Understanding and Using Variables

Professional Development Workshop

Understanding and Using Variables

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Variables

Some reasons to use a variable:

1. You don't have to *remember* which value is a certain servo position – the computer remembers for you

2. It makes your program easier to read and understand

3. Makes it easier to debug your program

4. You can do computation and store results in variables

Variables

- A **variable** is a ***named*** container that stores a **type** of **value**
  A **variable** has the following three components:
  a. the **type** of data it stores (holds),
  b. the **name**, and
  c. the **value** it is currently storing.

a b c

```
int arm_up;
arm_up = 1230;
```

Use `int` as your data type if you want to store whole numbers (integers)

- Visualize/think of a **variable** like a *storage space* that holds a value with a name on it…
  - Servo "up" position
  - Servo "down" position
  - Etc.

arm_up	1230
arm_down	400

Variable Names

Each **variable** is given a unique name so we can identify it…

- Variable names can be *almost* anything you would like.
- Variable names can contain **letters**, **numbers**, and **underscores** ("_").
- Variable names **cannot** begin with a **number**.
- Variable names should be ***meaningful*** and not generic like "x"

An Example:

```
int arm_up;          // variable "declaration"
arm_up = 1230;       // variable "initialization"
```

You can do the declaration and initialization at the same time

```
int arm_up = 1230;
```

Working with Variables

1. *Declaring* a variable:

```
int arm_up;
```

2. *Initializing/setting* a variable:

```
arm_up = 1230;
```

3. *Calling* a variable:

```
arm_up
```

What is `int`?

`int` stands for "integer". This means that the variable `arm_up` will have an integer (whole number) value.

See the Team Homebase resources for more information on data types

Using Variables for Drive Motors

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Variable declarations generally go inside a block of code (i.e., inside the **{ }**) immediately after the starting curly brace (i.e., **{**) and before any other code.

Source Code

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

int main()
{
      // left = 3
      // right = 0
      printf("Drive and turn\n");
      motor(3,100);
      motor(0,100);
      msleep(1000);

      motor(3,-50);
      motor(0,50);
      msleep(500):

      return 0;
}
```

Remove the forward slashes from your comments, add int for the data type and since it is now code add the semicolon

Source Code

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

int main()
{
      int left = 3;
      int right = 0;
      printf("Drive and turn\n");
      motor(left,100);
      motor(right,100);
      msleep(1000);

      motor(left,-50);
      motor(right,50);
      msleep(500):

      return 0;
}
```

Using Variables for Servo Motors

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Variable declarations generally go inside a block of code (i.e., inside the **{ }**), immediately after the starting curly brace (i.e., **{**) and before any other code.

Source Code

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

int main()
{
      // arm_port = 0;
      // arm_up = 1230;
      // arm_down = 400;
      printf("Wave servo\n");
      enable_servos();
      set_servo_position(0,1230);
      msleep(500);
      set_servo_position(0,400);
      msleep(500);

      return 0;
}
```

Source Code

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

int main()
{
      int arm_port = 0;
      int arm_up = 1230;
      int arm_down = 400;
      printf("Wave servo\n");
      enable_servos();
      set_servo_position(arm_port,arm_up);
      msleep(500);
      set_servo_position(arm_port,arm_down);
      msleep(500);

      return 0;
}
```

How many *potential* lines of code have to change if the arm servo is switched to port 3?

Slowing Down a Servo Iterating a Variable

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Move the Servo Arm Using a Loop

1. Set counter to 200.
2. Set servo position to counter.
3. Enable servos.
4. *Loop:* Is counter < 1800?
   Wait for 0.1 seconds.
   Add 100 to counter.
   Set servo position to counter.
5. Disable servos.
6. End the program.

Notice the counter variable value changes every time the loop comes around. This is known as iterating a variable

Source Code

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

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;
}

```