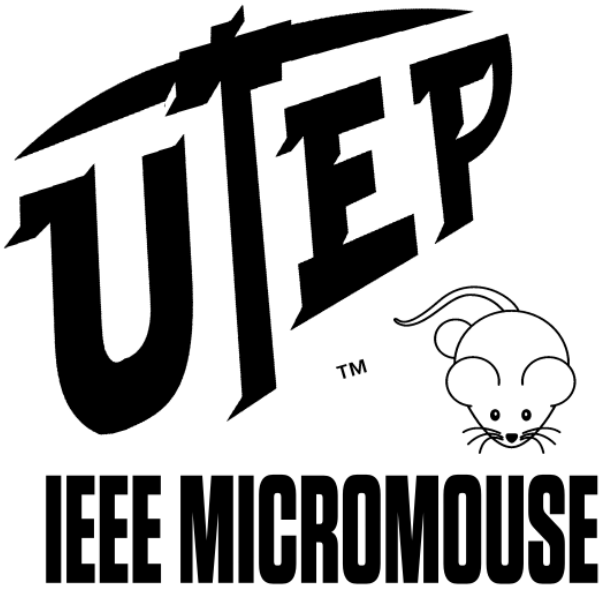


IEEE MICROMOUSE

Workshop Week 3:

Encoders and Arduino Libraries

Objective:

Learn how an encoder works and how to use it to go from angular translation to linear translation. Learn how to download and use new Arduino libraries.

Background Information:

One of the things that the mouse must do is move a specified distance accurately. From last week, you know that you can control the speed in which the mouse moves, but you cannot tell how far you moved on a given time interval. To know this, you must use some type of sensor, and for the micromouse this sensor is an encoder. So, what is an encoder? An encoder is a sensor that triggers voltage pulses at specific angle intervals. The encoder does this by using a photo interrupter or a hall effect sensor. Figures 1 and 2 show how this works on a mechanical level. But how does this translate to motion and direction? Well to know how much the wheel has rotated, you only need to count the number of pulses. For the direction, you need to know which channel signal is leading and which is lagging. In figure 1 and 2 you can see that the signals are always out of phase, or misaligned. On figure 3 you can observe how this misalignment identify the direction in which wheel encoder is moving by reading the value of the opposite signal channel on the encoder.

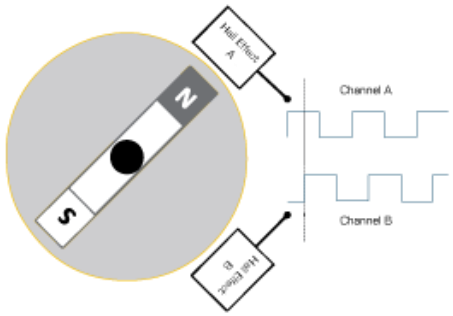


Figure 1 Hall effect encoder

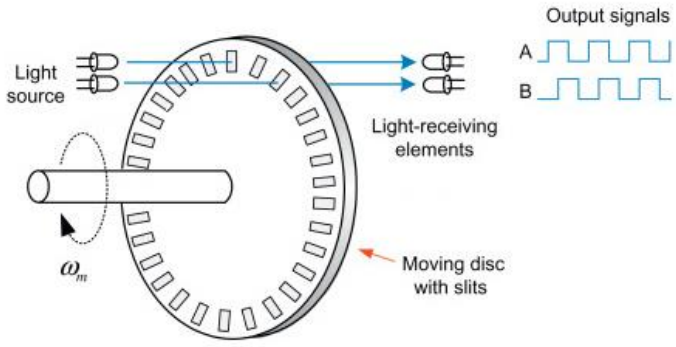


Figure 2 Photo interrupter encoder

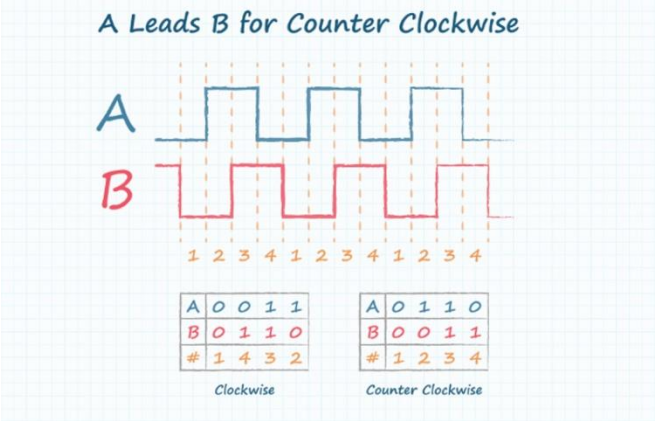
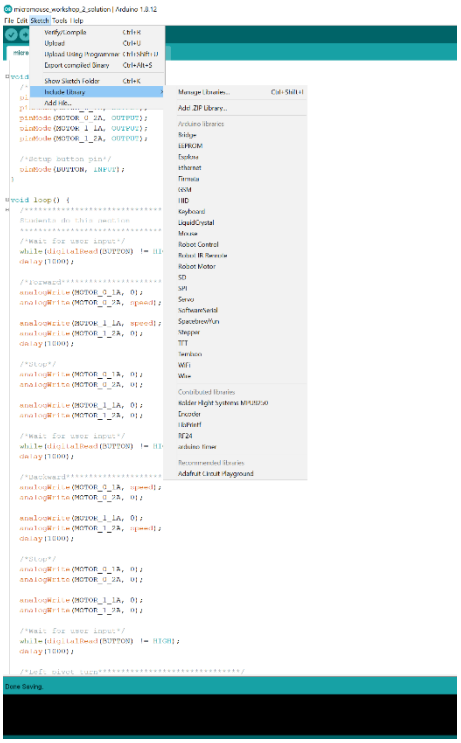


Figure 3 Pulse order for clockwise and counterclockwise encoder signals

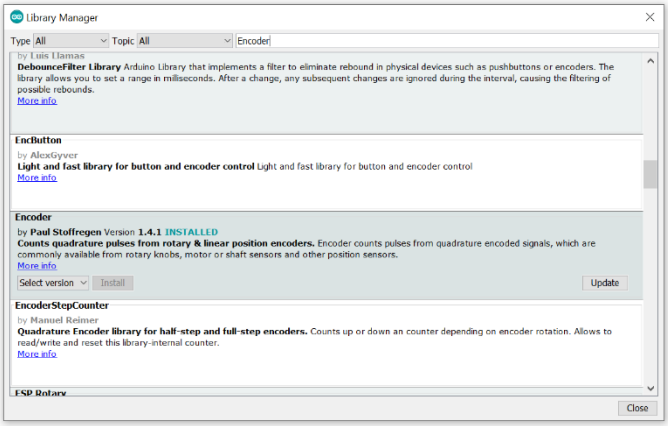
Downloading and using Arduino Libraries

One of the reasons why Arduino is so popular is because of the number of libraries that it has. If you have a gadget or device that you want to control, the Arduino community most likely has a library for it. For the micromouse we will use an encoder library to setup and read the value from both motor encoders. The steps to download the encoder library are:

1. In an Arduino sketch go to *Sketch/Include Libraries/Manage Libraries* or press CTRL+SHIFT+I



2. On the new window, on the search bar type *Encoder* and look for the Encoder library by Paul Stofreggen



3. Click on the install button. Let it download, once it's finished you can start using the library on all your micromouse sketches

To use this library, open a new micromouse Arduino sketch (sketch on last week's email) and write the following code:

```
/*This micromouse workshop will teach you:
 * 1. Downloading Arduino libraries
 * 2. Encoders for position and speed
 */

#include "micromouse.h"
#include <Encoder.h>

Encoder right_encoder(ENCODER_0_CHA, ENCODER_0_CHB);
Encoder left_encoder(ENCODER_1_CHB, ENCODER_1_CHA);

void setup()
{

}

void loop()
{

}

```

Compile and verify that your code has no errors. If Arduino does not throw any compilation errors, then you are set! You have now downloaded and used your first Arduino library, wasn't that easy?

Using encoders for linear displacement

Now that we can use the encoder library, let's use it to measure the linear displacement of the mouse. The first thing that we need to address is the relationship between angular and linear displacement. Remember that the encoder only measures angular displacement in the form of pulse counts. The second thing to consider is how to relate pulse count to angular displacement. In the end, we want to end up with a relationship that takes as an input the distance we want to travel in millimeters, and then output the encoder count we must have to travel that distance.

Linear Displacement → Angular Displacement → Encoder Pulse count

The relationship between the angular and linear displacement is given by the following equation

$$x = 2\pi r$$

Where x is the total distance travelled and r is the radius of the wheel. The radius of the wheel is $r = 21mm$. For example, in one full rotation of the wheel, we will displace the mouse a distance of $x = 2(21mm)\pi = 131.94mm$.

Now we need to relate the encoder pulses to angular displacement. To do this, we first need to know the pulse count in one wheel revolution. For the motors we are using this value is 800 counts/rev.

We can use this fact to come up with the following equation:

$$c(x) = (\frac{x}{131.94})(800)$$

Where c is the encoder pulse count and x is the desired distance for the mouse. For example, on the micromouse competition it is required that the mouse moves in "steps" of 180 millimeters, using the above equation, we can know how many encoders pulses we need to count until we reach the desired 180-millimeter distance $c(180) = \left(\frac{180}{131.94}\right)(800) = 1091.4 \simeq 1091\ pulses$

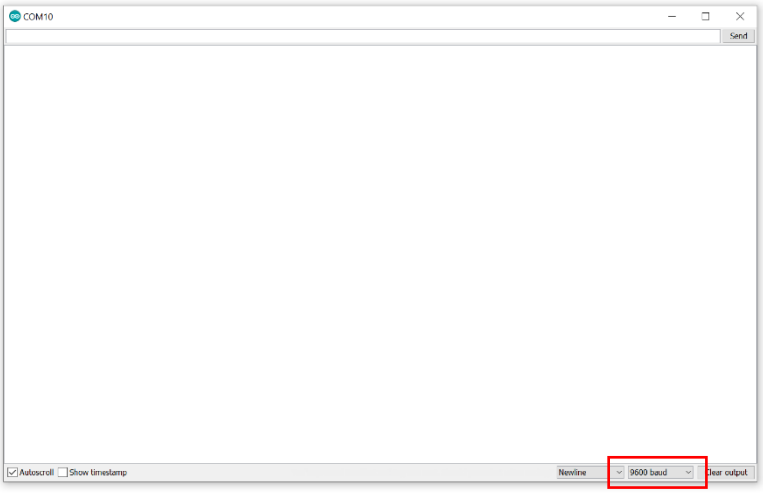
Last thing to consider is that we can use both encoders and average their values to measure distance. This will help smoothing any fluctuations we might have on one of the motors at any given time.

Enough theory! Let's write some code!

```
1.  /*This micromouse workshop will teach you:
2.   * 1. Downloading Arduino libraries
3.   * 2. Encoders for position and speed
4.   */
5.  #include "micromouse.h"
6.  #include <Encoder.h>
7.
8.  Encoder right_encoder(ENCODER_0_CHA, ENCODER_0_CHB);
9.  Encoder left_encoder(ENCODER_1_CHB, ENCODER_1_CHA);
10.
11. const int speed = 64;
12.
13. void setup()
14. {
15.   /*Setup motor pins*/
16.   pinMode(ENABLE, OUTPUT);
17.   pinMode(MOTOR_0_1A, OUTPUT);
18.   pinMode(MOTOR_0_2A, OUTPUT);
19.   pinMode(MOTOR_1_1A, OUTPUT);
20.   pinMode(MOTOR_1_2A, OUTPUT);
21.
22.   pinMode(BUTTON, INPUT);
23.   Serial.begin(9600);
24. }
25.
26. void loop()
27. {
28.
29.   /*Move motor forward at given speed*/
30.
31.   /*Forward distance to travel 1 cell is 180mm
32.    * 1 rev = 720 encoder counts, so (counts/rev) = 800.
33.    * Linear distance travelled on 1 wheel revolution is given
34.    * by x = 2*r*pi
35.    * where  r = wheel radius
36.    *        x = linear distance
37.    *
38.    * so x = 2*21*pi = 131.946mm
39.    *
40.    * To generalized
41.    * (x/131.946) * 800 = c
42.    * where x = desired distance and c = pulse count
43.    * counts/rev = encoder counts in one wheel revolution
44.    * (180mm/131.946mm) * 800 = 1091.4 ~= 1091
45.    */
46.   long int right_encoder_count;
47.   long int left_encoder_count;
48.   long int average_encoder_count = 0;
49.
50.   right_encoder_count = right_encoder.read();
51.   left_encoder_count = -left_encoder.read(); //This should be negative! This encoder drives
    the opposite way
52.   average_encoder_count = (right_encoder_count + left_encoder_count)/2;
53.
54.   /*print values to serial console*/
55.   Serial.print(right_encoder_count);
56.   Serial.print(“, ”);
57.   Serial.print(left_encoder_count);
58.   Serial.print(“, ”);
59.   Serial.println(average_encoder_count);
60.
61.   /*Stop motors*/
62.
63. }
```

To see the values of the left, right and average encoder count, open the Serial Monitor on Arduino

And select 9600 for the baud rate



After connecting and uploading the code to your Arduino, you will see data streaming to the console giving you the value of the left, right and average encoder count values. In future workshops we will discuss more in detail the serial console and how to use it in Arduino sketches, for now all you need to know is that the values will be printed to this console. Try to verify that one revolution matches the value of 800 count/rev. For the wheels of your mouse, this might be different although we don't think it will be by much.

Activity

Now is your turn to work on the micromouse. Using the information on encoders and the template code below, have your micromouse traverse 180mm forward, stop and then reset the encoder value back to zero. Try and measure the traveled distance of the mouse using a ruler or a measured piece of tape.

```
1.  /*This micromouse workshop will teach you:
2.   * 1. Downloading Arduino libraries
3.   * 2. Encoders for position and speed
4.   */
5.
6.  #include "micromouse.h"
7.  #include <Encoder.h>
8.
9.  Encoder right_encoder(ENCODER_0_CHA, ENCODER_0_CHB);
10. Encoder left_encoder(ENCODER_1_CHB, ENCODER_1_CHA);
11.
12. const int speed = 64;
13.
14. void setup()
15. {
16.   /*Setup motor pins*/
17.   pinMode(ENABLE, OUTPUT);
18.   pinMode(MOTOR_0_1A, OUTPUT);
19.   pinMode(MOTOR_0_2A, OUTPUT);
20.   pinMode(MOTOR_1_1A, OUTPUT);
21.   pinMode(MOTOR_1_2A, OUTPUT);
22.
23.   pinMode(BUTTON, INPUT);
24. }
25.
26. void loop()
27. {
28.   /*Wait for user input*/
29.   while(digitalRead(BUTTON) != HIGH);
30.   delay(1000);
31.
32.   /*Move both motors forward at given speed*/
33.   analogWrite(MOTOR_0_1A, 0);
34.   analogWrite(MOTOR_0_2A, speed);
35.
36.   analogWrite(MOTOR_1_1A, speed);
37.   analogWrite(MOTOR_1_2A, 0);
38.
39.   long int right_encoder_count;
40.   long int left_encoder_count;
41.   long int average_encoder_count = 0;
42.
43.   /*while distance != 180mm, keep measuring encoders and running the motors*/
44.   right_encoder_count = right_encoder.read();
45.   left_encoder_count = -left_encoder.read();
46.   average_encoder_count = (right_encoder_count + left_encoder_count)/2;
47.   delay(20);
48.
49.   /*Stop motors after you reach your destination*/
50.
51.   /*Reset encoders*/
52. }
53.
```

Extra Activity

Using the code above, have your mouse move forward 180mm and then go back to its original position. This is going to require for you to reset the encoders