Experiment #5 Acceleration

Objective

In experiment #4 you saw the results of driving pairs of wheels at different speeds. You learned that the robot does not move in a straight line over long distances and this can be compensated by small differences in speed control. You also learned that you can program a desired radius of curvature by varying the drive speed of side motors.

The objective of experiment #5 is to develop library functions that can drive the robots at speeds exceeding the stable starting speeds you have become familiar with. To reach high speeds with direction control the robot must be accelerated in gradual steps until the desired velocity is attained.

Acceleration

The basic math and physics of acceleration are straightforward. Under uniform acceleration the velocity at any time is the product of its acceleration and the time duration of the acceleration, v=at. The step acceleration is slightly more complicated, the wheels attempt to step at a rate determined by the delay time between steps. If the wheels attempt to step faster than the robot itself is moving a net force is applied through the wheels to accelerate the robot. If the step speed force exceeds the friction limits of the wheels the wheels will slip and direction control will be lost.

The acceptable acceleration step limit can be found with simple forward motion by adjusting the delay until maximum speed is obtained with no observable loss of direction control. This is somewhere between 1000 and 1600 microseconds per step. Actual velocities can be measured with a stop watch and the actual step delay of your motion can be calculated.

In theory, the acceptable step limit acceleration can be applied at any time the robot is moving stably at its commanded velocity so the robot can be commanded to accelerate from stop to 1x to 2x, 3x, etc. The acceleration limit is also time dependent. Since we are stepping our acceleration, each step must be held long enough for the robot to accelerate and reach its stable speed. Because the robot will be moving at different speeds the acceleration time intervals will cover increasing distances as the robot accelerates.

Acceleration function algorithm

The acceleration function is straightforward. The variable speed function is called repeatedly with recalculated delays and distances. The intermediate step delay is the division of the initial step delay by the integer count of the step. The intermediate distance is the minimum dead stop distance multiplied by the integer count of the step. i.e.

Step 1 Delay = 1000, distance = minStartDist
Step 2 Delay = 1000/2, distance = 2 * minStartDist
Step N Delay = 1000/N. distance = N x minStartDist

Where minStartDist is a global constant close to .2”

The process can be repeated for as many steps as desired. Actual delays and distances should be empirically determined. The number of steps determine the total distance required for acceleration and deceleration.

Procedure

Part one – measuring dead stop limits

1. Call your variable speed function in setup()
    a. Forward, straight, 72 inches
    b. Standard delay
2. Run robot and observe for any deviations from guided path
3. Decrease delay and repeat
4. Repeat 1-3 until observable deviations occur
5. Record results

Part two

1. Using the delay results from part one
2. Program an eight step acceleration profile
    a. 8 step acceleration
    b. Residual distance at full speed
    c. 8 step deccelleration
3. Compile and load program
4. Set robot on blocks, energize power, run one cycle
5. Align robot on board
    a. Right wheels aligned to painted stripe
    b. Align along long axis of board
    c. Start at one end
6. Push Arduino reset button and observe motion of robot
    a. Does the robot jerk when it starts? Solution – slow it down (Delay)
    b. Does it run straight or does it curve?
    c. How much and in which direction? Solution – small adjustments (<1%) to speed ratio
    d. Repeat 4 – 6 times to compensate for random alignment and starting error
7. Adjust starting distances and repeat steps 3-6 until optimum performance is obtained
    a. Optimum performance
        i. minimal drift
        ii. must be repeatable (average of 4 – 6 passes)

Source code follows!

The function “acceleration” is an implementation of the constant interval linear acceleration algorithm. It takes 4 arguments, the three basic motion variables; dir, dist, del and an interger N specifying the desired number of acceleration steps. If the number of desired acceleration steps requires more acceleration and deceleration distance than the specified dist the function recalculates the number of acceleration steps.

```
void acceleration(byte dir, float dist, long del,int N){
  float total_dis = N*(N+1)/2 * 3 * start_dis; // check commanded distance and number of steps
  if(total_dis > dist){
          float m = sqrt((dist/start_dis) * 2/3); // correct if necessary
          N = m;
                                }

  float mid_dis = dist-start_dis*3*N*(N+1)/2;

 for(int i =1; i<=N;i++){
          vars(dir, start_dis * i, del/i, straight, rightmotors, leftmotors);
                                }

  vars(dir, mid_dis, del/N, straight, rightmotors, leftmotors);

 for(int i=N; i>0; i--){
            vars(dir, start_dis *2*i, del/i, straight, rightmotors, leftmotors);
                                }
                                                                        }
```