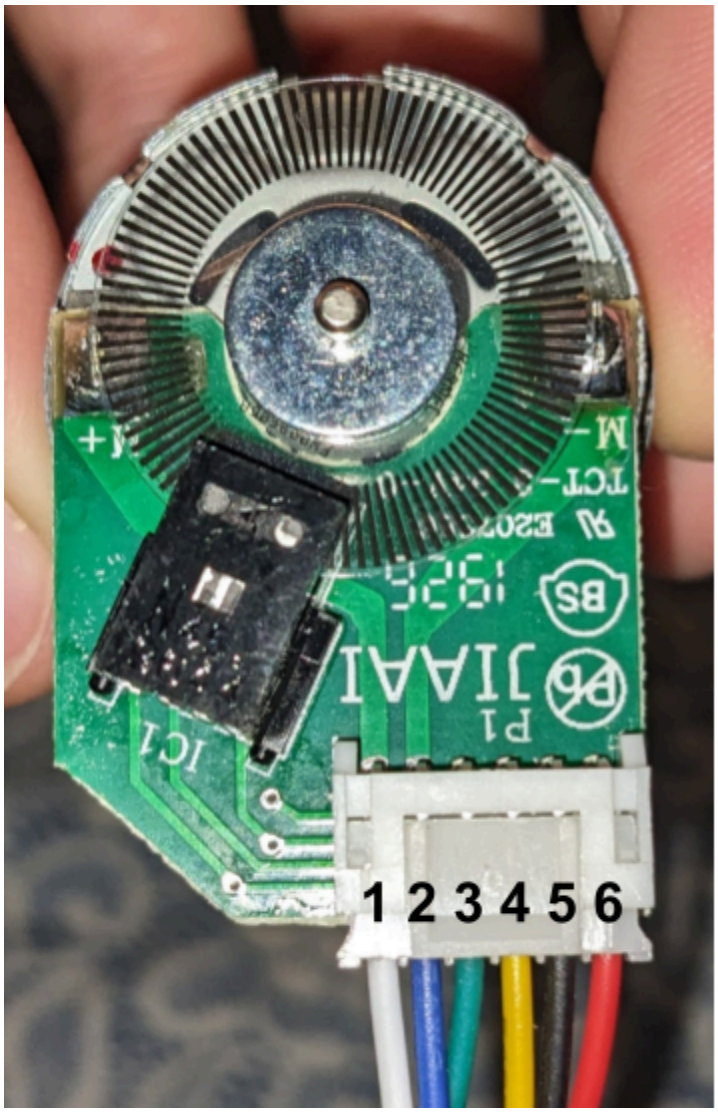
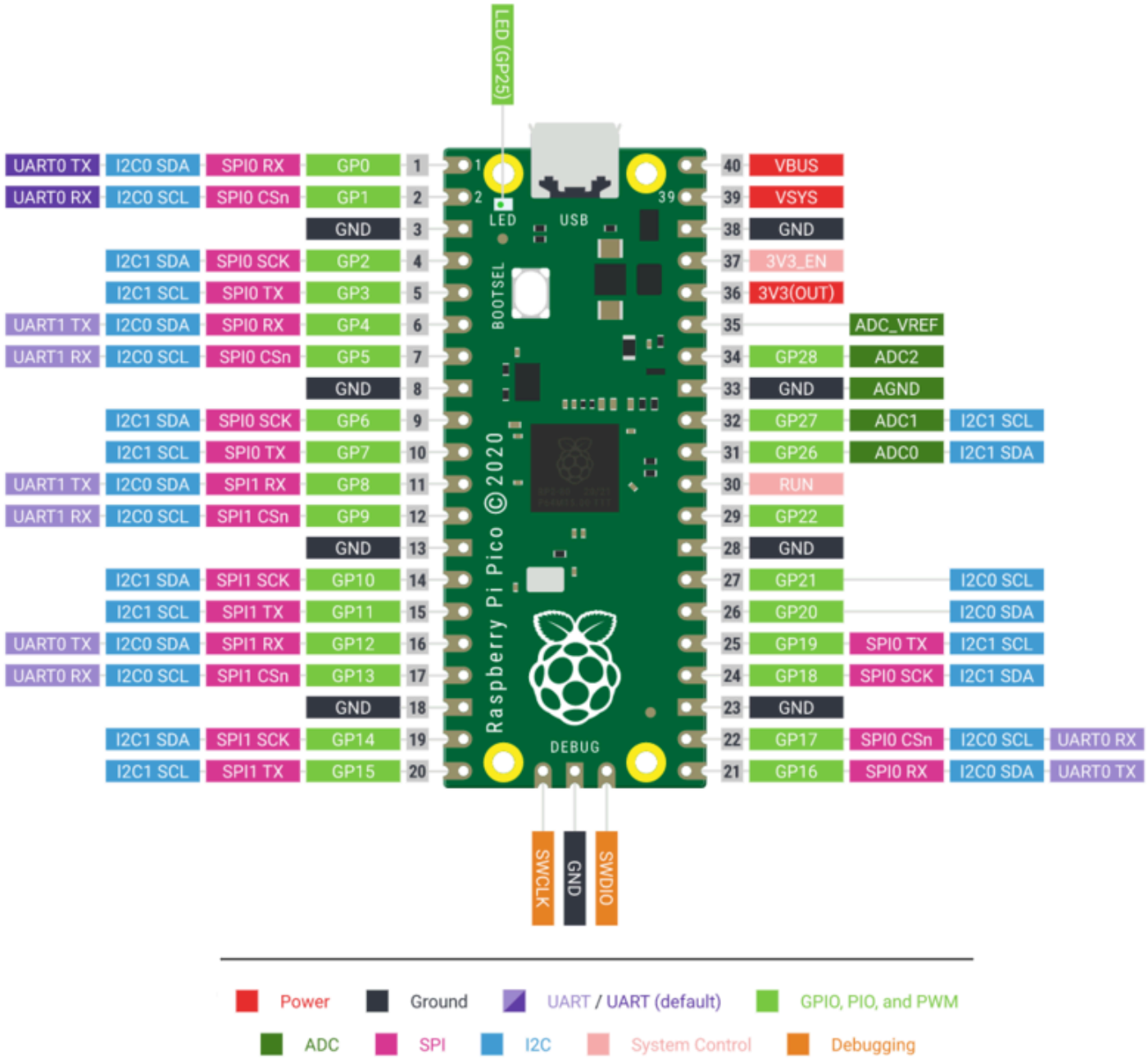
The motor pinout



Don’t trust the colors of the cable! Don’t trust the labels on the breakout board!
Pin 1 - Motor +
Pin 2 - Motor -
Pin 3 - Encoder GND
Pin 4 - Encoder B
Pin 5 - Encoder 3.3V
Pin 6 - Encoder A
The encoder has **96 lines**

The Raspberry Pi Pico H



https://www.raspberrypi.com/products/raspberry-pi-pico/

The Raspberry Pi Pico H breakout board contains an RP2040 microcontroller, with 2 ARM M0+ cores at 133MHz, 264k RAM, and 2M flash. It uses a MSD bootloader, so when you plug it into your computer with the BOOTSEL button pressed it will appear as a thumb drive named RP-RP2. Compiled code has the extension .uf2, drag a .uf2 file onto the RP-RP2 drive and the Pico will program itself.

Reading the encoder over USB

Program the Pico with pico_encoder_to_usb.uf2. Connect the encoder A to the Pico GP14, and encoder B to the Pico GP15, and the encoder 3.3V to the Pico 3V3(OUT) pin and the encoder GND to the pico GND. The Pico will enumerate as a USB device (on Windows, something like COM4, on Mac/Linux, something like /dev/tty.usbmodem1101). Open the port in Putty or screen, and the Pico will print the quadrature count and velocity at 10Hz. For 96 lines, the Pico will record 96*4 = 384 counts per revolution.

Reading the encoder over serial

Program the Pico with pico_encoder_to_serial.uf2.
Keep the encoder A and B pins in the Pico GP14 and GP15 pins.
Remove the Pico USB cable and connect the Pico 3V3(OUT) to the PIC 3.3V pin so that the PIC board is powering the Pico.
The Pico is programmed to communicate over UART on pins GP0/TX and GP1/RX with a baud of 230400.
Connect the Pico TX to the PIC U2RX, and the Pico RX to the PIC U2TX.
When you print ‘a’ from the PIC to the Pico, the Pico will print back the encoder count as an integer followed by a newline.
When you print ‘b’ from the PIC to the Pico, the Pico will reset the encoder count to 0 and will not reply.
On the PIC, see the encoder sample code.
The PIC will use the UART2 RX interrupt to read every letter that comes from the PICO. When the PIC gets a newline, it will sscanf the encoder count out of the character array, and set a flag variable to 1.
After using writeUART2(“a”), wait for the get_encoder_flag() function to return a 1, and then use the set_encoder_flag(0) function to clear the flag variable, and access the encoder count using the get_encoder_count() function.

```
// read encoder count
WriteUART2("a");
while(!get_encoder_flag()){}
set_encoder_flag(0);
char m[50];
int p = get_encoder_count();
sprintf(m,"%d\r\n",p);
NU32DIP_WriteUART1(m);
```