KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Servos

Servos

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Servos

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Servos

- A **servo motor** (or **servo** for short) is a motor that rotates to a specified **position between ~0° and ~180°**.
- Servos are great for raising an arm or closing a claw to grab something.
- Servo motors look very similar to non-servo motors, but there are differences…
  - A servo has **three wires** (orange, red, and brown) and a **black plastic plug**.
  - A non-servo motor has **two gray wires** and a **two-prong plug**.

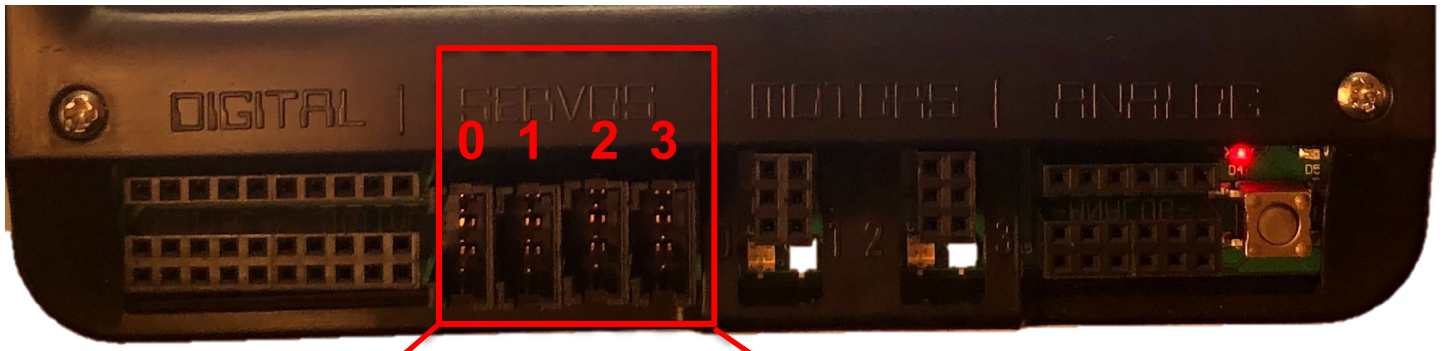
Large servo

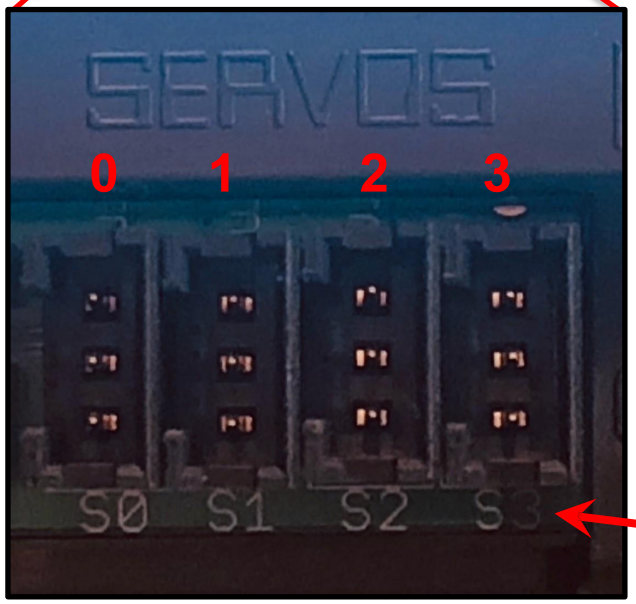
Micro servo

Wombat Servo Ports

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®





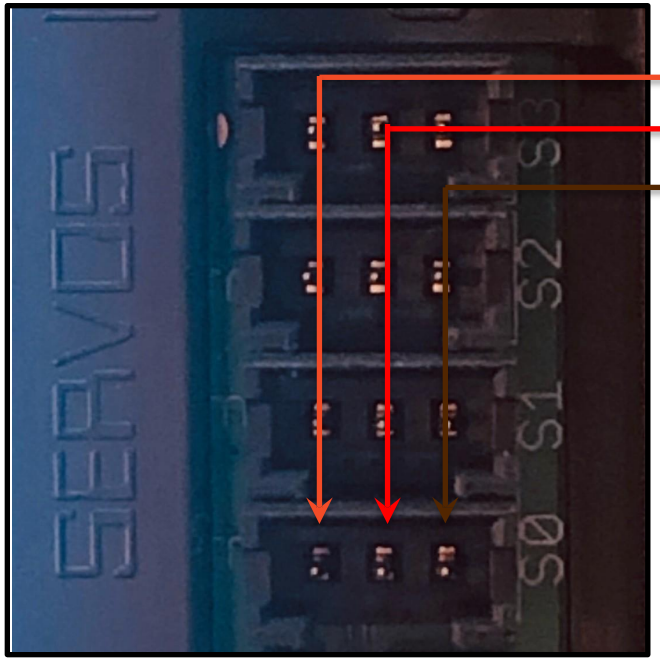
Servo Port Labels are on the board

Plugging in Servos

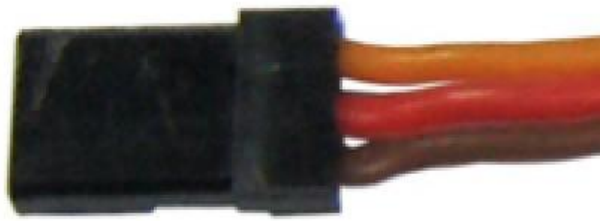
KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

- The KIPR Robotics Controller has 4 servo ports numbered **0** through **3**.
- Note that the orientation of the wires is very important:
  - (**S**) for the **orange** (**signal**) wire, which regulates servo position
    - Closest to the screen (orange “up”, brown “down”)
  - (**+**) for the **red** (**power**) wire
  - (**–**) for the **brown** (**ground**) wire (“the ground is down, down is negative”)



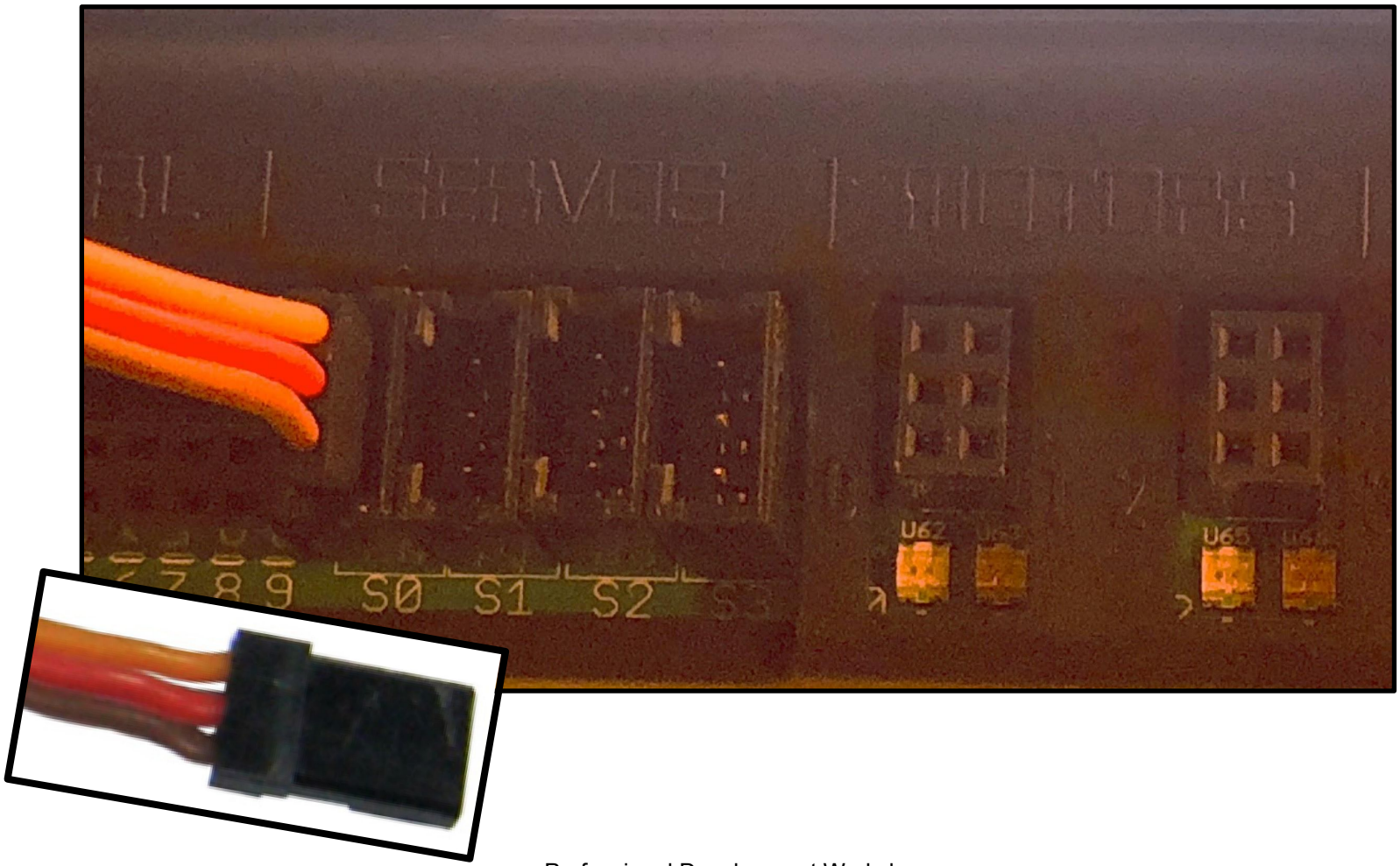
(S) signal wire
(+) power wire
(–) ground wire



NOTICE: orientation when plugging in the servos is very important

Plugged in Servos

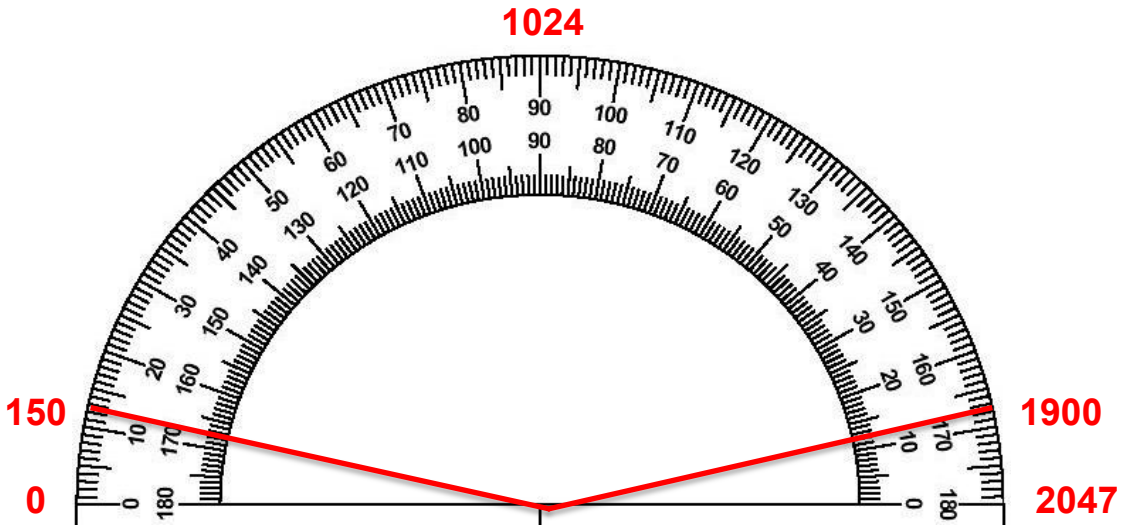
- One servo motor is plugged into Port **0**



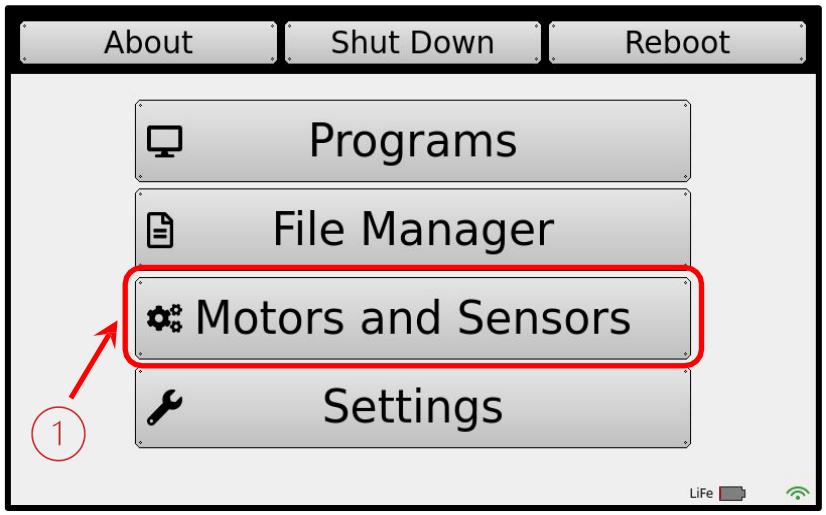
Servo Positions

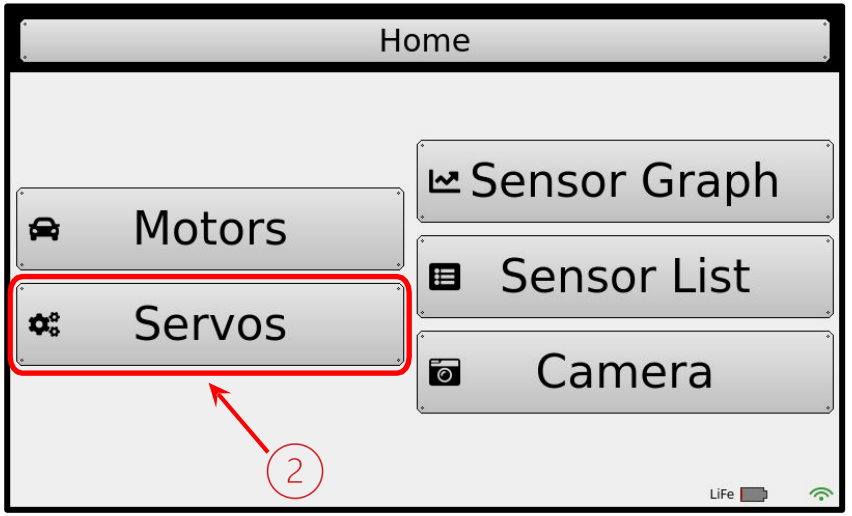
- Think of a servo like a protractor…
  - Angles in the **~180° range of motion** (between ~0° and ~180°) are divided into **2048 servo positions**.
  - These **2048 positions** range from 0 to 2047, but due to internal mechanical hard stop variability you should use **~150 to ~1900** (**remember:** computer scientists start counting with 0, not 1).
  - This allows for greater precision when setting a position (you have ~2048 different positions to choose from instead of just 180).

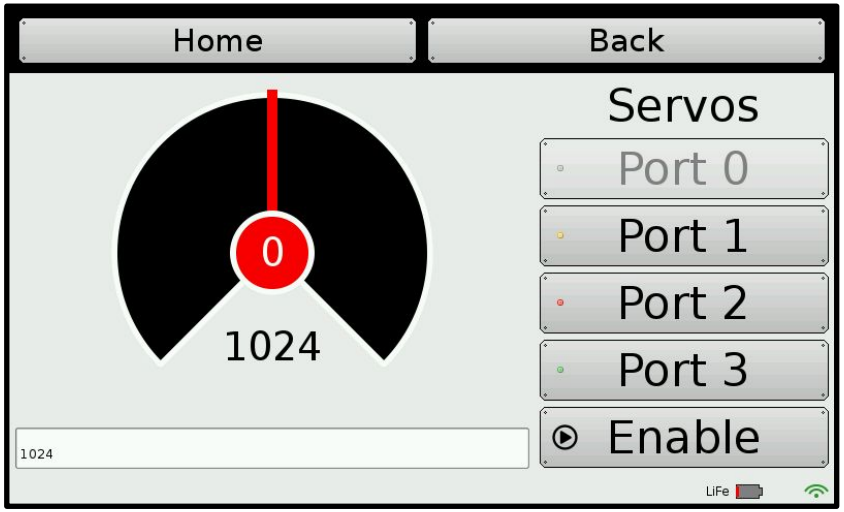
- The default position is **1024** (centered), however you should still use caution when setting up initial position.



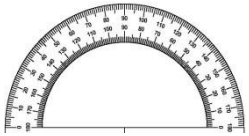
Servo Widget

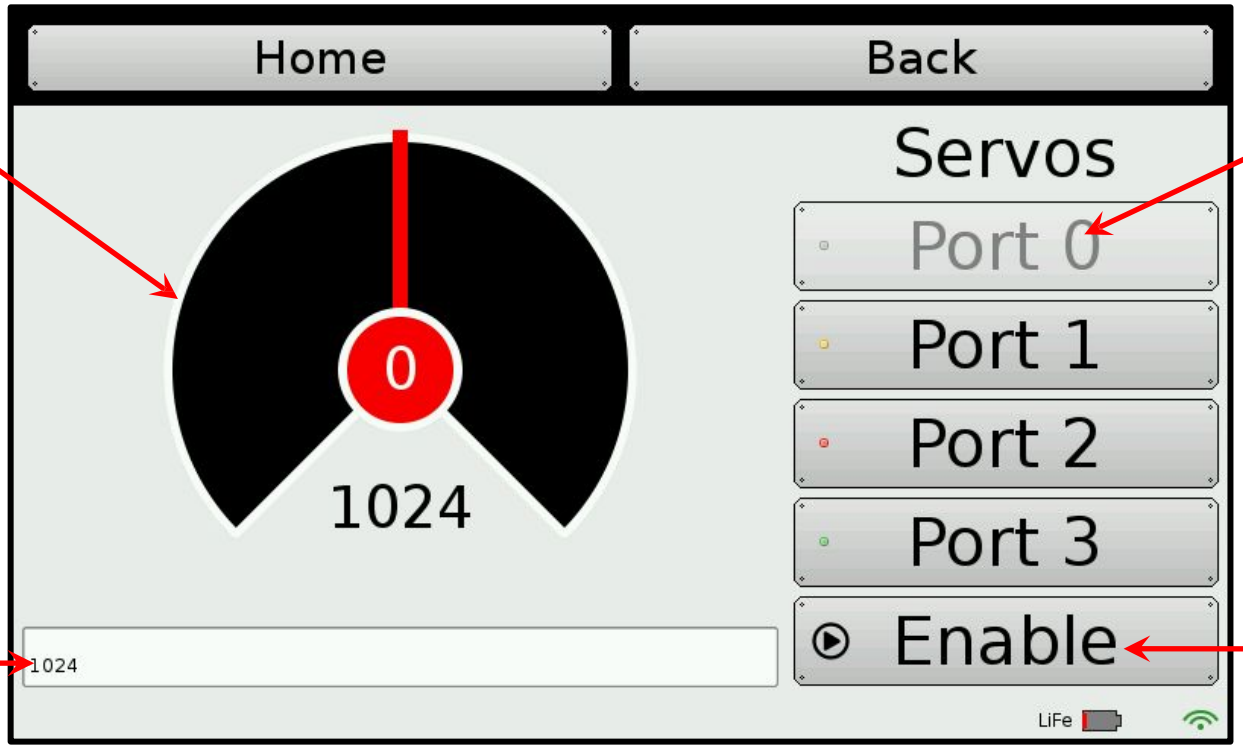






Testing with the Servo Widget





Select the servo port

The current servo position

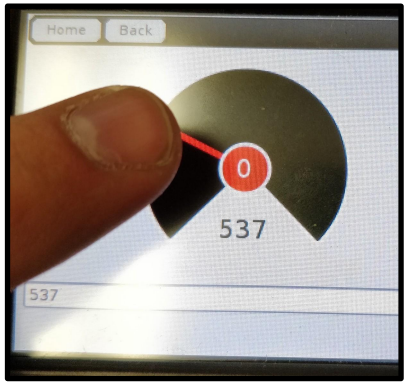
Enable servos

Testing with the Servo Widget

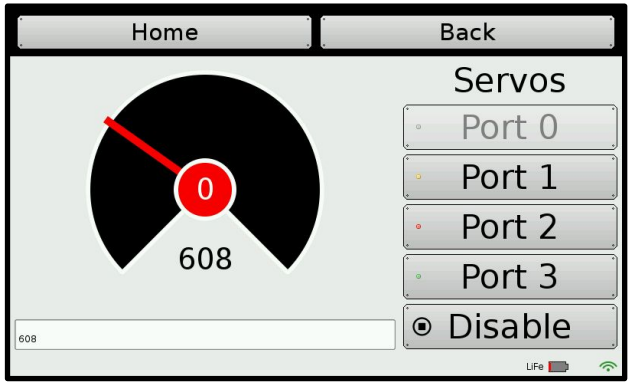
KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

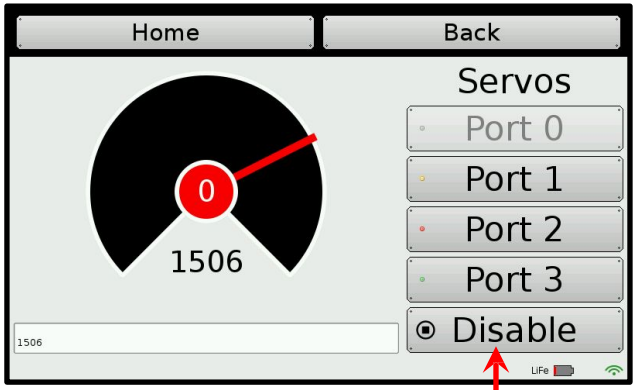
Use your finger to move the dial.



Servo @ 537



Servo @ 608



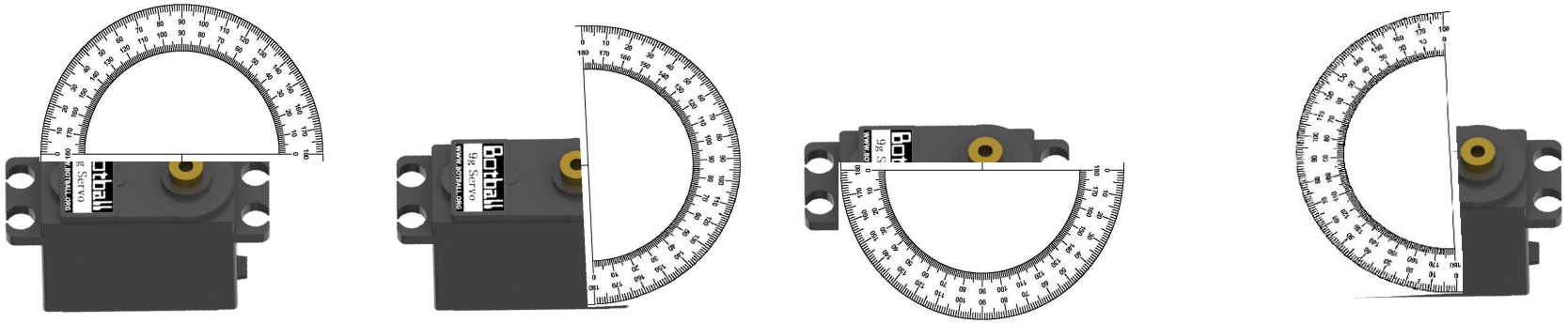
Servo @ 1506

Do <u>not</u> push a servo beyond its limits (less than ~150 or more than ~1900). This can burn out the servo motor!

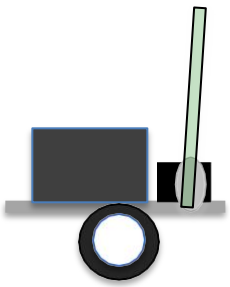
When you are finished, disable (turn off) the Servo

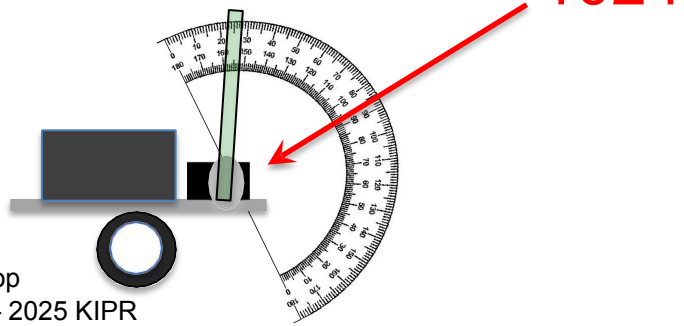
Centering the Servo Horn

- The Servo motor only has a range of motion of (rotates) ~180 degrees, but ***you cannot see by looking at the motor where this range of motion is located in relation to your robot***



- Using the Servo Widget screen, enable the servo on your robot. When you enable it, it will go to 1024. You can unscrew the servo horn on your arm or claw and place it in the center of the rotation if it is not already in the correct position





Servo Functions

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

- To help save power, servo ports by default are **not** active until they are **enabled**.
- Functions are provided for **enabling** or **disabling** all servo ports.
- A function is also provided for **setting the position** of an individual servo.

```
enable_servos();   // Enable (turn on) all servo ports.

set_servo_position(0,925);    // set servo on port #0 to position 925.

disable_servos();   // Disable (turn off) all servo ports.
```

- **Note:** it takes the servo TIME to move to a position so if you set it to another position without giving it TIME the CODE runs very fast and does not wait for the servo to move
- You can "**preset**" a servo position by calling **`set_servo_position()`** ***before*** calling **`enable_servos()`**. This will make the servo move towards this position immediately upon calling **`enable_servos()`**.

Servo Functions

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Example:

Source Code

```
1  #include <kipr/wombat.h>
2
3  int main()
4  {
5      enable_servos();
6
7      set_servo_position(0, 1500);
8      msleep(500);
9
10     set_servo_position(0, 925);
11     msleep(500);
12
13     set_servo_position(0, 675);
14     msleep(500);
15
16     disable_servos();
17     return 0;
18 }
19
```

(Note the use, and placement, of msleep to give the servo time to move to each new position)

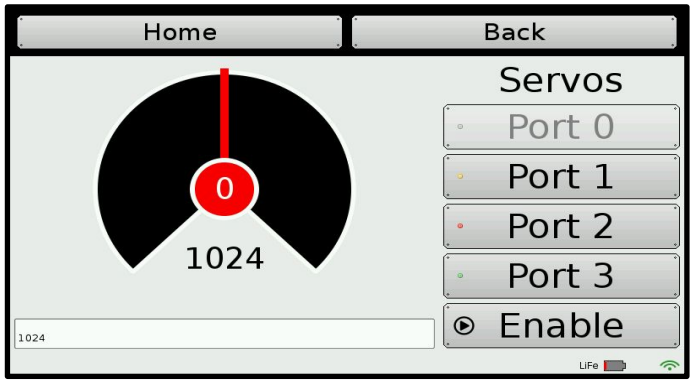
Wave the Servo Arm

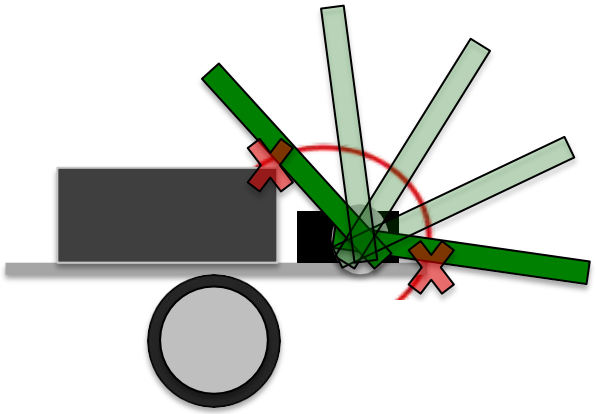
KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Description: Write a program that waves the DemoBot servo arm up and down.

- Remember to enable the servos at the beginning of your program, and disable the servos at the end of your program!
- **Warning**: The arm mounted on your DemoBot prevents the servo from freely rotating to all possible positions (it will run into the KIPR Wombat controller or the chassis of the robot)!
  - Do not keep trying to move a servo to a position it cannot reach, as this can burn out the servo and also consume a lot of power from your robot.
  - Use the Servo screen to determine the limits of the DemoBot arm, write these numbers down, and then use these numbers in your code.





Wave the Servo Arm

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

- Description: Write a program that waves the DemoBot servo arm up and down.
- Advanced: Write a function that does one wave. Call it from your main function.
- Analysis: What is the program supposed to do?

Pseudocode

1. Enable servos.
2. Move servo to up.
3. Wait for 3 seconds.
4. Move servo to down.
5. Wait for 3 seconds.
6. Disable servos.
7. End the program.

Comments

```
// 1. Enable servos.
// 2. Move servo to UP.
// 3. Wait for 3 seconds.
// 4. Move servo to DOWN.
// 5. Wait for 3 seconds.
// 6. Disable servos.
// 7. End the program.
```

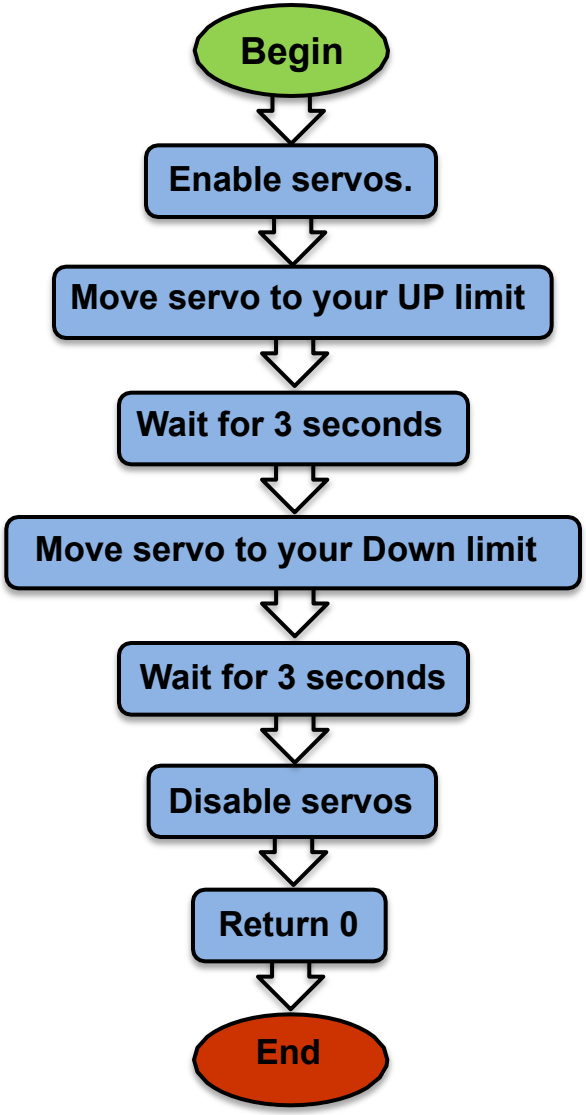
Wave the Servo Arm

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Analysis:

Flowchart



Commenting Within Your Programs

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

```
1  #include <kipr/wombat.h>
2
3  int main()
4  {
5      // arm = 0
6      // down = 400
7      // up = 1230
8      printf("Wave Servo Exercise\n");
9      return 0;
10 }
```

Make your comments after the first curly bracket and before the printf

Arm is plugged into servo port 0

Arm down position is 400

Arm up position is 1230

This (keeping track of <u>your</u> ports, positions, etc) could also be done in a notebook, but what if you misplace that notebook?