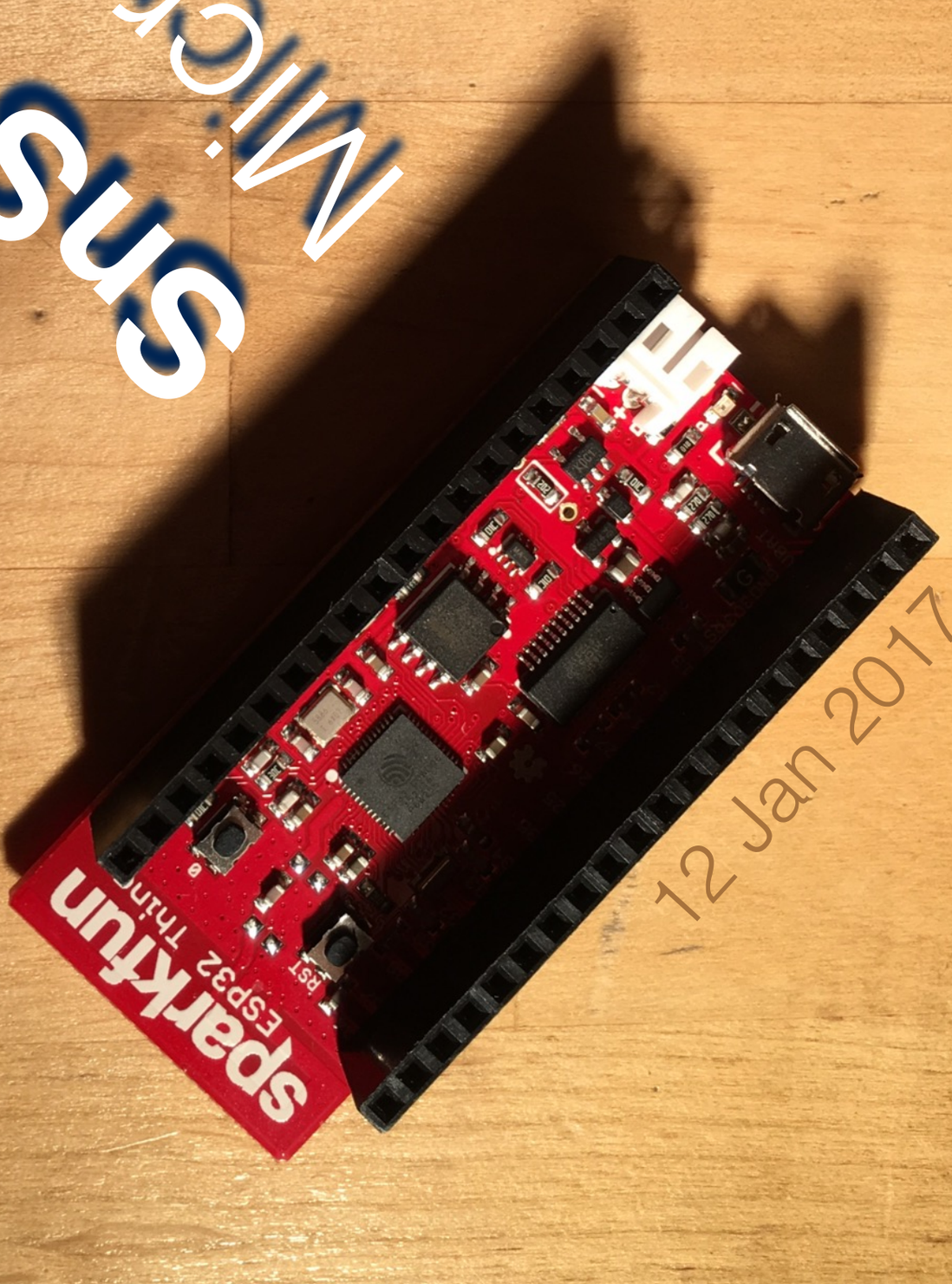
Sparkfun ESP32 Thing

Microcontrollers



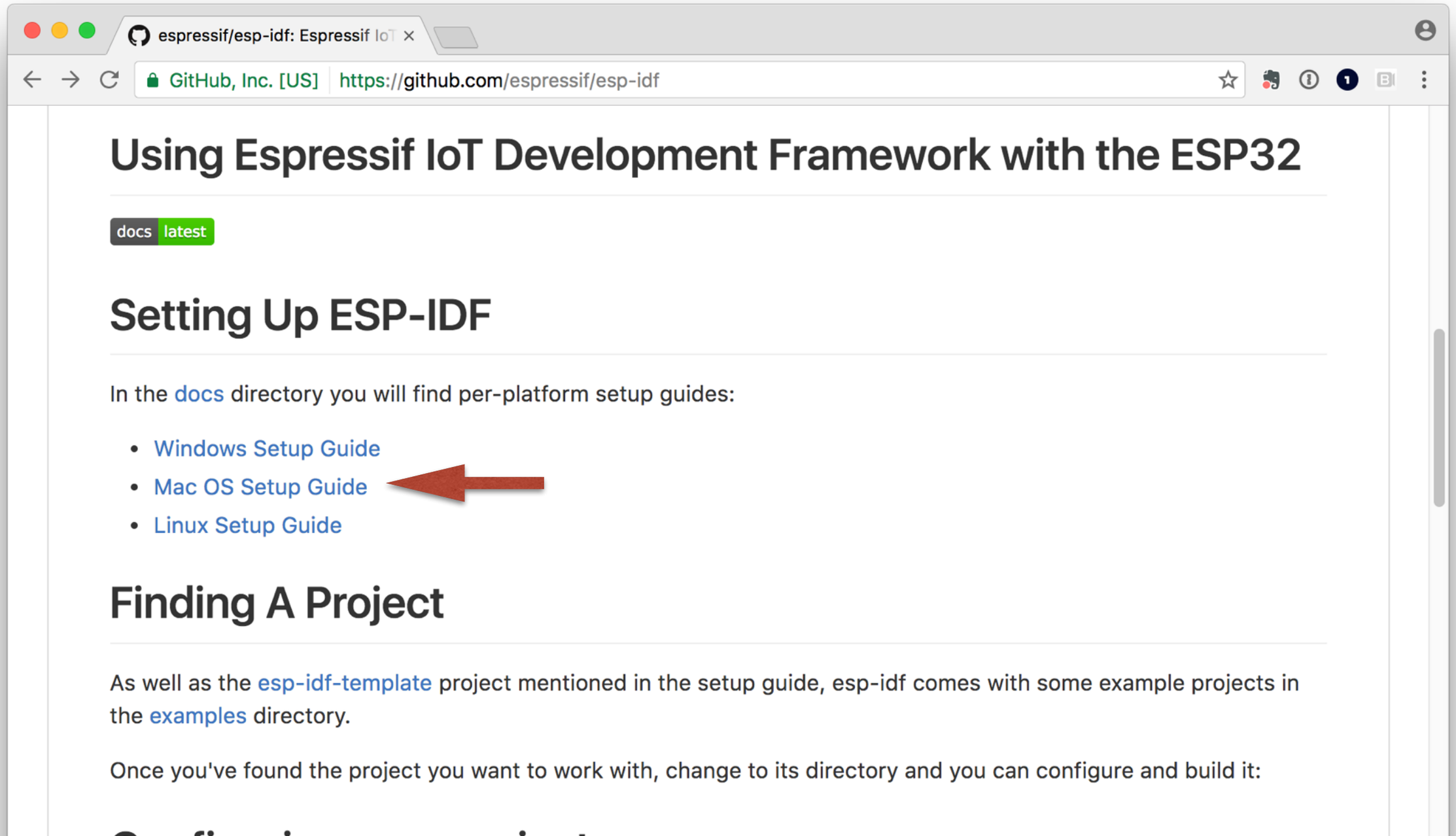
12 Jan 2017

Show of Hands

Outline

- Start Downloads
- Why Microcontroller?
- Introduction to ESP32 Hardware
- Software Environment
- Sensors
- Demos/Hands on

https://github.com/espressif/esp-idf



Microcontroller?

- Just a tiny computer.
- Typically no operating system or very minimal one
- Direct access to hardware
- Predictable instruction latency

Why microcontroller?

- Few layers between you and hardware
- Precise Timing
- Interact with devices
- Low power (battery powered)
- High MIPS/Watt

Why not microcontroller?

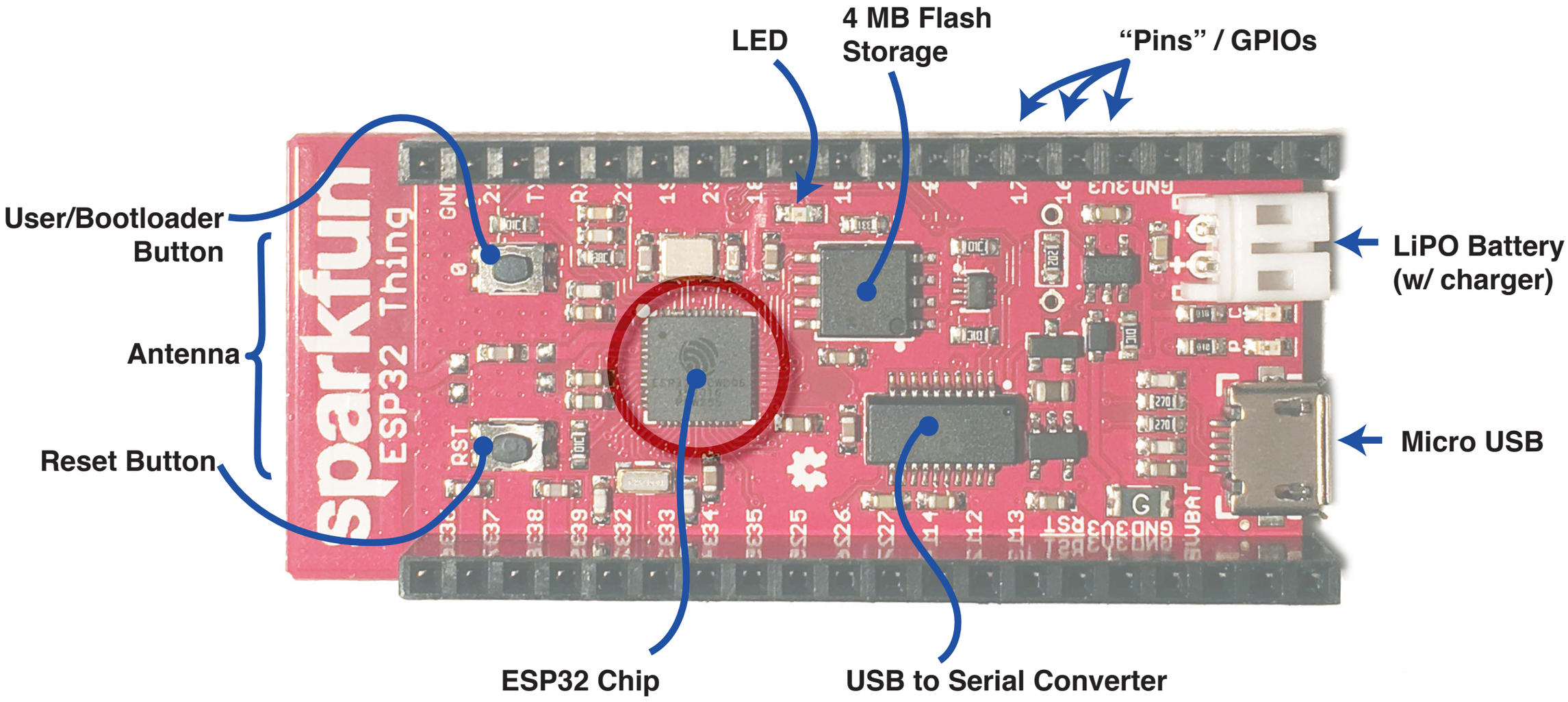
- Graphics
- Extensive user interaction
- Existing software
- Managed memory
- Always on, connected to mains power.
- High performance computation (pure MIPS)

Comparison

	Raspberry Pi 3	**ESP32**	**Arduino Uno**
Processor	1.2GHz quad-core ARMv8 (64 bit) ~2,200 DMIPS[1]	240 MHz dual-core Xtensa LX6 (32 bit) 600 DMIPS[2]	16MHz ATmega328P (8 bit) 5.2 DMIPS[3]
RAM	1GB DRAM	520 KB SRAM	2KB SRAM
Storage	SD Card (eg 32 GB)	Typ. 4 GB Flash (+ external)	32 KB
Networking	WiFi a/b/n Bluetooth low energy (BLE) Ethernet (integrated)	WiFi a/b/n Bluetooth low energy (BLE) Ethernet (external)	(none)
Peripherals/ Communications	HDMI, Audio UART I2C bus SPI bus	4 × SPI 2 × I²C, 2x I²S 3 × UART CAN bus 2.0 IR remote, 10x touch PWM LED, Motor Hall effect sensor more...	2x SPI 1x I²C 1x UART
Digital	27 pins	48 pins	14 pins
Analog	(none)	18 pins 12 bit ADC 2 pins 8bit DAC	6 pins 10 bit ADC
OS	Linux	Free RTOS	(none)
Electrical	3.3V	3.3V	5V
Cost	$40	$4 (module) $20 (dev board)	$25 official $5 clones

1) http://www.roylongbottom.org.uk/dhrystone%20results.html 2) https://espressif.com/en/products/hardware/esp32/overview 3) https://projectgus.com/2014/06/arduino-vs-raspberry-pi/

ESP 32 Thing

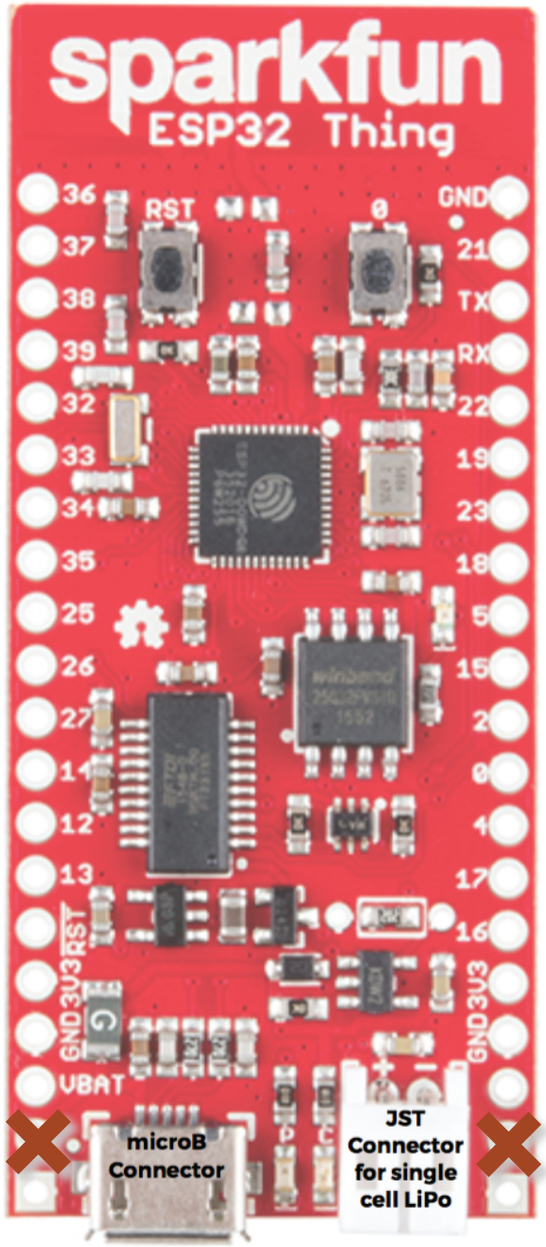

LED
4 MB Flash Storage
“Pins” / GPIOs
User/Bootloader Button
Antenna
Reset Button
LiPO Battery (w/ charger)
Micro USB
ESP32 Chip
USB to Serial Converter

PCB Antenna

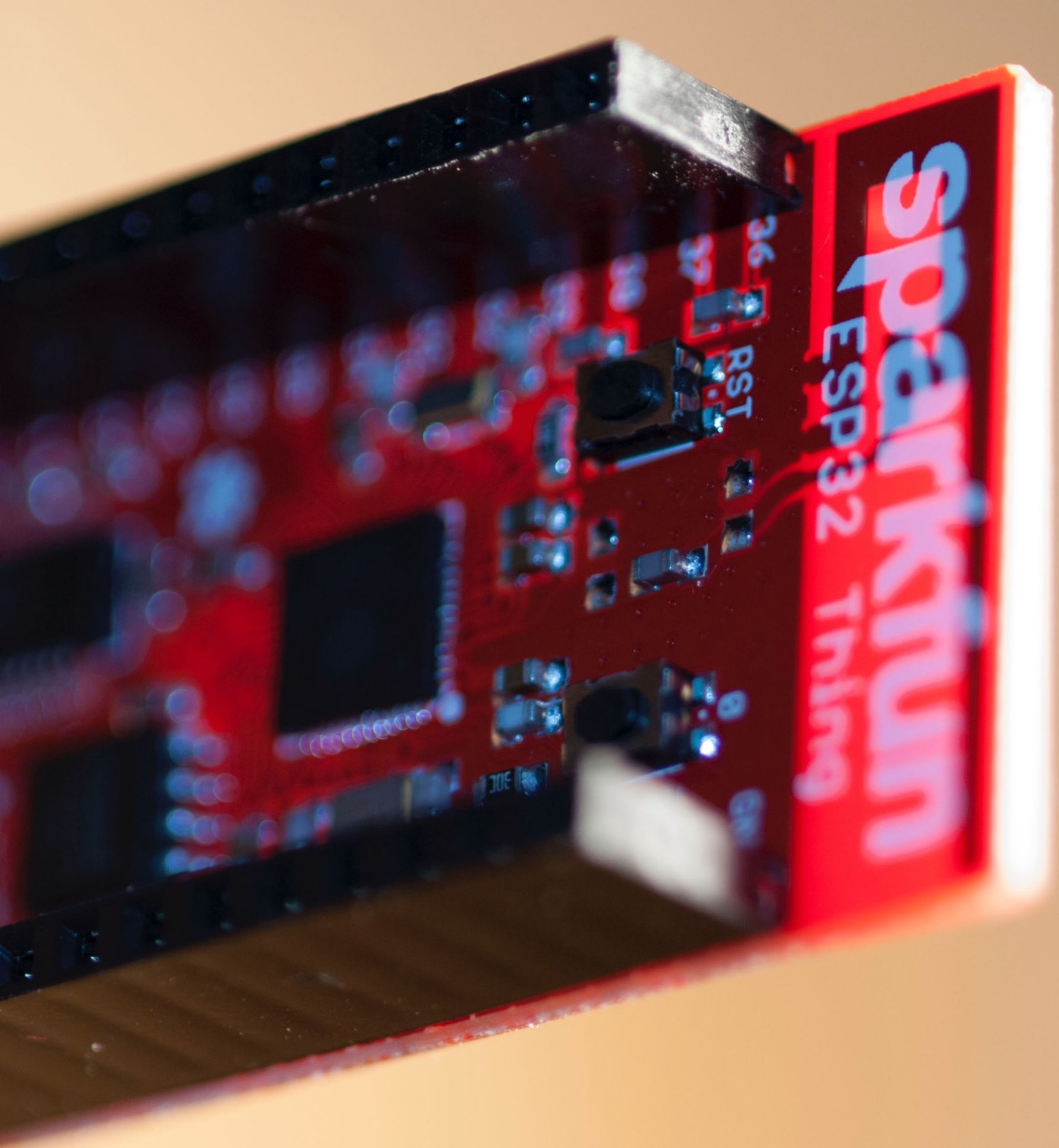
Reset button

Button: GPIO 0

	ADC1_0*	GPIO36*	SenseVP	36		GND				
	ADC1_1*	GPIO37*	CapVP	37		21	SDA	GPIO21	V_SPI_HD	
	ADC1_2*	GPIO38*	CapVN	38		TX	CLK3	GPIO1	U0_TXD	
	ADC1_3*	GPIO39*	SensVN	39		RX	CLK2	GPIO3	U0_RXD	
Touch9	ADC1_4	GPIO32	XTAL32	32		22	SCL	GPIO22	V_SPI_WP	U0_RTS
Touch8	ADC1_5	GPIO33	XTAL32	33		19	MISO	GPIO19	V_SPI_Q	U0_CTS
	VDET1	ADC1_6	GPIO34*	34		23	MOSI	GPIO23	V_SPI_D	
	VDET2	ADC1_7	GPIO35*	35		18	SCK	GPIO18	V_SPI_CLK	
	DAC1	ADC2_8	GPIO25	25		5	GPIO5	V_SPI_CS0	LED (Blue)	
	DAC2	ADC2_9	GPIO26	26		15	GPIO15	ADC2_3	HSPI_CS0	Touch3
	Touch7	ADC2_7	GPIO27	27		2	CS	GPIO21	ADC2_2	HSPI_WP
Touch6	HSPI_CLK	ADC2_6	GPIO14	14		0	CLK1	GPIO0	ADC2_1	Touch1
Touch5	HSPI_Q	ADC2_5	GPIO12	12		4	GPIO4	ADC2_0	HSPI_HD	Touch0
Touch4	HSP_ID	ADC2_4	GPIO13	13		17	GPIO17	U2_TXD		
			Reset	RST		16	GPIO16	U2_RXD		
			3.3V	3V3		3V3	3.3V			
			GND	GND		GND	GND			
			VBAT	VBAT		VBAT	VBAT			
			VUSB	VUSB		VUSB	VUSB			
			GND	GND		GND	GND			

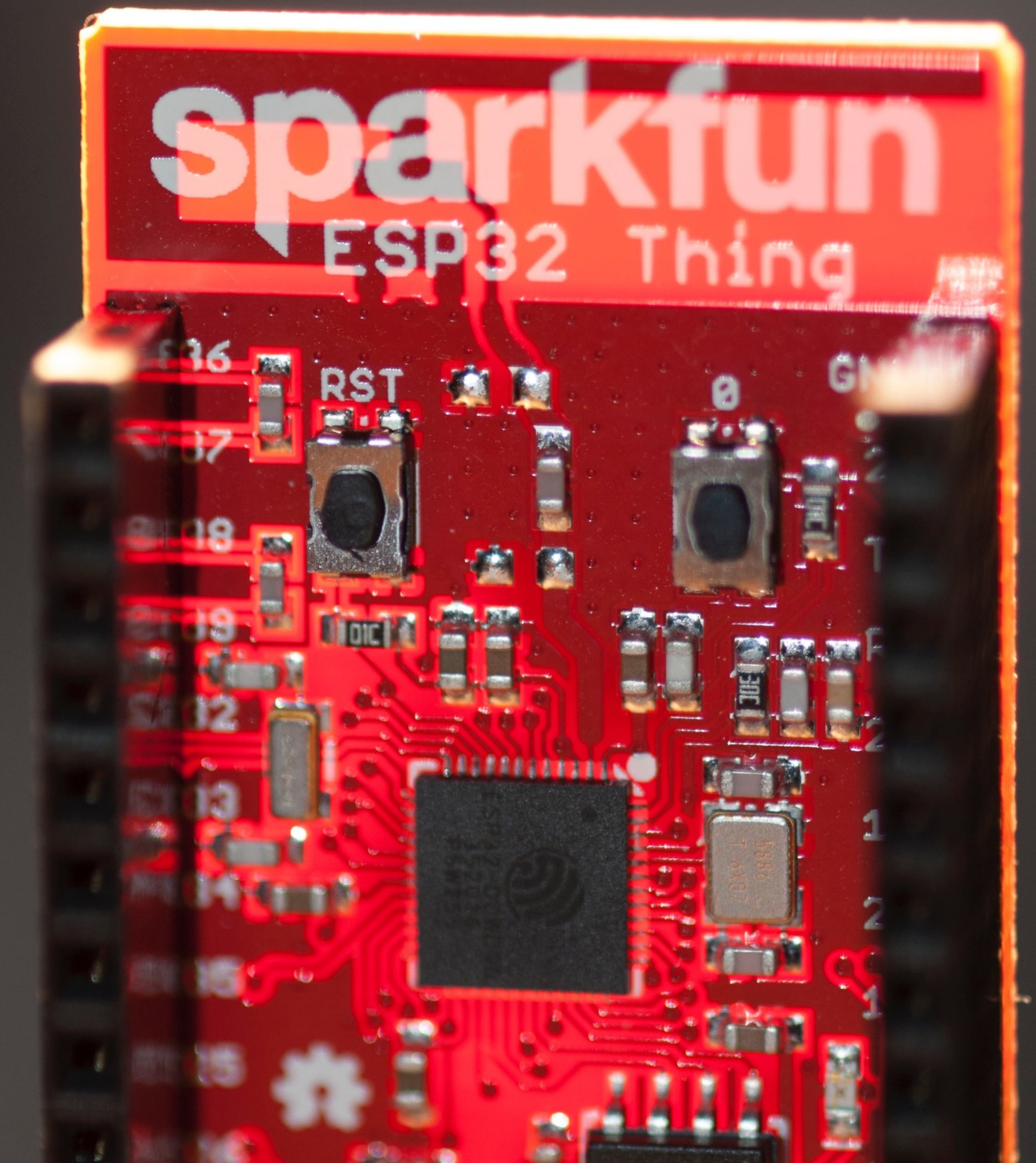


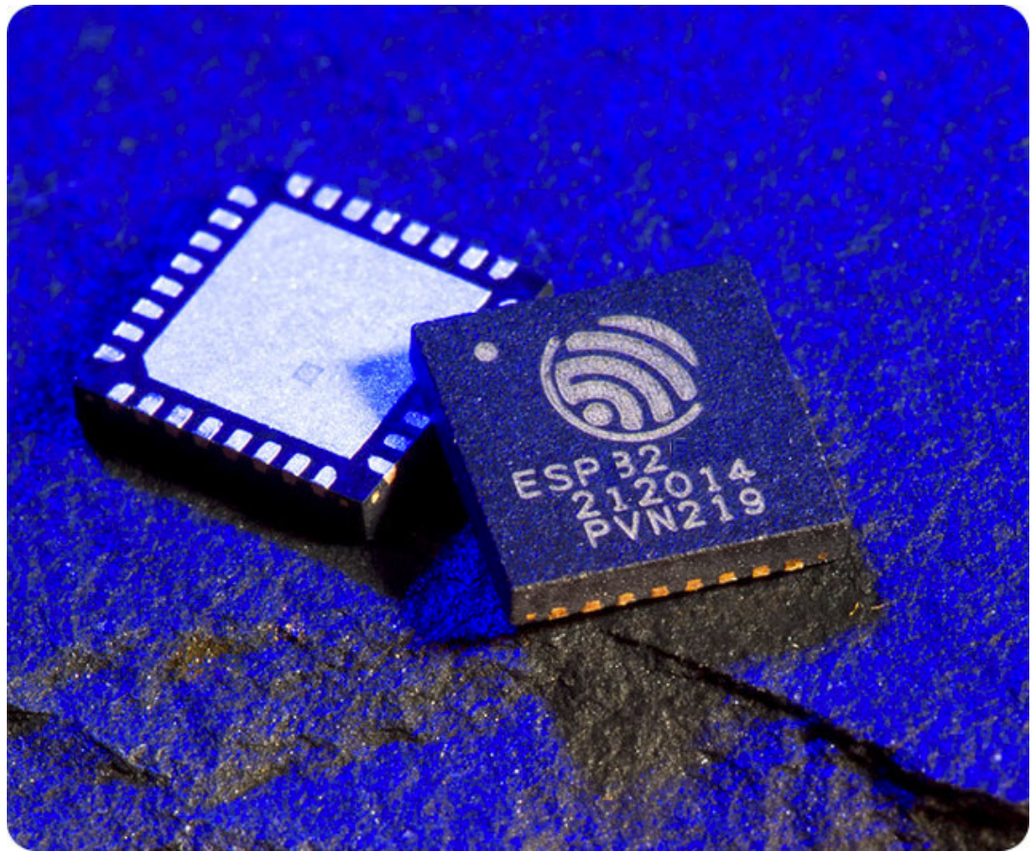
Power LED: Red
Charge LED: Yellow



Antenna

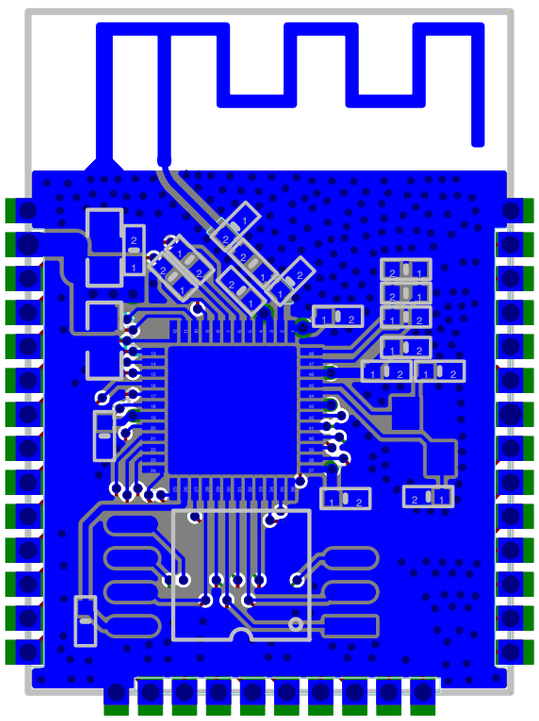
Feedline





What's in there?

- 2 CPUs
- RAM/ROM
- 2.4Ghz Radio
- Balun/TxRx Switch
- Many Peripherals



What else is needed?

- SPI Flash Chip
- ~12 passives
- Crystal (s)

Why ESP32?

- Dual Core
- Builtin WiFi/Bluetooth
- Cheap
- Can be low power
- Relatively open source
- A metric ton of peripherals
- Builds on success of earlier ESP8266, but much more capable.

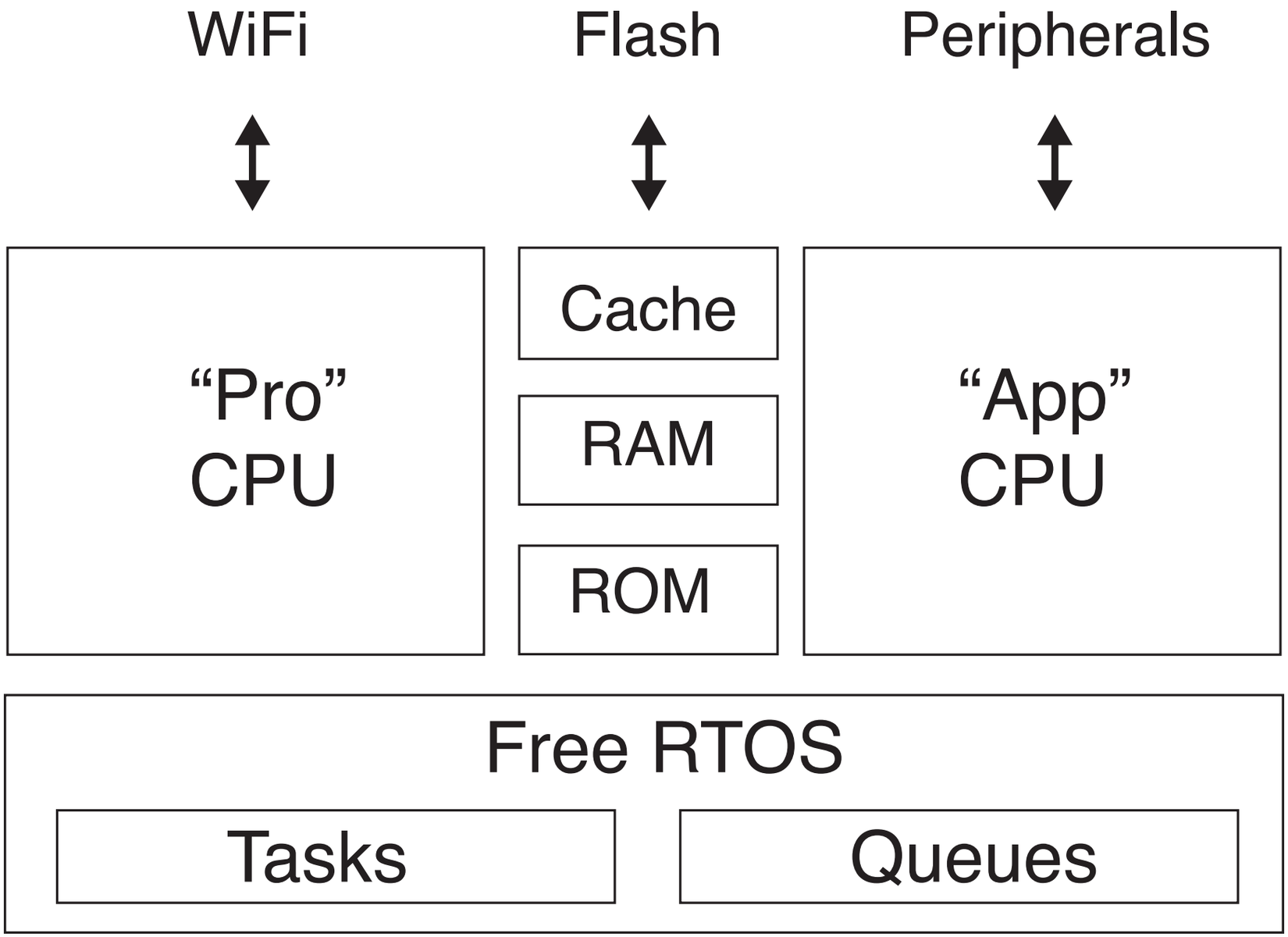
Peripherals

- WiFi (b/g/n, 150mb)
- Bluetooth (4.2 BR/EDR, BLE)
- 12-bit ADC 18 channels
- 2 × 8-bit D/A converters
- 10 × touch sensors
- Temperature sensor
- 4 × SPI
- 2 × I^2S
- 2 × I^2C
- 3 × UART
- SD Card (Host, Slave)
- Ethernet MAC
- CAN 2.0
- IR (TX/RX)
- Motor PWM
- LED PWM up to 16 channels
- Hall sensor
- Ultra low power analog pre-amplifier

Challenges

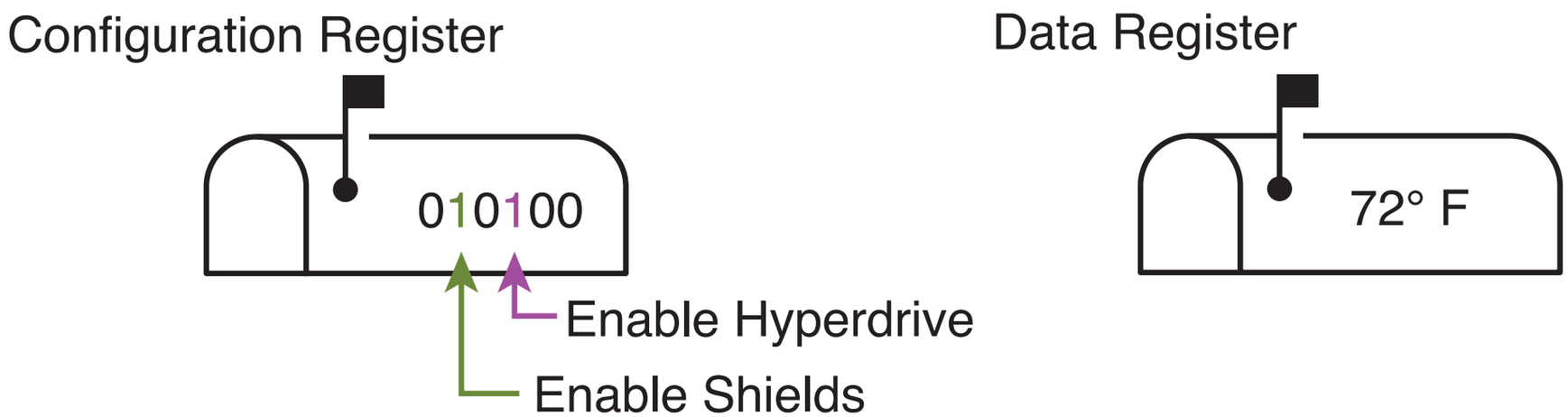
- Very new (docs/libraries still being written)
- Some silicon bugs (unlikely to affect you)
- Not a toy: lots to learn.

ESP32 Architecture

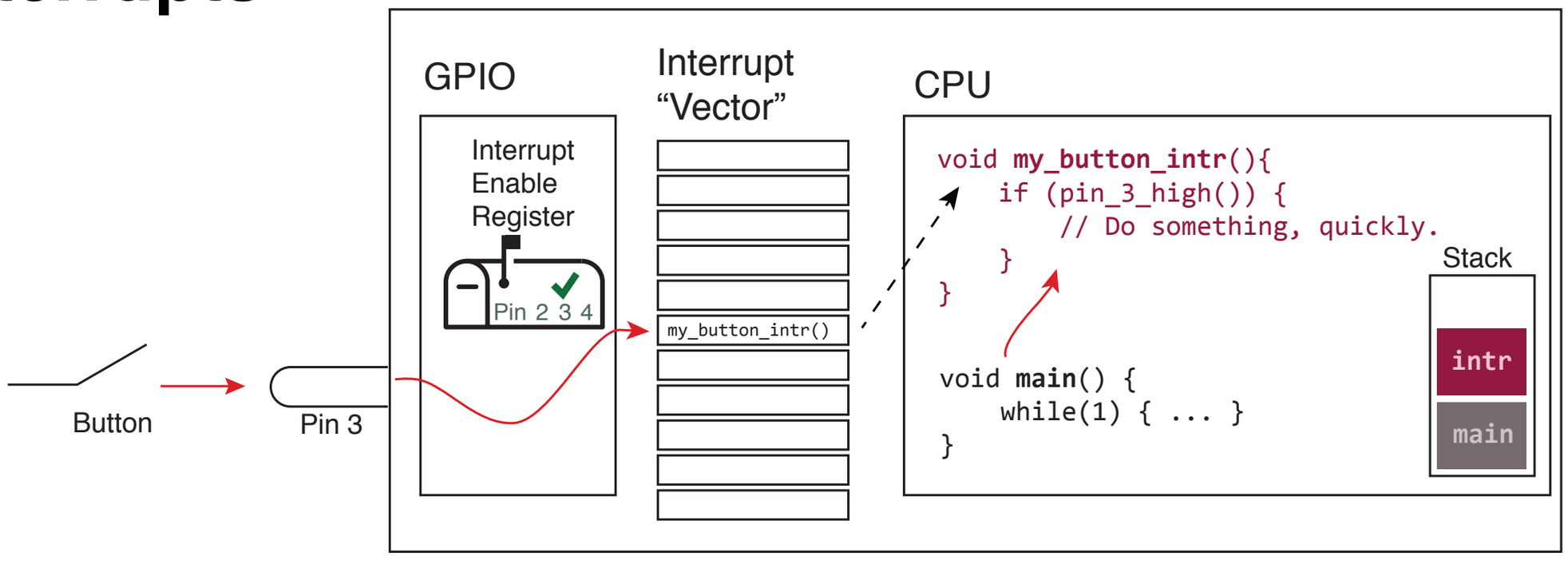


How to talk/listen to hardware?

Registers

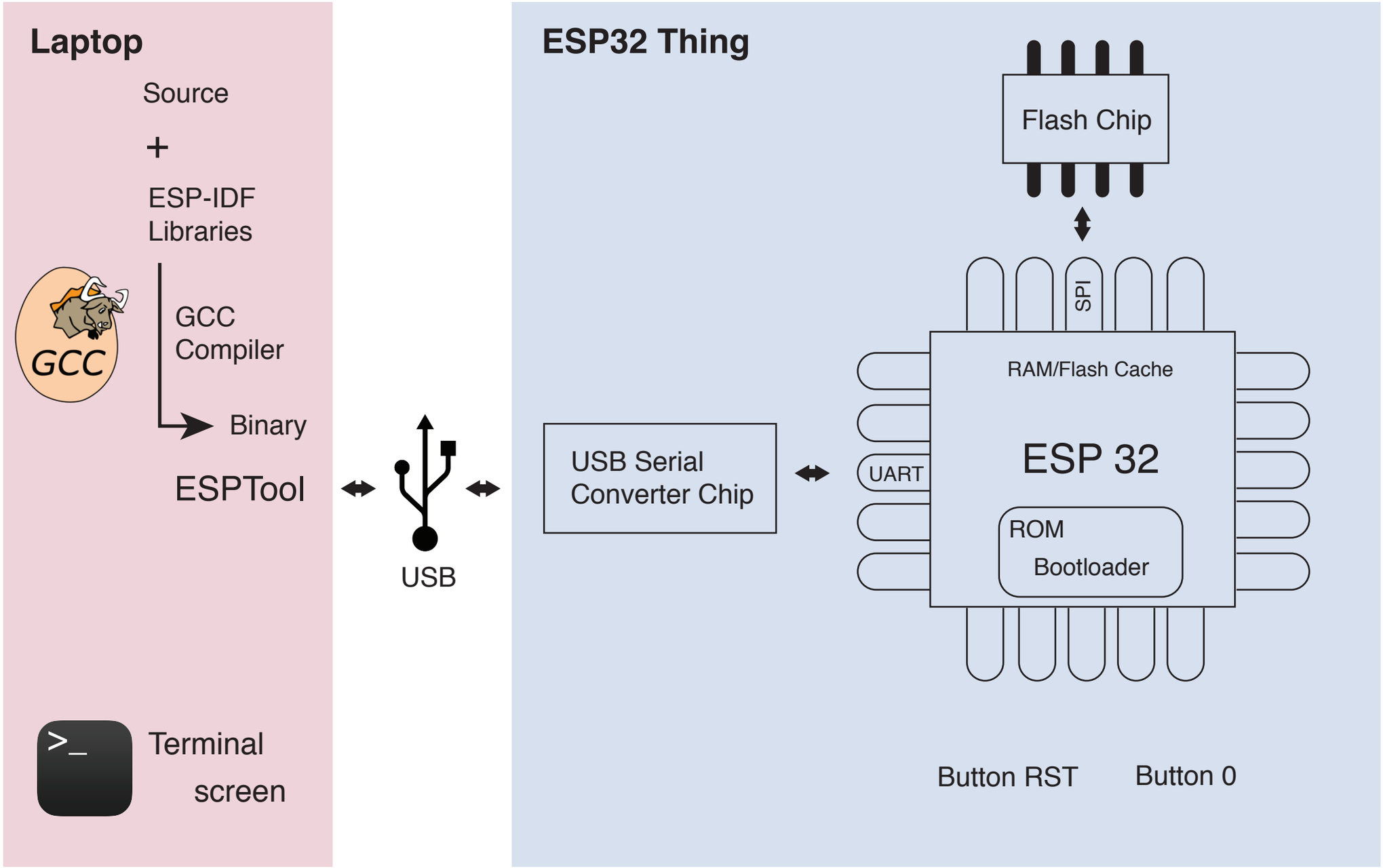


Interrupts



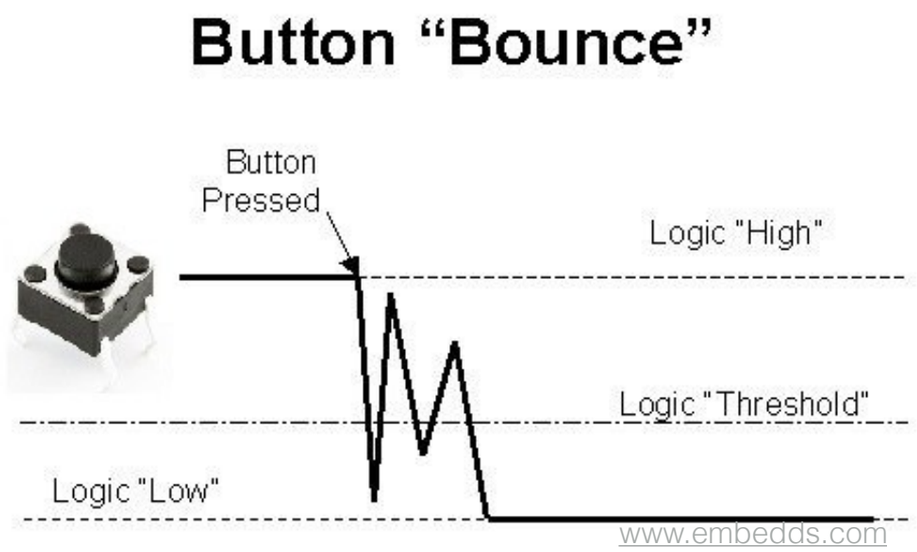
Memory Map

Development Environment



A tiny bit of electronics

- Voltage: **3.3V** (not 5V)
- Drive Current: **12 mA max** (datasheet p33) (relay)
- Polarity matters (+/-)
- Bounce:



Getting started

- `git clone --recursive` https://github.com/espressif/esp-idf
- Install Xtensa-GCC compiler from above link
- `export IDF_PATH=/path/to/above/checkout`
- `git clone --recursive https://github.com/cmason1978/esp32-examples`
- Plug ESP32 into USB port and `ls /tmp/tty.usb*`
- `make menuconfig`
- `make monitor`
- Ctrl-] to exit monitor

/Users/cmason/esp/esp-idf-template/sdkconfig - Espressif IoT Development Frame
> Serial flasher config

Serial flasher config

Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ]

(/dev/tty.usbserial-DN02BA3P) Default serial port
Default baud rate (921600 baud) --->
[*] Use compressed upload
Flash SPI mode (DIO) --->
Flash SPI speed (40 MHz) --->
Flash size (4 MB) --->
[*] Detect flash size when flashing bootloader
Before flashing (Reset to bootloader) --->
After flashing (Reset after flashing) --->
'make monitor' baud rate (115200 bps) --->

<Select> < Exit > < Help > < Save > < Load >

Some simple examples

```c
#include <stdio.h>
#include <freertos/FreeRTOS.h>
#include <freertos/task.h>
#include "esp_wifi.h"
#include "esp_system.h"

void app_main(void)
{
    gpio_set_direction(GPIO_NUM_5, GPIO_MODE_OUTPUT);
    int level = 0;

    while (true) {
        gpio_set_level(GPIO_NUM_5, level);
        level = !level;
        vTaskDelay(1000 / portTICK_PERIOD_MS);
        printf("Hello, world.\n");
    }
}
```

Troubleshooting

- `ld: cannot find -lrtc`

  Clone of esp-idf not recursive
  `cd esp-idf && git submodule update --init`

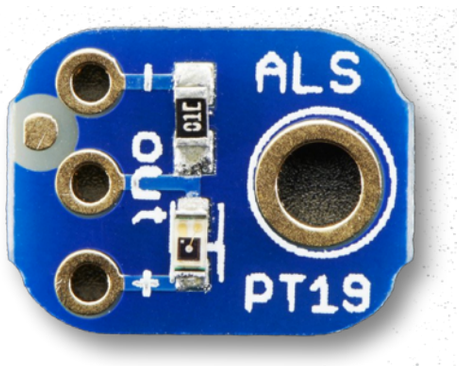
- `miniterm.py: error: no such option: --raw`

  pip install pyserial --upgrade

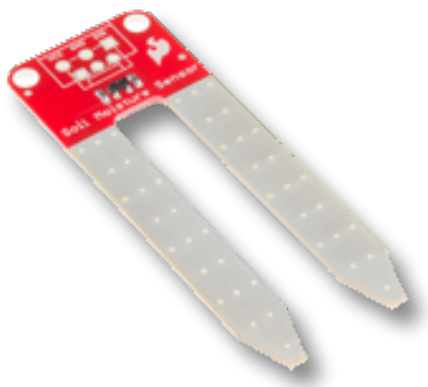
Sensors

Analog

Ambient Light



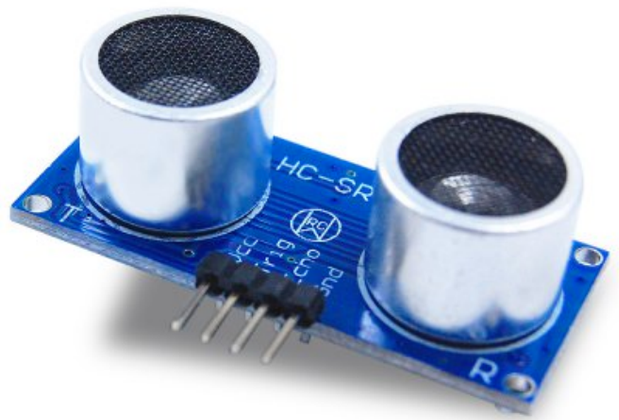
Soil Moisture



Heart Rate

Time-of-Flight

Ultrasonic Distance

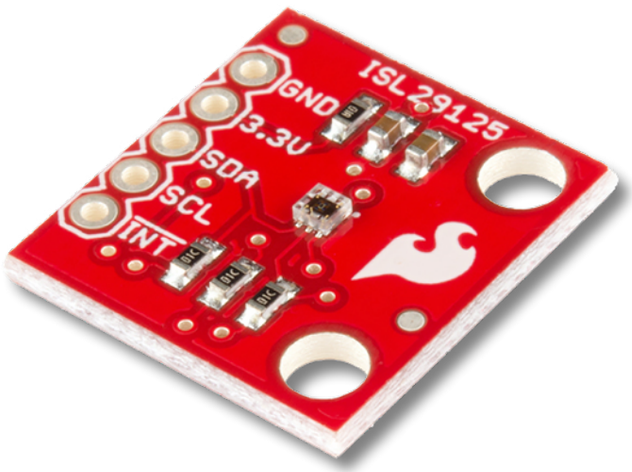


Digital

Temperature

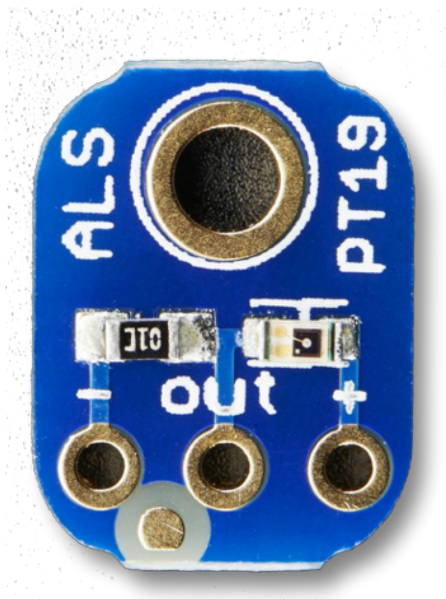
9 Axis IMU

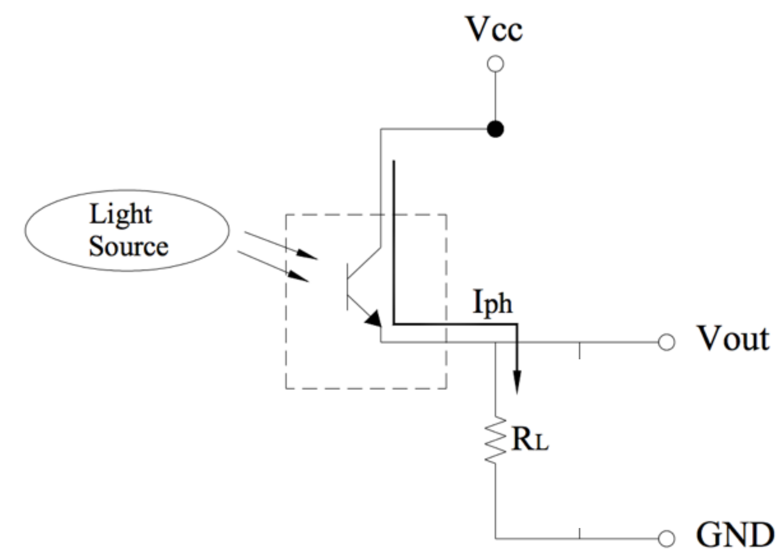
RGB Color

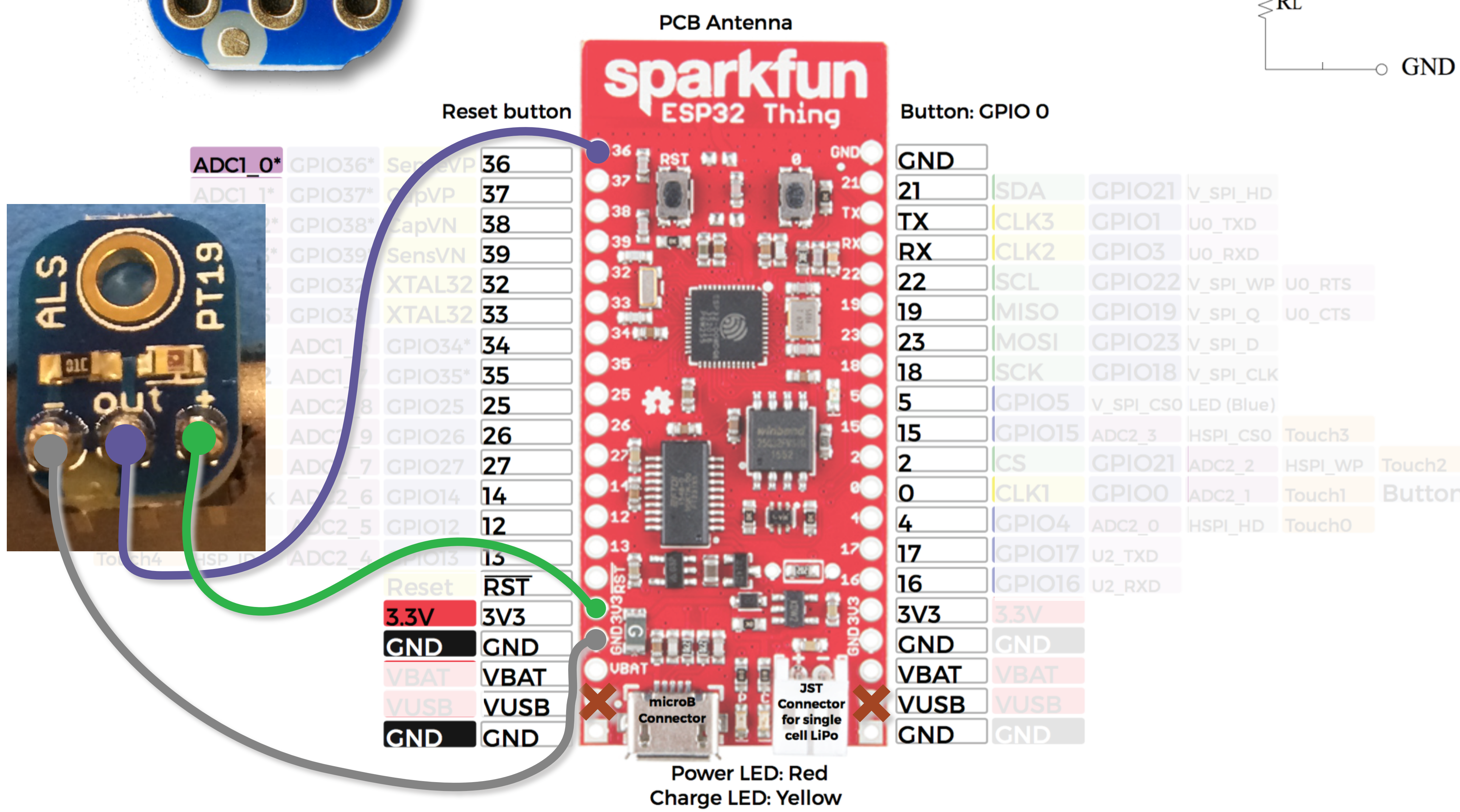


Analog Ambient Light Sensor

ALS-PT19

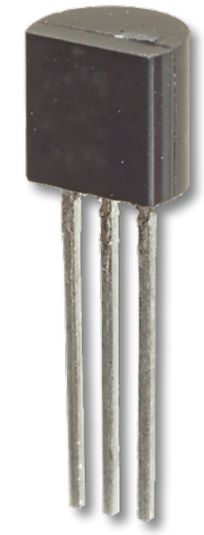






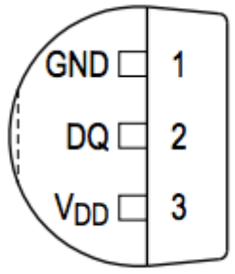
Digital Temperature Sensor MAX31820

“One Wire” Interface



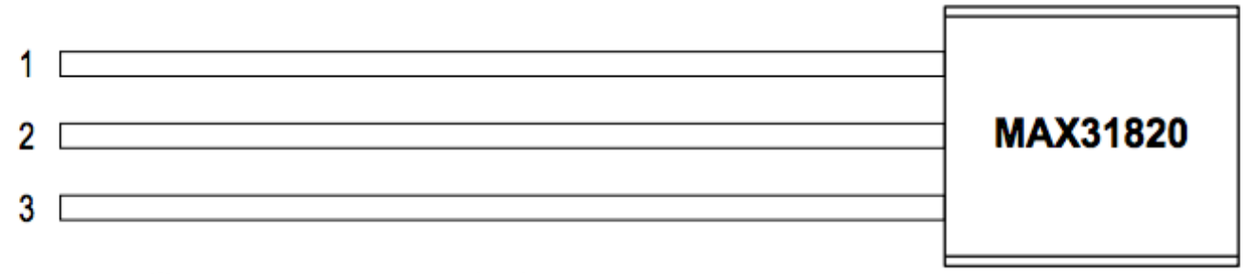
1 2 3

SIDE VIEW



TO-92

FRONT VIEW



1. Ground (-)
2. Data
3. Vdd (+)