KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Motors - Basic Driving and Turning

Professional Development Workshop

Check the Robot's Motor Ports

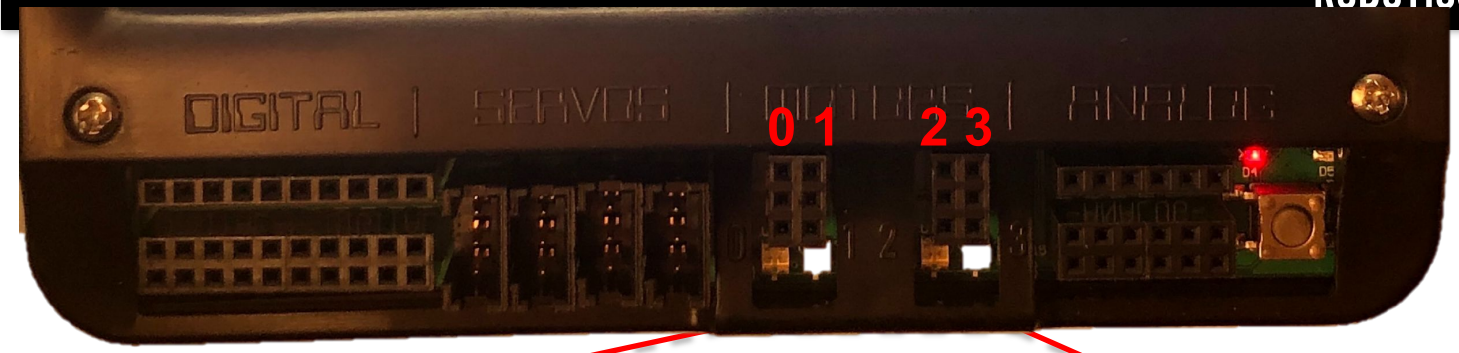
KISS INSTITUTE FOR PRACTICAL ROBOTICS

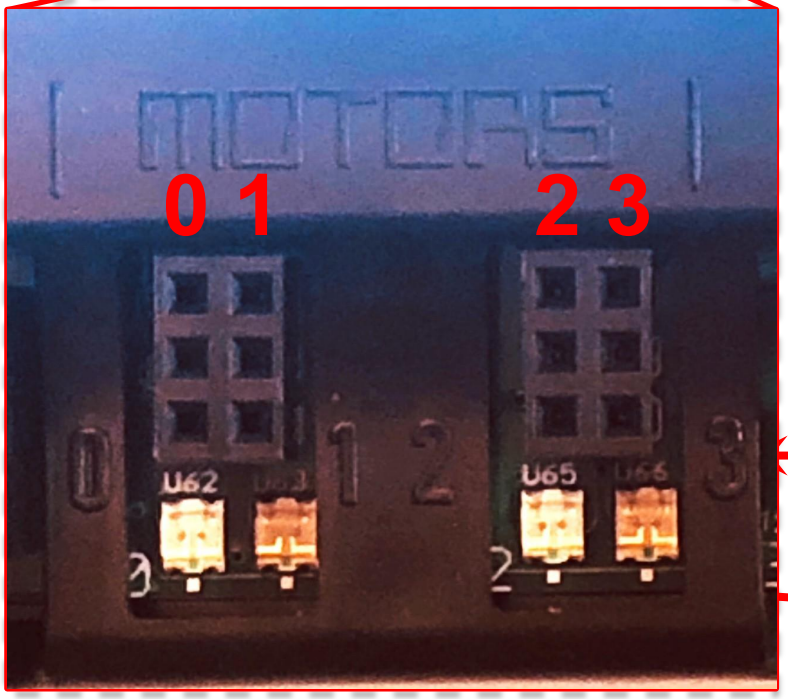
Botball®

- To program your robot to move, you need to know which **motor ports** your motors are plugged into.

- Computer scientists tend to start counting at 0, so the four **motor ports** are numbered **0**, **1**, **2**, and **3**.

Wombat Motor Ports





Motor Port Labels are on the Case

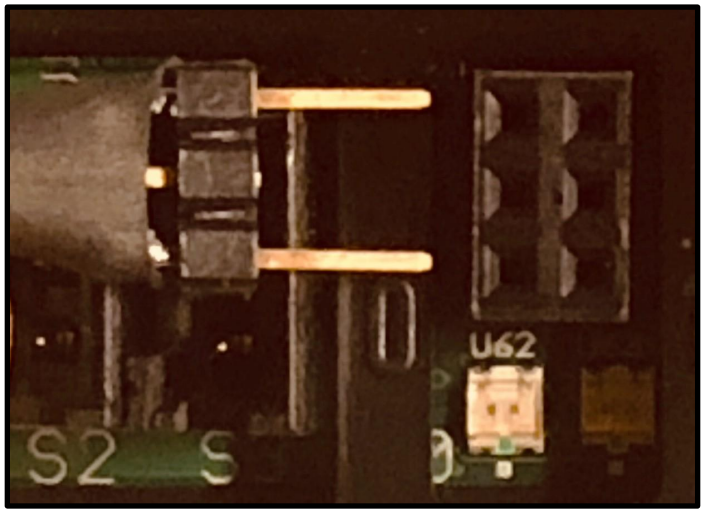
Motor Port Labels 0, 2 are also on the board

Plugging in Motors

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

- **Motors** have red wire and a black wire with a **two-prong plug**.
- The Wombat has 4 motor ports numbered **0** & **1** on left, and **2** & **3** on right.
- When a port is powered (receiving motor commands), it has a light that glows
  green for one direction and **red** for the other direction.
  - Plug orientation order determines motor direction.
  - By convention, **green** is **forward** (**+**) and **red** is **reverse** (**–**)
    - ***Unless you plug in the motors "backwards"***.



Drive motors have a two-prong plug.

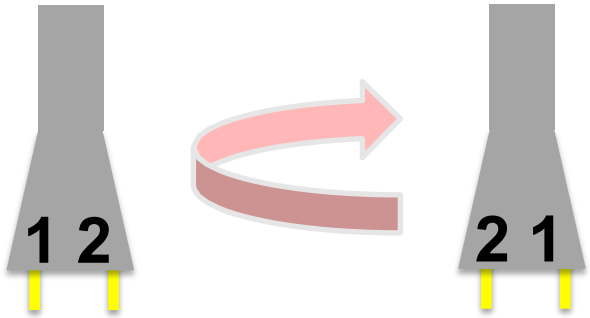
Plugged in Motor

Motor Plugged into Port 0 (right wheel on DemoBot)

Motor Direction

You want your motors going in the same direction; otherwise, your robot will go in circles!

- **Motors** have a red wire and a black wire with a **<u>two-prong plug</u>**.
- You can plug these in two different ways:
  - One direction is clockwise, and the other direction is counterclockwise.
  - The red and black wires help determine motor direction.



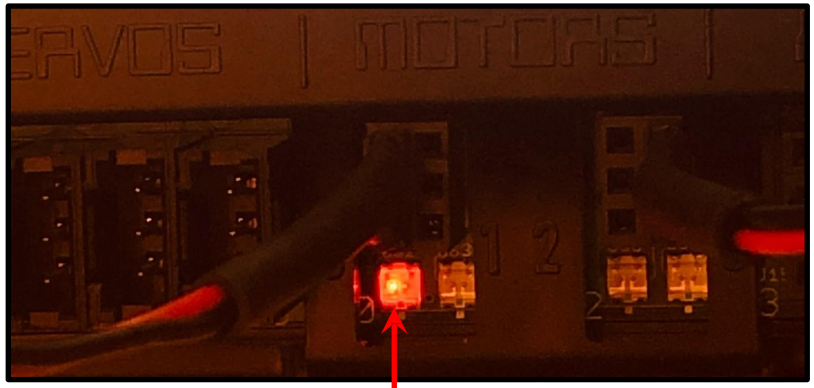
Motor Port and Direction Check

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

There is an easy way to check this!

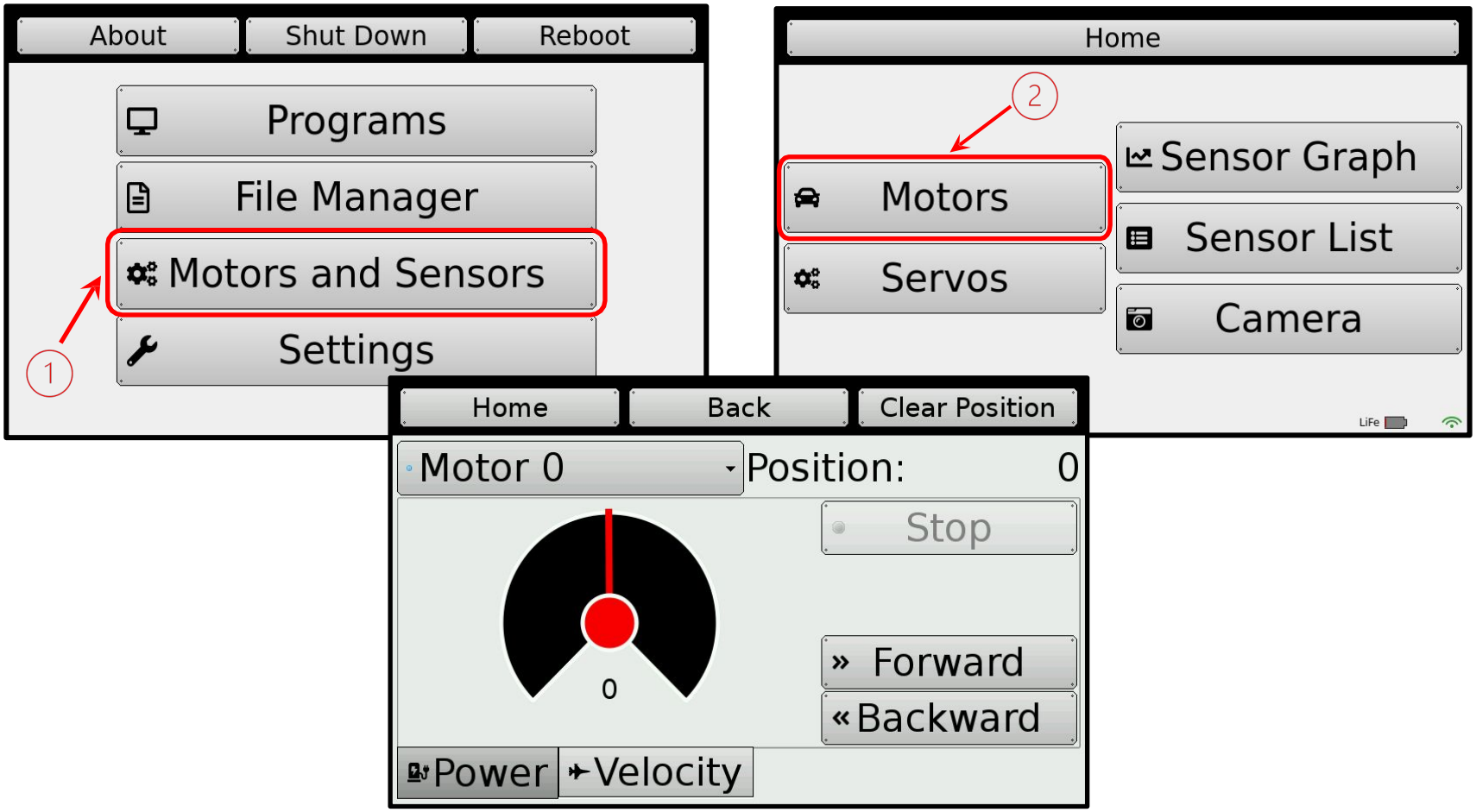
- Manually rotate the tire, and you will see an LED light up below the **motor port** (the **port #** is labeled on the board).
  - If the LED is **green**, it is going **forward** (**+**).
  - If the LED is **red**, it is going **reverse** (**−**).



- Use this trick to check the **port #**'s and **direction** of your **motors**.
  - If one is **red** and the other is **green**, turn one motor plug 180° and plug it back in.
  - The lights should both be **green** if the robot is moving forward.

Use the Motor Widget

1. Select "*Motors and Sensors*"
2. Select "*Motors*"



Common Motor Functions

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

There are several functions for motors.
We will begin with `motor()`

Motor port #
(between **0** and **3**)

```
motor(0, 100);
// Turns on motor port #0 at 100% power.
// Power should be between -100% and 100%.
msleep(# milliseconds);
// Wait for the specified amount of time.
ao();
// Turn off all of the motors.
```

A **positive number** should drive the motor **forward**; if not, rotate the motor plug 180°.

A **negative number** should drive the motor **reverse**.

If two drive motors are plugged in in opposite directions from each other, then the robot will go in a circle.

Moving the DemoBot

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

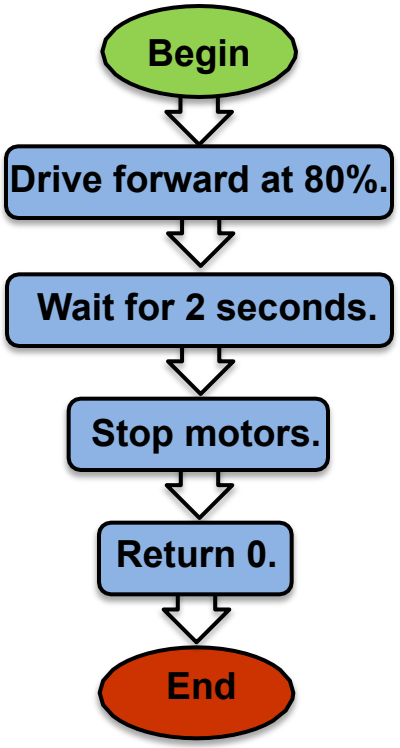
Description: Write a program that drives the DemoBot forward at 80% power for two seconds, and then stops.

Analysis: What is the program supposed to do?

Pseudocode Comments

1. Drive forward at 80%. `// 1. Drive forward at 80%.`
2. Wait for 2 seconds. `// 2. Wait for 2 seconds.`
3. Stop motors.`// 3. Stop motors.`
4. End the program. `// 4. End the program.`

Flowchart



Moving the DemoBot

Solution: Create a **new project**, create a **new file**, and enter your **pseudocode** (as **comments**) and **source code** in the **`main`** function.

- **Note:** remember to give your project and file descriptive, unique names!

Pseudocode (Comments)

```
1.  Drive forward at 80%.
2.  Wait for 2 seconds.
3.  Stop motors.
4.  End the program.
```

Source Code

```
1  #include <kipr/wombat.h>
2
3  int main()
4  {
5      motor(0, 80);
6      motor(3, 80);
7      msleep(2000);
8
9      ao();
10
11     return 0;
12 }
13
```

`//forward` (lines 5–7)

Execution: Compile and run your program.

Moving the DemoBot

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Reflection: What did you notice after you ran the program?

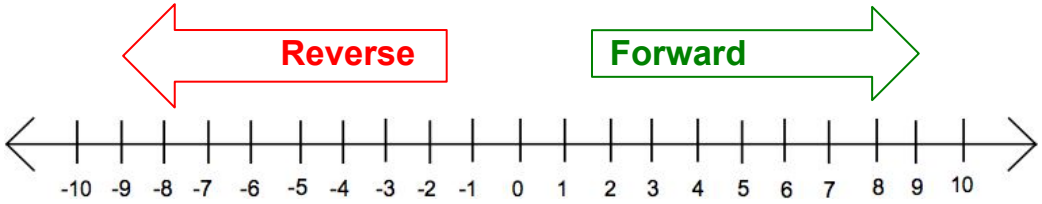
- Did the DemoBot move forward?
  - **Positive** (**+**) numbers should move the motors in a clockwise direction (**forward**); if not, rotate the motor plug 180° where it plugs into the Wombat.
  - If your robot moves in a circle, one motor is either not moving (is it plugged in?) or they are moving in opposite directions (rotate the motor plug 180°).
- Did the DemoBot drive straight?
- How could you adjust the code to make the robot drive straight?
- How can you make the robot drive backwards?
- How can you make the robot turn left or right?

Robot Driving Hints

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Remember your # line: positive numbers (+) go forward and negative numbers (−) go in reverse.



Driving straight: it is surprisingly difficult to drive in a straight line…

- **Problem:** Motors are not exactly the same.
- **Problem:** One tire has more resistance.
- **Problem:** The tires might not be aligned perfectly.

And many, many other reasons…

- **Solution:** You can adjust this by slowing down or speeding up the motors.

Making turns:

- **Solution:** Have one wheel go faster or slower than the other.
- **Solution:** Have one wheel move while the other one is stopped.
- **Solution:** Have one wheel move forward and the other wheel move in reverse (friction is less of a factor when both wheels are moving).

More Motor Functions

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

```
motor(0, 100);
// Turns on motor port #0 at 100% power.
```

- Is great for turning gears or winding up string on a pulley.
- Is not so great for driving robots as it is dependent on battery charge.

```
mav(0, 800);
// Move motor on port #0 at 800 ticks/sec.
```

- Is great for driving robots and not as dependent on battery charge.
- Greater precision of motor control (think of this as being like cruise control in a car).
- Must use `wait_for_milliseconds` function correctly.

```
mrp(0, 800, 3000);
// Move motor on port #0 forward 3000 ticks at 800 ticks/sec.
```

- Provides the most precise level of motor control.
- Most complicated to use (must do a lot of calculations to move correctly).

Other Motor Functions

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Move At Velocity:

```
mav(0, 1000);
```

Motor Port # (between 0 and 3)

Velocity (in ticks per second) between -1000 (reverse) and 1000 (forward)

Move Relative Position:

```
mrp(0, 1000, 3000);
```

Motor Position (in ticks); ~1000 ticks = 1 tire revolution

Move at Velocity

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Description: Write a program that drives the DemoBot forward at 1000 ticks per second for 3 seconds, then in reverse at 1000 ticks per second for 3 second, then stops.

Analysis: What is the program supposed to do?

Pseudocode

1. Drive forward at 1000 ticks/sec.
2. Wait for 3 seconds.
3. Drive reverse at 1000 ticks/sec.
4. Wait for 3 seconds.
5. Stop motors.
6. End the program.

Comments

```
// 1. Drive forward at 1000 ticks/sec.
// 2. Wait for 3 seconds.
// 3. Drive reverse at 1000 ticks/sec.
// 4. Wait for 3 seconds.
// 5. Stop motors.
// 6. End the program.
```

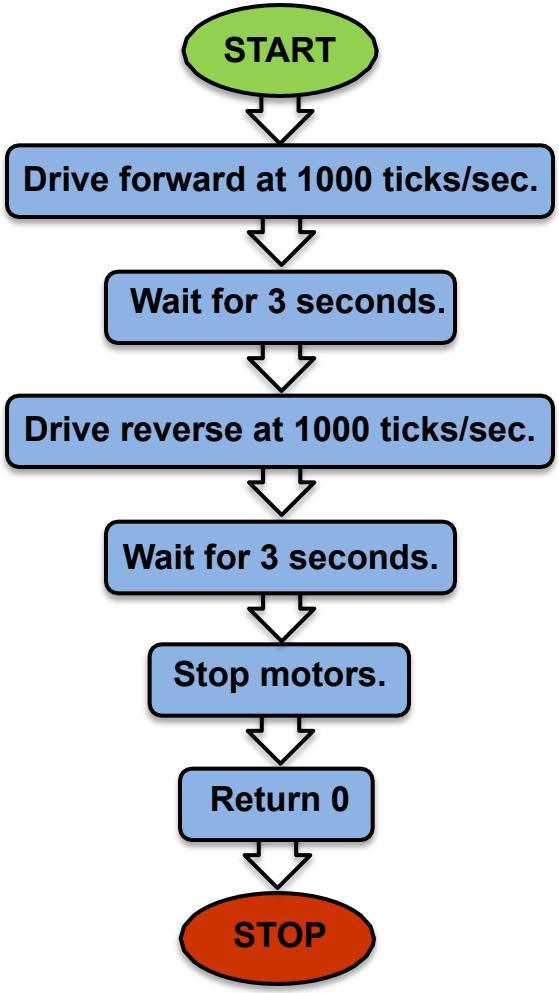
Move at Velocity

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Analysis:

Flowchart



Move at Velocity

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Solution:

Pseudocode (Comments)

```
int main()
{
  // 1. Drive forward at 1000 ticks/sec.
  // 2. Wait for 3 seconds.
  //3.  Drive reverse at 1000 ticks/sec.
  //4.  Wait for 3 seconds.
  // 5. Stop motors.
  // 6. End the program.
}
```

Source Code

```
1  #include <kipr/wombat.h>
2
3  int main()
4  {
5      // 1. Drive forward at 1000
6             ticks/sec
7      mav(0, 1000);
8      mav(3, 1000);
9      // 2. Wait for 3 seconds.
10     msleep(3000);
11     // 3. Drive reverse at 1000
12            tics/sec
13     mav(0, -1000);
14     mav(3, -1000);
15     // 4. Wait for 3 seconds.
16     msleep(3000);
17     // 5. Stop motors.
18     ao();
19     // 6. End the program.
20     return 0;
21 }
```

Execution: Compile and run your program on the Wombat.

Using Mecanum Wheels

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

- Now you have 4 motors/wheels to control
- The same motor functions work

Source Code

```
1  #include <kipr/wombat.h>
2
3  int main()
4  {
5      printf("MecanumWheelsFunctions\n");
6
7      motor(0,100);
8      motor(1,100);
9      motor(2,100);
10     motor(3,100);
11     msleep(3000); // Forward
12
13     ao();
14     msleep(500); // Pause
15     return 0;
16 }
17
```

Using Mecanum Wheels

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Source Code

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

int main()
{
      int RF = 0; // RF is right front
wheel
      int RR = 1; // RR is right rear wheel
      int LF = 2; // LF is left front wheel
      int LR = 3; // LR is left rear wheel
      printf("MecanumWheels\n");
      motor(RF,100);
      motor(RR,100);
      motor(LF,100);
      motor(LR,100);
      msleep(3000); // Drive Forward

      ao();
      msleep(500); // Pause

      motor(RF,-100);
      motor(RR,-100);
      motor(LF,-100);
      motor(LR,-100);
      msleep(3000); // Drive Backwards
      return 0;
}
```

Using Variables to keep track of motors will be helpful

Using Mecanum Wheels

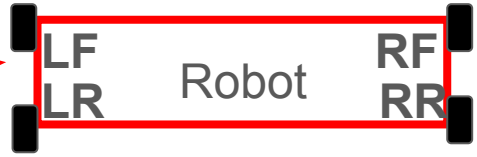
KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Source Code

```
1  #include <kipr/wombat.h>
2
3  int main()
4  {
5      int RF = 0; // RF is right front
6  wheel
7      int RR = 1; // RR is right rear wheel
8      int LF = 2; // LF is left front wheel
9      int LR = 3; // LR is left rear wheel
10
11     printf("MecanumWheels\n");
12
13     motor(LF,100);motor(RF,100);
14     motor(LR,100);motor(RR,100);
15     msleep(3000); // Drive Forward
16
17     ao();
18     msleep(500); // Pause
19
20     motor(LF,-100);motor(RF,-100);
21     motor(LR,-100);motor(RR,-100);
22     msleep(3000); // Drive Backwards
23
24     return 0;
25 }
```

- Placing your motor commands in a square like the robot may also help you keep track of your wheels




```
motor(LF,100); motor(RF,100);
motor(LR,100); motor(RR,100);
```

Using Mecanum Wheels

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Source Code

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

int main()
{
	int RF = 0; // RF is right front wheel
	int RR = 1; // RR is right rear wheel
	int LF = 2; // LF is left front wheel
	int LR = 3; // LR is left rear wheel

	printf("MecanumWheels\n");

	motor(LF,100);motor(RF,100);
	motor(LR,100);motor(RR,100);
	msleep(3000); // Drive Forward

	ao();
	msleep(500); // Pause

	motor(LF,100);motor(RF,-100);
	motor(LR,100);motor(RR,-100);
	msleep(500); // Right Turn

	ao();
	msleep(500); // Pause

	motor(LF,-100);motor(RF,100);
	motor(LR,-100);motor(RR,100);
	msleep(3000); // Left Turn

	return 0;
}
```

- Now you have 4 motors/wheels to control
- Forward, Backward, Radius/Arc, and Pivot Turns are the same as with two wheels

Using Mecanum Wheels

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Source Code

```
#include <kipr/wombat.h>
void L90(); // L90 is left 90 degree turn
void R90(); // R90 is right 90 degree turn
int RF = 0; // RF is right front wheel
int RR = 1; // RR is right rear wheel
int LF = 2; // LF is left front wheel
int LR = 3; // LR is left rear wheel
int main()
{
      printf("MecanumWheelsFunctions\n");

      R90();

      ao();
      msleep(500); // Pause

      L90();

      ao();
      msleep(500); // Pause
      return 0;
}
void L90()
{
      motor(LF,-100);motor(RF,100);
      motor(LR,-100);motor(RR,100);
      msleep(1000);
}
void R90()
{
      motor(LF,100);motor(RF,-100);
      motor(LR,100);motor(RR,-100);
      msleep(1000);
}
```

Write Functions for Your Turns

Moving Lateral / Sideways and Diagonal Movement

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Source Code

```c
#include <kipr/wombat.h>
int RF = 0; // RF is right front wheel
int RR = 1; // RR is right rear wheel
int LF = 2; // LF is left front wheel
int LR = 3; // LR is left rear wheel
int main()
{
      printf("MecanumWheelsFunctions\n");

      motor(LF,50);motor(RF,-50); // Sideways Right
      motor(LR,-50);motor(RR,50);
      msleep(3000);

      ao();
      msleep(500); // Pause

      motor(LF,-50);motor(RF,50); // Sideways Left
      motor(LR,50);motor(RR,-50);
      msleep(8000);

      ao();
      msleep(500); // Pause

      return 0;
}
```

Source Code

```c
#include <kipr/wombat.h>
int RF = 0; // RF is right front wheel
int RR = 1; // RR is right rear wheel
int LF = 2; // LF is left front wheel
int LR = 3; // LR is left rear wheel
int main()
{
      printf("MecanumWheelsFunctions\n");

      motor(LF,0);motor(RF,-50); // 45 Right
      motor(LR,-50);motor(RR,0);
      msleep(3000);

      ao();
      msleep(500); // Pause

      motor(LF,-50);motor(RF,0); // 45 Left
      motor(LR,0);motor(RR,-50);
      msleep(8000);

      ao();
      msleep(500); // Pause

      return 0;
}
```