

# Lecture 15

## ESP32 and Arduino

# Agenda

- 01. ESP32 Overview
- 02. Arduino Software
- 03. PicoKit ESP32 Hardware Caveats
- 04. Level Shifting 3.3V using MOSFETS

01

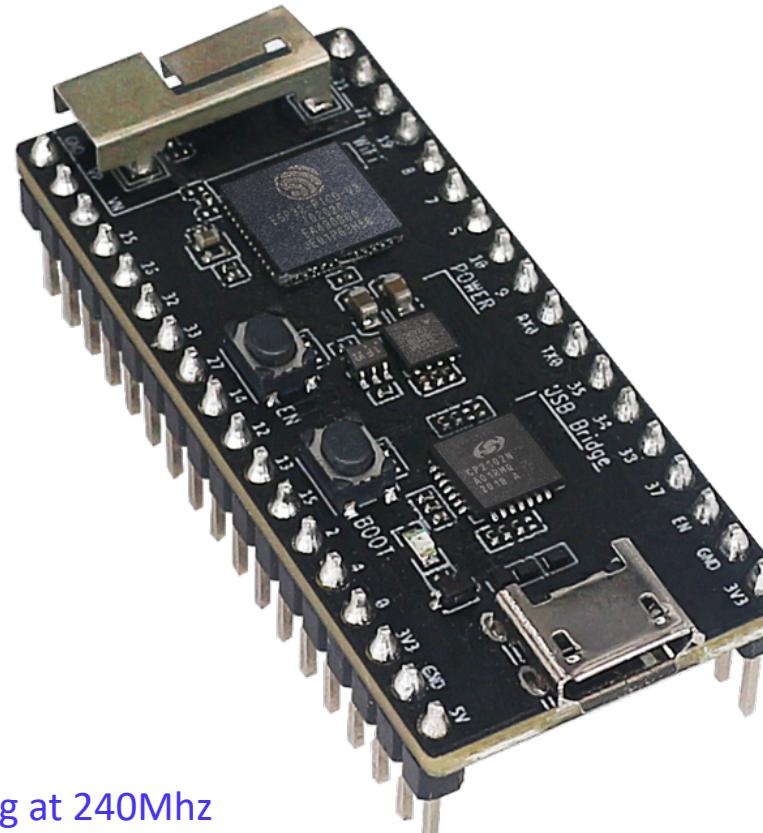
# ESP32 Overview

# Typical important functions on a microcontroller

- Computation (integer, FPU, single core, dual core)
- Memory (flash, RAM, EEPROM)
- Inputs (digital, analog)
- Outputs (digital, analog)
- Timers (input capture, output compare, PWM)
- Analog to Digital Converter (higher resolution, faster is better)
- Communication (USB, UART, SPI, I2C, wireless)

# ESP32

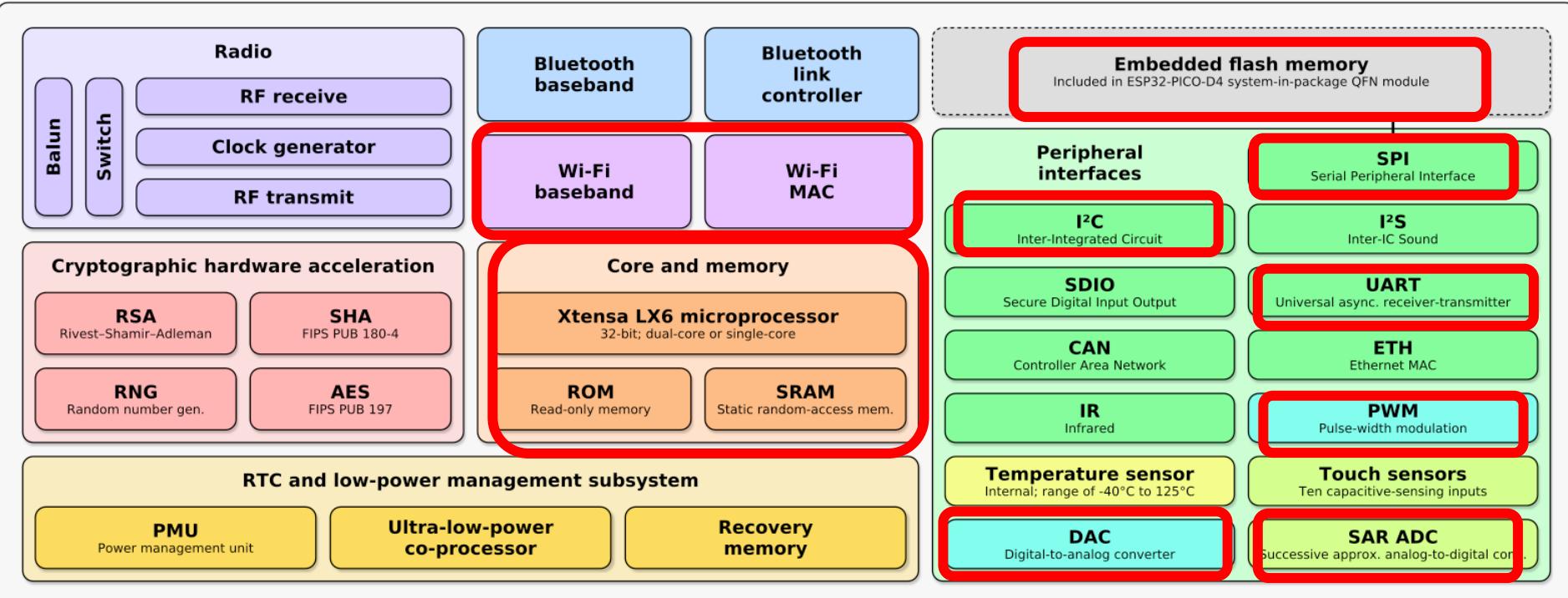
- ESP32 *Wifi Module plus Bluetooth*
  - ~50 foot range
- Operating voltage = 3.3V
- USB interface
- 520 KiB SRAM, 4MB of flash memory
- 160 MHz clock speed (**dual** core)
- Floating point coprocessor



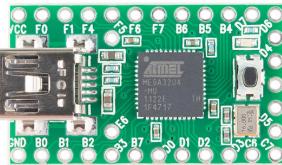
On ESP32 S2 no Bluetooth, no FPU, single core running at 240Mhz

# ESP32 Block Diagram

Espressif ESP32 Wi-Fi & Bluetooth Microcontroller — Function Block Diagram



# ATmega 32U4 VS ESP32

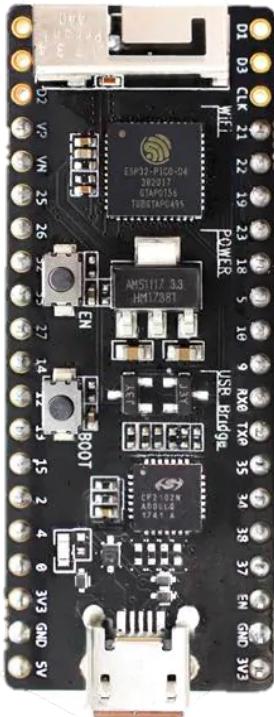


	ATMEGA 32U4	ESP32	ESP32 Difference
Processor Speed	16MHz, 8bit RISC	160Mhz, 32bit RISC	<b>~20x faster comp</b>
Memory	32k Flash; 2.5K RAM	4M Flash; 512K RAM	<b>&gt;100x's more</b>
GP Input/Output	21 @ 5V	17 to 32 @ 3.3V	<b>Similar</b>
IO Current	20 mA (40 mA max)	10mA (20-40mA max)	<b>Uncertain</b>
ADC	12 channels 10-bit	16 channels 12-bit	<b>33% More hi-res</b>
Communication	SPI/UART/I2C	SPI/I2S/UART/I2C/BT/...	<b>Much more</b>
cost	\$20	\$10	<b>half the price</b>

# ESP32 Board Variants

~\$10

No controllable LED



PicoKit

~\$10

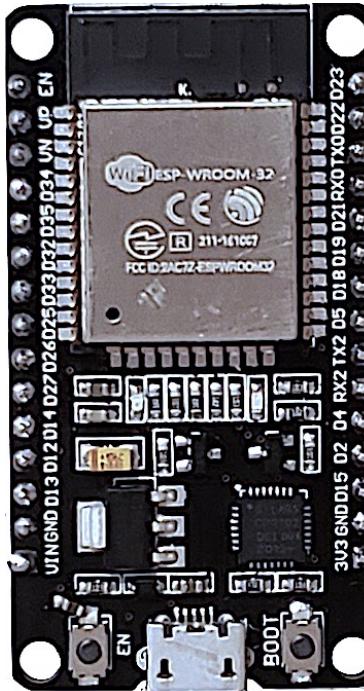
Pin labels underneath



HiLetGo  
NodeMCU

Cheapest, @ ~\$7

Too wide for protoboard  
(only one side exposes pins)



Dorhea  
NodeMCU

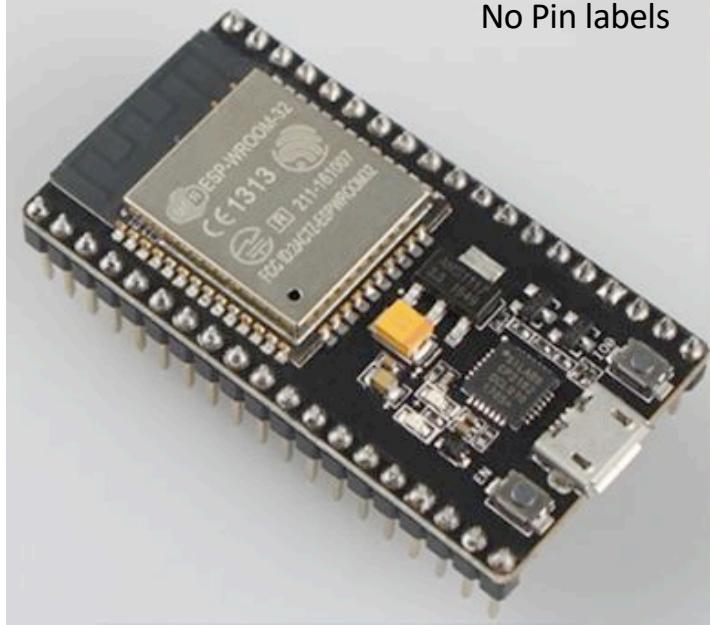
Newer, better board \$8  
single core, no FPU, no BT



ESP32 S2  
Saola 1R

# ESP32 Board Problems

- HiLetGo Board labeling
- Boards too wide.

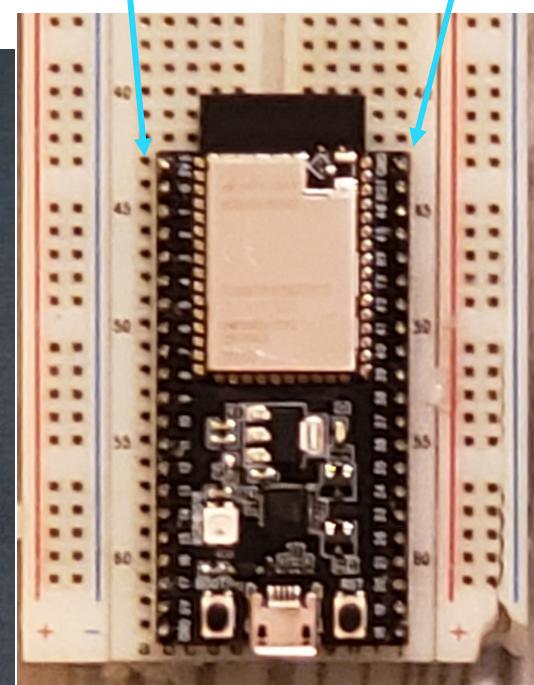


No Pin labels



Pin labels

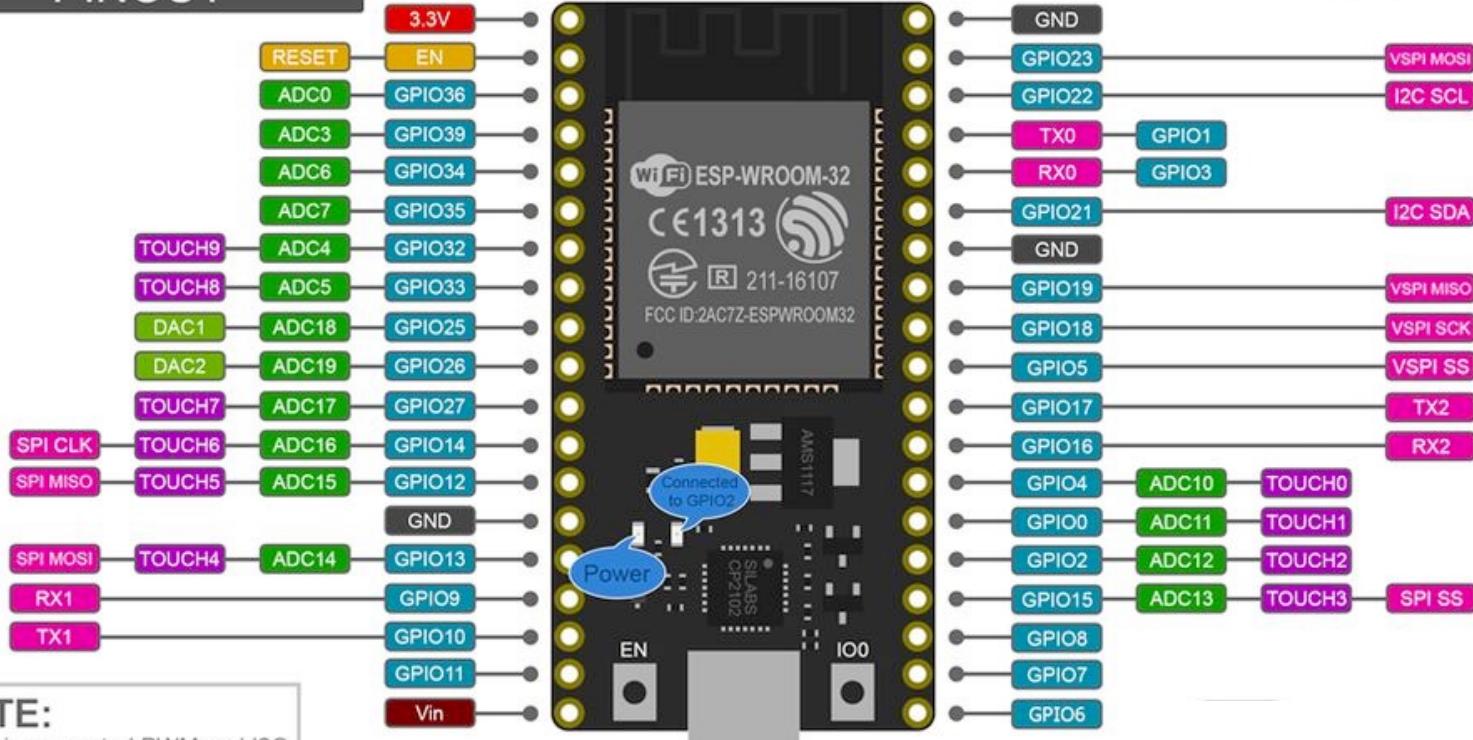
Only 1 column accessible  
No pins accessible



# NodeMCU-32S

## PINOUT

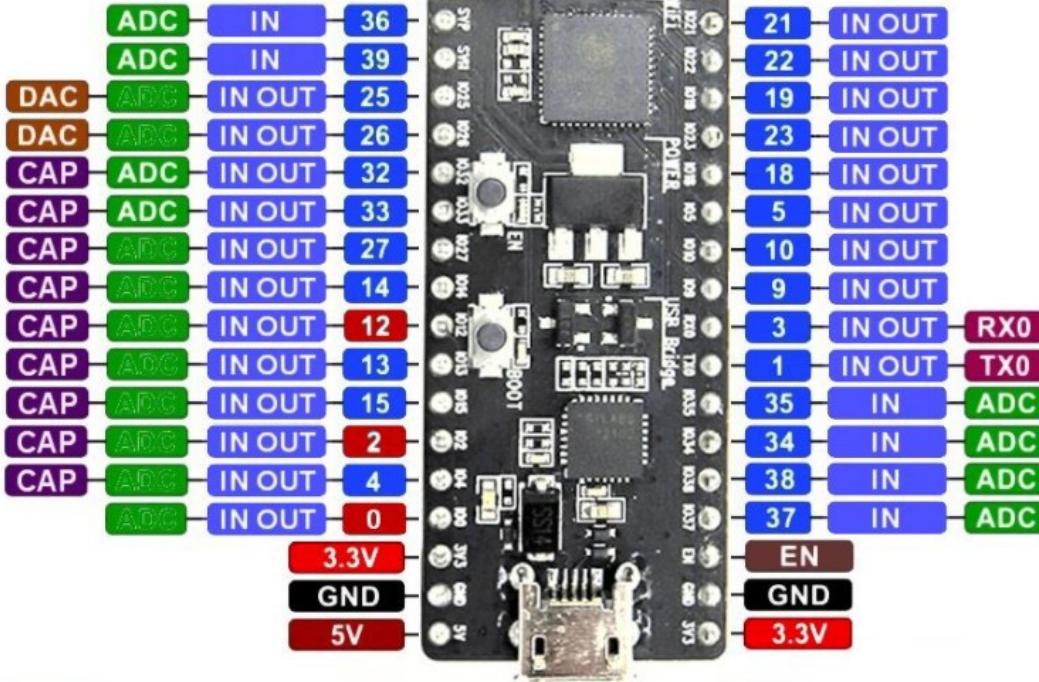
GPIO = General Purpose Input/Output  
GND = ground or negative power line  
VIN = positive power (5V typically)



### NOTE:

All pin supported PWM and I2C  
Pin current 6mA (Max. 12mA)

# Pico-kit v4 ESP32 Pinout



**IN** = Inputs only, and no pullups

**IN OUT** = DigIn / Counter / Period / Encoder  
DigOut / Pwm / Servo

**0** = Pin "0" must be high at startup and low while programming

**2** = Pin "2" must be low while programming

**12** = Pin "12" must be low at startup and while programming

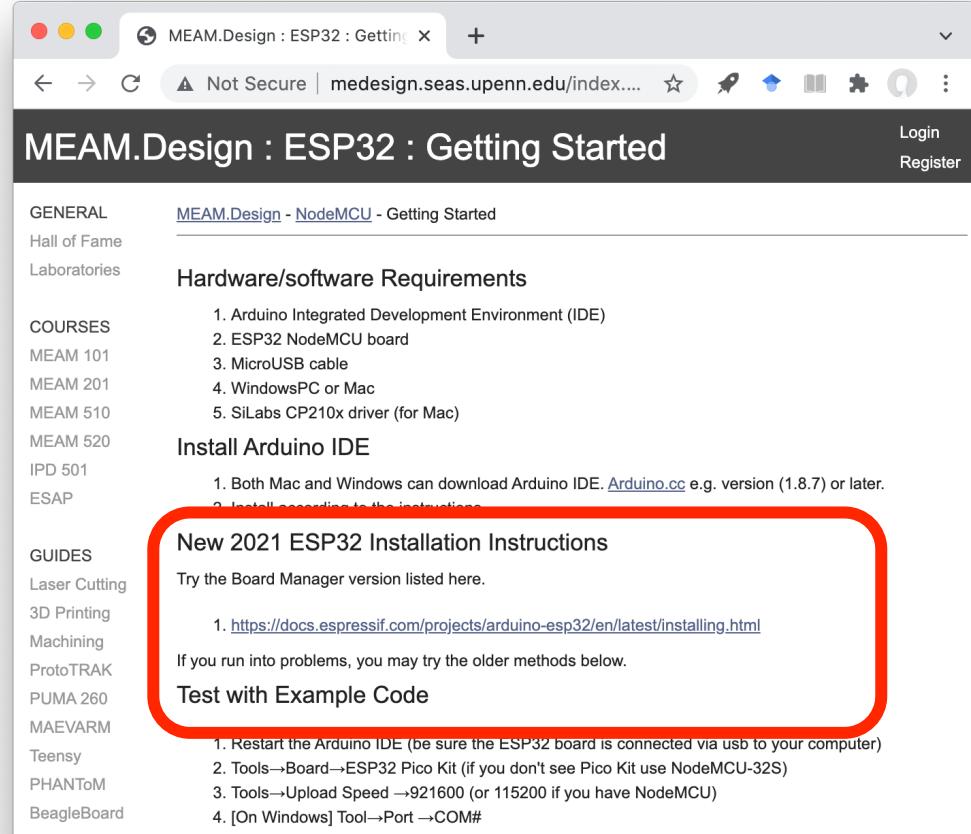
**ADC** = Adc when WiFi is not active

**RX0** **TX0** = Used by serial debugger

02

# Software - Arduino

# Arduino Set-up



MEAM.Design : ESP32 : Getting Started

GENERAL Hall of Fame Laboratories

COURSES MEAM 101 MEAM 201 MEAM 510 MEAM 520 IPD 501 ESAP

GUIDES Laser Cutting 3D Printing Machining ProtoTRAK PUMA 260 MAEVARM Teensy PHANToM BeagleBoard

**New 2021 ESP32 Installation Instructions**

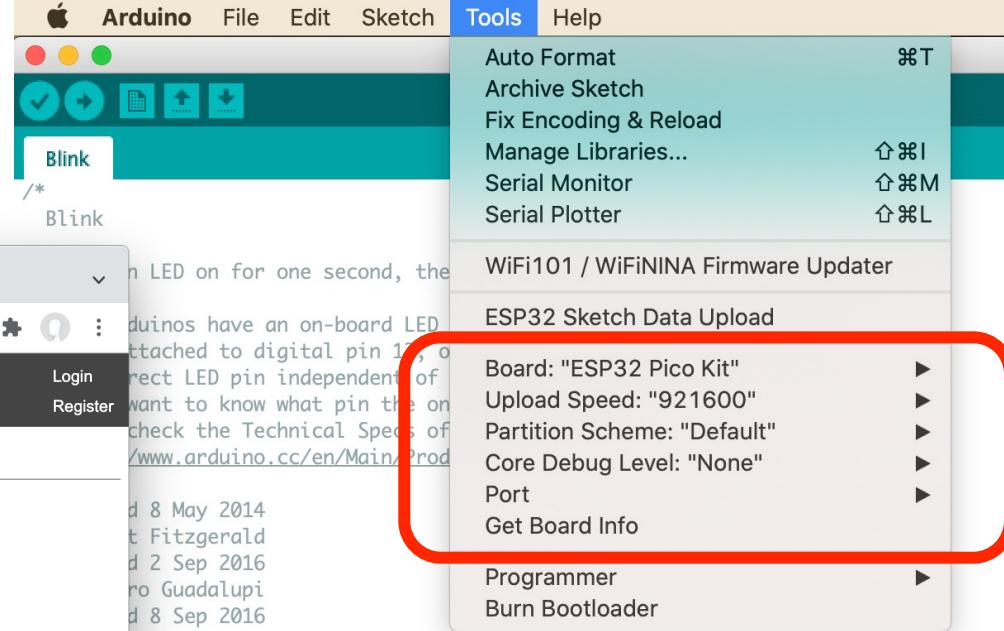
Try the Board Manager version listed here.

- <https://docs.espressif.com/projects/arduino-esp32/en/latest/installing.html>

If you run into problems, you may try the older methods below.

**Test with Example Code**

- Restart the Arduino IDE (be sure the ESP32 board is connected via usb to your computer)
- Tools→Board→ESP32 Pico Kit (if you don't see Pico Kit use NodeMCU-32S)
- Tools→Upload Speed→921600 (or 115200 if you have NodeMCU)
- [On Windows] Tool→Port→COM#



Arduino File Edit Sketch Tools Help

Auto Format Archive Sketch Fix Encoding & Reload Manage Libraries... Serial Monitor Serial Plotter WiFi101 / WiFiNINA Firmware Updater ESP32 Sketch Data Upload

Board: "ESP32 Pico Kit"  
Upload Speed: "921600"  
Partition Scheme: "Default"  
Core Debug Level: "None"  
Port  
Get Board Info

Programmer  
Burn Bootloader

<http://medesign.seas.upenn.edu/index.php/Guides/ESP32-gettingstarted>

# Arduino Code Breakdown

Every Arduino “Sketch” (essentially c++ program):

setup() runs once [first]

loop() repeats over and over.

```
void setup() {  
    // put your setup code here, to run once:  
    // These are comments “//”  
}  
  
void loop() {  
    // put your main code here, to run repeatedly:  
}
```

# "Hello World" in Arduino

1. Open:  
<Files-> Examples -> 01.Basic ->  
BareMinimum>

2. Add the lines:

    Serial.begin(115200);

in `setup()` section.

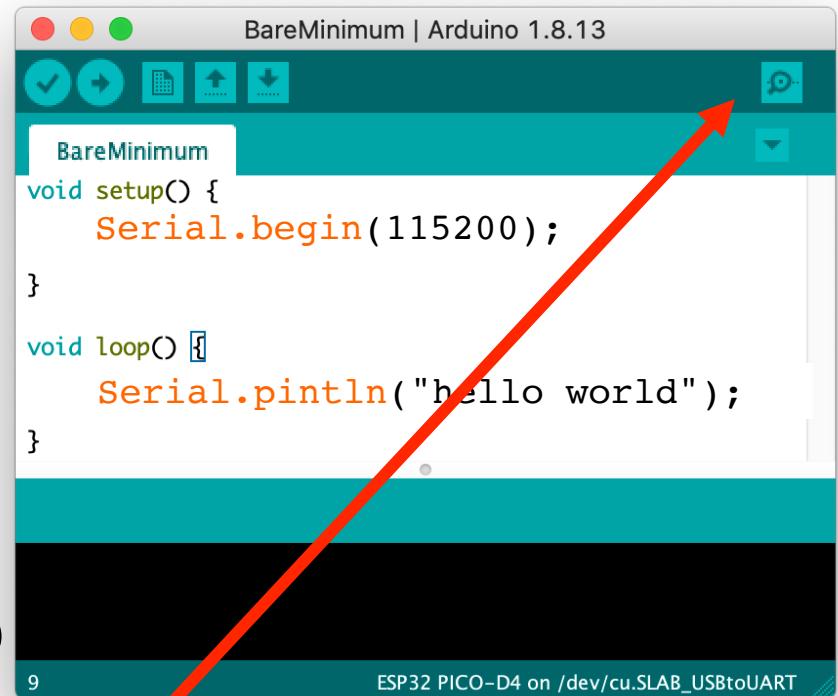
    Serial.println("hello world")

in `loop()` section.

3. Click (✓) <Verify> to compile

4. Click (→) <Upload> to Flash code to  
Arduino.

5. Click (magnifying glass) <serial  
monitor> (upper rt corner)



```
BareMinimum
void setup() {
    Serial.begin(115200);
}

void loop() {
    Serial.println("hello world");
}
```



# Restoring OscilloSorta to ESP32

- Same steps except find:
  - Oscillosorta folder in canvas -> files -> resources
- Copy whole folder (all files inside) to your Arduino directory
- Double click *OscilloSorta1\_3.ino*
- Arduino requires .ino file to be inside a folder of the same name
- Click (➔) <Upload> to Flash code to Arduino.

**Q1. In public chat: list 5 functions we use with Teensy pins**

# Command Summary

Teensy
set/clear (DDRx, pin#)
set/clear (PORTx, pin#)
bit_is_clear/set(PINx,pin#)
ADC routines...
Timer routines...
m_usb_init()
m_usb_tx_string(X)
_delay_ms()

## ESP32 Arduino

- `pinMode(pin#, INPUT/OUTPUT)`
- `digitalWrite(pin#, HIGH/LOW)`
- `digitalRead(pin#)`
- `analogRead(pin#)`
- ~~`analogWrite(pin#, value/255)`~~  
Not on ESP32
- ledc routines... (will cover later)
- `dacWrite(pin#)`
- `Serial.begin(baudrate)`
- `Serial.println("text")`
- `delay(ms)`

# Arduino Blink Sketch (like Lab 1)

*LED\_BUILTIN doesn't exist on pico*

```
void setup() {    LED_BUILTIN doesn't exist on pico
    pinMode(LED_BUILTIN, OUTPUT); // Initialize the LED_BUILTIN pin as an output
}

// the loop function runs over and over again forever
void loop() {
    digitalWrite(LED_BUILTIN, LOW);      // Turn the LED on
    delay(1000);                      // Wait for a second
    digitalWrite(LED_BUILTIN, HIGH);     // Turn the LED off
    delay(2000);                      // Wait for two secs
}
```

# Functions: in Blink

```
pinMode(pin,OUTPUT);
```

- Tells the micro that pin is an output pin (sets data direction register).
- Modes can be INPUT, OUTPUT, INPUT\_PULLUP, INPUT\_PULLDOWN

Available on ESP32,  
not most other MCUs

```
digitalWrite(pin,HIGH);
```

- Sets the state of digital pin # pin to HIGH (3.3V), LOW=0V,
- pin is the GPIO number, can be 0, 1, 2... 39 for ESP32
- Similar to set(PORT,PIN); and clear(PORT,PIN);

```
delay(1000);
```

- Delays further execution of code for 1000 milliseconds. (Busy waits)

# Digital Input/Output States

Three possible  
internal states:

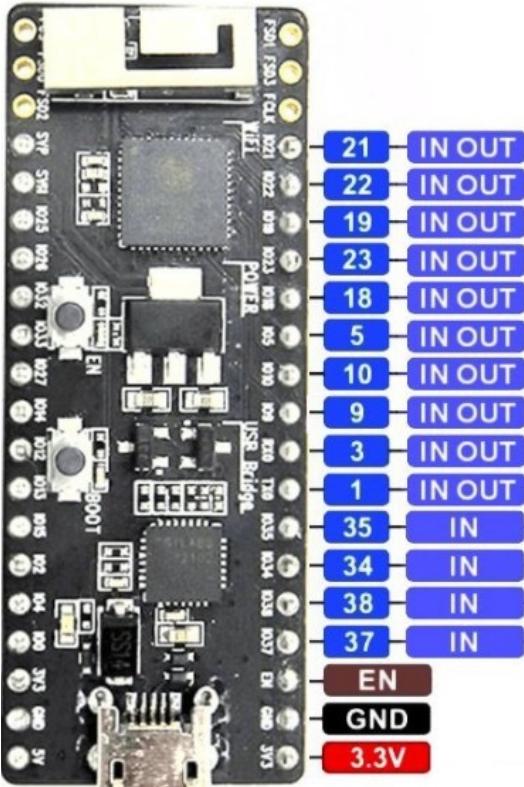
1. HIGH
2. LOW
3. Tri-state (high-  
impedance)

ESP32  
only

PINMODE	Internal	External	Result
OUTPUT	High	High	High
OUTPUT	High	Low	Contention
OUTPUT	High	Float	High
OUTPUT	Low	High	Contention
OUTPUT	Low	Low	Low
OUTPUT	Low	Float	Low
INPUT	Float	High	High
INPUT	Float	Low	Low
INPUT	Float	Float	unknown
INPUT_PULLUP	Float	High	High
INPUT_PULLUP	Float	Low	Low
INPUT_PULLUP	Float	Float	High
INPUT_PULLDOWN	Float	High	High
INPUT_PULLDOWN	Float	Low	Low
INPUT_PULLDOWN	Float	Float	Low

# How do we know what to call a pin?

*In Arduino/hardware/espressif/esp32/variants/pico32/ pins\_arduino.h*



```
#ifndef Pins_Arduino_h
#define Pins_Arduino_h

#include <stdint.h>

#define EXTERNAL_NUM_INTERRUPTS 16
#define NUM_DIGITAL_PINS        40
#define NUM_ANALOG_INPUTS       16

#define analogInputToDigitalPin(p) (((p)<20)?(esp32_adc20bit(p)):((p)<40)?(p):-1)
#define digitalPinToInterrupt(p) ((p)<34)?(p):-1
#define digitalPinHasPWM(p)      (p < 34)

static const uint8_t TX = 1;
static const uint8_t RX = 3;

static const uint8_t SDA = 21;
static const uint8_t SCL = 22;
```

# Other Functions

`digitalRead(pin);`

- Returns the state of digital pin # pin as **HIGH** or **LOW**,
- pin can be 0, 1, 2 ... 39

`delayMicroseconds(x);`

- Delays further execution of code for x microseconds.

`Serial.begin(115200);`

- Needed for serial comms.
- 115200 is the baudrate. 9600 is another standard rate.

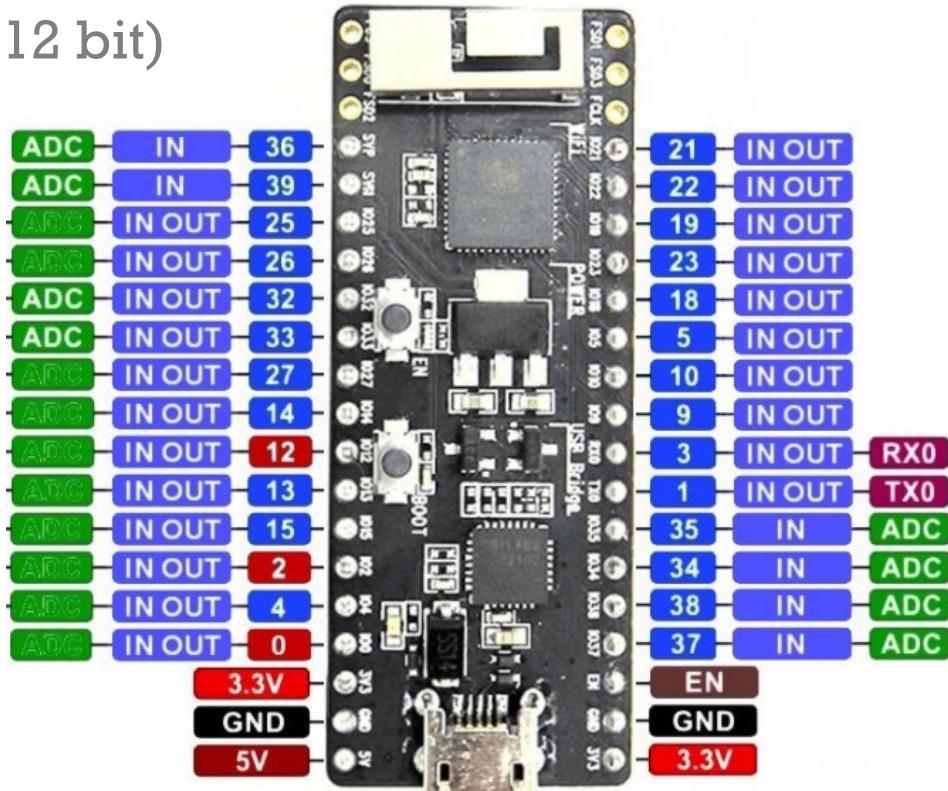
`Serial.println(x); Serial.print("string");`

- `println()` with carriage return, `print()` without carriage return.

# analogRead(pin)

- Reads the ESP32 ADC (one of 16 channels) associated with pin.
- Returns an integer from 0 – 4095 (12 bit)

```
void setup() {  
    Serial.begin(115200);  
}  
  
void loop() {  
    Serial.println(analogRead(A0));  
    delay(200);  
}
```



`analogRead(), map(value, inL, inH, outL, outH)`

- Can use arduino `map()` to convert linear units. E.g. from ADC counts (0-4095) to millivolts (0-3300).

<https://www.arduino.cc/reference/en/language/functions/math/map/>

The screenshot shows a web browser displaying the Arduino Reference website at <https://www.arduino.cc/reference/en/language/functions/math/map/>. The page title is "map()". The left sidebar has a tree menu with "FUNCTIONS" selected. The main content area shows the "map()" function documentation, which re-maps a number from one range to another. It includes a description, examples, and notes about the function's behavior.

Reference > Language > Functions > Math > Map

## map()

[Math]

### Description

Re-maps a number from one range to another. That is, a value of `fromLow` would get mapped to `toLow`, a value of `fromHigh` to `toHigh`, values in-between to values in-between, etc.

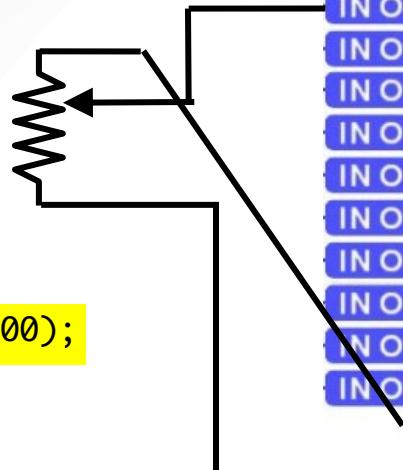
Does not constrain values to within the range, because out-of-range values are sometimes intended and useful. The `constrain()` function may be used either before or after this function, if limits to the ranges are desired.

The Arduino Reference text is licensed under a [Creative Commons Attribution-Share Alike 3.0 License](#).

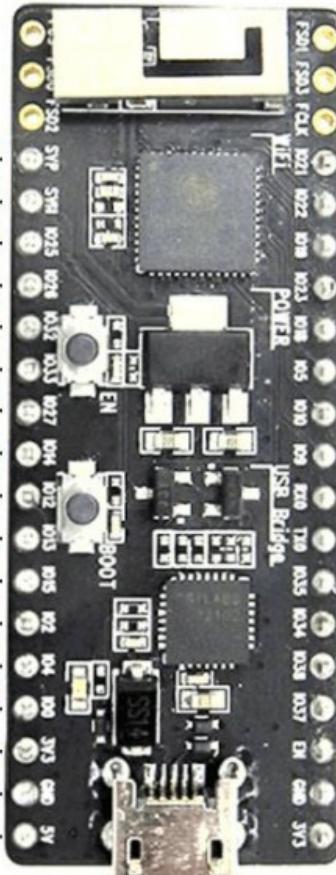
# Example reading a potentiometer

```
#define ANALOG_PIN A4 //pin 32
void setup() {
    Serial.begin(115200);
}

void loop() {
    int mv;
    mv = map(analogRead(ANALOG_PIN),0,4095,0,3300);
    Serial.println (mv);
    // prints 0-4095 <-> 0 - 3300mV
    delay(500);
}
```



IN	36
IN	39
IN OUT	25
IN OUT	26
IN OUT	32
IN OUT	33
IN OUT	27
IN OUT	14
IN OUT	12
IN OUT	13
IN OUT	15
IN OUT	2
IN OUT	4
IN OUT	0
3.3V	
GND	
5V	



**Q2 Circle what we would change to make the code read ADC channel A4 and print value in Volts (not mv)**

```
long map(long x, long in_min, long in_max, long out_min, long out_max)
{
    return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}

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

void loop() {
    int mv;
    mv = map(analogRead(A4),0,4095,0,3300); // map 0-4095 <-> 0 - 3300mV
    Serial.println (mv);
    delay(500);
}
```

**Q3 What would we write in this line to print Volts?**

## analogWrite() (actually pseudo “analog”)

- In Arduino analogWrite is Actually a modulated wave whose pulse width varies to set the effective voltage (PWM).

analogWrite(PIN1, 255);  100% duty cycle

analogWrite(PIN1, 192);  75% duty cycle

analogWrite(PIN1, 128);  50% duty cycle

analogWrite(PIN1, 64);  25% duty cycle

- A real analog write would use a digital to analog converter. It would output smooth voltage, not a modulated wave form.

- NOTE: This Arduino library function analogWrite is not yet available for ESP32!**

## dacWrite() (a real “analog” Write)

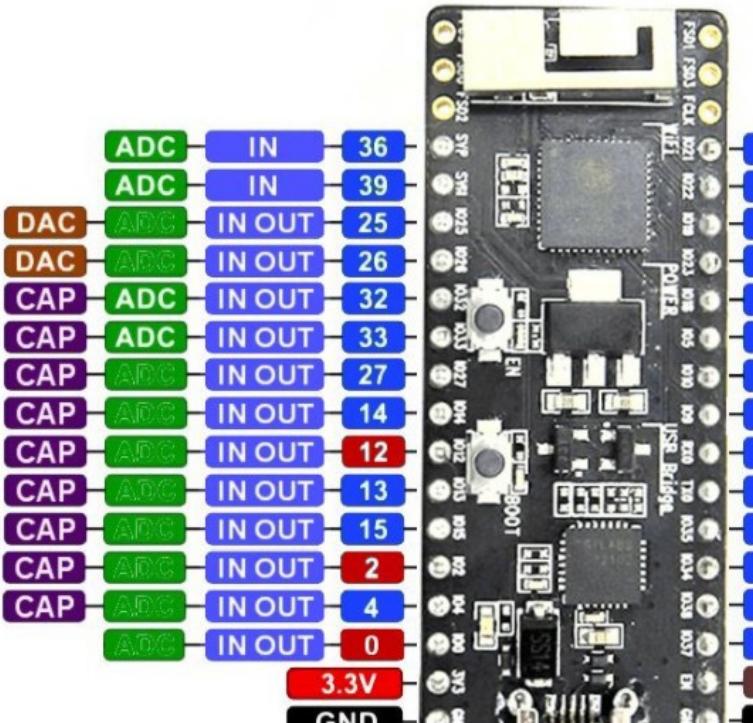
- ESP32 has two Digital to Analog converters (GPIO 25 or 26)

```
dacWrite(GPIOpin#, [0:255]); // 0 = 0V, 255=3.3V
```

```
void setup() {  
    Serial.begin(115200);  
}  
  
void loop() {  
    int i;  
    for(i=0;i<256;i++) {  
        dacWrite(25,i);  
        Serial.println(i);  
        delay(100);  
    }  
}
```

Ex: this one 

ADC	IN	36
ADC	IN	39
DAC	ADC IN OUT	25
DAC	ADC IN OUT	26
CAP	ADC IN OUT	32
CAP	ADC IN OUT	33
CAP	ADC IN OUT	27
CAP	ADC IN OUT	14
CAP	ADC IN OUT	12
CAP	ADC IN OUT	13
CAP	ADC IN OUT	15
CAP	ADC IN OUT	2
CAP	ADC IN OUT	4
ADC	IN OUT	0



# References online ([www.arduino.cc](https://www.arduino.cc/en/Reference/HomePage))

The screenshot shows the Arduino Reference website (https://www.arduino.cc/en/Reference/HomePage) in a web browser. The page has a teal header with the Arduino logo and navigation icons. The main content area is divided into three columns: 'Control Structures' (containing setup(), loop(), if, if...else, for, switch case, while, do... while), 'Constants' (containing HIGH | LOW, INPUT | OUTPUT | INPUT\_PULLUP, LED\_BUILTIN, true | false, integer constants, floating point constants), and 'Digital I/O' (containing pinMode(), digitalWrite(), digitalRead()). Below these are sections for 'Analog I/O' (analogReference(), analogRead(), analogWrite() - PWM).

- <a href="#">setup()</a>	<b>Constants</b>	<b>Digital I/O</b>
- <a href="#">loop()</a>	- <a href="#">HIGH   LOW</a>	- <a href="#">pinMode()</a>
<b>Control Structures</b>	- <a href="#">INPUT   OUTPUT   INPUT_PULLUP</a>	- <a href="#">digitalWrite()</a>
- <a href="#">if</a>	- <a href="#">LED_BUILTIN</a>	- <a href="#">digitalRead()</a>
- <a href="#">if...else</a>	- <a href="#">true   false</a>	<b>Analog I/O</b>
- <a href="#">for</a>	- <a href="#">integer constants</a>	- <a href="#">analogReference()</a>
- <a href="#">switch case</a>	- <a href="#">floating point constants</a>	- <a href="#">analogRead()</a>
- <a href="#">while</a>		- <a href="#">analogWrite() - PWM</a>
- <a href="#">do... while</a>		

New ⌘N

Open... ⌘O

Open Recent ▶

Sketchbook ▶

**Examples ▶**

Close ⌘W

Save ⌘S

Save As... ⌘S

Page Setup ⌘P

Print ⌘P

Blink | A



Blink

Blink

Turns an LED on for one second,

Most Arduinos have an on-board  
it is attached to digital pin 1  
the correct LED pin independent  
If you want to know what pin the  
model, check the Technical Spec

<https://www.arduino.cc/en/Main/>

modified 8 May 2014

by Scott Fitzgerald

modified 2 Sep 2016

by Arturo Guadalupi

modified 8 Sep 2016

by Colby Newman

## Built-in Examples

- 01.Basics
- 02.Digital
- 03.Analog
- 04.Communication
- 05.Control
- 06.Sensors
- 07.Display
- 08.Strings
- 09.USB
- 10.StarterKit\_BasicKit
- 11.ArduinoISP

## Examples for any board

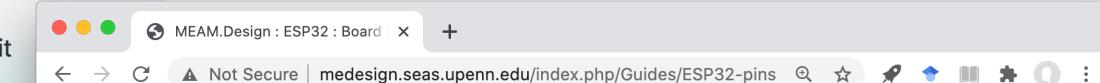
- Adafruit Circuit Playgroup
- Bridge
- Ethernet
- Firmata
- LiquidCrystal
- SD
- Stepper
- Temboo
- RETIRED

## Examples for NodeMCU

- ArduinoOTA
- BluetoothSerial
- DNSServer
- EEPROM
- ESP32
- ESP32 BLE Arduino
- ESPmDNS

# Many resources

<http://medesign.seas.upenn.edu/index.php/Guides/ESP32-pins>  
<http://www.esp32.com/>



## ESP32 Pico Kit

### Top Side Pins of Pico Kit

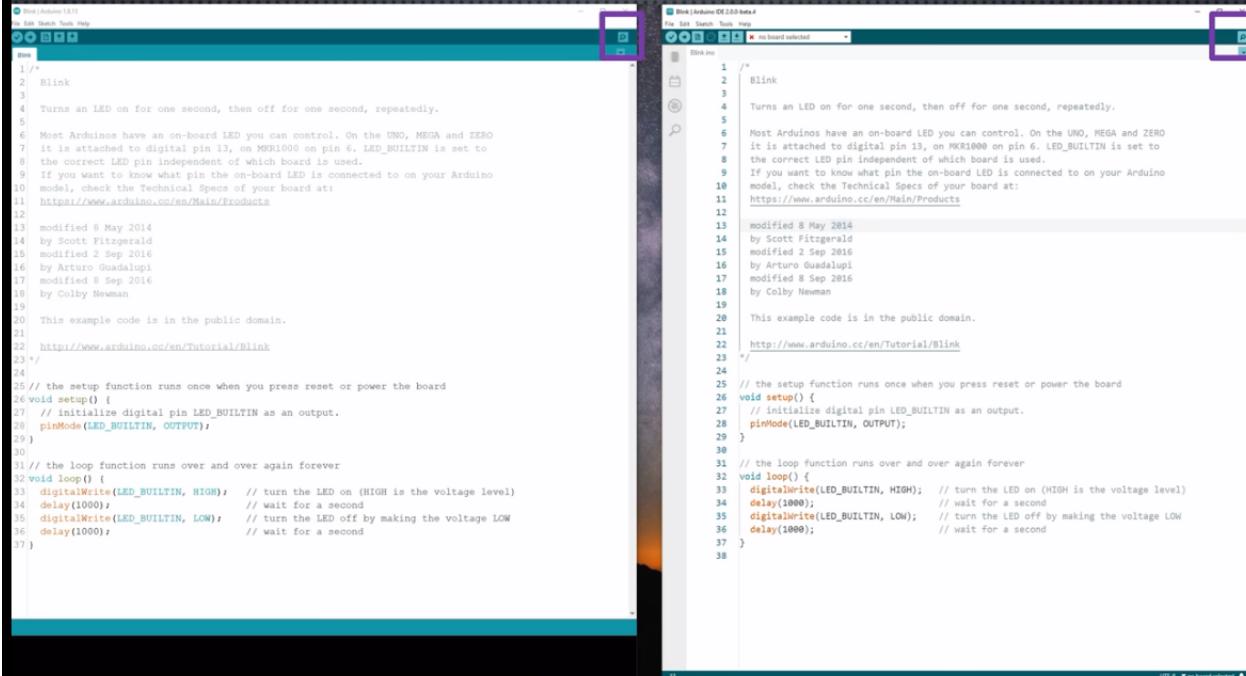
D1	Internal Flash, Don't use
D3	Internal Flash, Don't use
CLK	Internal Flash, Don't use
21 / VSPI HD	
22 / VSPI WP	
19 / VSPI Q	
23 / VSPI D	
18 / VSPI CLK	
5 / VSPI CS0	Boot strap pin, must be high on boot
10 / TXD1	
9 / RXD1	
RX0 / GPIO3	used for USB Serial.



# Arduino 2.0 Beta

<https://www.arduino.cc/en/Tutorial/getting-started-with-ide-v2/ide-v2-downloading-and-installing>

- Very similar
- Adds Code completion
- Adds debugging, but not ready.
- Better board support
- No ESP32 support
- Not likely to be ready before end of semester.



```
File: Arduino 2.0
File Edit Sketch Tools Help
File Edit Sketch Tools Help
Blink
1 // 
2 Blink
3
4 Turns an LED on for one second, then off for one second, repeatedly.
5
6 Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
7 it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
8 the correct LED pin independent of which board is used.
9 If you want to know what pin the on-board LED is connected to on your Arduino
10 model, check the Technical Specs of your board at:
11 https://www.arduino.cc/en/Main/Products
12
13 modified 8 May 2014
14 by Scott Fitzgerald
15 modified 2 Sep 2016
16 by Arturo Guadalupi
17 modified 8 Sep 2016
18 by Colby Newman
19
20 This example code is in the public domain.
21
22 http://www.arduino.cc/en/Tutorial/Blink
23 */
24
25 // the setup function runs once when you press reset or power the board
26 void setup() {
27 // initialize digital pin LED_BUILTIN as an output.
28 pinMode(LED_BUILTIN, OUTPUT);
29 }
30
31 // the loop function runs over and over again forever
32 void loop() {
33 digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
34 delay(1000); // wait for a second
35 digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
36 delay(1000); // wait for a second
37 }
```

```
File: Arduino 2.0 beta4
File Edit Sketch Tools Help
File Edit Sketch Tools Help
Blink.ino
1 /*
2 Blink
3
4 Turns an LED on for one second, then off for one second, repeatedly.
5
6 Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
7 it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
8 the correct LED pin independent of which board is used.
9 If you want to know what pin the on-board LED is connected to on your Arduino
10 model, check the Technical Specs of your board at:
11 https://www.arduino.cc/en/Main/Products
12
13 modified 8 May 2014
14 by Scott Fitzgerald
15 modified 2 Sep 2016
16 by Arturo Guadalupi
17 modified 8 Sep 2016
18 by Colby Newman
19
20 This example code is in the public domain.
21
22 http://www.arduino.cc/en/Tutorial/Blink
23 */
24
25 // the setup function runs once when you press reset or power the board
26 void setup() {
27 // initialize digital pin LED_BUILTIN as an output.
28 pinMode(LED_BUILTIN, OUTPUT);
29 }
30
31 // the loop function runs over and over again forever
32 void loop() {
33 digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
34 delay(1000); // wait for a second
35 digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
36 delay(1000); // wait for a second
37 }
38
```



Andreas Spiess

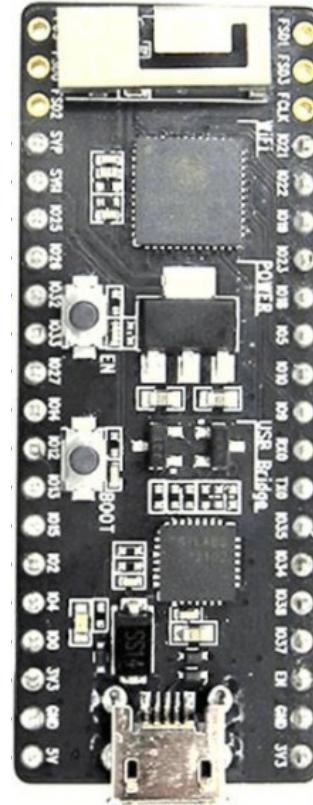
[https://www.youtube.com/watch?v=nll\\_5vxm3bk](https://www.youtube.com/watch?v=nll_5vxm3bk)

03

# ESP32 PicoKit Hardware Caveats

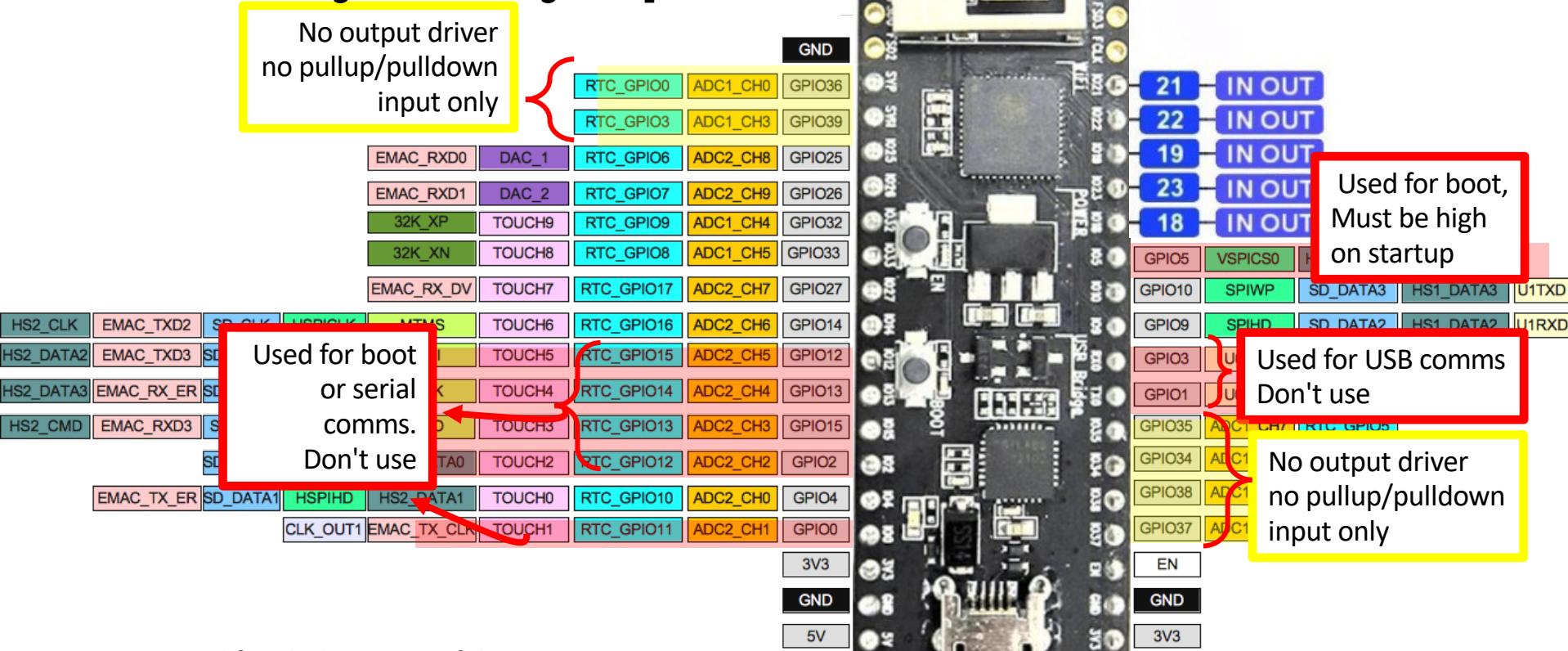
# General Purpose Input/Output (GPIO)

- 29 pins labeled as GPIO (only 20 are safe to use, see next slides for explanation)
- 14 GPIO pins can be input or output
- All use 3.3V
  - HIGH = 3.3V
  - LOW = 0V
- Internal pullup or pulldown
- **RED** LED turns on if powered.



**GPIO = General Purpose Input/Output**

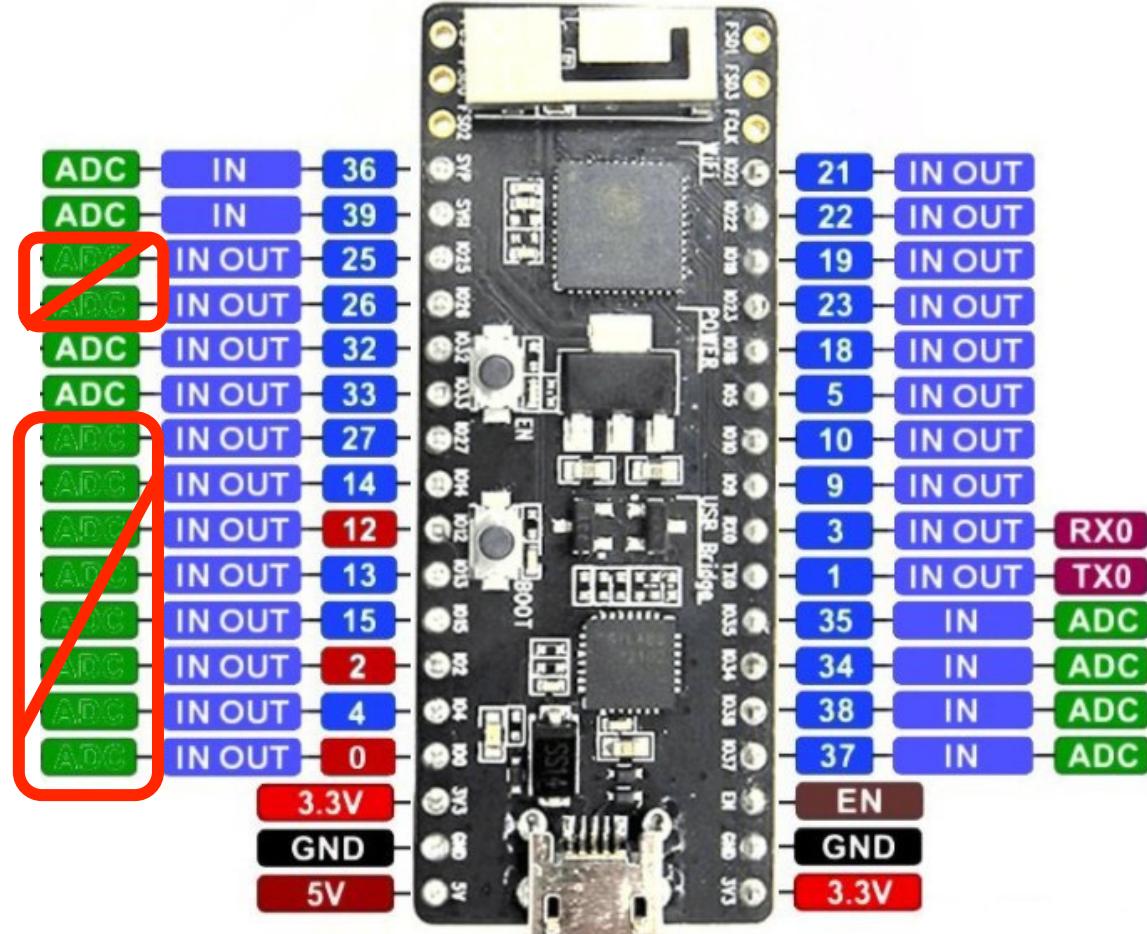
**GND = ground or negative power line**



used for Flash, can use if desperate  
but better left unconnected

# More ESP Issues – WiFi conflicts with some ADC

- On PicoKit 18 ADC pins are available.
- 10 of them ( $\text{ADC}\# > 9$ ) do not work when WiFi is on.



# Refer to medesign website for summary of caveats

<http://medesign.seas.upenn.edu/index.php/Guides/ESP32-pins>

- Don't use the pins that are highlighted in red

MEAM.Design : ESP32 : Board

Not Secure | medesign.seas.upenn.edu/index.php/Guides/ESP32-pins

## ESP32 Pico Kit

### Top Side Pins of Pico Kit

D1	Internal Flash, Don't use
D3	Internal Flash, Don't use
CLK	Internal Flash, Don't use
21 / VSPI HD	
22 / VSPI WP	
19 / VSPI Q	
23 / VSPI D	
18 / VSPI CLK	
5 / VSPI CS0	Boot strap pin, must be high on boot
10 / TXD1	
9 / RXD1	
RX0 / GPIO3	used for USB Serial.
TX0 / GPIO1	used for USB Serial.
35 / ADC1 CH7	No output driver. No pullup/pulldown. Can use input only.

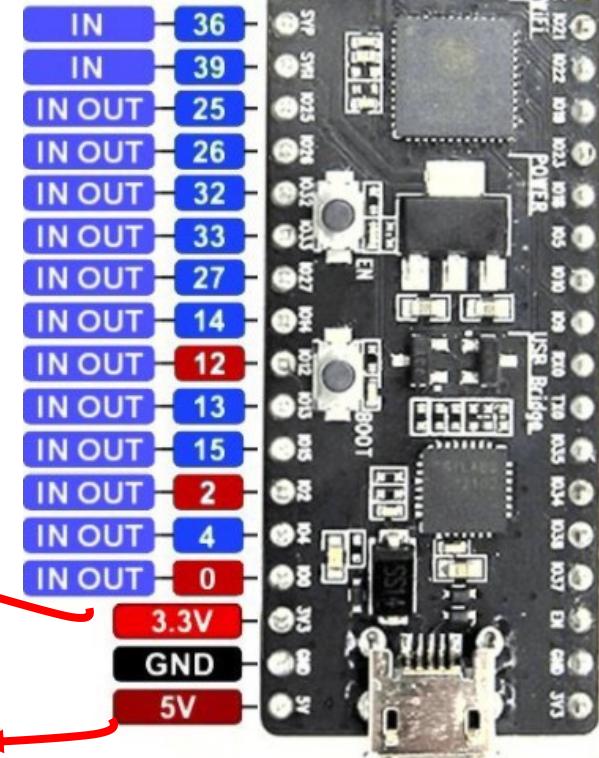


# PicoKit w/external power (e.g. battery)

- The PicoKit has a protection diode for your USB. So you can connect up to 12V on the Vin pin
  - even while USB is plugged in!
- Their design of the protection diode and voltage regulator is not great, so it causes problems we'll talk about later...

Both 3.3V source if USB powered or supply external 3.3V power

Both 5V source if USB or supply external power up to 12V



# ESP32 Issues

- If ESP32 mysteriously stop loading.
  - It is likely dead. We suspect that it is Electrostatic Static Discharge (ESD).
  - **Solution: Be sure to ground yourself always before touching ESP32 including hitting the download button.**
- NodeMCU and PicoKit may not work with all USB cables. The power design is not great.
  - **Use supplied USB cable. Later use external power.**



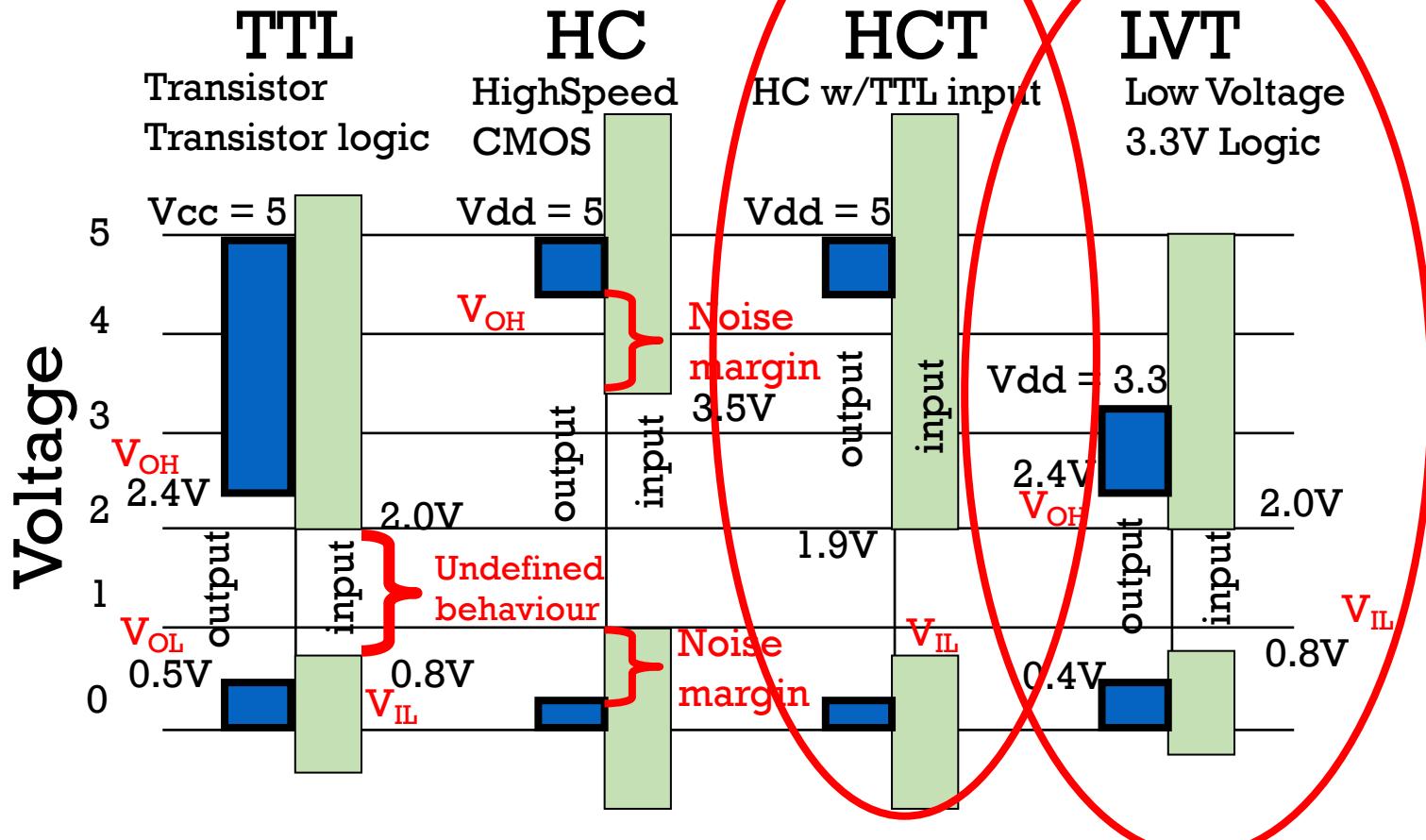
04

# Level Shifters 3.3 <-> 5V

# Logic Levels

Valid Output  
Levels

Valid  
Recognized  
Input Levels



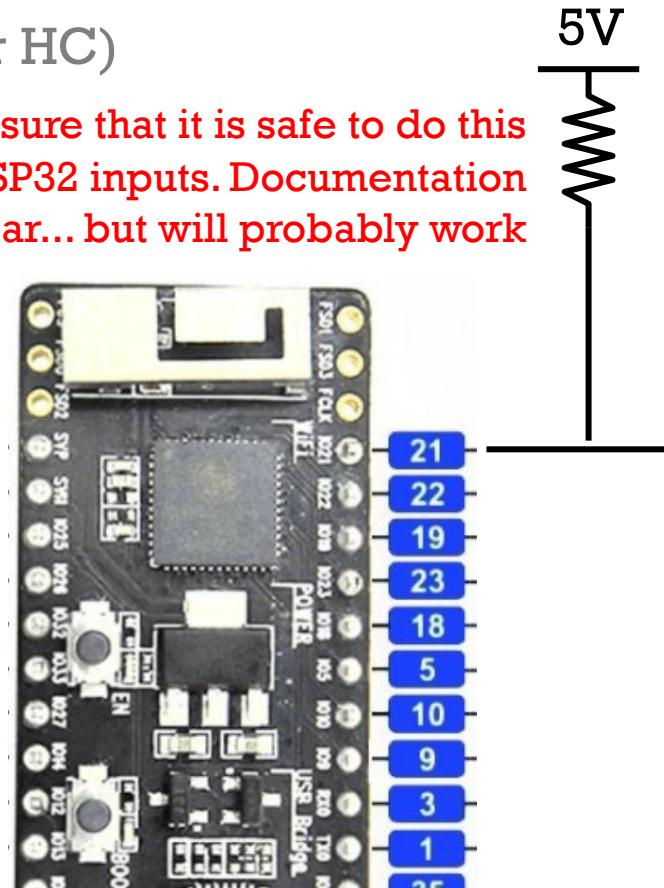
# Level shifting 3.3V to 5V with pullups

- For many devices we need 5V logic (TTL or HC)
  - 3.3V output should work on TTL and HCT Not sure that it is safe to do this with ESP32 inputs. Documentation not clear... but will probably work
  - 3.3V not guaranteed for HC
- In `setup()`

```
pinMode(21, INPUT);  
digitalWrite(21, LOW);
```

- Don't use `digitalWrite`.

- Instead use pullup:
  - For high values we use:  
`pinMode(21, INPUT);`
  - For low values we use:  
`pinMode(21, OUTPUT);`

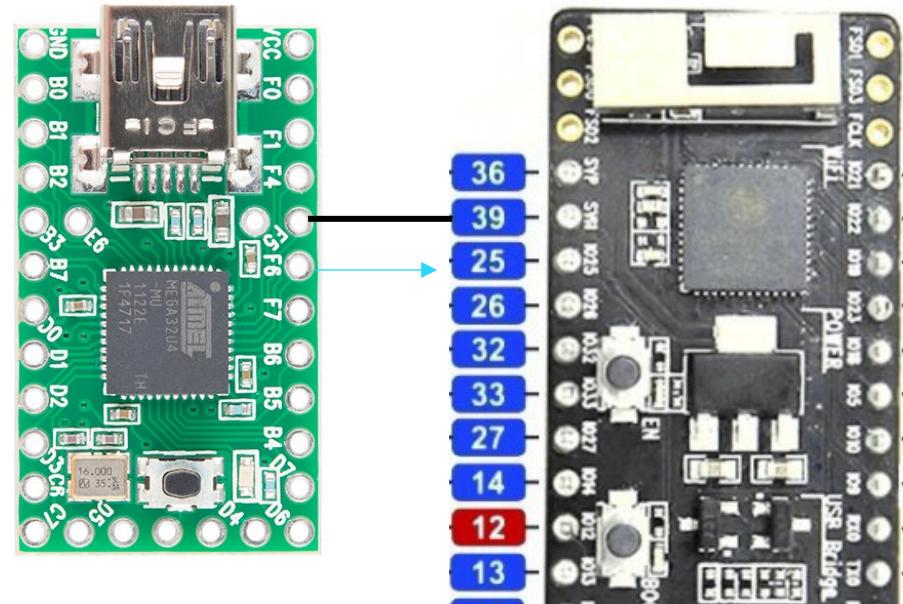


# Level shifting 5V output to 3.3V input

- Is ESP32 5V safe or 5V tolerant?
- The ESP32 documentation is not clear.. One spec says 3.9V max
- One Espressif engineer:

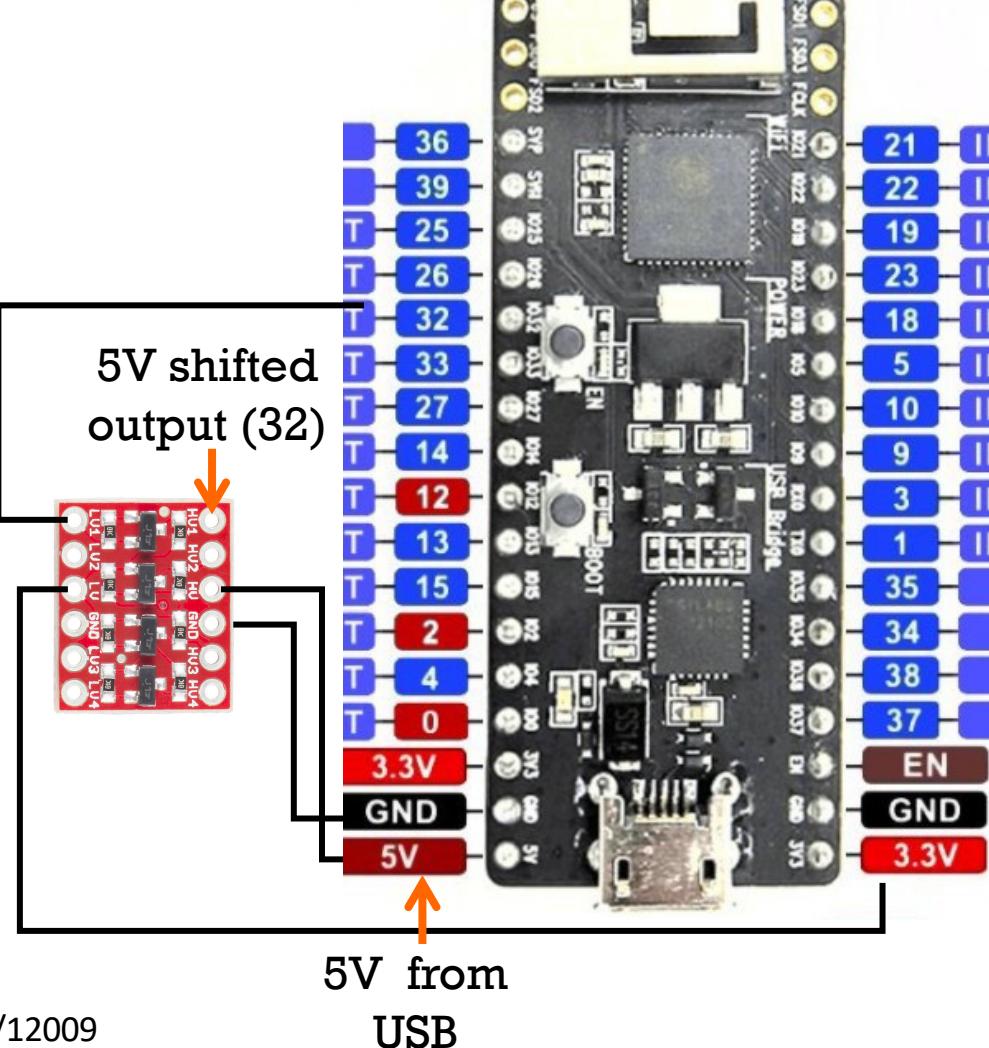
For "slow" 5V signals, it should be possible (though hacky) to use a single high value series resistor (1K or, ideally, more) to limit the current into the I/O pin's input protection circuitry, preventing it from damage. This is probably OK for a quick hack, but I would not recommend it for production hardware design or something where you need maximum reliability over the long term.

In those cases, level shifting is the best approach.



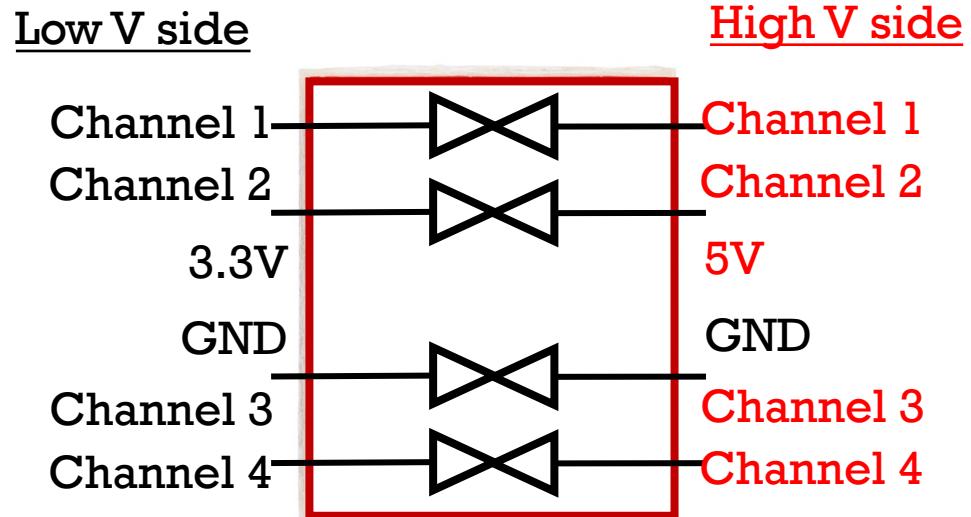
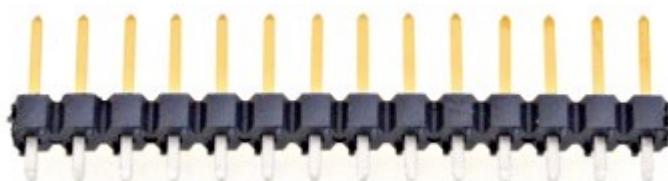
# Better Level Shifting

- Can use a level shifter and 5V from USB (Vin) as our 5V source.
- If Pin 32 is 3.3V, the output pf level shifter is 5V. Otherwise both are gnd.
- Four channel level shifter from sparkfun
  - Can shift 4 pins input or output

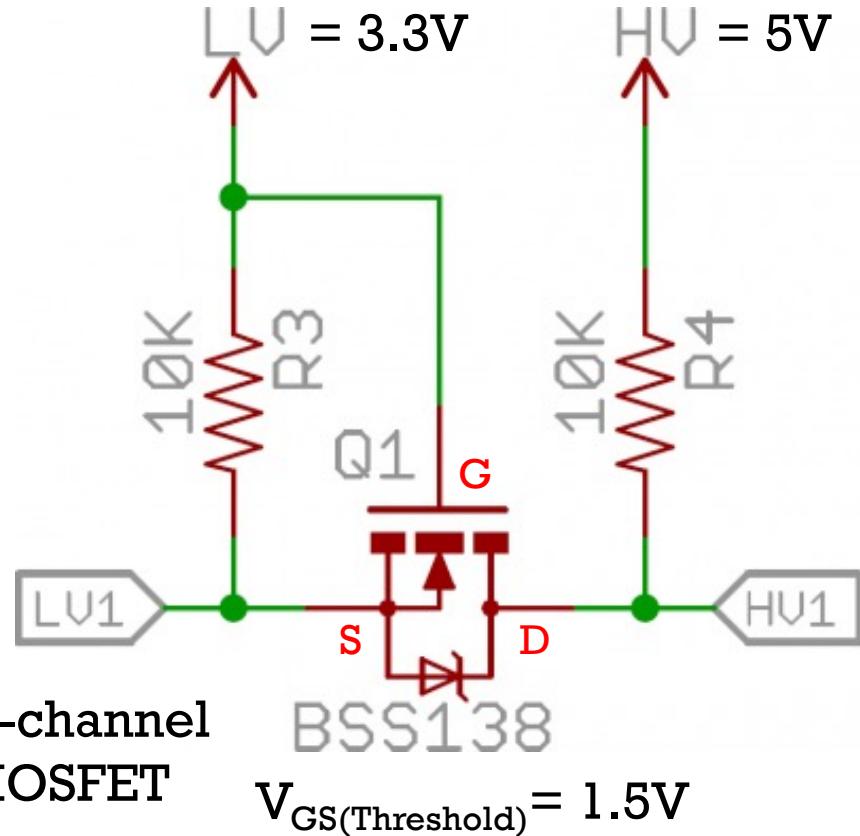


# Soldering level shifter (BOB 12009)

- Logic levels on one side are translated to corresponding logic levels on opposite side.
- Also provides a level of safety between devices.
- You will need to solder headers in first like with teensy
- Solder in breadboard.

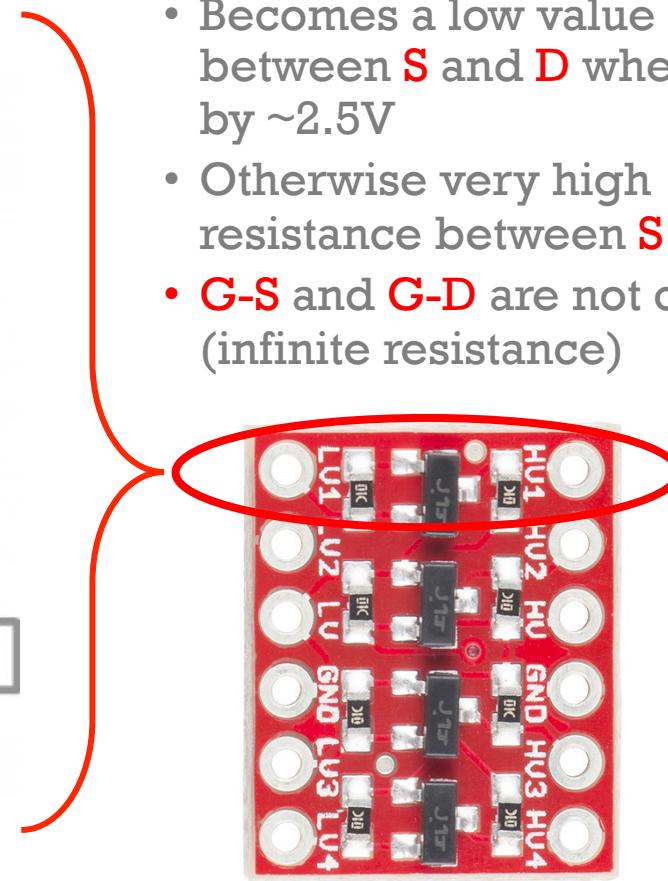


# Level Shifter



- N-channel MOSFET

- Becomes a low value resistor between **S** and **D** when  $V_G > V_S$  by  $\sim 2.5\text{V}$
- Otherwise very high resistance between **S** and **D**
- **G-S** and **G-D** are not connected (infinite resistance)



# Level Shifter

Four cases:

- LV1 is output HIGH driving HV1
- LV1 is output LOW driving HV1
- HV1 is output HIGH driving LV1
- HV1 is output LOW driving LV1



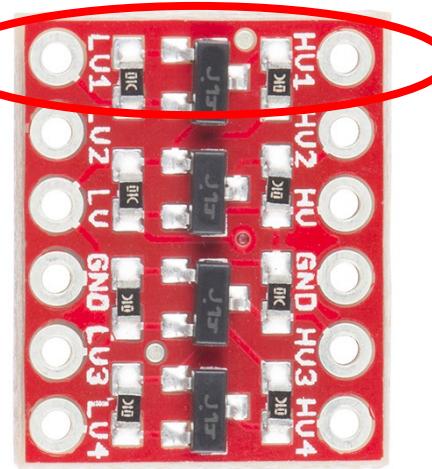
N-channel  
MOSFET

$$V_{GS(\text{Threshold})} = 1.5\text{V}$$

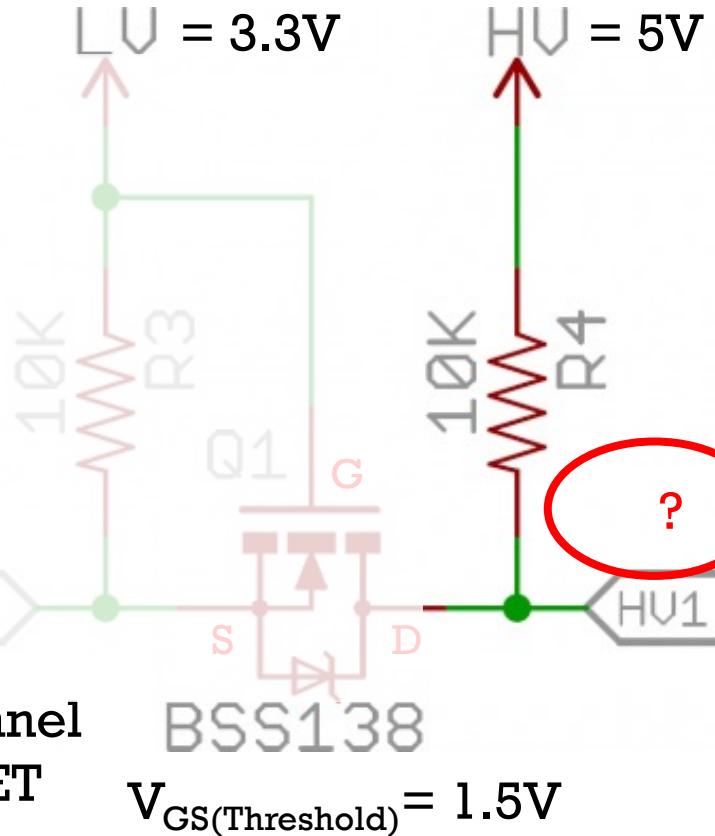
BSS138

- N-channel MOSFET

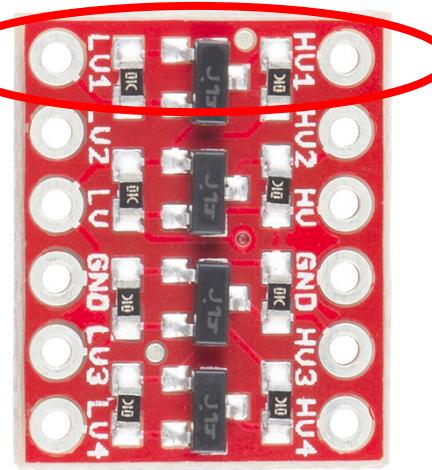
- Becomes a low value resistor between **S** and **D** when  $V_G > V_S$  by  $\sim 2.5\text{V}$
- Otherwise very high resistance between **S** and **D**
- **G-S** and **G-D** are not connected (infinite resistance)



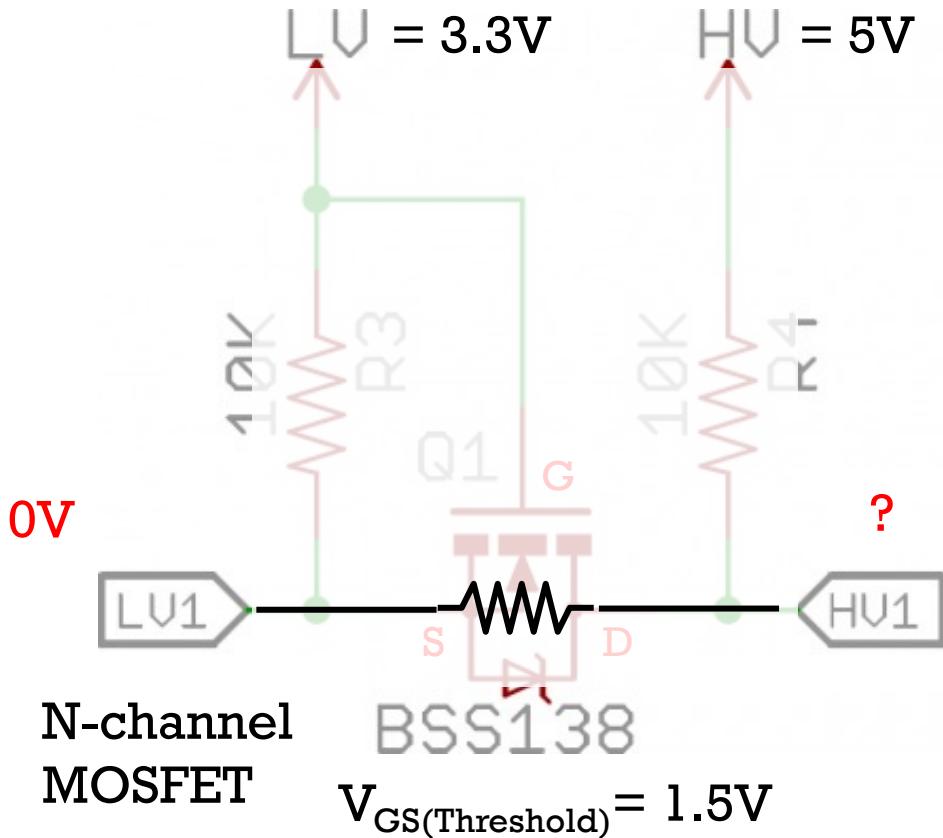
## Q4: Level Shifter



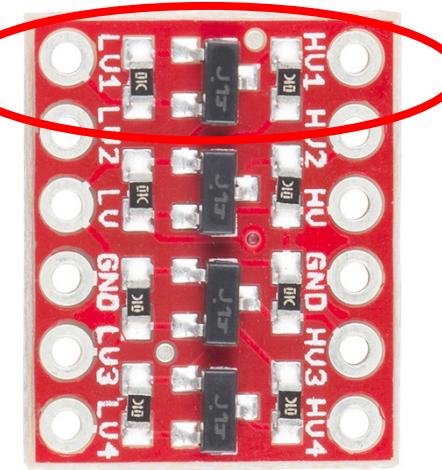
- With 3.3V at  $LV_1$ , what voltage is at  $HV_1$  and what is the max current supplied to  $HV_1$  to maintain proper logic levels?



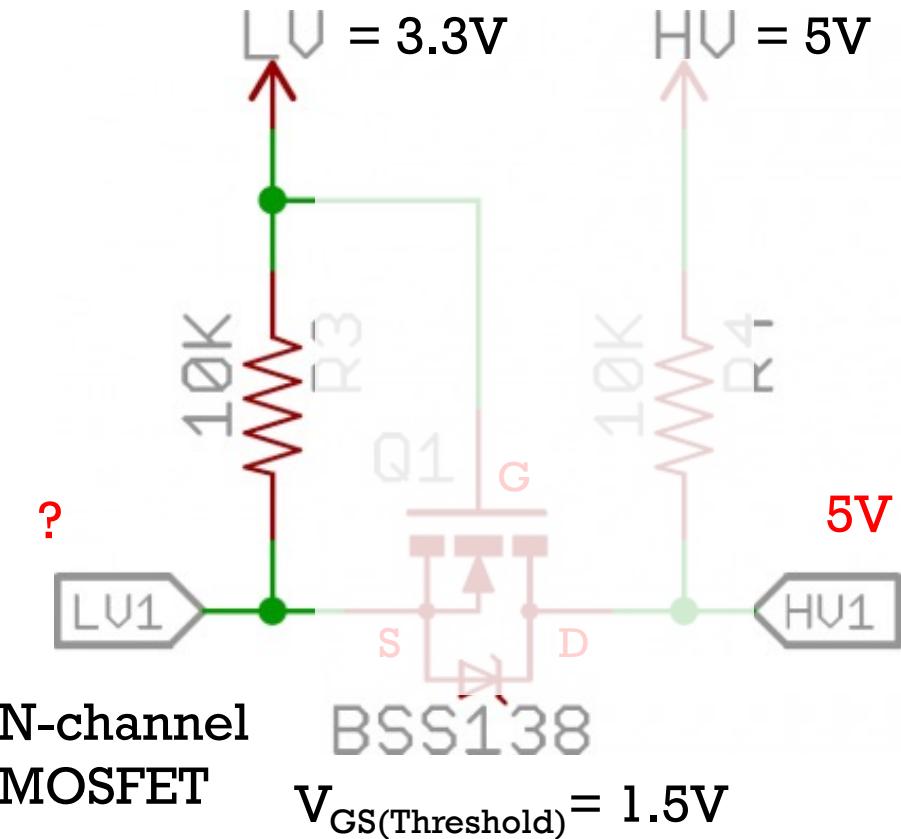
## Q5: Level Shifter



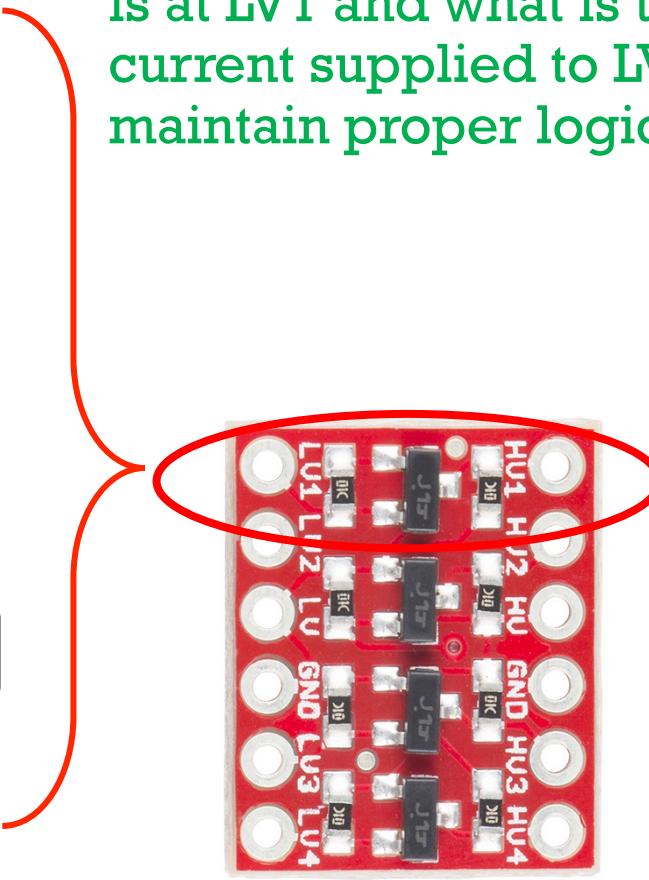
- With  $0\text{V}$  at  $LV_1$ , what voltage is at  $HV_1$  and what is the max current supplied to  $HV_1$  to maintain proper logic levels?



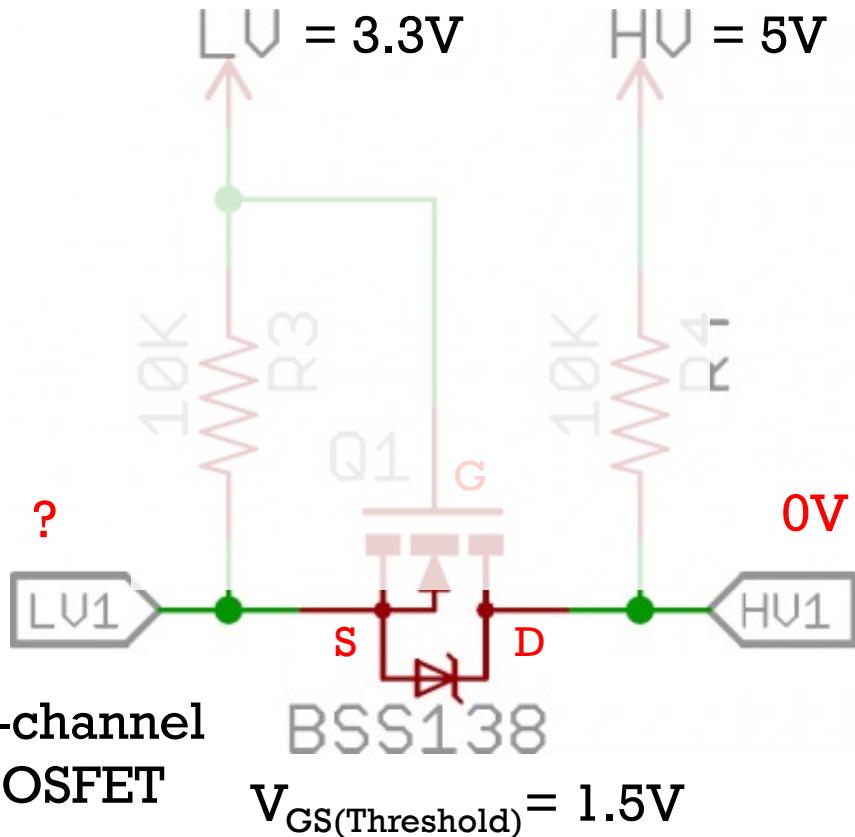
## Q6: Level Shifter



- With  $5\text{V}$  at  $HV_1$ , what voltage is at  $LV_1$  and what is the max current supplied to  $LV_1$  to maintain proper logic levels?



## Q7: Level Shifter



- With 0V at HV1, what voltage is at LV1 and what is the max current supplied to LV1 to maintain proper logic levels?

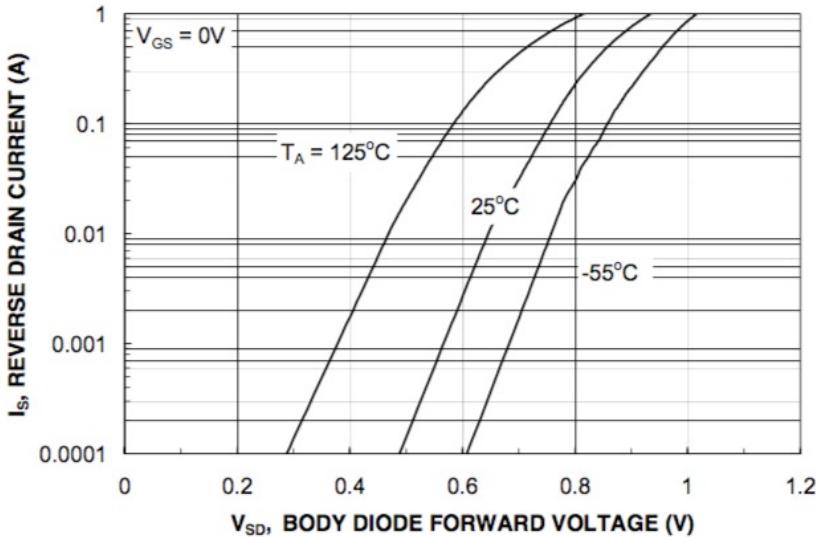
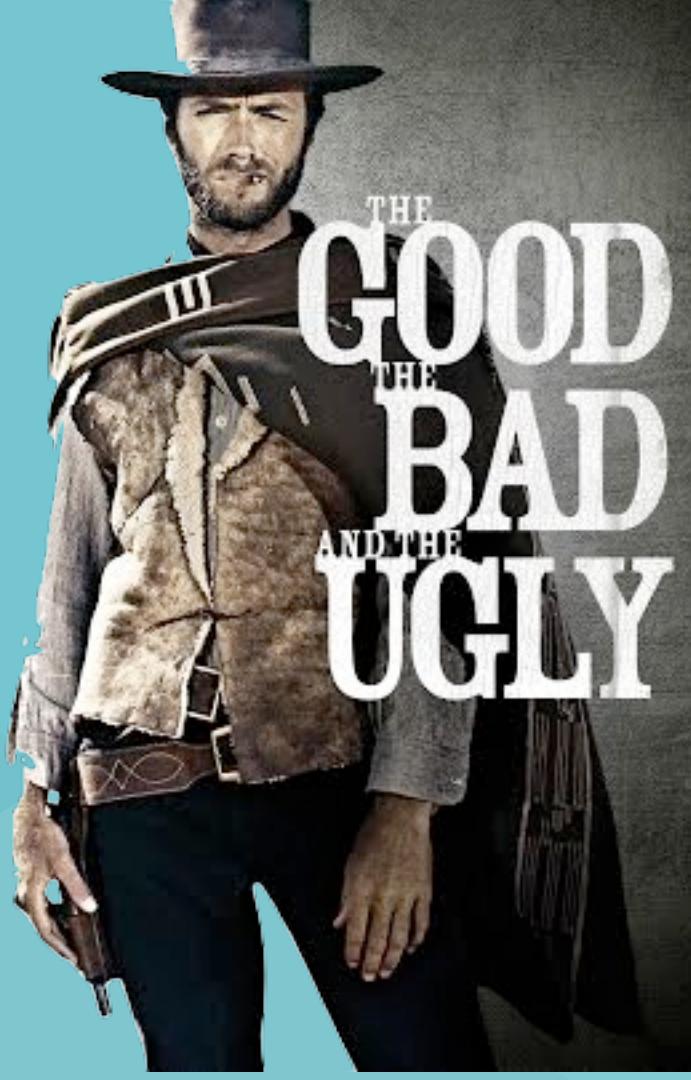


Figure 6. Body Diode Forward Voltage Variation with Source Current and Temperature.

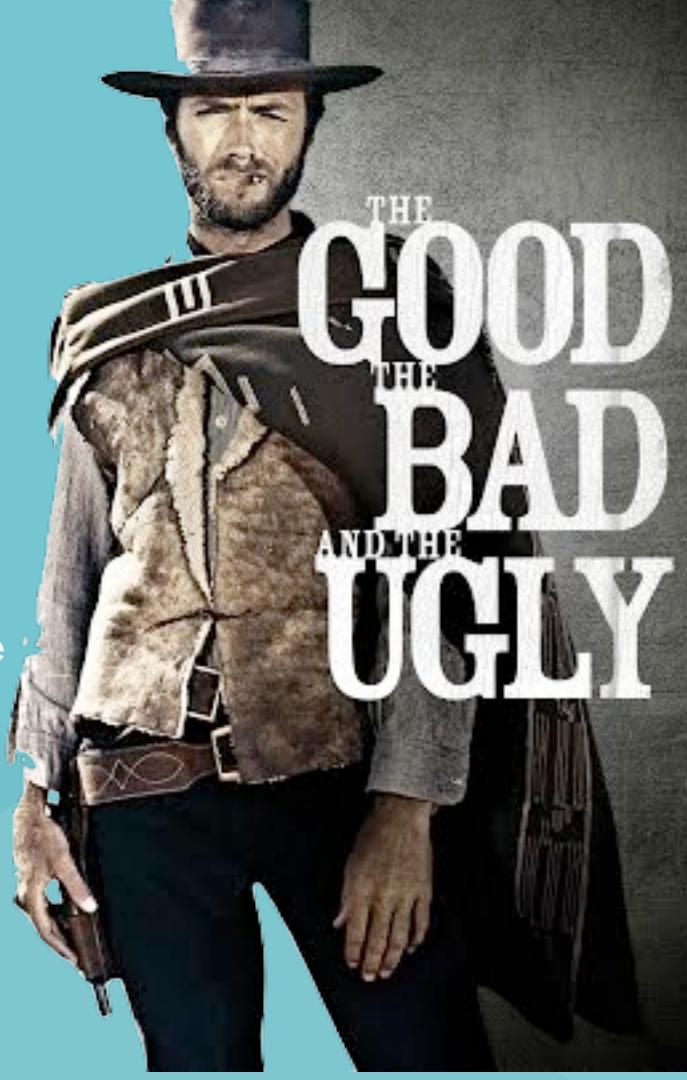
# The Good

1. WiFi on a chip - easy to setup a webpage
2. Full TCP/IP stack firmware w/RTOS
3. Very cheap (\$3 for module, \$10 for board)
4. Arduino SDK
5. DIY community is eating it up.
6. ~1M available program space
7. Enough I/O to do many things
  - A. 18 ADC
  - B. 2 SPI
  - C. 2 UART
  - D. I2S, I2C
  - E. ~17 useful GPIO



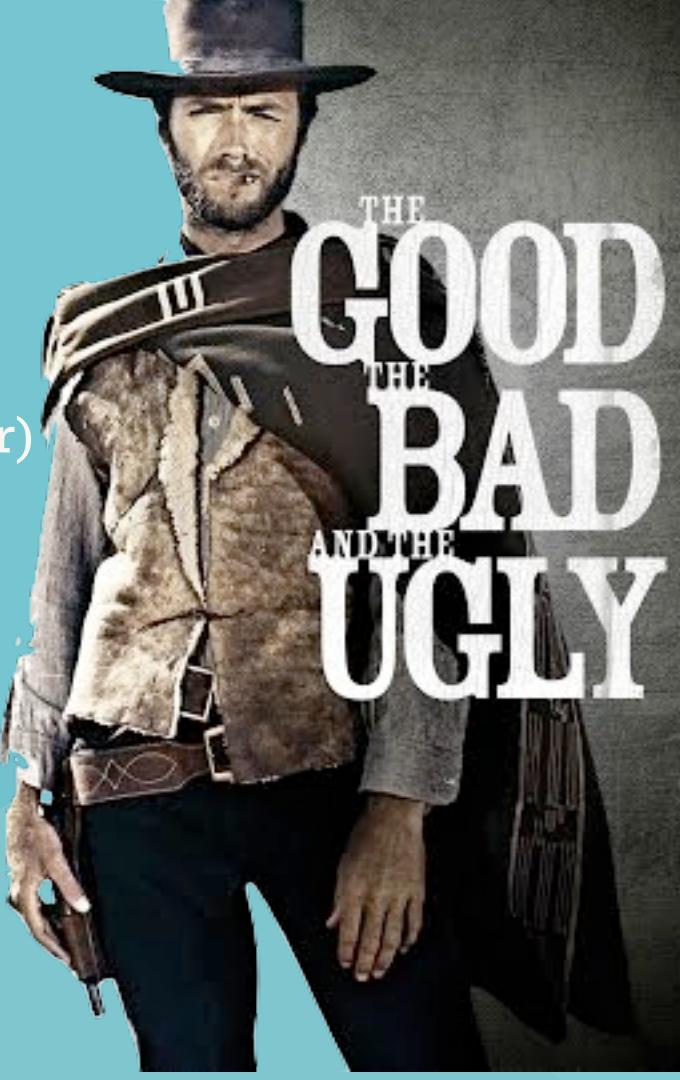
# The Bad

1. WiFi to internet is complicated.
2. Arduino SDK only partially implemented
  - A.No servo library
  - B.TCP is slow and timing is inconsistent
  - C.WiFi Latency ~130ms per packet
  - D.UDP is reasonably fast though ~500 packets /sec.
3. Timing not guaranteed (in arduino, PWM can be glitchy)
4. Downloading is much slower than Teensy
5. Power circuit design is borderline bad
6. QA for hiletgo is bad - ~2% failure out of box



# The Ugly

1. 3.3V interface (ports not quite 5V safe?)
  - A. Use 5V level shifters from sparkfun (elegant design)
2. Documentation is horrendous (getting better)
3. RTOS and dual core adds complexity
4. Many ports are multi-use in default conditions (e.g. GPIO used for startup)
5. Standard ADC accuracy is not great.
6. Randomly gets **hot** and dies... ESD?



# Summary

- ESP32 is much more powerful than teensy.
- ESP32 uses 3.3V logic
- ESP32 documentation is nonstandard and difficult to parse.
- When developing with ESP32 be careful of ESD!
- We will use Arduino SDK.