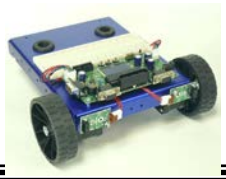


Lab 02

Odometry (Dead Reckoning): Go-To-Angle & Go-To-Goal

(Demonstration due in class on **Thursday**)

(Code and Memo due in Moodle drop box by midnight on **Sunday at midnight**)

Read this entire lab procedure before coming to lab.

****

Read this entire lab procedure starting the lab.

Purpose: The purpose of this lab is to use motion control to implement Go-To-Angle and Go-To-Goal behaviors on the CEENBot.

Objectives: At the conclusion of this lab, the student should be able to:

- Program the CEENBoT to move to a given angle
- Program the CEENBoT to move to a given goal position

Equipment: CEENBoT platform, '324 v2.21.
AVR In-System Programmer (ISP)
Masking Tape
Ruler

Theory:

In this lab, you will use odometry concepts to implement the go-to-angle and go-to-goal behaviors on the robot. Remember to continue to use the adjustment factors that you determined in Lab 01 for the robot motion commands for the left and right wheel to correct for systematic errors in odometry. Remember you will never be able to correct for **all** odometry error and must learn to implement the most ideal solution considering this variable. In order to use feedback control to implement these behaviors it is necessary to have sensors to measure the robot's actual position to compare to the desired position in order to calculate the error input for the controller. Figure 1 demonstrates the necessary feedback control system.

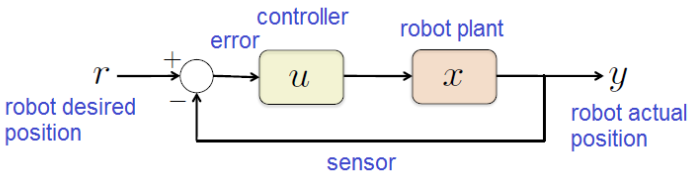
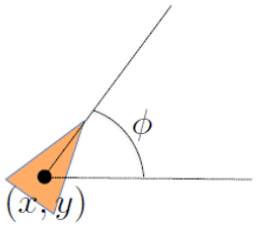


Figure 1: Robot Motion Feedback Control System

Since there are not any encoders on the CEENBot motors, we will estimate the measured angle or goal by using the precision of the steps or velocity and time to estimate position using forward kinematics. Figure 2 shows the dynamic model for implementing the Go-To-Goal behavior. Recall that on the CEENBot it is possible to specify number of steps or speed with a given time to move to a specific position.



- Inputs: v, ω
- Dynamics:

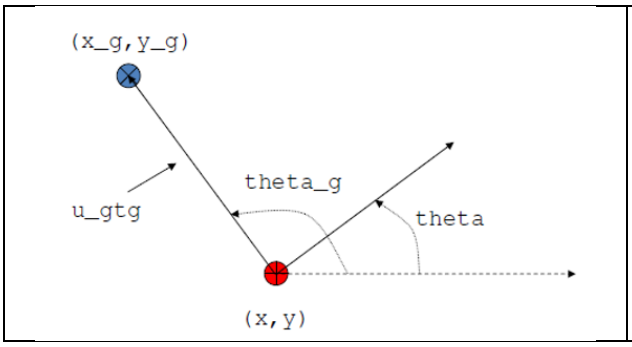
$$\begin{cases} \dot{x} = v\cos\phi \\ \dot{y} = v\sin\phi \\ \dot{\phi} = \omega \end{cases}$$

Figure 2: Unicycle robot dynamic model

In order to implement the behaviors, it is necessary to convert the desired distance or angle to the number of steps or velocity required for the robot's wheels. To find the number of steps, this requires the team to measure the diameter of the wheel and use the formula,

$$N_{steps} = 200\,\frac{total\ desired\ distance}{\pi \cdot wheel\ diameter}$$

In order to implement the Go-To-Goal behavior, the robot would calculate the angle to the goal point and use the Go-To-Angle behavior to turn first and then use the number of steps to get to the goal. Therefore, the Go-To-Angle behavior should be implemented first. To calculate the desired turn angle, use the following formula.



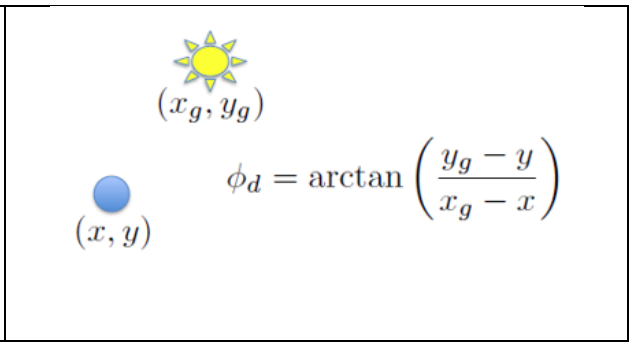


Figure 3: Go-To-Goal Notation

To find the number of steps necessary for the robot to spin to that angle, you must make the following calculation. Measure the wheel base of the robot and refer to this value as L. Therefore, when the robot spins in a circle it traces out a circle of circumference of πL. To find the number of steps to spin a complete circle use the following formula where D is the diameter of the wheels.

$$N_{steps} = 200\,\frac{\pi \cdot L}{\pi \cdot D} = 200\frac{circumference\ of\ robot\ spin}{circumference\ of\ wheel}$$

To spin the robot to any other angle in degrees, $\phi_{desired}$, use the following formula to calculate the number of steps where one wheel moves forward and the other moves in reverse.

$$N_{steps} = 200\,\frac{\pi \cdot L}{\pi \cdot D} \cdot \frac{\phi_{desired}}{360}$$

An alternative for implementing the Go-To-Goal behavior which may be a bit more difficult is to move the robot in a linear and angular motion at the same time as it converges on the goal point. In order to use this model, you will use the differential drive robot model to calculate the required motion given a constant linear velocity and variable angular velocity. This method would require the robot to incrementally calculate the vector to the goal position as

it moves to converge on the position within some error. The Go-To-Goal behavior could be implemented by using a P controller to create the behavior, $\omega = K(\phi_d - \phi)$.

****

LAB PROCEDURE

****

Part I – Go To Angle Behavior

1. Create a method to implement the Go-To-Angle behavior
2. The robot should move the given angle and move forward for a predetermined period of time
3. Take measurements to estimate the accuracy of the Go-To-Angle behavior

Part II – Go To Goal Behavior

1. Create a method to implement the Go-To-Goal behavior
2. The robot should move to the given goal position and stop within a certain error
3. Take measurements to estimate the accuracy of the Go-To-Goal behavior

Demonstration

Finally, create a program to demonstrate the go-to-angle and go-to-goal robot behaviors. The program should be as modular as possible with logically named functions and variables. Use the pushbutton switches to select the go-to-angle or go-to-goal behaviors. Then, use the pushbuttons to input a given angle for the robot to turn. The LCD should display the behavior as well as the requested angled. The robot should then turn the requested angle relative to the initial starting position. In the second half of the demonstration, use the pushbutton to input a given goal position assuming the robot starts at $(x,y,\phi) = (000)$. The LCD should display the requested behavior and goal position. The robot should then move to the goal point. You should also use the LCD and/or buzzers to indicate behaviors as well as robot intentions or current state.

Bring your robot fully charged to class on Thursday for the demonstration. Note that you always must re-flash the factory firmware and plug in the AC adapter in order for the robot to charge. Alternately, you can put the robot battery in the RC car battery charger. Note that this is a fast charger and will not last as long as the outlet charge.

More questions to answer in the lab memo

1. How did you calculate the turn angle for the robot given the stepper motor precision?
2. How did you calculate the odometry to move to the goal position given the stepper motor precision?
3. What type of accuracy/error did you have in the go-to-goal behavior?
4. What type of accuracy/error did you have in the go-to-angle behavior?
5. What could you do to improve the accuracy of the behaviors?
6. Did your team use the turn then forward approach for go-to-goal or move and turn at the same time? If so, what were the pros and cons of using your approach versus the other one?

Memo Guidelines:

Please use the following checklist to insure that your memo meets the basic guidelines.

- ✓ Format
  - o Begins with Date, To , From, Subject
  - o Font no larger than 12 point font
  - o Spacing no larger than double space
  - o Includes handwritten initials of both partners at the top of the memo next to the names
  - o Written as a paragraph not bulleted list
  - o No longer than three pages of text
- ✓ Writing
  - o Memo is organized in a logical order
  - o Writing is direct, concise and to the point
  - o Written in first person from lab partners
  - o Correct grammar, no spelling errors
- ✓ Content
  - o Starts with a statement of purpose
  - o Discusses the strategy or pseudocode for implementing the robot paths (may include a flow chart)
  - o Discusses the tests and methods performed
  - o States the results including error analysis
  - o Shows data tables with error analysis and required plots or graphs
  - o Answers all questions posed in the lab procedure
  - o Clear statement of conclusions

Grading Rubric:

The lab is worth a total of 30 points and is graded by the following rubric.

Points	Demonstration	Code	Memo
10	Excellent work, the robot performs exactly as required	Properly commented with a header and function comments, easy to follow with modular components	Follows all guidelines and answers all questions posed
7.5	Performs most of the functionality with minor failures	Partial comments and/or not modular with objects	Does not answer some questions and/or has spelling, grammatical, content errors
5	Performs some of the functionality but with major failures or parts missing	No comments, not modular, not easy to follow	Multiple grammatical, format, content, spelling errors, questions not answered
0	Meets none of the design specifications or not submitted	Not submitted	Not submitted

Here is an excerpt from the RangeSensors.c program so that you can see an example of a proper heading and properly commented code.

```
//RangeSensors.c
//A demo application to test robot locomotion
//CEENBoT peripherals and effectors
//Carlotta A. Berry
//Dane Bennington
//Jose Santos
//August 9, 2011
        //main function to run the robot
        void CBOT_main(void)
        {
                ATopstat = ATTINY_open();//open the tiny microcontroller
                LEopstat = LED_open(); //open the LED module

                //keep the microcontroller running
                while(1)
                {
                        btnValue=WaitButton();
                        if (btnValue==1)
                        {
                                //Check Light Sensors
                                //beep once
                                SPKR_play_beep( 500,500,100);//500 Hz for 500 ms
                                i = 0;
```

Submission Requirements:

You must submit you properly commented code as a zipped folder of the C code and the lab memo in a zipped folder by **11:59 pm on Sunday** to the Moodle Course Drop box. Your code should be modular with functions and classes in order to make it more readable. You should use the push buttons, buzzer and LCD to indicate the robot state during program execution.