KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Find the Wall & Line Following

Professional Development Workshop

Making a Choice

Program flow control with conditionals

`if-else` conditionals

`if-else` and Boolean operators Using `while` and `if-else`

Program Flow Control with Conditionals

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

- What if we want to execute a **block of code** *only if certain conditions are met*?

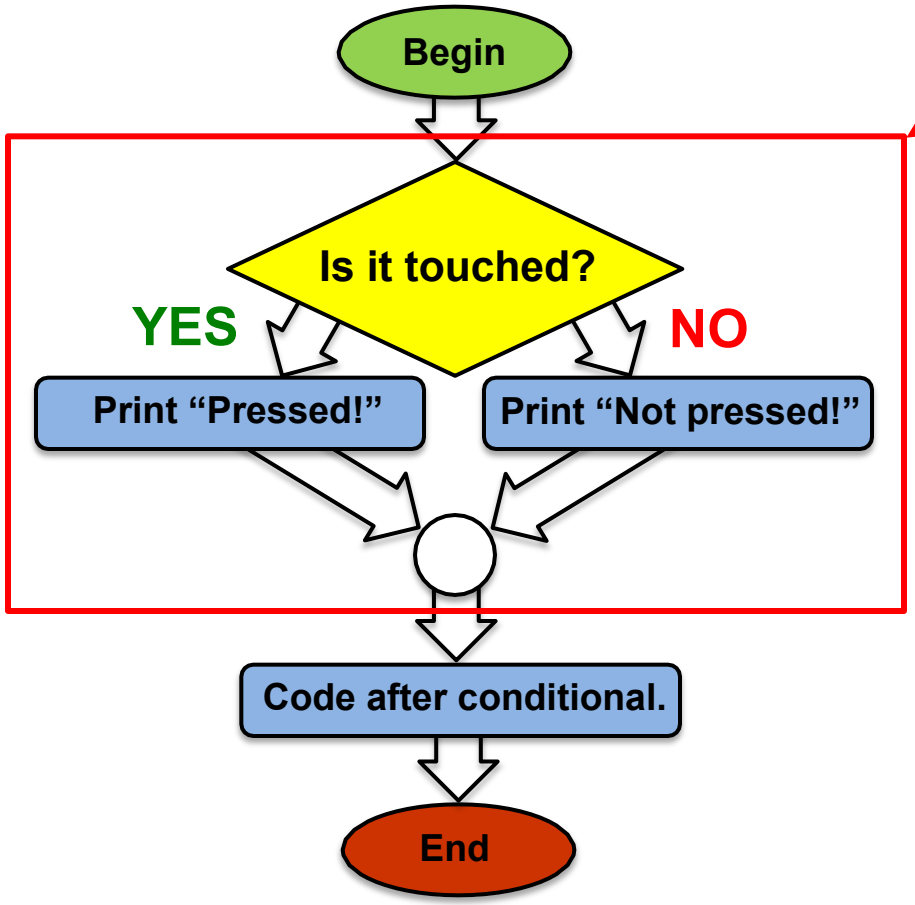
- We can do this using a **conditional**, which controls the **flow** of the program by executing ***a certain*** **block of code** if its conditions are met or a ***different*** **block of code** if its conditions are not met.
  - This is similar to a **loop**, but differs in that it **only executes once**.

Program Flow Control with Conditionals

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

This part of the code is the conditional.



if-else Conditionals

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

The `if-else` conditional checks to see if a **Boolean test** is **true** or **false**…

- If the **test** is **true**, then the `if` conditional **executes** the **block of code** that *immediately* follows it.
- If the **test** is **false**, then the `if` conditional **does not execute** the **block of code**, and the `else` **block of code** is **executed instead**.

Source

```
#include <kipr/wombat.h>
int main()
{
	if (digital(0) == 1)
	{
		printf("Touched!\n");
	}
	else
	{
		printf("Not touched!\n");
	}
	return 0;
}
```

What is this?

What does this say?

Notice: *In the same way that a* `while` *loop doesn't have a semicolon after the condition, neither does an if-else conditional.*

Using while and if-else

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

You can also put conditionals inside of (nested in) loops. This is beneficial when we want to keep checking a set of conditions over and over, instead of just a single time.

Source Code

```
1  #include <kipr/wombat.h>
2
3  int main()
4  {
5      while (digital(0) == 0)
6      {
7          if (analog(0) > 1600)
8          {
9              printf("It's dark in here!");
10         }
11         else
12         {
13             printf("I see the light!\n");
14         }
15     } // loop ends when the button is pressed
       return 0;
}
```

What should go inside the condition for the while loop?

Notice how the { and } braces line up for each block of code!

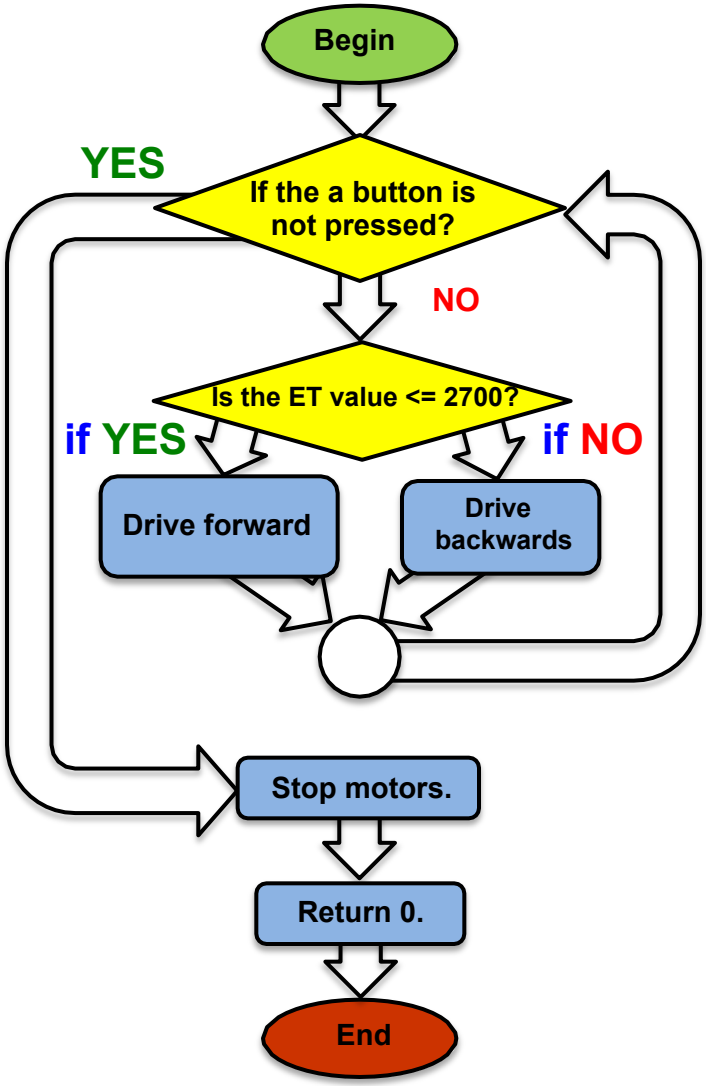
ET Drive forward to object

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Pseudocode (Task Analysis)

1. Check the a button, if it is not pressed
2. Drive forward as long as the value is <=2700 (or your determined value)
3. Drive backwards as long as the value is >2700 (or determined value)
4. Exit loop when a button is pressed
5. Shut everything off



ET Drive forward to object

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Source Code

```
1  #include <kipr/wombat.h>
2
3  int main()
4  {
5        printf("Drive to the object\n");
6        while (a_button() == 0) // A button is not pressed
7        {
8              if (analog(0) <= 2700) //Far away: drive forward
9              {
10                   motor(0, 80);
11                   motor(3, 90);
12             }
13             if (analog(0) > 2700) // Too close: back up
14             {
15                   motor(0, -80);
16                   motor(3, -90);
17             }
18       }
19       ao();
20       return 0;
21 }
22
```

Maintain Distance

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Description: Write a program for the KIPR Robotics Controller that makes the DemoBot maintain a specified distance away from an object, and stops when the touch sensor is touched.

Pseudocode

```
1.  Loop: Is not touched?

      If: Is distance too far?
        Drive forward.
      Else:
        If: Is distance too close?
          Drive reverse.
        Else:
          Stop motors.
2.  Stop motors.
3.  End the program.
```

Maintain Distance

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Solution:

Pseudocode

1. *Loop:* Is not touched?
   If: Is distance too far?
   Drive forward.
   Else:
   If: Is distance too close?
   Drive reverse.
   Else:
   Stop motors.
2. Stop motors.
3. End the program.

Source Code

```
1  #include <kipr/wombat.h>
2
3  int main()
4  {
5        while (digital(0) == 0)
6        {
7              if (analog(5) <= 1800)
8              {
9                    motor(0, 80);
10                   motor(3, 80);
11             }
12             else
13             {
14                   if (analog(5) > 2600)
15                   {
16                         motor(0, -75);
17                         motor(3, -75);
18                   }
19                   else // sensor value is 1800-2600
20                   {
21                         ao();
22                   }
23             }
24       }
25       ao(); // end of loop
26       return 0;
27 }
```


Botball®

Reflectance Sensor for Line Following

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

For this activity, you will need a **reflectance sensor**.

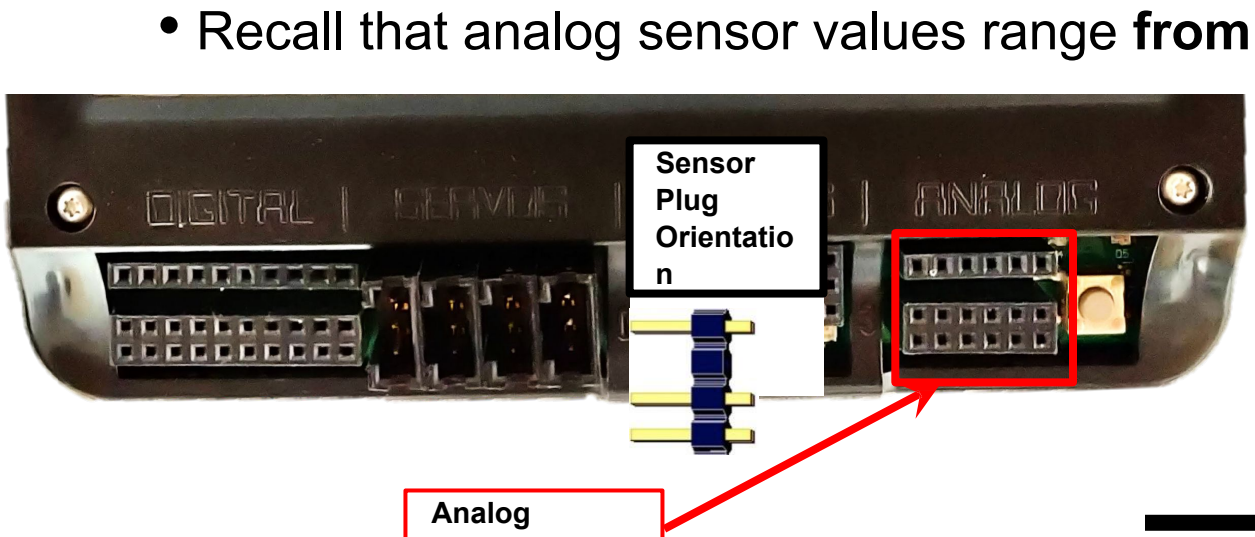
- This sensor is a short-range reflectance sensor.
- There is both an infrared (IR) *emitter* and an IR *detector* inside of this sensor.
- IR *emitter* sends out IR light → IR *detector* measures how much reflects back.
- The amount of IR reflected back depends on many factors, including **surface texture**, **color**, and **distance to surface**.

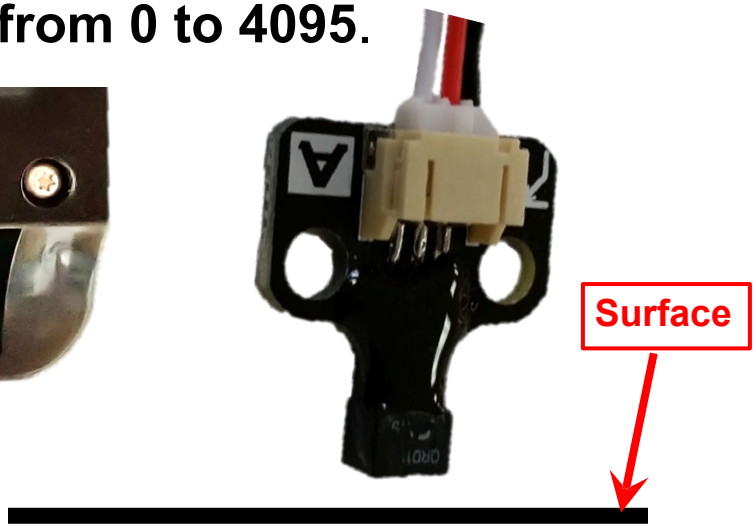
This sensor is **excellent** for line-following!

- **Black materials** typically **absorb <u>most</u> IR** → they **reflect <u>little</u> IR back**!
- **White materials** typically **absorb <u>little</u> IR** → they **reflect <u>most</u> IR back**!
- If this sensor is mounted at a *fixed height* above a surface, it is easy to distinguish a black line from a white surface.

Attach the Reflectance Sensor

- Attach the sensor on the front of your robot so that it is **pointing down at the ground** and is **approximately 1/8" from the surface**.
- A **reflectance sensor** is an **analog sensor**, so plug it into any of **analog sensor port #0 through 5**. Port 0 for this example.
  - Recall that analog sensor values range **from 0 to 4095**.

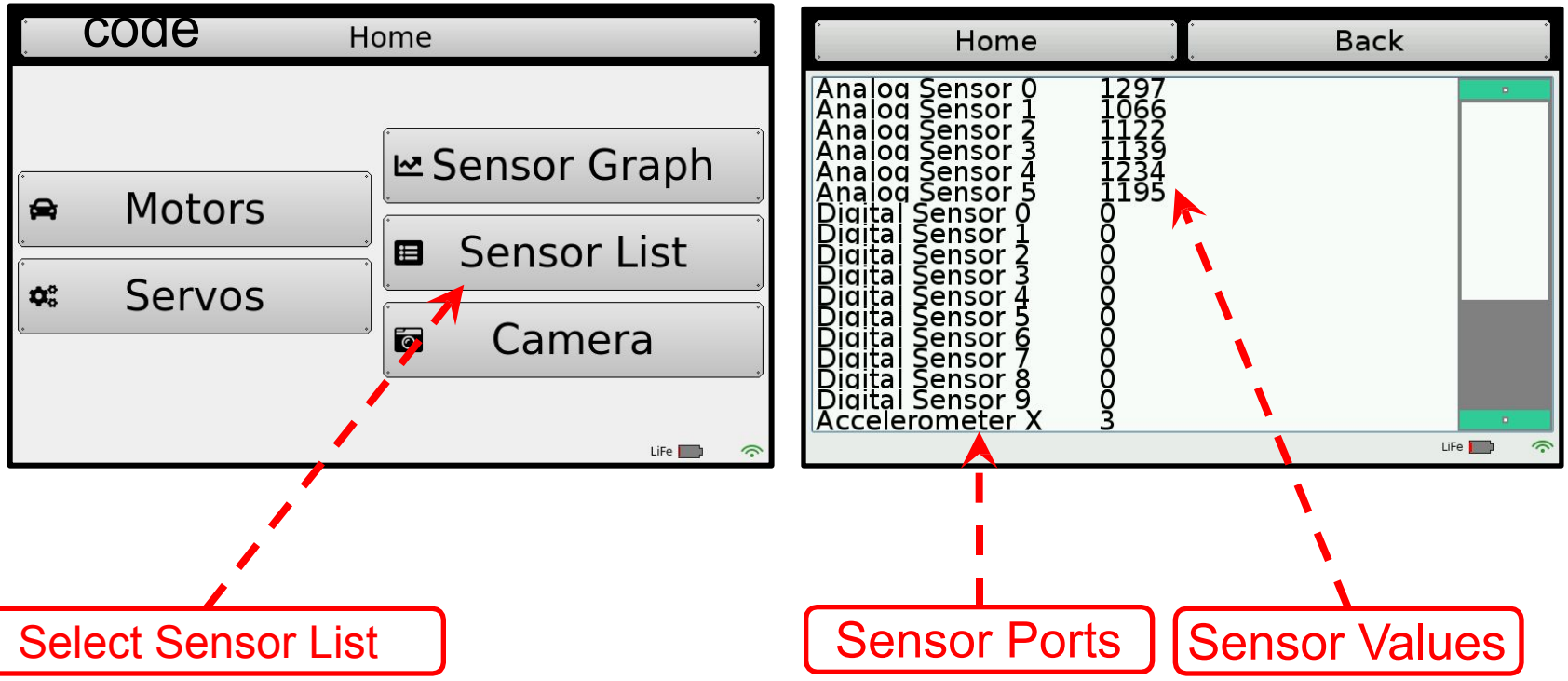




Reading Sensor Values from the Sensor List

You can access the Sensor Values from the Sensor List on your Wombat

- This is very helpful to get readings from all of the sensors you are using, and then know which values/ranges to use in your code

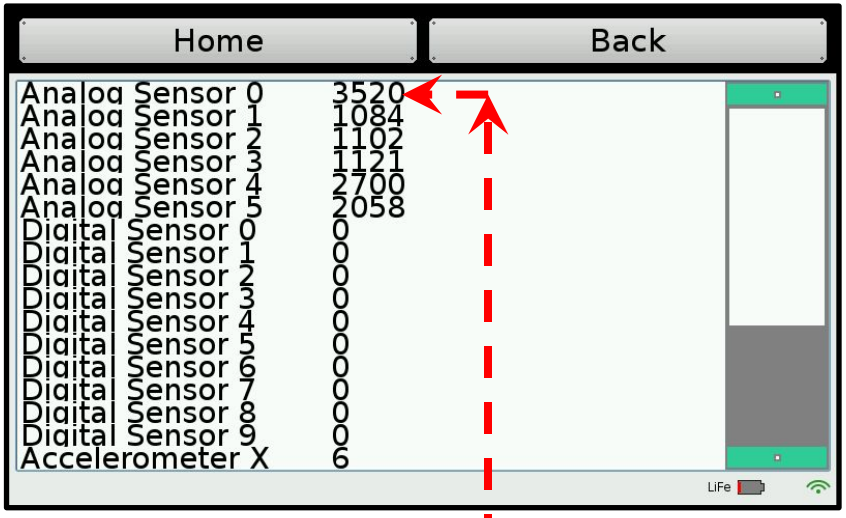


Reading Sensor Values from the Sensor List

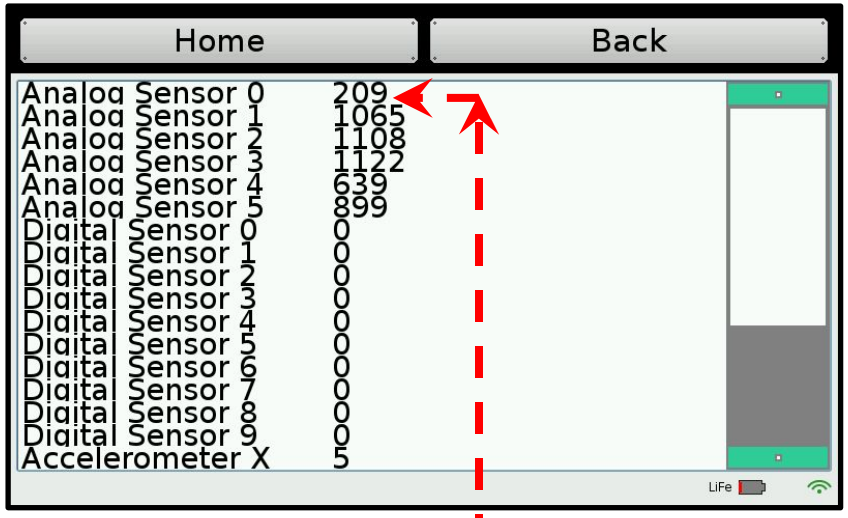
With the IR sensor plugged into analog port #0

- Over a white surface the value is (~200)
- Over a black surface the value is (~3000)

Your *values* will be different, but the *process* will be the same!



Values between ~2900-~3800 over the Black Surface



Values between ~175-~300 over the White Surface.

Line Following Strategy Using the Reflectance Sensor

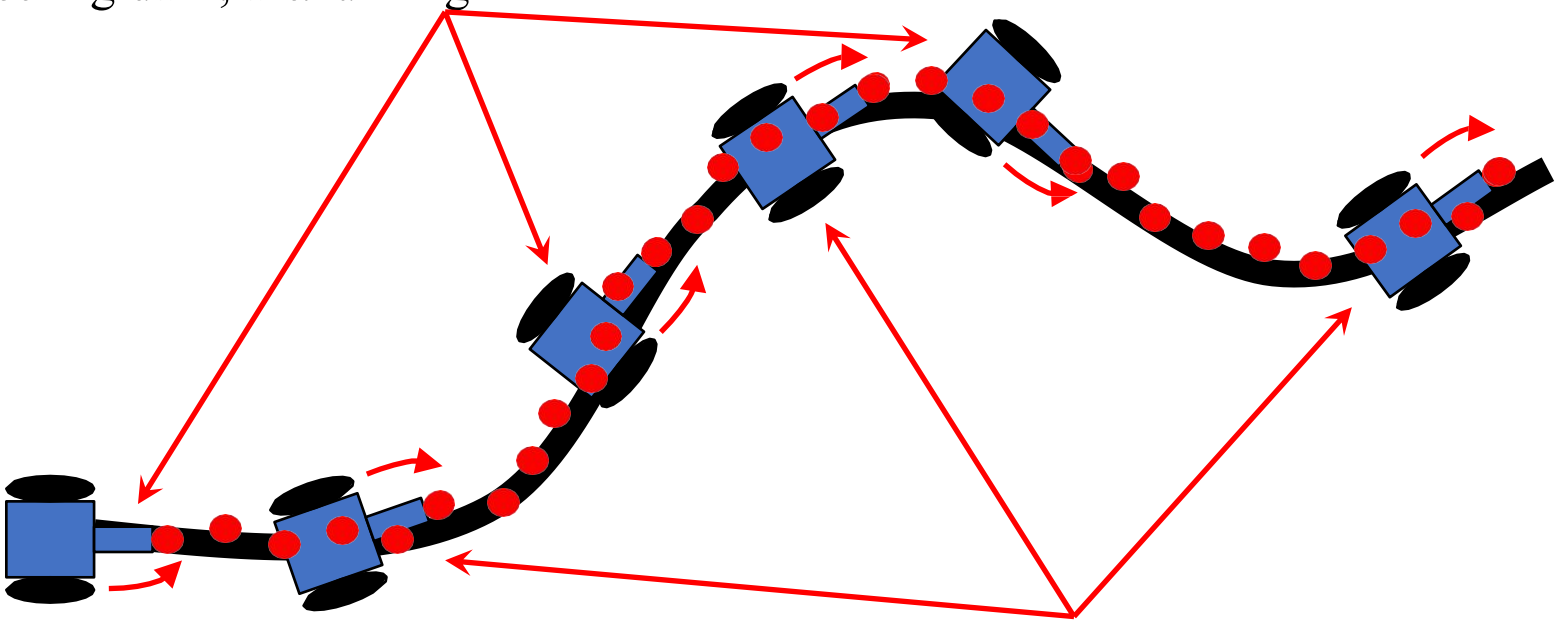
KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Line Following Strategy: `while` - Is the button pushed?

Follow the line's right edge by alternating the following 2 actions:

1. `if` detecting dark, arc/turn right



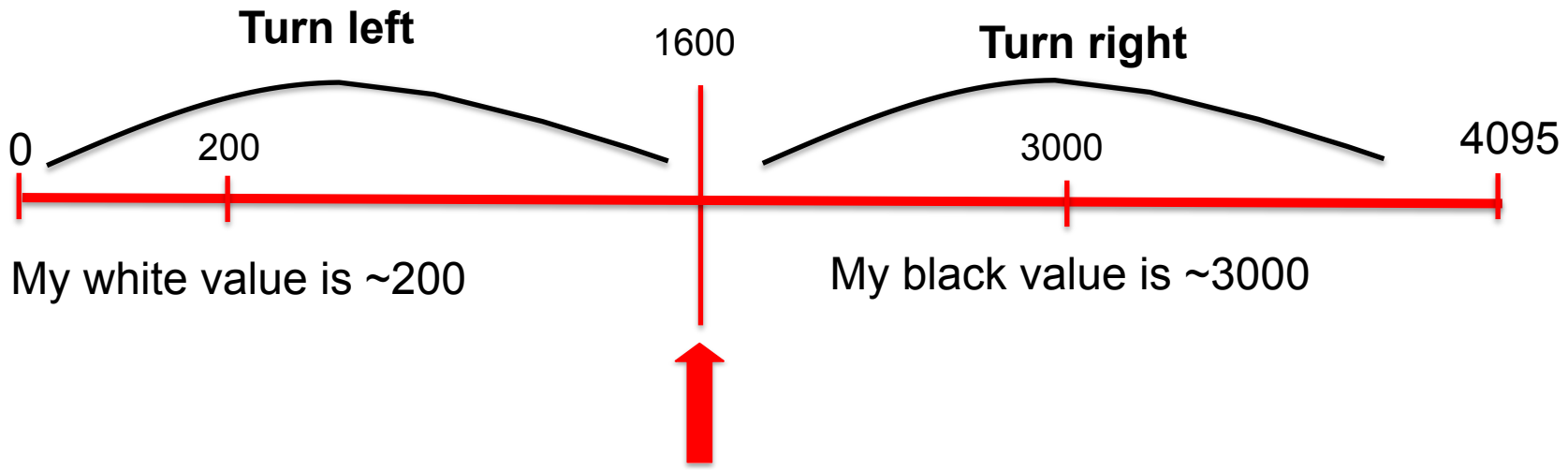
2. `if` detecting light, arc left.
3. Think about a sharp turn. What will your motor function look like? Remember the bigger the difference between the two motor powers the sharper the turn.

Understanding the IR Values

KISS INSTITUTE FOR PRACTICAL ROBOTICS

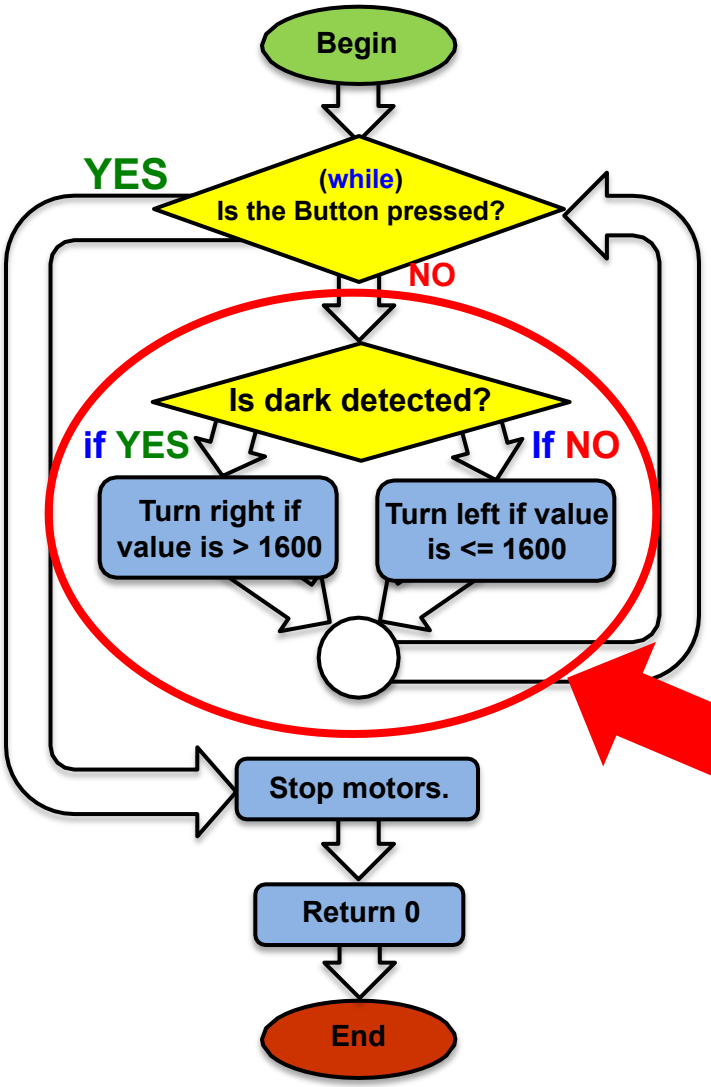
Botball®

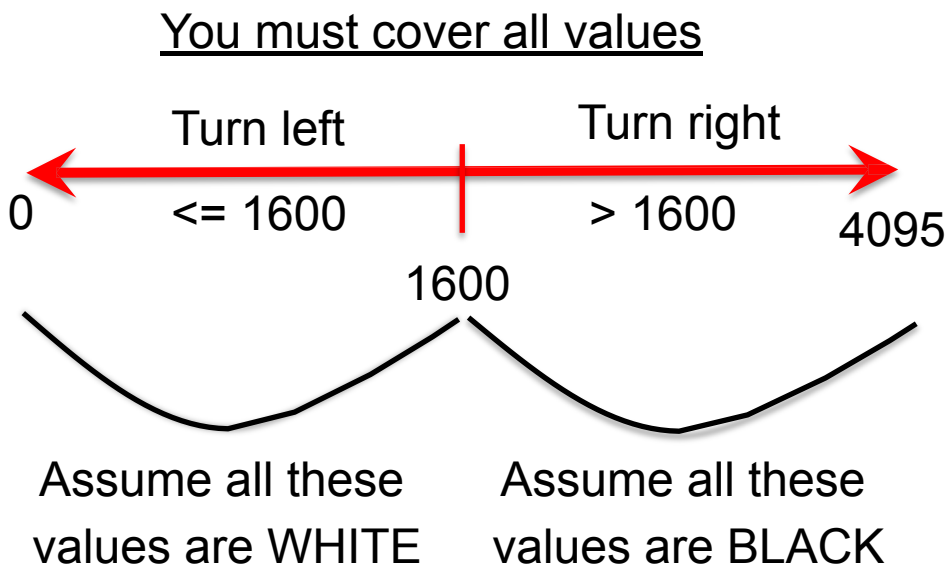
1. Place your IR analog sensor in one of the analog ports (0 to 5).
2. After mounting your IR sensor, check that the values are: white between
   175-225 and black between 2900-3100; write down your values.
3. Find your threshold or ***middle*** value (approximately)
4. This number will be the value you need for the find the black line activity.



Determine what your threshold or “half way” point will be. This example is ~1600.

Understanding while and if



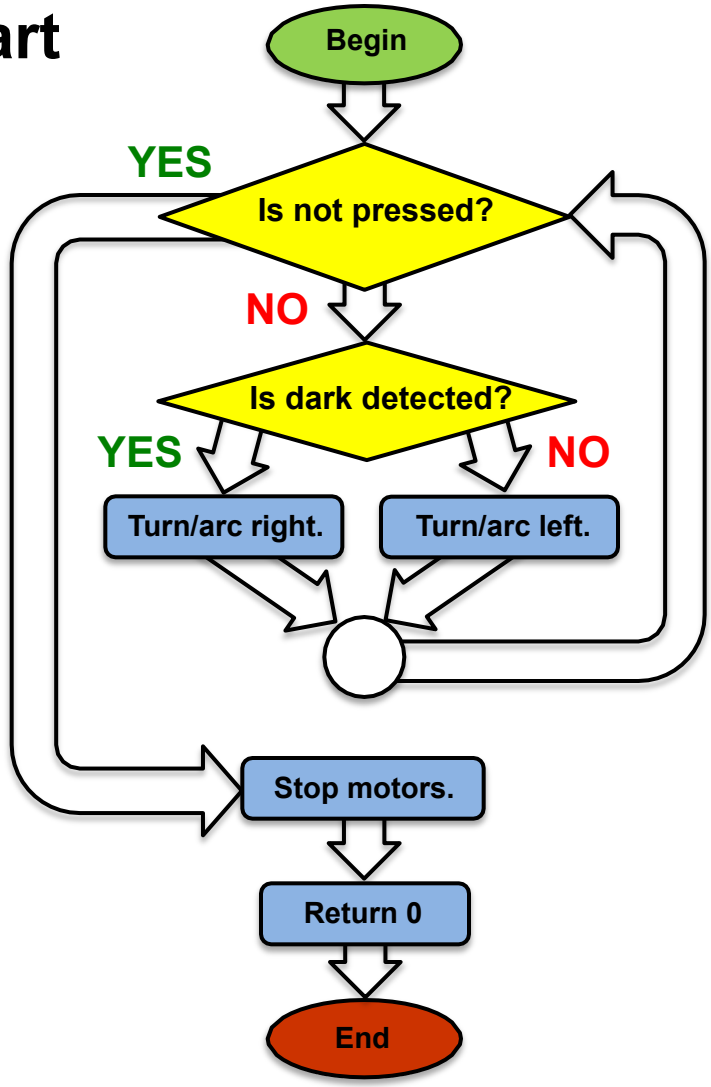


This is the part of the code that tells the Robot what to do when it sees black or white.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

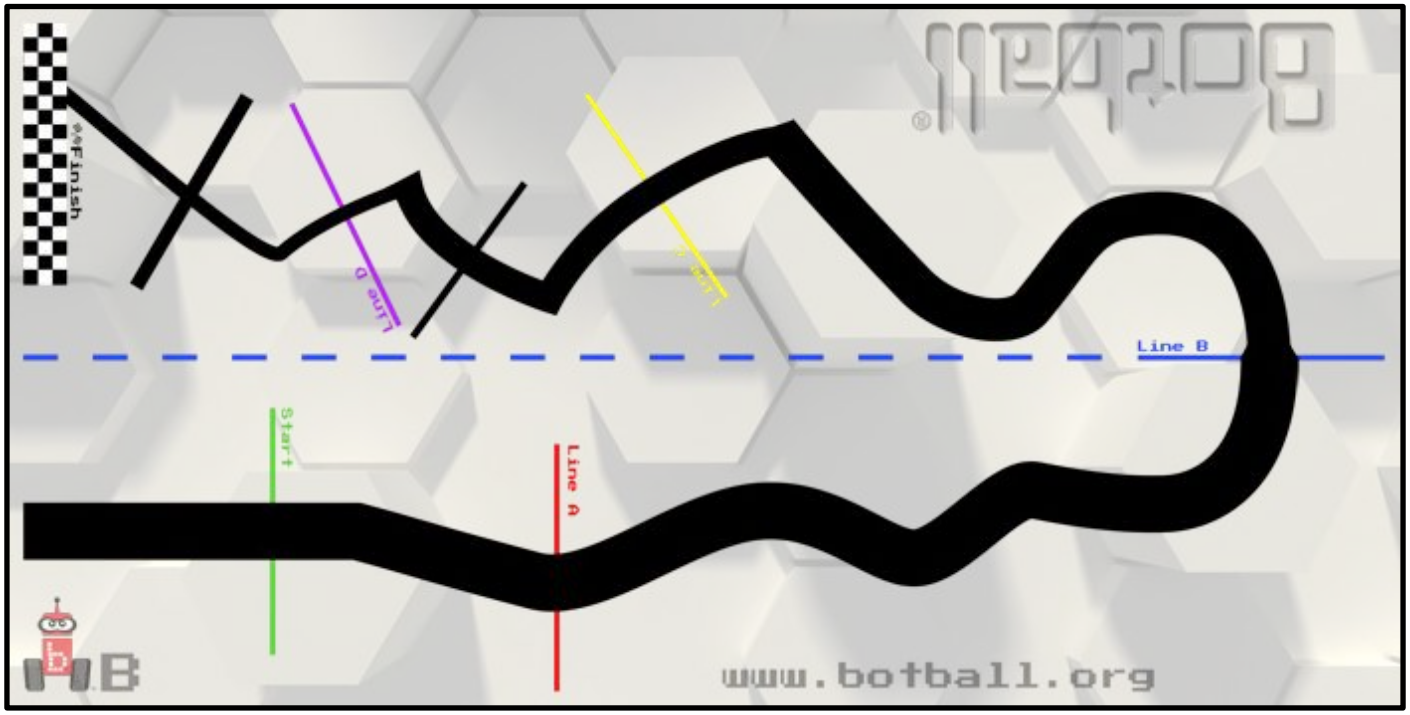
Botball®

Analysis: Flowchart



Line Following

Description: Starting with your DemoBot at the starting line of the KIPR Mat B. Write a program to have the robot travel along the path using the Top Hat sensor (line follow).



Line Following Solution

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Solution:

Pseudocode

1. *Loop:* Is not pressed?
   If: Is dark detected?
      Turn/arc right.
   Else:
      Turn/arc left.
2. Stop motors.
3. End the program.

Source Code

```
1  #include <kipr/wombat.h>
2
3  int main()
4  {
5      while (digital(0) == 0)
6      {
7          if (analog(0) <= 1600)
8          {
9              motor(0, 90);
10             motor(3, 5);
11         }
12         else
13         {
14             motor(0, 5);
15             motor(3, 90);
16
17         }
18     }
19     ao();
20     return 0;
21 }
22
```

Tips

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Change the threshold. Increase the “arc speed”.

Source Code

```
1  #include <kipr/wombat.h>
2
3  int main()
4  {
5      while (digital(0) == 0)
6      {
7          printf(“Follow the line\n”);
8          if (analog(0) > 1600)
9          {
10             motor(0, 90);
11             motor(3, 5);
12         }
13         else
14         {
15             motor(0, 5);
16             motor(3, 90);
17         }
18     }
19     ao();
20     return 0;
21 }
```

The value of 1600 or the “threshold” value is ½ way between the observed values.

Remember black reflects less IR than white but the value is higher.

Notice the Boolean operators > 1600 or <= 1600

Your value may be much lower due to lighting, placement of sensor and other factors.

Also increasing the “arc speed” (by making the ***difference*** between the two motor power values greater) may have a significant impact.

Line Following with Functions

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Botball®

Solution: (using two functions)

Pseudocode

```
1.   Loop: Is not pressed?
       If: Is dark detected?
          Turn/arc right.
       Else:
          Turn/arc left.
2.   Stop motors.
3.   End the program.
```

Source Code

```
1  #include <kipr/wombat.h>
2
3  void turn_right()
4  {
5        motor(0, 80);
6        motor(3, 10); // Turn/arc right
7  }
8  void turn_left()
9  {
10       motor(0, 10);
11       motor(3, 80); // Turn/arc left
12 }
13 int main()
14 {
15       while (digital(0) == 0)
16       {
17             if (analog(0) > 1600)
18             {
19                   turn_right();
20             }
21             else
22             {
23                   turn_left();
24             }
25       }
26       ao();
27       return 0;
28 }
```