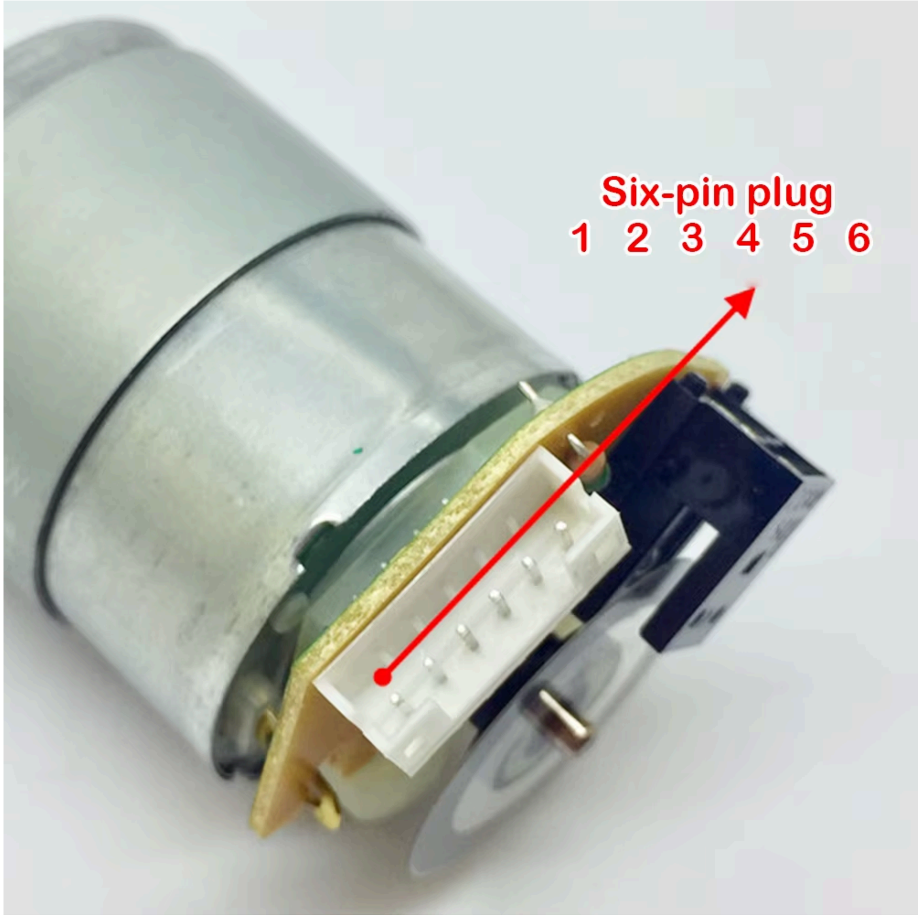
The motor pinout

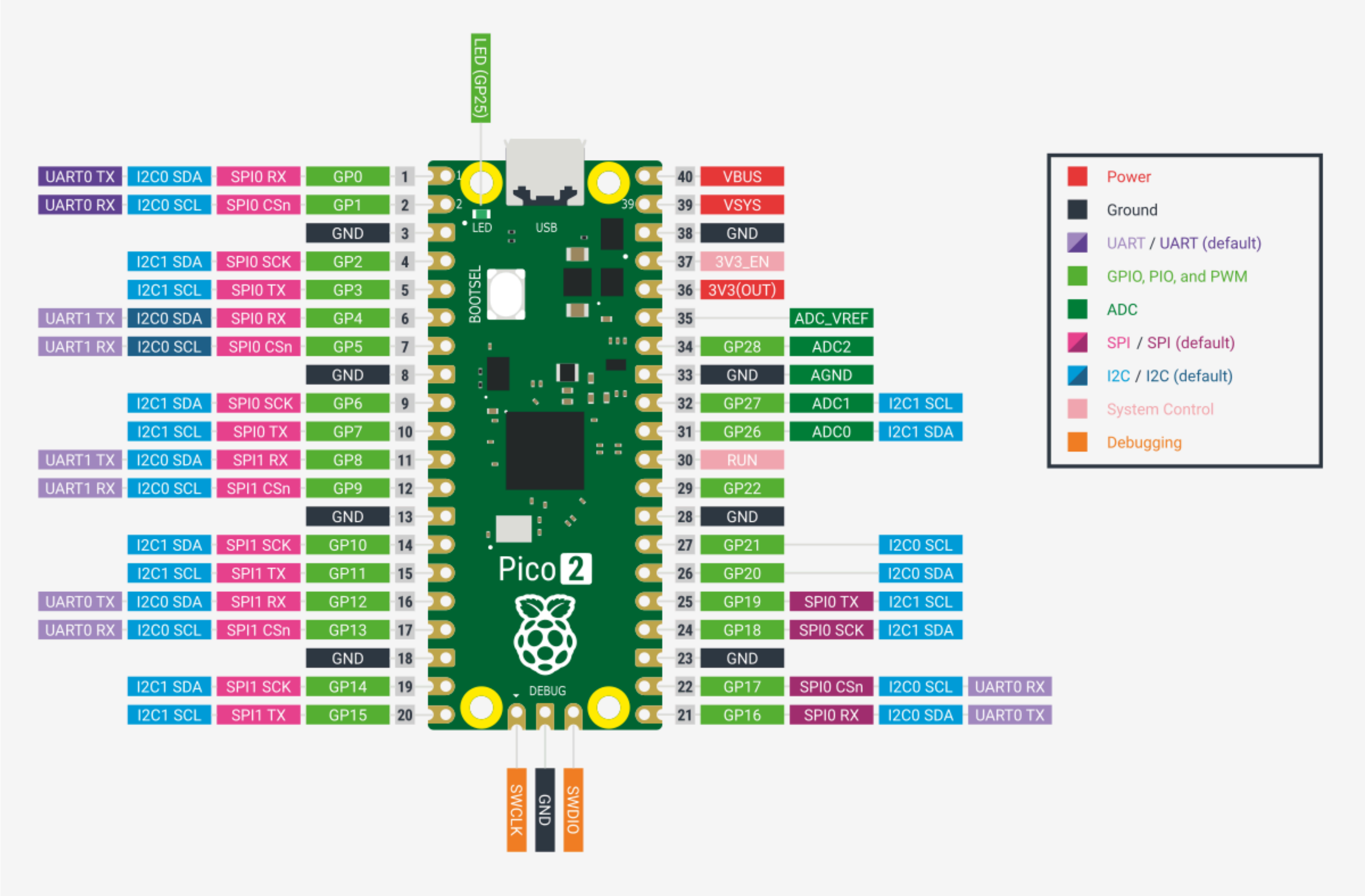


Don’t trust the colors of the cable!

1. Motor line -
2. Motor line +
3. Encoder VCC (3.3V)
4. Encoder GND
5. Encoder output A phase
6. Encoder output B phase

The encoder has **334 lines**

The Raspberry Pi Pico 2



https://www.raspberrypi.com/products/raspberry-pi-pico-2/

The Raspberry Pi Pico 2 breakout board contains an RP2350 microcontroller, with 2 ARM M33 cores (FPU!) at 150MHz, 512k RAM, and 4M flash. Or it can use 2 RISC-V cores. 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 RP2350. Compiled code has the extension .uf2. Drag a .uf2 file onto the RP-RP2 drive and the Pico will program itself.

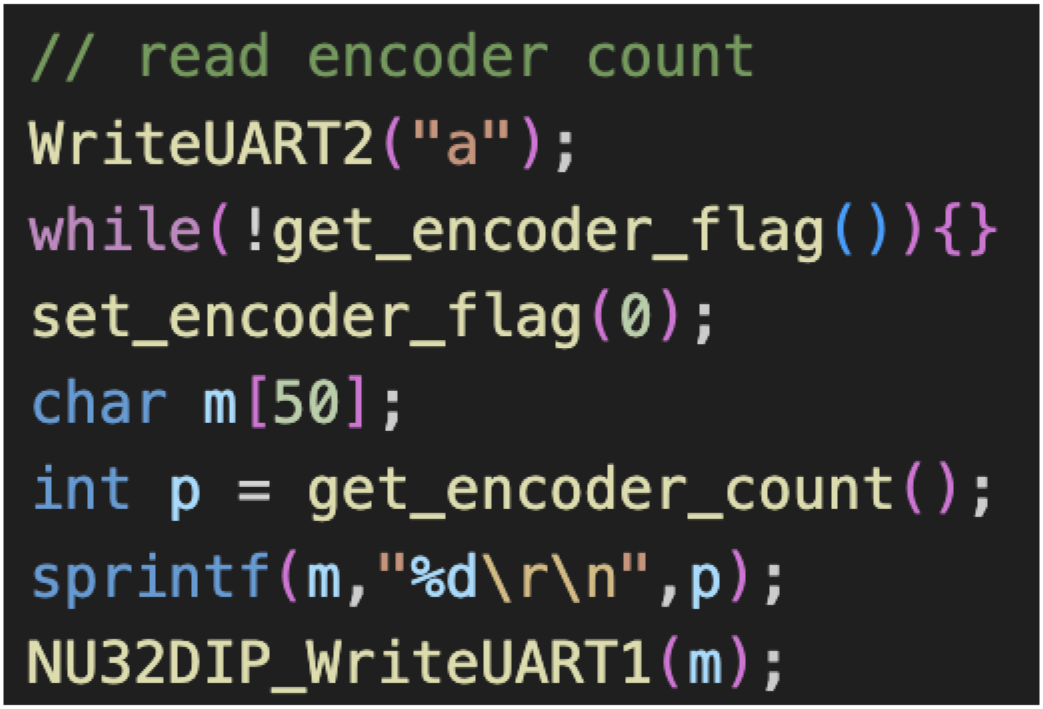
This board costs $5, and comes with a 3.3V regulator, USB port, buttons, resistors and capacitors. The PIC32MX170F256B by itself is $5.08!

Reading the encoder over USB

Program the Pico with pico2_encoder_to_usb.uf2. Connect the encoder phase A to the Pico GP14, and encoder phase B to the Pico GP15, and the encoder VCC 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 334 encoder lines lines, the Pico will record 334*4 = 1336 counts per revolution.

Reading the encoder over serial

Program the Pico with pico2_encoder_to_serial.uf2.
Keep the encoder A and B pins in the Pico GP14 and GP15 pins.
Remove the Pico USB cable. Connect the Pico GND to the PIC gnd, 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/UART0TX and GP1/UART0RX with a baud of 230400.
Connect the Pico TX to the PIC U2RX (pin B1), and the Pico RX to the PIC U2TX (pin B0).
Add encoder.c and encoder.h to your PIC project folder. Call UART2_Startup() in main().
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.
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.



```c
// 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);
```