Robotics With the XBC Controller

Session 10

Instructor: David Culp

Email: culpd@cfbisd.edu

Learning Goals

- The student will learn to use the BEMF functions in order to make precise turns and will use the functions to navigate a short obstacle course.
- Schedule for tonight
  - Odometry continued
  - Video of development lessons – West Bay
  - Interview with DeWitt Perry Students
  - Final Exam

Odometry Review

- Measuring distance based upon wheel rotations.
- The robots straight line distance (d) is the number of wheel rotations * wheel circumference (C).
  - Example:
    - C = 10cm
    - # rotations = 6.5
    - d = 100mm * 6.5 rotations = 65cm
- #rotations = pulses traveled / pulses per rotation

Odometry Review (Example)

- Wheel Diameter(D) = 3.18 cm
  - C = pi*D
  - Wheel C = 3.14159 *3.18cm = ~10cm
- Pulses per rotation = 1000
- Total pulses traveled = 3500
- How far has our robot traveled?
  - #rotations = pulses traveled / pulses per rotation
    - #rotations = 3500 / 1000 = 3.5 rotations
  - d = number of wheel rotations * wheel circumference.
    - d = 3.5 rotations * 10cm = 35cm

travel_dist function

```
/*
Function: travel_dist
Purpose: Will cause two wheels to travel a certain number of cm (it is possible to use more or less
      wheels)
Parameters:
    int vel- The speed to travel in clicks/sec
    float dist- The distance in cm to travel
*/
void travel_dist(int vel, float dist)
{
    //First calculate how far to travel
    float left_total_clicks_to_travel=(dist/wheel_circumference)*(float)LEFT_CLICKS_PER_ROT;
    float right_total_clicks_to_travel=(dist/wheel_circumference)*(float)LEFT_CLICKS_PER_ROT;
    mrp(LEFT_MOTOR, vel, (long)left_total_clicks_to_travel);
    mrp(RIGHT_MOTOR, vel, (long)right_total_clicks_to_travel);
    while( (get_motor_done(LEFT_MOTOR) == 0) || (get_motor_done(RIGHT_MOTOR) ==0) )
      { };
      ao(); // turn off the other motor when one is done to avoid turns at the end
}
```

Turning

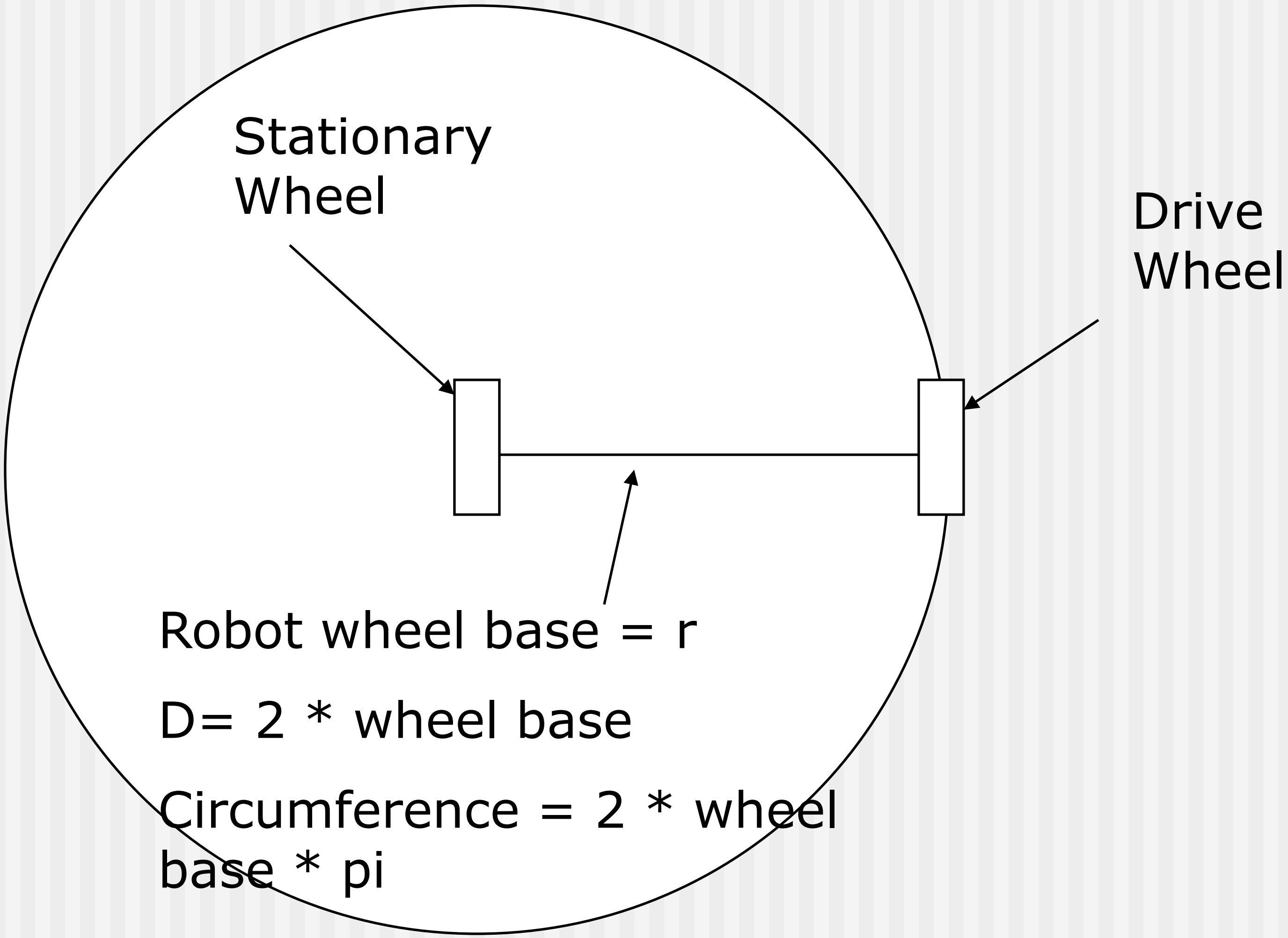
- If one wheel (the pivot wheel) is stationary in pivot turns, the drive wheel will travel in a circle, turning the robot with it.
- The robot then turns in a circle with a radius equal to the wheelbase of the robot, which is measured from the inside of the pivot wheel to the outside of the drive wheel.

Turning Continued

- The circumference of this circle can be calculated like the circumference of any other circle, using **2*PI* radius**.
- Full Turning Circle=(WHEEL_BASE*2.0)*PI

  [Note - to convert angles in degrees to radians: 360 deg = 2*PI rad]
- We can use this information to get the number of clicks we travel to turn a single degree.
- This can be multiplied by the number of degrees we want to turn to get how many clicks the drive wheel should move.

Illustration



Robot wheel base = r

D= 2 * wheel base

Circumference = 2 * wheel base * pi

Finding Clicks Per Degree

- Divide the circle’s circumference by that of the wheels.
- Then multiply the result by the number of clicks per wheel rotation and the ratio of 1/360 degrees.
- Clicks_per_degree=(full_circle/wheel_circumference)*(1.0/360.0)*RIGHT_CLICKS_PER_ROTATION;

New Additions to Our #defines and Variables

```
#define LEFT_MOTOR 0
#define RIGHT_MOTOR 2
#define LEFT_CLICKS_PER_ROT 350 /*WHEEL ROTATIONS! NOT motor rotations!*/
#define RIGHT_CLICKS_PER_ROT 350 /*WHEEL ROTATIONS! NOT motor rotations! */
#define WHEEL_DIAMETER 1.5 /* in cm */
#define WHEEL_BASE 12.0 /* in cm */
#define PI 3.14159

float wheel_circumference = WHEEL_DIAMETER*PI; //in cm
float full_circle=(WHEEL_BASE*2.0)*PI;//Total turning circle for the robot; also in cm
float
    left_clicks_per_degree=(full_circle/wheel_circumference)*(1.0/360.0)*(float)LEFT_
    CLICKS_PER_ROT;
float
    right_clicks_per_degree=(full_circle/wheel_circumference)*(1.0/360.0)*(float)RIGH
    T_CLICKS_PER_ROT;
```

Assignment 1

- Write a function called pivot_turn.
- Pivot turn takes the following parameters.
  - int motor – the motor # to use as the drive wheel.
  - int vel – the speed to move.
- void pivot_turn(int motor, int vel, float dist).
- The function calculates the number of BEMF pulses to move the drive wheel and moves it keeping the other wheel stationary.
- Use your function in a program to turn your robot an arbitrary # of degrees.

Solution

```
/*
Function: pivot_turn
Purpose: Will cause a dual drive robot to turn a certain number of degrees
Parameters:
   int motor_num- The number of the motor to use (port number)
   int vel- The speed to travel in clicks/sec
   float degrees- The distance in degrees to travel
*/
void pivot_turn(int motor_num, int vel, float degrees)
{
   //First calculate how far to travel
   float left_total_clicks_to_travel=left_clicks_per_degree*degrees;
   float right_total_clicks_to_travel=right_clicks_per_degree*degrees;
   if (motor_num == LEFT_MOTOR)
     {
      //Now move that number of pulses
      mrp(motor_num, vel, (long)left_total_clicks_to_travel);
      bmd(motor_num);
   }
   else
     {
         //Now move that number of pulses
      mrp(motor_num, vel, (long)right_total_clicks_to_travel);
      bmd(motor_num);
   }
}
```

A slightly more Elegant Solution

```
float clicks_per_degree[4];

void main()
{
   clicks_per_degree[LEFT_MOTOR]=(full_circle/wheel_circumference)*(1.0/360.0)*(float)LEFT_CLICKS_PER_ROT;
   clicks_per_degree[RIGHT_MOTOR]=(full_circle/wheel_circumference)*(1.0/360.0)*(float)RIGHT_CLICKS_PER_ROT;

   pivot_turn(LEFT_MOTOR,800,90.);
}

/*
Function: pivot_turn
Purpose: Will cause a dual drive robot to turn a certain number of degrees
Parameters:
   int motor_num- The number of the motor to use (port number)
   int vel- The speed to travel in clicks/sec
   float degrees- The distance in degrees to travel
*/
void pivot_turn(int motor_num, int vel, float degrees)
{
   //First calculate how far to travel
   float total_clicks_to_travel=clicks_per_degree[motor_num]*degrees;

   mrp(motor_num, vel, (long)total_clicks_to_travel);
   bmd(motor_num);

}
```

Assignment 2

- Set up an “odometry course”
  - Set a starting location of the robot.
  - Place objects in front of the robot.
  - Measure the distance from the objects to the robot and the distance between the objects.
  - Write a program using the two odometry functions to navigate your course.

Interviews + Final

- Video Interview with West Bay
- Short Interview with DeWitt Perry Students
- Final Exam Instructions + Goodbye and Good luck!