

Lecture 15

ESP32 and Arduino

Agenda

- Lab 4 Mobile Bases
- ESP32
- Arduino
- PicoKit ESP32 Hardware
- DEMO of TA MOBILE BASE

01

Lab 4: Mobile Bases

Mobile Bases

- Skid steer
 - Easy to make, hard to steer
- Ackerman steering
 - Hard to make, constrained motion
- Holonomic
 - Expensive, high mobility



Mobile Bases

- Differential Drive
 - Castor or skid in front
 - Easy to make
 - Lower cost
 - Recommended for simplicity
 - Compared to skid steer:
 - Similar control for steering
 - Less motor power required to turn



02

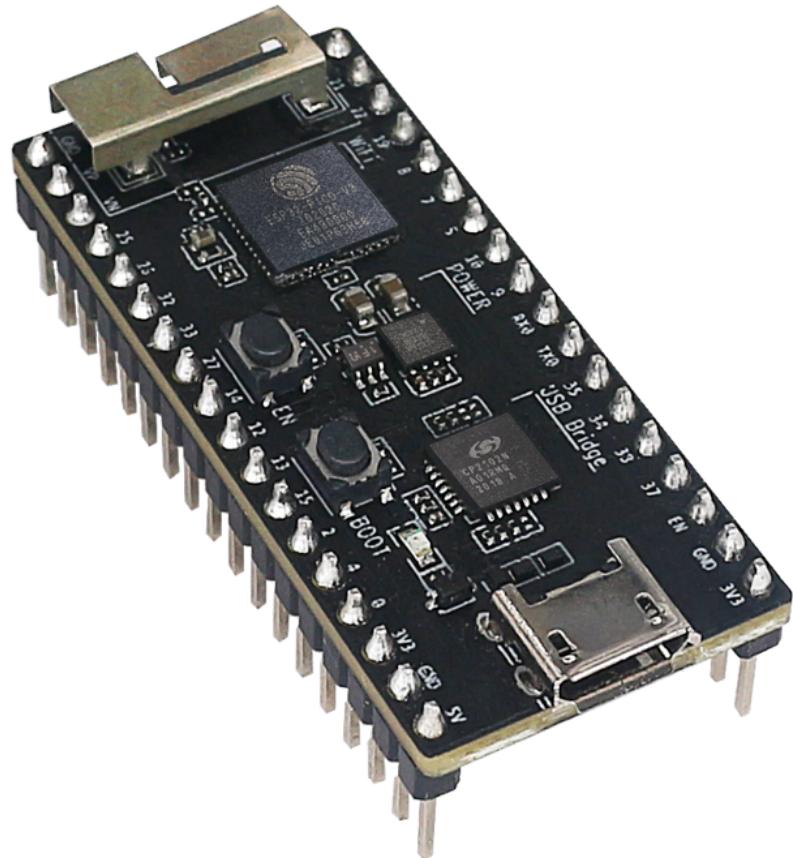
ESP32 Overview

Typical important functions on a microcontroller

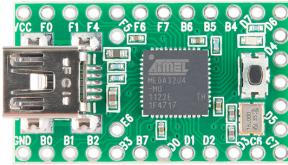
- Computation
- Memory
- Inputs
- Outputs
- Timers
- Analog to Digital Converter
- Communication

ESP32 Pico-Kit

- ESP32 *Wifi and Bluetooth Module*
 - ~50 foot range
- Operating voltage = 3.3V
- Voltage sources 3.3 – 12V
 - Through VIN pin or 5v through USB
- USB interface to PC or Mac
- 520 KiB SRAM, 4MB of flash memory
- 160 MHz clock speed (dual core)



Teensy VS NodeMCU

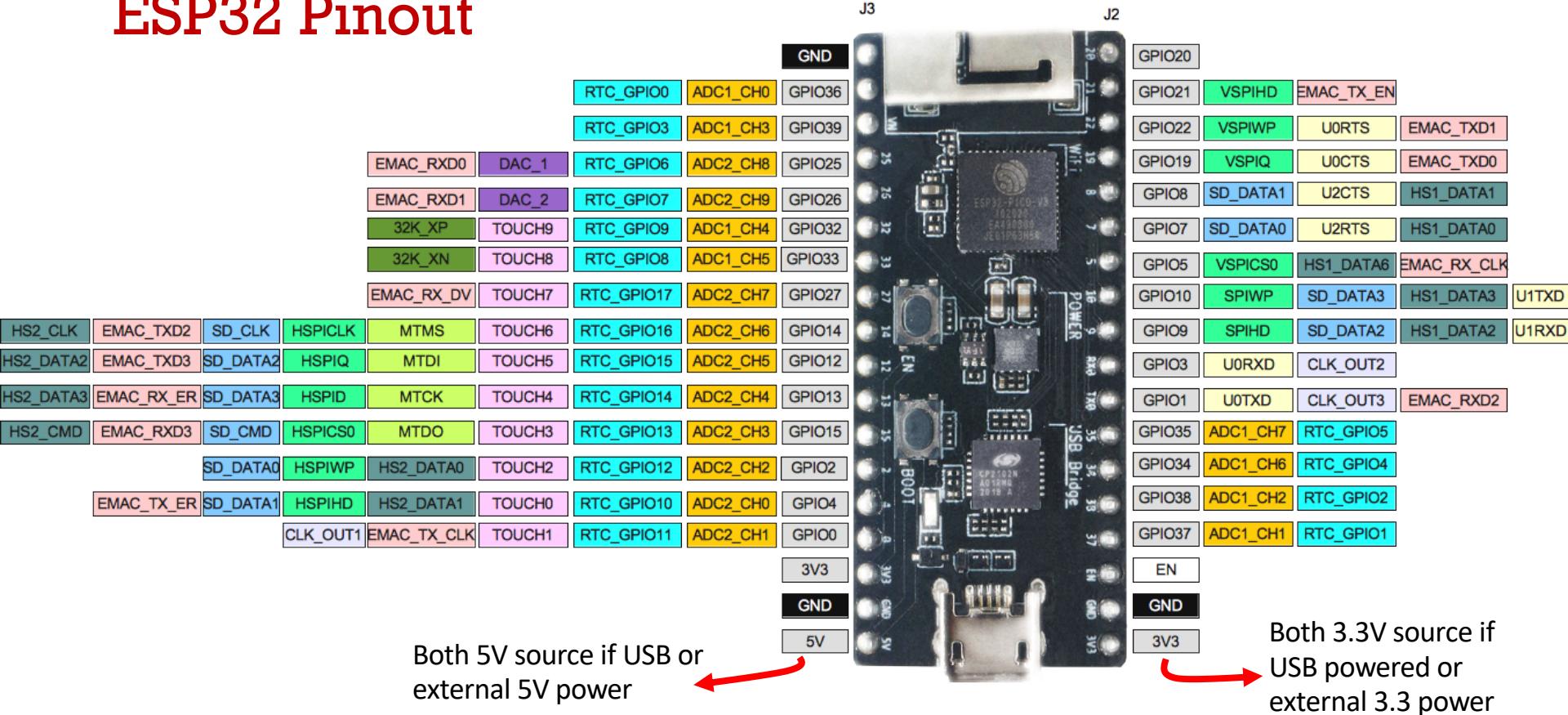


	ATMEGA 32U4	ESP32	ESP32 Difference
Processor Speed	16MHz, 8bit RISC	160Mhz, 32bit RISC	~20x faster comp
Memory	32k Flash; 2.5K RAM	4M Flash; 512K RAM	>100x's more
GP Input/Output	21 @ 5V	17 to 32 @ 3.3V	Similar
IO Current	20 mA (40 mA max)	10mA (20-40mA max)	Uncertain
ADC	12 channels 10-bit	16 channels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

Pico-kit

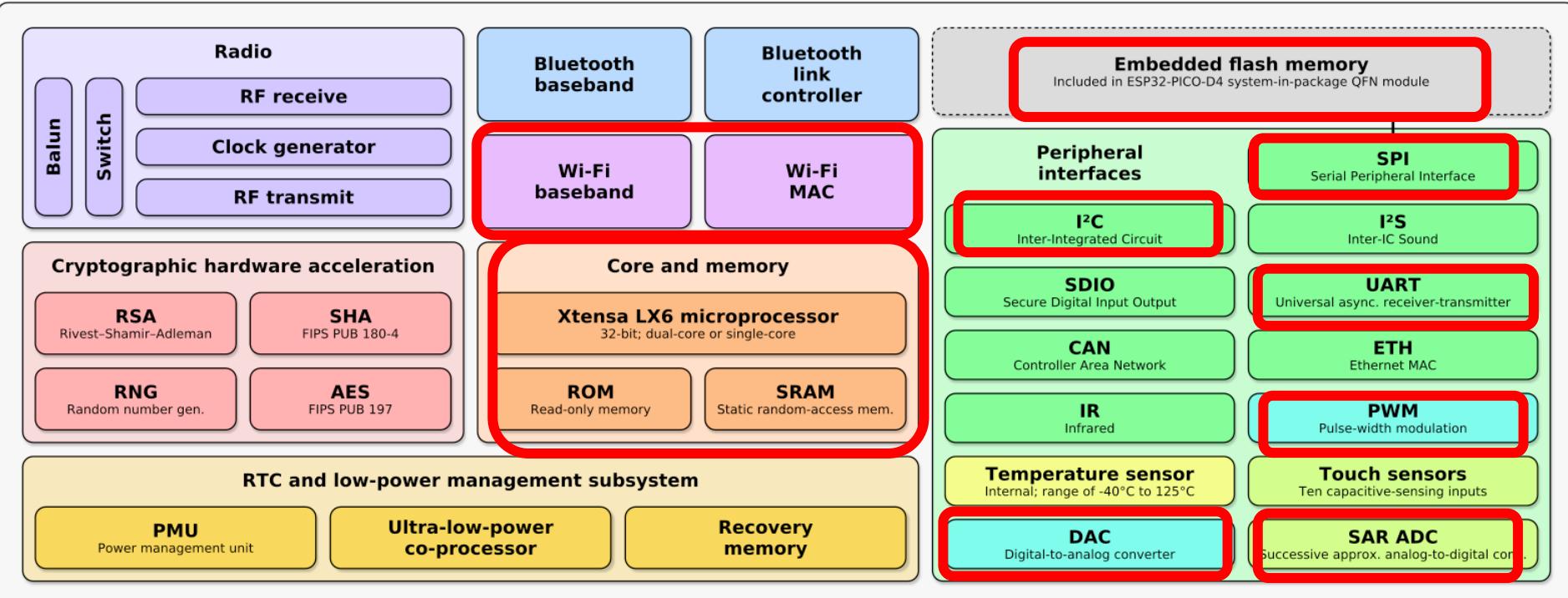
ESP32 Pinout

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



ESP32 Block Diagram

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



03

Software - Arduino

Arduino Set-up

MEAM.Design : ESP32 : Getting Started

GENERAL MEAM.Design - NodeMCU - Getting Started

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

Phidget

S62

ESP8266

Hardware/software Requirements

1. Arduino Integrated Development Environment (IDE)
2. ESP32 NodeMCU board
3. MicroUSB cable
4. WindowsPC or Mac
5. SiLabs CP210x driver (for Mac)

Install Arduino IDE

1. Both Mac and Windows can download Arduino IDE. [Arduino.cc](https://arduino.cc) e.g. version (1.8.7) or later.
2. Install according to the instructions

Windows PC Instructions

(You can use this video starting at 5:30 for a reference: <https://www.youtube.com/watch?v=mJcxnaR08Dg>)

1. Go to: <https://github.com/espressif/arduino-esp32>
2. Click Clone or Download→Download ZIP
3. Extract downloaded files.
4. Create new file path in your Arduino folder: C:\Users\ **YourName**\Documents\Arduino\hardware\espressif\esp32 and copy and paste the files from Step 3 here.
5. Go to the Tools folder and run get.exe
6. Test with the Blink Code (see below)

Mac instructions

(You may use this video starting at 1:45 for a reference: <https://www.youtube.com/watch?v=-P7RD-DWVuA&t=320s>)

1. Go to: <http://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers>
2. Find the "Download for Macintosh OSX" and download it [5]

The screenshot shows the Arduino IDE interface. At the top, there's a toolbar with icons for file operations like Open, Save, and Print. Below the toolbar is a menu bar with 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. The 'Tools' menu is currently open, displaying various options. A red box highlights the 'ESP32 Sketch Data Upload' option under the 'Sketch' section. To the right of the menu, there's a vertical stack of text snippets and user profiles. One snippet discusses an LED being connected to digital pin 13. Another snippet shows a commit history with entries from May 2014 to Sep 2016. At the bottom of the interface, there are links for 'Programmer' and 'Burn Bootloader'.

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“:

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)



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

Command Summary

Teensy

set/clear (DDR_x, pin#)

set/clear (PORT_x, pin#)

bit_is_clear/set(PIN_x,pin#)

m_usb_init()

m_usb_tx_uint(X)

_delay_ms()

ADC routines...

PWM timer routines...

ESP32 Arduino

- `pinMode(pin#, INPUT/OUTPUT)`

- `digitalWrite(pin#, HIGH/LOW)`

- `digitalRead(pin#)`

- `Serial.begin()`

- `Serial.println()`

- `delay()`

- `analogRead(pin#)`

- ~~`analogWrite(pin#)`~~ **not for ESP32**

- ledc routines... (will cover later)

- `dacWrite(pin#)`

Arduino Blink Sketch

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 (Note write LOW)
                                         // but actually the LED is on;
                                         // LED it is active low on the ESP

    delay(1000);                      // Wait for a second
    digitalWrite(LED_BUILTIN, HIGH);    // Turn the LED off by making pin HIGH
    delay(2000);                      // Wait for two secs
}
```

Functions: in Blink

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

- Tells the micro that pin is an output pin.
- Modes can be INPUT, OUTPUT, INPUT_PULLUP, INPUT_PULLDOWN

Available on ESP32,
not most others

```
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);

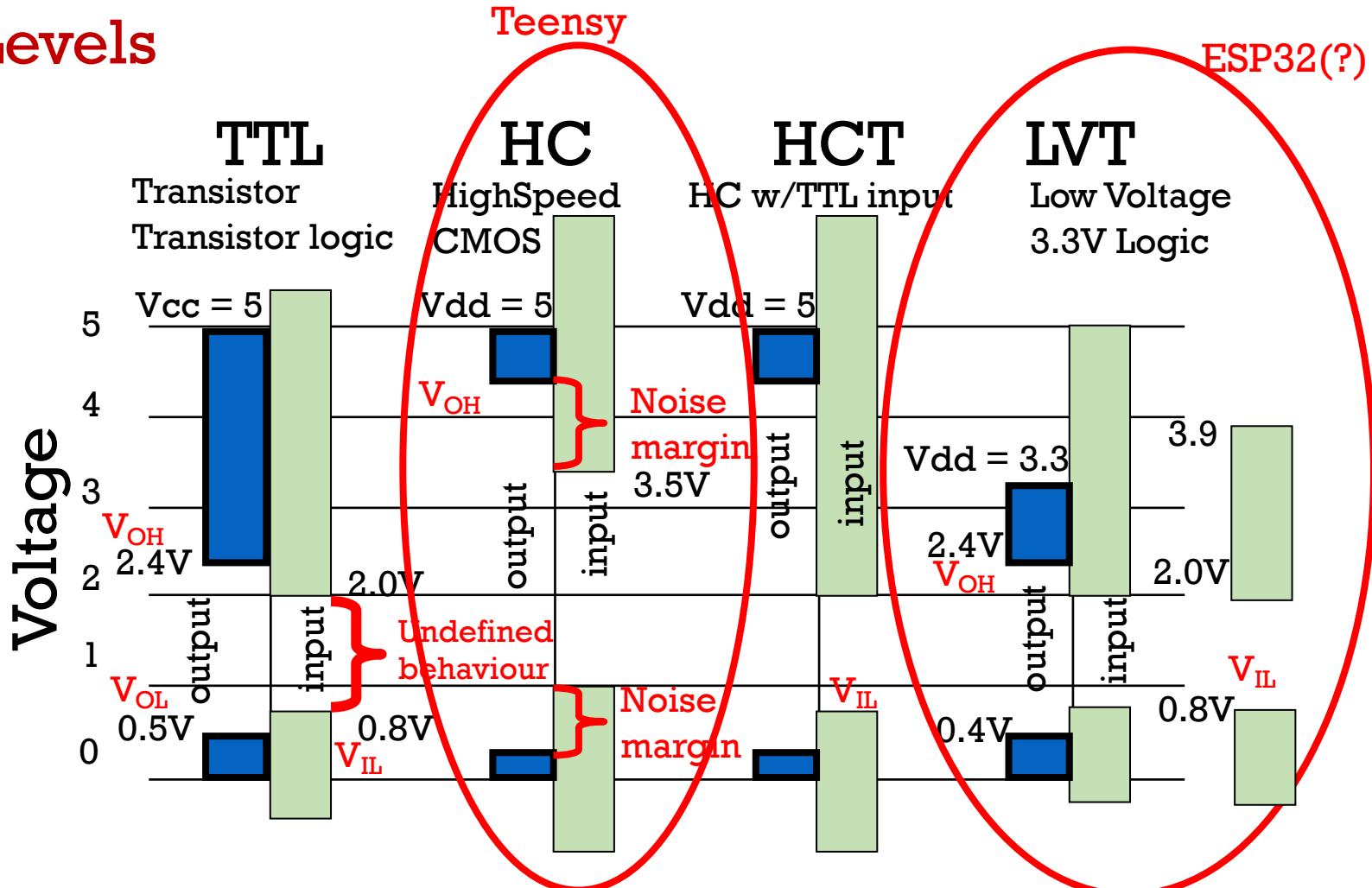
```
delay(1000);
```

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

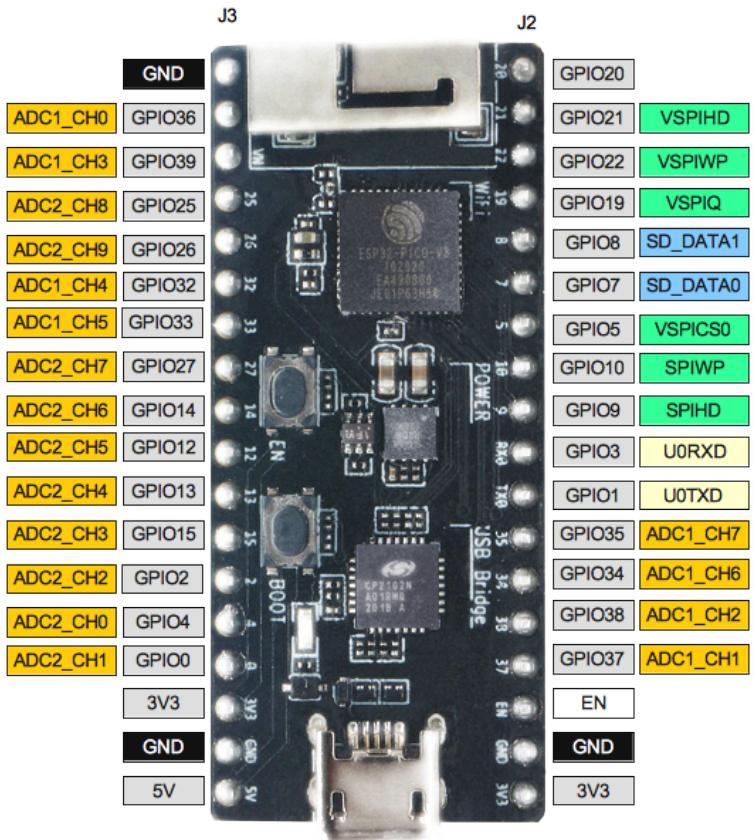
Logic Levels

Valid Output
Levels

Valid
Recognized
Input Levels



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)  
#define digitalPinToInterrupt(p) (((p)<40)  
#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. (Busy waits)

`Serial.begin(115200);`

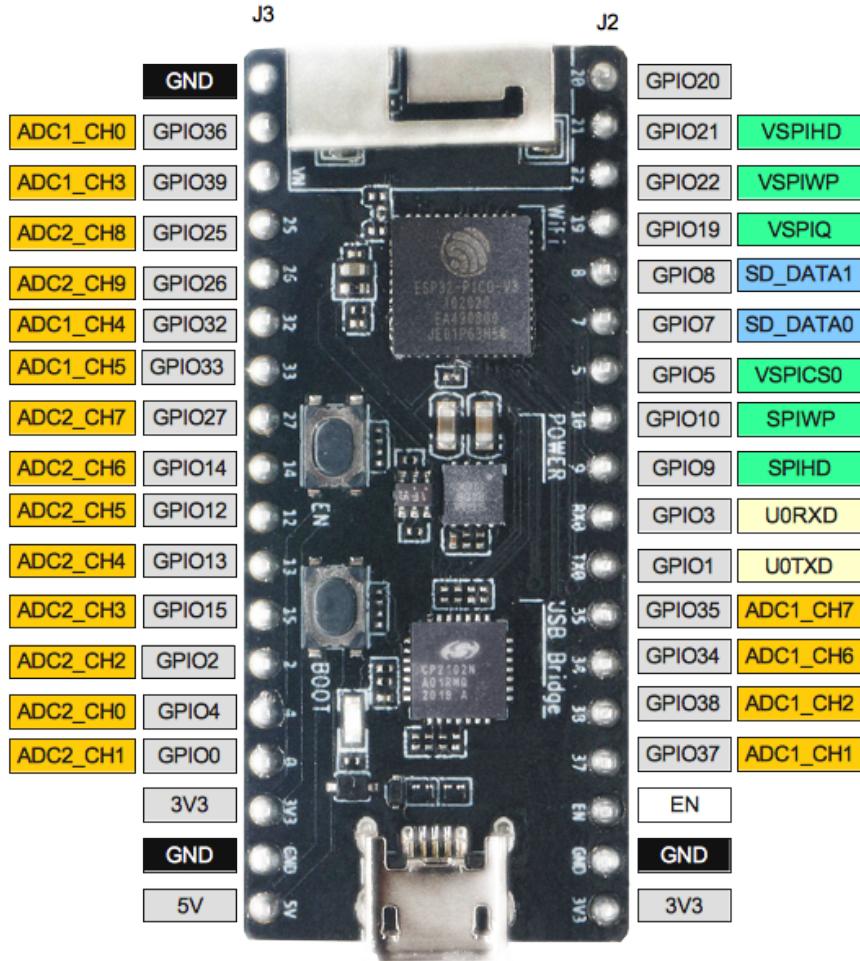
- Needed for serial comms.
- 115200 is the baudrate. 9600 is another standard rate.

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

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

Analog to Digital Converter

- Reads voltage at pin
- 18 available **ADC pins**
- 12bit SAR ADC
- Sample rate (about 6kHz)
- Can be referred to the ADC channel number A#, or GPIO number
 - ADC channel 0-> A0 = 36



analogRead(pin)

- Reads the ESP32 ADC channel 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 URL in the address bar is `https://www.arduino.cc/reference/en/language/functions/math/map/`. The page content includes the Arduino logo, a navigation bar with links to Home, Store, Software, Education, Resources, Community, and Help, and a sidebar with links to Language, Functions (which is currently selected), Variables, and Structure. The main content area shows the `map()` function's description, which states it re-maps a number from one range to another. The text also mentions that the function does not constrain values to within the range, and that the `constrain()` function may be used for that purpose.

Reference > Language > Functions > Math > Map

map()

[Math]

Description

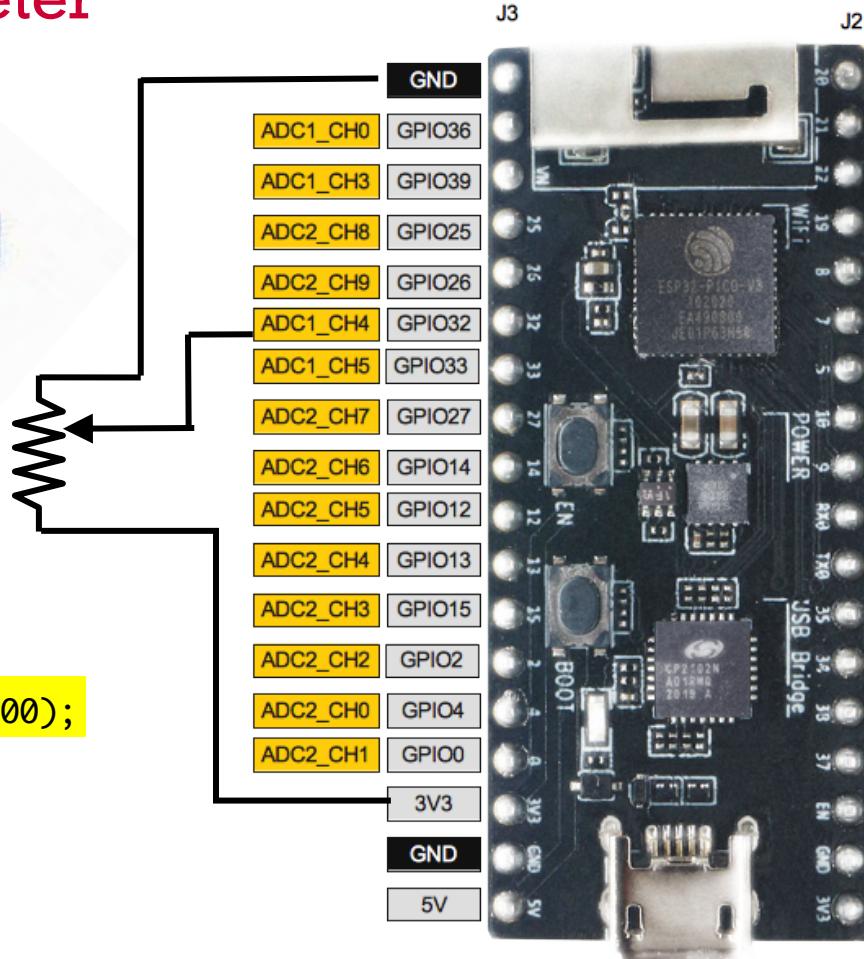
Re-maps a number from one range to another. That is, a value of `fromLow` would get mapped to `toLow`, a value of `fromHigh` to `toHigh`, values in-between to values in-between, etc.

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  
  
void setup() {  
    Serial.begin(115200);  
    Serial.print("ESP32 Analog Pin - GPIO");  
}  
  
void loop() {  
    int mv;  
    mv = map(analogRead(ANALOG_PIN),0,4095,0,3300);  
    Serial.println (mv);  
    // prints 0-4095 <-> 0 - 3300mV  
    delay(500);  
}
```



Q2 Circle what we would change to make the code read ADC channel 3 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;
#define ANALOG_PIN A4
void setup() {
    Serial.begin(115200);
    Serial.print("ESP32 Analog Pin - GPIO");
}
void loop() {
    int mv;
    mv = map(analogRead(ANALOG_PIN),0,4095,0,3300);
    Serial.println (mv);
    // prints 0-4095 <-> 0 - 3300mV
    delay(500);
}
```

Q4 (in Chat) What would 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).
- A real analog write would be a DAC – digital to analog converter. It would output smooth voltage, not a modulated wave form.
- **NOTE: This Arduino library function is not yet available for ESP32!**

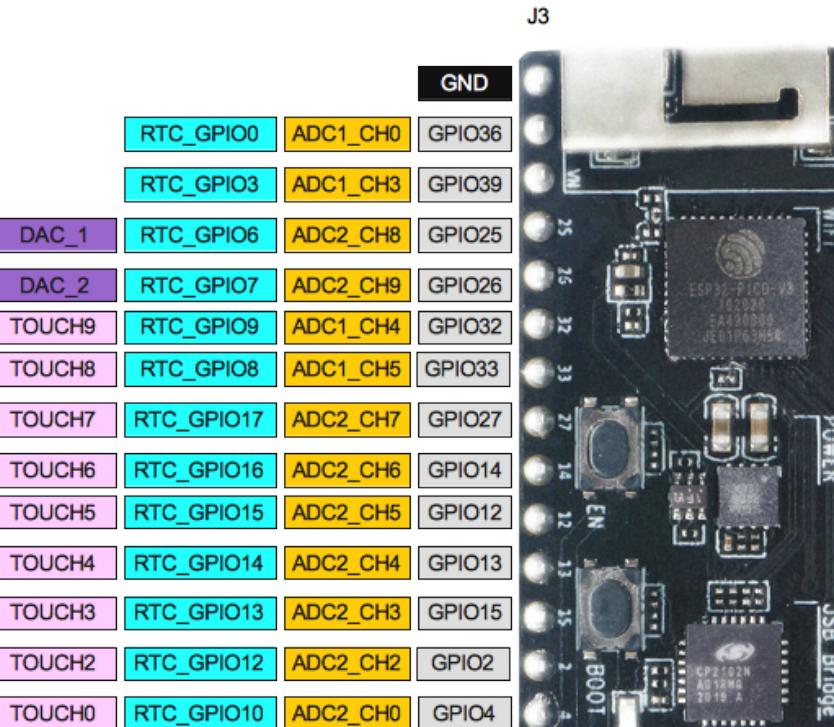
dacWrite() (a real “analog” Write)

- 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



RTC_GPIO0	ADC1_CH0	GPIO36
RTC_GPIO3	ADC1_CH3	GPIO39
DAC_1	RTC_GPIO6	ADC2_CH8
DAC_2	RTC_GPIO7	ADC2_CH9
TOUCH9	RTC_GPIO9	ADC1_CH4
TOUCH8	RTC_GPIO8	ADC1_CH5
TOUCH7	RTC_GPIO17	ADC2_CH7
TOUCH6	RTC_GPIO16	ADC2_CH6
TOUCH5	RTC_GPIO15	ADC2_CH5
TOUCH4	RTC_GPIO14	ADC2_CH4
TOUCH3	RTC_GPIO13	ADC2_CH3
TOUCH2	RTC_GPIO12	ADC2_CH2
TOUCH0	RTC_GPIO10	ADC2_CH0
		GPIO4

References online (www.arduino.cc)

The screenshot shows the Arduino Reference homepage. The top navigation bar includes the Arduino logo, a search icon, a shopping bag icon, and a menu icon. The main content area is organized into three columns:

- Control Structures** (orange header):
 - `setup()`
 - `loop()`
 - `if`
 - `if...else`
 - `for`
 - `switch case`
 - `while`
 - `do... while`
- Constants** (orange header):
 - `HIGH | LOW`
 - `INPUT | OUTPUT | INPUT_PULLUP`
 - `LED_BUILTIN`
 - `true | false`
 - `integer constants`
 - `floating point constants`
- Digital I/O** (orange header):
 - `pinMode()`
 - `digitalWrite()`
 - `digitalRead()`
- Analog I/O** (orange header):
 - `analogReference()`
 - `analogRead()`
 - `analogWrite() - PWM`

New ⌘N

Open... ⌘O

Open Recent ▶

Sketchbook ▶

Examples ▶

Close ⌘W

Save ⌘S

Save As... ⌘S

Page Setup ⌘P

Print ⌘P

Blink | A

Examples for any board

Adafruit Circuit Playgroup

Bridge

Ethernet

Firmata

LiquidCrystal

SD

Stepper

Temboo

RETIRED

Examples for NodeMCU

ArduinoOTA

BluetoothSerial

DNSServer

EEPROM

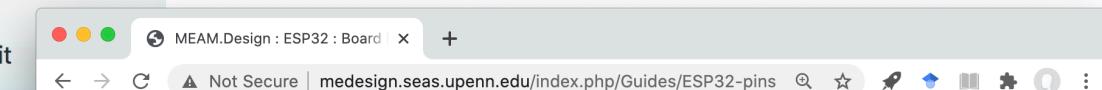
ESP32

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.

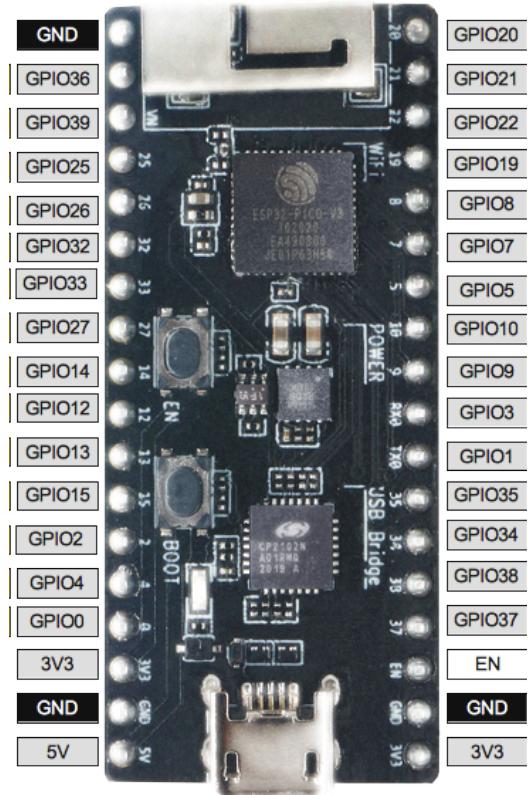


04

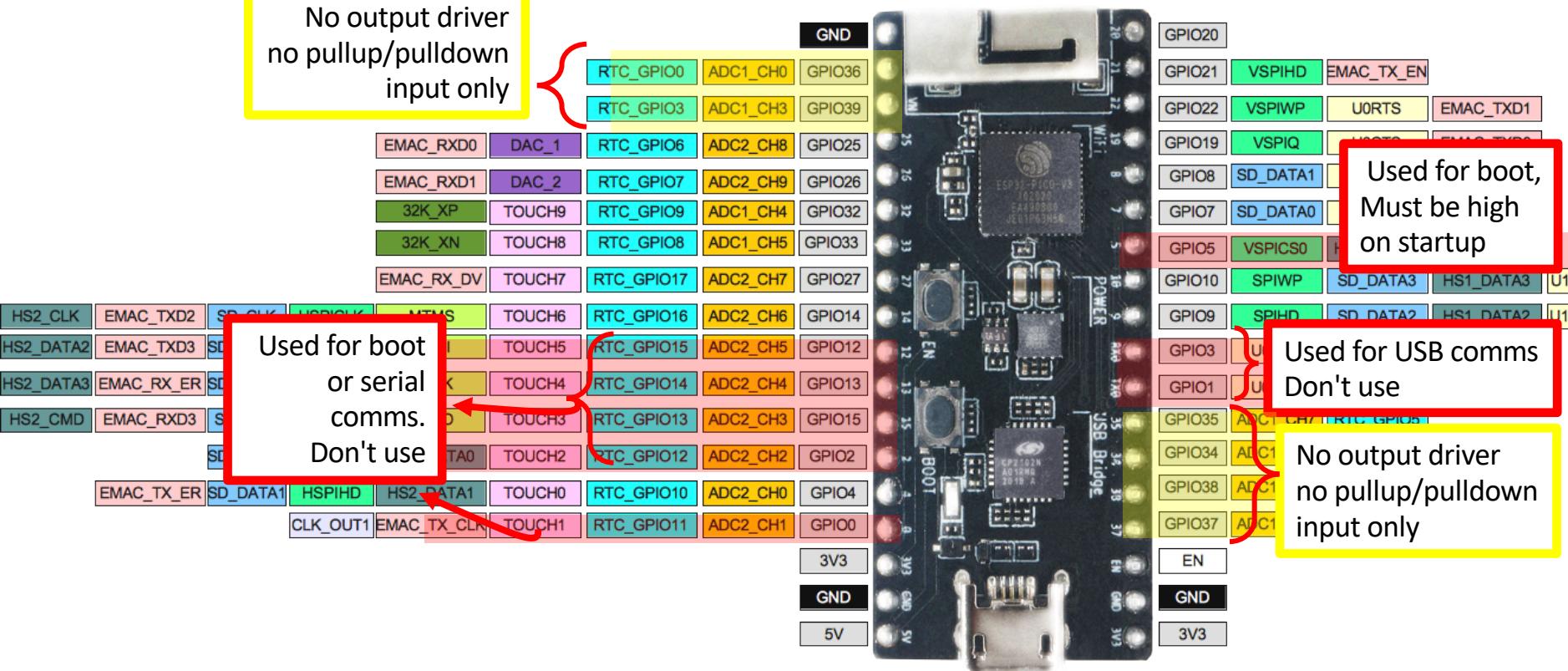
ESP32 PicoKit Hardware

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.

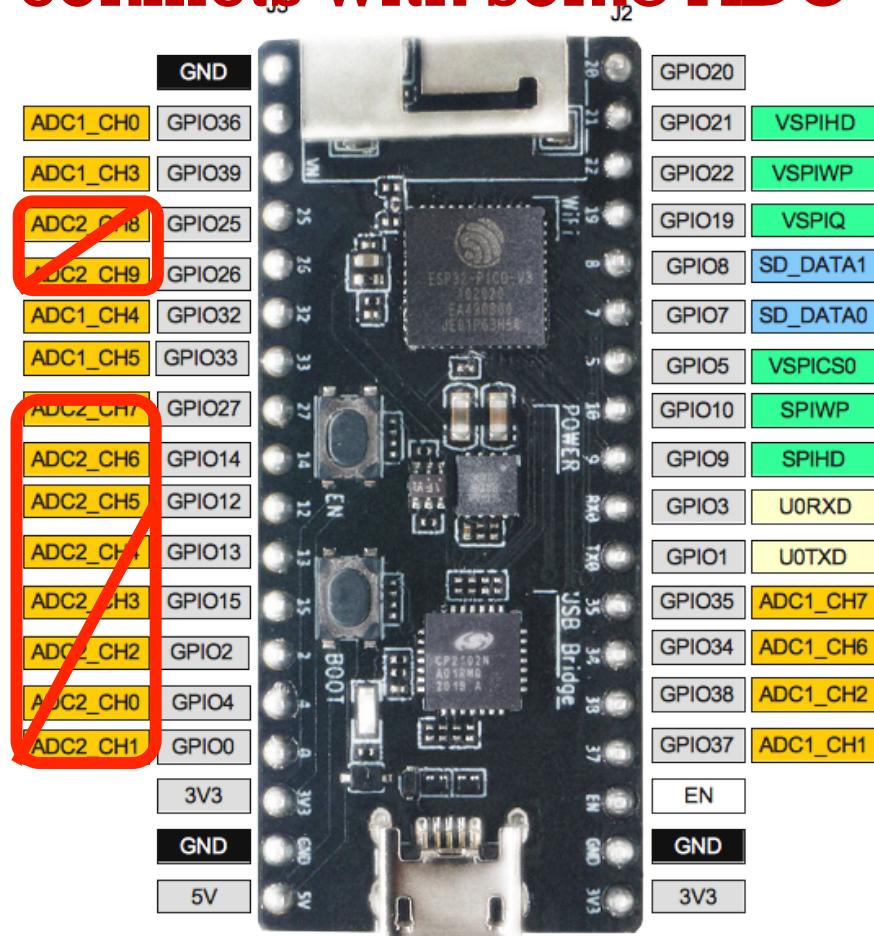


GPIO = General Purpose Input/Output
GND = ground or negative power line



More ESP Issues – WiFi conflicts with some ADC

- On PicoKit 18 ADC pins are available.
- 10 of them (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

ESP32 Pico Kit

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
- Phidget
- S62
- ESP8266
- ESP32

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.

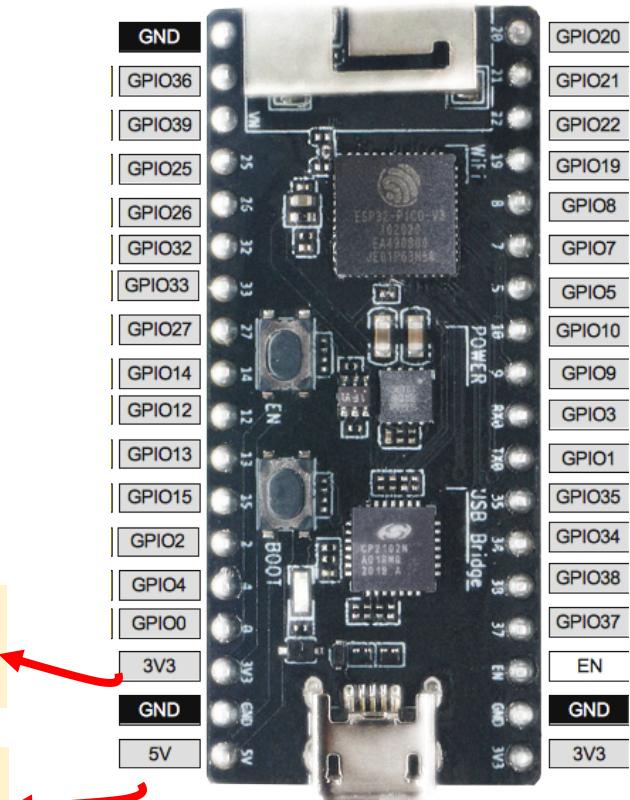


PicoKit w/external power (e.g. battery)

- 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



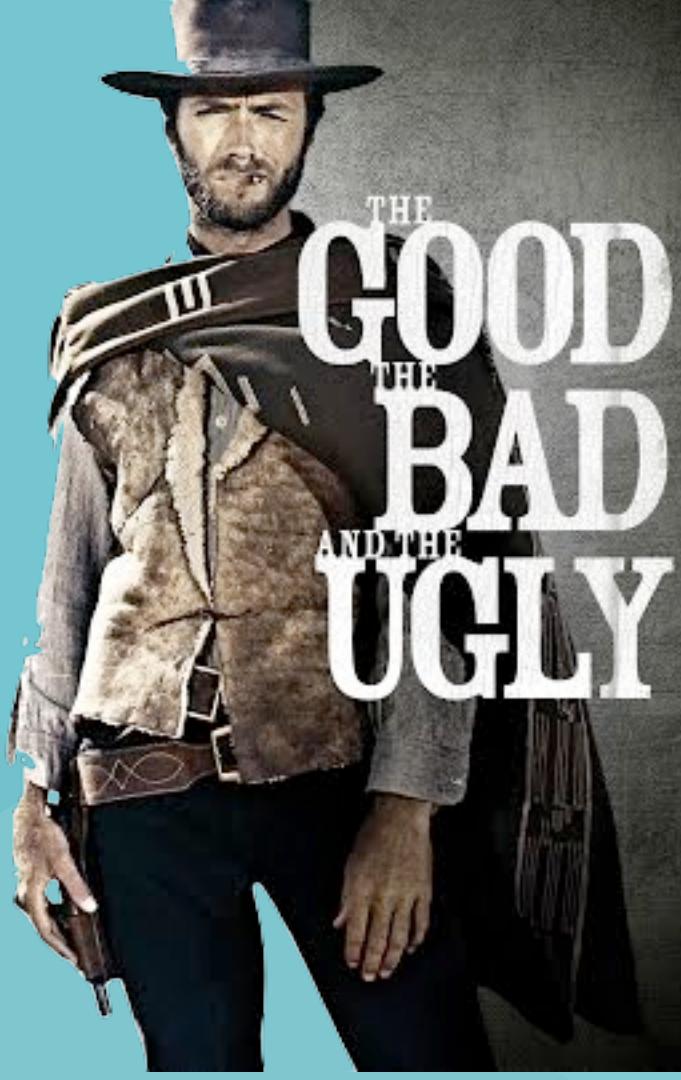
ESP32 Issues

- 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.**



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



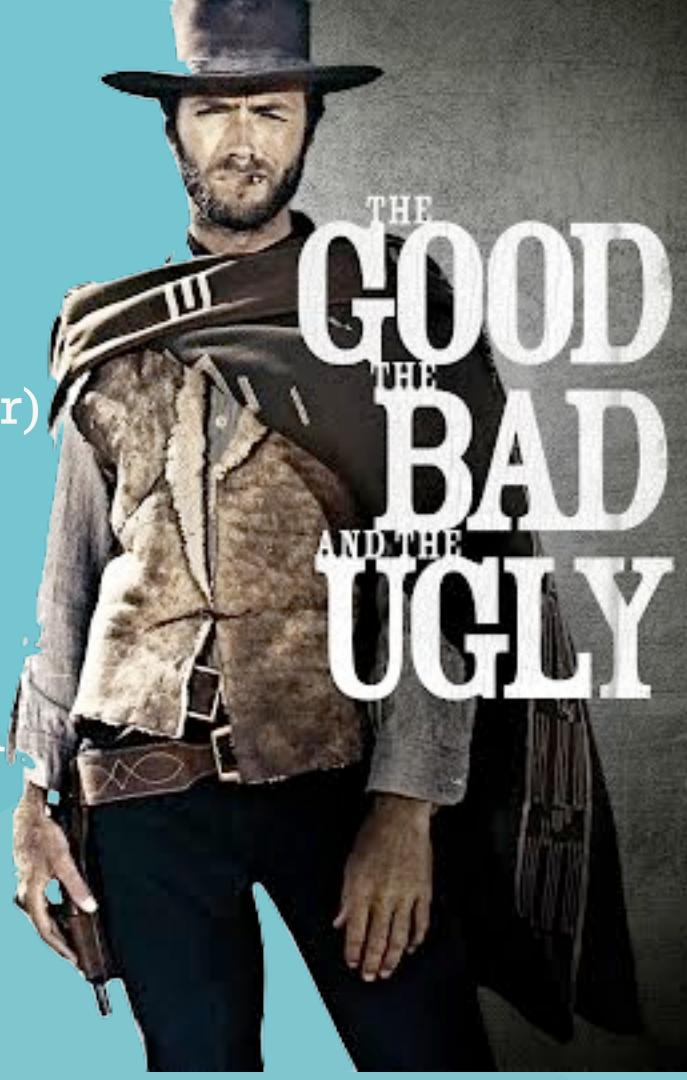
The Bad

1. WiFi only with port forwarding on router.
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 maybe 5V safe?)
 - A. Use 5V level shifters from sparkfun (elegant design)
2. Documentation is horrendous (getting better)
3. RTOS adds complication
4. Many ports are multi-use in default conditions (e.g. GPIO used for startup)
5. Ports can probably safely sink/source 10mA (vs 20mA for ATmega)
6. Randomly gets hot and dies... ESD?



NodeMCU version of ESP32 (we have some extras from previous years)



No Pin labels



Pin labels

Summary

- ESP32 is much more powerful than teensy.
- ESP32 uses 3.3V logic
- ESP32 documentation is very bad.
- When developing with ESP32 be careful of ESD!
- We will use Arduino SDK.