the code, you can either:
  - Press `F5` or go to `Run > Run Module` to run the entire script.
  - Select a specific code block and press `F9` or go to `Run > Run Selection` to run only the selected code.
- Save and Load Scripts:
  - You can save your Python scripts using `File > Save` or `File > Save As`. To open an existing script, use `File > Open`.
- Interactive Debugger:
  - IDLE also has an integrated debugger that can help you find and fix errors in your code. You can set breakpoints and step through your code line by line.
- 7. Auto-Completion and Syntax Highlighting: - IDLE provides features like auto-completion and syntax highlighting to make coding more convenient.
- Options and Preferences:
  - You can customize various settings and preferences in IDLE through the `Options` menu.
- Help and Documentation:
  - IDLE provides access to Python's built-in documentation. You can use the `Help` menu to access Python's documentation or to check for updates.
- Exit IDLE:
  - To exit IDLE, simply close the IDLE window.

Python IDLE is a good choice for beginners and for quickly testing small code snippets. However, for larger and more complex projects, you might want to consider using more feature-rich integrated development environments (IDEs) like PyCharm, Visual Studio Code, or Jupyter Notebook, which offer advanced features and better project management capabilities.

- Arduino IDE: Arduino is a physical computing platform based on a simple I/O board and a development environment that implements the Processing/Wiring language. Arduino can be used to develop stand-alone interactive objects or can be connected to software running on a computer

1. Code Editor: The Arduino IDE includes a text editor where you can write your Arduino sketches (programs). It supports standard C/C++ syntax with some Arduino-specific functions and libraries.

2. Library Manager: Arduino IDE comes with a Library Manager that allows you to easily add, update, and manage libraries used in your projects. Libraries provide pre-written code to extend the functionality of your Arduino projects.

3. Board Manager: You can select the type of Arduino board you are using from a list in the Arduino IDE. This setting determines the microcontroller and its specifications, including clock speed and available I/O pins.

4. Serial Monitor: The Serial Monitor is a tool within the IDE that allows you to send and receive serial data between your computer and the Arduino board. It's helpful for debugging and monitoring the behavior of your Arduino sketches.

5. Integrated Compiler: Arduino IDE includes a compiler (based on the AVR-GCC toolchain) that converts your Arduino code into machine code that can run on the Arduino board.

6. Upload Tool: You can upload your compiled code to the Arduino board directly from the IDE using a USB connection. The IDE handles the entire process of flashing the program onto the board.

7. Examples: Arduino IDE provides a range of example sketches that demonstrate various functions and features of Arduino boards and libraries. These examples serve as a valuable learning resource.

8. Auto-Formatting: The IDE can automatically format your code to improve readability by pressing `Ctrl+T` (or `Cmd+T` on macOS).

9. Debugging Tools: While the Arduino IDE doesn't offer advanced debugging capabilities, it provides basic features like printing debug information to the Serial Monitor and using built-in LEDs or external components for debugging.

10. Preferences: You can configure various preferences, such as code editor settings, compiler options, and additional hardware configurations.

11. Extensions and Plugins: Arduino IDE can be extended with plugins and extensions to add more functionality or support for additional hardware platforms.

12. Open Source: Arduino IDE is open-source software, which means that the community can contribute to its development and create custom versions or modifications.

13. Cross-Platform: The Arduino IDE is available for Windows, macOS, and Linux, making it accessible on various operating systems

The Arduino IDE is an excellent choice for beginners and hobbyists looking to get started with microcontroller programming and electronics projects. However, for more advanced development, some users prefer using alternative development environments such as PlatformIO with Visual Studio Code.

Chapter 3

Project Description

3.1 Introduction

Internet of things is a technology of the future that has already started to touch our homes. It uses a combination of hardware and software to enable control and management over appliances and devices.

By using the real time data we can predict the efficiency through given parameters (here distance) as we are the most flexible language which contain numerous packages. The main objective is for analysing the plots respectively within varied numerical values.

Which enables us to ask questions we haven't even considered asking before we had technologies optimized for providing these answers. However, graph databases still largely remain an untapped asset that can truly help the health care industry in dealing with real-time, saturated, and complex data.

3.2 Packages

Pip

pip is the package installer for Python. It is used to install, upgrade, and uninstall Python packages that are not included in the standard library You can use pip to install packages from the Python Package Index (PyPI) and other indexes. PyPI is a centralized and publicly accessible index of Python packages that includes frameworks, tools, and libraries to check whether pip is installed on your system or not, you can type the following command in the terminal: pip --version.

get-pip.py

The basic syntax of pip commands in the command prompt is: pip 'arguments'.

If you want to install a package using pip, you can use the command pip install package name. You can also specify a particular version of a package to install using pip install package name.

Matplotlib

Matplotlib is a powerful and very popular data visualization library in Python. Data visualization is an essential skill for all data analysts and Matplotlib is one of the most popular libraries for creating visualizations. Matplotlib is very flexible and customizable for creating plots. It does require a lot of code to make more basic plots with little customizations. When working in a setting where exploratory data analysis is the main goal, requiring many quickly drawn plots without as much emphasis on aesthetics.

Matplotlib is designed to work with NumPy arrays and pandas data frames. The library makes it easy to make graphs from tabular data

Use of Matplotlib:

Data Visualization: Matplotlib is widely used for visualizing data in various formats, such as line plots, scatter plots, bar charts, histograms, and more.

Exploratory Data Analysis: Matplotlib enables you to explore datasets, and identify patterns through interactive visualizations.

Cvzone

CVZone is a computer vision package that makes it easy to run image processing and AI functions. It uses OpenCV and Mediapipe libraries at its core. You can install the latest version of CVZone using pip. The package provides various modules such as Face Detection, Hand Tracking, and Pose Estimation. You can use these modules to perform various tasks such as detecting faces, tracking hands, and estimating poses in real-time video streams.

By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.

PySerial

This module encapsulates the access for the serial port. It provides back ends for Python running on Windows, OSX, Linux, BSD (possibly any POSIX compliant system) and Iron Python. The module named “serial” automatically selects the appropriate backend.

PySerial is a Python library that provides support for serial connections over a variety of different devices, including old-style serial ports, Bluetooth dongles, and infra-red ports. It also supports remote serial ports via RFC 2217. The module encapsulates the access for the serial port and provides backend for Python running on Windows, OSX, Linux, BSD (possibly any POSIX compliant system) and Iron Python. The module named “serial” automatically selects the appropriate backend. It is released under a free software license.

Arduino

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike. Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments.

Chapter 4

Requirement Analysis

4.1 Analysis Design

Analysis and design are two critical phases in the software development lifecycle, often referred to as the "Requirements Analysis" and "System Design" phases. These phases help ensure that a software project is well-planned and that the resulting system meets the desired objectives and quality standards. Let's explore each phase in more detail:

<u>Analysis Phase:</u>

<u>Purpose:</u> The analysis phase is the initial step in the software development process, where the focus is on understanding the user requirements and defining the requirements. The primary goal is to gather information, clarify objectives, and define what the software system needs to accomplish.

The analysis phase of project should result in three important deliverables;

- ✓ A business requirements report
- ✓ A conceptual system design plan
- ✓ High level strategy documents for the entire process

These documents will guide the rest of the project and ensure that all work supports the end goal of producing an application that fits the clients need.

4.2 Feasibility Study

A feasibility study is an assessment of the practicality of a project or system. A feasibility study aims to objectively and rationally uncover the strengths and weaknesses of an existing business

A well-designed feasibility study should provide a historical background of the business or project, a description of the product accounting statements, details of the operations and management, marketing research and policies, financial data, legal requirements and tax obligations

Three key considered in feasibility study

Time Feasibility: A time feasibility study will take into account the period in which the project is going to take up to its completion. A project will fail if it takes too long to be completed before it is useful. Typically, this means estimating how long the system will take to develop, and if it can be completed in a given time period using some methods like payback period. Time feasibility is a measure of how reasonable the project timetable

Financial visibility: financial viability can be judged on the following parameters:

- Total estimated cost of the project
- Financing of the project in terms of its capital structure, debt to equity ratio and promoter's share of total cost
- Existing investment by the promoter in any other business
- Projected cash flow and profitability

The financial viability of a project should provide the following information:

- Full details of the assets to be financed and how liquid those assets are.
- Rate of conversion to cash-liquidity (i.e., how easily the various assets can be converted to cash).
- Project's funding potential and repayment terms.
- Sensitivity in the repayments capability to the following factors:
    - Mild slowing of sales.
    - Acute reduction/slowing of sales.

- Small increase in cost.
- Large increase in cost.
- Adverse economic conditions.

Technical feasibility: This assessment is based on an outline design of system requirements, to determine whether the company has the technical expertise to handle completion of the project. When writing a feasibility report, the following should be taken to consideration:

- A brief description of the business to assess more possible factors which could affect the study
- The part of the business being examined
- The human and economic factor
- The possible solutions to the problem

At this level, the concern is whether the proposal is both *technically* and *legally* feasible (assuming moderate cost).

The technical feasibility assessment is focused on gaining an understanding of the present technical resources of the organization and their applicability to the expected needs of the proposed system. It is an evaluation of the hardware and software and how it meets the need of the proposed system

4.3 User Requirements:

User requirements are the needs, expectations, and preferences of the people who will use a system.

They are essential for designing and developing a system that meets the goals and solves the problems of the users.

It involves several steps and techniques that require collaboration, communication, and validation.

4.3.1 Understand The Context

The first step in defining user requirements is to understand the context of the system project. This means identifying the purpose, scope, and objectives of the system

The goal is to gain a clear and comprehensive picture of the problem domain, the user needs, and the expected outcomes of the system.

4.3.2 Define the Functional Requirements

The next step is to define the functional requirements of the system. These are the features and capabilities that the system must provide to enable the users to perform their tasks and achieve their goals.

4.3.3 Define the non-functional requirements

These are the quality attributes and constraints that affect the performance, usability, reliability, security, and maintainability of the system.

4.3.4 Define the user interface requirements

These are the aspects of the system that relate to the appearance, layout, navigation, and interaction of the user interface.

Validate the user requirement. This means checking that the user requirements are correct, complete, consistent, feasible, and testable.

Designing a user interface (UI) for plotting real-time graphs requires specific considerations to ensure that users can effectively monitor changing data in real-time. Below is a guideline for designing a UI for real-time graph plotting:

1. **Main Interface Layout**:

- Create a clean and uncluttered main interface. Real-time data graphs should be the central focus.

- Consider using a dark background with bright, contrasting colors for the graphs to enhance visibility.

2**. Graph Area**:

- Allocate a prominent portion of the interface to display the real-time graph.

- Use a responsive charting library or framework that can update and display data points in real-time.

3. Data Input:

- Include options for users to input or stream real-time data. This could be through file uploads, API endpoints, or hardware connections.

- Provide fields for users to specify data parameters and sources.

4. Graph Customization:

- Allow users to customize graph settings, such as scaling, axis labels, chart type (line, bar, scatter), and color schemes.

- Offer real-time customization options, such as changing the time window for data display.

5. Start/Stop Control:

- Include a "Start" button to initiate real-time data plotting and a "Stop" button to pause or halt data updates.

- Consider a "Pause" button for temporary halts without stopping the data source.

6. Real-Time Data Stream:

- Display a data stream or log alongside the real-time graph to show incoming data points.

- Provide timestamped data points for context.

7. Live Updates:

- Ensure that the graph updates in real-time as new data arrives.

- Implement smooth transitions or animations for data points to prevent abrupt jumps.

8. Data Buffering: Implement a data buffer to manage and display a limited history of data points. This helps prevent overwhelming the interface with excessive data.

9**. Data Filtering and Analysis**: Include options for users to filter and analyse the real-time data, such as setting data thresholds or applying mathematical operations.

10. Data Export: Allow users to export or save the real-time data and graphs for further analysis or reporting.

11.User Notifications: Use visual and auditory cues (e.g., colour changes, alarms, notifications) to alert users to critical events or data anomalies.

12.Performance Optimization: Optimize the UI and data handling to ensure smooth real-time updates without causing excessive CPU or memory usage.

13.User Assistance and Documentation: Provide tooltips, guides, or documentation to help users understand how to use the real-time plotting features effectively.

14. Error Handling: Implement error messages and alerts for issues related to data sources or real-time updates.

15.Customizable Alerts: Allow users to set custom alerts or triggers based on specific data conditions.

16.Responsive Design: Ensure that the UI adapts to various screen sizes and orientations, especially if the application is used on mobile devices.

17.User Testing: Conduct usability testing with potential users to refine the UI based on their feedback and needs.

18. Security: If the application deals with sensitive data, ensure data security measures are in place, such as encryption and authentication.

19.Cross-Platform Compatibility: - Consider developing the application for various platforms, such as web, desktop, and mobile, to reach a broader audience.

A responsive and intuitive UI will help users monitor and analyse data effectively as it evolves in real-time.

UI stands for "User Interface." It refers to the point of interaction between a human user and a computer system or software application. The user interface encompasses all the elements, controls, and design components that allow users to interact with and control the system or application.

UI design is a critical aspect of software development because it directly affects how users perceive and use a product. A well-designed user interface can enhance user experience, making the software more intuitive, efficient, and enjoyable to use. In contrast, a poorly designed UI can lead to user frustration and difficulties in using the software effectively.

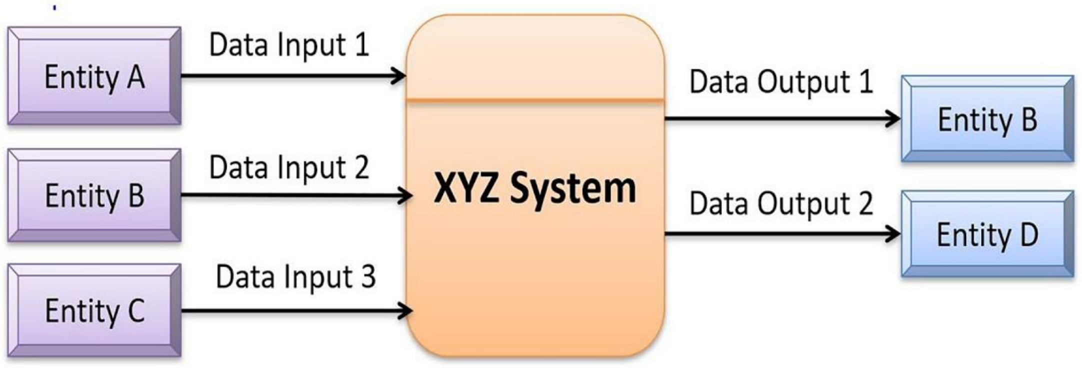
Chapter 5

System Design

5.1 System design

It is the core concept behind the design of any distributed systems. System Design is defined as a process of creating an architecture for different components, interfaces, and modules of the system and providing corresponding data helpful in implementing such elements in systems.

5.2 Data flow diagram



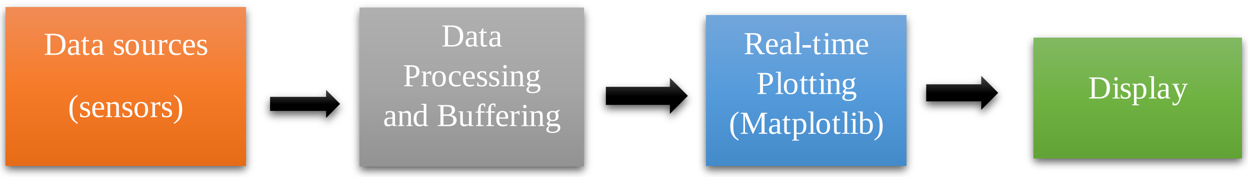
Here:

- ✓ entity A, B, C are the objects, human beings and etc
- ✓ XYZ is program or system which calculates the distance
- ✓ Entity D &E are the output data in the form of graph

It gives the graph according to the distance it is mainly used for the detection of objects.

5.3 Block diagram

Creating a block diagram for plotting a real-time graph using Matplotlib typically involves outlining the key components and how they interact with each other


Data sources
(sensors)
Data Processing and Buffering
Real-time Plotting (Matplotlib)
Display

Data Source: This is where your data originates, such as readings from the NodeMCU and Ultrasonic Sensor.

Data Processing and Buffering: This component processes incoming data, performs any necessary calculations or filtering, and stores it in a buffer. It prepares the data for plotting.

Real-time Plotting (Matplotlib): Matplotlib is used to create and update the real-time graph. It retrieves data from the buffer and plots it on the graph. The graph updates continuously based on a defined interval.

Display: The final output of the real-time graph is displayed here, typically on a computer screen or other output device.

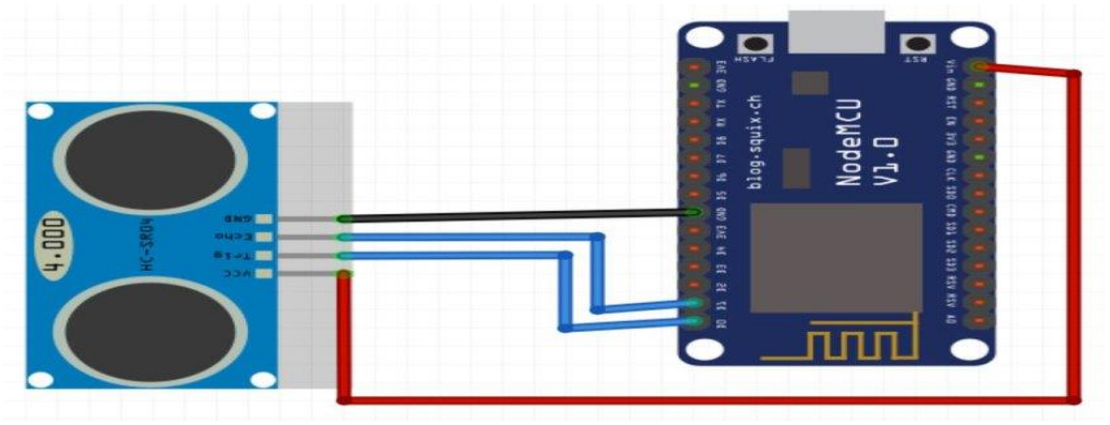
The actual implementation of these components can vary depending on your specific use case and the programming language you are using. As we are using python hence the libraries should be installed into your system. The key is to have a data source, data processing, and plotting components working together to achieve real-time graphing.

Chapter 6

IMPLEMENTATION

5.1 Prototype mode of the system

Creating a prototype for a NodeMCU and an ultrasonic sensor involves connecting the hardware and writing some code to read data from the sensor and possibly send it to a display or a computer. Below is a simple prototype using a NodeMCU (ESP8266) and an HC-SR04 ultrasonic sensor. This prototype will measure distances and display them on the NodeMCU's serial monitor



Connections

1. Connect the NodeMCU to your computer via USB.
2. Connect the ultrasonic sensor (HC-SR04) to the NodeMCU as follows:
   - VCC (Sensor) → 5V (NodeMCU)
   - GND (Sensor) → GND (NodeMCU)
   - TRIG (Sensor) → D1 (NodeMCU)
   - ECHO (Sensor) → D2 (NodeMCU)

Ensure you have the necessary libraries installed in the Arduino IDE (e.g., ESP8266 board support and library for the HC-SR04).

- Connect Power to NodeMCU:
- Power the NodeMCU using a USB cable connected to your computer or a compatible power source.
- Upload Code (as described in a previous response):
- Upload the Arduino code to the NodeMCU using the Arduino IDE.
- Monitor Data:
- Open the Arduino IDE's serial monitor (Tools > Serial Monitor) at the correct baud rate (115200 in the example code provided earlier). You should see the measured distance displayed in the serial monitor.
- This prototype setup allows you to physically connect and test your NodeMCU and ultrasonic sensor circuit

5.2 Implementation of hardware and software:

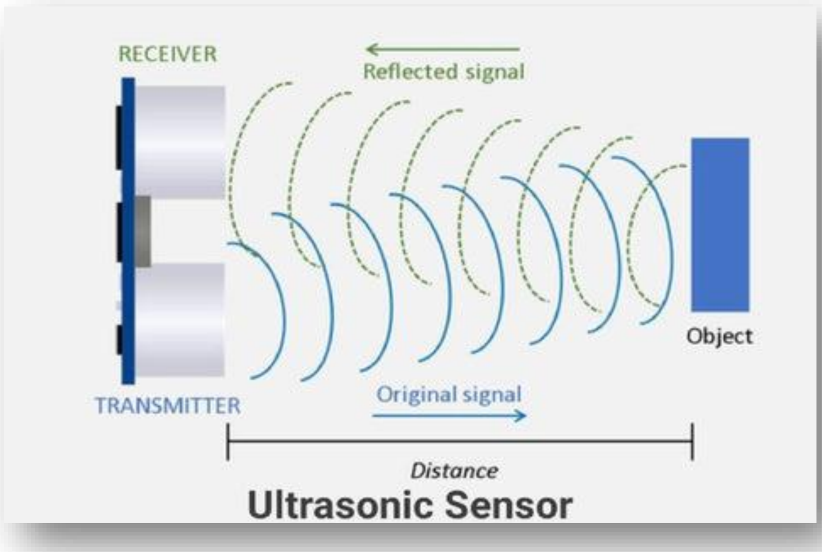
Ease of monitoring your local graph/data conditions in real time from anywhere in the world.

- Firstly the D5 and D6 pin of the Node MCU are defined as trig Pin and echo Pin respectively, as constants with integer data type.
- Then a variable "duration" is defined with a float data type because it will hold the value of the sound wave travel time (microseconds) which will have decimal points.
- The variable distance is in the integer data type because we require it to be in solid integers rather than floating point values with decimal points.
- In the <void setup ()> function the pin modes for trig Pin and echo Pin are set, moreover the serial communication between the PC and the Node MCU is initialized at a baud rate of 9600 bits per second.
- In the <void loop ()> function, firstly the trig Pin is cleared by setting it LOW and waiting for 2 milliseconds, and then it is set HIGH for 10 millisecond as to transmit a 10 ms sound wave trigger pulse, again setting it LOW afterwards.

- The time taken by the pulse to hit an object and returning back is calculated by the pulse In function which measures that after how long the echo Pin received a HIGH after the sound wave trigger pulse was transmitted, this time is saved in the duration variable.
- Finally the distance is calculated by multiplying the duration with the velocity of sound (0.034 centimeters /microseconds) as to employ the formula S=Vt, this distance is divided by two because it is calculated on the two-way time (pulse going to the object and returning back after hitting the object) and only one-way of it is sufficient.

5.2.1 Node MCU

The Node MCU (***N***ode ***M***icro ***C***ontroller ***U***nit) is an open-source software and hardware development environment built around an inexpensive System-on-a-Chip (SoC) called the ESP8266. The ESP8266, contains the crucial elements of a computer: CPU, RAM, networking (Wi Fi), and even a modern operating system and SDK. That makes it an excellent choice for Internet of Things (IoT) projects of all kinds.5.2.2. Ultrasonic sensor



Ultrasonic sensor working principle is either similar to sonar or radar which evaluates the target/object attributes by understanding the received echoes from sound/radio waves correspondingly. These sensors produce high-frequency sound waves and analyze the echo which is received from the sensor. The sensors measure the time interval between transmitted and received echoes so that the distance to the target is known.

Ultrasonic pins are:

VCC – This pin has to be connected to a power supply +5V.

TRIG – This pin is used to receive controlling signals from the Arduino board. This is the triggering input pin of the sensor

ECHO – This pin is used for sending signals to the Arduino board where the Arduino calculates the pulse duration to know the distance. This pin is the ECHO output of the sensor.

GND – This pin has to be connected to the ground.

Operating system windows 10: Windows 10 is used in current and latest version of windows. Making UML diagrams, canvases using Photoshop (version: CC16), E draw, and paint as well. Power point presentation, sensor coning in python IDE, connecting sensors and software for trail we are using android mobile application.

Python scripting language: Python is a basic language of processors and controllers such as Node MCU, Raspberry PI, etc. All the coding related to sensor would be done in this language. This language is very easy to implement because it's coding look like basic C language. Currently we are using 3.7 version of python IDE.

ARDUINO IDE: **The Arduino Integrated Development Environment** - or Arduino Software (IDE) - connects to the Arduino boards to upload programs and communicate with them. Programs written using Arduino Software (IDE) are called **sketches**. These sketches are written in the text editor and are saved with the file extension.

The Arduino Software (IDE) makes it easy to write code and upload it to the board offline. We recommend it for users with poor or no internet connection. This software can be used with any Arduino board.

Steps to be followed while

1) A **Toolbar with buttons** for common functions and a series of menus. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

2) The **message area**, gives feedback while saving and exporting and also displays errors.

3) The **text editor** for writing your code.

4) The **text console** displays text output by the Arduino Software (IDE), including complete error messages and other information.

5) **Connect your Arduino** board to your computer.

6) Now, you need to **select the right core & board**. This is done by navigating to **Tools > Board > Arduino AVR Boards > Board**. Make sure you select the board that you are using. If you cannot find your board, you can add it from **Tools > Board > Boards Manager**.

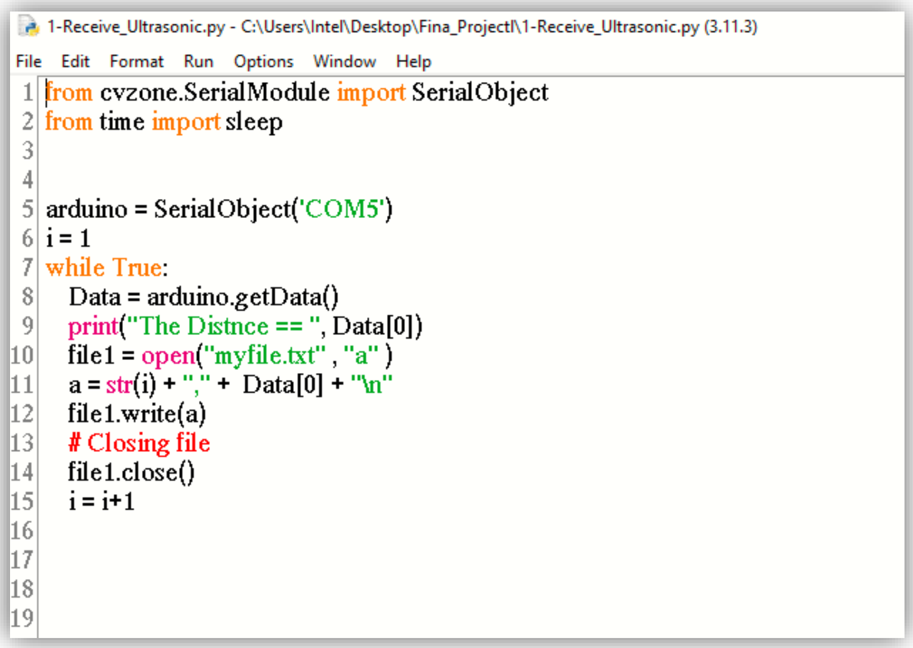
7) Now, let's make sure that your board is found by the computer, by **selecting the port**. This is simply done by navigating to **Tools > Port**, where you select your board from the list. To **upload it to your board**, simply click on the arrow in the top left corner. This process takes a few seconds, and it is important to not disconnect the board during this process. If the upload is successful, the message "Done uploading" will appear in the bottom output area.

8) **10.** Once the upload is complete, you should then see on your board the yellow LED with an L next to it start blinking. You can **adjust the speed of blinking** by changing the delay number in the parenthesis to 100, and upload the Blink sketch again. Now the LED should blink much faster

Matplotlib: Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

- Create publication quality plot.
- Make interactive figures that can zoom, pan, update.
- Customize visual style and layout.
- Export to many file formats.
- Embed in graphical user interface
- Use a rich array of third party to built on Matplotlib.

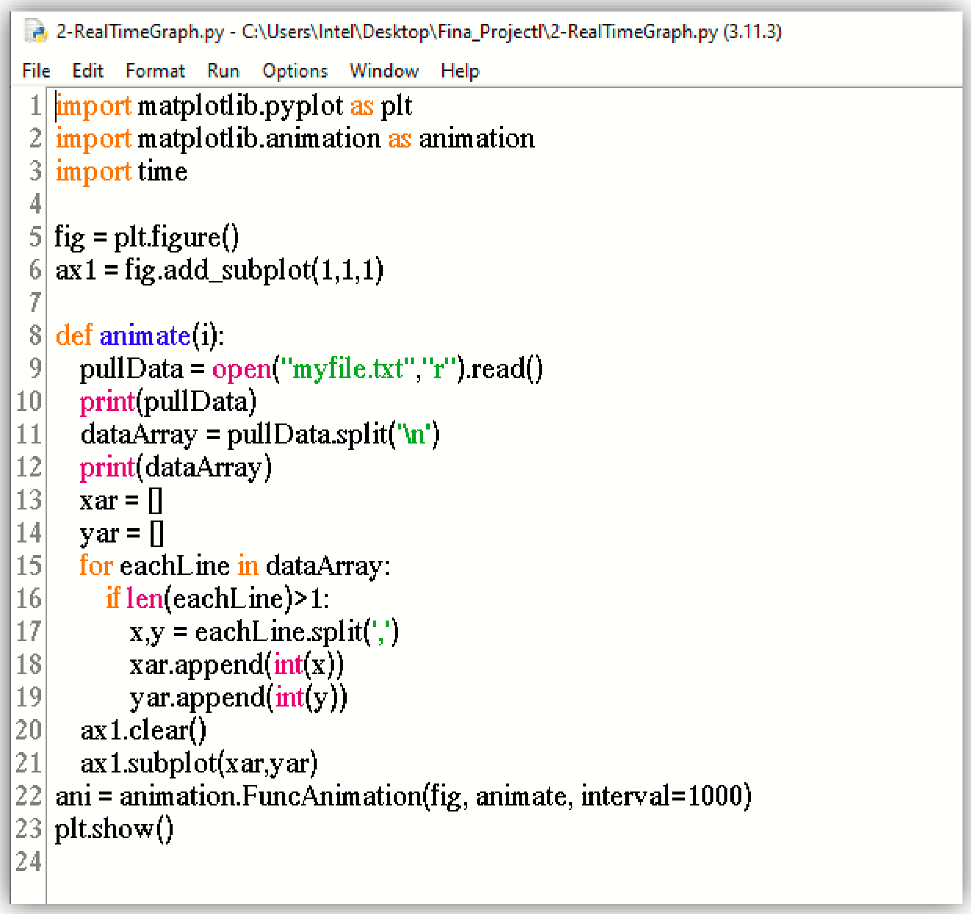
Code By running this code, it creates a file which contains the data-points. The serial object 'COM5' may vary, as it collects these data value from the sensor


1-Receive_Ultrasonic.py - C:\Users\Intel\Desktop\Fina_Project\1-Receive_Ultrasonic.py (3.11.3)
File Edit Format Run Options Window Help

```python
from cvzone.SerialModule import SerialObject
from time import sleep


arduino = SerialObject('COM5')
i = 1
while True:
    Data = arduino.getData()
    print("The Distnce == ", Data[0])
    file1 = open("myfile.txt" , "a" )
    a = str(i) + "," +  Data[0] + "\n"
    file1.write(a)
    # Closing file
    file1.close()
    i = i+1
```

This module retrieves the input values from the myfile.txt (previously created) and then it plots the graph


2-RealTimeGraph.py - C:\Users\Intel\Desktop\Fina_ProjectI\2-RealTimeGraph.py (3.11.3)
File Edit Format Run Options Window Help

```python
import matplotlib.pyplot as plt
import matplotlib.animation as animation
import time

fig = plt.figure()
ax1 = fig.add_subplot(1,1,1)

def animate(i):
    pullData = open("myfile.txt","r").read()
    print(pullData)
    dataArray = pullData.split('\n')
    print(dataArray)
    xar = []
    yar = []
    for eachLine in dataArray:
        if len(eachLine)>1:
            x,y = eachLine.split(',')
            xar.append(int(x))
            yar.append(int(y))
    ax1.clear()
    ax1.subplot(xar,yar)
ani = animation.FuncAnimation(fig, animate, interval=1000)
plt.show()
```

After executing these two programs we open the Arduino file which has the code for ultrasonic sensor and upload it not before checking the connecting appropriate drivers and boards.

While the program execution takes place you may notice in the serial monitor as it gives the distance real time.

Chapter 7

Testing

Software validation is the process to find errors /faults in a software application with the aim to deliver a quality product to the customer. Testing makes software predictable in nature.

Objectives of testing:

- Preventing defects by verifying requirements are implemented completely and correctly.
- Testing early in the life cycle by reviewing requirement documents.
- Finding defects. E.g. Integration, testing, System Testing, Regression Testing to detect defects at all levels.
- To make software predictable in nature. E.g.: software is
- behaving as per requirements

Types of testing:

- ✓ Testing the interfaces between components
- ✓ Tests communication between 2 components, not individual component functionality.
- ✓ Integration Testing is carried out by testers

Unit Testing

- ✓ It is a level of the software testing process where individual components of a system are tested.
- ✓ Objective is to validate that each unit of the software performs as designed.
- ✓ Concerned with functional correctness and completeness of individual program units

System testing:

- ✓ Testing complete, integrated system to evaluate the system's compliance with its specified requirements.
- ✓ Carried out by specialist's testers or independent testers.
- ✓ Covers both functional and non-functional requirements of the testing.

- ✓ Objective is verifying and validate both the business requirements as well as the applications architecture
- ✓ Testing application thoroughly to verify that it meets the technical and functional specifications

User acceptance testing:

- ✓ UAT is a process of verifying that a solution works for the user.
- ✓ User Acceptance Testing is the last phase of the software testing process.
- ✓ User Acceptance Testing is formal testing conducted to determine whether a system satisfies its acceptance criteria.

Effective testing is a fundamental practice in software development that helps reduce defects, improve software quality, and build confidence in the product. It is an iterative and ongoing process throughout the software development lifecycle.
IoT testing is an ongoing process that continues as your project evolves and new features are added. It's crucial to thoroughly test your IoT solution to ensure it operates reliably and securely in real-world scenarios.

Chapter 8
Maintenance

The process of tracking and enabling project activities in accordance with the project plan.
happens after the project team deploys the software and it's fully operational in the customer environment. During the maintenance phase, the customer monitors the software to ensure it continues to operate according to the coding specification.
In IOT maintenance phase is predictive maintenance:
Predictive maintenance requires the ability to process large amounts of data and run sophisticated algorithms, which cannot be achieved with local implementation.

A data lake stores the data gathered by sensors. It is still raw, so it may be inaccurate, erroneous or contain irrelevant items. It is presented as a number of sets of sensor readings measured at the corresponding time. When the data is needed for insights about battery's health, it is loaded to a big data warehouse. The big data warehouse stores cleansed structured data. It contains temperature, voltage and discharge parameters measured at a particular time and contextual information about batteries' types, locations, recharge dates, etc. Once the data is prepared, it is analyzed with machine learning (ML) algorithms. ML algorithms are applied to reveal hidden.

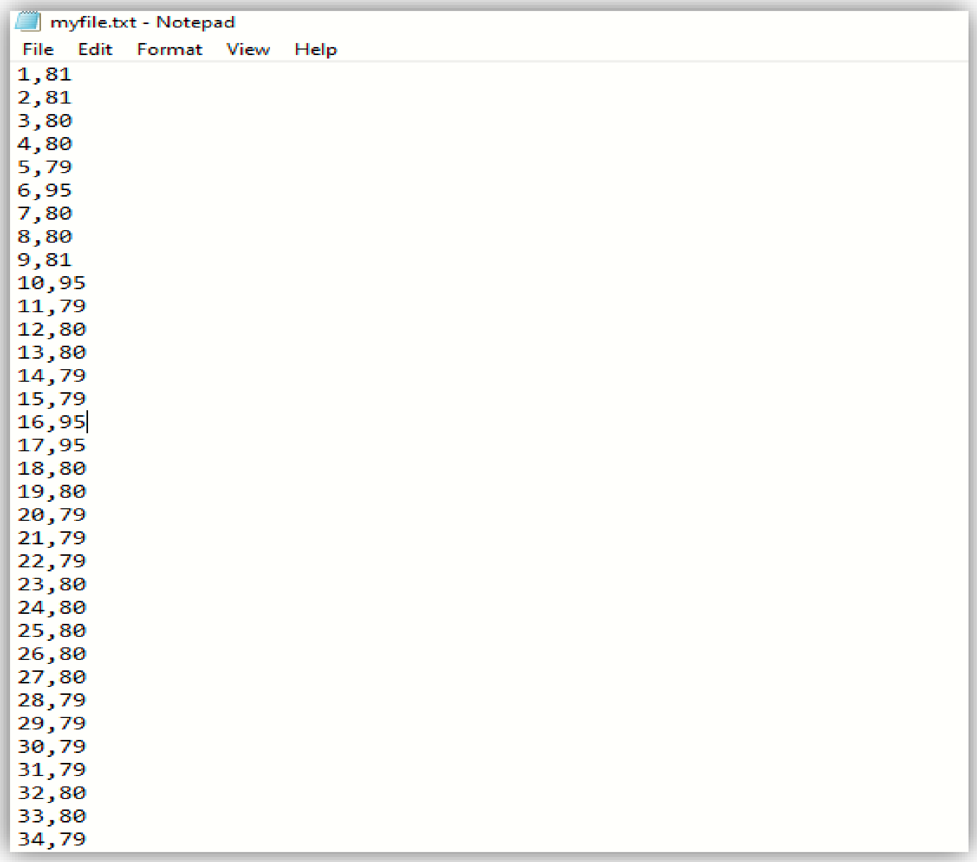
- Classification approach - models built according to this approach identify whether a battery is likely to self-discharge and show if the capacity of a battery is lower than normal or not.
- Regression approach - models provide the information on how many days/cycles are left until a battery's useful life ends.

correlations in data sets and detect abnormal data patterns. The recognized data patterns are reflected in predictive models. on how many days/cycles are left until a battery's useful life ends. Predictive models are regularly updated, say, once a month, and tested for accuracy. If the output differs from the expected one, they are revised, retrained and tested again, until they function as intended. Before getting down to machine learning, a good amount of exploratory analytics should be carried out. Data analysis is carried out to find dependencies and discover patterns and insights in the machine learning data sets. Moreover, during the exploratory analytics stage, various technical assumptions are assessed to help select the best-fit machine learning algorithm. User applications allow an IoT-based predictive maintenance solution to alert users of a potential battery failure.

Chapter 9
Screen shots

Outputs:

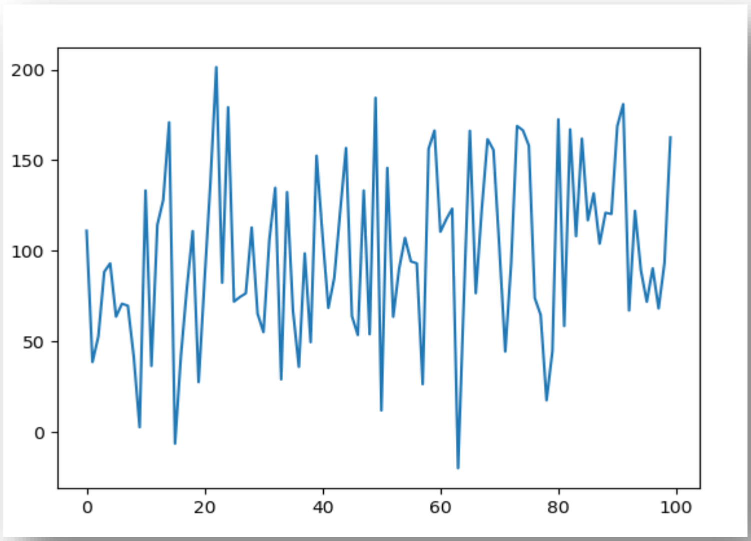
Myfile.txt is created which inputs the real time distance and hence is used to plot the chart

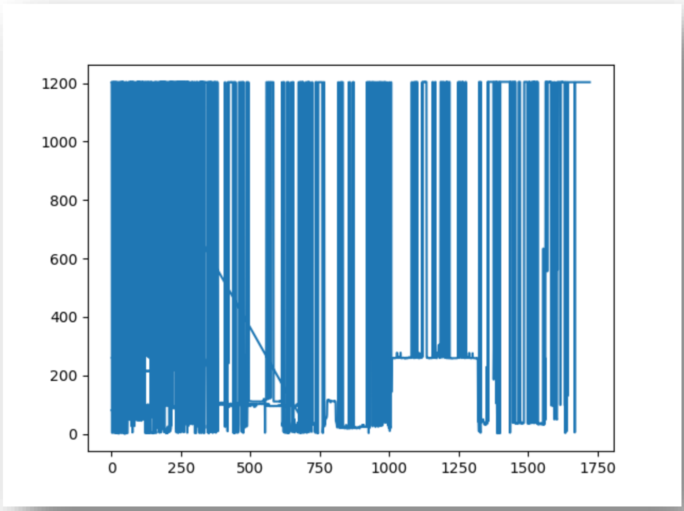


```
1,81
2,81
3,80
4,80
5,79
6,95
7,80
8,80
9,81
10,95
11,79
12,80
13,80
14,79
15,79
16,95
17,95
18,80
19,80
20,79
21,79
22,79
23,80
24,80
25,80
26,80
27,80
28,79
29,79
30,79
31,79
32,80
33,80
34,79
```

Typically, good quality ultrasonic sensors can provide accuracy within a few millimetres to centimetres, depending on the factors mentioned above. For critical applications requiring high precision, it's important to carefully select and calibrate the sensor, and to account for potential sources of error in the measurement setup

Graphs:





The sensor reads the values while we try the different distance as it detects the distance between the objects the graphs is plotted through these varied numbers. Hence engages us to study the environment in which this experiment occurs.

Chapter 10

Conclusion and scope

10.1 Conclusion

The environment for monitoring enables self-protection (i.e., smart environment) to the environment. To implement this need to use the sensor devices in the environment for collecting the data and analysis. By using sensor devices in the environment, we can bring the environment into real life. Then the collected data and analysis results will be available to the user through the Wi-Fi. The smart way to monitor the environment an efficient, low-cost embedded system is presented in this paper. It also sent the sensor parameters to the cloud. This data will be helpful for future analysis and it can be easily shared to other users also. This model can be expanded to monitor the developing cities and industrial zones for pollution monitoring. To protect the public health from pollution, this model provides an efficient and low-cost solution for continuous monitoring of environment.

In conclusion, generating real-time graphs in IoT is an indispensable aspect of data visualization, decision-making, and optimization. By harnessing the power of real-time data, IoT applications can improve efficiency, responsiveness, and overall user experience while unlocking new possibilities for various industries and domains.

The scope for real-time graphs continues to grow as businesses and organizations recognize the value of real-time data analysis for decision-making, optimization, and enhancing user experiences. Advances in technology, including faster data processing, improved data visualization tools, and the proliferation of IoT devices, contribute to the expanding role of real-time graphs in various sectors.

In conclusion, the generation and utilization of real-time graphs play a crucial role across a wide range of industries and applications. The ability to visualize data as it happens, rather than relying on static representations, provides organizations with valuable insights, enabling them to make timely decisions, optimize processes, enhance user experiences, and respond to dynamic situations effectively.

As technology continues to advance, the scope for real-time graphs will only expand further. From financial services to healthcare, manufacturing to transportation, and cybersecurity to entertainment, real-time data visualization is an indispensable tool for monitoring, analyzing, and leveraging data streams in the modern world.

To harness the full potential of real-time graphs, organizations must invest in robust data infrastructure, powerful visualization tools, and skilled data analysts and engineers. Additionally, staying up-to-date with emerging technologies and best practices in data analytics will be essential for maximizing the benefits of real-time data visualization in the years ahead.

In a data-driven world, the ability to capture, process, and present data in real time provides a competitive advantage and empowers decision-makers to adapt swiftly to changing circumstances. Real-time graphs are at the forefront of this data revolution, shaping the way we understand and interact with information in the digital age.

10.2 Future scope:

One can implement a few more sensors and connect it to the satellite as a global feature of this system. Adding more sensors to monitor other environmental parameters such as CO2, Pressure and Oxygen Sensor. In aircraft, navigation and the military there is a great scope of this real-time system. It can also be implemented in hospitals or medical institutes for the research & study in “Effect of Weather on Health and Diseases”, hence to provide better precaution alerts. Generating real-time graphs in the context of IoT (Internet of Things) has numerous benefits and applications. Here are some key conclusions regarding the importance and advantages of generating real-time graphs in IoT:

1. Real-Time Monitoring: Real-time graphs allow users to monitor IoT devices and sensor data as events occur, enabling prompt decision-making and immediate responses to critical situations. This is particularly crucial in applications such as industrial automation, healthcare monitoring, and environmental sensing.

2. Data Visualization: Graphical representations of IoT data help users understand complex patterns, trends, and anomalies quickly. Real-time graphs provide an intuitive and visual way to analyse data, enabling users to identify issues, correlations, and opportunities for optimization.

3. Predictive Insights: Real-time graphs, when combined with analytics and machine learning algorithms, can offer predictive insights. By analysing historical and current data trends, these graphs can help forecast potential issues or predict future trends, contributing to more informed decision-making.

4. Faster Problem Detection and Troubleshooting: With real-time graphs, anomalies and deviations from expected behaviour can be immediately detected. This facilitates timely troubleshooting and reduces downtime, leading to improved efficiency and productivity.

5. Enhanced User Experience: Real-time graphs can be integrated into IoT applications and dashboards, enhancing the overall user experience. Users can interact with the live data, customize views, and receive alerts when specific thresholds are breached.

6. Remote Monitoring and Control: Real-time graphs enable remote monitoring and control of IoT devices, allowing users to access and manage data from anywhere at any time. This flexibility is particularly valuable in scenarios where physical presence is challenging or risky.

7. <u>Adaptive Systems:</u> IoT systems can be made more adaptive by leveraging real-time graphs. By continuously analyzing incoming data and adjusting parameters in real-time, IoT applications can optimize their performance and responsiveness.

8. <u>Sustainability and Resource Management</u>: Real-time graphs can help in energy management, resource optimization, and waste reduction. By visualizing real-time data on energy consumption or resource usage, organizations and individuals can make more sustainable decisions.

9. <u>Scalability and Flexibility</u>: Real-time graphing solutions can be designed to handle large volumes of data from multiple IoT devices. Scalable and flexible graphing platforms can accommodate diverse data streams and support future expansions.

10.<u>Market and Business Insights</u>: Real-time graphs can provide valuable market insights and trends by analysing data from IoT-connected products and services. This data-driven approach can lead to more targeted marketing strategies and better understanding of customer behaviour.

1. **Environmental Monitoring:** Real-time graphs are used in environmental science to monitor air quality, weather conditions, and pollution levels. They provide valuable data for research and policy-making.
2. **Cybersecurity:** Real-time graphs are used to detect and respond to cybersecurity threats in real-time. They help visualize network traffic and identify suspicious activities.
3. **Social Media and Marketing:** Marketers use real-time data visualization to track social media trends, campaign performance, and customer engagement. It helps in making real-time adjustments to marketing strategies.
4. **Gaming and Entertainment:** Real-time graphics and visual effects are fundamental in the gaming and entertainment

industry. They enhance the gaming experience and create immersive virtual environments.

5. **Research and Science:** Real-time graphs play a significant role in scientific research, from physics experiments to climate modeling. They allow scientists to observe and analyze data as it is collect

The scope for IoT (Internet of Things) devices is vast and continues to expand as technology advances and new use cases emerge. IoT devices are essentially physical objects embedded with sensors, software, and connectivity capabilities, allowing them to collect and exchange data with other devices and systems over the internet

The scope for IoT devices is continually expanding due to advancements in sensor technology, wireless connectivity, and data analytics. As more industries and sectors recognize the value of IoT in improving efficiency, reducing costs, and enhancing decision-making, the IoT ecosystem will continue to evolve and shape the way we interact with the physical world. It's important to note that along with the opportunities, IoT also presents challenges related to data security, privacy, and interoperability, which need to be addressed as the scope of IoT continues to grow.

Chapter 11

BIBILIOGRAPHY

- https://www.scnsoft.com/blog/iot-predictive-maintenance-guide
- https://resources.experfy.com/iot/the-5-most-important-requirements-for-an-iot-project/
- https://www.pythonpool.com/python-serial-read/
- https://matplotlib.org/stable/index.html
- https://towardsdatascience.com/plotting-live-data-with-matplotlib-d871fac7500b
- https://wpforms.com/survey-data-visualization-tools/
- https://makersportal.com/blog/2018/8/14/real-time-graphing-in-python