KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Square Up Using Sensors

Professional Development Workshop

Square Up Using Bump Sensors

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

- Sometimes, it is useful to have a robot “square up” to then drive straight 90 degrees from a “wall”.
- This can be done in a number of ways. One common one is to use two bump (digital) sensors mounted at two “corners” of the back of the robot.
- What follows are **two** possible “algorithms/methods”

Square Up Using Bump Sensors

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Description: Use the pair of touch sensors on the back of the DemoBot to square up on a wall or PVC structure.

Background diagnostic work: You will need to plug your digital button sensors into digital ports. A good strategy might be to use the same port number as your motor port. E.g. right motor plugged into port 0, right button sensor plugged into digital 0.

Key Coding Concepts: Each of the digital sensors will need to be “married” to a wheel in code. One way to handle this is to nest two (if-else pairs) inside of a while loop. Essentially, one of these pairs will control the left wheel and one will control the right wheel.

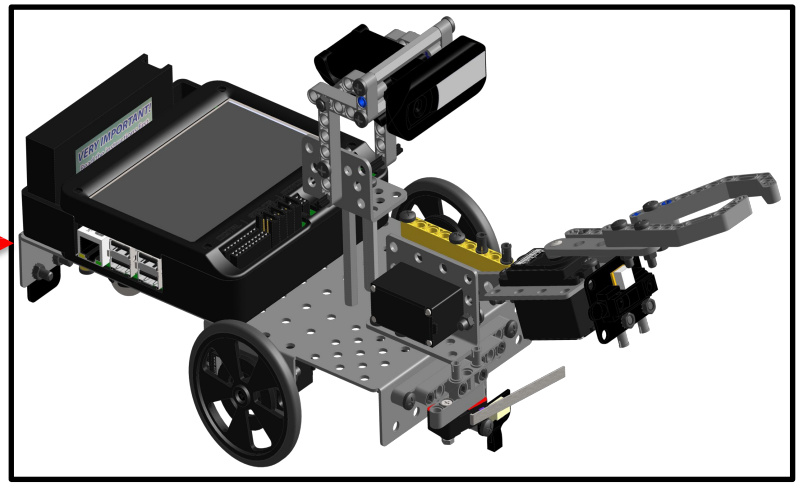
Method #1: The robot will move backward until it senses either back bump sensor is pushed. Upon a sensor being pushed, its corresponding wheel will freeze, the other wheel will continue to move backward until its sensor is pushed. At the point, the robot will exit the loop.

Bonus: Upon completing a square up, your robot will move forward 1000 ticks.

Large Touch Sensors Mounted on Back of Robot

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®



Square Up Method #2

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Description: Write a program that drives backward to orient your robot perpendicular to a “PVC wall”.

Analysis: What is the program supposed to do?

Pseudocode

1. Loop: Both sensors touched?
2. *If* **only** right sensor touched?
3. *Else If* **only** left sensor touched?
4. *Else* drive backward
5. End the program when both touched

Comments

```
// 1. Loop: Are both sensors pressed?

// 2. If right sensor is touched turn CCW

// 3. Else-If left sensor is touched turn
CW

// 4. Else drive backward

// 5. End the program.

// note: CCW means counter clockwise; CW
means clockwise
```

Square Up Method #2 Solution - Digital

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Source Code

```
#include <kipr/wombat.h>

int main()
{
      printf("Back Up to Square Up :-)\n");
      while ((digital(3) == 0) || (digital(0) == 0)) // Left or Right is not pressed
      {
            if ((digital(3) == 0) && (digital(0) == 1)) //Right is pressed (not Left)
            {
                  motor(3, -90);
                  motor(0, 10); // turn CCW backwards with right motor at zero
            }
            else if ((digital(3) == 1) && (digital(0) == 0))
            {
                  motor(3, 10);
                  motor(0, -90); // turn CW backwards with left motor at zero
            }
            else
            {
                  motor(3, -75);
                  motor(0, -75); // just keep going backwards
            }
      }
      ao();
      return 0;
}
```

Assumes that motor 0 and digital 0 are on the right side and motor 3 and digital 3 are on the left side.

Square Up Method #2 Solution - Reflectance

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Source Code

```
#include <kipr/wombat.h>

int main()
{
      printf("Back Up to Square Up :-)\n");
      while ((analog(0) < 1300) || (analog(5) < 1300)) // Left or Right not on black
      {
            if ((analog(0) < 1300) && (analog(5) >= 1300)) //Right not on black
            {
                  motor(3, -50);
                  motor(0, 50); // turn CCW
                  msleep(50);
            }
            else if ((analog(0) >= 1300) && (analog(5) < 1300)) //Left not on black
            {
                  motor(3, 50);
                  motor(0, -50); // turn CW
                  msleep(50);
            }
            else
            {
                  motor(3, 75);
                  motor(0, 75); // just keep going forwards
            }
            motor(0,0);
            motor(3,0);
            msleep(50);
      }
      return 0;
}
```

Assumes that motor 0 and digital 0 are on the right side and motor 3 and digital 5 are on the left side.