KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Driving Using the Gyrometer

Professional Development Workshop

Introduction

- The gyrometer is a sensor that measures **angular velocity**, which we can use to determine how much the robot is turning
- On a Wombat, the gyro sensors are accessed with the `gyro_x()`, `gyro_y()`, and `gyro_z()` functions
- Which axis you use depends on your robot, but for most, with the Wombat facing screen up, `gyro_z()` will be the most useful

Key Terms: Bias

- Your Wombat's `gyro_z()` reading is probably not 0, even when your robot is not moving
- To compensate for this, you must observe your readings and determine the bias, which is the **number you subtract from your gyro reading to make it 0**
- For example, if your reading hovers around 3 < `gyro_z()` < 5, then you might pick 4 for your bias, because that will bring your reading to about 0
- Later we will write a function to automatically calibrate the bias by taking the average of many readings

Key Terms: Deviation

- We will refer to the total amount we have turned away from straight as **deviation**
- In our code we will use a loop to track the total deviation

Motor Speed Adjustment

There are many possible approaches to incorporating the gyro, here is one:

First, calculate a speed delta based on the current deviation:

```
double delta = dev / 5;
```

Which you can then incorporate into your `mav` call:

```
mav(left_motor, speed + delta);
```

The `dev` gets adjusted at the end of each loop:

```
dev += gyro_z() - bias;
```

Gyro Example

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Source Code

```
1  #include <kipr/wombat.h>
2
3  double calibrate_gyro_z();
4  void drive_with_gyro(int speed, double time, double bias);
5
6  int main()
7  {
8      printf("Hello World\n");
9      double bias = calibrate_gyro_z();
10     drive_with_gyro(1000, 15, bias);
11     return 0;
12 }
```

Calibration Function

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Source Code

```
1  double calibrate_gyro_z()
2  {
3      int i, gz, iters = 1000;
4      double bias, total = 0.0;
5
6      for (i = 0; i < iters; i++) {
7          gz = gyro_z();
8          total += gz;
9          msleep(1);
10         printf("Gyro Z: %d\n", gyro_z());
11     }
12     bias = total / iters;
13     printf("New Bias: %f\n", bias);
14     return bias;
15 }
```

Drive Function

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Source Code

```
1  void drive_with_gyro(int speed, double time, double bias)
2  {
3      double start_time = seconds();
4      double delta, dev = 0;
5      int right_motor = 1, left_motor = 0;
6
7      while ((seconds() - start_time) < time) {
8          /* You may need to change this factor
9           * depending on how sensitive your gyro is
10          */
11         delta = dev / 5;
12         mav(right_motor, (-1 * speed) - delta);
13         mav(left_motor, speed + delta);
14
15         msleep(10);
16         dev += gyro_z() - bias;
17     }
18     ao();
19 }
```

Common Problems

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

- Depending on the sensitivity of your gyroscope, you may need to adjust the impact of the deviation by adjusting the number it is divided by
- Make sure that your motors have **opposite signs**! If they don't, your robot will "panic" because it is constantly turning when it thinks it's going straight.