UTEP™

IEEE MICROMOUSE

Workshop Week 9:

Turn control

Objective:

Control turn angles of the mouse in clockwise and counterclockwise rotations. Use the concepts from previous workshops to design a turn proportional controller. Finally, write and test the software for this week's workshop

Background information:

From previous workshops, we learned how to use proportional controllers to reach target distances from a maze cell and maze walls. This week we will design a turn controller for the mouse. The goal of this controller is to turn at accurate angles, so that we can precisely stop and turn when we find a wall in front of the mouse.

The same ideas form the previous controllers will be applied on this turn controller. First, we need to compute the error from a set angle and the actual angle. This is simply represented as:

$$Angle_{error} = Angle_{target} - Angle_{actual}$$

This error can be amplified and used as the output for the motors. The only difference this time is that both wheels must go in opposite directions, so the output of one motor will always be the negative of the other. The equations to achieve the motor outputs are as follows:

$$turn_{output} = kp_{turn} * Angle_{error}$$

$$Output_{left} = turn_{output}$$

$$Output_{right} = -turn_{output}$$

Our controller will correct for any disturbance, overshooting or undershooting our target angle. The turn profile we will use for this micromouse is the one shown in figure 1. Notice how the wheels run in opposite directions to create a tight turn.

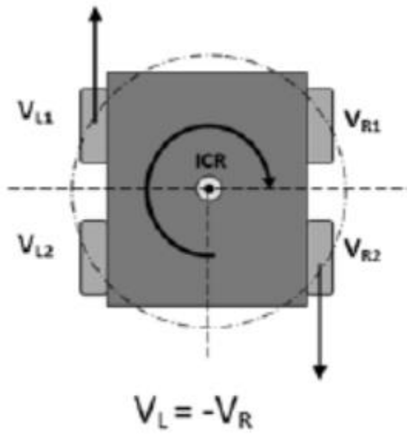

Figure 1 Turn profile for micromouse

Get gyro yaw angle

We are going to use the code from week 6 workshop to measure the yaw angle of the mouse. If you recall, using a gyroscope will allow us to measure the turn angle of the mouse by numerical integrating the raw output of the sensor. If you have not read this workshop, please do as most of the code will not be explained in here.

Let us enclose the gyroscope initialization and measurement into a couple of functions.

```
void gyro_init(void)
{
  gyro.initialize();
  gyro.CalibrateGyro();
}

float get_angle(float dt)
{
  int16_t raw_yaw_rate = gyro.getRotationZ();
  float yaw_rate = (float)raw_yaw_rate / 131.0;
  float yaw_angle = yaw_rate * dt;

  return yaw_angle;
}

```

Add these functions to your micromouse.cpp and micromouse files.

Main turn control loop

The control loop for the turn controller is vastly like the previous controller loops we designed. Again, the only differences on this controller are the sensor source and the motor output calculations. The code for this controller is as follows:

```
const float MIN_TURN_ERROR = 1.0;
const float turn_kp = 30.0;

void control_turn(float target_angle)
{
   /*Keep running the control loop until the error is less that the
   * minimum specified error (MIN_TURN_ERROR)
   */
  float yaw_angle = 0.0;
  float dt = 0.0;
  float t0;
  float turn_error;
  do
  {
    t0 = millis() / 1000.0;

    /*Read gyro yaw angle, remember that this is a measurement for
     * this time period so we need to sum the previous angle measurements
     */
    yaw_angle += get_angle(dt);

    /*Compute turn error*/
    turn_error = target_angle - yaw_angle;

    /*Proportional error for turn. Clamp output value*/
    float turn_output = clamp(turn_kp*turn_error, 80.0, -80.0);

    /*Calculate final output speed for each wheel*/
    int left_output = (int)(-turn_output);
    int right_output = (int)(turn_output);

    /*Find the turn direction of each wheel. Check the find_direction function*/
    bool left_dir = find_direction(left_output);
    bool right_dir = find_direction(right_output);

    /*Set speed and direction to motors*/
    motor_0_speed(abs(right_output), right_dir);
    motor_1_speed(abs(left_output), left_dir);

    /*delay the control loop and calculate delta time*/
    delay(50);
    dt = millis()/1000.0 - t0;
  }while(fabs(turn_error) > MIN_TURN_ERROR);

  /*Stop both motors*/
  motor_0_speed(0, true);
  motor_1_speed(0, true);
}

```

A few things to notice:

1. Turn speed affects the performance of the controller by overshooting or undershooting the set angle
2. The delay time also plays an important role on the controller. Making the delay shorter will increase the gyroscope drift
3. As in last week workshop, the proportional coefficient amplifies the velocity response to the error of the turn angle
4. The micromouse will always overshoot the target position. We don't know exactly why that is, but our best guess is that the gyroscope is not positioned exactly at the center of the mouse.

See that we also use the clamp() and find_direction() functions from workshop 8. If you do not have those, please go and see workshop 8. Add this control function to your micromouse.cpp and micromouse.h files.

Using the turn control on your sketch

Now you can use the code on your main sketch to turn the mouse to a given angle. Try this on your main sketch

```
#include "micromouse.h"

void setup()
{
  /*Initialize motors*/
  motors_init();
  gyro_init();
  ir_init();
  /*Initialize serial console*/
  Serial.begin(9600);
}

void loop()
{
  /*Wait for button press*/
  while(digitalRead(BUTTON) != HIGH);
  delay(1500);

  control_turn(75.0);

    /*Wait for button press*/
  while(digitalRead(BUTTON) != HIGH);
  delay(1500);

  control_turn(-75.0);
}

```