Lecture 18

Interrupts on ESP32
Power to circuits

Agenda

00 Demo and C++ Integer Type and Conversions

01 Timers and Interrupts on ESP32

02 Voltage Regulators

03 Battery Intro (more battery next time)

04 Level shifting

00

Lab 4 Demo and C++ integer types

Driving Robots in GM Lab

After you have your root working at home…

1. Program your ESP32 with new IP Address:
   192.168.43.*YOUR_ID*
2. Program a new SSID and Password:

```
const char* ssid     = "#Skyroam_1t9";
const char* password = "55687127";
```

3. Practice with TA during office hours
   1. Drop off your robot at GM
   2. Go home (or sit outside Towne and connect to internet)
   3. Ask TA to turn on and send you the ngrok URL to put in your browser.

Demo of Lab 4 from GM lab

- Robot installed (with Lab4 demo code from last lecture – including SSID, Passwd and IP address 192.168.143.9)
- TA is running webcam and zoom so we can watch.
- TA has sent ngrok url to me
- Good idea for you to try this when you can even if you don't have your robot done, so you can see the process and experience the lag.

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 one will be promoted to the larger. If operands are the same size but one is unsigned and signed, then signed is cast as unsigned.

```
int si = -1;
unsigned int ui = 1;
if (    si <(int)ui) printf("true"); // Q1 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;      // Q2: 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; // Q3: 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

01

PWM, Timers and interrupts on ESP32

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, e.g. 7311 years
  - 16bit pre-scalers

`timerBegin(timerno, prescale, countup);`

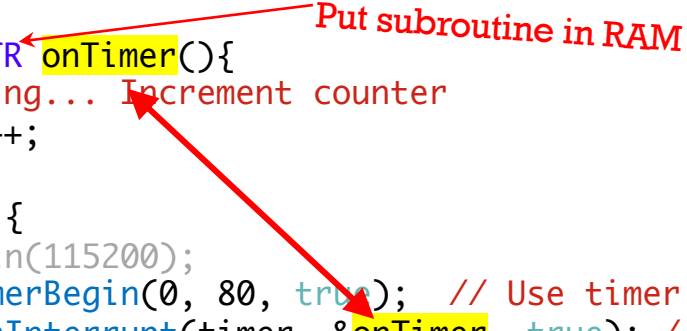
- *timerno* is which of four hardware timers [0, 1, 2, 3]
- *prescale* is the number you want to divide the clock (e.g. 80Mhz / 80 -> microsecond timer)
- *countup* = true means timer counts up instead of down

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... Increment counter
  isrCounter++;
}
void setup() {
  Serial.begin(115200);
  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, 1000000, true);       // Repeat the alarm (third parameter)
  timerAlarmEnable(timer);                     // Start an alarm
}

void loop() {
  static uint32_t oldCounter = 0;
  if (oldCounter != isrCounter) { // If Timer has fired
    Serial.printf("%d\n", isrCounter);
    oldCounter = isrCounter;
  }
}
```



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/

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 = xTaskGetTickCount();    //faster version of 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);
}
```

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

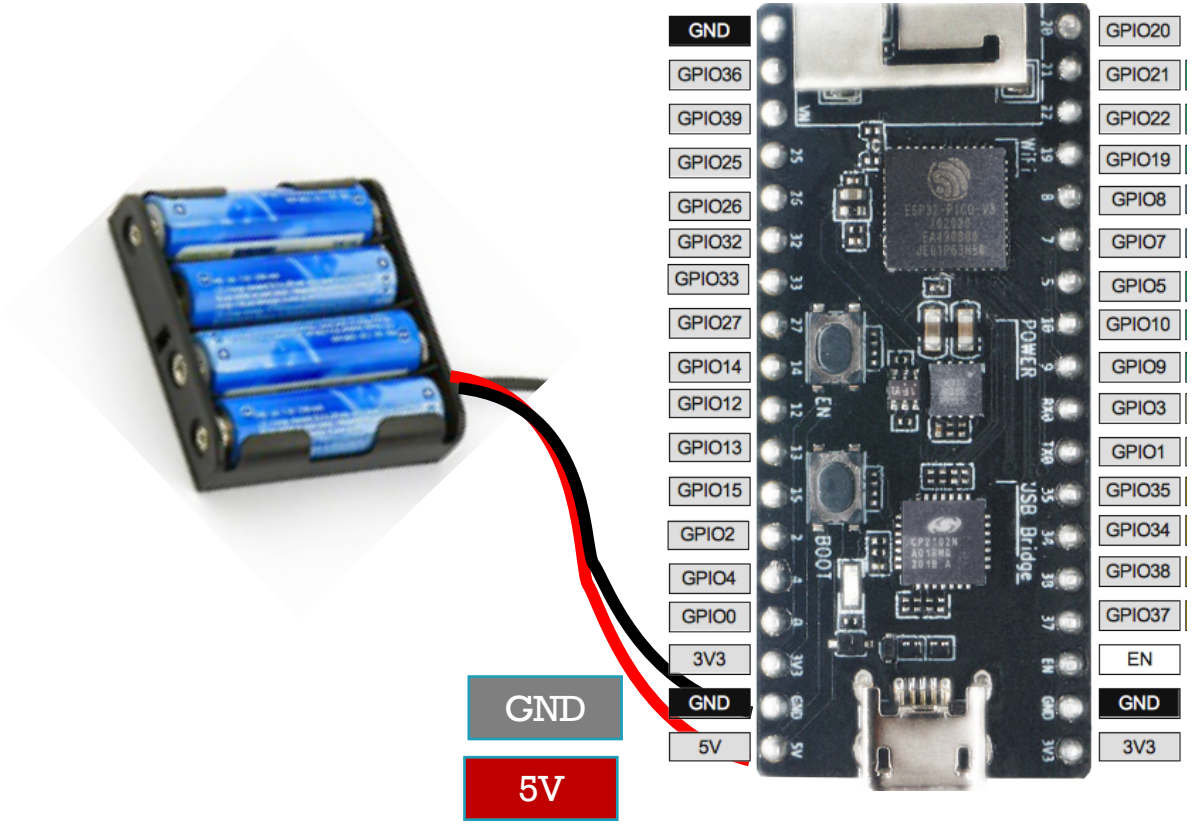
```
void loop() {
  bool currentState = digitalRead(BUTTONPIN);

  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);
}
```

02

Voltage Regulators

Can we safely attach 6V or 9V to the 5V pin?

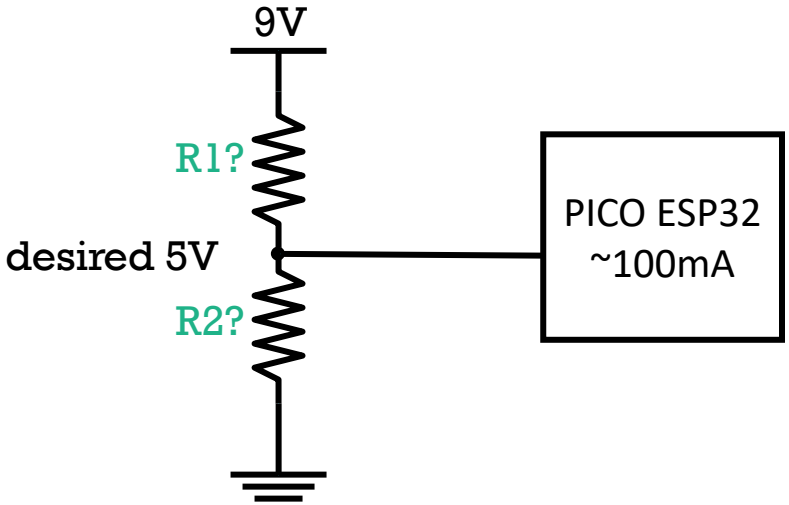


Voltage Regulators

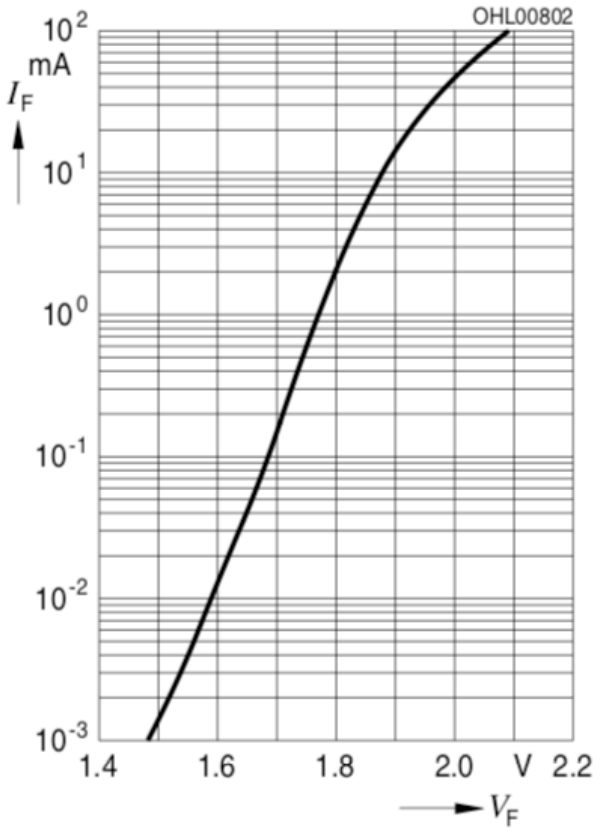
- You've got 9V battery and you need 5V to go to Vin.
- Can you use a voltage divider to supply 5
- What resistors do you choose?

Q4: Which will result closest to 5v?

A. R1 = 4Ω , R2 =5Ω
B. R1 = 40Ω , R2 = 50Ω
C. R1 = 4kΩ , R2 = 5kΩ
D. they are all the same



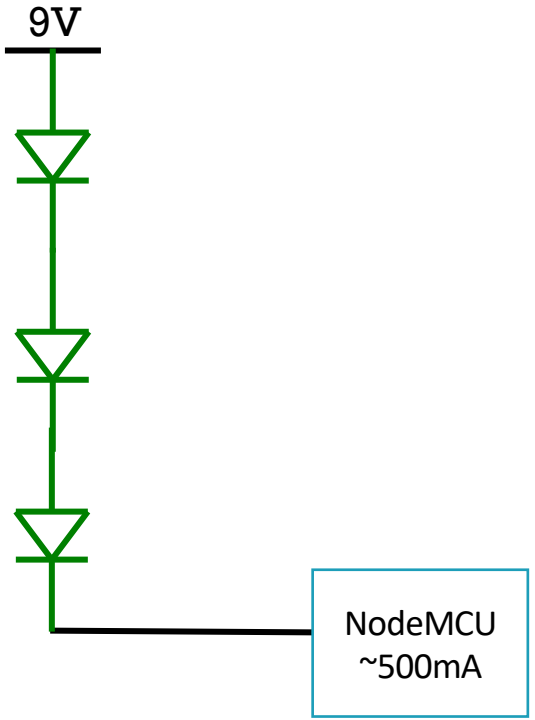
Voltage Regulators



ry and you need 5V to run the NodeMCU.

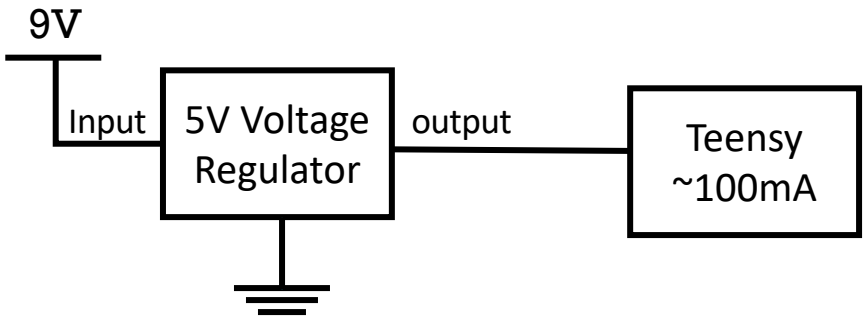
le diode drops?

Q5: What issues do we have with using 3 LEDs with Vf = 1.7V?



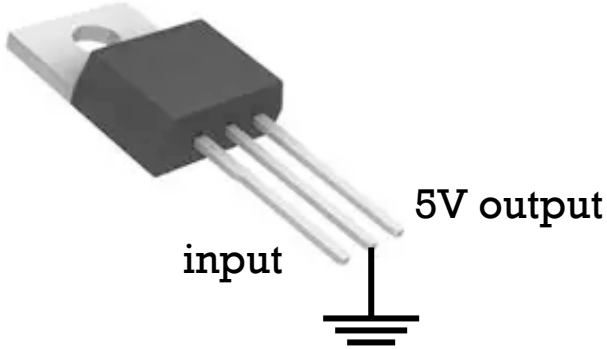
Voltage Regulators

- You've got 9V battery and you need 5V to run a Teensy.
- Use a voltage regulator
- They will get hot
  - if you have large input voltage
  - If you have large current.
- They have current limits
- Do NOT put them in parallel
  - They may fight each other and go unstable
  - If you need more current, find a different regulator.

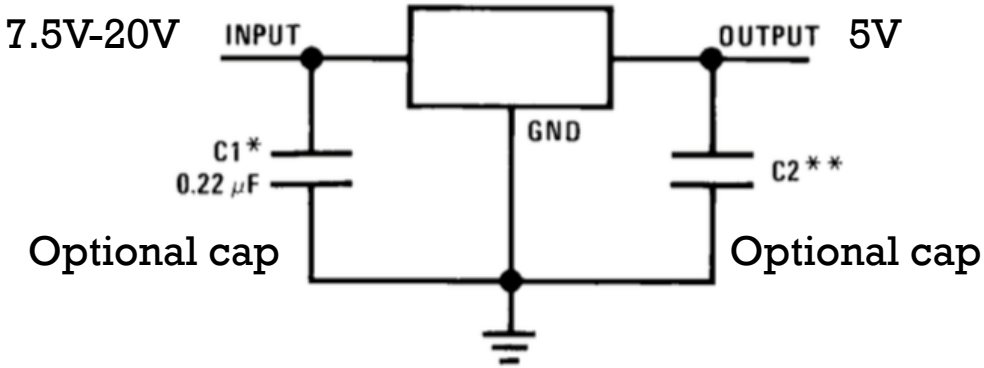


LM7805 5V regulator

Standard (old), low cost, low performance

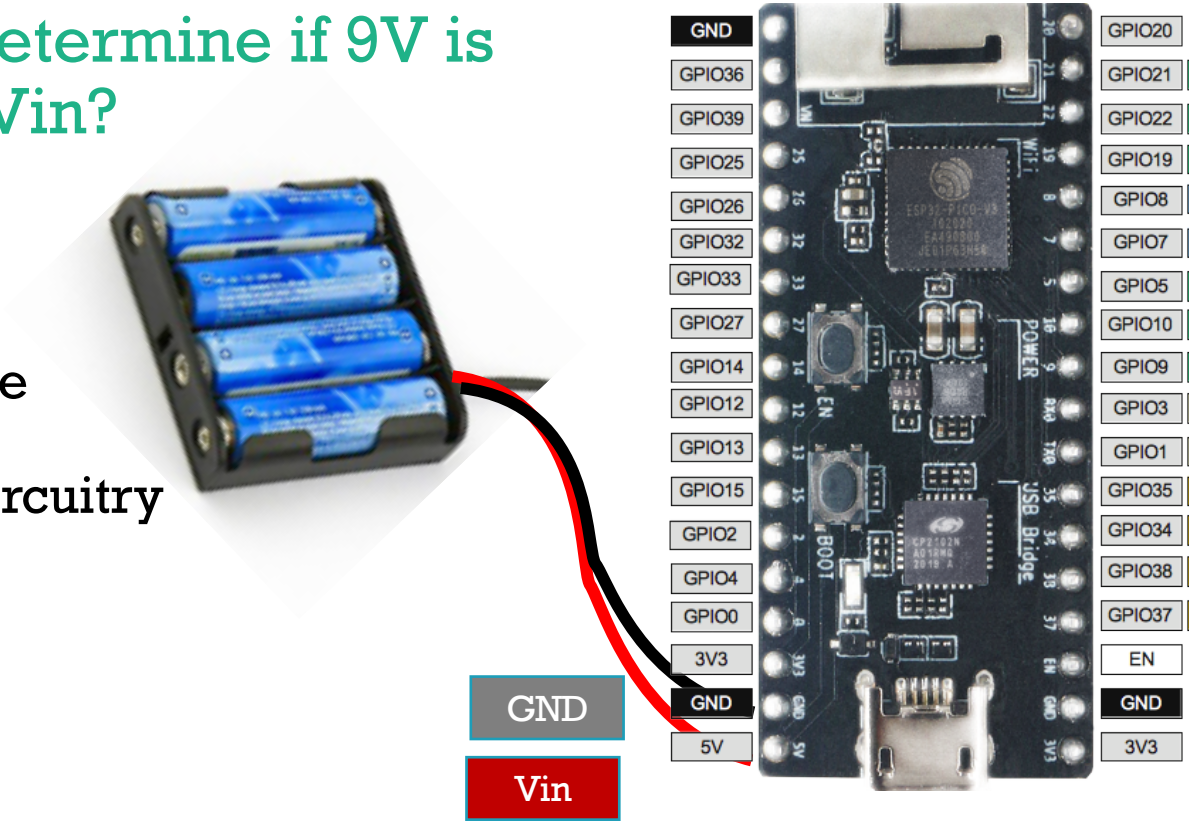


- Max output: 1.5A
- Input range: 7.5V to 20V
- Line regulation: 50mV
- Load regulation: 50mV
- Output usually: 5.0V
  but can vary 4.8 to 5.2V



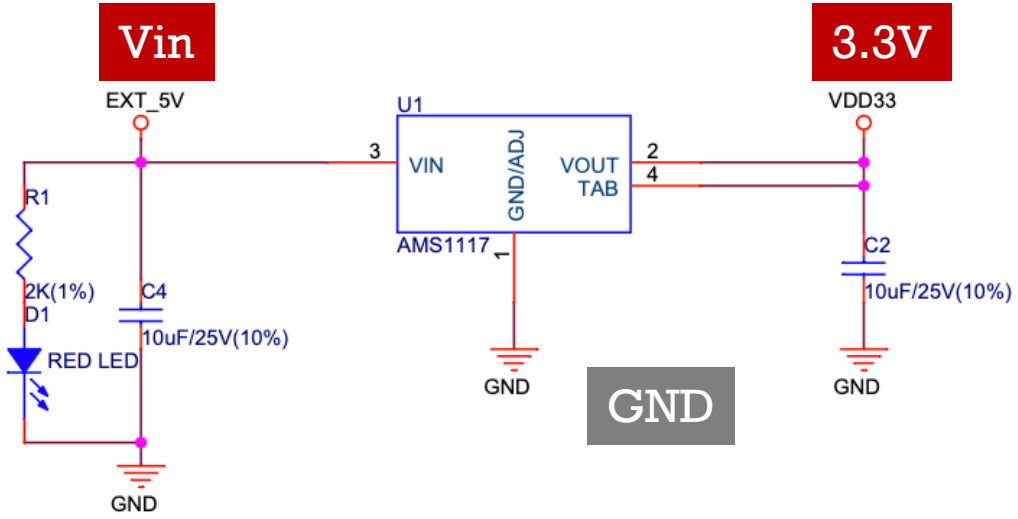
Q6: What component on the NodeMCU board do we need to know about to determine if 9V is safe to apply at Vin?

A) ESP32 input voltage range
B) Protection diode circuitry
C) Voltage regulator
D) USB Specifications



NodeMCU Voltage Regulator: AMS1117

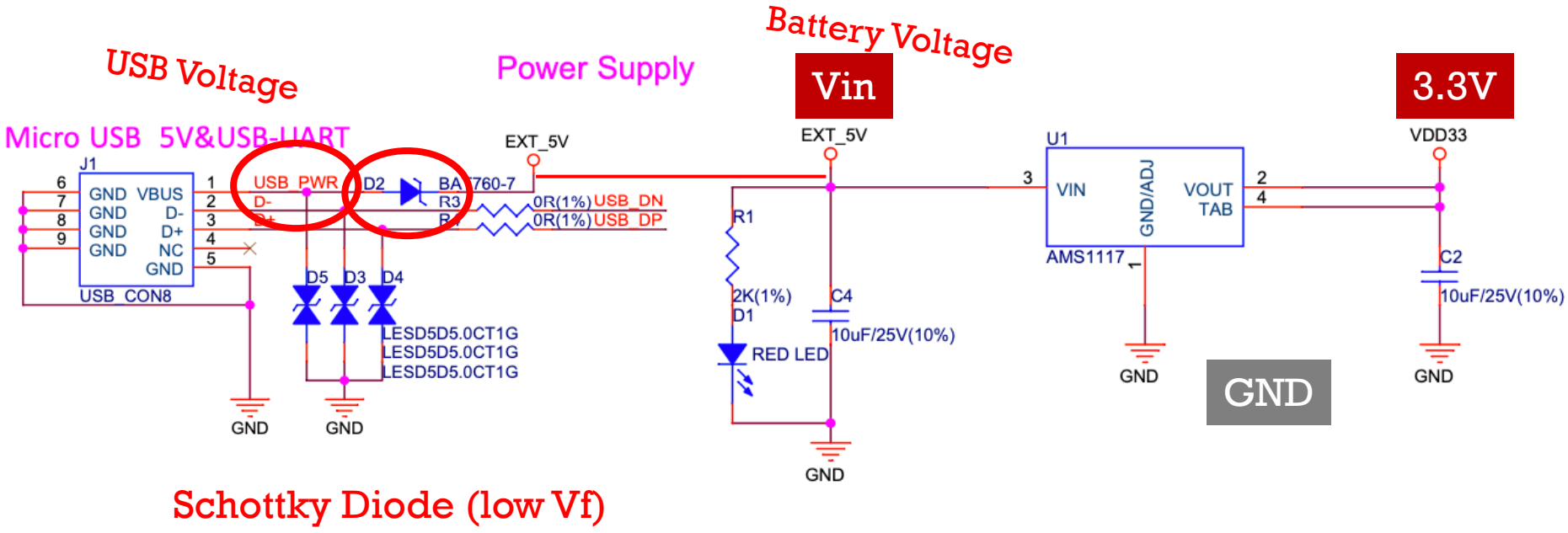
Power Supply



- AMS1117-3.3 outputs 3.3V
- Low Drop Out (LDO) meaning it can take closer to 3.3V as input
  - LDO is 1.1V@0mA to 1.3V@800mA
- Looking at datasheet: AMS117 works with input:

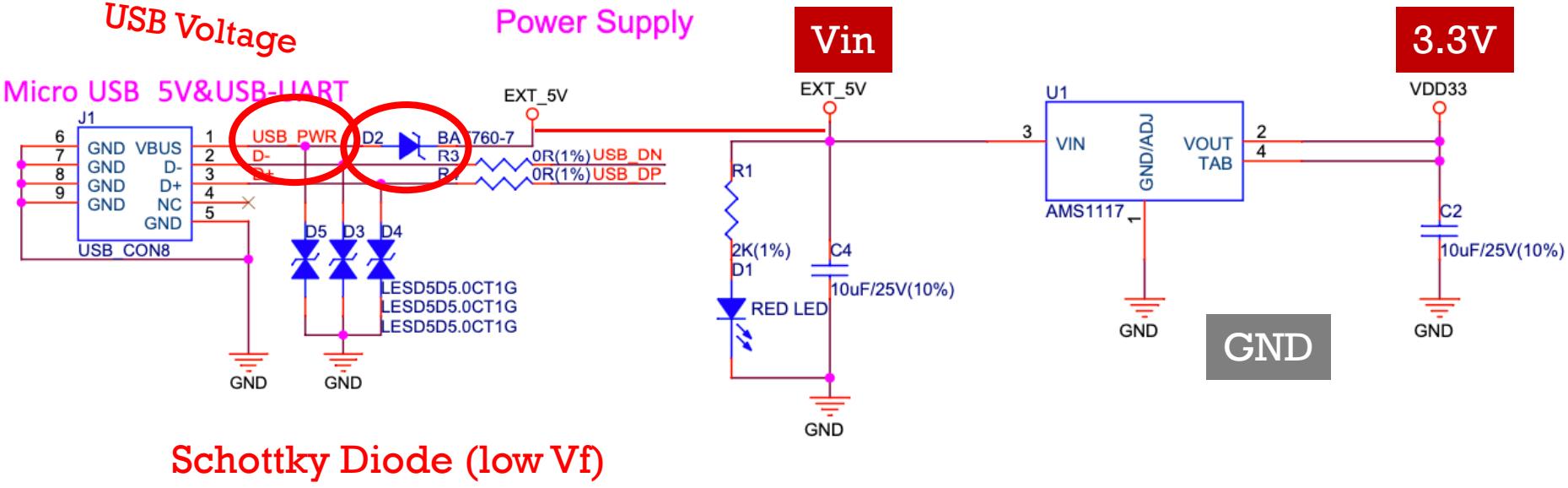
  4.6V to 12V

Q7: Is it okay to use both USB and battery at Vin?



Schottky Diode (low Vf)

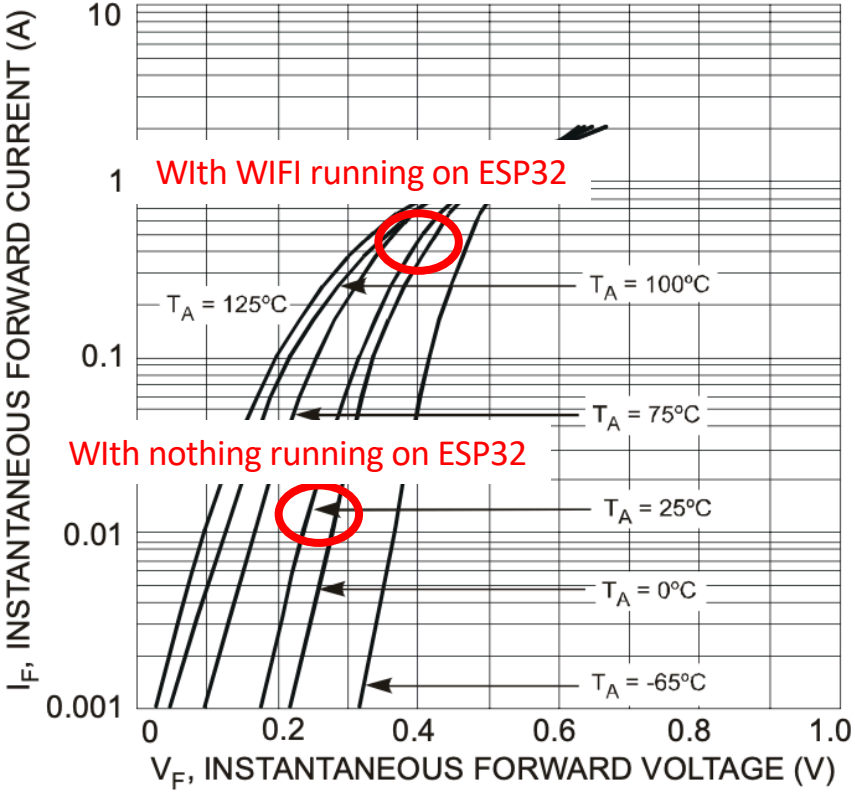
Q8 With USB power only, what will the voltage be at Vin?



Schottky Diode (low Vf)

Schottky Diodes (PICO uses BAT760-7)

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



VF about 0.25V without WiFi
VF about 0.4V with WiFi

- Power from USB is nominally 5V
- Power at Vin = 5V – Vf
- Power required at AMS117 > 4.6 worst case, though typically works at 4.4!=V

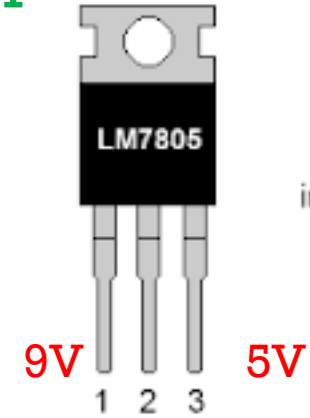
Solutions to borderline power issue:

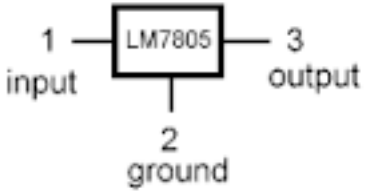
- Simple thing, add external 5V to 12V supply to Vin line on PICO (e.g. external battery) or power supply.
- Replace BAT760-7 with lower Vf (I found some about 0.1V lower)
- Replace AMS1117 with lower LDO, but major hassle to see if it fits right form factor of surface mount component
- Bypass the Schottkey diode. Dangerous, but okay if you remember never to plug in USB when it is battery powered. - WE DONT RECOMMEND THIS.
  - Or bypass Schottkey and never run battery powered Also not recommended.

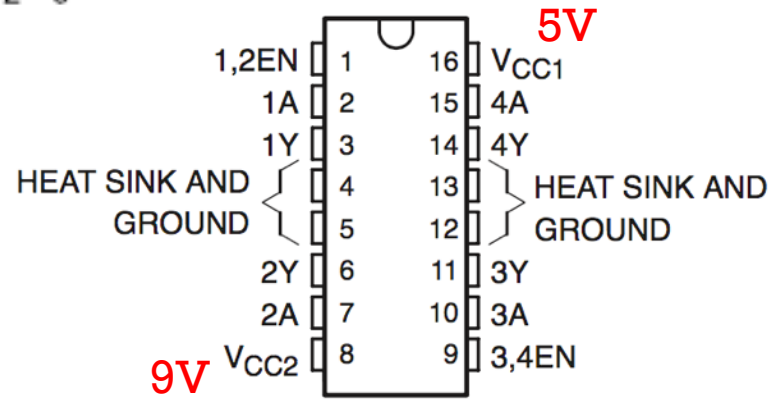
03

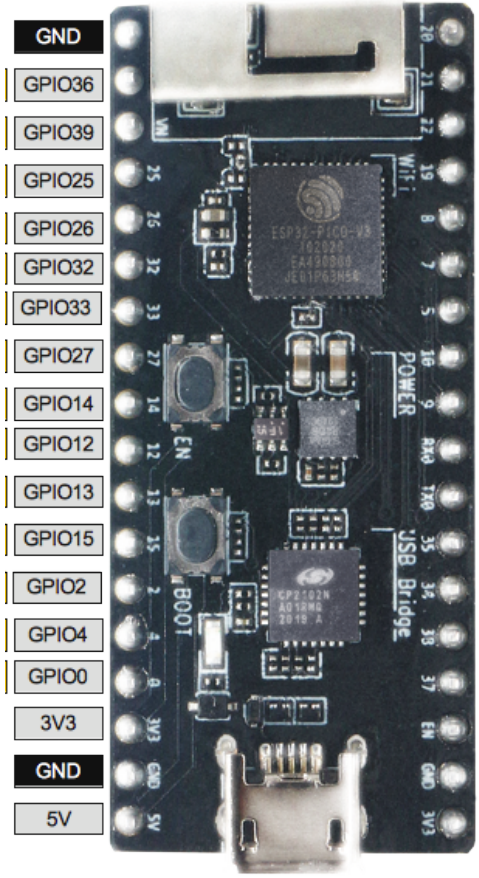
Battery Intro/Power routing

Q9: Power wiring: How should you route the power wires to your devices? Draw and hold one line connecting pins that have 5V

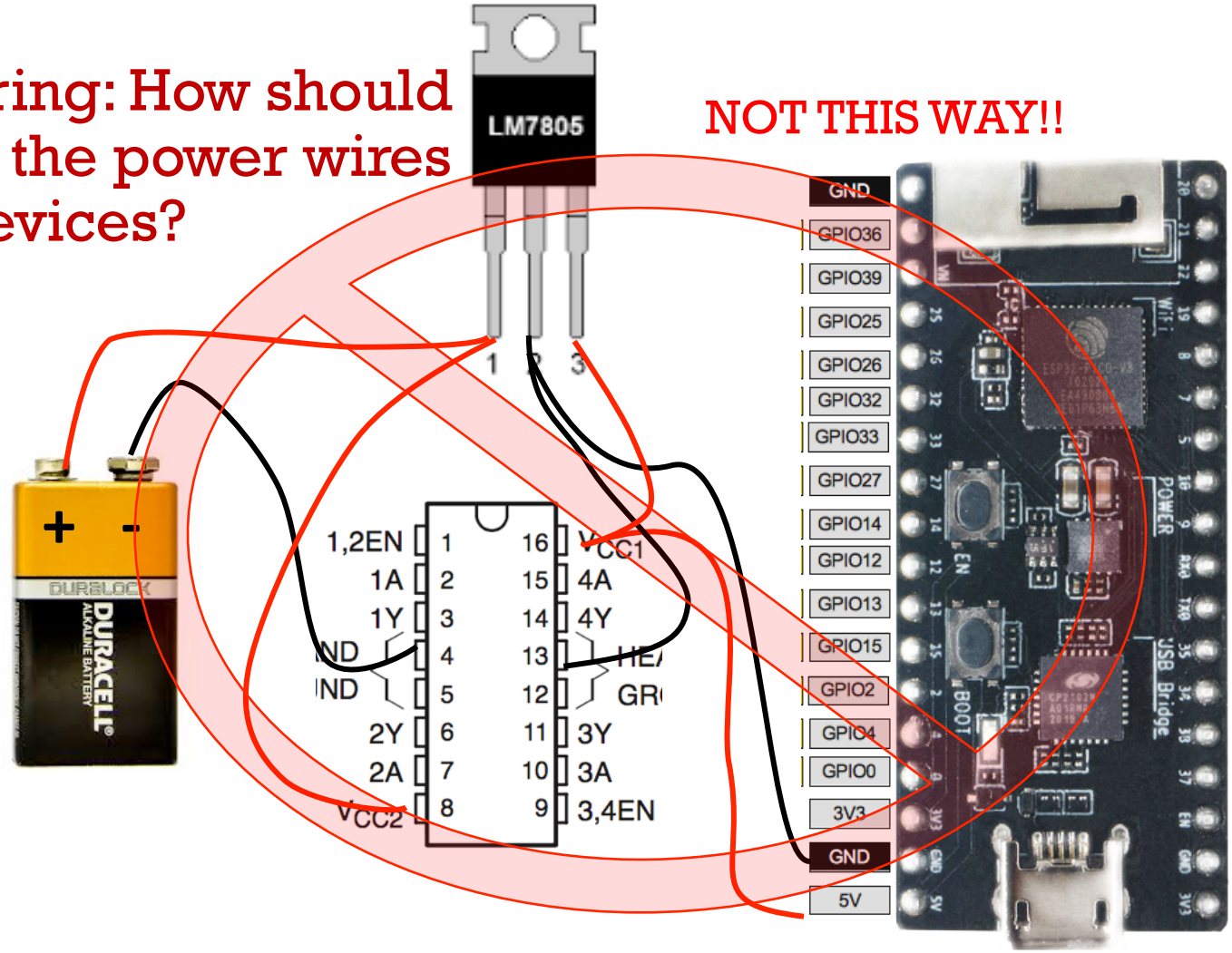




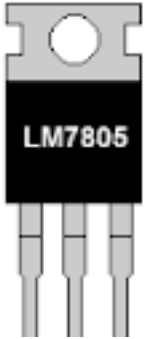




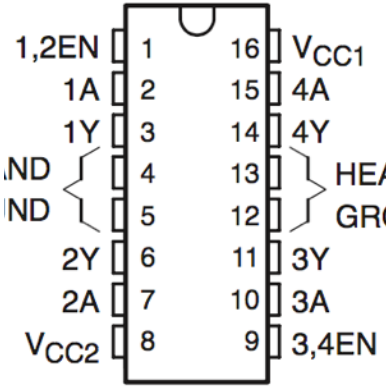
Power wiring: How should you route the power wires to your devices?

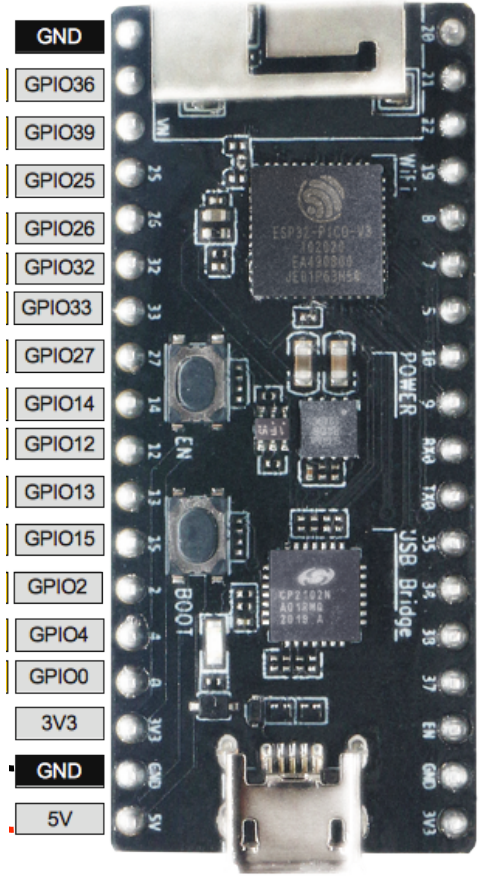


Power wiring: How should you route the power wires to your devices?



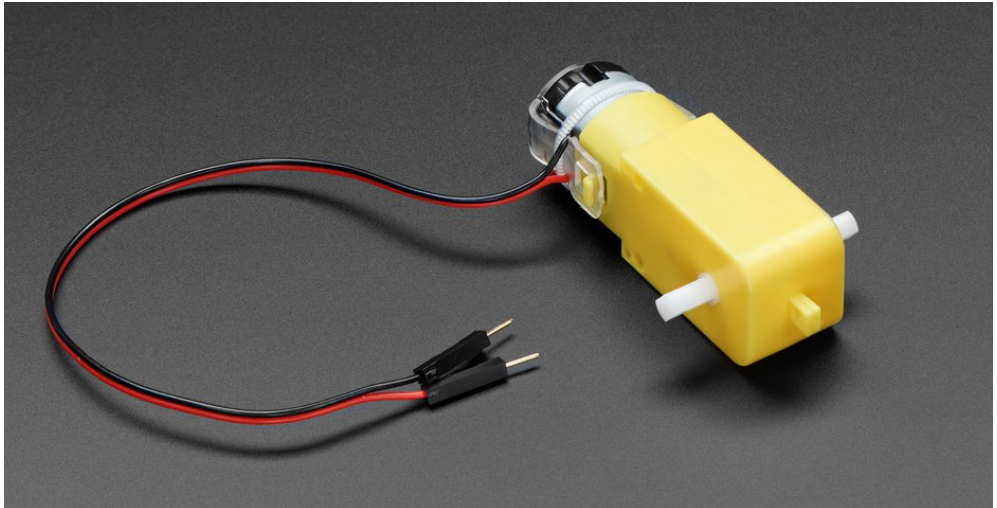
+V and gnd straight to battery, DON'T daisy chain.

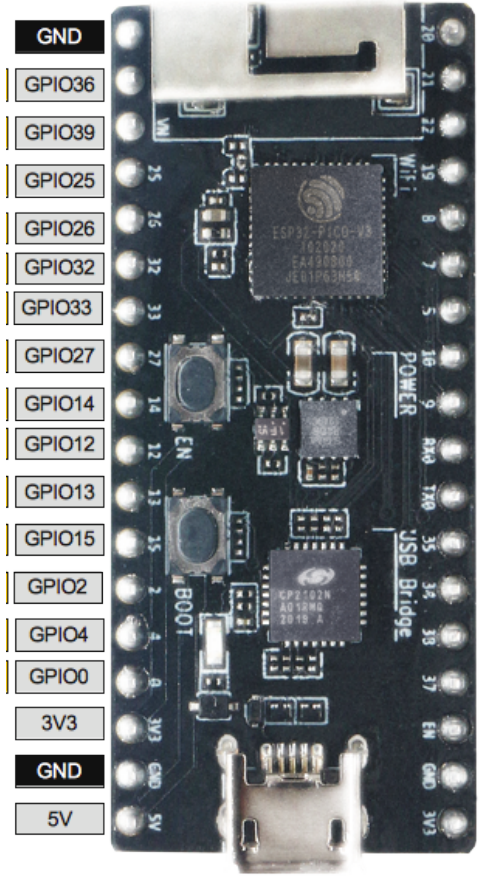




Noise issues in Lab 4.2

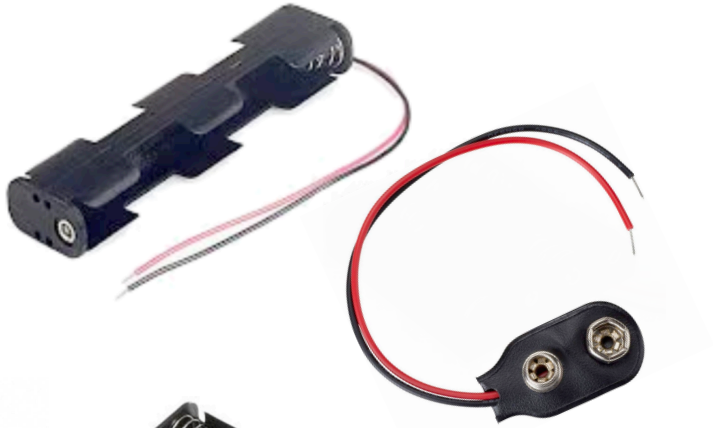
- It is likely that noise will be an issue as you drive motors.
- One solution is to use one battery for motors one battery for logic devices.





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 battery connectors in GMlab



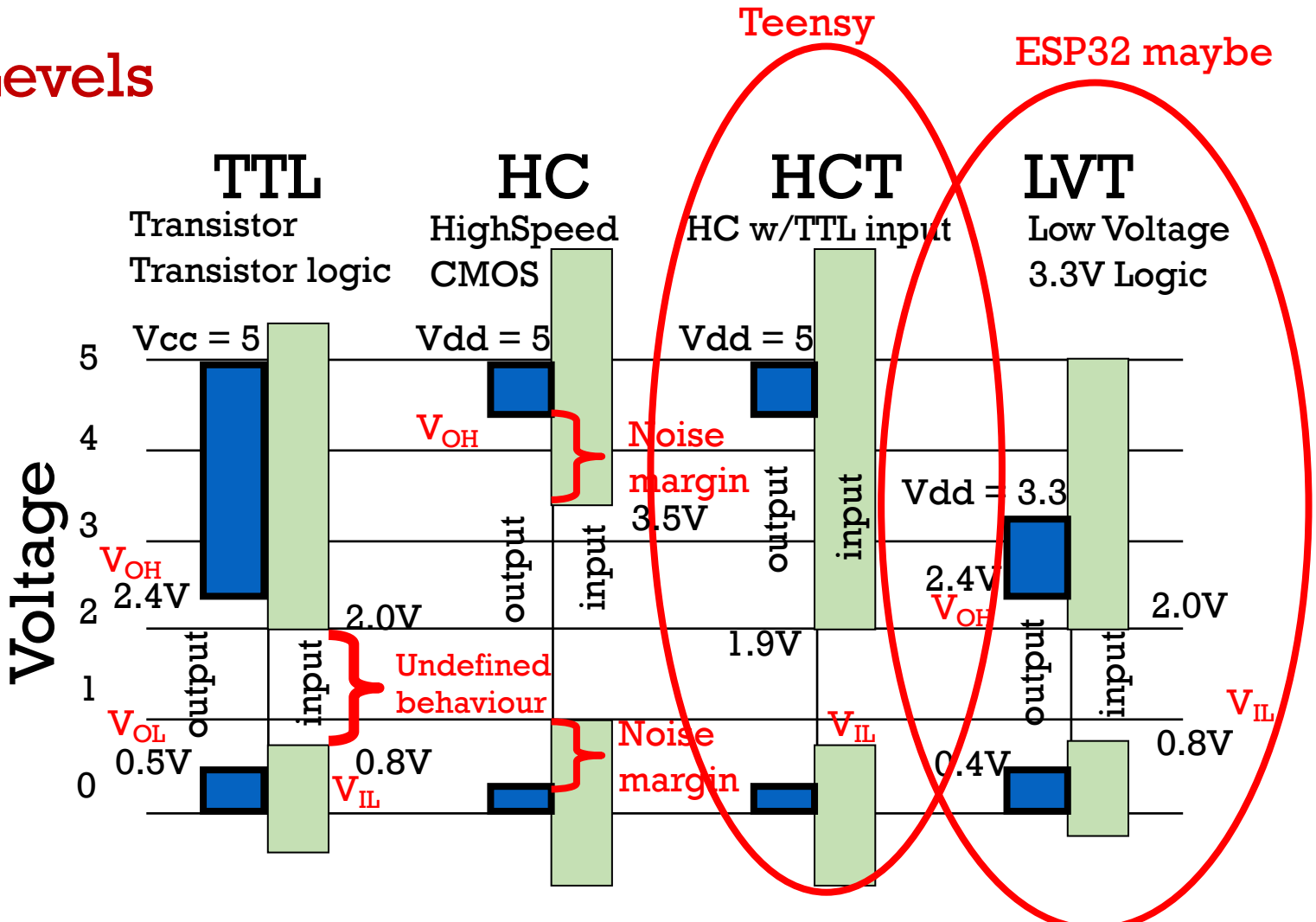
04

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 should work on TTL and HCT
  - 3.3V not guaranteed for HC
- In setup()

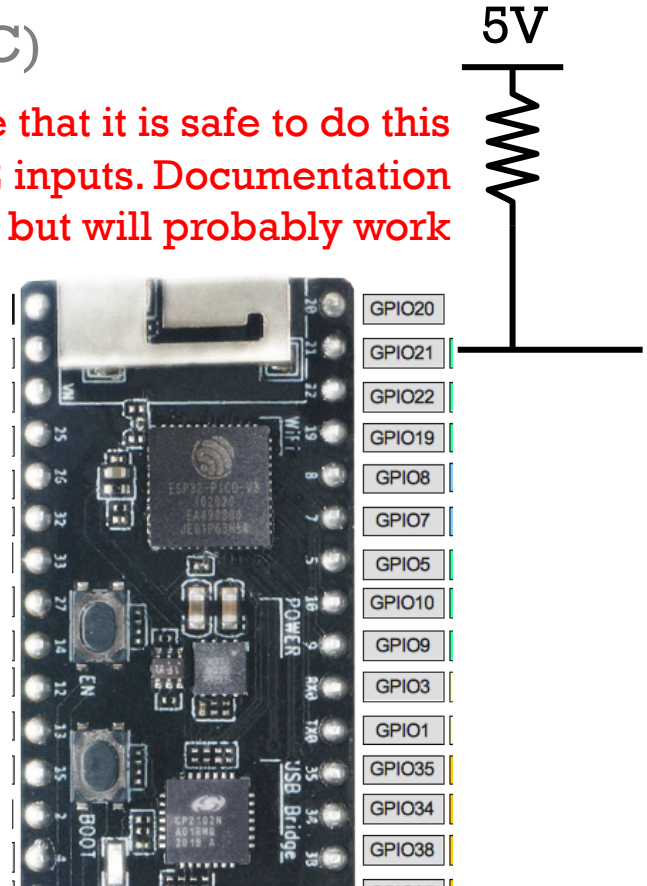
```
pinMode(23,INPUT);
digitalWrite(23,LOW);
```

- Don't use `digitalWrite.` Instead:
  - For high values we use:

```
pinMode(23,INPUT);
```

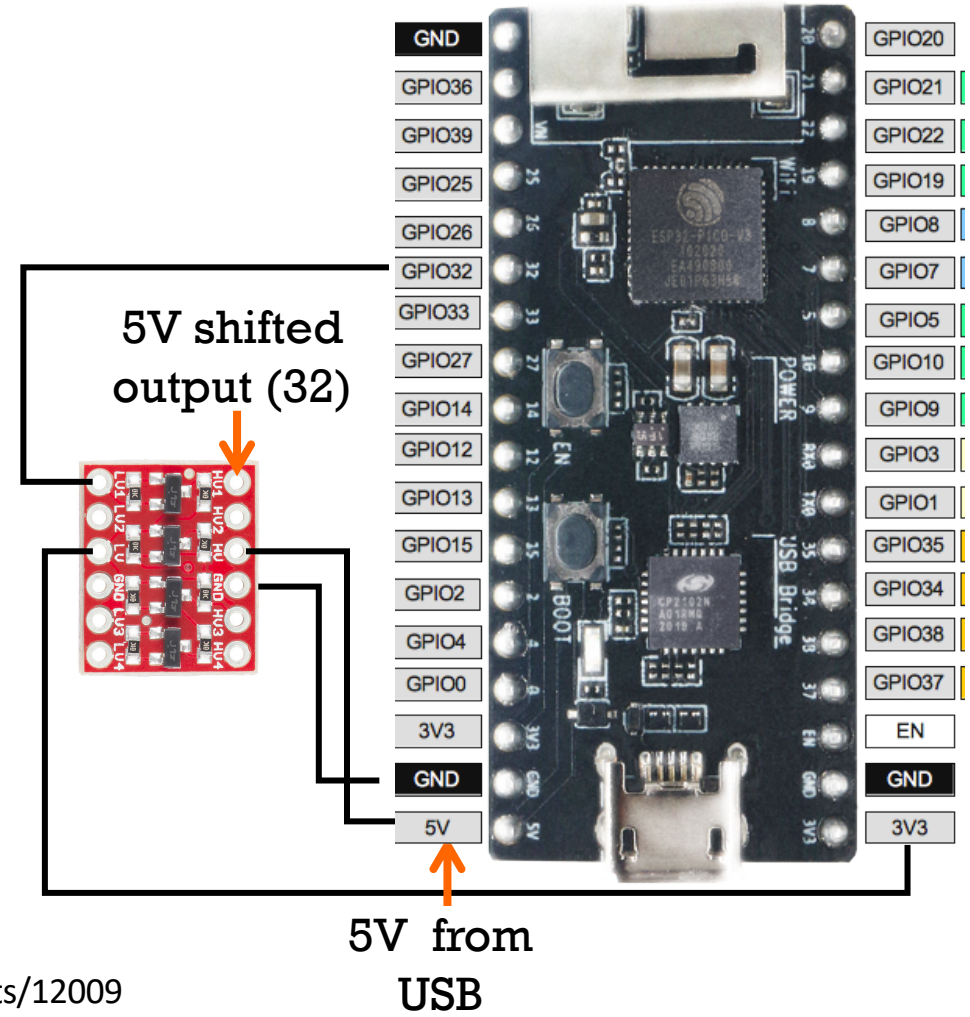
  - For low values we use:

```
pinMode(23,OUTPUT);
```



Better Level Shifting

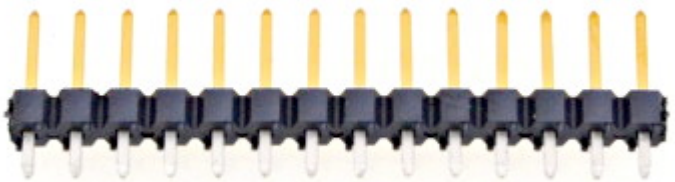
- Can use a level shifter and 5V from USB (Vin) as our 5V source.
- If Pin 32 is 3.3V, the output is 5V. Otherwise both are gnd.
- Four channel level shifter from sparkfun
  - Can shift 4 pins input or output

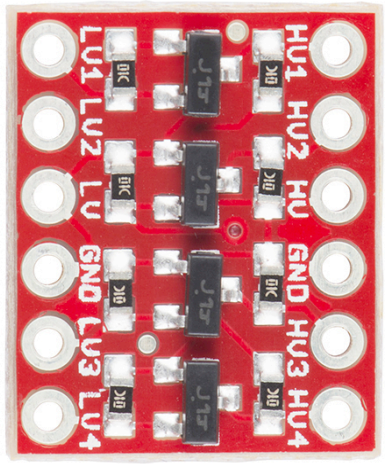


https://www.sparkfun.com/products/12009

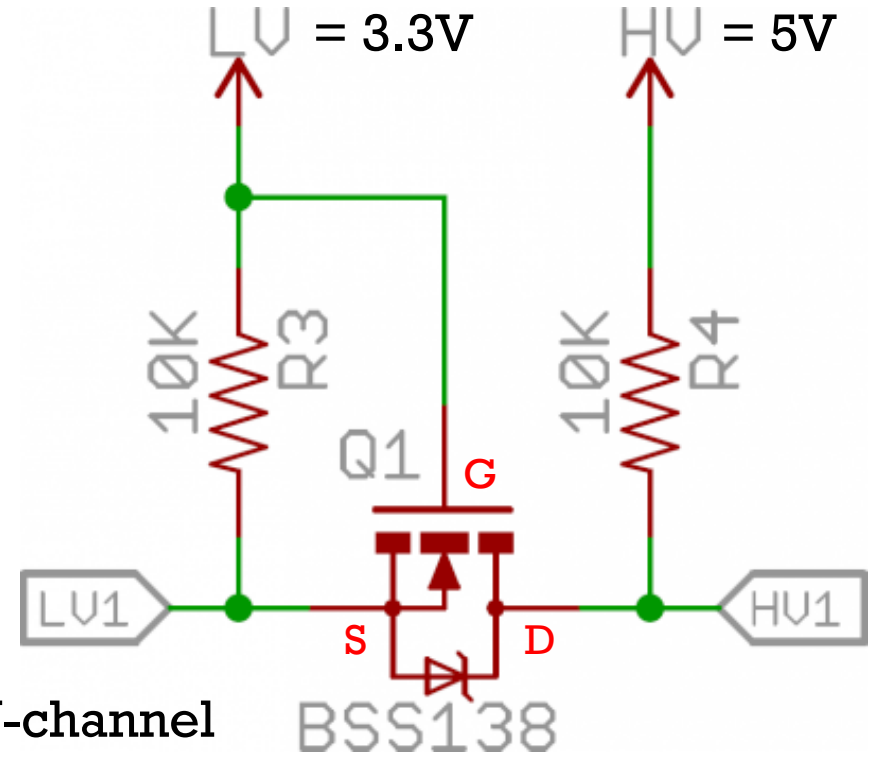
Soldering level shifter (BOB 12009)

- 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.





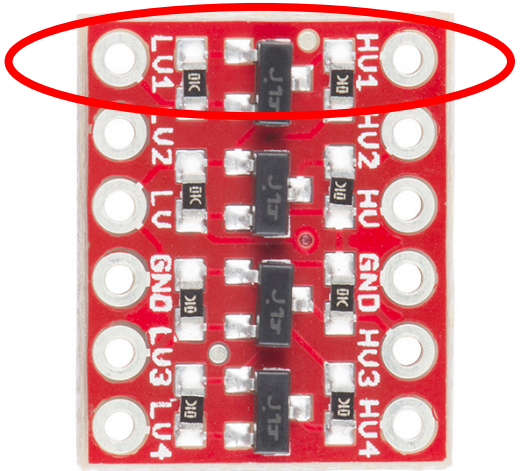
Level Shifter



N-channel MOSFET

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

- N-channel MOSFET
  - Becomes a low value resistor between S and D when $V_G > V_S$ by ~2.5V
  - Otherwise very high resistance between S and D
  - G-S and G-D are not connected (infinite resistance)



Level Shifter

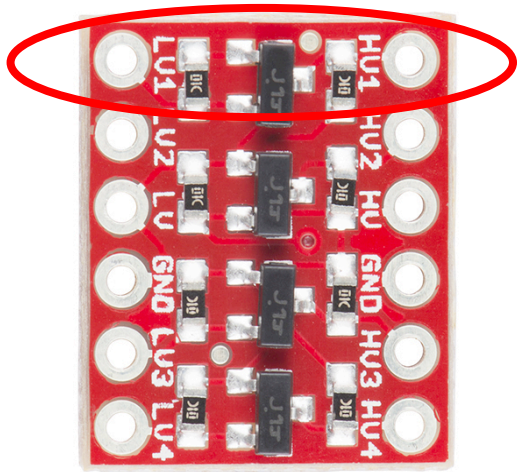
Four cases:

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

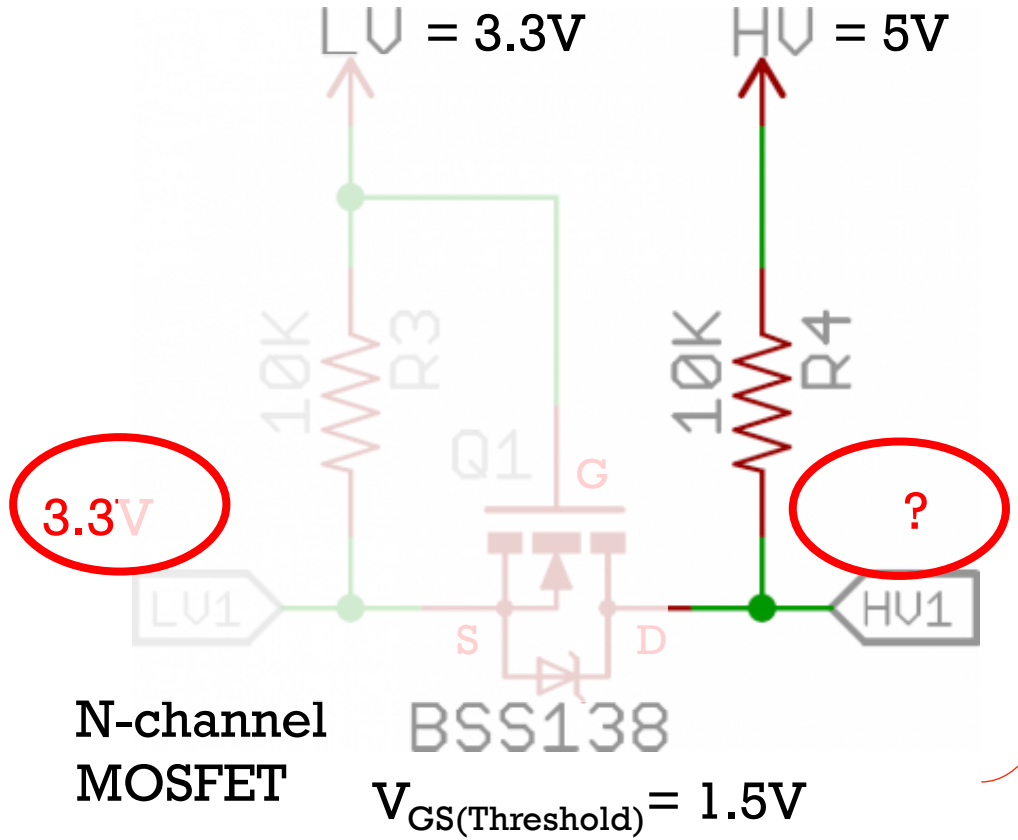
N-channel MOSFET

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

- N-channel MOSFET
  - Becomes a low value resistor between S and D when $V_G > V_S$ by ~2.5V
  - Otherwise very high resistance between S and D
  - G-S and G-D are not connected (infinite resistance)



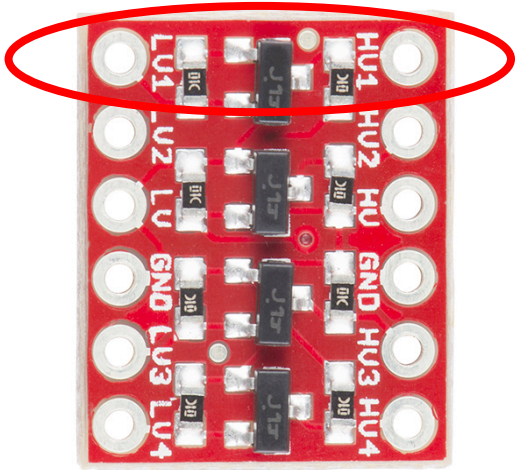
Q10: Level Shifter



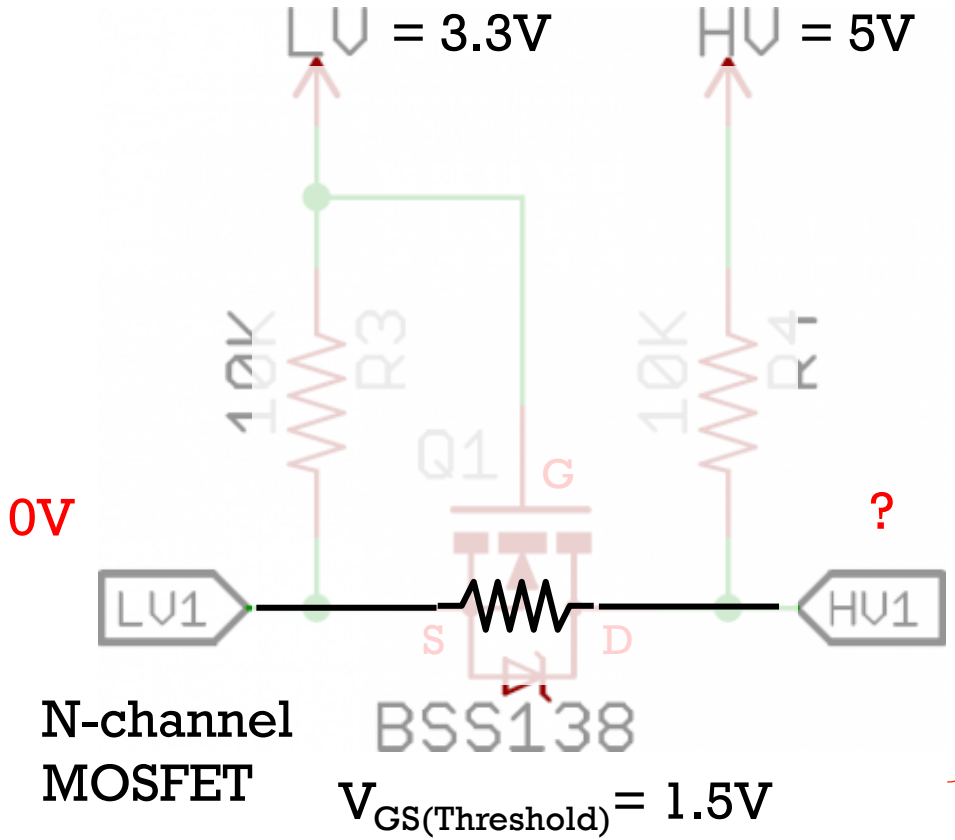
N-channel MOSFET

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

- With 3.3V at LV1, what voltage is at HV1 and what is the max current supplied to HV1 to maintain proper logic levels?



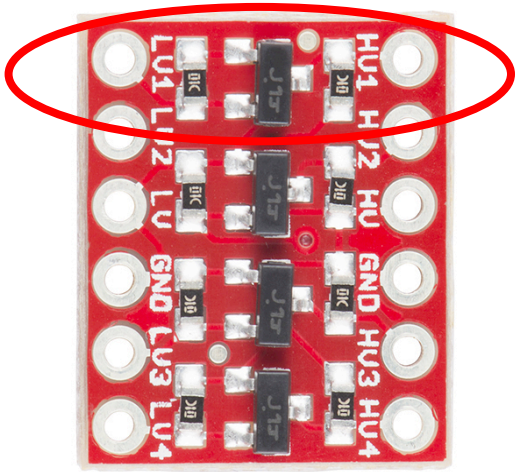
Q11: Level Shifter



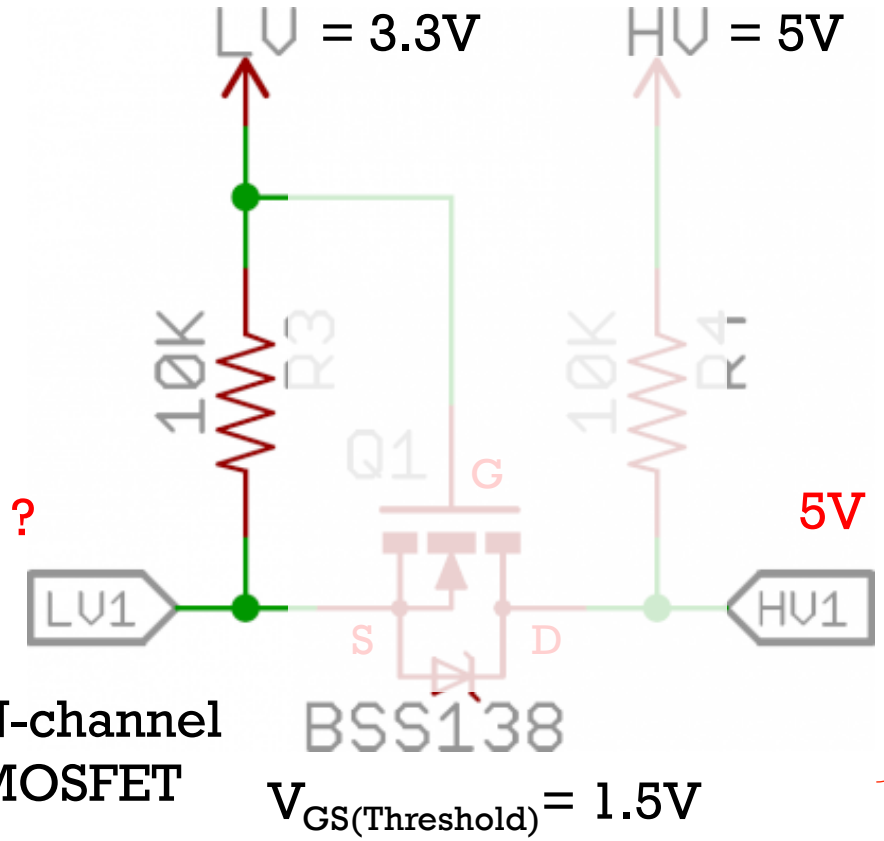
N-channel MOSFET

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

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



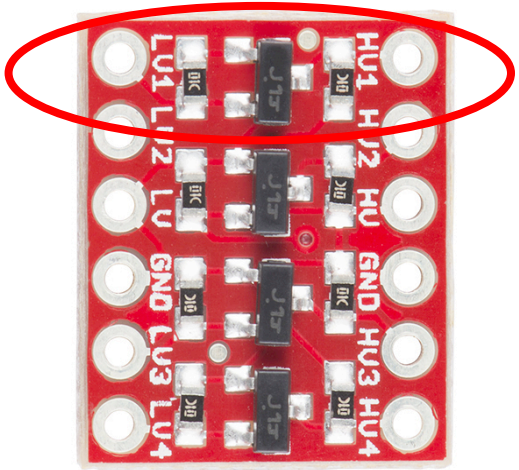
Q12: Level Shifter



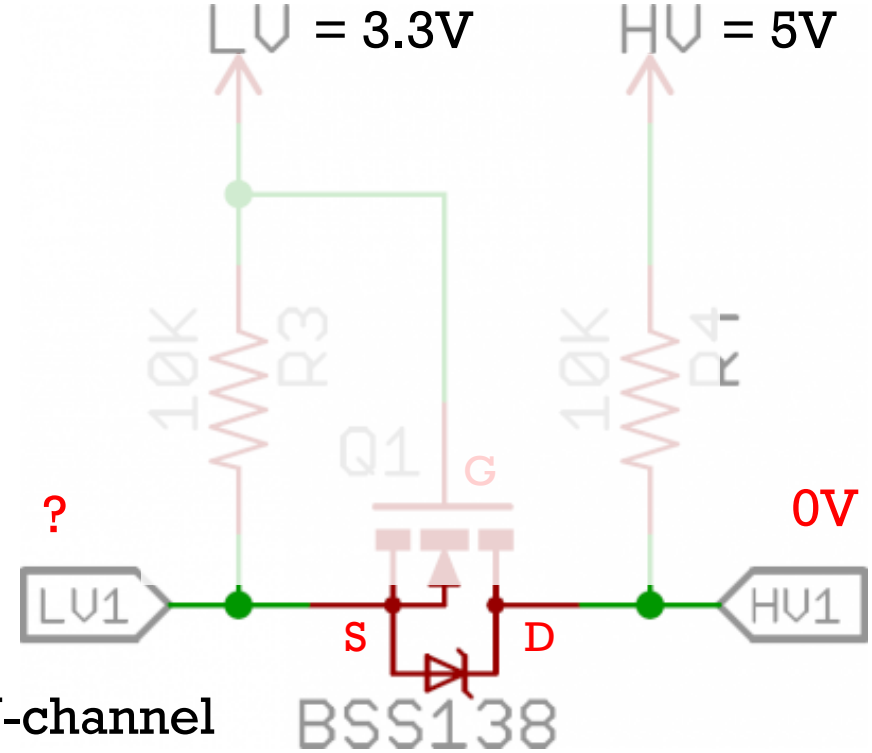
N-channel MOSFET

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

- With 5V at HV1, what voltage is at LV1 and what is the max current supplied to LV1 to maintain proper logic levels?



Q13: Level Shifter



N-channel MOSFET

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

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

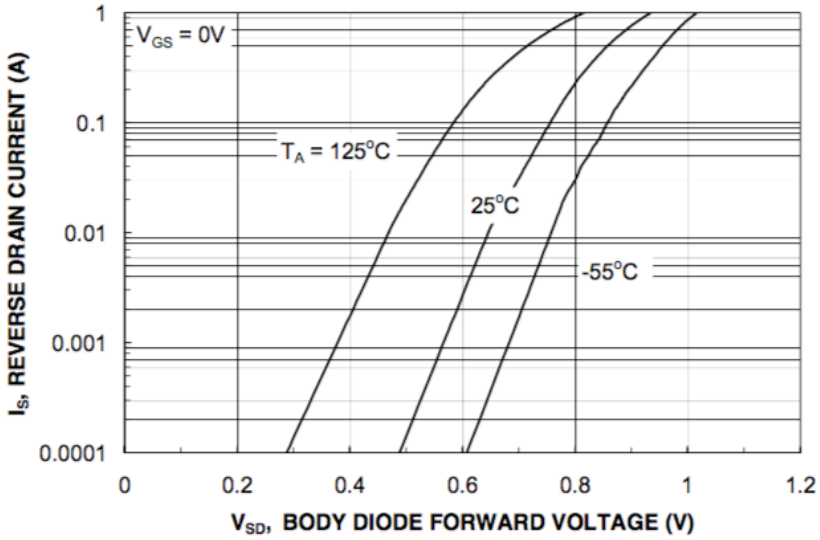


Figure 6. Body Diode Forward Voltage Variation with Source Current and Temperature.

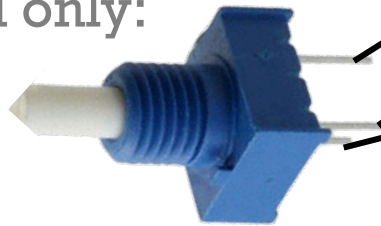
Summary

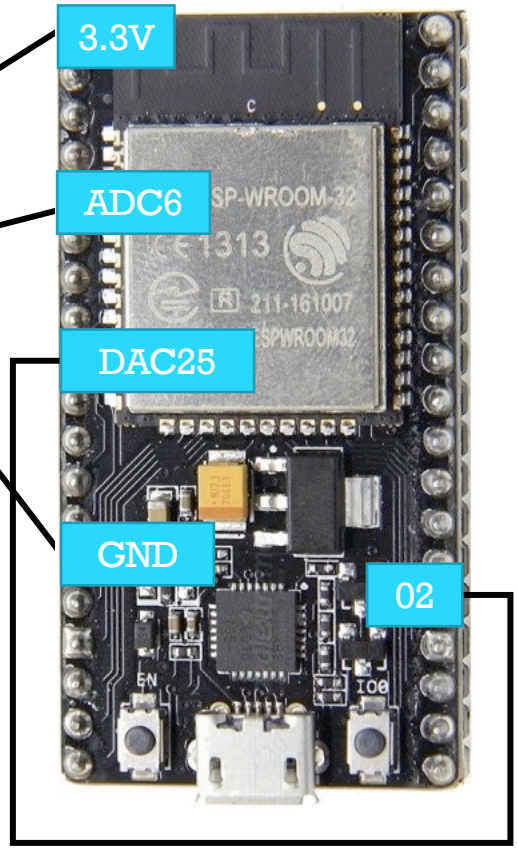
- Use Level shifters with ESP32 to interface with 5V both input and output
- Use voltage regulators if you need 5V. ESP32 has onboard regulator up to 12V.
- Important to route power wires directly to power source – DO NOT DAISY CHAIN.
- The PICO power design allows simultaneous USB and battery, but USB power is finicky.

Q-extra: Puzzle

- What is the shortest arduino code that will allow you to set the brightness of an LED with a potentiometer and only:
  - a ESP32,
  - a protoboard,
  - four wires.

[shortest means fewest characters]






```
void setup() {

}
void loop() {

}
```