KISS INSTITUTE FOR PRACTICAL ROBOTICS Botball®

Precision Driving - Motor Position Counters

Professional Development Workshop
KISS Institute for Practical Robotics © 1993 – 2025 KIPR

Precision Driving - Motor Position Counters

Slide	Topic
4	Motor Position Counter
5-6	Seeing Counters on Screen
7-9	Drive to a Specific Point
10-11	Drive to a Specific Point + Backup
12-13	Drive a Set Distance and Back Up to Start
14	Driving a Set Distance with Arguments
15-17	Drive Straight

KISS Institute for Practical Robotics © 1993 – 2025 KIPR

Precision Driving - Motor Position Counters

KISS Institute for Practical Robotics © 1993 – 2025 KIPR

Motor Position Counter

KISS INSTITUTE FOR PRACTICAL ROBOTICS Botball®

Each motor used by the DemoBot has a built-in motor position counter, which you can use to calculate the distance traveled by the robot!

Motor Port # (0 to 3)

```
get_motor_position_counter(0)   — OR —   gmpc(0)
// Tells us the number of ticks the motor on port #0 has rotated.
```

Motor Port # (0 to 3)

```
clear_motor_position_counter(0);   — OR —   cmpc(0);
// Resets the tick counter to 0 for the motor on port #0.
```

- The motor position is measured in "**ticks**".
- Botball motors have ***approximately* 1800 ticks per *revolution***.
- Use **wheel circumference divided by 1800** to calculate distance!

Similar to how a clock is divided into 60-second intervals (ticks).

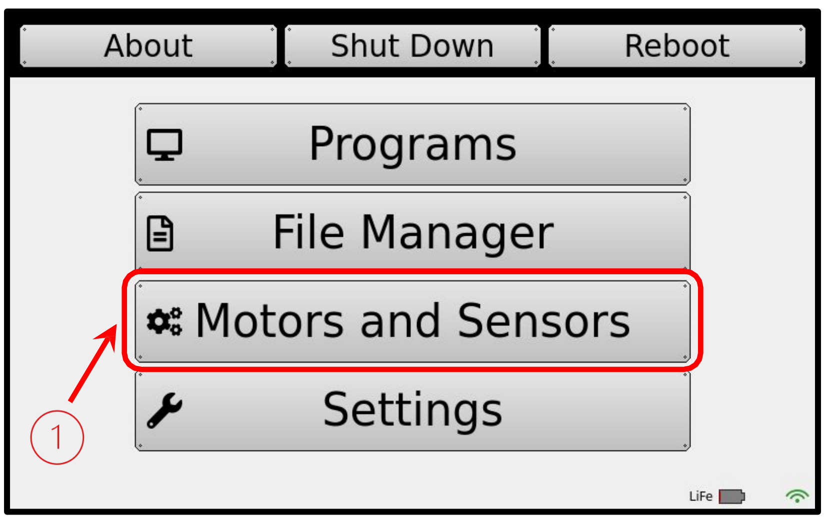
KISS Institute for Practical Robotics © 1993 – 2025 KIPR

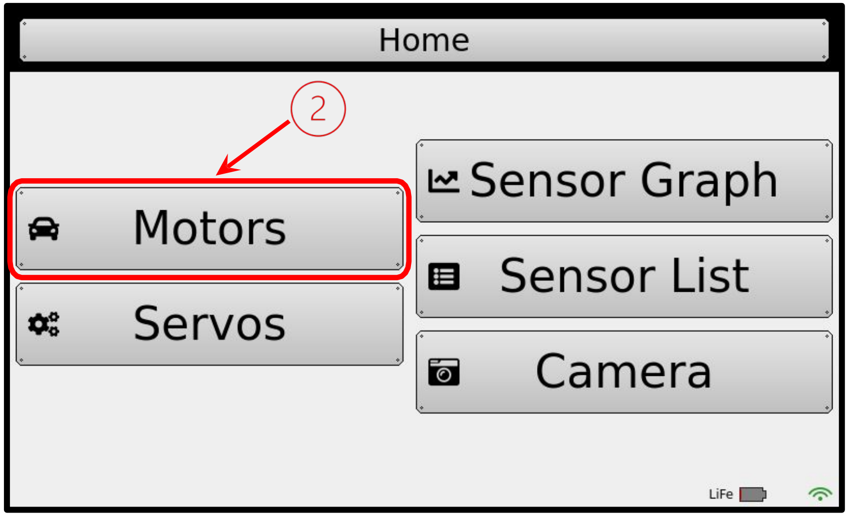
Seeing Counters on Screen

KISS INSTITUTE FOR PRACTICAL ROBOTICS Botball®

You can access the Motors from the Motors and Sensors section

- This is very helpful to test your motors and see the actual motor position counters "*in action*"





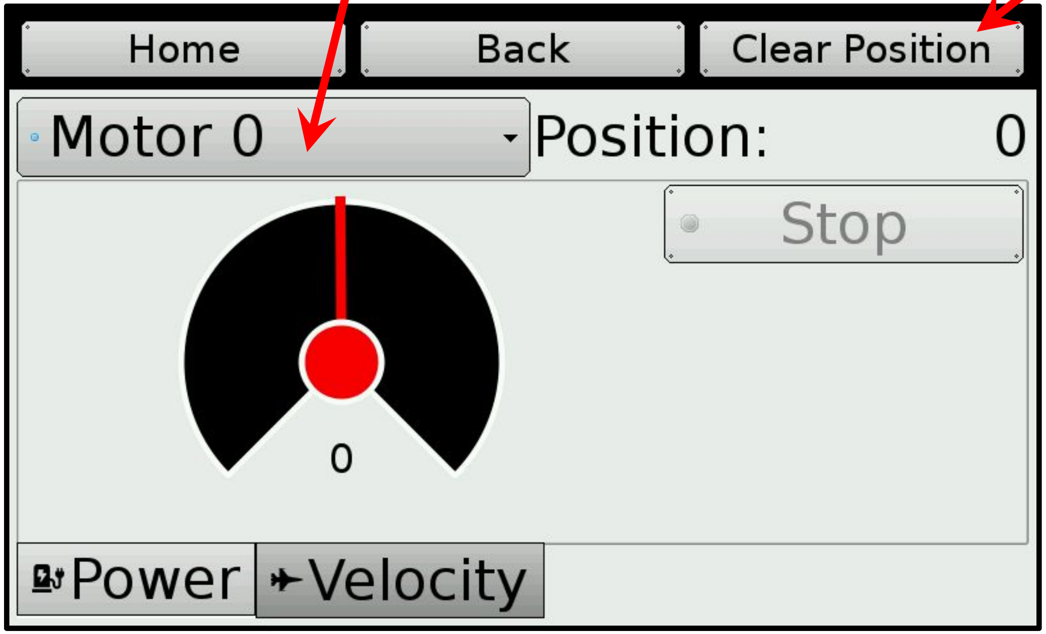
KISS Institute for Practical Robotics © 1993 – 2025 KIPR

Seeing Counters on Screen

Select motor port (allows you to select the motor of your choice)

To clear (reset) the counter

Motor Position in "ticks"



Use your hand to rotate the robot's wheel (plugged into port 0) and watch the position counter.

What happens if you turn the wheel in the opposite direction?

You can also place your robot on a surface and roll it forward to measure the # ticks from a starting position to another location or object

KISS Institute for Practical Robotics © 1993 – 2025 KIPR

Drive to a Specific Point

KISS INSTITUTE FOR PRACTICAL ROBOTICS Botball®

You can also place your robot on a surface and roll it forward to measure the # ticks from a starting position to another location or object.

Place the robot in the *start box* of **KIPR Mat A** and using the motors/widget screen:

1) reset the left motor counter
2) manually push the robot forward to *circle 9* on the mat
3) visually record/remember the tick count

Description: Write your program to drive the DemoBot forward that many “ticks” and then stop.

Pseudocode

Generate it!

KISS Institute for Practical Robotics © 1993 – 2025 KIPR

Drive to a Specific Point

Solution:

Pseudocode

1. Reset motor position counter.
2. Loop: Is counter < my distance?
3. Drive forward.
4. Stop motors.
5. End the program.

Source Code

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

int main()
{
    int distance = 4500; // in ticks

    cmpc(0);

    while (gmpc(0) < distance)
    {
        motor(0, 50);
        motor(3, 50);
    }
    ao();

    return 0;
}

```


KISS Institute for Practical Robotics © 1993 – 2025 KIPR

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

- How far did the robot travel? Was it always the same (you tested it more than once, right)?
  - Your robot most likely went FURTHER than you programmed it to (check the motors screen after it stops to see the actual final tick count). Why? Hint: inertia
  - Change your loop so that it actually goes to “distance - (actual - desired)”:

```
while(gmpc(0) < distance - (4832 - distance))
```

- How could you modify your program to travel a specific distance in millimeters? (**Hint:** Use **wheel circumference (in mm) divided by 1800** to calculate number of mm per tick!)

KISS Institute for Practical Robotics © 1993 – 2025 KIPR

Drive to a Specific Point + Backup

KISS INSTITUTE FOR PRACTICAL ROBOTICS Botball®

Description: Write your program to drive the DemoBot forward to a specific point and then back up to where you started.

Pseudocode

1. Drive forward.
2. Stop at specific distance
3. Drive backwards.
4. Stop at starting point.

Comments

// 1. Drive forward.
// 2. Loop: Is motor position at specific count?
// 3. Drive Backwards to specific distance.
// 4. End the program.

KISS Institute for Practical Robotics © 1993 – 2025 KIPR

Drive to a Specific Point + Backup

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Solution:

Now back up to position (tick count 0).

Note: clear counter not needed ***this*** *time*

Source Code

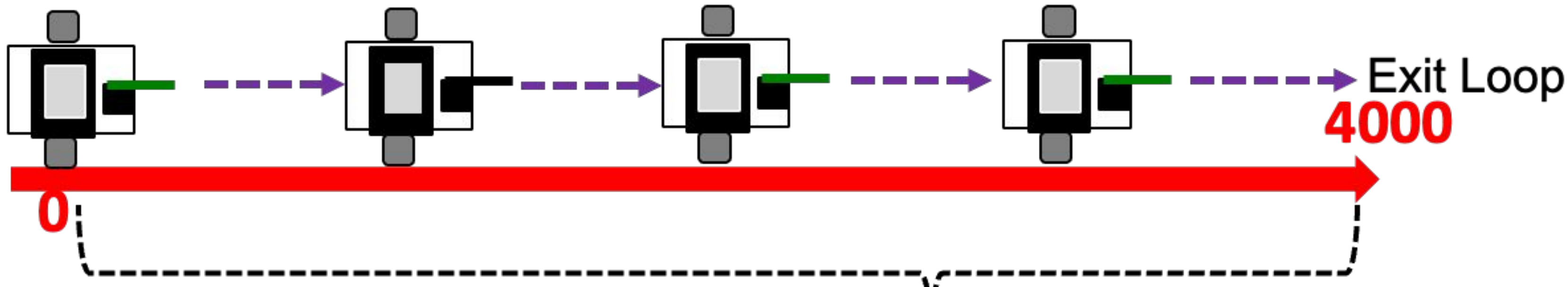
```
#include <kipr/wombat.h>

int main()
{
	int distance = 4500; // in ticks
	cmpc(0);
	while (gmpc(0) < distance)
	{
		motor(0, 50);
		motor(3, 50);
	}
	ao();
	msleep(2000); // see it stop?
	while (gmpc(0) > 0)
	{
		motor(0, -50);
		motor(3, -50);
	}
	ao();
	return 0;
}
```


KISS Institute for Practical Robotics © 1993 – 2025 KIPR

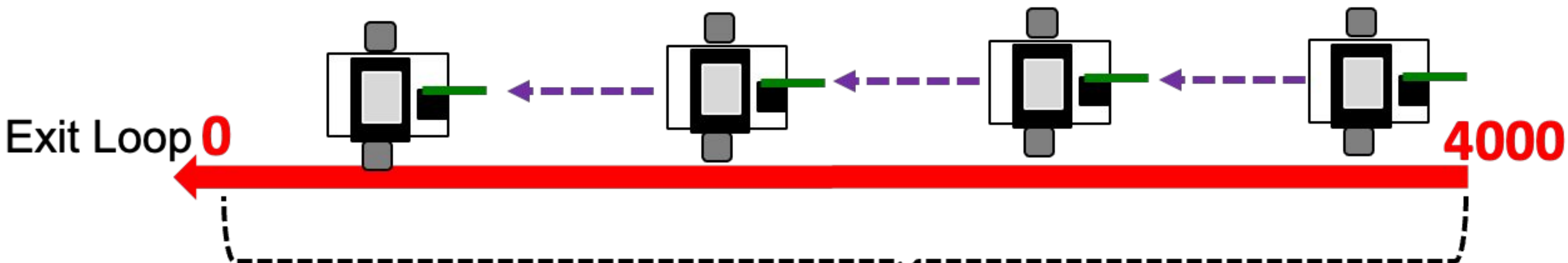
Drive a Set Distance and Back Up to Start

KISS INSTITUTE FOR PRACTICAL ROBOTICS Botball®

```
cmpc (0)
```




```
while (gmpc (0) <= 4000)
```




```
while (gmpc (0) >= 0)
```


KISS Institute for Practical Robotics © 1993 – 2025 KIPR

Drive a Set Distance and Back Up to Start

KISS INSTITUTE FOR PRACTICAL ROBOTICS Botball®

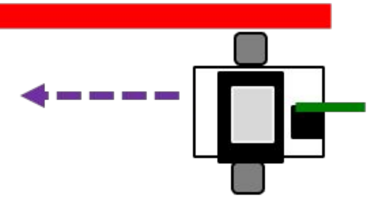
Activity:

1. Clear motor position counter.
2. Find the ticks to can on circle 9.
3. Account for the momentum. Do not clear mpc.
4. Drive set distance forward using motor position (while loop).
5. Next, back up to start position of 0 ticks (while loop).

Guided Questions:

What will your forward # be?

What will your final # be?



KISS Institute for Practical Robotics © 1993 – 2025 KIPR

Driving a Set Distance with Arguments

Source Code

```
#include <kipr/wombat.h>
void Drive(int Lpower, int Rpower, int distance);
int main()
{
     Drive (100,100,5000);

      motor (0,0);
     motor (3,0);
     msleep (500);

     return 0;
}
void Drive (int Lpower, int Rpower, int distance)
{
     cmpc(0);
     while (gmpc(0) < distance)
     {
         motor (0,Lpower);
         motor (3,Rpower);

     }
```

This Allows you to set your motor power and distance within the int main

KISS Institute for Practical Robotics © 1993 – 2025 KIPR

Drive Straight!

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

- Did the robot go straighter than in the previous program?
- How could you use this technique whenever you wanted to drive straight? (**Hint:** Consider writing a function with an argument for the distance.)
- How could you modify your program to go straight at different speeds?

KISS Institute for Practical Robotics © 1993 – 2025 KIPR

Drive Straight!

Description: Write a program that drives the DemoBot straight for 14000 ticks by adjusting the right motor power so that the position of the left motor is the same (or close) to the right.

Analysis: How can you adjust the left motor's position?

Pseudocode

```
1. Clear both motor counters
2. Loop: If total distance < 14000
         Move left motor 75% power
     If: Right is behind left
         speed up right
     Else:
         slow down right
3. Stop motors
4. End the program
```


KISS Institute for Practical Robotics © 1993 – 2025 KIPR

Solution:

Pseudocode

1. Clear both motor counters.
    Loop: check left position
    Power left motor at 75%.
    If: slower
        right motor at 100%
    Else: faster
        right motor at 50%
3. Stop motors.
4. End the program.

Source Code

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

int main()
{
	cmpc(3);
	cmpc(0);
	while (gmpc(3) < 14000)
	{
		motor(3, 75);
		if(gmpc(0) < gmpc(3))
		{
			motor(3, 100);
		}
		else
		{
			motor(3, 50);
		}
	}
	ao();
	return 0;
}

```


KISS Institute for Practical Robotics © 1993 – 2025 KIPR

Precision Turning

Description: Write a program that turns left 90 degrees and then turns right 90 degrees using motor position counter.

Hint: Remember how we manually moved our robots to find the correct position, and that inertia needs to be accounted for…

Pseudocode

1. Turn left 90 degrees.
2. Stop
3. Turn right 90 degrees.
4. Stop at same orientation as start.

Start "small" (try to accomplish the first turn before adding in / working on the second one)

KISS Institute for Practical Robotics © 1993 – 2025 KIPR

Turning Left 90 Degree with MPC

KISS INSTITUTE FOR PRACTICAL ROBOTICS Botball®

Source Code

```c
#include <kipr/wombat.h>
void L90 ();
int main()
{
     L90(); //L90 is a left 90 degree turn

      motor (0,0);
     motor (3,0);
     msleep (500);

     return 0;
}
void L90 ()
{
    cmpc(0);
    while (gmpc(0) < 1090)
    {
        motor (0,50);
        motor (3,-50);

    }
```

the 1090 is the number of tics for your robot to complete a 90 degree turn. Your number may be different

Turning Right 90 Degree with MPC

Source Code

```c
#include <kipr/wombat.h>
void R90 ();
int main()
{
      R90(); //R90 is a left 90 degree turn

       motor (0,0);
      motor (3,0);
      msleep (500);

      return 0;
}
void R90 ()
{
     cmpc(3);
     while (gmpc(3) < 1090)
     {
          motor (0,-50);
          motor (3,50);

     }
```

the 1090 is the number of tics for your robot to complete a 90 degree turn. Your number may be different

KISS Institute for Practical Robotics © 1993 – 2025 KIPR

Turning Any Degree with MPC

Source Code

```
#include <kipr/wombat.h>
void LTurn (int Lpower, int Rpower, float degrees);
int main()
{
    LTurn(50,50,90);

     motor (0,0);
    motor (3,0);
    msleep (500);

    return 0;
}
void LTurn (int Lpower, int Rpower, float degrees)
{
  cmpc(0);
  while (gmpc(0) < (degrees * 12.11)
  {
    motor (0,Lpower);
    motor (3,- Rpower);

  }
```

This Allows you to make your left turn at any degree with whatever motor power you decide

In previous activities we came up with 1090 ticks for a 90 degree turn. (1090*4)/360 = 12.11 tics per degree. We then multiply this by the number of desired degrees put into the LTurn function call. This is a Float data type because 12.11 * Degrees with be a decimal (float)

KISS Institute for Practical Robotics © 1993 – 2025 KIPR