VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELGAVI, KARNATAKA

"SMART GLASSES"

A Report
Submitted in fulfillment for the award of degree Of
Bachelor of Engineering In

ELECTRONCS AND COMMUNICATIONS ENGINEERING

By

MONISHA R	(4NI22EC413)
KEERTHANA H M	(4NI22EC410)
MILIND MANJUNATH P	(4NI22EC412)
PALAVELLI SIVANI SANDEEP	(4NI21EC074)

Under the guidance of

Internal
Anand Srivatsa
Associate professor
NIE, Mysuru-18

External
Puneeth S
Associate professor
NIE Mysuru -18

DEPARTMENT OF ELECTRONICS AND COMMUNICATIONS ENGINEERING
THE NATIONAL INSTITUTE OF ENGINEERING
MYSURU-570008

2024-2025

THE NATIONAL INSTITUTE OF ENGINEERING

Mananthawadi Road Mysuru-08

DEPARTMENT OF
ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

Certified that the project work entitled **SMART GLASSES** carried out by **Ms. Monisha R(4NI22EC413),Ms.Keerthana H M (4NI22EC410), Mr. MilindManjunath P (4NI22EC412), Mr. Palavelli Sivani Sandeep (4NI21EC074)** abonafide student of in fulfillment for the award of **Bachelor of Engineering** in Electronics And Communication of the Visvesvaraya Technological University,Belgaum during the year 2024-25. It is certified that all corrections/suggestionsindicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. The project report has been approved as it satisfies the academi crequirements in respect of Project work prescribed for the said Degree.

Name & Signature of the Guide

Name Signature of the HOD

External Viva

Name of the examiners

Signature with date

UNDERTAKING

We **MONISHA R, KEERTHANA H M,MILLIND MANJUNATH P, PALAVELLI SIVANI SANDEEP,** HEREBY UNDERTAKE THAT THE PROJECT WORK ENTITLED "SMARTGLASSES**"** IS CARRIED OUT BY ME INDEPENDENTLY UNDER THE GUIDANCE OF **ANAND SRIVATSA** ,**PUNEETH S ,** DEPARTMENT OF **ELECTRONICS AND COMMUNICATION**, NIE, MYSURU-08, IN FULFILLMENT OF THE REQUIREMENT FOR THE AWARD OF **BACHELOR OF ENGINEERING** IN **2025** BY THE VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM. THE PROJECT HAS BEEN MYORIGINAL WORK AND HAS NOT FORMED THE BASIS FOR THE AWARD OF ANY DEGREE , ASSOCIATE SHIP, FELLOWSHIP OR ANY OTHER SIMILAR TITLES.

SIGNATURE OF STUDENT

ACKNOWLEDGEMENT

It gives me immense pleasure to express my deepest sense of gratitude and sincere thanks to my highly respected and esteemed guide Mr. Anand Srivatsa Department of Electronics And Communication Mysore, for their valuable guidance, encouragement and help for completing this work. Their useful suggestions for this whole work and cooperative behavior are sincerely acknowledged. I would like to express my sincere thanks to Puneeth S for giving me this opportunity to undertake this project. I also wish to express my gratitude to Dr. S Parameshwara HOD (Electronics And Communication) for his kind hearted support. I also wish to express my indebtedness to my parents as well as my family member whose blessingsand support always helped me to face the challenges ahead. At the end I would like to express my sincere thanks to all my friends and others who helped me directly or indirectly during this project work.

ABSTRACT

This project explores the potential of standalone smart glasses, focusing on their integration of augmented reality (AR) and advanced digital capabilities independent of mobile devices. Smart glasses offer a hands-free interface that enhances user interaction with digital information and applications. The research begins with a review of current technologies and trends in wearable AR, followed by an analysis of technical specifications and design considerations specific to standalone operation. Emphasis is placed on user interface design, application development, and usability testing to optimize functionality and user experience. Results highlight the feasibility and benefits of standalone smart glasses across industries such as healthcare, education, and manufacturing. This project contributes to understanding the transformative impact of autonomous wearable technologies in enhancing productivity and daily life interactions.

CONTENTS

CHAPTER 1
1.1 INTRODUCTION TO EMBEDDED SYSTEMS

Embedded Technology is now in its prime and the wealth of knowledge available is mind blowing. However, most embedded systems engineers have a common complaint. There are no comprehensive resources available over the internet which deal with the various design and implementation issues of this technology. Intellectual property regulations of many corporations are partly to blame for this and also the tendency to keep technical know-how within a restricted group of researchers.

An embedded computer is frequently a computer that is implemented for a particular purpose. In contrast, an average PC computer usually serves a number of purposes: checking email, surfing the internet, listening to music, word processing, etc... However, embedded systems usually only have a single task, or a very small number of related tasks that they are programmed to perform.

Every home has several examples of embedded computers. Any appliance that has a digital clock, for instance, has a small embedded micro-controller that performs no other task than to display the clock. Modern cars have embedded computers onboard that control such things as ignition timing and anti-lock brakes using input from a number of different sensors.

Embedded computers rarely have a generic interface, however. Even if embedded systems have a keypad and an LCD display, they are rarely capable of using many different types of input or output. An example of an embedded system with I/O capability is a security alarm with an LCD status display, and a keypad for entering a password.

An embedded system can be defined as a control system or computer system designed to perform a specific task. Common examples of embedded systems include MP3 players, navigation systems on aircraft and intruder alarm systems. An embedded system can also be defined as a single purpose computer.
Most embedded systems are time critical applications meaning that the embedded system is working in an environment where timing is very important: the results of an operation are only relevant if they take place in a specific time frame. An autopilot in an aircraft is a time critical embedded system. If the autopilot detects that the plane for some reason is going into a stall then it should take steps to correct this within milliseconds or there would be catastrophic results.

1.2 APPLICATIONS OF EMBEDDED SYSTEM

Embedded systems are commonly found in consumer, cooking, industrial, automotive, medical, commercial and military applications.

Telecommunications systems employ numerous embedded systems from telephone switches for the network to cell phones at the end user. Computer networking uses dedicated routers and network bridges to route data.

Consumer electronics include MP3 players, mobile phones, videogame consoles, digital cameras, GPS receivers, and printers. Household appliances, such as microwave ovens, washing machines and dishwashers, include embedded systems to provide flexibility, efficiency and features. Advanced HVAC systems use networked thermostats to more accurately and efficiently control temperature that can change by time of day and season. Home automation uses wired- and wireless-networking that can be used to control lights, climate, security, audio/visual, surveillance, etc., all of which use embedded devices for sensing and controlling.

Transportation systems from flight to automobiles increasingly use embedded systems. New airplanes contain advanced avionics such as inertial guidance systems and GPS receivers that also have considerable safety requirements. Various electric motors brushless DC motors, induction motors and DC motors use electric/electronic motor controllers. Automobiles, electric vehicles, and hybrid vehicles increasingly use embedded systems to maximize efficiency and reduce pollution. Other automotive safety systems include anti-lock braking system (ABS), Electronic Stability Control (ESC/ESP), traction control (TCS) and automatic four-wheel drive.
Medical equipment uses embedded systems for vital signs monitoring, electronic stethoscopes for amplifying sounds, and various medical imaging (PET, SPECT, CT, and MRI) for non-invasive internal inspections. Embedded systems within medical equipment are often powered by industrial computers.[9]

Embedded systems are used in transportation, fire safety, safety and security, medical applications and life critical systems, as these systems can be isolated from hacking and thus, be more reliable. For fire safety, the systems can be designed to have greater ability to handle higher temperatures and continue to operate. In dealing with security, the embedded systems can be self-sufficient and be able to deal with cut electrical and communication systems.

1.3 CHARACTERISTICS OF EMBEDDED SYSTEM

Embedded systems are designed to do some specific task, rather than be a general-purpose computer for multiple tasks. Some also have real-time performance constraints that must be met, for reasons such as safety and usability; others may have low or no performance requirements, allowing the system hardware to be simplified to reduce costs.

Embedded systems are not always standalone devices. Many embedded systems consist of small parts within a larger device that serves a more general purpose. For example, the Gibson Robot Guitar features an embedded system for tuning the strings, but the overall purpose of the Robot Guitar is, of course, to play music. Similarly, an embedded system in an automobile provides a specific function as a subsystem of the car itself.

Embedded systems range from no user interface at all, in systems dedicated only to one task, to complex graphical user interfaces that resemble modern computer desktop operating systems. Simple embedded devices use buttons, LEDs, graphic or character LCDs (HD44780 LCD for example) with a simple menu system.

More sophisticated devices which use a graphical screen with touch sensing or screen- edge buttons provide flexibility while minimizing space used: the meaning of the buttons can change with the screen, and selection involves the natural behavior of pointing at what is desired. Handheld systems often have a screen with a "joystick button" for a pointing device.

Some systems provide user interface remotely with the help of a serial (e.g. RS- 232, USB, I²C, etc.) or network (e.g. Ethernet) connection. This approach gives several advantages: extends the capabilities of embedded system, avoids the cost of a display, simplifies BSP and allows one to build a rich user interface on the PC. A good example of this is the combination of an embedded web server running on an embedded device (such as an IP camera) or a network router. The user interface is displayed in a web browser on a PC connected to the device, therefore needing no software to be installed.

1.4PROCESSORS IN EMBEDDED SYSTEMS

Embedded processors can be broken into two broad categories. Ordinary microprocessors Embedded processors can be broken into two broad categories. Ordinary microprocessors (μP) use separate integrated circuits for memory and peripherals. Microcontrollers (μC) have on- chip peripherals, thus reducing power consumption, size and cost. In contrast to the personal computer market, many different basic CPU architectures are used, since software is custom- developed for an application and is not a commodity product installed by the end user. Both Von Neumann as well as various degrees of Harvard architectures are used. RISC as well as non-RISC processors are found. Word lengths vary from 4-bit to 64-bits and beyond, although the most typical remain 8/16-bit. Most architectures come in a large number of different variants and shapes, many of which are also manufactured by several different companies.

Numerous microcontrollers have been developed for embedded systems use. General- purpose microprocessors are also used in embedded systems, but generally require more support circuitry than microcontrollers.

(μP) use separate integrated circuits for memory and peripherals. Microcontrollers (μC) have on-chip peripherals, thus reducing power consumption, size and cost. In contrast to the personal computer market, many different basic CPU architectures are used, since software is custom-developed for an application and is not a commodity product installed by the end user. Both Von Neumann as well as various degrees of Harvard architectures are used. RISC as well as non-RISC processors are found. Word lengths vary from 4-bit to 64-bits and beyond, although the most typical remain 8/16-bit. Most architectures come in a large number of different variants and shapes, many of which are also manufactured by several different companies.

Numerous microcontrollers have been developed for embedded systems use. General- purpose microprocessors are also used in embedded systems, but generally require more support circuitry than microcontrollers.

.

CHAPTER 2
2.1 INTRODUCTION

Smart Glasses are a wearable technology designed to provide real-time data to users by displaying notifications, messages, and environmental information on an OLED screen. These glasses integrate wireless connectivity, a microphone for transcription, and battery-powered portability to enhance user experience.

2.2 SYSTEM DESIGN

The system consists of the following components:

1. **ESP32 DevKit** – The main processing unit, responsible for data reception and display.
2. **0.96" OLED Display** – Displays real-time notifications, alerts, and information.
3. **INMP441 Digital Microphone** – Captures voice input for transcription.
4. **LiPo Battery** – Provides portable power supply.
5. **TP4056** – Battery charging and protection module.
6. **Smartphone Connectivity** – Enables data transfer via Bluetooth/WiFi.

OBJECTIVES

- To display real-time notifications from a smartphone on an OLED screen.
- To provide call and message alerts on the display.
- To show date, time, temperature, and location information.
- To enable transcription using a digital microphone.
- To ensure portability through a rechargeable LiPo battery.

2.3 WORKING

1. The ESP32 DevKit connects to a smartphone via Bluetooth or WiFi.
2. The smartphone sends notifications, messages, and environmental data to the ESP32.
3. The ESP32 processes the received data and displays it on the 0.96" OLED display.
4. The INMP441 microphone captures voice input, which is processed for transcription.
5. The system is powered by a LiPo battery and managed by the TP4056 module.

2.4 ADVANTAGES

- Hands-free access to real-time information.
- Enhances productivity with instant notifications.
- Voice transcription for improved accessibility.
- Portable and rechargeable design.
- Low power consumption.

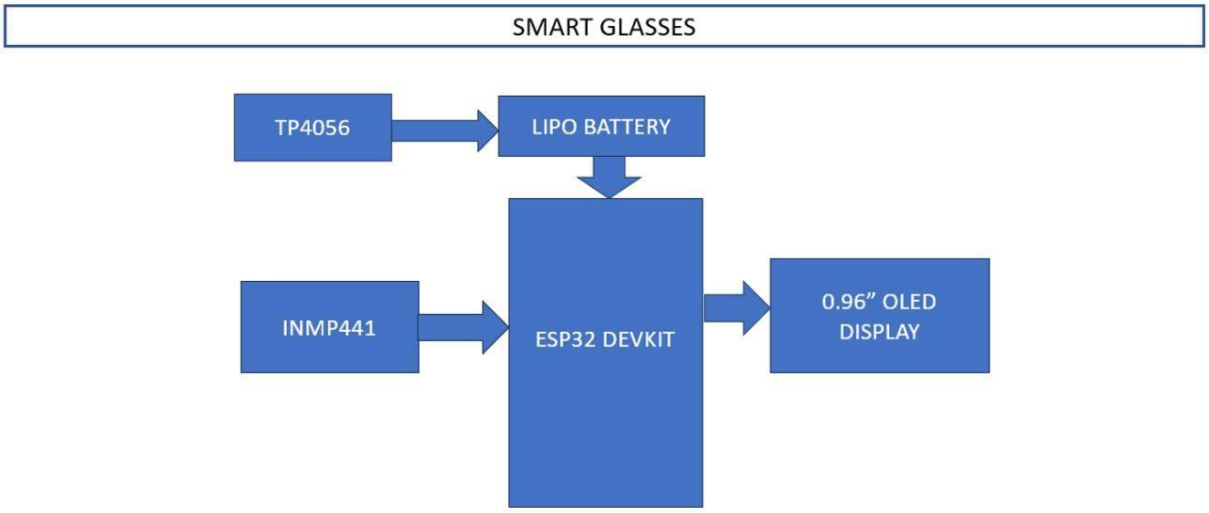
2.5 APPLICATIONS

- Assistive technology for visually impaired individuals.
- Wearable tech for professionals (engineers, doctors, etc.).
- Real-time notifications for cyclists and motorcyclists.
- Augmented reality applications.
- Personal assistant for daily tasks.

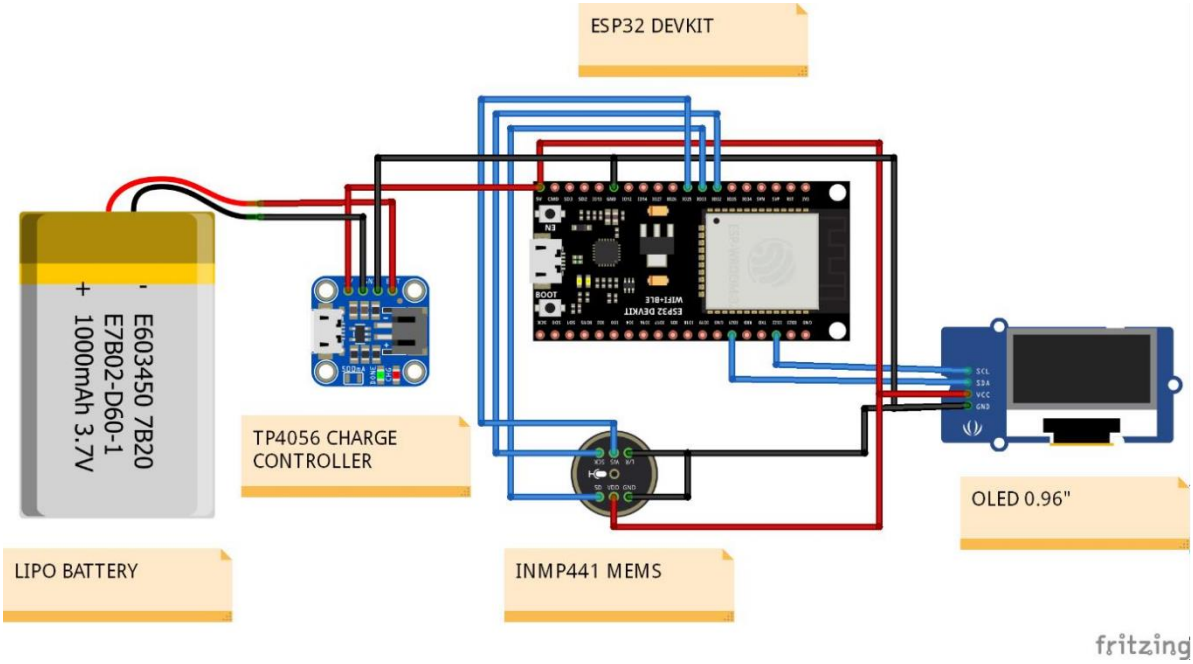
2.6 FUTURE SCOPE

- Integration of Augmented Reality (AR) for enhanced interaction.
- Expansion to support voice commands and AI assistants.
- Improved battery efficiency and lightweight design.
- Enhanced security features such as facial recognition.

2.7 BLOCK DIAGRAM



2.8 CIRCUIT DIAGRAM



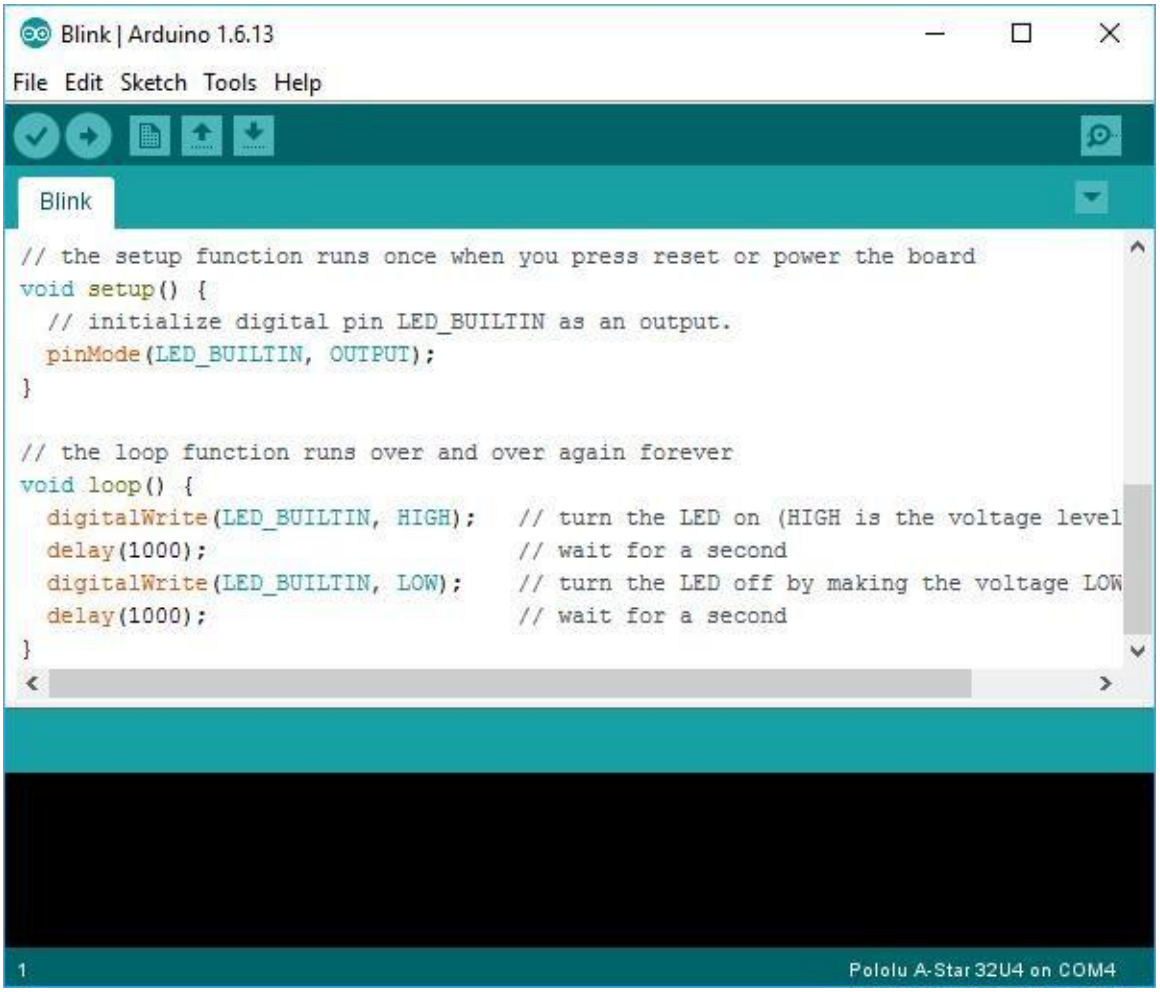
CHAPTER 3
SOFTWARE REQUIREMENTS

Software used in this project for uploading code onto Arduino is Arduino IDE.

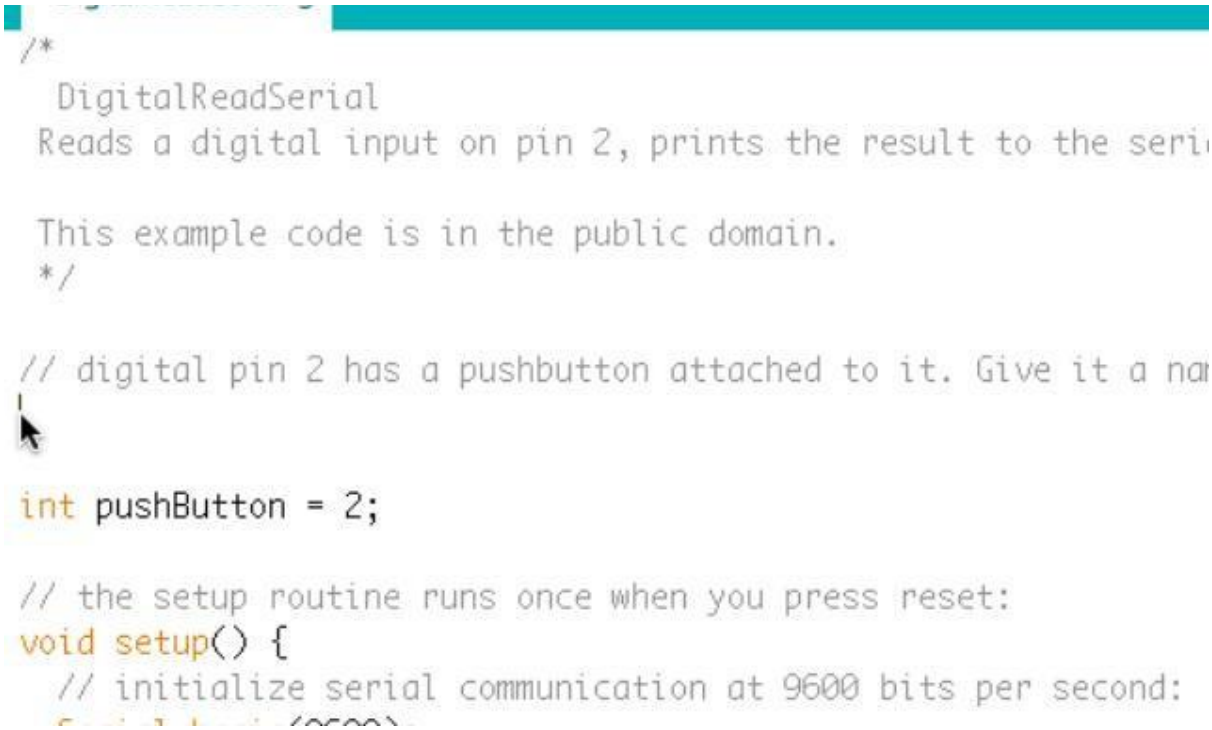
3.1 INTRODUCTION TO ARDUINO IDE

IDE stands for Integrated Development Environment. Pretty fancy sounding, and should make you feel smart any time you use it. The IDE is a text editor-like program that allows you to write Arduino code. When you open the Arduino program, you are opening the IDE. It is intentionally streamlined to keep things as simple and straightforward as possible. When you save a file in Arduino, the file is called a sketch – a sketch is where you save the computer code you have written. The coding language that Arduino uses is very much like C++ ("see plus plus"), which is a common language in the world of computing. The code you learn to write for Arduino will be very similar to the code you write in any other computer language – all the basic concepts remain the same – it is just a matter of learning a new dialect should you pursue other programming languages.

The code you write is "human readable", that is, it will make sense to you (sometimes), and will be organized for a human to follow. Part of the job of the IDE is to take the human readable code and translate it into machine-readable code to be executed by the Arduino. This process is called compiling. The process of compiling is seamless to the user. All you have to do is press a button. If you have errors in your computer code, the compiler will display an error message at the bottom of the IDE and highlight the line of code that seems to be the issue. The error message is meant to help you identify what you might have done wrong – sometimes the message is very explicit, like saying, "Hey – you forget a semicolon", sometimes the error message is vague. Why be concerned with a semicolon you ask? A semicolon is part of the Arduino language syntax, the rules that govern how the code is written. It is like grammar in writing. Say for example we didn't use periods when we wrote – everyone would have a heck of a time trying to figure out when sentences started and ended. Or if we didn't employ the comma, how would we convey a dramatic pause to the reader?



And let me tell you, if you ever had an English teacher with an overactive red pen, the compiler is ten times worse. In fact – your programs WILL NOT compile without perfect syntax. This might drive you crazy at first because it is very natural to forget syntax. As you gain experience programming you will learn to be assiduous about coding grammar.



3.1.2 INTRODUCTION ARDUINO LIBRARIES

Libraries are a collection of code that makes it easy for you to connect to a sensor, display, module, etc. For example, the built-in LiquidCrystal library makes it easy to talk to character LCD displays. There are hundreds of additional libraries available on the Internet for download. The built-in libraries and some of these additional libraries are listed in the reference. To use the additional libraries, you will need to install them.

Arduino libraries are managed in three different places: inside the IDE installation folder, inside the core folder and in the libraries folder inside your sketchbook. The way libraries are chosen during compilation is designed to allow the update of libraries present in the distribution. This means that placing a library in the “libraries” folder in your sketchbook overrides the other libraries versions.

The same happens for the libraries present in additional cores installations. It is also important to note that the version of the library you put in your sketchbook may be lower than the one in the distribution or core folders, nevertheless it will be the one used during compilation. When you select a specific core for your board, the libraries present in the core’s folder are used instead of the same libraries present in the IDE distribution folder.

Last, but not least important is the way the Arduino Software (IDE) upgrades itself: all the files in Programs/Arduino (or the folder where you installed the IDE) are deleted and a new folder is created with fresh content. This is why we recommend that you only install libraries to the sketchbook folder so they are not deleted during the Arduino IDE update process.

3.2 JAVASCRIPT

JavaScript is the scripting language of the Web.
JavaScript is used in millions of Web pages to add functionality, validate forms, detect browsers, and much more.

Introduction to JavaScript

JavaScript is used in millions of Web pages to improve the design, validate forms, detect browsers, create cookies, and much more.
JavaScript is the most popular scripting language on the Internet, and works in all major browsers, such as Internet Explorer, Mozilla Firefox, and Opera.

What is JavaScript?

- JavaScript was designed to add interactivity to HTML pages
- JavaScript is a scripting language
- A scripting language is a lightweight programming language
- JavaScript is usually embedded directly into HTML pages
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation)
- Everyone can use JavaScript without purchasing a license

Java and JavaScript are two completely different languages in both concept and design!
Java (developed by Sun Microsystems) is a powerful and much more complex programming language - in the same category as C and C++.

What can a JavaScript Do ?

- JavaScript gives HTML designers a programming tool - HTML authors are normally not programmers, but JavaScript is a scripting language with a very simple syntax! Almost anyone can put small "snippets" of code into their HTML pages
- JavaScript can put dynamic text into an HTML page - A JavaScript statement like this: document.write("
" + name + "
") can write a variable text into an HTML page
- JavaScript can react to events - A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on an HTML element
- JavaScript can read and write HTML elements - A JavaScript can read and change the content of an HTML element
- JavaScript can be used to validate data - A JavaScript can be used to validate form data before it is

submitted to a server. This saves the server from extra processing
- JavaScript can be used to detect the visitor's browser - A JavaScript can be used to detect the

visitor's browser, and - depending on the browser - load another page specifically designed for that browser
- JavaScript can be used to create cookies - A JavaScript can be used to store and retrieve information on the visitor's computer.

3.3 MQTT

3.3.1 Introduction to MQTT

What is MQTT?

MQTT is a lightweight, publish/subscribe messaging protocol designed for constrained devices and unreliable networks.

Pop quiz – "MQTT" stands for:
1. Message Queuing Telemetry Transport?
2. MQ Telemetry Transport?
3. MQTT?

What does "MQTT" stand for?

While it formerly stood for MQ Telemetry Transport, where MQ referred to the MQ Series, a product IBM developed to support MQ telemetry transport, MQTT is no longer an acronym.

MQTT is now simply the name of the protocol.

Although many sources label MQTT as a Message Queue Telemetry Transport protocol, this is not entirely accurate. While it is possible to queue messages in certain cases, MQTT is not a traditional message queuing solution. (from HiveMQ book).

Why Use MQTT?

MQTT is used because it's simple, efficient, and works well even when the internet connection is slow or unstable.

Asynchronous communication: Clients can publish data without worrying about whether subscribers are available, as the broker handles message routing. It allows devices to send and receive messages without waiting for an immediate response.
Low overhead and reduced power consumption: With a small packet size and a lightweight control mechanism, MQTT is efficient for transmitting data.
Support for Intermittent Connections: Devices in constrained environments often lose network connectivity. MQTT supports persistent sessions to handle this gracefully.

3.3.2 When to Choose Each Protocol

MQTT: Ideal for low-bandwidth, high-latency environments where devices need to conserve power, such as IoT devices (sensors, actuators, smart home systems). Suitable when asynchronous, decoupled communication is needed, such as in telemetry or distributed control systems.
HTTP: Best for web applications and REST-based APIs where resource exchange and direct request-response interactions are required (e.g., file sharing, data retrieval from a server). Inefficient for IoT scenarios due to its high overhead and statelessness.
CoAP: Designed for constrained IoT devices that need efficient, low-overhead communication. Ideal for sensor networks, smart energy management, and low-power wireless communication.
AMQP: Preferred in enterprise-level applications where reliable, secure, and transactional messaging is required (e.g., financial institutions, message queuing systems, distributed systems). Suitable for complex routing and store-and-forward messaging scenarios where full delivery guarantees and reliability are crucial.

MQTT in the Modern IoT and Cloud Ecosystem

Use cases: Home automation, industrial IoT, autonomous vehicles, smart cities.
Supported by cloud services: like AWS IoT, Azure IoT Hub, Google Cloud IoT Core, IBM Watson, Oracle IoT
Commercial brokers: HiveMQ, …
MQTT-like protocols: Apache Kafka (LinkedIn), Pulsar, ActiveMQ, …

3.3.3 MQTT Protocol Overview

Core MQTT Concepts

Publish: The action of sending a message to a specific topic on the broker.
Subscribe: The action of a client registering to receive messages from a specific topic.
Topic: A string used to organize and categorize messages, acting as a routing key for message distribution.
Broker: Central server that manages communication between clients by routing messages from publishers to subscribers.
Client: Any device or application that connects to the broker to publish or subscribe to topics.
QoS (Quality of Service): Defines the level of guarantee for message delivery (0, 1, or 2).
Retain Flag: Option to store the last message sent to a topic for future subscribers.
Will Message: A message defined by a client that the broker will send if the client unexpectedly disconnects.
Clean Session: An option to specify whether a client's subscription and message queue should be cleared when it disconnects.
Session Persistence: The broker can retain client subscriptions and undelivered messages between disconnections.

3.4 React Native

React Native is a framework developed by Facebook for creating native-style applications for Android & iOS under one common language, i.e. JavaScript. Initially, Facebook only developed React Native to support iOS. However, with its recent support of the Android operating system, the library can now render mobile UIs for both platforms.

What is React Native?

React Native is an open-source framework developed by Facebook that allows developers to build mobile applications using JavaScript and React. The key advantage of React Native is that it enables the development of apps for both iOS and Android platforms using a single codebase, which significantly reduces development time and effort.

Key Features of React Native

- **Cross-Platform Development:** Write once, run anywhere. React Native allows developers to create apps for both iOS and Android platforms using a shared codebase.
- **Native Performance:** React Native components are compiled into native code, which means that the apps deliver performance close to native apps.
- **Reusability:** React Native enables code reusability, which allows developers to use the same code for different platforms with minimal adjustments.
- **Hot Reloading:** This feature allows developers to instantly see the results of the latest change to the source code without rebuilding the app from scratch.
- **Component-Based Architecture:** Similar to React for the web, React Native uses a component-based architecture, making it easier to manage complex applications by breaking them down into smaller, reusable components.

How React Native Works?

React Native bridges the gap between JavaScript and native code. When you write code in React Native, it gets compiled into native components for iOS and Android. This means the app runs with the performance and look-and-feel of a native app.

Components of React Native

- **View:** This is the fundamental building block of UI in React Native. It maps to the native view on both iOS and Android.
- **Text:** Used for displaying text. It renders as native text elements on both platforms.
- **TextInput:** A component for text input fields.
- **ScrollView:** A component that provides a scrolling container.
- **StyleSheet:** A utility for defining styles. It works similarly to CSS.

Why Learn React Native?

- React Native allows developers to create mobile apps using website technology. So a developer who is handy in web development can easily develop a mobile app using React Native.
- React Native allows developers to build cross-platform apps that look and feel entirely Native since it uses JavaScript components that are both built on iOS and Android components.
- Since React Native uses JSX, a developer isn't required to learn complex languages.
- React Native uses fundamental Android and iOS building blocks to compile Native apps for both platforms in JavaScript. This makes handling the code base easier.
- React Native allows you to build apps faster. Instead of recompiling, you can reload an app instantly.
- In React Native we can easily develop and test features by using various libraries and tools like Expo, ESLint, Jest, and Redux.
- On top of being responsive and providing an impressive user experience, React Native apps are faster and agile.
- React Native uses Native components which makes the rendering and execution of the app much faster.

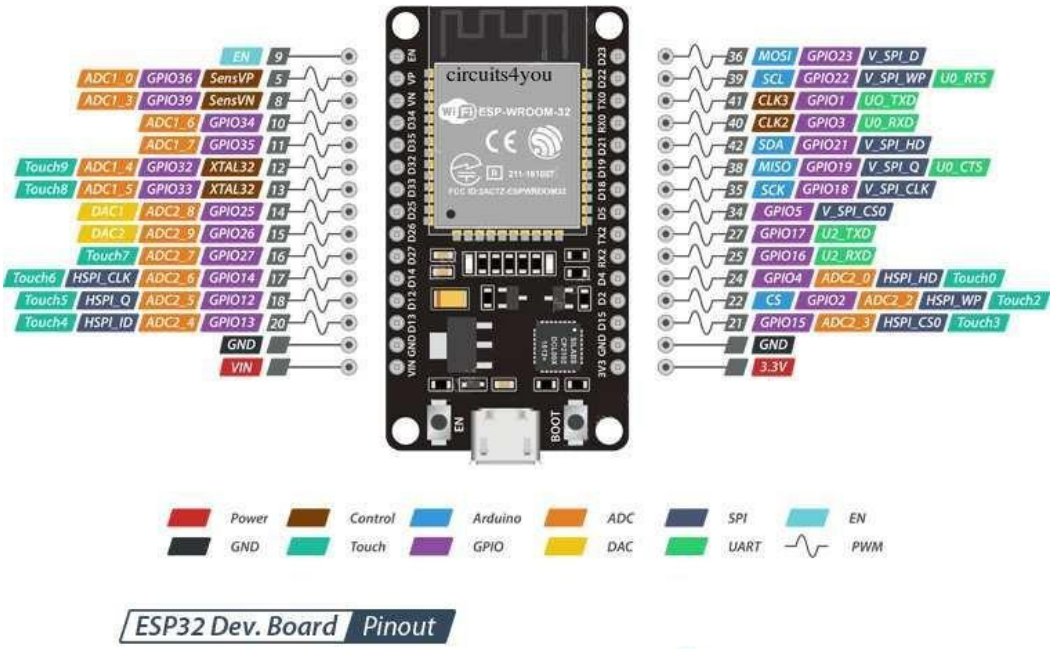
CHAPTER 4
HARDWARE REQUIREMENTS

Hardware Components of this project are

1. ESP32
2. OLED DISPLAY
3. LI-PO BATTERY
4. TP4056
5. INMP441

4.1 INTRODUCTION TO ESP32

NodeMCU is an open source IoT platform It includes firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module. The term "NodeMCU" by default refers to the firmware rather than the development kits. The firmware uses the Lua scripting language. It is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266. It uses many open source projects



4.1.1 HISTORY OF ESP32

NodeMCU was created shortly after the ESP8266 came out. On December 30, 2013, Espressif Systems began production of the ESP8266. The ESP8266 is a Wi-Fi SoC integrated with a Tensilica Xtensa LX106 core, widely used in IoT applications (see related projects). NodeMCU started on 13 Oct 2014, when Hong committed the first file of nodemcu-firmware to GitHub. Two months later, the project expanded to include an open- hardware platform when developer Huang R committed the gerber file of an ESP8266 board, named devkit v0.9. Later that month, Tuan PM ported MQTT client library from Contiki to the ESP8266 SoC platform, and committed to NodeMCU project, then NodeMCU was able to support the MQTT IoT protocol, using Lua to access the MQTT broker. Another important update was made on 30 Jan 2015, when Devsaurus ported the u8glib to NodeMCU project, enabling NodeMCU to easily drive LCD, Screen, OLED, even VGA displays.
In summer 2015 the creators abandoned the firmware project and a group of independent but dedicated contributors took over. By summer 2016 the NodeMCU included more than 40 different modules. Due to resource constraints users need to select the modules relevant for their project and build a firmware tailored to their needs.

4.1.2 ESP32 ARDUINO CORE

As Arduino.cc began developing new MCU boards based on non-AVR processors like the ARM/SAM MCU and used in the Arduino Due, they needed to modify the Arduino IDE so that it would be relatively easy to change the IDE to support alternate tool chains to allow Arduino C/C++ to be compiled down to these new processors.

They did this with the introduction of the Board Manager and the SAM Core. A "core" is the collection of software components required by the Board Manager and the Arduino IDE to compile an Arduino C/C++ source file down to the target MCU's machine language. Some creative ESP8266 enthusiasts have developed an Arduino core for the ESP8266 WiFi SoC that is available at the GitHub ESP8266 Core webpage. This is what is popularly called the "ESP8266 Core for the Arduino IDE" and it has become one of the leading software development platforms for the various ESP8266 based modules and development boards, including NodeMCUs.

4.1.3 FEATURES OF ESP32

- Finally, programable WiFi module.
- Arduino-like (software defined) hardware IO.
- Can be programmed with the simple and powerful Lua programming language or Arduino IDE.
- USB-TTL included, plug & play.
- 10 GPIOs D0-D10, PWM functionality, IIC and SPI communication, 1-Wire and ADC A0 etc. all in one board.

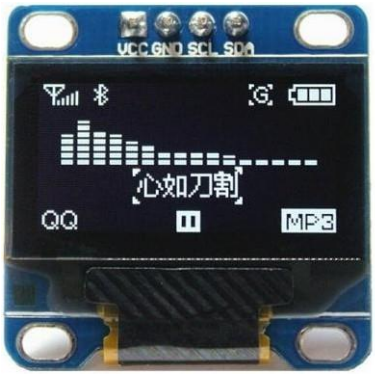
- WiFi networking (can be used as access point and/or station, host a web server), connect to internet to fetch or upload data.

- Event-driven API for network applications.

- PCB antenna.

4.2 INTRODUCTION TO SSD1306 OLED DISPLAY

SSD1306 is a single-chip CMOS OLED/PLED driver with controller for organic / polymer light emitting diode dot-matrix graphic display system. It consists of 128 segments and 64commons. This IC is designed for Common Cathode type OLED panel. The SSD1306 embeds with contrast control, display RAM and oscillator, which reduces the number of external components and power consumption. It has 256-step brightness control. Data/Commands are sent from general MCU through the hardware selectable 6800/8000 series compatible Parallel Interface, I 2 C interface or Serial Peripheral Interface. It is suitable for many compact portable applications, such as mobile phone sub-display, MP3 player and calculator, etc.



4.2.1 FEATURES

Resolution: 128 x 64 dot matrix panel

- Power supply o VDD = 1.65V to 3.3V for IC logic o VCC = 7V to 15V for Panel driving

- For matrix display o OLED driving output voltage, 15V maximum o Segment maximum source current: 100uA o Common maximum sink current: 15mA o 256 step contrast brightness current control

- Embedded 128 x 64 bit SRAM display buffer

- Pin selectable MCU Interfaces: o 8-bit 6800/8080-series parallel interface o 3 /4 wire Serial Peripheral Interface o I 2 C Interface.

- Screen saving continuous scrolling function in both horizontal and vertical direction
- RAM write synchronization signal
- Programmable Frame Rate and Multiplexing Ratio
- Row Re-mapping and Column Re-mapping
- On-Chip Oscillator • Chip layout for COG & COF
- Wide range of operating temperature: -40°C to 85°C

4.3 LI-PO BATTTERY

A Lithium Polymer (Li-Po) battery is a type of rechargeable battery that falls under the broader category of lithium-ion batteries. Li-Po batteries have gained popularity due to their lightweight and compact design, making them particularly suitable for use in various electronic devices, including smartphones, laptops, drones, remote-controlled vehicles, and portable consumer electronics.

Key characteristics and features of Li-Po batteries include:

1. Chemistry:
Li-Po batteries use a lithium-ion polymer electrolyte rather than a liquid electrolyte found in traditional lithium-ion batteries. This polymer electrolyte allows for flexibility in the battery's form factor and design.

2. Compact and Lightweight:
- Li-Po batteries are known for their slim and lightweight construction. The absence of a rigid metal casing allows for more flexibility in shaping the battery to fit specific device dimensions.

3. High Energy Density:
- Li-Po batteries generally have a high energy density, meaning they can store a significant amount of energy relative to their size and weight. This makes them well-suited for applications where space and weight are critical factors.

4. Flexibility in Design:
- The flexible packaging of Li-Po batteries allows manufacturers to create custom shapes and sizes to fit the specific requirements of different electronic devices. This flexibility in design is particularly advantageous in smaller and slimmer devices.

5. Voltage Stability:
- Li-Po batteries typically maintain a more stable voltage during discharge compared to other battery types, providing a more consistent power supply to electronic devices.

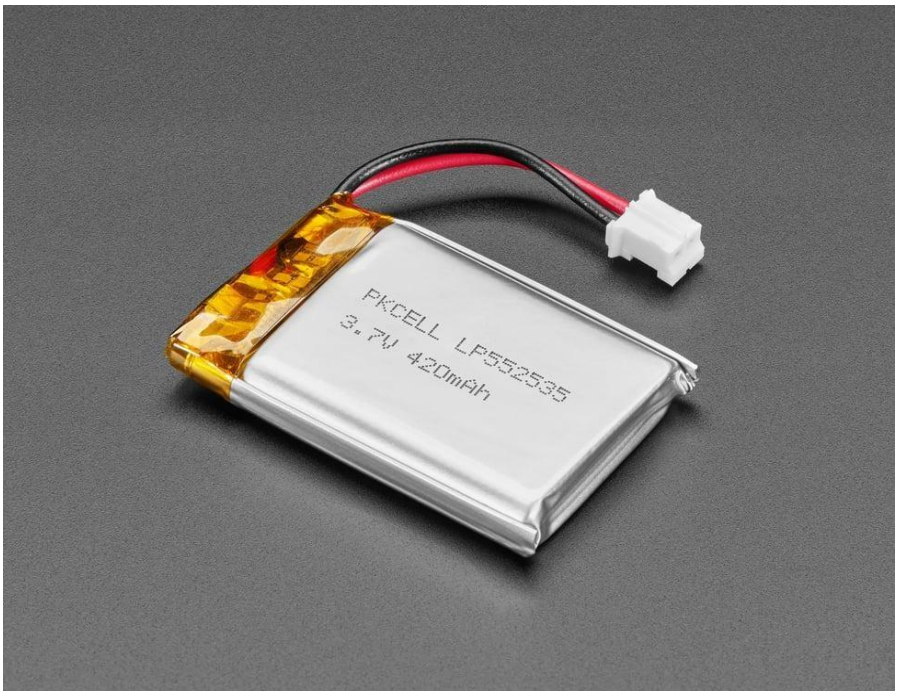
6. Fast Charging:
- Li-Po batteries often support faster charging compared to some other types of rechargeable batteries. This feature is essential for users who require quick recharging of their devices.

7. Prone to Puffing:
- One potential drawback of Li-Po batteries is their susceptibility to "puffing" or swelling, especially when exposed to overcharging or high temperatures. Puffing can affect the battery's performance and safety.

8. Applications:
- Li-Po batteries are widely used in various applications, including consumer electronics, radio-controlled vehicles, drones, wearable devices, and medical equipment.

9. Safety Considerations:
- Proper handling and charging practices are crucial to the safe use of Li-Po batteries. Overcharging, overheating, or physical damage can pose safety risks, including the potential for fire or swelling.

Despite certain safety considerations, Li-Po batteries have become a popular choice in many electronic devices due to their high energy density, lightweight design, and adaptability to various form factors. Advancements in battery technology continue to address safety concerns and improve the overall performance of Li-Po batteries for a wide range of applications.



4.4 INTRODUCTION ON TP4056 CHARGE CONTROLLER

The TP4056 is a commonly used single-cell lithium-ion/lithium-polymer (Li-Ion/LiPo) battery charge controller module. It is designed to manage the charging process of small rechargeable batteries, typically with capacities ranging from around 300mAh to 3000mAh. The TP4056 offers a compact and cost-effective solution for safely charging single-cell lithium batteries, making it popular in various portable electronic devices, such as power banks, smartphones, digital cameras, and IoT devices.

Key Features:

1. **Input Voltage**: The TP4056 can operate with an input voltage between 4.5V and 5.5V, which makes it suitable for direct connection to USB power sources.

2. **Charging Current**: It supports adjustable charging currents, often set to 1A by default. However, the charging current can be reduced by changing the value of the sense resistor connected to the module, allowing for safer charging of smaller battery capacities.

3. **Constant Current/Constant Voltage (CC/CV) Charging:** The TP4056 follows a two-stage charging process. In the initial constant current (CC) phase, it supplies a fixed current to the battery until its voltage reaches a predefined threshold. Then, it transitions to the constant voltage (CV) phase, where it maintains a steady voltage while gradually reducing the charging current until the battery is fully charged.

4. **Constant Current/Constant Voltage (CC/CV) Charging:** The TP4056 follows a two-stage charging process. In the initial constant current (CC) phase, it supplies a fixed current to the battery until its voltage reaches a predefined threshold. Then, it transitions to the constant voltage (CV) phase, where it maintains a steady voltage while gradually reducing the charging current until the battery is fully charged.

5. **Thermal Regulation:** The module is equipped with a thermal feedback loop that monitors the temperature during charging. If the temperature exceeds a certain limit, the charging current is reduced or suspended temporarily to prevent overheating.

6. **Battery Protection:** The TP4056 includes built-in protection features like overcharge protection, over-discharge protection, and short-circuit protection, which enhance the safety of the charging process and safeguard the battery from damage.

7. **Status Indicator:** Many TP4056 modules include LED indicators that provide visual feedback about the charging status. For example, they might display different colors to indicate charging, fully charged, and fault conditions.

8. **Low Standby Current:** When charging is not in progress, the TP4056 enters a low-power standby mode, minimizing power consumption.

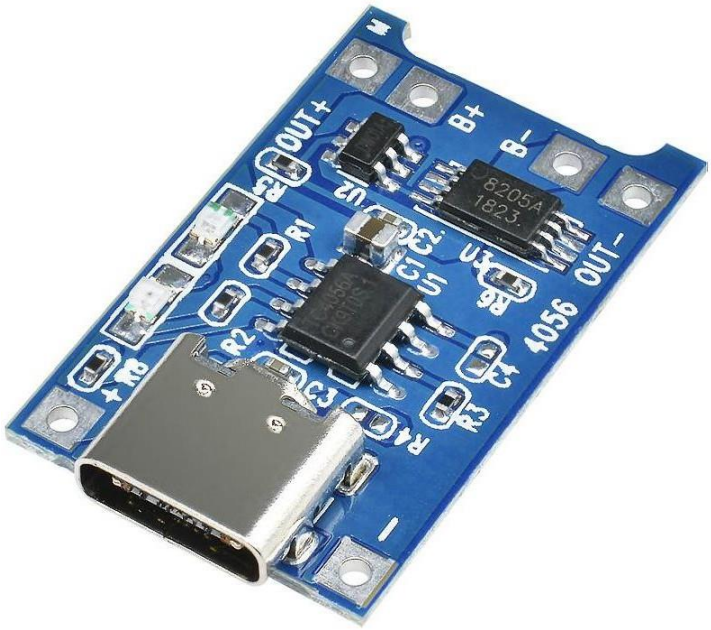
Usage Considerations:

1. **Cooling:** While the TP4056 includes thermal regulation, it's still recommended to provide proper ventilation or heat dissipation mechanisms when using it at higher charging currents, as excessive heat can affect its performance and longevity.

2. **Battery Compatibility**: Ensure that the module's specifications match the battery's requirements, especially in terms of voltage and capacity. Using the wrong settings can lead to improper charging and potential damage to the battery.

3. **Quality Control:** Due to its popularity, there are various clones and counterfeit versions of the TP4056 module on the market. It's advisable to purchase from reputable sources to ensure you're getting a genuine and well-tested product.

The TP4056 charge controller is a versatile and reliable solution for charging single-cell lithium batteries. Its ease of use, compact size, and built-in protection features make it a preferred choice for hobbyists, engineers, and designers working on portable electronics projects. Always refer to the manufacturer's datasheet and guidelines for accurate setup and safe usage of the module.



4.5 INMP441

The **INMP441** is a high-performance, low-power **MEMS (Micro-Electro-Mechanical Systems) microphone** with a digital I²S (Inter-IC Sound) output. It is commonly used in audio applications like voice recognition, audio recording, and noise detection.

Key Features:

- **Digital I²S Output:** Eliminates the need for an ADC.
- **Low Power Consumption:** Suitable for battery-powered applications.
- **High Signal-to-Noise Ratio (SNR):** Provides clear audio capture.
- **Omnidirectional Response:** Captures sound from all directions.
- **Wide Frequency Response:** Suitable for speech and music recording.

This microphone is widely used with microcontrollers like ESP32, Raspberry Pi, and Arduino for IoT and embedded audio applications.

ChatGPT said:

INMP441 MEMS Microphone – Detailed Overview
The **INMP441** is a **low-power, high-performance, digital MEMS microphone** that provides an I²S (Inter-IC Sound) output, making it ideal for audio applications that require digital audio data

transmission without the need for an analog-to-digital converter (ADC). It is widely used in IoT projects, voice recognition systems, and audio recording applications.

Key Specifications

Feature	Specification
Output Type	Digital I²S
Power Supply	1.8V – 3.3V
Current Consumption	~1.4 mA (typical)
Signal-to-Noise Ratio (SNR)	61 dBA **Acoustic Overload Point (AOP)** 120 dB SPL
Frequency Response	60 Hz – 15 kHz
Sensitivity	-26 dBFS
Directivity	Omnidirectional
Operating Temperature	-40°C to +85°C

Pinout Description

The INMP441 module typically comes with **5 pins**:

Pin	Function	Description
VDD	Power Supply	Connect to **3.3V** (Not 5V-tolerant)
GND	Ground	Connect to **GND**
SD	Serial Data Output	Sends the digital audio signal
WS	Word Select (L/R Clock)	Defines left/right audio channel
SCK	Serial Clock (Bit Clock)	Synchronizes data transmission

Working Principle

1. **Sound Detection:** The MEMS diaphragm inside the microphone vibrates in response to sound waves.

2. **Analog to Digital Conversion:** The MEMS sensor converts the vibrations into an electrical signal, which is digitized internally.

3. **I²S Transmission:** The digital audio signal is transmitted via the **I²S interface**, reducing noise and improving signal integrity.

4. **Microcontroller Processing:** The I²S signal is processed by microcontrollers like **ESP32, Raspberry Pi, or STM32**, which can analyze or store the data.

Advantages of INMP441

✓**No need for an external ADC** – Provides direct digital output.

✓**High SNR (61 dBA)** – Ensures clear audio capture.

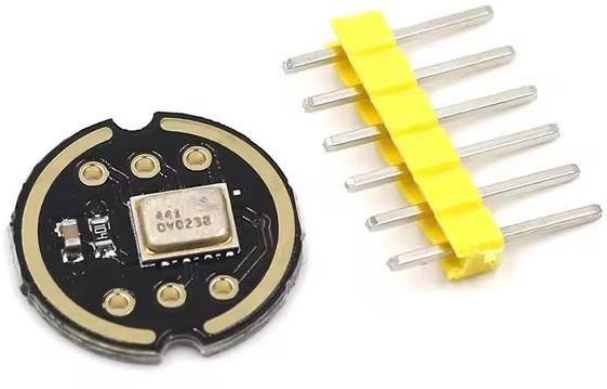
✓**Low power consumption** – Ideal for battery-operated devices.

✓**Omnidirectional pickup** – Captures sound from all directions.

✓**Compact and lightweight** – Easy to integrate into small devices.

Applications

- Voice Assistants (Google Assistant, Alexa, etc.)
- IoT Sound Detection Systems
- Audio Recording & Processing
- Speech Recognition
- Environmental Noise Monitoring
- Smart Home Automation



CHAPTER 5
PROGRAM CODE

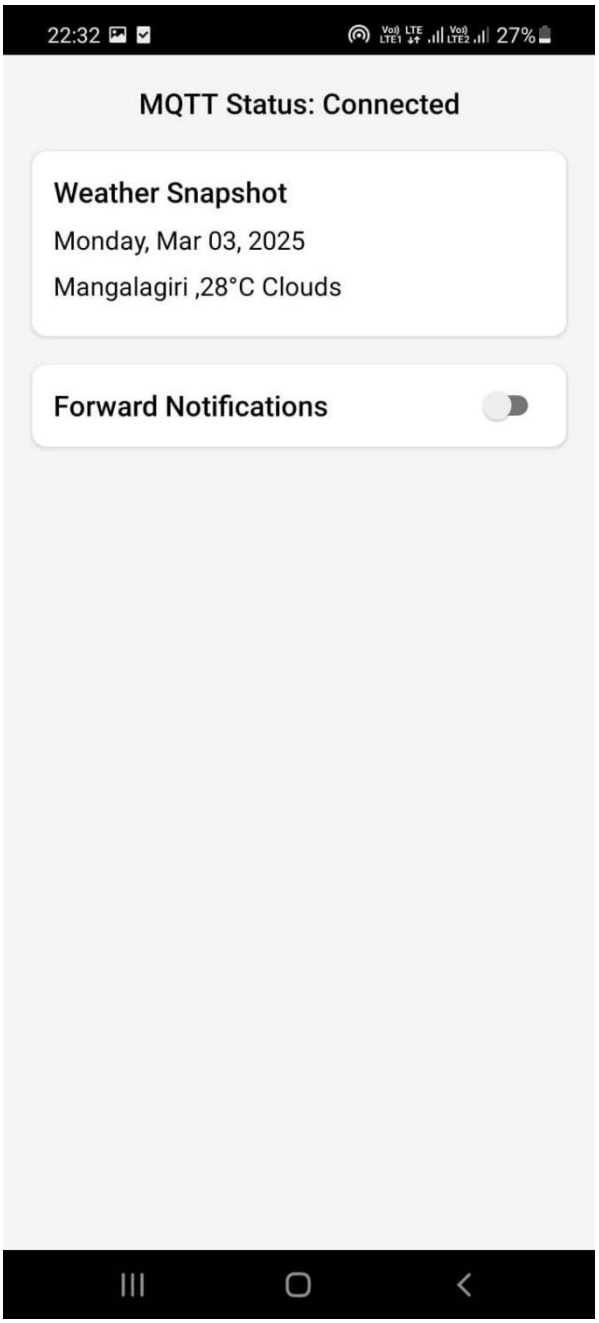
```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <WiFi.h>
#include <BluetoothSerial.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);
 BluetoothSerial SerialBT;

void setup() {
Serial.begin(115200);
SerialBT.begin("Smart Glasses");

if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {

Serial.println(F("SSD1306 allocation failed")); for(;;);
}

display.clearDisplay();
display.setTextSize(1);
display.setTextColor(WHITE);
display.setCursor(0,10);
display.println("Smart Glasses Ready");
display.display();
}



void loop() {

if (SerialBT.available())
{
String message = SerialBT.readString();
 display.clearDisplay();
display.setCursor(0,10);
display.println(message);
 display.display();
}
```

RESULT

Smart Glasses represent an innovative leap in wearable technology, designed to seamlessly integrate digital notifications, voice transcription, and environmental data into a compact and user-friendly device. At the core of the system is the ESP32 DevKit, which acts as the primary processing unit, interfacing with a 0.96" OLED display to provide real-time notifications and alerts. The integration of an INMP441 digital microphone allows for voice-based transcription, enhancing accessibility and hands-free communication. Connectivity is achieved through Bluetooth and WiFi, ensuring seamless synchronization with smartphones. The device is powered by a rechargeable LiPo battery, managed by the TP4056 module, ensuring portability and low power consumption. Applications of Smart Glasses span various industries, from assisting visually impaired individuals to offering real-time alerts for cyclists and professionals in high-paced environments. Future advancements aim to incorporate augmented reality, AI-driven voice commands, and enhanced battery efficiency, further solidifying Smart Glasses as an indispensable tool for modern users. The combination of embedded systems and real-time computing in this device highlights its potential to revolutionize personal and professional interactions, making technology more intuitive and accessible.


GND VDD SCK SDA
Notification:
com.whatsapp: 9963890
013 - Hello


GND VDD SCK SDA
Friday, Jan 31, 2025
Mysore ,21°C Clouds


22:32
27%
MQTT Status: Connected
Weather Snapshot
Monday, Mar 03, 2025
Mangalagiri ,28°C Clouds
Forward Notifications

CONCLUSION

Smart Glasses offer a revolutionary approach to wearable technology by integrating real-time notifications, voice transcription, and environmental data into a compact and portable device. With future advancements, these glasses can become an essential tool in various fields, improving accessibility, productivity, and convenience.

The development of Smart Glasses represents a significant step forward in wearable technology, merging convenience with real-time data access to create a seamless user experience. As digital transformation continues to reshape human interaction with technology, Smart Glasses provide a compelling alternative to conventional mobile devices, enabling hands-free access to notifications, voice transcription, and real-time environmental information. By leveraging embedded systems, these glasses integrate various hardware and software components to deliver a highly functional and user-friendly device, suitable for a wide range of applications.

The core advantage of Smart Glasses lies in their ability to enhance productivity and accessibility. Professionals such as engineers, doctors, and individuals with visual impairments can benefit immensely from the hands-free nature of these glasses. With real- time notifications and voice transcription, communication becomes more efficient, reducing the need for frequent smartphone interactions. Furthermore, their integration with smartphones through wireless connectivity allows seamless data exchange, ensuring that users stay informed and connected without distractions.

REFERENCES

1. https://ijcrt.org/papers/IJCRT2102109.pdf
2. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3638846
3. https://www.irjet.net/archives/V7/i7/IRJET-V7I796.pdf
4. https://smsjournals.com/index.php/SAMRIDDHI/article/view/2313
5. https://www.instructables.com/DIY-Smart-Augmented-Reality-Glasses-Using-Arduino/
6. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3638846