Experiment #4 Variable speed drive

Objective

In experiment #3 you saw the results of driving single pairs of wheels. The 45 degree, arc, and pivot motions are the hybrid motion results. The three hybrid motions are tunable. The 45 degree direction can be adjusted to any desired straight line direction. The radius of curvature of the arcs and pivots can also be adjusted. These adjustments are accomplished by driving the two associated pairs of wheels at different speeds rather than driving a single pair. Driving the single pair as you did in experiment #3 is simply a degenerate case of variable speed drive

The objective of experiment #4 is to develop library functions that simultaneously drive different pairs of motors at different speeds and study the effects of directions and speed ratios. The results have some surprises

Variable speed

To move the wheels at different speeds you must send them pulses at different rates. This can be done in many ways but a simple algorithm is given here. Source code is given below.

Motion function algorithm

```
Initialize delays for each different required speed – master/slave
Set master, slave, and step counters to 0
Loop until master wheel has completed its' commanded steps
        Compare master step count to master delay
                Greater than or equal
                        step motors,
                        set master wheel step count to 0
                        increment step counter
        Compare slave step count to slave delay
                Greater than or equal
                        step motors
                        set slave wheel step count to 0
        Increment master wheel step count
        Increment slave wheel step count
End loop
```

Calling variables

1. Delay – master step count
2. SpeedRatio – slave step count = Delay/SpeedRatio
3. Distance – distance of master wheels travel. It must be converted to step count.
4. FWD/REV – direction argument

Procedure

Part one

1. Program a variable speed function for forward motion
2. Call the function in setup()
    a. Forward motion
    b. Right side to left side speed ratio of 2:1
    c. Distance of 45 inches
    d. Master delay of 500 cycles
3. Compile and load the program
4. Set the robot up on the calibration board
    a. Align centerlines of robot to gridline crossing on board
    b. The robot is turning right so set it up with two feet of clearance on its right side
5. Run the program
6. Adjust the controlling parameters
    a. Adjust speed ratio to control radius of curvature (ROC)
    b. To control arc length balance distance and radius of curvature
7. Repeat step 6 until you obtain a 180 arc with a 12 inch radius of curvature

Part two

8. Call the variable speed function in setup()
    a. Forward motion
    b. Speed ratio 1:1
    c. Distance 72 inches
    d. Master delay 500 cycles
9. Compile and load program
10. Set robot on blocks, energize power, run one cycle
11. Align robot on board
    a. Right wheels aligned to painted stripe
    b. Align along long axis of board
    c. Start at one end
12. Push Arduino reset button and observe motion of robot
    a. Does the robot jerk when it starts? Solution – slow it down
    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
13. Adjust and repeat step 12 until optimum performance is obtained
    a. Optimum performance
        i. minimal drift
        ii. must be repeatable (average of 4 – 6 passes)
14. byte leftmotors = B????????,

The function vars takes six arguments, dir – direction byte, dist – distance to move, del – step delay, ratio – speed ratio between master and slave wheel pairs, master – motion byte for master pair, slave – motion byte for slave wheels. The execution time of the while loop controls the actual stepping rate. The del variable gives the number of while loops cycles are required for a single step

```
void vars( byte dir, float dist, long del, float ratio, byte master,byte slave){
  PORTL = dir;
  float stepf=dist*steps_per_inch;
  long steps=stepf;

 long masterCount =0;
 long slaveCount =0;
 long stepCount =0;

 float temp = del * ratio;
 long slaveDelay =  temp;

 while(stepCount < steps){
    if(masterCount > del){
       PORTL ^= master;//Step masterwheels
       masterCount = 0;
       stepCount++;
    }
    if(slaveCount > slaveDelay){
      PORTL ^= slave;//step slave wheels
      slaveCount = 0;//rest
    }
    masterCount++;
    slaveCount++;
 }

}
```