CSC 615 Assignment 4 – Motors & Speed Encoder

This is an INDIVIDUAL assignment. You can (and should) work in groups to research how to do the assignment, but each person should code their own version and make their own submission.

This is a physical class, so I will want to see what you do in action. Documentation, including short video clips (can use your cell phone) are required as part of the submission. It can be by the Hardware Manager of recorded via the WebCam.

You will also need to submit hardware drawings. These should be neat (can be either electronic or hand drawn, then scanned) of how the hardware is connected to the Raspberry Pi. This includes which pin (physical and GPIO), positive and negative flow, resisters, etc. I, and the Hardware Manager, should be able to rebuild your setup from this diagram and then run your program and get the same results. Also see https://www.circuit-diagram.org/editor/# if you want to use that (they have a Raspberry Pi template.

Assignment Description

1. Read the following pages "Component Knowledge".

2. This is the second of a two-part assignment. In this, the second part, you will use the wheel encoders to validate the speed that each wheel is running, in cm/sec. The purpose is to allow proper speed control in real world conditions where the same power may not yield consistent speed results.

3. You are to connect a motor to one of the channels on the Motor Shield (or other board). See the motor shield specifications to determine the pins and the pin numbers. The motor will have a speed encoder wheel and an infrared speed sensor.

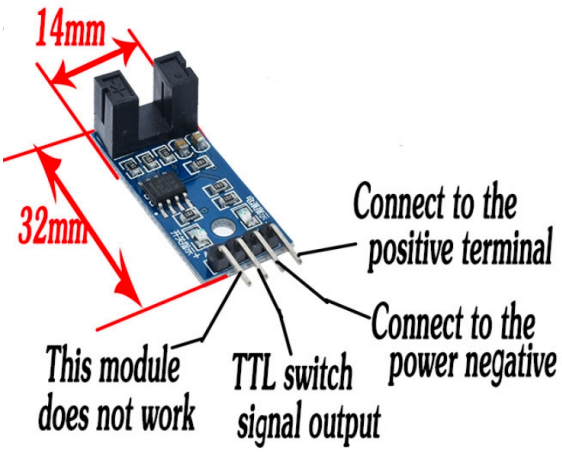
4. You will try to read the speed encoder sensor by directly connecting the sensor to a digital pin on the Raspberry Pi and creating a multi-threaded application where the thread is responsible for reading that pin and providing a speed to the main program.

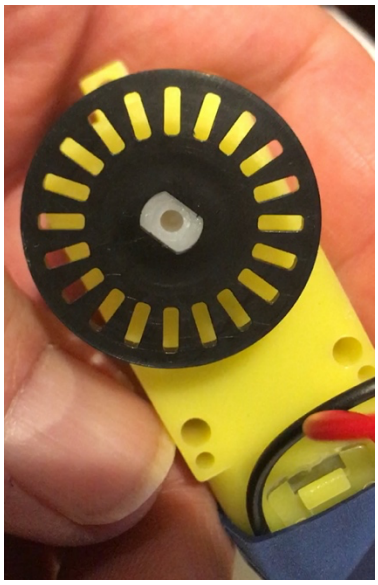
5. You must have console output that displays the power (PWM) being applied to each motor and the actual speed of that wheel as measured by the speed encoder. Record the output for the submission.

6. Submit your homework in your GitHub and on iLearn per the submission details below

Component Knowledge

Speed Sensor and Encoder Wheel





High Quality 4 PIN Infrared Speed Sensor Module

Module Description

1. Working voltage: 3.3--5.5V
2. The use of imported groove coupler sensor, the groove width of 5mm.
3. The sensor slots are blocked, the output high and the light is off; no shelter, output low and lights.
4. Output form: Digital switching output (0 and 1)
5. Using LM393 comparator output, type clean, good waveform, driving ability over 15mA.
6. A fixed bolt hole for easy mounting.

Module interface specification (3-wire)

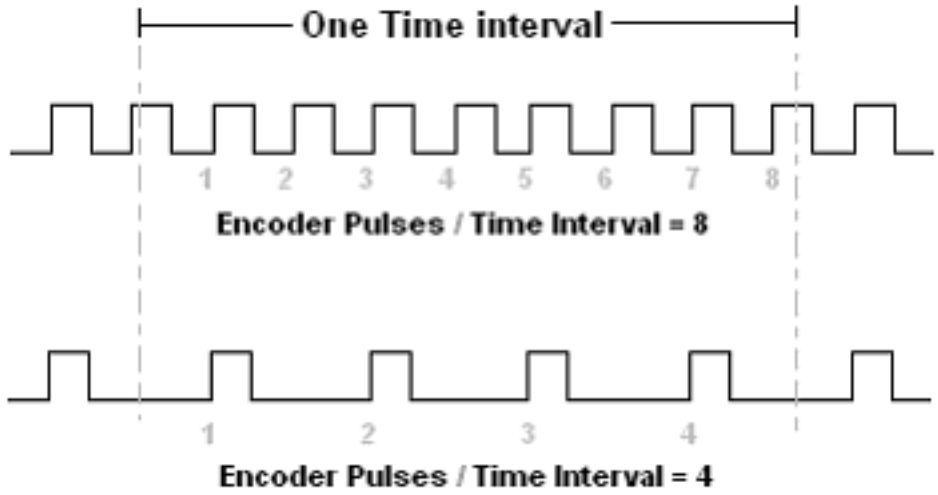
1 VCC 3.3V-5V external
2 GND External GND
3 DO small board digital output interface switch connected to the microcontroller IO port
4.AO: Invalid

Pulse counting

Pulse counting uses a sampling period (t) and the number of pulses (n) that are counted over the sampling period to determine the average time for one pulse (t/n). Knowing the number of pulses per revolution (N) for the encoder, the speed can be calculated.

$$\omega = 2\pi n/Nt$$

Where:
ω = angular speed (rad/s)
n = number of pulses
t = sampling period (s)
N = pulses per rotation



At low speeds, the resolution of pulse counting is poor, so this method is best applied in high speed applications.

Pulse timing

With the pulse timing method, a high-frequency clock signal is counted during one encoder period (the pitch, or interval between two adjacent lines or windows). The number of cycles of the clock signal (m), divided by the clock frequency (f), gives the time for the encoder period (the time for the encoder to rotate through one pitch). If the encoder PPR is denoted by N, the angular speed of the encoder is given by:

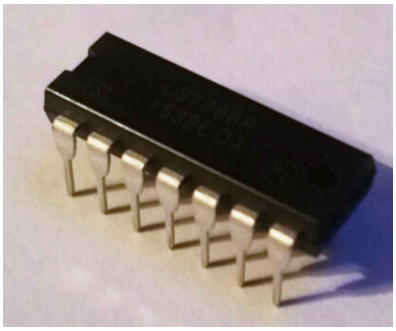
$$\omega = 2\pi f/Nm$$

Where:
ω = angular speed (rad/s)
f = clock frequency (Hz)
m = number of clock cycles
N = pulses per rotation

At high speeds, there may be too little time between pulses for pulse timing (also referred to as pulse frequency) to accurately measure clock cycles, so this method is best for low speed applications.

LSI LS7366R Incremental Encoder (for extra reference – Hardware Mangers Only)

LS7366R is a 32-bit CMOS counter, with direct interface for quadrature clocks from incremental encoders.
For communications with microprocessors or microcontrollers, it provides a 4-wire SPI/MICROWIRE bus.The four standard bus I/Os are SS/, SCK, MISO and MOSI. The data transfer between a microcontroller and a slave LS7366R is synchronous. The synchronization is done by the SCK clocks supplied by the microcontroller. Each transmission is organized in blocks of 1 to 5 bytes of data.



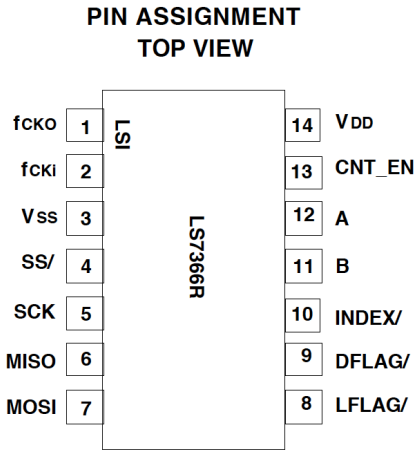


FIGURE 1

You will use the Pi's SPI interface to communicate with this chip which you will use to read the count from the speed sensor module.

Read the rest of the detailed specifications on this LS7366 located under Technical Documents on iLearn.

Submission Details

You need to submit the following files in your github repository:

1. All .c and .h source code files.

2. A makefile file to build your program (the file MUST be called **`makefile`**). The executable output files MUST be called **`assignment4`**.

3. A PDF that is clear and readable with your hardware diagram (make sure to indicate polarity and pin numbers).

4. A PDF showing the console output that has the power and actual speed for the motor. Also a short writeup with the issues and resolution and perceived accuracy. (combine the hardware PDF with this PDF and submit as one document in your github and in iLearn)

5. A mp4 file showing your motor in action (show both the motor and the motor shield) along with the console display.

All parts of the submissions must have your name and student ID number. For Video's please have at least a 2 second clip at the beginning with your Student ID card clearly visible. (In absence of **your student ID card print out your Name and ID number on paper and film that**).

Please post questions to the slack channel.

Grading Criteria

Grading criteria will be based on the following:

Completion and success of the assignment	25%
Code well structured, original and well documented	50%
Hardware Diagram	15%
Video	10%

Instructions followed (this includes submission requirements) This is only a detractor from your grade, i.e. failure to follow the instructions will result in a reduction from the grade calculated from the criteria above.	-50%