Work sheet for driving ActivityBot **Jeffrey La Favre** **November 6, 2015**

Calculations for driving straight with drive_goto()

Example code: **drive_goto(345, 345);**

The above code specifies that each wheel will rotate 345 encoder ticks forward.

An encoder tick is equivalent to a drive distance of 3.25 mm

1. List the number of millimeters of travel desired _____
2. Divide the distance to travel by 3.25 to find the number of encoder ticks _____
3. For forward motion, use positive values for number of encoder ticks and for reverse motion use negative values for number of encoder ticks. Each wheel, left and right, should receive the same number of encoder ticks to travel in a straight line.
4. Code format: **drive_goto(***# ticks for left wheel, # ticks for right wheel***)**

Calculations for a pivot turn using drive_goto()

In a pivot turn, one wheel remains stationary, forming the center point of the turn.

1. The robot will pivot one degree per 0.5672 ticks of the outboard wheel
2. List the number of degrees desired for the turn_____
3. Multiply the number of degrees by 0.5672 to get # ticks_____
4. Use the number of ticks in the drive_goto() statement

Example: calculate number of ticks for a 90 degree turn:

90 degrees X 0.5672 = 51 ticks

For a 90 degree right turn: **drive_goto(51, 0)**

For a 90 degree left turn: **drive_goto(0, 51)**

Calculations for a zero radius turn using drive_goto()

In a zero radius turn, one wheel moves forward while the other wheel moves backward an equal amount.

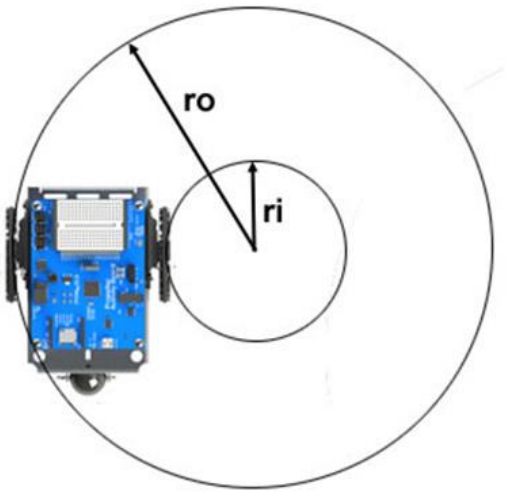
1. Use the procedure for a pivot turn to find the number of wheel ticks required for the turn.
2. Divide the number of wheel ticks by 2._____
3. Use the result of the division as the number of ticks for each wheel. If the number of total ticks required is an odd number, one wheel must get one tick more than the other wheel (you can't specify fractions of a tick).

For a 90 degree right turn: **drive_goto(26, -25)**

For a 90 degree left turn: **drive_goto(-25, 26)**

Calculations for an arc turn using drive_speed() or drive_ramp()

1. Select an outside radius in millimeters for the turn (ro)_____
2. Subtract 106 from the outside radius to find the inside radius (ri)_____
3. Divide ri by ro (ri/ro)_____ (radii ratio)
4. Select a wheel speed for outboard wheel (maximum allowed is 128)_____
5. Multiply the speed of the outboard wheel by the radii ratio (step 3) to find the speed of the inboard wheel_____



Once you have the speeds for the outboard and inboard wheels, you are ready to write your code.

Suppose you want to do an arc turn with an outside radius of 300 mm with an outboard wheel speed of 128. Then the inboard wheel speed would be 83.

For a right arc turn: **drive_speed(128, 83)**

For a left arc turn: **drive_speed(83, 128)**

Note: if you select an outside radius (ro) of 106 mm, then the arc turn becomes a pivot turn because the inboard wheel will have a radius of zero (ri = 0). In that case, there is no need for any calculations. Just select the speed you want for the outboard wheel and the speed for the inboard wheel will be zero

For a pivot turn using drive_speed or drive_ramp, here is code:

For right pivot turn: **drive_speed(128, 0)**

For left pivot turn: **drive_speed(0, 128)**

For zero radius turns usind drive_speed or drive_ramp, here is code:

For right zero radius turn **drive_speed(128, -128)**

For left zero radius turn **drive_speed(-128, 128)**

In above four code examples, the turn rate is set to maximum speed of 128. If you want the robot to run at a slower speed, then use the desired speed, but use negative values where listed and the zero values where listed.

Calculations for an arc turn of a specific number of degrees

The calculations given on page 2 allow you to code for an arc turn, but not for a specific number of degrees. In fact, your robot would run in a circle indefinitely if no other drive code was added. You can approximate a turn in degrees by adding a **pause()** on the line after the drive code, specifying the amount of time to make a turn. But a more accurate approach is to use the function **drive_getTicks()** to count the amount of wheel rotation during the turn. That is the approach used in this set of calculations.

1. Use the calculation sheet on page two to find the speeds for outboard and inboard wheels: outboard speed_____ inboard speed _____
2. Calculate the circumference of the outboard wheel circle (multiply ro by 6.283) circumference _____mm
3. Divide the circumference by 1170, which yields the number of wheel ticks per degree turn _____ wheel ticks/degree
4. Multiply number of wheel ticks/degree by the number of degrees for the desired turn _____wheel ticks to turn desired number of degrees

In order to write the code for the turn, you need three values: speed of outboard wheel, speed of inboard wheel, number of wheel ticks for outboard wheel. After you have completed the calculations above and on page 2, you will have all three values available. Here is an example code snippet for a left arc turn of 90 degrees with an outside radius of 163 mm, which requires 79 ticks for outboard wheel:

```
int ticksLeft[2];                                   //create variable array to store wheel ticks of left wheel
int ticksRight[2];                                  //create variable array to store wheel ticks of right wheel

int main()
{
  drive_getTicks(&ticksLeft[0], &ticksRight[0]);    //get current number of ticks traveled for each wheel
  ticksRight[1] = 0;                                //insures that ticksRight[1] is low number before while()
  ticksLeft[1] = 0;                                 //insures that ticksLeft[1] is low number before while()

  while(ticksRight[0] + 79 > ticksRight[1])         //stop driving in circle after 79 wheel ticks right wheel
  {
    drive_ramp(45, 128);                             //do arc turn with outside radius of 163 mm
    pause(20);                                       //wait 20 milliseconds and then repeat while loop
    drive_getTicks(&ticksLeft[1], &ticksRight[1]);  //get current number of ticks traveled for each wheel
  }

}
```

The three values you have calculated for your turn must be substituted in the above code. The value of wheel ticks replaces the value of 79 in the **while()** argument. The wheel speeds replace the values 45 and 128 in the **drive_ramp()** argument. Keep in mind that for a right turn, the left wheel must have the faster speed and for a left turn it is the right wheel that must have the faster speed. Also, if you are making a right turn, then use this for while code: **while(ticksLeft[0] + 79 > ticksLeft[1])** Again, replacing the value of 79 with the number of wheel ticks required for outboard wheel.

The above code alone will not result in an accurate turn unless the robot is traveling at the speed of the outboard wheel just prior to and after the turn. This is due to errors caused by acceleration and deceleration into and out of the turn when using drive_ramp().