KISS INSTITUTE FOR PRACTICAL ROBOTICS

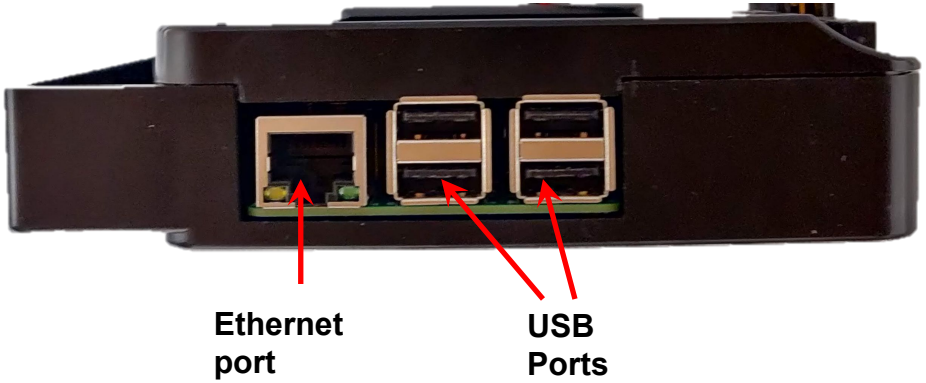
Botball®

Color Camera

Color Camera

For this activity, you will need the **camera**.

- The camera plugs into one of the USB (type A) ports on the back of the Wombat.
- **Warning:** Unplugging the camera while it is being accessed can freeze the Wombat, requiring it to be rebooted.

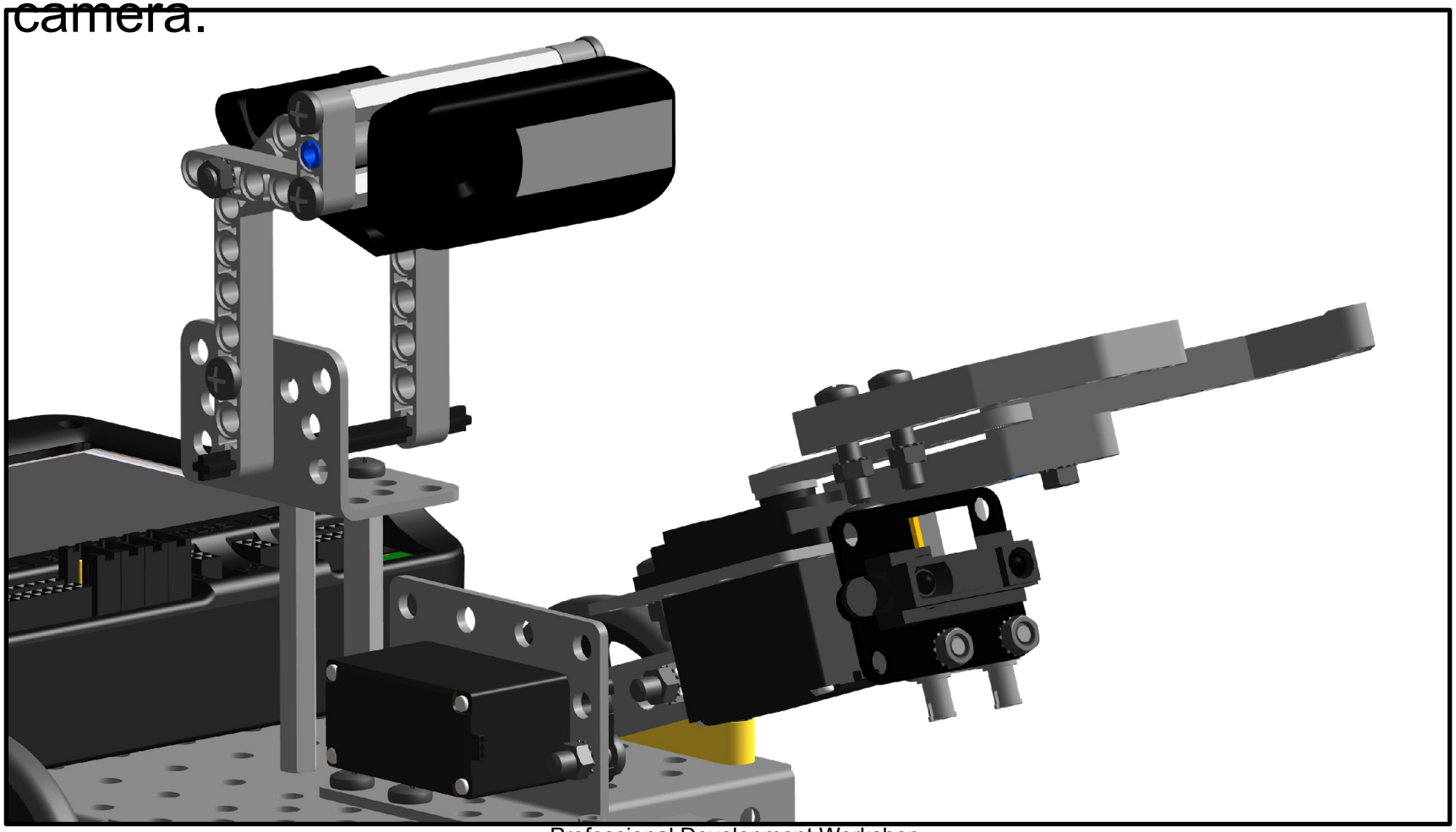


Setting the Color Tracking Channels

KISS INSTITUTE FOR PRACTICAL ROBOTICS

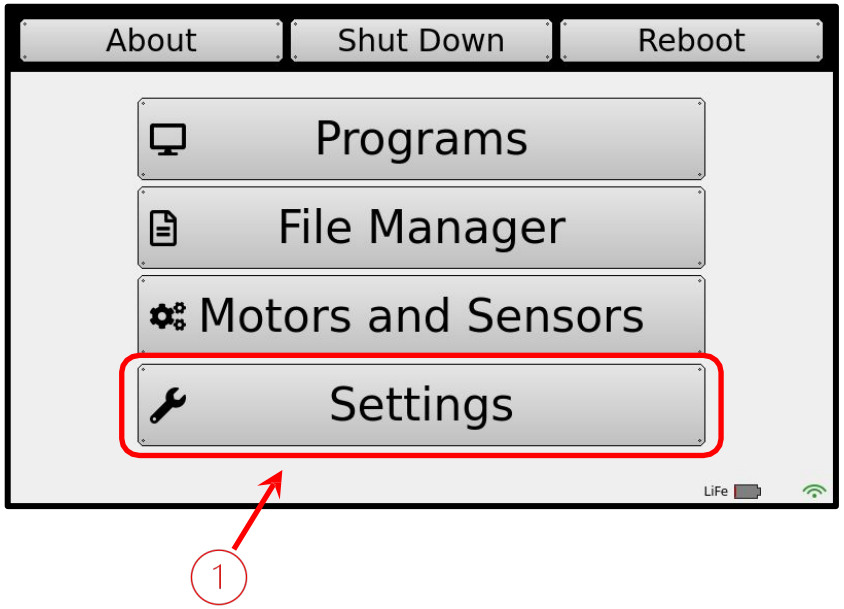
Botball®

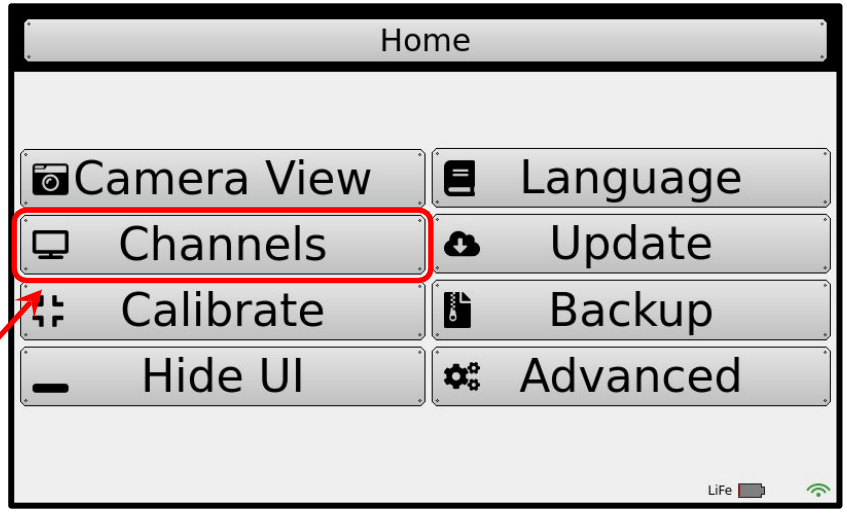
See the DemoBot build instructions for a way to mount the camera.



Setting the Color Tracking Channels

1. Select "*Settings*"
2. Select "*Channels*"



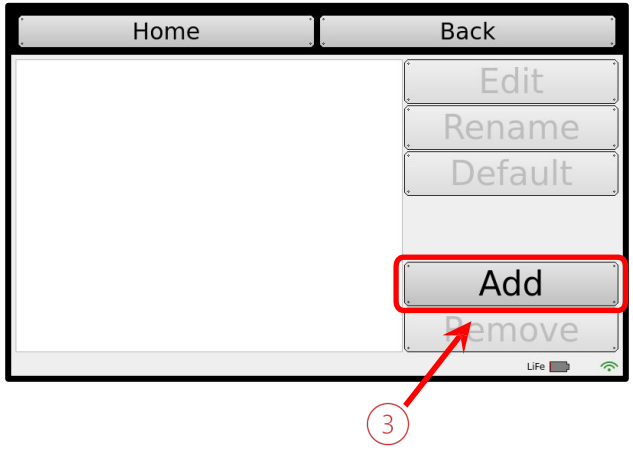


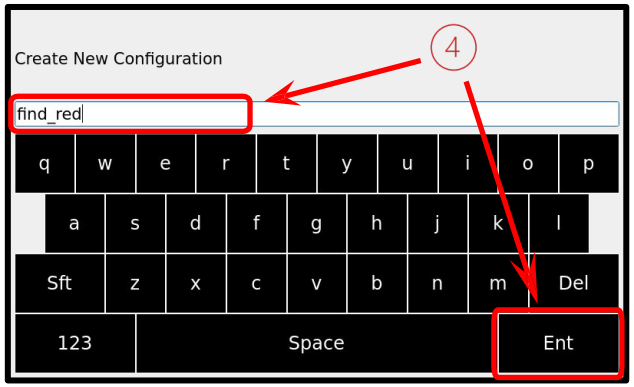
Setting the Color Tracking Channels

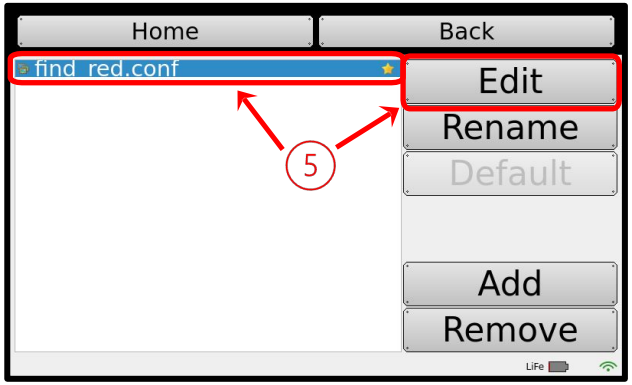
KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

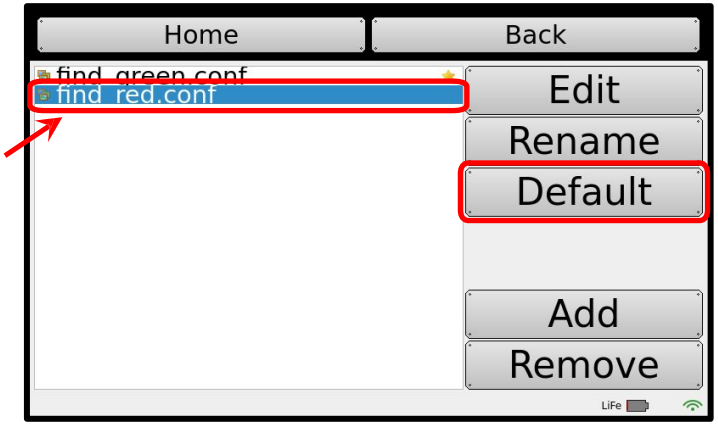
3. To specify a **camera configuration**, press the *Add* button.
4. Enter a configuration name, such as **find_red**, then press the *Ent* button.
5. Highlight the new configuration and press the *edit* button.





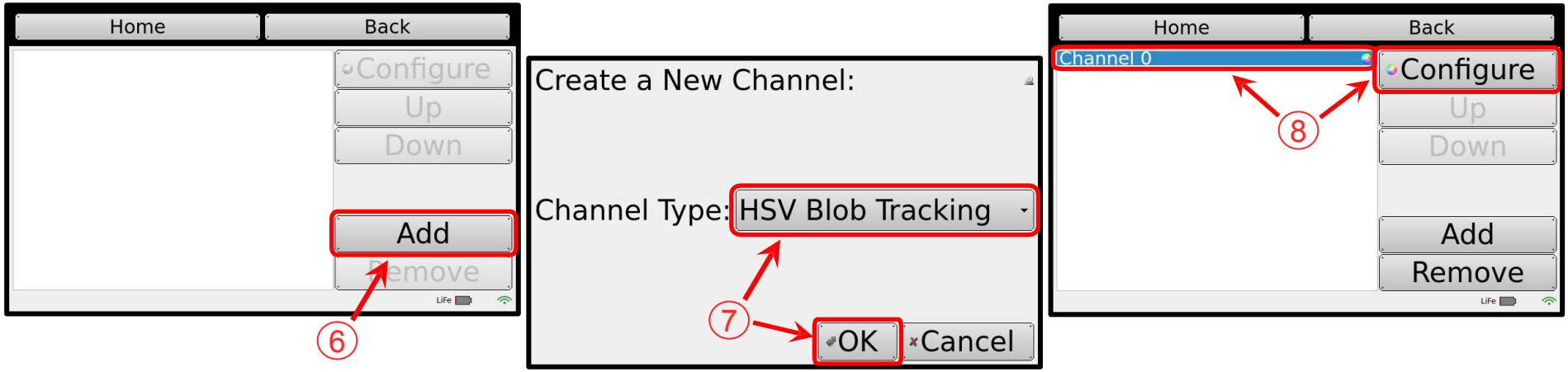


Note: if there is more than one configuration, select one, and press the "Default" button to make it be the one in use!



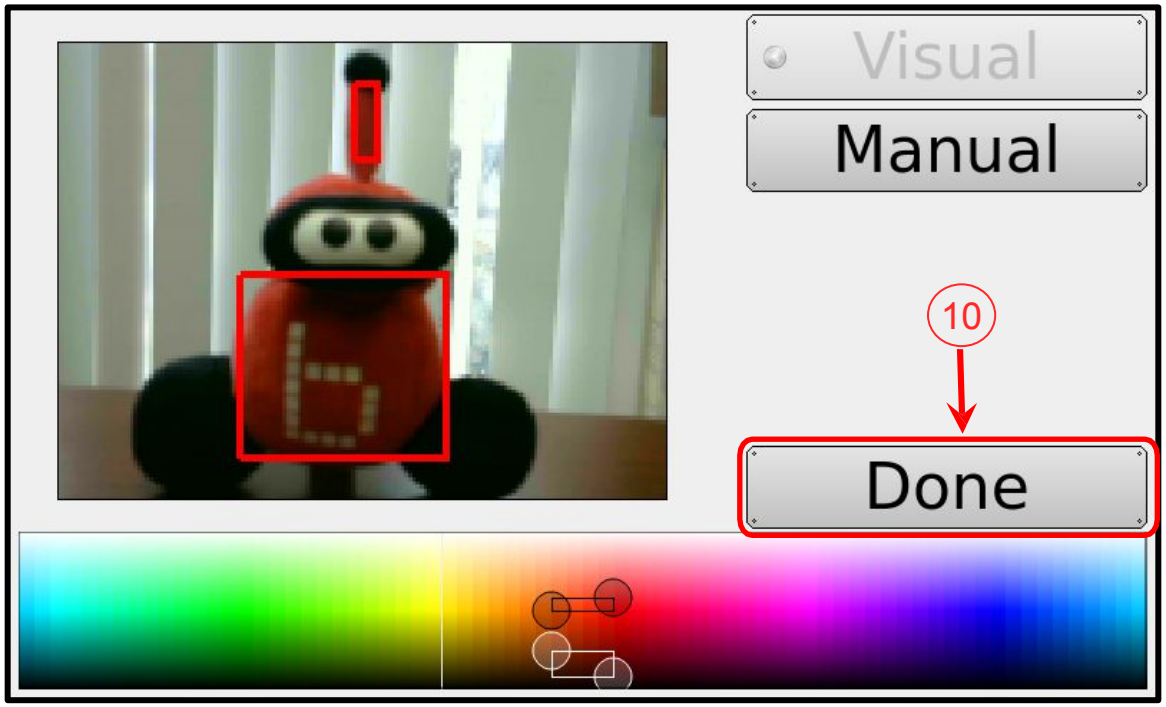
Setting the Color Tracking Channels

6. Press the *Add* button to add a channel to the configuration.
7. Select ***HSV Blob Tracking***, then *OK* to set this up to track a color.
8. Highlight the channel, then press *Configure* to edit settings.
   - The first channel is 0 by default. You can have up to four: **0**, **1**, **2**, and **3**.



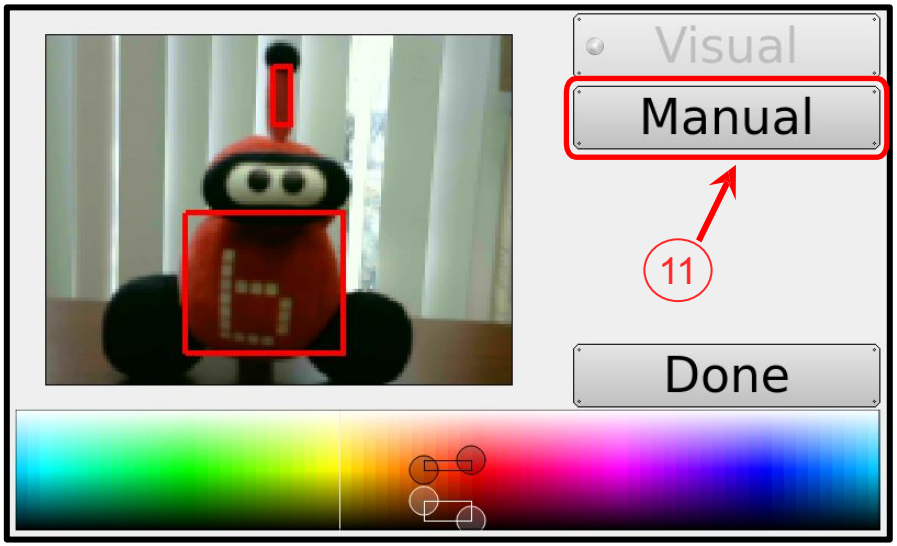
Setting the Color Tracking Channels

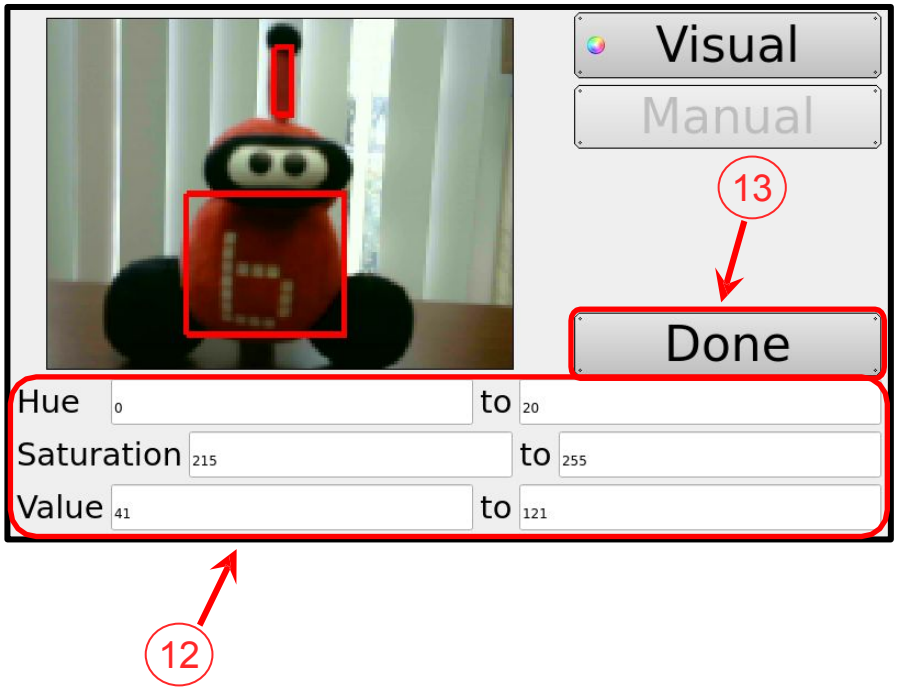
9. Place the colored object you want to track in front of the camera and **touch the object on the screen**.
   - A **bounding box** (**red**) will appear around the selected object.
10. Press the *Done* button.



Setting the Color Tracking Channels

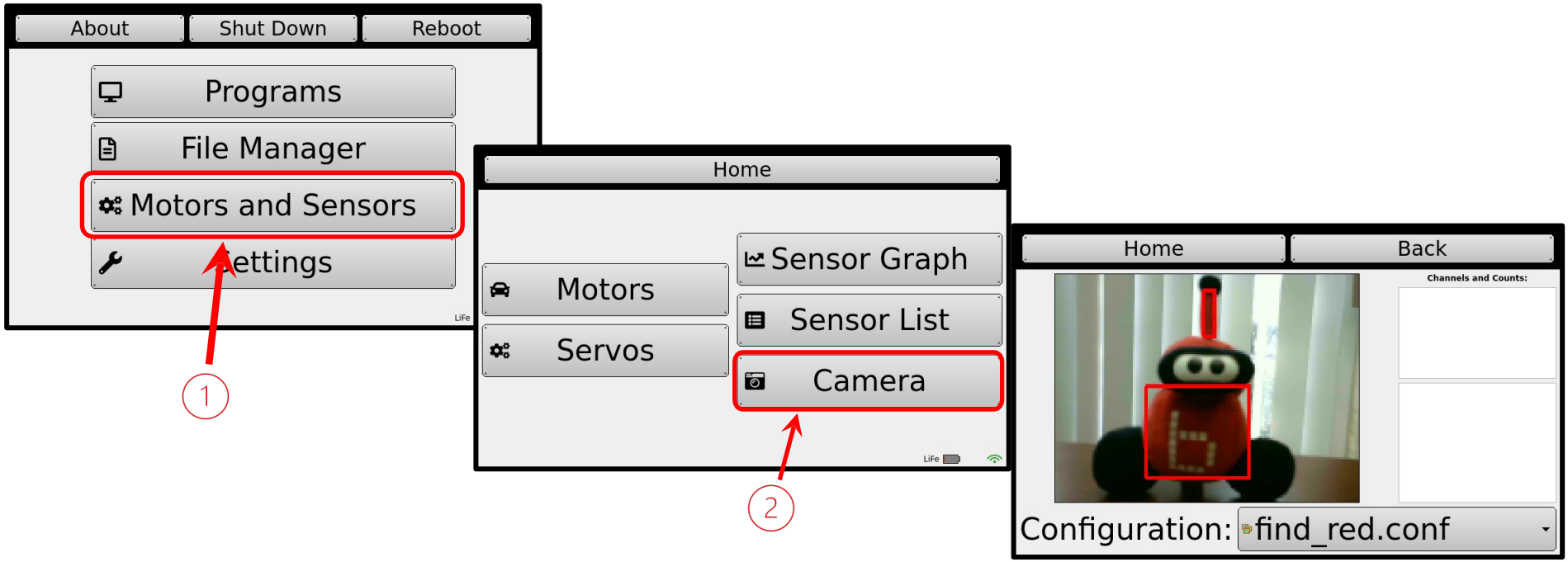
11. If you want to MANUALLY adjust the settings, select *Manual*
12. Adjust individual values
13. Press the *Done* button.





Verify Channel is Working

1. From the **Home** screen, press *Motors and Sensors* button.
2. Press the *Camera* button.
3. Make sure you select the configuration
4. Objects specified by the configuration should have a **bounding box**.



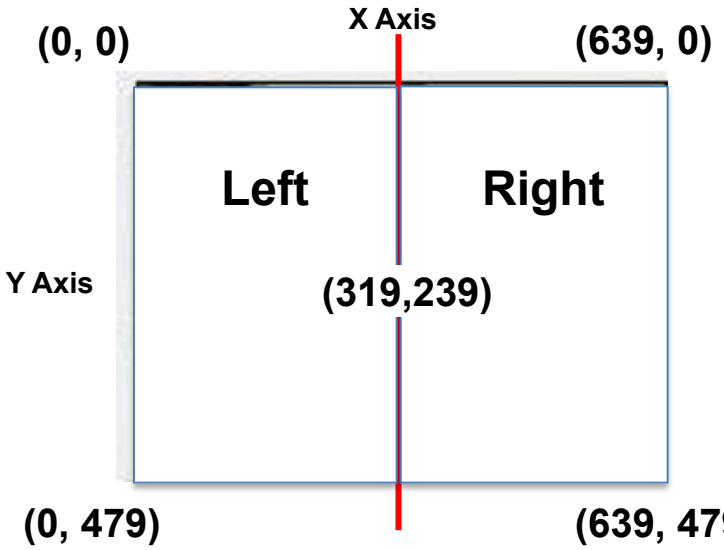
Tracking the Location of an Object

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

- You can use the **position** of the object in relation to the **center *x* (column)** of the image to tell if it is to the **left** or **right**.
  - The image is **160 columns wide**, so the **center column (*x*-value)** is 80.
  - An ***x*-value** of 319 is straight ahead.
  - An ***x*-value** between 0 and 318 is to the ***left***.
  - An ***x*-value** between 320 and 639 is to the ***right***.

- You can also use the **position** of the object in relation to the **center *y* (row)** of the image to tell **how far away** it is.



Object
0, 1, 2, …
(largest to smallest)

Channel #

```
get_object_center_x(0,0);
// The x-value of the tracked object.
// Note: number between 0 and 639.
```

Camera Functions

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

```
camera_open();
// Opens the connection to the camera.

camera_close();
// Closes the connection to the camera.

camera_update();
// Gets a new picture (image) from the camera and performs color tracking.

get_object_count(channel #)
// The number of objects being tracked on the specified color channel.

get_object_center_x(channel #, object #)
// The center x (column) coordinate value of the object # on the color
channel.

get_object_center_y(channel #, object #)
// The center y (row) coordinate value of the object # on the color channel.
```

Initial Camera Functions

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Resource

Programming statements always used with the camera:

```
camera_open();   // opens camera

camera_update(); // retrieves current image
```

If either of these two functions execute successfully they return 1, otherwise they return a value of 0

```
camera_close(); // closes camera
```

On older controllers, after opening the camera you should wait (msleep) three seconds before doing anything else; this gives the camera time to boot.

Initial Camera Functions

A commonly used camera function, almost always after **`camera_update()`** but often forgotten about. This function returns the number of objects “seen/found” in the **last** camera update (which could have been a while ago)

Source Code

```
1
2 if(get_object_count(0) > 0)
3 {
4     // code if object seen on channel (color) 0
5 }
```

Channel #: 0,1,2,3

- We chose 0 as our default
- This could be red or blue or green, etc. If you use a variable you could have and integer named **red_channel** and that would be easier to understand here

Number of objects should be greater than zero otherwise nothing was seen for the color represented by this channel

Camera Activity 1

Goal: Write a program that will allow you to check to see if the camera is tracking the color that you want it to see.

1. Setup one of the channels for **green** objects

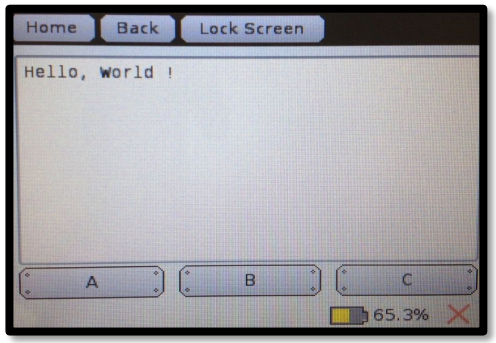
2. Write a program to look for **green** objects until the A button is pressed
   a) The program should print the words “I see green” when green objects come into view
   b) The program should print “Where is the green?” when it doesn’t seen green.

I See Green Continued

Example of code planning sheet:

1. Open the camera (starts communication between Controller & Camera)
2. Checks the status of the a_button
   a) We will use this step to create the **loop** that will keep your camera checking for images
3. Update the camera image (takes a snapshot of the current camera view)
4. Get an object count (the number of objects in the image)
5. Print “I see green.” (if green object seen, otherwise “Where is the green?”)
6. Remember if you want to stop the program you must press the A button: because you had a while loop that exits when a_button is pressed

Buttons



I See Green Example

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Source Code

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

int main()
{
	camera_open(); // opens and establishes communication with the camera
	while (a_button() == 0) // loops until the a button is pressed
	{
		camera_update(); // retrieves current camera image
		if (get_object_count(0) > 0) // does the camera see at least 1 green object?
		{
			printf("I see green.\n");
		}
		else
		{
			printf("Where is the green?\n");
		}
	}

	camera_close(); // disconnects from the camera
	return 0;
}
```

```
(get_object_count(0) > 0)
```

channel # (0 was the one we set for green)

number of objects

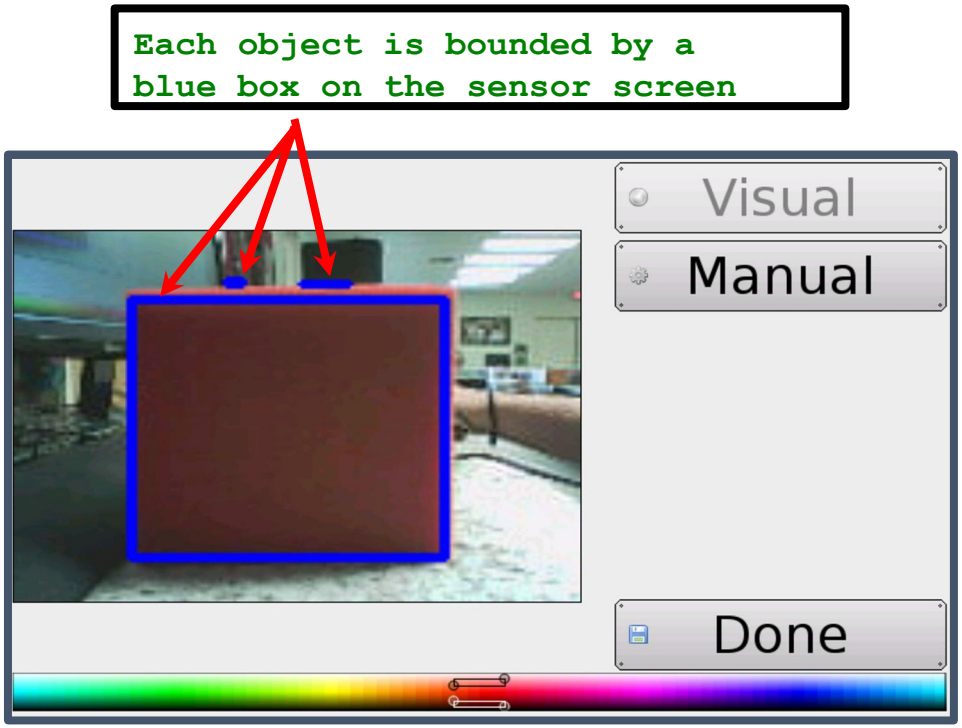
Getting the Object Count

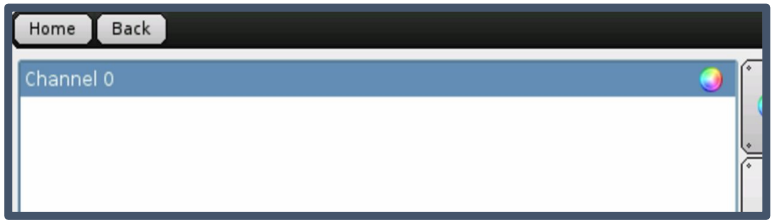
KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Resource

- Each object is numbered with the one with the largest area being object 0, the next largest being 1, and so on.
- The function below can be used to get the number of objects visible. This should only be done after a `camera_update()`





`get_object_count(channel#)`

Channel #: 0,1,2, or 3

- We setup 0 for green

Camera Activity 2

Goal: Print the number of objects the camera can see.

Activity:

1. Make sure you have configured your camera for this activity. Open a new project in your folder and write a program that does the following:
   a. Opens the camera
   b. Update the camera image
   c. Print the number of objects on the screen
   d. Close camera at the end
2. Proceed to the next slide for a sample solution.

Variations -

Run your program multiple times (or add a loop!) with different amount of objects (in the desired color, and other colors) in front of the camera and watch the number change (or not change).

Printing the Object Count

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Camera Activity 2: One possible solution

Source Code

```
1  #include <kipr/wombat.h>
2
3  int main()
4  {
5      int count; // Create a variable to represent the # of objects
6      camera_open(); // Opens camera
7
8      camera_update(); // Updates camera until it succeeds
9
10     count = get_object_count(0); // Capture number of objects seen
11     printf("There are %d objects on the screen.\n", count);
12     camera_close(); // Camera closed
13
14     return 0;
15 }
```

%d is a placeholder for an integer value

count is the integer value being placed into %d (note the use of a comma after the closed quote)

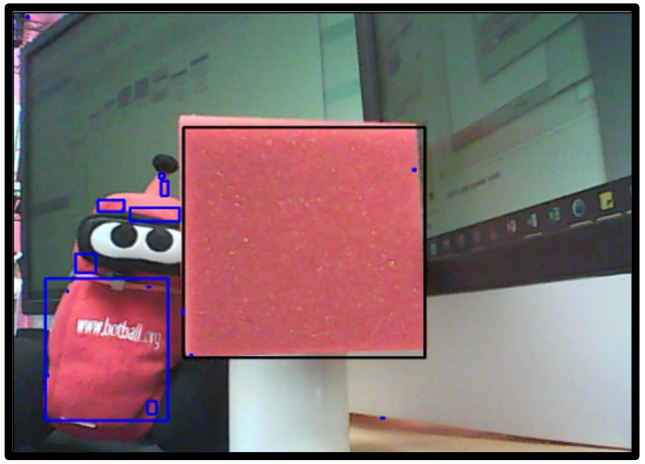
Output Example

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

1 Object

15 Objects



Do you see 15 objects in the second image?

Each is highlighted by a blue bounding box. Some are very, very small. The computer counts each group, no matter how small, as a separate object. What your eye sees as blue may or may not be the same as what the camera sees as blue. As an example, a bright white reflected spot off of a table may look white to you but the camera sees it as having a high concentration of blue light.

So, how do we figure out what objects are things we want the robot to interact with and which are just environmental noise?

There are other camera functions that we can use to get information about each object.

Screen Size

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Resource

The camera view is like a graph except the coordinate (0, 0) is in the ***top left corner***. The max width is **640** and the max height is **480**.

Object Centers

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Resource

Each object has a center. In this case the center would have the coordinates (x = 319, y = 239).

Getting the Object Center

Resource

These functions can be used to get the center x and center y values of an object:

```
get_object_center_x(channel#, object#) get_object_center_y(channel#, object#)
```

Note that the "first" **object#** (0) is the largest one of the color represented by **channel#**

Camera Activity 3

Goal: Find and print the center coordinates of an object with the camera

1. Make sure you have configured your camera for this activity. Open a new project in your folder and write a program that does the following:
   a. Opens the camera
   b. Update the camera image
   c. Check to see if there is at least one object on the screen
      i. get_object_center functions order the objects by size. The largest object has ID number 0.
   d. If there is at least one object, print the object center x and y coordinates
   e. Close camera connection

Variations -

Run your program multiple times with the object in different positions.

KISS INSTITUTE FOR PRACTICAL ROBOTICS Botball®

Activity 3 Template

Source Code

```
1  #include <kipr/wombat.h>
2
3  int main()
4  {
5
6      camera_open(); // Opens camera
7      camera_update(); // Updates camera until it succeeds
8
9
10
11     camera_close(); // Camera closed
12     return 0;
13 }
```

(A) Variables go here

(B) New camera code goes here

KISS INSTITUTE FOR PRACTICAL ROBOTICS Botball®

Activity 3: Possible Solution

Source Code

```
1
2  (A) Variables to be inserted in Camera Template (previous slide)
3      int x;
4      int y;
5
6  (B) New code to be inserted in Camera Template (previous slide)
7      if (get_object_count(0) > 0)
8      {
9          x = get_object_center_x(0,0);
10         y = get_object_center_y(0,0);
11         printf("The center of the object is (%d,%d).\n",x,y);
12     }
```

To print out the x and y values, you could have made two separate printf statements as done previously. The solution above demonstrates how to format and use multiple integer values in one printf. Note that the two %d are separated by a comma; as is the two value variables: x, y.

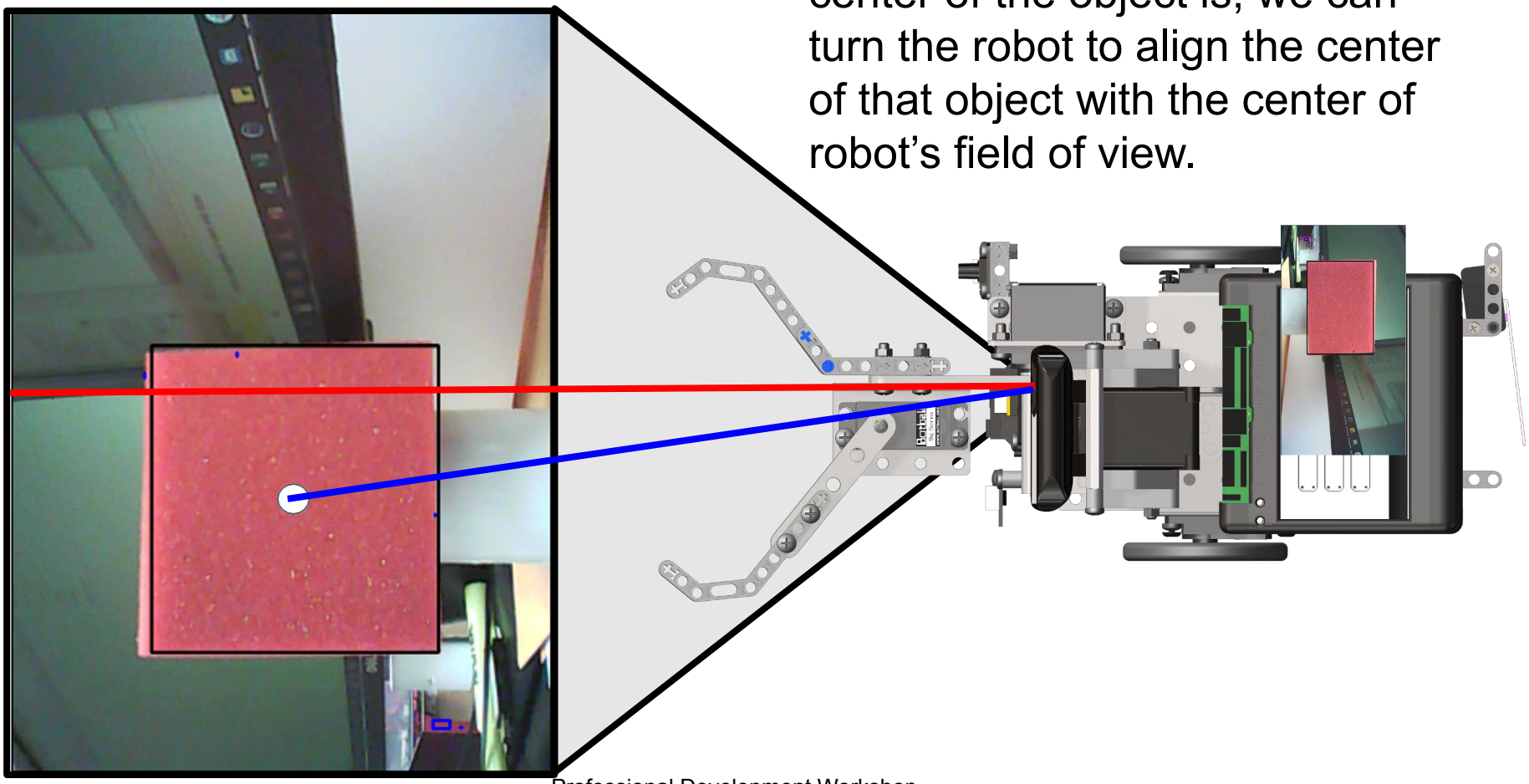
Turning Towards an Object

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Resource

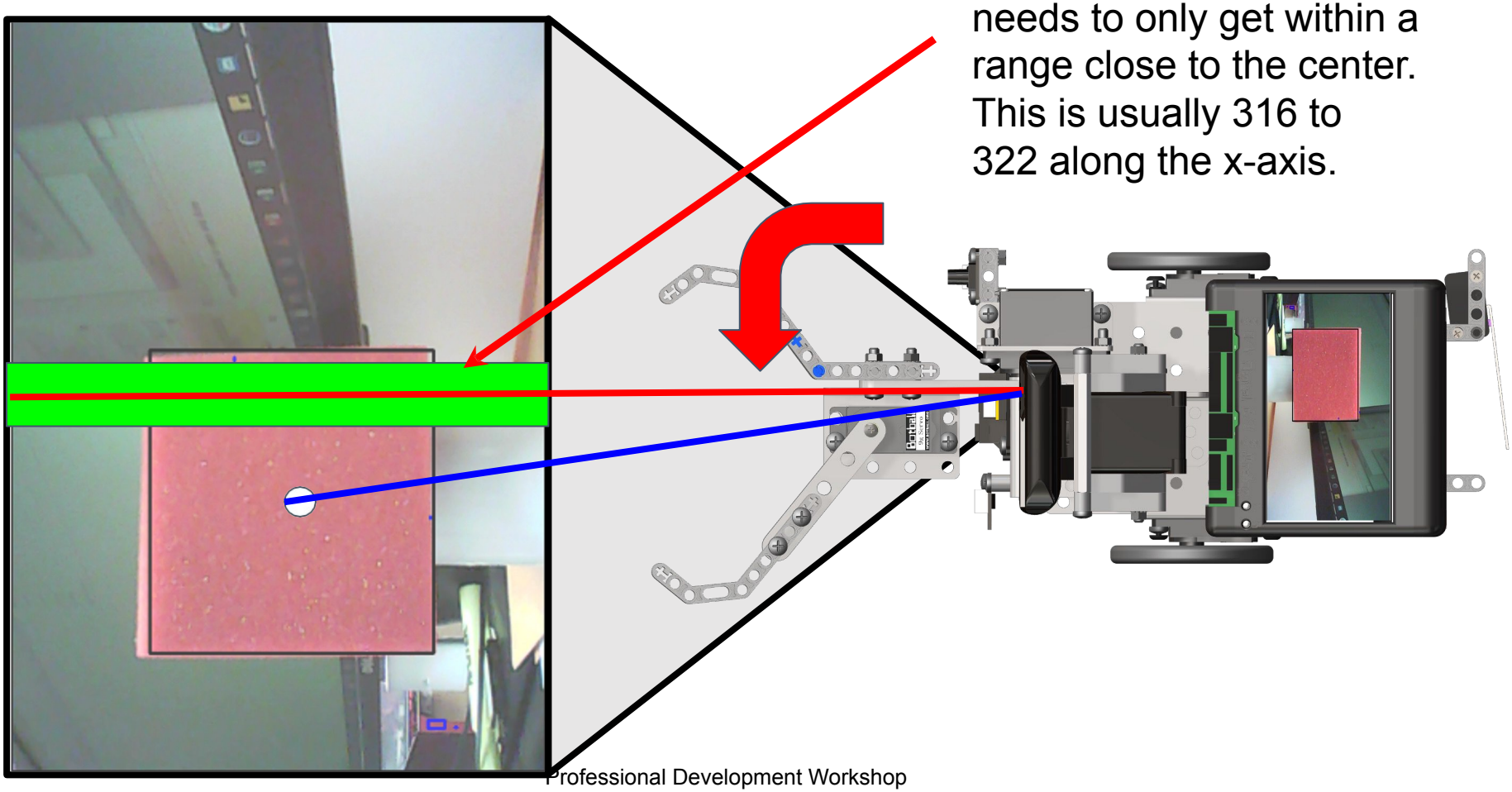
Now that we know where the center of the object is, we can turn the robot to align the center of that object with the center of robot's field of view.



Turning Towards an Object

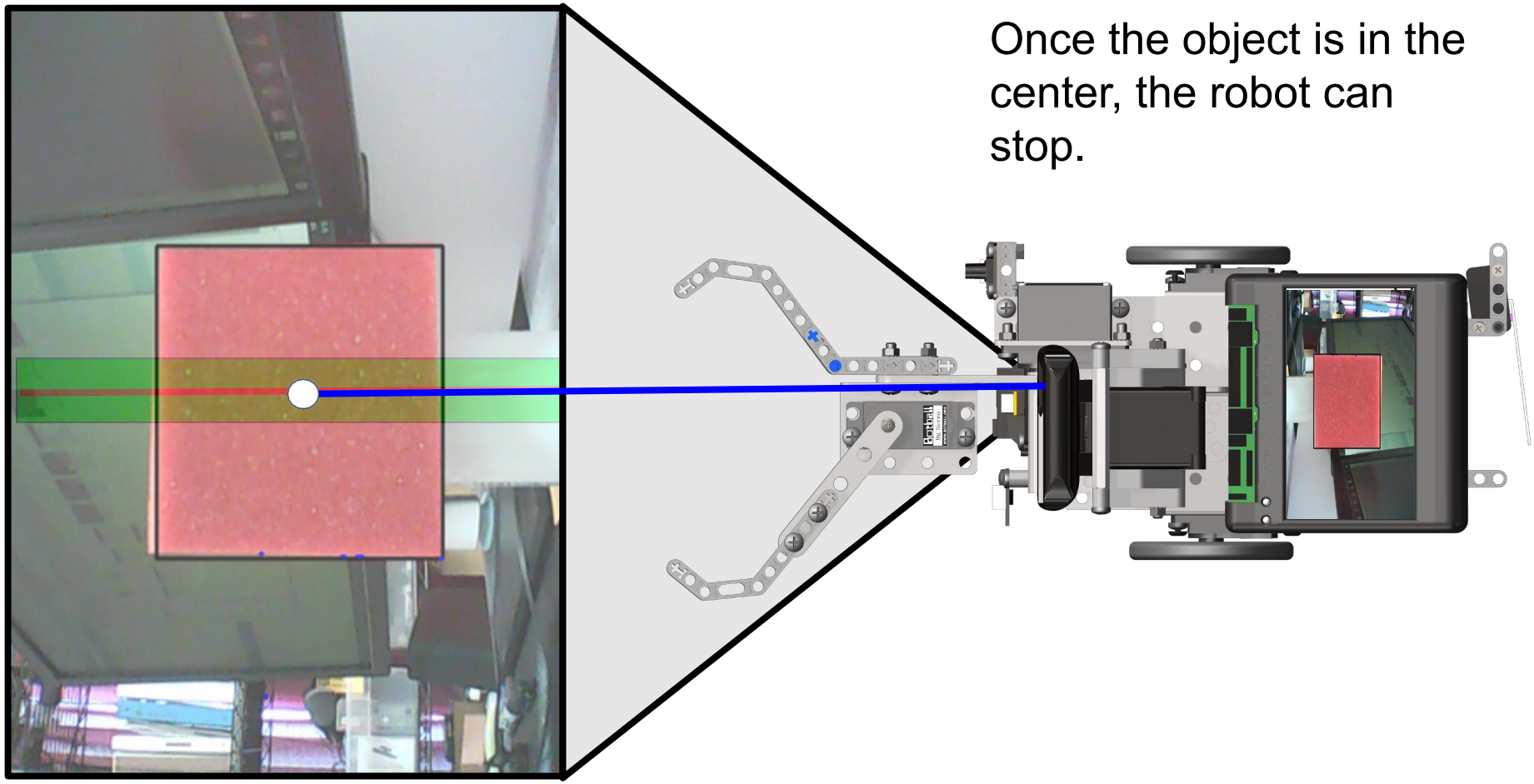
Resource

When the object is turning it needs to only get within a range close to the center. This is usually 316 to 322 along the x-axis.



Turning Towards an Object

Resource



Once the object is in the center, the robot can stop.

Camera Activity 4

Goal: Have a robot center itself on an object and print out the coordinates.
Activity:

1. Make sure you have configured your camera for this activity. Open a new project in your folder and write a program that does the following:
   a. Opens the camera
   b. If there is an object on the screen print the coordinates of the center
   c. Start turning until the object is in the center of the robot
   d. Print the new center coordinates of the object
   e. Close camera at the end
2. Proceed to the next slide for a sample solution.

Variations -
Have the object start off screen and have the robot turn until it sees it and it is centered.

Turning Towards an Object

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Activity 4 Template

Source Code

```
1  #include <kipr/wombat.h>
2
3  int main()
4  {
5      int stop = 0;
6
7      camera_open(); // Opens camera
8
9      while (stop == 0) // Updates camera image until stop pressed
10     {
11         camera_update();
12
13
14     }
15
16     camera_close(); // Camera closed
17     return 0;
18 }
```

(A) Variables go here

(B) Code to find object center goes here

(C) Code goes here to turn the robot

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Activity 4: Possible Solution

Source Code

```
1
2  (A) Variables to be inserted in Camera Template (previous slide)
3      int x;
4      int y;
5
6  (B) New code to be inserted in Camera Template (previous slide)
7      if (get_object_count(0) > 0)
8      {
9          x = get_object_center_x(0,0);
10         y = get_object_center_y(0,0);
11         printf("The center of the object is (%d,%d).\n",x,y);
12     }
```

Turning Towards an Object

Activity 4: Possible Solution

(C) New code to be inserted in Camera Template (previous slide)

```
if (get_object_count(0) > 0)
{
     if (get_object_center_x(0,0) < 316)
     {
          motor(0,-25);
          motor(3,25);
     }
     else if (get_object_center_x(0,0) > 322)
     {
          motor(0,25);
          motor(3,-25);
     }
     else
     {
          stop = 1;
          ao();
          if (get_object_count(0) > 0)
          {
               x = get_object_center_x(0,0);
               y = get_object_center_y(0,0);
               printf("The center of the object is (%d,%d).\n",x,y);
          }
     }
}
```

Output Examples

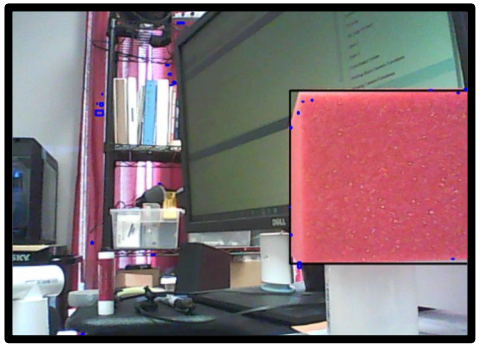
KISS INSTITUTE FOR PRACTICAL ROBOTICS

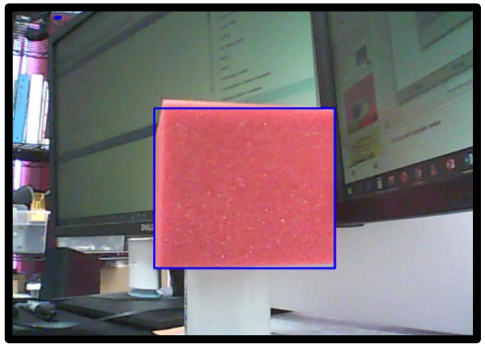
Botball®

Success

Runner

```
camera open complete
Linked to KIPR mods - index
(DEBUG) V4L: opening /dev/video0
The center of the object is (362,104).
The center of the object is (320,106).
The center of the object is (320,106).
closing camera
Cleaning up 2
~Wombat()
~Create()
Program exited with code 0
```





Fail

Runner

```
camera open complete
Linked to KIPR mods - index
(DEBUG) V4L: opening /dev/video0
[camera] [E]: No such object 0
[camera] [E]: No such object 0
```