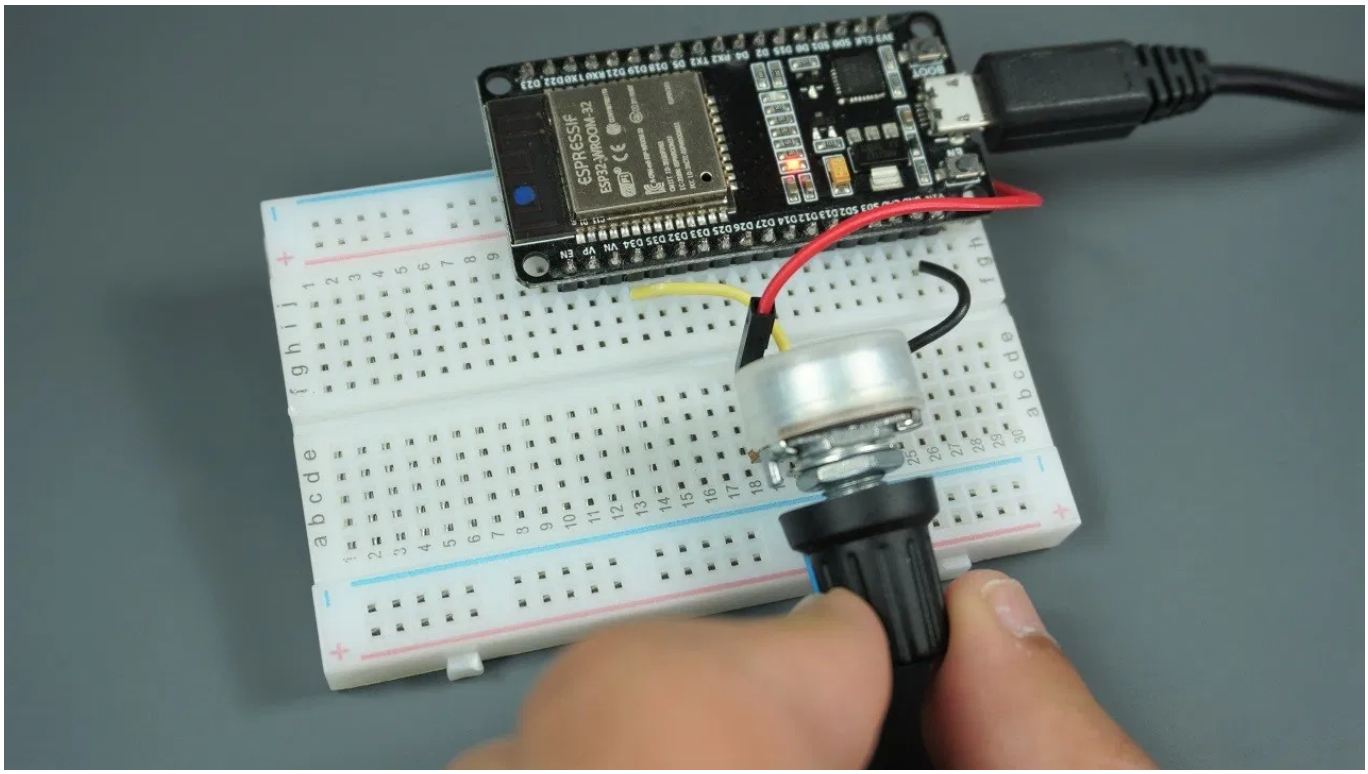
ESP32 ADC – Read Analog Values with Arduino IDE

This article shows how to read analog inputs with the ESP32 using Arduino IDE. Analog reading is useful to read values from variable resistors like potentiometers, or analog sensors.

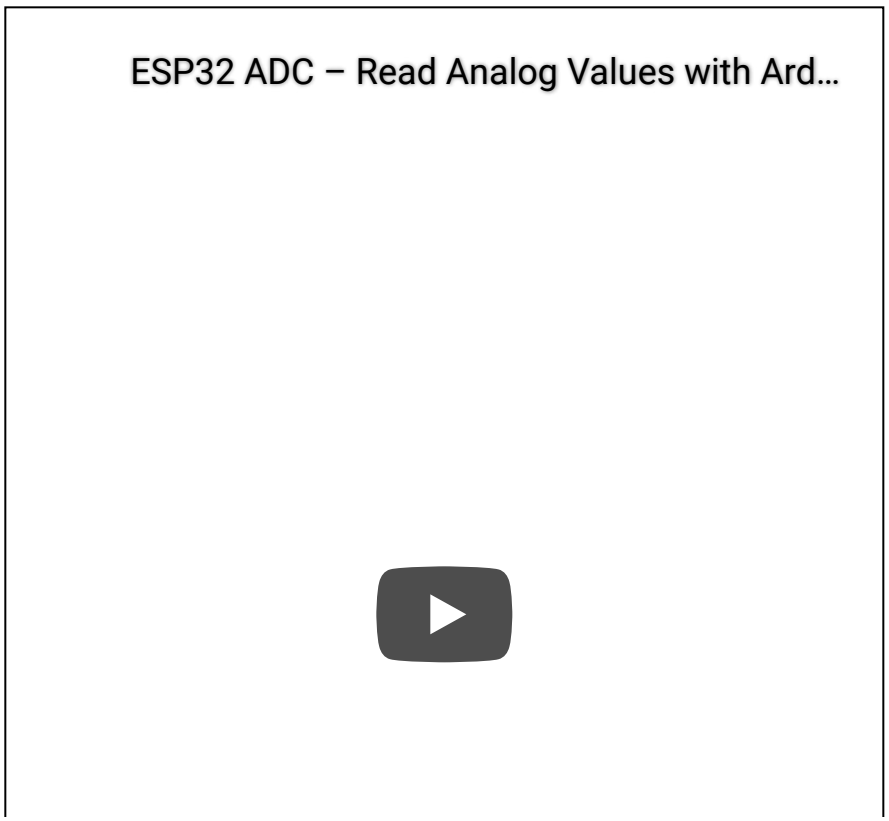


Reading analog inputs with the ESP32 is as easy as using the `analogRead(GPIO)` function, that accepts as argument, the GPIO you want to read.

We also have other tutorials on how to use analog pins with ESP board:

- ESP8266 ADC – Read Analog Values with Arduino IDE, MicroPython and Lua
- ESP32 Analog Readings with MicroPython

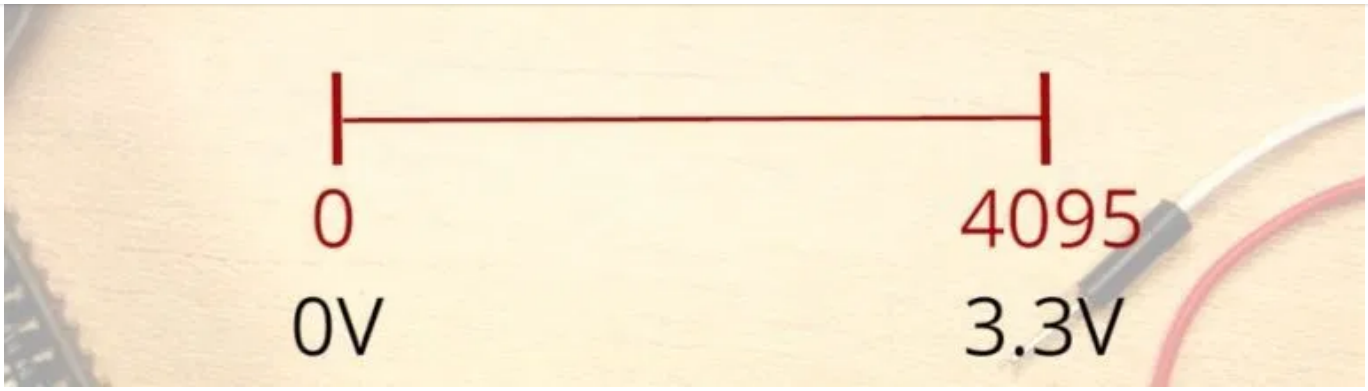
Watch the Video



Analog Inputs (ADC)

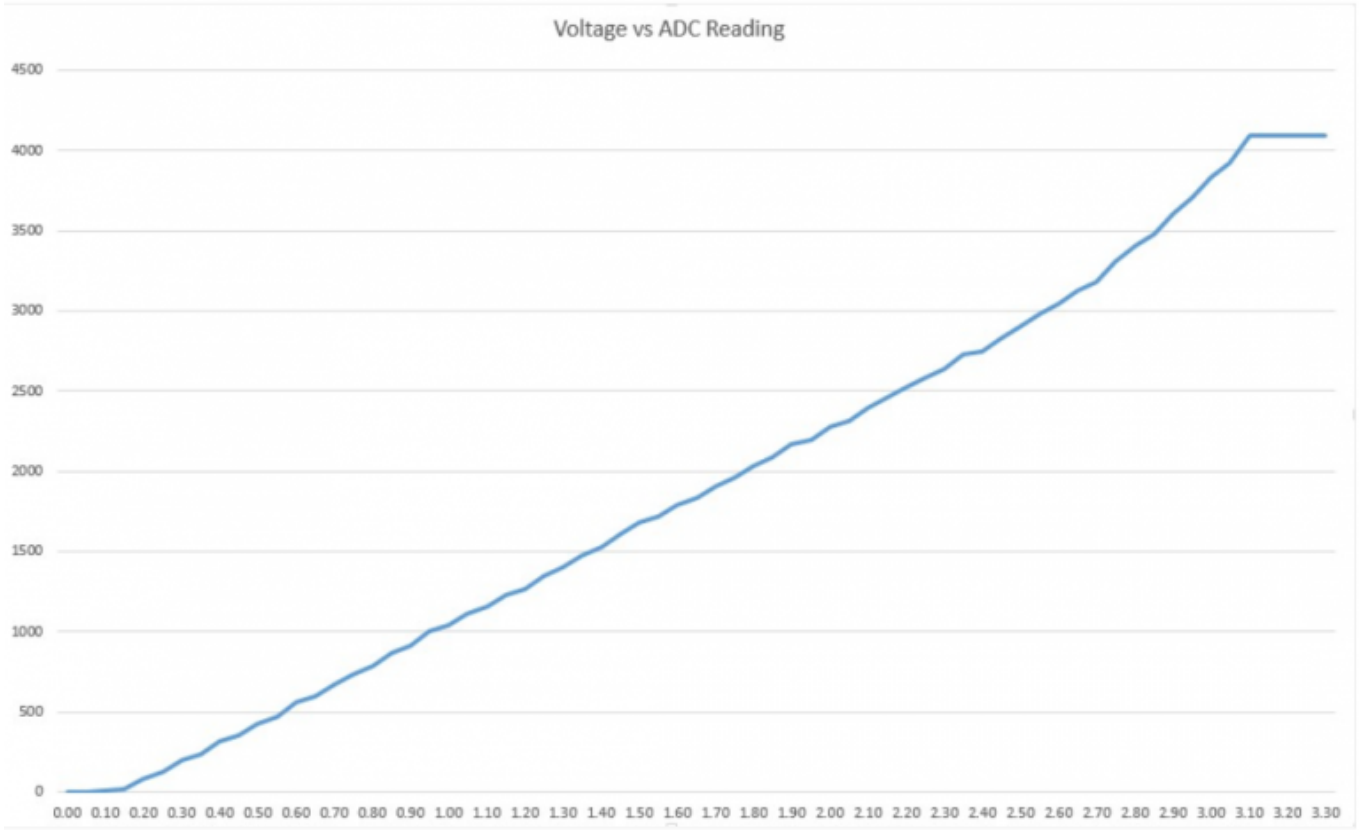
Reading an analog value with the ESP32 means you can measure varying voltage levels between 0 V and 3.3 V.

The voltage measured is then assigned to a value between 0 and 4095, in which 0 V corresponds to 0, and 3.3 V corresponds to 4095. Any voltage between 0 V and 3.3 V will be given the corresponding value in between.



ADC is Non-linear

Ideally, you would expect a linear behavior when using the ESP32 ADC pins. However, that doesn't happen. What you'll get is a behavior as shown in the following chart:



View source

This behavior means that your ESP32 is not able to distinguish 3.3 V from 3.2 V. You'll get the same value for both voltages: 4095.

There's a discussion on GitHub about this subject.

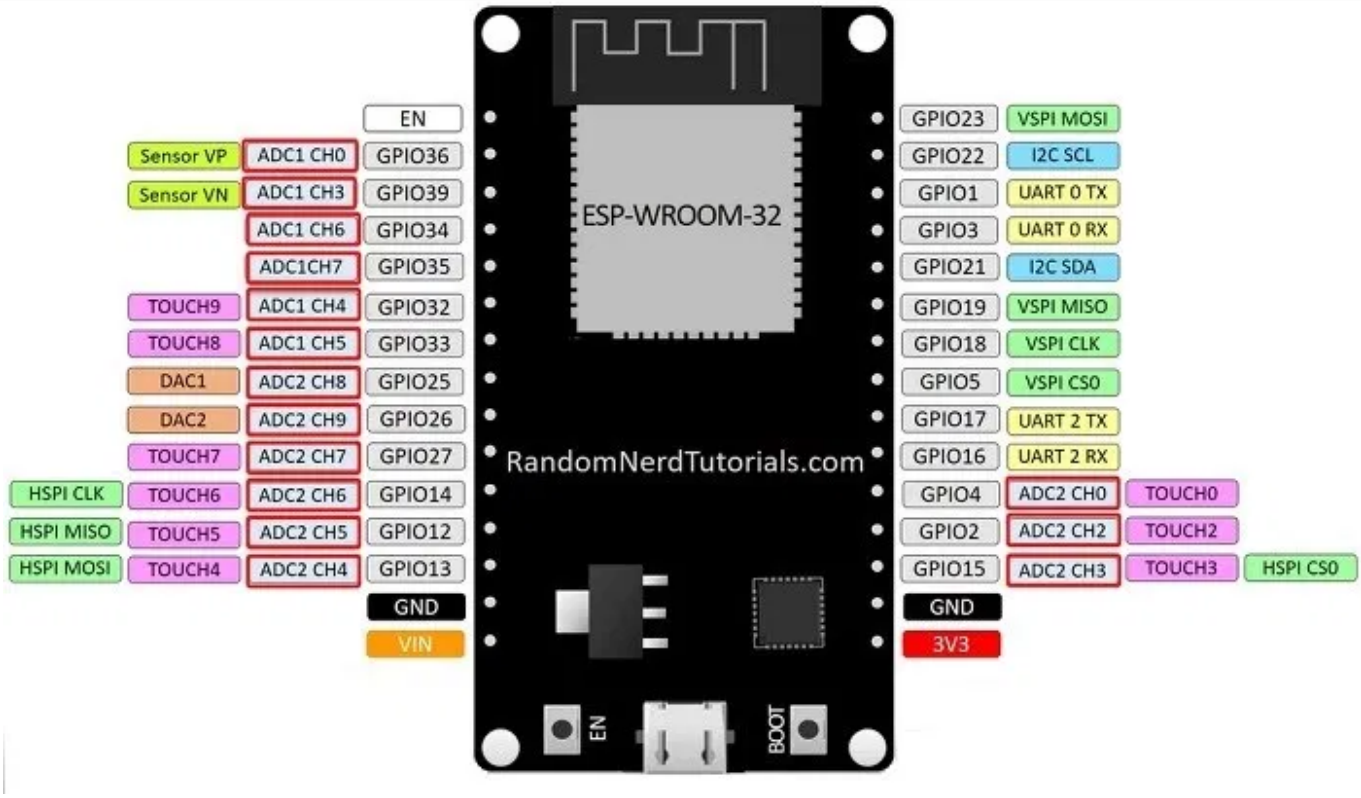
analogRead() Function

Reading an analog input with the ESP32 using the Arduino IDE is as simple as using the `analogRead()` function. It accepts as argument, the GPIO you want to read:

```
analogRead(GPIO);
```

The ESP32 supports measurements in 18 different channels. Only 15 are available in the DEVKIT V1 DOIT board (version with 30 GPIOs).

Grab your ESP32 board pinout and locate the ADC pins. These are highlighted with a red border in the figure below.



Learn more about the ESP32 GPIOs: ESP32 Pinout Reference.

These analog input pins have 12-bit resolution. This means that when you read an analog input, its range may vary from 0 to 4095.

> **Note:** ADC2 pins cannot be used when Wi-Fi is used. So, if you're using Wi-Fi and you're having trouble getting the value from an ADC2 GPIO, you may consider using an ADC1 GPIO instead, that should solve your problem.

Other Useful Functions

There are other more advanced functions to use with the ADC pins that can be useful in other projects.

- `analogReadResolution(resolution)` : set the sample bits and resolution. It can be a value between 9 (0 – 511) and 12 bits (0 – 4095). Default is 12-bit resolution.

resolution.

- `analogSetCycles(cycles)` : set the number of cycles per sample. Default is 8. Range: 1 to 255.
- `analogSetSamples(samples)` : set the number of samples in the range. Default is 1 sample. It has an effect of increasing sensitivity.
- `analogSetClockDiv(attenuation)` : set the divider for the ADC clock. Default is 1. Range: 1 to 255.
- `analogSetAttenuation(attenuation)` : sets the input attenuation for all ADC pins. Default is `ADC_11db` . Accepted values:
  - `ADC_0db` : sets no attenuation. ADC can measure up to approximately 800 mV (1V input = ADC reading of 1088).
  - `ADC_2_5db` : The input voltage of ADC will be attenuated, extending the range of measurement to up to approx. 1100 mV. (1V input = ADC reading of 3722).
  - `ADC_6db` : The input voltage of ADC will be attenuated, extending the range of measurement to up to approx. 1350 mV. (1V input = ADC reading of 3033).
  - `ADC_11db` : The input voltage of ADC will be attenuated, extending the range of measurement to up to approx. 2600 mV. (1V input = ADC reading of 1575).
- `analogSetPinAttenuation(pin, attenuation)` : sets the input attenuation for the specified pin. The default is `ADC_11db` . Attenuation values are the same from previous function.
- `adcAttachPin(pin)` : Attach a pin to ADC (also clears any other analog mode that could be on). Returns TRUE or FALSE result.
- `adcStart(pin)`, `adcBusy(pin)` and `resultadcEnd(pin)` : starts an ADC convertion on attached pin's bus. Check if conversion on the pin's ADC bus is currently running (returns TRUE or FALSE). Get the result of the conversion: returns 16-bit integer.

There is a very good video explaining these functions that you can watch here.

Read Analog Values from a Potentiometer with ESP32

For this example, you need the following parts:

- ESP32 DOIT DEVKIT V1 Board (read Best ESP32 development boards)
- Potentiometer
- Breadboard
- Jumper wires

You can use the preceding links or go directly to MakerAdvisor.com/tools to find all the parts for your projects at the best price!

Schematic

Wire a potentiometer to your ESP32. The potentiometer middle pin should be connected to GPIO 34. You can use the following schematic diagram as a reference.

add-on installed before proceeding:

- **Windows** instructions – ESP32 Board in Arduino IDE
- **Mac and Linux** instructions – ESP32 Board in Arduino IDE

Open your Arduino IDE and copy the following code.

```
// Potentiometer is connected to GPIO 34 (Analog ADC1_CH6)
const int potPin = 34;

// variable for storing the potentiometer value
int potValue = 0;

void setup() {
  Serial.begin(115200);
  delay(1000);
}

void loop() {
  // Reading potentiometer value
  potValue = analogRead(potPin);
  Serial.println(potValue);
  delay(500);
}
```

View raw code

This code simply reads the values from the potentiometer and prints those values in the Serial Monitor.

In the code, you start by defining the GPIO the potentiometer is connected to. In this example, GPIO 34 .

In the `setup()`, initialize a serial communication at a baud rate of 115200.

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

In the `loop()`, use the `analogRead()` function to read the analog input from the `potPin`.

```
potValue = analogRead(potPin);
```

Finally, print the values read from the potentiometer in the serial monitor.

```
Serial.println(potValue);
```

Upload the code provided to your ESP32. Make sure you have the right board and COM port selected in the Tools menu.

Testing the Example

After uploading the code and pressing the ESP32 reset button, open the Serial Monitor at a baud rate of 115200. Rotate the potentiometer and see the values changing.

The maximum value you'll get is 4095 and the minimum value is 0.

Wrapping Up

- The ESP32 DEVKIT V1 DOIT board (version with 30 pins) has 15 ADC pins you can use to read analog inputs.
- These pins have a resolution of 12 bits, which means you can get values from 0 to 4095.
- To read a value in the Arduino IDE, you simply use the `analogRead()` function.
- The ESP32 ADC pins don't have a linear behavior. You'll probably won't be able to distinguish between 0 and 0.1V, or between 3.2 and 3.3V. You need to keep that in mind when using the ADC pins.

We hope you've find this short guide useful. If you want to learn more about the ESP32, enroll in our course: Learn ESP32 with Arduino IDE.

Other ESP32 guides that you may also like:

- ESP32 OLED Display with Arduino IDE
- ESP32 with DHT Temperature and Humidity Sensor using Arduino IDE
- ESP32 Web Server with DHT readings
- 20+ ESP32 Projects and Tutorials

Thanks for reading.

Recommended Resources

Build a Home Automation System from Scratch » With Raspberry Pi, ESP8266, Arduino, and Node-RED.

Home Automation using ESP8266 eBook and video course » Build IoT and home automation projects.

What to Read Next...

ESP8266 Client-Server Wi-Fi Communication Between Two Boards (NodeMCU)

ESP8266 NodeMCU Access Point (AP) for Web Server

ESP32 I2C Communication: Set Pins, Multiple Bus Interfaces and Peripherals (Arduino IDE)

ESP8266 NodeMCU Web Server with Slider: Control LED Brightness (PWM)

ESP32 Web Server with BME680 – Weather Station (Arduino IDE)

7 Weekend Projects/Tutorials For the ESP8266 WiFi Module

Arduino IDE”

James
June 1, 2019 at 7:22 am

Thanks Rui for another very useful artical. I found the Other Useful Functions particularly useful.
Please check the analogSetWidth(width) they seems to be a typo and I am interested in what this does.
Please explain more the difference between Width, Cycles and Samples. I am sure samples is the number of times a reading is taken, but is the reading returned as an average? What is the limit and do you have a suggested maximum for this?
Also, is the analogSetAttenuation(attenuation) default set to ADC_11db? I am trying to understand this, because if 1V input = ADC reading of 3959 what would happen with a input of 3.3V. My experience says the default must be ADC_0db.

Reply

Ray Jones
June 1, 2019 at 10:29 am

One of the greatest problems with the ESP32 is it may have 18 ADC inputs, but if you use WiFi you cannot use any of the 10 inputs that direct to ADC2.
Yes 10 inputs cannot be used!

to the Internet!

Reply

Alex

June 28, 2020 at 5:30 pm

If you desperately need more analog inputs, try disconnecting from the WiFi, using the pins, then reconnecting and publishing any results.

Otherwise, it might be better to consider a simpler WiFi controller and some supporting ADC chips.

Reply

Blackneron

June 3, 2019 at 4:22 am

If you would like a more linear ADC behavior, you can check my Bitbucket repository at:

bitbucket.org/Blackneron/esp32_adc/

I tried to make an adjustment function for the ADCs of the ESP32.

Reply

June 3, 2019 at 8:42 am

That's great! Awesome job.
Thank you so much for sharing. It will certainly be useful for our readers.
Regards,
Sara

Reply

amjad
February 10, 2020 at 7:39 pm

thank you!

Reply

mahdar
May 29, 2020 at 1:15 pm

Sara Santos : i'd like to question, I have an error when trying the df-robot tds sensor using the esp32 board, the sensor reading results are zero, the ADC pin used is gpio 34, power5v. I have tried to replace it with another ADC pin but still the results of the tds readings are zero, please advise

Reply

May 30, 2020 at 9:28 am

Hi.
I've never tried a TDS sensor with the ESP32.
But that GPIO should work.
Measure with a multimeter to see if the sensor voltage output is actually changing. This way, you know that the problem is not the sensor.
Regards,
Sara

Reply

mahdar fajari
May 30, 2020 at 2:00 pm

ok thankyou so much

Reply

Norbert
June 10, 2020 at 11:32 pm

Hello Sara,

Minor typo on your function summary table:

"resultadcEnd(pin)" -> "adcEnd(pin)" to get the result (or similar).

Reply

David

August 19, 2020 at 2:58 pm

Hi Rui & co,
Could you advise the best pin(s) to use to get analog readings from the 36pin do-it ESP32?
I am using pin 39 now although I get a constant 4096 from an ldr.

I don't know what I'd do without Random Nerd Tutorials
David

Reply

Sara Santos

August 21, 2020 at 11:10 am

Hi David.
Try GPIO32 and GPIO 33.
Regards,
Sara

Reply

David

August 21, 2020 at 11:37 am

David

Reply

David
August 25, 2020 at 8:21 am

Thnx Sara,
Would using adc2 stop NOW from working reliably? Also does it particularly matter which is assigned first in a sketch – ADC pins, WiFi or NOW?
thank you
David

Reply

Sara Santos
August 25, 2020 at 4:27 pm

Hi.
I haven't tested ADC with ESP-NOW, so I'm not sure.
ESP-NOW doesn't use Wi-Fi, but it uses the Wi-Fi library, so I don't know if it conflicts.
Experiment and see what you get.
As for your second question, it doesn't matter the order as long as the variables are defined before they are called.

Regards,
Sara

Ray Jones

August 25, 2020 at 8:27 pm

My understanding of the ADC2 conflict with WiFi is nothing to do with software, but with internal power routing to the WiFi radio requires power to be diverted from the ADC2 peripheral.
Once you turn the radio on, the power to ADC2 is robbed, and as a result all readings become unreliable.

Sara Santos

August 26, 2020 at 4:58 pm

That's right.
Thanks for the insight.
Regards,
Sara

Leave a Comment

Name *

Email *

Website

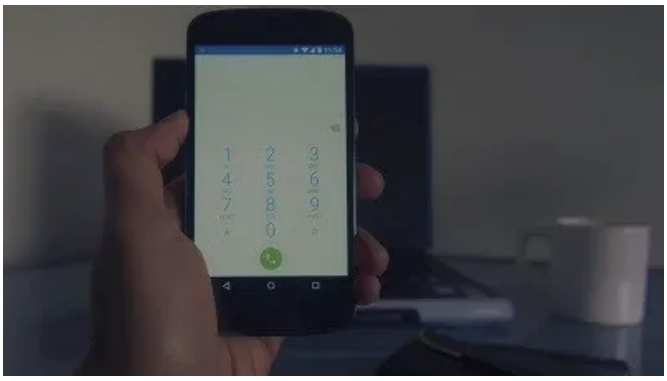
Notify me of follow-up comments by email.

Notify me of new posts by email.

Post Comment

Visit Maker Advisor – Tools and Gear for makers, hobbyists and DIYers »

Home Automation using ESP8266 eBook and video course » Build IoT and home automation projects.



Build a Home Automation System from Scratch » With Raspberry Pi, ESP8266, Arduino, and Node-RED.