ECEN 345/ECEN 3450-001

Mobile Robotics I

Prof. A. Gilmore (Instructor)
Carl Endrulat (TA)

Laboratory Assignment #2

Lab 2 - Dead Reckoning [Report]

Written by :
Nelson Lefebvre 03137910

Department of Electrical and Computer Engineering (ECE)
University of Nebraska Lincoln/Omaha

Due: 09/17/2022

• Overview:

Description : The main lesson of this lab is to make us understand how to determine in a primitive way how the robot locates itself and how to determine the realization of some movement of the robot in steps to reach the goal and then calculating the distance to see this margin of error. All of this to understand the impacts of the ability to program the robot for autonomous movement along a predetermined path.

Requirement: - CEENBoT-API – Getting Started guide

- CEENBoT-API: Programming Fundamentals
- Mobile Robotics I Module 4 In Class-1.pdf
- Ceenbot + Microchip studio + Ceenbot Utility tools (LAB 1)

Time invested in this project : For the experimentation i spend around 6h to figure exactly what was the best configuration (ex: for turning at 90*) and because I am not familiar with the ft and in measurement. And for the report I spent around 1-2h. So I spend around 7-8h for this lab.

Team : I have done this lab alone, so no repartition time for the tasks.

Problems encountered : For the technical problems I had several. In particular to move on the carpet and on the normal ground. During the demonstration of the code the distances were much less precise since I worked on a carpet. But also when I realized the code to make a square, I had some problems. When I rotated it to 90* it was more like a 145* and so I had to determine some values by trial and error. And I didn't manage to transcribe the rotation to make a circle because of lack of time.

• Background :

Here for the background in this lab we used two key terms that are interesting to define:

- Odometry : Technique to determine the position of the robot when it moves (often with sensors).
- Dead reckoning : The process of calculating the current position of some moving object by using a previously determined position.
- Locomotion : Locomotion refers to how a robot moves around from place to place.

And so in this lab we will use dead reckoning as locomotion to determine on the predefined long distance what is the position of the robot after several movements and thus determine the Odometry with this procedure. We will also discuss the veracity of this technique and the approximation that can be generated with several parameters.

• Procedure :

We can divide this Lab in 4 parts. In the 1st part we set up the simple movement of the robot, and see the basic problems of dead reckoning. That means we need to use simple mouvement like going from point A to point B in a straight line with different distances. For this we learn in the documentation that CeenBot robots do not work with traditional measurements like meters or feet but in step (as we use stepper motors). And so in our code it is important to implement the different formula seen below to be able to convert the different distances in Inches.

Supplementary Note – It is possible to create a relation between "steps" and "distance". Recall the *circumference formula*:

$$C = 2\pi r$$

If you measure the *radius* r of the wheel *very* carefully, you can compute the circumference C. Then, you can compute *how far* each "step" moves you (i.e., the *distance-per-step*) by using the conversion:

$$d_{\text{step}} = \frac{C}{200}$$

Finally, if D represents the total distance you want to travel, the "number of steps" required to travel that distance can be computed as:

$$N_{\text{steps}} = \frac{D}{d_{\text{step}}}$$

Then we modify different parameters to see the impact of the environment for the robot.

For the first one we measure the different distance error with a different acceleration entered in the code and we had this array :

Test nbr/Distance	6in	12in	18in	24in
1 (400)	6	11.85	17.75	23.8
2 (600)	5.9	11.8	17.7	23.7
3 (1200)	5.8	11.7	17.7	23.6
4 (2400)	5.6	11.6	17.6	23.5

For the 2nd one we use a different floor to see the difference :

Test carpet nbr/Distance	6in	12in	18in	24in

1 (400)	6	12	18	23.3
2 (600)	5.8	11.7	17.5	23.1
3 (1200)	5.8	11.7	17.4	23
4 (2400)	5.7	11.6	17.4	23

So we can see that the odometry error changes a lot between each acceleration of the type of the floor (environment). So we can conclude for the first part that the distance can be the predicted distance can be affected by different factors such as the type of terrain (more or less passable) and the acceleration used (which leads to a lower accuracy).

In the 2nd-3rd parts we test different parameters which are the trajectory of the object. In fact in the 2nd part we calculate the odometry error of a simple trajectory change which is a square. So it's basically just a 90* repeated 3 times to see how much it impacted the distance that we had predicted.

And we got this result:

Distance Error (1st corner): 0.041 ft

Distance Error (3rd corner): 0.3ft

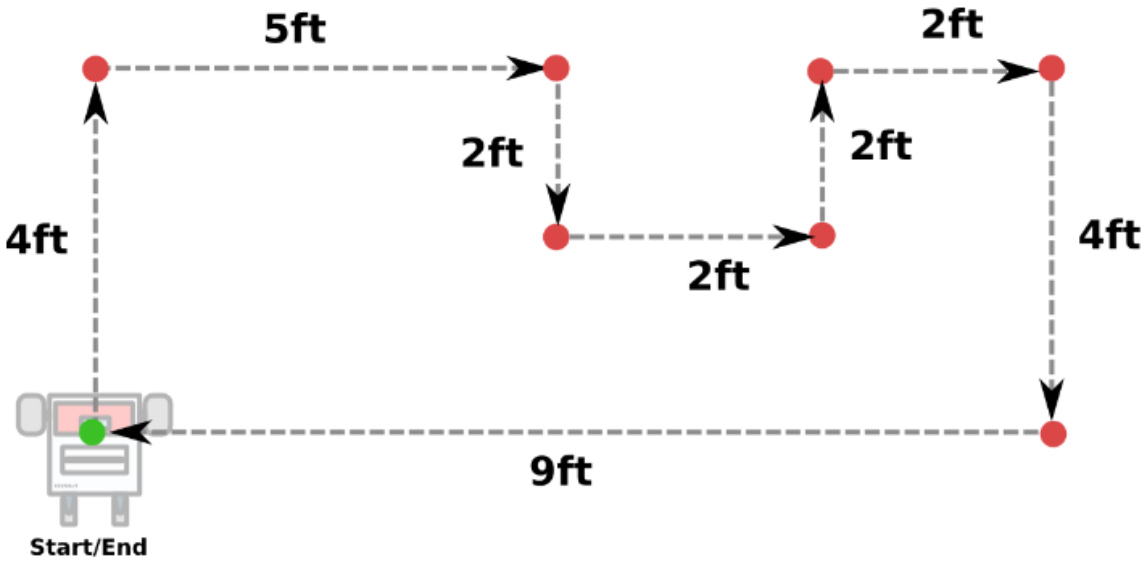
Distance Error(2nd corner): 0.35 ft

Distance Error (4th corner): 0.46 ft

$$e_{avg} = \frac{e_{d1}+e_{d2}+e_{d3}+e_{d4}}{4} = (0.041+0.3+0.35+0.46)/4 = 0.38$$

So as we can see even with simple 90* rotation, we result with an approximation of 0.38 of odometry error.

The next step is to try with a different figure with a more complicated trajectory (for example a maze) to see if the impact is notable:



The measurement and the calculation of the odometry error :

Distance Error (1): 0.2 ft
Distance Error(5): 0.5
Distance Error (2): 0.3 ft
Distance Error(6): 0.6
Distance Error (3): 0.4
Distance Error(7): 0.8
Distance Error (4): 0.4
Distance Error(8): 1.1

Determine the average error using:

$$e_{avg} = \frac{e_{d1}+e_{d2}+e_{d3}+e_{d4}+e_{d5}+e_{d6}+e_{d7}+e_{d8}}{8} = 4.3/8 = 0.53$$

So as we can see the more we add different trajectories the more the odometry error will be huge. It's basically like the chain reaction, the distance error will create more distance error than we had expected from the original plan.

Then we had the part 4 of the Lab that I didn't succeed.
I nevertheless had several possible approaches to solve the problem like calculating the circumference of the circle and then realizing a simple forward motion but turning the right wheel more slowly, thus potentially creating a circle.

Discussion about the Source Code :

Here the different source codes can be quite easily deconstructed. First of all they all use the same method for moving. We use the function :

```
STEPPER_move_stwt( STEPPER_BOTH,
```

```
        STEPPER_FWD, 6/d, 200, 2400, STEPPER_BRK_OFF, // Left
        STEPPER_FWD, 6/d, 200, 2400, STEPPER_BRK_OFF ); // Right
```

Who it takes in parameter respectively which stepper is used and then the action to be performed by the right wheel in 2nd argument and the left wheel in 5th. So by choosing STEPPER_FWD to move the wheel forward or STEPPER_REV to move the wheel backward you can move the vehicle in any direction (even left or right by turning the wheel in the desired direction backward and the other forward).

Then we are using the formula on the top if the code to calculate the accurate distance in IN or in FEET, in the function we will use 6/d to move 6in and 6*12/d to move 6ft for exemple. Then 200 is the number of steps and 2400 is the acceleration of the wheel.

And to take the measurement between all the movement we are using the function : TMRSRVC_delay(10000);

Whoever make the code wait 10,000ms wait to execute the next code.

To complete the maze and the square figure we just predicate by turning left or right the robot to complete it.

• Conclusion :

We can therefore conclude that the odometric error is strongly influenced by many factors and can create a large distance between what was predicted on paper and what actually happened. Whether it is the speed at which the robot is accelerating, the type of terrain, the speed at which the right or left wheel is turning, etc. And so I think it's probably more interesting to have a robot that reacts to its environment to detect obstacles in real time because then I think we reduce the margin of error for each distance.

CODE 1:

```
/*    Auth: Nelson Lefebvre
*     Date: 09/17/2022
*     Course: ECEN3450-Mobile Robotics I
*     Lab: Lab#4
 */

#include "capi324v221.h"

void CBOT_main( void )
{
//Formula to calculate the real life measure
   float c = 2*3.14*1.7;
   float d = c/200;
```

```
STEPPER_open(); // Open STEPPER module for use.

// Move BOTH wheels forward.
STEPPER_move_stwt( STEPPER_BOTH,
STEPPER_FWD, 6/d, 200, 2400, STEPPER_BRK_OFF, // Left
STEPPER_FWD, 6/d, 200, 2400, STEPPER_BRK_OFF ); // Right

// we clean the LCD screen and print the distance on it
LCD_open();
LCD_clear();
LCD_printf("6in\n");

// We wait 10000ms to take measure
TMRSRVC_delay( 10000 );



// Move BOTH wheels forward.
STEPPER_move_stwt( STEPPER_BOTH,
STEPPER_FWD, 12/d, 200, 2400, STEPPER_BRK_OFF, // Left
STEPPER_FWD, 12/d, 200, 2400, STEPPER_BRK_OFF ); // Right

// we clean the LCD screen and print the distance on it
LCD_open();
LCD_clear();
LCD_printf("12in\n");

// We wait 10000ms
TMRSRVC_delay( 10000 );

// Move BOTH wheels forward.
STEPPER_move_stwt( STEPPER_BOTH,
STEPPER_FWD, 18/d, 200, 2400, STEPPER_BRK_OFF, // Left
STEPPER_FWD, 18/d, 200, 2400, STEPPER_BRK_OFF ); // Right


// we clean the LCD screen and print the distance on it
LCD_open();
LCD_clear();
LCD_printf("18in\n");

// We wait 10000ms
TMRSRVC_delay( 10000 );

// Move BOTH wheels forward.
STEPPER_move_stwt( STEPPER_BOTH,
STEPPER_FWD, 24/d, 200, 2400, STEPPER_BRK_OFF, // Left
```

```
        STEPPER_FWD, 24/d, 200, 2400, STEPPER_BRK_OFF ); // Right


        // we clean the LCD screen and print the distance on it
        LCD_open();
        LCD_clear();
        LCD_printf("24in\n");

        // We wait 10000ms
        TMRSRVC_delay( 10000 );

}
```

CODE 2:

```
/*    Auth: Nelson Lefebvre
*     Date: 09/17/2022
*     Course: ECEN3450-Mobile Robotics I
*     Lab: Lab#2
*/

#include "capi324v221.h"

void CBOT_main( void )
{
    float c = 2*3.14*1.7;
    float d = c/200;


        STEPPER_open(); // Open STEPPER module for use.

//Loop to repeat the moving left 4 times, to make a square.
        for (int i = 0; i < 4; i++)
        {
        // Move BOTH wheels forward.
        // 4in*12=4ft
        STEPPER_move_stwt( STEPPER_BOTH,
        STEPPER_FWD, 4*12/d, 200, 400, STEPPER_BRK_OFF, // Left
        STEPPER_FWD, 4*12/d, 200, 400, STEPPER_BRK_OFF ); // Right

        // Then TURN RIGHT (~90-degrees)...
        STEPPER_move_stwt( STEPPER_BOTH,
        STEPPER_FWD, 126, 200, 400, STEPPER_BRK_OFF, // Left
        STEPPER_REV, 126, 200, 400, STEPPER_BRK_OFF ); // Right
        }

}
```

CODE 3:

```
/*      Auth: Nelson Lefebvre
*       Date: 09/17/2022
*       Course: ECEN3450-Mobile Robotics I
*       Lab: Lab#2
*/

#include "capi324v221.h"

void CBOT_main( void )
{
        float c = 2*3.14*1.7;
        float d = c/200;


            STEPPER_open(); // Open STEPPER module for use.

            // Move 4ft.
            STEPPER_move_stwt( STEPPER_BOTH,
            STEPPER_FWD, 4*12/d, 200, 400, STEPPER_BRK_OFF, // Left
            STEPPER_FWD, 4*12/d, 200, 400, STEPPER_BRK_OFF ); // Right

            // Then TURN RIGHT (~90-degrees)...
            STEPPER_move_stwt( STEPPER_BOTH,
            STEPPER_FWD, 126, 200, 400, STEPPER_BRK_OFF, // Left
            STEPPER_REV, 126, 200, 400, STEPPER_BRK_OFF ); // Right


        // We wait 10000ms
        TMRSRVC_delay( 10000 );

            // Move 5ft.
            STEPPER_move_stwt( STEPPER_BOTH,
            STEPPER_FWD, 5*12/d, 200, 400, STEPPER_BRK_OFF, // Left
            STEPPER_FWD, 5*12/d, 200, 400, STEPPER_BRK_OFF ); // Right

            // Then TURN RIGHT (~90-degrees)...
            STEPPER_move_stwt( STEPPER_BOTH,
            STEPPER_FWD, 126, 200, 400, STEPPER_BRK_OFF, // Left
            STEPPER_REV, 126, 200, 400, STEPPER_BRK_OFF ); // Right


        // We wait 10000ms
        TMRSRVC_delay( 10000 );

            // Move 2ft.
            STEPPER_move_stwt( STEPPER_BOTH,
            STEPPER_FWD, 2*12/d, 200, 400, STEPPER_BRK_OFF, // Left
```

```
   STEPPER_FWD, 2*12/d, 200, 400, STEPPER_BRK_OFF ); // Right

   // Then TURN LEFT (~90-degrees)...
   STEPPER_move_stwt( STEPPER_BOTH,
   STEPPER_REV, 126, 200, 400, STEPPER_BRK_OFF, // Left
   STEPPER_FWD, 126, 200, 400, STEPPER_BRK_OFF ); // Right

// We wait 10000ms
TMRSRVC_delay( 10000 );

   // Move 2ft.
   STEPPER_move_stwt( STEPPER_BOTH,
   STEPPER_FWD, 2*12/d, 200, 400, STEPPER_BRK_OFF, // Left
   STEPPER_FWD, 2*12/d, 200, 400, STEPPER_BRK_OFF ); // Right

   // Then TURN LEFT (~90-degrees)...
   STEPPER_move_stwt( STEPPER_BOTH,
   STEPPER_REV, 126, 200, 400, STEPPER_BRK_OFF, // Left
   STEPPER_FWD, 126, 200, 400, STEPPER_BRK_OFF ); // Right

// We wait 10000ms
TMRSRVC_delay( 10000 );

   // Move 2ft.
   STEPPER_move_stwt( STEPPER_BOTH,
   STEPPER_FWD, 2*12/d, 200, 400, STEPPER_BRK_OFF, // Left
   STEPPER_FWD, 2*12/d, 200, 400, STEPPER_BRK_OFF ); // Right

   // Then TURN RIGHT (~90-degrees)...
   STEPPER_move_stwt( STEPPER_BOTH,
   STEPPER_FWD, 126, 200, 400, STEPPER_BRK_OFF, // Left
   STEPPER_REV, 126, 200, 400, STEPPER_BRK_OFF ); // Right

// We wait 10000ms
TMRSRVC_delay( 10000 );

   // Move 2ft.
   STEPPER_move_stwt( STEPPER_BOTH,
   STEPPER_FWD, 2*12/d, 200, 400, STEPPER_BRK_OFF, // Left
   STEPPER_FWD, 2*12/d, 200, 400, STEPPER_BRK_OFF ); // Right

   // Then TURN RIGHT (~90-degrees)...
   STEPPER_move_stwt( STEPPER_BOTH,
   STEPPER_FWD, 126, 200, 400, STEPPER_BRK_OFF, // Left
   STEPPER_REV, 126, 200, 400, STEPPER_BRK_OFF ); // Right

// We wait 10000ms
TMRSRVC_delay( 10000 );
```

```
            // Move 4ft.
            STEPPER_move_stwt( STEPPER_BOTH,
            STEPPER_FWD, 4*12/d, 200, 400, STEPPER_BRK_OFF, // Left
            STEPPER_FWD, 4*12/d, 200, 400, STEPPER_BRK_OFF ); // Right

            // Then TURN RIGHT (~90-degrees)...
            STEPPER_move_stwt( STEPPER_BOTH,
            STEPPER_FWD, 126, 200, 400, STEPPER_BRK_OFF, // Left
            STEPPER_REV, 126, 200, 400, STEPPER_BRK_OFF ); // Right

        // We wait 10000ms
        TMRSRVC_delay( 10000 );

            // Move 9ft.
            STEPPER_move_stwt( STEPPER_BOTH,
            STEPPER_FWD, 9*12/d, 200, 400, STEPPER_BRK_OFF, // Left
            STEPPER_FWD, 9*12/d, 200, 400, STEPPER_BRK_OFF ); // Right

            // Then TURN RIGHT (~90-degrees)...
            STEPPER_move_stwt( STEPPER_BOTH,
            STEPPER_FWD, 126, 200, 400, STEPPER_BRK_OFF, // Left
            STEPPER_REV, 126, 200, 400, STEPPER_BRK_OFF ); // Right

        // We wait 10000ms
        TMRSRVC_delay( 10000 );



}
```