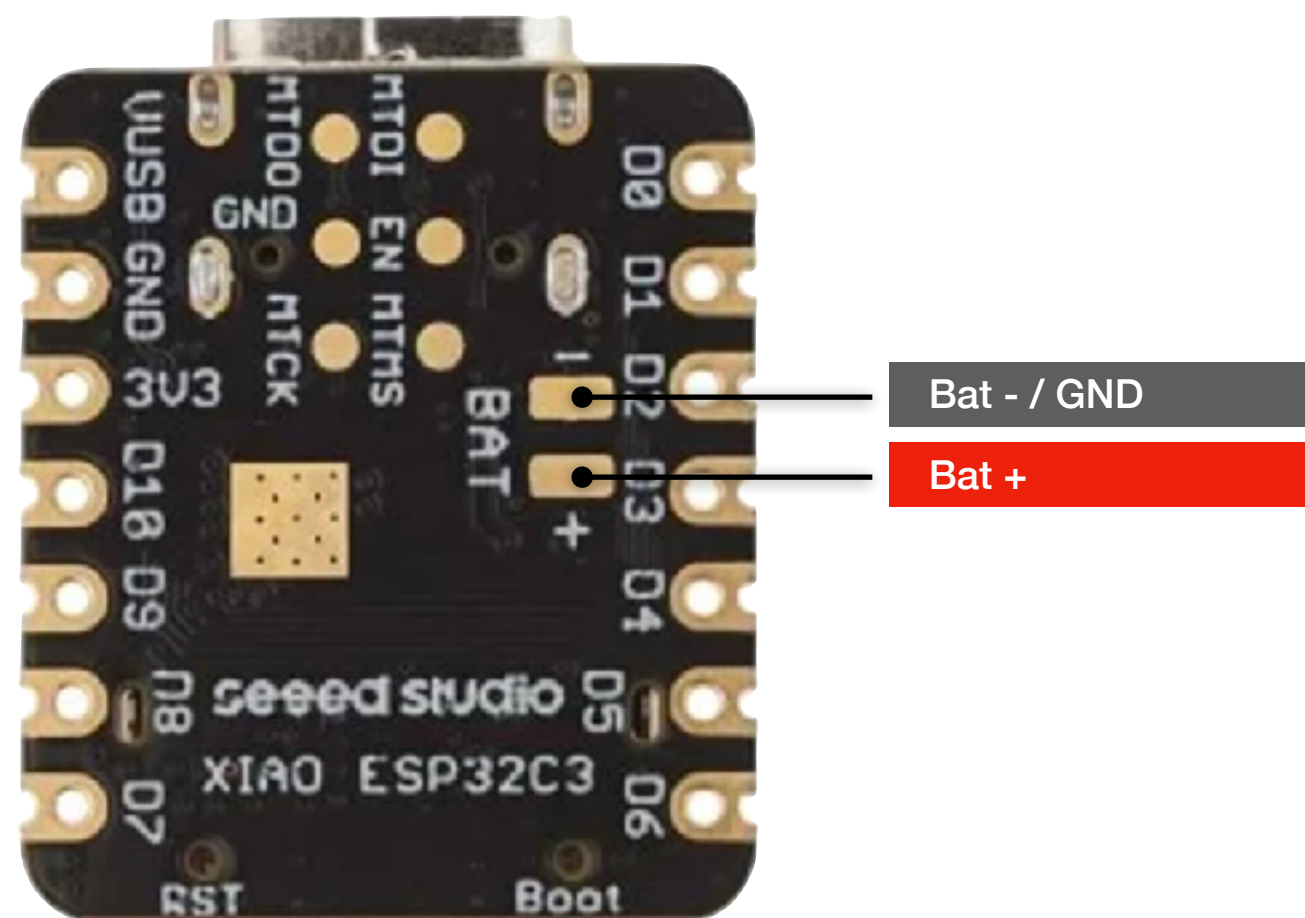
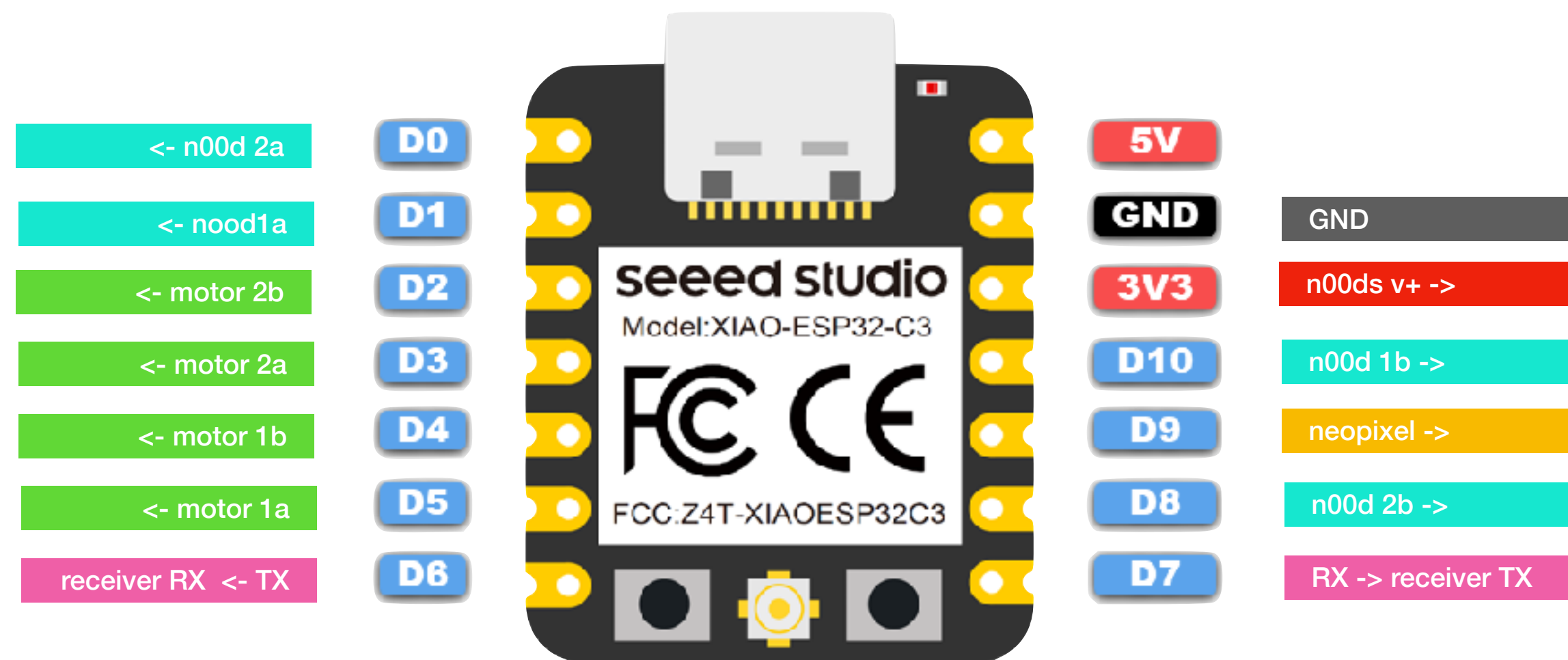
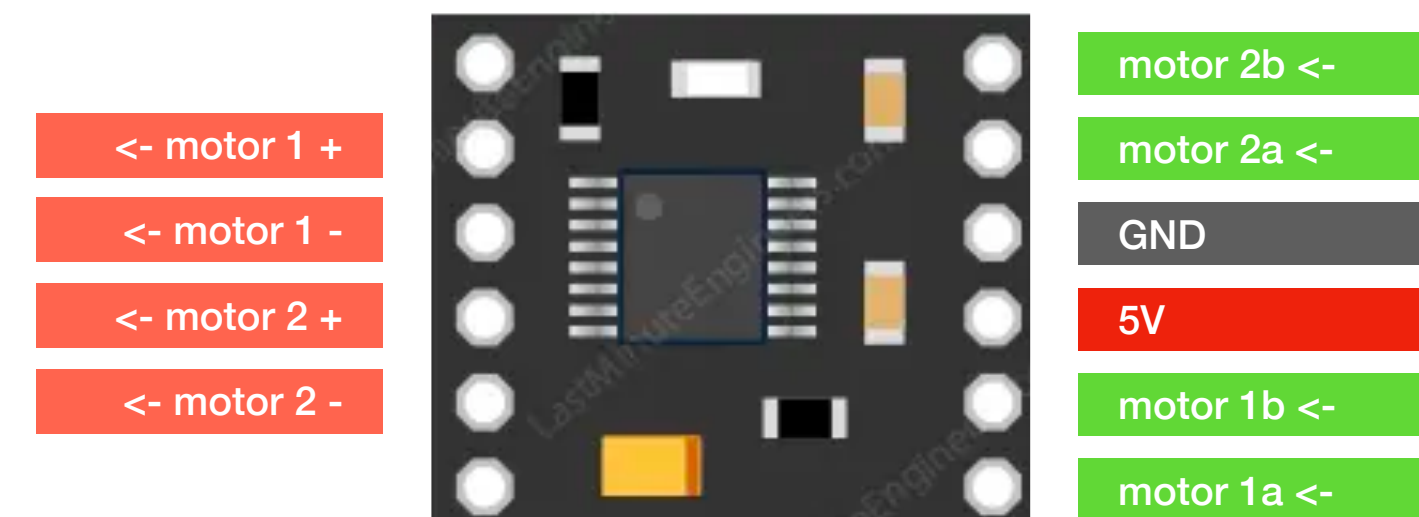


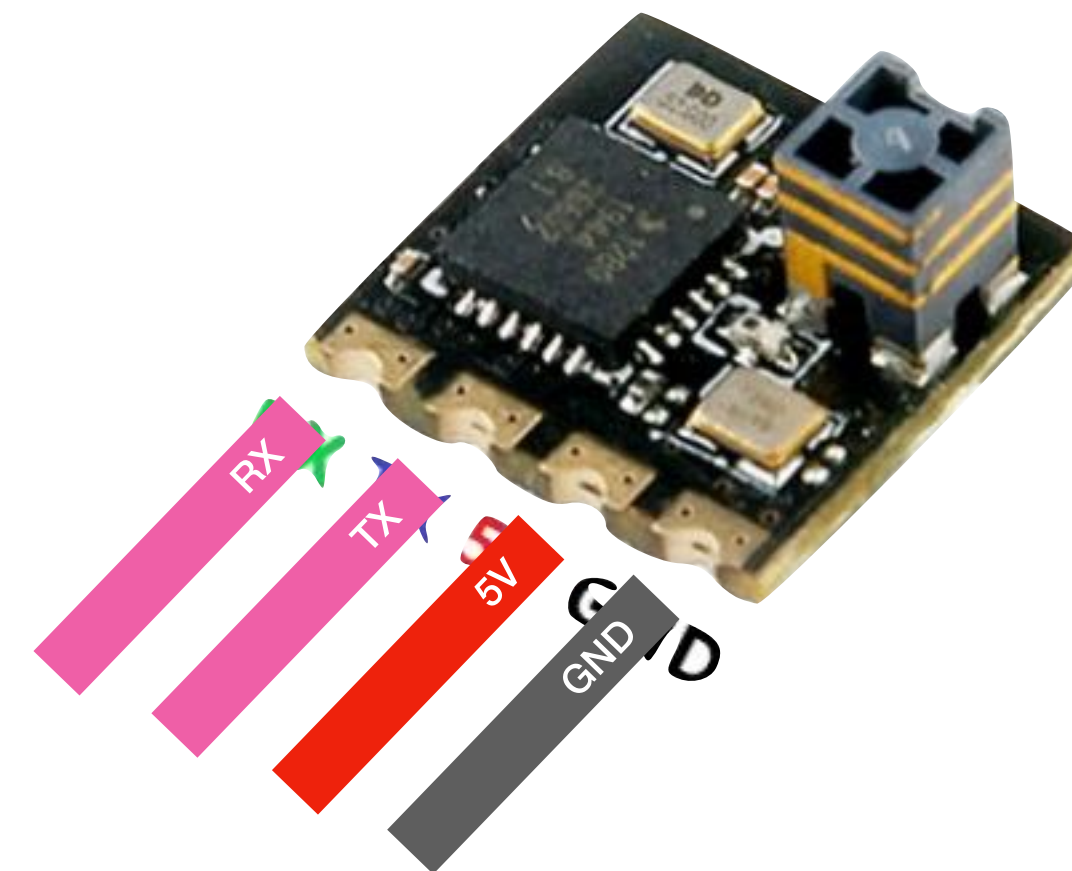
XIAO ESP32-C3



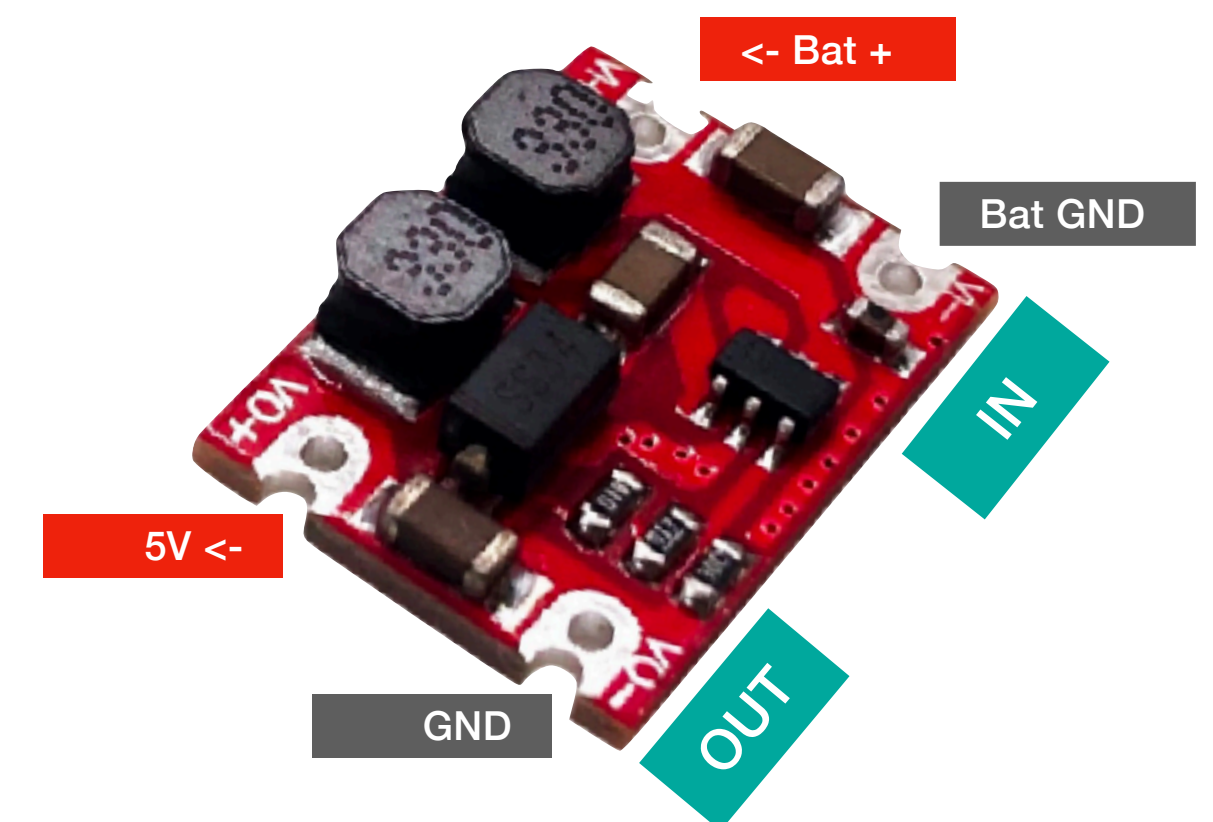
Generic DRV8833 motor driver



Happymodel EP2 TXCO ELRS rx



600ma 5v boost converter



XIAO ESP32-C3

receiver RX <- TX

<- motor 1a

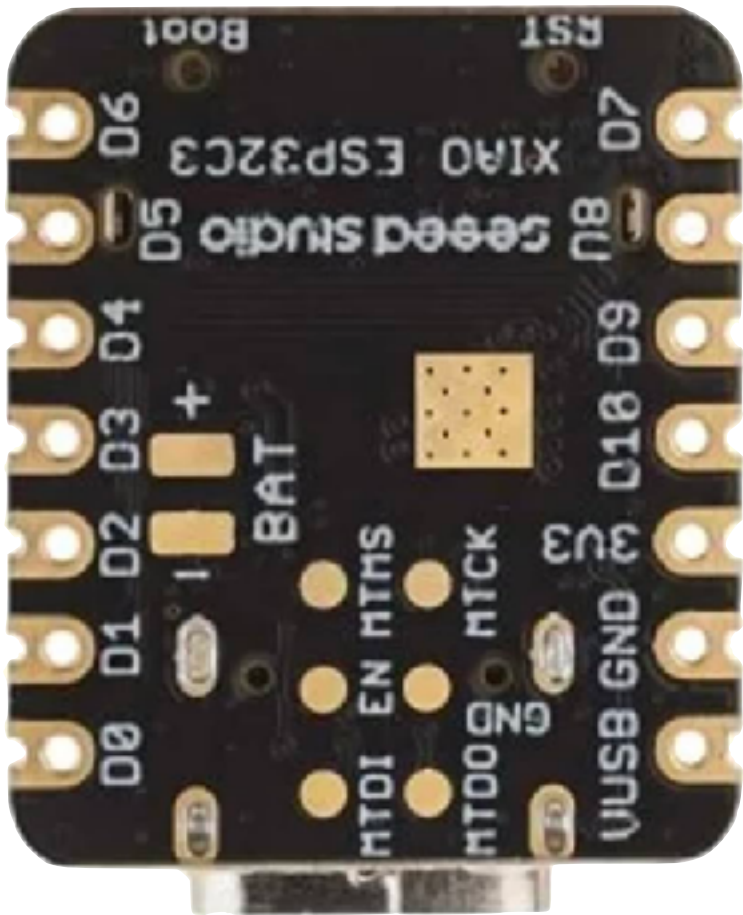
<- motor 1b

<- motor 2a

<- motor 2b

<- nood1a

<- n00d 2a



RX -> receiver TX

n00d 2b ->

neopixel ->

n00d 1b ->

n00ds v+ ->

GND

<- n00d 2a

<- nood1a

<- motor 2b

<- motor 2a

<- motor 1b

<- motor 1a

receiver RX <- TX

D0

D1

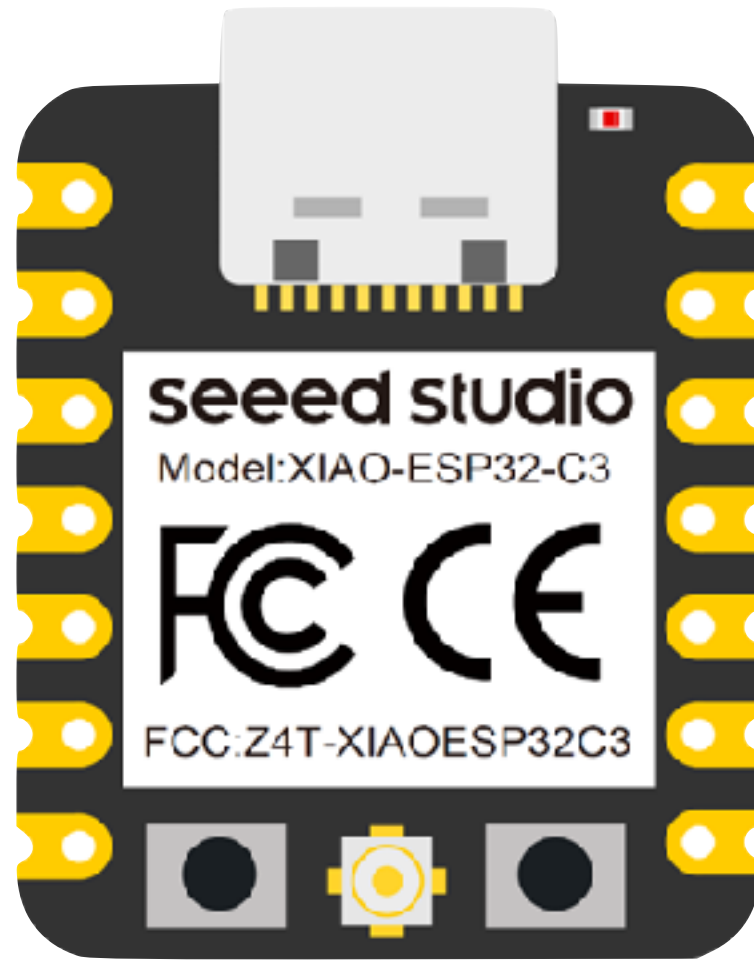
D2

D3

D4

D5

D6



5V

GND

3V3

D10

D9

D8

D7

GND

n00ds v+ ->

n00d 1b ->

neopixel ->

n00d 2b ->

RX -> receiver TX

GND

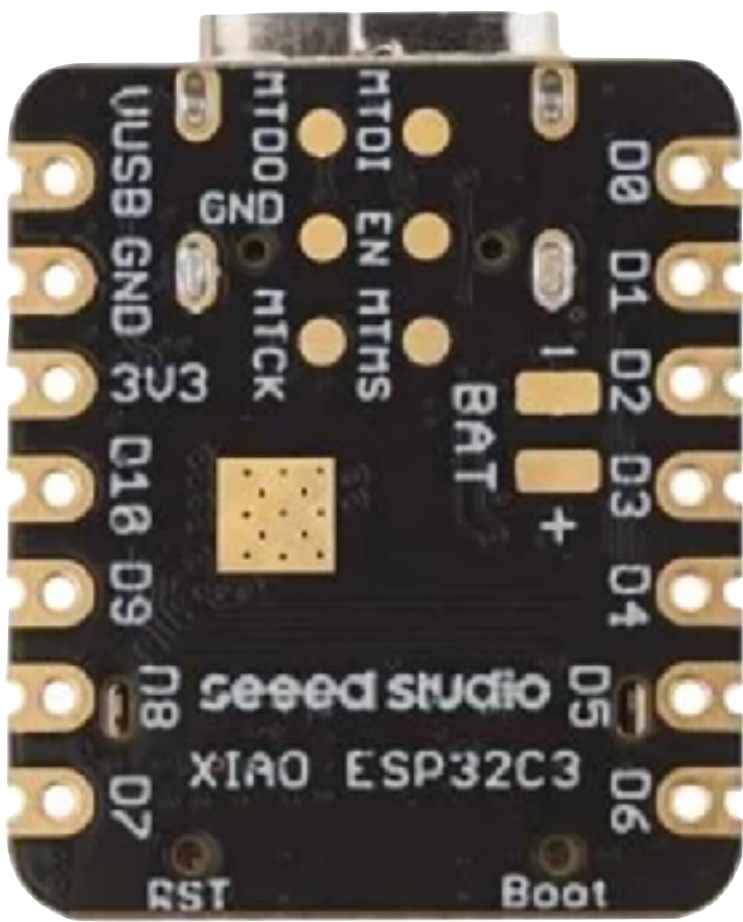
n00ds v+ ->

n00d 1b ->

neopixel ->

n00d 2b ->

RX -> receiver TX



<- n00d 2a

<- nood1a

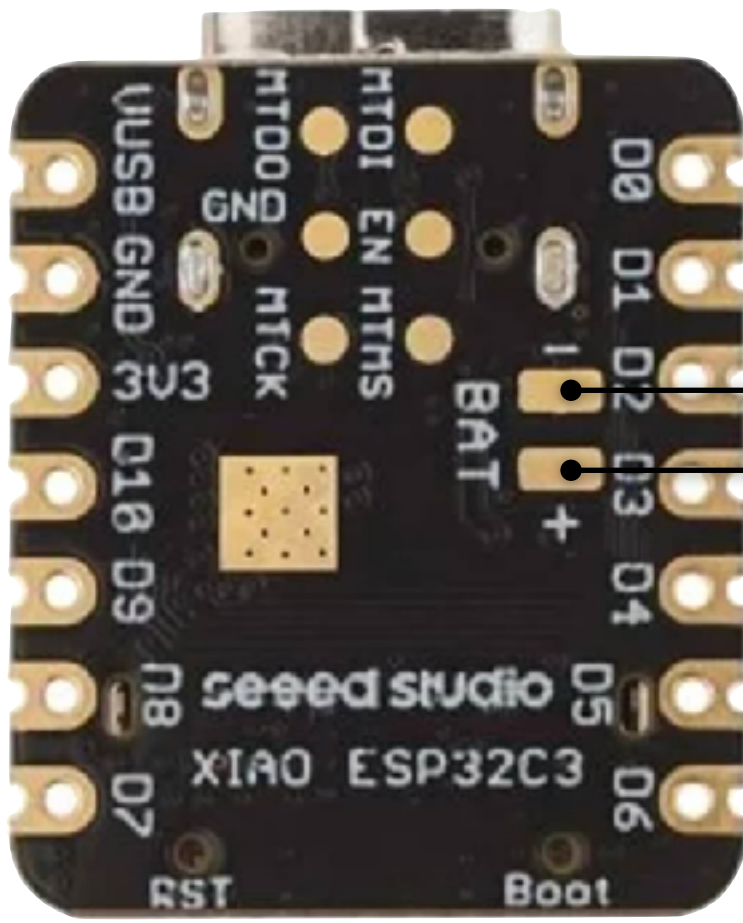
<- motor 2b

<- motor 2a

<- motor 1b

<- motor 1a

receiver RX <- TX



Bat - / GND

Bat +

Comm protocol of n00ds

- * Midi is 0-127
- * ExpressLRS is theoretically 0-2000, but the absolute range on the receiving end is in fact 191-1792. This is 1601 values in theory, and we map this back to 0-1023 to be able to apply binary logic

0-1023 is 10 bits

0 0 0 0 0 0 0 0 0 0

- * top 3 bits are used to encode the index of the n00d (1, 2, 3, 4) - psuedo big-endian

n00d blue - front

1 0 0 0 0 0 0 0 0 0

n00d blue - back

1 0 1 0 0 0 0 0 0 0

n00d pink - back

1 1 0 0 0 0 0 0 0 0

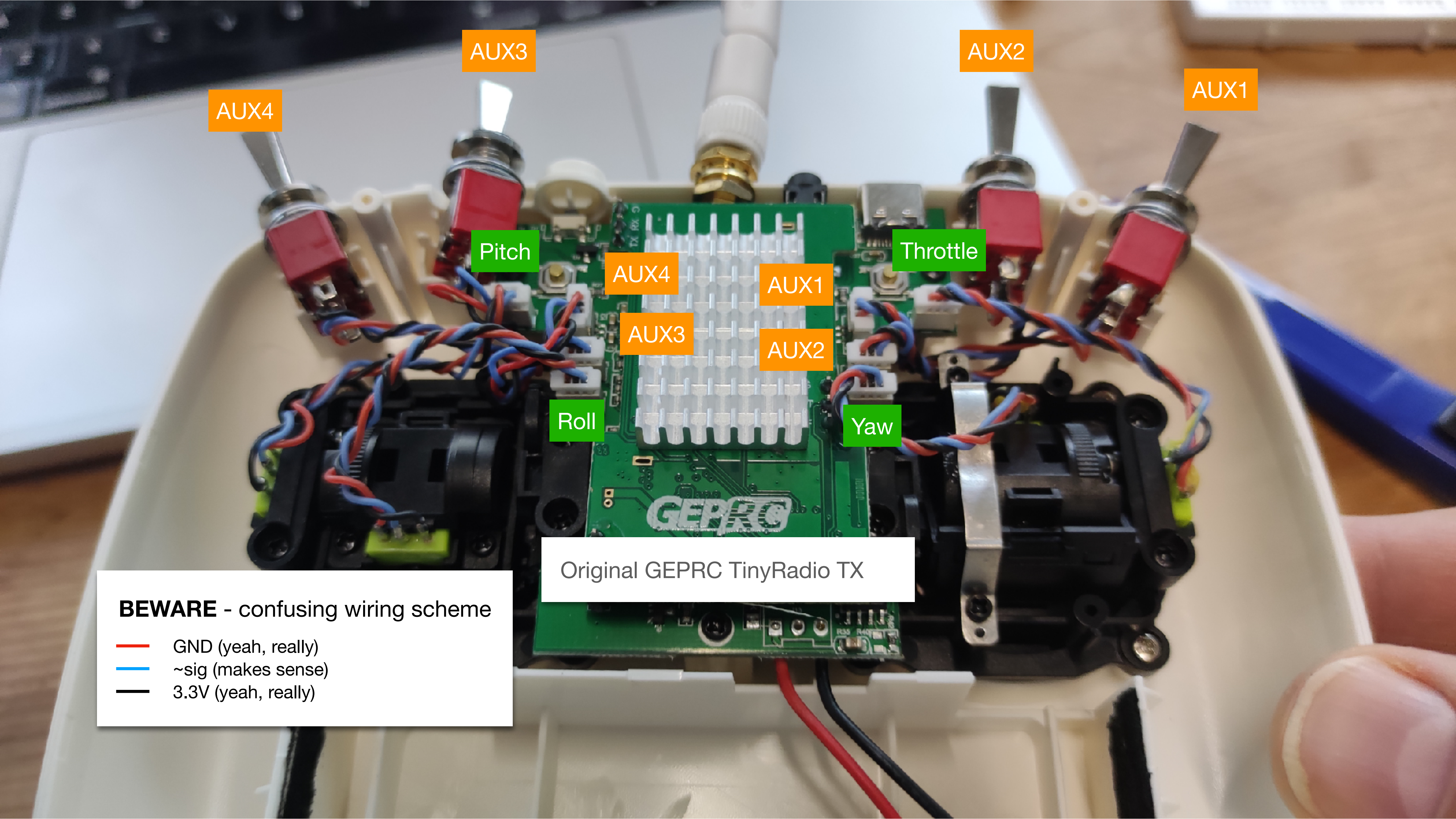
n00d pink - back

1 1 1 0 0 0 0 0 0 0

- * bottom 7 bits (0-127) are used for the value of the n00d
- * I effectively used 0-111. This is 16 values less, to prevent incidental binary overflow upwards in the case of signal jitter

n00d value (example)

1 0 0 1 0 1 1 0 0 1



AUX3

AUX2

AUX1

AUX4

Pitch

Throttle

AUX4

AUX1

AUX3

AUX2

Roll

Yaw

Original GEPRC TinyRadio TX

BEWARE - confusing wiring scheme

- GND (yeah, really)
- ~sig (makes sense)
- 3.3V (yeah, really)