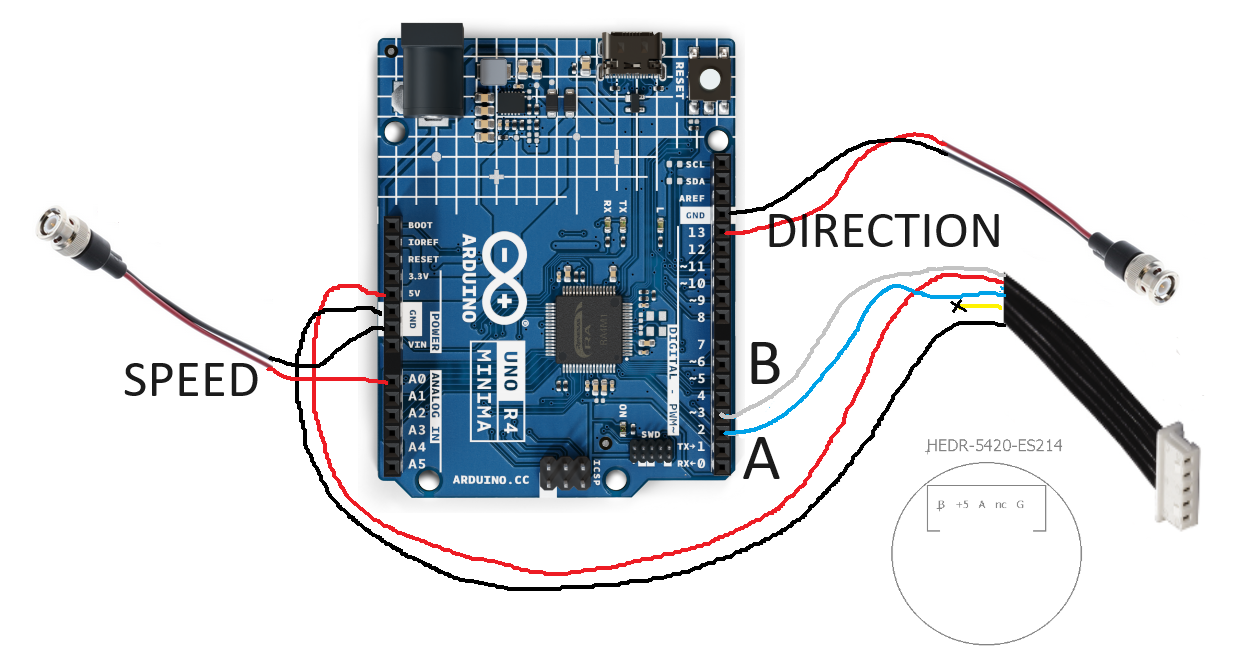
Encoder Interface



Steve Sawtelle  
Engineer, jET

Contents

[System Overview 2](#_Toc160118063)

[Hardware Development 2](#_Toc160118064)

[Assembly 2](#_Toc160118065)

[Firmware and Calibration 4](#_Toc160118066)

# System Overview

The Encoder Interface provides a method to connect a quadrature encoder to serial and analog interfaces. The emphasis is on simplicity and ease of assembly. The interface reads the encoder and calculates speed and direction based on time between encoder edge changes using the internal timer on the processor. Speed is updated every time it changes. If no changes are detected over a set time (typically 50 milliseconds) the speed is set to 0. Distance is also accumulated. The USB-serial connection prints the distance and speed every change. The analog output sets a voltage proportional to forward speed (as it is a unipolar output). The calibration of speed, distance, and output voltage can all be customized.

# Hardware Development

A top priority of this design is ease of assembly and programming. No circuit boards need to be made and only basic soldering skills are required. The processor boards are Arduino compatible.

A variety of processor boards are available that can be programmed under Arduino. There is a much smaller subset that is optimal to use for this interface. The requirements of the interface are:

1. 5-volt tolerant inputs, as the encoder is a 5-volt device.
2. Interrupt capability on the inputs.
3. Analog output for speed.
4. Digital-to-Analog Converter (DAC) output for direction.
5. Built-in USB for programming and serial output.

The simplest board to use is probably the Arduino R4 Minima. Other boards could also work, keeping in mind the constraints listed above.

# Assembly

If analog outputs are not needed the BNC cable assemblies can be ignored. Be sure to insulate wires and connections so that no shorts occur. Note: the encoder connector pinout is:

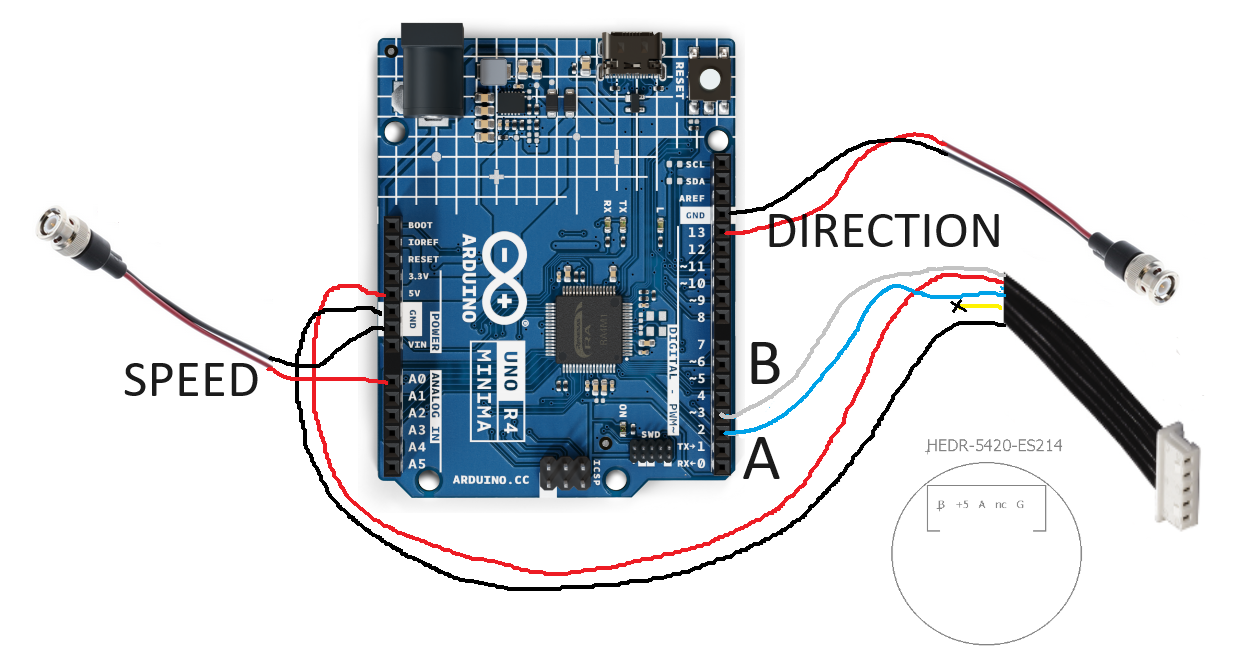
A close-up of a plug

Description automatically generated

**Arduino R4 Minima**

|  |  |  |  |
| --- | --- | --- | --- |
| Quantity | Description | Manufacturer | Part Number |
| 1 | Processor | Arduino | UNO R4 MINIMA |
| 2 | BNC connector | Pomonoa | 5069 |
| 1 | Cable assembly | Molex | 151340503 |
| 1 set | Pin jumper wire assortment | Pololu | 3965 |
| AR | Heat-shrink tubing |  |  |

The encoder cable assembly has connectors at both ends, so it can be cut in half. The connections to the board are made by soldering each BNC and Encoder cable end to a pin jumper (which have also been cut in half) and covering the connection with heat-shrink tubing. Hot melt glue can be used to help hold the pin connectors in place. The different colored wires shown are simply to make it easy to trace each wire.



# Firmware and Calibration

The firmware was developed under the Arduino environment. The Arduino R4 Board may need to be added to the Board Library.

Various values may need to be modified to suit a particular treadmill setup. The calibration can be checked by moving the treadmill belt a set distance and calculating the difference in the start and end points sent over the serial port. The analog output speed range also be adjusted.

MAXSPEED - the speed (mm/sec) that will cause the maximum voltage output (assuming 0 mm/sec is 0 volts).

SHOW\_MICROS – if defined, then the running micros value will be output to the serial port along when new data is sent.

SHOW\_REVERSE – if defined, then reverse running will subtract distance (and calculate negative speed).

SPEED\_TIMEOUT – Zero speed will be assumed if no encoder interrupts occur within this many microseconds.

SHOW\_ZERO – if defined, will output 0 speed after SPEED\_TIMEOUT microseconds.

CYCLINDER\_8IN – for the 8-inch cylindrical treadmill, extra interrupts are used to get additional resolution in the speed and distance since the treadmill moves further for each encoder count.