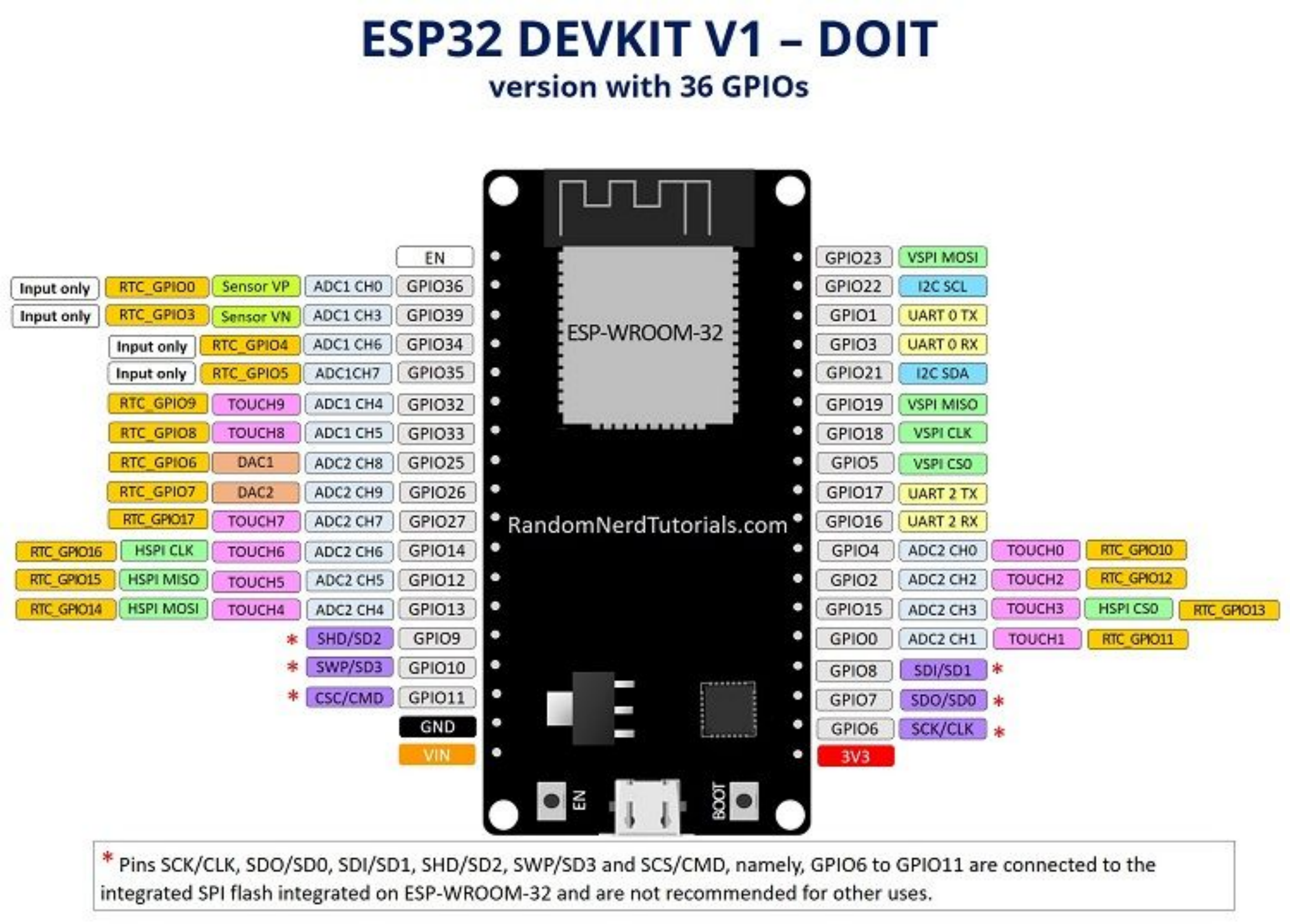
ESP32

Input and Output

- **Vin is primarily an input pin** for external power (5V-12V, regulated down to 3.3V).
- 3.3V is output only
- 5V may be used for 5V power source (depending on the board) and if powered by Vin may supply 5V output
- All GPIOs supply 3.3V as output when HIGH
- Avoidi powering via Vin or 5V is connected to USB
- 3.3V pin is never used for input power source



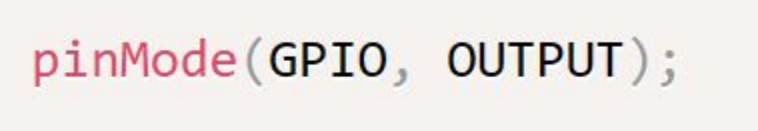
ESP32 DEVKIT V1 – DOIT

version with 36 GPIOs

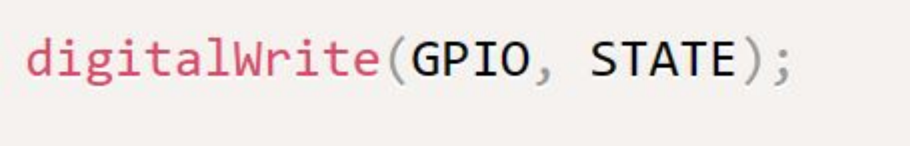
ESP-WROOM-32
RandomNerdTutorials.com
EN
BOOT
EN
Input only RTC_GPIO0 Sensor VP ADC1 CH0 GPIO36
Input only RTC_GPIO3 Sensor VN ADC1 CH3 GPIO39
Input only RTC_GPIO4 ADC1 CH6 GPIO34
Input only RTC_GPIO5 ADC1CH7 GPIO35
RTC_GPIO9 TOUCH9 ADC1 CH4 GPIO32
RTC_GPIO8 TOUCH8 ADC1 CH5 GPIO33
RTC_GPIO6 DAC1 ADC2 CH8 GPIO25
RTC_GPIO7 DAC2 ADC2 CH9 GPIO26
RTC_GPIO17 TOUCH7 ADC2 CH7 GPIO27
RTC_GPIO16 HSPI CLK TOUCH6 ADC2 CH6 GPIO14
RTC_GPIO15 HSPI MISO TOUCH5 ADC2 CH5 GPIO12
RTC_GPIO14 HSPI MOSI TOUCH4 ADC2 CH4 GPIO13
* SHD/SD2 GPIO9
* SWP/SD3 GPIO10
* CSC/CMD GPIO11
GND
VIN
GPIO23 VSPI MOSI
GPIO22 I2C SCL
GPIO1 UART 0 TX
GPIO3 UART 0 RX
GPIO21 I2C SDA
GPIO19 VSPI MISO
GPIO18 VSPI CLK
GPIO5 VSPI CS0
GPIO17 UART 2 TX
GPIO16 UART 2 RX
GPIO4 ADC2 CH0 TOUCH0 RTC_GPIO10
GPIO2 ADC2 CH2 TOUCH2 RTC_GPIO12
GPIO15 ADC2 CH3 TOUCH3 HSPI CS0 RTC_GPIO13
GPIO0 ADC2 CH1 TOUCH1 RTC_GPIO11
GPIO8 SDI/SD1 *
GPIO7 SDO/SD0 *
GPIO6 SCK/CLK *
3V3

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

Esp32 Digital Outputs



```
pinMode(GPIO, OUTPUT);
```



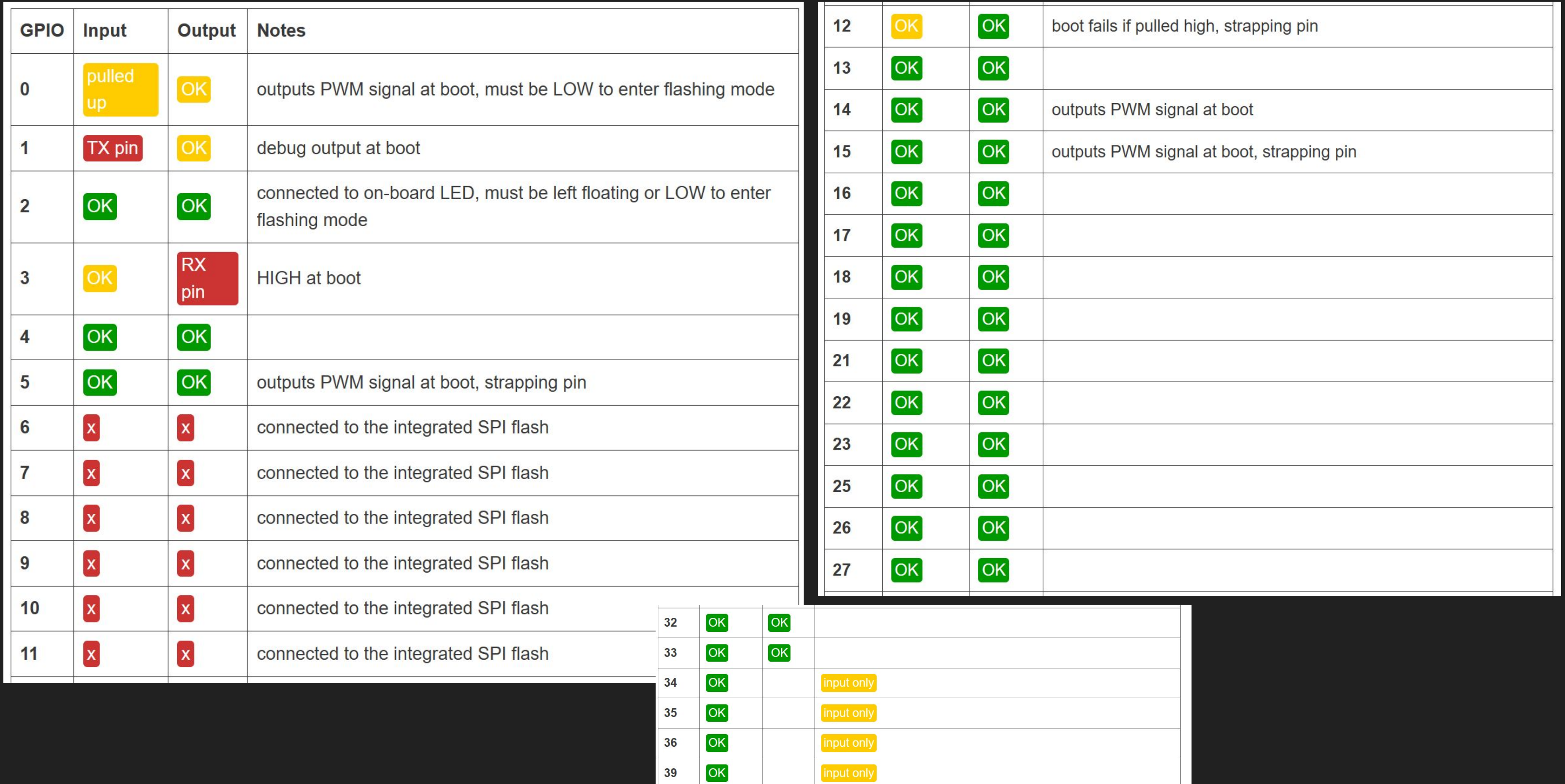
```
digitalWrite(GPIO, STATE);
```

All pins except for 6-11 and 34-36 and 39 can be used for output.

All pins output up to 3.3V. The Arduino output 5V.
When connected to a USB cable connected to computer the 3.3V and 5V pins supply power. When not connected you can connect these pins to power supplies to power the ESP32.

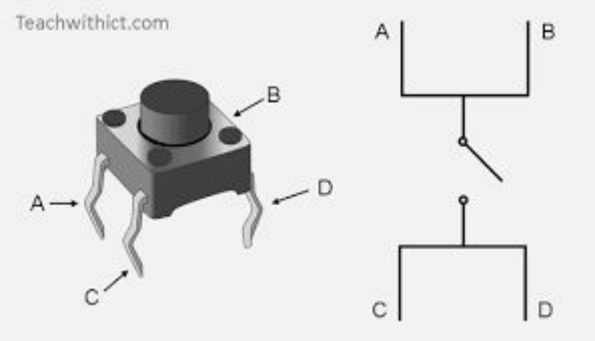
As a result we can use a smaller resistor when working with an LED i.e. 220 Ohms rather than our normal 330 Ohms on the Arduino.

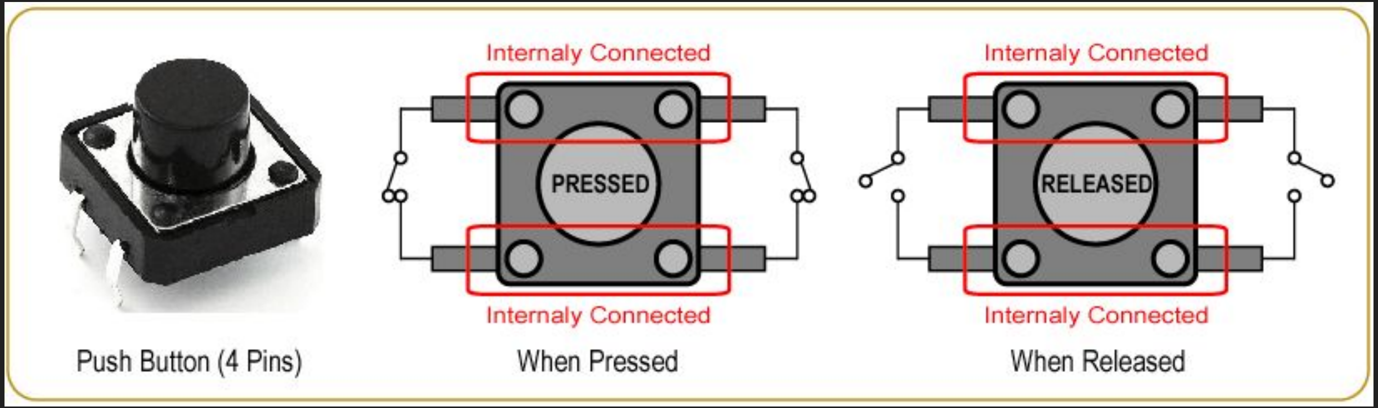
Reference Guide



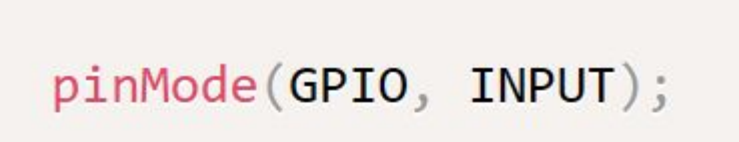
GPIO	Input	Output	Notes
0	pulled up	OK	outputs PWM signal at boot, must be LOW to enter flashing mode
1	TX pin	OK	debug output at boot
2	OK	OK	connected to on-board LED, must be left floating or LOW to enter flashing mode
3	OK	RX pin	HIGH at boot
4	OK	OK	
5	OK	OK	outputs PWM signal at boot, strapping pin
6	x	x	connected to the integrated SPI flash
7	x	x	connected to the integrated SPI flash
8	x	x	connected to the integrated SPI flash
9	x	x	connected to the integrated SPI flash
10	x	x	connected to the integrated SPI flash
11	x	x	connected to the integrated SPI flash
12	OK	OK	boot fails if pulled high, strapping pin
13	OK	OK	
14	OK	OK	outputs PWM signal at boot
15	OK	OK	outputs PWM signal at boot, strapping pin
16	OK	OK	
17	OK	OK	
18	OK	OK	
19	OK	OK	
21	OK	OK	
22	OK	OK	
23	OK	OK	
25	OK	OK	
26	OK	OK	
27	OK	OK	
32	OK	OK	
33	OK	OK	
34	OK		input only
35	OK		input only
36	OK		input only
39	OK		input only

Push Button-Internal Wiring

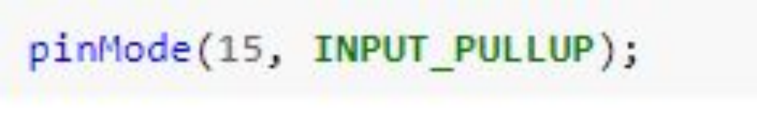




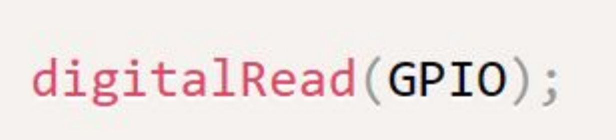
Esp32 Digital Inputs



```
pinMode(GPIO, INPUT);
```



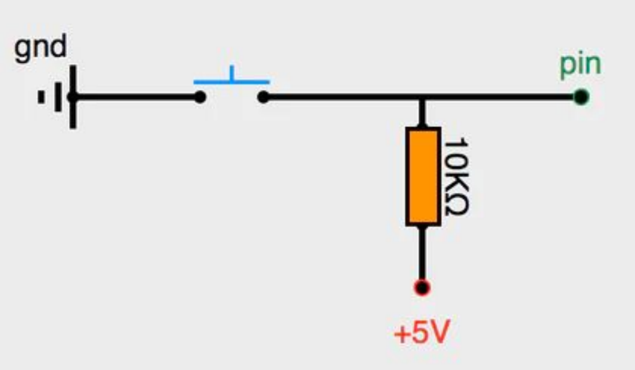
```
pinMode(15, INPUT_PULLUP);
```

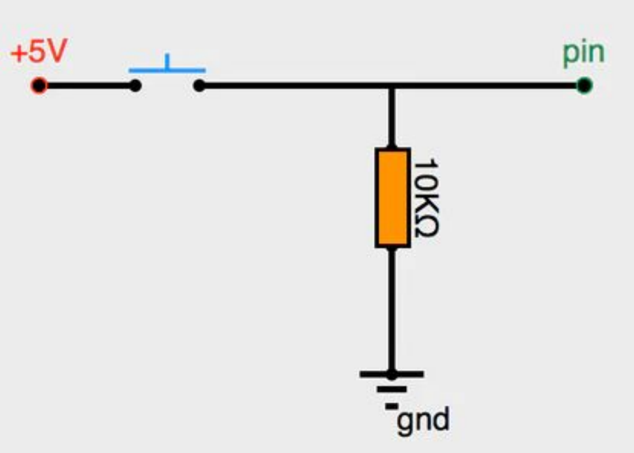


```
digitalRead(GPIO);
```

All pins except 6-11 can be used for input.

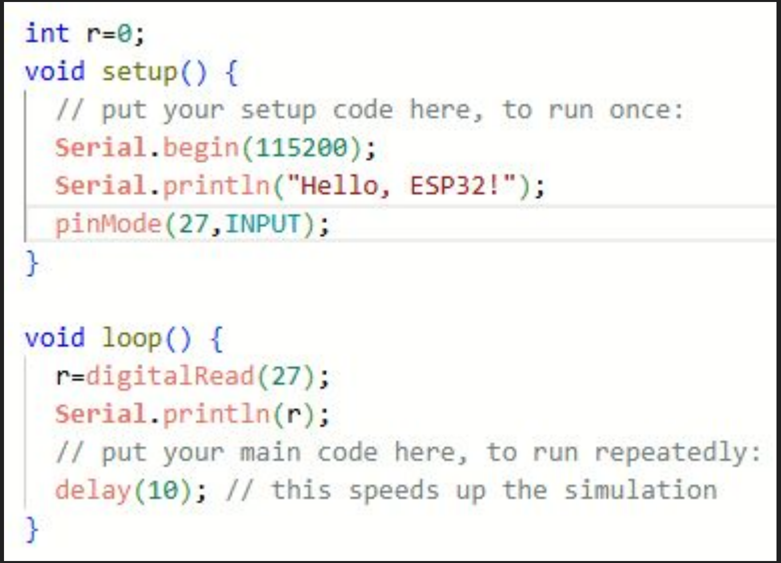
The ESP32 has internal pull up resistors so there is no need to add them into your circuit. This means your signal is HIGH until the button is pressed (and connected to GND) at which point signal goes LOW.
When setting up an input pin use the following to generate an input of 0 on a button until pressed in which case a reading of 1 is delivered.
pinMode(15, INPUT_PULLDOWN);





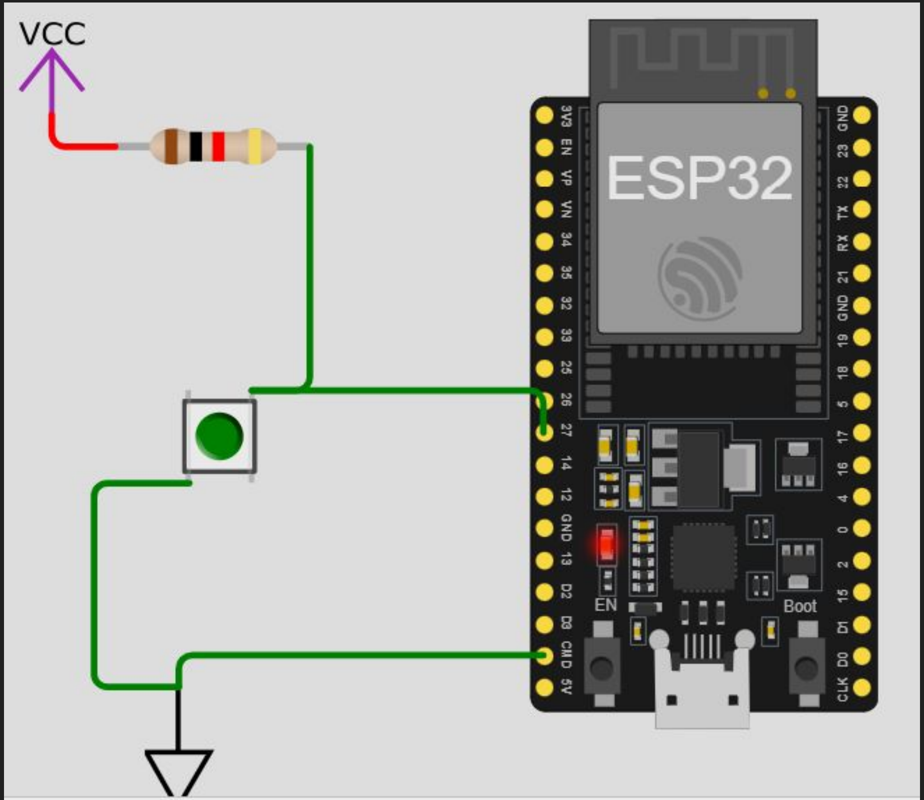
ESP32 with INPUT

Must include your own pullup or pulldown resistor. In this example I include a pullup resistor and so 1 until button is pressed. Always connect ESP GND to common ground connections.



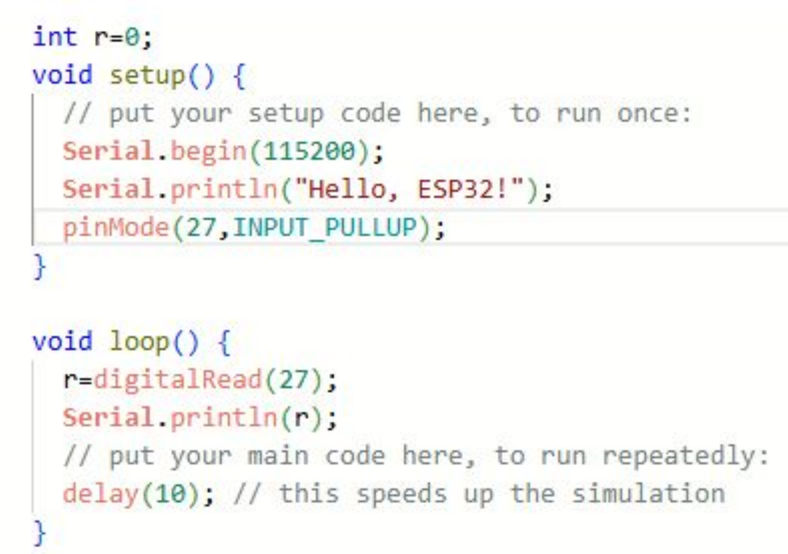
```
int r=0;
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  Serial.println("Hello, ESP32!");
  pinMode(27,INPUT);
}

void loop() {
  r=digitalRead(27);
  Serial.println(r);
  // put your main code here, to run repeatedly:
  delay(10); // this speeds up the simulation
}
```



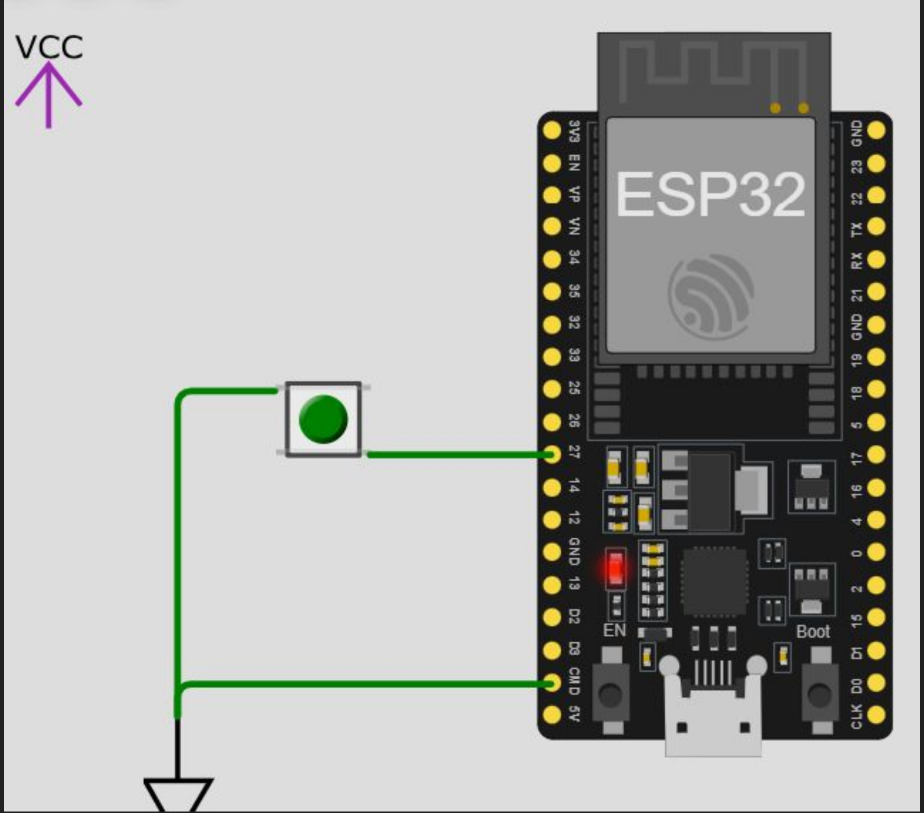
ESP32 with INPUT_PULLUP

1 until button is pressed



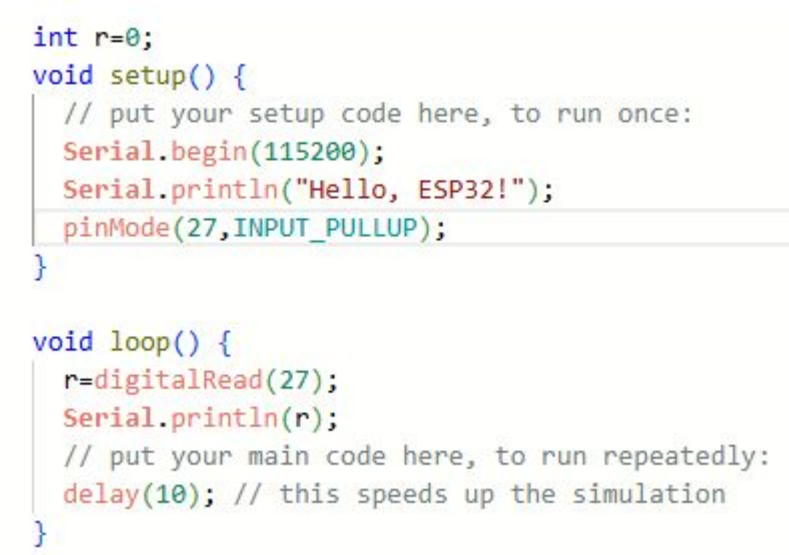
```
int r=0;
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  Serial.println("Hello, ESP32!");
  pinMode(27,INPUT_PULLUP);
}

void loop() {
  r=digitalRead(27);
  Serial.println(r);
  // put your main code here, to run repeatedly:
  delay(10); // this speeds up the simulation
}
```



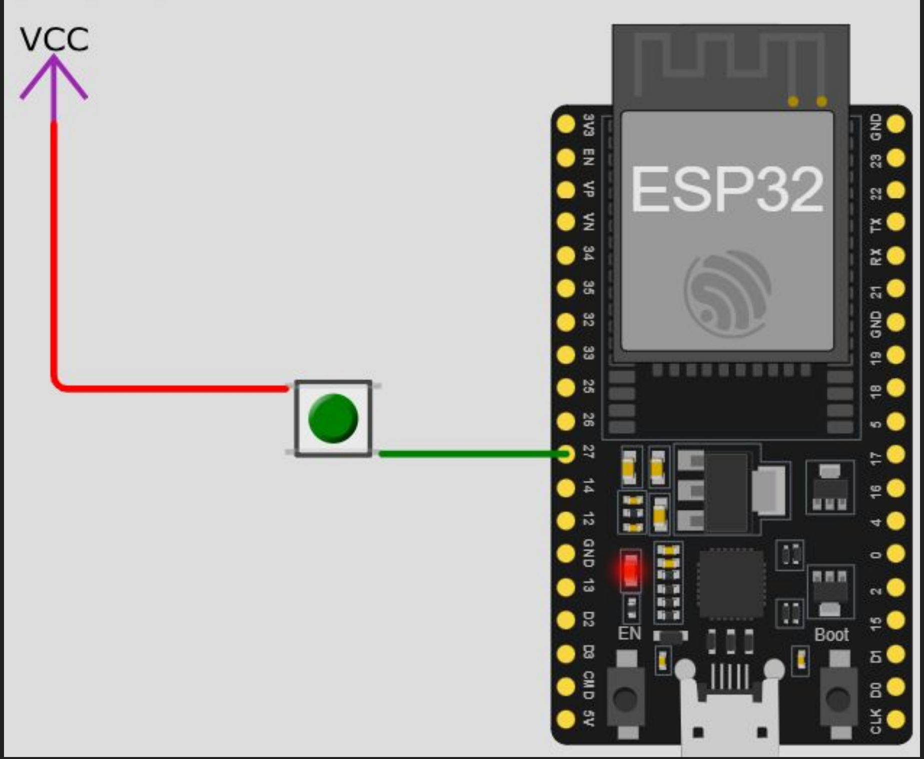
ESP32 with INPUT_PULLDOWN

0 until button is pressed

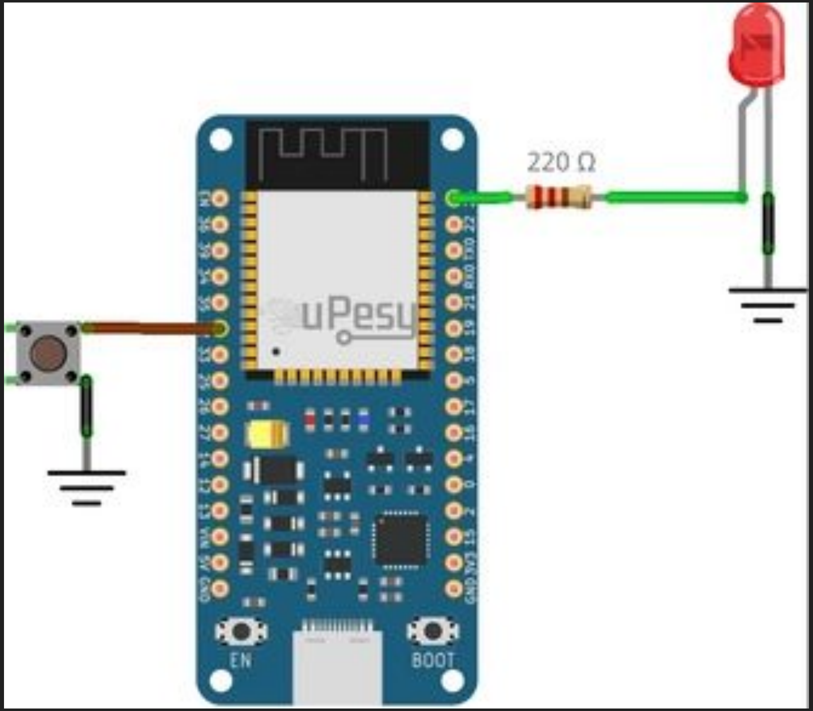


```
int r=0;
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  Serial.println("Hello, ESP32!");
  pinMode(27,INPUT_PULLUP);
}

void loop() {
  r=digitalRead(27);
  Serial.println(r);
  // put your main code here, to run repeatedly:
  delay(10); // this speeds up the simulation
}
```

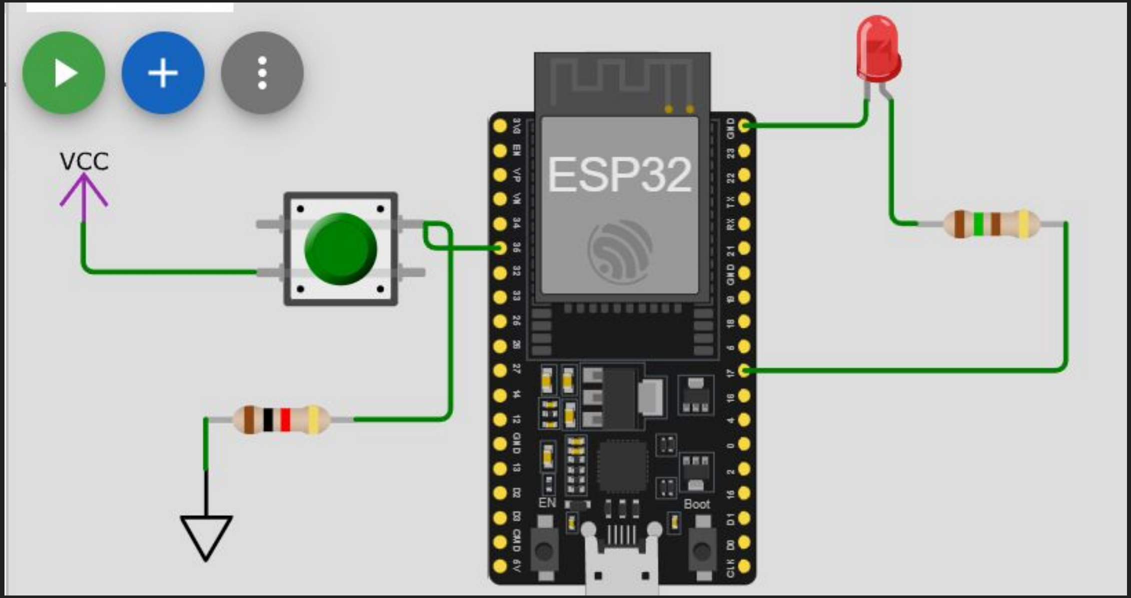


Standard Wiring



Buggy Results

You may get buggy results when avoiding adding in your own pull up or pull down resistors. If so, add them in yourself. As shown in the following image:



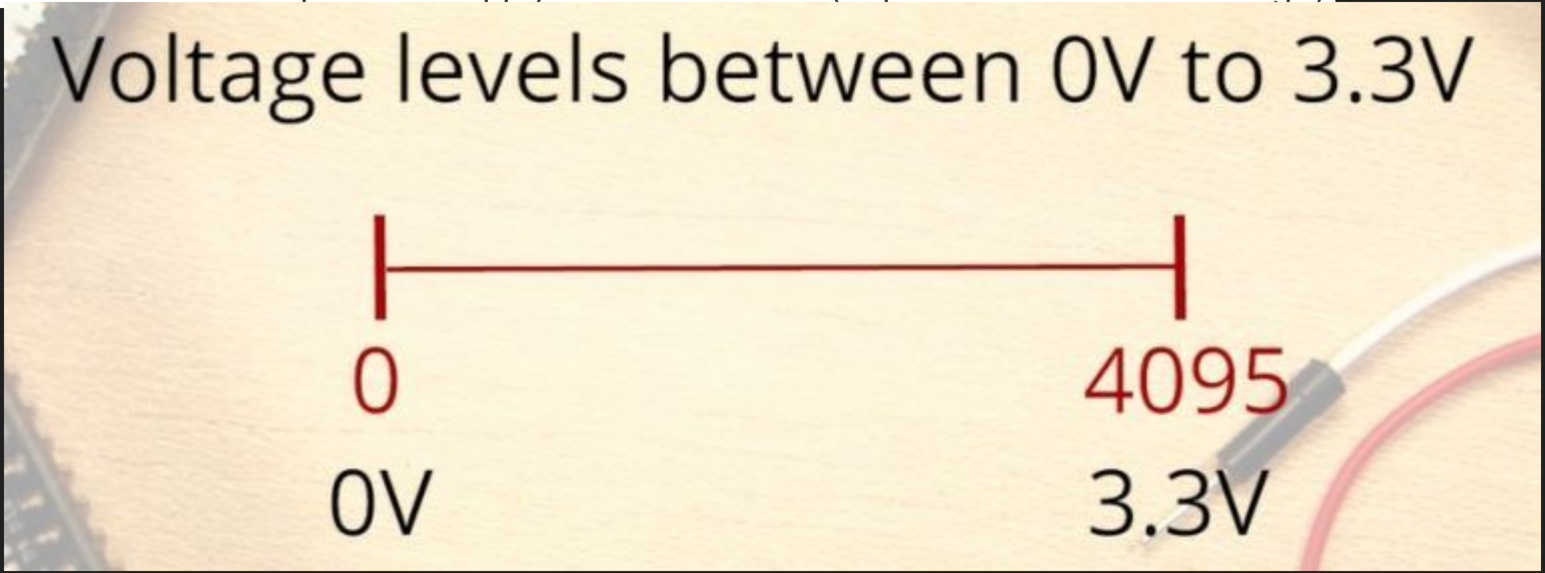
Note, a resistor of 1K to 10K should be used when pulling up or down your input pin.

Use INPUT in the pinMode when using your own.

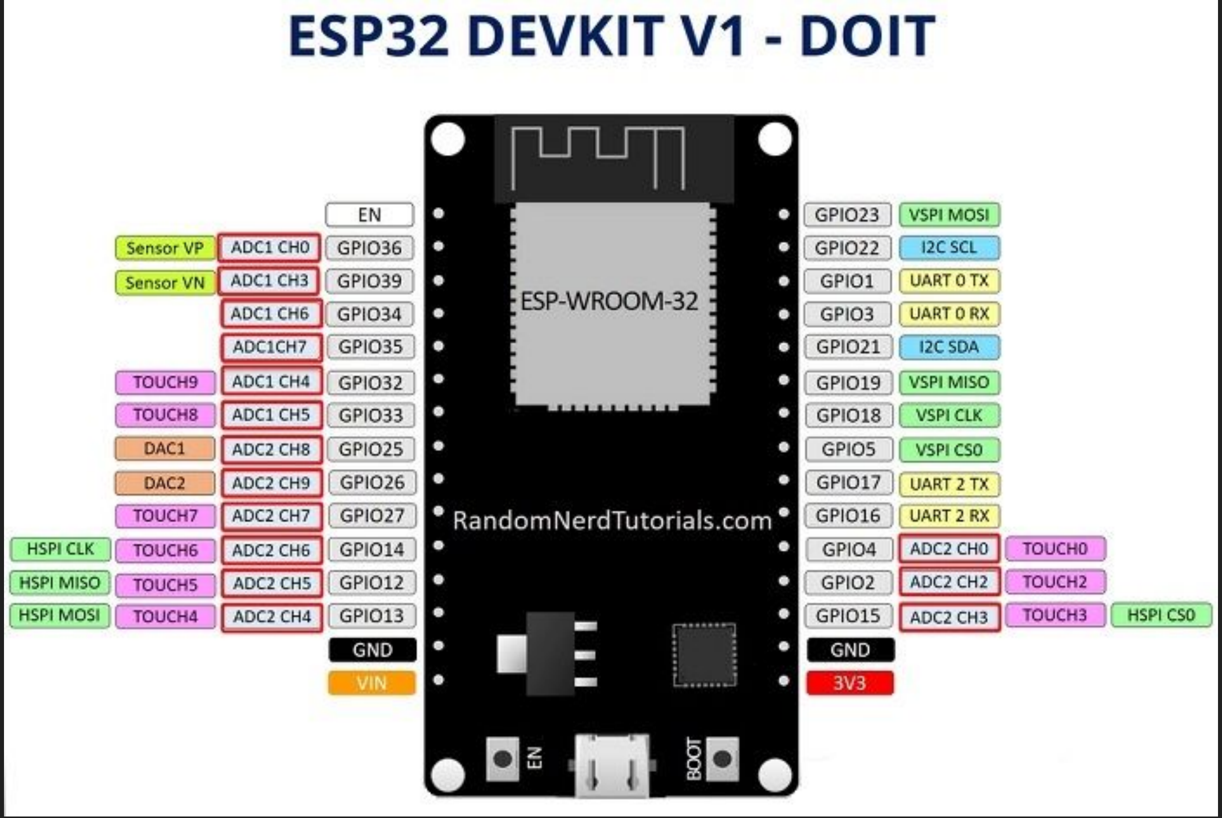
Analog Inputs

You can read analog values (which means being able to read varying voltage levels between 0 and 3.3V). These values range from 0 to 4095 as they are 12 bit analog to digital converters. Note these values have some range of error. For example 0V and 0.1V might register a 0. The Vin, 3.3V, and 5V pins are used to input power to the ESP32 (the ESP32 operates on 3.3V and so using Vin or the 5V pin will have its voltage changed by an onboard regulator) . Do not use these for power source but instead attach an output pin set HIGH to supply 3.3V to a component like a potentiometer.

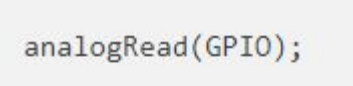
Also note that ESP32 pins can supply a max of 600mA.(depends on the board though).



Pinout
ADC-indicates an analog to digital converter capable pin

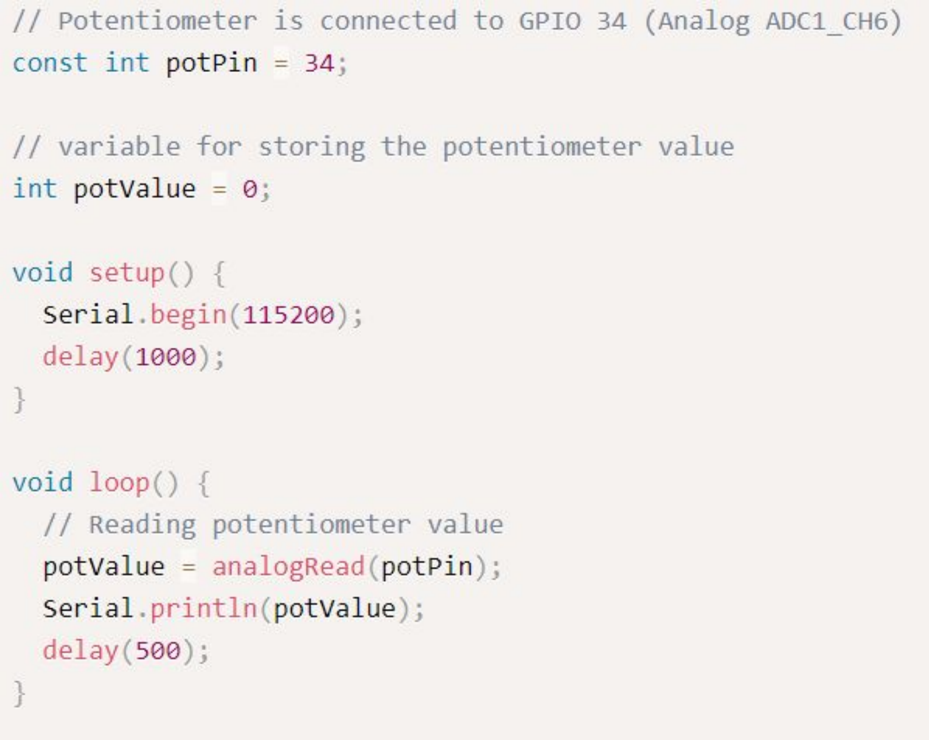


Code



```
analogRead(GPIO);
```

Note that there is no need to set the pinMode on analog input pins.

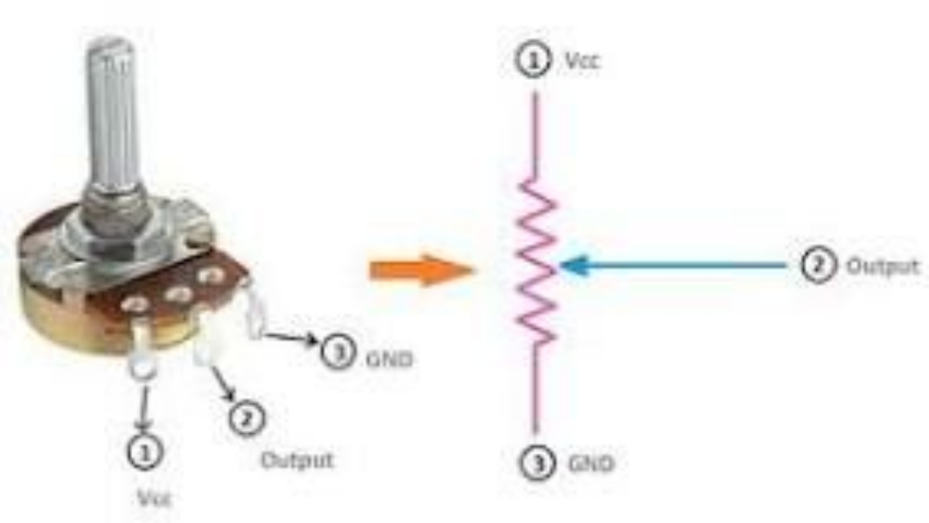


```
// Potentiometer is connected to GPIO 34 (Analog ADC1_CH6)
const int potPin = 34;

// 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);
}
```



POT as a Voltage Divider

Review: Voltages drop across resistors in proportion to their value relative to the total resistance in the circuit branch they are part of. For example, in the following circuit there are 2 resistors in a 12V circuit. The first resistor will drop the voltage 5/15 of 12V while the second drops its 10/15 of 12V.

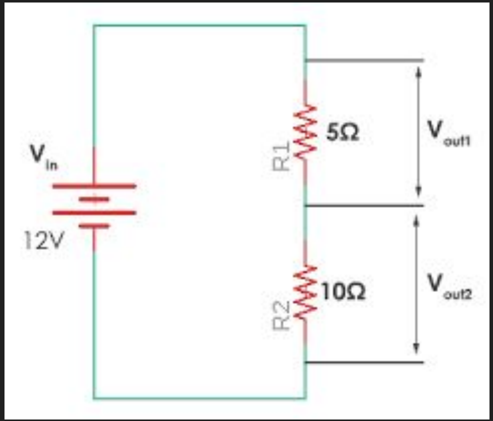
Math:

5/15 * 12 =4V

10/15 * 12 =8V

This means that after the first resistor there is 8V left. If we stuck a wire there to an input pin we could measure the 8V (technically esp32 can only handle 3.3V so the source voltage should be 3.3V).

The POT is used to change the proportion of the first and second, thus changing the voltage coming out of the centre pin.



You could also use Ohm’s law to calculate the voltage across each resistor:

Ohms law:

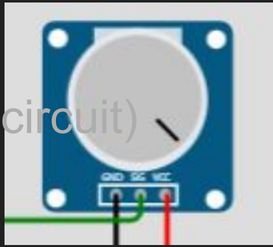
Rt=15 Ohms

I=V/R=12V/15 Ohms=0.8A

Vr1=0.8x5=4V

V42=0.8x10=8V

Questions and Exercises
(create and test the following in wokwi and then build and test it using a live circuit)



1. Go to https://lastminuteengineers.com/esp32-pinout-reference/#:~:text=RTC%20GPIO%20Pins&text=These%20pins%20are%20used%20to,as%20external%20wake%20up%20sources and briefly describe the following pins: RTC, EN, VIN, 3.3V, GND, Input Only GPIO pins, ADC, DAC, Touch, I2C, SPI, UART and PWM pins.
2. Create a circuit that blinks an LED on and off every second using pin 23 of the esp32. https://wokwi.com/projects/432675165505158145
3. Create a circuit that has a button and an LED. Only when the light is pressed does the LED light up and stay on. https://wokwi.com/projects/432675589966665729
4. Modify the last circuit so that when the same button is pressed again the LED turns off. (read https://esp32io.com/tutorials/esp32-button-debounce to learn about debouncing a button-you may also want to add a 50ms in each literation to help alleviate issues with button response) https://wokwi.com/projects/432676116989391873
5. Using the same circuit create a new program so that every time you click the button the LED flashes at a new random speed. https://wokwi.com/projects/432676666939258881
6. Create a new program so that only when the button is pressed 3 times will the LED light up. https://wokwi.com/projects/432677124762266625
7. Create two new circuits that use 2 buttons to control two LEDs. One button is used with a pull down resistor and the other with a pull up resistor. https://wokwi.com/projects/432689619856059393
8. Create a circuit in WOKWI using a POT to implement the code in the previous slide. Print out the range of values returned by calling analogRead() on the POT SIG pin. https://wokwi.com/projects/432689844012273665

1. Go to https://lastminuteengineers.com/esp32-pinout-reference/#:~:text=RTC%20GPIO%20Pins&text=These%20pins%20are%20used%20to,as%20external%20wake%20up%20sources and briefly describe the following pins: RTC, EN, VIN, 3.3V, GND, Input Only GPIO pins, ADC, DAC, Touch, I2C, SPI, UART and PWM pins.

Power & Control Pins

- RTC: GPIOs usable in deep sleep mode; wake-up capable.
- EN: Chip reset/enable pin. Pull low to reset.
- VIN: Supplies voltage (5–14V) to ESP32 via regulator.
- 3.3V: Outputs 3.3V from onboard regulator.
- GND: Ground connection.

GPIO Types

- Input-Only GPIOs: GPIOs 34–39; can't be outputs.
- ADC: Analog input pins (GPIOs 0, 2, 4, 12–15, 25–27, 32–39).
- DAC: Analog output on GPIO25 & GPIO26.
- Touch: Capacitive touch sensing GPIOs.

Communication Pins

- I2C: Default GPIO21 (SDA), GPIO22 (SCL); software reassignable.
- SPI: Default GPIOs 23 (MOSI), 19 (MISO), 18 (SCK), 5 (CS).
- UART: Serial comms, default TX (GPIO1), RX (GPIO3).

PWM

- Available on all output-capable GPIOs; used for dimming LEDs, motors, etc.