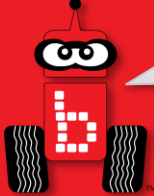


# You May Need to Update Your Wombat to use the *Create 3*

***NOTE: if your Wombat Firmware is  $\geq$  v30.2.1, you don't need to do this step***

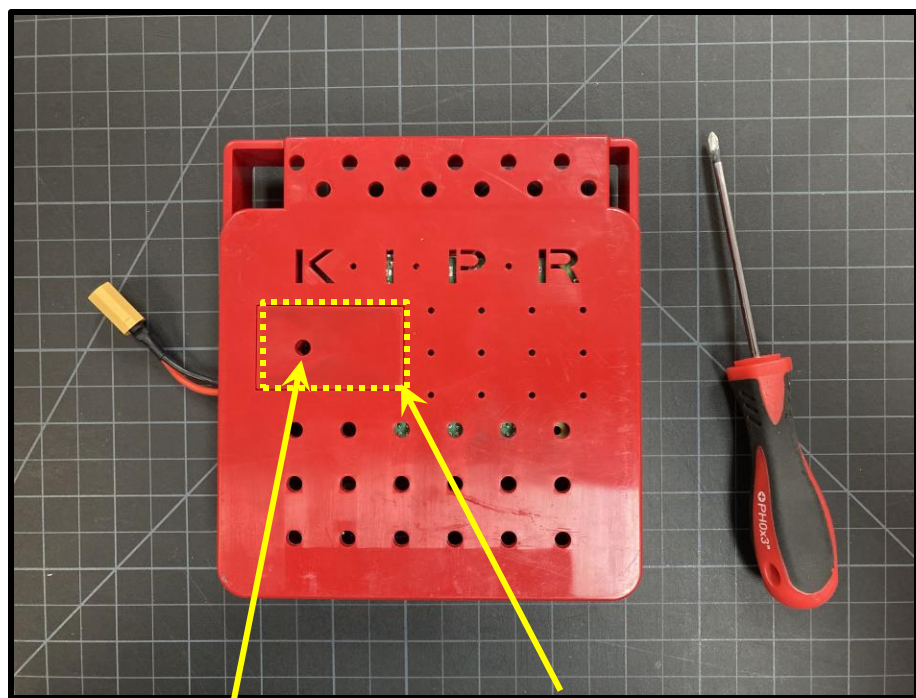
If your version is lower than 30.2.1 then you need to download the new wombat software image:

- [http://files.kipr.org/wombat/Wombat\\_v30.2.5.img](http://files.kipr.org/wombat/Wombat_v30.2.5.img)



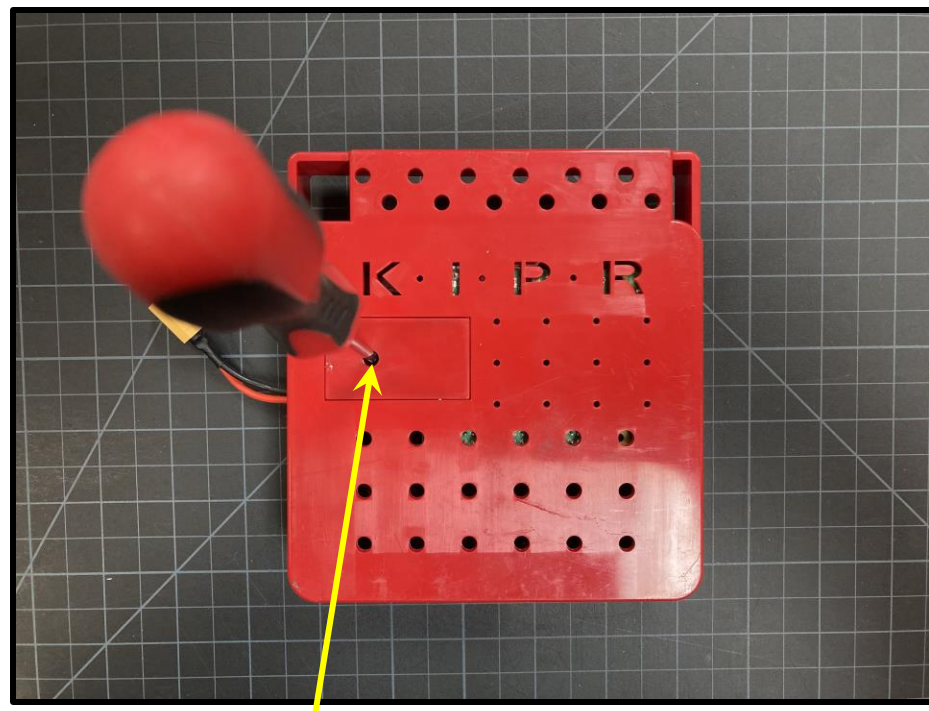
# Removing the SD Card from the Wombat

1. Make sure the wombat is unplugged and not powered
2. Remove the wombat from any chassis or framework so that you can access the rear panel

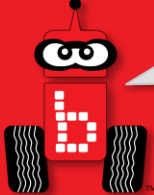


Door Screw

SD Card Door



Unscrew (screw won't come all the way out)



# Removing the SD Card from the Wombat Continued

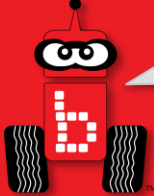
1. Remove the door (careful not to lose screw)
2. Carefully remove the SD card



SD Card



Flash the SD card (instructions on next slide)  
Come computers have a flasher or you can use an external USB flasher like pictured above



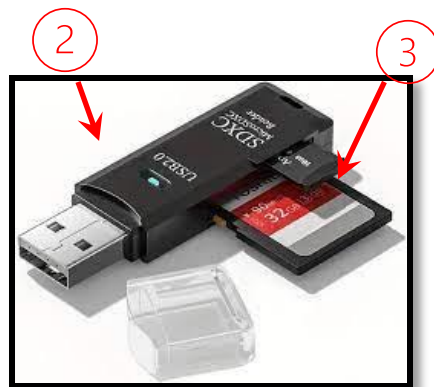
# Flashing the SD Card for Wombat

If you need to re-flash an SD card for wombat:

1. Download Balena Etcher: <https://etcher.balena.io/>. It is only available for Windows, Mac, or Linux (not Chromebooks)
2. Obtain an SD card writer
3. Insert micro SD into the smallest slot and plug USB into computer
4. Download desired software/image from the first slide if you haven't already:  
[http://files.kipr.org/wombat/Wombat\\_v30.2.5.img](http://files.kipr.org/wombat/Wombat_v30.2.5.img)
5. Open Balena Etcher on your computer

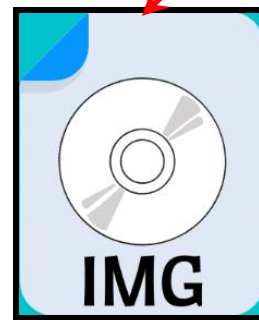


1

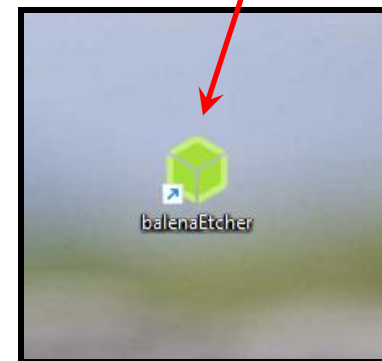


2

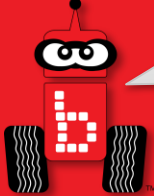
3



4

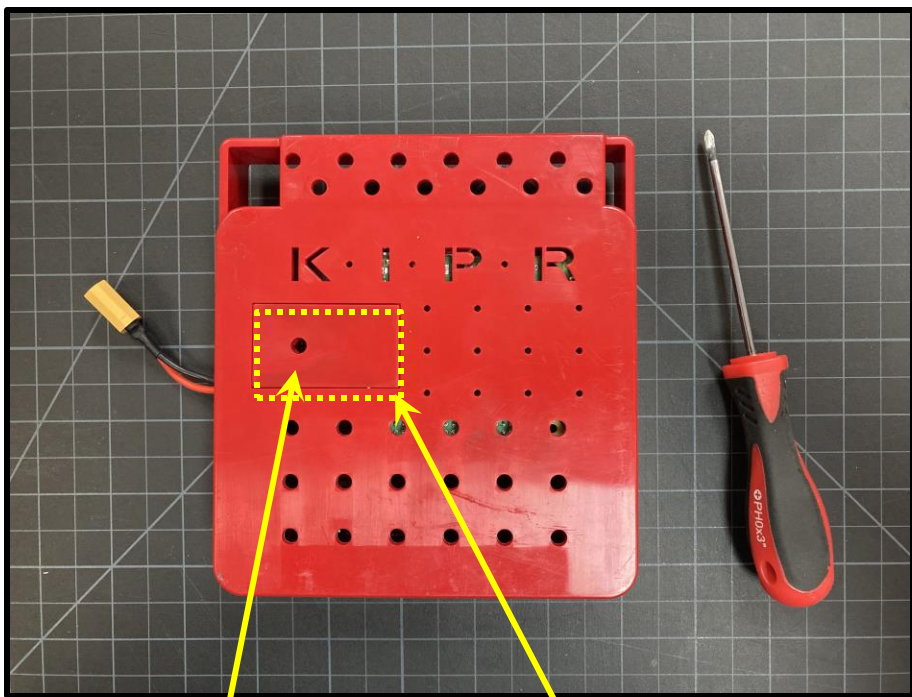


5



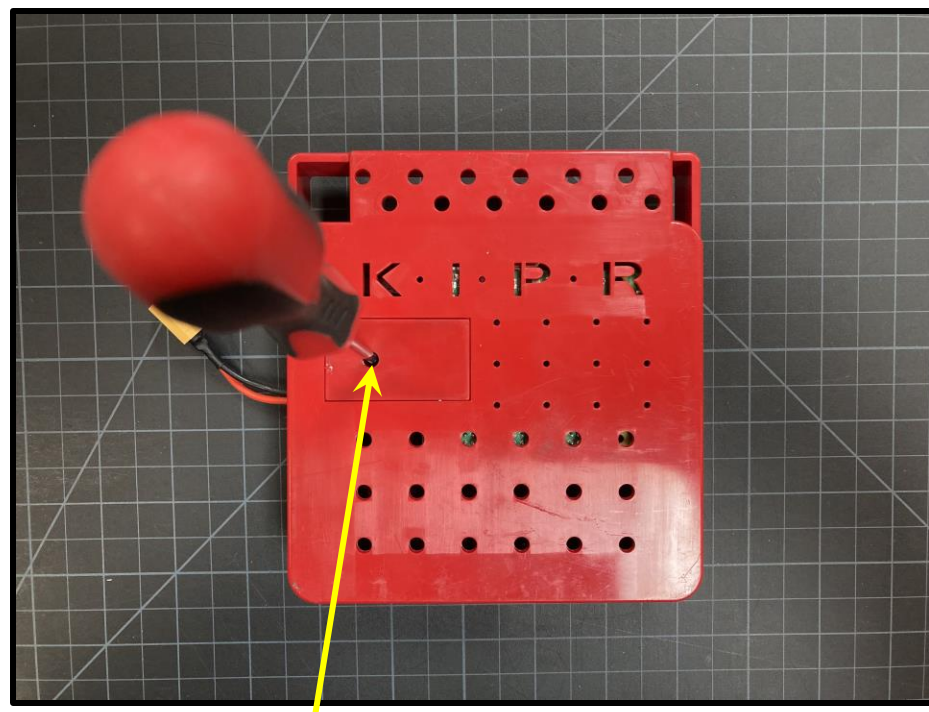
# Insert the Newly Flashed SD Card Into the Wombat

1. Insert the SD card into the Wombat
2. Replace the Door and screw into place

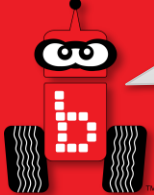


Door Screw

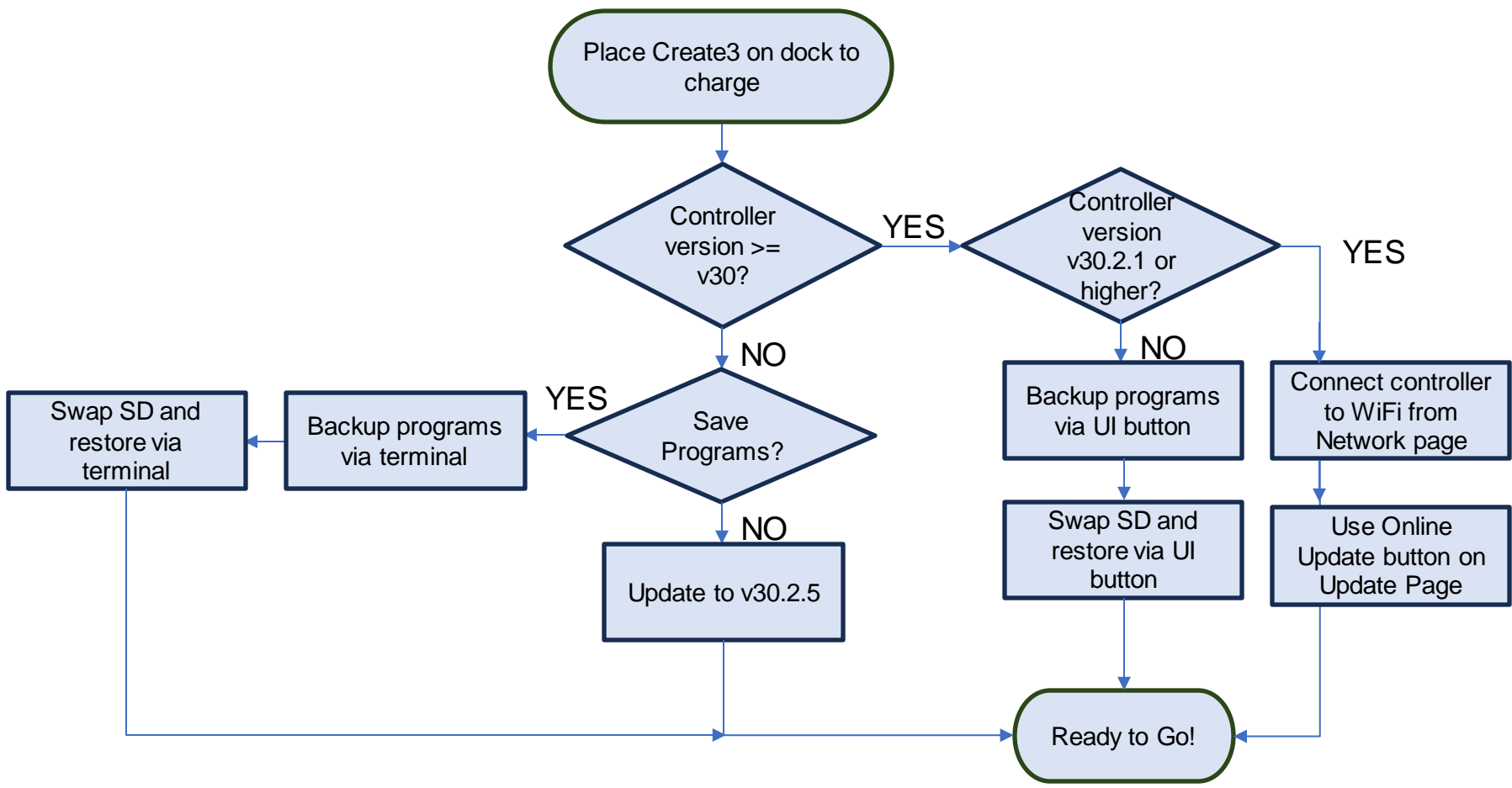
SD Card Door



Screw (Don't over tighten as this can easily be stripped)



# Updating the *Create 3*



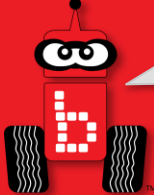


# Create 3 Downloads

The Create 3 works with the Wombat using **ONLY** versions of Humble (any firmware that starts with H), not Galactic (any version that start with G).

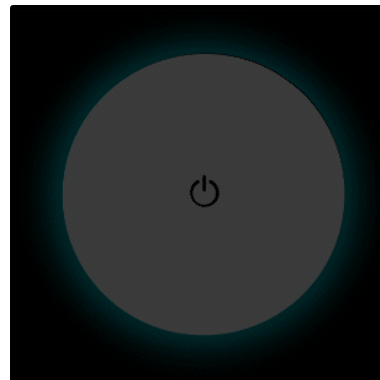
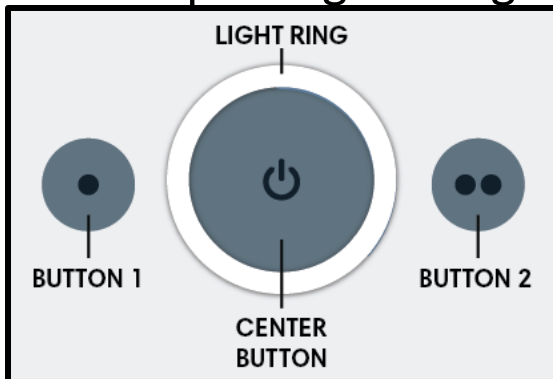
Download firmware for your Create 3:

- <https://edu.irobot.com/create3/firmware/H.2.4>

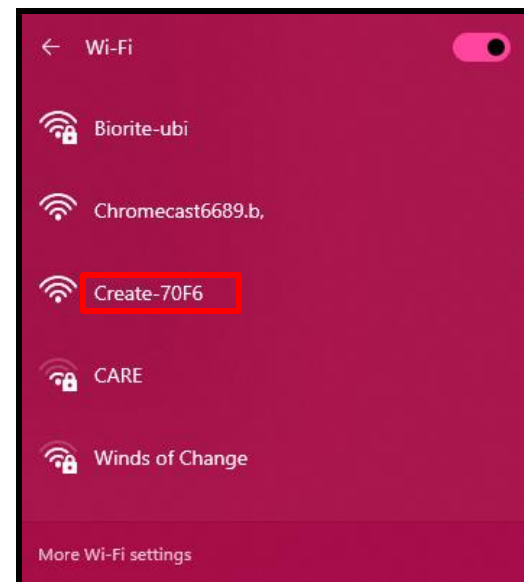
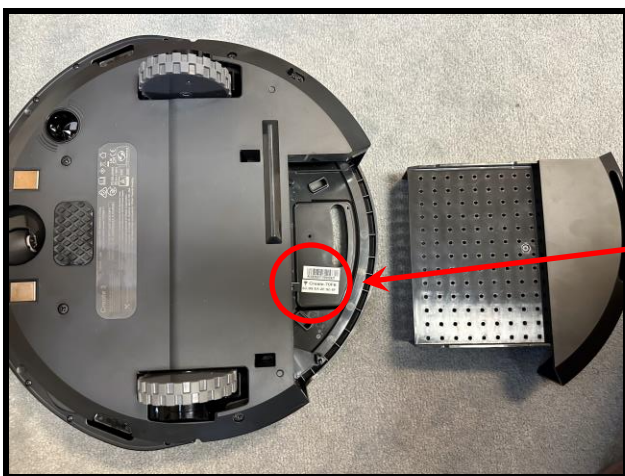


# Setting Up the *Create 3*

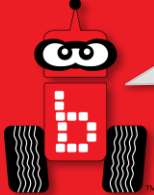
1. Hold down the ● and ●● buttons on your Create 3 until the light ring shows a spinning blue light.



2. Check Wi-Fi list and select your Create 3

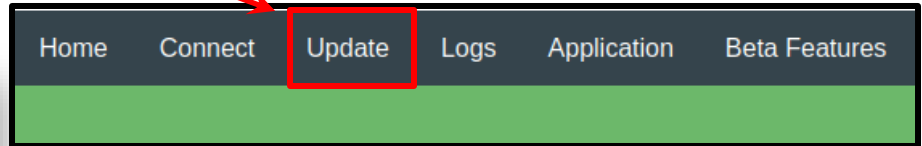
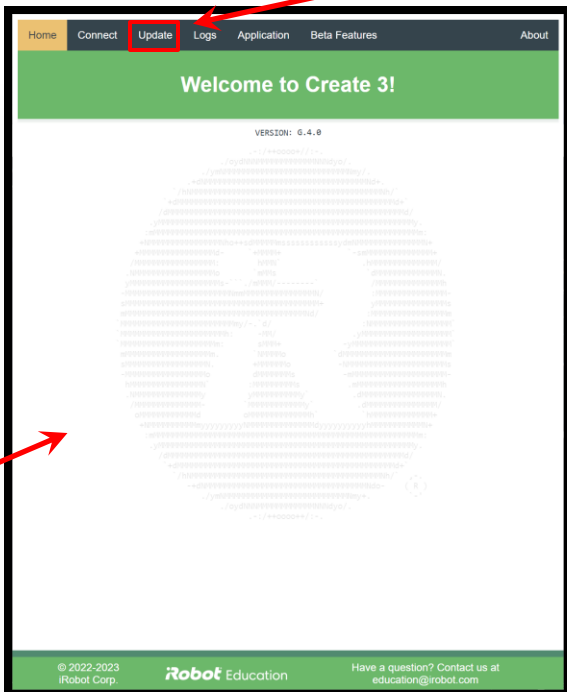
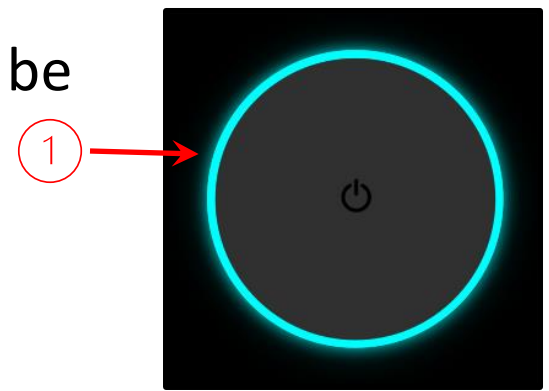


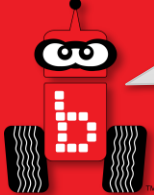




# Setting Up the Create 3

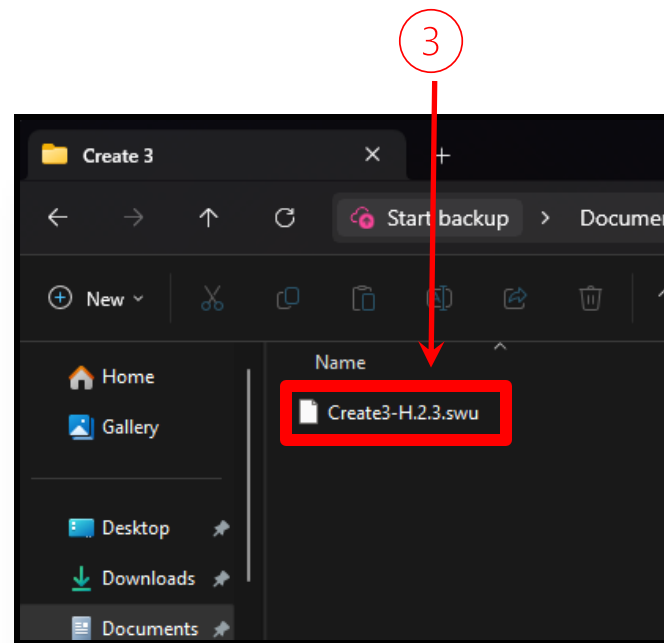
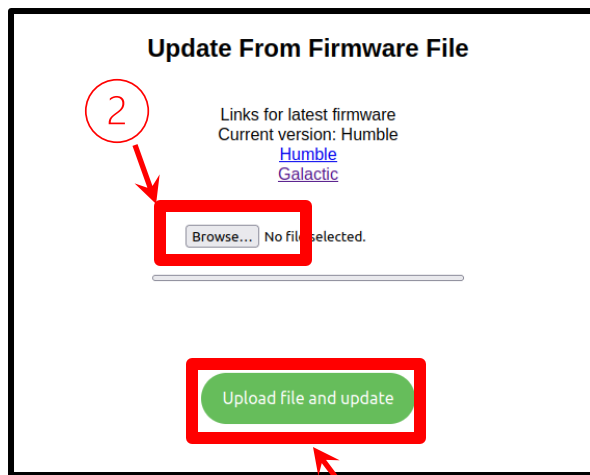
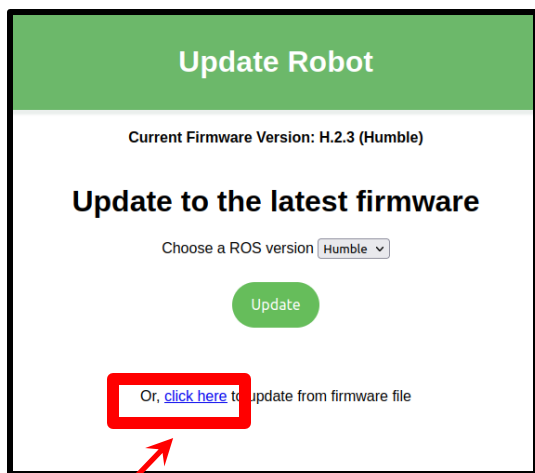
1. You'll hear a ding! The light ring will now be a solid blue
2. Open browser to **192.168.10.1**
3. Go to the Update tab

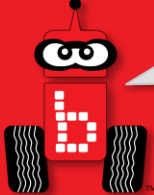




# Setting Up the *Create 3*

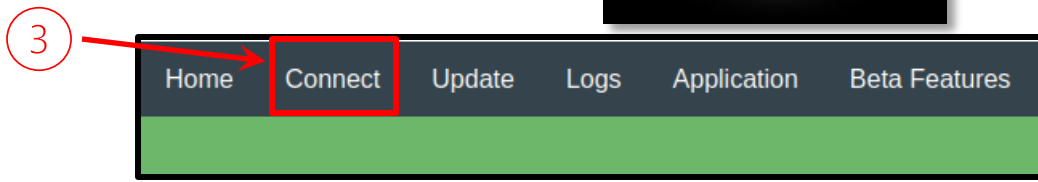
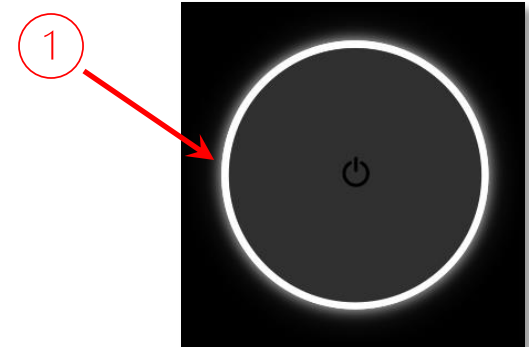
1. Select "click here" to update from firmware file.
2. Click Browse to open your file folder on your computer.
3. Navigate to where you saved the file and select.
4. Click "Upload file and update" to begin updating.





# Setting Up the *Create 3*

1. The Create will glow solid white and make a happy sound once finished
2. Reconnect to your Create3 and in a browser on your computer, navigate to:  
**192.168.10.1**
3. Go to the Connect Tab



**Connect Robot to Wi-Fi**

IP Address: 192.168.125.227

For detailed instructions, visit [edu.irobot.com/create3-setup](http://edu.irobot.com/create3-setup).

**Update Robot Names**

Host name (ROS Users):

Bluetooth name:

(Please note all fields are case-sensitive.)

**Connect to a 2.4 GHz Wi-Fi Network**

Type your Wi-Fi network name:

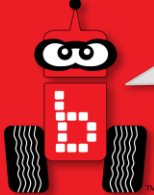
Wi-Fi Password:

Optional: additional radio bands are available for certain regions:

**WARNING: DO NOT CHANGE THIS!**

4. Find your Wombat name in the “Type your Wi-Fi network name” box
5. Type your Wombat’s password in the “Wi-Fi Password” box
6. Click Connect and wait for the Create to return to a solid white.

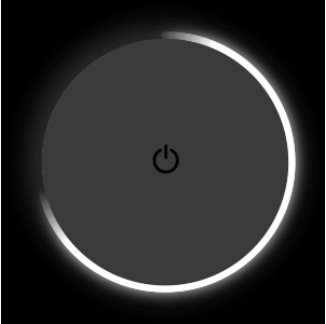
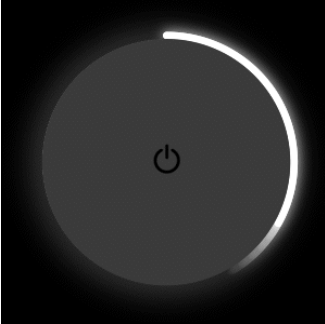
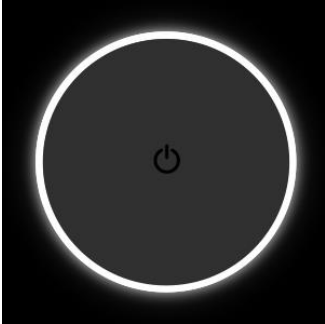
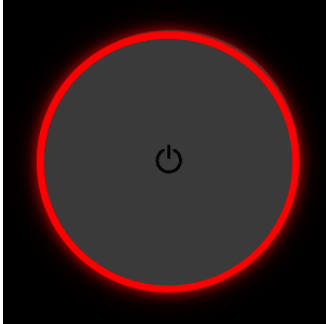
**NOTE: This may take a few minutes to connect**

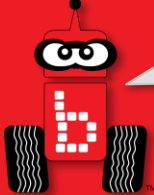


# Create 3 Light Ring

For reference, here's a list of the different light ring color codes:

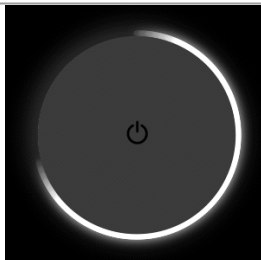
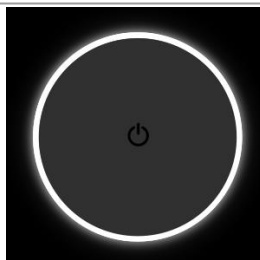
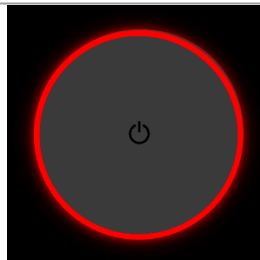
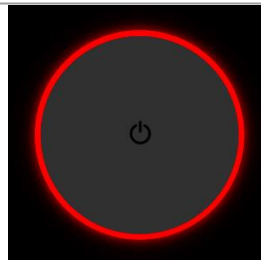
## While Charging (on the dock)

Spinning White	Partial White	Solid White	Pulsing Red
			
Robot is booting up. Wait for "happy sound" to play.	Robot is charging (Example shows 40%).	Robot is 100% charged.	Battery < 10%.

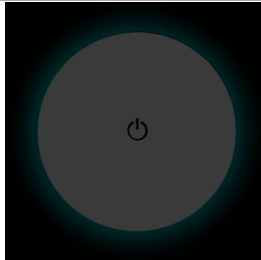
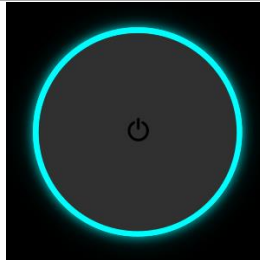


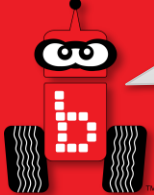
# Create 3 Light Ring

## While Idle

Spinning White	Solid White	Pulsing Red	Solid Red
			
Robot is booting up. Wait for "happy sound" to play.	Robot is powered on.	Battery < 10%. Place on charger.	Robot error. Cycle Power.

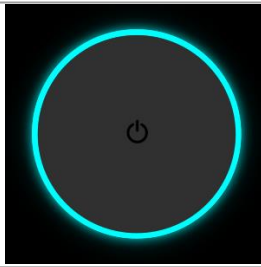
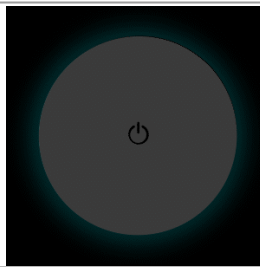
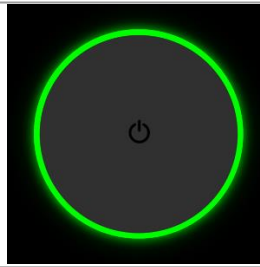
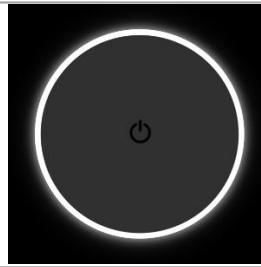
## While Connecting to Robot Access Point

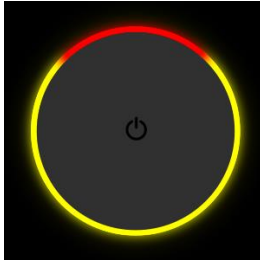
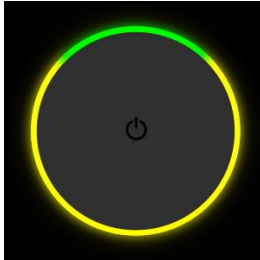
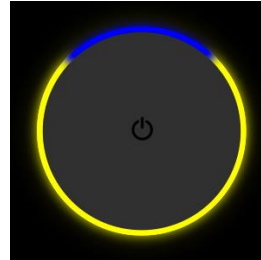
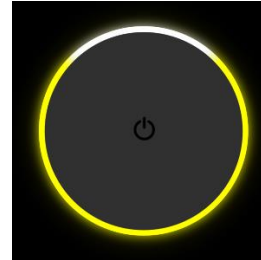
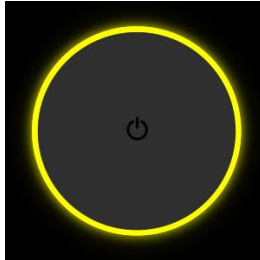
Spinning Cyan	Solid Cyan
	
Access Point is active. Select robot from device's Wi-Fi menu.	Device is connected to robot's Access Point page.

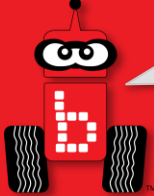


# Create 3 Light Ring

## While Connecting to Wi-Fi

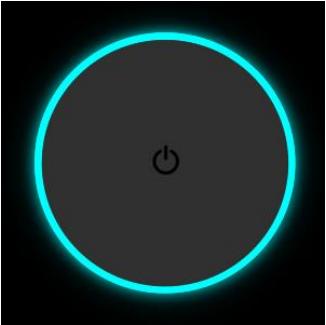
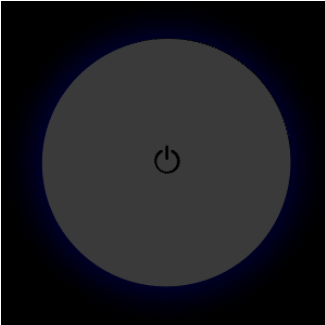
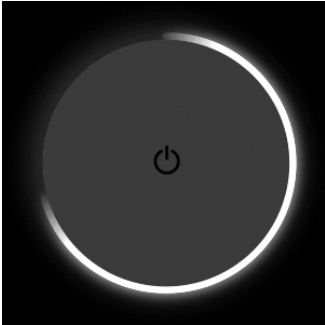
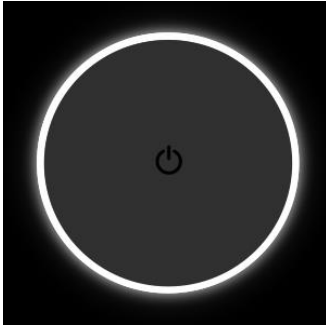
Solid Cyan	Spinning Cyan	Quick Green Flash	Solid White
			
Device is connected to robot's Access Point page.	Robot attempting to connect to Wi-Fi.	Success connecting to Wi-Fi.	Robot successfully disconnected from Access Point page.

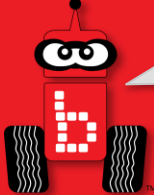
Yellow with Red	Yellow with Green	Yellow with Blue	Yellow with White	Solid Yellow
				
Failed Wi-Fi password.	Robot cannot connect to network access point.	DCHP failed to obtain valid IP addr. before time-out. Try again.	Access point located but failed association. Try again.	Failed to connect to Wi-Fi for unknown reason.



# Create 3 Light Ring

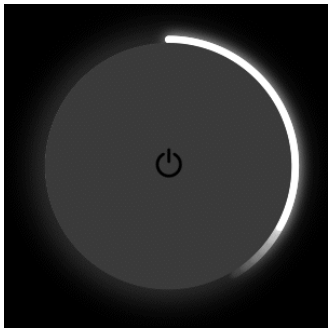
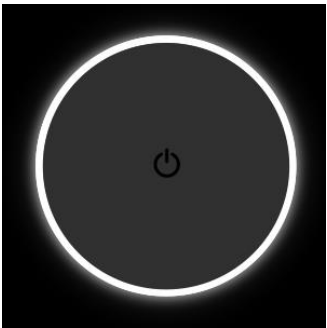
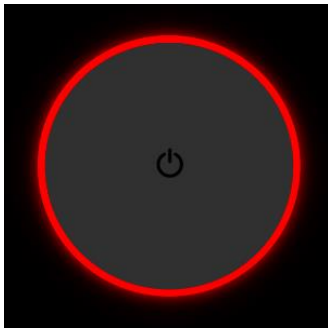
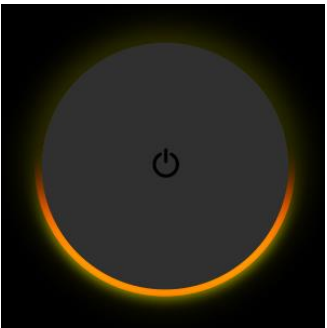
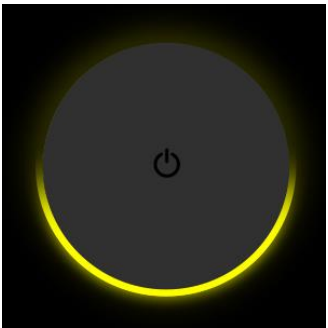
## While Updating Firmware (on the dock)

Solid Cyan	Spinning Blue	Spinning White	Solid White
			
Device is connected to robot's Access Point page.	Robot downloading update file.	Robot updating firmware. Do not remove from dock.	Update Successful.

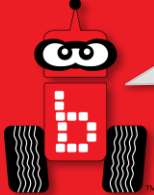


# Create 3 Light Ring

## While Operating

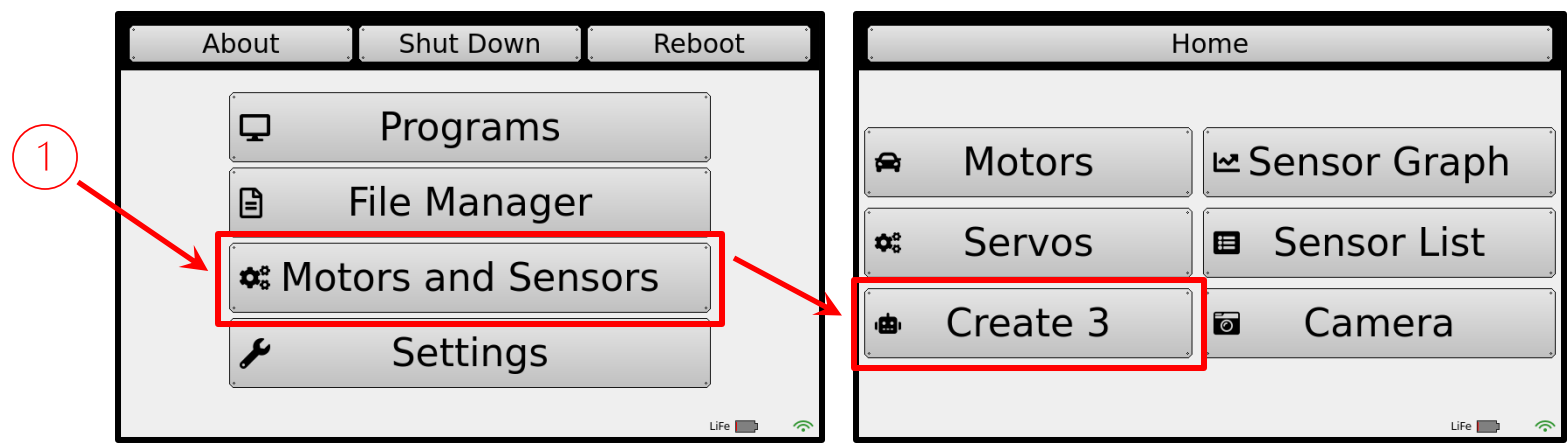
Spinning white	Solid White	Pulsing Red	Half Solid Orange	Half Solid Yellow
				
Robot is booting up. Wait for “happy sound” to play.	Default light color.	Robot updating firmware. Do not remove from dock.	Back-up safety activated.	Wheels disabled.



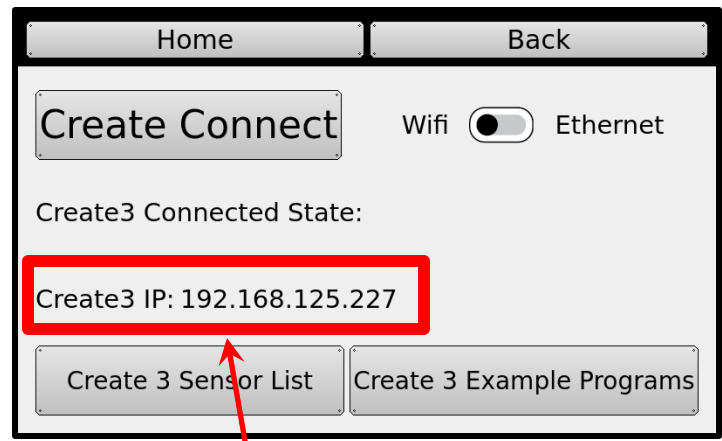


# Setting Up the *Create 3*

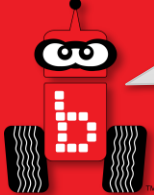
1. On your Wombat, go to Motors and Sensors -> Create 3



2. On the Create 3 page, look at the Create 3 IP. If you do not see one there, your Create may not be connected to your Wombat yet or you may want to try rebooting the Wombat.

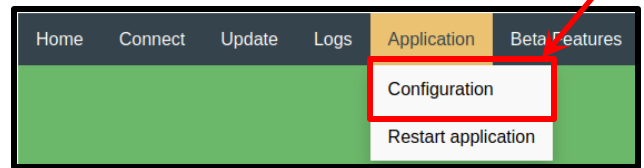


Note this IP address



# Setting Up the Create 3

1. Connect to your Wombat and go to the IP on your Create 3 page in your browser.
2. Go to the Application -> Configuration tab.
3. Ensure the "RMW\_IMPLEMENTATION" is set to **rmw\_fastrtps\_cpp**
4. Check the box next to "Enable Fast DDS discovery server"
5. Change the "Address and port of Fast DDS discovery server" to: **192.168.125.1:11811**
6. Click the green Save and wait for it to return to the page.



App config

---

Main Configuration

ROS 2 Domain ID (default 0):  3

ROS 2 Namespace:

RMW\_IMPLEMENTATION:  3

Enable Fast DDS discovery server?  4

Address and port of Fast DDS discovery server:  5

6 → Save

[Restart application](#) for changes to take effect.

---

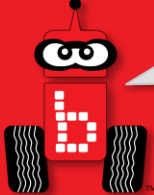
Application ROS 2 Parameters File

Note: If namespace above is not empty, node names to set parameters for must include namespace. (This does not validate yaml formatting; please validate separately.)

```

motion_control:
  ros_parameters:
    # safety_override options are
    # "none" - standard safety profile, robot cannot backup more
    # than an inch because of lack of cliff protection in rear, max
    # speed 0.306m/s
    # "backup_only" - allow backup without cliff safety, but keep
    # cliff safety forward and max speed at 0.306m/s
    # "full" - no cliff safety, robot will ignore cliffs and set
    # max speed to 0.46m/s
    safety_override: "full"
    reflexes_enabled: false
            
```

Save



# Setting Up the Create 3

1. ON THE SAME PAGE:  
Under the ROS 2 Parameters File, change `safety_override` from "**none**" to "**full**".
2. Also add:  
**`reflexes_enabled: false`**
3. Have someone double check your work. It is important not to mess this section up.

App config

---

Main Configuration

ROS 2 Domain ID (default 0):

ROS 2 Namespace:

RMW\_IMPLEMENTATION:

Enable Fast DDS discovery server?

Address and port of Fast DDS discovery server:

[Restart application](#) for changes to take effect.

---

Application ROS 2 Parameters File

Note: If namespace above is not empty, node names to set parameters for must include namespace. (This does not validate yaml formatting; please validate separately.)

```

motion_control:
  ros_parameters:
    # safety_override options are
    # "none" - standard safety profile, robot cannot backup more
    # than an inch because of lack of cliff protection in rear, max
    # speed 0.306m/s
    # "backup_only" - allow backup without cliff safety, but keep
    # cliff safety forward and max speed at 0.306m/s
    # "full" - no cliff safety, robot will ignore cliffs and set
    # max speed to 0.46m/s
    safety_override: "full"
    reflexes_enabled: false
  
```

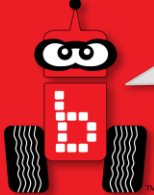
NOTE: Formatting matters!

[Save](#)

```

motion_control:
  ros_parameters:
    # safety_override options are
    # "none" - standard safety profile, robot cannot backup more
    # than an inch because of lack of cliff protection in rear, max
    # speed 0.306m/s
    # "backup_only" - allow backup without cliff safety, but keep
    # cliff safety forward and max speed at 0.306m/s
    # "full" - no cliff safety, robot will ignore cliffs and set
    # max speed to 0.46m/s
    safety_override: "full"
    reflexes_enabled: false
  
```

② → **safety\_override: "full"**  
aligned → **reflexes\_enabled: false**  
↑ **One Space**



# Setting Up the *Create 3*

1. Once you have made all the changes, Save.
2. Once the page refreshes, confirm that your changes are there.
3. Click Restart Application.
4. It may take several minutes for the robot to reboot. Once it is finished, it will make a happy sound and return to a solid white light.

**App config**

**Main Configuration**

ROS 2 Domain ID (default 0): 0

ROS 2 Namespace:

RMW\_IMPLEMENTATION: rmw\_fastrtps\_cpp

Enable Fast DDS discovery server?

Address and port of Fast DDS discovery server: 192.168.125.1:11811

Save

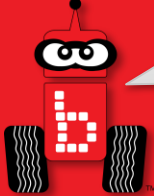
[Restart application for changes to take effect.](#)

**Application ROS 2 Parameters File**

Note: If namespace above is not empty, node names to set parameters for must include namespace. (This does not validate yaml formatting; please validate separately.)

```
motion_control:
  ros_parameters:
    # safety_override options are
    # "none" - standard safety profile, robot cannot backup more
    # than an inch because of lack of cliff protection in rear, max
    # speed 0.306m/s
    # "backup_only" - allow backup without cliff safety, but keep
    # cliff safety forward and max speed at 0.306m/s
    # "full" - no cliff safety, robot will ignore cliffs and set
    # max speed to 0.46m/s
    safety_override: "full"
    reflexes_enabled: false
```

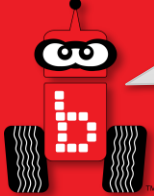
Save



# Create 3 Function Types

Most Create 3 functions can be split into 3 different types:  
**Actions, Publishers, and Subscribers.**

Actions	Publishers	Subscribers
<ul style="list-style-type: none"><li>• Send a goal for the Create 3 to complete (ex. Drive straight, rotate, drive an arc)</li><li>• A <code>create3_wait()</code> is needed after the last action in the program to make sure it runs</li><li>• <code>create3_wait()</code> tells the program to wait until the last action before it is completed</li><li>• Can be pushed past using a <code>create3_execute_next_command_immediately()</code> to move on to next command without completing the previous one</li></ul>	<ul style="list-style-type: none"><li>• Sends a state for the Create 3 to be in (ex. A specific velocity)</li><li>• Needs to be repeatedly sent to the Create 3 to keep it in that state</li></ul>	<ul style="list-style-type: none"><li>• Receives the state of something from the Create 3 (ex. Cliff sensors, IR sensors, bump sensors, odometry)</li><li>• Returns a value that can be used by the program</li></ul>



# Setting Up the *Create 3*

1. Reboot your Wombat by clicking the Shut Down button
2. Return to the Create 3 page and then Click **Create Connect**.
3. You should see a Connected text next to **Create 3 Connected State**.
4. You can now use Create 3 commands in your code.

**IMPORTANT**

1

2

3

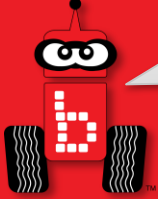
Home Back

Create Connect Wifi  Ethernet

Create3 Connected State:

Create3 IP: 192.168.125.227

Create 3 Sensor List Create 3 Example Programs



# Basic *Create 3* Functions

```
// Connects to your Create. Returns 1 if successful and 0 if not.
```

```
• create3_connect();
```

```
// Needed to make sure the most recent Create 3 action runs. Once called, any actions in the code will run until they are complete. Other non-Create 3 commands can be run while this is happening. It must be used if you want to wait for an action to run before another command.
```

```
• create3_wait();
```

```
// Runs whatever the next command in the code is immediately even if the previous one has not completed.
```

```
• create3_execute_next_command_immediately();
```

```
// Action: Drive the Create 3 a distance in meters at a specific speed. The max possible speed is 0.46 meters/sec. Both arguments can be decimals.
```

```
• create3_drive_straight(float distance, float max_linear_speed);
```

```
// Action: Rotates the Create 3 a certain number of degrees at a specific speed. Both commands are the same except the angle units are either degrees or radians and the speeds are degrees/sec or radians/sec.
```

```
• create3_rotate_degrees(float angle, float max_angular_speed);
```

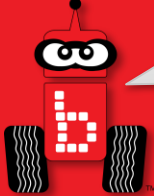
```
• create3_rotate_radians(float angle, float max_angular_speed);
```

```
// Action: Drives the Create 3 a certain number of degrees along an arc using the radius at a specific speed. The radius is in meters and the max_linear_speed is in meters/sec. Both commands are the same except the angle units are either degrees or radians.
```

```
• create3_drive_arc_degrees(float radius, float angle, float max_linear_speed);
```

```
• create3_drive_arc_radians(float radius, float angle, float max_linear_speed);
```

Professional Development Workshop

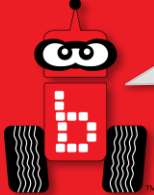


# Create 3 Sensors

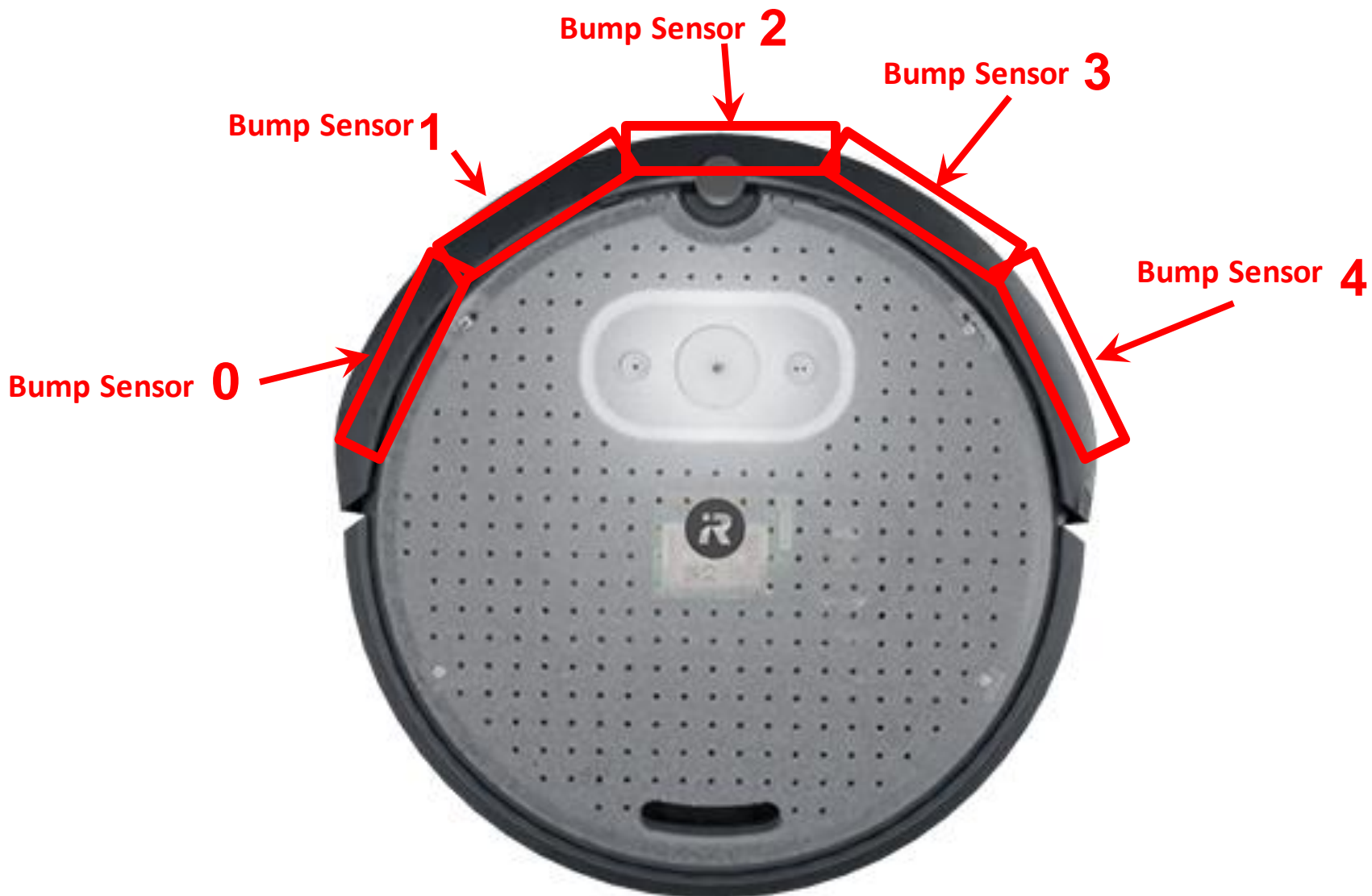
The Create 3 has 4 different types of sensors that can be used. All sensor functions are **subscribers** and will return values from the Create 3. The 4 different sensor types are:

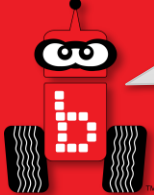
- **Bump sensors** – 5 digital sensors on the front bumper of the Create 3.
- **Cliff sensors** – 4 analog sensors on the bottom of the Create 3. These can detect edges and also the difference between black and white surface.
- **IR Sensors** – 7 analog sensors on the front of the Create 3. These can detect objects close to the Create 3 and give values based on distance.
- **Odometry** – The Create 3 has internal Odometry that can be used to calculate location, orientation, and speeds.



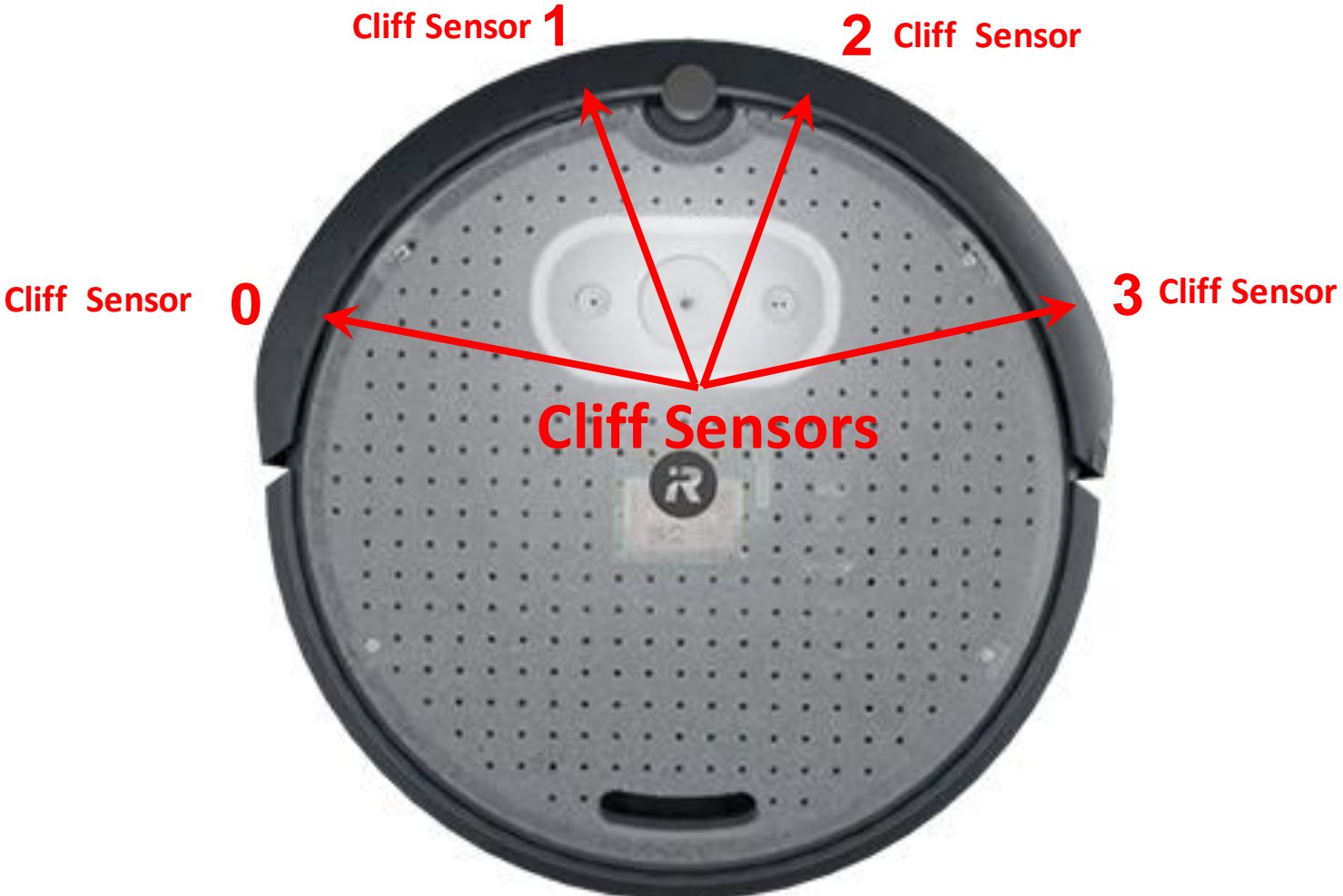


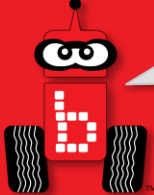
# Bump Sensor Locations



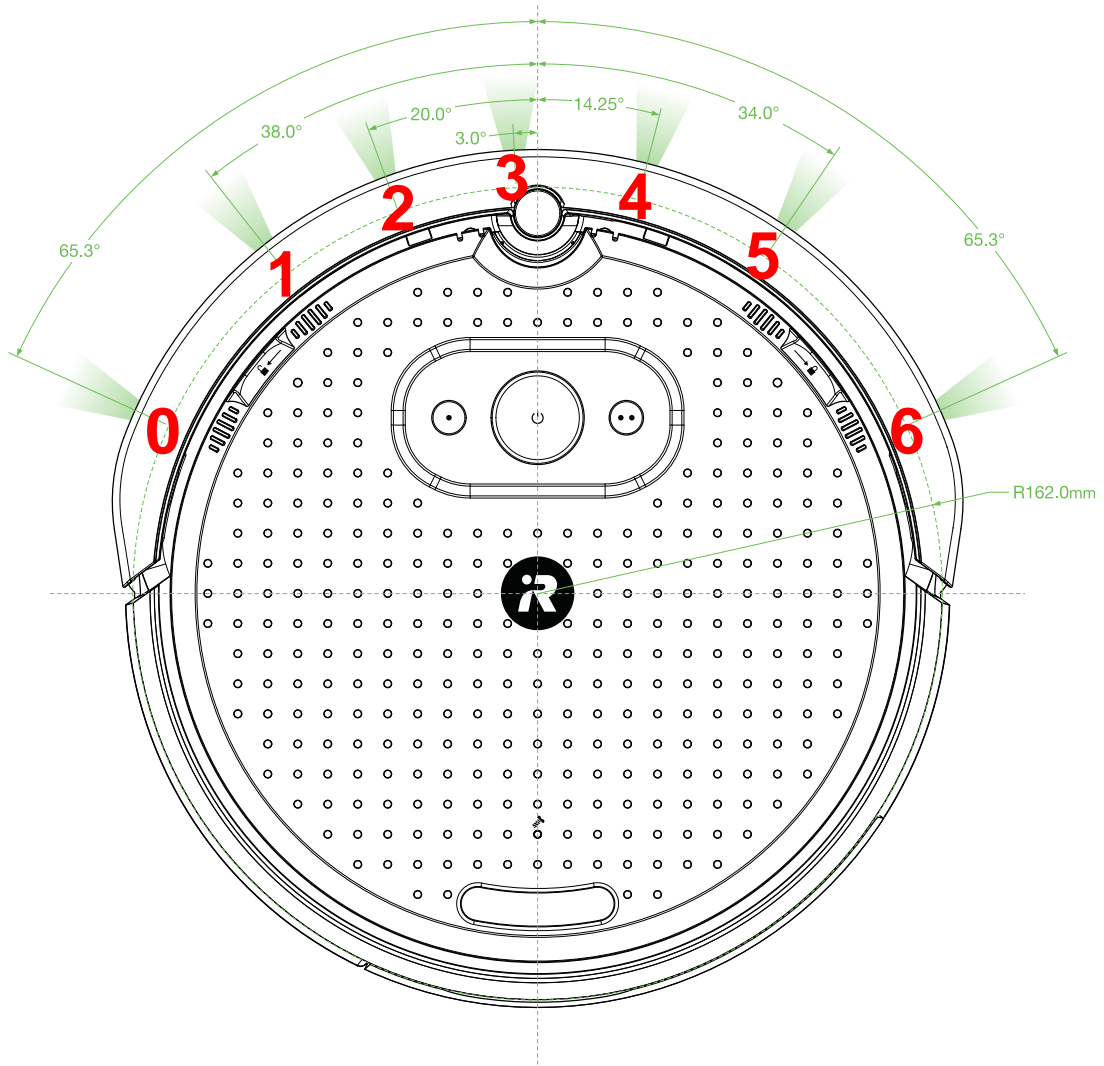


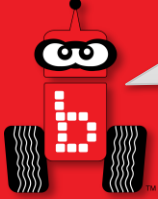
# Bump Cliff (IR Reflectance) Sensor Locations (Underside of robot)





# IR Sensor Locations





# Create 3 Sensor Functions

```
// Returns the value from a bump sensor, either 0 or 1. The sensor ID will be 0-4, with 0 being the farthest left and 4 being the farthest right.
```

```
• create3_sensor_bump(int sensor_id);
```

```
// Returns the value from a cliff sensor, with low numbers being for cliffs or edges and higher numbers being for normal surfaces. The sensor ID will be 0-3, with 0 being the farthest left and 3 being the farthest right.
```

```
• create3_sensor_cliff(int sensor_id);
```

```
// Returns the value from an IR sensor, with low numbers being for objects further away and high number being for objects that are closer. The sensor ID will be 0-6, with 0 being the farthest left and 6 being the farthest right.
```

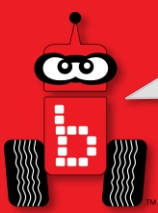
```
• create3_sensor_ir(int sensor_id);
```

```
// Returns the orientation about the Z axis in radians. The same function can be called but for x or y. However, the Create most likely won't be in a situation where it is rotated about either of those axes.
```

```
• create3_get_euler_z();
```

```
// Returns a Create3Pose which is a struct containing a Create3Vector3 that holds the current position and a quaternion which holds the current orientation.
```

```
• create3_pose_get();
```



# Create 3 Velocity Functions

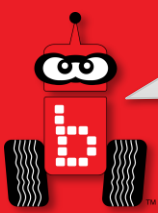
```
// Publisher: Sets the Create 3 velocity in the linear x direction (forward) in
meters/second and its rotational speed in radians/second in the angular z (around the
center).
• create3_velocity_set_components(double linear_x, double angular_z);

// Returns the current velocity about the z axis. This is useful for detecting speed
during rotations or arcs.
• create3_velocity_get_angular_z();

// Returns the current velocity in the linear x direction. This is useful for when the
Create is driving straight forward.
• create3_velocity_get_linear_x();

// Returns a Create3 Twist which is a struct containing a linear x velocity and an angular
z velocity.
• create3_velocity_get();

// Returns a Create3Odometry which is a struct containing a Create3Pose and a Create3Twist.
A Create3Pose contains a Create3Vector3 position and a Create3 quaternion. A
Create3Twist contains a linear x velocity and an angular z velocity.
• create3_odometry_get();
```

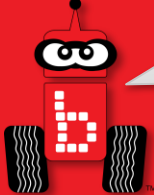


# Create 3 Light Ring

```
// Creates a Create3 LED color struct which can be used in other light ring functions
using typical r, g, b values.
• create3_led_color(int r, int g, int b);

// Creates a Create3 Light Ring struct which can be used in other light ring functions.
Each led can be set to a color. If you are using one color, you can set them all to the same
variable made using a single Create3LedColor variable.
• create3_lightring(
    Create3LedColor led0,
    Create3LedColor led1,
    Create3LedColor led2,
    Create3LedColor led3,
    Create3LedColor led4,
    Create3LedColor led5,
);

// Sets a Create3 LED animation to run for a specified amount of time. The two animation
types are Create3BlinkLights and Create3SpinLights and must be typed like that for
the animation_type argument. The Create3LightRing can be made using other light ring
functions. The time is in seconds and can be specified down to the decimal. This command is
good for learning structs as well as creating indicators during your code.
• create3_led_animation(
    Create3LedAnimationType animation_type,
    Create3LightRing lightring,
    double max_runtime
);
```



# Create 3 Connect/Disconnect Functions

All programs used with the *Create 3*

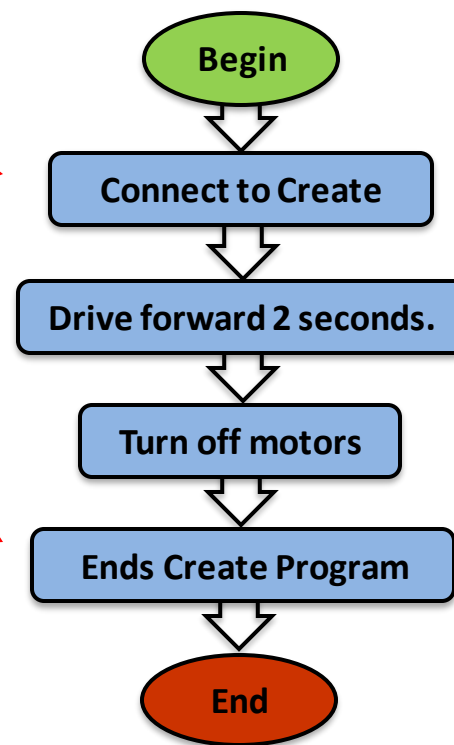
**MUST** start with

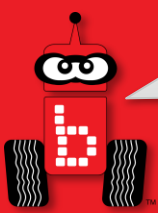
```
create3_connect();
```

and end with

```
create3_wait();
```

Flowchart





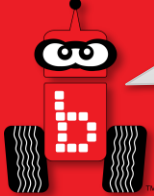
# Tournament Templates

```
int main() // for your Create robot
{
    create3_connect();
    // other initial items as needed (servo and camera calibration for example)

    wait_for_light(0); // change the port number to match the port you use
    shut_down_in(119); // shut off the motors and stop the robot after 119 seconds

    // Your code
    create3_wait();
    return 0;
}
```





# Moving the *Create 3* forward a set distance at a set velocity

**Description:** Write a program for the KIPR Robotics controller that drives the **Create3** forward to a distance of .5m at .46m/second for and then stops.

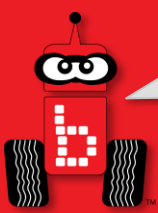
**Analysis:** What is the program supposed to do?

## Pseudocode

1. Connect to Create 3.
2. Drive forward for .5 m at .46m/sec
3. Create3 Wait.
4. End the program.

## Comments

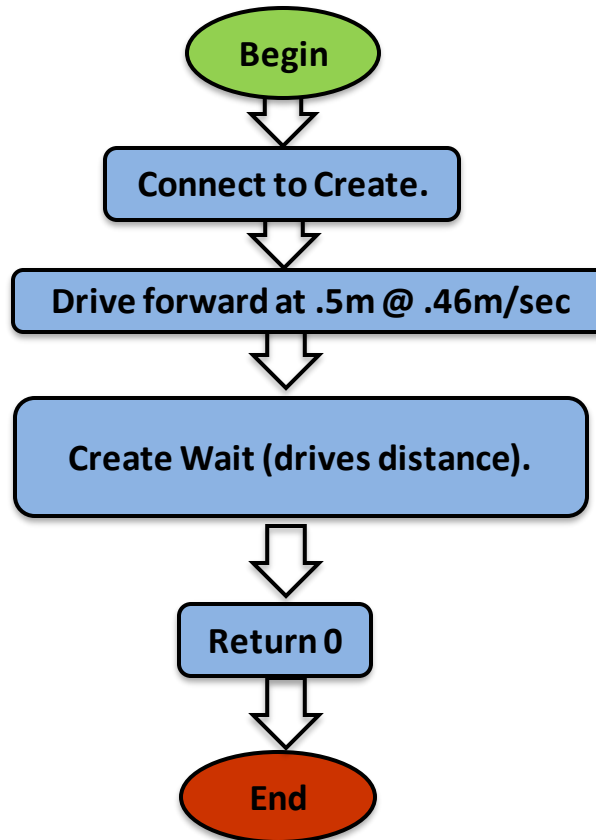
```
// 1. Connect to Create3.  
// 2. Drive forward to .5m @.46m/sec  
// 3. Gives Create 3 time to finish.  
// 4. End the program.
```

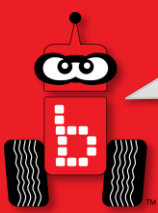


# Moving the *Create 3* forward a set distance at a set velocity

## Analysis:

### Flowchart





# Moving the *Create 3* forward a set distance at a set velocity

## Solution:

## Source Code

### Pseudocode

1. Connect to Create3.
2. Drive forward .5m @ .46m/sec
3. Create 3 waits to finish
4. Program ends

```
int main()
{
    create3_connect();
    create3_drive_straight (0.5, 0.46)
    create3_wait();

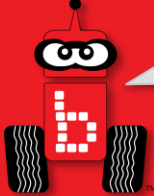
    return 0;
}
```

Speed in meters/second

Distance in meters

**\*REMEMBER .46m/sec is the fastest velocity for the Create3 Options are 0 to .46\***

**Execution:** Compile and run your program.



# Moving the *Create 3* backward a set distance at a set velocity

## Solution:

## Source Code

### Pseudocode

1. Connect to Create3.
2. Drive backward -0.5m @ 0.46m/sec
3. Create3 waits to finish
4. Program ends

```
int main()
{
    create3_connect();
    create3_drive_straight (-0.5, 0.46)
    create3_wait();

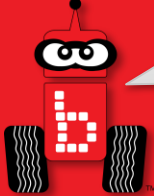
    return 0;
}
```

Speed in meters/second

Distance in meters

**\*REMEMBER .46m/sec is the fastest velocity for the Create3 Options are 0 to .46\***

**Execution:** Compile and run your program.



# Create 3 Turning - Rotation Functions

Rotating in Place can be controlled by desired angle or radian you want to rotate to and the speed at which you want to rotate. Both arguments are floats and will take a decimal

```
create3_rotate_degrees(float angle, float max_angular_speed);
```

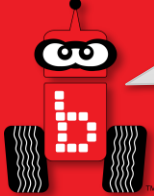
```
create3_rotate_degrees(90, 45);
```

Rotates to 90degrees @ 45 degrees/second (Minimum 15 - Maximum 100)

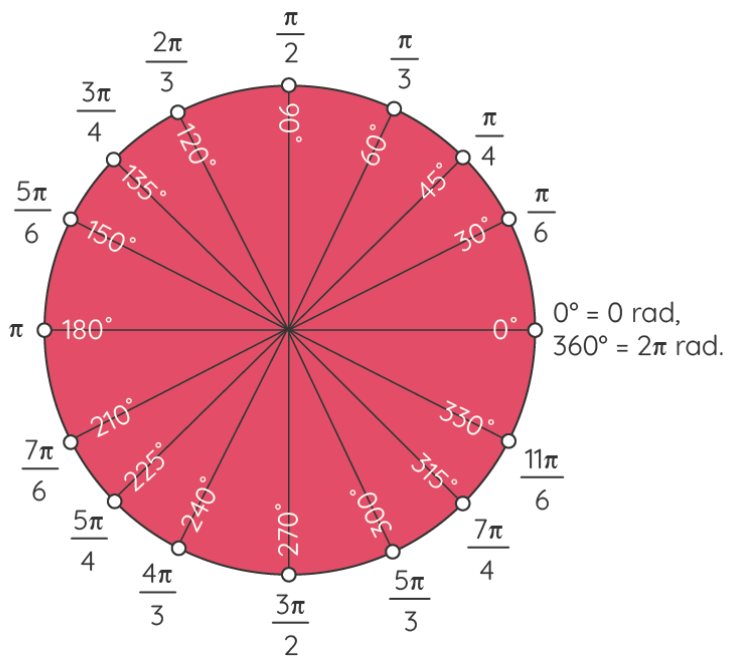
If you want to use Si units, Radians or rad then:

```
create3_rotate_radians(0.25, 0.25);
```

Rotates to .25rad @ .25 rad/second



# Degrees and Radian



- **Angles are measured using two basic units: degrees and radians (SI units).**
- **One complete counterclockwise revolution is equal to  $2\pi$  rad in radians.**
- **$1^\circ$  equals 0.017453 radians and 1 rad equals  $57.2958^\circ$ .**
- **To convert an angle from radians to degrees, we multiply it by  $180^\circ/\pi$ .**
- **To convert an angle from degrees to radians, we multiply it by  $\pi/180^\circ$**

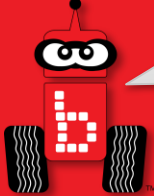


# Using *Create 3* Rotation Turn Functions

## Examples:

```
int main()
{
    create3_connect();
    create3_rotate_degrees(90, 45);
    create3_wait();
    return 0;
}
```

```
int main()
{
    create3_connect();
    create3_rotate_radians(.25, .25);
    create3_wait();
    return 0;
}
```



# Create3 Turning- Arc Functions

Rotating in Place can be controlled by desired angle or radian you want to rotate to and the speed at which you want to rotate. Both arguments are floats and will take a decimal

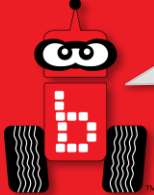
```
create3_drive_arc_degrees(float radius, float angle, float max_linear_speed);
```

```
create3_drive_arc_degrees (.25, 90, .25);
```

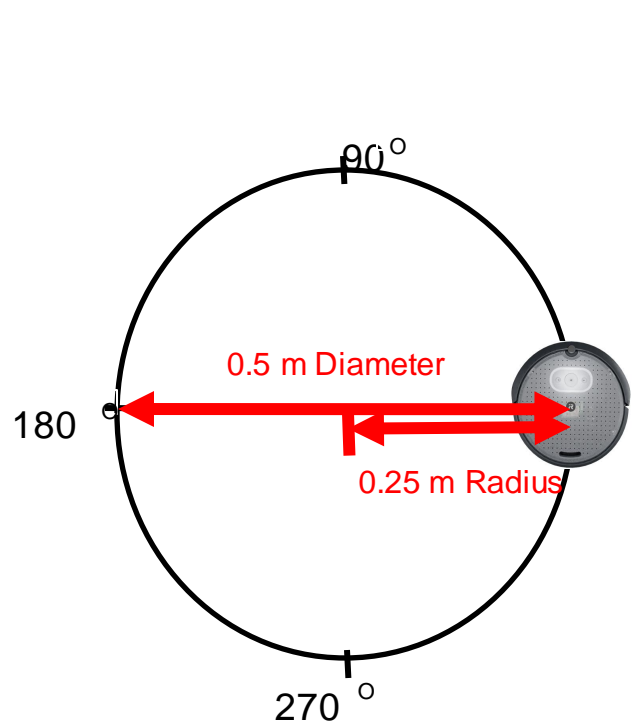
Radius of arc in meters, turn in angle or rad, speed in m/sec

```
create3_drive_arc_radians (.25, .25, .25);
```

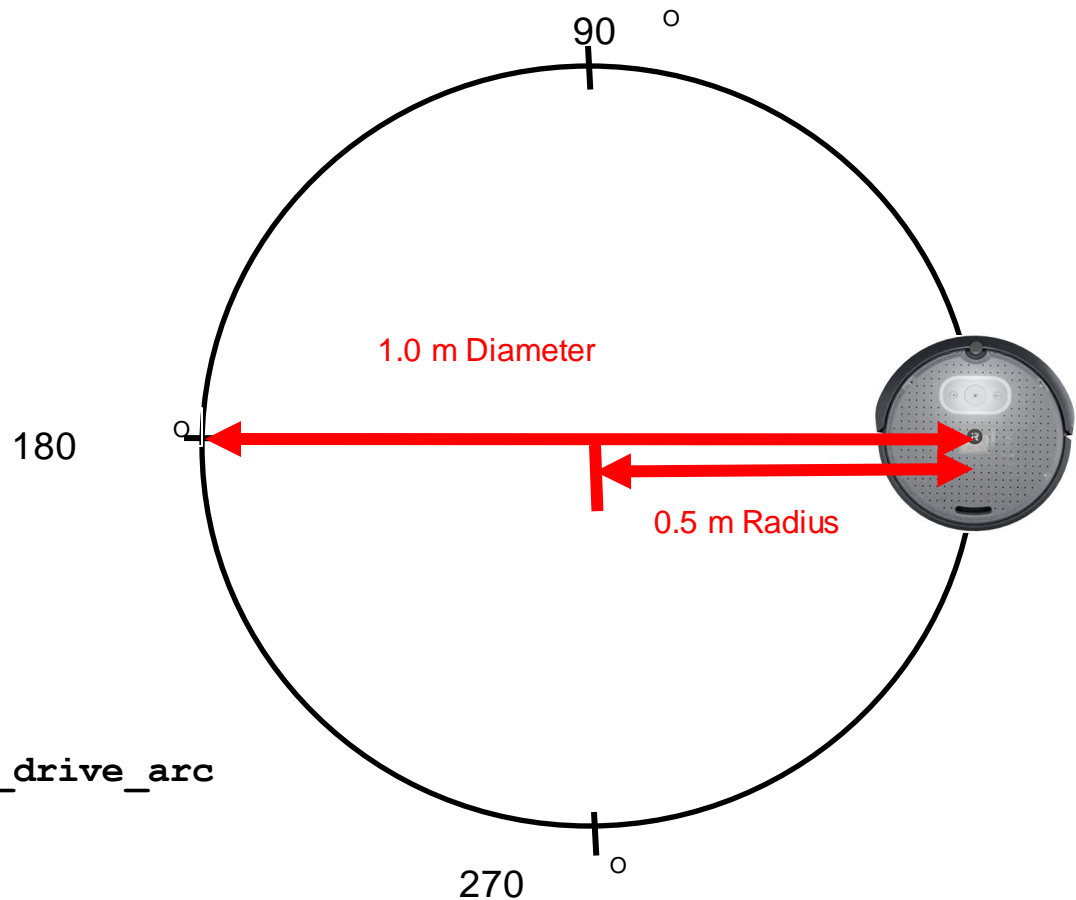




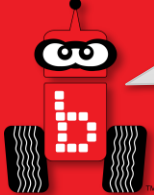
# Arc Radius (meters)



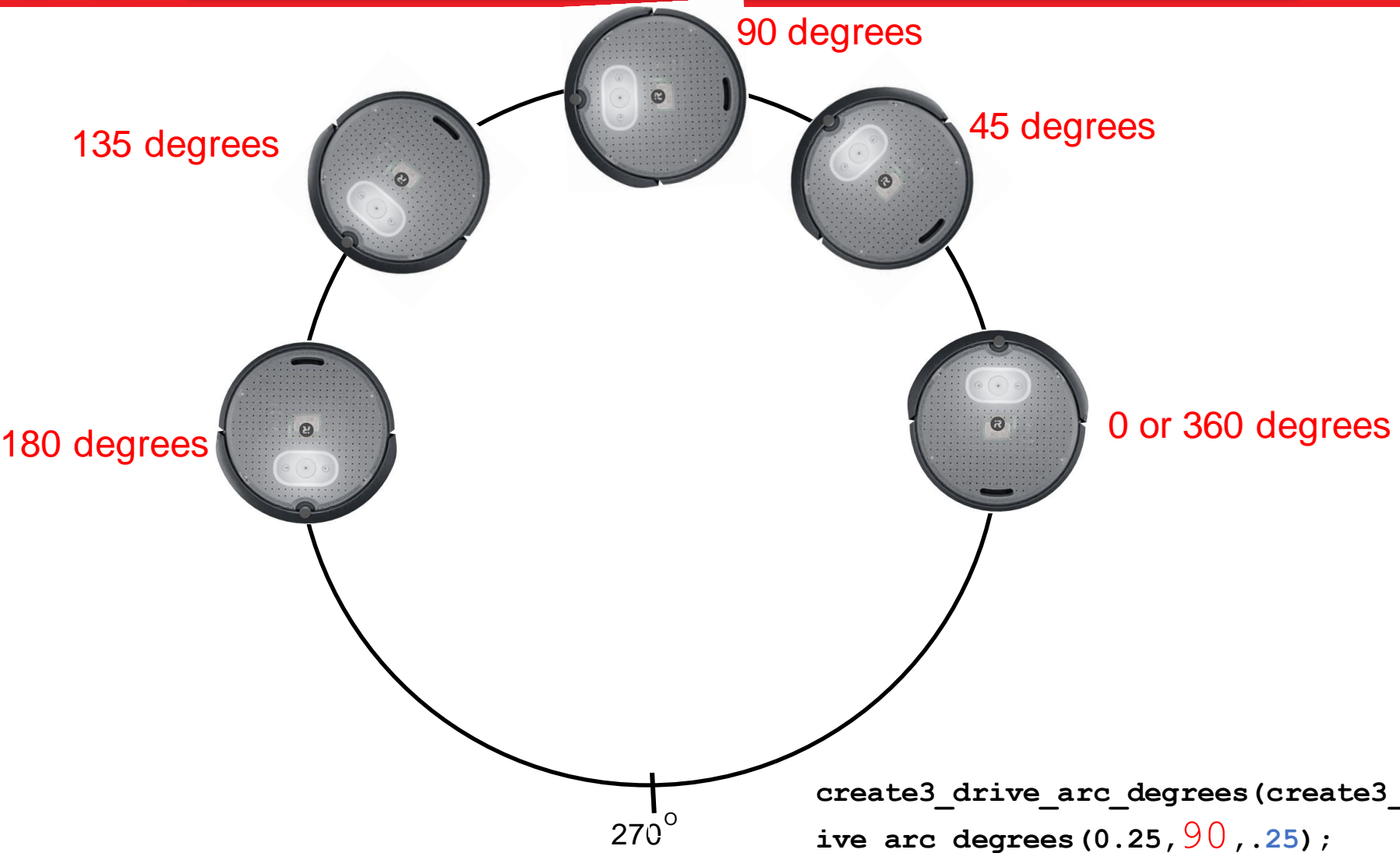
```
create3_drive_arc_degrees(create3_drive_arc_degrees(0.25, 90, .25);
```



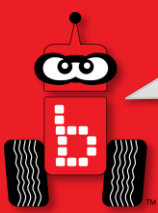
```
create3_drive_arc_degrees(create3_drive_arc_degrees(0.50, 90, .25);
```



# Arc Degrees (degrees)



```
create3_drive_arc_degrees(create3_dr  
ive_arc_degrees(0.25, 90, .25);
```

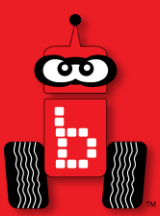


# Using *Create3* Arc Turn Functions

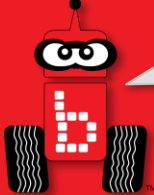
## Examples:

```
int main()
{
    create3_connect();
    create3_drive_arc_degrees(.25, 90, .25);
    create3_wait();
    return 0;
}
```

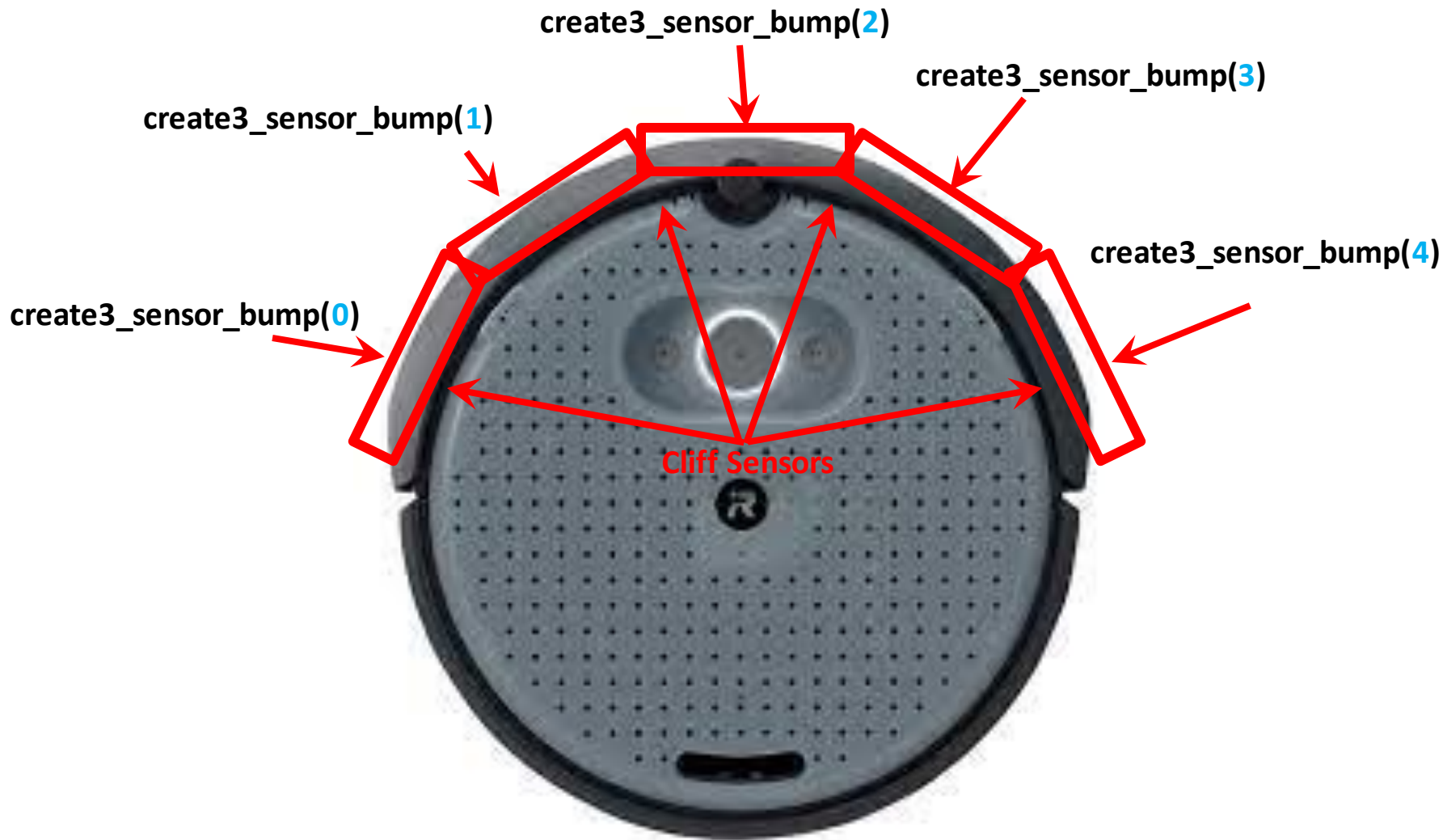
```
int main()
{
    create3_connect();
    create3_drive_arc_radians(.25, .25, .25);
    create3_wait();
    return 0;
}
```

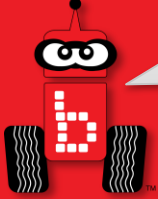


# iRobot *Create 3* Sensors



# Create 3 Sensor Functions





# Drive Until Bumped

**Description:** Write a program for the KIPR Wombat that drives the *Create3* forward until a bumper is pressed, and then stops.

**Analysis:** What is the program supposed to do?

## Pseudocode

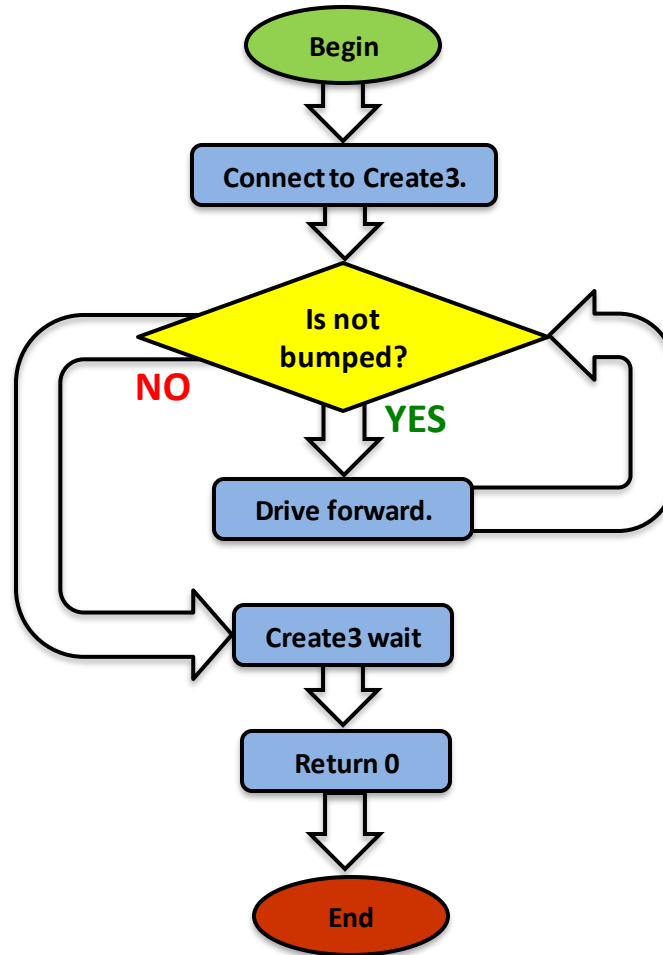
1. Connect to Create 3.
2. Drive forward for .25 m/sec @ 0 degrees (straight)
3. Create3 Wait.
4. End the program.

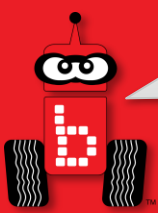
## Comments

```
// 1. Connect to Create 3.  
// 2. Loop: Is not bumped?  
// 2.1. Drive forward.  
// 3. Give Create commands time to  
complete  
// 4. End the program.
```



# Drive Until Bumped





# Drive Until Bumped

```
int main()
{
    create3_connect();
    double velocity = 0.15;
    while (velocity != 0)
    {
        if (create3_sensor_bump(1) == 1 || create3_sensor_bump(2) == 1 || create3_sensor_bump(3) == 1)
        {
            printf("Bumped\n");
            velocity = 0;
        }
        create3_velocity_set_components(velocity, 0);
    }
    create3_wait();
    return 0;
}
```

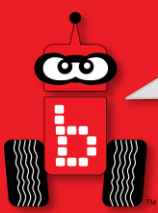
Bump sense is improved if you look at 3 bump sensors at the same time. 1 or 2 or 3

```
create3_veocity_set_components(.25, 0);
```

speed in m/sec

Rotation speed in degrees 0 = straight



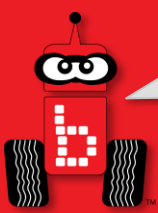


# Printing Create3 Cliff Sensor Values

```
int main()
{
    create3_connect();
    while (a_button() == 0)
    {
        printf("value is : %d \n", create3_sensor_cliff(2));
        msleep(300);
    }

    create3_wait();
    return 0;
}
```

Cliff (reflectance) sensors can be used to find a black line or to follow a black line. Printing the values from your board and our tournament boards to the screen to set your threshold is important.

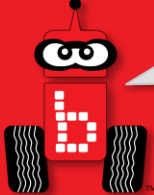


# Find Black With the Create3

**Description:** Make the iRobot Create3 find a line.

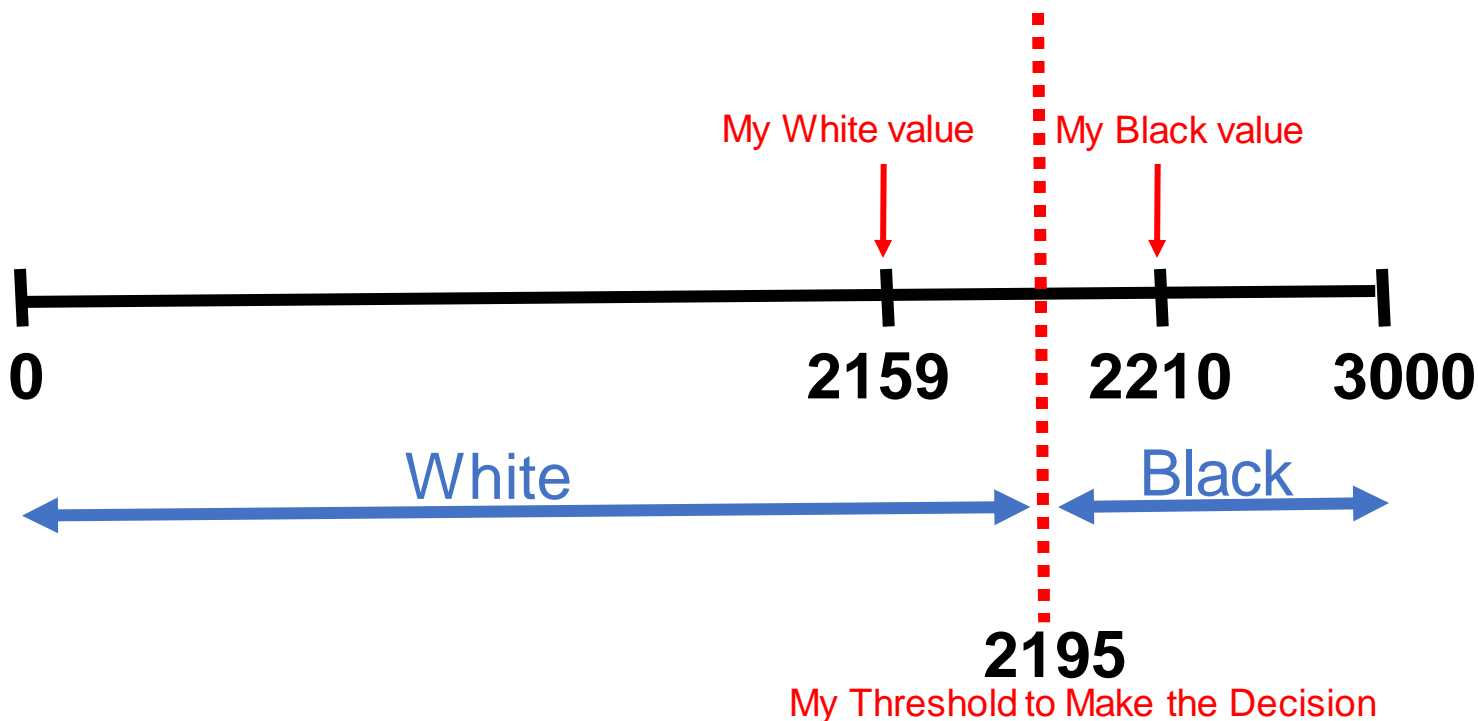
What you need to know: You can use the Create front cliff sensor and the `create3_velocity_set_components()` commands to accomplish this

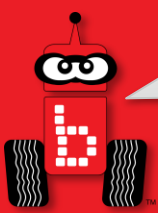
Point to note: The scale and value for the `create3_sensor_cliff()` may differ from the values of the analog sensor ports on the Wombat. You may consider finding the black and white values for this sensor by printing values of each to your screen.



# Create3 Cliff Sensor Values

1. Using the previous slide to print values, document the values when your Create3 Cliff sensors are over white and over black





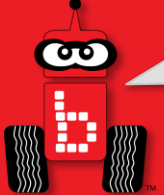
# Find Black Line

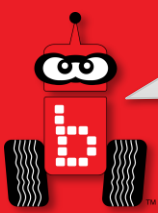
```
int main()
{
    create3_connect();
    double velocity = 0.15;
    while (velocity != 0)
    {
        if (create3_sensor_cliff(1) < 2195 || create3_sensor_cliff(2) < 2195)
        {
            printf("Found Black\n");
            velocity = 0;
        }
        create3_velocity_set_components(velocity, 0);
    }
    create3_wait();
    return 0;
}
```

```
create3_veocity_set_components(.25, 0);
```

speed in m/sec

Rotation in degrees 0 = straight





# Line Follow With the Create

**Description:** Make the iRobot Create3 follow a line. The Create3 will follow a black line.

**What you need to know:** You can use the Create front cliff sensor and the `create3_velocity_set_components()` commands to accomplish this

**Point to note:** The scale and value for the `create3_sensor_cliff()` may differ from the values of the analog sensor ports on the Wombat. You may consider finding the black and white values for this sensor by printing values of each to your screen.

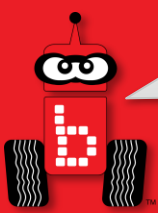


# Line Follow with Create3 Solution

```
int main()
{
    create3_connect();
    printf("Follow the non-yellow brick road!\n");

    while (a_button == 0)
    {
        if (create3_sensor_cliff(1) >= 2195)
        {
            create3_velocity_set_components(0.1, 0.3);
        }
        else
        {
            create3_velocity_set_components(0.1, -0.3);
        }
    }

    create3_wait();
    return 0;
}
```



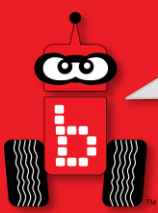
# Printing Create3 IR Sensor Values

```
int main()
{
    create3_connect();
    while (a_button() == 0)
    {
        printf("value is : %d \n", create3_sensor_ir(3));
        msleep(300);
    }

    create3_wait();
    return 0;
}
```

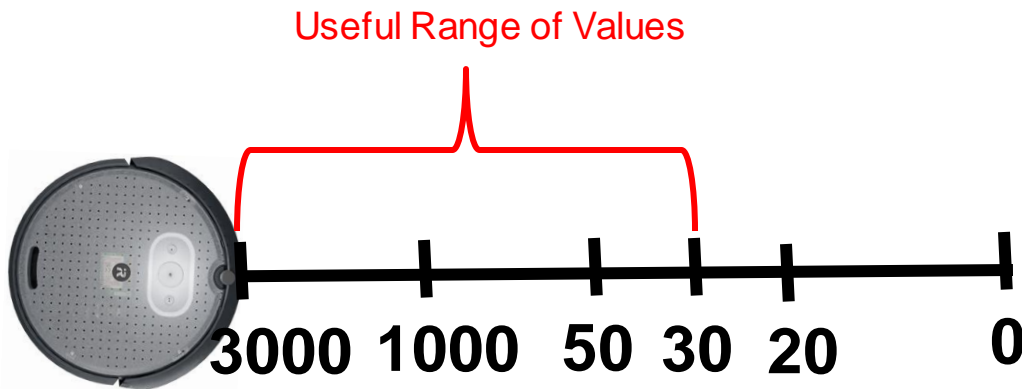
Cliff (reflectance) sensors can be used to find a black line or to follow a black line. Printing the values from your board and our tournament boards to the screen to set your threshold is important.

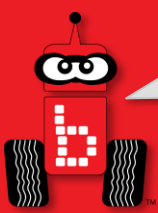




# Create3 IR Sensor Values

1. IR (Rangefinder) sensor values. Using the previous slide figure out what sensor values equate to distance





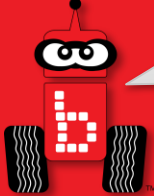
# Drive to Object (Game Piece or PVC)

```
int main()
{
  create3_connect();
  double velocity = 0.15;
  while (velocity != 0)
  {
    if (create3_sensor_ir(3) > 100 || create3_sensor_ir(4) > 100)
    {
      printf("Found Object\n");
      velocity = 0;
    }
    create3_velocity_set_components(velocity, 0);
  }
  create3_wait();
  return 0;
}
```

```
create3_veocity_set_components(.25, 0);
```

speed in m/sec

Rotation in degrees 0 = straight



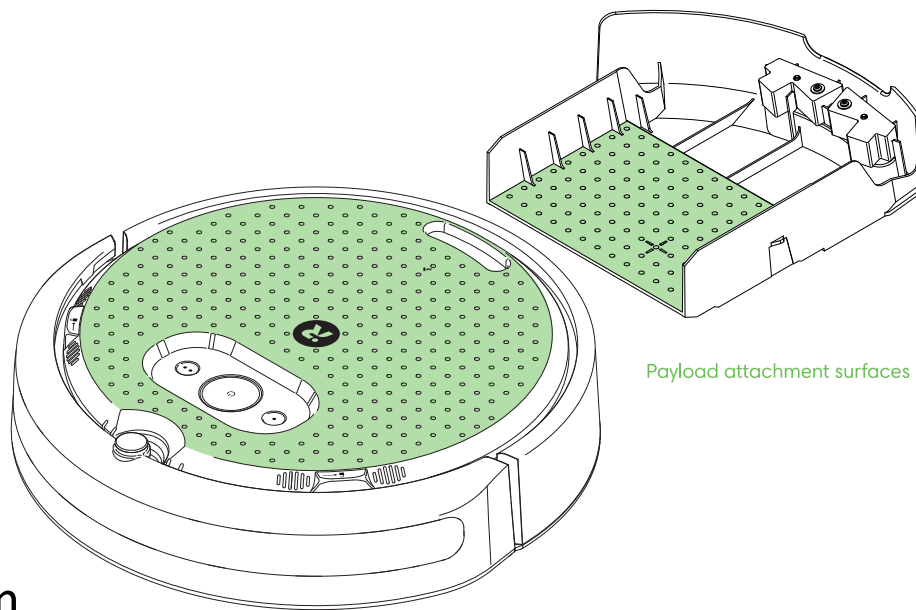
# Building on the *Create 3*

The Create 3 has two different surfaces where things may be attached. All hole spacing is 12mm so LEGO holes will line up every other hole.

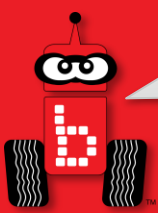
- Faceplate – The top surface of the Create 3. It can be twisted counter-clockwise to remove it and attach a payload.
- Internal Cargo Bay – Can be removed from the back to attach a payload inside.

All holes on the Create can have items attached using M3 screws (those are the same ones used to attach lever sensors to robots).

These holes may be drilled out to use larger screws at the user's own expense. Replacement faceplates may be purchased from iRobot.



Payload attachment surfaces



# Switching from Wifi to Ethernet

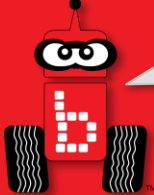
As of version v30.2.5 on the Wombat you can now switch between an Ethernet and a WiFi connection on the Wombat. The Ethernet connection has been shown to be more stable and less prone to issues in testing.

To switch between the two you will need:

- Ethernet cable
- USB-C to Ethernet Adapter
- Wombat on version 30.2.5 or higher
- Create 3 on version H.2.4 or higher
- Computer

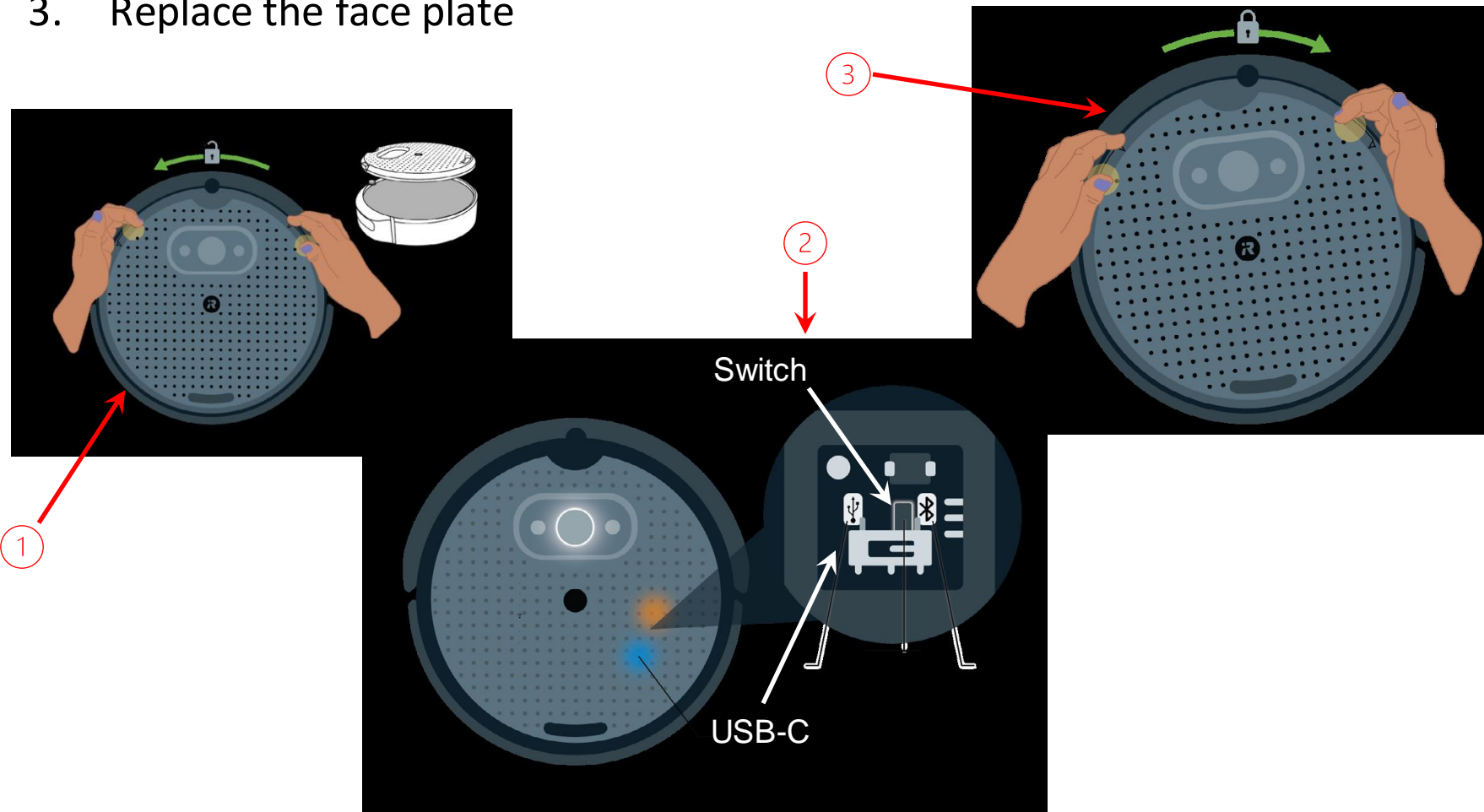
Make sure that your Wombat and Create 3 are on the correct versions or the following steps may not work.

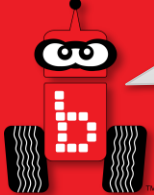
Teams that received their Create 3 earlier in the season or had an earlier workshop may not have received a USB-C to Ethernet adapter or Ethernet cable. If you need one, please email [tcorbly@kipr.org](mailto:tcorbly@kipr.org) about getting one sent to you.



# Switching from Wifi to Ethernet

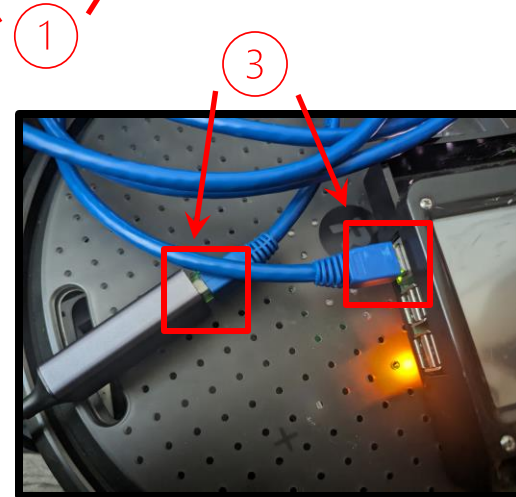
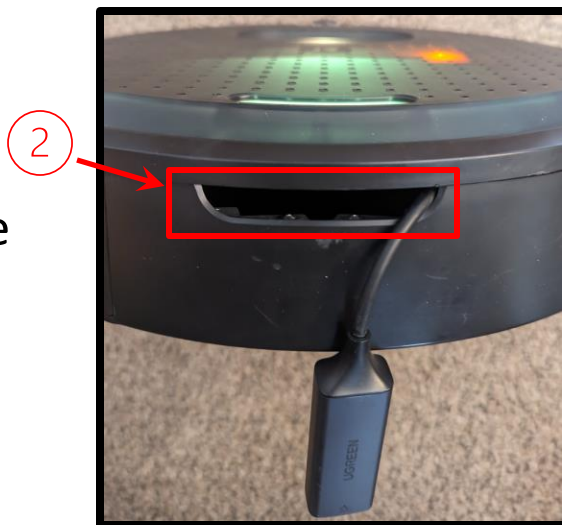
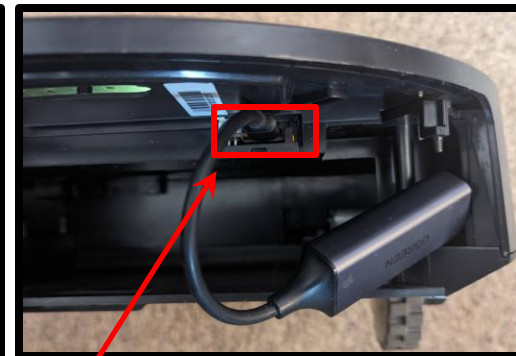
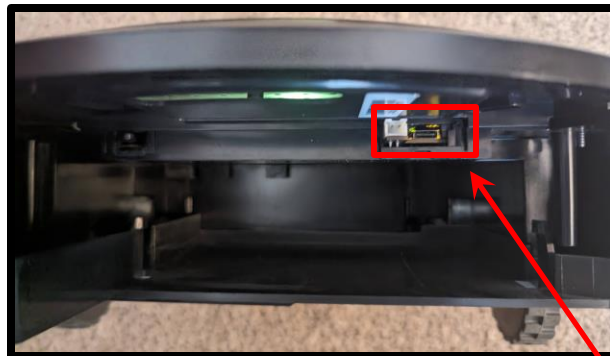
1. Turn the face plate of the Create to the left to unlock
2. Flip the switch to the USB-C side
3. Replace the face plate

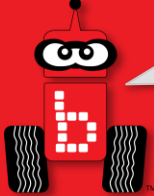




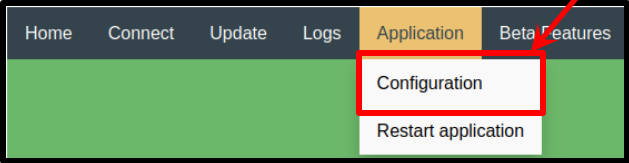
# Switching from Wifi to Ethernet

1. Pull out the cargo bay from your Create 3. You will see a USB-C port inside that you can plug your adapter into.
2. Put the cargo bay back in and run the adapter out the hole in the back of the Create 3.
3. Plug your Ethernet cable into the adapter and your Wombat
4. Place the Create 3 on the dock.





# Switching from Wifi to Ethernet

1. Connect to your Wombat and go to the IP on your Create 3 page in your browser.
2. Go to the Application -> Configuration tab. 
3. Change the "Address and port of Fast DDS discovery server" to: **192.168.186.3:11811**
4. Click the green Save and wait for it to return to the page.
5. Click Restart application and wait for the Create 3 to make a chime and return to a solid white light on top.

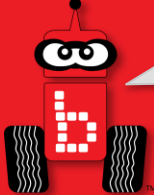


The screenshot shows the 'App config' interface. The 'Main Configuration' section includes the following fields:

- ROS 2 Domain ID (default 0): 0
- ROS 2 Namespace: (empty)
- RMW\_IMPLEMENTATION: rmw\_fastrtps\_cpp
- Enable Fast DDS discovery server?
- Address and port of Fast DDS discovery server: 192.168.186.3:11811

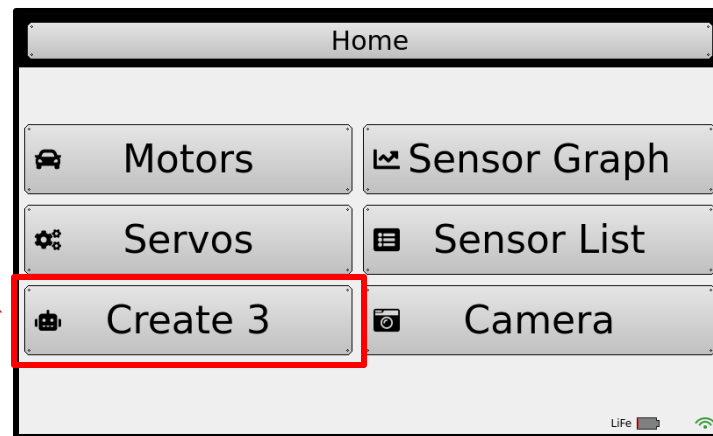
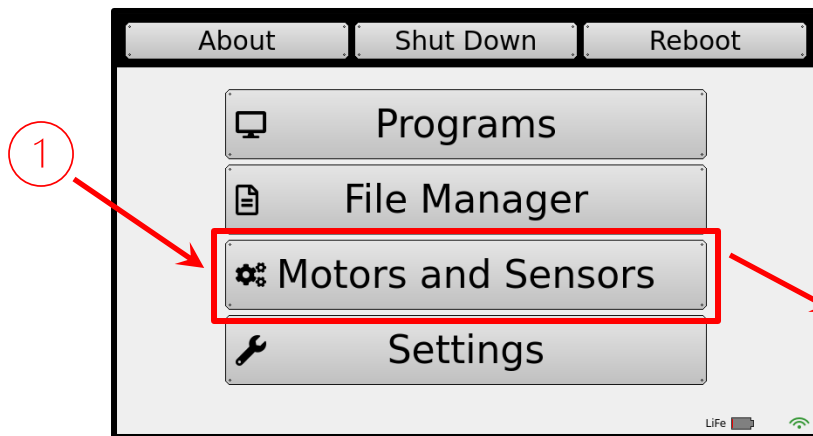
Annotations in the image:

- 2: Points to the 'Configuration' tab in the application menu.
- 3: Points to the 'Address and port of Fast DDS discovery server' input field.
- 4: Points to the 'Save' button.
- 5: Points to the 'Restart application' button.

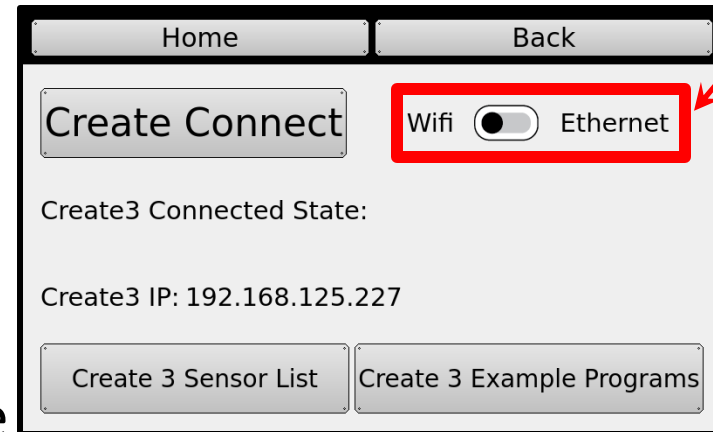


# Switching from Wifi to Ethernet

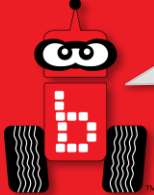
1. On your Wombat, go to Motors and Sensors -> Create 3



2. On the Create 3 page, tap the Wifi – Ethernet toggle.
3. When it asks if you would like to switch, say Yes and wait for it to reboot.
4. Return to this page after the reboot and test it by clicking Create Connect.

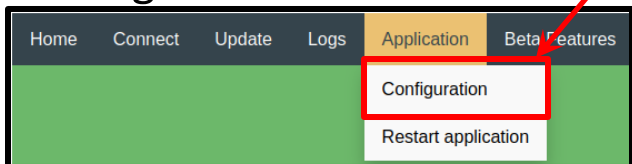




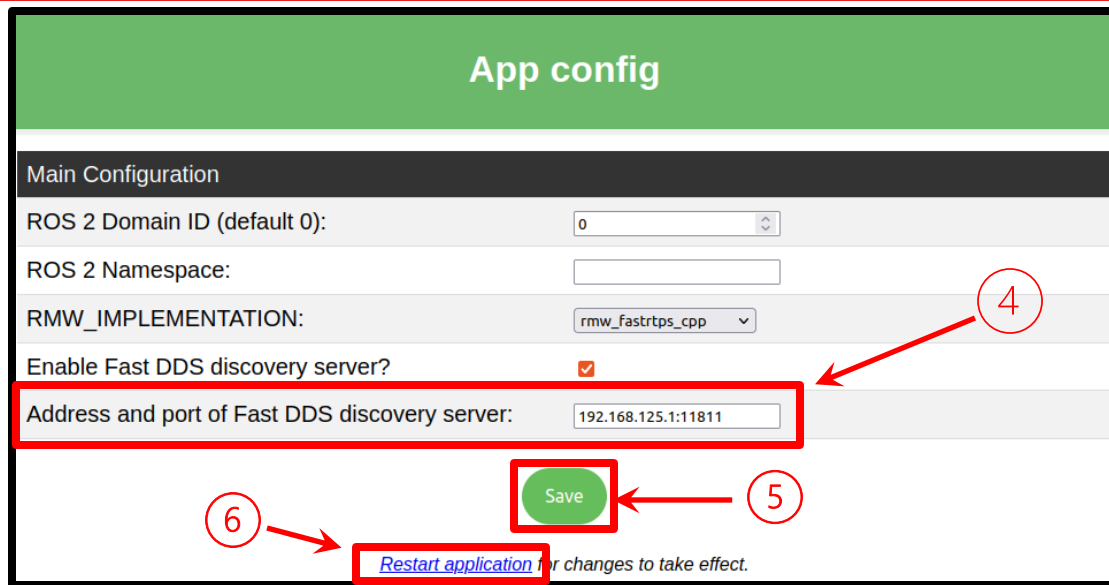


# Switching from Ethernet to Wifi

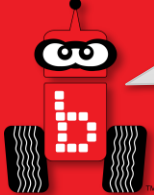
1. Unplug your Ethernet Cable from the Wombat and the Create 3.
2. Connect to your Wombat and go to the IP on your Create 3 page in your browser.
3. Go to the Application -> Configuration tab.



4. Change the "Address and port of Fast DDS discovery server to: **192.168.125.1:11811**
5. Click the green Save and wait for it to return to the page.
6. Click Restart application and wait for the Create 3 to make a chime and return to a solid white light on top.

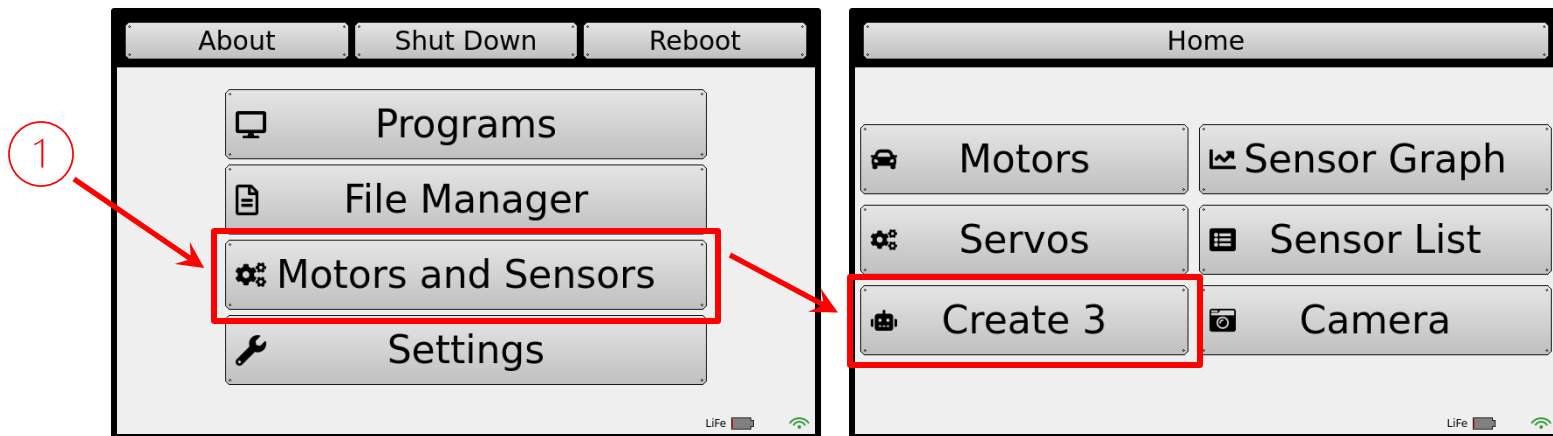


**Note: If the Create shows a red light while rebooting, pull it off the dock briefly and put it back on and wait for it to return to white.**

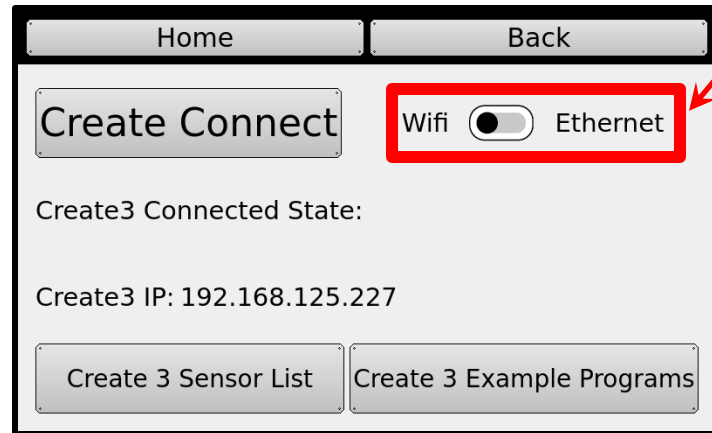


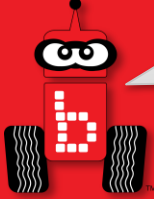
# Switching from Ethernet to Wifi

1. On your Wombat, go to Motors and Sensors -> Create 3



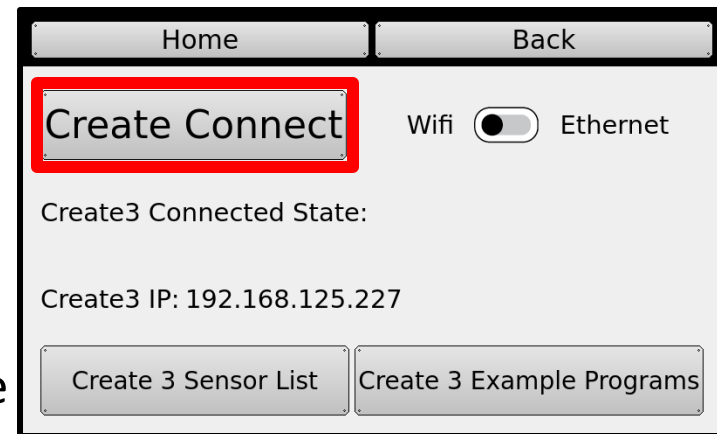
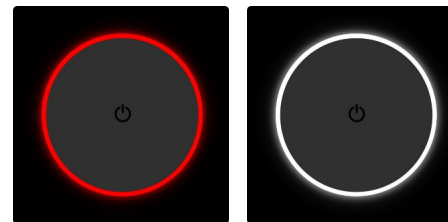
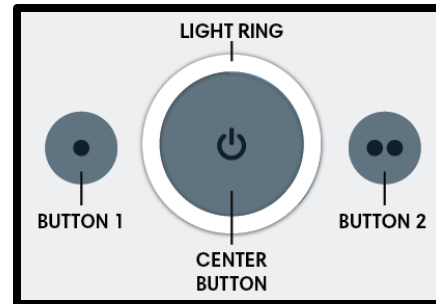
2. On the Create 3 page, tap the Wifi – Ethernet toggle.
3. When it asks if you would like to switch, say Yes and wait for it to reboot.

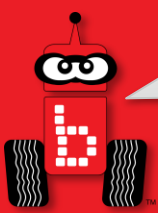




# Switching from Ethernet to Wifi

1. Hold down the center power button on your Create 3 until it powers down. Place it back on the dock.
2. It will most likely turn red. To fix this, pull it off the dock briefly and then put it back on and wait for it to return to a solid white.
3. Reboot your Wombat.
4. Wait a couple minutes after reboot before trying the Create Connect button on the Create 3 page.
5. If it fails to connect or the UI crashes, reboot the Wombat once more.



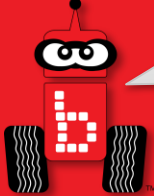


# Create 3 Troubleshooting

It is common for the Create 3 to experience bugs as the KIPR software for it is new as well as the Create 3 platform. If you are uncomfortable using the Troubleshooting techniques in upcoming slides always feel free to contact KIPR staff at:

- Phone: 405-579-4609
- Email: [tcorbly@kipr.org](mailto:tcorbly@kipr.org), [eharrington@kipr.org](mailto:eharrington@kipr.org), [support@kipr.org](mailto:support@kipr.org)



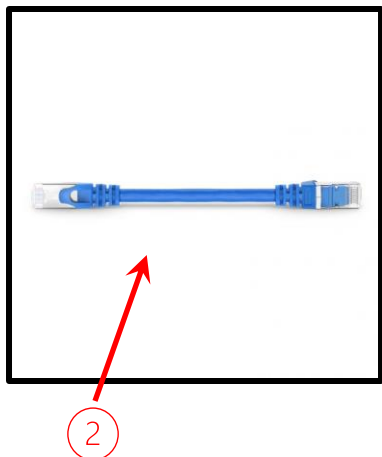


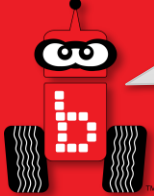
# Create 3 Troubleshooting

It is common to lock up the Create 3 if trying to force too many processes at once. This lockup looks like a spinning white light ring that lasts a long time.

You will need:

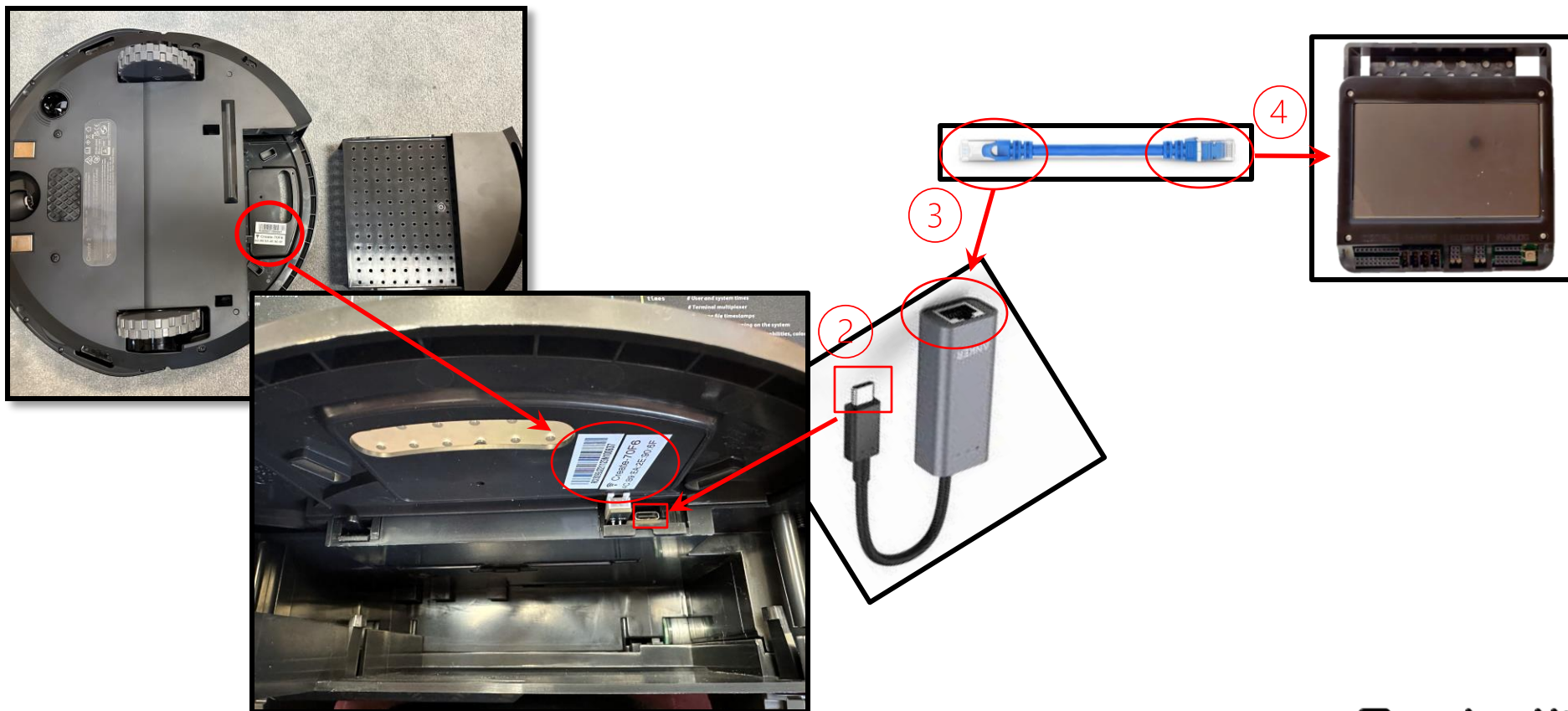
1. USB-C to ethernet adapter (not included)
2. Ethernet cable (not included)
3. Your Wombat
4. Your Create 3
5. Keyboard

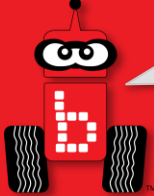




# Create 3 Troubleshooting

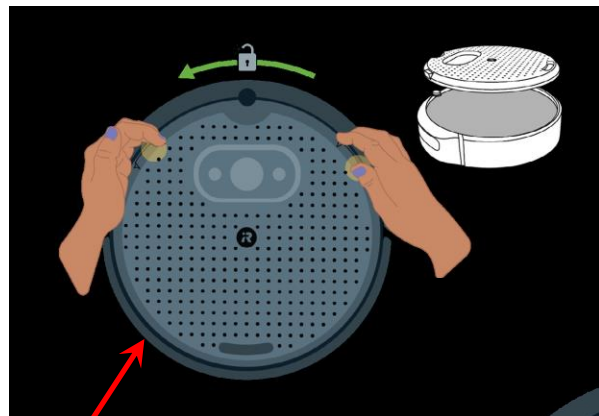
1. Open the cargo bay on the Create
2. Plug in your USB-C to Ethernet adapter to the Create
3. Plug one end of the Ethernet cable to your adapter
4. Plug the other end of the Ethernet cable to your Wombat



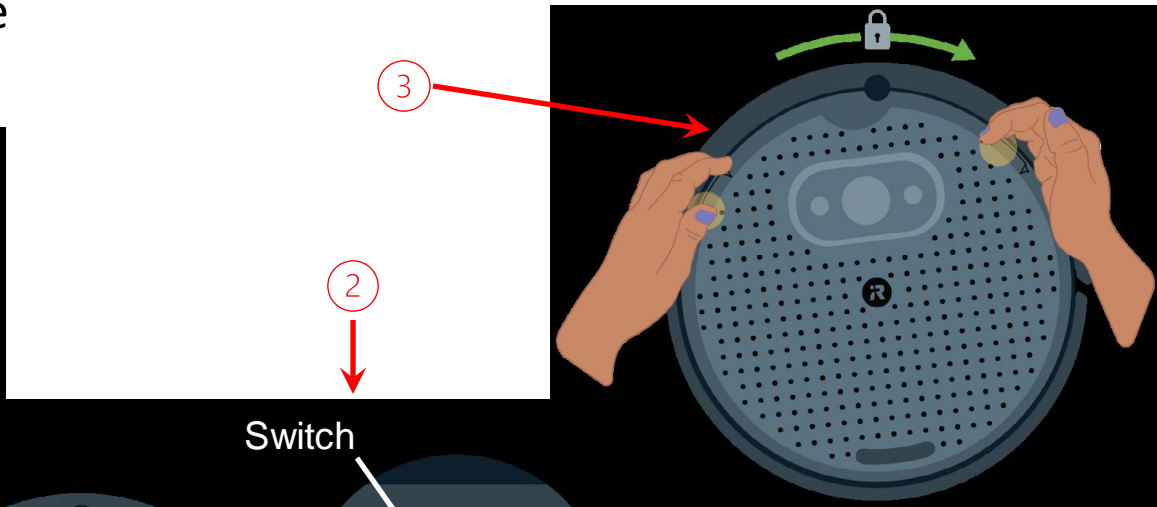


# Create 3 Troubleshooting

1. Turn the face plate of the Create to the left to unlock
2. Flip the switch to the USB-C side
3. Replace the face plate

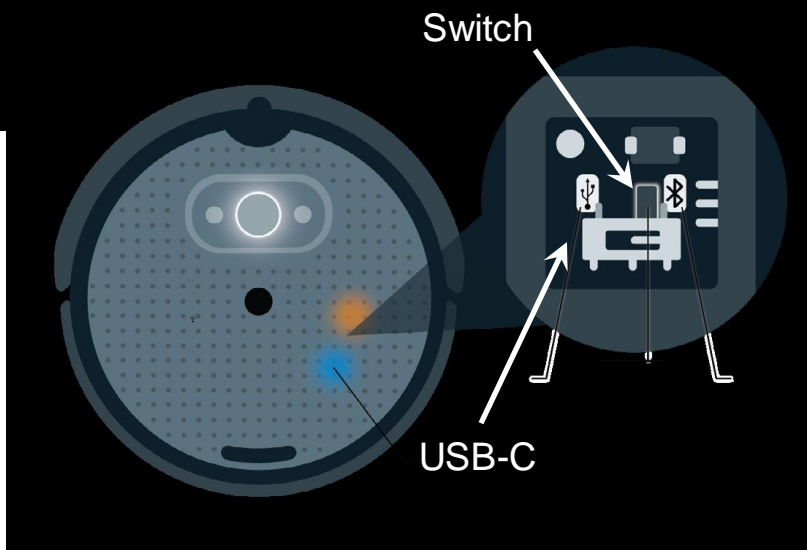


1



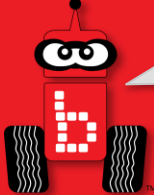
3

2



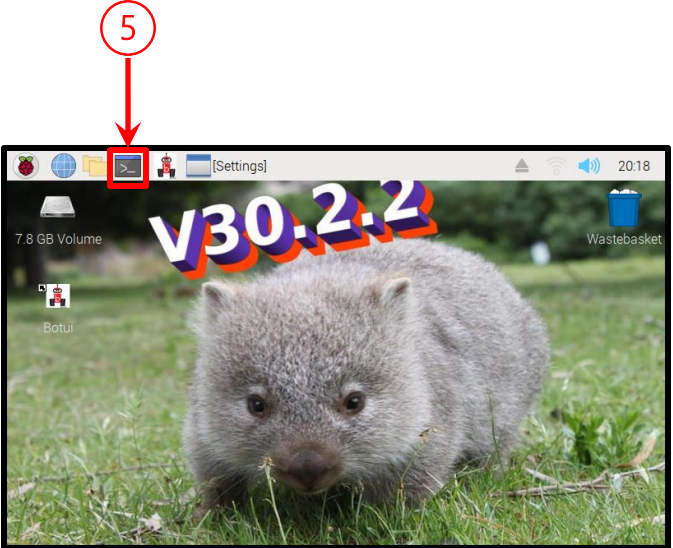
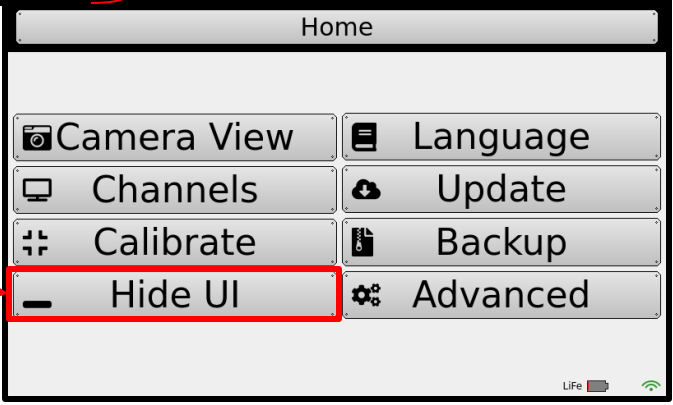
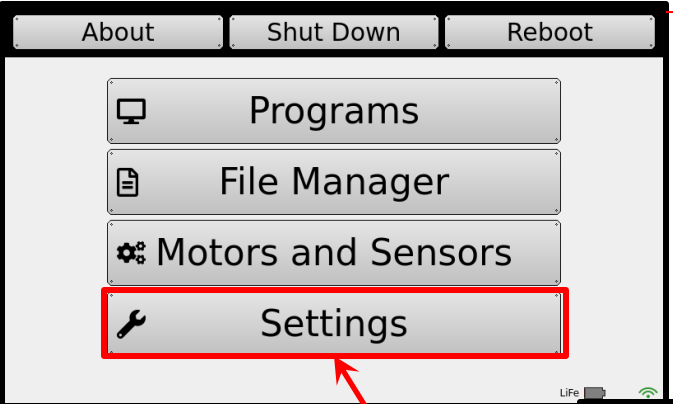
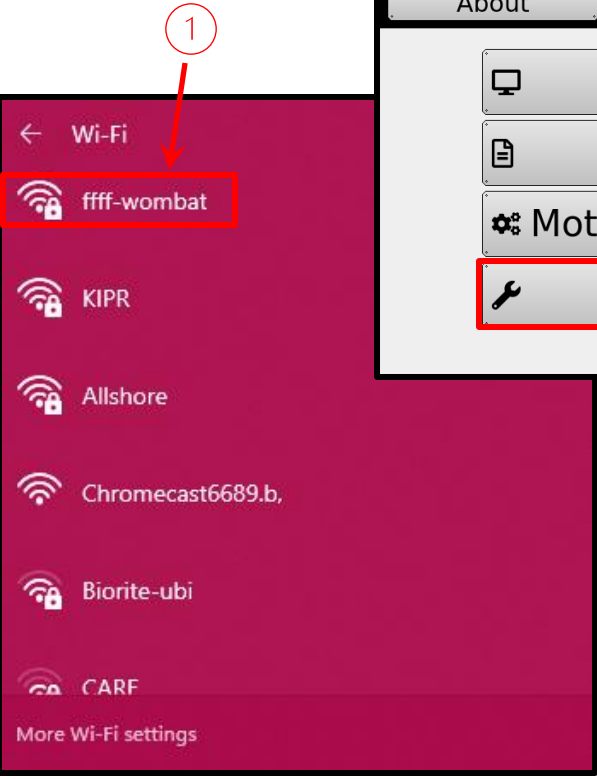
Switch

USB-C

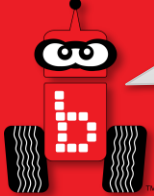


# Create 3 Troubleshooting

1. Connect to your Wombat's network on your computer's Wi-Fi settings
2. Navigate to the Home page on your Wombat
3. Click Settings
4. Click Hide UI
5. Click the Terminal icon at the top of the screen







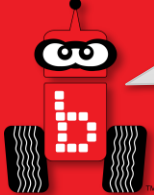
# Create 3 Troubleshooting

1. Type in “**sudo ip a add 192.168.186.3/24 dev eth0**”
2. Type this until you see “**RTNETLINK answers: File Exists**”
3. Type “**ifconfig**”
4. Verify eth0’s inet shows “**192.168.186.3**”

```
kivr@wombat:~$ sudo ip a add 192.168.186.3/24 dev eth0
kivr@wombat:~$ sudo ip a add 192.168.186.3/24 dev eth0
RTNETLINK answers: File exists
kivr@wombat:~$ ifconfig
eth0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 192.168.186.3 netmask 255.255.255.0 broadcast 0.0.0.0
    ether b8:27:eb:5e:0b:14 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 2575 bytes 756133 (738.4 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 2575 bytes 756133 (738.4 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.125.1 netmask 255.255.255.0 broadcast 192.168.125.255
```



# Create 3 Troubleshooting

1. On your computer's browser, type "**192.168.186.2**" ← This is different than what you type in previous slide!
2. Find the About button at the top right of the Create's webserver
3. Scroll all the way to the bottom of the About Page
4. Select "Factory Reset"
5. You will have to go through the set-up process again (slide 4)

The screenshot shows the webserver interface with a navigation bar at the top containing: Home, Connect, Update, Logs, Application, Beta Features, and About. The main content area displays system information, including system date, uptime, memory usage, and network configurations for eth0, lo, and wlan0. A terminal window is open, showing the command `>>>>>> system hciconfig` and the output `Factory_Reset`. A red arrow points from the 'About' button in the top right to the terminal window. Another red arrow points from the terminal window to a red box containing `>>>>>> system hciconfig` and `Factory_Reset`. A red circle with the number '4' is placed near the arrow pointing to the terminal window.