Automatic Bottle filling Machine using Micro-controller

A Project report submitted in partial fulfillment
of the requirements for the degree of B. Tech in Electrical Engineering

by

DEBANTA CHATTERJEE (11701618050)
CHIRANTAN BHATTACHARYA (11701618051)
SOUGATA SENAPATI (11701618025)
DEBLEENA DAS (11701618047)

Under the supervision of

Mr. Budhaditya Biswas
Assistant Professor
Department of Electrical Engineering

Department of Electrical Engineering
RCC INSTITUTE OF INFORMATION TECHNOLOGY
CANAL SOUTH ROAD, BELIAGHATA, KOLKATA – 700015, WEST BENGAL
Maulana Abul Kalam Azad University of Technology (MAKAUT)

This work has been dedicated to the memory of our beloved teacher

Mr. Debobrata Bhattacharya

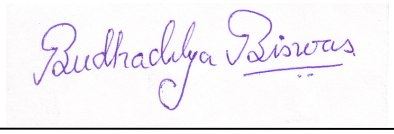
Professor, Applied Electronics & Instrumentation Engineering

Department of Electrical Engineering
RCC INSTITUTE OF INFORMATION TECHNOLOGY
CANAL SOUTH ROAD, BELIAGHATA, KOLKATA – 700015, WEST BENGAL
PHONE: 033-2323-2463-154, FAX: 033-2323-4668
Email: hodeercciit@gmail.com, Website: http://www.rcciit.org/academic/ee.aspx

CERTIFICATE

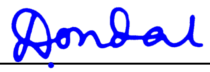
To whom it may concern

This is to certify that the project work entitled **Automatic Bottle filling Machine using Micro-controller** is the bonafide work carried out by **DEBANTA CHATTERJEE (11701618050), CHIRANTAN BHATTACHARYA (11701618051), SOUGATA SENAPATI (11701618025), DEBLEENA DAS (11701618047)**, the students of B.Tech in the Department of Electrical Engineering, RCC Institute of Information Technology (RCCIIT), Canal South Road, Beliaghata, Kolkata-700015, affiliated to Maulana Abul Kalam Azad University of Technology (MAKAUT), West Bengal, India, during the academic year 2021-22, in partial fulfillment of the requirements for the degree of Bachelor of Technology in Electrical Engineering and that this project has not submitted previously for the award of any other degree, diploma and fellowship.



(Budhaditya Biswas)
Assistant Professor
Department of Electrical Engineering
RCC Institute of Information Technology

Countersigned by



(Prof. Dr. Debasish Mondal)
HOD, Electrical Engineering Dept
RCC Institute of Information Technology

(External Examiner)

ACKNOWLEDGEMENT

It is our great fortune that we have got opportunity to carry out this project work under the supervision of **Mr. Budhaditya Biswas** in the Department of Electrical Engineering, RCC Institute of Information Technology (RCCIIT), Canal South Road, Beliaghata, Kolkata-700015, affiliated to Maulana Abul Kalam Azad University of Technology (**MAKAUT**), West Bengal, India. We express our sincere thanks and deepest sense of gratitude to our guide for his constant support, unparalleled guidance and limitless encouragement.

We would like to convey our gratitude to all the faculty members and staffs of the Department of Electrical Engineering, RCCIIT for their whole hearted cooperation to make this work turn into reality.

We are very thankful to **Mr. Nitai Banerjee** for his support and effort to build the shaft of the motor in the mechanical workshop.

We would also like to convey our gratitude to **Prof. Dr. Ashoke Mondal** to guide us to build the radio frequency shielding and the driver board to interface the DC pump.

We are very thankful to our department and to the authority of RCCIIT for providing all kinds of infrastructural facility towards the research work.

Thanks to the fellow members of our group for working as a team.

DEBANTA CHATTERJEE (11701618050) Debanta Chatterjee

CHIRANTAN BHATTACHARYA (11701618051) Chirantan Bhattacharya

SOUGATA SENAPATI (11701618025) Sougata Senapati

DEBLEENA DAS (11701618047) Debleena Das.

To

The Head of the Department
Department of Electrical Engineering
RCC Institute of Information Technology
Canal South Rd. Beliagahata, Kolkata-700015

Respected Sir,

In accordance with the requirements of the degree of Bachelor of Technology in the Department of Electrical Engineering, RCC Institute of Information Technology, we present the following thesis entitled "**Automatic Bottle filling Machine using Micro-controller**". This work was performed under the valuable guidance of Mr. Budhadtiya Biswas, Assistant Professor in the Dept. of Electrical Engineering.

We declare that the thesis submitted is our own, expected as acknowledge in the test and reference and has not been previously submitted for a degree in any other Institution.

Yours Sincerely,

DEBANTA CHATTERJEE (11701618050) Debanta Chatterjee

CHIRANTAN BHATTACHARYA (11701618051) Chirantan Bhattacharya

SOUGATA SENAPATI (11701618025) Sougata Senapati

DEBLEENA DAS (11701618047) Debleena Das.

Contents

List of Figures

List of Tables

ABBREVIATIONS AND ACRONYMS

OLED – Organic Light Emitting Diode
SoC **–** System on a chip
IC - Integrated Circuit
PCB – Printed Circuit Board
μC – Micro Controller
BJT - Bi-polar Junction Transistor
SPDT - Single Pole Double Throw
NO - Normally Open
NC - Normally Closed
COM – Common
LED - Light Emitting Diode
POT – Potentiometer
SMPS – Switch Mode Power Supply
ISM – Industrial, scientific and medical
USB – Universal serial bus
SPI – Serial Peripheral Interface
I^2C **–** Inter-Integrated Circuit
GPIO – General Purpose Input Output
API **–** Application Program Interface
SDA **–** Serial Data
SCL **–** Serial Clock

ABSTRACT

Automatic Bottle Filling Machines are most commonly used in beverages and soft drink industries. The field of automation has had a notable impact in a wide range of industries beyond manufacturing. Automation is the use of control systems and information technologies to reduce the need for human work in the production of goods and services. In the scope of industrialization, automation is a step beyond mechanization. Whereas mechanization provides human operators with machinery to assist them with the muscular requirements of work, automation greatly decreases the need for human sensory and mental requirements as well. Filling is a task carried out by a machine that packages liquid products such as cold drinks or water.

There are two ways to filling a bottle automatically. One using a conveyor belt using position sensor (generally IR sensors or Infrared sensors) and the use of the PLC. But that involves high cost. Second one using fixed position of bottles and the time calculation to place the bottle bellow the liquid pipe for filling it us. This eliminates the cost of position sensors, conveyor belt and PLC.

In this project a stepper motor is used to move the bottle which is controlled by the controller and time calculation is used to open the pump for filling up the bottle. A limit switch is used to find the initial position of the platform containing the bottle. This project is prototype to such machines used to fill the liquid in the bottle with a fixed quantity or fixed level, it eliminates the chances unevenness from bottle to bottle or inaccuracy which is a very common mistake while filling manually. These projects reduce the labour effort and make work more accurate and reliable.

CHAPTER 1

(Introduction)

1.1 INTRODUCTION

The project intends to design, implement and monitor an "**Automatic Bottle Filling Machine using Micro-Controller**" using position control mechanism with the help of Micro-Controller (ESP-32). A prototype has been developed to illustrate the project. In this project the limit switch sets the initial position and the stepper motor rotates for a fixed angle which is calculated as (Total angle of rotation) / (No. of bottles placed) and the time for filling liquid in the bottle is calculated previously. OLED displays the status of the bottle filling and buzzer generates an audio signal once each bottle is filled.

The developed prototype consists of the following main section.

- Rotating platform – responsibility is to carry the bottles and placed the bottle bellow the water pipe.
- Stepper motor – it moves the rotating platform
- Bearing and pully assembly – this part holds the rotating platform and synchronize with the movement of the stepper motor
- Control board – it consists of ESP32 microcontroller, stepper motor driver, relay driver, OLED, voltage regulator etc. The heart of the project. Controls the stepper motor, water pump and display all the information in the OLED.

1.2 SEQUENTIAL PROCESS CONTROL

A **Process Control** in continuous industrial production processes is a discipline that uses industrial control systems to achieve a production level of consistency, economy and safety which could not be achieved purely by human manual control. It is implemented widely in industries such as automotive, mining, dredging, oil refining, pulp and paper manufacturing, chemical processing

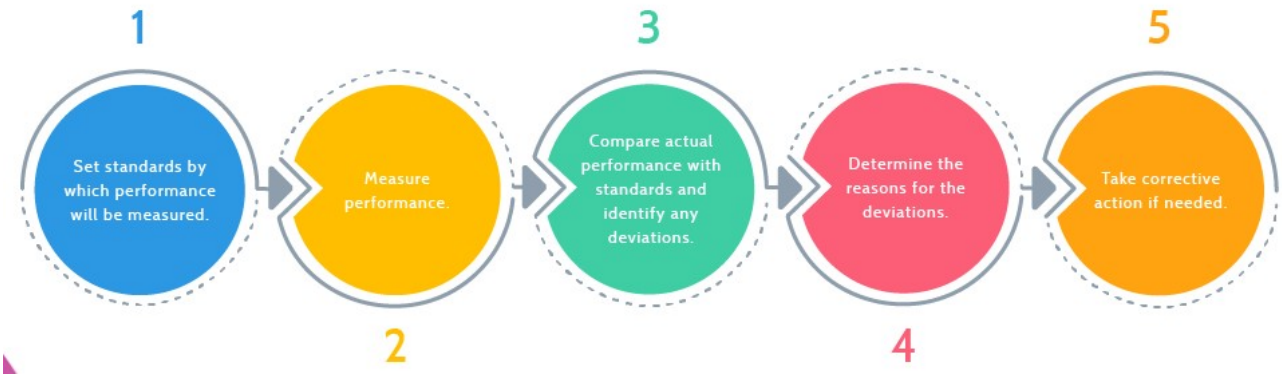


Figure 1: Concept of Industrial Process Control

and power generating plants. Now a control system in which the individual steps are processed in a predetermined order, progression from one sequence step to the next being dependent on defined conditions being satisfied, is a **Sequential Process Control**. Such a system is generally time-controlled, in which the step transition conditions are functions of time only or it may also be external-event dependent, where the conditions are functions of Input signals only or it also may be combinations of these (and perhaps more complex) conditions.

Early process control breakthroughs came most frequently in the form of water control devices. Later process controls inventions involved basic physics principles. With the dawn of the Industrial Revolution in the 1760s, process controls inventions were aimed to replace human operators with mechanized processes.

Process control of large industrial plants has evolved through many stages. Initially, control would be from panels local to the process plant. However, this required a large manpower resource to attend to these dispersed panels, and there was no overall view of the process. The next logical development was the transmission of all plant measurements to a permanently-manned central control room. Effectively this was the centralization of all the localized panels, with the advantages of lower manning levels and easier overview of the process. With the coming of electronic processors, graphic displays and different communication protocols it became possible to replace these discrete controllers with computer-based algorithms, hosted on a network of input/output racks with their own control processors and also to replace wired control with wireless system.

Now generally process control is associated with industrial process industries where batch control of repeated processes is required. A great example of process control is temperature control during any industrial process. Different temperatures are maintained for different times spans according to settings. This also can be done manually but that will require large man power and time and will consist of heavy inaccuracy. Using this microcontroller based automated system, this can be done effective from anywhere at any time with great accuracy.

1.3 Overview and benefits

Maintained measurement and control in manufacturing processes helps facilitate a business overall success. That’s easier said than done, though. Overseeing the regulation of a large variety of processes can be extremely overwhelming. That’s where the implementation of process control instrumentation comes in.

The automatic water bottle filling mainly developed using PLC and conveyor belt. It includes much cost. The same can be developed using synchronized process control which eliminates the use of the sensors and conveyor belt. The cost also reduces much in this process.

Although process control technology has advanced rapidly since the mid-1980s, the latest systems still follow the traditional hierarchical or pyramid-like structure. The lowest level of the pyramid works to make sure a particular process doesn't vary by more than an allowed amount. It monitors the operation of each part of the process, identifies unwanted changes and initiates any necessary corrective actions. Lower-level controls can't handle complex situations like equipment faults. These have to be dealt with either manually, by an operator, or by other controls at a higher level of the hierarchy. Further up the pyramid the system controls the overall production process and makes sure it continues to operate efficiently.

Process control systems are central to maintaining product quality. Using proper instrumentation, control systems maintain the proper ratio of ingredients. Without this standard of control, products would vary and quality would be impaired. With improved quality comes higher levels of safety too. The process control systems automatically warn you of any abnormalities which minimizes the risk of accidents. By shifting focus to cost-effective and objective-reaching technologies, the ability to take on more work will increase significantly.

1.4 Organisation of thesis

The thesis is organised into seven chapters including the chapter of introduction. Each chapter is different from the other and is described along with the necessary theory required to comprehend it.

Chapter 2 deals with the literature reviews. From this chapter we can see before our project who else works on this topic and how our project is different and advance from those projects.

Chapter 3 deals with the theory required to do the project. The basic of process control with microcontroller and the interfacing of the stepper motor, OLED and pump control are described here. The overview of the project and software simulation of the project is also listed in this chapter.

Chapter 4 deals with the hardware modelling of the projects. The main features, photographs, step by step operation of the prototype, component listing and the hardware interfacing of the required components are described here.

Chapter 5 describes the basic operation of the circuit. A flow chart is presented on the actions that would take in the controller beginning from the positioning of the bottles and filling it. Advantages and disadvantages and cost estimation are listed in this chapter.

Chapter 6 concludes the work performed so far. The possible limitations in proceeding research towards this work are discussed. The future work that can be done in improving the current scenario is mentioned. The future potential along the lines of this work is also discussed.

Chapter 7 References are listed in this chapter

Appendix A, B & C Hardware description, software coding and datasheets are listed here.

CHAPTER 2

(Literature Review)

[1] Deepika Saikia, Prajakta Powar, Animesh Gaurav "Automatic Bottle Filling System", International Research Journal of Engineering and Technology (IRJET), e-ISSN: 2395-0056, p-ISSN: 2395-0072

Liquid filling machine is capable of incorporating a tank, bottle or container to fill the liquid. Liquid filling machines are important equipment in various industries like cosmetics, food, etc. where liquids are to be packed in various types of containers. This machine helps in reducing wastage and can be easily and efficiently packed into containers that to very fast. In our proposed technology we have designed an automatic liquid filling machine which will work on flow meters. We have use microcontroller, flow meter, solenoid valve, stepper motor, and proximity sensor and conveyor belt. The paper includes working of the machine, simulation result and also the scope. Our project is a combination of electronics and mechanical work and it is used in industrial production and it is also suitable for small scale industries.

[2] Ashwini P. Somawanshi, Supriya B. Asutkar, Sachin A. More "Automatic Bottle Filling Using Microcontroller Volume Correction", International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181, Vol. 2 Issue 3, March – 2013

The field of automation has had a notable impact in a wide range of industries beyond manufacturing. Automation is the use of control systems and information technologies to reduce the need for human work in the production of goods and services. In the scope of industrialization,
automation is a step beyond mechanization. Whereas mechanization provides human operators with machinery to assist them with the muscular requirements of work, automation greatly decreases the need for human sensory and mental requirements as well. Filling is a task carried out by a machine that packages liquid products such as cold drinks or water. The bottle filling project serves as an interdisciplinary engineering design experience. It introduces aspects of computer, electronics and mechanical engineering, including the following five primary knowledge areas: 1) Machining & Fabrication 2) Electronics circuit prototyping and Programming 3) Sensor and Actuator application 4) Mechanical design 5) Project Planning 6) Presentation Skills.

[3] Aniruddh Guha, Adarsh Ganveer, Manjari Kumari, Ajay Singh Rajput, "AUTOMATIC BOTTLE FILLING MACHINE", International Research Journal of Engineering and Technology (IRJET), e-ISSN: 2395-0056, p-ISSN: 2395-0072

The world is increasing moving towards Automation, i.e., the process of performing various tasks without or with minimum human intervention. It increases the overall efficiency and output of a process. It involves establishing control loops using microcontrollers like Arduino or PLCs, which control the working of the entire plant. Filling is an operation in which a predetermined amount of liquid needs to be precisely filled in the bottle. It is used by soft drinks industry, packaged water industry and various pharmaceuticals. The operation was earlier carried out by humans and involved placing one bottle at a time on the conveyor belt and filling it. The process then was slow, involved spilling of liquid and unequal quantities of liquid in bottles. The process is now carried out by PLCs in large manufacturing units now. PLC machines are very expensive. Due to their high costs, filling is still carried out manually in small manufacturing units. This results in shortcomings in the operation and at drives up labour costs. This problem compels us to design a system with reduced costs. This can be achieved by using Arduino as a microcontroller. The proposed project will reduce cost for small scale industries and help them in setting up automated plants.

[4] J.Dharanidharan, R. Puviarasi, "AUTOMATIC BOTTLE FILLING MACHINE", International Conference on Recent Trends in Computing, Communication and Networking Technologies (ICRTCCNT'19) Oct 18-19, 2019, Kings Engineering College, Chennai, TamilNadu, India.

The field of automation includes a notable impact in a very wide selection of industries on the far side producing. Within the standard technique main half is system which has 'C' program, Arduino or microcontroller to regulate entire system. The matter arrested in manual filling method like spilling of water whereas filling it in bottle, equal amount of water might not be crammed, delay thanks to natural activities could happens. To rectify the higher than mentioned issues the projected system in designed. In projected system the advanced technology of PLC is employed. With this technique that operates mechanically, each method may be swish and therefore the method of replenishment will scale back employee price and operation price. with mounted hard and fast set amount or fixed level, it eliminates the possibilities like varied amount from bottle to bottle or quality which might be happen with the manually filling. This project scale back labour impact and build additional correct and reliable.

[5] Arthur Pius Santiago [2010]: Insecticide Bottle Filling and Capping machines in De La Salle University. The 11th Asia pacific Industrial Engineering and Management Systems Conference. The 14th Asia Pacific Regional Meeting of the International Foundation for production Research.

Insecticide Bottle Filling and Capping Machines have been one of the industry related projects in the Manufacturing Engineering and Management Department of De La Salle University Manila. The initial attempt was supported by Mapecon Philippines, Inc., a private business entity which formulates its own insecticide. This paper presents machine prototypes to replace the manual process. The prototypes shown were able to consistently fill and cap 500ml bottles of Big R Insecticide. Experiments were conducted to determine the production rate of the prototypes as well as to test whether the volume of insecticide dispensed are within company specified limits.

[6] Samarth Nainani, Akshata Rupawate, Shoaib Sayyed, Siddharth Poojary, Vaishali Bodhale, "AUTOMATIC BOTTLE FILLING SYSTEM USING ARDUINO UNO", International Research Journal of Modernization in Engineering Technology and Science, Volume:03/Issue:04/April-2021, e-ISSN: 2582-5208

The current state of industries is to embrace new technologies to proceed towards automation. The identical vision is exercised in bottle-filling plants. To help small-scale industries all operations are nearly automated. And in small-scale industries, the operations are still carried out by humans which involved some imperfections. The automation of bottle filling involves the use of PLC which are used in large-scale industries and are very costly. The study emphasizes on reduction in cost using the Arduino micro-controller. The manual filling process has many problems like spilling water while filling it in a bottle, etc. This work generally emphasizes small industries and we aim to make these small-scale industries more efficient and to eliminate problems faced by small-scale bottle filling industries. With this technique that operates automatically, every process can be smooth and the process of refilling will cut back the hands price and operation time.

[7] Kiran Chaudhari, Aaqib Momin, Ritu Magare, Omkar More, Aniket Kantale, "Automated Filling Machine", International Research Journal of Engineering and Technology (IRJET), e-ISSN: 2395-0056, p-ISSN: 2395-0072.

India: a country with population of over a billion, the demand for food and other daily necessities in people's everyday life keeps on rising sharply. The world is modernizing at a rapid rate and the current COVID-19 pandemic situation has resulted in entrepreneurs becoming more technology oriented over labour force. This brings up the need for more automation in every sector where there is involvement of human work force be it small scale or an industry. This research project focuses on reducing the customer to vendor gap by eliminating the involvement of labour at small grocery or other enterprises.

[8] Bipin Mashilkar, Pallavi Khaire and Girish Dalve, "Automated Bottle Filling System," International Research Journal of Engineering and Technology (IRJET), e-ISSN: 2395-0056, p-ISSN: 2395-0072; Volume 02, Issue 07; October 2015.

The field of automation has a notable impact in a wide range of industries beyond manufacturing. Automation plays an increasingly important role in the world economy. Filling is a task carried out by a machine that packages liquid products such as cold drinks or water. In past, humans were the main method for controlling a system. More recently, electricity has been used for control and electrical control is based on microcontrollers for various purposes like medicines, pharmaceutical plants, chemical plants etc. There microcontrollers control the complete working of the system. It is common to use microcontrollers to make simple logical control decision. The automation in bottle filling industry comes with increased electrical components. Essential requirements of each component in the system is important to be studied in ordered to understand how each part works in coordination with other parts in the system. This study mainly includes design, fabrication and control system for automated bottle filling system. The main part is control system which includes embedded C programming in Arduino microcontroller to control various components in system. A conveyor system with sensors and electromagnetic valve is fabricated for this purpose. The entire sequence of operation is controlled by Arduino microcontroller.

[9] Nisarg A Solanki, Pratik G Raj, Saumil P Patel, Charmish D Rajput, "Automatic Liquid Filling Machine", International Journal of Engineering Research & Technolog

Liquid filling machines are equipment used for packaging of various liquid products, mainly food and cold drinks. Depending on the different products, the different containers to be filled can either be a bottle or bag. These machines are usually found in manufacturing industry to promote quality and efficiency on the manufacturing process. In our proposed technology we suggest automatic liquid filling machine which will work on gear pump. Gear pump will be synchronized with encoder will give command to rotate particular rotation and hence pump will deliver particular volume. Pump will be connected with nozzle to transfer material into bottles. Volume setting from one size to another size will be done by changing command to gear pump. Once it is calibrated, volume setting will be done in seconds. So it will give more production and will save lot of manpower.

CHAPTER 3

(Theory)

3.1 Microcontroller (MCU):

A microcontroller is a compact integrated circuit designed to govern a specific operation in an embedded system. A typical microcontroller includes a processor, memory and input/output (I/O) peripherals on a single chip.

3.1.1 How do microcontrollers work?

A microcontroller is embedded inside of a system to control a singular function in a device. It does this by interpreting data it receives from its I/O peripherals using its central processor. The temporary information that the microcontroller receives is stored in its data memory, where the processor accesses it and uses instructions stored in its program memory to decipher and apply the incoming data. It then uses its I/O peripherals to communicate and enact the appropriate action.

Microcontrollers are used in a wide array of systems and devices. Devices often utilize multiple microcontrollers that work together within the device to handle their respective tasks.

3.1.2 What are the elements of a microcontroller?

The core elements of a microcontroller are:

1. The processor (CPU) -- A processor can be thought of as the brain of the device. It processes and responds to various instructions that direct the microcontroller's function. This involves performing basic arithmetic, logic and I/O operations. It also performs data transfer operations, which communicate commands to other components in the larger embedded system.
2. Memory -- A microcontroller's memory is used to store the data that the processor receives and uses to respond to instructions that it's been programmed to carry out. A microcontroller has two main memory types:
3. Program memory, which stores long-term information about the instructions that the CPU carries out. Program memory is non-volatile memory, meaning it holds information over time without needing a power source.
4. Data memory, which is required for temporary data storage while the instructions are being executed. Data memory is volatile, meaning the data it holds is temporary and is only maintained if the device is connected to a power source.

- I/O peripherals -- The input and output devices are the interface for the processor to the outside world. The input ports receive information and send it to the processor in the form of binary data. The processor receives that data and sends the necessary instructions to output devices that execute tasks external to the microcontroller.

Other supporting elements of a microcontroller include:

- Analog to Digital Converter (ADC) -- An ADC is a circuit that converts analog signals to digital signals. It allows the processor at the center of the microcontroller to interface with external analog devices, such as sensors.
- Digital to Analog Converter (DAC) -- A DAC performs the inverse function of an ADC and allows the processor at the center of the microcontroller to communicate its outgoing signals to external analog components.

- System bus -- The system bus is the connective wire that links all components of the microcontroller together.
- Serial port -- The serial port is one example of an I/O port that allows the microcontroller to connect to external components. It has a similar function to a USB or a parallel port but differs in the way it exchanges bits.

3.1.3 Microcontroller features

- A microcontroller's processor will vary by application. Options range from the simple 4-bit, 8-bit or 16-bit processors to more complex 32-bit or 64-bit processors. Microcontrollers can use volatile memory types such as random access memory (RAM) and non-volatile memory types -- this includes flash memory, erasable programmable read-only memory (EPROM) and electrically erasable programmable read-only memory (EEPROM).
- Generally, microcontrollers are designed to be readily usable without additional computing components because they are designed with sufficient onboard memory as well as offering pins for general I/O operations, so they can directly interface with sensors and other components.

3.1.4 Microcontroller applications

Microcontrollers are used in multiple industries and applications, including in the home and enterprise, building automation, manufacturing, robotics, automotive, lighting, smart energy, industrial automation, communications and internet of things (IoT) deployments.

One very specific application of a microcontroller is its use as a digital signal processor. Frequently, incoming analog signals come with a certain level of noise. Noise in this context means ambiguous values that cannot be readily translated into standard digital values. A microcontroller can use its ADC and DAC to convert the incoming noisy analog signal into an even outgoing digital signal.

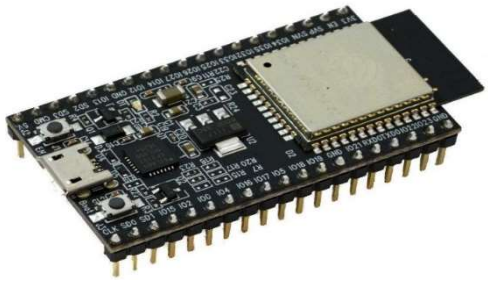


Figure 2: ESP 32 Microcontroller

3.1.5 Microcontrollers vs. microprocessors

The distinction between microcontrollers and microprocessors has gotten less clear as chip density and complexity has become relatively cheap to manufacture and microcontrollers have thus integrated more "general computer" types of functionalities. On the whole, though, microcontrollers can be said to function usefully on their own, with a direct connection to sensors and actuators, where microprocessors are designed to maximize compute power on the chip, with internal bus

connections (rather than direct I/O) to supporting hardware such as RAM and serial ports. Simply put, coffee makers use microcontrollers; desktop computers use microprocessors.

3.2 ESP32 microcontroller

ESP32 is created by Espressif Systems with a series of SoC (System on a Chip) and modules which are low cost with low power consumption.

This new ESP32 is the successor to the well-known ESP8266(became very popular with its inbuilt WiFi). ESP32 not only has Built in WiFi but also has Bluetooth and Bluetooth Low Energy. In other words, we can define ESP32 as "ESP8266 on Steroids".

ESP32 chip ESP32-D0WDQ6 is based on a Tensilica Xtensa LX6 dual core microprocessor with an operating frequency of up to 240 MHz.

The small ESP32 package has a high level of integrations such as:

- Antenna switches
- Balun to control RF
- Power amplifier
- Low noise reception amplifier
- Filters and power management modules

On top of all that, it achieves very low power consumption through power saving features including clock synchronization and multiple modes of operation. The ESP32 chip's quiescent current is less than 5 μA which makes it the ideal tool for your battery powered projects or IoT applications.

3.2.1 ESP32 Functional Blocks and Features

Although in the previous table you can notice some main technical characteristics of the ESP32, the truth is not everything is in the table. In fact, many details are missing. To get to know all the features of this **magnificent SoC** it is necessary to refer

- ESP32 Technical Datasheet
- ESP32 Technical Reference Manual

3.2.2 ESP32 Architectural Block diagram

Below is the **Architectural block diagram of ESP32** which shows all the **functional blocks of ESP32** SOC.

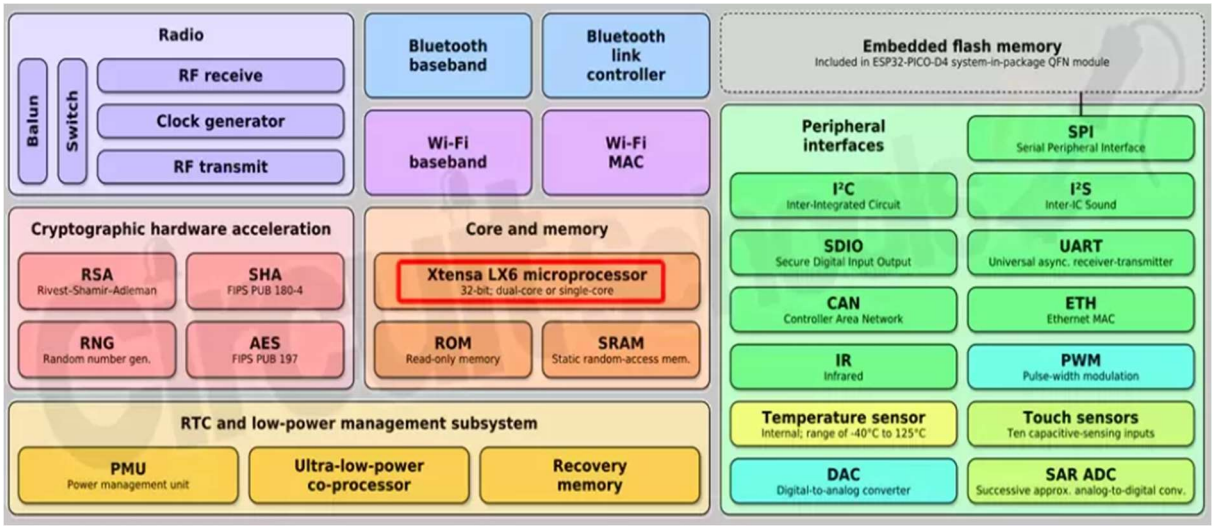


Figure 3: ESP32 Architectural Block diagram

3.2.3 ESP32 Core

As we have already mentioned that the ESP32 has **dual core low-power Tensilica Xtensa 32-bit LX6** microprocessors.

Memory

In most of the microcontrollers based on Arduino, there are three types of memories:

- Program memory: to store the sketch.
- SRAM memory: to store the variables that are used in the code.
- EEPROM memory: to store variables that do not lose their value even when the device is turned off.

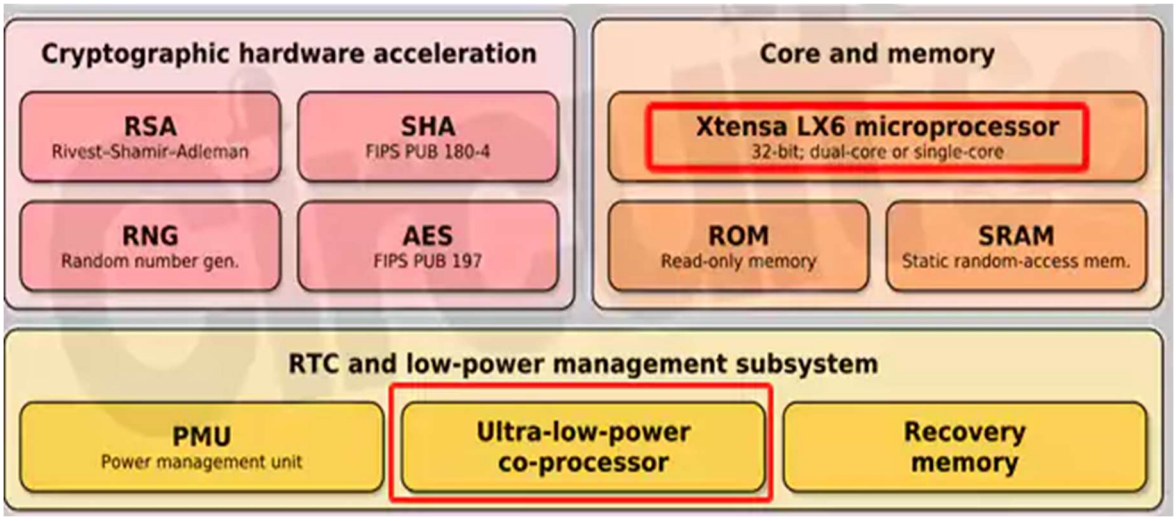


Figure 4: ESP32 memory Block diagram

It can be observed from the above core block image, it has an **ultra-low-power co-processor** that is used to perform **analog-digital conversions** and other operations while the device is operating in deep sleep low-power mode. In this way, a very low consumption by the SoC is achieved.

It is important to note that these processors offer great typical advantages of a digital signal processor:

- Operating frequency: 240 MHz (executes instructions 15 times faster than an Arduino UNO board)
- It allows to perform operations with real numbers (numbers with commas) very efficiently.
- Allows you to multiply large numbers instantly.

In ESP32 this does not happen, in fact there are more types of memories that are usually classified into internal and external.

The internal memories are those that are already included in the SoC, and the external are those that can be added to expand the capacity of the system.

Many ESP32- based development boards add external memory for a better performing system.

3.2.4 ESP32 Internal memories and their functions:

- ROM memory (448 KiB): this memory is write-only, that is, you cannot reprogram it. This is where the codes that handle the Bluetooth stack, the Wi-Fi physical layer control, some general-purpose routines, and the bootloader to start the code from external memory are stored.
- Internal SRAM memory (520 KiB): this memory is used by the processor to store both data and instructions. Its advantage is that it is much easier for the processor to access than the external SRAM.
- RTC SRAM (16 KiB): this memory is used by the co-processor when the device operates in deep sleep mode.
- Efuse (1 Kilobit): 256 bits of this memory are used by the system itself and the remaining 768 bits are reserved for other applications.
- Flash embedded (Embedded flash): This memory is where our application code is stored. The amount of memory varies depending on the chip used:
  0 MB (chips ESP32-D0WDQ6, ESP32-D0WD, ESP32-S0WD)
  2 MB (chip ESP32-D2WD)
  4 MB (Chip ESP32-PICO-D4)

For ESP32s that do not have embedded memory or simply when memory is insufficient for your application, it is possible to add more memory externally:

- Up to 16 MB of external flash memory can be added. This way you can develop more complex applications.
- It also supports up to 8 MB of external SRAM memory.

Therefore, it is difficult for you to find yourself limited in memory when implementing an application using this platform.

3.2.5 ESP32 Pinout diagram and Pins

It can be seen from the above image of **ESP32 WROOM module pinout diagram**, all the different types of pins are mentioned in different colors which we are going to explain in detail below.

Digital pins

The ESP32 has a total of 34 digital pins. These pins are similar to Arduino digital pins which allows you to add LED display, OLED display, sensors, buttons, buzzers, etc. to our projects.

Most of these pins support the use of internal pull-up, pull-down, and high impedance status as well. This makes them ideal for connecting buttons and matrix keyboards, as well as for applying LED control techniques such as the well-known Charlieplexing.

ESP32 WROOM module has 25 GPIO pins out of which there are only input pins, pins with input pull up and pins without internal pullup.

Maximum current drawn per a single GPIO is 40mA according to the "Recommended Operating Conditions" section in the ESP32 datasheet.

Input only pins:

- GPIO 34
- GPIO 35
- GPIO 36
- GPIO 39

Pins with pull up INPUT_PULLUP

- GPIO14
- GPIO16
- GPIO17
- GPIO18
- GPIO19
- GPIO21
- GPIO22
- GPIO23

Pins without internal pull up

- GPIO13
- GPIO25
- GPIO26
- GPIO27
- GPIO32
- GPIO33

ADC (Analog to digital converters)

Some of the pins listed in the pinout diagram can also be used to interact with analog sensors, same as analog pins of an Arduino board.

For this, the ESP32 has a 12-bit (0-4096 resolution which means when voltage observed is 0 the value is **0** and when max voltage like **3.3v** is observed the value goes to 4096), 18-channel analog to digital converter, which means you can take readings from up to 18 analog sensors.

This allows you to develop very compact connected applications, even when using multiple analog sensors.

Analog input pins:

- ADC1_CH0 (GPIO 36)
- ADC1_CH1 (GPIO 37)
- ADC1_CH2 (GPIO 38)
- ADC1_CH3 (GPIO 39)
- ADC1_CH4 (GPIO 32)
- ADC1_CH5 (GPIO 33)
- ADC1_CH6 (GPIO 34)
- ADC1_CH7 (GPIO 35)
- ADC2_CH0 (GPIO 4)
- ADC2_CH1 (GPIO 0)
- ADC2_CH2 (GPIO 2)
- ADC2_CH3 (GPIO 15)
- ADC2_CH4 (GPIO 13)
- ADC2_CH5 (GPIO 12)
- ADC2_CH6 (GPIO 14)
- ADC2_CH7 (GPIO 27)
- ADC2_CH8 (GPIO 25)
- ADC2_CH9 (GPIO 26)

DAC (Digital to Analog Converters)

PWM signals are used on most Arduino boards to generate analog voltages. The ESP32 has two 8 bits digital to analog converters.

This allows two pure analog voltage signals to be generated. These converters can be used to:

- Control an analog circuit
- Manipulate the intensity of an LED
- Can even add a small amp and speaker to your project to play a song.

DAC Pins:

- DAC1 (GPIO25)
- DAC2 (GPIO26)

Capacitive Touch GPIOs

In case if somebody wants to develop applications with **no mechanical buttons**, they can use the touch sensitive pins on ESP32s to achieve it.

These pins are capable of detecting the small variations produced when approaching a finger to the pin. In this way, it is possible to create all kinds of controls such as buttons or slide bars without the need for mechanical components.

Capacitive Touch pins:

- T0 (GPIO 4)
- T1 (GPIO 0)

- T2 (GPIO 2)
- T3 (GPIO 15)
- T4 (GPIO 13)
- T5 (GPIO 12)
- T6 (GPIO 14)
- T7 (GPIO 27)
- T8 (GPIO 33)
- T9 (GPIO 32)

RTC

As we already learnt about the RTC GPIO support in the core section. The GPIOs which are routed to the **RTC low-power management subsystem** can be used when the ESP32 is in deep sleep. These RTC GPIOs can be used to wake up the ESP32 from deep sleep when the **Ultra-Low Power (ULP) co-processor** is running. The following GPIOs can be used as an external wake up source.

- RTC_GPIO0 (GPIO36)
- RTC_GPIO3 (GPIO39)
- RTC_GPIO4 (GPIO34)
- RTC_GPIO5 (GPIO35)
- RTC_GPIO6 (GPIO25)
- RTC_GPIO7 (GPIO26)
- RTC_GPIO8 (GPIO33)
- RTC_GPIO9 (GPIO32)
- RTC_GPIO10 (GPIO4)
- RTC_GPIO11 (GPIO0)
- RTC_GPIO12 (GPIO2)
- RTC_GPIO13 (GPIO15)
- RTC_GPIO14 (GPIO13)
- RTC_GPIO15 (GPIO12)
- RTC_GPIO16 (GPIO14)
- RTC_GPIO17 (GPIO27)

SD / SDIO / MMC driver

This peripheral allows the ESP32 to interact with SD and MMC cards directly. In fact, by combining this controller with the analog digital converter it is possible to improve our little audio player.

UART

Many microcontrollers have UART modules, which on Arduino are known as Serial ports. These allow asynchronous communications between two devices using only two pins.

The ESP32 has three UART ports:

- UART0
- UART1

- UART2

All of these are compatible with RS-232, RS-485 and IrDA protocols.

I2C

The ESP32 have two interfaces I2C or TWI that support the operating modes master and slave. Its features include:

- Standard mode (100 Kbit/s)
- Fast mode (400 Kbit/s)
- 7 and 10 bit addressing

I2C Pins

- GPIO 21 (SDA)
- GPIO 22 (SCL)

SPI

The ESP32 also has SPI communication. It has three fully functional buses:

- Four transfer modes: this means that it is compatible with all or almost all SPI and QSPI devices available on the market.
- All SPI ports are capable of high speeds (theoretically up to 80 MHz).
- 64-byte buffer for transmission and reception.

By default, the pin mapping for SPI is:

Table1: SPI pin mapping

SPI	MOSI	MISO	CLK	CS
VSPI	GPIO 23	GPIO 19	GPIO 18	GPIO 5
HSPI	GPIO 13	GPIO 12	GPIO 14	GPIO 15

Infrared remote controller

The ESP32 also allows the transmission and reception of signals using various infrared protocols (the same as those used by the television remote).

Therefore, you can also use your ESP32 to create your own remote control that allows you to interact with your TV or your stereo.

PWM

Like the ESP8266, the ESP32 also supports the use of analog outputs using PWM. The big difference is in ESP32 it is possible to use up to 16 pins as PWM outputs where ESP8266 only supports 8 and Arduino UNO board that only supports 6.

PWM pins:

All the PWM pins are indicated with the below symbol in the ESP32 Pinout Diagram above.

3.2.6 How to select an ESP32 development board?

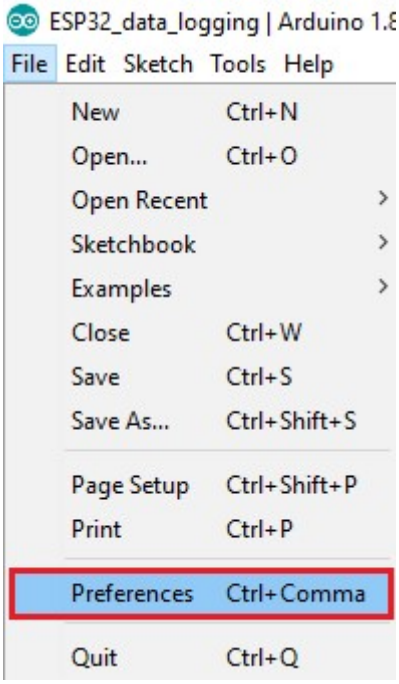
Before selecting an ESP32 development board, you need to take into account certain aspects:

- **Pin numbers and configuration:** it is important to have access to the board's pinout in order to make correct use of it.
- **Serial -USB interface and voltage regulator**: These two features are found in practically all development boards. These are the ones that allow the board to be connected directly to the computer to be energized and programmed.
- **Battery connector**: if you are thinking of venturing into low-consumption systems with batteries, you can opt for boards that already include battery connectors.
- **Extra functions**: many development boards for ESP32 come with extra features such as cameras, OLED displays, LoRa modules, etc.

3.3 Installing ESP32 Add-on in Arduino IDE

To install the ESP32 board in your Arduino IDE, follow these next instructions:

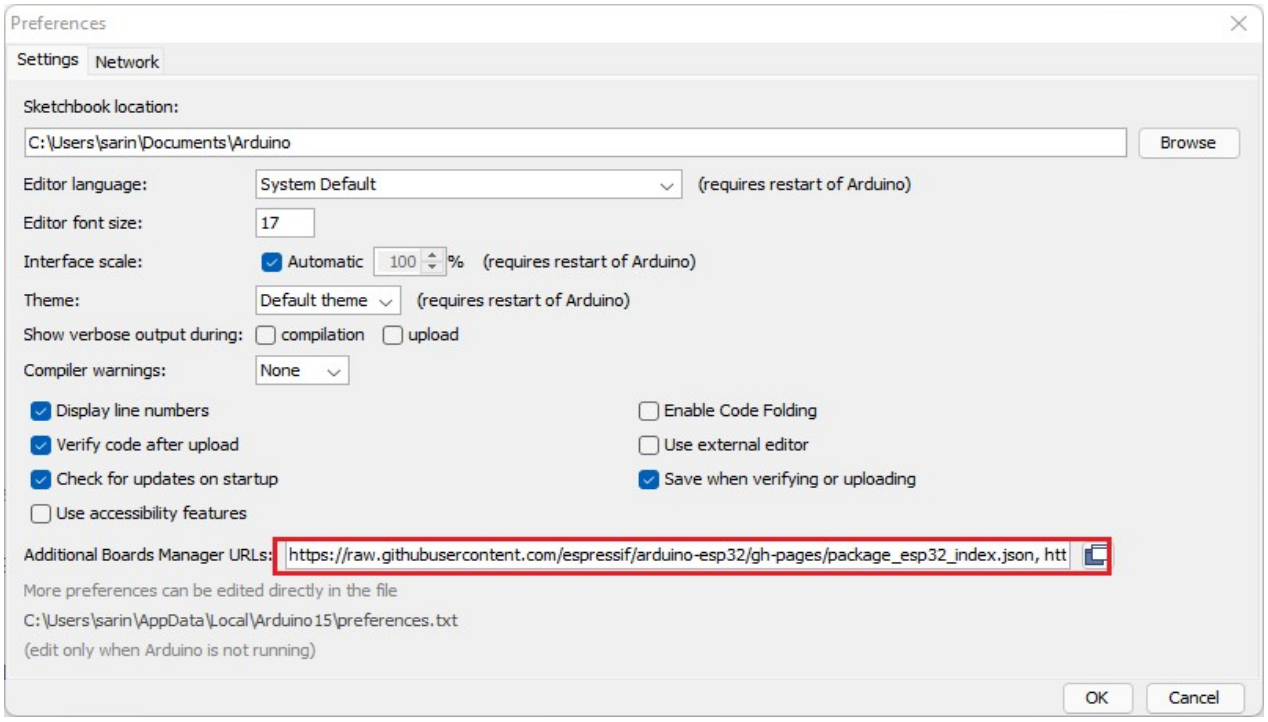
1. In your Arduino IDE, go to **File**> **Preferences**



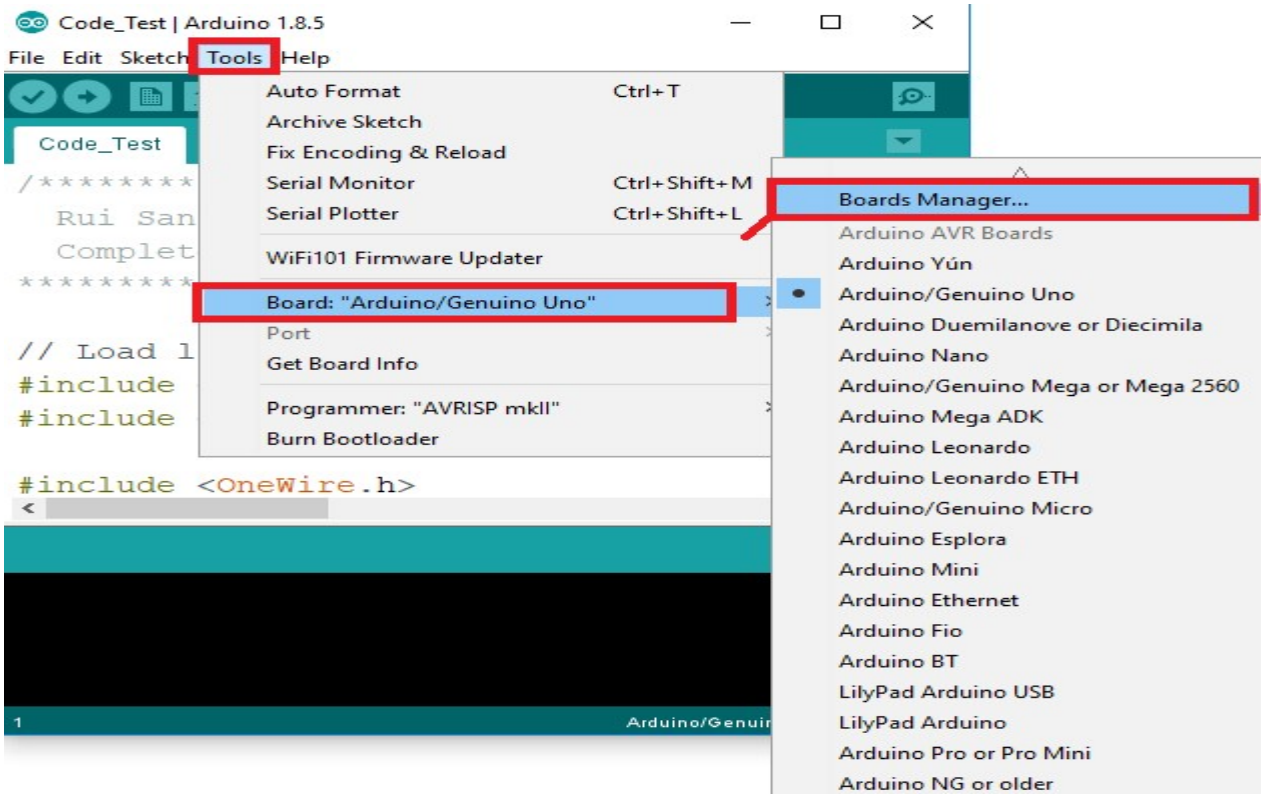
2. Enter the following into the "Additional Board Manager URLs" field:

   https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json

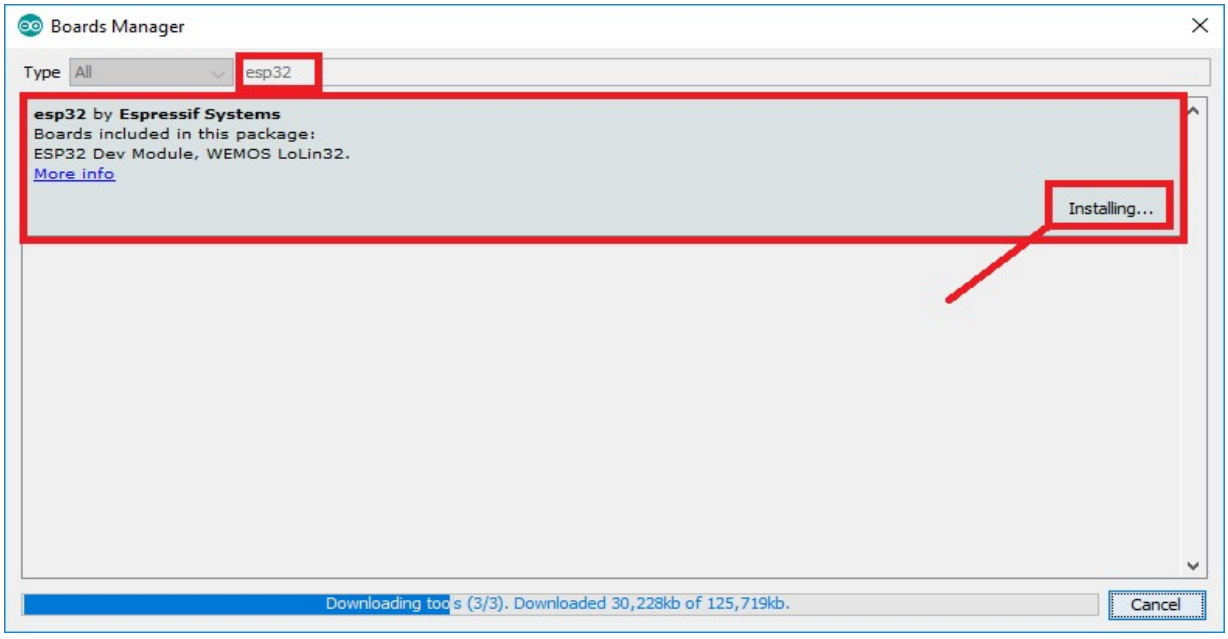
   Then, click the "OK" button:



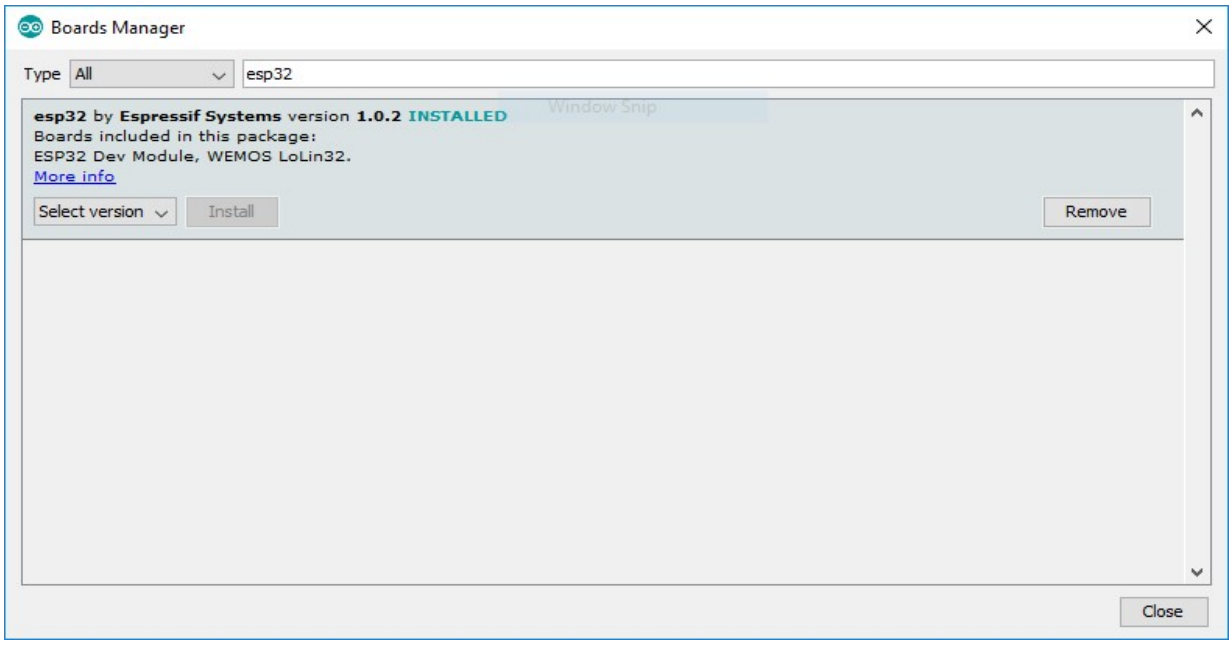
3. Open the Boards Manager. Go to **Tools** > **Board** > **Boards Manager…**



4. Search for **ESP32** and press install button for the "**ESP32 by Espressif Systems**":



5. That's it. It should be installed after a few seconds.



3.4 Stepper Motor Basics

A stepper motor is an electric motor whose main feature is that its shaft rotates by performing steps, that is, by moving by a fixed number of degrees. This feature is obtained thanks to the internal structure of the motor, and allows to know the exact angular position of the shaft by simply counting how may steps have been performed, with no need for a sensor. This feature also makes it fit for a wide range of applications.

3.4.1 Stepper Motor Working Principles

As all with electric motors, stepper motors have a stationary part (the stator) and a moving part (the rotor). On the stator, there are teeth on which coils are wired, while the rotor is either a permanent magnet or a variable reluctance iron core. We will dive deeper into the different rotor structures later. Figure 5 shows a drawing representing the section of the motor is shown, where the rotor is a variable-reluctance iron core.

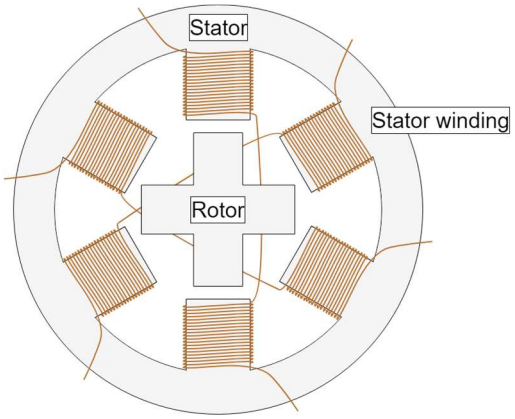


Figure 5: Cross-Section of a Stepper Motor

The basic working principle of the stepper motor is the following: By energizing one or more of the stator phases, a magnetic field is generated by the current flowing in the coil and the rotor aligns with this field. By supplying different phases in sequence, the rotor can be rotated by a specific amount to reach the desired final position. Figure 6 shows a representation of the working principle. At the beginning, coil A is energized and the rotor is aligned with the magnetic field it produces. When coil B is energized, the rotor rotates clockwise by 60° to align with the new magnetic field. The same happens when coil C is energized. In the pictures, the colors of the stator teeth indicate the direction of the magnetic field generated by the stator winding.

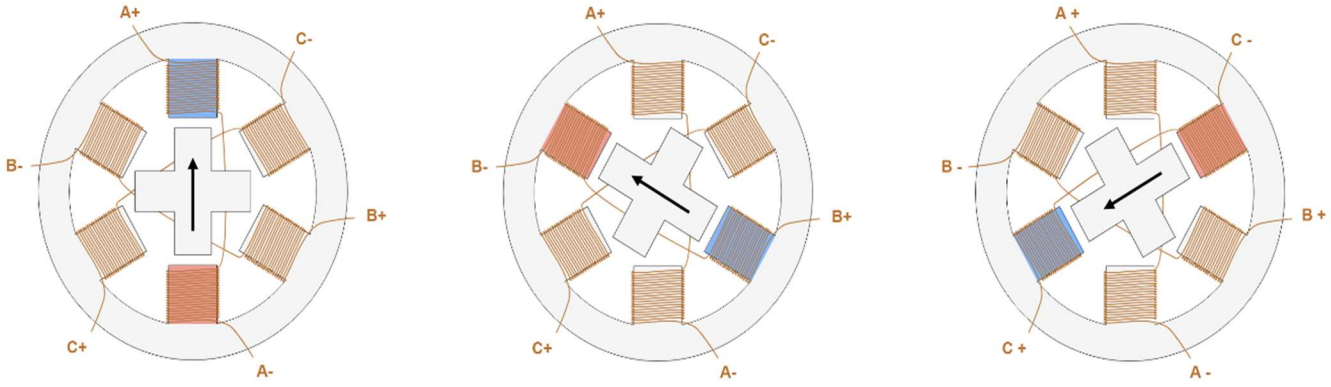


Figure 6: Stepper Motor Steps

3.4.2 Stepper Motor Control

We have seen previously that the motor coils need to be energized, in a specific sequence, to generate the magnetic field with which the rotor is going to align. Several devices are used to supply the necessary voltage to the coils, and thus allow the motor to function properly. Starting from the devices that are closer to the motor we have:

- A transistor bridge is the device physically controlling the electrical connection of the motor coils. Transistors can be seen as electrically controlled interrupters, which, when closed allow the connection of a coil to the electrical supply and thus the flow of current in the coil. One transistor bridge is needed for each motor phase.
- A pre-driver is a device that controls the activation of the transistors, providing the required voltage and current, it is in turn controlled by an MCU.
- An MCU is a microcontroller unit, which is usually programmed by the motor user and generates specific signals for the pre-driver to obtain the desired motor behavior.

Figure 7 shows a simple representation of a stepper motor control scheme. The pre-driver and the transistor bridge may be contained in a single device, called a **driver**.

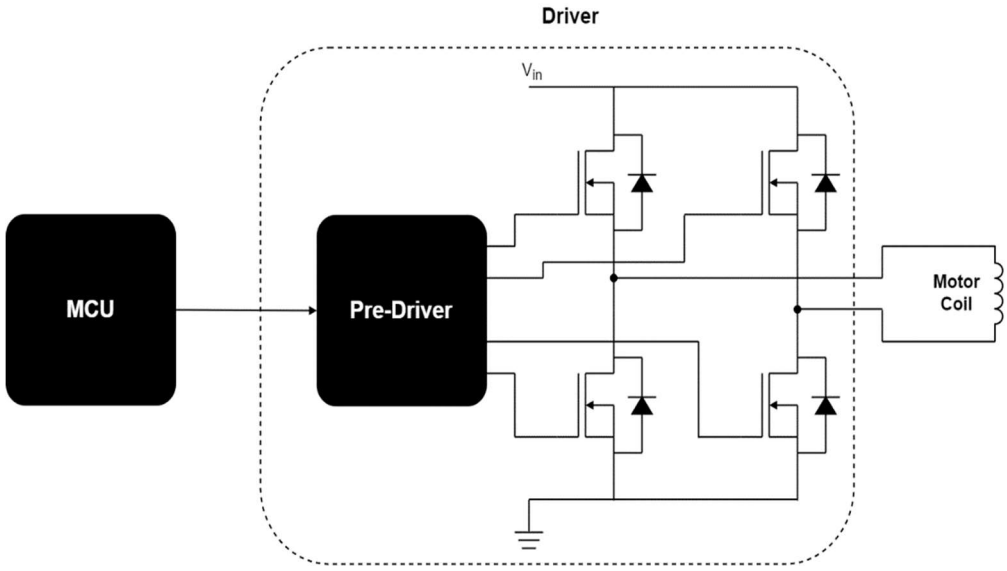


Figure 7: Motor Control Basic Scheme

3.4.3 Stepper Motor Driver Types

There are different stepper motor drivers available on the market, which showcase different features for specific applications. The most important characteristics include the input interface. The most common options are:

- Step/Direction – By sending a pulse on the Step pin, the driver changes its output such that the motor will perform a step, the direction of which is determined by the level on the Direction pin.
- Phase/Enable – For each stator winding phase, Phase determines the current direction and triggers Enable if the phase is energized.
- PWM – Directly controls the gate signals of the low-side and high-side FETs.

Another important feature of a stepper motor driver is if it is only able to control the voltage across the winding, or also the current flowing through it:

- With voltage control, the driver only regulates the voltage across the winding. The torque developed and the speed with which the steps are executed only depend on motor and load characteristics.
- Current control drivers are more advanced, as they regulate the current flowing through the active coil in order to have better control over the torque produced, and thus the dynamic behavior of the whole system.

Unipolar/Bipolar Motors

Another feature of the motor that also affects control is the arrangement of the stator coils that determine how the current direction is changed. To achieve the motion of the rotor, it is necessary not only to energize the coils, but also to control the direction of the current, which determines the direction of the magnetic field generated by the coil itself.

3.4.4 Stepper Motor Uses and Applications

Due to their properties, stepper motors are used in many applications where a simple position control and the ability to hold a position are needed, including:

- 3D printing equipment
- Textile machines
- Printing presses
- Gaming machines
- Medical imaging machinery
- Small robotics
- CNC milling machines
- Welding equipment

While these applications are the most common, they're a fraction of what stepper motors can be used for. Generally speaking, any application that requires highly accurate positioning, speed control, and low speed torque can benefit from the use of stepper motors.

3.5 A4988 Stepper Motor Driver Chip

At the heart of the module is a Micro stepping Driver from Allegro – A4988. It's small in stature (only 0.8″ × 0.6″) but still packs a punch.

Figure 8: Stepper Motor Driver A4988

The A4988 stepper motor driver has output drive capacity of up to 35 V and ± 2A and lets you control one bipolar stepper motor at up to 2A output current per coil like NEMA 17.

The driver has built-in translator for easy operation. This reduces the number of control pins to just 2, one for controlling the steps and other for controlling spinning direction.

The driver offers 5 different step resolutions viz. full-step, half-step, quarter-step, eighth-step, and sixteenth-step.

3.5.1 A4988 Motor Driver Pinout

The A4988 driver has total 16 pins that interface it to the outside world. The connections are as follows:

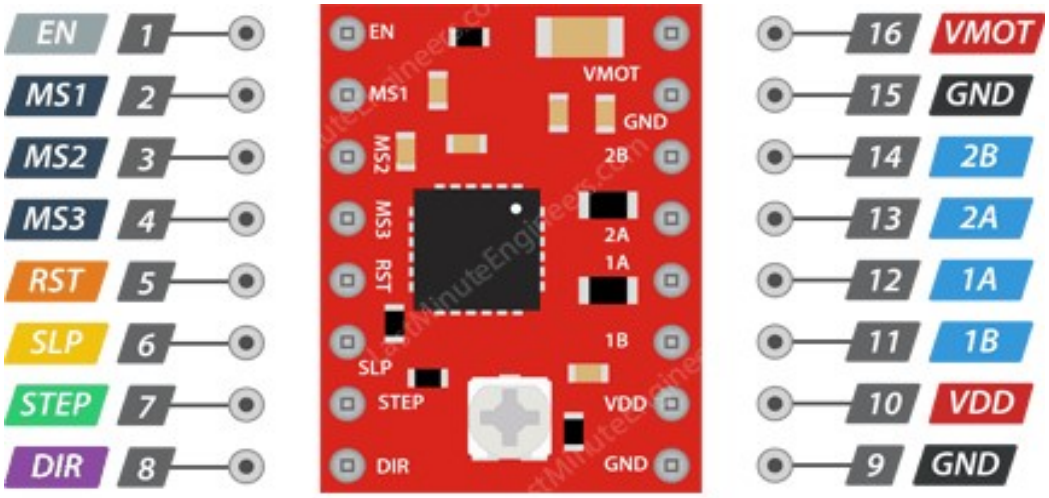


Figure 9: A4988 pin diagram

Let's familiarize ourselves with all the pins one by one.

3.5.2 Power Connection Pins

The A4988 actually requires two power supply connections.

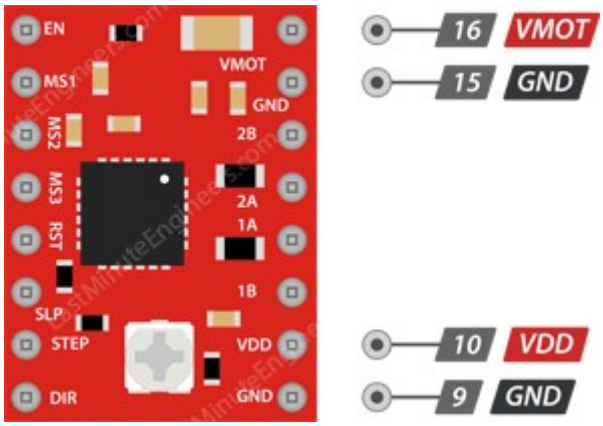


Figure 10: A4988 power pins

VDD & GND is used for driving the internal logic circuitry which can be 3V to 5.5 V. Whereas,

VMOT & GND supplies power for the motor which can be 8V to 35 V.

According to datasheet, the motor supply requires appropriate decoupling capacitor close to the board, capable of sustaining 4A. In our project the stepper motor is connected to 12 V. We use 100 µF capacitor between **VMOT & GND.**

3.5.3 Micro-step Selection Pins

The A4988 driver allows micro stepping by allowing intermediate step locations. This is achieved by energizing the coils with intermediate current levels.

For example, if you choose to drive NEMA 17 having 1.8° or 200 steps per revolution in quarter-step mode, the motor will give 800 micro steps per revolution.

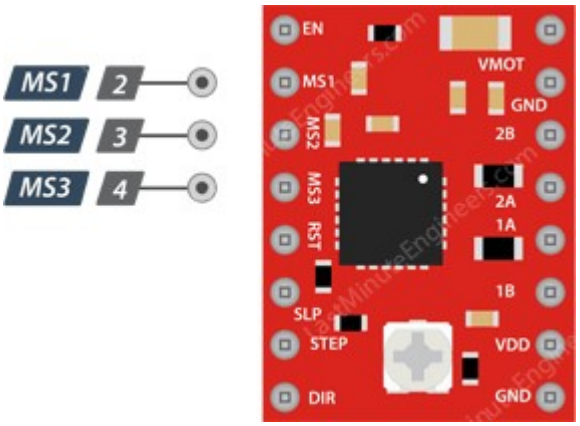


Figure 11: A4988 micro-step pin selection

The A4988 driver has three step size(resolution) selector inputs viz. MS1, MS2 & MS3. By setting appropriate logic levels to these pins, we can set the motors to one of the five step resolutions.

Table 2: Micro-stepping selection

MS1	MS2	MS3	Micro-step Resolution
Low	Low	Low	Full step
High	Low	Low	Half step
Low	High	Low	Quarter step
High	High	Low	Eighth step
High	High	High	Sixteenth step

These three micro step selection pins are pulled LOW by internal pull-down resistors, so if we leave them disconnected, the motor will operate in full step mode.

3.5.4 Control Input Pins

The A4988 has two control inputs viz. STEP and DIR.

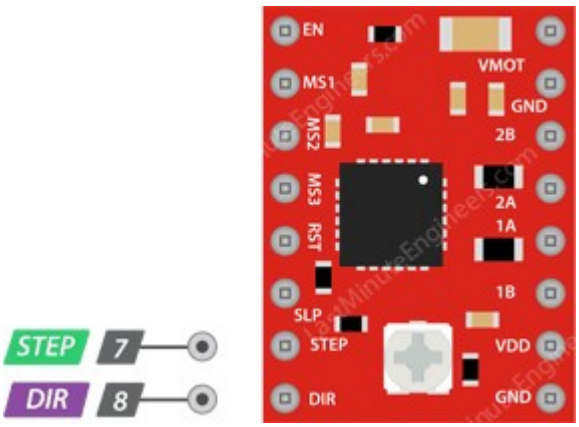


Figure 12: A4988 control pins

STEP input controls the micro-steps of the motor. Each HIGH pulse sent to this pin steps the motor by number of micro-steps set by Micro-step Selection Pins. The faster the pulses, the faster the motor will rotate.

DIR input controls the spinning direction of the motor. Pulling it HIGH drives the motor clockwise and pulling it LOW drives the motor counterclockwise.

3.5.5 Pins for Controlling Power States

The A4988 has three different inputs for controlling its power states viz. EN, RST, and SLP.

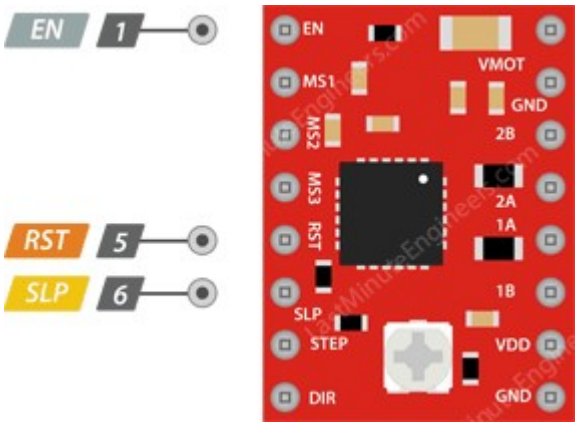


Figure 13: A4988 power state control pins

EN Pin is active low input, when pulled LOW (logic 0) the A4988 driver is enabled. By default, this pin is pulled low so the driver is always enabled, unless you pull it HIGH.
SLP Pin is active low input. Meaning, pulling this pin LOW puts the driver in sleep mode, minimizing the power consumption. You can invoke this especially when the motor is not in use to conserve power.
RST is also an active low input. When pulled LOW, all STEP inputs are ignored, until you pull it HIGH. It also resets the driver by setting the internal translator to a predefined home state. Home state is basically the initial position from where the motor starts and it's different depending upon the micro-step resolution.

3.5.6 Output Pins

The A4988 motor driver's output channels are broken out to the edge of the module with 1B, 1A, 2A & 2B pins.

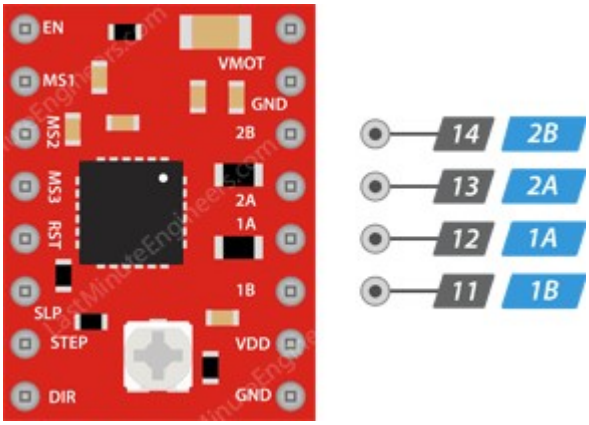


Figure 14: A4988 output pins

You can connect any bipolar stepper motor having voltages between 8V to 35 V to these pins.

Each output pin on the module can deliver up to 2A to the motor. However, the amount of current supplied to the motor depends on system's power supply, cooling system & current limiting setting.

3.6 Interface OLED Graphic Display Module with ESP32

OLED Display:

The OLED display module breaks out a small monochrome OLED display. It's 128 pixels wide and 64 pixels tall, measuring 0.96″ across. It's micro, but it still packs a punch – the OLED display is very readable due to the high contrast, and you can fit a deceivingly large number of graphics on there.

3.6.1 Pin Description

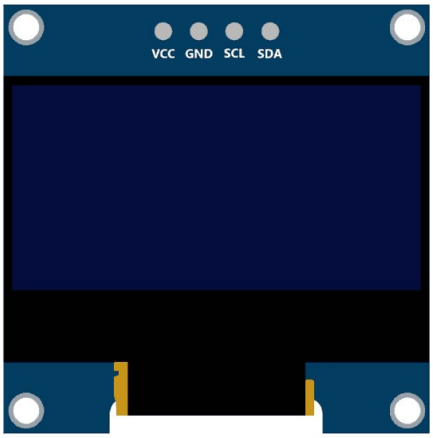


Figure 15: 128x64 I2C based OLED module.

VCC: This is the power pin for the module. A supply of 3.3V or 5V can be provided to this pin to power the display.

GND: This is the ground pin for the module.

SCL and SDA: These are the serial clock and serial data pins for I2C Interface.

3.6.2 Wiring OLED display module to ESP32

Connections are fairly simple. Start by connecting VCC pin to the 3.3V output on the ESP32 and connect GND to ground.

Next, Connect the SCL pin to the I2C clock D22 pin on your ESP32 and connect the SDA pin to the I2C data D21 pin on ESP32. Refer to ESP32 Pinout.

The following diagram shows you how to wire everything.

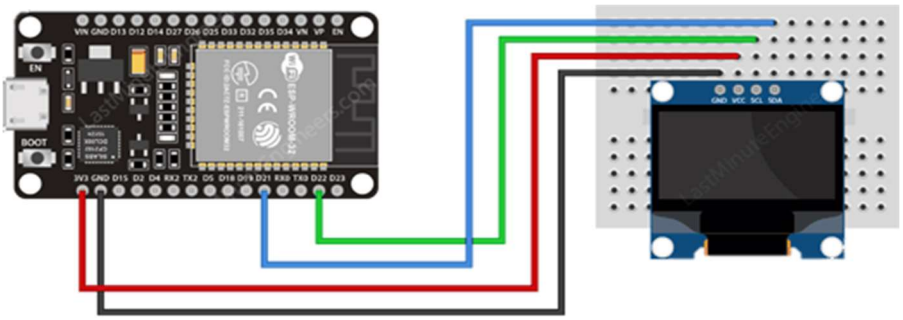


Figure 16: Wiring Connections for OLED Display Module with ESP32

3.7 Overview of the Project:

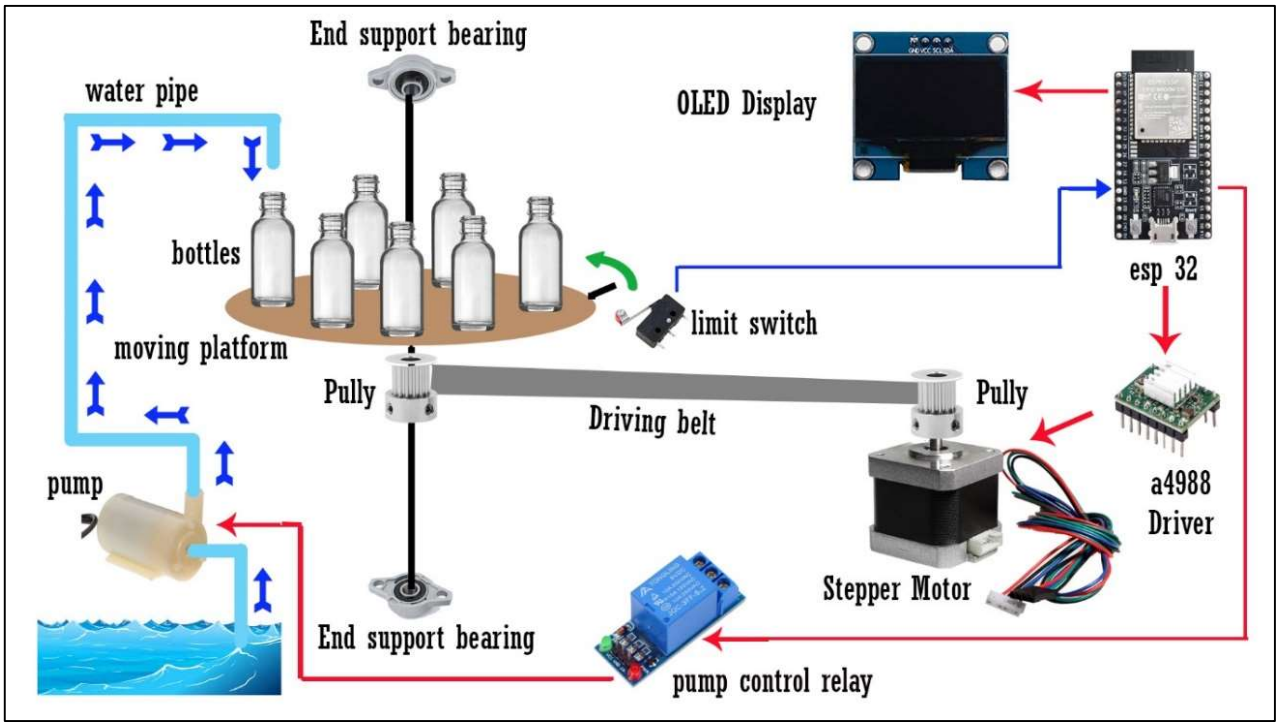


Figure 17: Overview of the project

This project is the perfect example of sequential process. Without the use of the any sensors the bottles are filling perfectly. The bottles are placed in the predefine places in the rotating platform

with the help of some cylindrical piece of plastic. These piece of plastic holds the bottles in the places securely. The rotating platform rotates with the help of the stepper motor. The motor moves the platform in perfect 60^0 in every step and ensure that the bottles are placed exactly bellow the water pipe. This process repeats until all the bottles filled up. The whole process is monitored are displayed in the built in OLED display. Also, a buzzer is attached in the board to give some audio feedback.

3.8 Circuit Diagram:

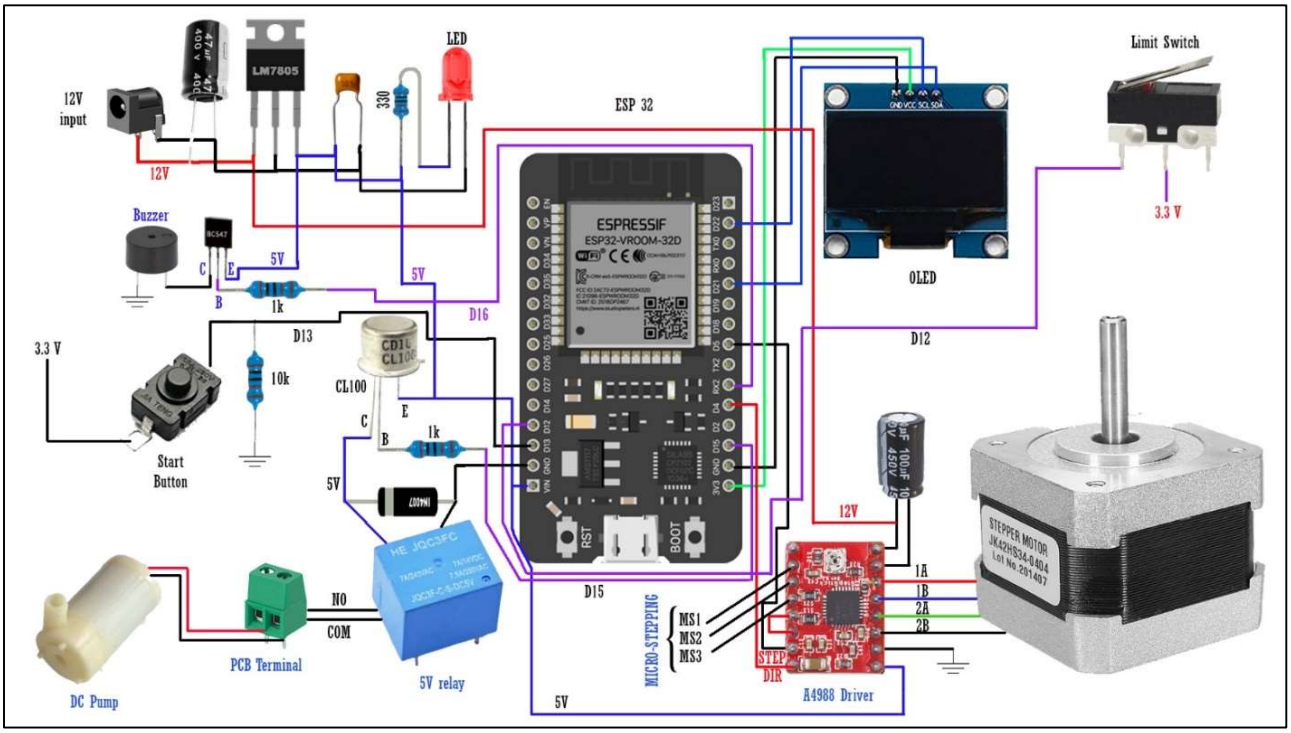


Figure 18: Circuit diagram of the developed prototype

CHAPTER 4

(Hardware Modeling)

4.1 Main features of the prototype

The features of the developed prototype are:

- Automatic and accurate bottle filling
- Can be used with any type of liquid
- Reduce Man-Dependency
- Real time status display in the OLED
- Extension for relay board connection
- Buzzer connected for audio feedback
- Single power supply and on-board voltage regulator
- Micro stepping option for smooth movement of the motor

4.2 Photographs of the main controller board

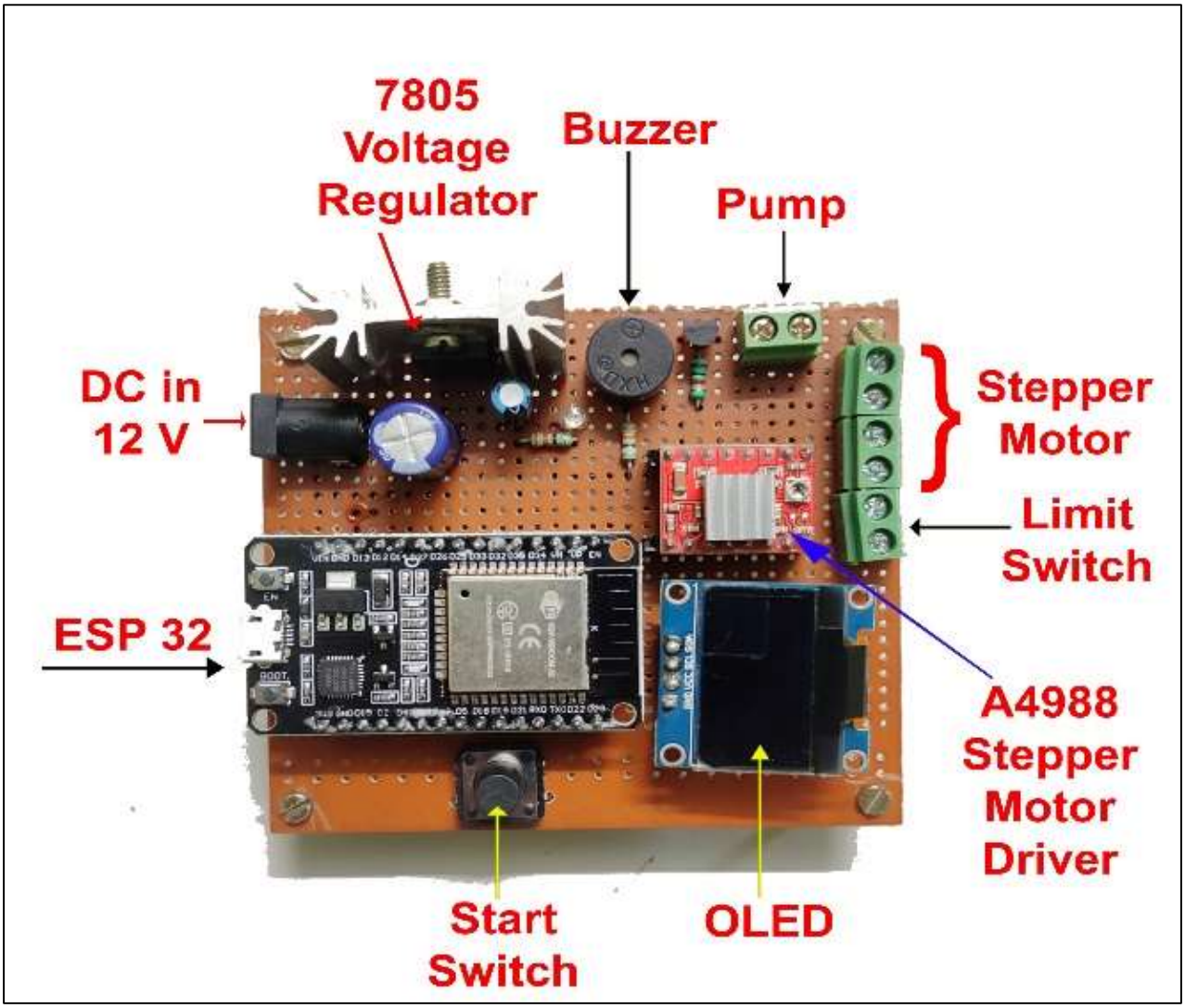


Figure 19: Main Controller and Relay board

4.3 Step by step operation of the prototype

1. Connect the DC adapter (12V, 1A) to the DC socket
2. Voltage Regulator (LM7805) provides 5 Volt input for ESP32 And Motor Driver(A4988)
3. The Pump is fed with 5V DC
4. First the Stepper Motor will run in clock wise direction until the limit switch is triggered
5. After the triggering of the limit switch the initial position is determine
6. The Pump will be on for a calculated time to fill the bottles
7. Next the stepper motor will rotate the supporting platform for 60 degrees and pump will fill the bottles and OLED will display the status of the filled bottles
8. After filling all the bottles, the OLED displayed the final message and the whole system will stop.

4.4 Components required

Table 3: Component listing

Sl. No.	Component	Qtn
1.	ESP32	1
2.	NEMA 17 Stepper Motor	1
3.	A4988 Stepper motor driver	1
4.	Static Relay (5 volt)	1
5.	0.96” OLED (I2C)	1
6.	CK100 transistor (NPN)	1
7.	General blank PCB (KS 100)	1
8.	5 mm LED	1
9.	Berg terminals	14
10.	Small DC Pump (3-6 V)	1
11.	8mm zinc alloy pillow block flange bearing	2
12.	GT2 timing pully (20 tooth, 5mm)	1
13.	GT2 timing pully (40 tooth, 8mm)	1
14.	GT2 timing belt (280 mm long, 6mm width)	1
15.	8mm support rod	20 cm
16.	Limit switch	1
17.	Push button	1
18.	BC547 transistor	1
19	Male PCB Header Connector	1
20.	3mm water pipe	3 ft

4.5 Hardware interfacing

4.5.1 Relay Driver interfacing with ESP32

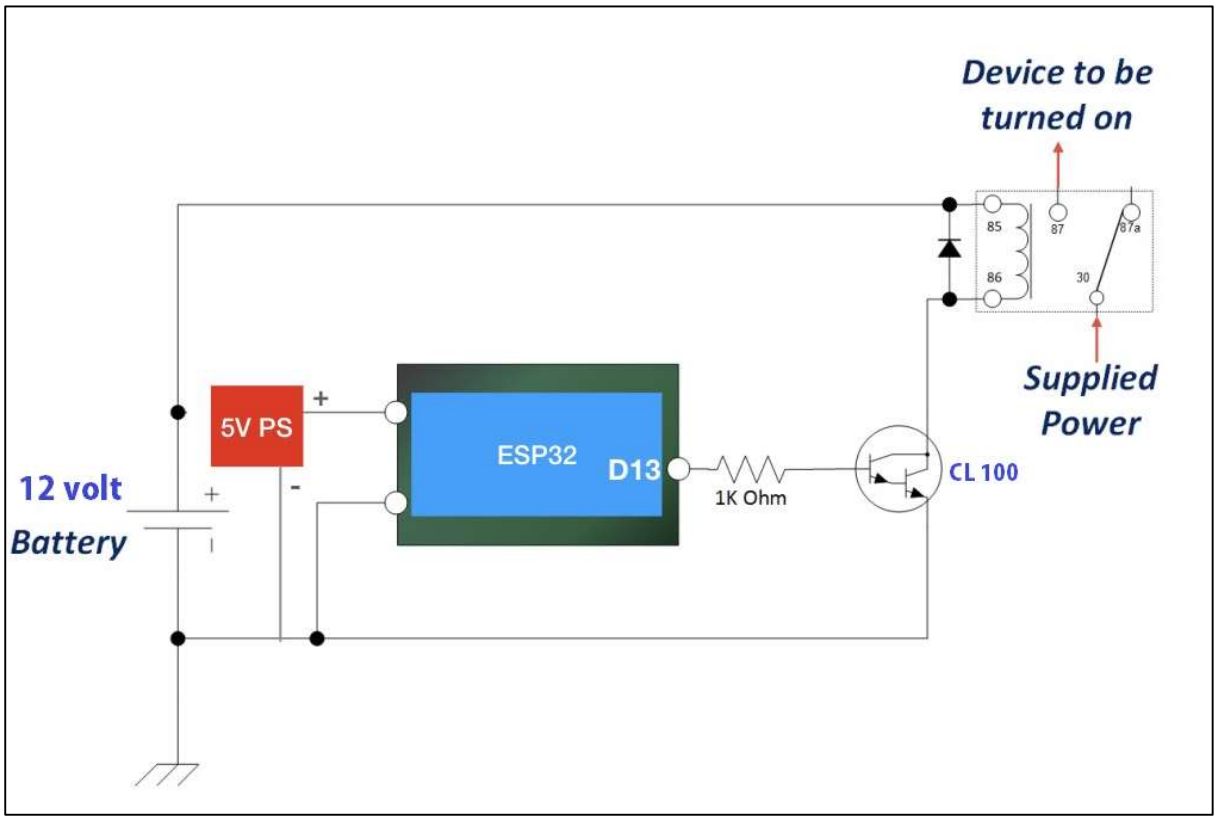


Figure 20: Relay interfacing with ESP32

The ESp32 microcontroller runs in 3.3-volt logic. Its GPIO pins maximum give 3.3 volt as logic high level and maximum 40 mA of current. Here we need a voltage driven relay driver instead of current driven. CK100 is a NPN Silicon Planar Transistors. The esp32 GPIO output pin is connected to the base of the transistor through 1k resistor, collector is connected to the +5 volt and emitter is connected to the ground. For a '0' from microcontroller the corresponding relay is turned off and a '1' from microcontroller is turned on the relay. For free wheeling a diode needs to be connected parallel to the relay coil with reverse bias as shown in the figure 20.

4.5.2 A4988 Interfacing with ESP32

To connect the ESP32 board with the stepper motor and driver we will use all the pins of the driver except for the enable pin and the micro step resolution selection pins. Connect the output pins of the driver with the respective motor pins. Connect the STEP pin and the DIR pin with any appropriate GPIO pin of ESP32 board. We have used GPIO12 to connect with DIR and GPIO14 to connect with STEP. As we want to operate our stepper mode in full mode hence, we will leave the MS1, MS2 and MS3 pins as they are. The RST pin will be connected with SLP so that the driver is enabled. Moreover, the VCC and GND pins will be connected with Vin and GND pin from ESP32 respectively. The VMOT will be connected with an external power supply ranging between 8-35V. We are using 12V external power supply. Make sure the GND pins are connected with the respective common grounds.

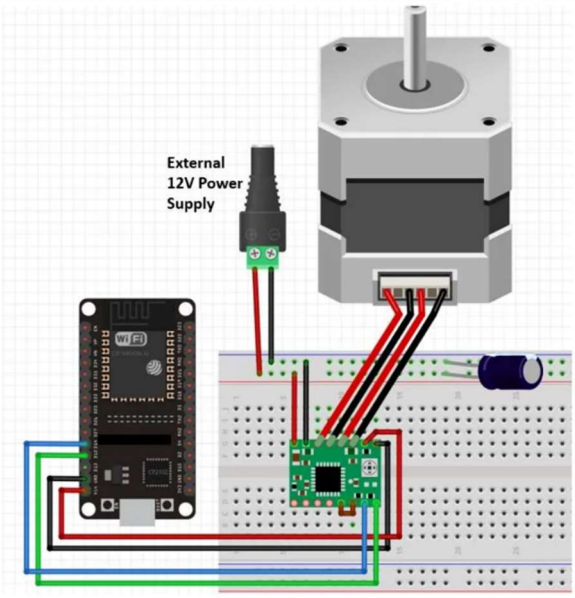


Figure 21: A4988 interfacing with ESP32

4.5.3 ESP32 OLED Display with Arduino IDE

The combination of OLED with ESP32 is so popular that there are some boards of ESP32 with the OLED integrated. We'll, however, assume that you will be using a separate OLED module with your ESP32 board. If you have an OLED module, it perhaps looks like the image below.

Figure 22: 0.96" OLED I2C module

The OLED (**O**rganic **L**ight **E**mitting **D**iode) display that we'll use in this tutorial is the SSD1306 model: a monocolor, 0.96-inch display with 128×64 pixels as shown in the above figure.

The OLED display doesn't require backlight, which results in a very nice contrast in dark environments. Additionally, its pixels consume energy only when they are on, so the OLED display consumes less power when compared to other displays.

The model we're using has four pins and communicates with any microcontroller using I2C communication protocol. There are models that come with an extra RESET pin or that communicate using SPI communication protocol.

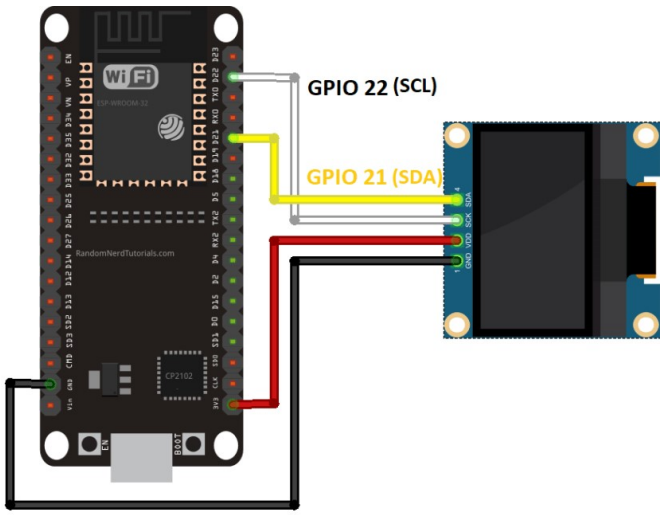
OLED Display SSD1306 Pin Wiring

Because the OLED display uses I2C communication protocol, wiring is very simple. Use the following table as a reference.

Table 4: OLED interfacing with ESP32

Pin	ESP32
Vin	3.3V
GND	GND
SCL	GPIO 22
SDA	GPIO 21

Alternatively, it can follow the next schematic diagram to wire the ESP32 to the OLED display.



In this example, we're using I2C communication protocol. The most suitable pins for I2C communication in the ESP32 are GPIO 22 (SCL) and GPIO 21 (SDA).

If you're using an OLED display with SPI communication protocol, use the following GPIOs.

- GPIO 18: CLK
- GPIO 19: MISO
- GPIO 23: MOSI
- GPIO 5: CS

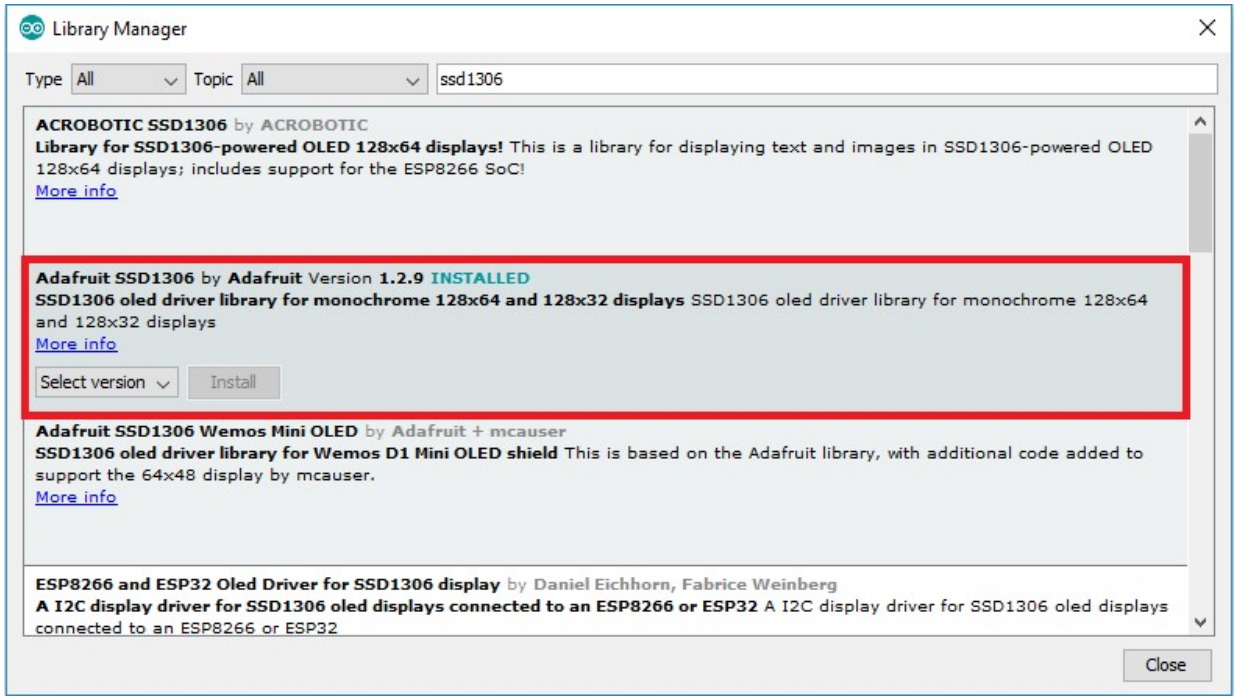
Installing SSD1306 OLED Library – ESP32

There are several libraries available to control the OLED display with the ESP32. In this project we'll use two Adafruit libraries: **Adafruit_SSD1306 library** and **Adafruit_GFX library**.

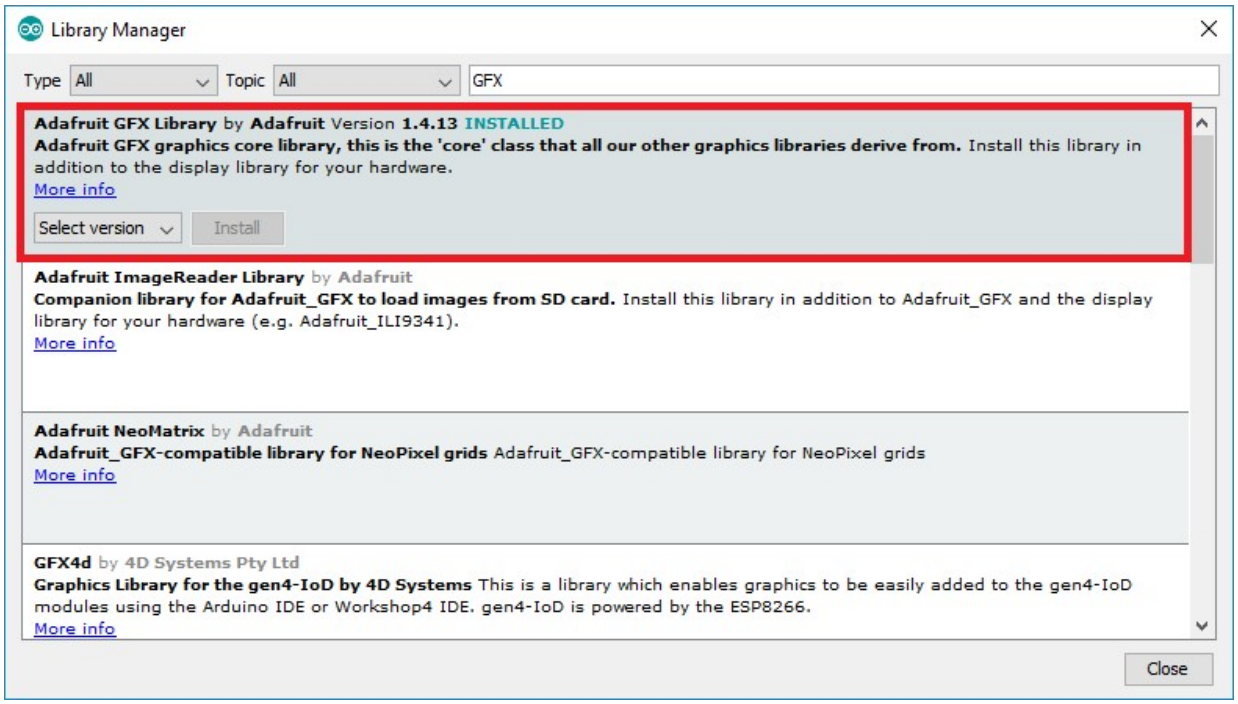
Follow the next steps to install those libraries.

1. Open your Arduino IDE and go to **Sketch** > **Include Library** > **Manage Libraries**. The Library Manager should open.

2. Type “**SSD1306**” in the search box and install the SSD1306 library from Adafruit.



3. After installing the SSD1306 library from Adafruit, type “**GFX**” in the search box and install the library.

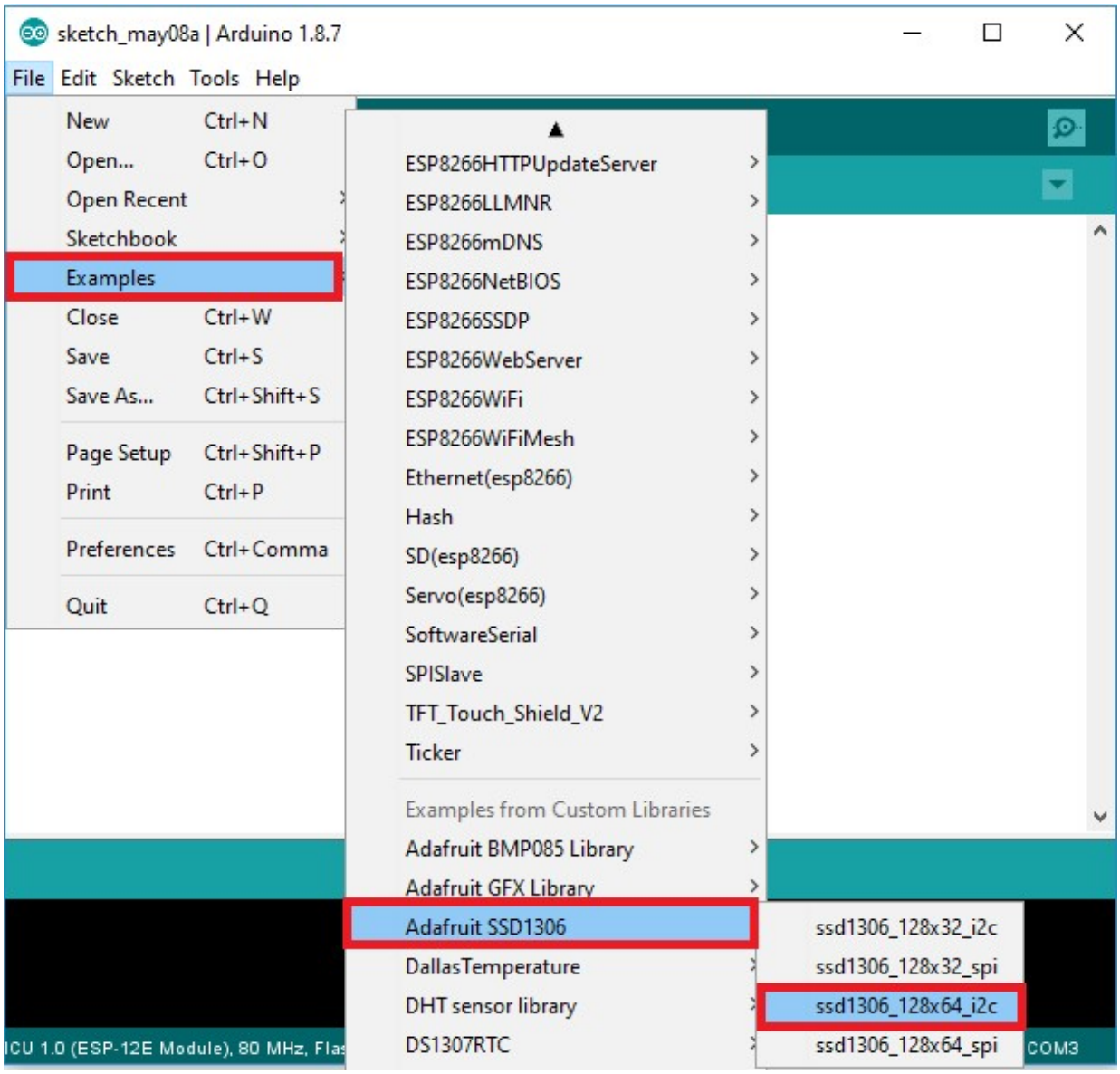


4. After installing the libraries, restart your Arduino IDE.

Testing OLED Display with ESP32

After wiring the OLED display to the ESP32 and installing all required libraries, you can use one example from the library to see if everything is working properly.

In your Arduino IDE, go to **File** > **Examples** > **Adafruit SSD1306** and select the example for the display you’re using.



Write Text – OLED Display

The Adafruit library for the OLED display comes with several functions to write text. In this section, you'll learn how to write and scroll text using the library functions.

"Hello, world!" OLED Display

The following sketch displays Hello, world! message in the OLED display.

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

#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

void setup() {
```

```
  Serial.begin(115200);
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3D for 128x64
    Serial.println(F("SSD1306 allocation failed"));
    for(;;);
  }
  delay(2000);
  display.clearDisplay();

  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0, 10);
  // Display static text
  display.println("Hello, world!");
  display.display();
}

void loop() {

}
```

After uploading the code, this is what you'll get in your OLED:

Let's take a quick look on how the code works.

Importing libraries

First, you need to import the necessary libraries. The `Wire` library to use I2C and the Adafruit libraries to write to the display: `Adafruit_GFX` and `Adafruit_SSD1306`.

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

Initialize the OLED display

Then, you define your OLED width and height. In this example, we're using a 128×64 OLED display. If you're using other sizes, you can change that in the SCREEN_WIDTH, and SCREEN_HEIGHT variables.

```
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels
```

Then, initialize a display object with the width and height defined earlier with I2C communication protocol (&Wire).

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

The (-1) parameter means that your OLED display doesn't have a RESET pin. If your OLED display does have a RESET pin, it should be connected to a GPIO. In that case, you should pass the GPIO number as a parameter.

In the setup(), initialize the Serial Monitor at a baud raute of 115200 for debugging purposes.

```
Serial.begin(115200);
```

Initialize the OLED display with the begin() method as follows:

```
if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
  Serial.println("SSD1306 allocation failed");
  for(;;); // Don't proceed, loop forever
}
```

This snippet also prints a message on the Serial Monitor, in case we're not able to connect to the display.

```
Serial.println("SSD1306 allocation failed");
```

In case you're using a different OLED display, you may need to change the OLED address. In our case, the address is 0x3C.

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

After initializing the display, add a two second delay, so that the OLED has enough time to initialize before writing text:

```
delay(2000);
```

Clear display, set font size, color and write text

After initializing the display, clear the display buffer with the clearDisplay () method:

```
display.clearDisplay();
```

Before writing text, you need to set the text size, color and where the text will be displayed in the OLED.

Set the font size using the setTextSize() method:

```
display.setTextSize(1);
```

Set the font color with the setTextColor() method:

```
display.setTextColor(WHITE);
```

WHITE sets white font and black background.

Define the position where the text starts using the setCursor(x,y) method. In this case, we're setting the text to start at the (0,0) coordinates – at the top left corner.

```
display.setCursor(0,0);
```

Finally, you can send the text to the display using the println() method, as follows:

```
display.println("Hello, world!");
```

Then, you need to call the display() method to actually display the text on the screen.

```
display.display();
```

Draw Shapes in the OLED Display

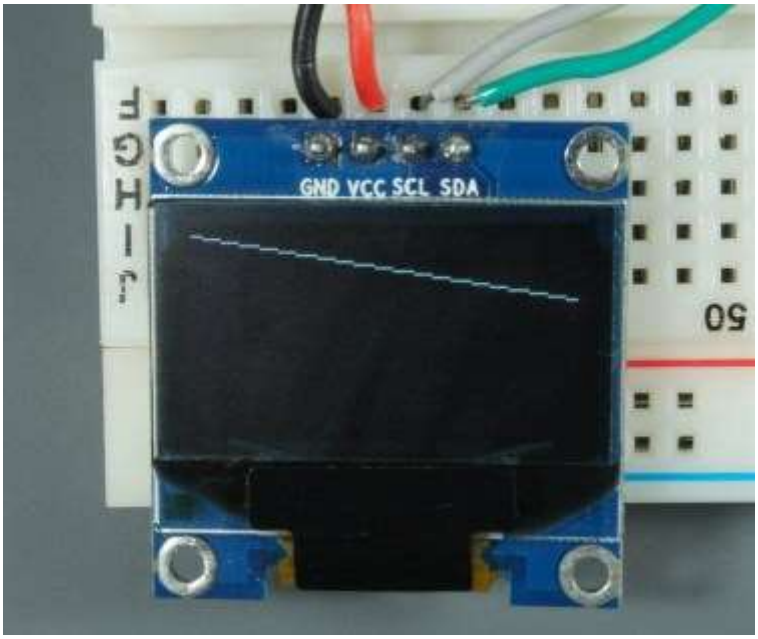
The Adafruit OLED library provides useful methods to draw pixels, lines and shapes. Let's take a quick look at those methods.

Draw a pixel

To draw a pixel in the OLED display, you can use the drawPixel(x, y, color) method that accepts as arguments the x and y coordinates where the pixel appears, and color. For example:

```
display.drawPixel(64, 32, WHITE);
```

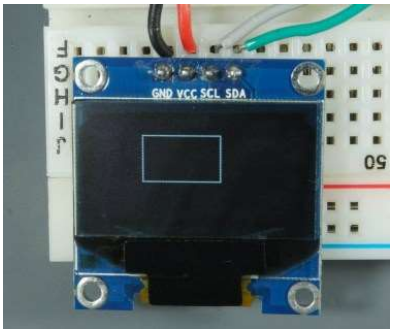
Draw a line



Use the drawLine(x1, y1, x2, y2, color) method to create a line. The (x1, y1) coordinates indicate the start of the line, and the (x2, y2) coordinates indicates where the line ends. For example:

```
display.drawLine(0, 0, 127, 20, WHITE);
```

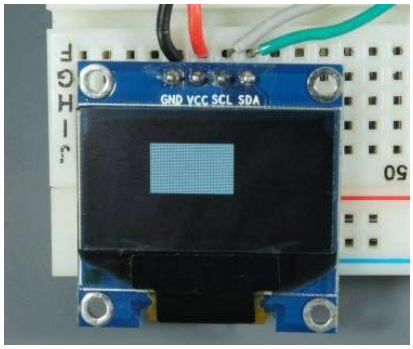
Draw a rectangle



The drawRect(x, y, width, height, color) provides an easy way to draw a rectangle. The (x, y) coordinates indicate the top left corner of the rectangle. Then, you need to specify the width, height and color:

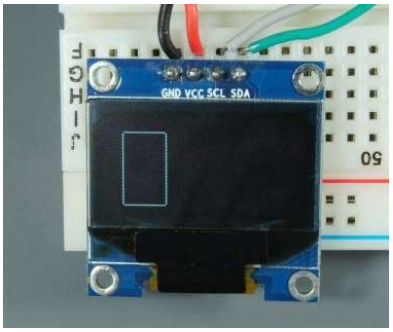
```
display.drawRect(10, 10, 50, 30, WHITE);
```

You can use the fillRect(x, y, width, height, color) to draw a filled rectangle. This method accepts the same arguments as drawRect().



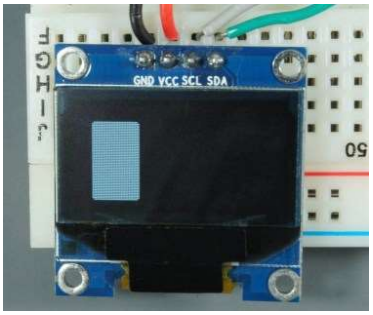
The library also provides methods to displays rectangles with round corners: drawRoundRect() and fillRoundRect(). These methods accepts the same arguments as previous methods plus the radius of the corner. For example:

```
display.drawRoundRect(10, 10, 30, 50, 2, WHITE);
```



Or a filled round rectangle:

```
display.fillRoundRect(10, 10, 30, 50, 2, WHITE);
```



Draw a circle

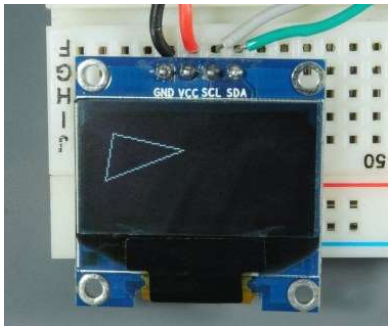
To draw a circle use the drawCircle(x, y, radius, color) method. The (x,y) coordinates indicate the center of the circle. You should also pass the radius as an argument. For example:

```
display.drawCircle(64, 32, 10, WHITE);
```

In the same way, to build a filled circle, use the fillCircle() method with the same arguments:

```
display.fillCircle(64, 32, 10, WHITE);
```

Draw a triangle

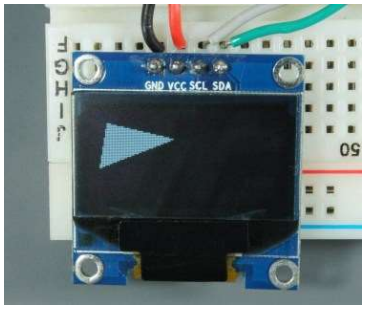


Use the the drawTriangle(x1, y1, x2, y2, x3, y3, color) method to build a triangle. This method accepts as arguments the coordinates of each corner and the color.

```
display.drawTriangle(10, 10, 55, 20, 5, 40, WHITE);
```

Use the fillTriangle() method to draw a filled triangle.

```
display.fillTriangle(10, 10, 55, 20, 5, 40, WHITE);
```



CHAPTER 5

(Logic & Operation)

5.1 INTRODUCTION

After assembling the system, what remains is to observe its operation and efficiency of the system. The total system is divided in several sub systems, like

- ESP32 Section
- Stepper Motor Section
- OLED Section
- Relay Section

The operation of the whole circuit is depending on every sections performance.

5.2 Flow Chart

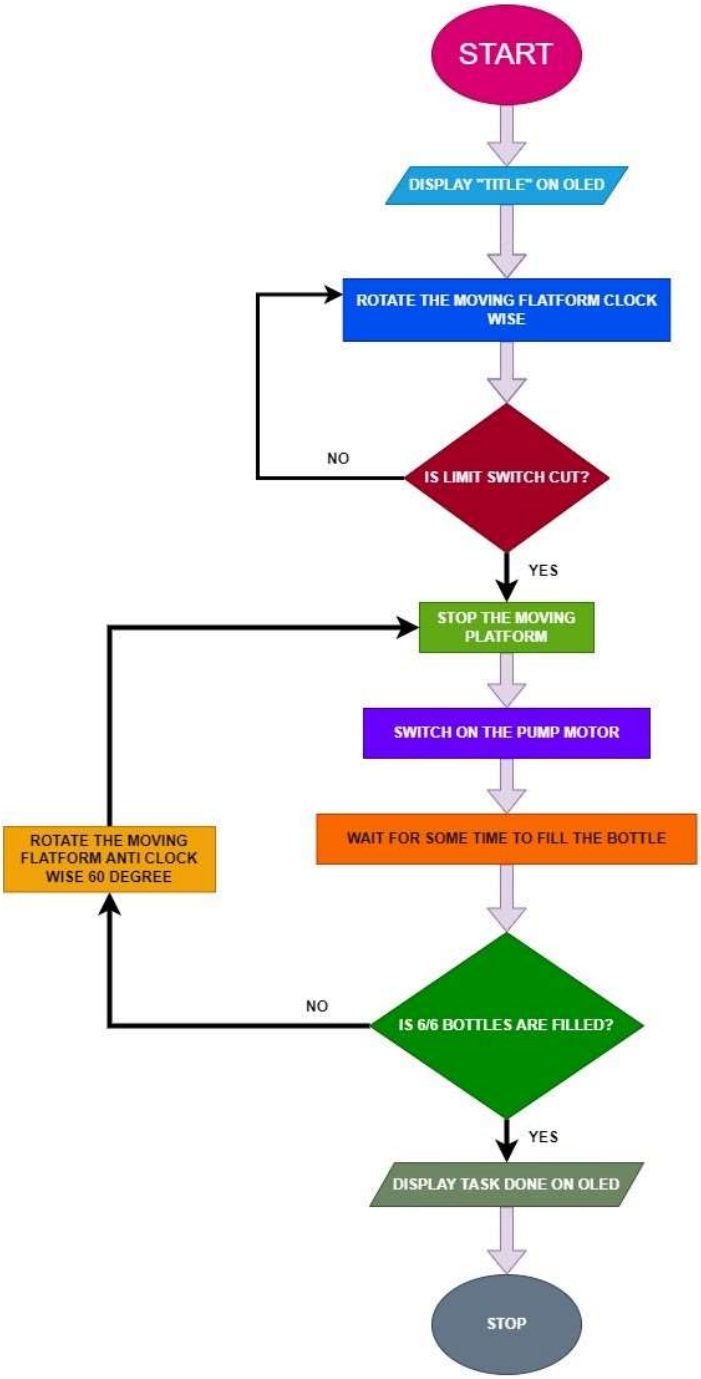


Figure 23: Flow chart of the program

5.3 Principle & Operations

This project is the perfect example of sequential process. Without the use of the any sensors the bottles are filling perfectly. The bottles are placed in the predefine places in the rotating platform with the help of some cylindrical piece of plastic. These piece of plastic holds the bottles in the places securely. The rotating platform rotates with the help of the stepper motor. The motor moves the platform in perfect 60^0 in every step and ensure that the bottles are placed exactly bellow the water pipe. This process repeats until all the bottles filled up. The whole process is monitored are displayed in the built in OLED display. Also, a buzzer is attached in the board to give some audio feedback.

5.3.1 Advantages of the ESP32

- ***Low cost:*** The ESP32 is less costly than any other IOT based Devices. Because the wifi module which is used in it is of lowest cost.
- ***Network API:*** ESP32 has easily configurable network API.
- ***Integrated Wifi Module:*** It is an easily accessible wifi module.

5.3.2 Disadvantages

- ***Less documentation*****:** There is little documentation available regarding ESP32 so for understanding the operation, GPIO, ADC there is a little problem.
- ***3.3-volt operation:*** Unlike Arduino, ESP32 works on 3.3-volt logic. So, interfacing different sensors and component to the ESP32 special caution must be given.

5.4 Cost estimation of the project

In this project we have used the cheapest IOT module NODE MCU. So the total cost of the project is reduced compare to the other IOT project. The total estimated cost of the complete project is listed in table 3.

Table 5: Costing of the projects

Sl. No.	Component	Price
1.	ESP32	400
2.	NEMA 17 Stepper Motor	750
3.	A4988 Stepper motor driver	200
4.	Static Relay (5 volt)	25
5.	0.96” OLED (I2C)	200
6.	CK100 transistor (NPN)	10
7.	General blank PCB (KS 100)	40
8.	5 mm LED	2
9.	Berg terminals	15
10.	Small DC Pump (3-6 V)	55
11.	8mm zinc alloy pillow block flange bearing	318

12.	GT2 timing pully (20 tooth, 5mm)	160
13.	GT2 timing pully (40 tooth, 8mm)	190
14.	GT2 timing belt (280 mm long, 6mm width)	149
15.	8mm support rod	90
16.	Limit switch	35
17.	Push button	1
18.	BC547 transistor	1
19	Male PCB Header Connector	20
20.	3mm water pipe	10
Total		**2671/-**

5.5 Photographs of the prototype

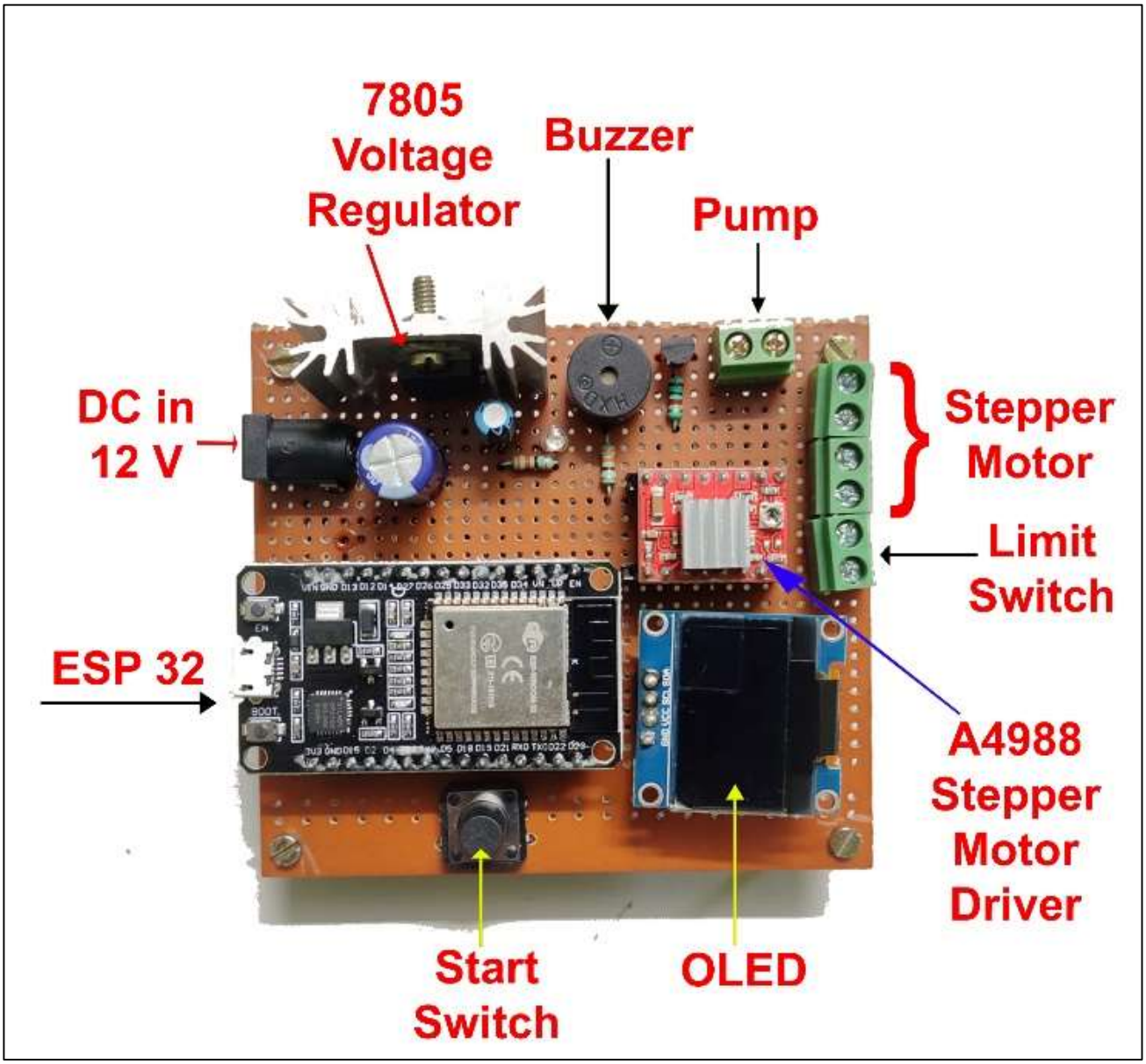


Figure 24: Main Controller Board

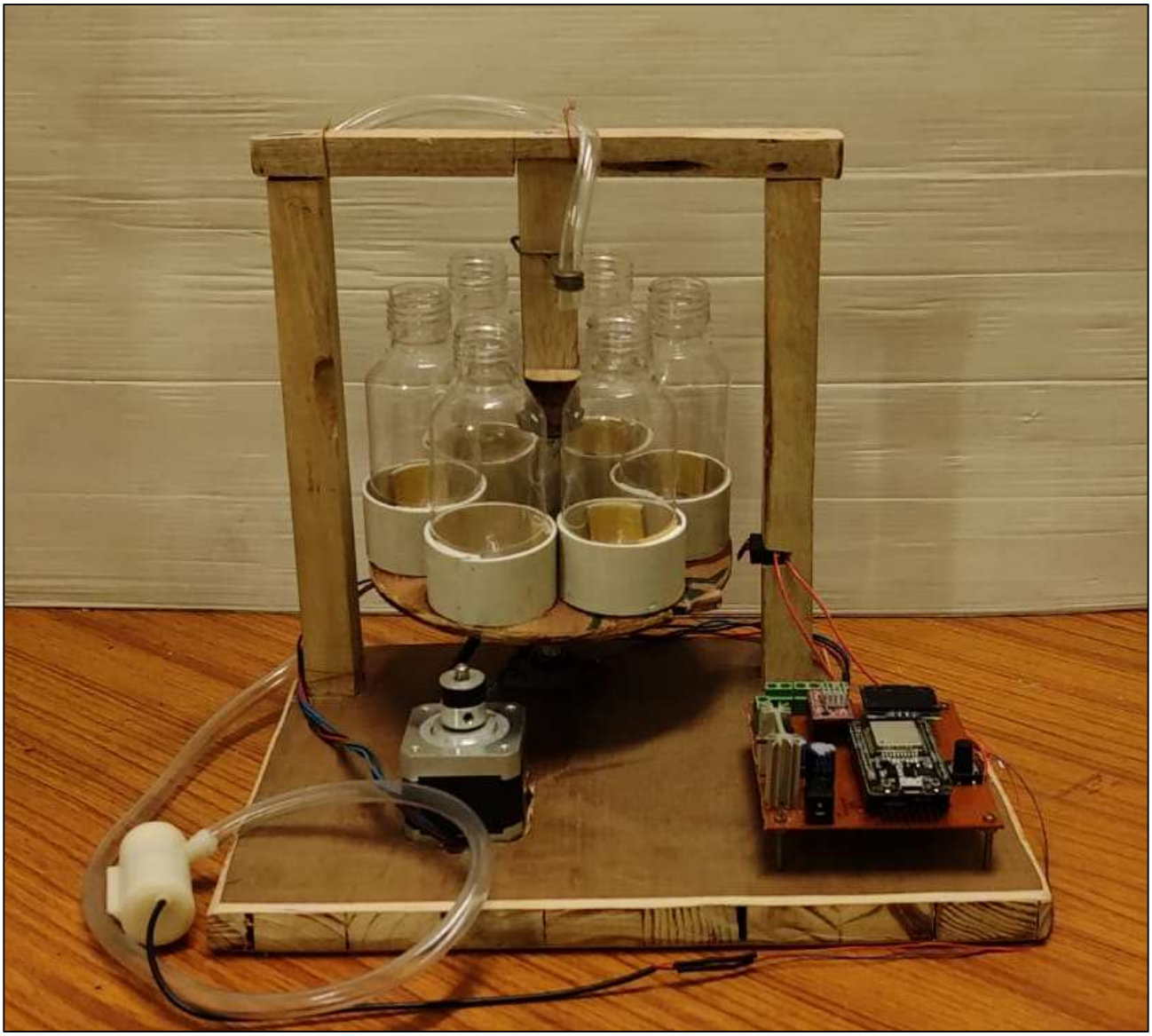


Figure 25: The Prototype

Figure 26: Pully and driving belt

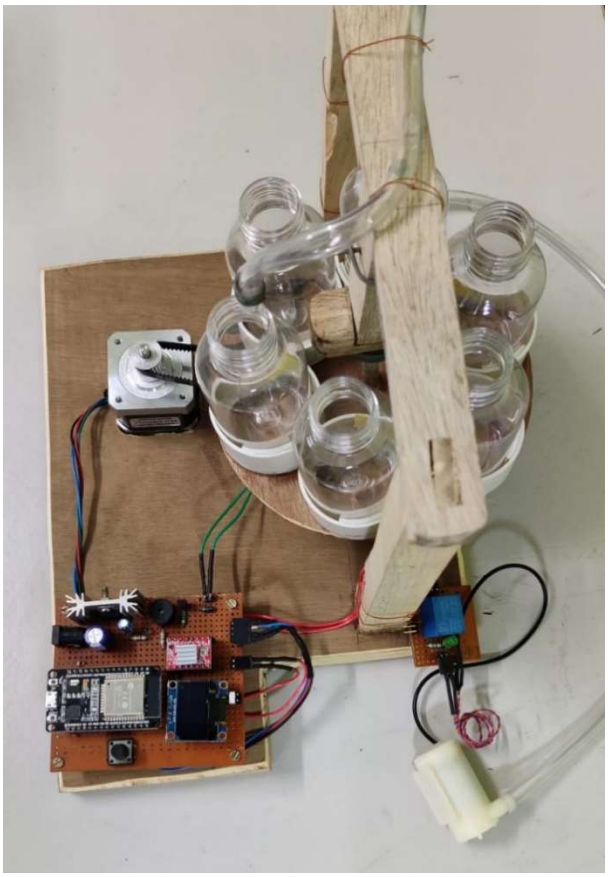


Figure 27: Complete setup

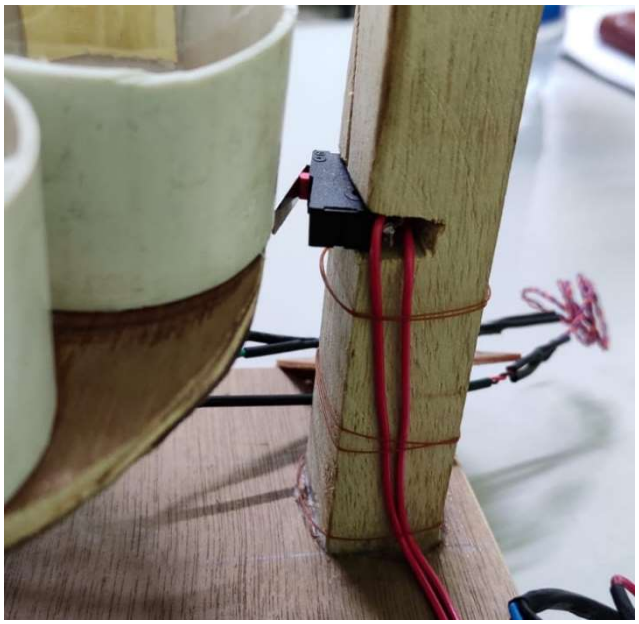


Figure 28: Limit switch

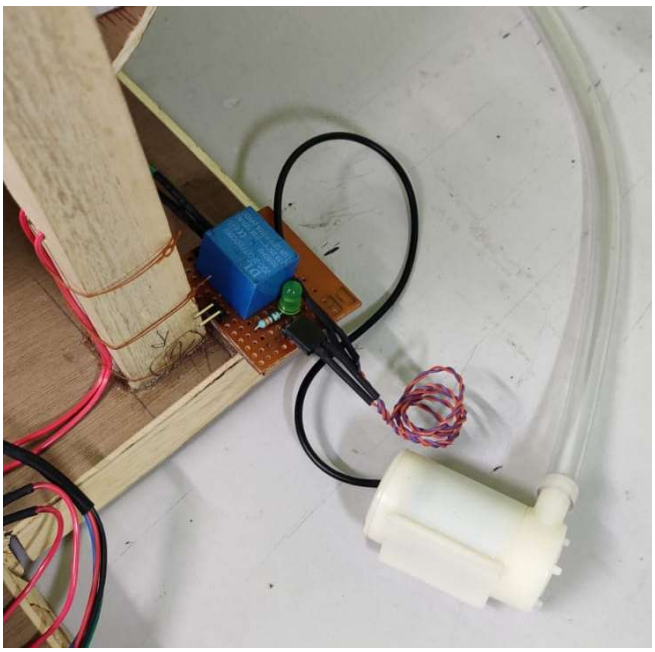


Figure 29: Pump and driver board

Chapter 6

(Conclusion & Future Scope)

6.1 Conclusion

Here we developed a prototype which automatically fills a number of bottles with the help of a micro-controller. It will help in reducing human effort and error. Our circuit consists of ESP32 as a main controller, OLED as a display device, Stepper motor as a main driving gear, Pump to fill the liquid in the bottle. The prototype worked satisfactorily.

6.2 Result

The experimental model was made according to the circuit diagram and the results were as expected. The OLED displays properly the status of the operation at each step. After all the bottles are filled the circuit stops. Here special attention must be taken to design the relay driver. During the testing it was found that the DC pump creates much RFI (Radio Frequency Interference) which affect the stepper motor driver A4988. When the pump starts the driver malfunctioned and the stepper motor behaves strangely. One noise filter circuit also developed to reduce the interference.

6.3 Future work

In this developed prototype the bottles need to placed manually and after the filling user need to replace the bottle manually. In our future work we try to develop a system which fully automatically filling the bottles. we will also try to use IoT and connect with our module, then it can be controlled with the help of a remote device, which will make it to stop the circuit remotely in case of any error.

Chapter 7

(References)

[1] Deepika Saikia, Prajakta Powar, Animesh Gaurav "Automatic Bottle Filling System", International Research Journal of Engineering and Technology (IRJET), e-ISSN: 2395-0056, p-ISSN: 2395-0072

[2] Ashwini P. Somawanshi, Supriya B. Asutkar, Sachin A. More "Automatic Bottle Filling Using Microcontroller Volume Correction", International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181, Vol. 2 Issue 3, March – 2013

[3] Aniruddh Guha, Adarsh Ganveer, Manjari Kumari, Ajay Singh Rajput, "AUTOMATIC BOTTLE FILLING MACHINE", International Research Journal of Engineering and Technology (IRJET), e-ISSN: 2395-0056, p-ISSN: 2395-0072

[4] J.Dharanidharan, R. Puviarasi, "AUTOMATIC BOTTLE FILLING MACHINE", International Conference on Recent Trends in Computing, Communication and Networking Technologies (ICRTCCNT'19) Oct 18-19, 2019, Kings Engineering College, Chennai, TamilNadu, India.

[5] Arthur Pius Santiago [2010]: Insecticide Bottle Filling and Capping machines in De La Salle University. The 11th Asia pacific Industrial Engineering and Management Systems Conference. The 14th Asia Pacific Regional Meeting of the International Foundation for production Research.

[6] Samarth Nainani, Akshata Rupawate, Shoaib Sayyed, Siddharth Poojary, Vaishali Bodhale, "AUTOMATIC BOTTLE FILLING SYSTEM USING ARDUINO UNO", International Research Journal of Modernization in Engineering Technology and Science, Volume:03/Issue:04/April-2021, e-ISSN: 2582-5208

[7] Kiran Chaudhari, Aaqib Momin, Ritu Magare, Omkar More, Aniket Kantale, "Automated Filling Machine", International Research Journal of Engineering and Technology (IRJET), e-ISSN: 2395-0056, p-ISSN: 2395-0072.

[8] Bipin Mashilkar, Pallavi Khaire and Girish Dalve, "Automated Bottle Filling System," International Research Journal of Engineering and Technology (IRJET), e-ISSN: 2395-0056, p-ISSN: 2395-0072; Volume 02, Issue 07; October 2015.

[9] Nisarg A Solanki, Pratik G Raj, Saumil P Patel, Charmish D Rajput, "Automatic Liquid Filling Machine", International Journal of Engineering Research & Technology.

Appendix A

(Hardware description)

Transformer less AC to DC power supply circuit using dropping capacitor

Production of low voltage DC power supply from AC power is the most important problem faced by many electronics developers and hobbyists. The straight forward technique is the use of a step-down transformer to reduce the 230 V or 110V AC to a preferred level of low voltage AC. But *SMPS* power supply comes with the most appropriate method to create a low-cost power supply by avoiding the use of bulky transformer. This circuit is so simple and it uses a voltage dropping capacitor in series with the phase line. Transformer less power supply is also called as capacitor power supply. It can generate 5V, 6V, 12V 150mA from 230V or 110V AC by using appropriate zener diodes.

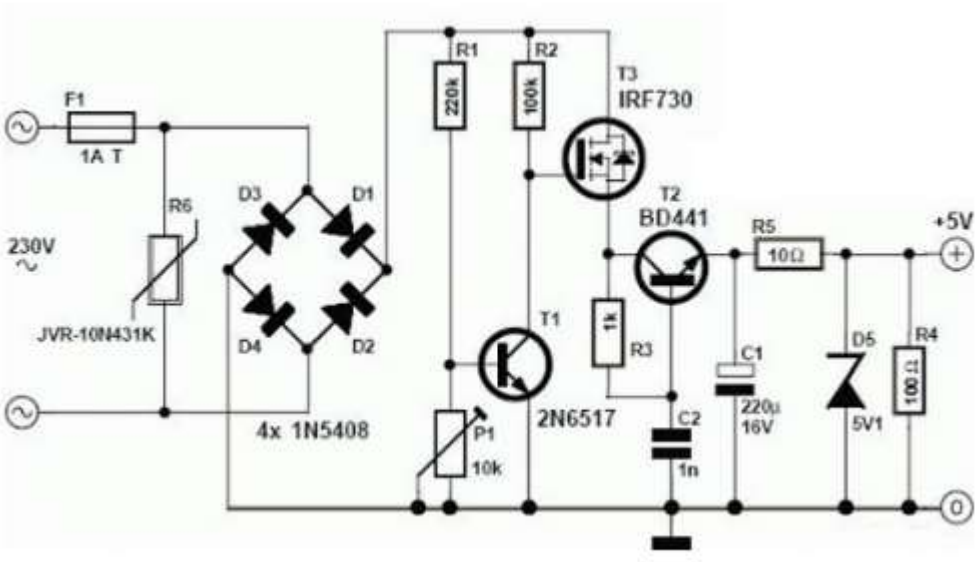


Figure 30: Transformer less SMPS 5-volt power supply

Working of Transformer less capacitor power supply

- This transformer less power supply circuit is also named as capacitor power supply since it uses a special type of AC capacitor in series with the main power line.
- A common capacitor will not do the work because the mains spikes will generate holes in the dielectric and the capacitor will be cracked by passing of current from the mains through the capacitor.
- X rated capacitor suitable for the use in AC mains is vital for reducing AC voltage.
- A X rated dropping capacitor is intended for 250V, 400V, 600V AC. Higher voltage versions are also obtainable. The dropping capacitor is non polarized so that it can be connected any way in the circuit.
- The 470kΩ resistor is a bleeder resistor that removes the stored current from the capacitor when the circuit is unplugged. It avoids the possibility of electric shock.
- Reduced AC voltage is rectified by bridge rectifier circuit. We have already discussed about bridge rectifiers. Then the ripples are removed by the 1000µF capacitor.

- This circuit provides 24 volts at 160 mA current at the output. This 24 volt DC can be regulated to necessary output voltage using an appropriate 1 watt or above zener diode.

- Here we are using 6.2V zener. You can use any type of zener diode in order to get the required output voltage.

Resistor

Figure 31: Resistor

Resistance is the opposition of a material to the current. It is measured in Ohms Ω. All conductors represent a certain amount of resistance, since no conductor is 100% efficient. To control the electron flow (current) in a predictable manner, we use resistors. Electronic circuits use calibrated lumped resistance to control the flow of current. Broadly speaking, resistor can be divided into two groups viz. fixed & adjustable (variable) resistors. In fixed resistors, the value is fixed & cannot be varied. In variable resistors, the resistance value can be varied by an adjuster knob. It can be divided into (a) Carbon composition (b) Wire wound (c) Special type. The most common type of resistors used in our projects is carbon type. The resistance value is normally indicated by color bands. Each resistance has four colors, one of the bands on either side will be gold or silver, this is called fourth band and indicates the tolerance, others three band will give the value of resistance (see table). For example, if a resistor has the following marking on it say red, violet, gold. Comparing these colored rings with the color code, its value is 27000 ohms or 27 kilo ohms and its tolerance is ±5%. Resistor comes in various sizes (Power rating). The bigger the size, the more power rating of 1/4 watts. The four-color rings on its body tells us the value of resistor value.

Color Code of the resistor

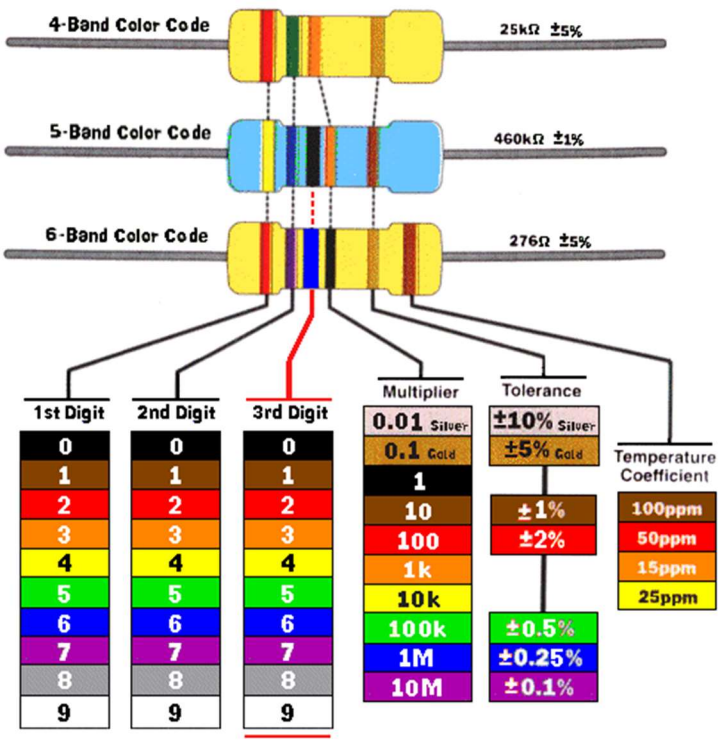


Figure 32: Color Code for resistance

RELAY

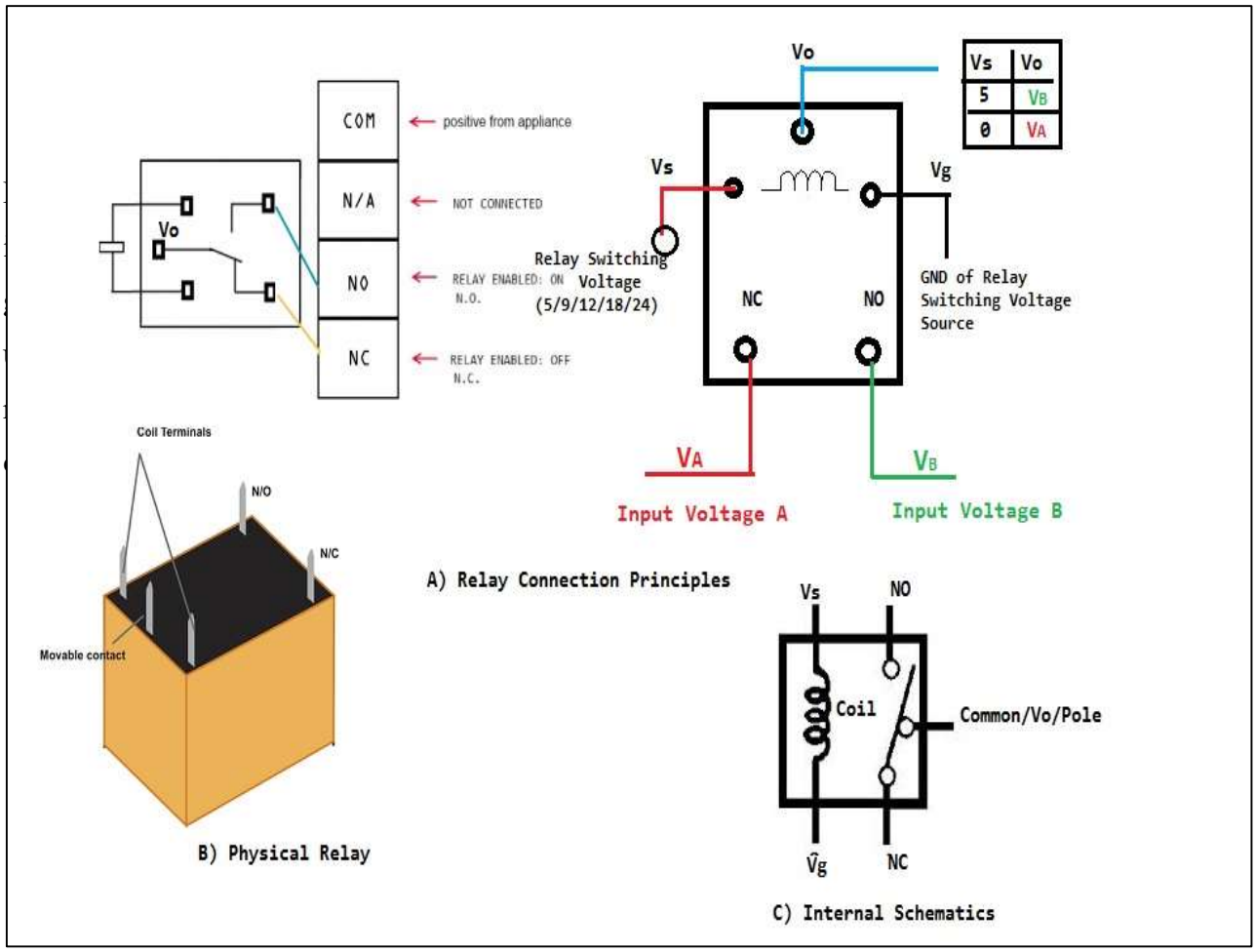


Figure 33: 6-volt Cube Relay

A relay is an electrically operated switch. Current flowing through the coil of the relay creates a magnetic field which attracts a lever and changes the switch contacts. The coil current can be on or off so relays have two switch positions and they are double throw (changeover) switches.

The relay's switch connections are usually labeled COM (POLE), NC and NO:

COM/POLE= Common, NC and NO always connect to this, it is the moving part of the switch.

NC = Normally Closed, COM/POLE is connected to this when the relay coil is not magnetized.

NO = Normally Open, COM/POLE is connected to this when the relay coil is MAGNETIZED and vice versa.

OLED

An organic light-emitting diode (OLED) is a light-emitting diode (LED) in which the emissive electroluminescent layer is a film of organic compound that emits light in response to an electric current. This organic layer is situated between two electrodes; typically, at least one of these electrodes is transparent. OLEDs are used to create digital displays in devices such as television screens, computer monitors, portable systems such as smart phones, handheld game consoles and PDAs. A major area of research is the development of white OLED devices for use in solid-state lighting applications.

Figure 34: 0.96” I2C OLED display module

ESP32

ESP32 is a series of low-cost, low-power system on a chip microcontrollers with integrated Wi-Fi and dual-mode Bluetooth. The ESP32 series employs either a Tensilica Xtensa LX6 microprocessor in both dual-core and single-core variations, Xtensa LX7 dual-core microprocessor or a single-core RISC-V microprocessor and includes built-in antenna switches, RF balun, power amplifier, low-noise receive amplifier, filters, and power-management modules. ESP32 is created and developed by Espressif Systems, a Shanghai-based Chinese company, and is manufactured by TSMC using their 40 nm process. It is a successor to the ESP8266 microcontroller.

Figure 35: ESP 32 microcontroller

Piezo buzzer

A buzzer or beeper is an audio signaling device, which may be mechanical, electromechanical, or piezoelectric. Typical uses of buzzers and beepers include alarm devices, timers and confirmation of user input such as a mouse click or keystroke. A piezoelectric element may be driven by an oscillating electronic circuit or other audio signal source, driven with a piezoelectric audio amplifier. Sounds commonly used to indicate that a button has been pressed are a click, a ring or a beep.

Figure 36: Piezo Buzzer

Blank PCB

A **printed circuit board** (**PCB**) mechanically supports and electrically connects electronic components using conductive tracks, pads and other features etched from copper sheets laminated onto a non-conductive substrate. PCBs can be *single sided* (one copper layer), *double sided* (two copper layers) or *multi-layer* (outer and inner layers). Multi-layer PCBs allow for much higher component density. Conductors on different layers are connected with plated-through holes called vias. Advanced PCBs may contain components - capacitors, resistors or active devices - embedded in the substrate.

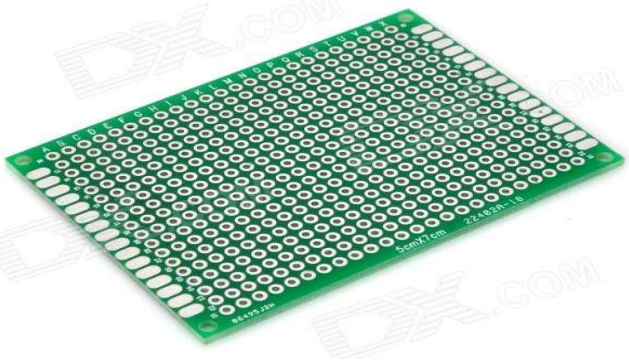


Figure 37: Blank glass epoxy PCB Board

FR-4 glass epoxy is the primary insulating substrate upon which the vast majority of rigid PCBs are produced. A thin layer of copper foil is laminated to one or both sides of an FR-4 panel. Circuitry interconnections are etched into copper layers to produce printed circuit boards. Complex circuits are produced in multiple layers.

Printed circuit boards are used in all but the simplest electronic products. Alternatives to PCBs include wire wrap and point-to-point construction. PCBs require the additional design effort to lay

out the circuit, but manufacturing and assembly can be automated. Manufacturing circuits with PCBs is cheaper and faster than with other wiring methods as components are mounted and wired with one single part. Furthermore, operator wiring errors are eliminated.

Stepper Motor

A stepper motor, also known as step motor or stepping motor, is a brushless DC electric motor that divides a full rotation into a number of equal steps. The motor's position can be commanded to move and hold at one of these steps without any position sensor for feedback (an open-loop controller), as long as the motor is correctly sized to the application in respect to torque and speed.

Figure 38: NEMA 17 stepper motor

Stepper motors effectively have multiple "toothed" electromagnets arranged as a stator around a central rotor, a gear-shaped piece of iron. The electromagnets are energized by an external driver circuit or a micro controller. To make the motor shaft turn, first, one electromagnet is given power, which magnetically attracts the gear's teeth. When the gear's teeth are aligned to the first electromagnet, they are slightly offset from the next electromagnet. This means that when the next electromagnet is turned on and the first is turned off, the gear rotates slightly to align with the next one. From there the process is repeated. Each of those rotations is called a "step", with an integer number of steps making a full rotation. In that way, the motor can be turned by a precise angle.

A4988 Stepper Motor Driver

The A4988 is a complete micro-stepping motor driver with built-in translator for easy operation. It is designed to operate bipolar stepper motors in full-, half-, quarter-, eighth-, and sixteenth-step modes, with an output drive capacity of up to 35 V and ±2 A. The A4988 includes a fixed off-time current regulator which has the ability to operate in slow or mixed decay modes.

Figure 39: A4988 Stepper motor driver

Appendix B

(Software coding)

PROGRAM CODE:

```
// Automatic Water bottle filling machine using NODE MCU///

#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define OLED_RESET -1  //14
Adafruit_SSD1306 display(OLED_RESET);

#if (SSD1306_LCDHEIGHT != 64)
#error("Height incorrect, please fix Adafruit_SSD1306.h!");
#endif

const int stepPin = 5; // A4988 step pin
const int dirPin = 4; // A4988 direction pin
int buttonPin;  // start button pin
int r;  // Limit Switch
int buzzer = 16;  // Buzzer Connected
int m = 15; // Pump Connected

void setup()
{
 buttonPin = 13; //whatever pin your START button is plugged into
 r = 12; //limit switch pin number.
 pinMode(buttonPin, INPUT_PULLUP);
 pinMode(r, INPUT);


display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
 delay(100);
 display.clearDisplay();

 display.drawRoundRect(1, 1, 126, 62, 4, WHITE);
 display.setTextSize(1);
 display.setTextColor(WHITE);
 display.setCursor(17, 4);
 // Display static text
 display.println("Automatic Bottle");
 display.setCursor(20, 15);
 display.println("Filling Machine");
 display.drawFastHLine(1, 27, 125, WHITE);
 // display.startscrollright(0x00, 0x00);
 display.setCursor(22, 37);
 display.println("Press START to");
 display.setCursor(15, 47);
 display.println("fill the bottles");
 display.display();

 // Set the two pins as Output
 pinMode(stepPin, OUTPUT);
 pinMode(dirPin, OUTPUT);
 pinMode(buzzer, OUTPUT);
 pinMode(m, OUTPUT);
 digitalWrite(m, LOW);     // initially the pump switched off

   while(digitalRead(r) ==LOW)
   {

      digitalWrite(dirPin, LOW); // Enable the motor to move in a particular direction
      // Makes 200 pulses for making one full cycle rotation
      for(int x=0; x<2000; x++)
      digitalWrite(stepPin, HIGH);
      delayMicroseconds(2000);
      digitalWrite(stepPin, LOW);
      delayMicroseconds(2000);

      if(digitalRead(r) == HIGH)
      break;
   }
   while(digitalRead(buttonPin) ==LOW)
   {
      display.setTextColor(WHITE);
      display.setCursor(22, 37);
      display.println("Press START to");
      display.setCursor(15, 47);
      display.println("fill the bottles");
      display.display();
      delay (500);
      display.setTextColor(BLACK);
      display.setCursor(22, 37);
      display.println("Press START to");
      display.setCursor(15, 47);
      display.println("fill the bottles");
      display.display();
      delay (500);
```

```
      if(digitalRead(buttonPin)      ==
HIGH)
      break;
   }
   }

void loop()
   {
   // Check button pressed, if so enter
program condition (inside if statement)
    if(digitalRead(buttonPin)
==HIGH) //functions based off of
button pulling input pin low
    {
     digitalWrite(buzzer,HIGH);
     //tone(buzzer, 800);
     delay(1000);
     //tone(buzzer, 0);
     digitalWrite(buzzer,LOW);

     display.setTextColor(BLACK);
     display.drawFastHLine(1,      27,
125, WHITE);
     //      display.startscrollright(0x00,
0x00);
     display.setCursor(22, 37);
     display.println("Press      START
to");
     display.setCursor(15, 47);
     display.println("fill the bottles");
     display.setTextColor(WHITE);
     display.setCursor(22, 37);
     display.println("Start Filling");
     display.display();


     {
      delay(500);
      display.setCursor(30, 47);
      display.println("Bottle 1/6");
      display.display();
      digitalWrite(m,HIGH);
      delay(1850);
      digitalWrite(m,LOW);
      delay(1000); //one second delay
      digitalWrite(buzzer,HIGH);
      delay(200);
      digitalWrite(buzzer,LOW);
     }
     {
      digitalWrite(dirPin, HIGH); //
Enables the motor to move in a
particular direction
      // Makes 200 pulses for making
one full cycle rotation
      for(int x=0; x<553; x++)
      {
       digitalWrite(stepPin, HIGH);
       delayMicroseconds(2000);
       digitalWrite(stepPin, LOW);
       delayMicroseconds(2000);
      }
      delay(500);
      display.setTextColor(BLACK);
      display.setCursor(30, 47);
      display.println("Bottle 1/6");
      display.display();
      display.setTextColor(WHITE);
      display.setCursor(30, 47);
      display.println("Bottle 2/6");
      display.display();
      digitalWrite(m,HIGH);
      delay(1850);
      digitalWrite(m,LOW);
      delay(1000); //one second delay
      digitalWrite(buzzer,HIGH);
      delay(200);
      digitalWrite(buzzer,LOW);
     }
     {
      {
      digitalWrite(dirPin, HIGH); //
Enables the motor to move in a
particular direction
      // Makes 200 pulses for making
one full cycle rotation
      for(int x=0; x<545; x++)
      {
       digitalWrite(stepPin, HIGH);
       delayMicroseconds(2000);
       digitalWrite(stepPin, LOW);
       delayMicroseconds(2000);
      }
      delay(500);
      display.setTextColor(BLACK);
      display.setCursor(30, 47);
      display.println("Bottle 2/6");
      display.display();
      display.setTextColor(WHITE);
      display.setCursor(30, 47);
      display.println("Bottle 3/6");
      display.display();
      digitalWrite(m,HIGH);
      delay(1850);
```

```
        digitalWrite(m,LOW);
        delay(1000); //one second delay
        digitalWrite(buzzer,HIGH);
        delay(200);
        digitalWrite(buzzer,LOW);
       }
       {
        digitalWrite(dirPin, HIGH); // Enables the motor to move in a particular direction
        // Makes 200 pulses for making one full cycle rotation
        for(int x=0; x<553; x++)
        {
         digitalWrite(stepPin, HIGH);
         delayMicroseconds(2000);
         digitalWrite(stepPin, LOW);
         delayMicroseconds(2000);
        }
        delay(500);
        display.setTextColor(BLACK);
        display.setCursor(30, 47);
        display.println("Bottle 3/6");
        display.display();
        display.setTextColor(WHITE);
        display.setCursor(30, 47);
        display.println("Bottle 4/6");
        display.display();
        digitalWrite(m,HIGH);
        delay(1850);
        digitalWrite(m,LOW);
        delay(1000); //one second delay
        digitalWrite(buzzer,HIGH);
        delay(200);
        digitalWrite(buzzer,LOW);
       }
       {
        digitalWrite(dirPin, HIGH); // Enables the motor to move in a particular direction
        // Makes 200 pulses for making one full cycle rotation
        for(int x=0; x<510; x++)
        {
         digitalWrite(stepPin, HIGH);
         delayMicroseconds(2000);
         digitalWrite(stepPin, LOW);
         delayMicroseconds(2000);
        }
        delay(500);
        display.setTextColor(BLACK);
        display.setCursor(30, 47);
        display.println("Bottle 4/6");
        display.display();
        display.setTextColor(WHITE);
        display.setCursor(30, 47);
        display.println("Bottle 5/6");
        display.display();
        digitalWrite(m,HIGH);
        delay(1850);
        digitalWrite(m,LOW);
        delay(1000); //one second delay
        digitalWrite(buzzer,HIGH);
        delay(200);
        digitalWrite(buzzer,LOW);
       }
       {
        digitalWrite(dirPin, HIGH); // Enables the motor to move in a particular direction
        // Makes 200 pulses for making one full cycle rotation
        for(int x=0; x<510; x++)
        {
         digitalWrite(stepPin, HIGH);
         delayMicroseconds(2000);
         digitalWrite(stepPin, LOW);
         delayMicroseconds(2000);
        }
        delay(500);
        display.setTextColor(BLACK);
        display.setCursor(30, 47);
        display.println("Bottle 5/6");
        display.display();
        display.setTextColor(WHITE);
        display.setCursor(30, 47);
        display.println("Bottle 6/6");
        display.display();
        digitalWrite(m,HIGH);
        delay(1850);
        digitalWrite(m,LOW);
        delay(1000); //one second delay
        digitalWrite(buzzer,HIGH);
// at the end buzzer bips 3 times
        delay(200);
        digitalWrite(buzzer,LOW);
        delay(200);
        digitalWrite(buzzer,HIGH);
        delay(200);
        digitalWrite(buzzer,LOW);
        delay(200);
        digitalWrite(buzzer,HIGH);
        delay(200);
        digitalWrite(buzzer,LOW);
       }
       {

        delay(500);
```

```
display.setTextColor(BLACK);
display.setCursor(22, 37);
display.println("Start Filling");
display.setCursor(30, 47);
display.println("Bottle 6/6");
display.display();
display.setTextColor(WHITE);
display.setCursor(35, 37);
display.println("Job Done");
        display.setCursor(10, 47);
        display.println("Replace the
Bottles");
        display.display();
        delay (2000);
       }
     }
   }
 }
```

Appendix C

(Data sheets)

ESP32-WROOM-32 (ESP-WROOM-32)

Datasheet

Version 2.4

Espressif Systems

1. Overview

ESP32-WROOM-32 (ESP-WROOM-32) is a powerful, generic Wi-Fi+BT+BLE MCU module that targets a wide variety of applications, ranging from low-power sensor networks to the most demanding tasks, such as voice encoding, music streaming and MP3 decoding.

At the core of this module is the ESP32-D0WDQ6 chip*. The chip embedded is designed to be scalable and adaptive. There are two CPU cores that can be individually controlled, and the clock frequency is adjustable from 80 MHz to 240 MHz. The user may also power off the CPU and make use of the low-power co-processor to constantly monitor the peripherals for changes or crossing of thresholds. ESP32 integrates a rich set of peripherals, ranging from capacitive touch sensors, Hall sensors, SD card interface, Ethernet, high-speed SPI, UART, I2S and I2C.

Note:

* For details on the part number of the ESP32 series, please refer to the document ESP32 Datasheet.

The integration of Bluetooth, Bluetooth LE and Wi-Fi ensures that a wide range of applications can be targeted, and that the module is future proof: using Wi-Fi allows a large physical range and direct connection to the internet through a Wi-Fi router, while using Bluetooth allows the user to conveniently connect to the phone or broadcast low energy beacons for its detection. The sleep current of the ESP32 chip is less than 5 μA, making it suitable for battery powered and wearable electronics applications. ESP32 supports a data rate of up to 150 Mbps, and 20.5 dBm output power at the antenna to ensure the widest physical range. As such the chip does offer industry-leading specifications and the best performance for electronic integration, range, power consumption, and connectivity.

The operating system chosen for ESP32 is freeRTOS with LwIP; TLS 1.2 with hardware acceleration is built in as well. Secure (encrypted) over the air (OTA) upgrade is also supported, so that developers can continually upgrade their products even after their release.

Table 1 provides the specifications of ESP32-WROOM-32 (ESP-WROOM-32).

Table 1: ESP32-WROOM-32 (ESP-WROOM-32) Specifications

Categories	Items	Specifications
Certification	RF certification	FCC/CE/IC/TELEC/KCC/SRRC/NCC
	Wi-Fi certification	Wi-Fi Alliance
	Bluetooth certification	BQB
	Green certification	RoHS/REACH
Wi-Fi	Protocols	802.11 b/g/n (802.11n up to 150 Mbps)
		A-MPDU and A-MSDU aggregation and 0.4 μs guard interval support
	Frequency range	2.4 GHz ~ 2.5 GHz
Bluetooth	Protocols	Bluetooth v4.2 BR/EDR and BLE specification
	Radio	NZIF receiver with -97 dBm sensitivity
		Class-1, class-2 and class-3 transmitter
		AFH
	Audio	CVSD and SBC

Categories	Items	Specifications
Hardware	Module interface	SD card, UART, SPI, SDIO, I2C, LED PWM, Motor PWM, I2S, IR
		GPIO, capacitive touch sensor, ADC, DAC
	On-chip sensor	Hall sensor, temperature sensor
	On-board clock	40 MHz crystal
	Operating voltage/Power supply	2.7 ~ 3.6V
	Operating current	Average: 80 mA
	Minimum current delivered by power supply	500 mA
	Operating temperature range	-40°C ~ +85°C
	Ambient temperature range	Normal temperature
	Package size	18±0.2 mm x 25.5±0.2 mm x 3.1±0.15 mm
Software	Wi-Fi mode	Station/SoftAP/SoftAP+Station/P2P
	Wi-Fi Security	WPA/WPA2/WPA2-Enterprise/WPS
	Encryption	AES/RSA/ECC/SHA
	Firmware upgrade	UART Download / OTA (download and write firmware via network or host)
	Software development	Supports Cloud Server Development / SDK for custom firmware development
	Network protocols	IPv4, IPv6, SSL, TCP/UDP/HTTP/FTP/MQTT
	User configuration	AT instruction set, cloud server, Android/iOS app

2. Pin Definitions

2.1 Pin Layout

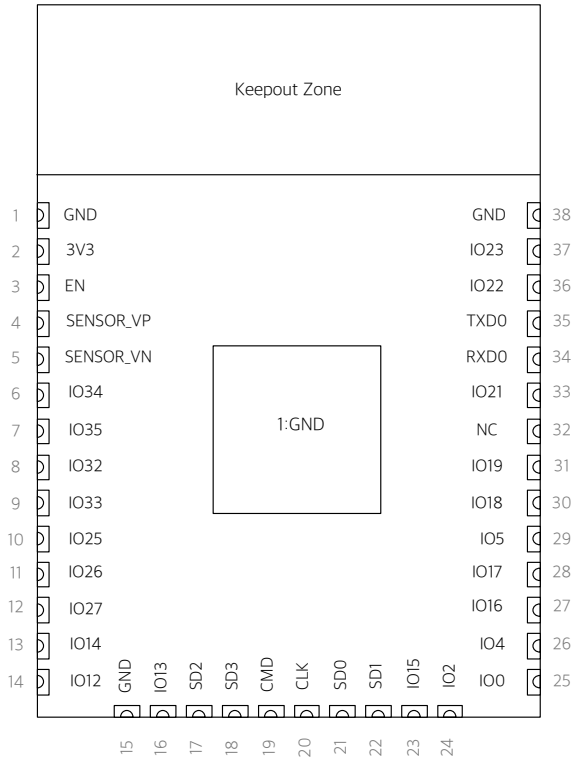


Figure 1: ESP32-WROOM-32 (ESP-WROOM-32) Pin layout

2.2 Pin Description

ESP32-WROOM-32 (ESP-WROOM-32) has 38 pins. See pin definitions in Table 2.

Table 2: Pin Definitions

Name	No.	Type	Function
GND	1	P	Ground
3V3	2	P	Power supply.
EN	3	I	Chip-enable signal. Active high.
SENSOR_VP	4	I	GPIO36, SENSOR_VP, ADC_H, ADC1_CH0, RTC_GPIO0
SENSOR_VN	5	I	GPIO39, SENSOR_VN, ADC1_CH3, ADC_H, RTC_GPIO3
IO34	6	I	GPIO34, ADC1_CH6, RTC_GPIO4
IO35	7	I	GPIO35, ADC1_CH7, RTC_GPIO5
IO32	8	I/O	GPIO32, XTAL_32K_P (32.768 kHz crystal oscillator input), ADC1_CH4, TOUCH9, RTC_GPIO9
IO33	9	I/O	GPIO33, XTAL_32K_N (32.768 kHz crystal oscillator output), ADC1_CH5, TOUCH8, RTC_GPIO8
IO25	10	I/O	GPIO25, DAC_1, ADC2_CH8, RTC_GPIO6, EMAC_RXD0
IO26	11	I/O	GPIO26, DAC_2, ADC2_CH9, RTC_GPIO7, EMAC_RXD1
IO27	12	I/O	GPIO27, ADC2_CH7, TOUCH7, RTC_GPIO17, EMAC_RX_DV

Name	No.	Type	Function
IO14	13	I/O	GPIO14, ADC2_CH6, TOUCH6, RTC_GPIO16, MTMS, HSPICLK, HS2_CLK, SD_CLK, EMAC_TXD2
IO12	14	I/O	GPIO12, ADC2_CH5, TOUCH5, RTC_GPIO15, MTDI, HSPIQ, HS2_DATA2, SD_DATA2, EMAC_TXD3
GND	15	P	Ground
IO13	16	I/O	GPIO13, ADC2_CH4, TOUCH4, RTC_GPIO14, MTCK, HSPID, HS2_DATA3, SD_DATA3, EMAC_RX_ER
SHD/SD2*	17	I/O	GPIO9, SD_DATA2, SPIHD, HS1_DATA2, U1RXD
SWP/SD3*	18	I/O	GPIO10, SD_DATA3, SPIWP, HS1_DATA3, U1TXD
SCS/CMD*	19	I/O	GPIO11, SD_CMD, SPICS0, HS1_CMD, U1RTS
SCK/CLK*	20	I/O	GPIO6, SD_CLK, SPICLK, HS1_CLK, U1CTS
SDO/SD0*	21	I/O	GPIO7, SD_DATA0, SPIQ, HS1_DATA0, U2RTS
SDI/SD1*	22	I/O	GPIO8, SD_DATA1, SPID, HS1_DATA1, U2CTS
IO15	23	I/O	GPIO15, ADC2_CH3, TOUCH3, MTDO, HSPICS0, RTC_GPIO13, HS2_CMD, SD_CMD, EMAC_RXD3
IO2	24	I/O	GPIO2, ADC2_CH2, TOUCH2, RTC_GPIO12, HSPIWP, HS2_DATA0, SD_DATA0
IO0	25	I/O	GPIO0, ADC2_CH1, TOUCH1, RTC_GPIO11, CLK_OUT1, EMAC_TX_CLK
IO4	26	I/O	GPIO4, ADC2_CH0, TOUCH0, RTC_GPIO10, HSPIHD, HS2_DATA1, SD_DATA1, EMAC_TX_ER
IO16	27	I/O	GPIO16, HS1_DATA4, U2RXD, EMAC_CLK_OUT
IO17	28	I/O	GPIO17, HS1_DATA5, U2TXD, EMAC_CLK_OUT_180
IO5	29	I/O	GPIO5, VSPICS0, HS1_DATA6, EMAC_RX_CLK
IO18	30	I/O	GPIO18, VSPICLK, HS1_DATA7
IO19	31	I/O	GPIO19, VSPIQ, U0CTS, EMAC_TXD0
NC	32	-	-
IO21	33	I/O	GPIO21, VSPIHD, EMAC_TX_EN
RXD0	34	I/O	GPIO3, U0RXD, CLK_OUT2
TXD0	35	I/O	GPIO1, U0TXD, CLK_OUT3, EMAC_RXD2
IO22	36	I/O	GPIO22, VSPIWP, U0RTS, EMAC_TXD1
IO23	37	I/O	GPIO23, VSPID, HS1_STROBE
GND	38	P	Ground

Note:

* Pins SCK/CLK, SDO/SD0, SDI/SD1, SHD/SD2, SWP/SD3 and SCS/CMD, namely, GPIO6 to GPIO11 are connected to the integrated SPI flash integrated on ESP32-WROOM-32 (ESP-WROOM-32) and are not recommended for other uses.

2.3 Strapping Pins

ESP32 has five strapping pins, which can be seen in Chapter 6 Schematics:

- MTDI
- GPIO0
- GPIO2
- MTDO
- GPIO5

Software can read the value of these five bits from the register "GPIO_STRAPPING".

During the chip's system reset (power-on reset, RTC watchdog reset and brownout reset), the latches of the strapping pins sample the voltage level as strapping bits of "0" or "1", and hold these bits until the chip is powered down or shut down. The strapping bits configure the device boot mode, the operating voltage of VDD_SDIO and other system initial settings.

Each strapping pin is connected with its internal pull-up/pull-down during the chip reset. Consequently, if a strapping pin is unconnected or the connected external circuit is high-impendence, the internal weak pull-up/pull-down will determine the default input level of the strapping pins.

To change the strapping bit values, users can apply the external pull-down/pull-up resistances, or apply the host MCU's GPIOs to control the voltage level of these pins when powering on ESP32.

After reset, the strapping pins work as the normal functions pins.

Refer to Table 3 for detailed boot modes' configuration by strapping pins.

Table 3: Strapping Pins

Voltage of Internal LDO (VDD_SDIO)					
Pin	Default	3.3V		1.8V	
MTDI	Pull-down	0		1	
Booting Mode					
Pin	Default	SPI Boot		Download Boot	
GPIO0	Pull-up	1		0	
GPIO2	Pull-down	Don't-care		0	
Debugging Log Printed on U0TXD During Booting?					
Pin	Default	U0TXD Toggling		U0TXD Silent	
MTDO	Pull-up	1		0	
Timing of SDIO Slave					
Pin	Default	Falling-edge Input Falling-edge Output	Falling-edge Input Rising-edge Output	Rising-edge Input Falling-edge Output	Rising-edge Input Rising-edge Output
MTDO	Pull-up	0	0	1	1
GPIO5	Pull-up	0	1	0	1

Note:
Firmware can configure register bits to change the settings of "Voltage of Internal LDO (VDD_SDIO)" and "Timing of SDIO Slave" after booting.

3. Functional Description

This chapter describes the modules and functions integrated in ESP32-WROOM-32 (ESP-WROOM-32).

3.1 CPU and Internal Memory

ESP32-D0WDQ6 contains two low-power Xtensa® 32-bit LX6 microprocessors. The internal memory includes:

- 448 kB of ROM for booting and core functions.
- 520 kB (8 kB RTC FAST Memory included) of on-chip SRAM for data and instruction.
  - 8 kB of SRAM in RTC, which is called RTC FAST Memory and can be used for data storage; it is accessed by the main CPU during RTC Boot from the Deep-sleep mode.
- 8 kB of SRAM in RTC, which is called RTC SLOW Memory and can be accessed by the co-processor during the Deep-sleep mode.
- 1 kbit of eFuse, of which 320 bits are used for the system (MAC address and chip configuration) and the remaining 704 bits are reserved for customer applications, including Flash-Encryption and Chip-ID.

3.2 External Flash and SRAM

ESP32 supports up to four 16-MB of external QSPI flash and SRAM with hardware encryption based on AES to protect developers' programs and data.

ESP32 can access the external QSPI flash and SRAM through high-speed caches.

- Up to 16 MB of external flash are memory-mapped onto the CPU code space, supporting 8, 16 and 32-bit access. Code execution is supported.
- Up to 8 MB of external flash/SRAM are memory-mapped onto the CPU data space, supporting 8, 16 and 32-bit access. Data-read is supported on the flash and SRAM. Data-write is supported on the SRAM.

ESP32-WROOM-32 (ESP-WROOM-32) integrates 4 MB of external SPI flash. The 4-MB SPI flash can be memory-mapped onto the CPU code space, supporting 8, 16 and 32-bit access. Code execution is supported. The integrated SPI flash is connected to GPIO6, GPIO7, GPIO8, GPIO9, GPIO10 and GPIO11. These six pins cannot be used as regular GPIO.

3.3 Crystal Oscillators

The ESP32 Wi-Fi/BT firmware can only support 40 MHz crystal oscillator for now.

3.4 RTC and Low-Power Management

With the use of advanced power management technologies, ESP32 can switch between different power modes.

- Power modes
  - Active mode: The chip radio is powered on. The chip can receive, transmit, or listen.
  - Modem-sleep mode: The CPU is operational and the clock is configurable. The Wi-Fi/Bluetooth baseband and radio are disabled.
  - Light-sleep mode: The CPU is paused. The RTC memory and RTC peripherals, as well as the ULP co-processor are running. Any wake-up events (MAC, host, RTC timer, or external interrupts) will wake up the chip.
  - Deep-sleep mode: Only the RTC memory and RTC peripherals are powered on. Wi-Fi and Bluetooth connection data are stored in the RTC memory. The ULP co-processor can work.
  - Hibernation mode: The internal 8-MHz oscillator and ULP co-processor are disabled. The RTC recovery memory is powered down. Only one RTC timer on the slow clock and some RTC GPIOs are active. The RTC timer or the RTC GPIOs can wake up the chip from the Hibernation mode.

The power consumption varies with different power modes/sleep patterns and work statuses of functional modules. Please see Table 4 for details.

Table 4: Power Consumption by Power Modes

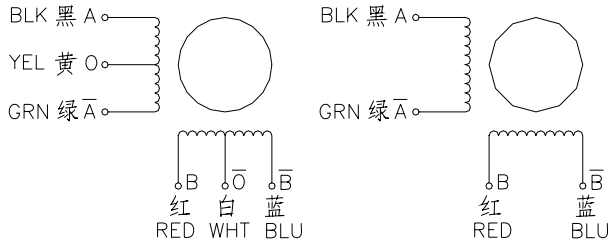
Power mode	Description	Power consumption
Active (RF working)	Wi-Fi TX packet 14 dBm ~ 19.5 dBm	Please refer to ESP32 Datasheet.
	Wi-Fi / BT TX packet 0 dBm	
	Wi-Fi / BT RX and listening	
	Association sleep pattern (by Light-sleep)	1 mA ~ 4 mA @DTIM3
Modem-sleep	The CPU is powered on.	Max speed 240 MHz: 30 mA ~ 50 mA
		Normal speed 80 MHz: 20 mA ~ 25 mA
		Slow speed 2 MHz: 2 mA ~ 4 mA
Light-sleep	-	0.8 mA
Deep-sleep	The ULP co-processor is powered on.	150 μA
	ULP sensor-monitored pattern	100 μA @1% duty
	RTC timer + RTC memory	10 μA
Hibernation	RTC timer only	5 μA
Power off	CHIP_PU is set to low level, the chip is powered off	0.1 μA

Note:

- When Wi-Fi is enabled, the chip switches between Active and Modem-sleep mode. Therefore, power consumption changes accordingly.
- In Modem-sleep mode, the CPU frequency changes automatically. The frequency depends on the CPU load and the peripherals used.
- During Deep-sleep, when the ULP co-processor is powered on, peripherals such as GPIO and I2C are able to work.
- When the system works in the ULP sensor-monitored pattern, the ULP co-processor works with the ULP sensor periodically; ADC works with a duty cycle of 1%, so the power consumption is 100 μA.

2 Phase Hybrid Stepper Motor
17HS series-Size 42mm(1.8 degree)

Wiring Diagram:



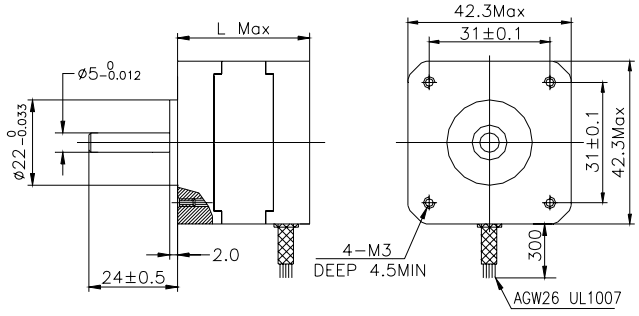
UNI-POLAR(6 LEADS) BI-POLAR(4LEADS)

Electrical Specifications:

Series Model	Step Angle (deg)	Motor Length (mm)	Rated Current (A)	Phase Resistance (ohm)	Phase Inductance (mH)	Holding Torque (N.cm Min)	Detent Torque (N.cm Max)	Rotor Inertia (g.cm²)	Lead Wire (No.)	Motor Weight (g)
17HS2408	1.8	28	0.6	8	10	12	1.6	34	4	150
17HS3401	1.8	34	1.3	2.4	2.8	28	1.6	34	4	220
17HS3410	1.8	34	1.7	1.2	1.8	28	1.6	34	4	220
17HS3430	1.8	34	0.4	30	35	28	1.6	34	4	220
17HS3630	1.8	34	0.4	30	18	21	1.6	34	6	220
17HS3616	1.8	34	0.16	75	40	14	1.6	34	6	220
17HS4401	1.8	40	1.7	1.5	2.8	40	2.2	54	4	280
17HS4402	1.8	40	1.3	2.5	5.0	40	2.2	54	4	280
17HS4602	1.8	40	1.2	3.2	2.8	28	2.2	54	6	280
17HS4630	1.8	40	0.4	30	28	28	2.2	54	6	280
17HS8401	1.8	48	1.7	1.8	3.2	52	2.6	68	4	350
17HS8402	1.8	48	1.3	3.2	5.5	52	2.6	68	4	350
17HS8403	1.8	48	2.3	1.2	1.6	46	2.6	68	4	350
17HS8630	1.8	48	0.4	30	38	34	2.6	68	6	350

***Note:** We can manufacture products according to customer's requirements.

Dimensions: unit=mm



Motor Length:

Model	Length
17HS2XXX	28 mm
17HS3XXX	34 mm
16HS4XXX	40 mm
16HS8XXX	48 mm

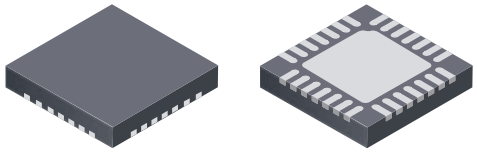
DMOS Microstepping Driver with Translator And Overcurrent Protection

Features and Benefits

- Low $R_{DS(ON)}$ outputs
- Automatic current decay mode detection/selection
- Mixed and Slow current decay modes
- Synchronous rectification for low power dissipation
- Internal UVLO
- Crossover-current protection
- 3.3 and 5 V compatible logic supply
- Thermal shutdown circuitry
- Short-to-ground protection
- Shorted load protection
- Five selectable step modes: full, $^1/_2$, $^1/_4$, $^1/_8$, and $^1/_{16}$

Package:

28-contact QFN
with exposed thermal pad
5 mm × 5 mm × 0.90 mm
(ET package)



Approximate size

Description

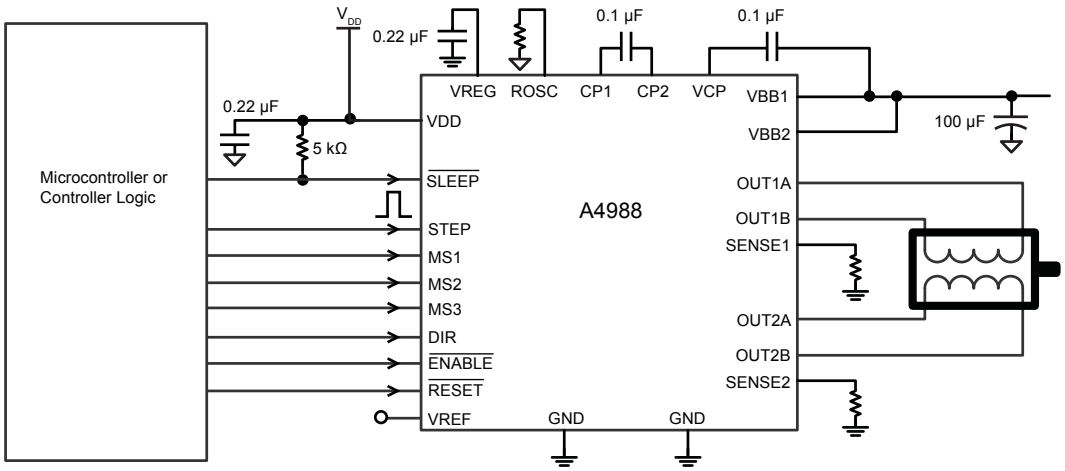
The A4988 is a complete microstepping motor driver with built-in translator for easy operation. It is designed to operate bipolar stepper motors in full-, half-, quarter-, eighth-, and sixteenth-step modes, with an output drive capacity of up to 35 V and ±2 A. The A4988 includes a fixed off-time current regulator which has the ability to operate in Slow or Mixed decay modes.

The translator is the key to the easy implementation of the A4988. Simply inputting one pulse on the STEP input drives the motor one microstep. There are no phase sequence tables, high frequency control lines, or complex interfaces to program. The A4988 interface is an ideal fit for applications where a complex microprocessor is unavailable or is overburdened.

During stepping operation, the chopping control in the A4988 automatically selects the current decay mode, Slow or Mixed. In Mixed decay mode, the device is set initially to a fast decay for a proportion of the fixed off-time, then to a slow decay for the remainder of the off-time. Mixed decay current control results in reduced audible motor noise, increased step accuracy, and reduced power dissipation.

Continued on the next page…

Typical Application Diagram



Description (continued)

Internal synchronous rectification control circuitry is provided to improve power dissipation during PWM operation. Internal circuit protection includes: thermal shutdown with hysteresis, undervoltage lockout (UVLO), and crossover-current protection. Special power-on sequencing is not required.

The A4988 is supplied in a surface mount QFN package (ES), 5 mm × 5 mm, with a nominal overall package height of 0.90 mm and an exposed pad for enhanced thermal dissipation. It is lead (Pb) free (suffix –T), with 100% matte tin plated leadframes.

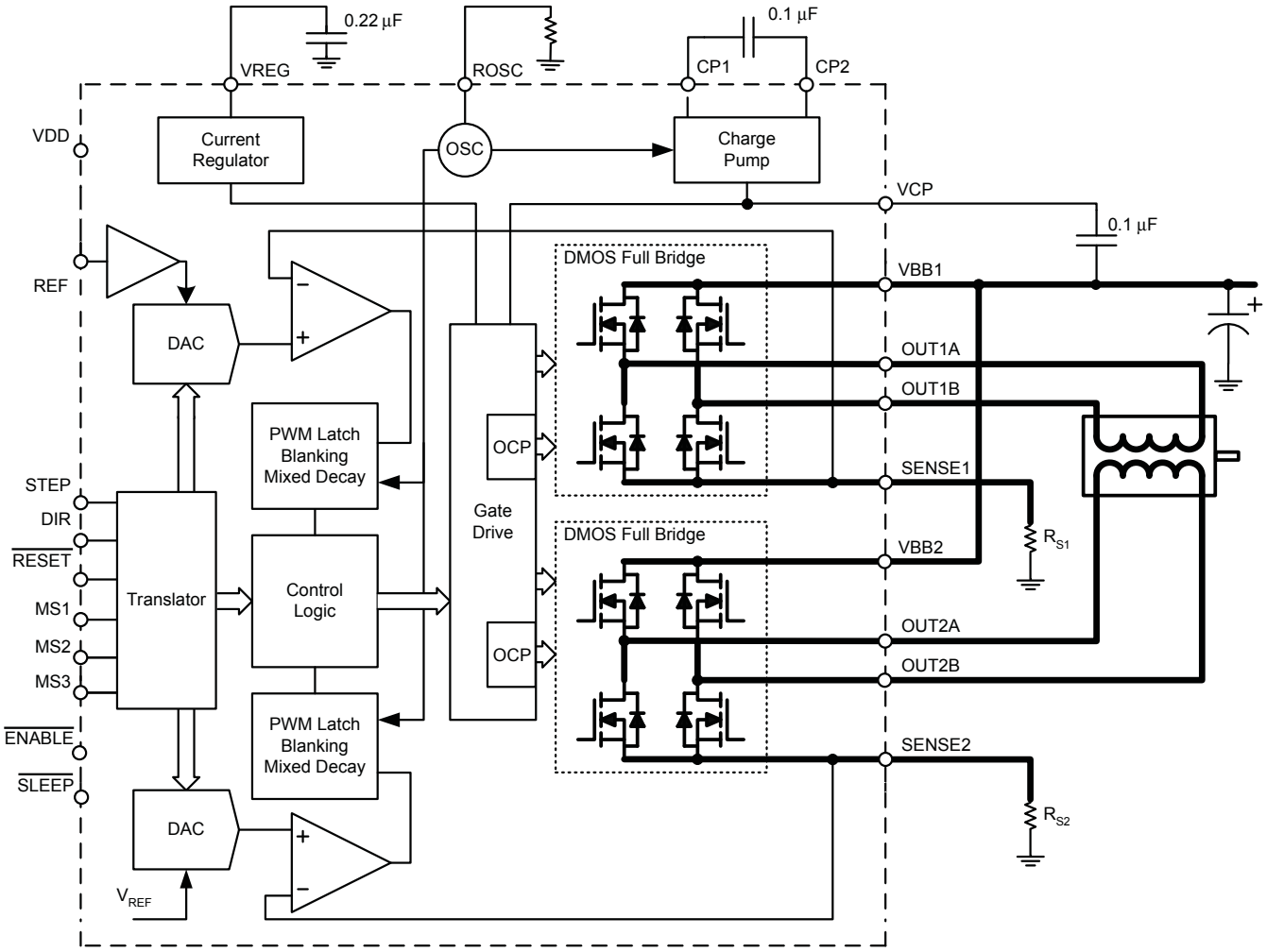
Selection Guide

Part Number	Package	Packing
A4988SETTR-T	28-contact QFN with exposed thermal pad	1500 pieces per 7-in. reel

Absolute Maximum Ratings

Characteristic	Symbol	Notes	Rating	Units
Load Supply Voltage	V_{BB}		35	V
Output Current	I_{OUT}		±2	A
Logic Input Voltage	V_{IN}		–0.3 to 5.5	V
Logic Supply Voltage	V_{DD}		–0.3 to 5.5	V
Motor Outputs Voltage			–2.0 to 37	V
Sense Voltage	V_{SENSE}		–0.5 to 0.5	V
Reference Voltage	V_{REF}		5.5	V
Operating Ambient Temperature	T_A	Range S	–20 to 85	ºC
Maximum Junction	T_J(max)		150	ºC
Storage Temperature	T_{stg}		–55 to 150	ºC

Functional Block Diagram



ELECTRICAL CHARACTERISTICS[1] at T_A = 25°C, V_{BB} = 35 V (unless otherwise noted)

Characteristics	Symbol	Test Conditions	Min.	Typ.[2]	Max.	Units
Output Drivers						
Load Supply Voltage Range	V_{BB}	Operating	8	–	35	V
Logic Supply Voltage Range	V_{DD}	Operating	3.0	–	5.5	V
Output On Resistance	R_{DSON}	Source Driver, I_{OUT} = –1.5 A	–	320	430	mΩ
		Sink Driver, I_{OUT} = 1.5 A	–	320	430	mΩ
Body Diode Forward Voltage	V_F	Source Diode, I_F = –1.5 A	–	–	1.2	V
		Sink Diode, I_F = 1.5 A	–	–	1.2	V
Motor Supply Current	I_{BB}	f_{PWM} < 50 kHz	–	–	4	mA
		Operating, outputs disabled	–	–	2	mA
Logic Supply Current	I_{DD}	f_{PWM} < 50 kHz	–	–	8	mA
		Outputs off	–	–	5	mA
Control Logic						
Logic Input Voltage	$V_{IN(1)}$		V_{DD}×0.7	–	–	V
	$V_{IN(0)}$		–	–	V_{DD}×0.3	V
Logic Input Current	$I_{IN(1)}$	V_{IN} = V_{DD}×0.7	–20	<1.0	20	µA
	$I_{IN(0)}$	V_{IN} = V_{DD}×0.3	–20	<1.0	20	µA
Microstep Select	R_{MS1}	MS1 pin	–	100	–	kΩ
	R_{MS2}	MS2 pin	–	50	–	kΩ
	R_{MS3}	MS3 pin	–	100	–	kΩ
Logic Input Hysteresis	$V_{HYS(IN)}$	As a % of V_{DD}	5	11	19	%
Blank Time	t_{BLANK}		0.7	1	1.3	µs
Fixed Off-Time	t_{OFF}	OSC = VDD or GND	20	30	40	µs
		R_{OSC} = 25 kΩ	23	30	37	µs
Reference Input Voltage Range	V_{REF}		0	–	4	V
Reference Input Current	I_{REF}		–3	0	3	µA
Current Trip-Level Error[3]	err_I	V_{REF} = 2 V, %$I_{TripMAX}$ = 38.27%	–	–	±15	%
		V_{REF} = 2 V, %$I_{TripMAX}$ = 70.71%	–	–	±5	%
		V_{REF} = 2 V, %$I_{TripMAX}$ = 100.00%	–	–	±5	%
Crossover Dead Time	t_{DT}		100	475	800	ns
Protection						
Overcurrent Protection Threshold[4]	I_{OCPST}		2.1	–	–	A
Thermal Shutdown Temperature	T_{TSD}		–	165	–	°C
Thermal Shutdown Hysteresis	T_{TSDHYS}		–	15	–	°C
VDD Undervoltage Lockout	V_{DDUVLO}	V_{DD} rising	2.7	2.8	2.9	V
VDD Undervoltage Hysteresis	$V_{DDUVLOHYS}$		–	90	–	mV

[1]For input and output current specifications, negative current is defined as coming out of (sourcing) the specified device pin.

[2]Typical data are for initial design estimations only, and assume optimum manufacturing and application conditions. Performance may vary for individual units, within the specified maximum and minimum limits.

[3]V_{ERR} = [(V_{REF}/8) – V_{SENSE}] / (V_{REF}/8).

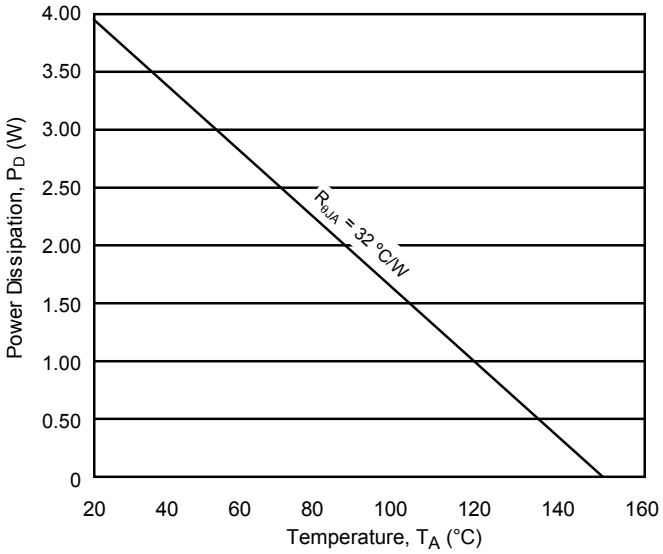
[4]Overcurrent protection (OCP) is tested at T_A = 25°C in a restricted range and guaranteed by characterization.

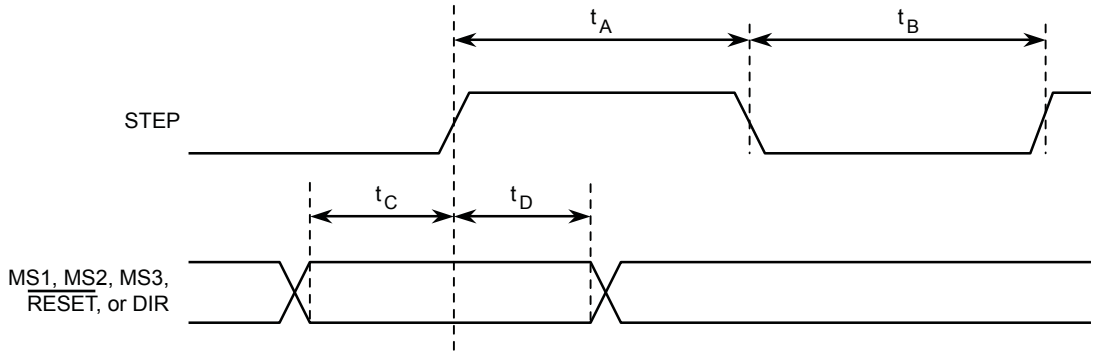
THERMAL CHARACTERISTICS

Characteristic	Symbol	Test Conditions*	Value	Units
Package Thermal Resistance	$R_{\theta JA}$	Four-layer PCB, based on JEDEC standard	32	ºC/W

*Additional thermal information available on Allegro Web site.

Power Dissipation versus Ambient Temperature





Time Duration	Symbol	Typ.	Unit
STEP minimum, HIGH pulse width	t_A	1	µs
STEP minimum, LOW pulse width	t_B	1	µs
Setup time, input change to STEP	t_C	200	ns
Hold time, input change to STEP	t_D	200	ns

Figure 1. Logic Interface Timing Diagram

Table 1. Microstepping Resolution Truth Table

MS1	MS2	MS3	Microstep Resolution	Excitation Mode
L	L	L	Full Step	2 Phase
H	L	L	Half Step	1-2 Phase
L	H	L	Quarter Step	W1-2 Phase
H	H	L	Eighth Step	2W1-2 Phase
H	H	H	Sixteenth Step	4W1-2 Phase