

# ADS1115 I2C external ADC with ESP32 in Arduino IDE

This guide shows you how to interface external adc AD1115 with ESP32 to measure analog voltage with high accuracy. ADS1115 provides data over I2C communication. It consists of four analog channels. At the end of this guide, I will provide an example of analog voltage measurement with ADS1115 and ESP32.

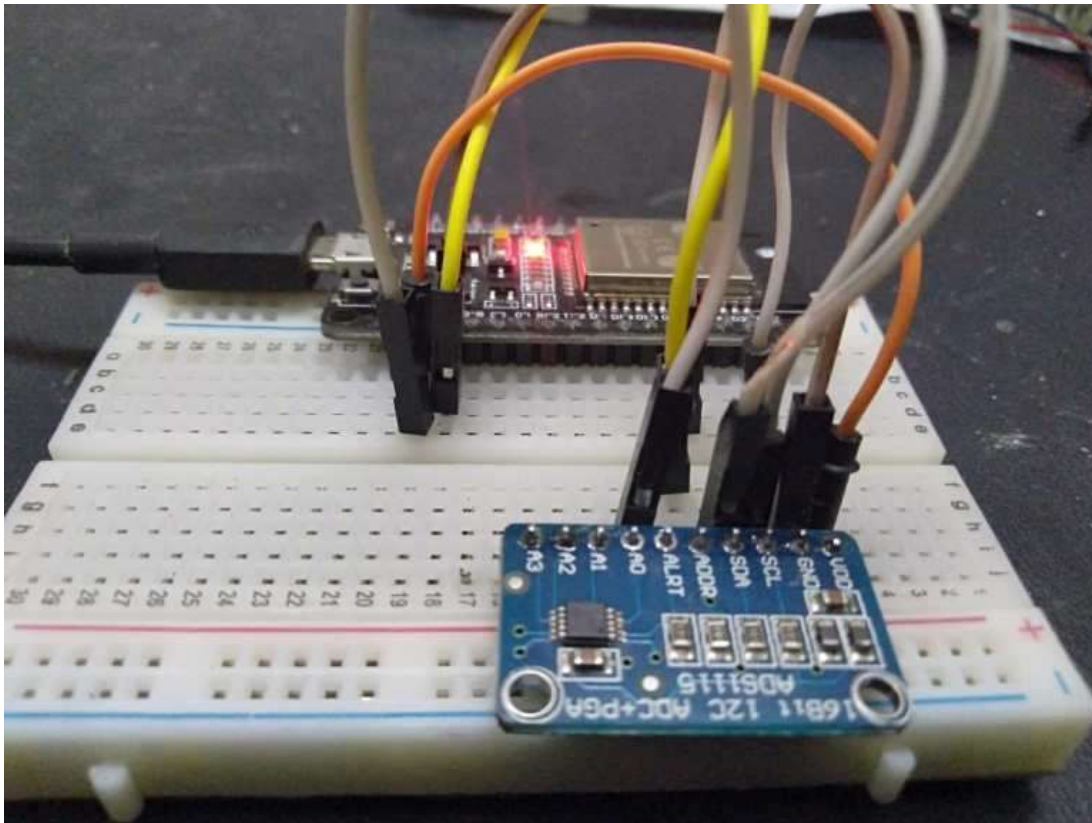


Table of Contents



## Why we need to use external ADC with ESP32?

Although, ESP32 has two built-in ADC modules namely; ADC0 and ADC1 and each channel is of 12-bits. But, the main issue of ESP32 ADC's is that it has a non-linear characteristics and it exhibits non-linear behavior as explained in already posted

**ESP32 ADC tutorial** . It can not differentiate between 1mv and 2mv signals which means it offers very low resolution. In order to resolve these issues, we can use external high resolution programmable ADC IC.

Reasons to use External analog to digital converter are mentioned below:

**Low resolution issue** : Other than the issue of **non-linear behavior** and low resolution of built-in analog to digital converter, ESP32 has all excellent features required for internet of things project. Therefore, we can use an external ADC with this development board to resolve problem of measuring analog signal with high accuracy.

**Interfacing issues** : For example, you want to interface LM35 temperature sensor with ESP32, you can not connect it directly due to low resolution and inaccurate behavior of built analog to digital converter of ESP32. LM35 temperature sensor gives output of 1mv per one degree centigrade of temperature. Built-in adc can not measure 1mV accurately and so is ESP32.

## Introduction to ADS1115 External ADC

- ADS1115 is a 16-bit analog to digital converter that consists of four analog channels.
- It can be used in to measure both positive and negative voltages.
- It works in Single Ended, Differential and comparator mode. In single ended differential mode, it is used to measure positive voltages only and in comparator mode, it can measure both positive and negative voltages (useful when you want to measure voltage of battery).
- It works on I2C communication. We can interface it with any microcontroller having I2C communication support.
- It has a programmable comparator.
- It can be used in four modes.

ADS1115 is a 16 bit I2C ADC having four analog channels. It provides output in signed integer format. From total 16 bits, one bit is assigned for positive and negative number. Therefore, only 15 bits are used to measure the voltage and ADC resolution is calculated according to 15 bits.

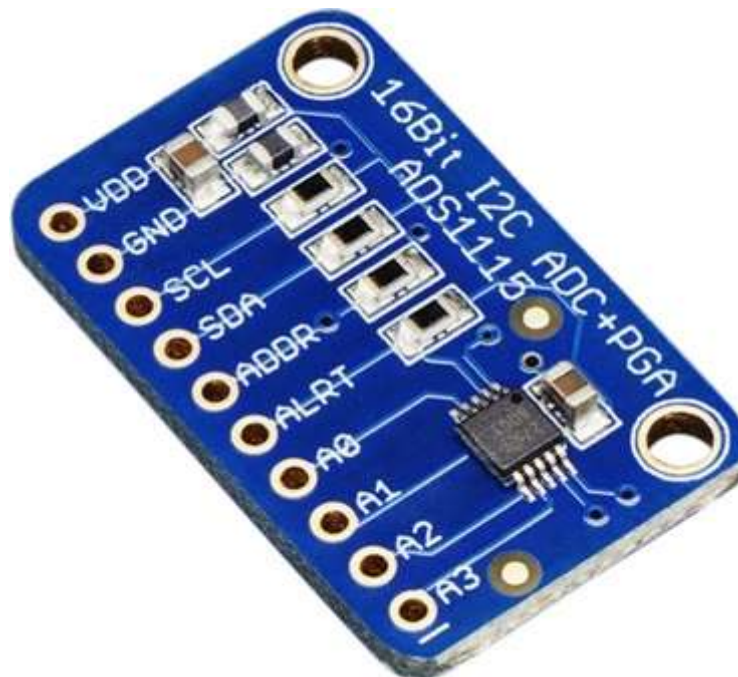
- We get  $2^{15} = 32,768$  possible output values for every analog input.
- Possible output digital values will be between 0-32767.

- These 15 bits are used to get the magnitude of voltage through I2C communication.
- The weight of every bit can be defined with Programmable Gain Amplifier (PGA) by setting a reference voltage value.
- In default mode, the reference voltage value is +/-6.144 volts.
- 0 volts represents 0 digital value and 6.144 volts represents 32767 as a digital value.
- But we can not measure 6.144 volts directly with this chip. The maximum voltage which we can measure depends on the supply voltage to chip which is 5.5 volts in case of ADS1115.

## Resolution of ADS1115

- The resolution of this external ADC is calculated by dividing 6.144 with 32767 which is equal to **0.1875mV**.
- So the minimum analog voltage we can measure with ADS1115 external I2C ADC is 0.1875mV which is almost 50% greater accuracy than built-in analog to digital converter module of ESP32.

## Pinout of ADS1115



It consists of four pins and table below shows functionality of each pin.

Pin name	Functionality

<b>Vdd ( Power supply pin)</b>	Connect power supply between 2.2 – 5.5 volts
<b>GND ( Common reference pin )</b>	Connect with ground pin of power supply
<b>SCL</b>	I2C SCL ( serial clock pin)
<b>SDA</b>	I2C SDA ( Serial data pin)
<b>ADDR</b>	I2C slave select pin or address
<b>ALRT</b>	Alert/Ready
<b>A0</b>	Analog channel 0
<b>A1</b>	Analog channel 1
<b>A2</b>	Analog channel 2
<b>A3</b>	Analog channel 3

Addressing modes can be set in four different mode by connecting address pin with SCL, SDA, GND or VDD. Table below provides description on connection of address pin with SCL, SDA, GND or VDD pin according to value of address.

<b>ADDR pin value</b>	<b>Connection with Pin</b>
0x48	Connect address pin to <b>GND</b>
0x49	Connect address pin to <b>VDD</b>
0x4A	Connect address pin to <b>SDA</b>
0x4B	Connect address pin to <b>SCL</b>

ADDR pin is used for selection of four different devices with one pin. So, we can connect it with any of four pins listed in table. You can explore further about addressing modes by reading datasheet.

## DATASHEET ADS1115

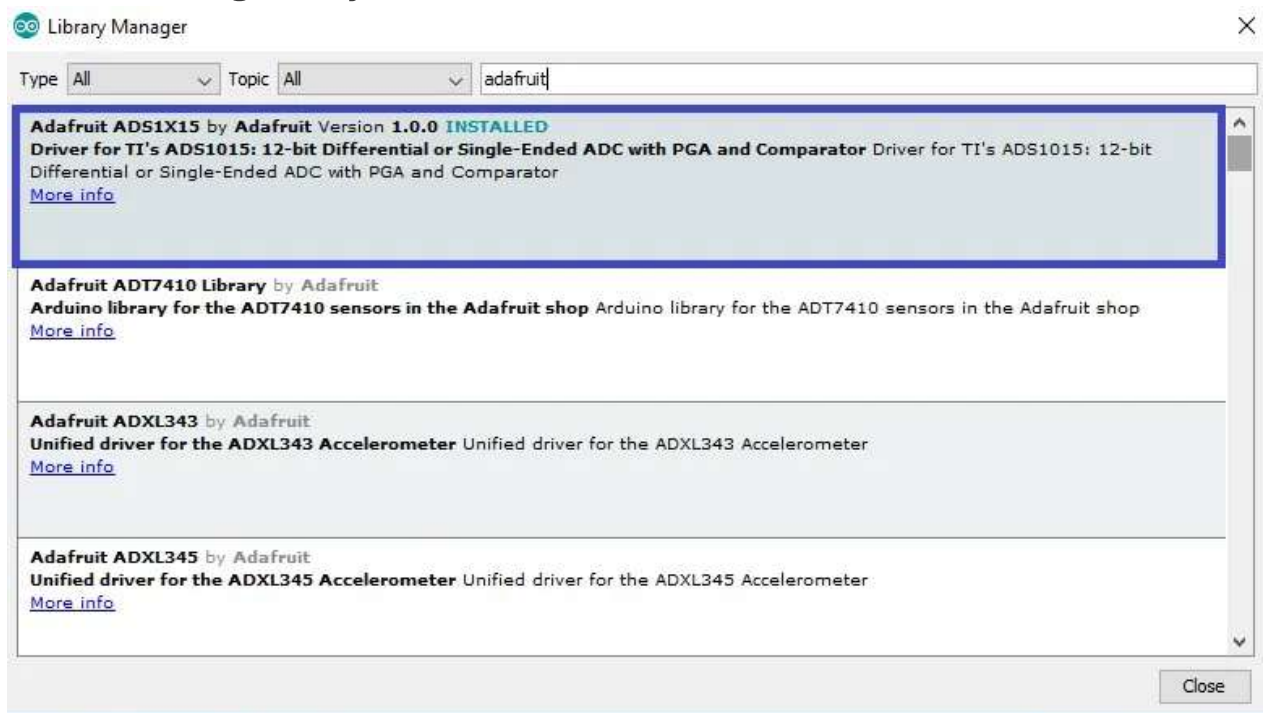
## ADS1115 library in Arduino IDE

we will be using Arduino IDE to program ESP32. If you are using Arduino IDE first time with ESP32, you can check this tutorials:

- [How to install ESP32 in Arduino IDE](#)

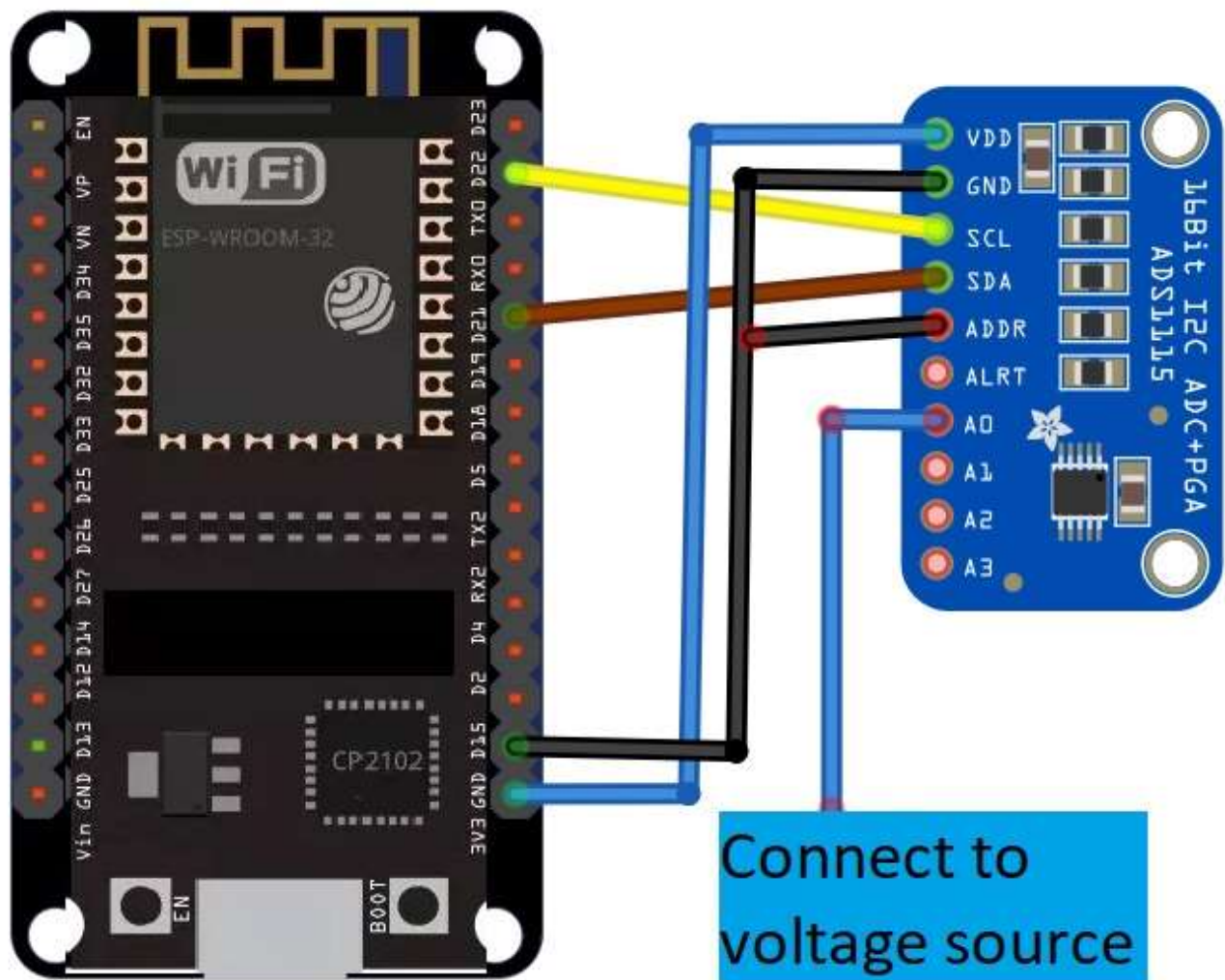
As mentioned in previous sections, ADS1115 communicates data to ESP32 over I2C communication. We need to use I2C pins of ESP32 to read data. To handle communication and receive measured voltage over I2C, we can use **ADS1115 library** of Adafruit. Follow these steps to install library.

- To install library, open your Arduino IDE and follow this menu **Sketch > Include Library > Manage Libraries** and click on manage libraries.
- After that type “Adafruit in search bar” and select ADS1X15 to install it.
- After installing library click on close button and restart Arduino IDE.



## Interfacing ADS1115 external ADC with ESP32

Now make the connections according to this layout given below:



ESP32	ADS1115 external ADC
VDD	VDD
GND	GND
GPIO21 ( SDA Pin )	SDA
GPIO22 ( SCL Pin )	SCL
GND	ADDR
Analog voltage signal	A0

ESP32 GPIO21 and GPIO22 is used for I2C communication. Connect Analog signal to which voltage you want to measure with A0 pins of ADS1115.

## Code to measure analog voltage

Now open your Arduino IDE and paste this code to code editor window. After compiling code, upload it to ESP32 by clicking on upload button.

```
#include <Wire.h>
#include <Adafruit_ADS1015.h>

Adafruit_ADS1115 ads(0x48);
float Voltage = 0.0;

void setup(void)
{
  Serial.begin(9600);
  ads.begin();
}

void loop(void)
{
  int16_t adc0;

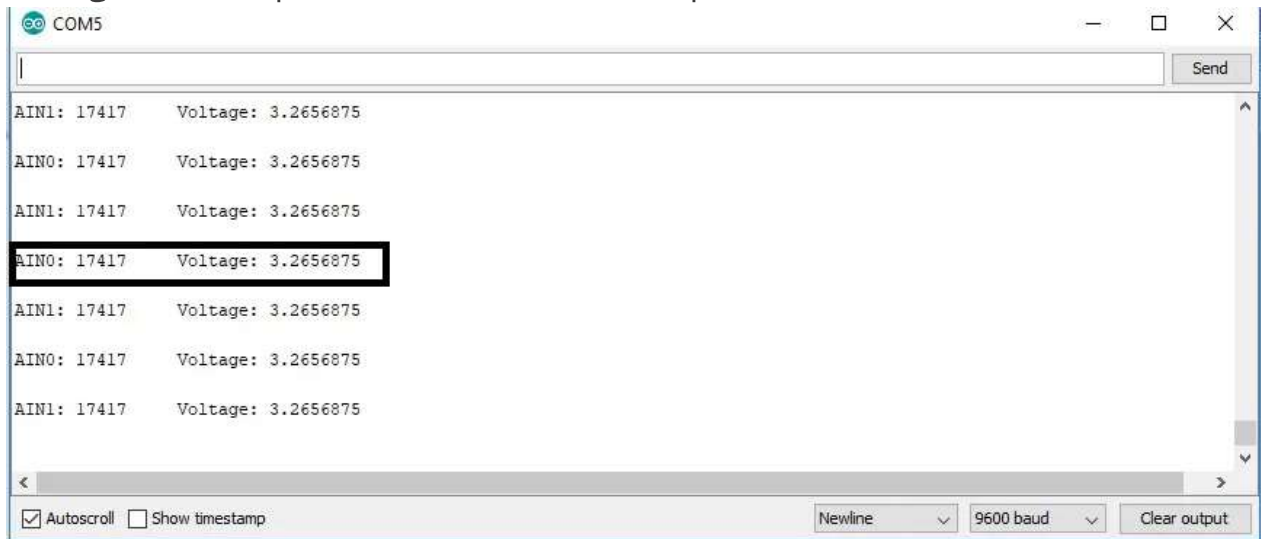
  adc0 = ads.readADC_SingleEnded(0);
  Voltage = (adc0 * 0.1875)/1000;

  Serial.print("AIN0: ");
  Serial.print(adc0);
  Serial.print("\tVoltage: ");
  Serial.println(Voltage, 7);
  Serial.println();

  delay(1000);
}
```

- After uploading code, open serial monitor, you will see the output of voltage and adc0 on the serial monitor.
- For demonstration, I have connected vdd pin of ESP32 with analog channel zero.

- As you can see, we are getting close to 3.3 volts on serial monitor which is operating voltage of ESP32 board, you can verify this output by measuring voltage on VDD pin of ESP32 with the help of voltmeter.



## Code working

First two lines used to add the library of ADS1115 and I2C communication.

```
#include <Wire.h>
#include <Adafruit_ADS1015.h>
```

First line defines the sensor type and address mode, we are using ADS1115. But this library can be used with other versions like ADS1015. Second line declares a variable of float type to store voltage value.

```
Adafruit_ADS1115 ads(0x48);
float Voltage = 0.0;
```

Inside the `setup()` function, `Serial.begin()` function defines the baud rate of 9600 and `ads.begin()` function starts to read adc value.

```
Serial.begin(9600);
ads.begin();
```

Inside the `loop()` function, first we initialize the 16 bit long integer variable `adc0` which is used to store output of analog channel zero.

```
int16_t adc0;
```

`ds.readADC_SingleEnded(0)` function read analog channel zero adc value and store this value in `adc0` integer variable. This statement converts  $(adc0 * 0.1875) / 1000$  adc value into voltage where 0.1875 is a resolution of ADS1115.

```
adc0 = ads.readADC_SingleEnded(0);  
Voltage = (adc0 * 0.1875) / 1000;
```

`Serial.print()` functions prints value of ADC and voltage on serial monitor. `delay()` function adds a delay of one second after every reading.

```
Serial.print("AIN0: ");  
Serial.print(adc0);  
Serial.print("\tVoltage: ");  
Serial.println(Voltage, 7);  
Serial.println();  
delay(1000);
```

In summary, in this tutorial, you learned the following concepts:

- Introduction to ADS1115 external I2C based ADC
- How to use it with ESP32
- How to measure voltage with ADS1115 and ESP32

You can check ESP32 projects also:

- [Create ESP32 Web server in Arduino IDE](#)
- [DHT11/DHT22 interfacing and display value on Web server](#)
- [BMP180 with ESP32 and web server](#)
- [Control Servo motor with Web server using ESP32](#)

 ESP32

## Subscribe to Blog via Email