Botball Lesson Plan

Title: Moving the robot in a straight line

Concept / Topic to Teach: Driving the robot forward and backwards

Standards Addressed:

Goal:
By the end of this activity, students will be able to drive the robot forwards and backwards using mtp (move to position), mav (move at velocity).

Anticipatory Set:
This is important because basic movement commands are the simplest way to navigate a robot and form the foundation of more advanced techniques.

Time Required:

Required Materials: Computer with KISS-C, Demo bot, download cable

Activity Procedure:
1. Open KISS-IDE
    a. Target: CBCv2
    b. New Program
2. Watch Video
3. Try it out
    a. mtp
    b. mav

Assessment:
Tape two sheets of paper on the table a set distance from each other, allow students to measure the distance, students must start the robot on the first sheet of paper, and park on the second sheet paper.

Extension Activities:
Have each of the students drive the robot from one piece of paper to the other.
After all the students have made an attempt, cut the sheet of paper in half.
Students that successfully completed the last challenge attempt the new paper size.
Repeat until one student remains, they win!

Do the above, but make the robot return to its starting position

Basic Motor Movement Handout

clear_motor_position_counter(motor port #);
resets the value of that port to 0

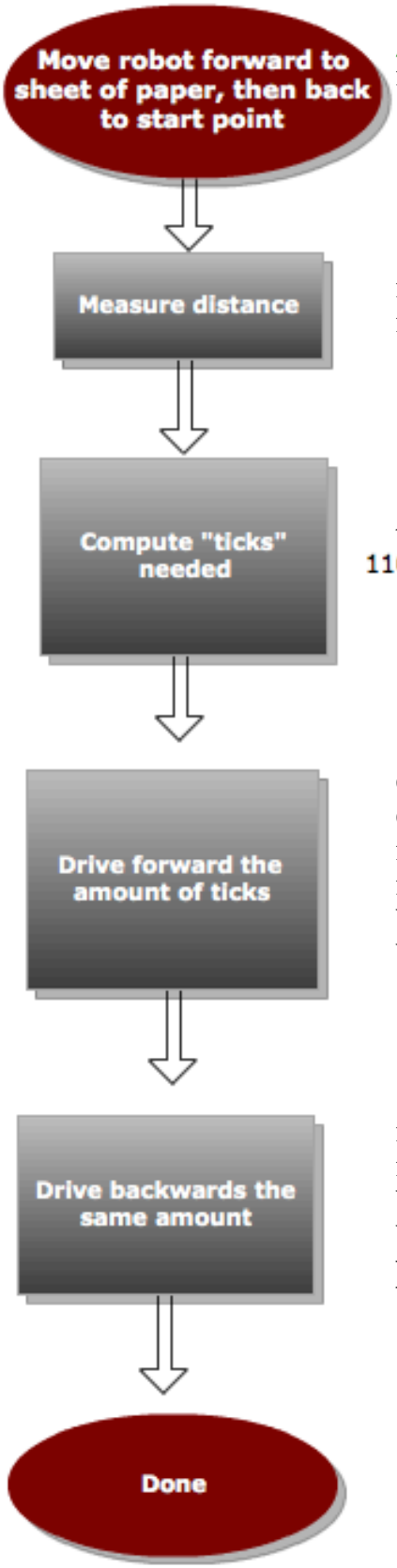
mtp(motor port # , speed , desired position);
motor port # : corresponds to the motor port on CBC (four ports labeled 0-3)
speed: measured in ticks/second (range 0-1,000)
desired position: can be negative or positive to go forwards or backwards

must be followed by
bmd(motor port #);
bmd: causes the program to wait until the motor has moved to its final position

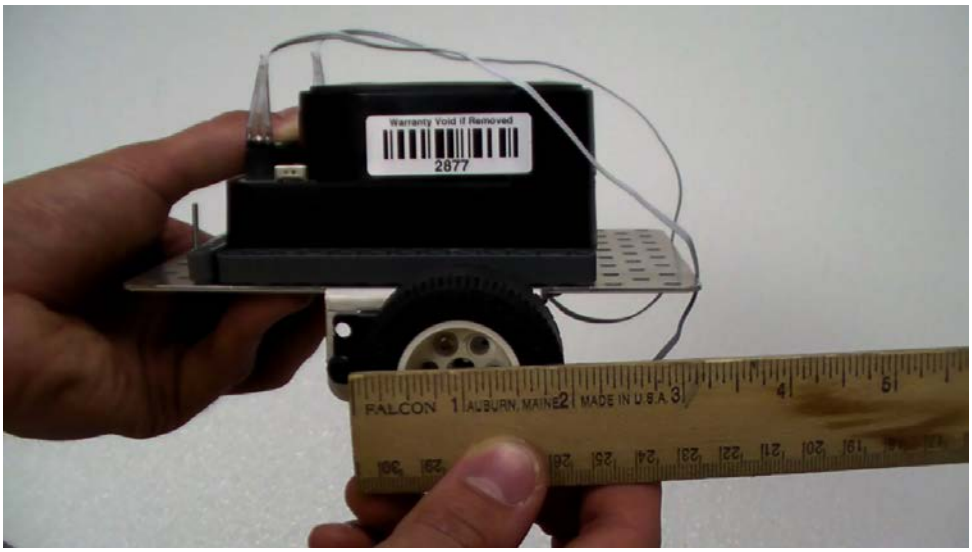
```
1 int main()
2 {//driving a robot forward
3     clear_motor_position_counter(0);
4     clear_motor_position_counter(3);
5     mtp(0,500,1000);
6     mtp(3,500,1000);
7     bmd(0);
8     bmd(3);
9 }
```

mav(port #, speed);
sleep(time in seconds);

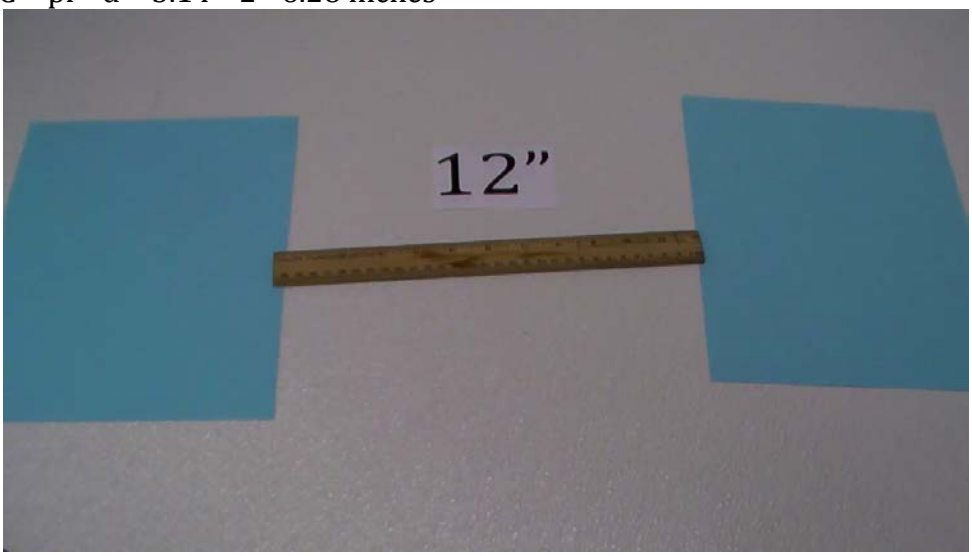
```
1 int main()
2 {//driving a robot forward
3     mav(0,500);
4     mav(3,500);
5     sleep(2.);
6 }
```




```
//driving a robot forward
int main() {//circumference = pi * d ,

int inches=12, ticks;//inches: distance to travel,
int d=2; d= diameter of wheel

ticks= (inches/(3.14159*d)*1100);
1100 ticks = 1 circumference of the wheel

clear_motor_position_counter(0);
clear_motor_position_counter(3);
mtp(0,300,ticks);
mtp(3,300,ticks);
bmd(0);
bmd(3);

mtp(0,300,0);
mtp(3,300,0);
bmd(0);
bmd(3);
}
```



The diameter of the wheel is 2 inches.
The distance the wheel travels in one rotation is the circumference.
C = pi * d = 3.14 * 2 =6.28 inches



Dividing the distance to travel by the circumference determines how many rotations the wheel must turn.
12/6.28 = 1.91 rotations
multiply by 1100 ticks/rotation
=2102 ticks