



Internet of Things: Theory and Applications

Module 2: ESP32 based IoT Systems – Part B

Introduced by: Eng. Mohamed Hatem

Teaching Assistant at the Faculty of Computers and Data Science in Alexandria University

Graduate of Nanotechnology and Nano-electronics Engineering Program at the Zewail City of Sciences, Technology, and Innovation

Graduate Student Member at IEEE

Session Outline:

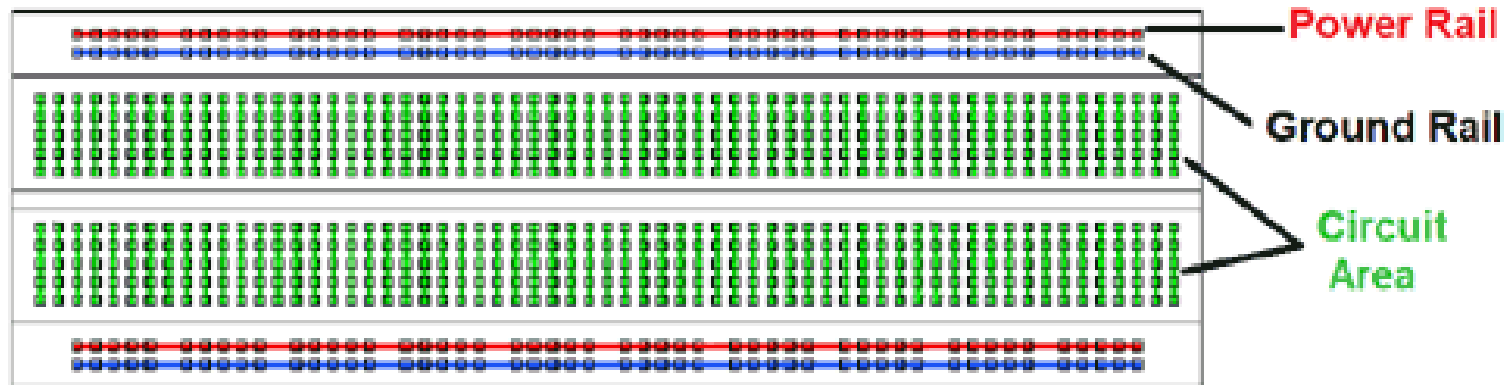
- Section 0: A Recap on Practical Basics of Electronics & Embedded Systems
- Section 1: Sensors Interfacing using ESP32
- Section 2: Actuators Interacting using ESP32
- Section 3: Output Modules Interfacing using ESP32
- Section 4: Input Modules Interfacing using ESP32



A Recap on Practical Basics of Electronics & Embedded Systems

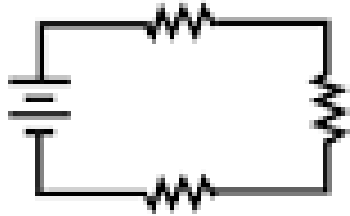
Internet of Things: Theory and Applications

What is the Breadboard?

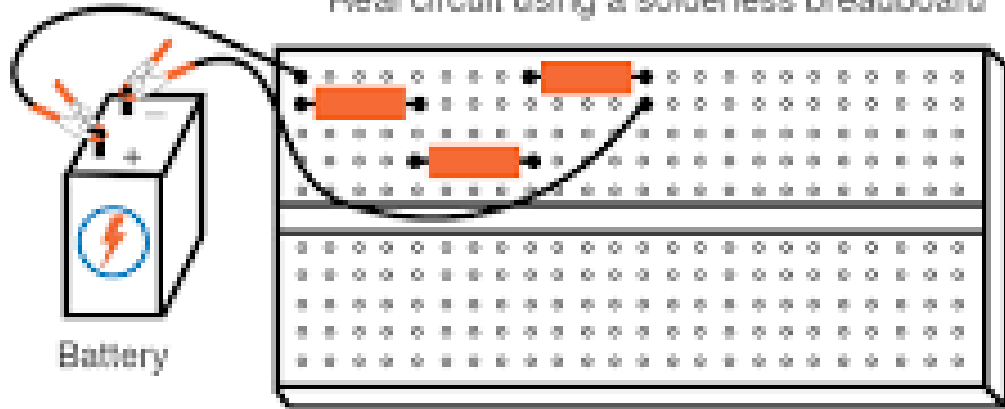


Series and Parallel Connections on Breadboard

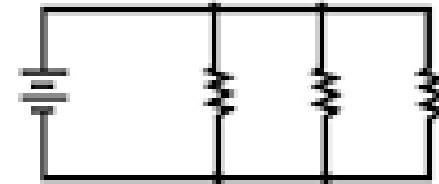
Schematic diagram



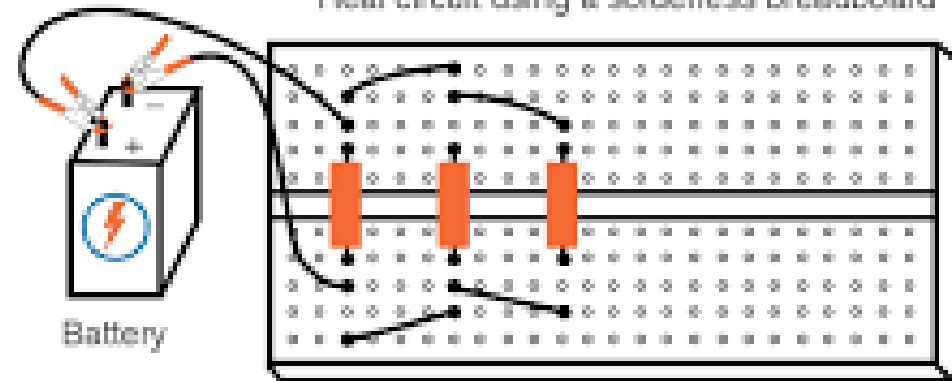
Real circuit using a solderless breadboard



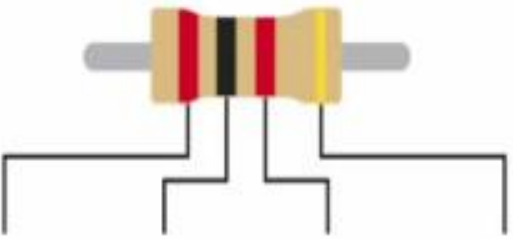
Schematic diagram



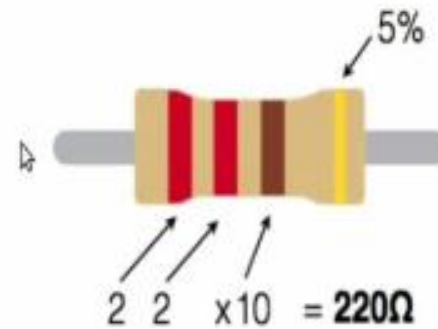
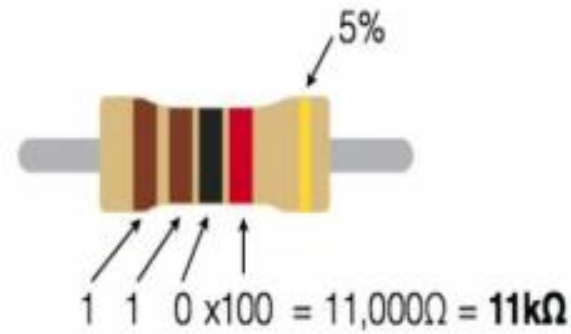
Real circuit using a solderless breadboard



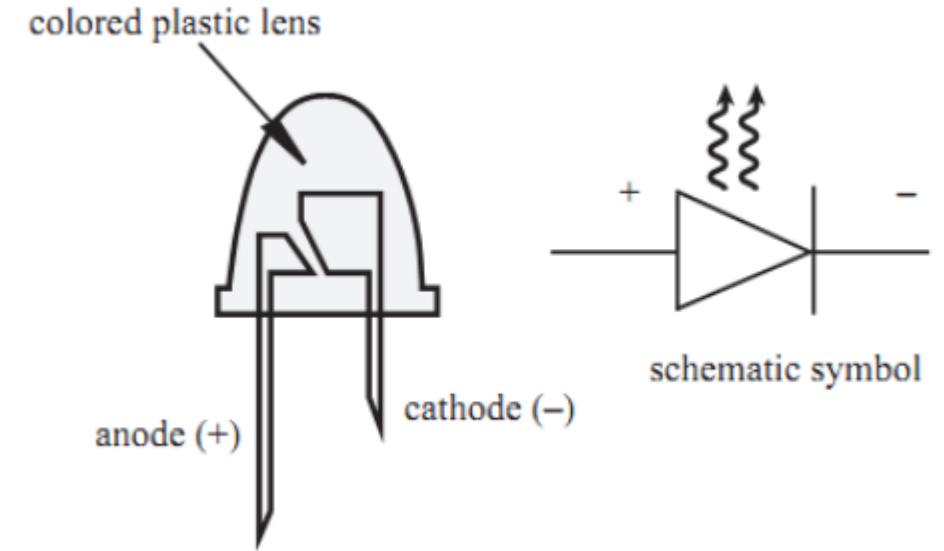
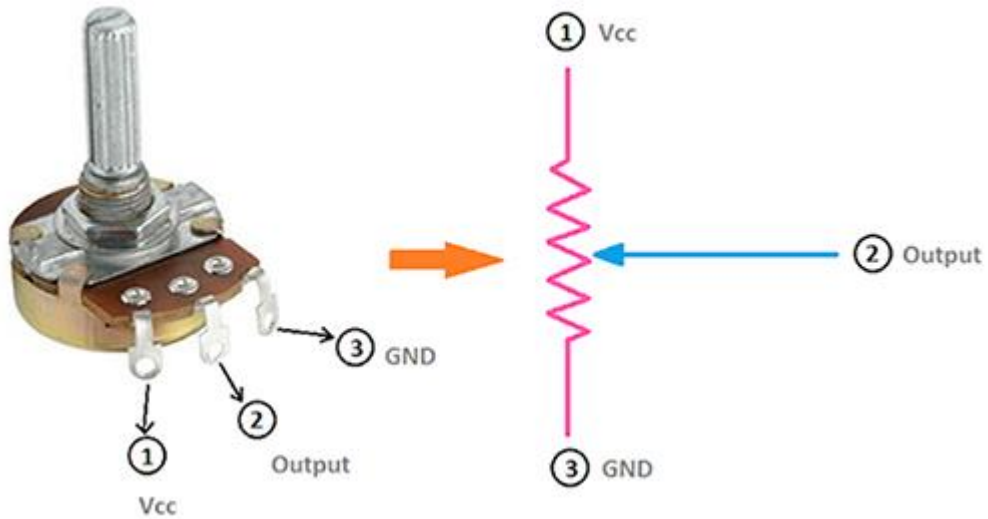
Resistors & How to Read them?



1st digit	2nd digit	Multiplier	Tolerance
0	0	x1	
1	1	x10	±1%
2	2	x100	±2%
3	3	x1K	
4	4	x10K	
5	5	x100K	
6	6	x1M	
7	7		
8	8	x0.1	±5%
9	9	x0.01	±10%



Potentiometers & LEDs

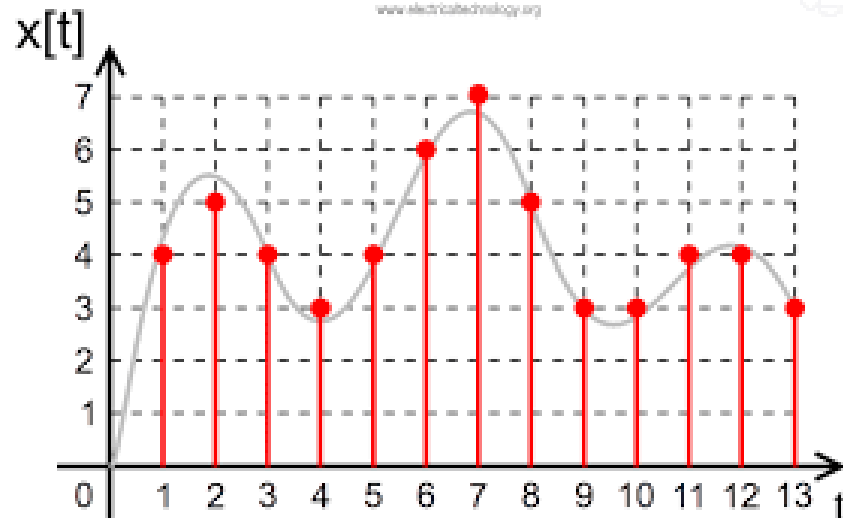


What is Analog to Digital Converter?

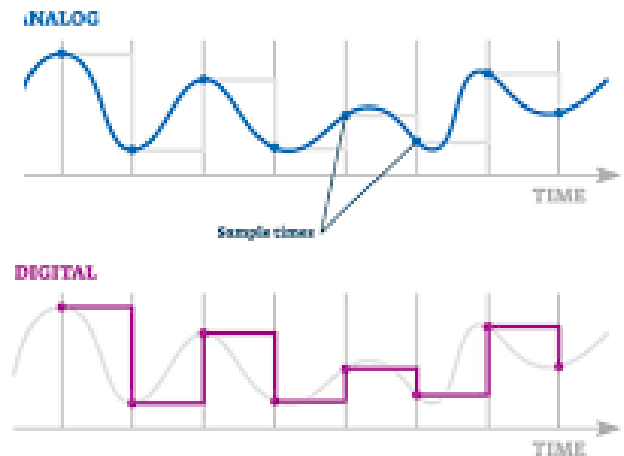
ADC

ANALOG TO DIGITAL CONVERTER

Types, Working, Block Diagram & Applications



ANALOG VS DIGITAL SIGNAL

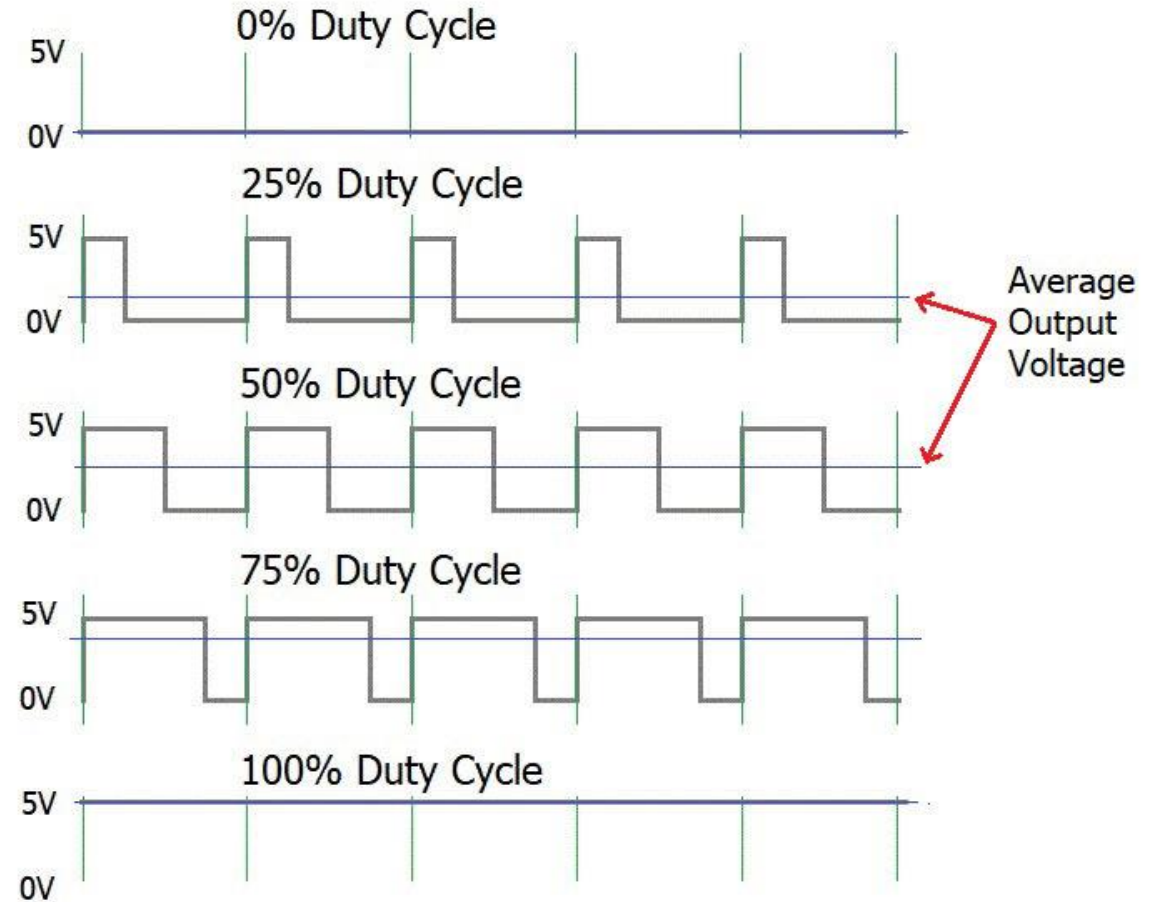
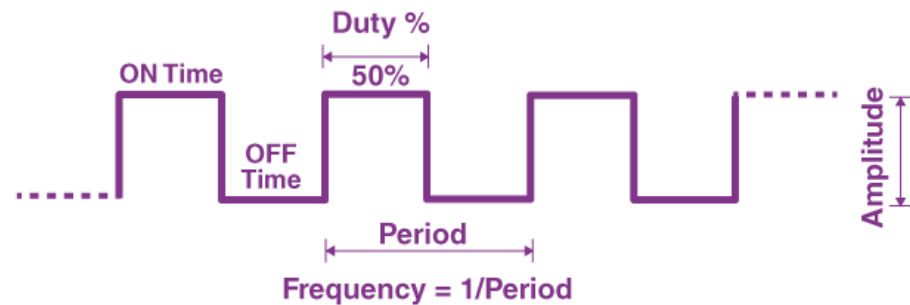


What is Pulse Width Modulation?

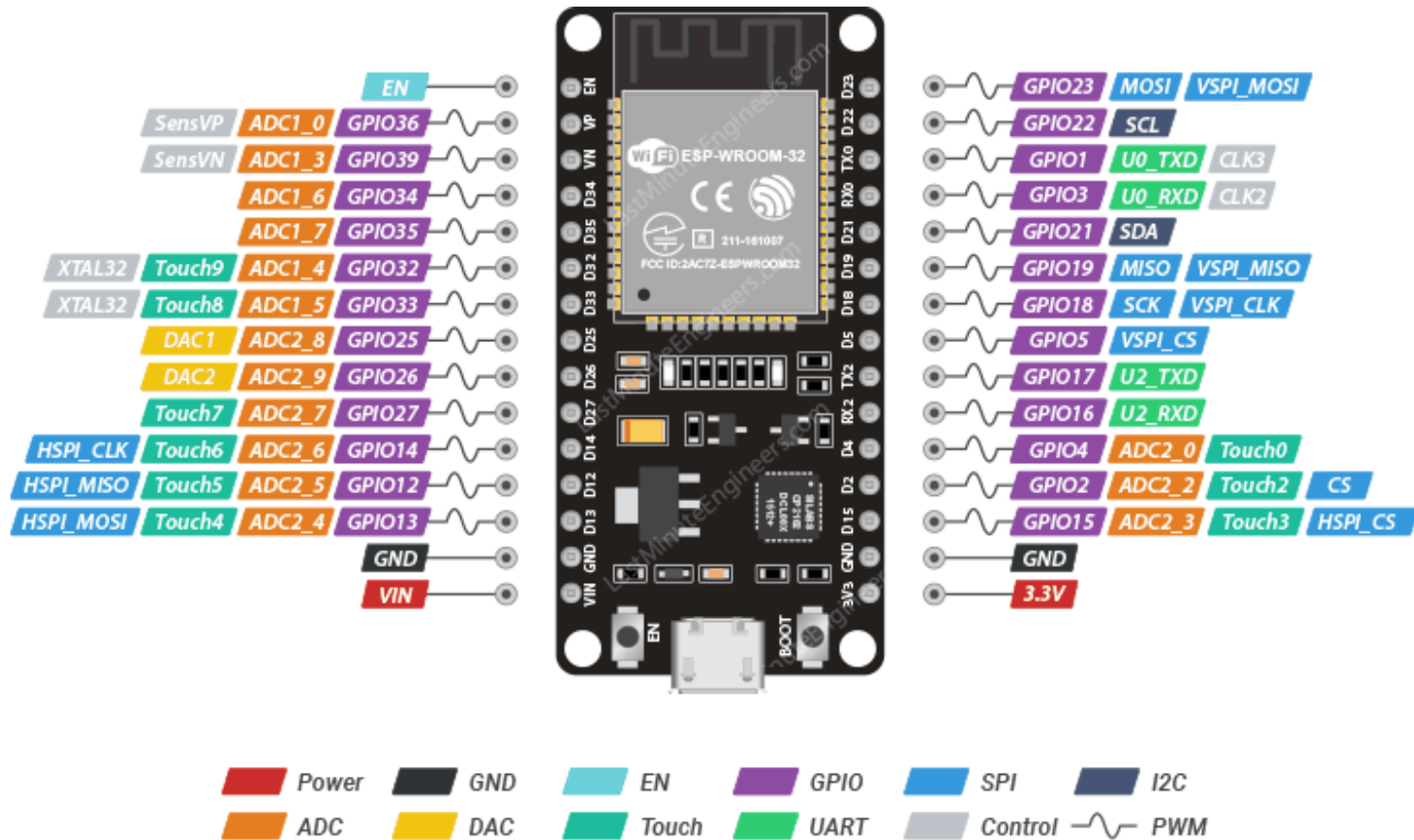


What is Pulse Width Modulation?

$$\text{DUTY CYCLE} = \frac{T_{\text{high}}}{\text{PERIOD}} \times 100$$

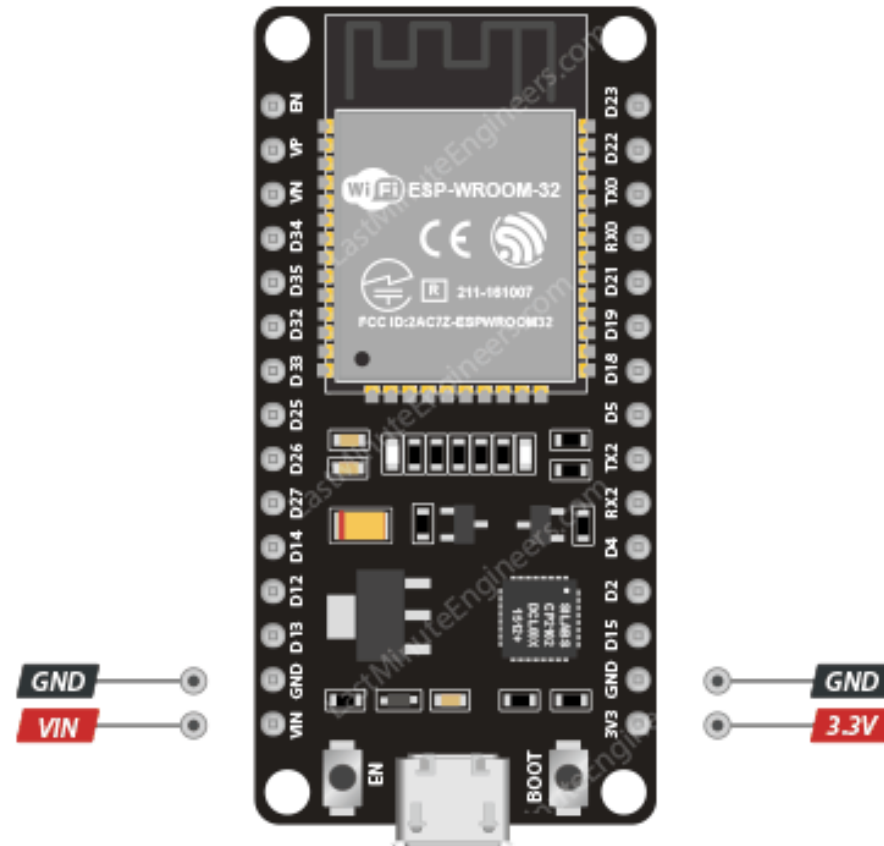


Layout of ESP32

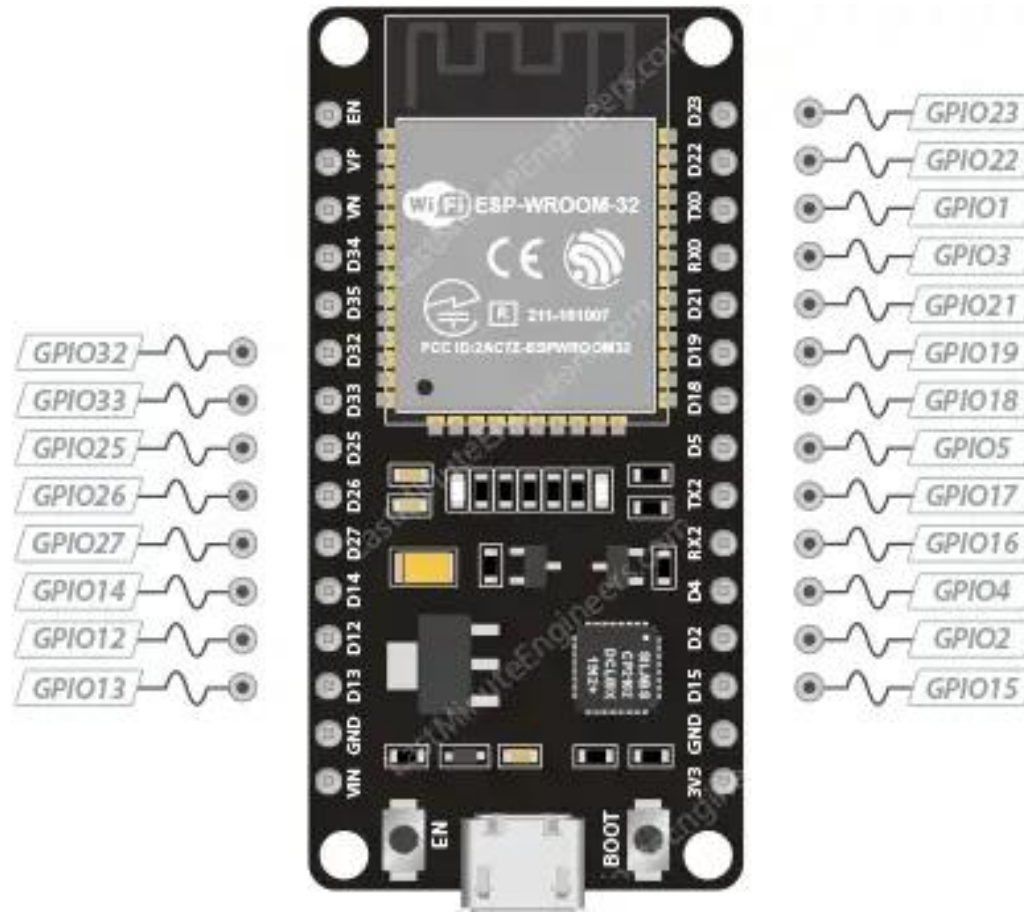


ESP32 Dev. Board Pinout

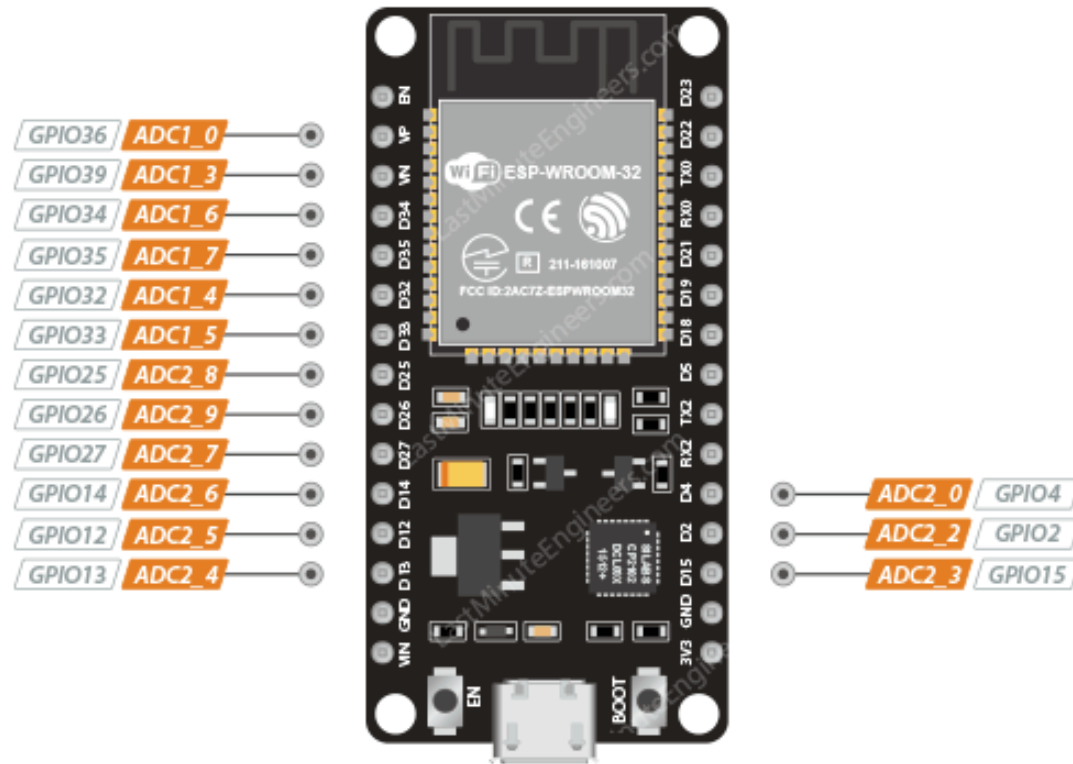
Power Pins



PWM Pins



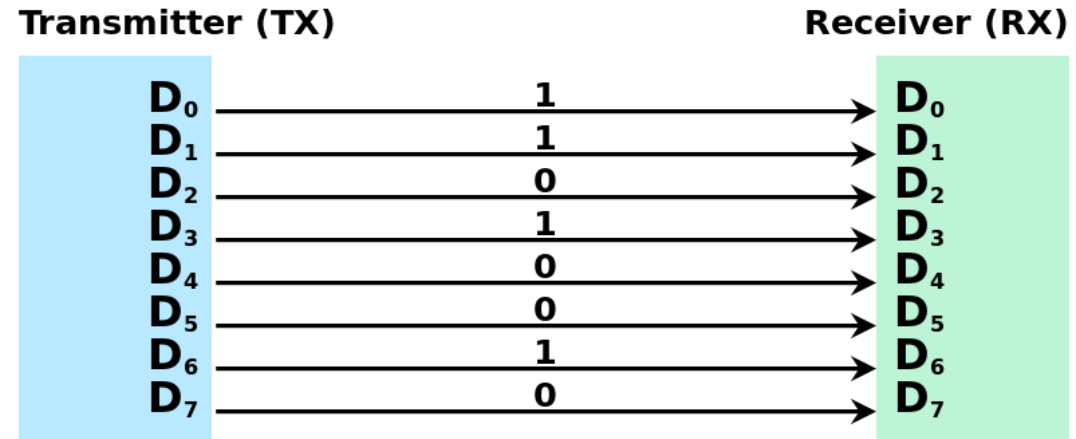
ADC Pins



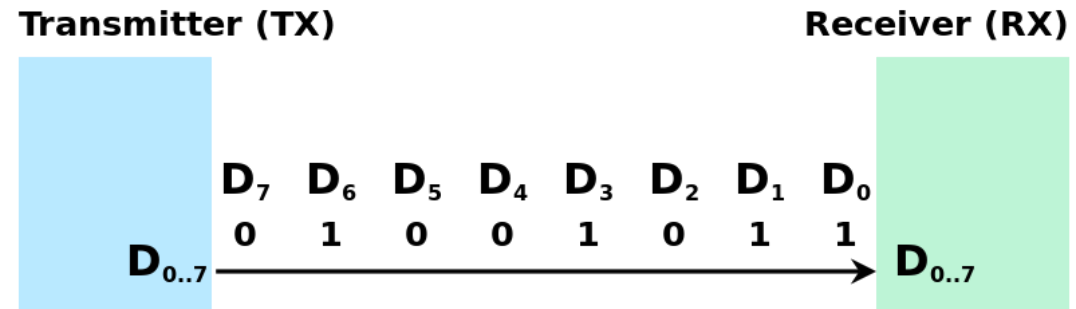
What is Serial Communication & What is the Baud Rate ?

- Baud rate is the measure of the number of changes to the signal (per second) that propagate through a transmission medium.

Parallel interface example



Serial interface example



Important ESP32 Arduino Syntax & Functions

- Void Set up > Done Once
- Void Loop > Done Forever
- Digital Write > Write 0 or 1
- Digital Read > Read 0 or 1
- Analog Write > Write from 0 to 255
- Analog Read > Read from 0 to 4095
- Map Function > Maps from Min1 & Max1 to Min2 & Max2
- Serial Print and Serial Print In > Prints on Serial Monitor



Sensors Interfacing using ESP32

Internet of Things: Theory and Applications

What are Transducers?

- The transducers are used for **measurements of the system parameters.**
- The transducer is a device that **produces a measurable response to a change in a physical condition** such as temperature, pressure, humidity, flow, light intensity, vibration, etc.

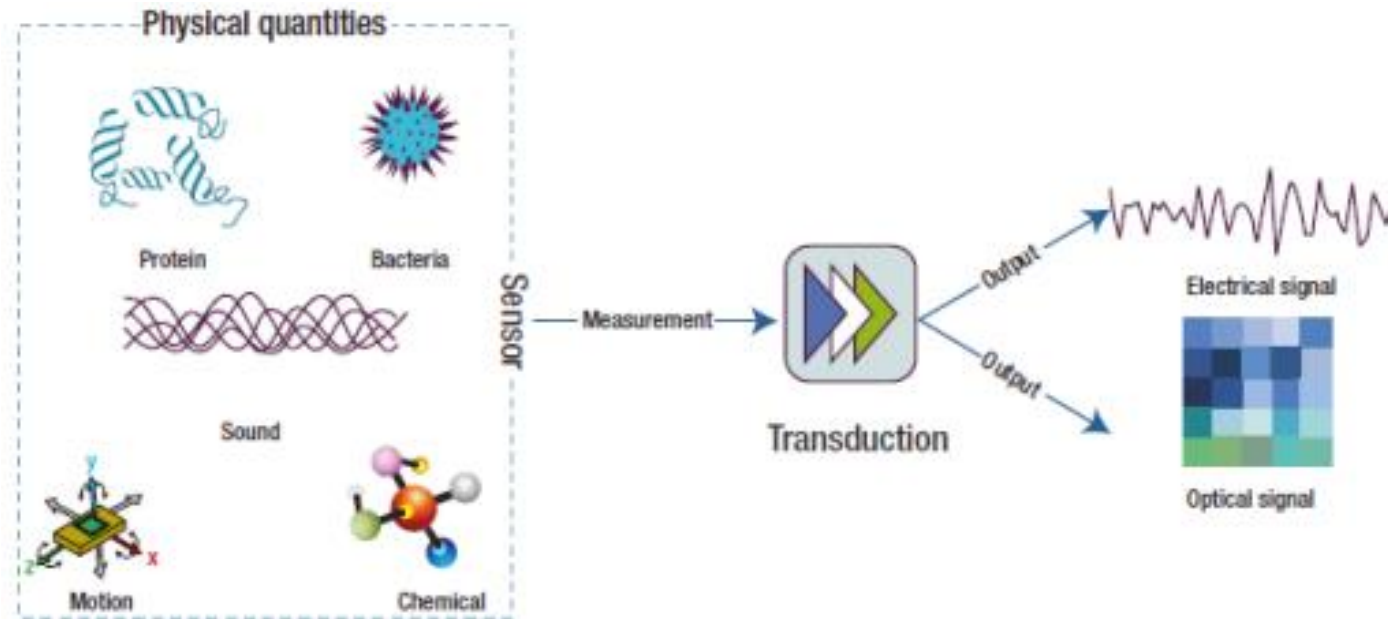
Classification of Transducers

- **Passive equivalences** are **resistance, inductance and capacitance**. The transducers of this category are made up of **specific material**, such that their electrical properties like resistance, inductance, and capacitance vary in response to change in input measurands.
- **Active equivalences** are those which directly provide **electrical signals** like voltage, current either in the form of DC or AC.

What are Sensors?

- The Sensor is a device that **receives a stimulus and responds with an electrical signal.**
- The sensor is an **special type of transducer (device that converts one type of energy into another)**

Sensing Process



Common Types of Sensors & Transducers

- Limit Switches/Touch Sensors
- Magnetic Proximity Detectors
- Capacitive Proximity Detectors
- Infra Red Sensors
- Laser Sensors
- Ultra Sonic Sensors
- Temperature Sensors/Switches
- Pressure Sensors/Switches
- Speed Sensors
- Humidity Sensors

Common Types of Sensors & Transducers

- Position Sensors
- Absolute Shaft Encoders
- Relative Shaft Encoders
- Wind Velocity Sensors
- Thickness Sensors
- Tension Sensors
- Flow Sensors/Switches
- Color Sensors
- Depth Sensors
- Flatness Sensors

Defining Some Sensors Characteristics

- **Accuracy:** The agreement between the actual value and the measured value
- **Resolution:** The change in measured variable to which the sensor will respond
- **Repeatability:** Variation of sensor measurements when the same quantity is measured several times
- **Range:** Upper and lower limits of the variable that can be measured which includes full scale range and operating voltage range.
- **Sensitivity:** The change in input required to generate a unit change in output
- **Linearity**
- **Transfer Function:** $S=F(x)$, where x is the measurand and S the electrical signal (commonly Voltage)

Types of Errors

- **Error**: the difference between the measured value and true value.
- **Systematic Errors (Controllable Errors)**: These errors are consistent and repeatable, leading to a constant deviation from the true value. Systematic errors can be managed and corrected through calibration and adjustment methods.
- **a. Zero Error**: The sensor reading is nonzero when the input signal is zero. This error can be calibrated out by applying an offset correction.
- **b. Scale Error**: The sensor's output does not scale linearly with the input signal. Scaling factors can be adjusted during calibration.
- **c. Proportional Error**: The sensor exhibits a constant proportional deviation from the true value. Calibration can help correct this error.

Types of Errors

- **Random Errors (Uncontrollable Errors):** These errors are unpredictable and vary with each measurement. They are caused by various factors, such as noise, environmental variations, or inherent limitations in the sensor's design.
- **a. Noise:** Random fluctuations in the measured signal that can be caused by electrical interference or external disturbances.
- **b. Drift:** Gradual changes in the sensor's output over time, even with a constant input signal.
- **c. Environmental Interference:** Changes in the measurement due to external factors like temperature, humidity, or electromagnetic fields.
- **d. Resolution Error:** The smallest increment that can be detected by the sensor, leading to rounding or quantization errors.

Controllable Errors in Sensors

- Controllable errors in sensors **refer to the inaccuracies or deviations in the sensor readings** that can be managed, minimized, or calibrated to improve their precision and reliability. Unlike uncontrollable errors, which are inherent to the sensor's design and cannot be adjusted, **controllable errors can be addressed through various methods to enhance the sensor's performance.** Some common controllable errors in sensors include:
- **Calibration Error:** This occurs when the sensor's output does not match the true value of the measured parameter. Calibration techniques can be employed to adjust the sensor's output to align it with the actual value.
- **Zero Offset Error:** It is the deviation in the sensor's output when the input signal is zero. Zero offset can be corrected by applying a correction factor during calibration.

Controllable Errors in Sensors

- **Linearity Error:** This error arises when the sensor's response is not perfectly linear across the measurement range. Linearization techniques or piecewise calibration can be used to improve linearity.
- **Hysteresis Error:** Hysteresis is the difference in sensor output for the same input value when approached from different directions. This error can be minimized through compensation methods during calibration.
- **Temperature Drift:** Sensors can exhibit changes in output with variations in temperature. Temperature compensation methods can be applied to reduce the impact of temperature drift.
- **Noise and Interference:** External factors, such as electromagnetic interference or mechanical vibrations, can introduce noise in sensor readings. Shielding and filtering techniques can be used to mitigate these effects.
- **Response Time:** The time taken by a sensor to reach a stable output after a change in the measured parameter can affect accuracy. Improving sensor design or using signal processing techniques can help reduce response time.
- **Sensitivity Drift:** Sensitivity drift occurs when the sensitivity of the sensor changes over time. Periodic recalibration can help maintain consistent sensitivity.

Accuracy vs Precision



**High Accuracy
High Precision**



**Low Accuracy
High Precision**



**High Accuracy
Low Precision**



**Low Accuracy
Low Precision**

Common IoT & Embedded Systems Sensors

1) Proximity Sensors:

- Mechanical Proximity Sensors,
- Inductive Proximity Sensors
- Capacitance Proximity Sensors
- **Optical Proximity Sensors**
- Ultrasonic Proximity Sensor

Common IoT & Embedded Systems Sensors

2) Position and Velocity Measurements

- Linear Glass Scales
- Potentiometers
- Linear Variable Differential Transformer
- Eddy Current Probes
- Capacitive Sensors
- Speed and Position Encoders

Common IoT & Embedded Systems Sensors

- 3) Acceleration Measurements using IMU
- 4) Force and Pressure Measurements
- 5) Temperature Measurements
- 6) Flow Measurements
- 7) Level Measurements
- 8) Embedded Computer Vision

Proximity Sensors

- Widely used in general industrial automation conveyor lines (counting, jam detection, etc.)
- Usually digital (on/off) sensors detecting the presence or absence of an object but there are also analog proximity sensors.
- Consist of:
 - **Sensor head:** optical, inductive, capacitive
 - **Detector circuit**
 - **Amplifier**
 - **Output circuit:** TTL, solid state relay

Types of Proximity Sensors

- **Mechanical Proximity Sensors:** Mechanical proximity sensors are simple and robust devices that use physical contact to detect the presence or absence of an object. When the object comes in contact with the sensor's actuator, it triggers a mechanical response, indicating the presence of the object. These sensors are commonly used in industrial applications for limit and position sensing.
- **Inductive Proximity Sensors:** Inductive proximity sensors work based on the principle of electromagnetic induction. They generate an electromagnetic field and detect changes in the field when a conductive object comes close to the sensor. The presence of the object alters the field, triggering the sensor to detect its presence. Inductive proximity sensors are used in various industrial automation applications for detecting metallic objects without physical contact.
- **Capacitive Proximity Sensors:** Capacitive proximity sensors detect the presence of an object by measuring changes in capacitance. When an object enters the sensor's electric field, it alters the capacitance, which is then detected by the sensor. These sensors are sensitive to both conductive and non-conductive materials, making them suitable for detecting various objects in industrial and consumer electronics applications.

Types of Proximity Sensors

- **Optical Proximity Sensors:** Optical proximity sensors use light beams to detect the presence of an object. They emit infrared or visible light, and when the light is reflected back to the sensor, it detects the presence of the object. Optical proximity sensors are widely used in devices like smartphones, printers, and automated vending machines for object detection and position sensing.
- **Ultrasonic Proximity Sensors:** Ultrasonic proximity sensors use sound waves to detect objects and measure distances. They emit high-frequency sound waves, and when these waves encounter an object, they reflect back to the sensor. The sensor calculates the time taken for the waves to return, providing information about the distance to the object. Ultrasonic proximity sensors are commonly used in robotics, automotive parking systems, and industrial automation for distance measurement and object detection.

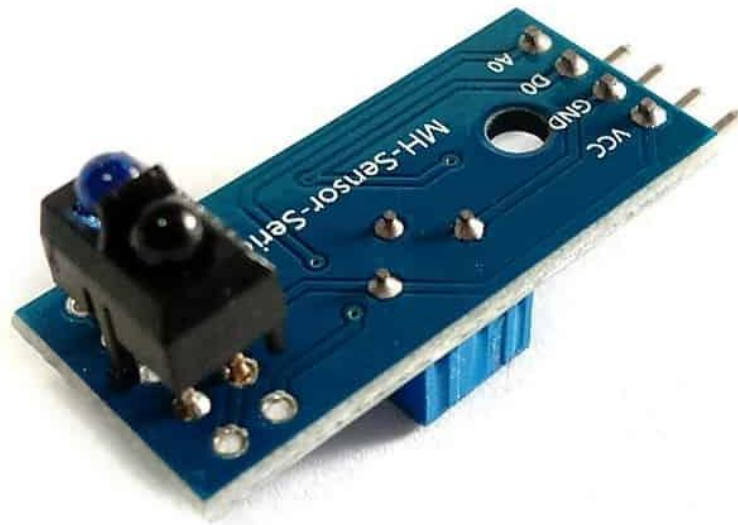
Optical Proximity – Infrared Optical based Sensors

- Proximity sensors are devices that **detect the presence or absence of an object within a certain range without any physical contact.** They are widely used in various industries and applications for automation, object detection, and position sensing.
- Proximity sensors can be based on different principles, such as electromagnetic induction, capacitance, optical, ultrasonic, or **infrared (IR) radiation.**
- An example of a proximity sensor is **the MH series IR analog and digital output sensor.**

Optical Proximity – Infrared Optical based Sensors

- This sensor is designed to detect **the presence of an object within a specific range using infrared radiation.** It emits infrared light and measures the reflection or absorption of the light when it encounters an object. The sensor then converts this information into an analog or digital output signal, indicating the presence or absence of the object.
- The MH series IR sensor is available in **both analog and digital output variants.** The analog output sensor provides a **continuous voltage or current signal that varies proportionally with the distance of the detected object.** The output signal can be connected to an analog input of a microcontroller or PLC for further processing.
- On the other hand, the **digital output sensor provides a binary output signal, typically in the form of a logic high or low,** indicating whether an object is within the sensing range or not. This type of sensor is often used in applications where a simple presence or absence detection is sufficient.

Infrared MH Sensor Series



Infrared MH Sensor Series

- This is an easy to use sensor module you can use to **sense the intensity of infrared light reflection on surface.** You can use this as the line tracking sensor to make line following robots, or proximity sensor to sense object or obstacle in front of the optical sensor pair.
- **Note :** this sensor can **works with both analog and digital mode.** You can use its analog output **(AO) to get the intensity reading or digital output (DO), adjust the potentiometer (a.k.a. trimmer) to set the digital output sensitivity.**

Infrared MH Sensor Series

- **Specific Features**

- Sensor IC : TCRT5000
- Range of detection : ~60 mm
- Sensitive to black and white color surfaces

- **Common Features**

- Logical IC : LM393
- Operating voltage : 3.3 – 5V
- Output current : 15 mA
- Adjustable sensitivity via potentiometer
- Comes with LED indicators for POWER and OUTPUT
- Fixed bolt holes for easy installation
- Dimension : 32 x 14 mm

Infrared MH Sensor Series Pinout

Label	Meaning	Connection
VCC	Power source	Connect to 3.3V or 5V of system
GND	Ground	Connect to GND of system
DO	Digital Output	Connect to any digital IO pin
AO	Analog Output	Connect to analog input pin

Infrared MH Sensor Series

Working Principle:

- The MH Series IR sensor works based on the principle of infrared (IR) light reflection. It consists of two main components: **a transmitter LED and a photodiode. When the sensor is powered on, the transmitter LED emits infrared light. This emitted light strikes the nearby objects within the sensor's detection range.** The objects reflect some of the infrared light back towards the sensor. The photodiode in the sensor receives this reflected infrared light.
- The **amount of reflected infrared light received by the photodiode depends on the distance between the sensor and the object.** The photodiode generates an electric current proportional to the intensity of the received infrared light. This generated electric current is then converted into a corresponding voltage signal.

Infrared MH Sensor Series

Math Formula (Analog Output):

Analog output of the MH Series IR sensor is represented by a voltage signal. The sensor provides a resolution of 4095 analog steps (12-bit resolution) to convert the generated electric current into a voltage value. The analog output voltage can be calculated using the following formula:

- Analog Output Voltage (V_{out}) = (Analog Value / 4095) * V_{ref}

Where:

- Analog Value: The analog value read from the sensor (ranging from 0 to 4095).
- V_{ref} : The reference voltage provided to the analog input pin of the microcontroller. The ESP32 microcontroller has an internal voltage reference (V_{ref}) of 1100mV, which is typically used as the default reference voltage for the ADC.

The V_{ref} is typically the voltage supplied to the microcontroller's analog reference pin, which sets the maximum voltage range for the analog input.

Infrared MH Sensor Series

Applications in IoT:

The MH Series IR sensor with analog and digital output finds application in various IoT projects, including:

- 1) Distance measurement for obstacle avoidance in robotics and drones.
- 2) Object detection in home automation systems.
- 3) Proximity sensing for smart lighting and security systems.
- 4) Gesture recognition in wearable devices and human-machine interfaces.
- 5) Industrial automation for detecting the presence of objects on conveyor belts.

Infrared MH Sensor Series

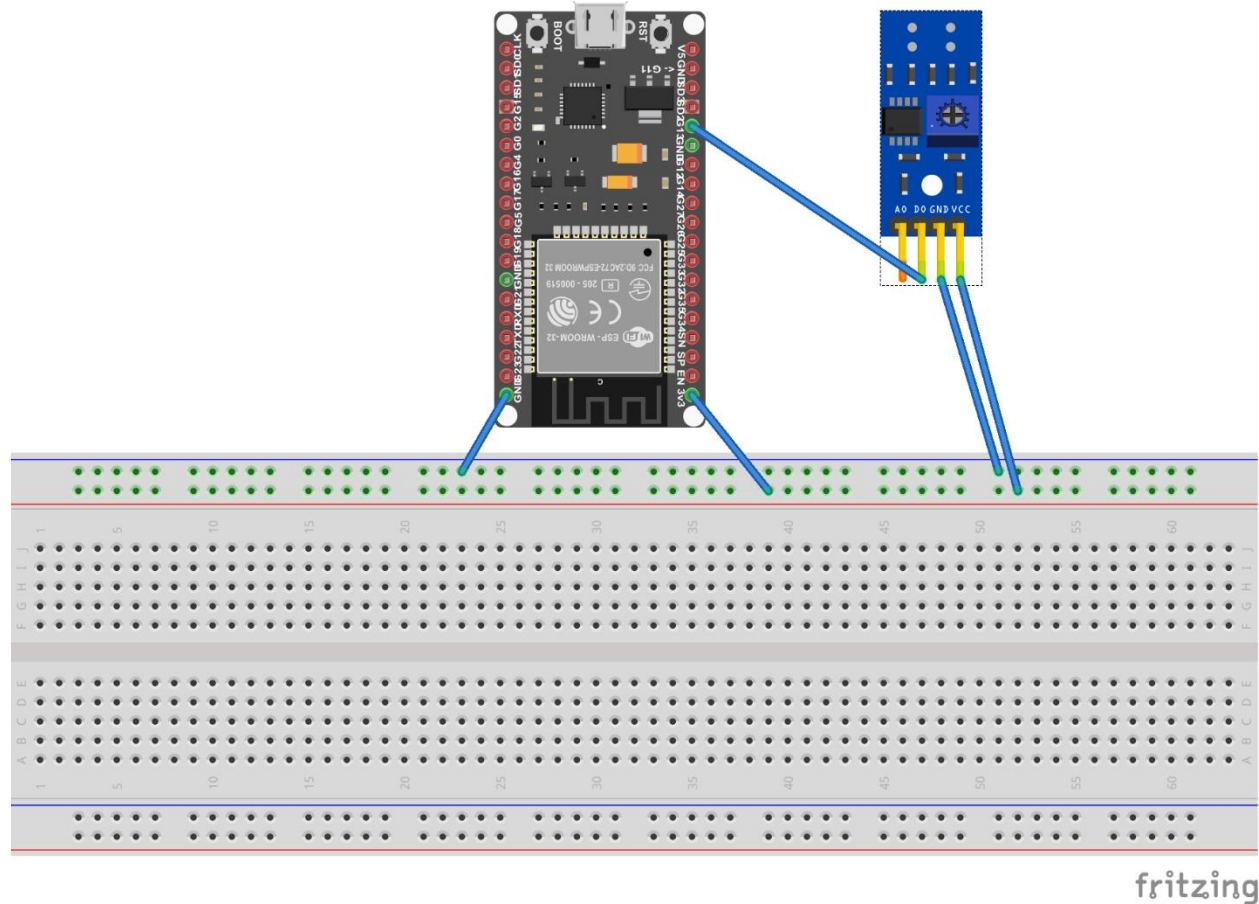
Connecting to ESP32: To connect the MH Series IR sensor with analog output to an ESP32 microcontroller, follow these steps:

- 1) Connect the output of the IR sensor to one of the analog input pins (ADC) of the ESP32.
- 2) Connect the sensor's ground (GND) pin to the ground (GND) pin of the ESP32.
- 3) Connect the sensor's power supply (VCC) pin to the appropriate voltage source (3.3V or 5V, depending on the sensor's specifications) of the ESP32.

Example Problem

- Design and interface an MH Series IR sensor with an ESP32 microcontroller to **detect the presence of objects and print the output digitally.** The objective is to develop a system that can accurately sense the presence of objects within its proximity using the IR sensor and communicate this information to the ESP32, which will then display the result as a digital output.

Schematic



Code

```
#define SENSOR_PIN 13 // ESP32 pin GPIO18 connected to OUT pin 13
void setup() {
    Serial.begin(115200);
    pinMode(SENSOR_PIN, INPUT);
}
```

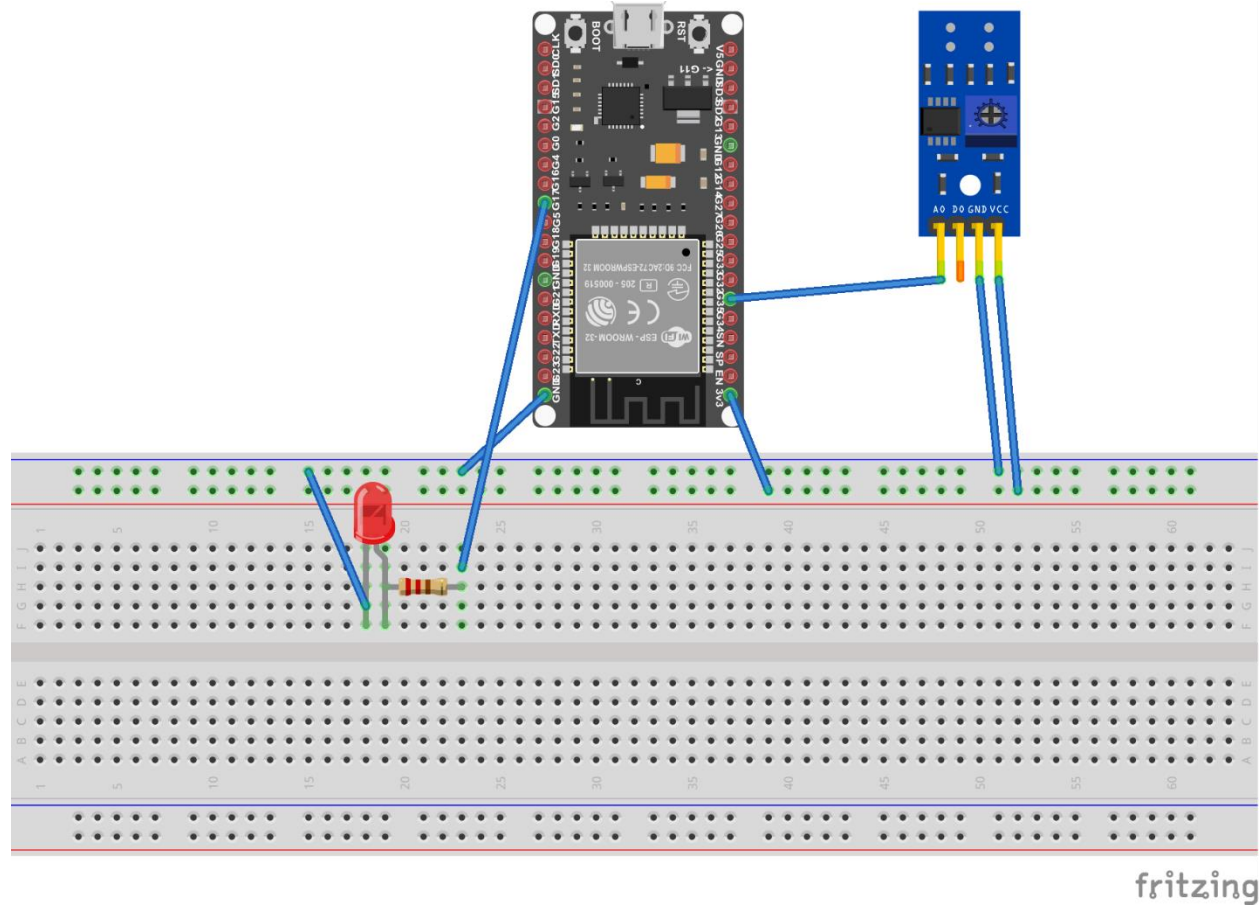

Code

```
void loop() {  
    // read the state of the the input pin:  
    int state = digitalRead(SENSOR_PIN);  
  
    if (state == LOW)  
        Serial.println("The obstacle is present");  
    else  
        Serial.println("The obstacle is NOT present");  
  
    delay(100);  
}
```

Practice Problem

- Design and interface an MH Series IR sensor with an ESP32 microcontroller to **detect the presence of objects and print the output as analog output.** The objective is to develop a system that can accurately sense the presence of objects within its proximity using the IR sensor and communicate this information to the ESP32, which will then display the result as a analog output and **lights a LED and prints a notification in serial monitor once an object detected.**

Schematic



Code

```
int IRSensor = 35; // connect IR sensor module to ESsp32 pin 35
int LED = 17; // connect LED to ESP32 pin 17

void setup(){
    Serial.begin(115200); // Intiate Serial at 115200 Baud Rate.
    pinMode(IRSensor, INPUT); // IR Sensor pin INPUT
    pinMode(LED, OUTPUT); // LED Pin Output
}
```

Code

```
void loop(){
  int sensorStatus = analogRead(IRSensor); // Set the GPIO as Input
  if (sensorStatus > 1000) // Check if the pin high or not
  {
    // if the pin is high turn off the onboard Led
    digitalWrite(LED, LOW); // LED LOW
    delay(500);
    Serial.println(sensorStatus);
    Serial.println("Motion Detected!");
  }
  // print Motion Detected! on the serial monitor window
}
```

Code

```
    else {  
        //else turn on the onboard LED  
        digitalWrite(LED, HIGH); // LED High  
        delay(500);  
        Serial.println(sensorStatus);  
        Serial.println("Motion Ended!");  
        // print Motion Ended! on the serial monitor window  
    }  
}
```



Actuators Interfacing using ESP32

Internet of Things: Theory and Applications

What are Actuators?

- Actuators are devices or components that **convert electrical, mechanical, or thermal energy into physical motion or action.**
- They are an essential part of **control systems** and are commonly used in various applications to produce a desired effect or response.
- Actuators are responsible for **bringing about a change in the physical world** based on input signals received from sensors or a control system.

Actuating Process

- The actuating process refers to the action of initiating a physical movement or response in a system or device by using actuators.
- Actuators are components that convert various forms of energy, such as electrical, mechanical, or thermal, into mechanical motion or action.
- The actuating process typically involves the following steps:
- Input Signal: The actuating process starts with an input signal that serves as a command or control to activate the actuator. This signal can come from a sensor, a control system, a human operator, or any other source.
- Energy Conversion: Once the input signal is received, the actuator converts the provided energy into mechanical motion or action. The type of energy used depends on the type of actuator, such as electrical energy for motors, compressed air for pneumatic actuators, or pressurized fluid for hydraulic actuators.

Actuating Process

- **Physical Motion**: As the actuator receives and converts the energy, it initiates physical motion or action. For example, an electrical motor may rotate its shaft, a solenoid may extend or retract its plunger, or a hydraulic actuator may generate linear motion in a piston.
- **System Response**: The physical motion or action generated by the actuator leads to a response in the system. This response could be moving a mechanical part, opening or closing a valve, adjusting a position, or performing any other desired function.
- **Feedback (Optional)**: In some cases, the actuating process may involve feedback mechanisms. Feedback allows the system to sense its output or performance and adjust the actuator's action accordingly. This feedback loop ensures precise control and stability in the system.
- **Completion**: The actuating process is complete when the desired physical motion or action is achieved, and the system or device performs its intended function.

Characteristics of Actuators

- Actuators are devices that convert various forms of energy into mechanical motion to control or move mechanisms, systems, or processes. The characteristics of actuators vary based on their type and application, but some common characteristics include:
- **Type:** Actuators come in various types, such as electric, hydraulic, pneumatic, piezoelectric, and magnetic, each with its own unique working principles and advantages.
- **Output Motion:** Actuators produce linear or rotary motion, depending on their design and application. Linear actuators move in a straight line, while rotary actuators rotate around an axis.
- **Force Output:** Actuators exert force to perform mechanical work. The force output can range from small forces in micro-actuators to high forces in heavy-duty industrial actuators.
- **Speed:** Actuators can operate at different speeds, from slow and precise movements in precision applications to rapid movements in high-speed applications.
- **Precision and Accuracy:** Some actuators, like servo motors and piezoelectric actuators, offer high precision and accuracy in controlling motion, making them suitable for applications requiring precise positioning.
- **Control:** Actuators can be controlled through various methods, such as electrical signals, hydraulic pressure, or pneumatic pressure. This control allows for precise and flexible operation.

Characteristics of Actuators

- **Energy Efficiency:** Actuators vary in their energy efficiency, with some designs consuming minimal energy to perform their tasks, while others may require significant power input.
- **Size and Compactness:** Actuators are available in various sizes and forms, from compact micro-actuators to large industrial actuators.
- **Durability and Reliability:** Industrial actuators need to withstand harsh environments and continuous operation, requiring durability and reliability in their design.
- **Noise and Vibrations:** Actuators can generate noise and vibrations during operation, which may be critical considerations in certain applications.
- **Integration and Interfacing:** Actuators need to be compatible and easily integrated into the overall system, and they may require specific interfaces for control and communication.
- **Cost:** The cost of actuators can vary significantly based on their type, performance, and application, with some high-performance actuators being more expensive than standard ones.
- **Environmental Considerations:** Actuators used in outdoor or hazardous environments may need to be designed to withstand exposure to dust, moisture, temperature variations, and corrosive substances.
- **Lifetime and Maintenance:** Actuators should have a long service life and may require regular maintenance to ensure optimal performance.

Actuators Errors

- Actuator errors refer to deviations or inaccuracies that can occur in the output motion or performance of actuators. These errors can affect the actuator's ability to precisely control the motion or force it produces. Some common actuator errors include:
- **Backlash:** Backlash is the amount of free play or clearance present in mechanical components of the actuator, such as gears or linkages. It can result in lost motion, where the actuator does not respond immediately to control signals, leading to inaccuracies in positioning.
- **Hysteresis:** Hysteresis occurs when the actuator's output motion depends not only on the current input signal but also on its past history. It can lead to non-linear responses and inconsistencies when changing the input direction.
- **Deadband:** Deadband is a range of input signals for which the actuator does not respond or produce any output motion. This can cause inaccuracies near the actuator's setpoint and result in limited control resolution.

Actuators Errors

- **Nonlinearity:** Actuator nonlinearity refers to deviations from an ideal linear response to input signals. Nonlinearity can lead to distortions in the output motion, making it challenging to achieve precise and accurate positioning.
- **Friction and Stiction:** Friction is the resistance encountered by moving parts of the actuator, while stiction is the static friction that needs to be overcome before motion can start. These can introduce delays and fluctuations in the actuator's response.
- **Temperature Sensitivity:** Actuators may be sensitive to temperature variations, leading to changes in their performance characteristics. This can result in variations in output motion under different operating temperatures.
- **Mechanical Wear:** Over time, mechanical components in the actuator may wear out, leading to increased play, reduced precision, and decreased overall performance.
- **Electromagnetic Interference:** Electromagnetic interference from external sources can affect the actuator's control electronics, leading to erratic behavior or inaccuracies in its motion.

Core Classification of Actuators

- Actuators can be classified into several core categories based on the type of energy they use to produce mechanical motion or action. The main classifications of actuators are:
- **Electrical Actuators:** These actuators use electrical energy to generate mechanical motion. They are widely used in various applications and can be further classified into:
 - DC Motors: Convert electrical energy into rotational motion.
 - Servo Motors: Precise motors with position and velocity control.
 - Stepper Motors: Move in discrete steps, ideal for precise positioning.
 - Solenoids: Linear actuators that use electromagnetism to move a plunger.
- **Pneumatic Actuators:** These actuators use compressed air or gases to generate motion. Common types include:
 - Pneumatic Cylinders: Linear actuators with controlled motion based on air pressure.
 - Pneumatic Valves: Control the flow of air or gas in pneumatic systems.
- **Hydraulic Actuators:** These actuators use pressurized fluid, typically hydraulic oil, to generate motion. Types include:
 - Hydraulic Cylinders: Linear actuators with controlled motion based on hydraulic pressure.
 - Hydraulic Motors: Convert hydraulic pressure into rotational motion.

Core Classification of Actuators

- **Mechanical Actuators:** These actuators directly convert mechanical energy into motion. Examples include:
 - Levers and Linkages: Simple mechanical systems for converting input motion to output motion.
 - Gears and Gearboxes: Used to change the speed and direction of motion.
- **Thermal Actuators:** These actuators use heat energy to produce motion. Types include:
 - Bimetallic Actuators: Use differential expansion of two metals to generate motion.
 - Shape Memory Alloys (SMAs): Change shape based on temperature changes.
- **Magnetic Actuators:** These actuators use magnetic fields to produce motion. Examples include:
 - Electromagnetic Actuators: Generate motion based on electromagnetic forces.
- **Piezoelectric Actuators:** These actuators use the piezoelectric effect to generate motion when subjected to an electric field.
- **Shape Memory Actuators:** These actuators change shape based on external stimuli like heat or electricity.

Common Types of Electrical Actuators

- Electric actuators are devices that convert electrical energy into mechanical motion. There are several types of electric actuators, each with its own unique characteristics and applications. Here are the main types of electric actuators:
- **DC Motors:** Direct Current (DC) motors are widely used in various applications due to their simplicity and efficiency. They consist of a rotor and a stator, and when an electric current is applied to the coils in the stator, it creates a magnetic field that causes the rotor to rotate. DC motors are available in various configurations, including brushed and brushless DC motors.
- **Servo Motors:** Servo motors are a type of DC motor that offers precise control of angular position, velocity, and acceleration. They are commonly used in robotics, CNC machines, and other applications where precise motion control is required. Servo motors include feedback mechanisms, such as encoders, to provide accurate positioning.
- **Stepper Motors:** Stepper motors are special types of DC motors that move in discrete steps or increments. They do not require feedback for position control and are commonly used in applications that require precise positioning, such as 3D printers, CNC machines, and robotics.

Common Types of Electrical Actuators

- **Linear Actuators:** Linear actuators convert rotary motion into linear motion. They are used when linear motion is required, and they come in various types, including ball screw actuators, lead screw actuators, and linear motors. Linear actuators find applications in robotics, medical devices, and industrial automation.
- **Solenoids:** Solenoids are electromagnetic devices that generate linear motion when an electrical current is applied. They consist of a coil of wire wrapped around a ferromagnetic core. When the coil is energized, it creates a magnetic field that attracts the core, causing linear motion. Solenoids are used in applications such as door locks, valves, and relays.
- **Voice Coils:** Voice coil actuators are electromagnetic actuators that produce linear motion. They consist of a coil of wire placed in a magnetic field. When an electrical current is applied to the coil, it generates a force that moves the coil in the magnetic field, producing linear motion. Voice coil actuators are commonly used in audio devices, autofocus systems, and vibration control.
- **Piezoelectric Actuators:** Piezoelectric actuators use the piezoelectric effect to produce motion. When an electric field is applied to certain materials, such as piezoelectric crystals, they change shape. Piezoelectric actuators offer precise control and are used in nanopositioning, microelectromechanical systems (MEMS), and high-precision applications.
- **Electromagnetic Actuators:** Electromagnetic actuators use electromagnetic forces to produce motion. They can take various forms, including solenoids, voice coils, and linear motors. These actuators find applications in various industries, including automotive, aerospace, and robotics.

Motors as Actuators

- Motors are essential components used as actuators in various systems to convert electrical energy into mechanical motion. They play a crucial role in a wide range of applications, from industrial automation to robotics and automotive systems. Let's delve into the features and characteristics of motors as actuators:
- **Features:**
- **Force:** Motors can generate force to move objects or perform tasks. The force output depends on the motor's design, size, and power rating.
- **Speed:** Motors can operate at different speeds, ranging from slow to high-speed rotations, depending on the application's requirements.
- **Torque:** Torque is the rotational force generated by the motor. It determines the motor's ability to overcome resistance and drive mechanical loads.
- **Power:** Motors have power ratings that define the amount of energy they can convert into mechanical work. Power is a crucial factor in determining the motor's performance.
- **Efficiency:** Motor efficiency is a measure of how well it can convert electrical energy into mechanical energy without significant losses. Higher efficiency motors are desirable for energy-saving applications.

Motors Types

- **DC Motors:** These motors operate using direct current (DC) and are widely used in various applications. They come in brushed and brushless variants, each with its advantages. Brushed DC motors are simple and cost-effective, while brushless DC motors offer higher efficiency and longer lifespan.
- **AC Motors:** Alternating current (AC) motors are commonly used for high-power applications and are available in various types, including induction motors, synchronous motors, and permanent magnet motors.
- **Servo Motors:** Servo motors are precise and controlled motors used in applications requiring accurate positioning and speed control. They are often used in robotics, CNC machines, and automation systems.
- **Stepper Motors:** Stepper motors move in discrete steps, making them suitable for precise motion control. They find applications in 3D printers, scanners, and other position-sensitive systems.
- **Linear Motors:** Linear motors provide linear motion instead of rotational motion. They are used in applications where direct linear movement is required, such as high-speed transportation systems.

About Servo Motor

- **Overview:** Servo motors are a type of rotary actuator widely used in various applications that require precise and controlled angular motion. They are favored for their accuracy, reliability, and ability to hold a specific position without drifting. Servo motors find applications in robotics, industrial automation, aerospace, RC (remote control) vehicles, camera gimbals, and more.
- **Working Principle:** The basic working principle of a servo motor involves a closed-loop control system. It consists of three main components:
 - **1) DC Motor:** At the core of a servo motor, there is a DC motor, usually a brushed or brushless type. The motor's rotation generates torque, allowing the output shaft to move.
 - **2) Position Feedback Device:** A position feedback device, such as a potentiometer or an optical encoder, is mounted on the output shaft of the motor. This device continuously measures the shaft's angular position and provides feedback to the control system.
 - **3) Control System:** The control system is responsible for comparing the desired position (setpoint) with the actual position (feedback) obtained from the position feedback device. If there is a difference between the two, the control system generates an error signal, which is used to adjust the motor's position and bring it closer to the setpoint.

About Servo Motor

Closed-Loop Control: The closed-loop control system ensures that the servo motor maintains its position accurately. The process involves the following steps:

- The user or system specifies the desired position or angle for the servo motor to reach (setpoint).
- The control system compares the setpoint with the actual position obtained from the position feedback device.
- If there is a difference between the setpoint and the feedback, the control system calculates the error signal.
- The error signal is used to adjust the motor's operation, and the motor moves to reduce the error and align with the setpoint.
- Once the motor's actual position matches the setpoint within a specified tolerance, the control system stops adjusting, and the servo motor holds its position.

About Servo Motor

Applications:

- Servo motors have a wide range of applications due to their precise control and position holding capabilities. Some common applications include:
- **Robotics:** Servo motors are extensively used in robotic arms and joints to achieve precise and controlled movements.
- **Industrial Automation:** They are used in CNC machines, conveyor systems, and other automation applications.
- **Camera Gimbals:** Servo motors stabilize cameras and camcorders, ensuring smooth and stable footage.
- **RC Vehicles:** Servo motors control steering and throttle in radio-controlled cars, boats, and airplanes.
- **Aerospace:** They are employed in aircraft control surfaces and flaps.
- **Medical Devices:** Servo motors are used in surgical robots and precision medical equipment.
- **Electronics:** Servo motors are found in 3D printers and RC hobby projects.

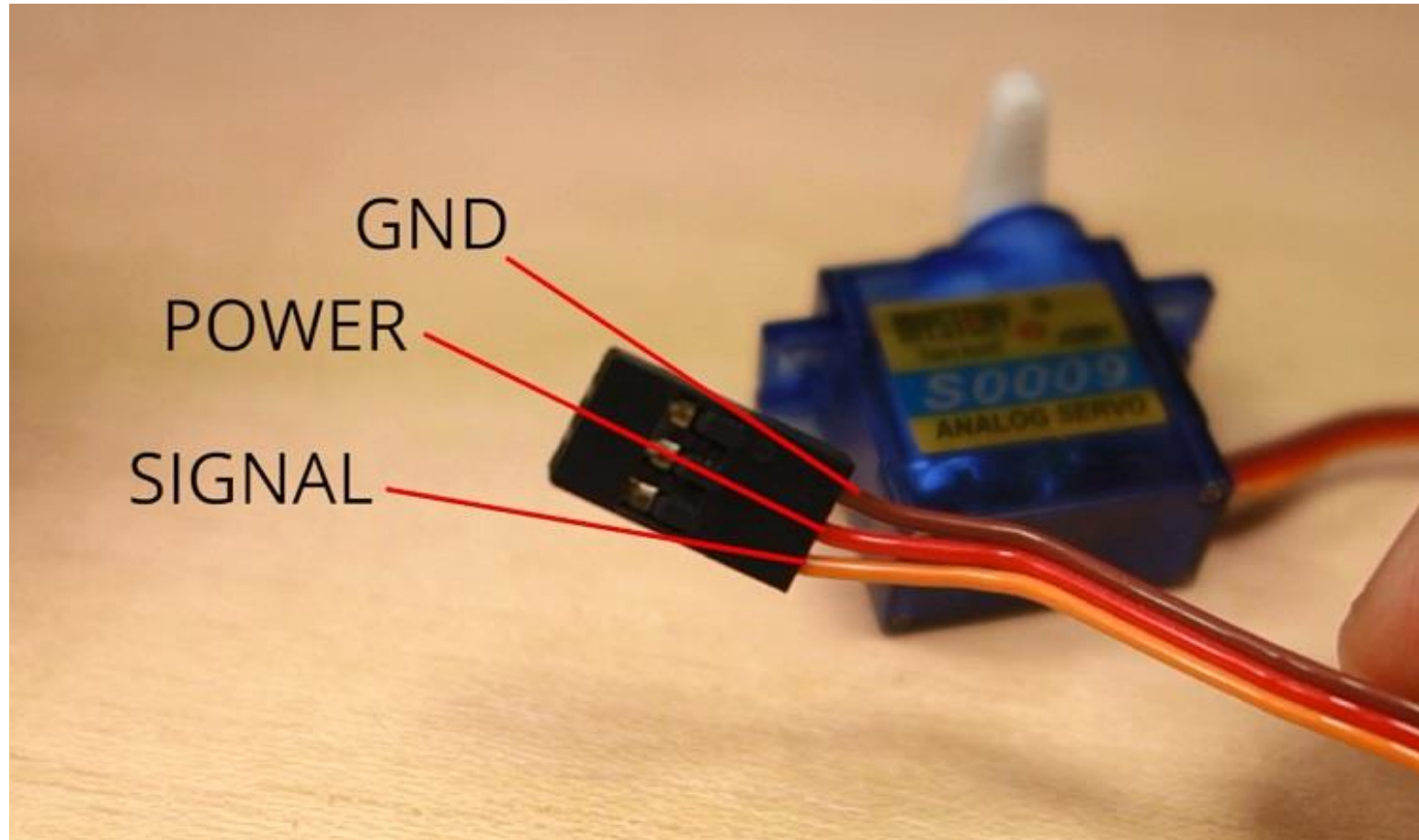
Types of Servo Motors

- There are several types of servo motors based on their construction, working principles, and applications. Some common types of servo motors include:
- **DC Servo Motor:** These motors use a DC power supply and have a brushed or brushless DC motor at their core. They are widely used in industrial automation, robotics, and motion control systems.
- **AC Servo Motor:** AC servo motors use an AC power supply and are popular in applications requiring high torque and precision, such as CNC machines and robotics.
- **Linear Servo Motor:** Instead of producing rotary motion, linear servo motors provide linear motion. They are commonly used in high-precision positioning systems and applications where traditional mechanical actuators may not be suitable.
- **Continuous Rotation Servo Motor:** Unlike standard servo motors, continuous rotation servo motors can rotate continuously in either direction. They are often used in robotics and other applications requiring continuous motion.

Types of Servo Motors

- **Mini Servo Motor:** Mini servo motors are compact and lightweight, making them ideal for small-scale projects, RC vehicles, and micro-robotics.
- **Digital Servo Motor:** Digital servo motors incorporate advanced electronics and microcontrollers, offering higher accuracy, faster response times, and more precise control compared to analog servo motors.
- **Analog Servo Motor:** Analog servo motors are commonly used in hobby projects and simple applications. They have a basic control circuit and respond to analog control signals to set their position.
- **High Torque Servo Motor:** High torque servo motors provide greater rotational force, making them suitable for applications requiring heavy loads or moving parts.
- **Low-Cost Servo Motor:** These servo motors are budget-friendly and find applications in educational projects and DIY applications.
- **Industrial Servo Motor:** Industrial servo motors are designed for rugged environments and heavy-duty applications, such as manufacturing and automation.
- **Hobby Servo Motor:** Hobby servo motors are commonly used in RC hobbies, toy applications, and small-scale projects.

SG90 DC Servo Motor



SG90 DC Servo Motor

- **Working Principle:** The SG90 servo motor operates on the principle of using a control signal to position its shaft at a specific angle. It consists of a small DC motor, a potentiometer (pot) for feedback, and a control circuit. The control circuit interprets the incoming control signal (PWM signal) and adjusts the motor's position accordingly by comparing the desired angle with the actual angle detected by the potentiometer. As a result, the motor shaft rotates to the desired position, allowing precise control over the angle.

SG90 DC Servo Motor

- **Math Formula:** The SG90 servo motor is controlled using a Pulse Width Modulation (PWM) signal. The PWM signal consists of a series of pulses with varying pulse widths. The duration of the pulse determines the position of the servo motor. The formula to calculate the pulse width for a specific angle (θ) is:
- **Pulse Width (in microseconds) = $(\theta / \text{Angle Range}) * (\text{Maximum Pulse Width} - \text{Minimum Pulse Width}) + \text{Minimum Pulse Width}$**

where:

- θ is the desired angle of the servo motor (0 to the maximum angle supported by the servo).
- Angle Range is the total range of angles supported by the servo (usually 180 degrees for most servos).
- Maximum Pulse Width is the pulse width corresponding to the maximum angle.
- Minimum Pulse Width is the pulse width corresponding to the minimum angle.

SG90 DC Servo Motor

- **Connection to ESP32:** The SG90 servo motor can be easily connected to an ESP32 microcontroller using three wires: power (VCC), ground (GND), and signal (control). The power and ground wires are connected to the appropriate pins on the ESP32 to provide the motor with the necessary voltage and ground connection. The control wire is connected to a PWM-capable pin on the ESP32 to send the control signal to the servo.

SG90 DC Servo Motor

- **Applications in IoT:** The SG90 servo motor is widely used in IoT projects for its precise positioning and control capabilities. Some common applications in IoT include:
- **Robotic Arms and Manipulators:** SG90 servos are used in robotic arms and manipulators to control the movement and positioning of various components.
- **Surveillance Cameras:** Servo motors can be used to pan and tilt surveillance cameras to monitor a wider area.
- **Smart Home Automation:** Servos can be used in smart home automation projects to control doors, windows, blinds, and other movable components.
- **Solar Tracking Systems:** In solar tracking systems, servos can be used to adjust the angle of solar panels to maximize their exposure to sunlight.
- **Remote Controlled Vehicles:** Servo motors are commonly used in RC cars, planes, and boats for steering and other control functions.
- **Industrial Automation:** In industrial applications, servos are used in conveyor systems, CNC machines, and other automated processes that require precise motion control.

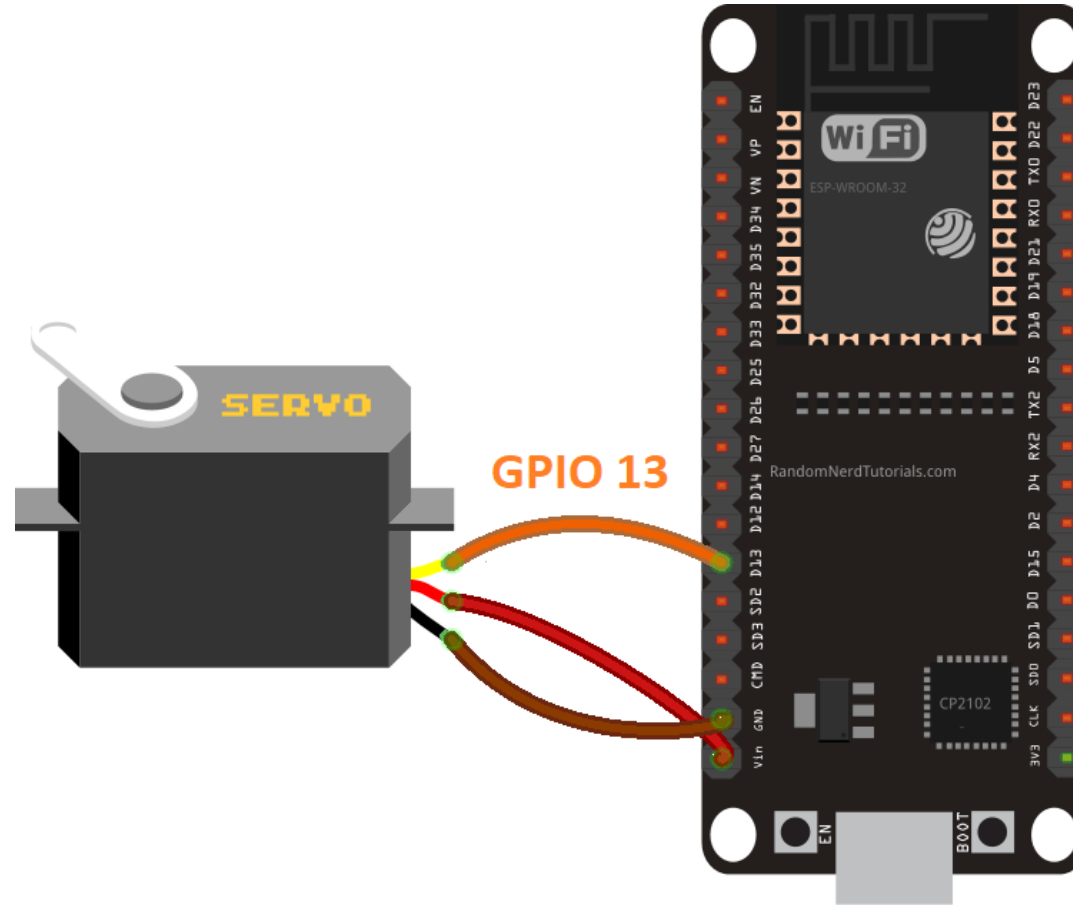
Types of SG90 Servo Motor based on Angle

- **Servo Motors with 180° Maximum Rotation:** These servo motors have a limited range of motion and can rotate up to 180 degrees. They are commonly referred to as "180° servo motors" or "limited rotation servo motors." These servos are suitable for applications where precise control within a limited range of motion is required, such as in robotics, RC cars, and small-scale mechanical projects.
- **Servo Motors with 360° Continuous Rotation:** Unlike standard servo motors, these servos can rotate continuously in either direction, offering a full 360-degree range of motion. They are commonly known as "360° servo motors" or "continuous rotation servo motors." Continuous rotation servos are commonly used in applications where continuous motion is needed, such as robot wheels, pan-tilt cameras, and motion platforms.

Example

- Get SG90 DC 180 degrees servo motor and let it go to 0 degrees and wait a second then to 90 degrees and wait for second then to 180 degrees and wait a second.

Schematic



Code

```
#include <Servo.h>

Servo servoMotor; // Create a servo object to control the SG90 servo motor

void setup() {
    servoMotor.attach(13); // Attach the servo to digital pin 9
}

void loop() {
    // Move the servo to the 0-degree position (minimum angle)
    servoMotor.write(0);
    delay(1000); // Wait for 1 second

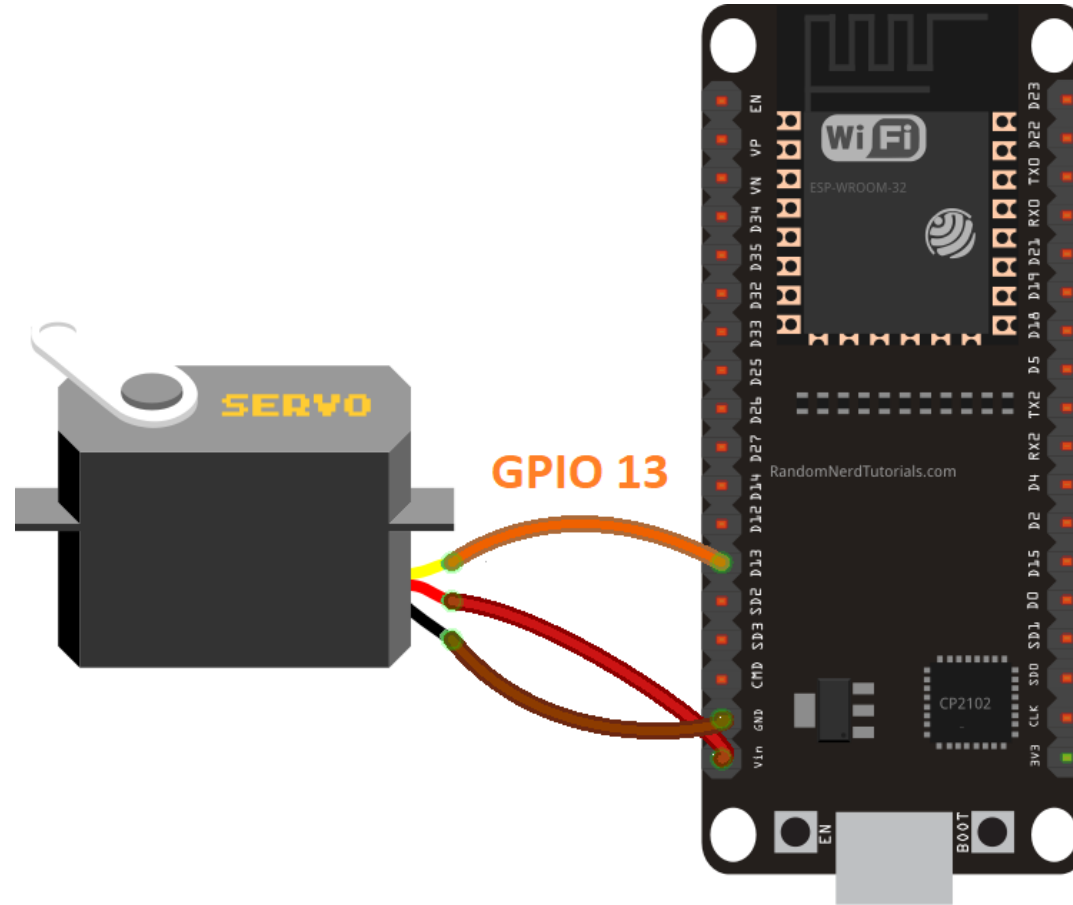
    // Move the servo to the 90-degree position (mid-angle)
    servoMotor.write(90);
    delay(1000); // Wait for 1 second

    // Move the servo to the 180-degree position (maximum angle)
    servoMotor.write(180);
    delay(1000); // Wait for 1 second
}
```

Practice

- Control the servo motor to be swept from 0 to 180 degrees and vice versa.

Schematic



Code

```
#include <ESP32Servo.h>

Servo myservo;  // create servo object to control a servo
// 16 servo objects can be created on the ESP32

int pos = 0;    // variable to store the servo position
// Recommended PWM GPIO pins on the ESP32 include 2,4,12-19,21-23,25-27,32-33
int servoPin = 13;
```

Code

```
void setup() {  
    Serial.begin(115200);  
    myservo.attach(servoPin); // attaches the servo on pin 18 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  
    }  
}
```



Output Modules Interfacing using ESP32

Internet of Things: Theory and Applications

What are Output Modules?

- Output modules in embedded systems refer to the **hardware or software components responsible for generating and controlling output signals to external devices or systems.** These modules take processed data or information from the embedded system's core processing unit and convert it into a form suitable for external devices to understand or act upon.
- Output modules are essential in various embedded applications where the system needs to **interact with the external environment.** **They can control various output devices such as motors, actuators, displays, LEDs, relays, speakers, and more.** The purpose of output modules is to convey information, trigger actions, or provide feedback to users or other systems.

Types of Output Modules

- **Motor Control Modules:** These modules interface with motors (DC motors, stepper motors, servo motors) and control their speed, direction, and rotation to perform mechanical actions.
- **Display Modules:** Output modules with displays, such as LCDs or OLEDs, are used to present information, status, or data to users or operators.
- **LED Control Modules:** These modules control LEDs to provide visual indicators or notifications.
- **Actuator Control Modules:** Actuators are used to control mechanical elements in a system, such as opening or closing valves, adjusting positions, or actuating switches.
- **Audio Output Modules:** These modules generate sound through speakers or audio output devices, enabling the system to produce audible alerts, alarms, or voice prompts.
- **Communication Modules:** In embedded systems that require communication with external devices or networks, output modules are used to control data transmission and reception.
- **Haptic Feedback Modules:** These modules generate tactile feedback through vibration or touch, enhancing the user's interaction with the system.

About Display Output Modules

- Display output modules are **electronic components or devices used to present information or data in a human-readable form to users. They are an essential part of various electronic systems and devices where visual feedback or data representation is required.** Display output modules are designed to transform digital or analog data into visual or graphical representations, making it easier for users to interpret and interact with the information.
- Here are some common types of display output modules:
- **Liquid Crystal Display (LCD):**
 - LCDs are widely used display output modules due to their compact size, low power consumption, and clear visibility in various lighting conditions.
 - They consist of a matrix of liquid crystal cells that change their orientation in response to an applied electric field, thus altering the light transmission and displaying characters or graphics.
 - LCDs are commonly used in calculators, digital watches, home appliances, industrial equipment, and various consumer electronics.

About Display Output Modules

- **Light Emitting Diode (LED) Display:**

- LED displays utilize an array of light-emitting diodes to produce characters, numbers, or graphical patterns.
- They are energy-efficient, offer high brightness, and are commonly used in indicators, signage, scoreboards, and alphanumeric displays.

- **Quantum Dot Light Emitting Diode (QLED) Display:**

- QLED displays are a variation of LED LCD technology that incorporates quantum dots to enhance color accuracy, brightness, and overall image quality.
- Quantum dots improve the color reproduction of the display, resulting in more vibrant and lifelike images.
- QLED displays are commonly used in high-end televisions and monitors, providing a wide color gamut and high peak brightness levels.

About Display Output Modules

- **Seven-Segment Display:**

- Seven-segment displays are a type of LED display with seven individual segments that can display numeric digits from 0 to 9 and some characters (A-F).
- They are often used in digital clocks, timers, measuring instruments, and other devices where numerical data representation is essential.

- **Organic Light Emitting Diode (OLED) Display:**

- OLED displays use organic compounds that emit light when an electric current is applied, allowing for pixel-level control of brightness.
- OLEDs offer "true" blacks, infinite contrast ratios, and wide viewing angles, making them ideal for smartphones, televisions, and wearable devices.

About Display Output Modules

- **Dot Matrix Display:**

- Dot matrix displays consist of an array of individual LEDs or pixels that can be controlled individually to display text, symbols, and graphics.
- They provide more flexibility in data representation and are commonly used in information boards, scrolling tickers, and graphic interfaces.

- **Thin Film Transistor (TFT) Display:**

- TFT displays are a type of LCD that uses thin-film transistor technology for improved image quality and faster response times.
- They are commonly used in smartphones, tablets, computer monitors, and high-end display applications.

Getting an Display Output Process

- The process of getting output from display output modules involves several steps, depending on the type of display and the technology used. Generally, the process can be broken down into the following stages:
- **Data Input:**
 - The first step is to provide the data that needs to be displayed. This data can be in digital or analog form, depending on the display and the input interface available.
 - For example, in the case of an LCD, digital data, such as characters or graphics, is typically sent to the display controller.
- **Data Conversion and Processing:**
 - In some cases, data may need to be converted or processed before being sent to the display output module.
 - For example, in a 7-segment LED display, the digital data representing a number may need to be converted into a binary format to activate the appropriate segments for each digit.

Getting an Display Output Process

- **Display Controller:**

- Many display output modules, especially those with advanced features, use a display controller to handle the data input, processing, and communication with the display itself.
- The display controller can manage tasks like refreshing the display at regular intervals to prevent flickering, adjusting brightness levels, and interfacing with other components.

- **Communication Interface:**

- Depending on the display technology, **there may be different communication interfaces** used to transmit data from the microcontroller or system to the display output module.
- Common communication interfaces include Serial Peripheral Interface (SPI), Inter-Integrated Circuit (I2C), Parallel Interface, or Universal Asynchronous Receiver/Transmitter (UART).

- **Pixel or Segment Activation:**

- For displays like LCDs or OLEDs, the display controller activates individual pixels or segments based on the provided data to create the desired image or characters.
- In LED displays, each segment or pixel is controlled individually to display the required information.

Getting an Display Output Process

- **Light Emission:**

- In LED, QLED, and OLED displays, the data sent to the individual pixels or segments controls the light emission, resulting in the desired colors and brightness levels.
- For LCDs, the liquid crystal cells align or twist in response to the electric field, allowing or blocking the passage of light from a backlight source, resulting in the display of characters or images.

- **Refreshing (if applicable):**

- Some display output modules, such as LCDs, require periodic refreshing to maintain the displayed information. The display controller ensures that the information is continuously updated to prevent any distortion or fading.

- **User Interaction:**

- In some cases, display output modules may interact with users through touch-sensitive interfaces (e.g., capacitive touchscreens) or physical buttons.

Characteristics of Display Output Modules

- Display output modules have various characteristics that define their performance, capabilities, and suitability for specific applications. Here are some key characteristics of display output modules:
- **Display Technology:**
 - Display output modules come in different technologies, such as LCD, LED, OLED, QLED, TFT, and others.
 - Each technology has its advantages and limitations in terms of color accuracy, contrast ratio, viewing angles, response time, and power consumption.
- **Resolution:**
 - Resolution refers to the number of pixels or dots that a display can represent, usually expressed as width x height (e.g., 1920x1080 pixels for Full HD).
 - Higher resolution displays offer sharper and more detailed images.
- **Color Gamut:**
 - Color gamut defines the range of colors that a display can reproduce within a specific color space, such as sRGB or DCI-P3.
 - Wider color gamut displays can reproduce more vibrant and accurate colors.

Characteristics of Display Output Modules

- **Brightness:**

- Brightness measures the intensity of light emitted by the display in nits or cd/m^2 .
- Higher brightness levels are essential for readability in well-lit environments.

- **Contrast Ratio:**

- Contrast ratio indicates the difference in brightness between the darkest and brightest parts of the display.
- Higher contrast ratios lead to more distinct and lifelike images.

Characteristics of Display Output Modules

- **Viewing Angles:**

- Viewing angles determine the range of angles from which the display remains visible without significant color distortion or dimming.
- Displays with wide viewing angles provide consistent image quality from various perspectives.

- **Refresh Rate:**

- Refresh rate indicates how many times per second the display updates its content.
- Higher refresh rates (e.g., 60Hz, 120Hz, 240Hz) result in smoother motion, crucial for gaming and fast-paced content.

- **Response Time:**

- Response time measures how quickly pixels can change from one color to another.
- Lower response times reduce motion blur and ghosting in fast-moving images.

Characteristics of Display Output Modules

- **Power Consumption:**

- Power consumption affects the device's overall energy efficiency and battery life, making it an essential consideration for portable devices.

- **Size and Form Factor:**

- The physical dimensions and form factor of the display module determine its suitability for specific applications.
- Displays come in various sizes, from small screens for wearable devices to large panels for televisions.

- **Touch Sensitivity (for some displays):**

- Touch-sensitive displays allow users to interact directly with the screen through taps, swipes, and gestures.

Characteristics of Display Output Modules

- **Durability and Longevity:**

- Some displays are designed to withstand harsh environmental conditions, making them suitable for outdoor or industrial applications.
- Display longevity is important, especially for devices that are intended to be in use for extended periods.

- **Connectivity and Interface:**

- The display module's interface and connectivity options (e.g., HDMI, VGA, I2C, SPI) determine how it can be connected and controlled by the host system.

- **Price and Availability:**

- The cost and availability of display output modules influence the selection for a particular project or product.

Errors in Display Output Modules

It seems like there might be a typo in your question. If you are referring to errors related to display output modules, I'll assume you mean common issues or problems that can occur with display output modules. Here are some common errors or challenges that may arise with display output modules:

- **Dead Pixels:** Some display technologies, like LCD and OLED, may suffer from dead pixels, where individual pixels remain permanently black or white. Dead pixels can be distracting and affect image quality.
- **Backlight Bleeding:** LCD displays with LED backlighting may exhibit backlight bleeding, which causes uneven illumination along the edges of the screen, especially noticeable in dark scenes.
- **Image Burn-In:** OLED displays are susceptible to image burn-in, where persistent static images or logos can cause ghost images to remain visible even after the content changes.
- **Color Inconsistency:** Lower-quality displays may suffer from color inconsistency, where different parts of the screen display colors slightly differently.
- **Screen Flickering:** Flickering can occur due to incompatible refresh rates or other issues, leading to eye strain and discomfort.

Errors in Display Output Modules

- **Viewing Angle Issues:** Some displays experience color shifts or fading when viewed from certain angles, limiting the range of optimal viewing positions.
- **Response Time Artifacts:** Displays with slow response times can produce motion blur or ghosting in fast-paced content.
- **Screen Tearing:** Screen tearing happens when the display's refresh rate does not synchronize with the frame rate of the content being displayed, resulting in visual artifacts.
- **Connectivity Problems:** Issues with connection interfaces, such as loose cables or incompatible connectors, can cause display problems.

Errors in Display Output Modules

- **Display Driver or Controller Errors:** Problems with the display driver or controller can lead to incorrect rendering or distorted images.
- **Display Signal Interference:** External factors, such as electromagnetic interference, can disrupt the display signal and cause visual glitches.
- **Power Supply Problems:** Insufficient or unstable power supply to the display module may result in flickering or other issues.
- **Touchscreen Inaccuracy:** In touch-sensitive displays, inaccurate touch detection or calibration issues may hinder accurate input.
- **Overheating:** Displays that generate excessive heat can degrade performance or even cause damage over time.

Core Classification of Display Output Modules

- Output display modules can be classified into several core categories based on their display technology and the way they present visual information. The main classifications of output display modules are as follows:
- **LCD (Liquid Crystal Display):**
 - LCDs are one of the most common and widely used display technologies.
 - They use liquid crystal cells that change their orientation in response to an electric field to control the passage of light, creating characters, graphics, and images.
 - LCDs are commonly found in computer monitors, televisions, smartphones, calculators, and various consumer electronics.
- **LED (Light Emitting Diode) Display:**
 - LED displays use an array of light-emitting diodes to produce characters, numbers, or graphical patterns.
 - They are energy-efficient, offer high brightness, and have a longer lifespan compared to other technologies.
 - LED displays are commonly used in digital signage, scoreboards, traffic signs, and indicators.

Core Classification of Display Output Modules

- **OLED (Organic Light Emitting Diode) Display:**

- OLED displays use organic compounds that emit light when an electric current is applied, eliminating the need for a backlight.
- OLEDs offer "true" blacks, infinite contrast ratios, wide viewing angles, and fast response times.
- They are commonly found in smartphones, televisions, wearable devices, and high-end consumer electronics.

- **QLED (Quantum Dot Light Emitting Diode) Display:**

- QLED displays are a variation of LED LCD technology that incorporates quantum dots to enhance color accuracy, brightness, and overall image quality.
- Quantum dots improve the color reproduction of the display, resulting in more vibrant and lifelike images.
- QLED displays are commonly used in high-end televisions and monitors.

Core Classification of Display Output Modules

- **TFT (Thin Film Transistor) Display:**

- TFT displays are a type of LCD that uses thin-film transistor technology for improved image quality and faster response times.
- TFTs are commonly used in computer monitors, tablets, smartphones, and other applications where high-quality graphics are required.

- **E-Ink (Electronic Ink) Display:**

- E-Ink displays use electrophoretic technology to produce characters and images that mimic ink on paper.
- They have high visibility even in bright sunlight, and their low power consumption makes them ideal for e-readers and electronic shelf labels.

Core Classification of Display Output Modules

- **Projection Displays:**

- Projection displays use light beams to project images onto a surface, such as a screen or wall.
- They are commonly used in projectors for presentations, home theaters, and large-scale displays.

- **Touchscreen Displays:**

- Touchscreen displays are integrated with touch-sensitive technology that allows users to interact directly with the screen by touching or swiping.
- Touchscreens are commonly used in smartphones, tablets, ATMs, information kiosks, and interactive displays.

About LCD

Overview of LCD (Liquid Crystal Display):

- LCD stands for Liquid Crystal Display, a widely used display technology known for its compact size, low power consumption, and versatility. LCDs consist of liquid crystal cells sandwiched between two layers of glass or plastic. They are commonly used in various electronic devices, including calculators, digital watches, computer monitors, smartphones, and IoT devices, to display text, numbers, and graphics.

About LCD

Working Principle of LCD:

- The working principle of an LCD is based on the optical properties of liquid crystals. Liquid crystals are molecules that have a unique ability to **align their orientation in response to an electric field**. When an electric current is applied to specific areas of the liquid crystal cells, their alignment changes, affecting the passage of light through the cell.
- LCDs typically consist of **two polarizing filters placed at right angles to each other, with liquid crystal cells in between**. When no electric field is applied, the liquid crystal molecules align themselves in a twisted configuration, preventing light from passing through the second polarizing filter. This state is called "off" or "dark."
- When an **electric field is applied, the liquid crystal molecules straighten, allowing light to pass through the second polarizing filter and producing a visible image**. By controlling the electric field across different areas of the liquid crystal cells, the display can form characters, numbers, and graphics.

About LCD

Applications of LCD in IoT:

LCDs are widely used in IoT devices due to their simplicity, low power consumption, and cost-effectiveness. They provide a user-friendly interface for IoT devices to display real-time data, status information, and user prompts. Common applications of LCD in IoT include:

- **IoT Weather Stations:** Displaying temperature, humidity, and weather conditions.
- **Home Automation:** Displaying control options and status for smart home devices.
- **Health Monitoring:** Displaying vital signs and health-related data.
- **Industrial IoT:** Displaying sensor readings, machine status, and production data.
- **Smart Agriculture:** Displaying soil moisture, temperature, and crop information.

About LCD

There are several types of LCDs based on their construction and functionality. Some common types include:

- **Character LCD:** Used for displaying alphanumeric characters and symbols.
- **Graphical LCD:** Capable of displaying custom graphics and images.
- **TFT LCD:** High-quality color displays with thin-film transistor technology.

Interfacing LCD with ESP32

- An LCD (Liquid Crystal Display) is a common output device used in ESP32 projects to display information in a user-friendly manner. It allows you to show text, numbers, symbols, and even graphics on a screen. Here's some information about using an LCD with Arduino:
- **Components Needed:**
 - ESP32 Dev Kit
 - LCD module (commonly 16x2 or 20x4 character LCD)
 - Potentiometer (for adjusting LCD contrast)
 - Breadboard and jumper wires
- **Working Principle with ESP32:** LCDs work based on the properties of liquid crystals, which can change their orientation when an electric field is applied. The LCD module consists of a grid of tiny segments or pixels that can be activated to display characters and graphics. ESP32 communicates with the LCD module using the parallel or I2C interface to send commands and data for display.

Interfacing LCD with ESP32

- **Library:** To work with the LCD module, you need to use an appropriate Arduino library. Popular libraries for controlling LCDs include "LiquidCrystal" (for parallel interface) and "LiquidCrystal_I2C" (for I2C interface).

Interfacing LCD with ESP32

- **Data pins (D0 to D7):** Connect these pins to the corresponding data pins of the LCD.
- **RS (Register Select) pin:** Connect to the Register Select pin of the LCD to differentiate between command and data.
- **Enable (E) pin:** Connect to the Enable pin of the LCD to enable data transfers.
- **R/W (Read/Write) pin:** Connect to Ground (Low) for write-only mode or Vcc (High) for read/write mode.
- **Vcc and Ground:** Connect to the power supply of the LCD.
- **Backlight Control (optional):** Connect to control the LCD backlight.

Controlling LCD using ESP32

- The 4-bit and 8-bit configurations refer to the number of data lines used to transfer information between a microcontroller (such as an ESP32) and an LCD (Liquid Crystal Display). Both configurations are used to interface microcontrollers with LCDs and have their own advantages and limitations.
- 4-Bit Configuration:
 - In the 4-bit configuration, only four data lines (D4 to D7) are used to transfer data between the microcontroller and the LCD.
 - The other four data lines (D0 to D3) are not connected or are left unutilized.
 - To send an 8-bit data byte to the LCD, it requires two separate 4-bit transfers: first, the higher 4 bits (D4 to D7) are sent, followed by the lower 4 bits (D0 to D3).
 - This configuration reduces the number of required GPIO pins on the microcontroller, which can be beneficial if GPIO pins are limited or needed for other purposes.
 - However, the 4-bit configuration is slightly slower compared to the 8-bit configuration since it requires two transfers to send a complete 8-bit data byte.

Controlling LCD using ESP32

- 8-Bit Configuration:
 - In the 8-bit configuration, all eight data lines (D0 to D7) are used to transfer data between the microcontroller and the LCD.
 - This means that a complete 8-bit data byte can be sent in a single transfer, making it faster than the 4-bit configuration.
 - The 8-bit configuration requires more GPIO pins on the microcontroller compared to the 4-bit configuration.
 - However, it offers better performance and higher data transfer rates compared to the 4-bit configuration.

Controlling LCD using ESP32

- Which Configuration is Better? The choice between the 4-bit and 8-bit configurations depends on the specific requirements and constraints of your project. Here are some factors to consider when deciding which configuration is better:
- GPIO Pin Availability: If GPIO pins on the microcontroller are limited or needed for other purposes, the 4-bit configuration may be preferred as it uses fewer GPIO pins.
- Speed: If fast data transfer is a priority, the 8-bit configuration is better since it can send a complete 8-bit data byte in a single transfer, making it slightly faster than the 4-bit configuration.

Controlling LCD using ESP32

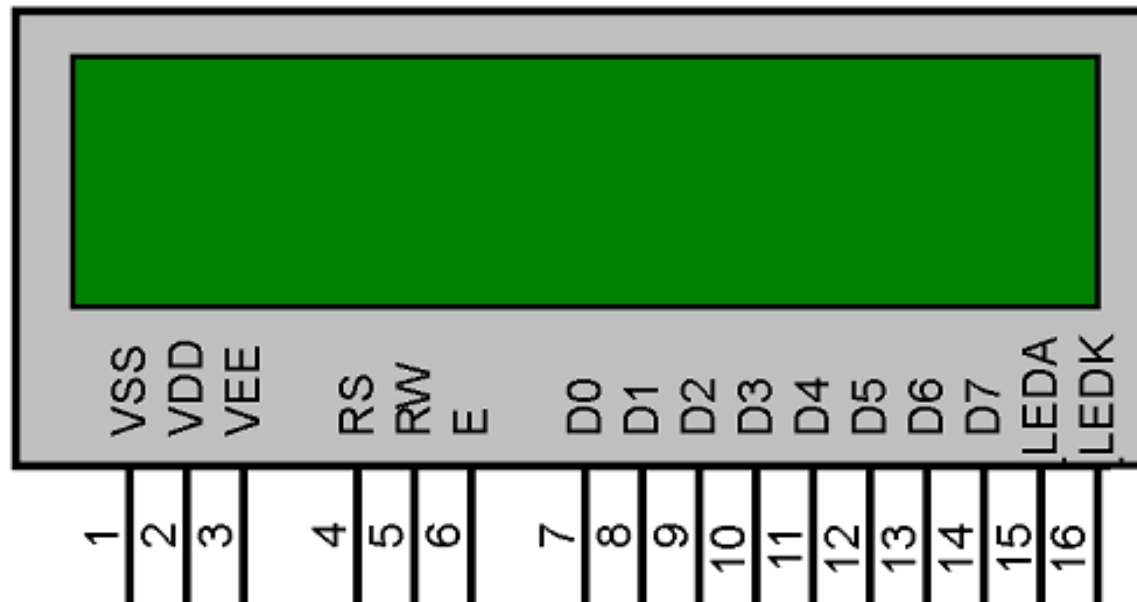
- Complexity: The 4-bit configuration requires additional software logic to split the 8-bit data byte into two separate 4-bit transfers, which may add some complexity to the code.
- Display Update Rate: For applications where the display content needs to be updated frequently, the 8-bit configuration's faster data transfer rate may be more desirable.
- Project Constraints: Consider other project constraints such as power consumption, PCB layout, and ease of implementation when making your decision.
- In general, the 8-bit configuration is commonly used in most projects as it provides better performance and speed, and the ESP32 and similar microcontrollers have sufficient GPIO pins to accommodate the connections. However, if you have strict constraints on GPIO pin availability or need to minimize complexity, the 4-bit configuration can be a viable alternative.

Which Configuration Used?

- We are mainly are to use the 4-bit configuration due to limited GPIOs pins in the LCD.

Interfacing LCD 16*2 with ESP32

- **16×2 LCD** is named so because; it has 16 Columns and 2 Rows.
- All LCD types has 16 pins as the following:



Interfacing LCD 16*2 with ESP32

Pin No.	Pin Name	Pin Type	Pin Description	Pin Connection
Pin 1	Ground	Source Pin	This is a ground pin of LCD	Connected to the ground of the MCU/ Power source
Pin 2	VCC	Source Pin	This is the supply voltage pin of LCD	Connected to the supply pin of Power source
Pin 3	VO/VEE	Control Pin	Adjusts the contrast of the LCD.	Connected to a variable POT that can source 0-5V
Pin 4	Register Select	Control Pin	Toggles between Command/Data Register	Connected to a MCU pin and gets either 0 or 1. 0 -> Command Mode 1-> Data Mode
Pin 5	Read/Write	Control Pin	Toggles the LCD between Read/Write Operation	Connected to a MCU pin and gets either 0 or 1. 0 -> Write Operation 1-> Read Operation

Interfacing LCD 16*2 with ESP32

Pin No.	Pin Name	Pin Type	Pin Description	Pin Connection
Pin 6	Enable	Control Pin	Must be held high to perform Read/Write Operation	Connected to MCU and always held high.
Pin 7-14	Data Bits (0-7)	Data/Command Pin	Pins used to send Command or data to the LCD.	<u>In 4-Wire Mode</u> Only 4 pins (0-3) is connected to MCU <u>In 8-Wire Mode</u> All 8 pins(0-7) are connected to MCU
Pin 15	LED Positive	LED Pin	Normal LED like operation to illuminate the LCD	Connected to +5V
Pin 16	LED Negative	LED Pin	Normal LED like operation to illuminate the LCD connected with GND.	Connected to ground

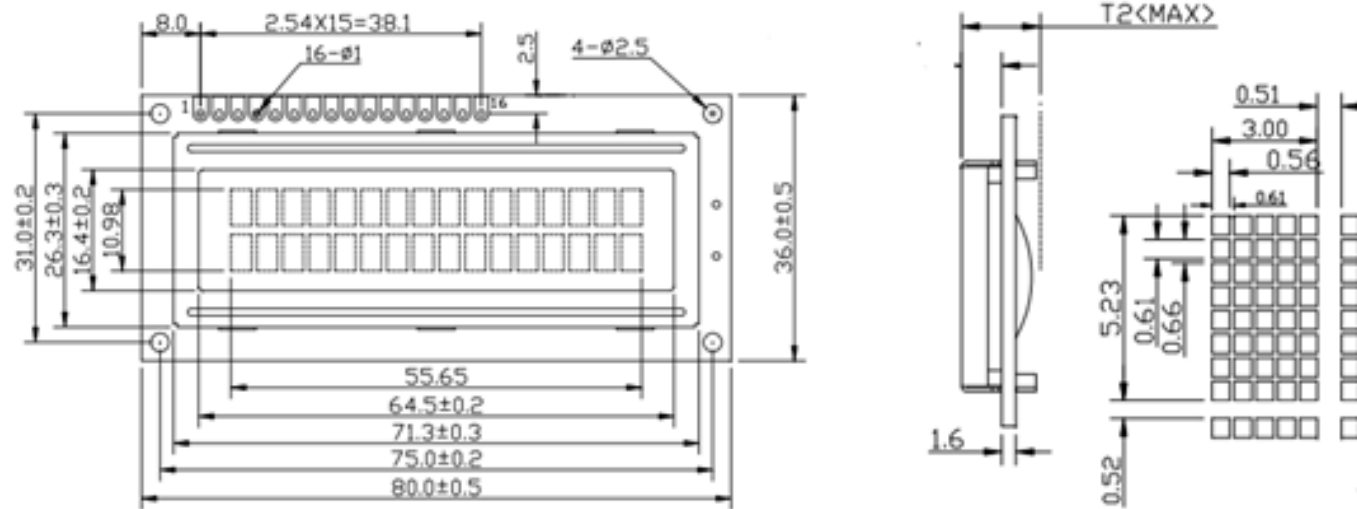
Power & Upload NOTES

- Don't EVER connect to Vin while connecting to the laptop.
- Don't use any two input powers on ESP32 EVER.
- Instead use a Power Bank to power the ESP32 after uploading the code on it.
- While uploading the code, don't connect to any TX, RX, or VIN pins ever.

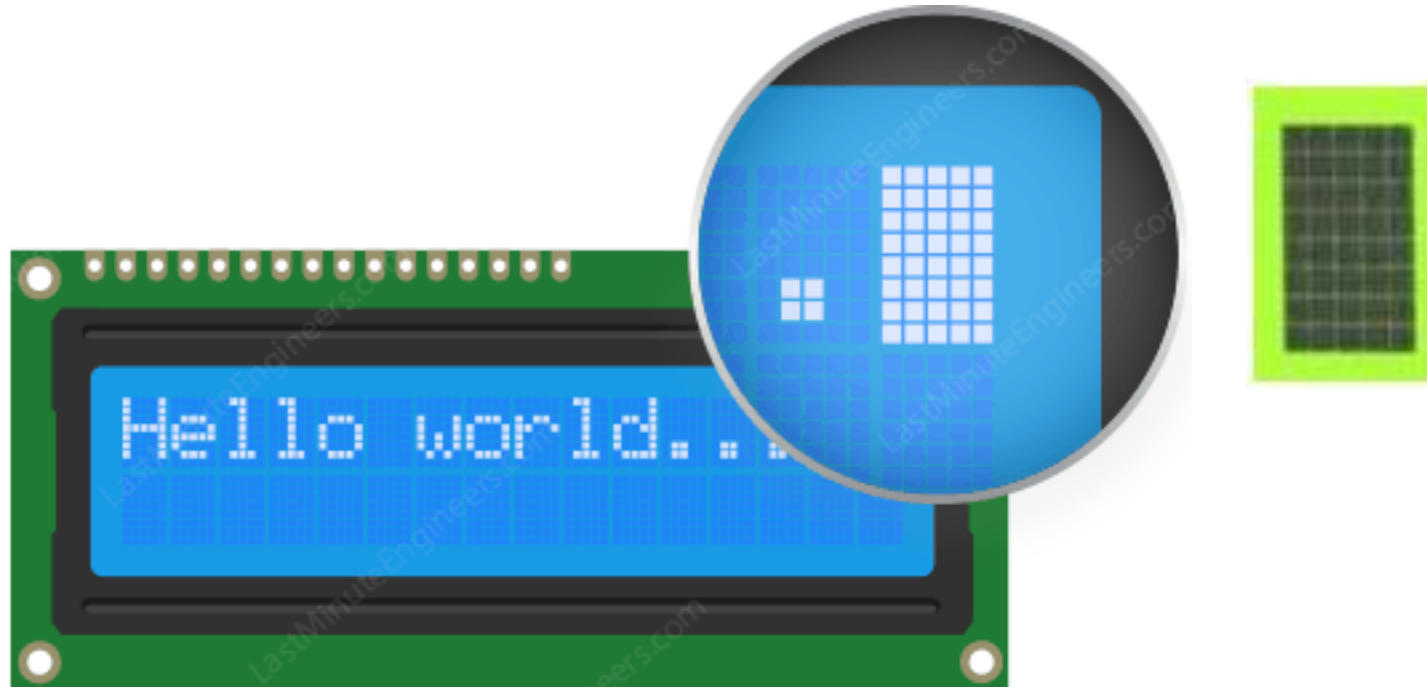
Technical Specs of LCD 16*2

- Operating Voltage is 4.7V to 5.3V
- Current consumption is 1mA without backlight
- Alphanumeric LCD display module, meaning can display alphabets and numbers
- Consists of two rows and each row can print 16 characters.
- Each character is build by a 5×8 pixel box
- Can work on both 8-bit and 4-bit mode
- It can also display any custom generated characters
- Available in Green and Blue Backlight

Technical Specs of LCD 16*2




Technical Specs of LCD 16*2



Custom Character of LCD 16*2

Start selecting pixels



Clear all

```
byte  
Character[8] =  
{  
    0b000000,  
    0b000000,  
    0b01010,  
    0b11111,  
    0b11111,  
    0b01110,  
    0b00100,  
    0b000000  
};
```

Copy this code to your sketch

Custom Character of LCD 16*2

	Custom char					Bitmap					Binary	HEX
	4	3	2	1	0							
0	■	■	■	■	■	0	0	1	0	0	0 0 1 0 0	0x4
1	■	■	■	■	■	0	1	1	1	0	0 1 1 1 0	0xE
2	■	■	■	■	■	0	1	1	1	0	0 1 1 1 0	0xE
3	■	■	■	■	■	0	1	1	1	0	0 1 1 1 0	0xE
4	■	■	■	■	■	1	1	1	1	1	1 1 1 1 1	0x1F
5	■	■	■	■	■	0	0	0	0	0	0 0 0 0 0	0x0
6	■	■	■	■	■	0	0	1	0	0	0 0 1 0 0	0x4
7	■	■	■	■	■	0	0	0	0	0	0 0 0 0 0	0x0

Example

- Print your “IOT Training” on LCD upon parallel interfacing with ESP32 Dev Kit.

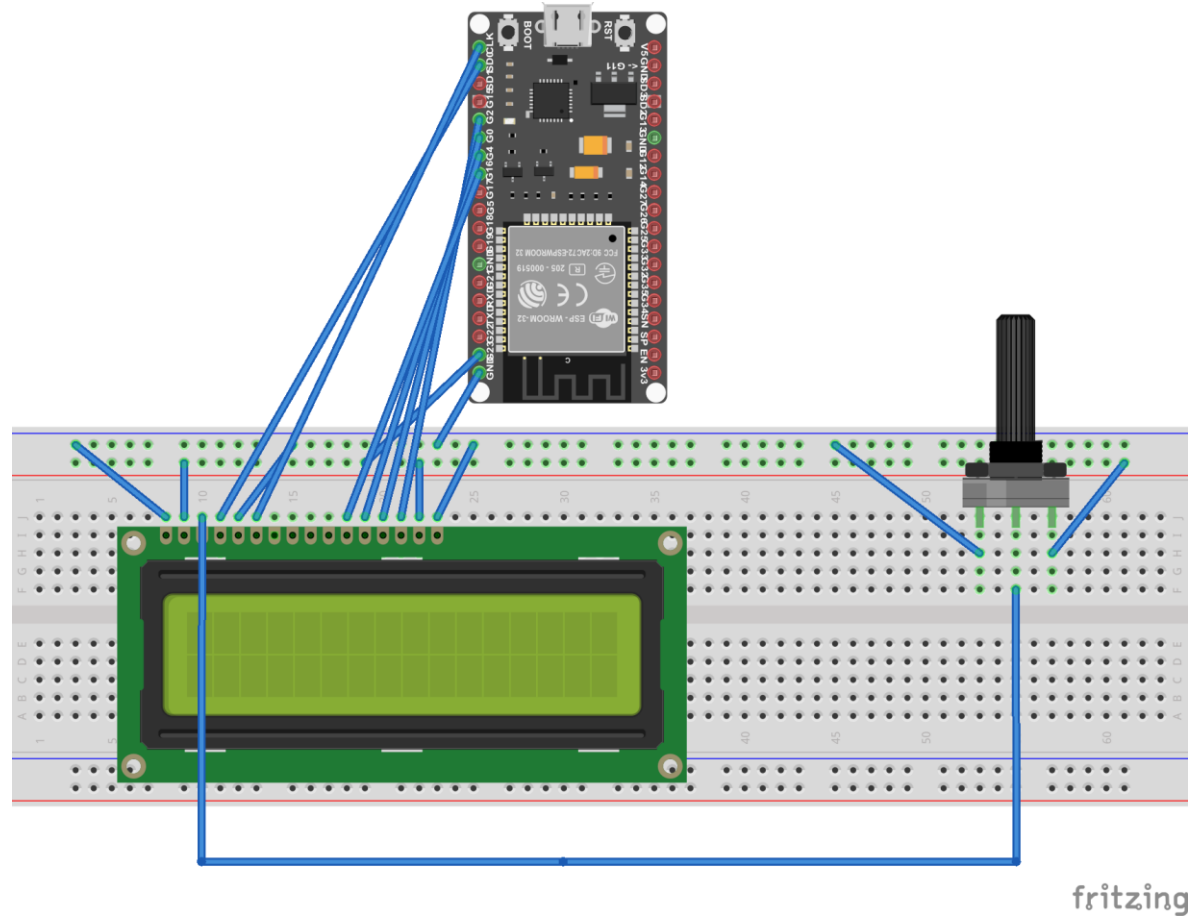
Components

- ESP32 Development Board
- 16*2 LCD display
- 10K Potentiometer
- Connecting wires
- Breadboard

Connections

LCD	ESP32	Potentiometer
1,16 (GND, LED Negative - K)	GND	GND
2,15 (VCC , LED Positive A)	VIN	VCC
VEE		Signal Pin
RS	GPIO 22	
R/W	GND	
EN	GPIO 23	
D4	GPIO 5	
D5	GPIO 18	
D6	GPIO 19	
D7	GPIO 21	

Schematic



Code

```
#include <LiquidCrystal.h>

// Create a LiquidCrystal object named 'lcd' and define the pin
connections
// Parameters: rs, enable, d4, d5, d6, d7
LiquidCrystal lcd(22, 23, 5, 18, 19, 21);

// The setup function runs once when the microcontroller is powered up or
reset
```

Code

```
void setup()
{
    // Initialize the LCD with 16 columns and 2 rows
    lcd.begin(16, 2);

    // Clear the LCD screen to start with a clean display
    lcd.clear();
}
```

Code

```
void loop()
{
    // Set the cursor position to row 0, column 1 (indexed at 0)
    lcd.setCursor(1, 0);
    // Print the string "IOT Training" at the current cursor position
    // This will display "IOT Training" on the first row of the LCD,
    starting from the second column.
    lcd.print("IOT Training");
    // The loop function will repeatedly execute these print statements,
    // continuously refreshing the displayed text on the LCD.
}
```

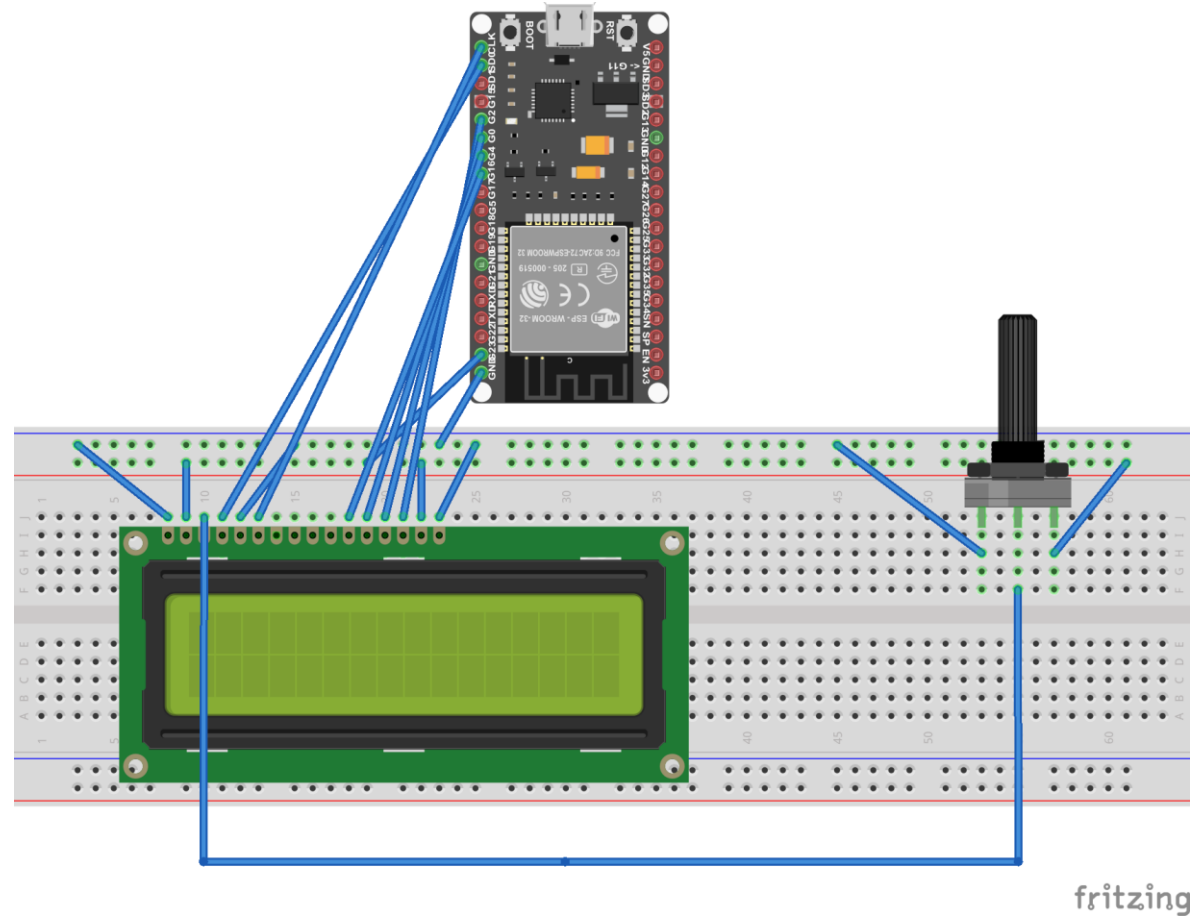

Practice Problem

- Print “I  IOT” on the LCD screen using ESP32 parallel interfacing.

Connections

LCD	ESP32	Potentiometer
1,16 (GND, LED Negative - K)	GND	GND
2,15 (VCC , LED Positive A)	VIN	VCC
VEE		Signal Pin
RS	GPIO 22	
R/W	GND	
EN	GPIO 23	
D4	GPIO 5	
D5	GPIO 18	
D6	GPIO 19	
D14	GPIO 21	

Schematic



Code

```
#include <LiquidCrystal.h>
// Custom character definition for heart symbol (♥)
byte heart[8] = {
    0b00000,
    0b01010,
    0b11111,
    0b11111,
    0b01110,
    0b00100,
    0b00000,
    0b00000
};
```

Code

```
LiquidCrystal lcd(22, 23, 5, 18, 19, 21);

void setup()
{
    lcd.begin(16, 2);
    lcd.createChar(0, heart); // Load the heart symbol to
custom character index 0
    lcd.clear();
}
```

Code

```
void loop()
{
    lcd.setCursor(0, 0);

    // Display the custom character (heart symbol) using
    lcd.print() instead of lcd.write()
    lcd.print(" I ");
    lcd.write(byte(0)); // Write the custom character (heart
    symbol) to the LCD
    lcd.print(" IOT");
```

Code

```
lcd.setCursor(0, 1);  
  lcd.print("                "); // Clear the second row before  
  printing the next message  
  
  // The loop function will repeatedly execute these print  
  statements,  
  // continuously refreshing the displayed text on the LCD.  
  // Since there are no delays or other control statements in this  
  code,  
  // the text will update rapidly, potentially causing flickering  
  on the display.  
  // To avoid flickering, consider adding appropriate delays or  
  other control mechanisms in the loop.  
}
```



Input Modules Interfacing using ESP32

Internet of Things: Theory and Applications

What are Input Modules?

- Input modules are electronic devices or components used to capture and process various types of input signals or data from external sources. These modules are crucial for interfacing with sensors, switches, and other input devices in electronic systems. Input modules convert the physical world's analog or digital signals into digital data that can be processed and used by microcontrollers, computers, or other digital systems. They play a vital role in enabling devices to interact with their environment and respond to external stimuli. Some common types of input modules include:
- **Sensor Modules:** These input modules contain sensors designed to detect and measure specific physical properties, such as temperature, humidity, pressure, light, motion, sound, distance, etc. Sensor modules provide valuable information about the surroundings and allow electronic systems to respond accordingly.

What are Input Modules?

- **Keypad Modules:** Keypad input modules consist of a set of buttons arranged in a matrix. Users can input data or commands by pressing these buttons, making them suitable for various user interfaces and control applications.
- **Switch Modules:** Switch input modules are used to detect the state of mechanical or electronic switches. They determine whether a switch is open or closed, indicating specific user actions or system states.

What are Input Modules?

- **Potentiometers:** Potentiometer input modules are variable resistors used to control analog voltage levels. They are often used for volume control, brightness adjustment, and other analog input applications.
- **Analog-to-Digital Converters (ADC):** ADC modules convert analog signals (such as voltage or current) from sensors or other analog devices into digital values that can be processed by digital systems.
- **Digital-to-Analog Converters (DAC):** DAC modules, on the other hand, do the opposite by converting digital values into analog signals, allowing digital systems to control analog devices.

What are Input Modules?

- **Communication Modules:** These modules facilitate data exchange between different devices or systems. Examples include Bluetooth modules, Wi-Fi modules, and Ethernet modules.
- **Biometric Modules:** Biometric input modules capture and process biometric data, such as fingerprints, iris patterns, or facial features, for identification and authentication purposes.
- **Microphone and Audio Input Modules:** These modules capture audio signals from microphones or audio sources and convert them into digital data for further processing.
- **Gesture Recognition Modules:** Gesture recognition input modules can detect and interpret hand or body movements, enabling gesture-based control of electronic devices.

Types of Input Modules

- **Sensor Modules:**

- Temperature Sensor Module
- Humidity Sensor Module
- Pressure Sensor Module
- Light Sensor Module
- Proximity Sensor Module
- Motion Sensor Module (e.g., PIR sensor)
- Sound Sensor Module (e.g., Microphone module)
- Gas Sensor Module (e.g., CO2 sensor, methane sensor)
- Flame Sensor Module
- Water Level Sensor Module

Types of Input Modules

- Accelerometer Module
- Gyroscope Module
- Compass Module (e.g., Magnetometer)
- Force Sensor Module
- Flex Sensor Module
- Heart Rate Sensor Module
- Fingerprint Sensor Module
- Iris Recognition Module
- RFID Module

Types of Input Modules

- **Keypad and Switch Modules:**

- Matrix Keypad Module
- **Membrane Keypad Module**
- Capacitive Touch Sensor Module
- Tactile Switch Module
- Push Button Module
- Slide Switch Module
- DIP Switch Module
- Rotary Encoder Module
- Toggle Switch Module

Types of Input Modules

- **Potentiometer Modules:**
 - Rotary Potentiometer Module
 - Linear Potentiometer Module
- **Analog Input Modules:**
 - Analog-to-Digital Converter (ADC) Module
 - Digital-to-Analog Converter (DAC) Module

Types of Input Modules

- **Communication Modules:**

- Bluetooth Module
- Wi-Fi Module
- Ethernet Module
- Zigbee Module
- LoRa Module
- GSM/GPRS Module
- GPS Module
- NFC Module
- IR Receiver Module
- RF Receiver Module
- RF Transmitter Module

Types of Input Modules

- **Biometric Input Modules:**
 - Fingerprint Scanner Module
 - Iris Scanner Module
 - Facial Recognition Module
 - Voice Recognition Module
- **Microphone and Audio Input Modules:**
 - Electret Microphone Module
 - Sound Sensor Module
 - Audio Amplifier Module

Types of Input Modules

- **Gesture Recognition Modules:**

- Gesture Sensor Module
- Image Recognition Module (for hand gesture recognition)

- **Joystick and Gamepad Modules:**

- Analog Joystick Module
- Gamepad Controller Module

- **Barcode and QR Code Scanner Modules:**

- Barcode Scanner Module
- QR Code Scanner Module

Types of Input Modules

- **Proximity Detection Modules:**
 - Proximity Sensor Module (IR-based or ultrasonic)
- **Capacitive and Inductive Sensor Modules:**
 - Capacitive Proximity Sensor Module
 - Inductive Proximity Sensor Module
- **Hall Effect Sensor Module**
- **Raindrop Sensor Module**
- **Flame Detection Sensor Module**

Types of Input Modules

- **Soil Moisture Sensor Module**
- **Laser Sensor Module**
- **Current Sensor Module**
- **Gas Leakage Sensor Module**
- **Color Sensor Module**
- **Hall Effect Sensor Module**
- **Reed Switch Module**

About Keypad & Switch Input Modules

- **Keypad Modules:**

- Consist of a set of buttons arranged in a matrix pattern.
- Used for data entry and user interaction in electronic systems.
- Each button represents a specific character, number, or function.
- Pressing a button generates an electrical signal for the system to detect.
- Commonly found in consumer electronics, security systems, and industrial control.
- Matrix keypads are popular, with different sizes like 3x3, 4x4, etc.
- Microcontroller interprets the keypress and performs actions accordingly.
- Offer tactile feedback and reliable for frequent user interactions.

About Keypad & Switch Input Modules

- **Switch Modules:**

- Used to detect the state of mechanical or electronic switches.
- Determine whether a switch is open or closed, representing specific user actions or system states.
- Various types include tactile switches, push buttons, slide switches, DIP switches, and more.
- Important for user interfaces, control systems, and circuitry configurations.
- Can be combined with other components for various functionalities.
- Microcontroller reads switch status for decision-making and control.
- Reliable and widely used in various electronic devices and systems.

Classification Keypad & Switch Input Modules

- **Types of Keypad Modules:**

- **Matrix Keypad Module:**

- Consists of a set of buttons arranged in a matrix pattern (rows and columns).
- Each button represents a specific character, number, or function.
- When a button is pressed, it creates a unique combination of row and column connections, allowing the microcontroller to identify the pressed key based on the activated row and column.
- Commonly used for user input and data entry in various electronic devices, such as calculators, security systems, and embedded systems.

Classification Keypad & Switch Input Modules

- **Membrane Keypad Module:**

- Utilizes a thin, flexible membrane with conductive pads and symbols printed on it.
- When a button is pressed, the top and bottom layers of the membrane make contact, completing an electrical circuit and generating a signal.
- Provides a cost-effective and compact solution for user input in devices like remote controls, microwave ovens, and home appliances.

- **Capacitive Touch Sensor Module:**

- Employs capacitive sensing technology to detect touch or proximity of a conductive object (e.g., human finger).
- No physical button press is required, making it ideal for touch-sensitive interfaces with no moving parts.
- Commonly used in smartphones, tablets, touchscreens, and other modern devices.

Classification Keypad & Switch Input Modules

- **Types of Switch Modules:**

- **Tactile Switch Module:**

- Also known as momentary or push-button switches.
- Provides momentary contact, meaning it stays open or closed only as long as the user presses the button.
- Commonly used for control functions, reset buttons, and user interfaces.

- **Slide Switch Module:**

- Operates by sliding a lever or actuator to change the switch position from ON to OFF or vice versa.
- Offers a simple and straightforward way to control the state of a circuit.
- Frequently found in electronic devices, power supplies, and lighting controls.

Classification Keypad & Switch Input Modules

- **DIP Switch Module:**

- Dual In-line Package (DIP) switches are small switches mounted on a PCB.
- Each switch can be toggled ON or OFF to configure the circuit's settings or select options.
- Used in electronic circuits for configuration, address setting, and mode selection.

- **Toggle Switch Module:**

- A lever-operated switch that stays in its position (ON or OFF) until physically toggled again.
- Ideal for maintaining a constant state, such as power ON/OFF for devices like lamps and power supplies.

About Membrane Keypad Input Modules

- A Membrane Keypad Module is an input device that utilizes a thin, flexible membrane with conductive pads and symbols printed on its surface. When a user presses a button, the top and bottom layers of the membrane make contact, completing an electrical circuit and generating an electrical signal. This signal is then detected and processed by a microcontroller or electronic system, allowing it to recognize the pressed key.

Characteristics of Membrane Keypad Input Modules

- **Cost-Effective and Compact:** Membrane keypads are cost-effective to manufacture due to their simple design and use of flexible materials. They are also compact and can be made in various sizes and shapes, making them suitable for small devices with space constraints.
- **Tactile Feedback:** Most membrane keypads provide tactile feedback when pressed, giving users a physical indication that their input has been registered. This tactile response enhances the user experience.
- **Durable and Reliable:** Membrane keypads are durable and resistant to wear, making them suitable for applications that involve frequent use. Their sealed design protects the internal components from dust, dirt, and moisture, increasing their reliability and lifespan.

Characteristics of Membrane Keypad Input Modules

- **Customizable Symbols:** The symbols or labels printed on the membrane can be customized to represent specific functions or characters, providing clear and intuitive user interfaces.
- **Simple Installation:** Membrane keypads are typically designed to be easily mounted or adhered to a flat surface, simplifying the installation process.
- **No Moving Parts:** Unlike mechanical switches, membrane keypads have no moving parts, which reduces the risk of mechanical failure and enhances their longevity.

Sending an Input Process in Membrane Keypad Input Modules

- The process of sending input to a Membrane Keypad Module involves several steps. Below is a detailed explanation of each step:
- **Physical Interaction:**
 - The user physically interacts with the membrane keypad by pressing a button.
 - When the user presses a button, it causes the top layer of the flexible membrane to make contact with the bottom layer, creating an electrical connection.
- **Electrical Connection:**
 - When the top and bottom layers of the membrane come into contact, the conductive pads on the two layers form an electrical connection.
 - This connection completes an electrical circuit for the specific button that the user pressed.
- **Conductive Material:**
 - The conductive pads on the membrane keypad are typically made of a conductive material, such as a conductive ink or graphite, that allows electric current to flow through it.

Sending an Input Process in Membrane Keypad Input Modules

- **Matrix Configuration:**

- Membrane keypads are often designed in a matrix configuration with rows and columns of buttons.
- Each button is associated with a specific row and column, creating a unique combination for each button.

- **Scanning by Microcontroller:**

- The membrane keypad is connected to a microcontroller or electronic system.
- The microcontroller scans the rows and columns of the matrix in sequence to detect button presses.

- **Reading the Keypress:**

- During the scanning process, the microcontroller checks for continuity or electrical connection in each row and column pair.
- When a button is pressed, the microcontroller identifies the corresponding row and column that form a connection.

Sending an Input Process in Membrane Keypad Input Modules

- **Decoding the Keypress:**

- The microcontroller uses the row and column information to decode the button press.
- It determines which button was pressed based on the unique row-column combination.

- **Output Processing:**

- Once the microcontroller identifies the button press, it processes the input data accordingly.
- Depending on the application, the microcontroller may respond to the button press by performing a specific action, executing a command, or collecting data.

Sending an Input Process in Membrane Keypad Input Modules

- **User Interface Interaction:**

- In applications with user interfaces, the microcontroller may update the display or provide feedback to the user to indicate that the input was received.

- **Continuous Scanning:**

- The microcontroller continues to scan the keypad matrix repeatedly to detect any changes in button states.
- This continuous scanning process allows the system to respond to real-time user input.

Errors in Membrane Keypad Input Modules

- **Common errors or issues that may arise with Membrane Keypad Modules include:**
- **Ghosting and Key Jamming:**
 - Ghosting occurs when pressing one button inadvertently triggers neighboring buttons due to electrical interference.
 - Key jamming happens when multiple buttons appear to be pressed simultaneously, leading to incorrect input detection.
- **Debouncing and Contact Bounce:**
 - Contact bounce occurs when the keypad's conductive pads make and break contact rapidly during a button press, resulting in multiple signals.
 - Debouncing errors can lead to multiple input detections for a single button press.
- **Worn or Damaged Membrane:**
 - Over time, the membrane keypad may wear out or become damaged, causing inconsistent or non-responsive button presses.
- **Open or Short Circuits:**
 - Faulty connections or damaged traces on the keypad can result in open circuits (no electrical connection) or short circuits (unintended connections).

Errors in Membrane Keypad Input Modules

- **Intermittent Connections:**

- Loose connections between the keypad and the microcontroller or other components can lead to intermittent button readings.

- **Incorrect Pin Mapping:**

- Mismatched pin connections between the keypad and the microcontroller can lead to incorrect button readings.

- **Incorrect Keycode Interpretation:**

- Improperly mapped keycodes in the microcontroller can cause the system to interpret button presses incorrectly.

- **Environmental Factors:**

- Environmental conditions such as humidity, moisture, and dust can affect the reliability of the keypad's conductive pads.

Errors in Membrane Keypad Input Modules

- **Poor Quality Keypads:**

- Low-quality membrane keypads may have inconsistent or inaccurate button readings.

- **Erosion of Conductive Pads:**

- In harsh conditions or with excessive usage, the conductive pads on the membrane may erode, affecting their conductivity.

- **Incorrect Placement or Alignment:**

- Misalignment or improper placement of the membrane keypad during assembly can lead to misreadings.

Self-Adhesive Membrane Keypad 4x4 - 16 Keys:



About Self-Adhesive Membrane Keypad 4x4 - 16 Keys:

Overview of Self-Adhesive Membrane Keypad 4x4 - 16 Keys:

- A self-adhesive membrane keypad 4x4 with 16 keys is a type of user interface input device designed for easy integration into various electronic systems. It is a compact and cost-effective solution that consists of a thin, flexible membrane with 16 tactile buttons arranged in a 4x4 matrix pattern. The keypad is self-adhesive, allowing for easy mounting on flat surfaces, making it ideal for applications where space is limited.

Working Principle of Self-Adhesive Membrane Keypad 4x4 - 16 Keys:

- The working principle of the self-adhesive membrane keypad relies on the electrical connection formed when a button is pressed. Each button on the keypad has a conductive pad on the bottom layer of the membrane, which, when pressed, comes into contact with a corresponding conductive pad on the top layer. This contact completes an electrical circuit, and the microcontroller or electronic system connected to the keypad can detect the button press.
- The 4x4 matrix configuration allows the system to identify the specific button pressed by scanning the rows and columns of the matrix. By reading the electrical connections between the rows and columns, the microcontroller can determine the exact button that the user has pressed.

About Self-Adhesive Membrane Keypad 4x4 - 16 Keys:

- **Applications in IoT (Internet of Things):**
- The self-adhesive membrane keypad 4x4 - 16 keys finds applications in various IoT devices and smart systems. Some common applications include:
- **Home Automation Systems:** In IoT-based home automation systems, the keypad can be used to control lights, fans, appliances, and other smart devices.
- **Security Systems:** The keypad can serve as an input method for IoT-based security systems, such as alarm panels or access control systems, where users can enter PIN codes for authentication.

About Self-Adhesive Membrane Keypad 4x4 - 16 Keys:

- **Smart Locks:** In smart door lock systems, the keypad can be used for PIN code entry to unlock doors without the need for physical keys.
- **IoT Prototyping and DIY Projects:** The self-adhesive keypad is popular in prototyping and DIY IoT projects due to its ease of use and compact size.
- **IoT-Based Industrial Control Systems:** The keypad can be incorporated into IoT-enabled industrial control panels for controlling machinery, processes, and automation.
- **Healthcare Devices:** In IoT-based healthcare devices and medical equipment, the keypad can be used for data entry, parameter adjustment, or user inputs.
- **IoT-Based Consumer Electronics:** The keypad can be integrated into various IoT-enabled consumer electronics, such as smart remote controls, IoT-based gaming devices, and more.

4*4 Membrane Keypad Interfacing with ESP32

Brief about Self-Adhesive Membrane Keypad 4x4 with ESP32 Interfacing:

- A self-adhesive membrane keypad 4x4 with 16 keys is a user interface input device that can be easily integrated with the ESP32 microcontroller for various applications. The keypad consists of a thin, flexible membrane with 16 tactile buttons arranged in a 4x4 matrix pattern. The self-adhesive feature allows for easy mounting on flat surfaces, making it convenient for use in projects with limited space.

Needed Components for ESP32 Interfacing:

- For interfacing the self-adhesive membrane keypad 4x4 with the ESP32, you will need the following components:
- **ESP32 Development Board:** The main microcontroller that will be responsible for reading the button presses and controlling the system.

4*4 Membrane Keypad Interfacing with ESP32

- **Self-Adhesive Membrane Keypad 4x4:** The input module with 16 tactile buttons arranged in a 4x4 matrix pattern.
- **Resistors (Optional):** If needed, pull-up or pull-down resistors to ensure stable readings from the keypad.
- **Breadboard and Jumper Wires:** For prototyping and making connections between the keypad and ESP32.
- **Power Supply:** Sufficient power source to power the ESP32 and the keypad.

4*4 Membrane Keypad Interfacing with ESP32

Libraries for ESP32 Interfacing:

- For interfacing the self-adhesive membrane keypad 4x4 with the ESP32, you can use the following libraries:
- **Arduino IDE Libraries:** Libraries that come pre-installed with the Arduino IDE, providing various functionalities like reading digital inputs and managing matrix keypad scans.

4*4 Membrane Keypad Interfacing with ESP32

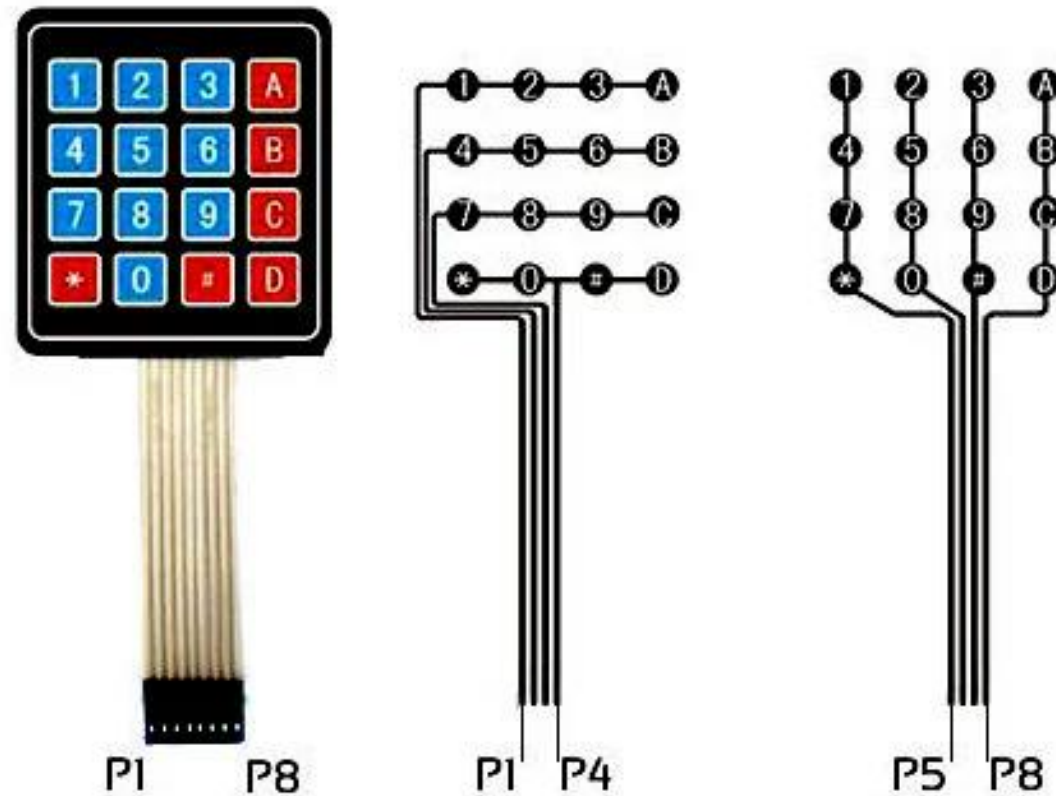
- **Keypad Library:** A popular third-party library for handling matrix keypad scans and reading button presses efficiently.
- **Wire (I2C) Library (if using I2C-based keypads):** If the keypad has an I2C interface, you'll need the Wire library for I2C communication.

4*4 Membrane Keypad Interfacing with ESP32

Module Pinout:

- The self-adhesive membrane keypad 4x4 with 16 keys is a matrix keypad, and its pinout may vary depending on the specific model or manufacturer. However, the basic pin configuration typically follows a 4x4 matrix pattern.
- For interfacing the matrix keypad with the ESP32, you will typically need to connect the following pins:
- **Row Pins (4): Connect these pins from the keypad to GPIO pins on the ESP32.**
- **Column Pins (4): Connect these pins from the keypad to GPIO pins on the ESP32.**
- The specific GPIO pins on the ESP32 will depend on your choice, and you can configure them in the code using the Keypad library or custom code for matrix scanning.
- Remember to refer to the datasheet or pinout diagram of the specific self-adhesive membrane keypad you are using for accurate pin configurations and functionalities. Additionally, make sure to use appropriate pull-up or pull-down resistors, if required, to ensure reliable readings from the keypad.

How 4*4 Membrane Keypad Works?



Pinout of 4*4 Membrane Keypad

Pin Number	Description
ROWS	
1	PIN1 is taken out from 1st ROW
2	PIN2 is taken out from 2nd ROW
3	PIN3 is taken out from 3rd ROW
4	PIN4 is taken out from 4th ROW
COLUMN	
5	PIN5 is taken out from 1st COLUMN
6	PIN6 is taken out from 2nd COLUMN
7	PIN7 is taken out from 3rd COLUMN
8	PIN8 is taken out from 4th COLUMN

Controlling 4*4 Keypad on ESP32

Technical Specs

- Maximum Rating: 24 VDC, 30mA
- Number of buttons: 16
- Dimensions: 77 x 70 x 0.8 mm
- Belt length with connector: 83 mm
- Type of connector: 1 x 8 pin – female raster 2.54 (goldpin connector)

Controlling 4*4 Keypad with ESP32

Components Needed:

- ESP32 Dev Kit
- Keypad module (e.g., 4x4 or 3x4 keypad)
- Breadboard and jumper wires

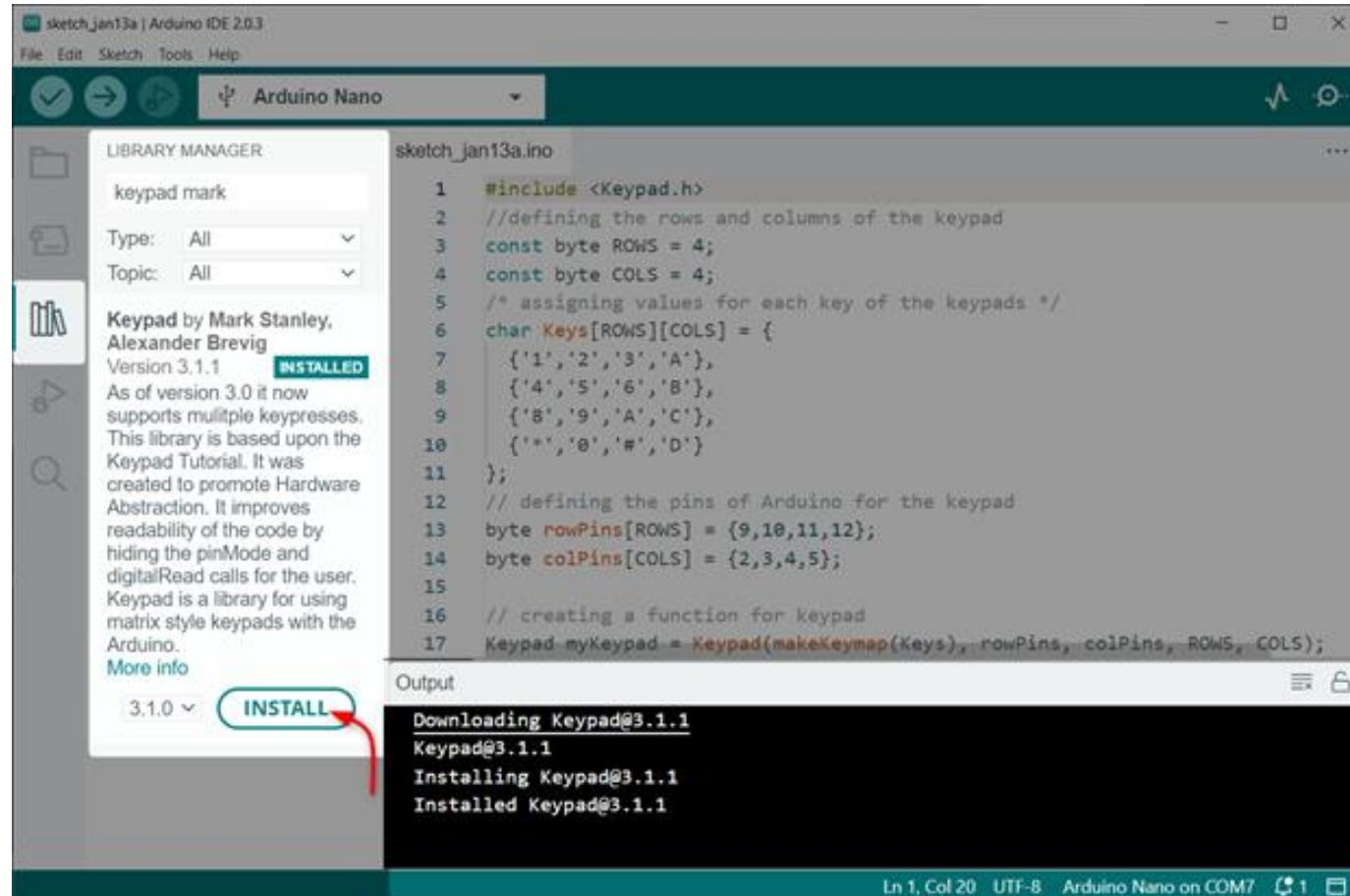
Controlling 4*4 Keypad with ESP32

- **Library:** To work with a keypad, you can use the "Keypad" library in Arduino IDE. This library simplifies the process of reading input from the keypad and supports both 3x4 and 4x4 keypad configurations.
- **Connections:** The connections between the keypad and ESP32 depend on the type of keypad module. In general, you need to connect the keypad rows and columns to specific ESP32 pins. For example, a 4x4 keypad might have eight pins (four rows and four columns) and can be connected as follows:
 - Keypad Rows (R1 to R4) to ESP32 pins
 - Keypad Columns (C1 to C4) to ESP32 pins

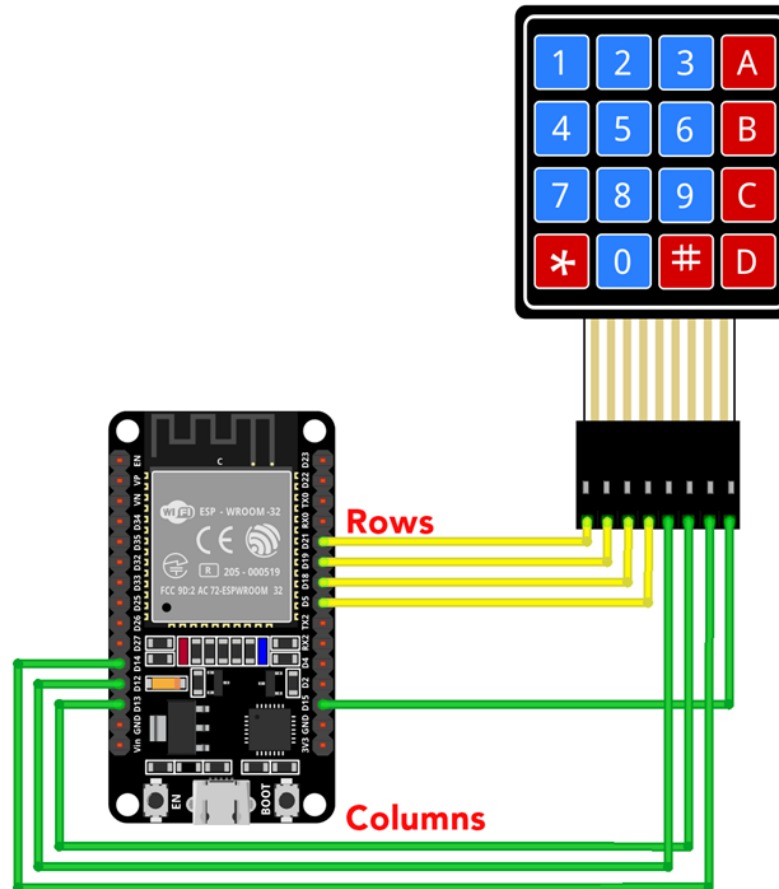
Example

- Print the keypad characters in the serial monitor.

Controlling 4*4 Keypad with ESP32 – Install the Library



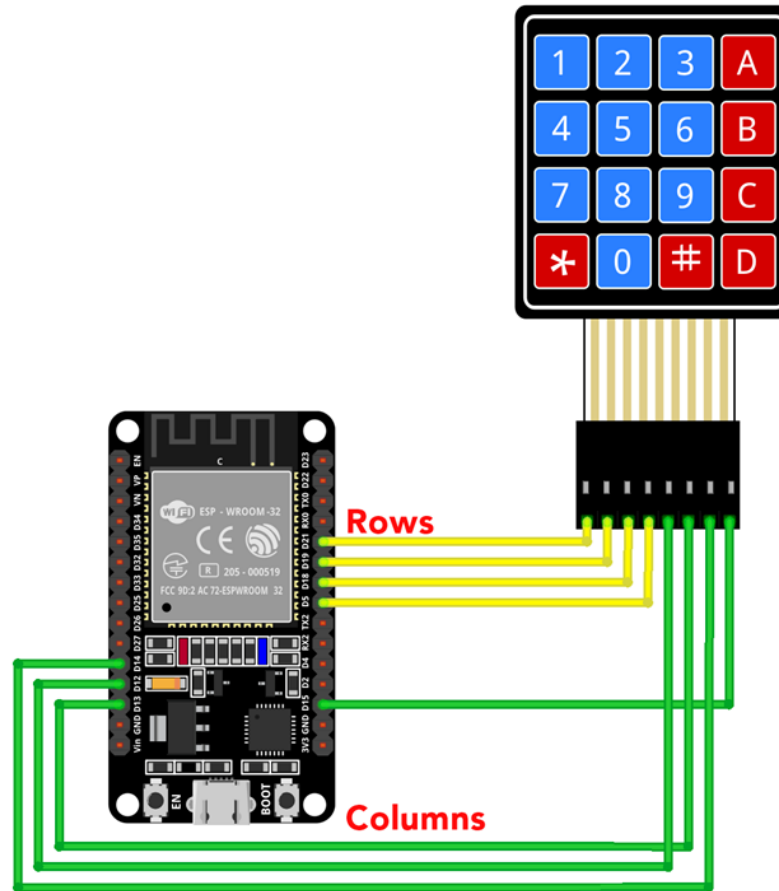
Connecting 4*4 Membrane Keypad to ESP32



Connecting 4*4 Membrane Keypad to ESP32 Types

keypad Pin	ESP32
Row 1	D21
Row 2	D19
Row 3	D18
Row 4	D5
Column 1	D12
Column 2	D13
Column 3	D14
Column 4	D15

Schematic



Code

```
#include <Keypad.h>    /* Included keypad library */

#define ROW_NUM        4 /* Define keypad Rows */
#define COLUMN_NUM     4 /* Define keypad Columns */

char keys[ROW_NUM][COLUMN_NUM] = {
    {'1', '2', '3', 'A'},
    {'4', '5', '6', 'B'},
    {'7', '8', '9', 'C'},
    {'*', '0', '#', 'D'}
};
```

Code

```
byte pin_rows[ROW_NUM]      = {21, 19, 18, 5}; /* Initialized ESP32  
Pins for Rows */  
byte pin_column[COLUMN_NUM] = {12, 13, 14, 15}; /* Initialized  
ESP32 Pins for Columns */  
  
/* Function for keypad */  
Keypad keypad = Keypad(makeKeymap(keys), pin_rows, pin_column,  
ROW_NUM, COLUMN_NUM);  
  
void setup() {  
    Serial.begin(115200); /* Baud Rate for Serial Communication */  
}
```

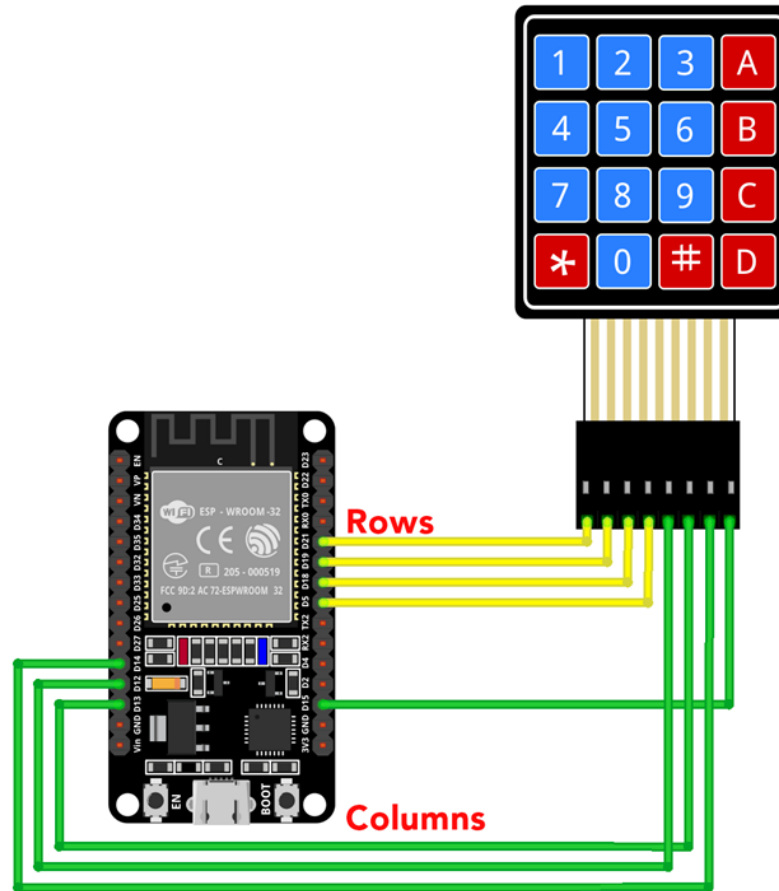
Code

```
void loop() {  
    char key = keypad.getKey();    /* Take input from  
keypad */  
    if (key) {                      /* If Key is pressed  
display the output */  
        Serial.println(key);  
    }  
}
```

Practice

- Blink the embedded LED of ESP32 with Keypad

Schematic



Code

```
#include <Keypad.h>

#define ROW_NUM      4 // Number of keypad Rows
#define COLUMN_NUM   4 // Number of keypad Columns

char keys[ROW_NUM][COLUMN_NUM] = {
    {'1', '2', '3', 'A'},
    {'4', '5', '6', 'B'},
    {'7', '8', '9', 'C'},
    {'*', '0', '#', 'D'}
};
```

Code

```
byte pin_rows[ROW_NUM] = {21, 19, 18, 5}; // ESP32  
GPIO pins for Rows
```

```
byte pin_column[COLUMN_NUM] = {12, 13, 14, 15}; //  
ESP32 GPIO pins for Columns
```

```
Keypad keypad = Keypad(makeKeymap(keys), pin_rows,  
pin_column, ROW_NUM, COLUMN_NUM);
```


Code

```
const int embeddedLedPin = 2; // ESP32 embedded LED pin (GPIO 2)

void setup() {
    Serial.begin(9600); // Initialize the Serial Monitor
    pinMode(embeddedLedPin, OUTPUT); // Set the embedded LED pin as
output
    digitalWrite(embeddedLedPin, LOW); // Turn off the embedded LED
initially
}

void loop() {
    char key = keypad.getKey();
```

Code

```
if (key) {  
    Serial.println(key);  
    // Perform actions based on the pressed key  
    switch (key) {  
        case '1':  
            digitalWrite(embeddedLedPin, HIGH); // Turn  
on the embedded LED  
            break;
```

Code

```
case '2':  
    digitalWrite(embeddedLedPin, LOW); // Turn off  
the embedded LED  
    break;  
default:  
    // Do nothing for other keys  
    break;  
}  
}  
}
```