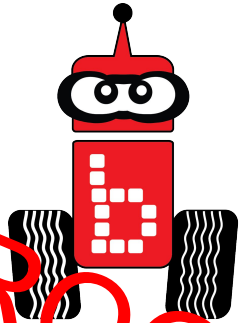


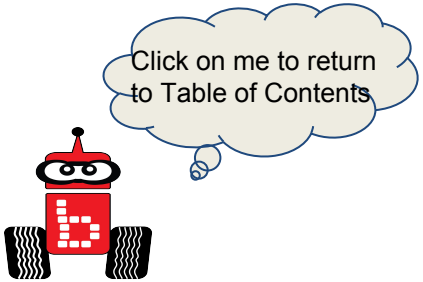
IN PROGRESS

Counters With Servos and more

- **Key Concepts**
  - **Students will learn how to program their robots to be able to move using a counter**
- **Pacing**
  - **30-45 minutes per Activity**

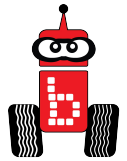
KISS INSTITUTE FOR PRACTICAL ROBOTICS

Table of Contents
Page 1 of 2



KISS INSTITUTE FOR PRACTICAL ROBOTICS

Moving Your Robot Using the `counter()` Function



Goal:

- Students will familiarize themselves with the functions `msleep()` and `motor()`
- Students will understand how to move their robots in the following manner: forwards, backwards, straight, circles, right and left turns

Standards:

Common Core State Standards Math Practices

CCSSMP1: Make sense of problems and persevere in solving them

CCSSMP2: Reason abstractly and quantitatively

CCSSMP4: Model with mathematics

CCSSMP6: Attend to precision

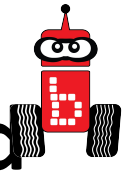
CCSSMP8: Look for and express regularity in repeated reasoning

Next Generation Science and Engineering Practice

1: Asking questions and defining problems
2: Developing and using models
3: Planning and carrying out investigations
4: Analyzing and interpreting data
5: Using mathematics and computational thinking
6: Constructing explanations and designing solutions
7: Engaging in argument from evidence obtaining, evaluating, and communicating information

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Moving Your Robot Using the `counter()` Function Continued



Standards Continued:

2016 ISTE Standards

Empowered Learner

1c: Students use technology to seek feedback that informs and improves their practice and to demonstrate their learning in a variety of ways.

1d: Students understand the fundamental concepts of technology operations, demonstrate the ability to choose, use and troubleshoot current technologies and are able to transfer their knowledge to explore emerging technologies.

Knowledge Constructor

3d: Students build knowledge by actively exploring real-world issues and problems, developing ideas and theories and pursuing answers and solutions.

Innovative Designer

4a: Students know and use a deliberate design process for generating ideas, testing theories, creating innovative artifacts or solving authentic problems.

4b: Students select and use digital tools to plan and manage a design process that considers design constraints and calculated risks.

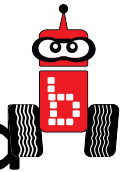
4c: Students develop, test and refine prototypes as part of a cyclical design process.

4d: Students exhibit a tolerance for ambiguity, perseverance and the capacity to work with open-ended problems.

Computational Thinker

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Moving Your Robot Using the `counter()` Function Continued



Standards Continued:

2016 ISTE Standards

Computational Thinker

5b: Students collect data or identify relevant data sets, use digital tools to analyze them, and represent data in various ways to facilitate problem-solving and decision-making.

5c: Students break problems into component parts, extract key information, and develop descriptive models to understand complex systems or facilitate problem-solving.

5d: Students understand how automation works and use algorithmic thinking to develop a sequence of steps to create and test automated solutions.

Creative Communicator

6a: Students choose the appropriate platforms and tools for meeting the desired objectives of their creation or communication.

6c: Students communicate complex ideas clearly and effectively by creating or using a variety of digital objects such as visualizations, models or simulations.

Global Collaborator

7c: Students contribute constructively to project teams, assuming various roles and responsibilities to work effectively toward a common goal.

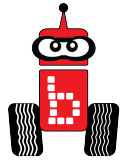
KISS INSTITUTE FOR PRACTICAL ROBOTICS

Prerequisite

- Know how to use while loops
- Know how to use Variables (Unit 12)
  - Review variables
- Know how to create your own functions

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Variables as Counters



•Remember that variables can be modified over time, so how could this be useful?

```
int counter;
counter = 0;
```

counter 0

```
// some code later
counter = counter + 1; // adding one to the counter
```

•They can be used to help remember (or keep count) for us how many times something has been done (which can be useful for some loops).

The “trick” to understanding this is that the RIGHT side is done first which means ***counter “is assigned” counter (currently 0) plus one (or 0 + 1)***

Counter 1

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Learning to Make a Square
Activity 1

Description: Write a program for the KIPR Wallaby that drives the robot along a path in the shape of a square *using loops and counter*.

Analysis: What is the program supposed to do?

Pseudocode Comments

1. .Set Variable "side_counter" to 0. `// Set variable "counter" to 0.`
2. Loop: Is "counter" < 4 `//Loop: Is "counter" < 4?`
3. Drive forward. `//Drive forward.`
4. Turn right 90°. `//Turn right 90-degrees.`
5. Add 1 to "counter". `// Add 1 to "counter".`
6. Stop motors. `// Stop motors.`
7. End the program. `// End the program.`

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Making a Square with counters

Psuedocode

```
int main()
{
  // Set "counter" to 4.
  // Loop: Is "counter" < 4?
  // Drive forward.
  // Turn right 90-degrees.
  // Add 1 to "counter".
  // Stop motors.
  // End the program.
```

=

Source Code

```
int main()
{
  int counter = 0;
  while (counter < 4)
  {
      motor(0,90);
      motor(2,90);
      msleep(4000); // forward

      motor(0,100);
      motor(2,0);
      msleep(1500); //right turn

      side_counter = counter + 1;
  }
  ao();
   return 0;
}
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Review Creating Your Own Functions

These are 3 components of a function and should be created in this order:

1. Function prototype
   a. Create a **function prototype** on the line *after* `#include <kipr/botball.h>` and *before* `int main()`.

2. *Function definition*
   a. *Define* the new function (i.e., specify what the robot will actually do). The **function definition** goes after the last "`}`" in your program.

3. *Function Call*
   a. *Call* (use) the new function by typing the **function name** within a block of code.

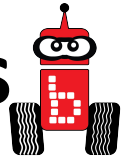
KISS INSTITUTE FOR PRACTICAL ROBOTICS

Create a Function
Activity 2

1. Create a function that will drive your robot forward for msleep(3000).
2. Compile and run your program.
3. What observations can you make? How will this be helpful

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Review Creating Your Own Functions



Notice the placement of the three components.

This is the
function prototype

This is the
Function call

This is the
function definition.

A **function definition** does not have a semicolon at the end.

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

void drive_forward();

int main()
{
  printf("Hello, World\n");
  drive_forward():
  return 0;
}

void drive_forward()
{
  motor(0,100); // forward
  motor(3,100);
  msleep(1000);

  ao();
}
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Draw a Square
Activity 3

1. Create new functions and use variables to create a square.
2. Compile and run the program.
3. What observations and benefits can you make about using variables and functions?

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Drawing a Square Using Variables and Creating Functions - Activity 2

Psuedocode

```
int main()
{
  // Set "counter" to 4.
  // Loop: Is "counter" < 4?
  // Drive forward.
  // Turn right 90-degrees.
  // Add 1 to "counter".
  // Stop motors.
  // End the program.
```

Source Code

```
void drive_forward_and_turn_right();
int main()
{
  int counter = 0 ;
  while (counter < 4)
  {
      drive_forward_and_turn_right();

      counter = counter + 1;
  }
  ao();
  return 0;
}
void drive_forward_and_turn_right()
{
  motor(0, 90);
  motor(2, 90);
  msleep(4000);
  motor(0,  70);
  motor(2, -70);
  msleep(1500);
  ao();
}
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Move the Servo Arm Using a Loop
Activity 4

Description: Write a program for the KIPR Wallaby that moves the robot servo arm from position 200 to 1800 in increments of 100.

•Remember to **enable the servos** at the beginning of your program, and **disable the servos** at the end of your program!

Analysis: What is the program supposed to do?

Pseudocode Comments (SAMPLE)

1.Set counter to 200. `//Set counter to 200`
2.Set servo position to counter. `//Set servo position to counter`
3.Enable servos. `//Enable servos.`
4.Loop: Is counter < 1800? `//Loop: Is counter < 1800?`
1.Wait for 100 milliseconds. `//Wait for 100 milliseconds.`
2.Add 100 to counter. `//Add 100 to servo position.`
3.Set servo position to counter. `//Set servo position to counter.`
5.Disable servos. `//Disable servos.`
6.End the program. `//End the program.`

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Moving the Servo Using a Loop

Psuedocode

```
int main()
{
  // Set counter to 200.
  // Set servo position to counter.
  // Enable servos.
  // loop: Is counter < 1800?
  // Wait for 0.1 seconds.
  // Add 100 to counter.
  // Set servo position to counter.
  // Disable servos.
  // End the program.
```

=

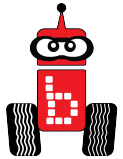
Source Code

```
int main()
{
  int counter = 200;
  set_servo_position(0, counter);
  enable_servos();

  while (counter < 1800)
   {
     msleep(100);
     counter = counter + 100;

     set_servo_position(0,counter);
   }
     msleep(100);
     disable_servos();

  return 0;
}
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Assessment

Climbing the Mountain Using Loops:



Setup: Use Surface-A. Place three 12 oz. empty soda in the orange garage.
Hint: You may need a ramp to set the cans upright.

Desired Outcome: The robot will drive out and pick up a can(s) in the orange garage and place them upright on the mountain(2 reams of paper stacked on top of each other) that is in the starting box.
Limitations:

1. The robot must start completely behind the vertical projection of the inside of the start line.
2. The robot cannot cross the orange garage solid lines.
3. The robot must enter through the dotted lines.
4. Students can put the cans anywhere within the orange garage as long as they do not touch the walls of the garage.
5. Students may place the reams of paper anywhere with the starting box as long as they are stacked on top of each other.
6. Cans must be upright and not touching any part of the robot on the mountain.

Completion: When the robot successfully place two cans on the mountain.
Extra Optimization: Students try to get all three cans on the mountain with the fastest time.

KISS INSTITUTE FOR PRACTICAL ROBOTICS