Color Sorting Machine using Micro-controller

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

by

SOUMYADIP BARMAN (11701618022)
SUBRATA MONDAL (11701618015)
ABHIJNAN SARKAR (11701618067)

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 **Color Sorting Machine using Micro-controller** is the bonafide work carried out by **SOUMYADIP BARMAN (11701618022), SUBRATA MONDAL (11701618015) and ABHIJNAN SARKAR (11701618067)**, 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 2016-17, 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

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

Countersigned by

Mondal

(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 also 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 Sir** 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 **Mr. Nijam Uddin Molla Sir** to help us in our soldering process to build the circuit.

We are very thankful 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.

SOUMYADIP BARMAN (11701618022) Soumyadip Barman

SUBRATA MONDAL (11701618015) Subrata Mondal

ABHIJNAN SARKAR (11701618067) Abhijnan Sarkar

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 "**Color Sorting 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,

SOUMYADIP BARMAN (11701618022) Soumyadip Barman

SUBRATA MONDAL (11701618015) Subrata Mondal

ABHIJNAN SARKAR (11701618067) Abhijnan Sarkar

Contents

List of Figures

List of Tables

ABBREVIATIONS AND ACRONYMS

SCL – Serial Clock
SDA – Serial Data
SoC – System on a chip
IC - Integrated Circuit
PCB – Printed Circuit Board
µC – Micro Controller
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
UART - Universal asynchronous receiver-transmitter
PWM – Pulse Width Modulation
ULP – Ultra Low Power Processor
MS – Micro Step
RST – Reset
SLP - Sleep

ABSTRACT

Machines can perform highly repetitive tasks better than humans. Worker fatigue on assembly lines can result in reduced performance, and cause challenges in maintaining product quality. An employee who has been performing an inspection task over and over again may eventually fail to recognize the colour of product. Automating many of the tasks in the industries may help to improv the efficiency of manufacturing system. The purpose of this prototype is to design and implement a system which automatically separates products based on their colour. This machine consists of four parts: rotating platform, colour sensor servo motor and stepper motor. The output and input of these part was interfaced using Arduino/Node MCU/ESP32 microcontroller

Sorting of products is a very difficult industrial process. Continuous manual sorting creates consistency issues. This prototype designed for automatic sorting of objects based on the colour. TCS230 sensor will be used to detect the colour of the product and the microcontroller will be used to control the overall process. The identification of the colour is based on the frequency analysis of the output of TCS230 sensor. One rotating platform will be used to bring the product in front of the colour sensor which is controlled by the stepper motor. After recognizing the colour of the product one servo motor is used to place the product in separate compartments. The experimental results promise that the prototype will fulfil the needs for higher production and precise quality in the field of automation.

CHAPTER 1

(Introduction)

1.1 INTRODUCTION

The project intends to design and implement an "**Color Sorting Machine using Micro-Controller**" using position control mechanism and color sensing technology with the help of Micro-Controller (ESP-32). A prototype has been developed to illustrate the project. In this project the stepper motor brings the color discs from the holder to exact top position of the color sensor TSC230. The color sensor TSC230 identify the color of the disc and send the RGB value of the color to the microcontroller. The microcontroller then identifies the color from the pre define value and instructs the servo motor to rotate the sliding platform to the respective color pot. In the sext rotation of the stepper motor the disc fall down to the particular color pot.

The developed prototype consists of the following main section.

- Disc holder – it holds the color disc in the vertical position just above the rotating disc positioner.
- disc positioner – it brings the color discs from the holder and place it exactly above the color sensor TSC230
- Stepper motor – it moves the disc positioner
- Servo motor – it controls the sliding platform so that the discs can accumulate to the correct color pot
- Control board – it consists of ESP32 microcontroller, stepper motor driver, voltage regulator etc. The heart of the project. Controls the stepper motor, servo motor and the color sensor TSC230.
- Color sensor – this sensor senses the color of any object and break it the RGB value

1.2 Color sorting process

In the cutting-edge-day scenario of competitive manufacturing in commercial zone performance of manufacturing holds the important component for achievement. It's miles essential to beautify manufacturing pace, lower the labour charge and reduce the breakdown time of production gadget. Merchandise should be taken care of in numerous ranges of manufacturing and manual sorting is time consuming and labour extensive. This project discusses about the automatic sorting tool which helps the sorting mechanism to kind based at the coloration. For sensing TCS230 coloration sensor has been used. With the aid of reading the frequency of the output of the sensor, colour primarily based absolutely sorting is completed. Layout of an innovative prototype referred to as item sorting system by means of spotting the only of a kind shade of the item has been leader goal of the challenge. Accumulating the objects from the hopper and distributes those objects to their accurate area based on their coloration even they'll be unique in coloration. Many paintings environments aren't suitable for manual sorting and a few areas are risky for humans to paintings on. This prototype is built as a simple digital gadgets like microcontroller for processing, Servo motors for actions and coloration sensor for recognizing exclusive-coloured devices.

As the name suggests, colour sorting is simply to sort the things according to their colour. It can be easily done by seeing it but when there are too many things to be sorted and it is a repetitive task then automatic colour sorting machines are very useful. These machines have colour sensor to sense the colour of any objects and after detecting the colour servo motor grab the thing and put it into respective pot. They can be used in different application areas where colour identification, colour distinction and colour sorting are important. Some of the application areas include Agriculture Industry (Grain Sorting on the basis of colour), Food Industry, Diamond and Mining Industry, Recycling etc. The applications are not limited to this and can be further applied to different industries.

The operation of colour sorting starts with a stepper motor. The function of the motor is to collect the colour disc from a disc holder and put the colour disc above the colour sensor TSC230. The sensor senses the colour and break the colour in RGB values and then fed the RGB values to the microcontroller. The controller identifies the colour based on RGB values and direct the stepper motor to position the sliding platform to the respective colour pot. The colour disc slides down to the respective colour pot in the next rotation of the stepper motor. Block diagram of system is as shown in Fig 1.

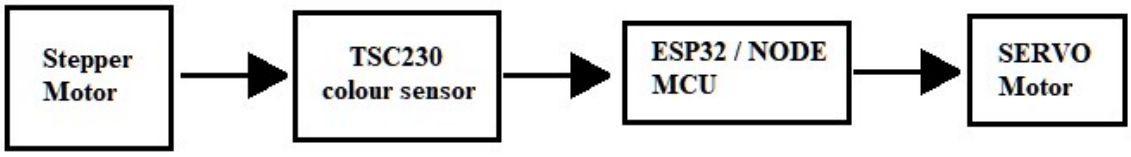


Figure 1: Basic blocks of the colour sensing and sorting process

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 colour sorting machine mainly developed using high speed colour sensing sensor, conveyor belt and driving mechanism. It includes much cost. The same can be developed using synchronized process control which eliminates the use of the driving mechanism 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, TSC230 colour sensor and servo motor 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] ***Ch. Shravani, G. Indira, V. Appalaraju, "Arduino Based Color Sorting Machine using TCS3200 Color Sensor", International Journal of Innovative Technology and Exploring Engineering (IJITEE), ISSN: 2278-3075, Volume-8, Issue- 6S4, April 2019.***

Sorting of object is an essential mechanical process in which difficult work is quite required. Chronic manual arranging makes consistency troubles. Machines can perform mainly dreary assignments superior to human beings. Laborer exhaustion on sequential manufacturing structures can result in decreased execution, and purpose troubles in retaining up object fine. A employee who has been appearing research undertaking over and over may additionally in the end forget about to recognize the color of item, but a machine in no way. On this paper a compact record close to arranging of articles based totally on shading has been implemented making use of TCS3200 shading sensor with SERVOMOTORS associated with AURDINO UNO.

[2] ***K.Sasidhar, Shahwar Farooqi, Mohammed Abdul Moin, M Sachin, "Design and Development of a Colour Sorting Machine using PLC and SCADA", International Journal of Research and Scientific Innovation (IJRSI), Volume V, Issue VII, July 2018, ISSN 2321–2705.***

The purpose of this project is to present a Programmable Logic Control (PLC) and SCADA based control system that is applied to the Colour Sorting Machine. In many industrial applications, there is a need of sorting. Sorting can be done by using many ways according to the dimensions, colours, weight, using machine vision (image processing), material of an object etc. For example, in Thermal Power Station, electromagnetic sorting technique is used to sort ferromagnetic materials from coal. This project consists of components such as PLC, SCADA software, conveyors, colour sensors, electronic system and motors. The objects are being sorted according to their respective colour. The main conveyor is supported of two branches to load the distinguished object on to the respective one as separated by the electronic system and detected by the proximity sensors. In this project, SCADA provides a user-friendly environment to establish an easy communication between humans and process. SCADA shows the activation of various parts of the system, i.e. conveyors, motors, LDRs and electronic devices.

[3] **Kunhimohammed C. K, Muhammed Saifudeen K. K, Sahna S, Gokul M. S and Shaeez Usman Abdulla, "Automatic Color Sorting Machine Using TCS230 Color Sensor And PIC Microcontroller", International Journal of Research and Innovations in Science and Technology, Volume 2, Issue 2, 2015, ISSN(Online): 2394-3858 ISSN(Print) : 2394-3866.**

Sorting of products is a very difficult industrial process. Continuous manual sorting creates consistency issues. This paper describes a working prototype designed for automatic sorting of objects based on the color. TCS230 sensor was used to detect the color of the product and the PIC16F628A microcontroller was used to control the overall process. The identification of the color is based on the frequency analysis of the output of TCS230 sensor. Two conveyor belts were used, each controlled by separate DC motors. The first belt is for placing the product to be analyzed by the color sensor, and the second belt is for moving the container, having separated compartments, in order to separate the products. The experimental results promise that the prototype will fulfill the needs for higher production and precise quality in the field of automation.

[4] **Aung Thike, Zin Zin Moe San, Dr. Zaw Min Oo, "Design and Development of an Automatic Color Sorting Machine on Belt Conveyor", International Journal of Science and Engineering Applications Volume 8–Issue 07,176-179, 2019, ISSN: -2319–7560.**

Automatic color sorting is very much convenient in industry. Color and size are the most important features for accurate classification and sorting of product which can be done by using some optical sensors or analyzing their pictures. Color sorting machines are machines that are used on the production lines in bulk food processing and other industries. They separate items by their colors, detecting the colors if things that pass before them and using mechanical or pneumatic ejection devices to divert items whose colors do not fall within the acceptable range. The Color sorting machine using Arduino is a fascinating and renowned project for techies, who would like to combine electronics, machine building and programming. The Color Sorting Machine is used for sorting mainly RGB colors. A simple robot arm is used to apply a color sorting to a physical system. The objects are placed to the conveyor belt using robot arm with servo motors. One conveyor belt is used, which is controlled by DC motors.

[5] ***Aye Myat Myat Myo, Zar Chi Soe, "Automatic Color Sorting Machine Using Arduino Mega Microcontroller", International Journal of Latest Technology in Engineering, Management & Applied Science (IJLTEMAS), Volume VIII, Issue VIII, August 2019, ISSN 2278-2540.***

In this digital world, color processing gives us a greater impact in different industries to solve the consistency issue of continuous manual sorting. This paper will be a new approach to recognize and sort the objects continuously and keep them in a designated location. Nowadays, image or colors processing attract massive attention as it leads to possibility of widening scope of application in different field with the help of modern technology. A color sorter is researched, designed and created with Arduino Mega microcontroller, TCS230 color sensor, servo motor and other electronic components. This work involves sensors that sense the object's color and sends the signal to the Arduino. The microcontroller sends signal to circuit which drives the various motors to allow the object and place it in the specified location. Based upon the detection, the hole moves to the specified location, releases the object and comes back to the original position. The system has the ability to sort the object according to their colors into respective color station in minimum time.

[6] ***Lim Jie Shen, Irda Hassan, "Design and development of colour sorting robot", Journal of Engineering Science and Technology EURECA 2014 Special Issue January (2015) 71– 81, © School of Engineering, Taylor's University.***

This paper shows a new approach for continuous recognition and sorting of objects into desired location. Image or colours processing nowadays attract massive attention as it leads to possibility of widening scope of application in different field with the help of modern technology. A colour sorting robot is researched, designed and created with Arduino Uno microcontroller, TCS3200D Colour Sensor, SG90 Tower Pro Servo Motor and other electronic components. The system has the ability to sort the object according to their colours into respective colour station in minimum time. Specific programming code for this system is written.

[7] ***Dharmannagari Vinay Kumar Reddy, "Sorting of objects based on colour by pick and place robotic arm and with conveyor belt arrangement", International Journal of mechanical engineering and robotic research (IJMERR), Vol. 3, No. 1, January 2014, ISSN 2278 – 0149.***

In many situations, autonomous robots can provide effective solutions to grueling tasks. In this case, it is desirable to create an autonomous robot that can identify objects from the conveyor belt and relocate them if the object meets certain criteria. Dealing with a large number of objects is a very menial task, this is an excellent application for a robot of this type. In this case, to keep costs and design complexity low, the robot is designed around the platform and uses several different sensors to collect information about the robots environment to allow the robot to react accordingly. This paper aims at the problem I am attempting to solve is to create an autonomous robot that can identify objects when placed on the conveyor belt based on color sensing and then sort by relocating them to a specific location. It will be using a picking arm which uses a controller motor to pick the particular object from the conveyor belt and place it according to the color sensing. Micro controller (AT89S52) allows dynamic and faster control. Liquid Crystal Display (LCD) makes the system user-friendly. AT89S52 Micro controller is the heart of the circuit as it controls all the functions.

[8] ***Mr.V.A.Aher, Mayur Dukre, Ganesh Abhang, Trupti Thorat, "Colour based object sorting machine", International Research Journal of Engineering and Technology (IRJET), Volume 08, Issue 02, Feb 2021, ISSN(print): 2395-0072, ISSN(online): 2395-0056.***

Sorting is a process in which two or more objects of similar, yet different characteristics are arranged in a systematic order. This is generally carried through manually or by using sensors in automation. Automatic color sorting is very much convenient in industry. Color and size are the most important features for accurate classification and sorting of product which can be done by using some optical sensors or analyzing their pictures. Color sorting machines are machines that are used on the production lines in bulk food processing and other industries. They separate items by their colors, detecting the colors if things that pass before them and using mechanical or pneumatic ejection devices to divert items whose colors do not fall within the acceptable range. The Color sorting machine using microcontroller is a fascinating and renowned project for techies, who would like to combine electronics, machine building and programming. The Color Sorting Machine is used for sorting mainly RGB colors. A simple robot arm is used to apply a color sorting to a physical system. The objects are placed to the conveyor belt using robot arm with servo motors. One conveyor belt is used, which is controlled by DC motors.

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.

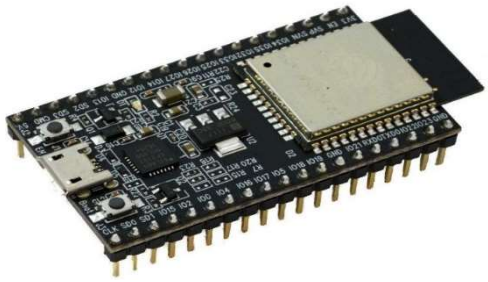


Figure 2: ESP 32 Microcontroller

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.

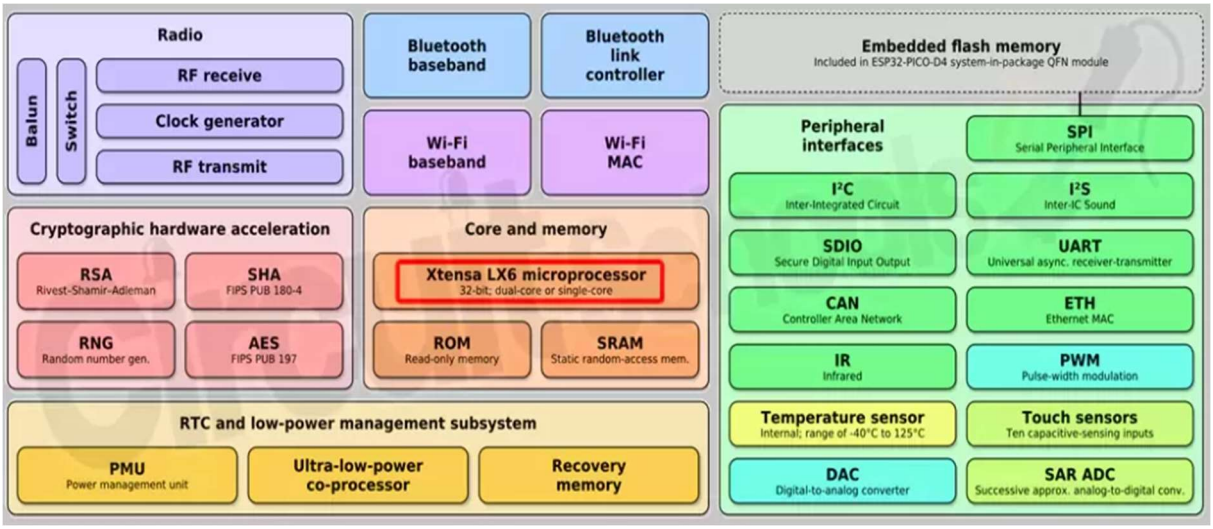


Figure 3: ESP32 Architectural Block diagram

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.

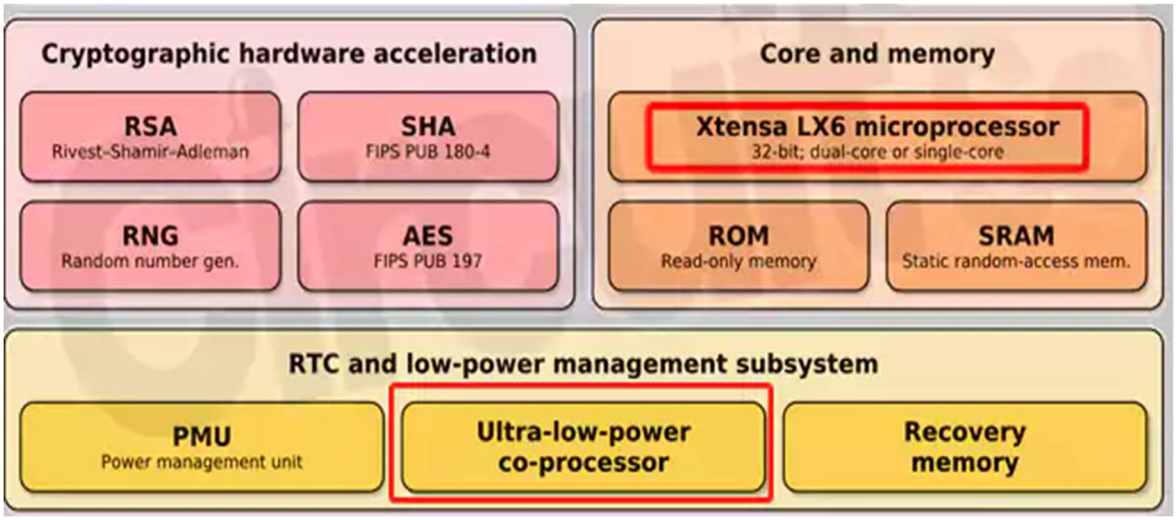


Figure 4: ESP32 memory Block diagram

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:

Table 1: 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?

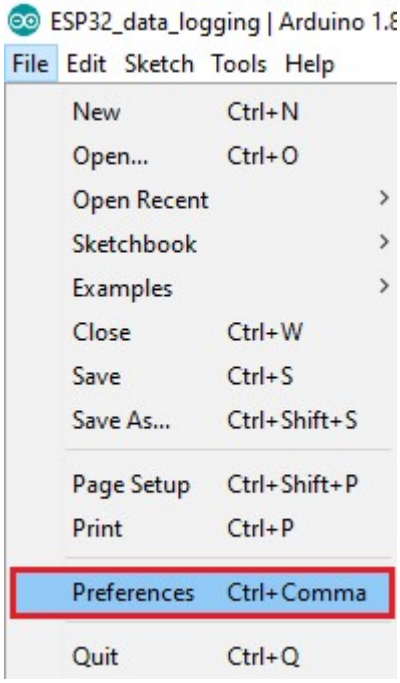
Before selecting an ESP32 development board, you need to take into account certain aspects:

- **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:

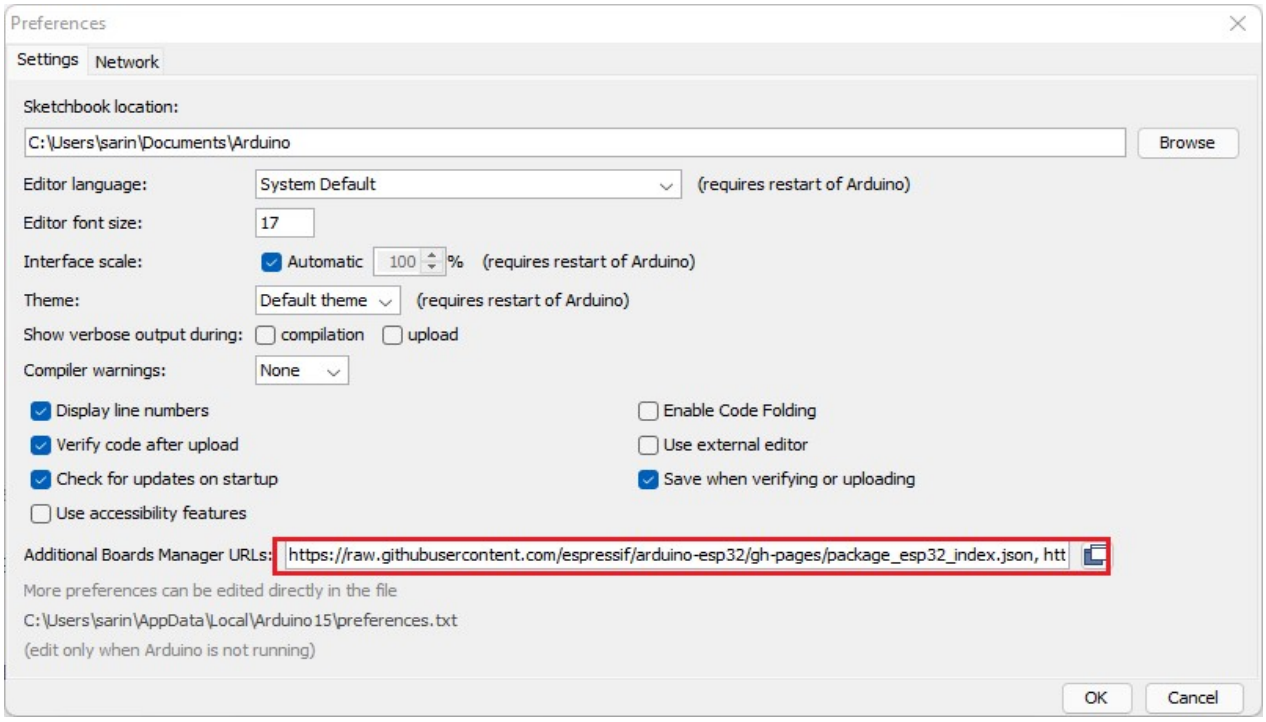
1. In your Arduino IDE, go to **File**> **Preferences**



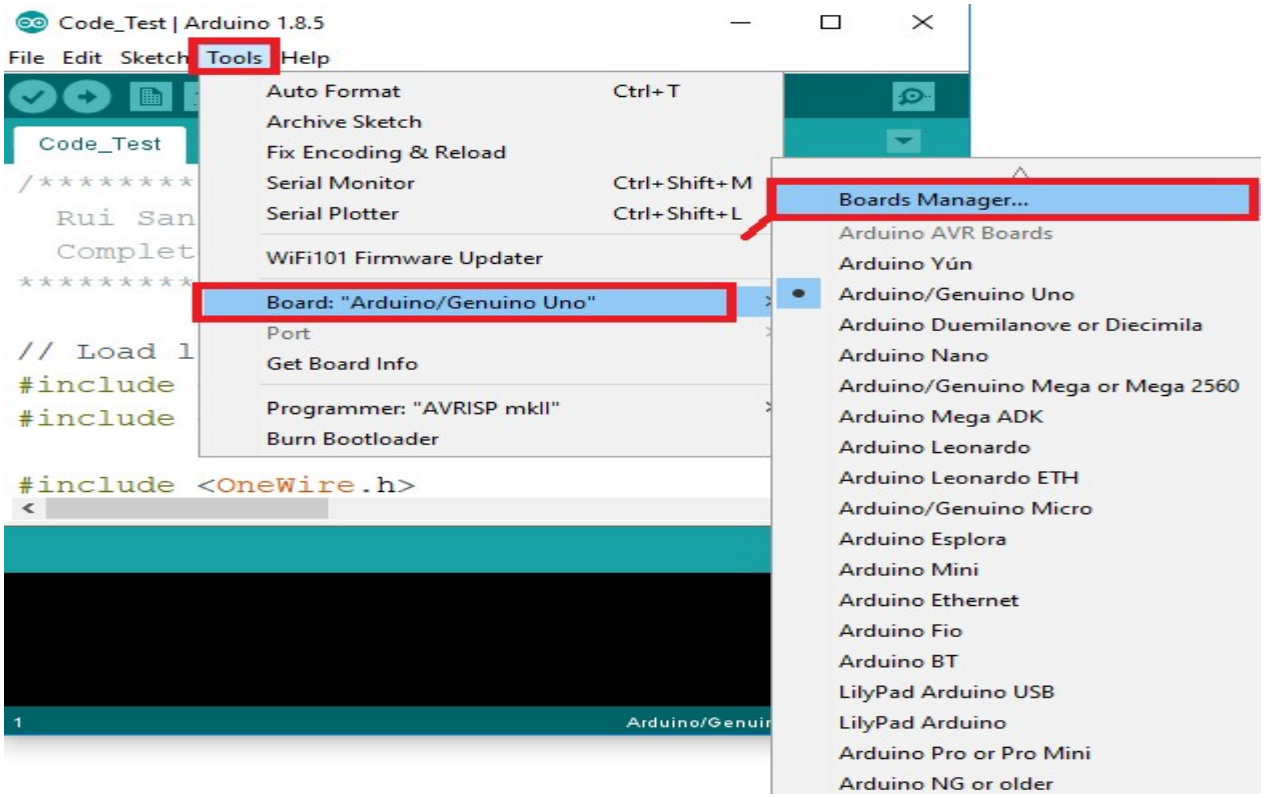
2. Enter the following into the "Additional Board Manager URLs" field:

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

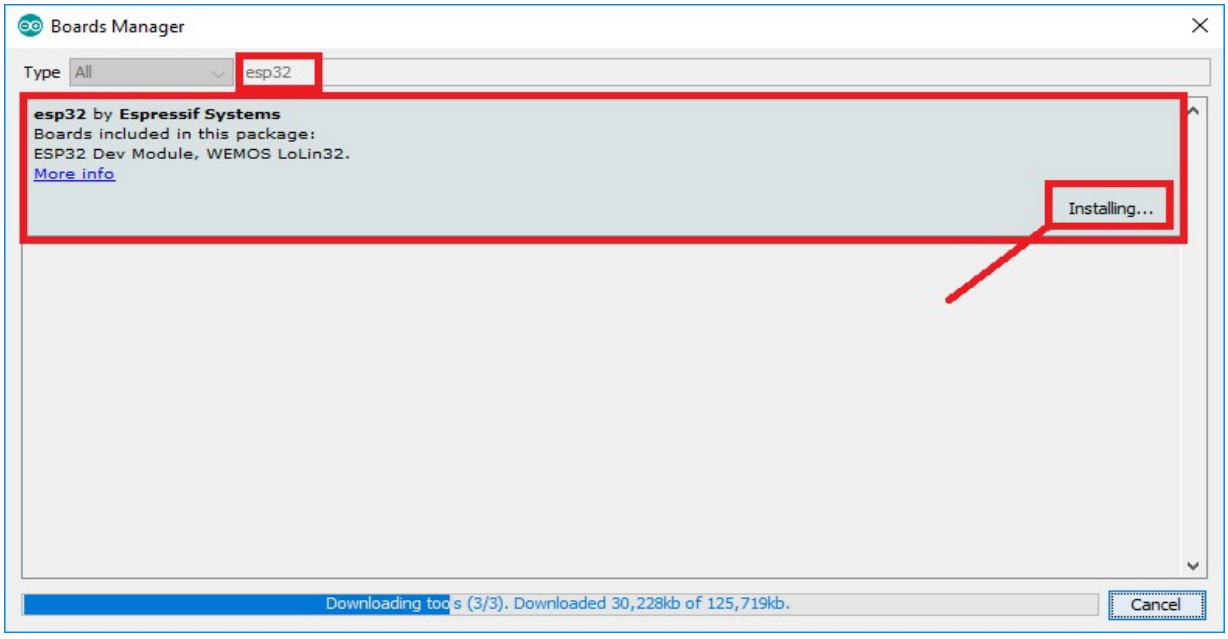
   Then, click the "OK" button:



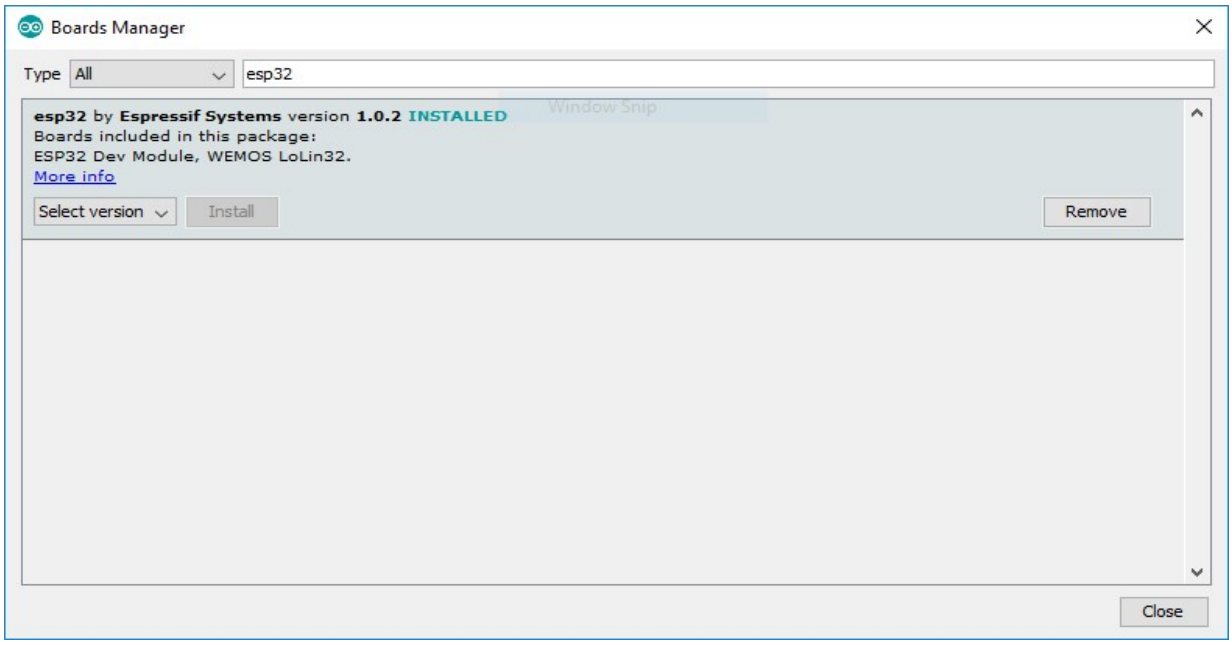
3. Open the Boards Manager. Go to **Tools** > **Board** > **Boards Manager…**



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.

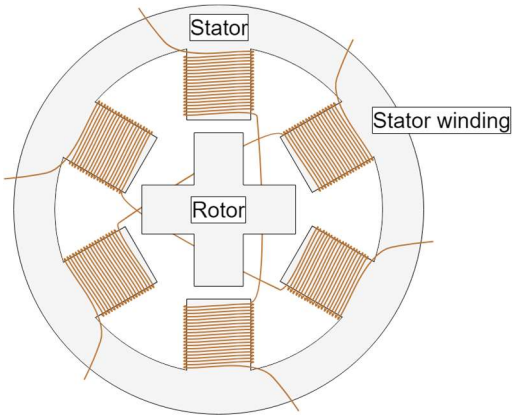


Figure 5: Cross-Section of a Stepper Motor

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.

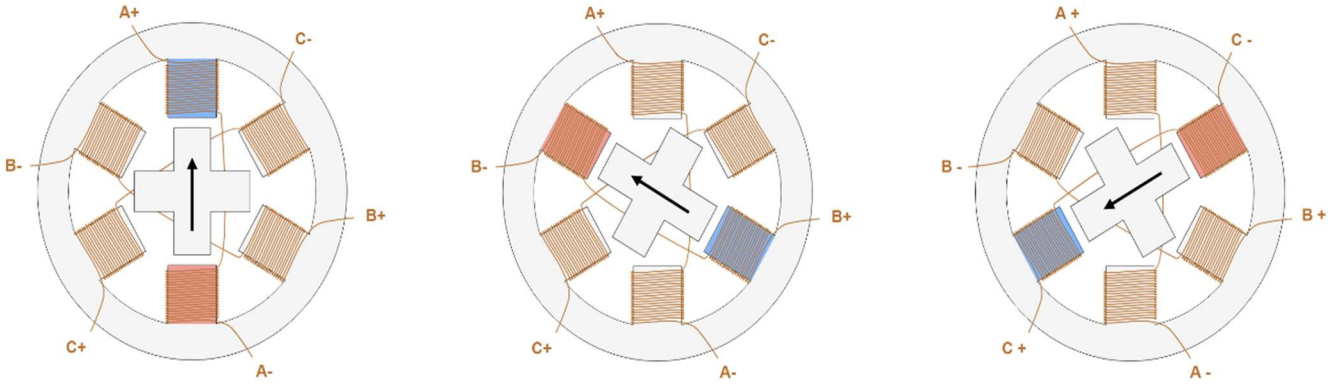


Figure 6: Stepper Motor Steps

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**.

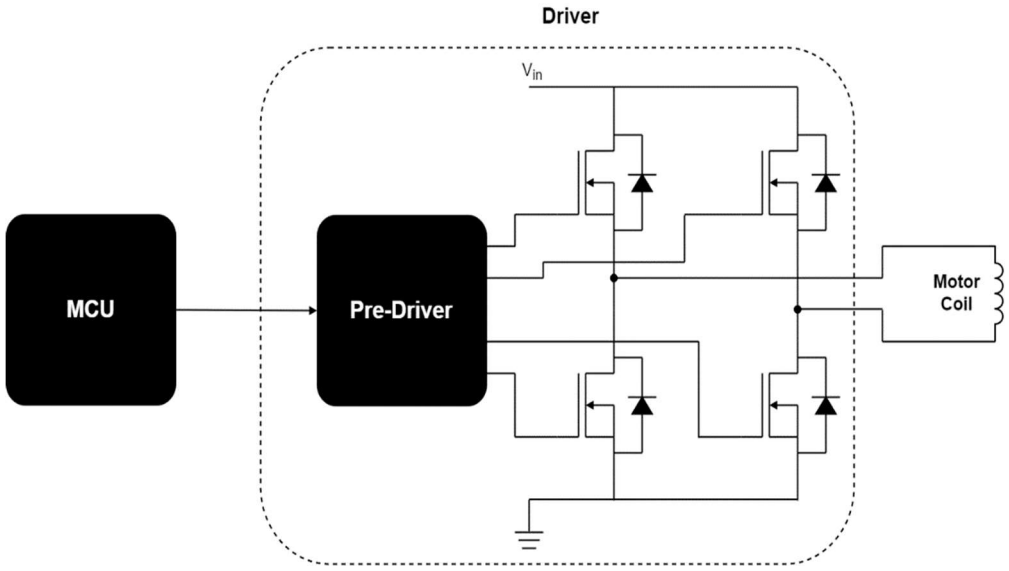


Figure 7: Motor Control Basic Scheme

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.

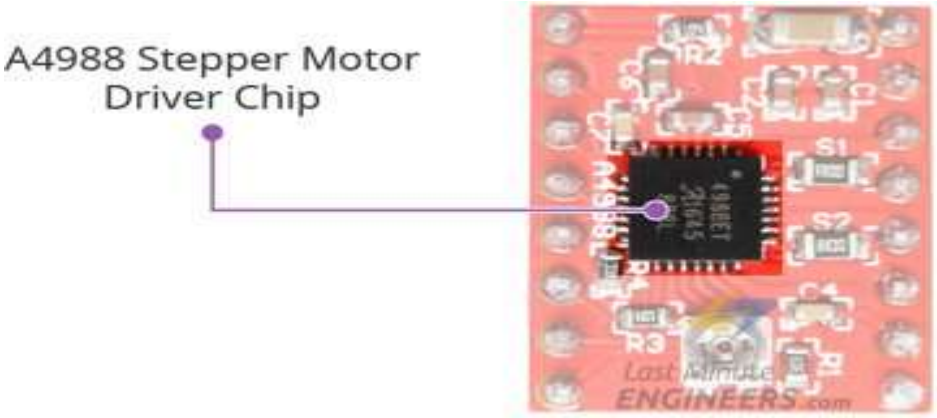


Figure 8: Stepper Motor Driver A4988

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:

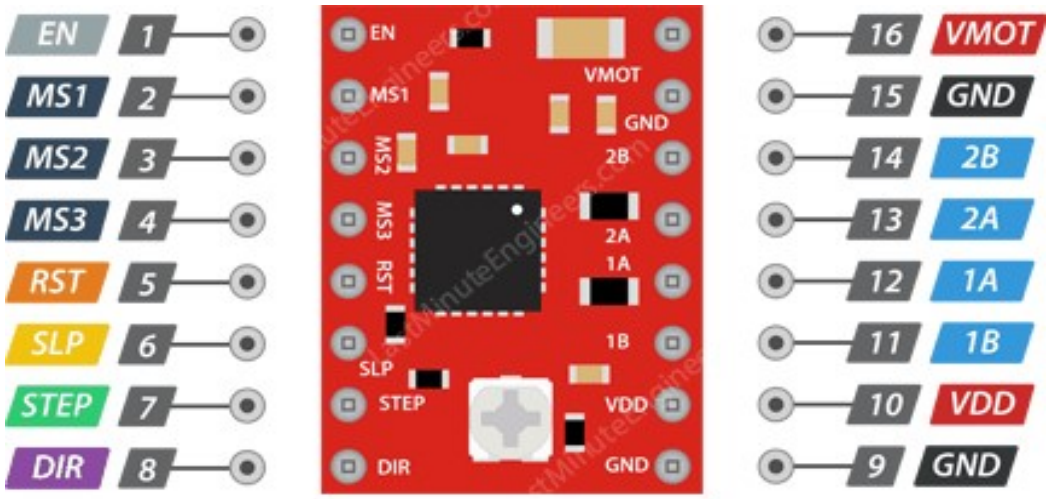


Figure 9: A4988 pin diagram

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.

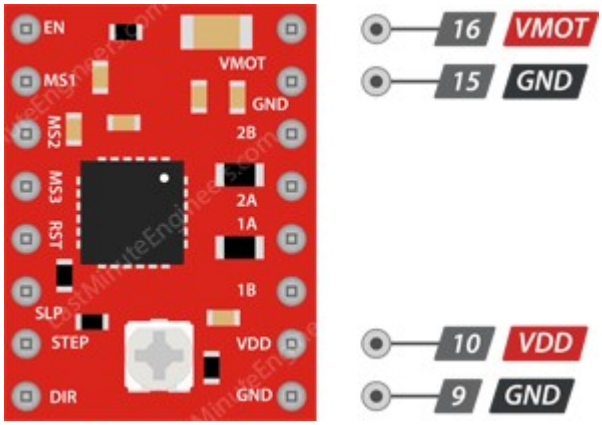


Figure 10: A4988 power pins

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.

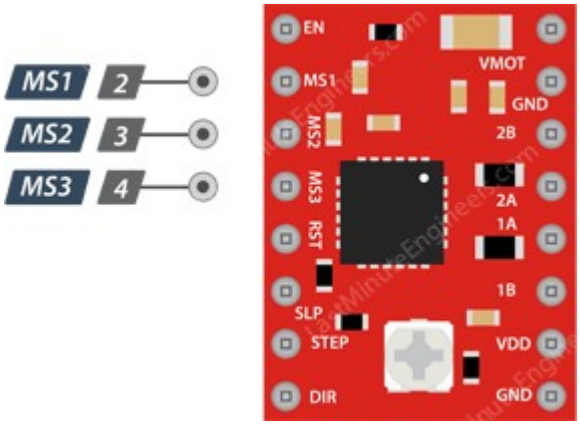


Figure 11: A4988 micro-step pin selection

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.

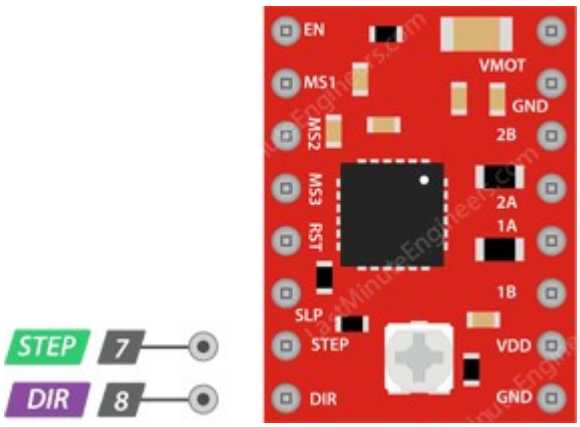


Figure 12: A4988 control pins

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.

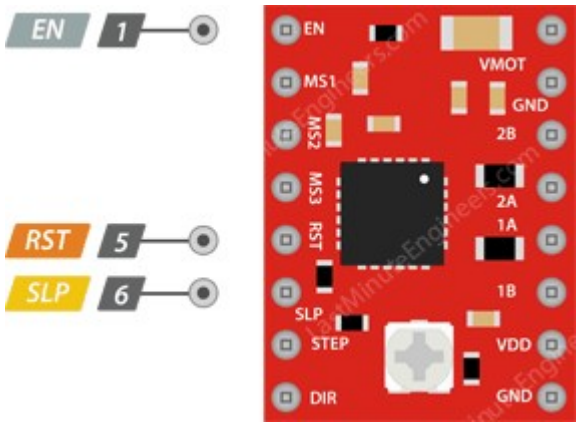


Figure 13: A4988 power state control pins

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.

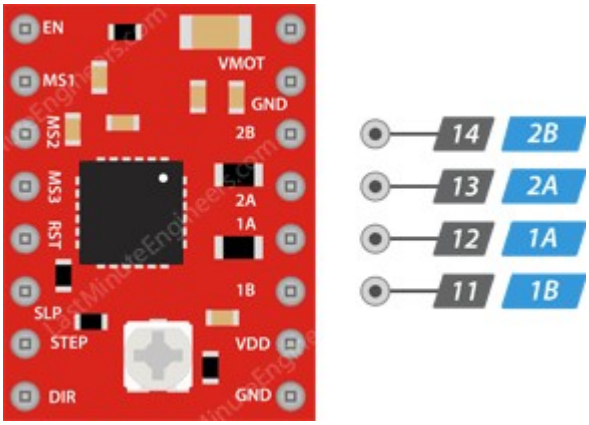


Figure 14: A4988 output pins

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 Interfacing TCS230/TCS3200 Color Sensor with Node MCU / ESP32

Color sensors provide more reliable solutions to complex automation challenges. They are used in various industries including the food and beverage, automotive and manufacturing industries for purposes such as detecting material, detecting color marks on parts, verifying steps in the manufacturing process and so on.

While expensive color sensors are used in industrial applications, inexpensive sensors such as TCS230 color sensor can be used for less stringent applications.

The TCS230 color sensor (also branded as the TCS3200) is quite popular, inexpensive and easy to use. Before we use this color sensor in our project, it would be good to see how a color sensor actually works.

3.6.1 How Color Sensors Work

White light is made up of three primary colors (Red, green and blue), which have different wavelengths. These colors combine with each other to form different shades of colors.

When white light falls on any surface, some wavelengths of light are absorbed and some are reflected, depending on the properties of the surface material. The color we see is a result of which wavelengths are reflected back into our eyes.

Figure 15: Color sensing principle

Now coming back to the sensor, a typical color sensor includes a high-intensity white LED that projects a modulated light onto the object. To detect the color of reflected light, almost all the color sensors consists of a grid of color-sensitive filter, also known as 'Bayer Filter' and an array of photodiodes underneath, as shown in the picture below.

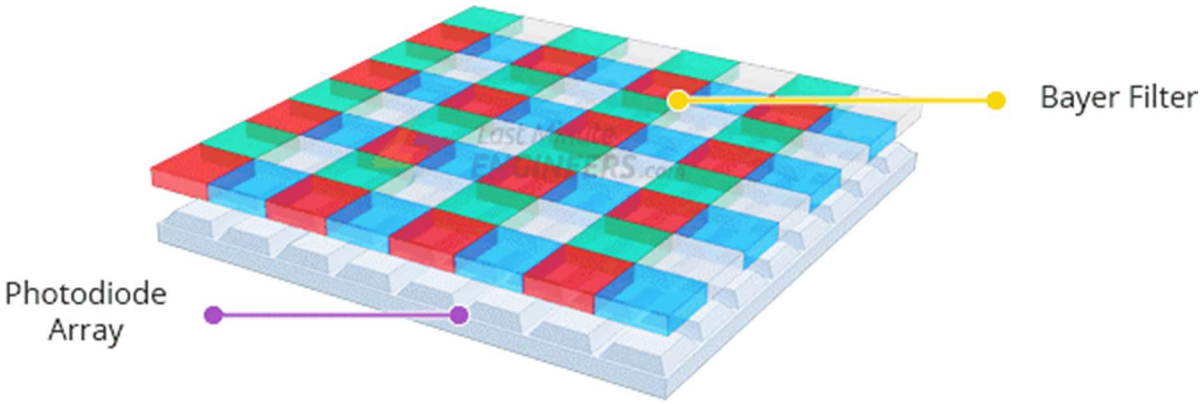


Figure 16: Color sensor array

A single pixel is made up of 4 filters, one red, one blue, one green and one clear filter (no filter). This pattern is also known as the 'Bayer Pattern'. Each filter passes light of just a single color to the photodiode beneath, while the clear filter passes light as it is, as shown below. This extra light passed through the clear filter is a major advantage in low light conditions.

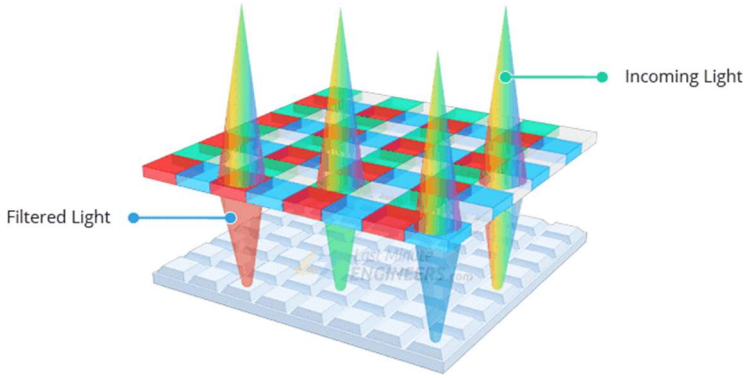


Figure 17: Color sensor filtration

The processing chip then addresses each photodiode (one color at a time), and measures the intensity of the light. As there is an array of photodiodes, the results are first averaged and then sent out for processing. By measuring the relative level of red, green and blue light, the color of the object is determined.

3.6.2 TSC230 Color Sensor Module

At the heart of the module is an inexpensive RGB sensor chip from Texas Advanced Optoelectronic Solutions – TCS230. The TCS230 Color Sensor is a complete color detector that can detect and measure an almost infinite range of visible colors.

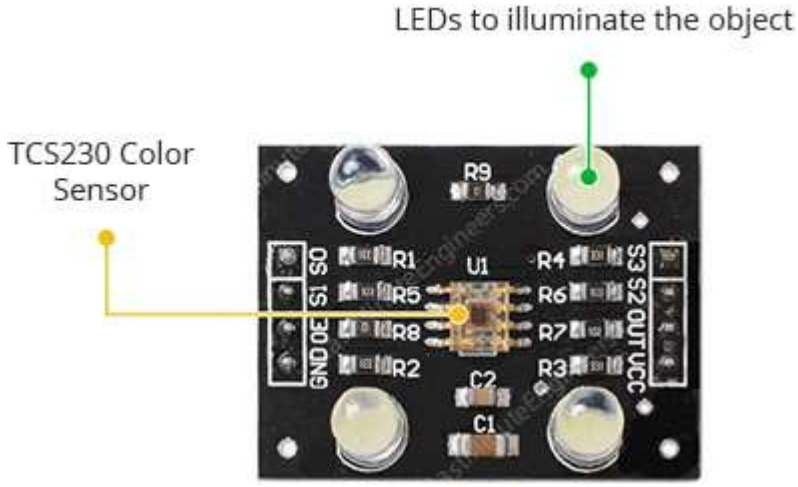


Figure 18: Color sensor module TSC230

The sensor itself can be seen at the center of the module, surrounded by the four white LEDs. The LEDs light up when the module is powered up and are used to illuminate the object being sensed. Thanks to these LEDs, the sensor can also work in complete darkness to determine the color or brightness of the object.

The TCS230 operates on a supply voltage of 2.7 to 5.5 volts and provides TTL logic-level outputs.

3.6.3 TCS230 Operation

The TCS230 detects color with the help of an 8 x 8 array of photodiodes, of which sixteen photodiodes have red filters, 16 photodiodes have green filters, 16 photodiodes have blue filters, and remaining 16 photodiodes are clear with no filters.

If you look closely at the sensor, you can actually see these filters.

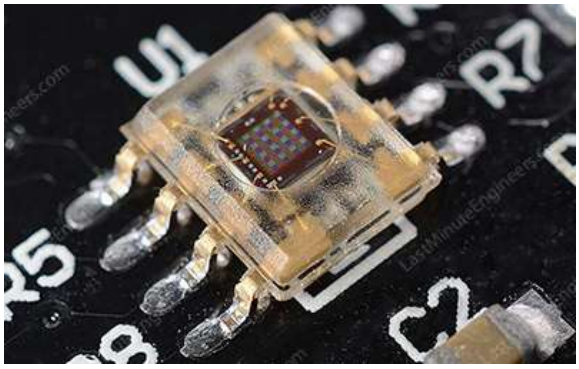


Figure 19: sensor array inside the module

Each 16 photodiodes are connected in parallel, so using two control pins S2 and S3 you can choose which of them to read. So for example, if you want to detect only red color, you can select 16 red-filtered photodiodes by setting the two pins to LOW according to the table.

Similarly, you can choose different types of photodiodes by different combinations of S2 and S3.

Table 3: Photodiode filter selection

S2	S3	Photodiode type
LOW	LOW	Red
LOW	HIGH	Blue
HIGH	LOW	Clear (No filter)
HIGH	HIGH	Green

An internal current-to-frequency converter converts readings from photodiodes into a square wave whose frequency is proportional to the intensity of the chosen color. The range of the typical output frequency is 2HZ~500KHZ.

The sensor has two more control pins, S0 and S1, which are used for scaling the output frequency. The frequency can be scaled to three different preset values of 2%, 20% or 100%. This frequency-scaling function allows the sensor to be used with a variety of microcontrollers and other devices.

Table 4: output frequency scaling

S0	S1	Output frequency scaling
LOW	LOW	Power down
LOW	HIGH	2%
HIGH	LOW	20%
HIGH	HIGH	100%

One can get different scaling factor by different combinations of S0 and S1. For the Node MCU / ESP32 most applications use the 20% scaling.

3.6.4 TSC230 Color Sensor Module Pinout

The following diagram shows the pinout of a common TCS230 module.

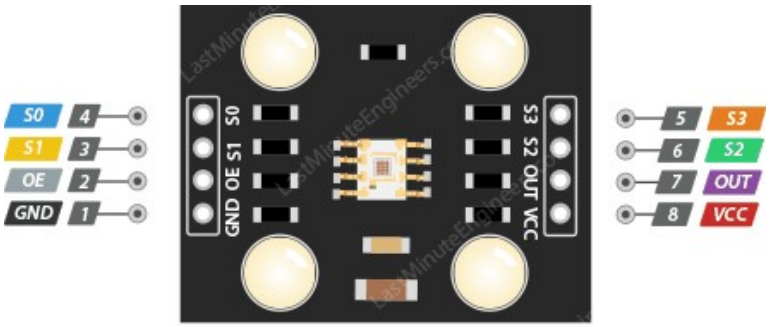


Figure 20: TSC230 module pinout

GND is a ground pin.
OE is the Output Enable pin. This pin is rarely used and on most modules is permanently enabled. If not already enabled then pull it LOW.
S0 & S1 pins are used to select the frequency scaling.
S2 & S3 pins are used to select the color array.

OUT pin is a TTL level square wave.
VCC pin supplies power to the module. Connect it to the 2.7V to 5.5V power supply.

3.6.5 Wiring TSC230 Color Sensor to Node MCU / ESP32

Hooking up the TSC 230 to a Node MCU / ESP32 is very simple. Every pin is used except the Output Enable pin, and the module is powered safely from the 5-volt output of the Node MCU / ESP32.

Below is the hookup for the experiments with the TSC230:

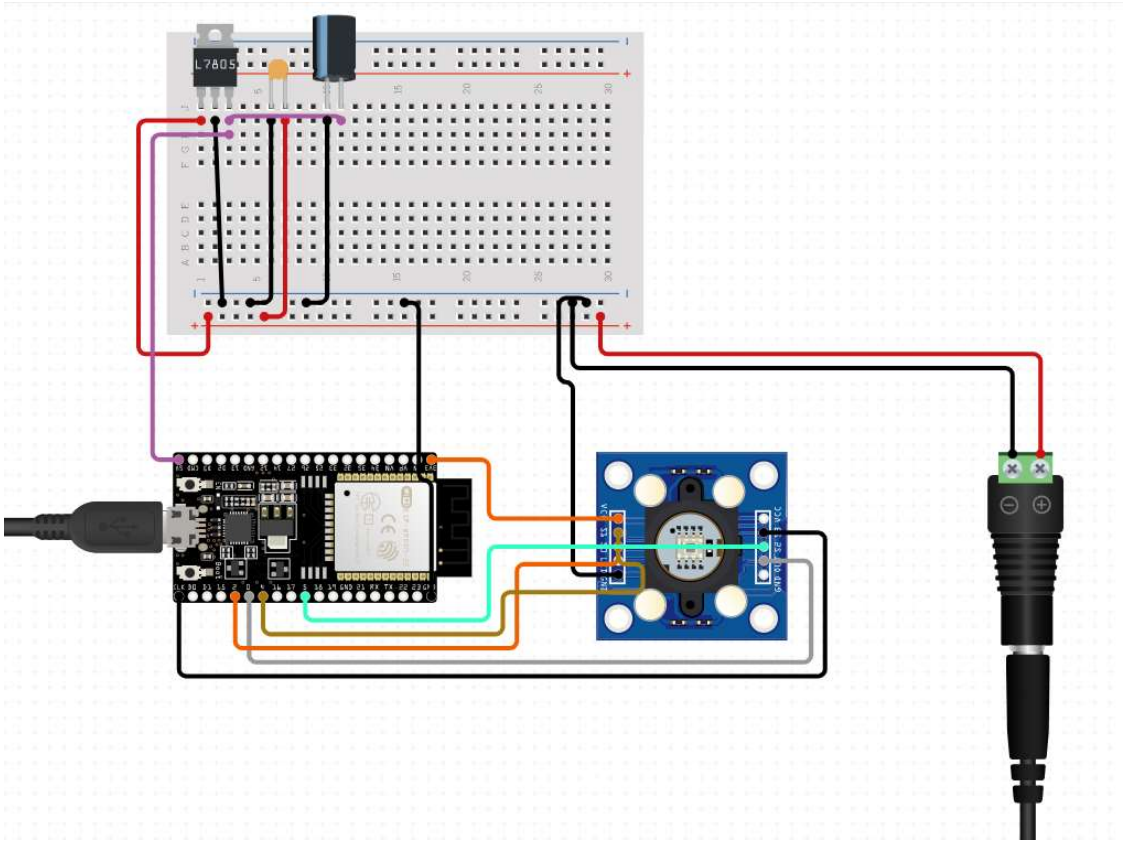


Figure 21: Interfacing TSC230 with ESP32

None of the pins used on the Node MCU / ESP32 are critical because the module does not require any pin-specific features, so if anybody want to use different pins they can do so safely. Just be sure to change the pin numbers in the code to reflect any changes to the wiring.

Once the sensor is connected to the Node MCU / ESP32 it's time to write some code!

3.6.6 Calibrating the Sensor

Two sketches are used to work with the TCS230 color sensor.

1. The first sketch (calibration sketch) will help us to obtain the raw data from the sensor.
2. The second sketch (main Node MCU / ESP32 sketch) will use the raw data previously received to display RGB values for the color being sensed.

both sketches will use the same hardware hookup.

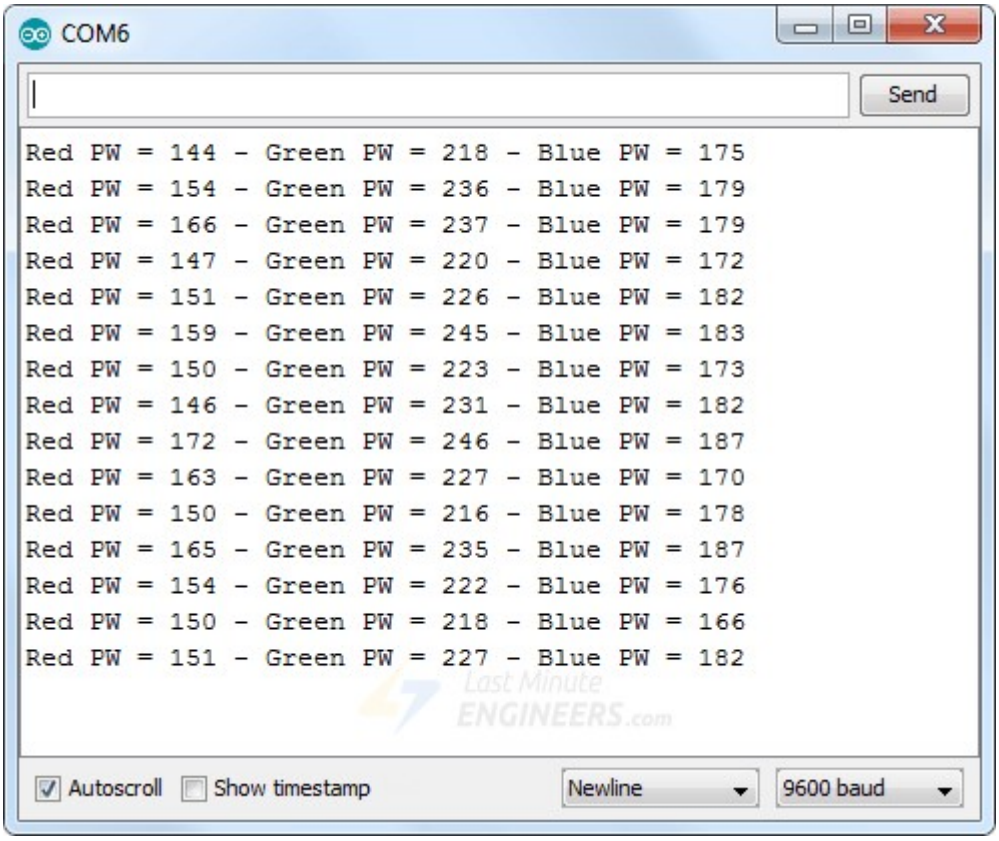
Following is the calibration sketch. This sketch addresses the TCS230 sensor color-by-color and reads the pulse width of the output pin. The output is then displayed on the serial monitor.

Load the sketch to the Node MCU / ESP32 and mount the sensor so that it is facing the objects. Start by finding a reference object for white and black color. These reference objects will produce readings at both maximum and minimum values for all three colors.

```
// Define color sensor pins
#define S0 4
#define S1 5
#define S2 6
#define S3 7
#define sensorOut 8

// Variables for Color Pulse Width Measurements
int redPW = 0;
int greenPW = 0;
int bluePW = 0;

void setup() {
  // Set S0 - S3 as outputs
  pinMode(S0, OUTPUT);
  pinMode(S1, OUTPUT);
  pinMode(S2, OUTPUT);
  pinMode(S3, OUTPUT);

  // Set Pulse Width scaling to 20%
  digitalWrite(S0,HIGH);
  digitalWrite(S1,LOW);

  // Set Sensor output as input
  pinMode(sensorOut, INPUT);

  // Setup Serial Monitor
  Serial.begin(9600);
}

void loop() {
  // Read Red Pulse Width
  redPW = getRedPW();
  // Delay to stabilize sensor
  delay(200);

  // Read Green Pulse Width
  greenPW = getGreenPW();
  // Delay to stabilize sensor
  delay(200);

  // Read Blue Pulse Width
  bluePW = getBluePW();
  // Delay to stabilize sensor
  delay(200);

  // Print output to Serial Monitor
  Serial.print("Red PW = ");
  Serial.print(redPW);
  Serial.print(" - Green PW = ");
  Serial.print(greenPW);
  Serial.print(" - Blue PW = ");
  Serial.println(bluePW);
}
```

```
// Function to read Red Pulse Widths
int getRedPW() {
  // Set sensor to read Red only
  digitalWrite(S2,LOW);
  digitalWrite(S3,LOW);
  // Define integer to represent Pulse Width
  int PW;
  // Read the output Pulse Width
  PW = pulseIn(sensorOut, LOW);
  // Return the value
  return PW;
}

// Function to read Green Pulse Widths
int getGreenPW() {
  // Set sensor to read Green only
  digitalWrite(S2,HIGH);
  digitalWrite(S3,HIGH);
  // Define integer to represent Pulse Width
  int PW;
  // Read the output Pulse Width
  PW = pulseIn(sensorOut, LOW);
  // Return the value
  return PW;
}

// Function to read Blue Pulse Widths
int getBluePW() {
  // Set sensor to read Blue only
  digitalWrite(S2,LOW);
  digitalWrite(S3,HIGH);
  // Define integer to represent Pulse Width
  int PW;
  // Read the output Pulse Width
  PW = pulseIn(sensorOut, LOW);
  // Return the value
  return PW;
}
```

Once upload the sketch will get such readings. Record the readings you get at both extremes.



3.6.7 Code Explanation:

The sketch begins with defining the pins used to connect the TSC230. Some variables are also defined to represent the pulse widths of the red, green and blue color array.

```
#define S0 4
#define S1 5
#define S2 6
#define S3 7
#define sensorOut 8

int redPW = 0;
int greenPW = 0;
int bluePW = 0;
```

In the setup, we define the S0-S3 pins as outputs. These pins will be used to select the frequency scaling and the color we wish to address. The S0 and S1 pins are used to set the frequency scaling to 20%, which is a common value when using this color sensor with an Arduino. Next, The sensors Output pin is defined as an input to the Arduino, this is where we will receive the square wave. Finally, we set up the serial monitor.

```
void setup() {
  // Set S0 - S3 as outputs
  pinMode(S0, OUTPUT);
  pinMode(S1, OUTPUT);
  pinMode(S2, OUTPUT);
  pinMode(S3, OUTPUT);
```

```
  // Set Pulse Width scaling to 20%
  digitalWrite(S0,HIGH);
  digitalWrite(S1,LOW);

  // Set Sensor output as input
  pinMode(sensorOut, INPUT);

  // Setup Serial Monitor
  Serial.begin(9600);
}
```

In the loop section, we call three functions getRedPW(), getGreenPW() and getBluePW() to obtain the pulse width. Let's examine getRedPW() as an example.

The getRedPW() function gets the red pulse width. It starts by setting the S2 and S3 pins to select the red filter. This is the only step where this function differs from its green and blue counterparts.

Next, an integer is defined to store the pulse width. The pulse width is then determined using the Arduino pulseIn() function. This function measures the pulse width, note that we have configured it to measure the width of the LOW part of the pulse. The result is time in milliseconds. This value is then returned and the function terminates.

```
int getRedPW() {
  // Set sensor to read Red only
  digitalWrite(S2,LOW);
  digitalWrite(S3,LOW);
  // Define integer to represent Pulse Width
  int PW;
  // Read the output Pulse Width
  PW = pulseIn(sensorOut, LOW);
  // Return the value
  return PW;
}
```

Back in the loop, we call three functions to read the color pulse widths, adding a delay of 200ms between them to allow the sensor to stabilize. We then print the values on the serial monitor and repeat the loop.

```
void loop() {
  // Read Red Pulse Width
  redPW = getRedPW();
  // Delay to stabilize sensor
  delay(200);

  // Read Green Pulse Width
  greenPW = getGreenPW();
  // Delay to stabilize sensor
  delay(200);
```

```
  // Read Blue Pulse Width
  bluePW = getBluePW();
  // Delay to stabilize sensor
  delay(200);

  // Print output to Serial Monitor
  Serial.print("Red PW = ");
  Serial.print(redPW);
  Serial.print(" - Green PW = ");
  Serial.print(greenPW);
  Serial.print(" - Blue PW = ");
  Serial.println(bluePW);
}
```

Arduino Code – Reading RGB Values from the TCS230

Once you have taken your readings you can upload the next sketch where we will read RGB values from the TCS230 color sensor.

Before uploading the sketch, enter the six calibration values you obtained from the calibration sketch in the top of the sketch. replace the “0” with your actual values.

```
// Define color sensor pins
#define S0 4
#define S1 5
#define S2 6
#define S3 7
#define sensorOut 8

// Calibration Values
// *Get these from Calibration Sketch
int redMin = 0; // Red minimum value
int redMax = 0; // Red maximum value
int greenMin = 0; // Green minimum value
int greenMax = 0; // Green maximum value
int blueMin = 0; // Blue minimum value
int blueMax = 0; // Blue maximum value

// Variables for Color Pulse Width Measurements
int redPW = 0;
int greenPW = 0;
int bluePW = 0;

// Variables for final Color values
int redValue;
int greenValue;
int blueValue;
```

```
void setup() {
  // Set S0 - S3 as outputs
  pinMode(S0, OUTPUT);
  pinMode(S1, OUTPUT);
  pinMode(S2, OUTPUT);
  pinMode(S3, OUTPUT);

  // Set Sensor output as input
  pinMode(sensorOut, INPUT);

  // Set Frequency scaling to 20%
  digitalWrite(S0,HIGH);
  digitalWrite(S1,LOW);

  // Setup Serial Monitor
  Serial.begin(9600);
}

void loop() {
  // Read Red value
  redPW = getRedPW();
  // Map to value from 0-255
  redValue = map(redPW, redMin,redMax,255,0);
  // Delay to stabilize sensor
  delay(200);

  // Read Green value
  greenPW = getGreenPW();
  // Map to value from 0-255
  greenValue = map(greenPW, greenMin,greenMax,255,0);
  // Delay to stabilize sensor
  delay(200);

  // Read Blue value
  bluePW = getBluePW();
  // Map to value from 0-255
  blueValue = map(bluePW, blueMin,blueMax,255,0);
  // Delay to stabilize sensor
  delay(200);

  // Print output to Serial Monitor
  Serial.print("Red = ");
  Serial.print(redValue);
  Serial.print(" - Green = ");
  Serial.print(greenValue);
  Serial.print(" - Blue = ");
  Serial.println(blueValue);
```

```
}


// Function to read Red Pulse Widths
int getRedPW() {
  // Set sensor to read Red only
  digitalWrite(S2,LOW);
  digitalWrite(S3,LOW);
  // Define integer to represent Pulse Width
  int PW;
  // Read the output Pulse Width
  PW = pulseIn(sensorOut, LOW);
  // Return the value
  return PW;
}

// Function to read Green Pulse Widths
int getGreenPW() {
  // Set sensor to read Green only
  digitalWrite(S2,HIGH);
  digitalWrite(S3,HIGH);
  // Define integer to represent Pulse Width
  int PW;
  // Read the output Pulse Width
  PW = pulseIn(sensorOut, LOW);
  // Return the value
  return PW;
}

// Function to read Blue Pulse Widths
int getBluePW() {
  // Set sensor to read Blue only
  digitalWrite(S2,LOW);
  digitalWrite(S3,HIGH);
  // Define integer to represent Pulse Width
  int PW;
  // Read the output Pulse Width
  PW = pulseIn(sensorOut, LOW);
  // Return the value
  return PW;
}
```

Load the sketch and observe the results with samples of different colors. You can make minor adjustments to calibration values if necessary.

Code Explanation

You will notice that the majority of this sketch is exactly the same as the previous sketch, except:

The six calibration values you obtained from the calibration sketch are entered in the top of the sketch.

```
// Calibration Values
int redMin = 0; // Red minimum value
int redMax = 0; // Red maximum value
int greenMin = 0; // Green minimum value
int greenMax = 0; // Green maximum value
int blueMin = 0; // Blue minimum value
int blueMax = 0; // Blue maximum value
```

Three new variables are defined for the RGB values we want to output.

```
int redValue;
int greenValue;
int blueValue;
```

In the loop section, we read each of the values using the same function used in the previous sketch. Then we use the Arduino map() function to convert these values into RGB values, using our calibration values as a reference.

Note that we have reversed the range (Min value is mapped to 255 amd Max value is mapped to 0) because our functions return pulse width, not frequency.

```
// Read Red value
redPW = getRedPW();
// Map to value from 0-255
redValue = map(redPW, redMin,redMax,255,0);
// Delay to stabilize sensor
delay(200);
```

Finally, the output the values on the serial monitor. These final readings will correspond to the RGB values of the item being scanned.

3.7 Interfacing Servo Motor with ESP32

Servo Motors are one of the most important actuators in the realm of robotics being applied in use cases from RC planes to automated door locks. Therefore, it is important to know how to interface a servo motor with ESP32 controller.

3.7.1 Connecting the Servo Motor to the ESP32

Servo motors have three wires: power, ground, and signal. The power is usually red, the GND is black or brown, and the signal wire is usually yellow, orange, or white.

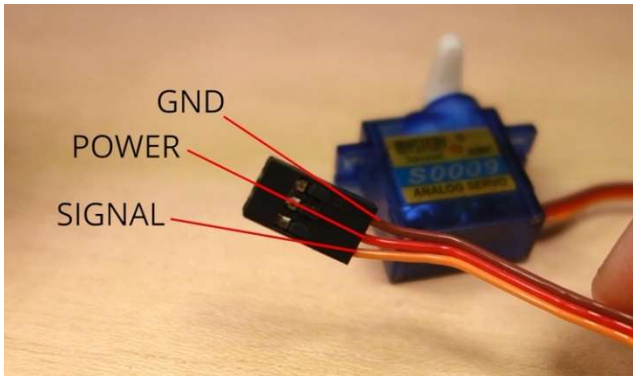


Table 5: Servo motor pin-out

Wire	Color
Power	Red
GND	Black, or brown
Signal	Yellow, orange, or white

When using a small servo like the S0009 as shown in the figure below, power it directly from the ESP32.

But using more than one servo or other type, probably need to power up servos using an external power supply.

In case of small servo like the S0009, you need to connect:

- GND -> ESP32 GND pin;
- Power -> ESP32 VIN pin;
- Signal -> GPIO 13 (or any PWM pin).

Note: in this case, you can use any ESP32 GPIO, because any GPIO is able to produce a PWM signal. However, we don't recommend using GPIOs 9, 10, and 11 that are connected to the integrated SPI flash and are not recommend for other uses.

3.7.2 Schematic

In our examples we'll connect the signal wire to GPIO 13. So, you can follow the next schematic diagram to wire your servo motor.

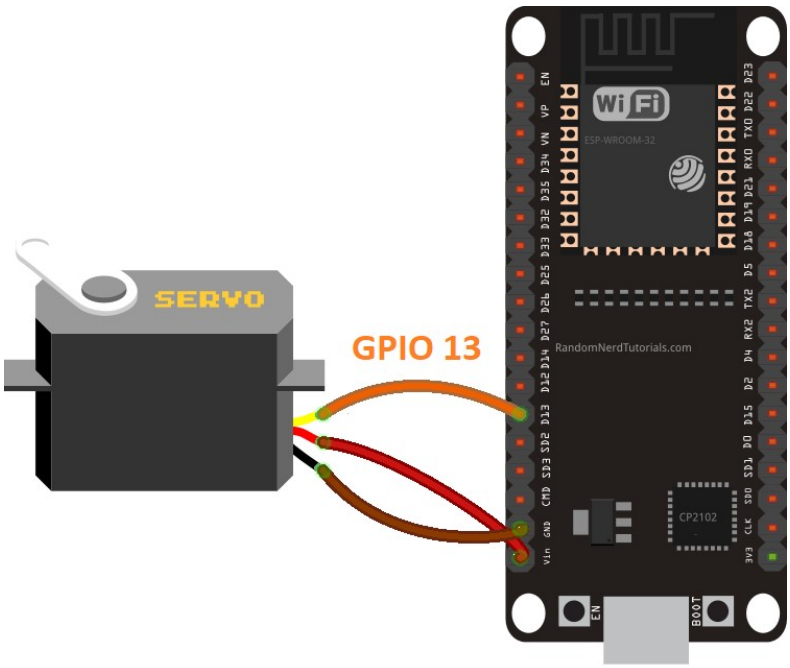
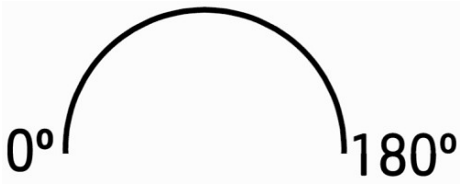


Figure 22: servo interfacing with ESP32

(This schematic uses the ESP32 DEVKIT V1 module version with 36 GPIOs – if you're using another model, please check the pinout for the board you're using.)

3.7.3 How to Control a Servo Motor?

You can position the servo's shaft in various angles from 0 to 180º. Servos are controlled using a pulse width modulation (PWM) signal. This means that the PWM signal sent to the motor will determine the shaft's position.



To control the motor, you can simply use the PWM capabilities of the ESP32 by sending a 50Hz signal with the appropriate pulse width. Or you can use a library to make this task much simpler.

3.7.4 Preparing the Arduino IDE

There's an add-on for the Arduino IDE allows you to program the ESP32 using the Arduino IDE and its programming language. Follow one of the next tutorials to prepare your Arduino IDE to work with the ESP32, if you haven't already.

3.7.5 Installing the ESP32_Arduino_Servo_Library

The ESP32 Arduino Servo Library makes it easier to control a servo motor with your ESP32, using the Arduino IDE. Follow the next steps to install the library in your Arduino IDE:

1. Download the ESP32_Arduino_Servo_Library. You should have a .zip folder in your Downloads folder

2. Unzip the .zip folder and you should get *ESP32-Arduino-Servo-Library-Master* folder
3. Rename your folder from ~~*ESP32-Arduino-Servo-Library-Master*~~ to *ESP32_Arduino_Servo_Library*
4. Move the *ESP32_Arduino_Servo_Library* folder to your Arduino IDE installation libraries folder
5. Finally, re-open your Arduino IDE

3.7.6 Testing an Example

After installing the library, go to your Arduino IDE. Make sure you have the ESP32 board selected, and then, go to **File** > **Examples** > **ServoESP32** > **Simple Servo**.

```
#include <Servo.h>
Servo myservo;  // create servo object to control a servo
// twelve servo objects can be created on most boards
int pos = 0;    // variable to store the servo position
void setup() {
  myservo.attach(13);  // attaches the servo on pin 13 to the servo object
}
void loop() {
  for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
  for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
}
```

3.7.7 Understanding the code

This sketch rotates the servo 180 degrees to one side, and 180 degrees to the other. Let's see how it works.
First, you need to include the Servo library:

```
#include <Servo.h>
```

Then, you need to create a servo object. In this case it is called myservo.

```
Servo myservo;
```

setup()
In the setup(), you initialize a serial communication for debugging purposes, and attach GPIO 13 to the servo object.

```
void setup() {
  myservo.attach(13);
}
```

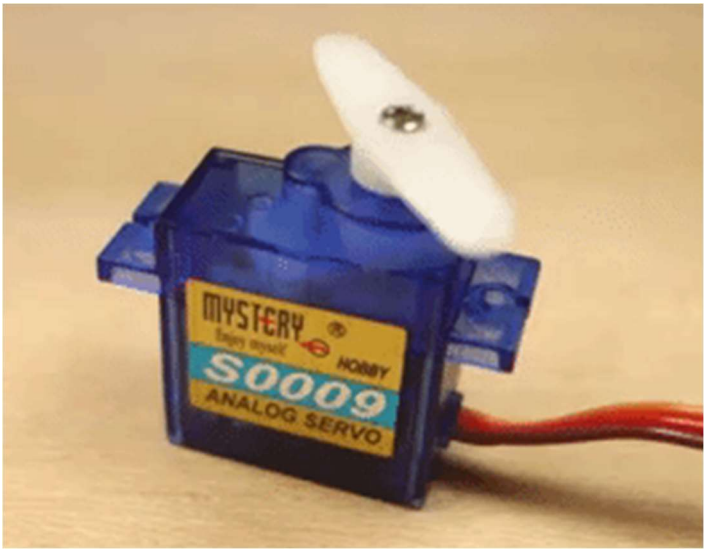
loop()

In the loop(), we change the motor's shaft position from 0 to 180 degrees, and then from 180 to 0 degrees. To set the shaft to a particular position, you just need to use the write() method in the servo object. You pass as an argument, an integer number with the position in degrees.

```
myservo.write(pos);
```

3.7.8 Testing the Sketch

Upload the code to your ESP32. After uploading the code, you should see the motor's shaft rotating to one side and then, to the other.



3.8 Overview of the Project:

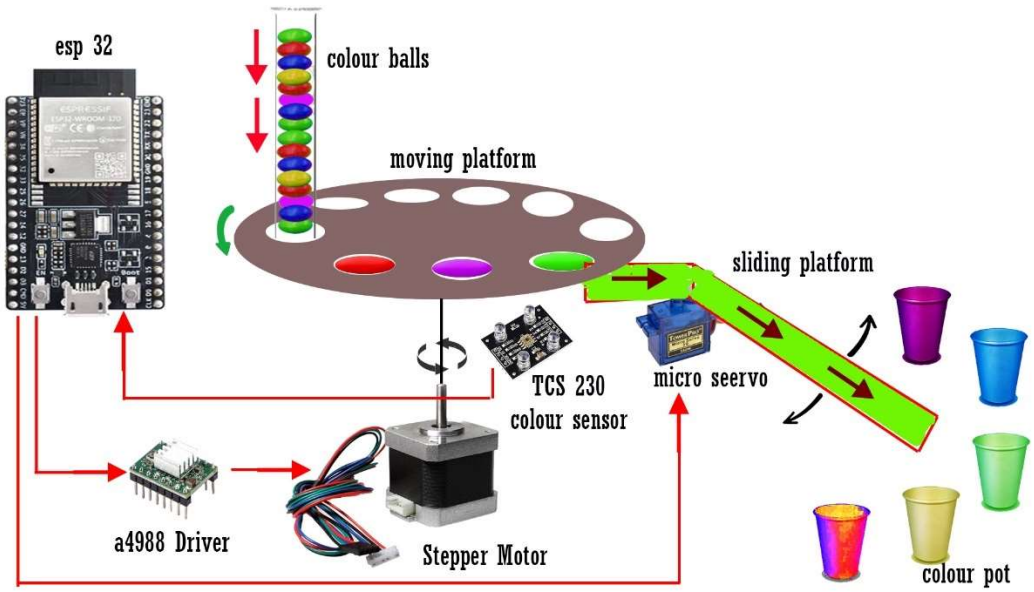


Figure 23: Overview of the project

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.

3.9 Circuit Diagram:

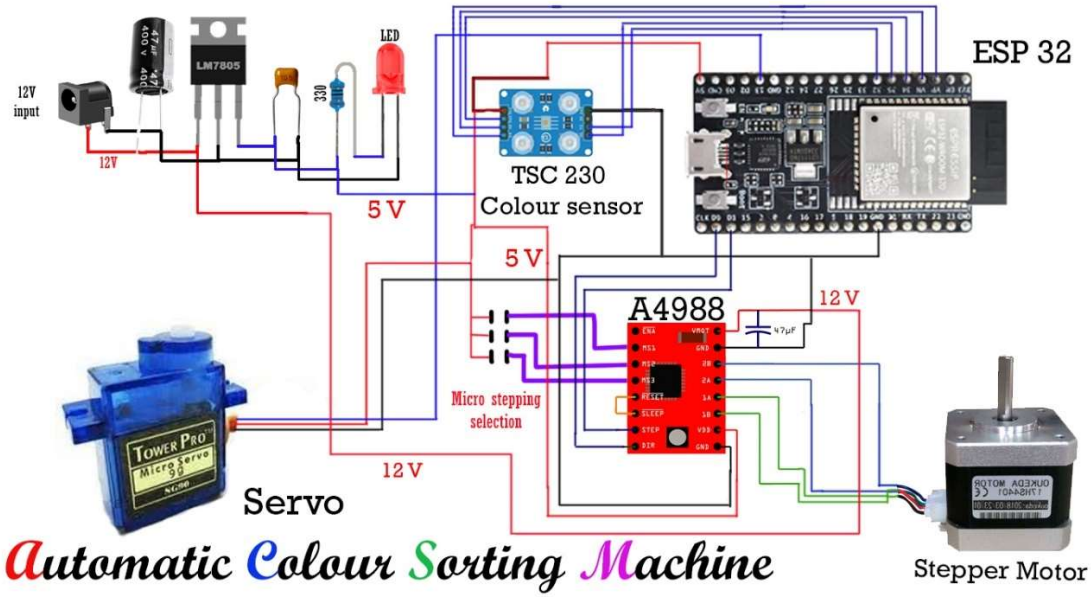


Figure 24: Circuit diagram of the developed prototype

CHAPTER 4

(Hardware Modeling)

4.1 Main features of the prototype

The features of the developed prototype are:

- Automatic, fast and accurate color sorting
- With little modification can be used with any type of color material
- Reduce human-Dependency
- On board stepper motor driver
- On board berg connector for stepper motor, servo and color sensor
- 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

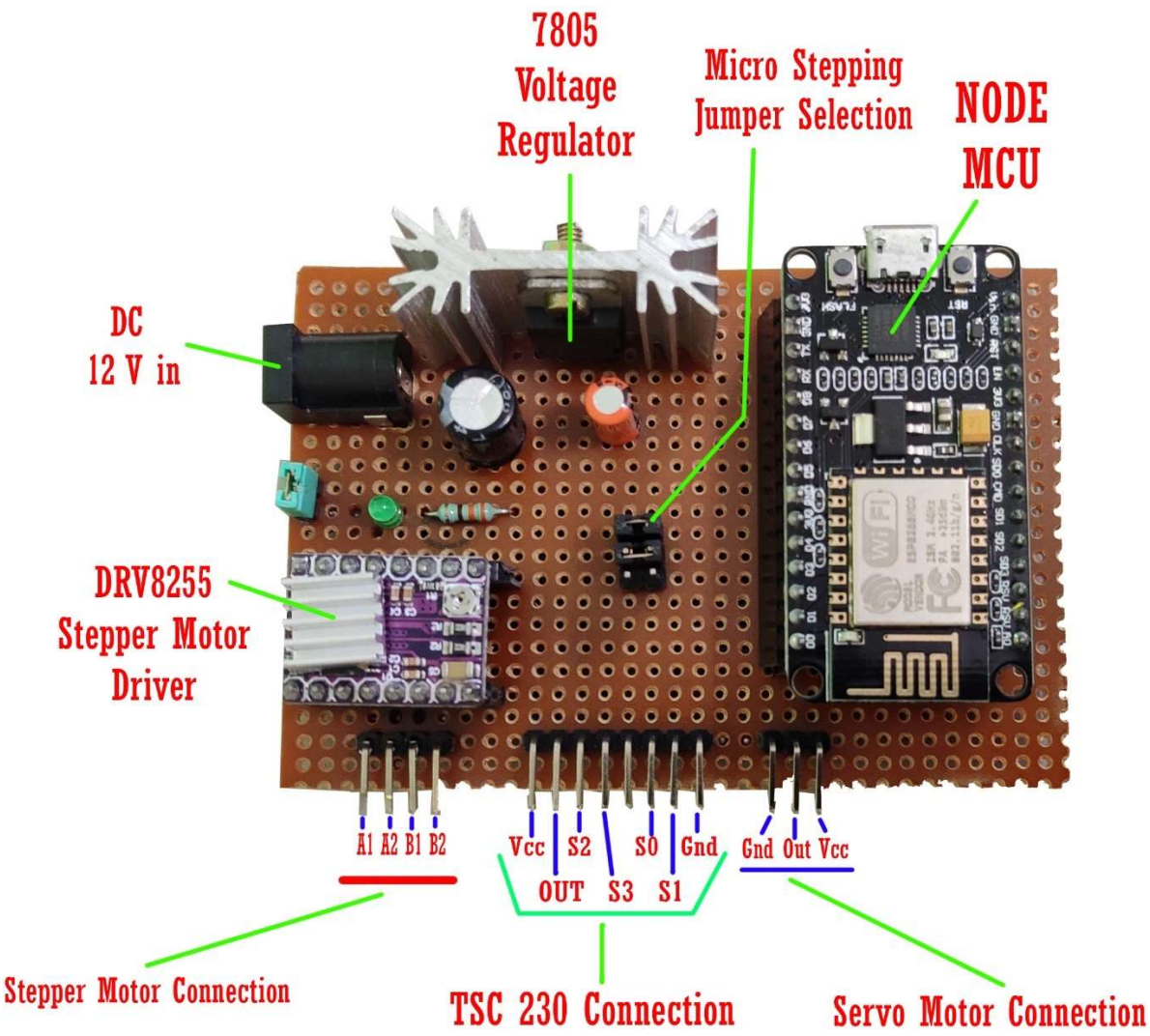


Figure 25: Main Controller board

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, TSC230 and servo motor. 12 v input directly goes to the stepper motor driver(A4988)
3. The stepper motor rotates 60^0 and place the color disc above the color sensor
4. TSC230 sense the color of the disc and send the RGB value of the color to the ESP32
5. ESP32 identify the color from the preset value and instruct the servo motor to place the sliding platform in the correct position
6. In the next rotation of the stepper motor the color disc fall down and slides to the proper color pot
7. This process repeats until the color disc holder become empty

4.4 Components required

Table 6: Component listing

Sl. No.	Component	Qtn
1.	ESP32	1
2.	NEMA 17 Stepper Motor	1
3.	A4988 Stepper motor driver	1
4.	Micro servo motor	1
5.	TSC230 color sensor	1
6.	LM7805 regulator	1
7.	General blank PCB (KS 100)	1
8.	5 mm LED	1
9.	Berg terminals	14
10.	DC power socket	1
11.	100 uF electrolytic capacitor	2
12.	10 uF electrolytic capacitor	1
13.	8mm straight rod	8inch
15.	5mm to 8mm shaft connector	1
16.	5mm round disc connector	1

4.5 Hardware interfacing

4.5.1 TSC230 interfacing with microcontroller

Figure 26: TSC230 color sensor

The TCS3200 color sensor – shown in the figure below – uses a TAOS TCS3200 RGB sensor chip to detect color. It also contains four white LEDs that light up the object in front of it.

Here's the sensor specifications:

- Power: 2.7V to 5.5V
- Size: 28.4 x 28.4mm (1.12 x 1.12″)
- Interface: digital TTL
- High-resolution conversion of light intensity to frequency
- Programmable color and full-scale output frequency
- Communicates directly to microcontroller

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

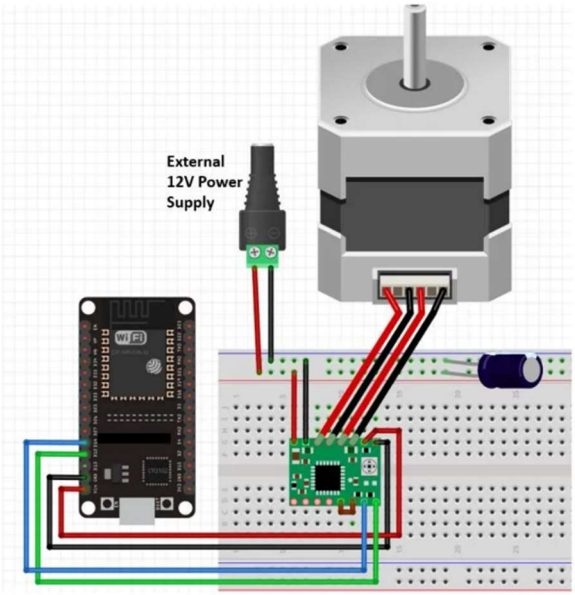
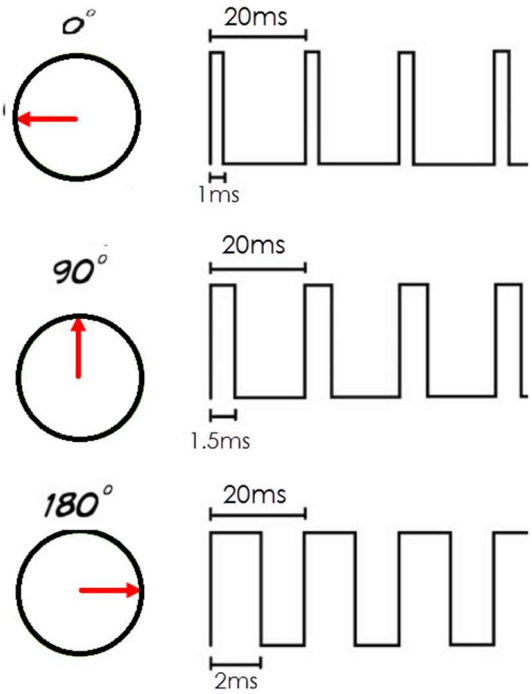


Figure 27: A4988 interfacing with ESP32

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.

4.5.3 Servo motor Interfacing with ESP32



Dc servo motors are popular among the diy circuit makers. They are also used in toys and are known as RC servo motors. RC servo motors are small, cheap and easy to use with systems involving microcontrollers or chips (Used in Toys) dedicated for a singular purpose. RC servos can rotate only 180 degree. Their purpose is to provide precise location in 0-to-180-degree angular field. Most popular RC servo motor is tower pro micro servo sg90. Sg90 servo motor works on 4.8 volts. Small torque produced by sg90 at 4.8 volts can displace a load of 1.8 kg per cm. sg90 servo motor shaft rotation depends on pwm signal frequency and duty cycle. Pwm frequency requirement for most RC servo motors is 50 Hz. They can rotate between 0 to 180 degree on pwm signal duty cycle between 1 milli seconds to 2 milli seconds. 1 milli second duty cycle at 50 hz frequency moves the servo shaft to 0-degree angle. 1.5 ms moves to 90 degree and 2 ms to 180 degree. One can calculate the duty cycles for other angles by yourself. For example, for 45-degree shaft rotation the duty cycle will be 45/180= 0.25 so 1(0 degree) +0.25= 1.25 ms.

Interfacing Diagram

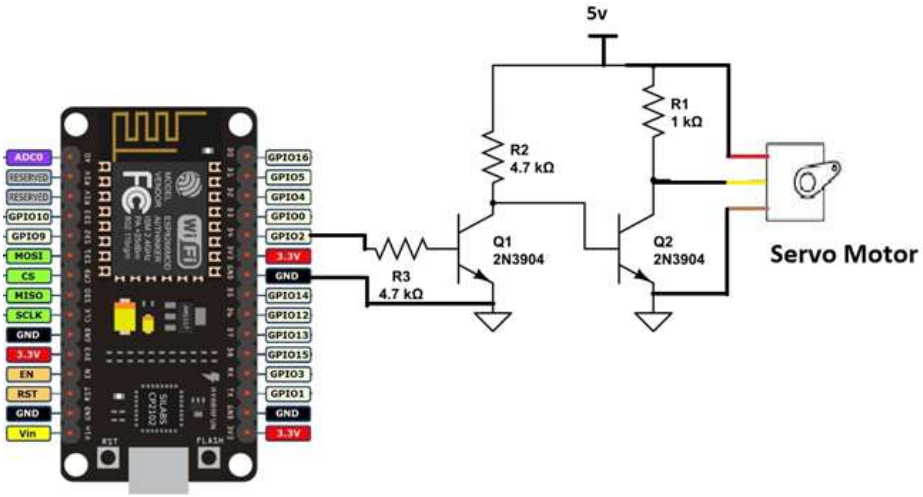


Figure 28: Interfacing Servo Motor with ESP32

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
- TSC230 color sensor section
- Servo motor section

The operation of the whole circuit is depending on every section's performance.

5.2 Flow Chart

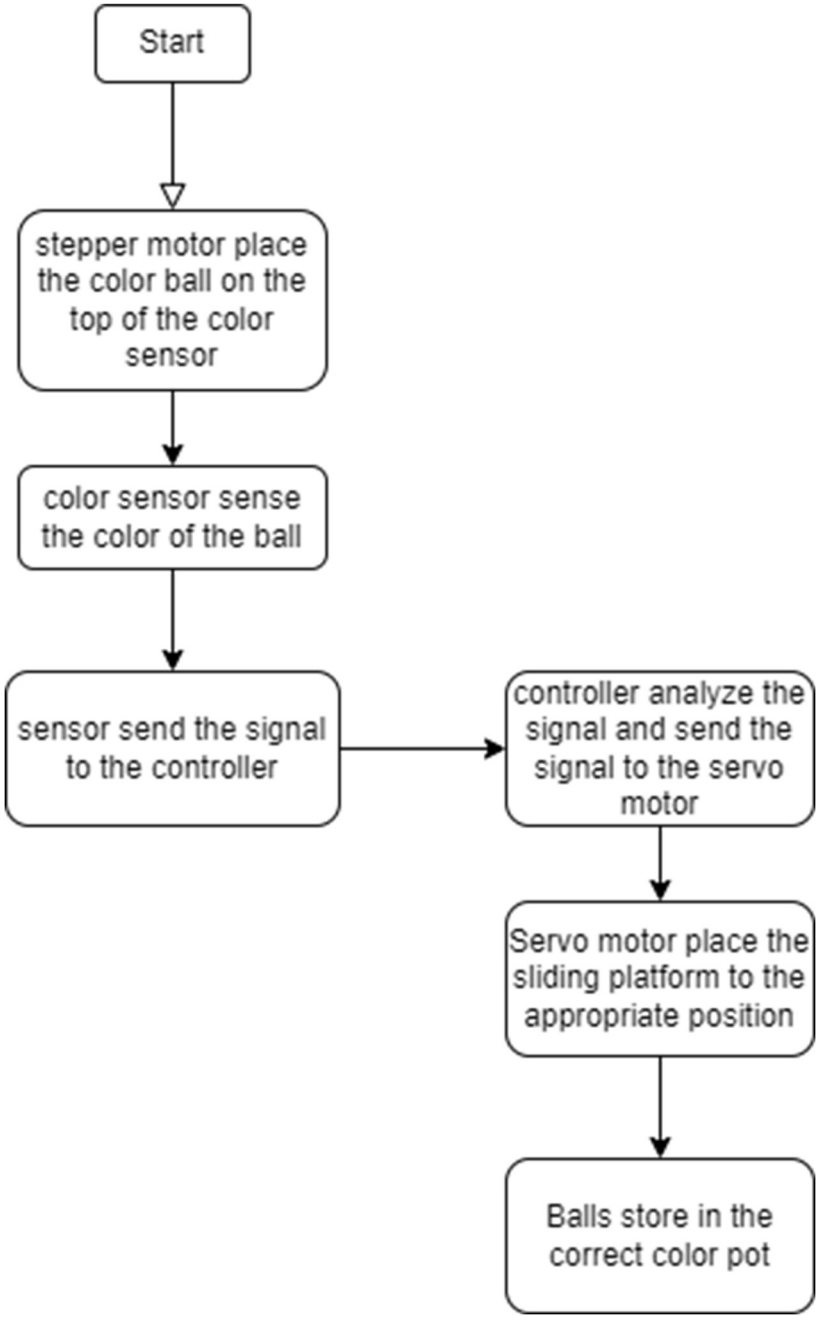


Figure 29: Flow chart of the program

5.3 Principle & Operations

At first, the colour discs are put into the colour disk holder and then align the colour positioner under the colour disk holder. The stepper motor rotates 60^0 in every step to place with colour disk above the TSC230 colour sensor. After that colour sensor check the colour and convert the colour in their RGB values it sends those values to the microcontroller. ESP32 detects the colour of the discs based on the predefine values of the colour discs. According to the pre-set value controller gave information to Servo motor to rotate and place the slider on the particular colour pot accordingly. The process continues till all the colour discs placed on the respectively colour pot.

5.4 Cost estimation of the project

In this project we have used the cheapest IOT module ESP32. So the total cost of the project is reduced compare to the other embedded system project. The total estimated cost of the complete project is listed in table 3.

Table 7: Costing of the projects

Sl. No.	**Component**	**Cost**
1.	ESP32	400
2.	NEMA 17 Stepper Motor	750
3.	A4988 Stepper motor driver	200
4.	Micro servo motor	100
5.	TSC230 color sensor	300
6.	LM7805 regulator	5
7.	General blank PCB (KS 100)	40
8.	5 mm LED	1
9.	Berg terminals	14
10.	DC power socket	5
11.	100 uF electrolytic capacitor	2
12.	10 uF electrolytic capacitor	2
13.	8mm straight rod	90
15.	5mm to 8mm shaft connector	80
16.	5mm round disc connector	120
Total Cost		**2109/-**

5.5 Photographs of the prototype

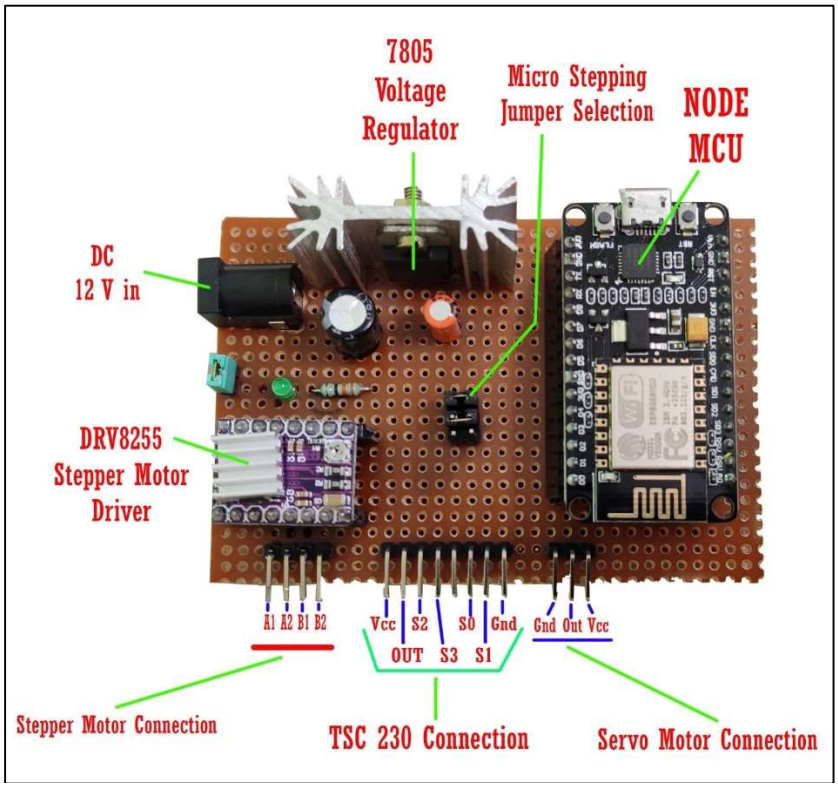
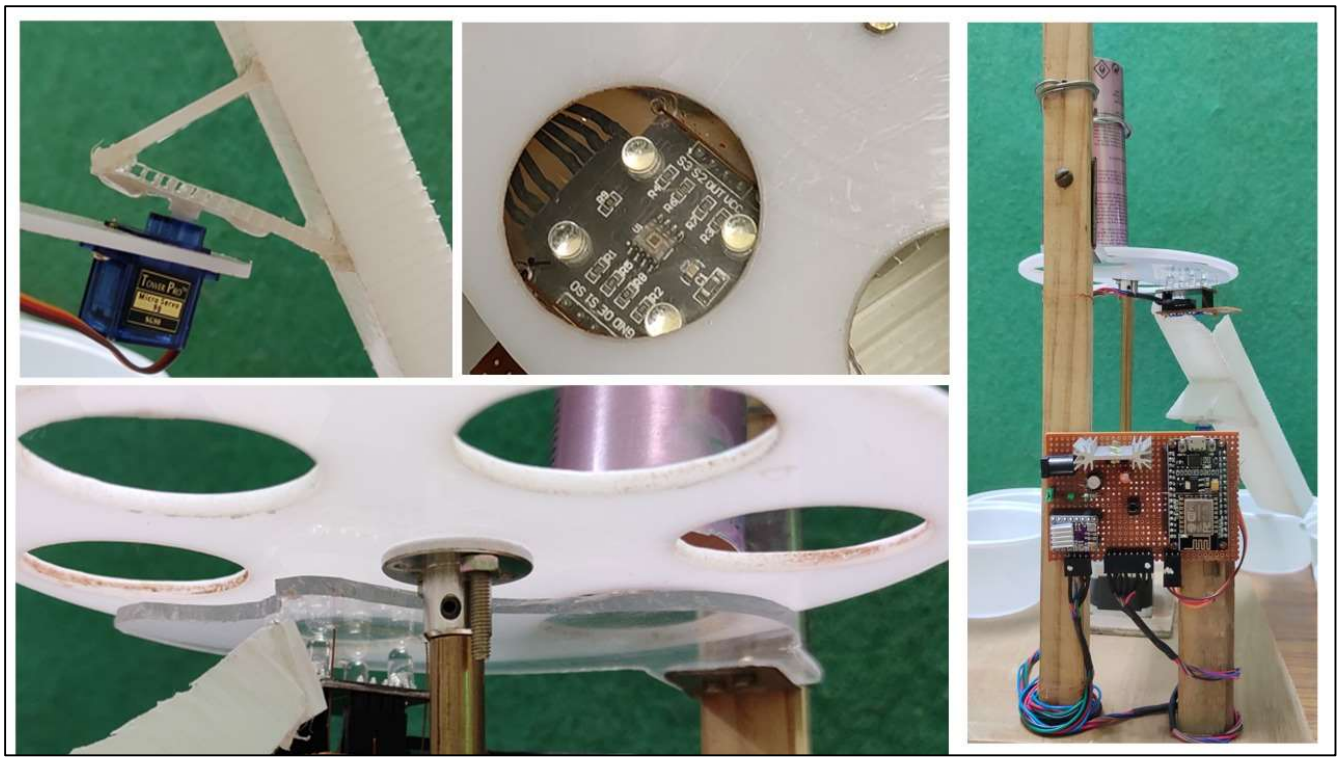


Figure 30: Main Controller Board

Figure 31: The Prototype



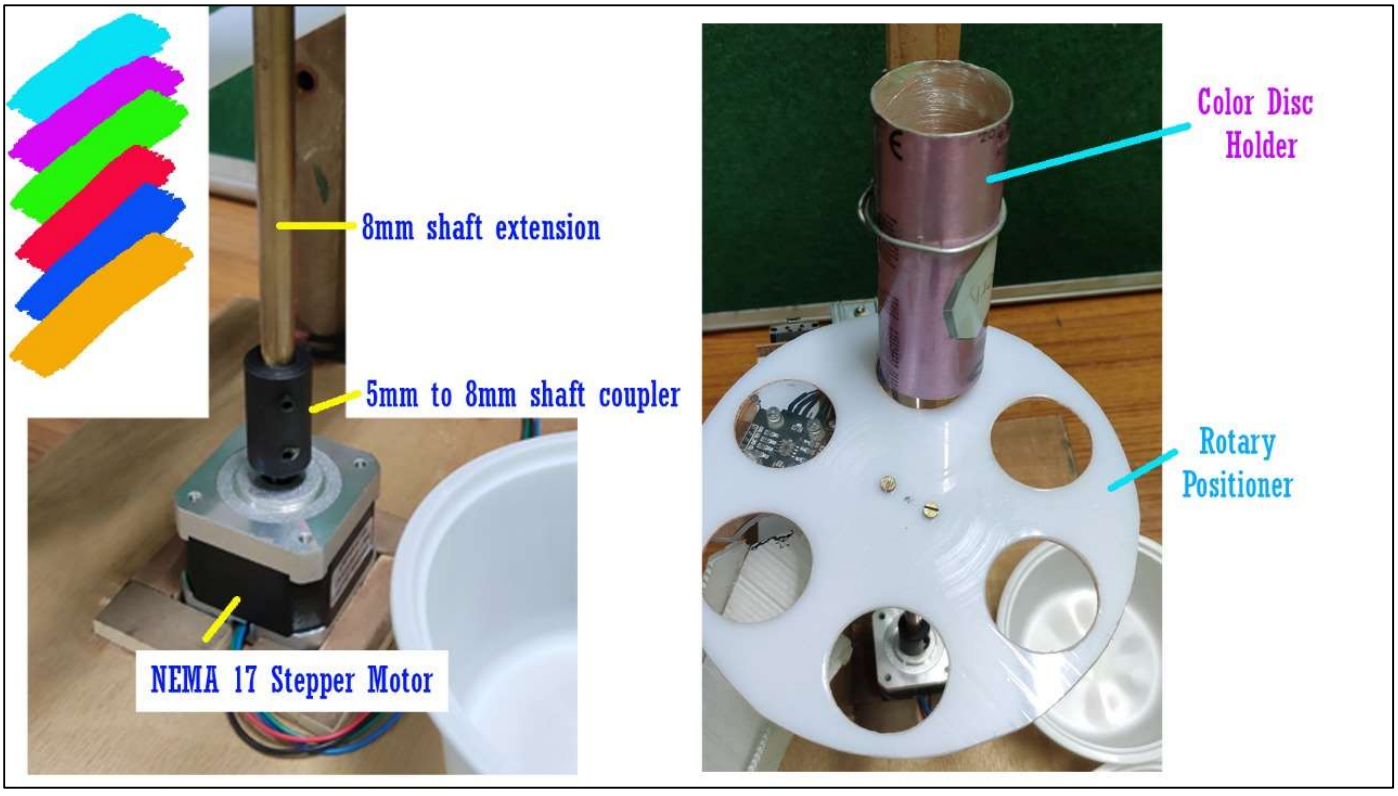


Figure 32: Different sections of the developed prototype

Chapter 6

(Conclusion & Future Scope)

6.1 Conclusion

Here we developed a prototype which automatically sort out color discs in the respective color pots with the help of a micro-controller. It will help in reducing human effort and error. Our circuit consists of ESP32, TSC230, Stepper motor with driver and servo motor. First after switching on, the stepper motor places the color disc above the TSC230 color sensor. After sensing the color of the disc, the servo motor place, the color discs to the respective color pots with the help of a sliding platform. All the motors are controlled by the microcontroller.

6.2 Result

The prototype model was made according to the circuit diagram and the results were as expected. The discs are sorted out satisfactorily using the help of color sensor and sets of motors.

6.3 Future work

Here we used a NEMA 17 stepper motor to position the colour disc on the top of the colour sensor TSC230. It works satisfactorily with the help of the stepper motor driver A4988/DRV8255. Only problem is that the stepper motor we used here is an open loop stepper motor. The motor works on the pulses generated by the microcontroller. There is no feedback system to check that all the pulses generated by the controller are used by the motor or not. If the motor skips some of the pulses, then the rotation is not as expected. This is the main disadvantages of the open loop stepper motor. We planned in the future to use a close loop stepper motor to get the feedback information that all the pulsed generated by the controller must be used by the stepper motor. Then the accuracy of the system increases drastically.

Figure 33: Close loop stepper motor

Chapter 7

(References)

Reference

[1] Ch.Shravani, G. Indira, V. Appalaraju, "Arduino Based Color Sorting Machine using TCS3200 Color Sensor", International Journal of Innovative Technology and Exploring Engineering (IJITEE), ISSN: 2278-3075, Volume-8, Issue- 6S4, April 2019.

[2] K.Sasidhar, Shahwar Farooqi, Mohammed Abdul Moin, M Sachin, "Design and Development of a Colour Sorting Machine using PLC and SCADA", International Journal of Research and Scientific Innovation (IJRSI), Volume V, Issue VII, July 2018, ISSN 2321–2705.

[3] Kunhimohammed C. K, Muhammed Saifudeen K. K, Sahna S, Gokul M. S and Shaeez Usman Abdulla, "Automatic Color Sorting Machine Using TCS230 Color Sensor And PIC Microcontroller", International Journal of Research and Innovations in Science and Technology, Volume 2, Issue 2, 2015, ISSN(Online): 2394-3858 ISSN(Print): 2394-3866.

[4] Aung Thike, Zin Zin Moe San, Dr. Zaw Min Oo, "Design and Development of an Automatic Color Sorting Machine on Belt Conveyor", International Journal of Science and Engineering Applications Volume 8–Issue 07,176-179, 2019, ISSN:-2319–7560.

[5] Aye Myat Myat Myo, Zar Chi Soe, "Automatic Color Sorting Machine Using Arduino Mega Microcontroller", International Journal of Latest Technology in Engineering, Management & Applied Science (IJLTEMAS), Volume VIII, Issue VIII, August 2019, ISSN 2278-2540.

[6] Lim Jie Shen, Irda Hassan, "Design and development of colour sorting robot", Journal of Engineering Science and Technology EURECA 2014 Special Issue January (2015) 71– 81, © School of Engineering, Taylor's University.

[7] Dharmannagari Vinay Kumar Reddy, "Sorting of objects based on colour by pick and place robotic arm and with conveyor belt arrangement", International Journal of mechanical engineering and robotic research (IJMERR), Vol. 3, No. 1, January 2014, ISSN 2278 – 0149.

[8] Mr.V.A.Aher, Mayur Dukre, Ganesh Abhang, Trupti Thorat, "Colour based object sorting machine", International Research Journal of Engineering and Technology (IRJET), Volume 08, Issue 02, Feb 2021, ISSN(print): 2395-0072, ISSN(online): 2395-0056.

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.

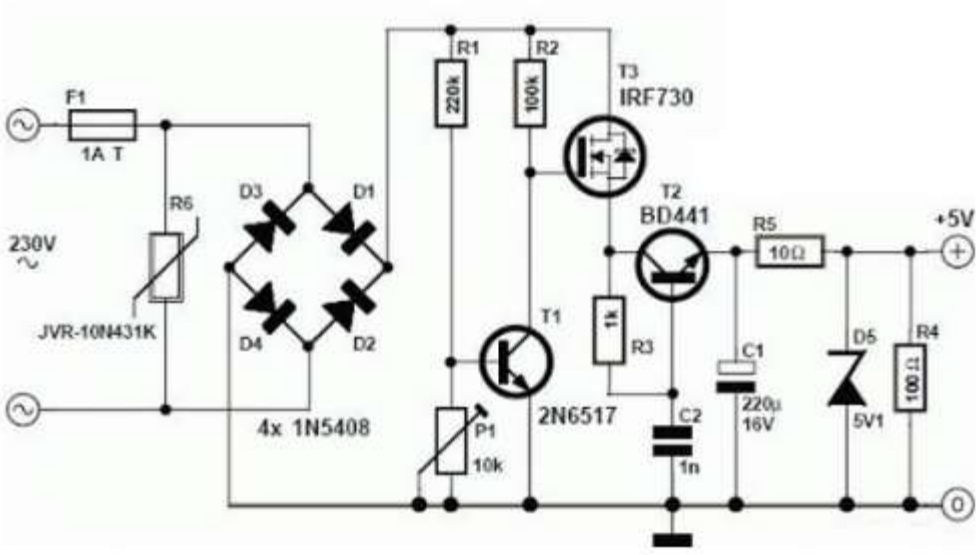


Figure 34: Transformer less SMPS 5 volt power supply

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 35: 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 band 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

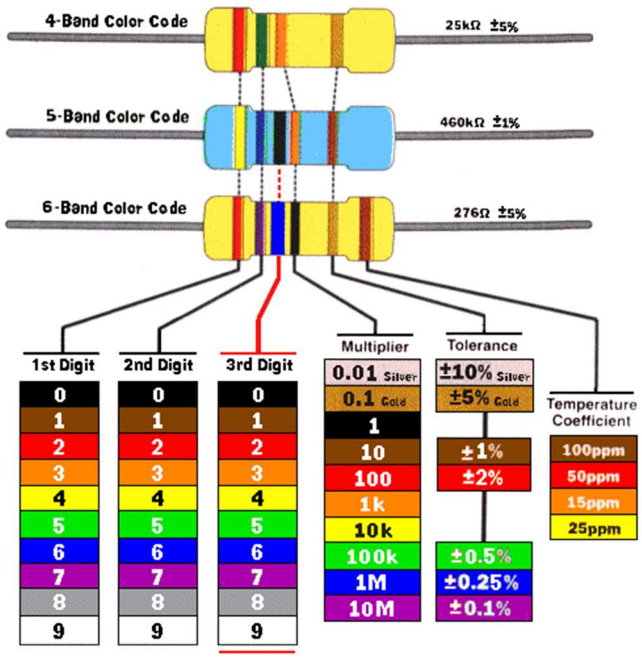


Figure 36: Color Code for resistance

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 37: ESP32 microcontroller

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.

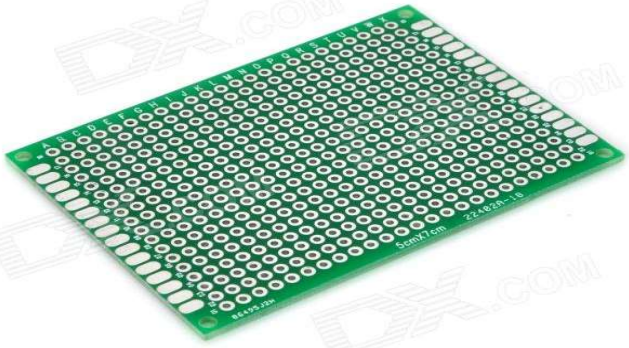


Figure 38: Blank glass epoxy PCB Board

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 39: 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 40: A4988 Stepper motor Driver

TSC230 / TSC3200 Colour Sensor

The TCS230 senses colour light with the help of an 8 x 8 array of photodiodes. Then using a Current-to-Frequency Converter the readings from the photodiodes are converted into a square wave with a frequency directly proportional to the light intensity.

Figure 41: TSC230 color sensor module

Appendix B

(Software coding)

PROGRAM CODE:

```
#include<Servo.h>
Servo servo_1;
int pos;

#define S0 15
#define S1 13
#define S2 12
#define S3 14
#define sensorOut 16
// Define pin connections & motor's steps per revolution
const int dirPin = D1;
const int stepPin = D2;


int Red = 0;
int Green = 0;
int Blue = 0;

void setup()
{
  servo_1.attach(D3, 500, 2400);
pinMode(S0, OUTPUT);
pinMode(S1, OUTPUT);
pinMode(S2, OUTPUT);
pinMode(S3, OUTPUT);
pinMode(sensorOut, INPUT);
pinMode(stepPin, OUTPUT);
pinMode(dirPin, OUTPUT);

// Setting frequency-scaling to 20%
digitalWrite(S0,HIGH);
digitalWrite(S1,HIGH);

Serial.begin(9600);
}

void loop()
{
// Set motor direction clockwise
  digitalWrite(dirPin, HIGH);

  // Spin motor slowly
  for(int x = 0; x < 267; x++)
```

```
  {
   digitalWrite(stepPin, HIGH);
   delayMicroseconds(2000);
   digitalWrite(stepPin, LOW);
   delayMicroseconds(2000);
  }
  delay(500); // Wait a second

// Setting red filtered photodiodes to be read
digitalWrite(S2,LOW);
digitalWrite(S3,LOW);
// Reading the output frequency
Red = pulseIn(sensorOut, LOW);
// Printing the value on the serial monitor
Serial.print("R= ");//printing name
Serial.print(Red);//printing RED color frequency
Serial.print(" ");
delay(50);
// Setting Green filtered photodiodes to be read
digitalWrite(S2,HIGH);
digitalWrite(S3,HIGH);
// Reading the output frequency
Green = pulseIn(sensorOut, LOW);
// Printing the value on the serial monitor
Serial.print("G= ");//printing name
Serial.print(Green);//printing RED color frequency
Serial.print(" ");
delay(50);
// Setting Blue filtered photodiodes to be read
digitalWrite(S2,LOW);
digitalWrite(S3,HIGH);
// Reading the output frequency
Blue = pulseIn(sensorOut, LOW);
// Printing the value on the serial monitor
Serial.print("B= ");//printing name
Serial.print(Blue);//printing RED color frequency
Serial.println(" ");

if(Red <= 6 && Red >=3 && Green <= 13 && Green >= 11 && Blue <= 11 && Blue >=
10){
    Serial.println(" - ORANGE Detected!");
    servo_1.write (0);
  }
if(Red <= 9 && Red >=7 && Green <= 13 && Green >= 11 && Blue <= 11 && Blue >= 9){
    Serial.println(" - BROWN Detected!");
    servo_1.write (30);
  }
```

```
if(Red <= 6 && Red >=3 && Green <= 6 && Green >= 4 && Blue <= 8 && Blue >= 4){
    Serial.println(" - YELLOW Detected!");
    servo_1.write (60);
 }
if(Red <= 16 && Red >=14 && Green <= 18 && Green >= 15 && Blue <= 14 && Blue >=
12){
    Serial.println(" - BLUE Detected!");
    servo_1.write (90);
 }
if(Red <= 16 && Red >=14 && Green <= 14 && Green >= 10 && Blue <= 14 && Blue >=
12){
    Serial.println(" - Green Detected!");
    servo_1.write (120);
 }
if(Red <= 8 && Red >=6 && Green <= 17 && Green >= 15 && Blue <= 14 && Blue >=
12){
    Serial.println(" - RED Detected!");
    servo_1.write (150);
 }
delay(500);
}
```

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

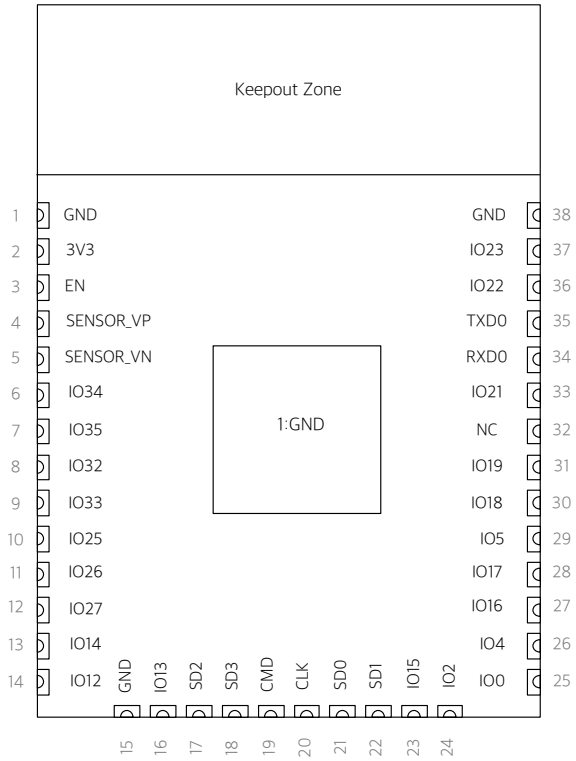


Figure 1: ESP32-WROOM-32 (ESP-WROOM-32) Pin layout

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:

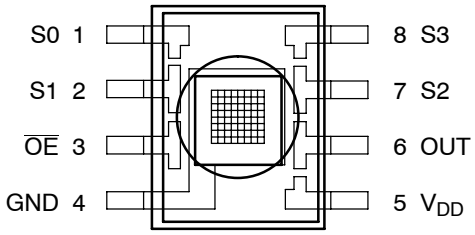
- When Wi-Fi is enabled, the chip switches between Active and Modem-sleep mode. Therefore, power consumption changes accordingly.
- In Modem-sleep mode, the CPU frequency changes automatically. The frequency depends on the CPU load and the peripherals used.
- During Deep-sleep, when the ULP co-processor 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.

TCS230
PROGRAMMABLE
COLOR LIGHT-TO-FREQUENCY CONVERTER

TAOS046M – OCTOBER 2007

- **High-Resolution Conversion of Light Intensity to Frequency**
- **Programmable Color and Full-Scale Output Frequency**
- **Communicates Directly With a Microcontroller**
- **Single-Supply Operation (2.7 V to 5.5 V)**
- **Power Down Feature**
- **Nonlinearity Error Typically 0.2% at 50 kHz**
- **Stable 200 ppm/°C Temperature Coefficient**
- **Low-Profile Lead (Pb) Free and RoHS Compliant Surface-Mount Package**

PACKAGE D
8-LEAD SOIC
(TOP VIEW)

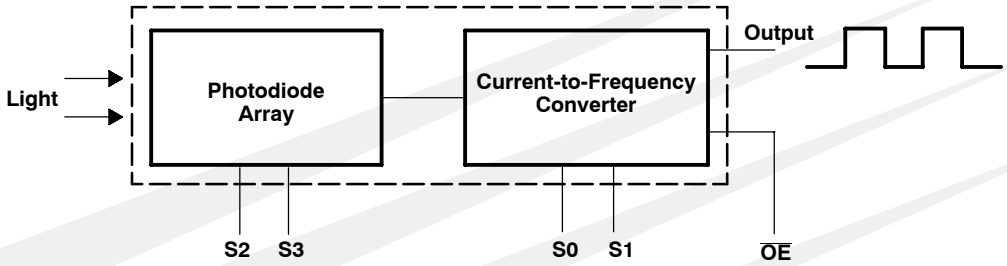


Description

The TCS230 programmable color light-to-frequency converter combines configurable silicon photodiodes and a current-to-frequency converter on a single monolithic CMOS integrated circuit. The output is a square wave (50% duty cycle) with frequency directly proportional to light intensity (irradiance). The full-scale output frequency can be scaled by one of three preset values via two control input pins. Digital inputs and digital output allow direct interface to a microcontroller or other logic circuitry. Output enable ($\overline{OE}$) places the output in the high-impedance state for multiple-unit sharing of a microcontroller input line.

The light-to-frequency converter reads an 8 x 8 array of photodiodes. Sixteen photodiodes have blue filters, 16 photodiodes have green filters, 16 photodiodes have red filters, and 16 photodiodes are clear with no filters. The four types (colors) of photodiodes are interdigitated to minimize the effect of non-uniformity of incident irradiance. All 16 photodiodes of the same color are connected in parallel and which type of photodiode the device uses during operation is pin-selectable. Photodiodes are 120 μm x 120 μm in size and are on 144-μm centers.

Functional Block Diagram



Texas Advanced Optoelectronic Solutions Inc.
1001 Klein Road • Suite 300 • Plano, TX 75074 • (972) 673-0759
www.taosinc.com

Terminal Functions

TERMINAL NAME	TERMINAL NO.	I/O	DESCRIPTION
GND	4		Power supply ground. All voltages are referenced to GND.
$\overline{OE}$	3	I	Enable for f_o (active low).
OUT	6	O	Output frequency (f_o).
S0, S1	1, 2	I	Output frequency scaling selection inputs.
S2, S3	7, 8	I	Photodiode type selection inputs.
V_{DD}	5		Supply voltage

Table 1. Selectable Options

S0	S1	OUTPUT FREQUENCY SCALING (f_o)		S2	S3	PHOTODIODE TYPE
L	L	Power down		L	L	Red
L	H	2%		L	H	Blue
H	L	20%		H	L	Clear (no filter)
H	H	100%		H	H	Green

Available Options

DEVICE	T_A	PACKAGE – LEADS	PACKAGE DESIGNATOR	ORDERING NUMBER
TCS230	–40°C to 85°C	SOIC–8	D	TCS230D

Absolute Maximum Ratings over operating free-air temperature range (unless otherwise noted)†

Supply voltage, V_{DD} (see Note 1) . . . . . 6 V
Input voltage range, all inputs, V_I . . . . . –0.3 V to V_{DD} + 0.3 V
Operating free-air temperature range, T_A . . . . . –40°C to 85°C
Storage temperature range . . . . . –40°C to 85°C
Solder conditions in accordance with JEDEC J–STD–020A, maximum temperature (see Note 2) . . . 260°C

† Stresses beyond those listed under “absolute maximum ratings” may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under “recommended operating conditions” is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTES: 1. All voltage values are with respect to GND.
2. The device may be hand soldered provided that heat is applied only to the solder pad and no contact is made between the tip of the solder iron and the device lead. The maximum time heat should be applied to the device is 5 seconds.

Recommended Operating Conditions

		MIN	NOM	MAX	UNIT
Supply voltage, V_{DD}		2.7	5	5.5	V
High-level input voltage, V_{IH}	V_{DD} = 2.7 V to 5.5 V	2		V_{DD}	V
Low-level input voltage, V_{IL}	V_{DD} = 2.7 V to 5.5 V	0		0.8	V
Operating free-air temperature range, T_A		–40		70	°C

Electrical Characteristics at T_A = 25°C, V_{DD} = 5 V (unless otherwise noted)

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
V_{OH}	High-level output voltage	I_{OH} = –4 mA	4	4.5		V
V_{OL}	Low-level output voltage	I_{OL} = 4 mA		0.25	0.40	V
I_{IH}	High-level input current				5	μA
I_{IL}	Low-level input current				5	μA
I_{DD}	Supply current	Power-on mode		2	3	mA
		Power-down mode		7	15	μA
	Full-scale frequency (See Note 2)	S0 = H, S1 = H	500	600		kHz
		S0 = H, S1 = L	100	120		kHz
		S0 = L, S1 = H	10	12		kHz
	Temperature coefficient of output frequency	λ ≤ 700 nm, –25°C ≤ T_A ≤ 70°C		±200		ppm/°C
k_{SVS}	Supply voltage sensitivity	V_{DD} = 5 V ±10%		±0.5		%/V

NOTE 3: Full-scale frequency is the maximum operating frequency of the device without saturation.

Operating Characteristics at V_{DD} = 5 V, T_A = 25°C, S0 = H, S1 = H (unless otherwise noted) (See Notes 3, 4, 5, 6, and 7).

PARAMETER		TEST CONDITIONS	CLEAR PHOTODIODE S2 = H, S3 = L			BLUE PHOTODIODE S2 = L, S3 = H			GREEN PHOTODIODE S2 = H, S3 = H			RED PHOTODIODE S2 = L, S3 = L			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	MIN	TYP	MAX	MIN	TYP	MAX	
f_O	Output frequency	E_e = 47.2 μW/cm², λ_p = 470 nm	16	20	24	11.2	16.4	21.6							kHz
		E_e = 40.4 μW/cm², λ_p = 524 nm	16	20	24				8	13.6	19.2				kHz
		E_e = 34.6 μW/cm², λ_p = 640 nm	16	20	24							14	19	24	kHz
f_D	Dark frequency	E_e = 0		2	12		2	12		2	12		2	12	Hz
R_e	Irradiance responsivity (Note 8)	λ_p = 470 nm		424			348			81			26		Hz/(μW/cm²)
		λ_p = 524 nm		495			163			337			35		
		λ_p = 565 nm		532			37			309			91		
		λ_p = 640 nm		578			31			29			550		
	Saturation irradiance (Note 9)	λ_p = 470 nm		1410			1720								μW/cm²
		λ_p = 524 nm		1210						1780					
		λ_p = 565 nm		1130						1940					
		λ_p = 640 nm		1040									1090		
R_v	Illuminance responsivity (Note 10)	λ_p = 470 nm		565			464			108			35		Hz/lx
		λ_p = 524 nm		95			31			65			7		
		λ_p = 565 nm		89			6			52			15		
		λ_p = 640 nm		373			20			19			355		
	Nonlinearity (Note 11)	f_O = 0 to 5 kHz		±0.1%			±0.1%			±0.1%			±0.1%		% F.S.
		f_O = 0 to 50 kHz		±0.2%			±0.2%			±0.2%			±0.2%		% F.S.
		f_O = 0 to 500 kHz		±0.5%			±0.5%			±0.5%			±0.5%		% F.S.
	Recovery from power down			100			100			100			100		μs
	Response time to output enable ($\overline{OE}$)			100			100			100			100		ns

NOTES: 4. Optical measurements are made using small-angle incident radiation from a light-emitting diode (LED) optical source.
5. The 470 nm input irradiance is supplied by an InGaN light-emitting diode with the following characteristics: peak wavelength λ_p = 470 nm, spectral halfwidth $\Delta\lambda_{1/2}$ = 35 nm, and luminous efficacy = 75 lm/W.
6. The 524 nm input irradiance is supplied by an InGaN light-emitting diode with the following characteristics: peak wavelength λ_p = 524 nm, spectral halfwidth $\Delta\lambda_{1/2}$ = 47 nm, and luminous efficacy = 520 lm/W.
7. The 565 nm input irradiance is supplied by a GaP light-emitting diode with the following characteristics: peak wavelength λ_p = 565 nm, spectral halfwidth $\Delta\lambda_{1/2}$ = 28 nm, and luminous efficacy = 595 lm/W.
8. The 640 nm input irradiance is supplied by a AlInGaP light-emitting diode with the following characteristics: peak wavelength λ_p = 640 nm, spectral halfwidth $\Delta\lambda_{1/2}$ = 17 nm, and luminous efficacy = 155 lm/W.
9. Irradiance responsivity R_e is characterized over the range from zero to 5 kHz.
10. Saturation irradiance = (full-scale frequency)/(irradiance responsivity).
11. Illuminance responsivity Rv is calculated from the irradiance responsivity by using the LED luminous efficacy values stated in notes 4, 5, and 6 and using 1 lx = 1 lm/m².
12. Nonlinearity is defined as the deviation of f_O from a straight line between zero and full scale, expressed as a percent of full scale.

TYPICAL CHARACTERISTICS

PHOTODIODE SPECTRAL RESPONSIVITY

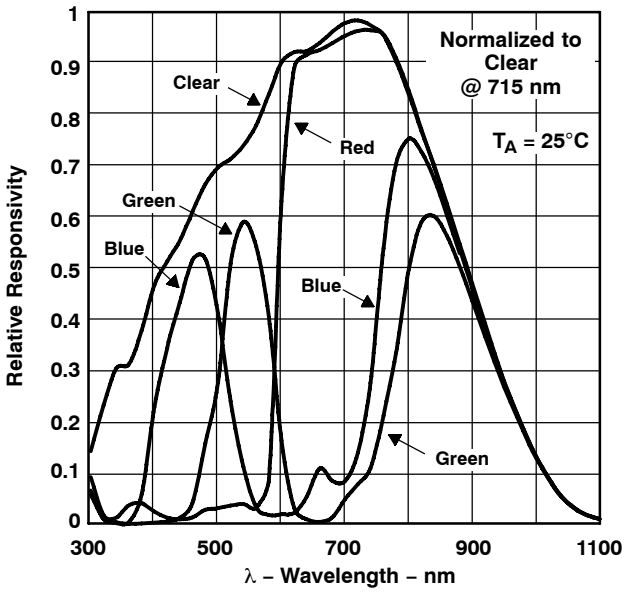
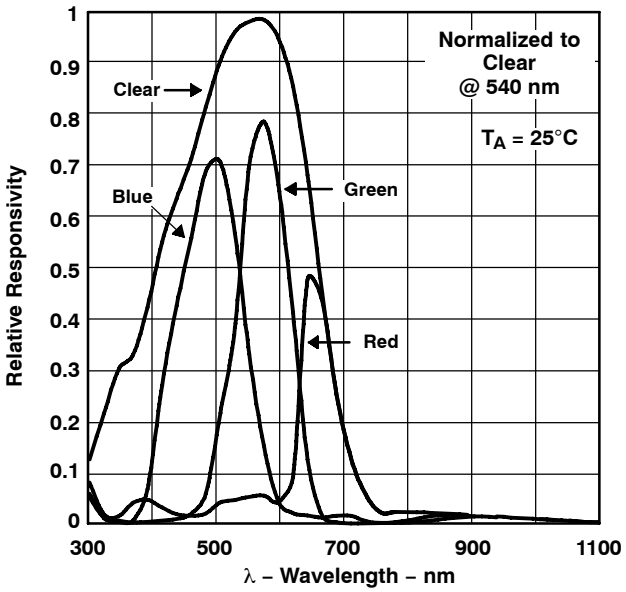


Figure 1

PHOTODIODE SPECTRAL RESPONSIVITY WITH EXTERNAL IR-BLOCKING FILTER†



† Typical IR filter examples include Schott BG18, Schott BG39, and Hoya CM500.

Figure 2

NORMALIZED OUTPUT FREQUENCY vs. ANGULAR DISPLACEMENT

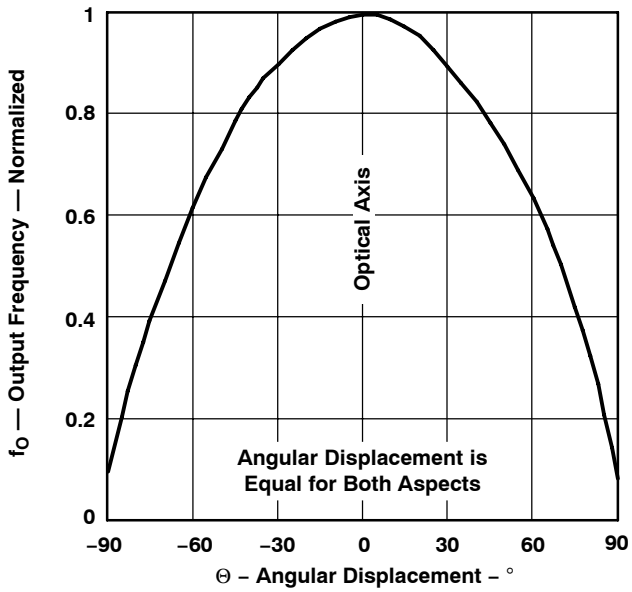
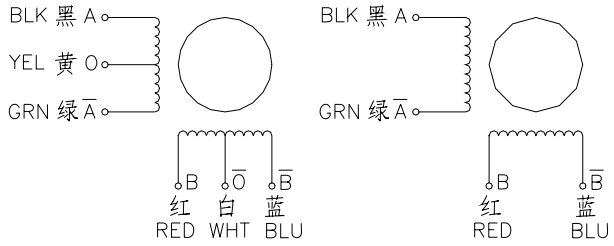


Figure 3

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

Wiring Diagram:



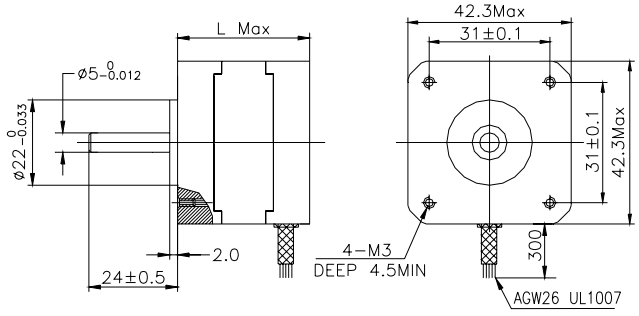
UNI-POLAR(6 LEADS) BI-POLAR(4LEADS)

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

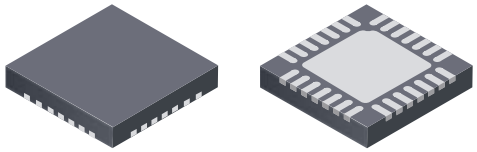
DMOS Microstepping Driver with Translator And Overcurrent Protection

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

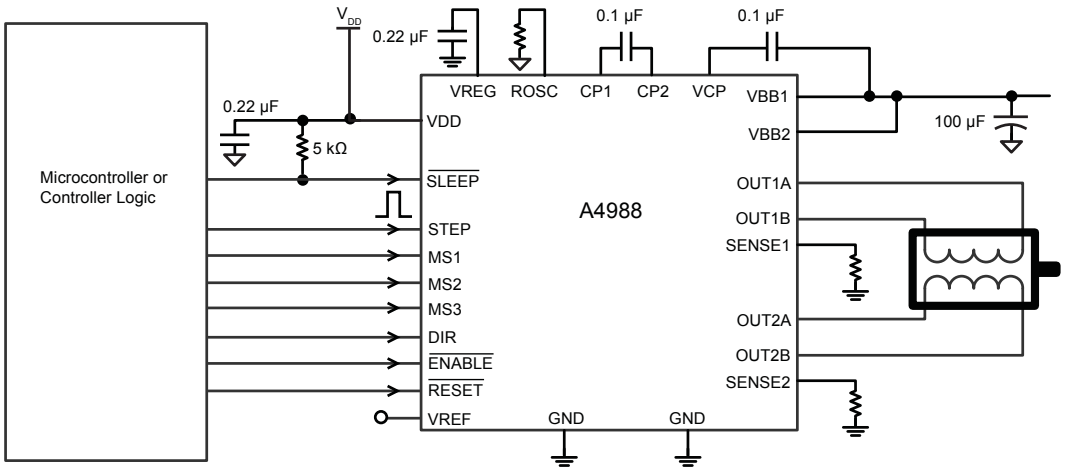
The A4988 is a complete microstepping motor driver with built-in translator for easy operation. It is designed to operate bipolar stepper motors in full-, half-, quarter-, eighth-, and sixteenth-step modes, with an output drive capacity of up to 35 V and ±2 A. The A4988 includes a fixed off-time current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

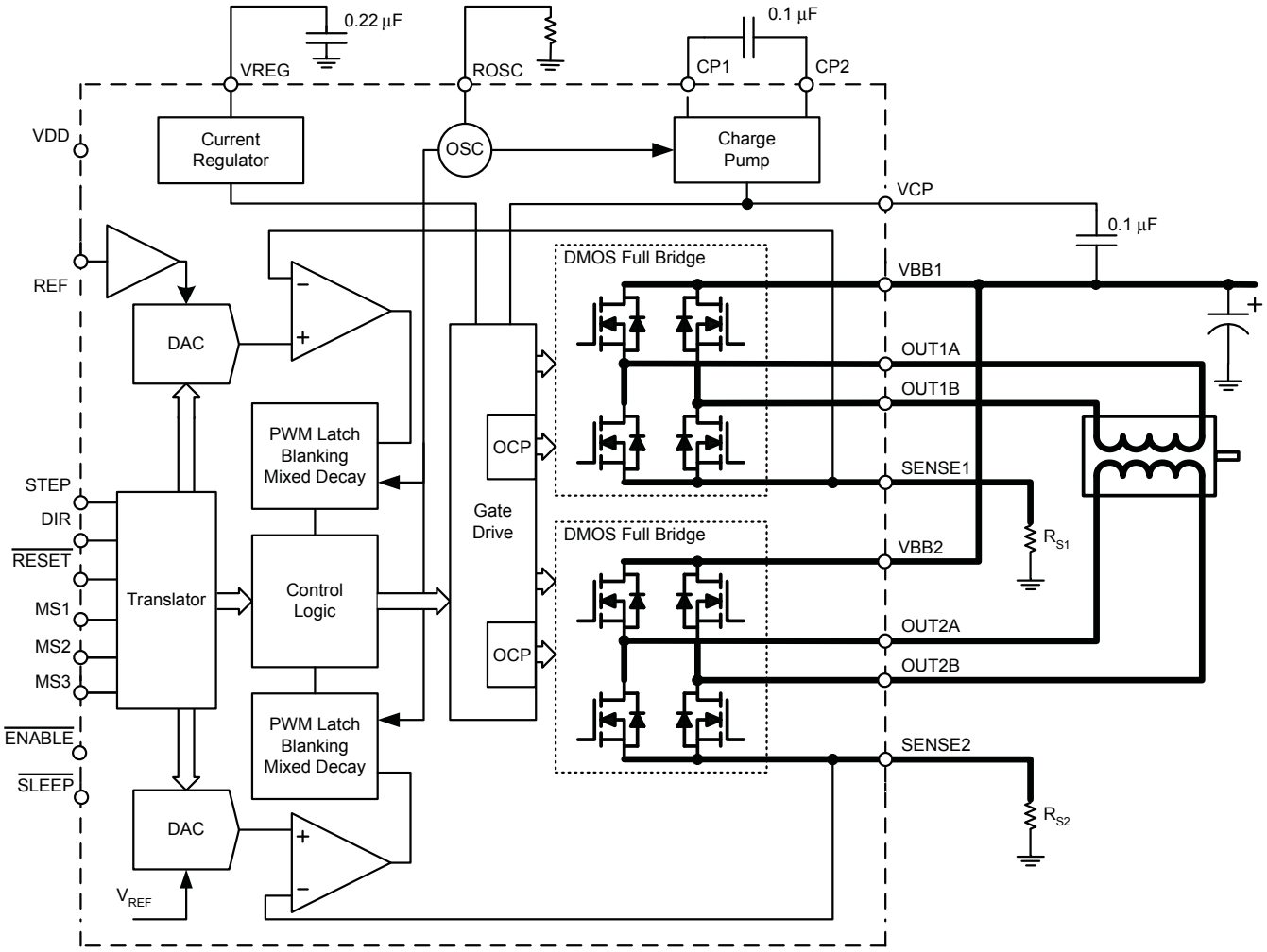
Selection Guide

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).

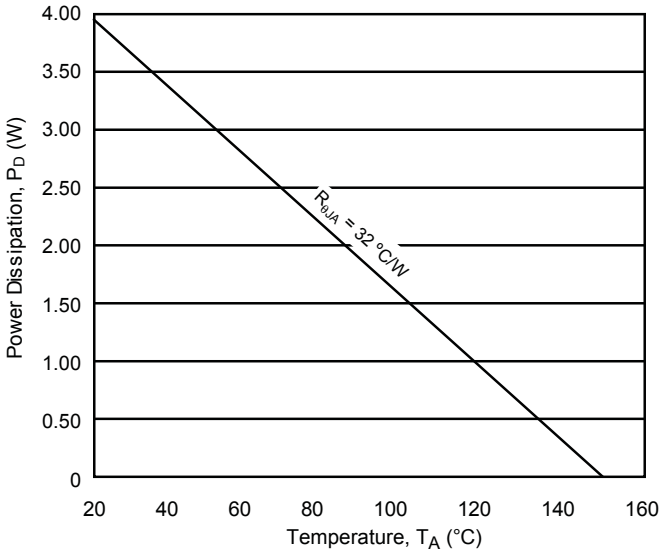
[4]Overcurrent protection (OCP) is tested at T_A = 25°C in a restricted range and guaranteed by characterization.

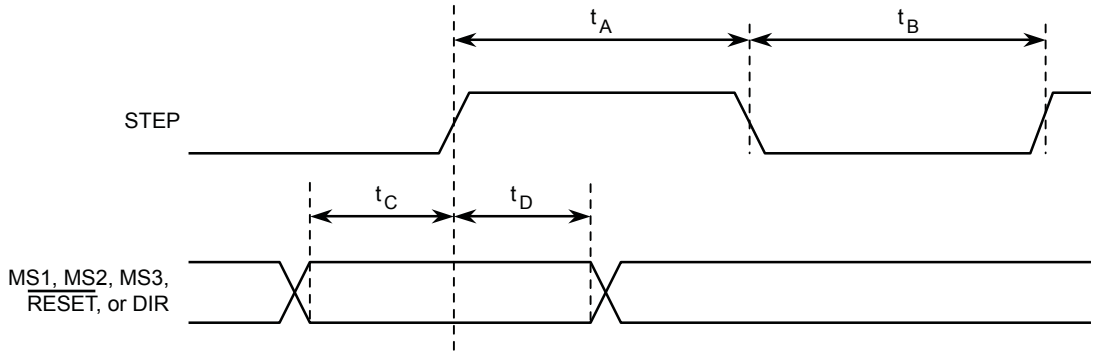
THERMAL CHARACTERISTICS

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