Department of Mechanical and Industrial Engineering

Mobile Robotic Control Research (Summer 1 2022)

Angular Velocity and Control of a Mobile Robot

Submitted by
Shane Meyer

Abstract

The research consisted of writing code for a Pololu 3pi+ 32u4 robot that would help to understand the robot's movements and angular velocities for its left and right motors utilizing its built-in encoders. The previous researcher left off with writing code that programmed two Pololu 3pi+ 32u4 robots, where one followed a set pattern of varying speed, the other robot used an ultrasonic sensor to follow the first robot and keep a set distance. Two codes were created during Summer 1, one code that for both the left and right motors: calculated the revolutions of each motor at time 1 and time 2, the angular displacement at time 1 and time 2, the angular velocities, displayed all these values in an easy-to-read manner, as well as displaying time 1 and time 2 and the change in time between the two measurements. The second code was based off the first code and only displayed the time stamp and the angular velocities of both motors, which was made to easily copy these values into Excel where the data was analyzed. Using the data, the difference between the angular velocities was calculated and then the percentage that the velocities were the same was calculated as well as the standard deviation of the differences and the average difference. An error value was then added to the speeds of the left and right motor to try and increase the number of times the angular velocities of the motors were the same. It was found that a value of 2 produced the best results with the angular velocities being the same 53.5% of the time and had an average difference value of 0.087 rad/s difference between the left and right angular velocities. More values or another way to correct each motor's angular velocities should be pursued next.

Date Submitted: June 27th, 2022
Date Performed: May 9th – June 27th 2022
Course Instructor: Professor Sipahi
Research TA: Haonan Fan

Introduction

Under standing the coding language used for the Pololu robots was the first process that was required. The coding language in place was Arduino, specifically version 1.8, that utilizes the serial monitor. Once this was understood the underlying issue could be analyzed which was that the robot following a set pattern was supposed to travel in a straight line, but something was causing it to veer off course causing the following robot to lose 'sight' of the lead robot. To understand why this was happening the left and right motors of each robot were looked at. A code was written to calculate the angular displacement and angular velocities of each motor to see if there were any discrepancies between the two.

Once the data was calculated the it could be seen that there was in fact a discrepancy between the two motors. One would be faster than the other at one time then the other would be faster at another time which caused the angular displacements to be different and the robot would then tilt towards the slower side and cause its path to change.

To try and combat this the error between the two motors was calculated and then multiplied by a constant which was then added to the speed value of each motor to try and correct the difference between them.

Theory and Methods

To make these calculations and find the discrepancy some values must be calculated and recorded to later use to find the angular displacement and angular velocity. The first value that needed to be found was the number of revolutions each motor had made at two times, time 1 and time 2. This was able to be found by taking the number of encoder revolutions and dividing it by the number of encoder counts/revolution. From the spec sheet it was found that the robot makes 358.3 counts. This value was then multiplied by 2 pi to find the angular displacement. The angular difference was calculated by subtracting the angular displacment at time 1 from time 2. This value is needed to find the angular velocity of each motor. The angular velocity is calculated by the equation

$$\omega = \frac{\Delta\theta}{\Delta t}$$

where ω is the angular velocity, $\Delta\theta$ is the change in angular displacement, and Δt is the change in time. The change in time (delta t) was found by taking the time at the first encoder count then delaying the next encoder count by 10 milliseconds and taking the time again. Now, the delta t was not always a perfect 10 milliseconds so it was calculated by subtracting time 1 from time 2 and using that as a variable delta t in the equation.

Using the code that displayed all the data it was clear that there were differences between the two motors. An error function was created that was the difference between the motors angular velocities then multiplied by a constant. This value was then added to the speed value of each motor the next cycle. This value caused one motor to slow down and the other speed up to try and reach an equilibrium speed that was calculated based on the previous angular velocities.

Using the acquired data it was transferred to Excel in the format of time, left angular velocity, right angular velocity. The difference was then calculated between each motors angular velocity

and then the number of times the difference was zero was calculated and then converted to a percentage. Using that percentage it could be seen whether or not the constant used to calculated the values added to each motors speed was helping to have the motors reach equilibrium or if the constant made the discrepancies worse.

Discussion and Analysis

The research conducted was straight forward and served a purpose to correct an internal issue with the Pololu 3pi+ 32u4 robots. There may be user error in the code which could be a potential source of error for the research as well as the robot could be faulty which causes the imbalances between the left and right motors.

Conclusions

The object of this research was to identify what each motor was doing and how they differed and related to eachother thorugh angular displacement amd angular velocity. Two codes were created, one to see what each motor is doing at a specific time and one that prints the data in an easy format to transfer to Excel. The data in Excel was then analyzed. The revolutions of each motor was calculated and used to find angular displacement and angular velocity. It was found that each motor reaches a max angular velocity of 22.8 rad/sec at speed 100 in arduino and that each motor does not always experience this velocity causing the robot to not travel in a straight line. A constant of 2 multiplied by the error then added to each speed gave the best results and had the same angular velocities for each motor the most amount of times which was 53.5% which was an increase of 4.6% the highest out of all values tried. It also had the closest average difference between the left and right motors which was 0.087 rad/sec.

The next steps for this research would be to either find a constant to multiply the error by to bring the angular velocities of each motor closer togther to try and eliminate the discrepancy between the angular displacement of each motor and therefore causes the robot to travel the intended path. If a constant is not able to be found a new way to correct the angular velocities must be used. According to the spec sheet the Pololu 3pi+ 32u4 is able to receive 5 volts, making sure that the robot is actually receiving this power is another step that must be done as well as making sure the cable connecting the robot to the computer is able to send 5 volts to the robot. Another process that would be helpful would be transferring the data from arduino to Excel directly instead of having to copy and paste it over. Using the PLX-DAQ extenstion for Excel was looked into and may be a viable option for this. Lastly fabricating a cover for each robot would be the final step in this process.

Appendix A: Arduino Code

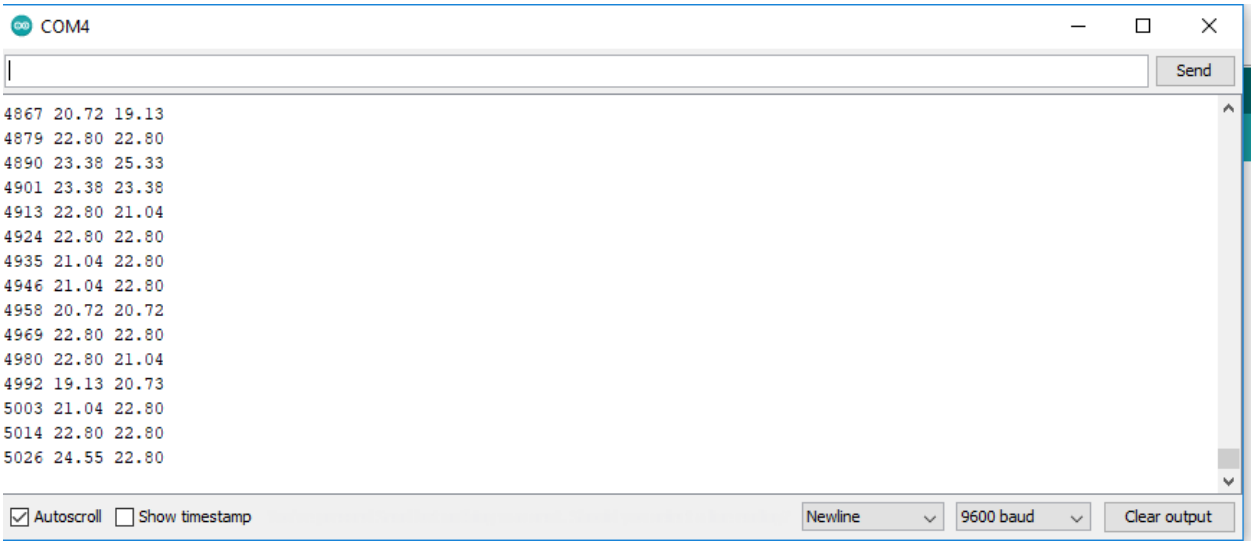
SPEED__W_CORRECTION_MEYER_S_26JUNE22

```
#include <Pololu3piPlus32U4.h>
#include <math.h>

using namespace Pololu3piPlus32U4;

Encoders encoders;                                  // Initializing parts of arduino
Buzzer buzzer;
Motors motors;
ButtonA buttonA;
ButtonC buttonC;
ButtonB buttonB;
const double pi = 3.141592653589793238462;          // pi as a constant used for calculations
char report[80];                                    // report used to display encoder values
unsigned long myTime;                               // variable used for displaying the elapsed time
float encoderCounts = 358.3;                        // the amount of times the encoder counts per revolution of the wheels
unsigned long oldTime;                              // variable used for calculating delta t
unsigned long t1;                                 // variable used for calculating delta t
unsigned long t2;                                // variable used for calculating delta t
const unsigned long period = 10;                    // the period that should be delta t

float al = 0;                           // Action Left, increases speed by x amount
float ar = 0;                           // Action Right, increases speed by x amount


void setup() {
    int16_t countsLeft = encoders.getCountsAndResetLeft();          //Clears the number of revolutions left motor made (left # on serial monitor)
    int16_t countsRight = encoders.getCountsAndResetRight();        //Clears the number of revolutions right motor made (right # on serial monitor)
}

void loop() {
    int16_t countsLeft = encoders.getCountsLeft();          //the number of revolutions of left wheel at t1
    int16_t countsRight = encoders.getCountsRight();        //the number of revolutions of right wheel at t1

    t1 = millis();
    motors.setSpeeds(100+al, 100+ar);           // sets speed of both motors to 100
    delay(10);
    int16_t countsLeft2 = encoders.getCountsLeft();         //the number of revolutions of left wheel at t2
    int16_t countsRight2 = encoders.getCountsRight();       //the number of revolutions of right wheel at t2
    t2 = millis();

    // Calculations
    float dt = (t2 - t1);                                   // calculates dt between encoder values at t1 and t2
    float lrev = countsLeft / encoderCounts;                // calculates the left revolutions made at t1
    float rrev = countsRight / encoderCounts;               // calculates the right revolutions at t1
    float leftThetal = lrev * (2 * pi);
    float rightThetal = rrev * (2 * pi);
    float lrev2 = countsLeft2 / encoderCounts;
```

SPEED__W_CORRECTION_MEYER_S_26JUNE22

```
    float lrev = countsLeft / encoderCounts;                    // calculates the left revolutions made at tl
    float rrev = countsRight / encoderCounts;                   // calculates the right revolutions at tl
    float leftThetal = lrev * (2 * pi);
    float rightThetal = rrev * (2 * pi);
    float lrev2 = countsLeft2 / encoderCounts;
    float rrev2 = countsRight2 / encoderCounts;
    float leftTheta2 = lrev2 * (2 * pi);
    float rightTheta2 = rrev2 * (2 * pi);
    float wl = (leftTheta2 - leftThetal) / (dt * .001);
    float wr = (rightTheta2 - rightThetal) / (dt * .001);

//error

float e = (wl - wr);

al = -e * 2.01;
ar = e * 2.01;

 if(buttonC.getSingleDebouncedRelease()) {
  delay(10);
    motors.setSpeeds(0,0);
  }


    /*

      snprintf_P(report, sizeof(report),          //Sends info/displays on serial monitor
        PSTR("%6d %6d"),
        countsLeft, countsRight);
    Serial.print('\n');
    Serial.println(report);
      snprintf_P(report, sizeof(report),          //Sends info/displays on serial monitor
        PSTR("%6d %6d"),
        countsLeft2, countsRight2);
    Serial.print('\n');
    Serial.println(report);
*/

      //Serial.print("Time: ");
       myTime = millis();
       Serial.print(myTime); // prints time since program started
       Serial.print(' ');
    Serial.print(wl);
       Serial.print(' ');
    Serial.print(wr);
    Serial.print('\n');
}
```

Serial monitor display for code 2



Code 1

FINAL_ENCODER_VALUES_MEYER_S_23JUNE22

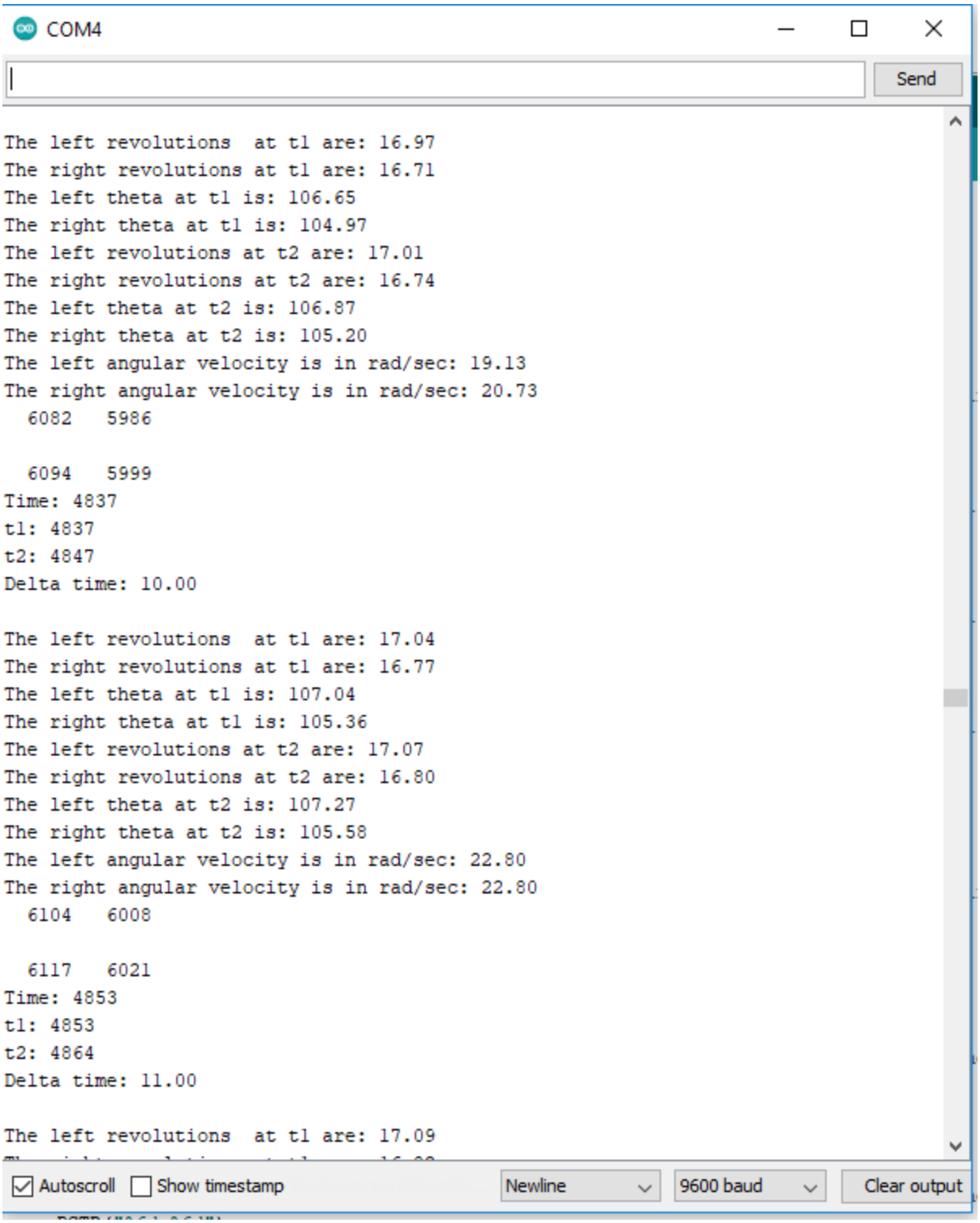
```
/*This code is better for seeing what each individual motor is doing, will display
 * all the following values for the left and right encoder: revolutions at t1 and t2,
 * angular displacement at t1 and t2, angular velocity, encoder counts at t1 and t2,
 * t1, t2, and delta t
*/
#include <Pololu3piPlus32U4.h>                    // Needed libraries
#include <math.h>
using namespace Pololu3piPlus32U4;

Encoders encoders;                                  //Initializes arduino variables
Buzzer buzzer;
Motors motors;
ButtonA buttonA;
ButtonC buttonC;
ButtonB buttonB;
const double pi = 3.14159265358979238462;           // pi as a constant used for calculations
char report[80];                                    // report used to display encoder values
unsigned long myTime;                               // variable used for displaying the elapsed time
float encoderCounts = 358.3;                        // the amount of times the encoder counts per revolution of the wheels
unsigned long t1;                          // variable used for calculating delta t
unsigned long t2;                         // variable used for calculating delta t

void setup() {
    int16_t countsLeft = encoders.getCountsAndResetLeft();           //Clears the number of revolutions left motor made (left # on serial monitor)
    int16_t countsRight = encoders.getCountsAndResetRight();         //Clears the number of revolutions right motor made (right # on serial monitor)
}

void loop() {

    int16_t countsLeft = encoders.getCountsLeft();           //the number of revolutions of left wheel at t1
    int16_t countsRight = encoders.getCountsRight();         //the number of revolutions of right wheel at t1

    t1 = millis();                       // Takes time right before action starts, needs to be before so time can pass between measurements
    motors.setSpeeds(100, 100);          // sets speed of both motors to 100 (Action that robot performs is put here)
    Serial.print("t1: ");                // Records time after button is pressed (t1) (Once the wheels start spinning)
    Serial.println(t1);                  //Prints T1

    delay(10);                           // Does not delay action (wheel spin in this case, but does delay 10 milliseconds between encoder measurements)
    int16_t countsLeft2 = encoders.getCountsLeft();         //the number of revolutions of left wheel at t2
    int16_t countsRight2 = encoders.getCountsRight();       //the number of revolutions of right wheel at t2
    t2 = millis();                                         //Takes time at t2
    Serial.print("t2: ");                                  // Prints t2
    Serial.println(t2);

    //Calculations, and printing info on serial monitor
    float dt = (t2 - t1);                    // calculates dt between encoder values at t1 and t2
    Serial.print("Delta time: ");                     // Displays what actual dt, which is not always 10 milliseconds
```

FINAL_ENCODER_VALUES_MEYER_S_23JUNE22

```
float dt = (t2 - t1);                    // calculates dt between encoder values at t1 and t2
Serial.print("Delta time: ");                        // Displays what actual dt, which is not always 10 milliseconds
Serial.println(dt);

// Calculations made at t1
Serial.print('\n');                                          // prints on a new line
float lrev = countsLeft / encoderCounts;                     // calculates the left revolutions made at t1
Serial.print("The left revolutions  at t1 are: ");
Serial.print(lrev);                                          // displays the left revolutions at t1
Serial.print('\n');                                          // prints on a new line
float rrev = countsRight / encoderCounts;                    // calculates the right revolutions at t1
Serial.print("The right revolutions at t1 are: ");
Serial.print(rrev);                                           // displays right revolutions at t1
Serial.print('\n');                                           // prints on a new line
float leftThetal = lrev * (2 * pi);                           // Calculates the left angular displacement
Serial.print("The left theta at t1 is: ");                    // Displays angular displacement
Serial.print(leftThetal);
Serial.print('\n');                                           // prints on a new line
float rightThetal = rrev * (2 * pi);                          // Calculates the right angular displacement
Serial.print("The right theta at t1 is: ");
Serial.print(rightThetal);

// Calculations made at t2
Serial.print('\n');                                            // prints on a new line
float lrev2 = countsLeft2 / encoderCounts;
Serial.print("The left revolutions at t2 are: ");
Serial.print(lrev2);
Serial.print('\n');                                          // prints on a new line
float rrev2 = countsRight2 / encoderCounts;
Serial.print("The right revolutions at t2 are: ");
Serial.print(rrev2);
Serial.print('\n');                                           // prints on a new line
float leftTheta2 = lrev2 * (2 * pi);
Serial.print("The left theta at t2 is: ");
Serial.print(leftTheta2);
Serial.print('\n');                                           // prints on a new line
float rightTheta2 = rrev2 * (2 * pi);
Serial.print("The right theta at t2 is: ");
Serial.print(rightTheta2);
Serial.print('\n');                                           // prints on a new line

//Calculating Angular velocity (omega)
float wl = (leftTheta2 - leftThetal) / (dt * .001);
Serial.print("The left angular velocity is in rad/sec: ");
Serial.print(wl);
Serial.print('\n');                                          // prints on a new line
float wr = (rightTheta2 - rightThetal) / (dt * .001);
```

```
    snprintf_P(report, sizeof(report),          //Displays encoder counts for left and right at t1
    PSTR("%6d %6d"),
    countsLeft, countsRight);
    Serial.print('\n');
    Serial.println(report);
    snprintf_P(report, sizeof(report),          //Displays encoder counts for left and right at t2
    PSTR("%6d %6d"),
    countsLeft2, countsRight2);
    Serial.print('\n');
    Serial.println(report);

      Serial.print("Time: ");
      myTime = millis();
      Serial.println(myTime); // prints time since program started

}
```

Serial monitor display for code 1


COM4
Send
The left revolutions at t1 are: 16.97
The right revolutions at t1 are: 16.71
The left theta at t1 is: 106.65
The right theta at t1 is: 104.97
The left revolutions at t2 are: 17.01
The right revolutions at t2 are: 16.74
The left theta at t2 is: 106.87
The right theta at t2 is: 105.20
The left angular velocity is in rad/sec: 19.13
The right angular velocity is in rad/sec: 20.73
6082 5986
6094 5999
Time: 4837
t1: 4837
t2: 4847
Delta time: 10.00
The left revolutions at t1 are: 17.04
The right revolutions at t1 are: 16.77
The left theta at t1 is: 107.04
The right theta at t1 is: 105.36
The left revolutions at t2 are: 17.07
The right revolutions at t2 are: 16.80
The left theta at t2 is: 107.27
The right theta at t2 is: 105.58
The left angular velocity is in rad/sec: 22.80
The right angular velocity is in rad/sec: 22.80
6104 6008
6117 6021
Time: 4853
t1: 4853
t2: 4864
Delta time: 11.00
The left revolutions at t1 are: 17.09
Autoscroll
Show timestamp
Newline
9600 baud
Clear output