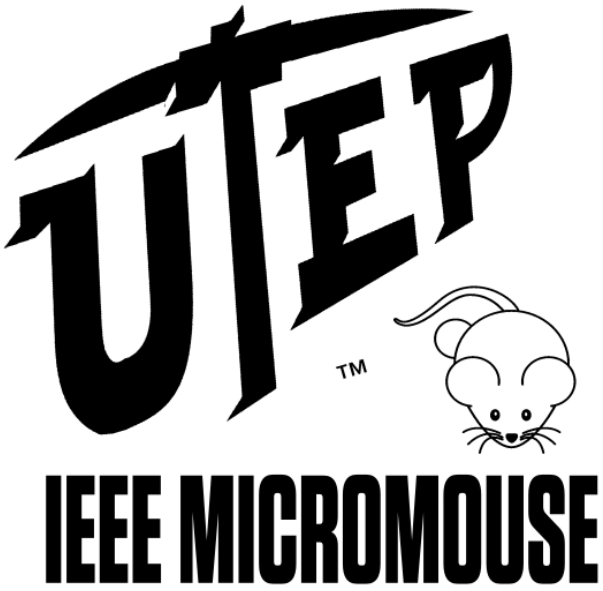


IEEE MICROMOUSE

Workshop Week 6:

Gyroscope and micromouse turns

Objective:

Learn what is a gyroscope and how to use it to measure angular displacement on the micromouse. Use this knowledge to accurately turn the micromouse to a desired angle. Learn about the different turn algorithms for micromouse.

Background information:

On previous workshops, you learned how to control the distance traveled by your micromouse using magnetic encoders. While this will get you far on the maze, the mouse still needs to perform turns at junctions and dead ends. To do this accurately, we need to continuously measure the angular displacement of the mouse on its z-axis. To measure this angle, we need to introduce a new sensor on your micromouse kit: the gyroscope.

A gyroscope is a Micro Electromechanical System (MEMS) that measures the angular rate on a body. A 3-axis gyroscope can measure the angular rate on all 3 perpendicular axis X, Y and Z. On the micromouse kit, you will find the MPU6050 3 axis accelerometer and gyroscope Inertial Measurement Unit (IMU). This device can measure angular rates and accelerations on all 3 axes. For the micromouse project, we will only use the Z-axis output measurement from the gyroscope.

As previously stated, the gyroscope can only measure angular rates, which means that we get the angular speed in degrees per second. Angular rate data can be useful, but we are more interested on getting the angular displacement of the mouse. To achieve this, we are going to have to numerically integrate the angular speed measurement coming from the z-axis gyroscope, more details on this later.

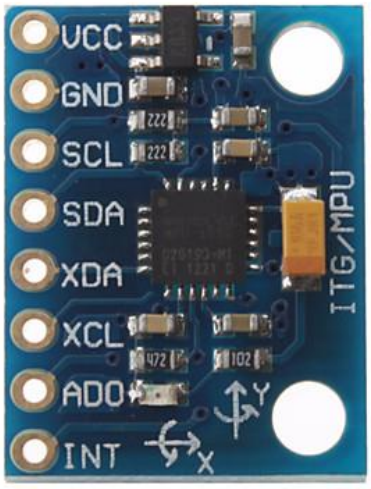


Figure 1 MPU6050 Inertial Measurement Unit

Another important concept is the different ways in which the mouse can perform a turn. The micromouse is part of a family of robotics that are known as differential drive robots. An important characteristic of a differential drive robot is its ability to control each motor independently from each other. Having this ability allows us to perform different turn maneuvers. The image below shows the different options when rotating a differential drive robot. You can see that we have 3 options, the first option is to drive both wheels at different speeds, one lower than the other. The second option is to only move one motor while the other is stopped, this will create a tighter turn profile around the center of the mouse. The third and last option is to drive both wheels at the same speed at opposite directions. This type of turn is very easy to understand, and it is helpful when turning in tight spaces.

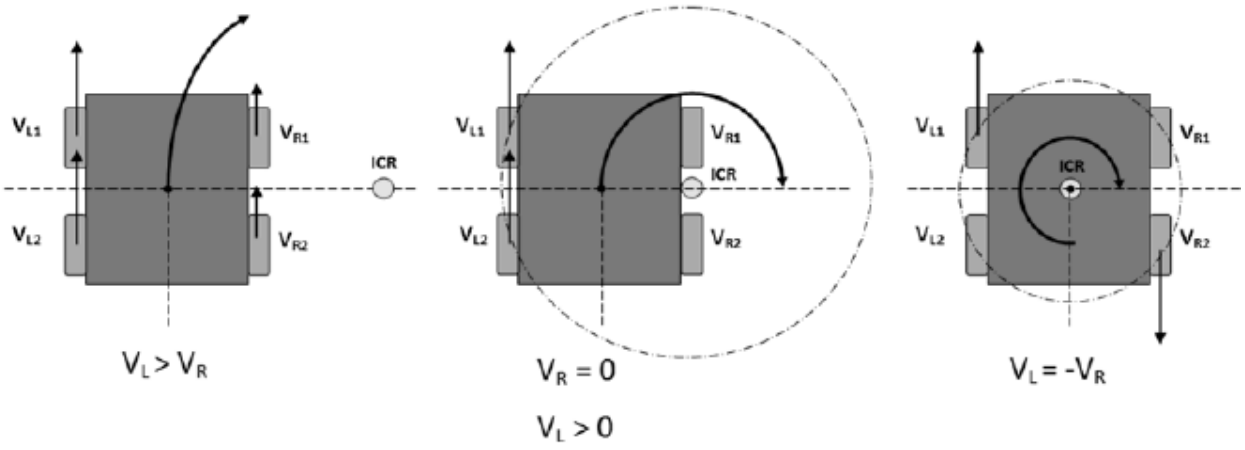
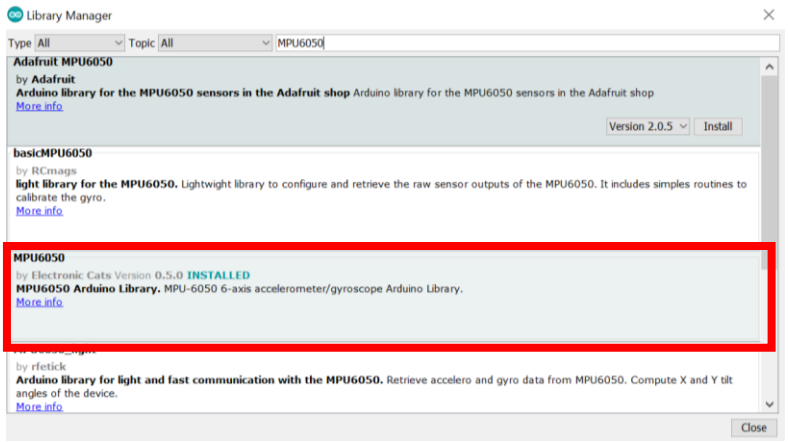


Figure 2 Turn profiles for a differential drive robot

The MPU6050 gyroscope

We first need to download a new library to start using the MPU6050. If you don't remember how to do this, please go back to workshop 3 and read the section on how to download new Arduino libraries. You will need to download the MPU6050 library by Electronic Cats



Once you have the library installed, write the following code to measure the angular rate on about the z-axis of the mouse

```
1.  #include "micromouse.h"
2.  #include "MPU6050.h"
3.
4.  MPU6050 gyro;
5.
6.  void setup()
7.  {
8.    Serial.begin(9600);
9.
10.   gyro.initialize();
11.   gyro.CalibrateGyro();
12. }
13.
14. void loop()
15. {
16.   int16_t raw_yaw_rate = gyro.getRotationZ();
17.   float yaw_rate = (float)raw_yaw_rate / 131.0;
18.
19.   Serial.println(yaw_rate);
20. }
21.
22.
```

Open the serial console and you will see the angular rate being displayed in degrees per second. On line 17, you see this line of code

```
1.  float yaw_rate = (float)raw_yaw_rate / 131.0;
```

This line of code may be confusing if you are new to using the MPU6050. The division by *131.0* is required to obtain the proper scale for the gyroscope measurement. On the MPU6050 datasheet, you will find the different scales for the raw measurements. This determines the sensitivity of the gyroscope and full-scale range of the measurement. By default, the MPU6050 has a full-scale range of +-250 degrees per second, which is the first entry on the table below.

Gyroscope ADC Word Length			16	bits
Sensitivity Scale Factor	FS_SEL=0		131	LSB/(°/s)
	FS_SEL=1		65.5	LSB/(°/s)
	FS_SEL=2		32.8	LSB/(°/s)
	FS_SEL=3		16.4	LSB/(°/s)

Figure 3 MPU6050 scales for raw measurements

As a side note, on line 16 you will encounter this variable declaration

```
1.  int16_t raw_yaw_rate = gyro.getRotationZ();
```

The *int16_t* data type means that we are declaring a 16-bit integer, which is the resolution of the gyroscope.

Numerically integrating gyroscope angular speed

As previously explained, the gyroscope can only measure angular rates. This measurement is not useful in this state since we want to measure the angular displacement of the mouse. To get the angle, we are going to use a bit of calculus.

If you remember from a basic physics course, speed is equal to the distance measured divided by the amount of time it has passed

$$s = \frac{d}{\Delta t}$$

Speed is defined as the instantaneous velocity at time *t,* and it has no direction. Now using the previous equation, you can solve for the distance traveled *d.*

$$d = s * \Delta t$$

Again, this is the instantaneous displacement at time *t*, and it will only tell us about the displacement at that specific time interval. If we want to maintain the **TOTAL** displacement, then we must add all the previous displacements. Imagine that our velocity profile looks like the one below, to find the total displacement from this function, we need to slice it into equal time chunks multiply them by the speed (instantaneous velocity) then add that result to the previous sum.

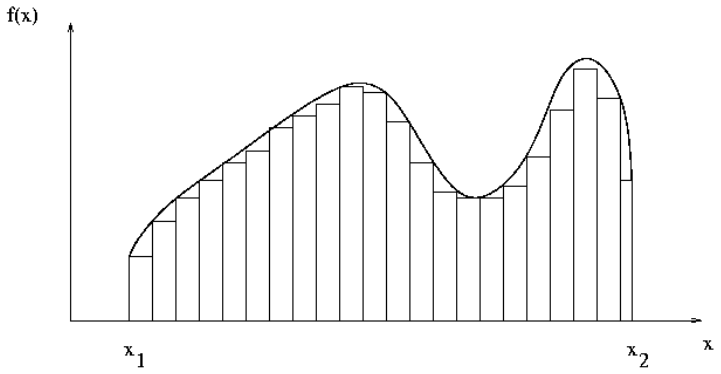


Figure 4 Numerical integration shown graphically. Imagine this function as the velocity profile of your mouse

With this knowledge we can write a simple numerical integration algorithm that uses the following 2 equations

$$s_0 = 0.0$$

$$s_{n+1} = s_{n-1} + s_n * \Delta t$$

This all applies to angles as well, just replace linear displacement by angular displacement

Transforming these equations into code looks like this

```
1.  #include "micromouse.h"
2.  #include "MPU6050.h"
3.
4.  MPU6050 gyro;
5.
6.  float yaw_angle = 0.0;
7.  float dt = 0.0;
8.  float t0;
9.
10. void setup()
11. {
12.   motors_init();
13.
14.   pinMode(BUTTON, INPUT);
15.   Serial.begin(9600);
16.
17.   gyro.initialize();
18.   gyro.CalibrateGyro();
19. }
20.
21. void loop()
22. {
23.   t0 = millis()/1000.0;
24.
25.   int16_t raw_yaw_rate = gyro.getRotationZ();
26.   float yaw_rate = (float)raw_yaw_rate / 131.0;
27.
28.   /*Numerical integration*/
29.   yaw_angle += yaw_rate * dt;
30.
31.   Serial.print("angle= ");
32.   Serial.print(yaw_angle);
33.   Serial.print(", ");
34.   Serial.print("dt= ");
35.   Serial.println(dt);
36.
37.   dt = millis()/1000.0 - t0;
38. }
39.
```

Open the serial console, and you will see the angle displacement of your micromouse. Try rotating your micromouse about the Z-axis.

Not everything is perfect with this approach and there are a few things to watch out for:

1. The error on the numerical integration will accumulate over time. After a few minutes the measurement will be unreliable
2. You might saturate the gyroscope measurement by turning the mouse too fast. If this happens the measurements will be incorrect.
3. The initial orientation of the mouse is assumed to be 0 degrees at startup. This obviously will not make much sense in some frame of references.

Accurate turning using a gyroscope

Now that you know how to use a gyroscope, lets use it to turn the micromouse 90 degrees. The code to do this looks like this

```
1.  #include "micromouse.h"
2.  #include "MPU6050.h"
3.
4.  MPU6050 gyro;
5.
6.  float yaw_angle = 0.0;
7.  float dt = 0.0;
8.  float t0;
9.
10. void setup()
11. {
12.   motors_init();
13.
14.   pinMode(BUTTON, INPUT);
15.   Serial.begin(9600);
16.
17.   gyro.initialize();
18.   gyro.CalibrateGyro();
19.
20.   /*Set motor speed at startup*/
21.   motor_0_speed(64, true);
22.   motor_1_speed(64, false);
23. }
24.
25. void loop()
26. {
27.   t0 = millis()/1000.0;
28.
29.   int16_t raw_yaw_rate = gyro.getRotationZ();
30.   float yaw_rate = (float)raw_yaw_rate / 131.0;
31.
32.   /*Numerical integration*/
33.   yaw_angle += yaw_rate * dt;
34.
35.   Serial.print("angle= ");
36.   Serial.print(yaw_angle);
37.   Serial.print(", ");
38.   Serial.print("dt= ");
39.   Serial.println(dt);
40.
41.   dt = millis()/1000.0 - t0;
42.
43.   if(yaw_angle > 90.0)
44.   {
45.     //stop motors
46.     motor_0_speed(0, true);
47.     motor_1_speed(0, true);
48.   }
49. }
50.
```

The angle might not be correct, so tweak it accordingly.

Exercise:

Use the previous code to turn the mouse -90 and 180 degrees.