Lecture 19

Power routing
Interrupts on ESP32

Agenda

01 Power Routing

02 Building website elements

03 Interrupts on ESP32

00 Demo and C++ Integer Type and Conversions (if time)

Stuff

- COVID19 is rising slightly at Penn. 31 new student cases last week (highest level in 5 weeks). Please be careful.
- Lab 4 field (same as final project field) is in GMlab.
- Level shifters are in ministore.

Purchasing

- Orders will be delivered to the GMLab. We plan to place them on the table near the entrance of the door.
- We plan to set up a video monitoring system so please only take what you know to be yours.
- The web ordering process is still in development. We may get a look as the project starts. Until then, continue to email *yim@seas* with your order.
- Please use the sample spreadsheet on Canvas → resources/

`PURCHREQ_20211012_Smith_MEAM510_example.xlsx`

Lab 4 Team Coaches

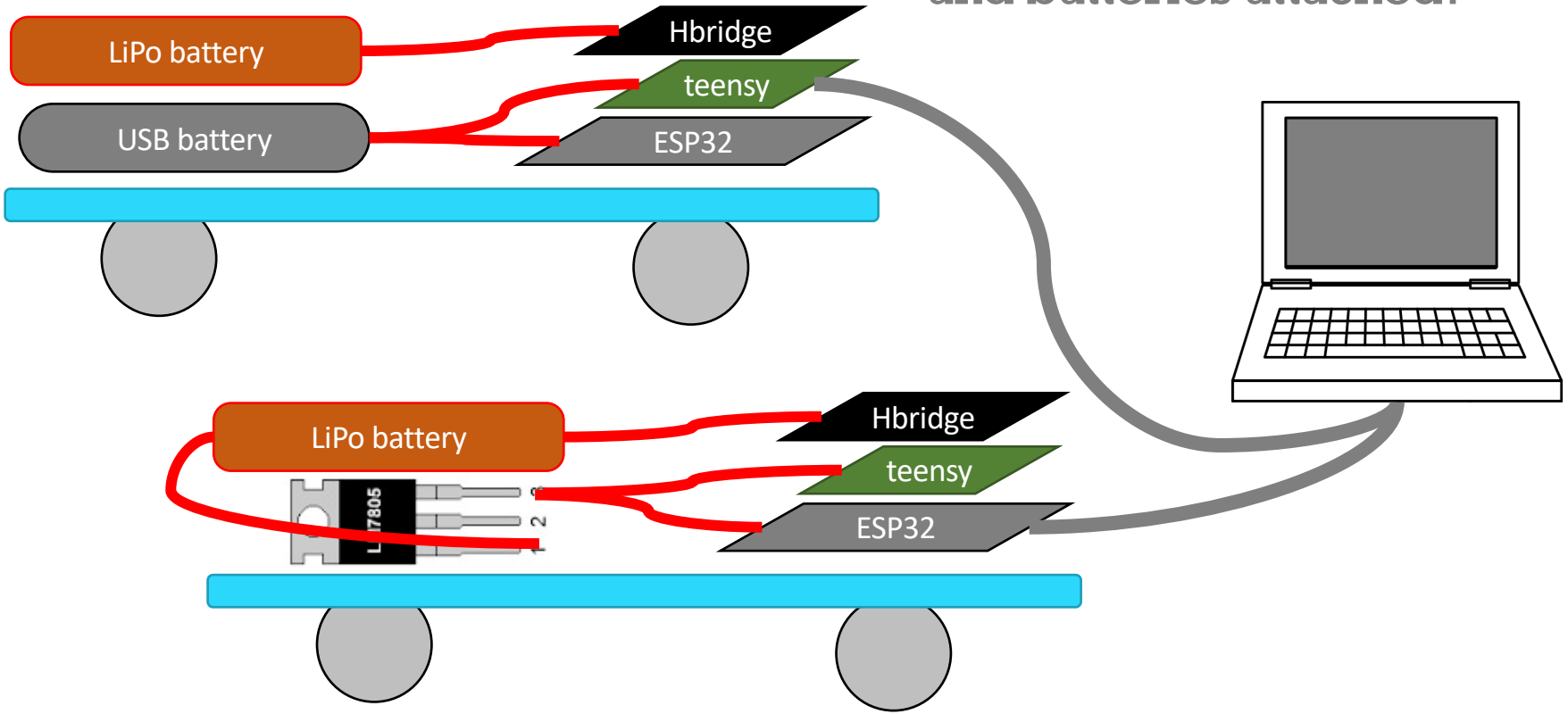
Lab 4 Group #	TA Coach
1	Ryan
2	Mark
3	Greg
4	Lili
5	Andreas
6	Zhenyu
7	Sheil
8	Zac
9	Brian
10	Will
11	Lili
12	Zhenyu
13	Greg
14	Zhenyu
15	Lili
16	Sheil
17	Zhenyu
18	Zac
19	Brian
20	Will
21	Andreas
22	Greg
23	Brian
24	Zac
25	Ryan
26	Brian
27	Andreas
28	Sheil
29	Sheil
31	Lili
32	Ryan
34	Will
36	Will
37	Andreas
40	Zac

01a

Power in ESP32 and Teensy

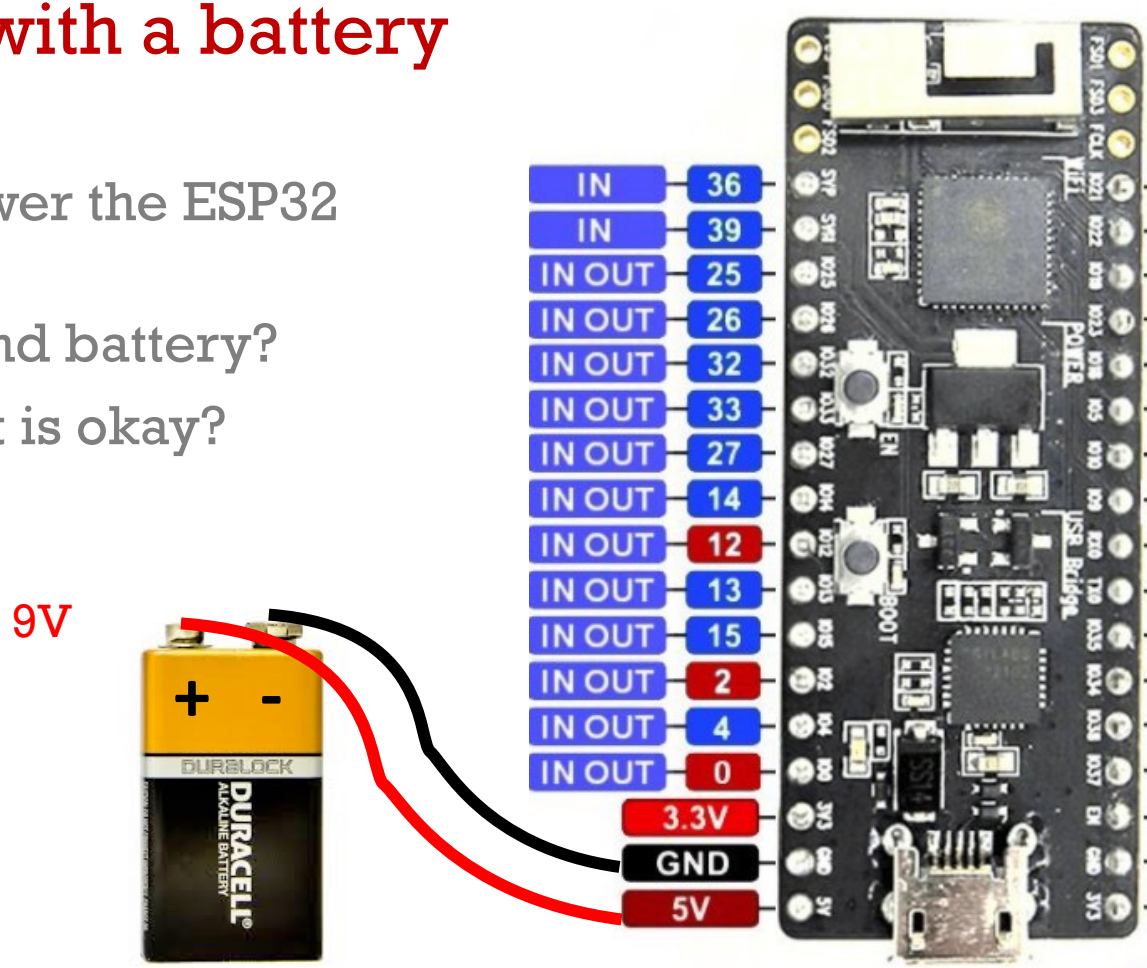
Powering system

- When developing code is it safe to have both laptop and batteries attached?



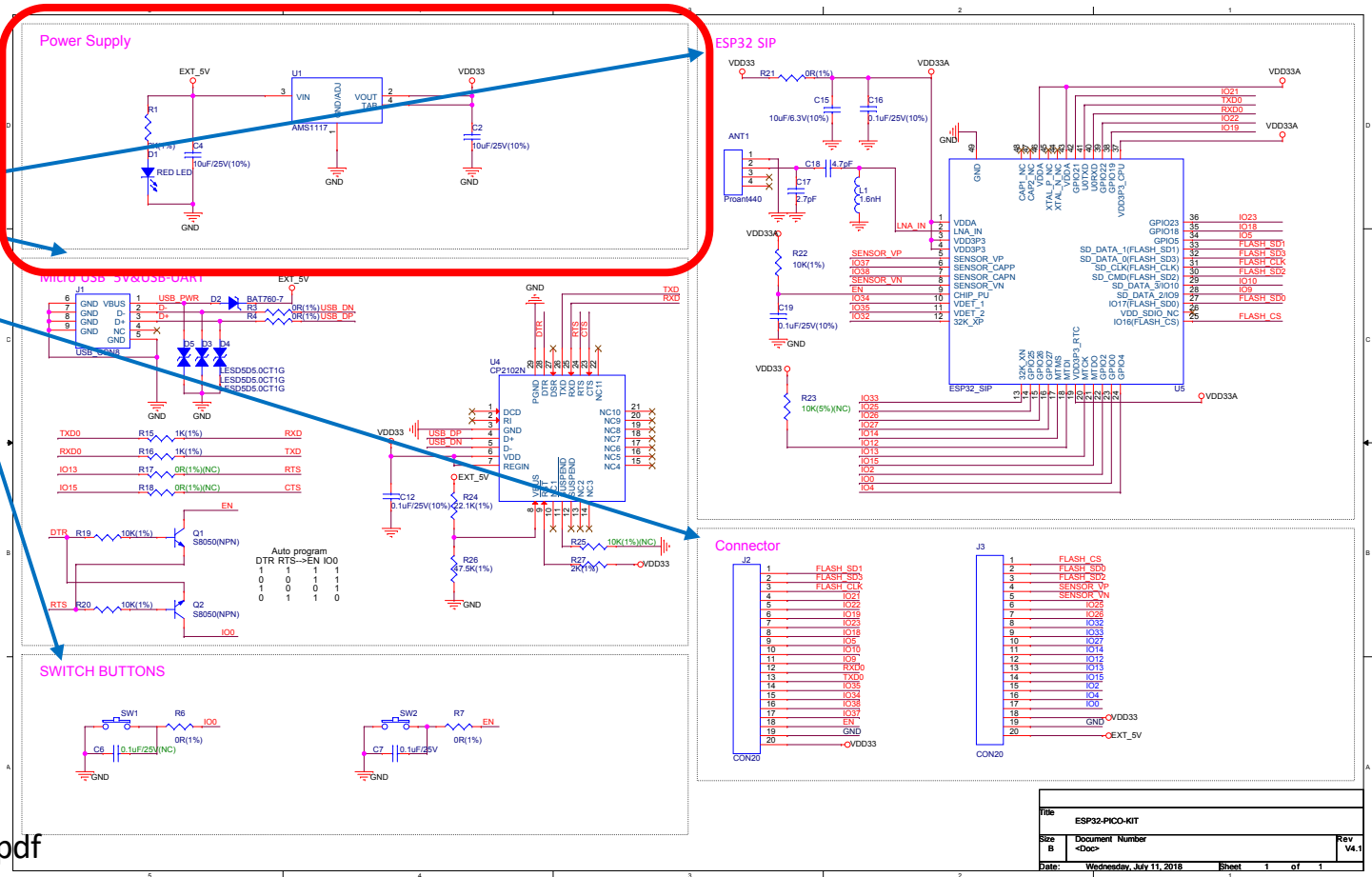
Powering boards with a battery

- Can we apply 9V to power the ESP32 PicoKit?
- Is it okay to have USB and battery?
- How can we find out if it is okay?



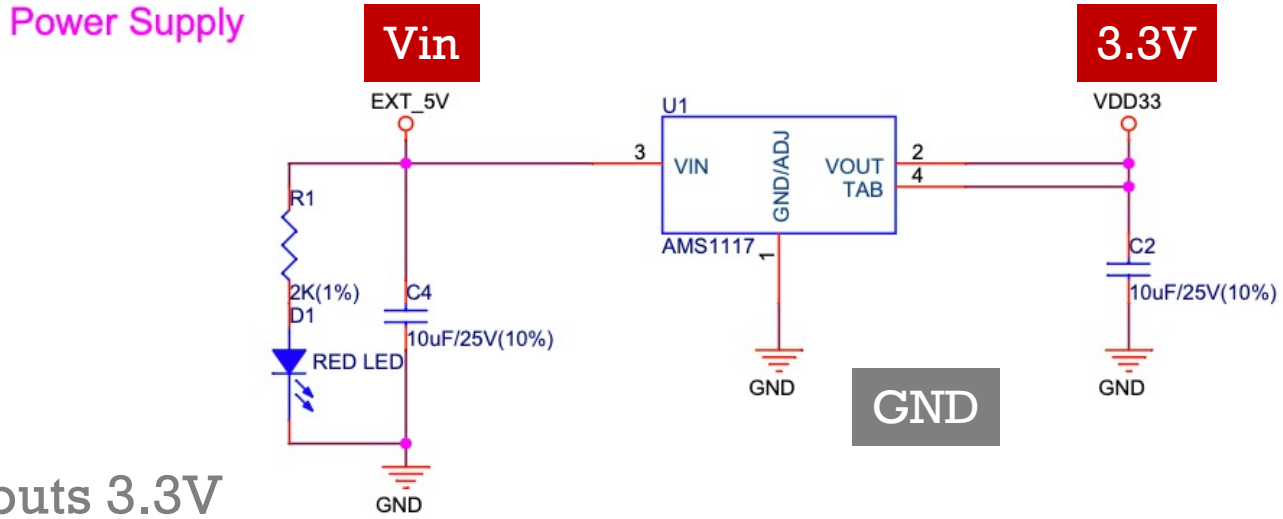
If we look at the board schematic what section(s) would likely tell us if we can use 9V?

A) Power Supply
B) ESP32 SIP
C) Micro USB 5V
D) Connector
E) Switch Buttons



On canvas:
esp32-piko-kit-v4.1_schematic.pdf

PicoKit Voltage Regulator: AMS1117

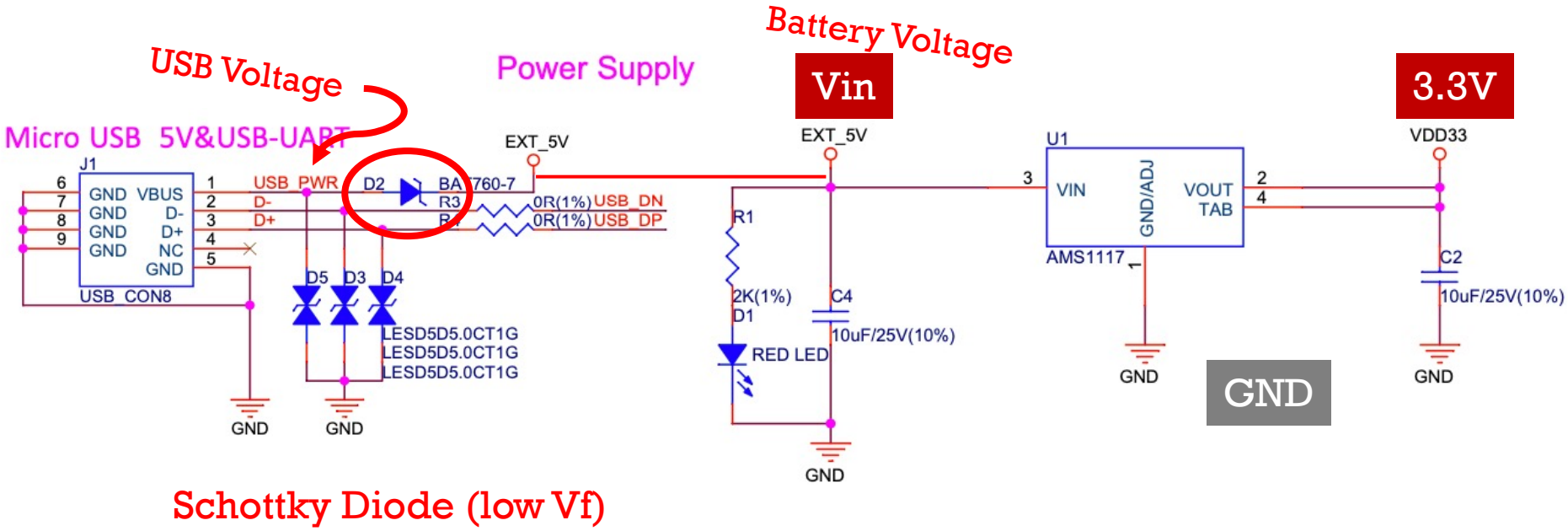


- AMS1117-3.3 outputs 3.3V
- Low Drop Out (LDO) means it can take closer to 3.3V as input
  - Dropout voltage is 1.1V@0mA to 1.3V@800mA (add this to 3.3V)
- Looking at datasheet: AMS1117 works with input:

  4.6V to 12V @ 300mA

On canvas: esp32-piko-kit-v4.1_schematic.pdf

Is it okay to use both USB and battery at Vin?

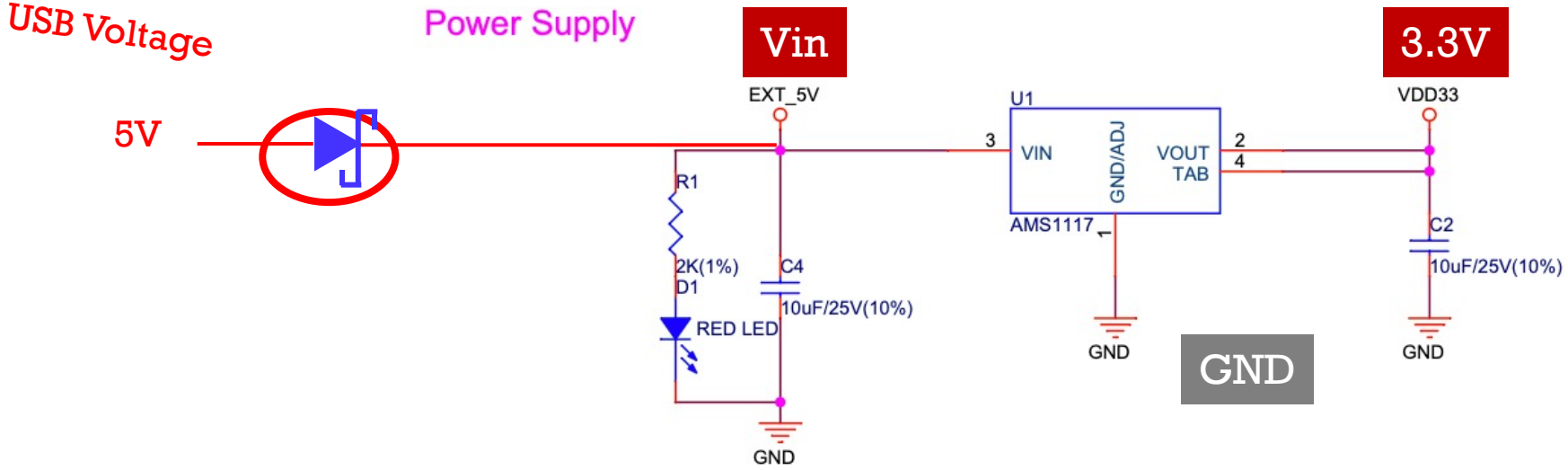


Schottky Diode (low Vf)

- Yes, the Schottky diode protects the USB port from any battery source you may put on the Vin line.

On canvas: esp32-piko-kit-v4.1_schematic.pdf

Q6 With USB power only, what will the voltage be at Vin?



Schottky Diode

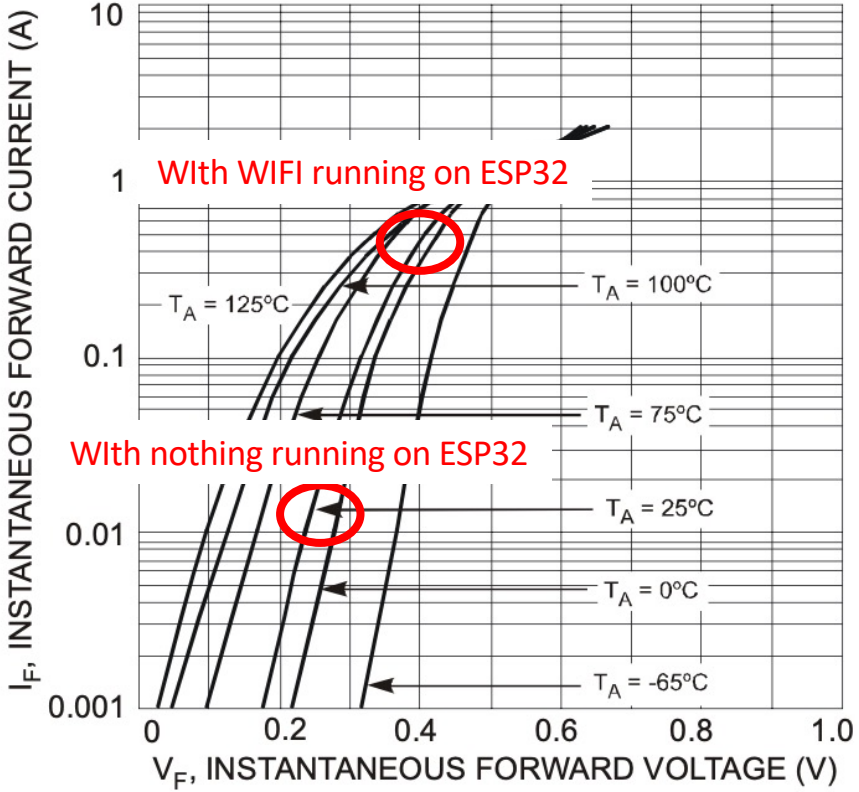
Have low Vf ranging from 0V to 0.7V typically

Depending on current through diode and temperature of diode

On canvas: esp32-piko-kit-v4.1_schematic.pdf

Schottky Diodes (PICO uses BAT760-7)

- Advantage of schottky is low Vf, but varies with load and temp.

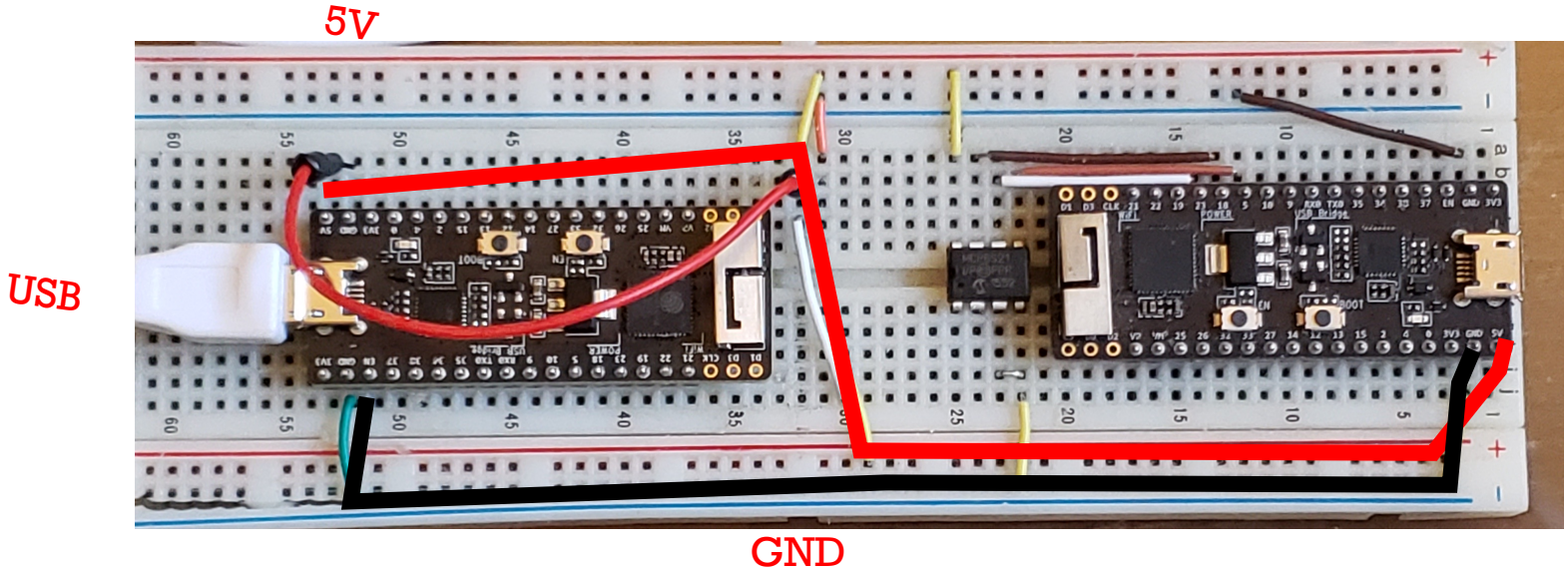


VF about 0.25V at 15mA without WiFi
VF about 0.4V at 500mA with WiFi

- Voltage from USB is nom. 5V
- Voltage at Vin = 5V – Vf = 4.6
- AMS1117 requires V > 4.6 worst case.
- It barely works… USB specs allow 5% at 5V. Cable resistance can play a factor.

Cascading power supply with ESP32?

- What if we want to power 2 ESP32s from one USB?
- 5V – 0.25V – 0.25V = 4.5V – okay to boot (barely).
- Turn on WiFi, 5V – 0.4 – 0.4 = 4.2V. NOT OKAY! Both crash.

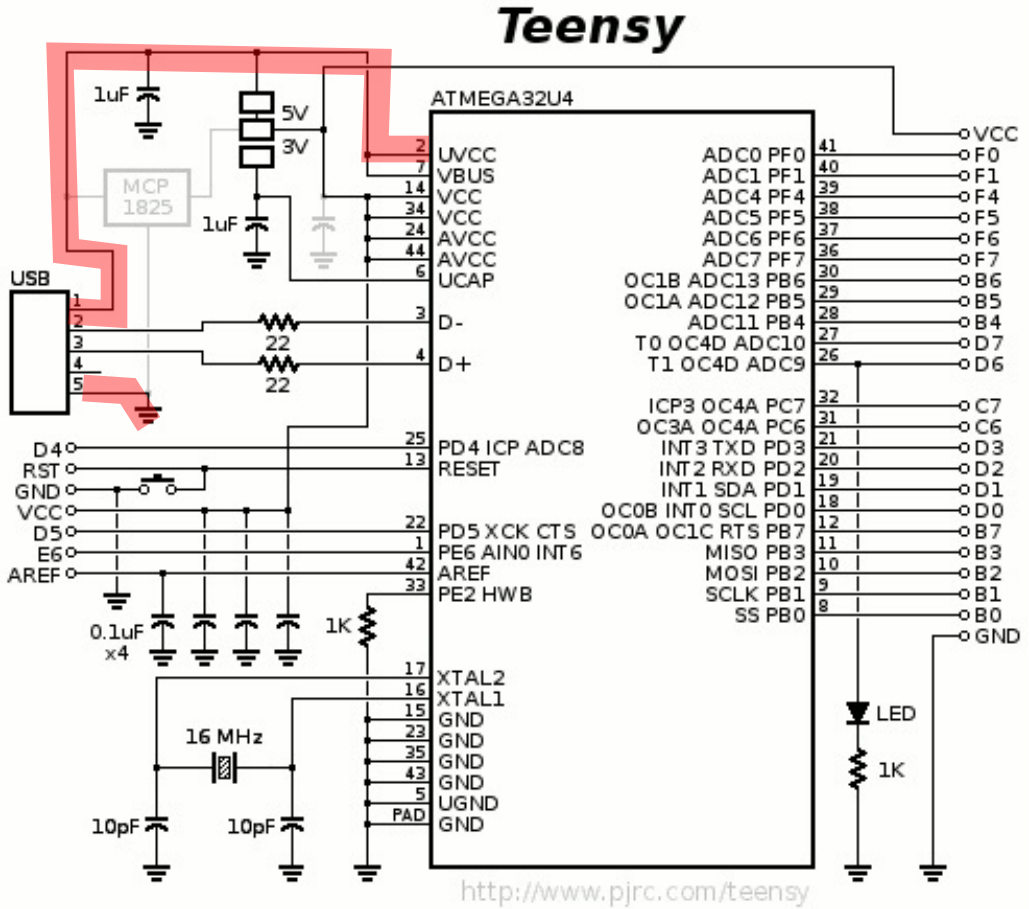


Solutions to borderline power issue:

- Simplest: avoid diode, add external 5V to 12V supply to Vin line on PICO (e.g. external battery) or power supply.
- Replace BAT760-7 with lower Vf (I found some about 0.1V lower) – hard to desolder SMD diode – not recommended
- Replace AMS1117 with lower LDO, but major hassle to see if it fits right form factor of surface mount component – hard to desolder SMD diode – not recommended
- Bypass the Schottkey diode. (jumper over it) Dangerous, but okay if you remember never to plug in USB when it is battery powered. (Teensy approach) - not recommended

What about Teensy?

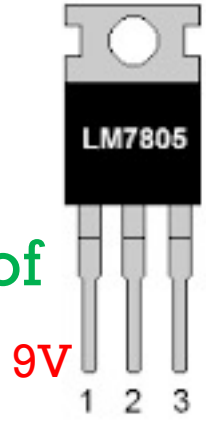
- USB 5V goes straight to VCC
- No protection, but ATmega32U4 has wide range of input.
- Current only limited by USB port, USB connector and traces on PCB

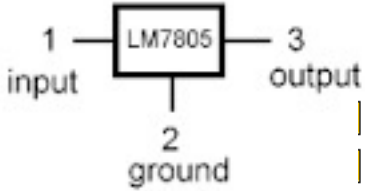


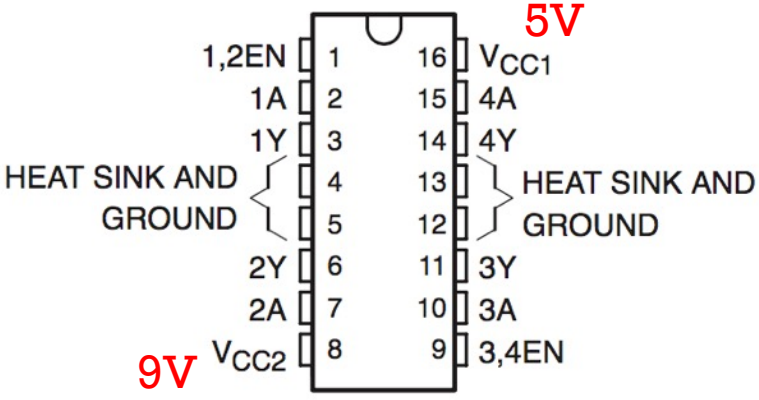
01b

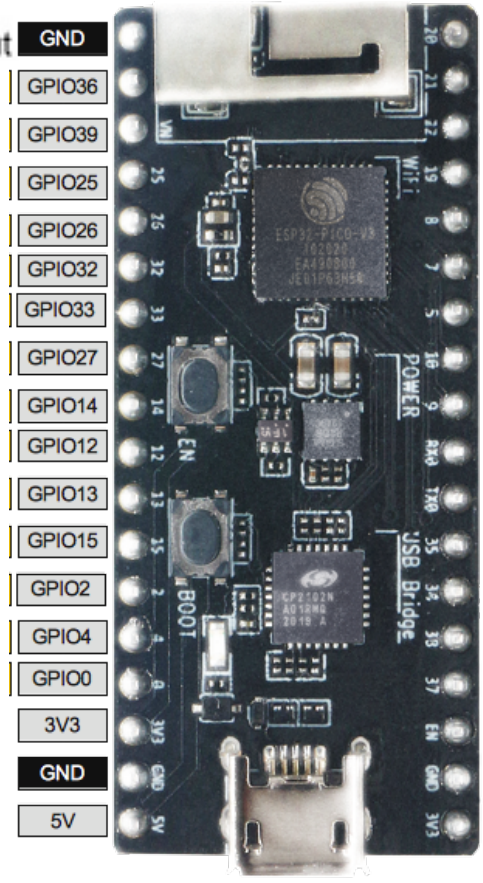
Power routing

Q7: How should you route the power wires to your devices?
(Keep in mind resistance of lines)



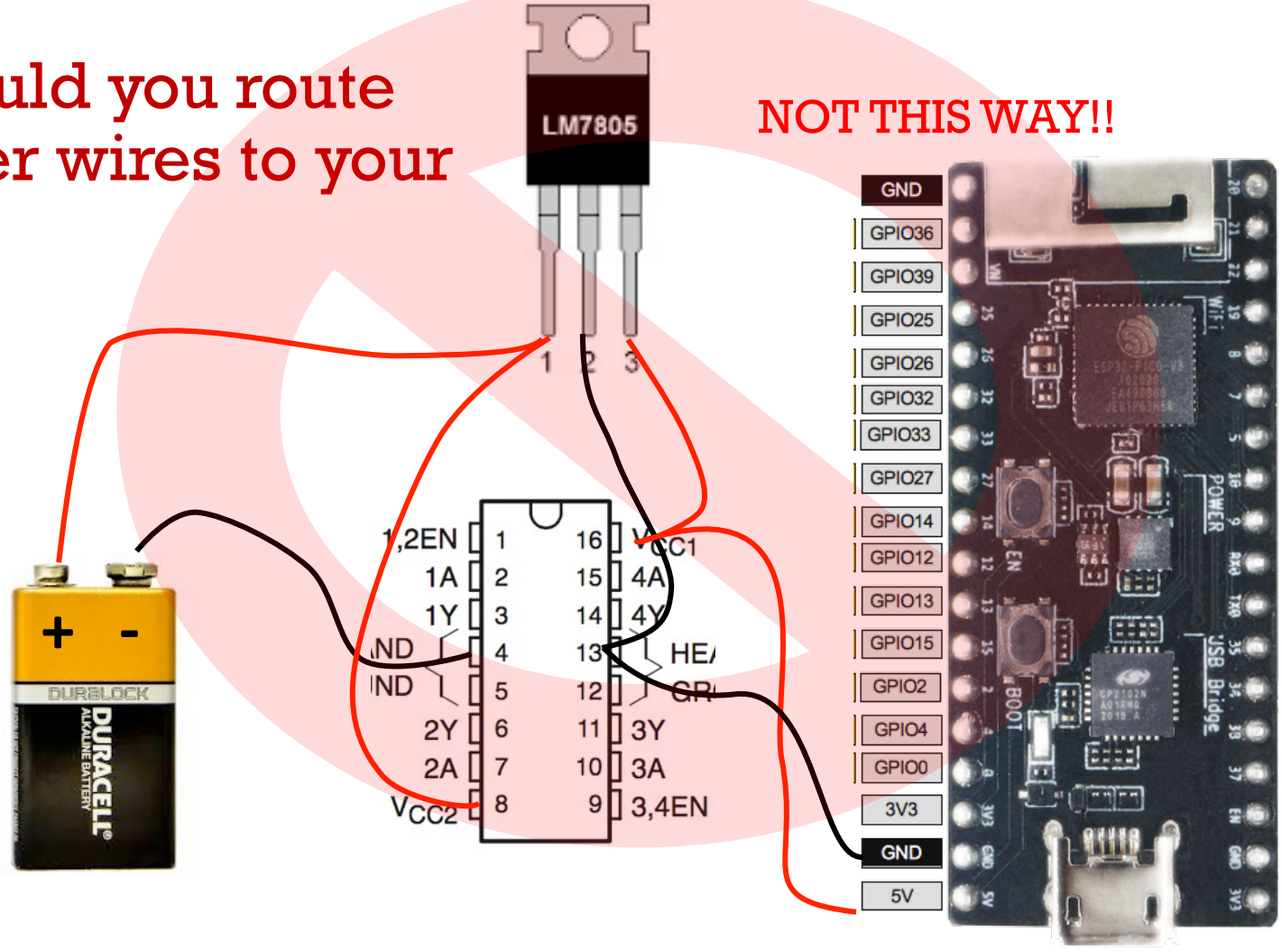




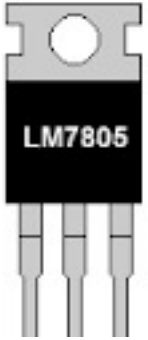


How should you route the power wires to your devices?

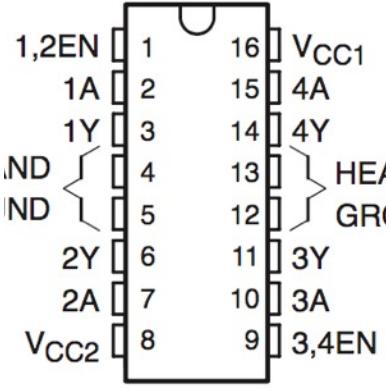
NOT THIS WAY!!

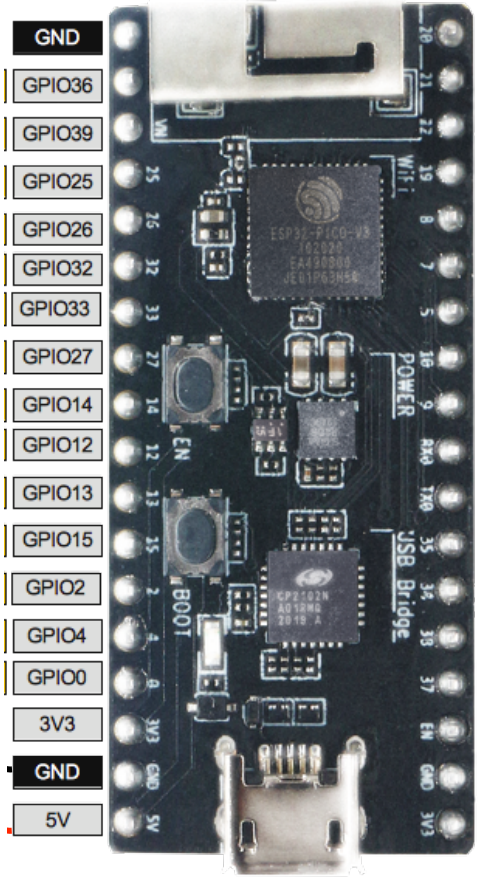


How should you route the power wires to your devices?



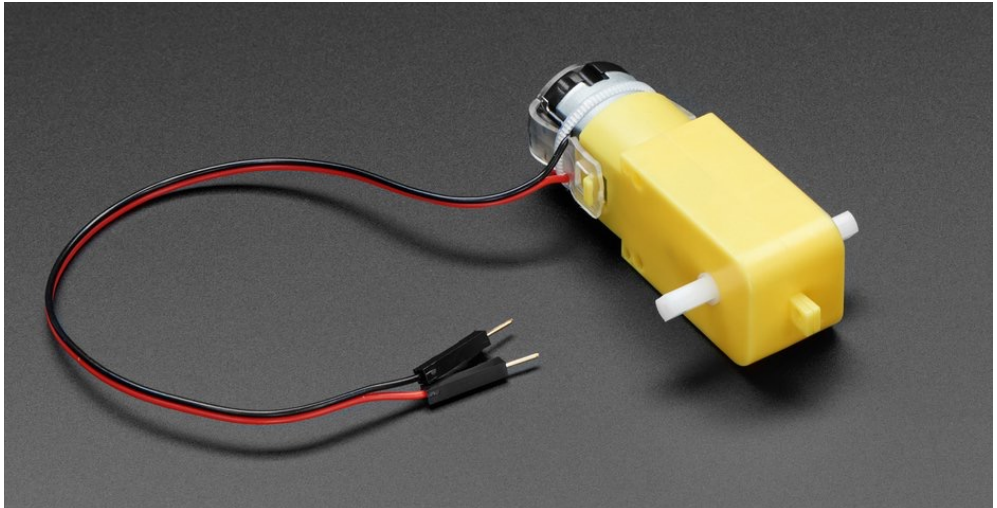
+V and gnd straight to battery, DON'T daisy chain.

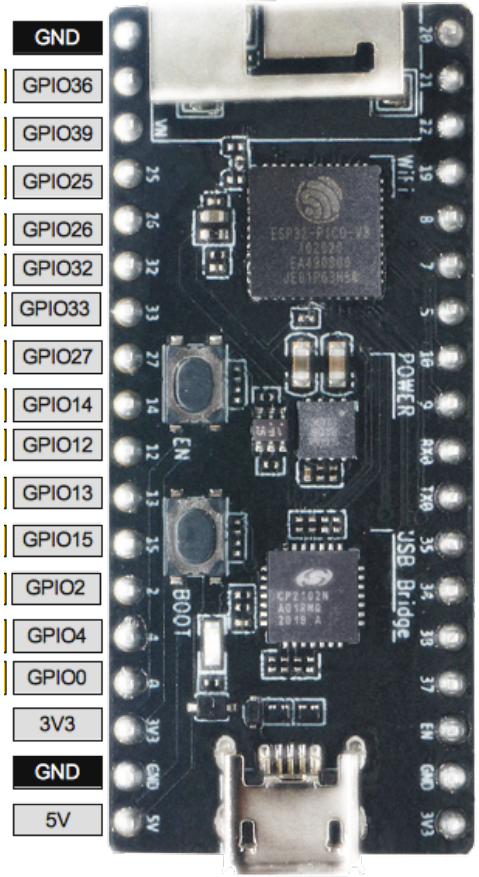




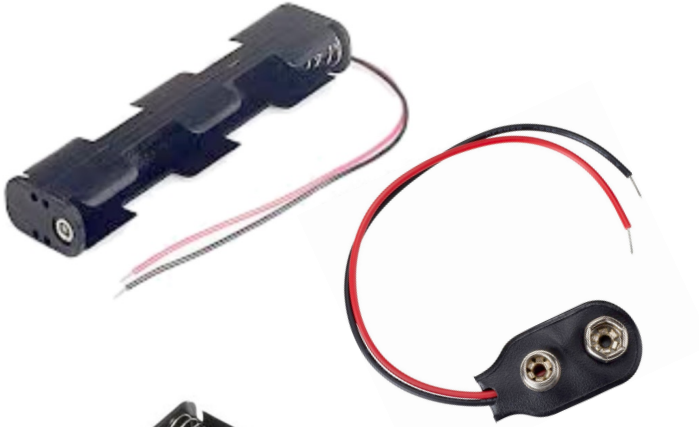
Noise issues in Lab 4.2

- It is likely that noise will be an issue as you drive motors. Sensors won't read correctly, your ESP32 may reset.
- One solution is to use one battery for motors one battery for logic devices.





Lab 4.2 Batteries



- You may use any battery you like.
- If you want to use standard batteries (e.g. AA or 9V)
- I do not recommend 9V batteries in general. (High cost, low energy)
- We have in stock
  - Battery connectors in GMlab (free)
  - USB charger 5V 5Ah, 2A $8
  - Variety of LiPos
    - 7.4V 1.3Ah, 40A $10 (limited supply),
    - 3.2V 3Ah, unknown $5

Summary

- Important to route power wires directly to power source – DO NOT DAISY CHAIN.
- You can supply upto 12V on PicoKit, Teensy can handle up to 5.5V.
- The PicoKit and NodeMCU power designs barely work. They won't work with long USB cables or with extra USB connections
- You can use both battery and USB with ESP32.
- You CANNOT use both battery to Vin and USB with Teensy

02

Building Website Components

Building Website Elements

Live Demo using W3Schools

https://www.w3schools.com/howto/howto_js_rangeslider.asp

- Changing text:
  - document.getElementById("label").innerHTML = "text";
- Adding buttons:
  - <button id="label"> text </button>
- Adding variables
  - Use parseInt() to force string to be number
- Formatting text
  - <H1>, <H2>, <P> for inline paragraph, for inline text

Run ›

Result Size: 275 x 562

Get your website

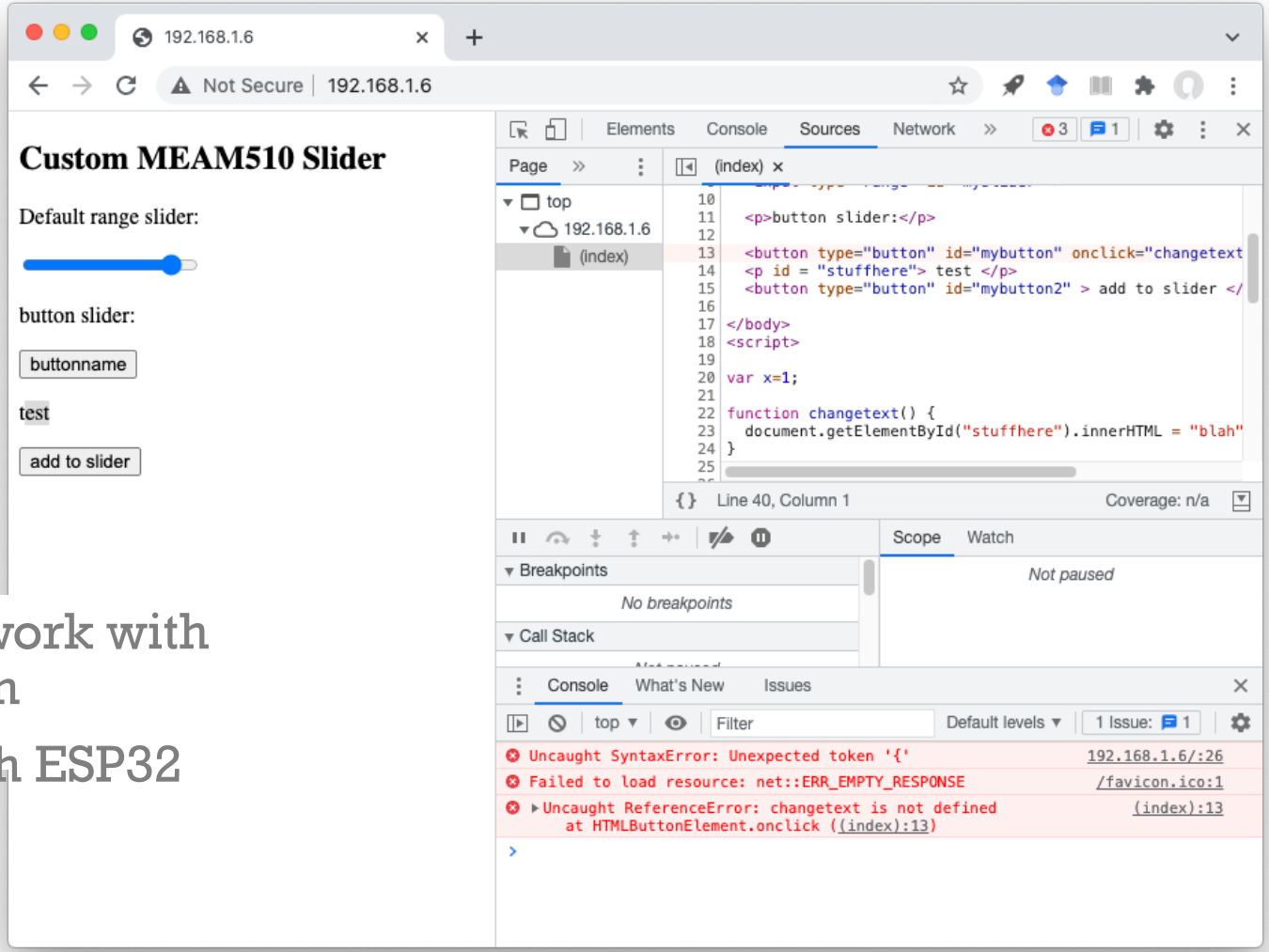
```
<!DOCTYPE html>
<html>
<body>

<h1>MEAM510 Slider</h1>

  <p>Default range slider:</p>
  <input type="range" id="slider"> <span id="moretext"> </span>
  <p id="sometext"> </p>
  <button id="mybutton"> button1 </button>
  <button id="mybutton2" onclick="addslider()"> 2 </button>

</body>
<script>
var x=1;
slider.onchange = function() {
  document.getElementById("sometext").innerHTML = this.value - x;
}

mybutton.onclick = function() {
  document.getElementById("moretext").innerHTML = "newtext";
}

function addslider() {
  x = x + 1;
}

</script>
</html>
```

MEAM510 Slider

Default range slider:

button1 2

Developer Tools can help debug



- Note, doesn't work with w3schools.com
- Does work with ESP32

03

Timers and interrupts on ESP32

All timer examples in this lecture are on canvas

- Canvss -> Files -> Resources -> ESP32 Arduino Lecture Examples
- Be sure to put into your own folder of the same name as Arduino requires.


int_TimerButtonpress.ino


int_TimerCapture.ino


int_TimerCompare.ino


interrupt-debounc-lecture.ino


interruptTimer.ino

Interrupts on ESP32

- **Internal**: peripherals like timer, or software generated interrupts
- **External**: digital pin (high/low level or edge triggered) interrupts
- Need to "attach" subroutine to interrupt e.g.:

```
timerAttachInterrupt(timerstruct, &timerISR, true);
attachInterrupt(#interruptpin, ISR, mode );
```

  - Note that sometimes you attach a pointer to a function (e.g. `&function`) and sometimes it is just the function name without **&**.

Timers on ESP

- Four hardware timers
  - 64bit wide (for ref, Teensy timers were 8, 10 and 16 bit wide)
    - Run at clock speed 80Mhz
    - More bits means less concern about rolling over, i.e., @80Mhz it takes 7311 years before a 64 bit counter reaches the limit 0xFFFFFFFF FFFFFFFF
  - 16bit pre-scalers

`timerBegin(timerno, prescale, countup);`

- *timerno* is which of the four hardware timers [0, 1, 2, 3]
- *prescale* is the number to divide the clock (e.g. 80Mhz / 80 -> uS timer)
- *countup* `true` means timer counts up instead of down

`timerAlarmWrite(timer, timeToAlarm, autorepeat);`

- *timer* structure returned by `timerBegin()`
- *timeToAlarm* number of counts until interrupt occurs
- *autorepeat* if `true` timer will repeat over and over

Timer Interrupt on ESP32 [ex: ISR once per second]

```
hw_timer_t* timer = NULL;
volatile uint32_t isrCounter = 0;

void IRAM_ATTR onTimer(){
// do something...  For example increment a counter
  isrCounter++;
}
void setup() {
  Serial.begin(115200);
  timer = timerBegin(0, 80, true); // Use timer 0 [0:3], Set prescaler = 80, count up
  timerAttachInterrupt(timer, &onTimer, true); // Attach onTimer() to our timer.

// Set alarm to call onTimer after 1 second (value in microseconds).
  timerAlarmWrite(timer, 1000000, true);       // once per sec, and autorepeat
  timerAlarmEnable(timer);                     // Start an alarm now
}
void loop() {
  static uint32_t oldCounter = 0;
  if (oldCounter != isrCounter) {  // Could do anything in loop()
    Serial.printf("%d\n", isrCounter);   // example; if Timer has fired
    oldCounter = isrCounter;
  }
}
```

Need to put subroutine in RAM for fast interrupts

On canvas: interruptTimer.ino

Output compare

timerAlarm example to generate PWM

```
hw_timer_t* timer = NULL;
uint32_t duty = 0;
uint32_t per = 10000; // period in uS

void IRAM_ATTR onTimer(){
  if (digitalRead(21)) {     // Use GPIO pin 21
    digitalWrite(21,LOW);
    timerAlarmWrite(timer, (100-duty)*per/100,true);   // Set low time
  }
  else  {
    digitalWrite(21, HIGH);
    timerAlarmWrite(timer, duty*per/100,true);       // Set high period
  }
}

void setup() {
  pinMode(21, OUTPUT);
  timer = timerBegin(0, 80, true); // Use timer 0 [0:3],  Set prescaler = 80.
  timerAttachInterrupt(timer, &onTimer, true); // Attach onTimer() to our timer.
// Set alarm to call onTimer after 1 second (value in microseconds).
  timerAlarmWrite(timer, duty*per/100, true);  // Repeat the alarm (third parameter)
  timerAlarmEnable(timer);                     // Start an alarm
}

void loop() {
  duty = 30;  // some code to set duty cycle
}
```

On canvas: int_TimerCompare.ino

Pin generated (external) interrupt

```
attachInterrupt(#interruptpin, ISR, mode );
```

- #interruptpin use digitalPinToInterrupt(GPIOPIN)
- ISR is name of interrupt service routine
- Mode -> [HIGH, LOW, RISING, FALLING, CHANGE]
- attachInterrupt is an Arduino routine available to all MCU
- As alterntate example timerAttachInterrupt() is ESP32 specific (won't appear in orange in Arduino IDE)

https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/

Input Capture example

```
uint32_t buttonPressTime;

void IRAM_ATTR handleButtonInterrupt() {
   buttonPressTime = millis();  // millis returns ms since prog started.
}

void setup(){
  Serial.begin(115200);
  pinMode(0, INPUT);              // GPIO 0 has button on PicoKit board
  attachInterrupt(digitalPinToInterrupt(0), handleButtonInterrupt, FALLING);
}

void loop() {
  static int oldbuttonPressTime;
  if (oldbuttonPressTime != buttonPressTime) {
    Serial.printf("Button pressed at %d\n",buttonPressTime);
    oldbuttonPressTime = buttonPressTime;
  }
}
```

On canvas: int_TimerCapture.ino

Input Capture example (down time)

```
uint32_t buttonDownTime, buttonUpTime;

void IRAM_ATTR handleButtonInterrupt() {
   if (digitalRead(0)) buttonUpTime = millis();
   else buttonDownTime = millis();
}
void setup(){
  Serial.begin(115200);
  pinMode(0, INPUT);              // GPIO 0 has button on PicoKit board
  attachInterrupt(digitalPinToInterrupt(0), handleButtonInterrupt, CHANGE);
}

void loop() {
  static int oldbuttonPressTime;
  if (oldbuttonUpTime != buttonUpTime) {
    Serial.printf("Button pressed at %d\n",buttonUpTime-buttonDownTime);
    oldbuttonUpTime = buttonUpTime;
  }
}
```

On canvas: int_TimerButtonpress.ino

Interrupts on ESP32
– multicore- multithread complications

- Memory (IRAM -> Instruction RAM, DRAM -> Data RAM)
  - Faster access otherwise uses ROM (e.g. FLASH) potential complications with cache. IRAM_ATTR, Important for ESP32 ISR.

```
void IRAM_ATTR onTimer()
```

- Semaphores, Mutex (mux) and CRITICAL_ISR
  - Multi-threaded, multi-core, RTOS can potentially screw things up if multiple threads access same resource.
  - These devices help to manage.

```
volatile SemaphoreHandle_t timerSemaphore;

portMUX_TYPE timerMux = portMUX_INITIALIZER_UNLOCKED;
portENTER_CRITICAL_ISR(&timerMux);
  isrCounter++;
portEXIT_CRITICAL_ISR(&timerMux);
```

https://docs.espressif.com/projects/esp-idf/en/latest/api-guides/general-notes.html
https://techtutorialsx.com/2017/10/07/esp32-arduino-timer-interrupts/

Pin generated (external) interrupt example [part 1/2]

```
/* Debouncing a button interrupt example
 */
#define BUTTONPIN    0
#define DEBOUNCETIME 20 // in ms
volatile int numberOfButtonInterrupts = 0;
volatile bool lastState;
volatile uint32_t debounceTimeout = 0;
portMUX_TYPE mux = portMUX_INITIALIZER_UNLOCKED;

// Interrupt Service Routine - Keep it short!
void IRAM_ATTR handleButtonInterrupt() {
  portENTER_CRITICAL_ISR(&mux);
   numberOfButtonInterrupts++;
   lastState = digitalRead(BUTTONPIN);
   debounceTimeout = millis();
  portEXIT_CRITICAL_ISR(&mux);
}

void setup(){
  Serial.begin(115200);
  pinMode(BUTTONPIN, INPUT);  // Pull up to 3.3V on input - GPIO 0 has external pullup
  attachInterrupt(digitalPinToInterrupt(BUTTONPIN), handleButtonInterrupt, FALLING);
}
```

On canvas: interrupt-debounc-lecture.ino

Pin generated (external) interrupt example [part 2/2]

```
void loop() {
  bool currentState = digitalRead(BUTTONPIN);

  if ((numberOfButtonInterrupts != 0)  // interrupt has triggered
    && (currentState == lastState)     // pin state same as last interrupt
    && (millis() - debounceTimeout > DEBOUNCETIME ))
  { //  low for at least DEBOUNCETIME,
    if (currentState == LOW)  Serial.printf("Button is pressed\n");
    else   Serial.printf("Button is RELEASED\n");
    Serial.printf("Triggered %d times\n", numberOfButtonInterrupts);
    portENTER_CRITICAL_ISR(&mux);  // can't change it unless, atomic - Critical section
     numberOfButtonInterrupts = 0; // acknowledge keypress and reset interrupt counter
    portEXIT_CRITICAL_ISR(&mux);
    delay(1);
  }
  delay(1);
}
```

On canvas: interrupt-debounc-lecture.ino

Summary

- Timer functions can be run from interrupts
- Since they are usually regular, easier to debug than random intterupts
- millis() returns milliseconds since program started
- Have to be careful about writing to same variable from multiple places (since ESP32 is multi core)

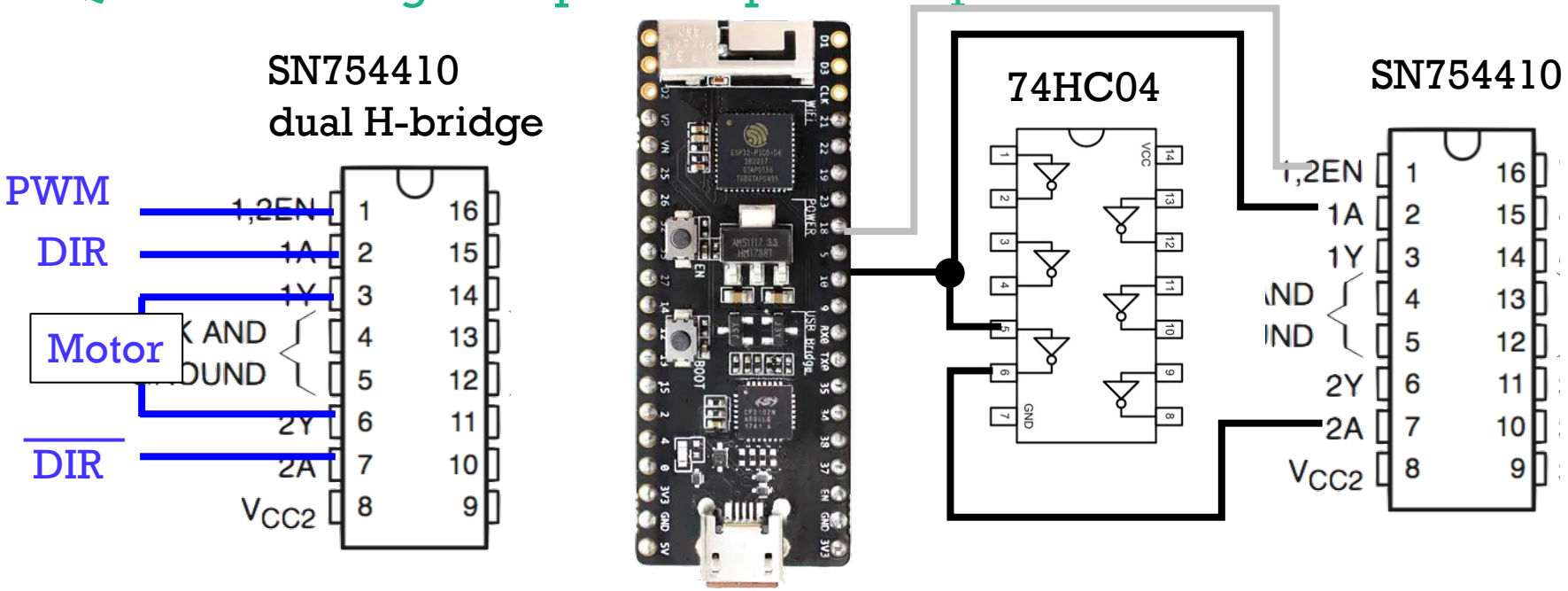
00

Datasheet Practice
C++ integer types

Datasheet Practice: ESP32/HC04/SN754410

- To save ESP32 GPIO pins, we want to use an inverter for the 2A input (always inverse of 1A input)

Q2: Are the digital inputs/outputs compatible?



Q1: Circle the important specs

Datasheet Practice:

6.2 Recommended Operating Conditions

over operating free-air temperature range (unless otherwise noted)

			MIN	NOM	MAX	UNIT
V_{CC}	Supply voltage		4.5		5.5	V
V_{IH}	High-level input voltage	V_{CC} = 4.5 V to 5.5 V	2			V
V_{IL}	Low-level input voltage	V_{CC} = 4.5 V to 5.5 V			0.8	V
V_I	Input voltage		0		V_{CC}	V
V_O	Output voltage		0		V_{CC}	V
t_t	Input transition time	V_{CC} = 4.5 V			500	ns
		V_{CC} = 5.5 V			400	
T_A	Operating free-air temperature		–55		125	°C

74HC04 Datasheet

6.4 Electrical Characteristics

over operating free-air temperature range; typical values measured at T_A = 25°C (unless otherwi

PARAMETER		TEST CONDITIONS		V_{CC}	Operating free-air temperatur					
					25°C			–40°C to 85°C		
					MIN	TYP	MAX	MIN	TYP	MAX
V_{OH}	High-level output voltage	V_I = V_{IH} or V_{IL}	I_{OH} = –20 µA	4.5 V	4.4			4.4		
			I_{OH} = –4 mA	4.5 V	3.98			3.84		
V_{OL}	Low-level output voltage	V_I = V_{IH} or V_{IL}	I_{OL} = 20 µA	4.5 V			0.1			0.1
			I_{OL} = 4 mA	4.5 V			0.26			0.33

.3 DC Characteristics (3.3 V, 25 °C)

ESP32 Datasheet

Table 13: DC Characteristics (3.3 V, 25 °C)

Symbol	Parameter		Min	Typ	Max	Unit
C_{IN}	Pin capacitance		-	2	-	pF
V_{IH}	High-level input voltage		0.75×VDD[1]	-	VDD[1]+0.3	V
V_{IL}	Low-level input voltage		–0.3	-	0.25×VDD[1]	V
I_{IH}	High-level input current		-	-	50	nA
I_{IL}	Low-level input current		-	-	50	nA
V_{OH}	High-level output voltage		0.8×VDD[1]	-	-	V
V_{OL}	Low-level output voltage		-	-	0.1×VDD[1]	V
I_{OH}	High-level source current (VDD[1] = 3.3 V, V_{OH} >= 2.64 V, output drive strength set to the maximum)	VDD3P3_CPU power domain [1, 2]	-	40	-	mA
		VDD3P3_RTC power domain [1, 2]	-	40	-	mA
		VDD_SDIO power domain [1, 3]	-	20	-	mA

SN754410 Datasheet

7.2 Recommended Operating Conditions

over operating free-air temperature range (unless otherwise noted)

		MIN	MAX	UNIT
V_{CC1}	Logic supply voltage	4.5	5.5	V
V_{CC2}	Output supply voltage	4.5	36	V
V_{IH}	High-level input voltage	2	5.5	V
V_{IL}	Low-level input voltage	–0.3 (1)	0.8	V
T_J	Operating virtual junction temperature	–40	125	°C
T_A	Operating free-air temperature	–40	85	°C

Standard integer types

`int`	**on Teensy is 2 bytes (−32,768 to 32,767)**
`int`	**on ESP32 is 4 bytes (−2,147,483,648 to 2,147,483,647)**
`uint8_t`	1 byte unsigned, (i.e. unsigned char)
`uint16_t`	2 byte unsigned, (i.e. unsigned short int)
`uint32_t`	4 byte unsigned, (i.e. unsigned long int)
`uint64_t`	8 byte unsigned, (i.e. unsigned long long int)
`int8_t`	1 byte signed, (i.e. signed char)
`int16_t`	2 byte signed, (i.e. signed short int)
`int32_t`	4 byte signed, (i.e. signed long int)
`int64_t`	8 byte signed, (i.e. signed long long int)

Variable type conversion

- **Automatic promotion:** If two operands are different, the smaller is promoted to the larger. If operands are the same size but unsigned and signed, then signed is cast as unsigned. Float is bigger than int.

```
int si = -1;
unsigned int ui = 1;
if (    si <(int)ui) printf("true"); // Q3 Does it print true?
```

Mixing signed and unsigned may have unexpected results

```
float x;
short int y = 4;
x =   y/3.0;    // y is promoted to a float, x = 1.33333
x =         y/3;// Q4 What does x equal?
```

Promotion happens at evaluation of operation

https://wiki.sei.cmu.edu/confluence/display/c/INT02-C.+Understand+integer+conversion+rules

Variable type conversion -cont.

- Bitwise operations and promotions to int.

```
uint8_t port = 0x12;   // for reference 0x12 = b00010010
uint8_t result_8 = (uint8_t) (~port) >> 4; // Q5: What is result_8?
```

- Passed parameters are promoted according to prototype

```
hw_timer_t *timer;
float value = 12.0;
timerAlarmWrite(timer,  value, true);
```

Supposed to be an unsigned int 64 bit

- Use google to find code prototypes "esp32 timerAlarm"
- Prototype in esp32-hal-timer.h

```
timerAlarmWrite(hw_timer_t *timer, uint64_t value, bool reload);
```

https://wiki.sei.cmu.edu/confluence/display/c/INT02-C.+Understand+integer+conversion+rules
https://github.com/espressif/arduino-esp32/blob/master/cores/esp32/esp32-hal-timer.h

Random Useful Websites:

Basic Circuits

http://www.seattlerobotics.org/encoder/mar97/basics.html

Sensors

http://www.seattlerobotics.org/encoder/jul97/basics.html

Filters

http://mysite.du.edu/~etuttle/electron/elect15.htm

Batteries (more detail)

http://www.mpoweruk.com/index.htm

http://batteryuniversity.com/

Answer in Chat

Answer how you feel about each topic below with:

1. I don't understand this topic at all
2. I don't know now, but know what to do to get by
3. I understand some, but expect to get the rest later
4. I understand completely already

A. Power Routing

B. Building Websites

C. Timers and interrupts on ESP32