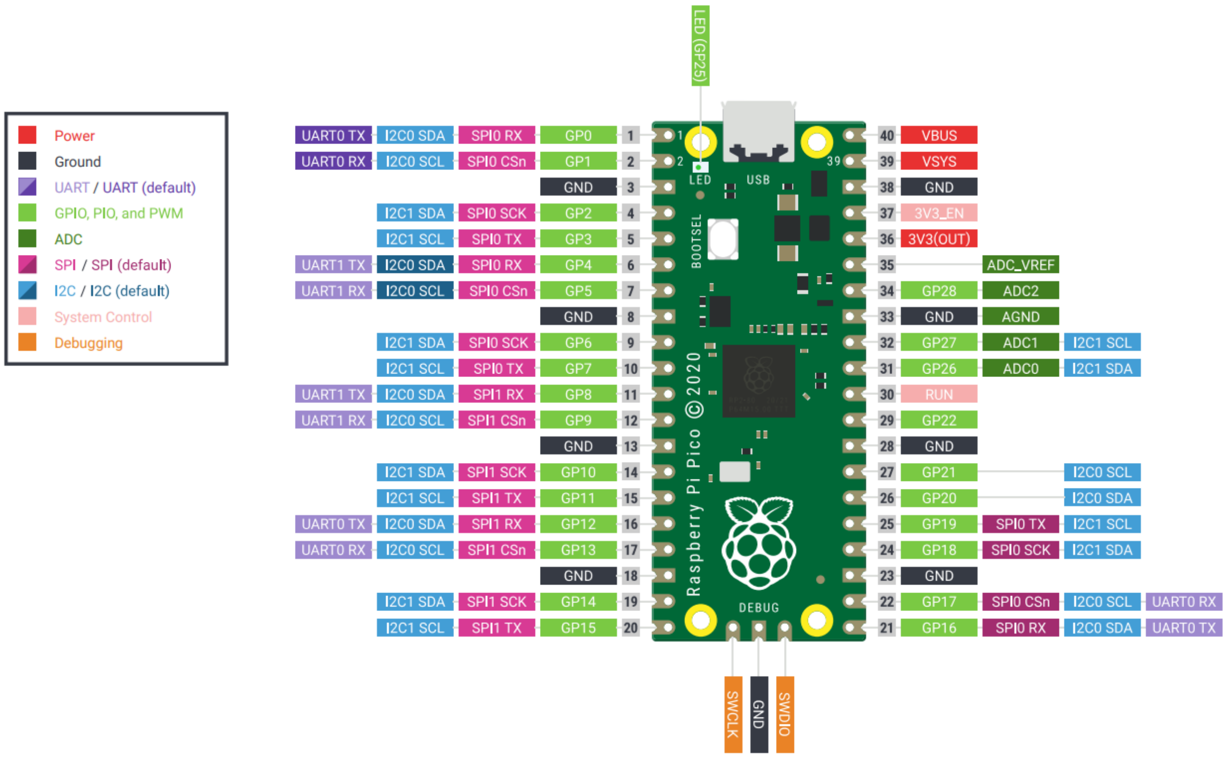
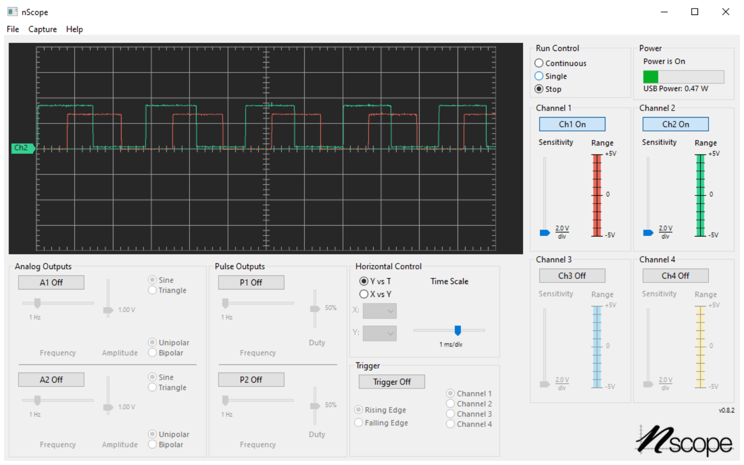
Reading the motor encoder

The motor is from Aliexpress.
The breakout board plugged into the motor cable has the following connections:
Pin 1 - Encoder B
Pin 2 - Encoder A
Pin 3 - Encoder Ground
Pin 4 - Encoder 3.3V
Pin 5 - Motor -
Pin 6 - Motor +
The encoder has 334 lines.

Use the Raspberry Pi Pico to power the encoder, and check that the encoder A and B pins output quadrature square waves when you spin the motor. The Pico has the following pins:



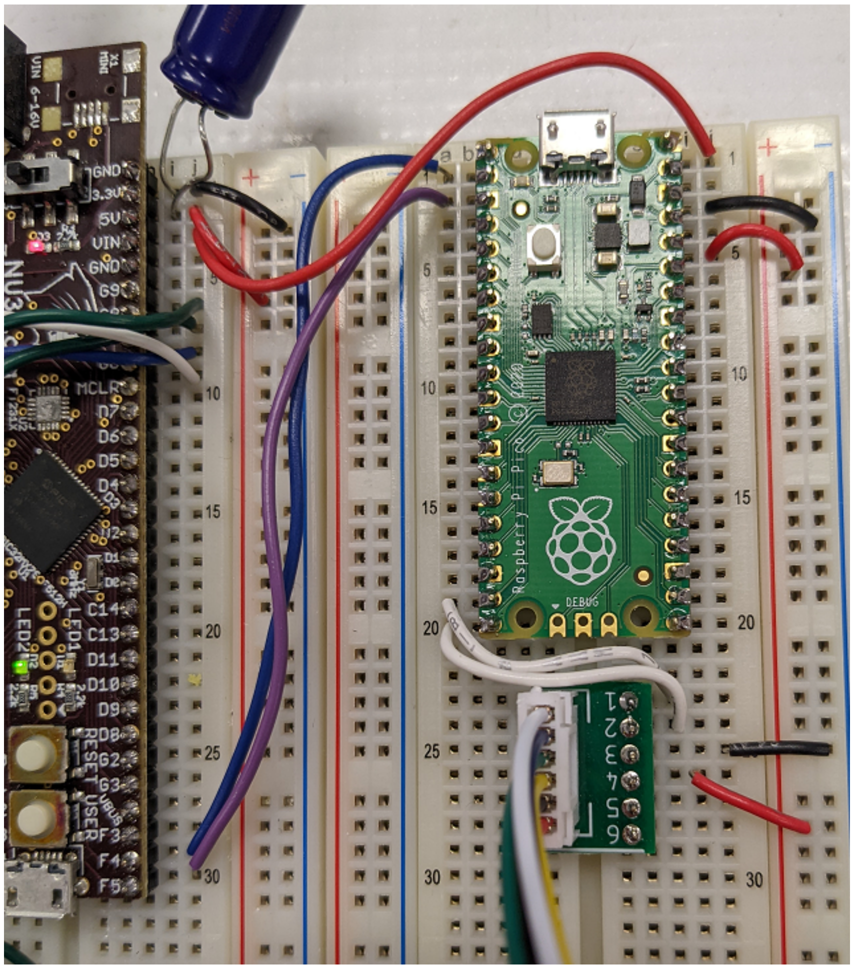
Put the Pico at the top of a breadboard, with the USB connector facing out. Connect GND and 3V3(OUT) to the breadboard rails, connect the encoder to power, and plug in the USB cable. Plug A and B into nScope Ch1 and Ch2, and verify that the encoder works.



Remove A and B from the nScope, and connect A to the Pico GP14 and B to GP15. Load pico_encoder_to_usb.uf2 by unplugging the Pico USB, and holding the BOOTSEL button while plugging the USB back in. The Pico will enumerate as a drive called RPI_RP2. Copy pico_encoder_to_usb.uf2 to the drive. The drive will disappear and the Pico will enumerate as a new serial port. Find the name of the port (on Windows, in Device Manager, will look like COM4; in OSX and linux, ls /dev/tty.*, will look like /dev/tty.usbmodem###). Open the Port (on Windows, open Putty, select the serial radio button, type in the port name and click open; on OSX and linux, type screen and the port name). The Pico will print the encoder position in quadrature encoder ticks (334*4 ticks per revolution) at 10Hz.

When you have verified that the encoder works, load pico_encoder_to_serial.uf2 on the Pico. With this code, the Pico will not make a USB port. Instead the Pico will respond to a UART command of ‘a’ by replying with the encoder position. A command of ‘b’ will set the position to 0, but will not reply. The Pico UART uses a baud of 230400 and pins UART0 TX/GP0 and UART0 RX/GP1.

Connect the Pico TX to the NU32 U2RX/F4 pin and the Pico RX to the NU32 U2TX/F5 pin. Connect the NU32 Ground to the Pico Ground. Disconnect the Pico USB and connect the NU32 5V to the Pico VBUS to power the Pico.



On the NU32, use encoder.c and encoder.h to talk to the encoder.
UART2_Startup() initializes UART2 on the NU32, with a baud of 230400 and an interrupt is generated when a character is received. Characters can be sent using WriteUART2(). You can request a position from the Pico by using WriteUART2(“a”). You can set the position to zero by using WriteUART2(“b”).
When the Pico sends a position, an interrupt will be generated for each character, and the character is stored in rx_message. When the ‘\n’ newline character is received, the position is sscanf’d out of rx_message and put in the variable pos and the newPosFlag is set to 1.
So when you want to get the motor position,

```
WriteUART2(“a”); // ask for a position
while(!get_encoder_flag()){} // wait for the new position while the flag is 0
set_encoder_flag(0); // remember to clear the flag so it works again next time
int pos = get_encoder_count(); // get the position to do whatever with it
```

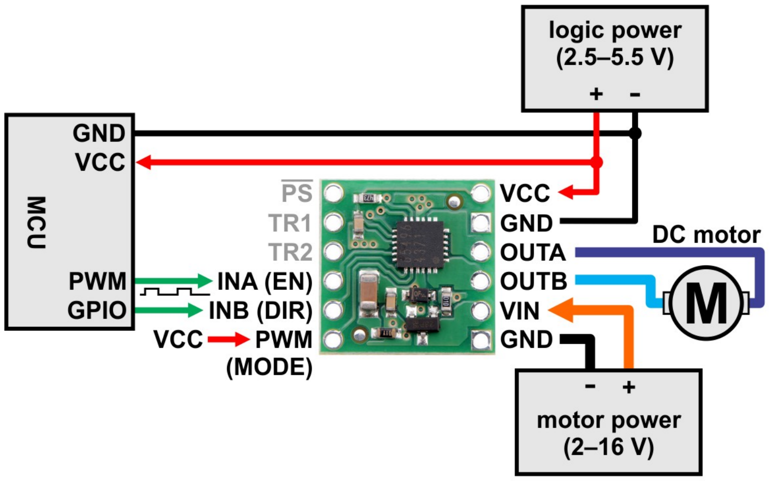
Reading the current sensor

The current sensor is a breakout board from Aliexpress. The circuit has a 0.1 ohm current sense resistor built onto the circuit board. The motor has a resistance of about 6 ohms, and the battery has a voltage of about 6V, so the stall current should be about 1A. (The actual stall current will be less, because the battery has 1 ohm output impedance, I get about 700mA).
The breakout board uses a INA219B I2C op amp. It uses 3.3V for VCC and Ground for GND. The data pins are SDA and SCL, connect SDA to the NU32 SDA1/D9 pin and SCL to the NU32 SCL1/D10 pin. The I2C pins need pull-up resistors to work, these are built into the breakout board.
The sensed current flows from Vin+ to Vin-. The current is returned as a signed 10 bit integer. Use ina219.c and ina219.h to access the INA219 sensor, which require the use of i2c_master_noint.c and i2c_master_noint.h. You can download all 4 files from Canvas.

To use the INA219, call INA219_Startup() after NU32_Startup(), and when you want to read the current in milliamps, use INA219_read_current(), which returns a float (converted to milliamps from the signed 10 bit number). It is not necessary to calibrate the sensor, it is pre calibrated! So 28.4.7 and 28.4.8 become much simpler.

Using the h-bridge

The h-bridge is the BD65496MUV breakout board from Pololu.
VCC is the logic power, use 3.3V from the NU32 or the Pico. Connect both GND to ground.
The battery goes from VIN to GND, and must also be connected to the NU32 ground. Be careful with the battery, it is easy to plug it in backwards and fry the h-bridge or NU32, or it can short in your bag. Always remove one of the AAs from the pack when not using.
Connect the PWM(MODE) pin to 3.3V, so that the chip interprets INA as a speed command and INB as a direction.



Pick a PWM pin from the NU32 for INA (like D0), and any regular digital output for INB (like D1).
At this point you can follow along with the book for 28.4.9.
Note that the current sensor must be placed in series with the motor, so the connections are something like:
OUTA to M+, M- to Vin+, Vin- to OUTB

Other tips

- If you want to use an enum (28.4.6 Operating Mode) it looks like:

```
enum mode_t {IDLE, PWM, ITEST, HOLD, TRACK}; // the definition
enum mode_t mode; // the declaration
mode = IDLE; // how to set it
if (mode == ITEST){...} // how to compare it
```

- The matlab/python function read_plot_matrix expects you to first print how many data points you want to send, then the data points in columns. For example, the NU32 would print:
    - 3\n
    - 1 2\n
    - 3 4\n
    - 5 6\n
        - To send 3 data points from 2 arrays
-