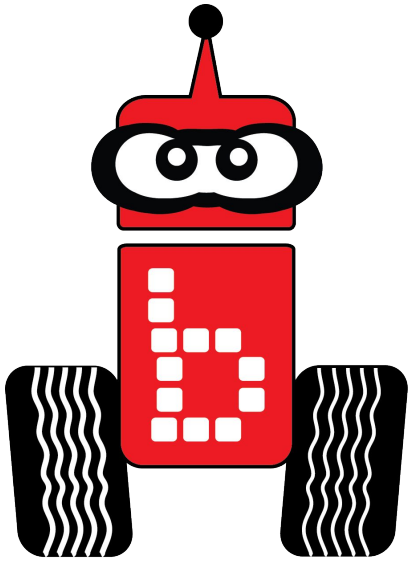


Using Sensors

- **Key Concepts:**
  - **Understand and apply the concepts of using digital and analog sensors**

- **Pacing:**
  - **Over several class periods**

Table of Contents

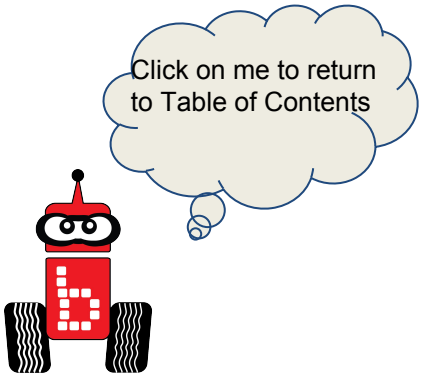


Table of Contents (cont.)

Table of Contents (Cont.)

Standards

Goal:

- Students will understand and apply the concepts of using digital and analog sensors
- Students will familiarize themselves with the functions `msleep()` and `motor()`
- Students will understand how to move their robots in the following manner: forwards, backwards, straight, circles, right and left turns

Standards:

Common Core State Standards Math Practices

CCSSMP1: Make sense of problems and persevere in solving them

CCSSMP2: Reason abstractly and quantitatively

CCSSMP4: Model with mathematics

CCSSMP6: Attend to precision

CCSSMP8: Look for and express regularity in repeated reasoning

Next Generation Science and Engineering Practice

1: Asking questions and defining problems
2: Developing and using models
3: Planning and carrying out investigations
4: Analyzing and interpreting data
5: Using mathematics and computational thinking
6: Constructing explanations and designing solution
7: Engaging in argument from evidence obtaining, evaluating, and communicating information

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Standards Continued

Standards Continued:

2016 ISTE Standards

Empowered Learner

1c: Students use technology to seek feedback that informs and improves their practice and to demonstrate their learning in a variety of ways.

1d: Students understand the fundamental concepts of technology operations, demonstrate the ability to choose, use and troubleshoot current technologies and are able to transfer their knowledge to explore emerging technologies.

Knowledge Constructor

3d: Students build knowledge by actively exploring real-world issues and problems, developing ideas and theories and pursuing answers and solutions.

Innovative Designer

4a: Students know and use a deliberate design process for generating ideas, testing theories, creating innovative artifacts or solving authentic problems.

4b: Students select and use digital tools to plan and manage a design process that considers design constraints and calculated risks.

4c: Students develop, test and refine prototypes as part of a cyclical design process.

4d: Students exhibit a tolerance for ambiguity, perseverance and the capacity to work with open-ended problems.

Computational Thinker

5a: Students formulate problem definitions suited for technology-assisted methods such as data analysis, abstract models and algorithmic thinking in exploring and finding solutions.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Using Sensors

Goals:

- To help students understand how to implement a touch sensor into the design of their claw.
- To compare and contrast using a touch sensor with a claw to grab an object versus using no sensor in a claw to grab an object.
- To understand the importance of using while loops and sensors to make a smarter robot.

Standards:

CCSSMP1: Make sense of problems and persevere in solving them, CCSSMP2: Reason abstractly and quantitatively, CCSSMP4: Model with mathematics, CCSSMP6: Attend to precision,CCSSMP8: Look for and express regularity in repeated reasoning.

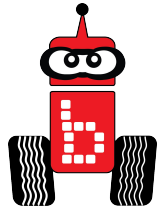
KISS INSTITUTE FOR PRACTICAL ROBOTICS

Learning About Sensors

1. Use the Jigsaw strategy to learn about sensors. Use the following slides;
    a. Decision Making and Sensors
    b. Smarter Robots
    c. Sensor Functions
    d. Checking Values
    e. What is a Digital Sensor?
    f. The Large Touch Sensor

KISS INSTITUTE FOR PRACTICAL ROBOTICS

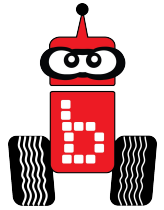
Decision Making and Sensors



- You should now realize how hard it is to be consistent with dead reckoning.

- Now we will add decision making and sensors to make our robots smarter.

Smarter Robots



When you log onto your computer you must enter a password. The program checks this against a stored value and if it matches, the program opens.

If the password doesn't match, the program runs a different set of code that prompts you to try again or even locks you out!

To make a smart robot, we need to check and compare sensor values

Sensor values are either:

- **Digital**- Returns a value of 0 or 1 (true or false), (touched not touched)
  - Sensors include; small touch, large touch and lever
- **Analog**- Return whole number values between 0-4095 (12 bit analog = 2^{12} or 4096- remember we start counting at 0)
  - Sensors include; Light, small top-hat and ET (rangefinder)

*You can find sensor information in the sensor and motor manual located in the help file on your Wallaby (Upper right hand corner of compiler screen)

Digital or Analog? Activity-1

1. Place your bag of sensors in front of you.
2. The goal is to sort your sensors into two sides; analog and digital. Sort them on the basis of; if it works like a switch or button it goes on one side otherwise put it on the other side
3. Note* If you have old “link” sensors with white wires, they do not work on wallabies.

Digital

Larch Touch Sensor

Large Lever Touch Sensor

Analog

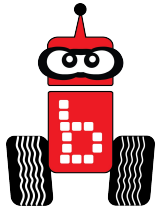
Light Sensor

Small IR Reflectance Sensor

“ET”-rangefinder

Printable Boolean Operators

Checking Values



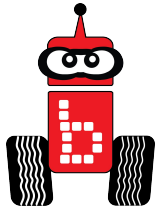
When writing code you use OPERATORS that allow the program to check a value stored against another value to determine if it is True or False.

Boolean operators

> Greater than	5 > 4 is TRUE
< Less than	4 < 5 is TRUE
>= Greater than or equal	4 >= 4 is TRUE
<= Less than or equal	3 <= 4 is TRUE
== Equal to	5 == 5 is TRUE
!= Not equal to	5 != 4 is TRUE

*Until you are familiar with the Operators that you will be using, you can use the “hint sheet” for easy reference.

Sensor Ports



You call for the analog sensor value with a function

- You have 6 analog ports (0-5)

```
analog(Port#);        analog(1);
```

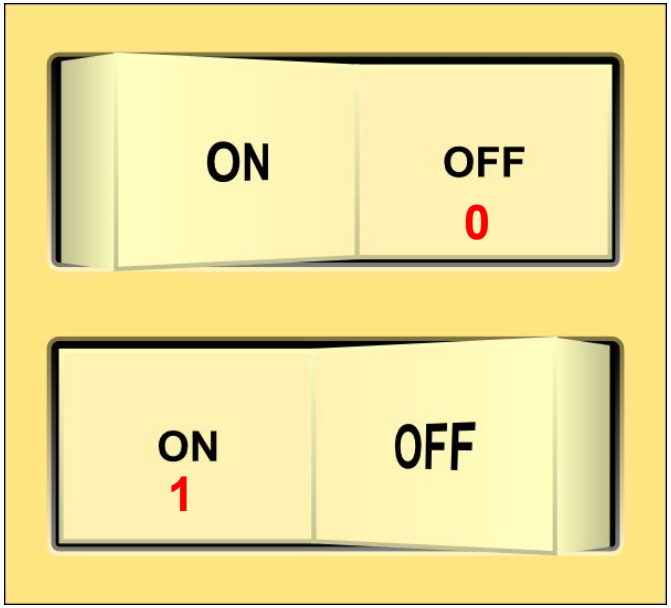
You call for the digital sensor value with a function

- You have 10 digital ports (0-9)

```
digital(Port#);       digital(8);
```

What is a Digital Sensor?

- You may want to think about these digital sensors being similar to a light switch; You only have two choices, off, not touched (0) and on, touched (1).



Touch Sensors

Description

- The touch sensors are mechanical switches. Pressing the switch brings two contacts together completing the circuit. Because of the way this sensor works, it is either touched and returns a value of (1) or not touched and returns a value of (0).

Uses

- These sensors are used to detect if the robot is in physical contact with something, like a wall.

Understanding The Touch Sensor
Activity 1

Description:

- The touch/bump sensor is a mechanical switch. Pressing the switch brings two contacts together completing the circuit. Because of the way this sensor works, it is either touched (1) or not touched (0).

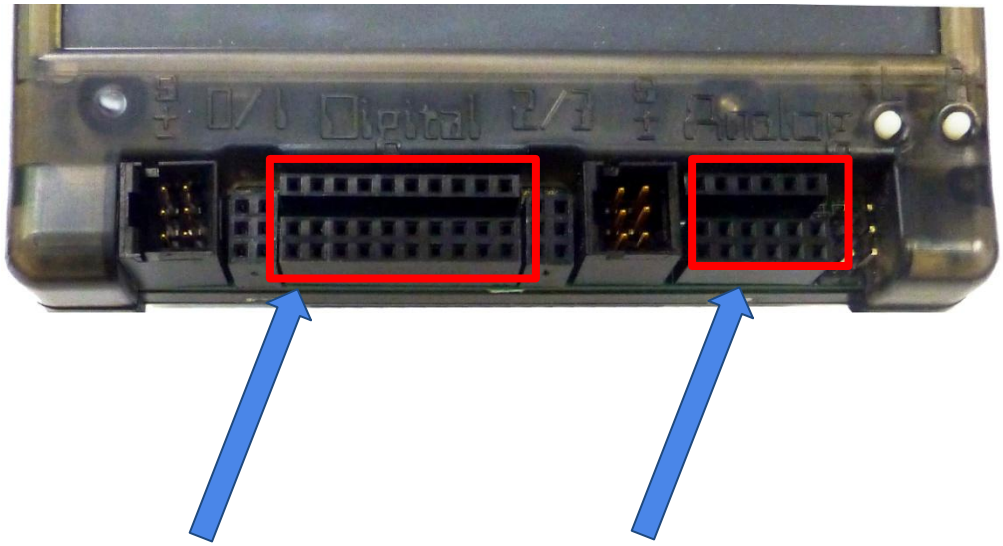
Uses:

- This sensor is used to detect if the robot is in physical contact with something, like a wall.

Activity:

1. Use the Thinking Notes Strategy to read,discuss and follow the directions on the following slides to use the touch sensor.
   - Sensor Ports
   - Plug in Your Touch Sensor
   - Reading Sensor Values
   - Check Touch Sensor
   - Interpreting Values
   - It's Dark
   - Looping
   - Understanding `while` loops

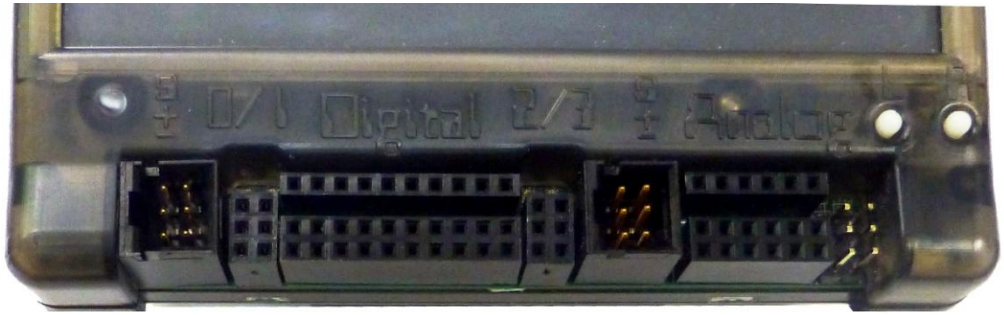
Sensor Ports

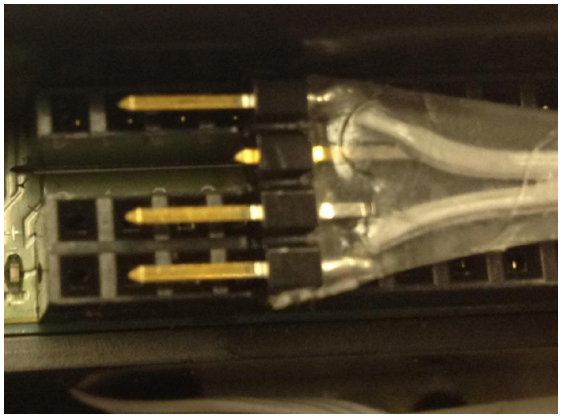


Digital ports #0-9

Analog ports #0-5

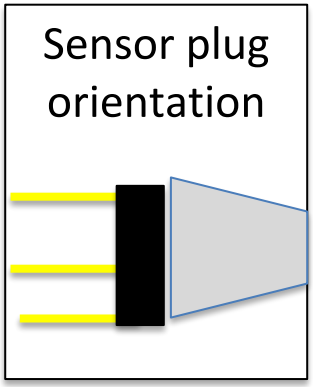
Plug in Your Touch Sensor





Plug your touch sensor into digital port o

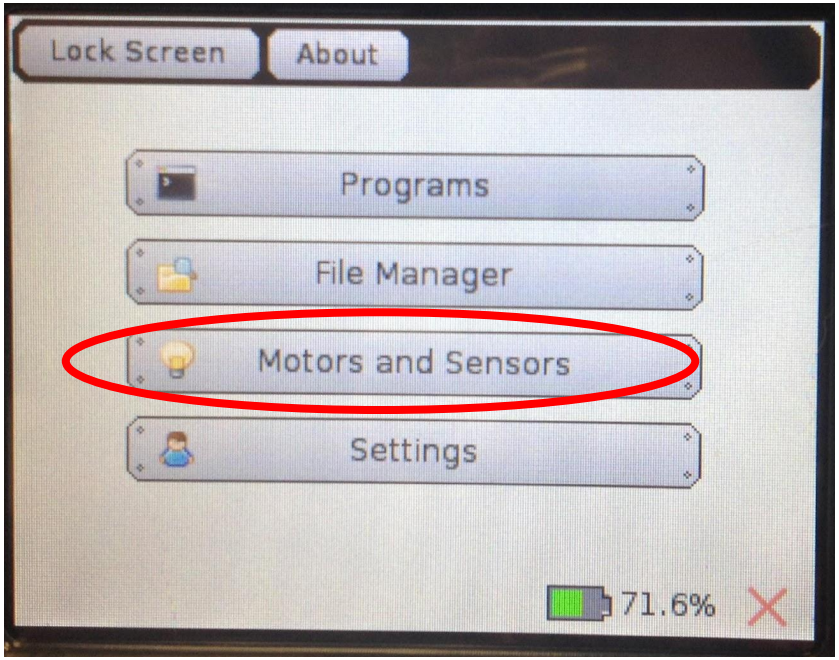
Sensor plug orientation

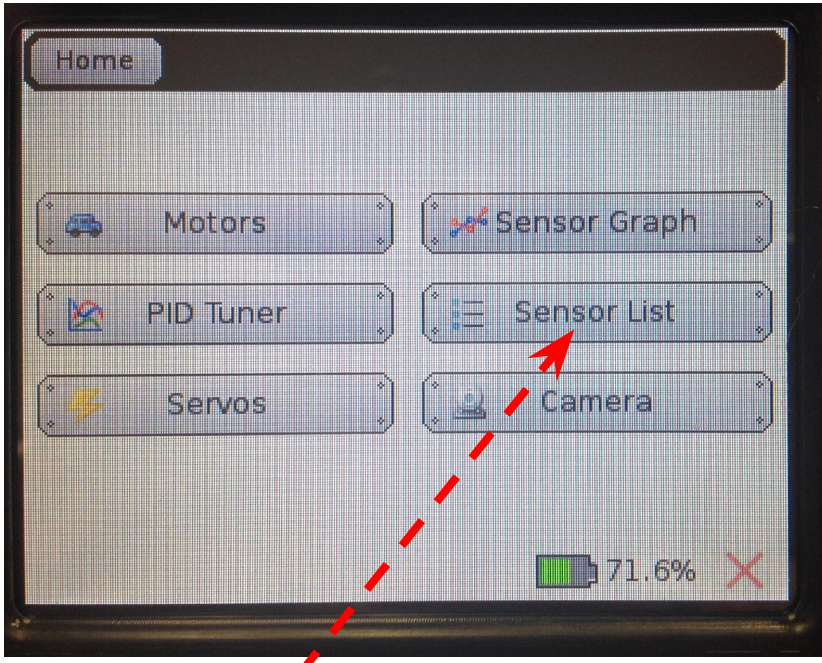


Reading Sensor Values From the Sensor List

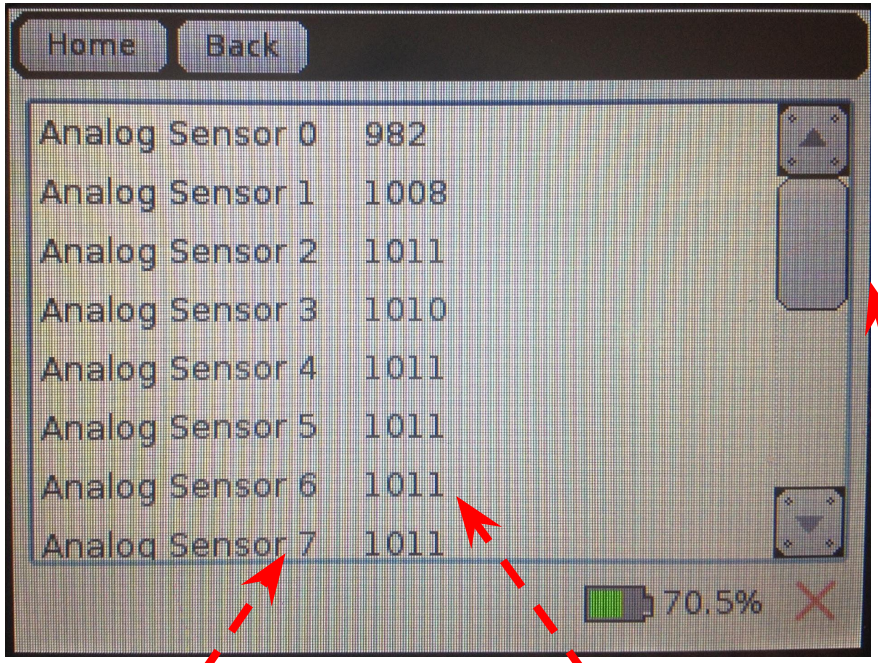
You can access the Sensor Values from the Sensor List on your Wallaby

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



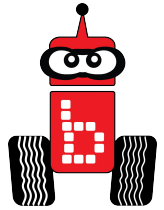


Check Touch Sensor on Wallaby Screen



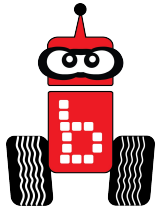
Scroll down to the digital sensor and read the value when your touch sensor is pressed and when it is not pressed

How does the Wallaby Interpret Sensor Values



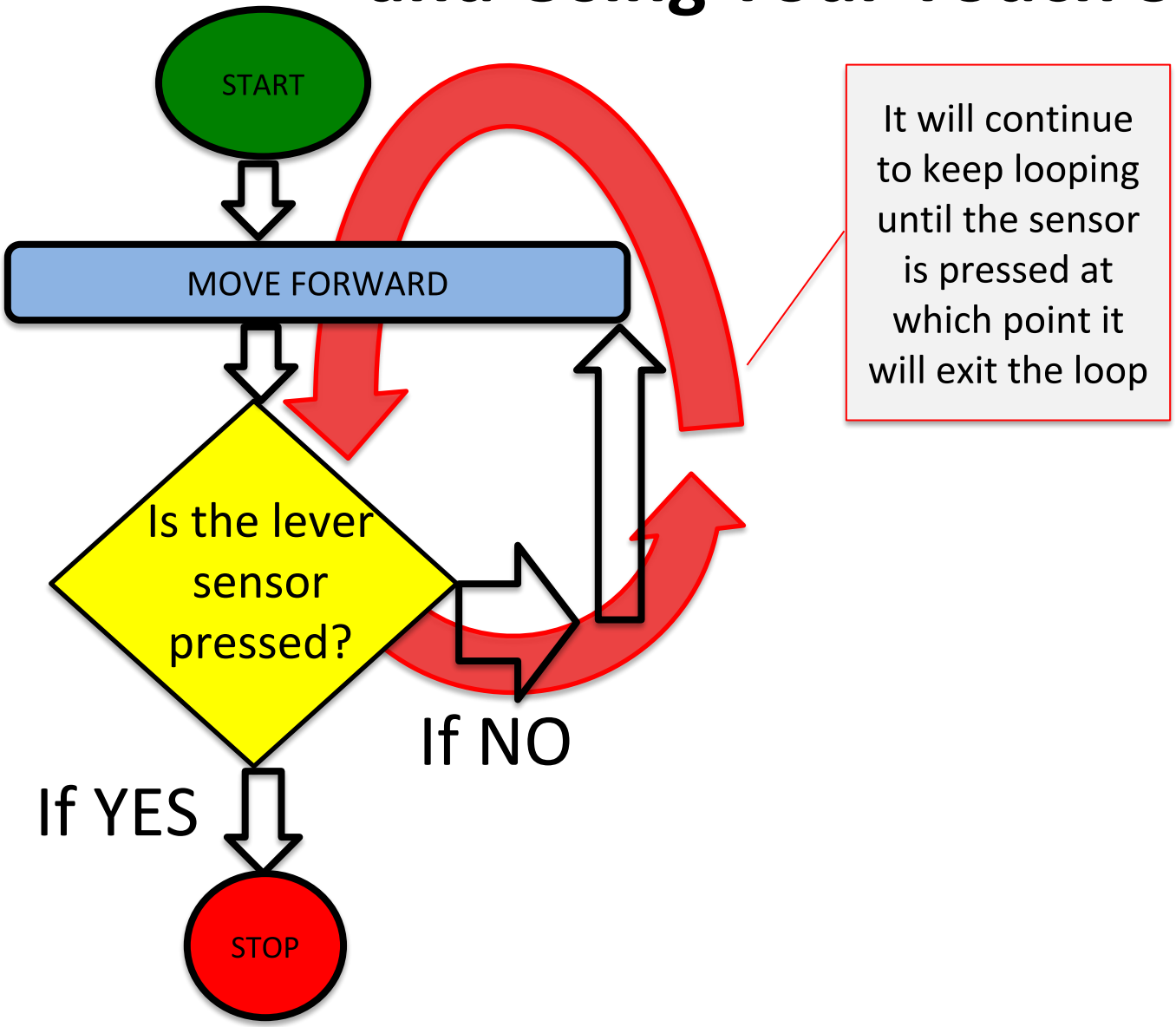
- Remember your robot controller reads the code at ~8 million lines per second
  - This is why we used the `msleep();` function to give the motors and servos time to move
- We must give the robot time to read the sensor values we are checking
  - Instead of having the program sleep (it can't read any values while sleeping), we simply need it to keep repeating the code (looping) to give it time to repeatedly read the sensor values

It's Dark, I need my Sensors!



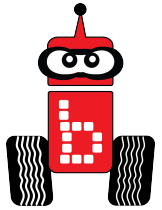
- A great analogy to use for sensors is the scenario of you walking in the dark to get a flashlight when the power goes off.
  - When you wake up at night and the lights are off, (it's dark) you put up your **digital sensors** (your hands) and you check for walls and objects in your way.
    - Do you only check one time? No!
    - You keep checking (repeating) and checking (repeating). In computer language this is called **looping!**
    - You repeat this until your sensors sense the wall and

Looping Your Program and Using Your Touch Sensor



It will continue to keep looping until the sensor is pressed at which point it will exit the loop

Understanding `while` Loops



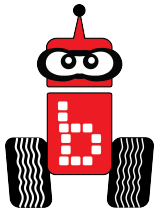
We accomplish this loop with a `while` statement.

`while` statements keep the block of code running (repeating/looping) so that sensor values can be continually checked and a decision can be made.
The while statement checks to see if something is true or false (Boolean operators).

```
while ( condition )
{
  Code to execute while
  the condition is true
}
```

Notice there is no terminating semicolon after the while statement

While Statements Activity 1



Brainstorm **`while`** statement conditions with a partner.

Example:

```
while it is raining
{
      my umbrella is open and over my head
}
close umbrella and put it away
```

1.While

2.{

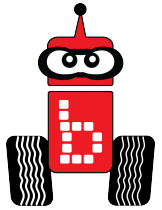
do what?

}

3. exit loop

4. close umbrella and put it away

While Statement Activity 2



1. Read and discuss this page with your elbow partner
2. Go to the next slide to complete activity 2.

```
while (digital (port#)==0)
{
   motor(0,100);
   motor(3,100);
}
```

Notice **no** terminating statement

Type of sensor; analog, digital, analog

Port number; analog (0-5) digital (0-9)

Boolean logic;

> Greater than

>= Greater than or equal

< Less than

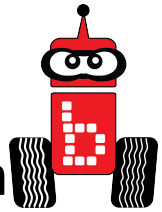
<= Less than or equal

== Equal to

!=Not equal to

Code to execute while the condition is true

While Statements Activity-2



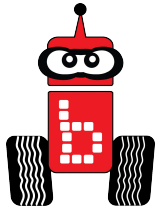
Brainstorm **while** statements with a partner utilizing a touch sensor. Use the format below.

Example:

```
while (digital (1) ==0)  // while not touched
{
      go forward
}
// as soon as the touch sensor is touched jump out of the loop
```

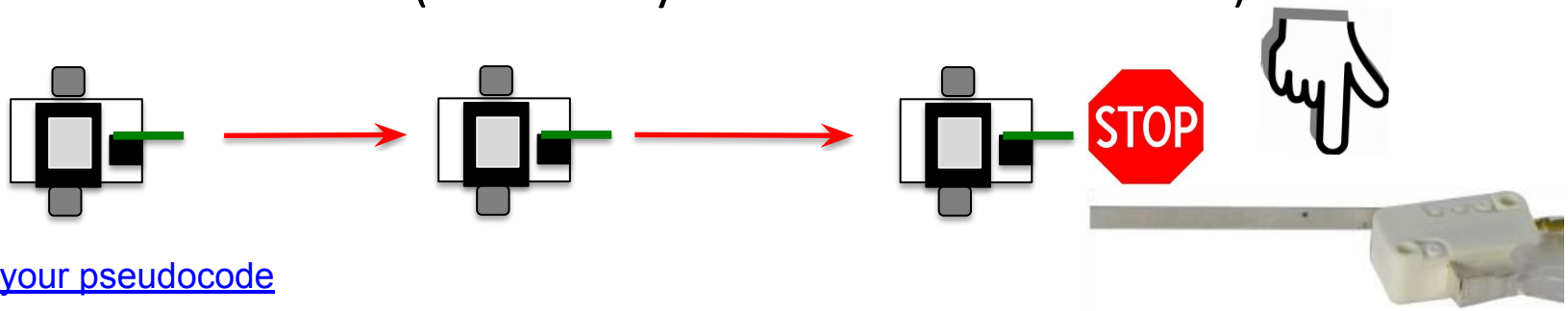
1. while (digital (port #) is == ?)
2. {
   do what?
   }
3. exit loop

Drive Until Bump Activity 3



Robot will drive forward until the long touch sensor is pressed

1. Open a new project, name it, "*your name* Drive until Bump".
2. Use the JBC Code Planning Notebook paper to pre-think your program.
3. Type, compile and run your program that will have your robot drive forward until the long touch sensor is pressed.
    - You can hold the sensor while the robot is moving and manually trigger it to stop. It helps if you hold the robot up so that it is not actually moving across the floor.
    - Write a program using a `while` statement that drives the robot forward until the lever sensor is activated.
        - Remember what digital port you plugged the sensor into (#10 is easy because it is on the end).

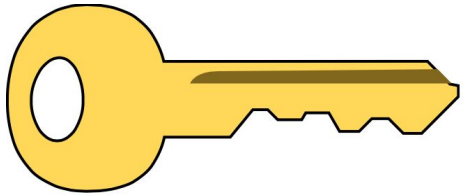


Check your pseudocode

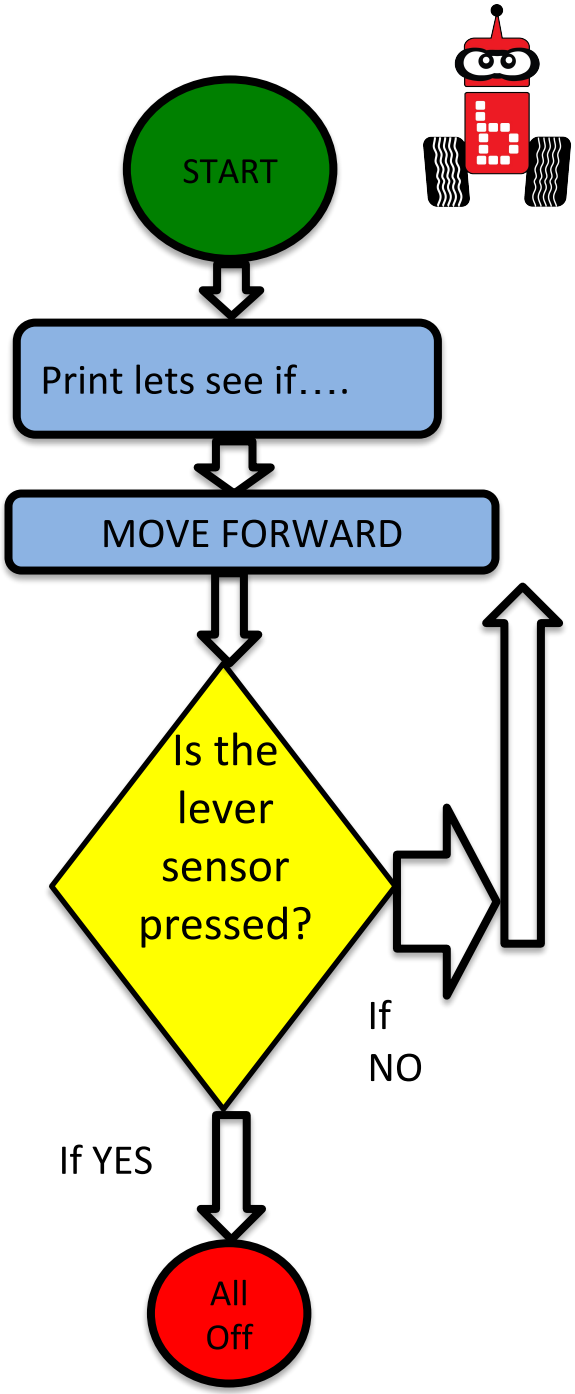
Drive Until Bump

Pseudocode (Task Analysis)

1. Print let's see if we can stop with a touch sensor
2. Check the sensor value in digital port 15 and when not pressed == 0 (aka true)
3. Keep checking and drive forward
4. Exit loop when sensor value in digital port is pressed == 1 (aka NOT true)
5. Shut everything off



Click Key for Solutions



Pseudocode in your notebook

int main()

{

printf(“Carol Drive until Bump”);

1. enable
2. arm horizontal
3. claw open

while (digital(0)==0)

{

motor(0,100);

motor(3,100);

}

1. after } jumped out, close claw
2. go back

ao();

return 0;

}

Another example of Pseudocode

1. enable
2. arm horizontal
3. claw open
4. while loop ==0
5. forward, no sleep
6. after } jumped out, close claw
7. go back
8. disable

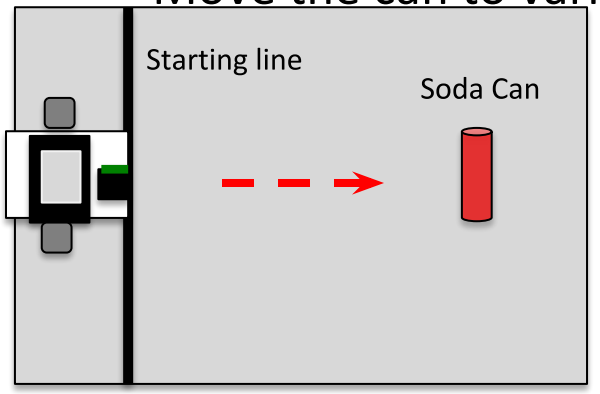
Bump the Can and Go Home

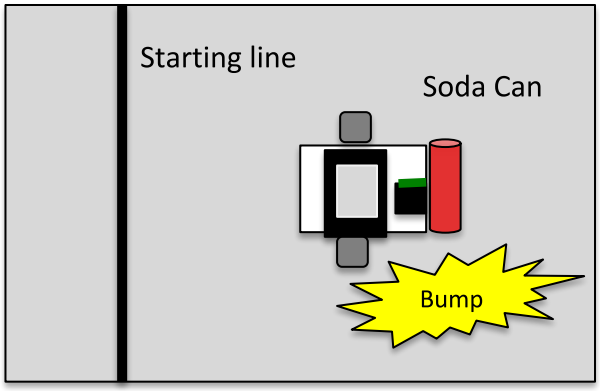
Activity 4

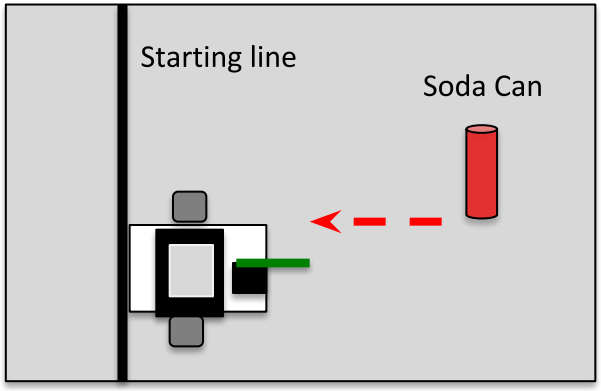
A variation on Touch, Closest to and Recycle the Can.

Engineering*

- Students need to attach the long lever sensor to the front of their robot so that it will touch the object first
- Use the long lever sensor to detect when you have touched the can and then return to the starting line
- Move the can to various distances

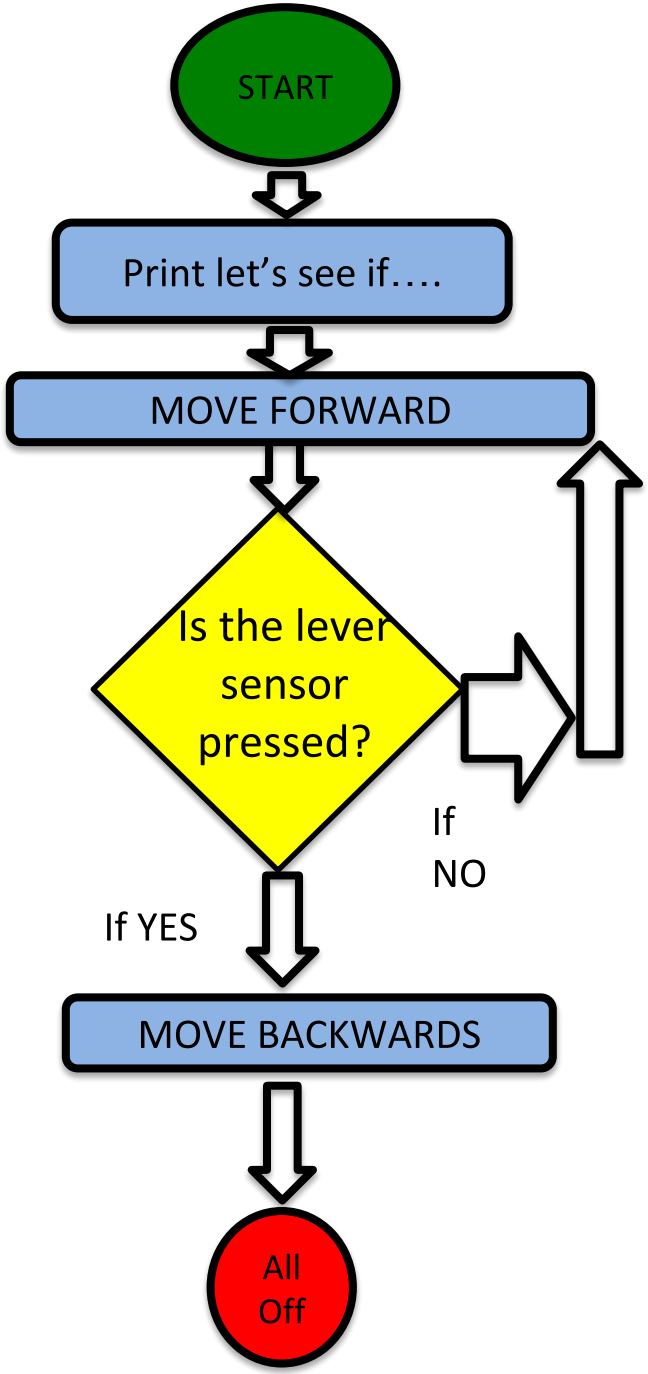


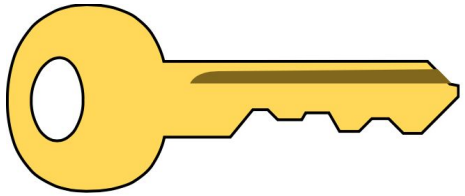




[See flowchart psuedocode for Bump Can and Go Home]

Teacher Hint!





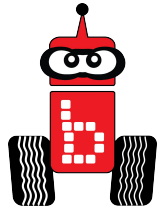
Click Key for Solutions

While Statement Activity 3

1. Change the expected sensor value from a 0 to a 1
2. Predict how the robot will behave
3. Run the program
4. Discuss results
5. Run again this time holding the touch sensor closed

```
#include <kipr/botball.h>
int main ()
{
printf("Drive_until_bump\n");
while (digital (0)==1)
{
    motor(0,100);
    motor(3,100);
}
 ao();
 return 0;
}
```

“Smart Claw” Activity 5

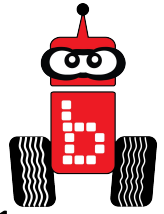


Preparation

- Have a robot with a claw capable of grabbing a can (refer to Servo lessons)
- Add a touch sensor to the robot’s claw so that the claw will close when the can touches the sensor
  - You can use LEGO, Bolts, Tape, #M Sticky Dots to attach the sensor to the robot
    - Make sure the sensor is solidly attached
    - Make sure the can can actually trigger the sensor in the claw

1. Open a new project, name it, “*your name* smart claw”.
2. Use the JBC Code Planning Notebook paper to pre-think your program.
3. Type, compile and run your program that combines the touch sensor and claw to grab a can.

Assessments



Assessment 15: Tag and Bring Home

Setup: Use Surface-A. One empty 12oz soda can randomly placed in circle 2, 6, or 11.

Desired Outcome: The robot will go out, sense the can, and then return it to the starting box.

Limitations:

1. All robots must be autonomous (no remote controls, wireless communication, or touching the robot after starting a run).
2. The robot must start completely behind the vertical projection of the inside of the start line.
3. The students must have their robot lined up and ready to go before the can is placed on the mat.
4. Once the can is placed, the student starts the robot (student cannot reposition, change program, etc.).
5. If the robot brings the can back to the starting box (can must break the vertical projection of the inside outside boundary of the starting line) the student can remove the can and reposition their robot for another run.
6. The teacher will take the can and place it again at random in circle 2, 6, or 11 (except not in the same circle as any previous successful runs).

Completion:

- A completion occurs when they have returned at least two cans and brought them back to the starting box.

Learning About Analog Sensors

Analog Activity 1:

1. Read and follow the slides:
   - Analog Sensors
   - Checking Values
   - Reading from the Sensor List
2. After reading and following the slides, go to ET Activity 1.

Analog Sensors

- Returns the analog value of the port (a value in the range 0-4095). Analog ports are numbered 0-5.
- Light sensors, range finders and reflectance are examples of sensors you would use in analog ports.

"ET"-rangefinder

Small IR Reflectance Sensor

Light Sensor

ET Activity 1

1. Extend your arm in front of you with your thumb pointed up.
2. Focus on your thumb and then slowly bring your thumb toward your face.
3. What happens when your thumb gets close to your face?
   - Did it get blurry? Yes! It got within the focal point of your eyes (where you could focus on it and make it clear)
4. The ET sensor also has a focal point and if the object is too close the sensor cannot tell if it is close or far away.
5. When attaching your et sensor to your robot consider the ~4 cm distance from you sensor to its focal point

Understanding the ET Sensor
Activity 2

ET Activity 2: Infrared “ET" Distance Sensor

1. Read, discuss and follow the slides:
   - ET Information
   - Check ET Sensor on Wallaby Screen
   - ET sensor Values
   - Optical Rangefinder
   - Learning to Use an Analog Sensor
2. After reading and following the slides, go ET Activity 3.

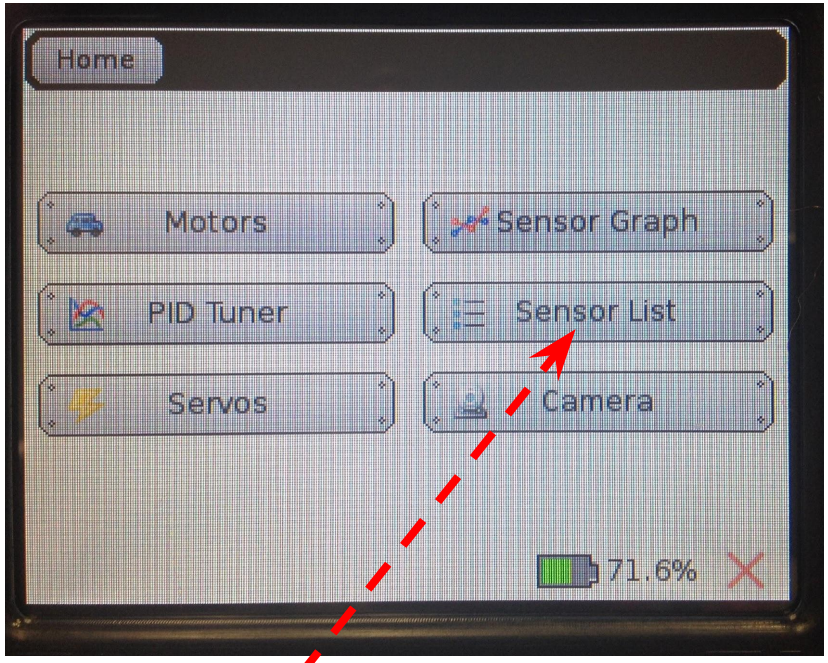
ET Sensor Information

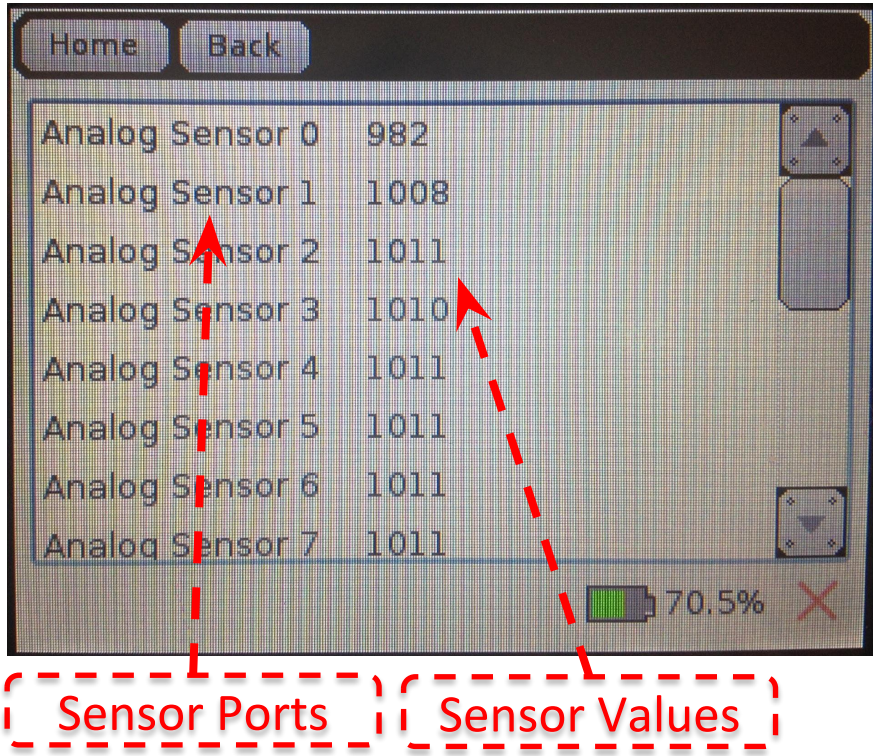
- Plug your ET sensor into an analog port
- ET sensor uses a special function:
- **Low values:** indicate greater distance (farther from robot)
- **High values:** indicate shorter distance (closer to robot)
- Optimal range is between 4” and 40’.
- 0”-3” values are not optimal.
- Objects closer than the focal point(4”) will have the same readings as those far away.

Reading Sensor Values From the Sensor List

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

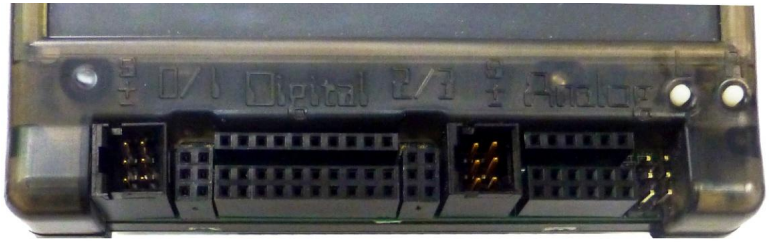
- It is very helpful to get readings from all of the sensors you are using, and then you can then use the values when writing your code



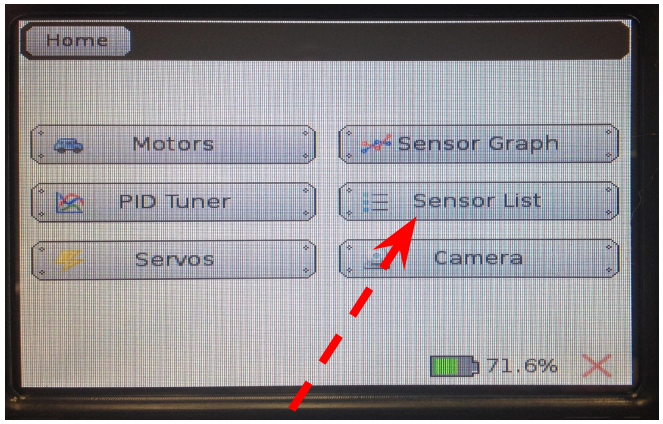


Check ET Sensor on Wallaby Screen

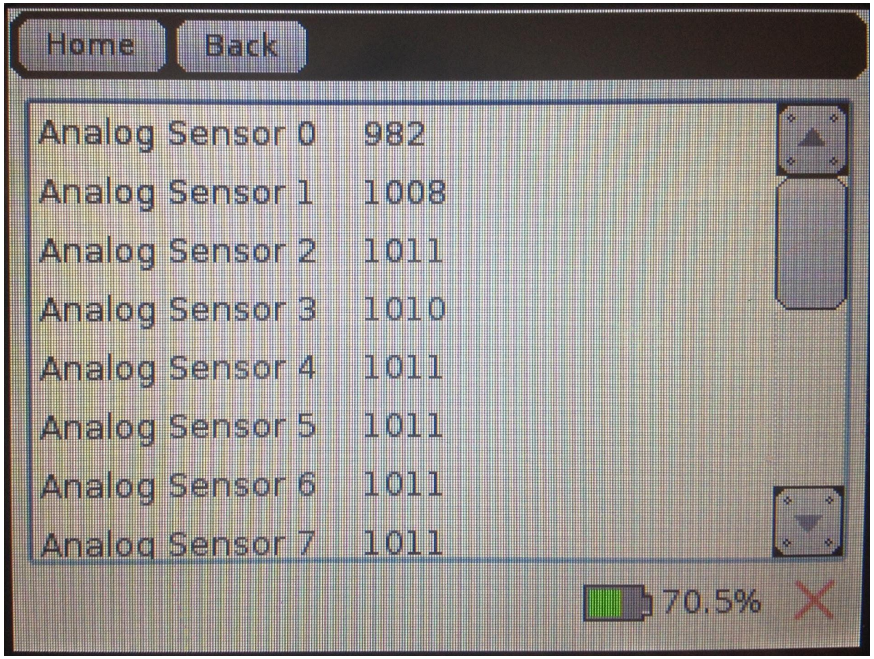
1. Plug the ET sensor into any of the analog ports (0-5)



Plug your ET into any Analog port



2. Select Sensor List

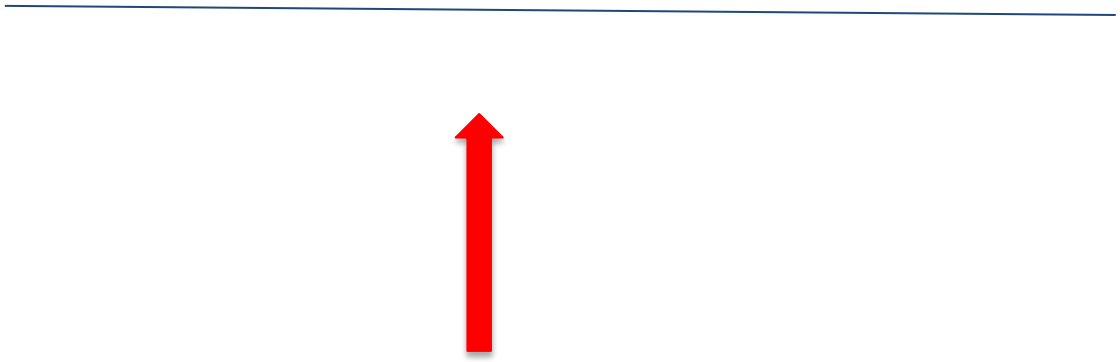


3. Read the values by placing your hand in front of the et sensor.
Move your hand close to the sensor and then far away. Notice the values.

4. The point where you see the values suddenly change from getting larger to getting smaller is your “Focal point”. See next slide.

Create a number line to represent your et sensor

Find your focal point; where the numbers change



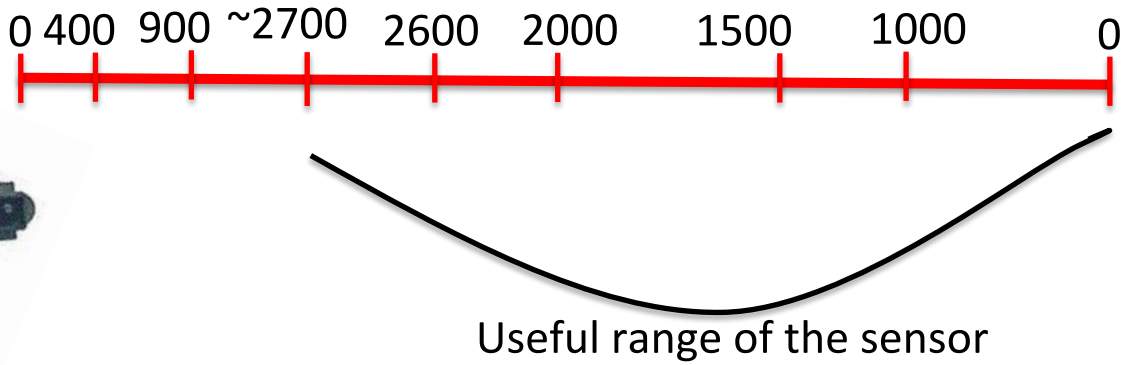
Find your focal point

ET sensor Values

Focal Point

Objects that are inside the focal point return a smaller #, too close to object

Objects that are farther away return a smaller number



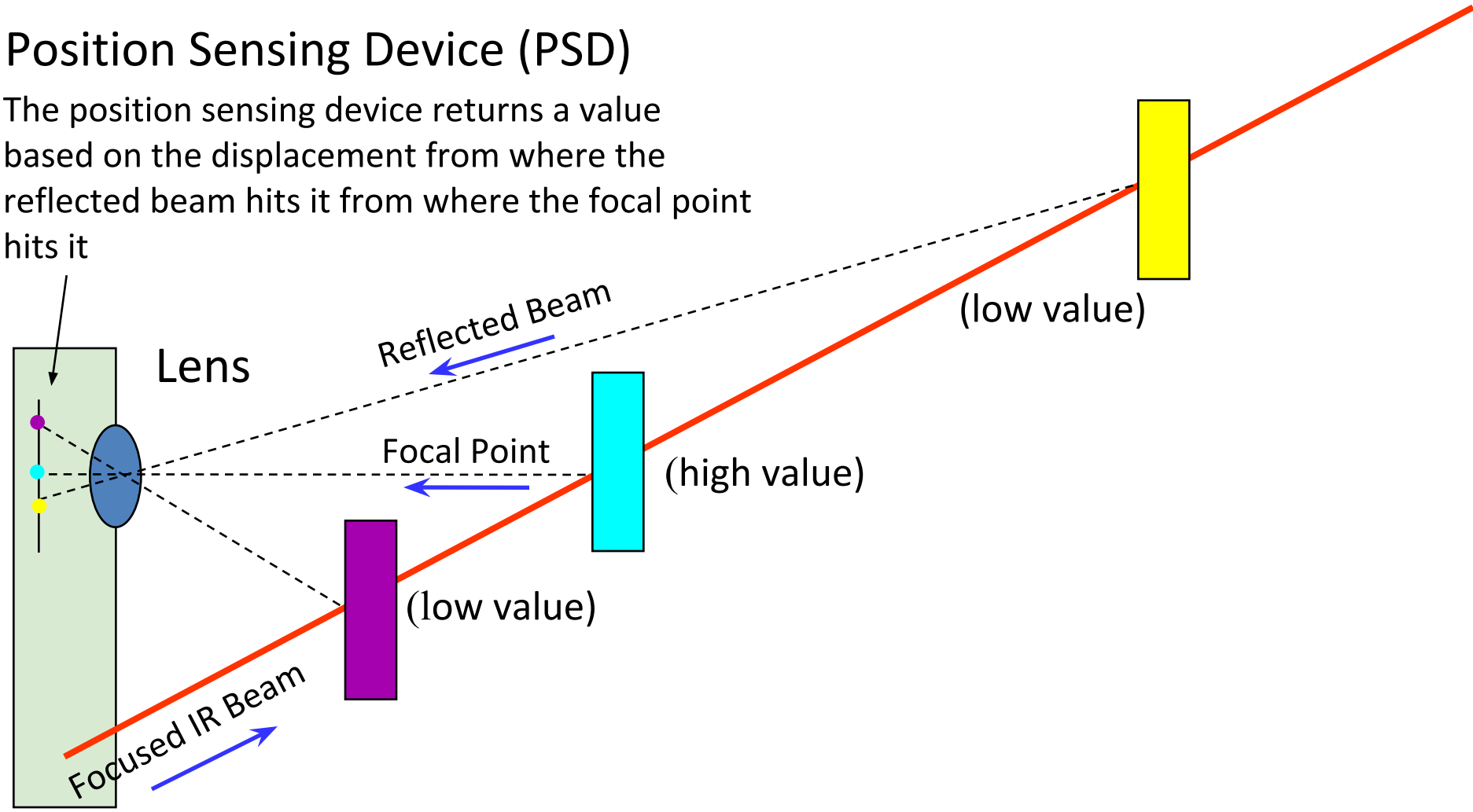
See Learning to use an Analog Sensor. You may need to adjust the value up or down a little for your desired distance from an object.
Optimal distance is about 4”

Optical Rangefinder

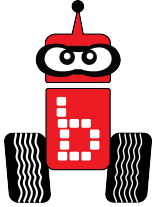
- Sensor emits a narrow infrared beam, and measures the angle of the beam return using a position-sensitive detector (PSD):

Position Sensing Device (PSD)

The position sensing device returns a value based on the displacement from where the reflected beam hits it from where the focal point hits it



Learning to Use an ET Analog Sensor



```
while (analog (port#)<=?)
```

Notice **no** terminating statement

Type of sensor; analog, digital,

Port number; analog digital

Boolean logic

> Greater than

>= Greater than or equal

< Less than

<= Less than or equal

== Equal to

!=Not equal to

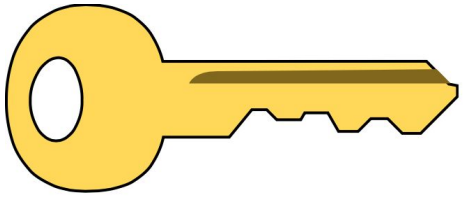
```
{
   motor (0,100);
   motor (3,100);
}
```

What you want it to repeat while checking to see if the while statement is true

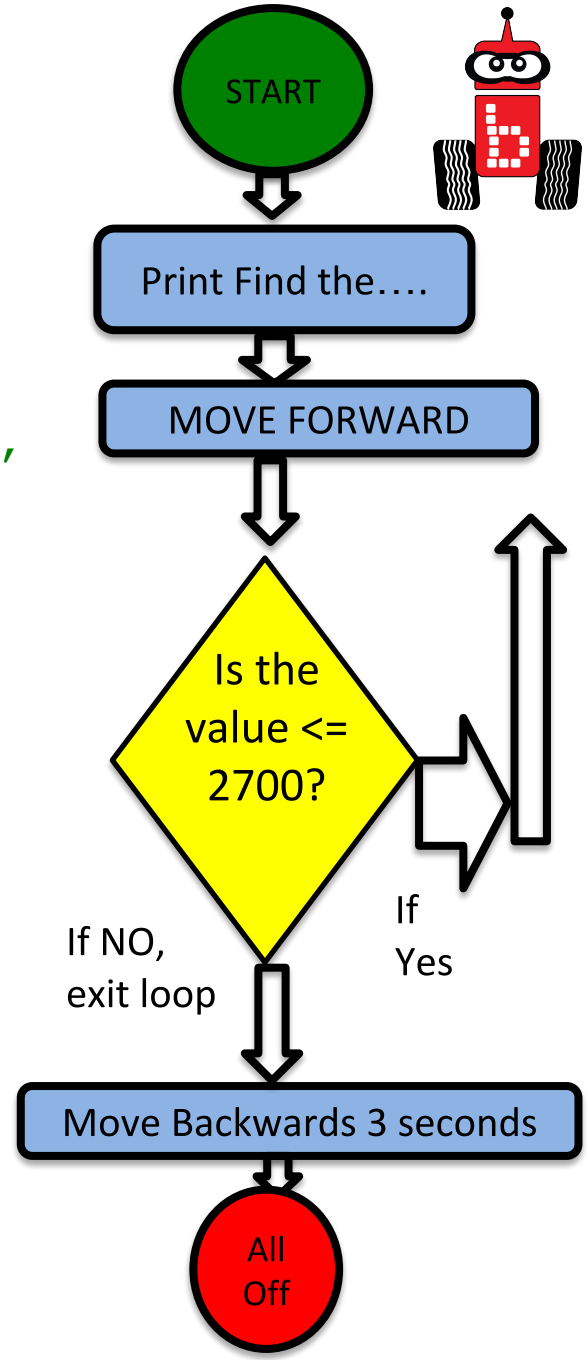
ET Activity 3
Find the Wall and Back Up

Use the JBC Code Planning Notebook paper and write the Pseudocode (Task Analysis)

1. `//Print Find the Wall and Back Up`
2. `//Check the sensor value in analog port 1, Is the value <=2700?`
3. `Drive forward as long as the value is <=2700 (or your determined value)`
4. `//Exit loop when value is 2700(or your determined value) or greater`
5. `//Back up for 3 seconds`
6. `//Shut everything off`
7. Open a new project, name it, "*your name* FInd the Wall".
8. Follow your psuedocode to type, compile and run your program.



Click Key for Solutions



Conditionals with ET

Objective: Robot will make choices. It will go forward or go backwards depending on the value of return from the ET.

- To make choices we need to use an "if statement".

`if` Statements = Choices

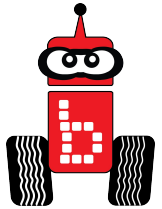
`if` statements allow the code being run to make a choice (If the bump sensor is pressed, do this)

Just like the `while` statement no semicolon is used after the `if` statement

```
if (value)
{
  Execute this block of code- whatever is
  between curly braces
}
```

*You can use `if` statements within a `while` loop

Checking Values



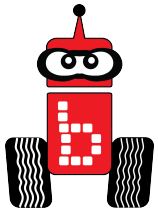
- When writing code you use OPERATORS that allow the program to check a value stored against another value to determine if it is True or False.

Boolean operators

> Greater than	5 > 4 is TRUE
< Less than	4 < 5 is TRUE
>= Greater than or equal	4 >= 4 is TRUE
<= Less than or equal	3 <= 4 is TRUE
== Equal to	5 == 5 is TRUE
!= Not equal to	5 != 4 is TRUE

You can print the JBC Tent and distribute a copy to each student.

While Statements

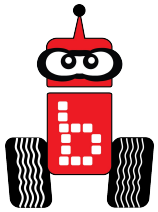


Brainstorm **`while`** **`and`** **`if`** statement conditions with a partner.

Example:

```
while I'm walking on the sidewalk
{
        if there is no one coming
        {
                walk in the middle of the sidewalk
        }
        If someone is coming
        {
                get to the right side of the sidewalk
        }
}
exit the loop if I'm not on the sidewalk
```

While Statements



Brainstorm **`while`** **`and`** **`if`** statement conditions with a partner.

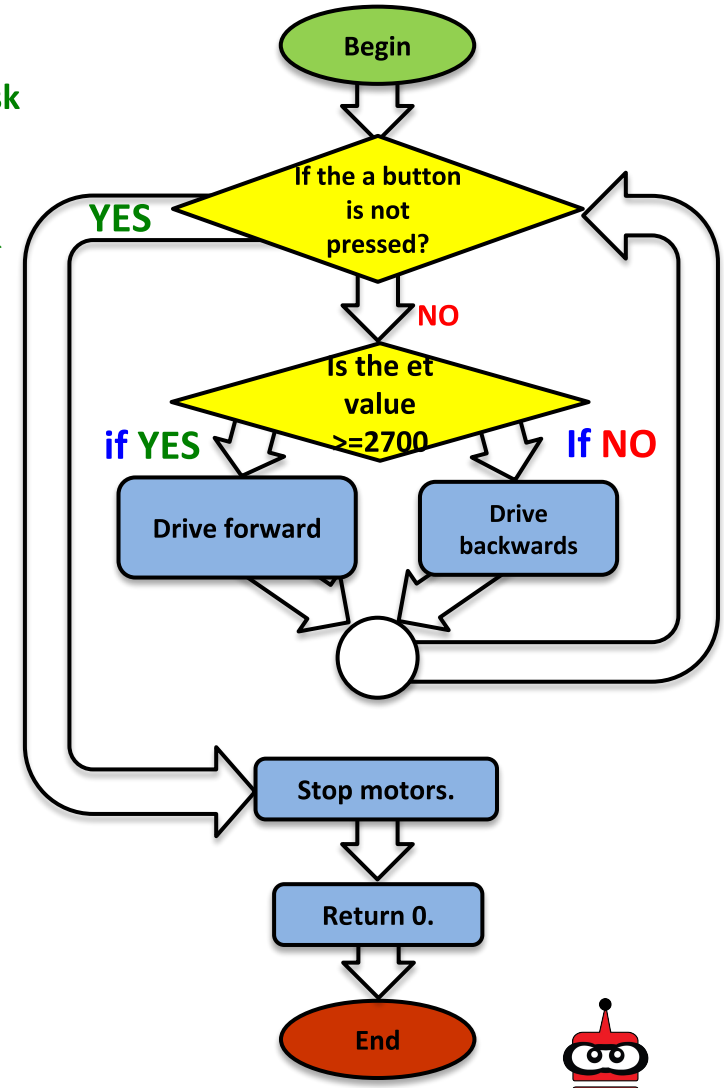
Example:

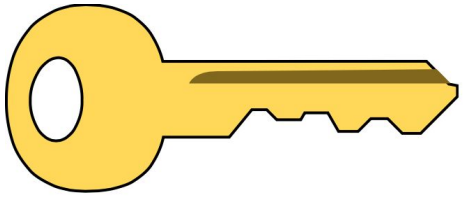
```
while the a button is not pressed
{
        if et range is ??
        {
            go forward
        }
        if et range is ??
        {
            go backwards
        }
}
exit the loop if the a button is pressed
```

ET Activity 4
Find the Wall and Back Up and Go forward

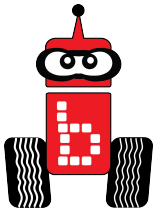
Use the JBC Code Planning Notebook paper and write the Pseudocode (Task Analysis)

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`
6. Open a new project, name it, "*your name* FInd the Wall".
7. Follow your psuedocode to type, compile and run your program.

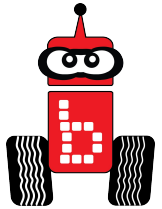




Click Key for Solutions



Assessment



Assessment 16: Proximity

Setup: Use Surface-A. One ream (500 sheets) of standard copy paper.

Desired Outcome: On two separate runs, the robot has to sense the wall (ream of paper) that has been randomly placed on the mat and drive out to it, stopping within approximately 4 1/4” (the width of a piece of paper folded in half lengthwise) of the wall without touching it.

Limitations:

1. All robots must be autonomous (no remote controls, wireless communication, or touching the robot after starting a run).
2. The robot must start completely behind the vertical projection of the inside of the start line.
3. Once the robot is in starting position, a ream of paper is placed on edge (long side down and parallel to the starting line) at either circles 4, 6, 9 or 11.
4. Once the ream of paper is set, students can push “run” on their robot.
5. Robot must come to a complete stop within approximately 4 1/4” (the width of a piece of paper folded in half lengthwise) without touching the wall with any part of the robot.

Completion: When the robot goes out, senses the wall and stops within approximately 4 1/4"” of the wall without touching it on two different runs.

Extra Optimization: Require students to stop closer or further away.

Reflectance Sensor
Activity 1

1. **Read and follow the directions on the following slides:**
   - Analog Small and Large Top Hat Sensors
   - Reflectance Sensor Ports
   - Plug in Your Reflectance Sensor
   - Reading Your Sensor Values
   - Understanding the IR Sensor
2. After reading and following the slides, go to Find the Black Line Activity 1

Analog Sensor: Small Top Hat Sensors

This sensor is really a short range reflectance sensor. There is an infrared (IR) emitter and an IR collector in this sensor. The IR emitter sends out IR light and the IR collector measures how much is reflected back.

- Amount of IR reflected back depends on surface texture, color and distance to surface

This sensor is excellent for line following

- Black materials typically absorb IR and reflect very little IR and white materials typically absorb little IR and reflect most of it back
  - If this sensor is mounted at a fixed height above a surface, it is easy to distinguish a black surface from a white surface
  - Connect to analog port 0-5

Reflectance Sensor Ports

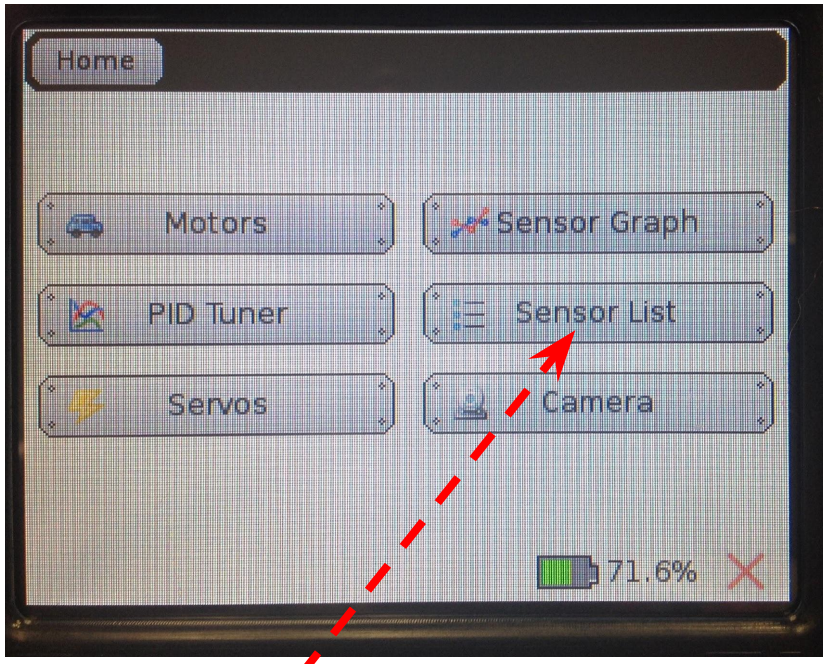
1. This is an **`analog()`** sensor so plug it into any of your analog ports
   - Values will be between 0-4095
   - Mount the sensor on the front of your robot so that it is pointing to the ground and ~1/4” from the surface

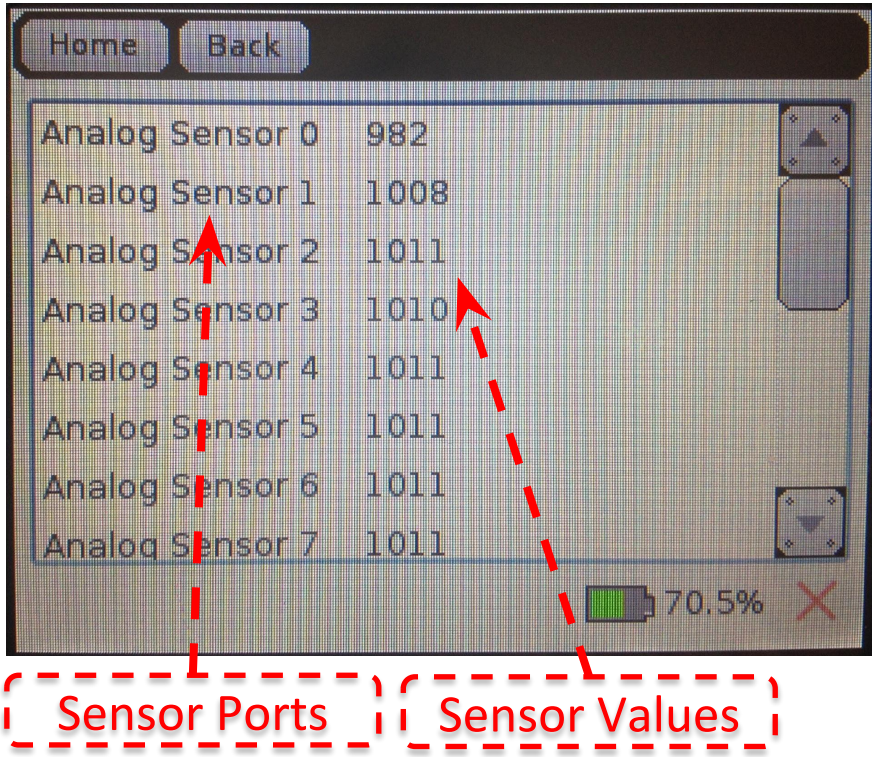
Surface

Reading Sensor Values From the Sensor List

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

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



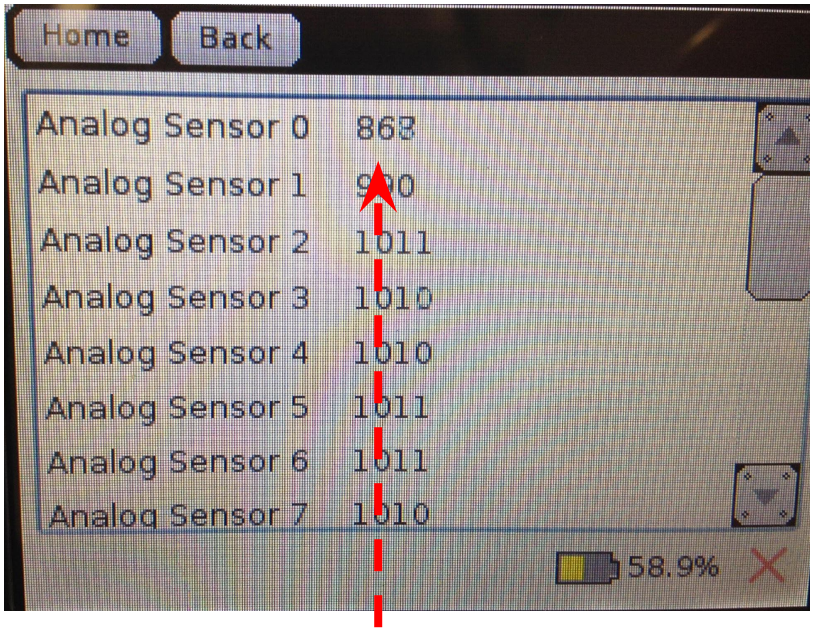


- Continue to next slide

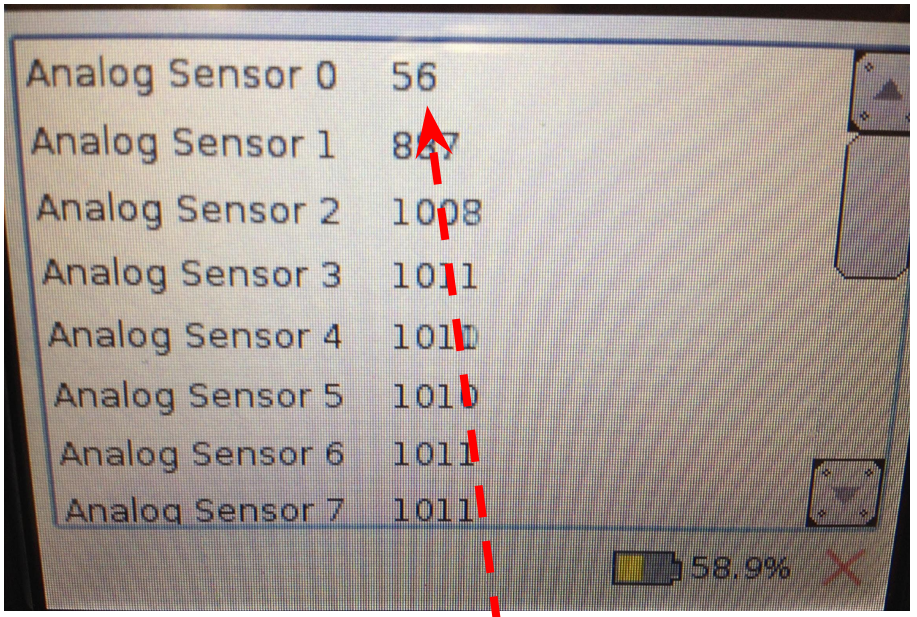
Reading Sensor Values From the Sensor List (Cont.)

With the IR sensor plugged into analog port #0

- Over a white surface the value is (56)
- Over a black surface the value is (1250)



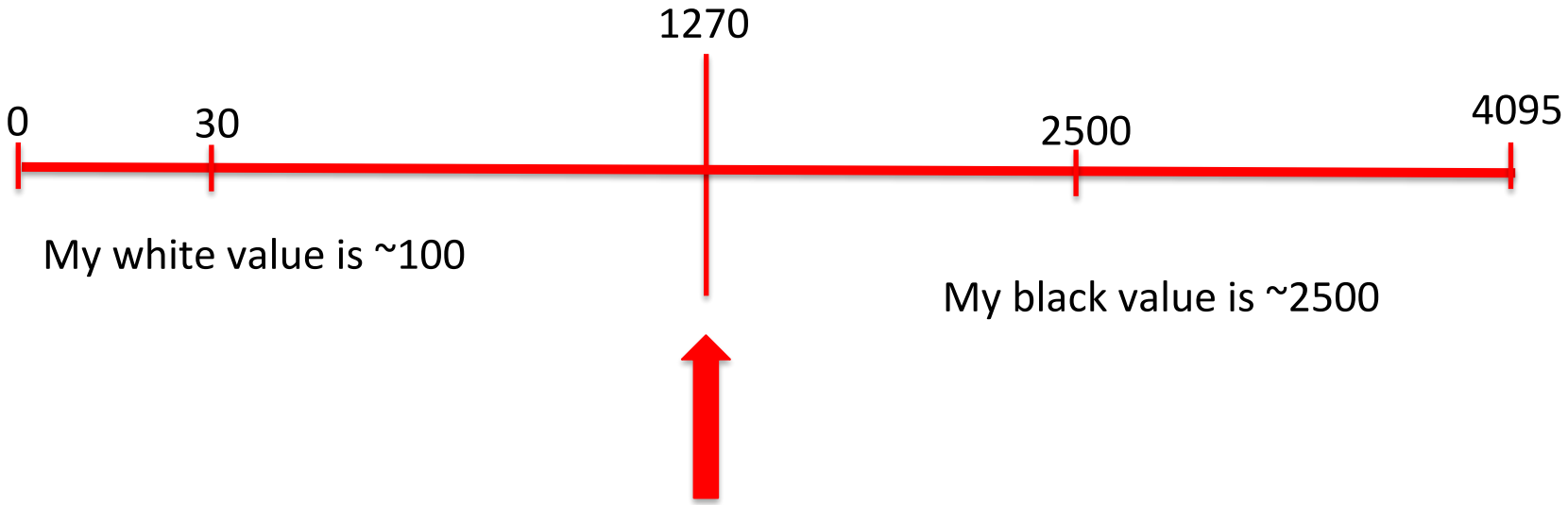
Your IR sensor is correctly mounted when you have values between 1200- 1500 on the Black Surface



Your IR sensor is correctly mounted when you have values between 30-60 on the White Surface.

Understanding the IR Values

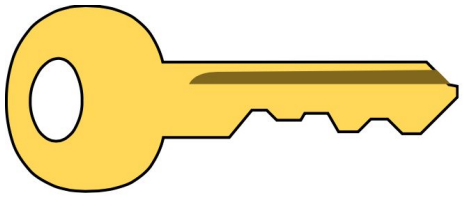
1. Place your IR analog sensor in one of the analog ports (0-5).
2. After mounting your IR sensor, check that the values are: white 30-60 and black 1200-1500; write down your values.
3. Find your threshold or middle value.
4. This number will be the value you need for the find the black line activity.



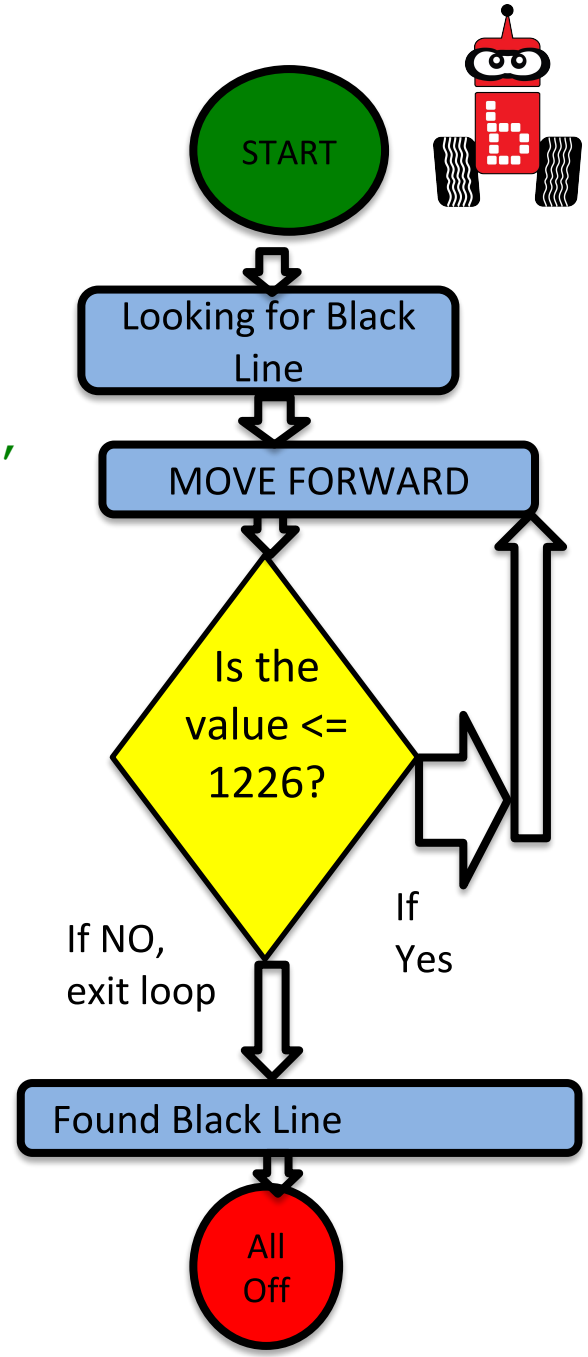
Find the Black Line
Activity 2

Use the JBC Code Planning Notebook paper and write the Pseudocode (Task Analysis)

1. `//Prints looking for black line`
2. `//Check the sensor value in analog port 0, <=1225`
3. `drive forward as long as the value is <=1225`
4. `//Exit loop when value is 1226 or greater`
5. `//Prints Found Black Line`
6. `//Shut everything off`
7. Open a new project, name it, "*your name* Find the Line".
8. Follow your psuedocode to type, compile and run your program.



Click Key for Solutions



Learning about `if` Statements

1. Read, discuss and follow the slides:
   - `if` Statements
   - Line Following Strategy
   - Buttons
   - Understanding `while` and `if`

2. After reading and following the slides, go to Line Following Activity 1

`if` Statements = Choices

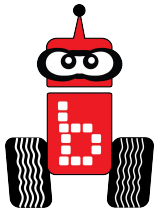
`if` statements allow the code being run to make a choice (If the bump sensor is pressed, do this)

Just like the `while` statement no semicolon is used after the `if` statement

```
if (value)
{
  Execute this block of code- whatever is
  between curly braces
}
```

*You can use `if` statements within a `while` loop

While Statements

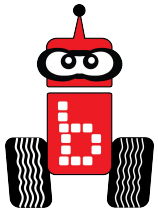


Brainstorm **`while`** **`and`** **`if`** statement conditions with a partner.

Example:

```
while it is raining
{
        if my girlfriend is under the umbrella with me
        {
                shift the umbrella over her head
        }
        If my girlfriend leaves
        {
                shift the umbrella to cover my head
        }
}
exit the loop if the rain stops
```

While Statements



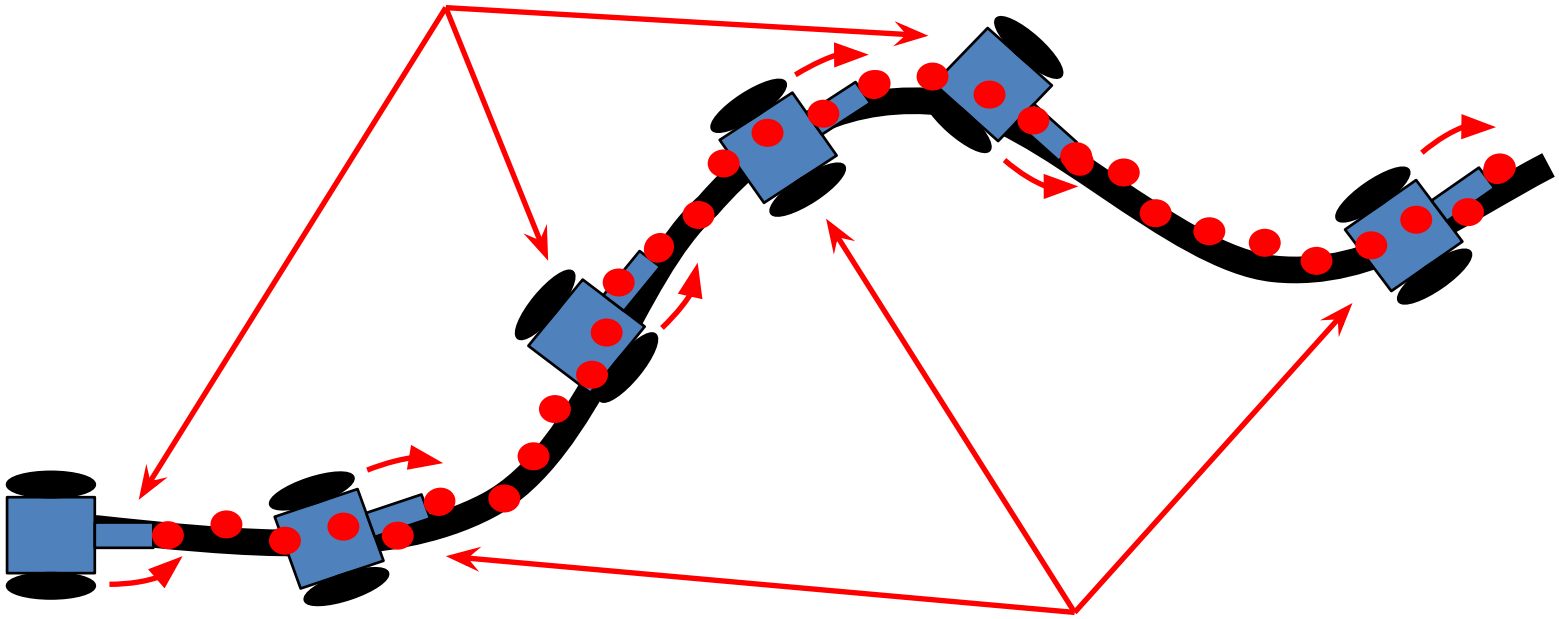
Brainstorm **`while`** **`and`** **`if`** statement conditions with a partner.

Example:

```
while I'm walking on the sidewalk
{
        if there is no one coming
        {
                walk in the middle of the sidewalk
        }
        If someone is coming
        {
                get to the right side of the sidewalk
        }
}
exit the loop if I'm not on the sidewalk
```

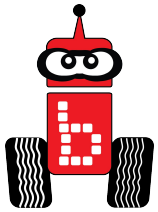
Line Following Strategy

Line Following Strategy: While - Is the button pushed?
Follow the line's left edge by alternating the following 2 actions:
1. If detecting dark, arc/turn left.



2. If detecting light, arc right.
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.

While Statements

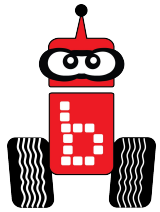


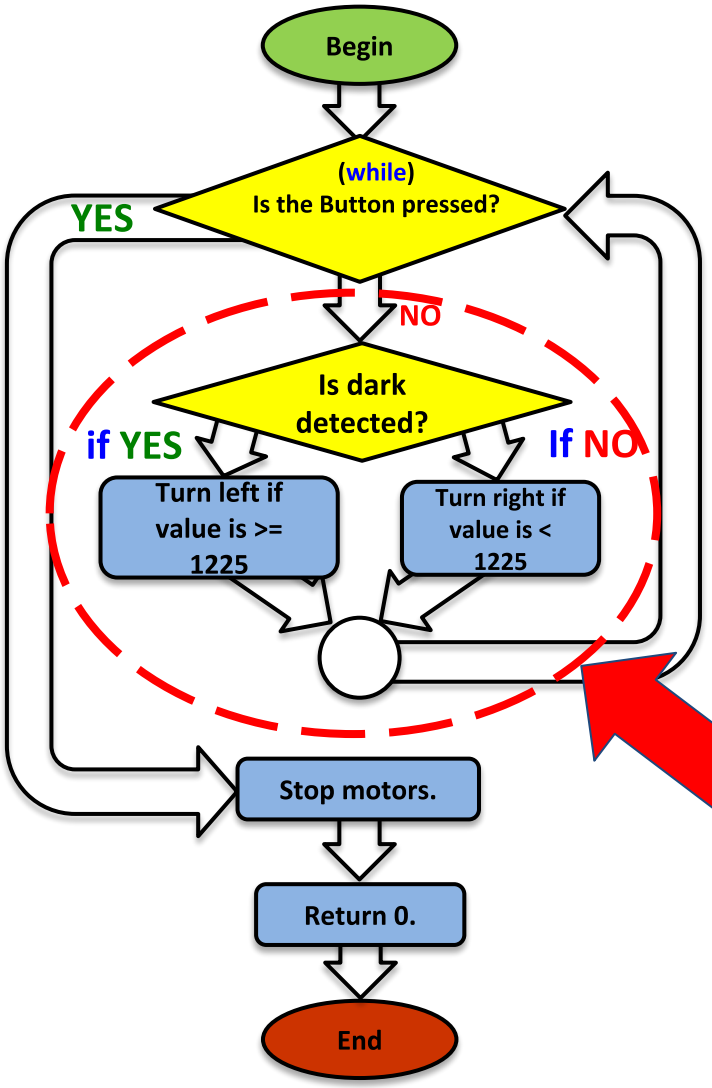
Brainstorm **`while`** **`and`** **`if`** statement conditions with a partner.

Example:

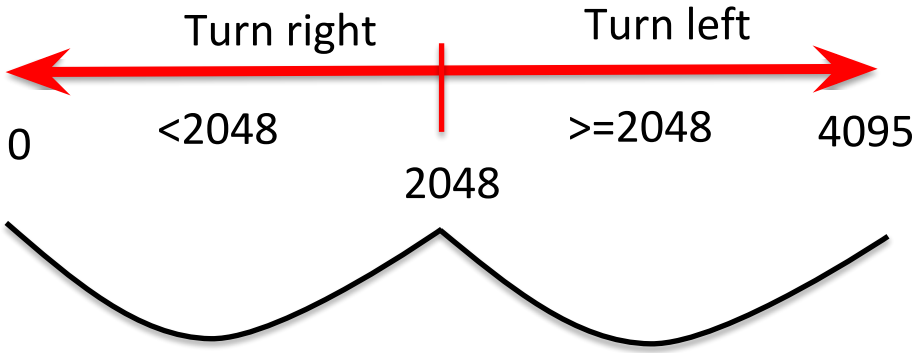
```
while following the line if the button is not pushed
{
        if black is detected
        {
                turn a sharp left
        }
        If white is detected
        {
                turn a sharp right
        }
}
exit the loop if the rain stops
```

Understanding `while` and `if`





You must cover all values



Assume all these values are WHITE

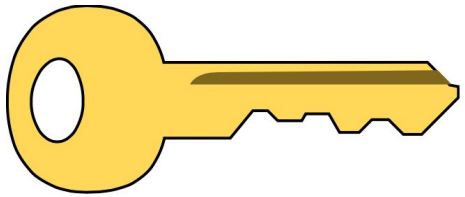
Assume all these values are BLACK

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

Line Following
Using `while` and `if`
`Activity 1`

Use the JBC Code Planning Notebook paper and write the Pseudocode (Task Analysis)

1. `Print "line following".`
2. `Checks the status of the Button=while statement`
3. `Checks the value from the reflectance sensor=if`
4. `Turns left if value is >= 1225 (or your determined value)=inside curly brackets`
5. `Turns right if value is < 1225 (or your determined value)=if statement followed by brackets`
6. Open a new project, name it, "*your name* Line Following".
7. Follow your psuedocode to type, compile and run your program.
8. If you are having trouble consider:
   - You have three variables to change.
     - How sharp or tight you are turning.
     - The value the sensor is reading.
     - The placement of the sensor
9. After programing with success look at Tip 1 , Tip 2, Tip 3.

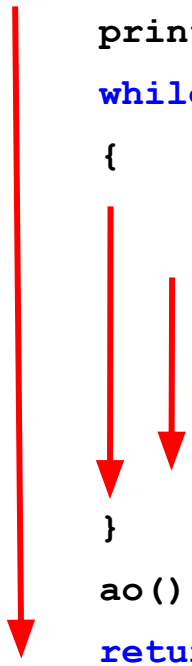


Click Key for Solutions

Tip 1

1. Align and tab over matching curly braces.

```
int main ()
{
    printf("Sarah line follow\n");
    while (digital (10)==0)
    {
        if (analog (5)>1225)
        {
          motor (0,100);
          motor (3,100);
        }
    }
    ao();
    return 0;
}
```



(Hint: Hit the Indent button at the top of the compiler and see what happens.)

Tip 2

2. Use `while`, `if` and `else`.

```
int main ()
{
  printf("Sarah line follow\n");
  while (digital(0) == 0)
  {
    if (analog(0) >= 1225)
    {
      motor (0,100);
      motor (3,-10);
    }
    else
    {
      motor (0,-10);
      motor (3,100);
    }
  }
  ao();
  return 0;
}
```

You can replace the second **`if`** statement with **`else`**

Tip 3

3. Change the threshold.

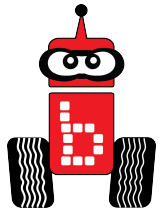
```
int main ()
{
    printf("Sarah line follow\n");
    while (digital (15)==0)
    {
        if (analog (5)>1225)
        {
            motor (0,100);
            motor (3,100);
        }
        else
        {
            motor (0,100);
            motor (3,100);
        }
    }
    ao();
    return 0;
}
```

The value of 1225 or the “threshold” value is ½ way between the observed values.
Remember black reflects less IR than white so the value is lower.
Notice the Boolean operators >= 1225 or < 1225
Your value may be much lower due to lighting, placement and turns

You can replace the second `if` statement with `else`

Assessment

Assessment 17: Walk the Line



Setup: Use Surface-B.

Desired Outcome: The robot will follow the black line from start to finish.

Limitations:

1. All robots must be autonomous (no remote controls, wireless communication, or touching the robot after starting a run).
2. The robot must start completely behind the vertical projection of the inside of the start line.
3. The robot must be following the line. Dead reckoning will not be allowed.
4. Lines are only counted as touched if all the driving wheels touch the colored line.

Completion: When the robot's drive wheels touch or cross the blue line.

Extra Optimization: Challenge students to make it all the way to the finish line.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Buttons

Having buttons (Digital) on the controller can be very useful when programming your robot

On the KIPR Wallaby there is physical button (named *side*) and 6 soft buttons (named *a,b,c,x,y,z*) on the screen

- All have *name*`_button()` functions which return 1 if the button is being pressed and 0 otherwise

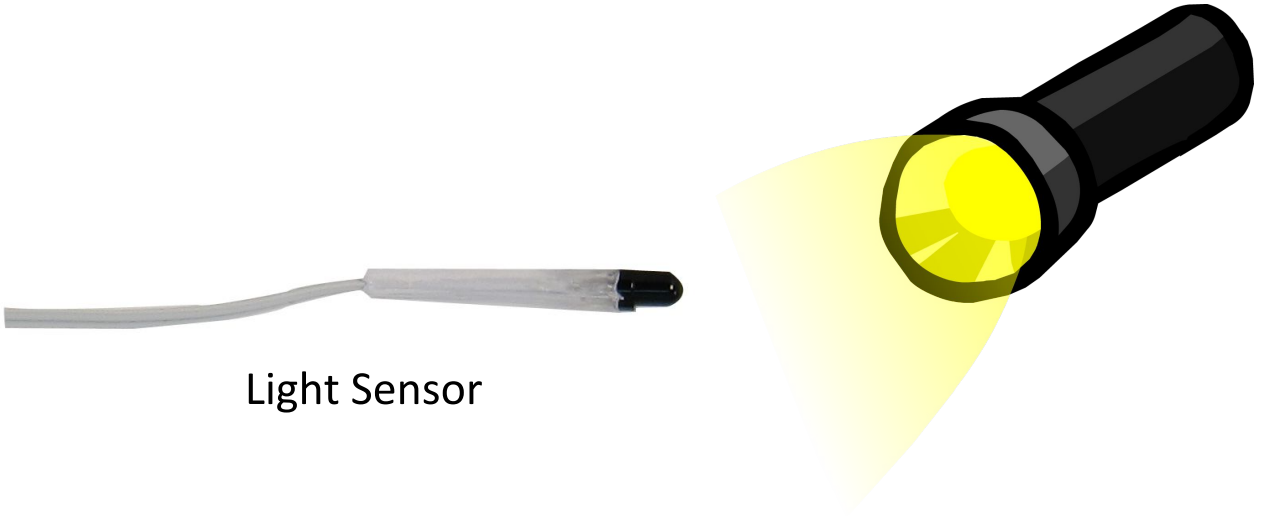
Example: a_button() ==0;

Using the Light Sensor

1. Get a light sensor (refer to picture below)

   The light sensor senses *infrared light* (human cannot see infrared), so light must be emitted from an *incandescent light*, not an *LED light*

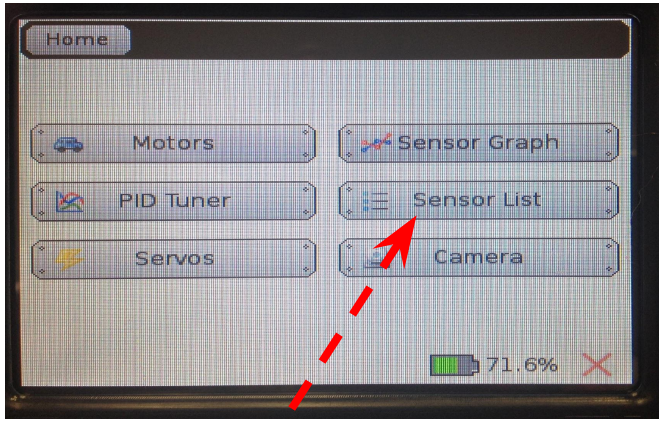
   - You can use a non LED flashlight.



Light Sensor

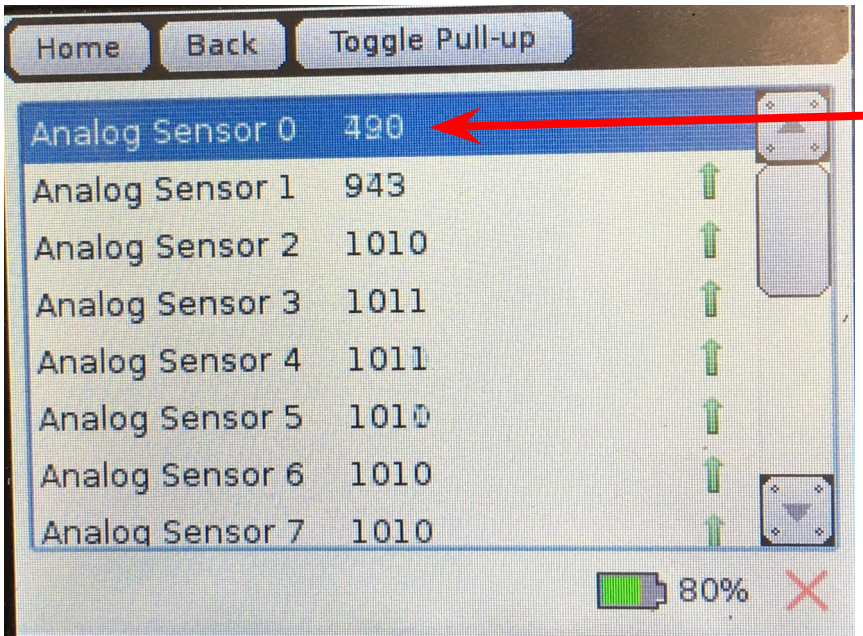
Using the Light Sensor

1. Plug the light sensor into analog port 0



Plug your light sensor into analog port 0

2. Select Sensor List



3. Read the values without your light shining on the sensor and then again when you shine the light on your sensor

*The ***more*** light (infrared) detected, the ***lower*** the reported value.

`wait_for_light()` function

The `wait_for_light` function allows your program to run when your robot senses a light.

- **Note:** When the program calls this function, it has a built-in calibration routine that will come up on the robot screen
  (a step-by-step guide to this calibration routine is on a following slide).

```
wait_for_light(0);
```

Port #

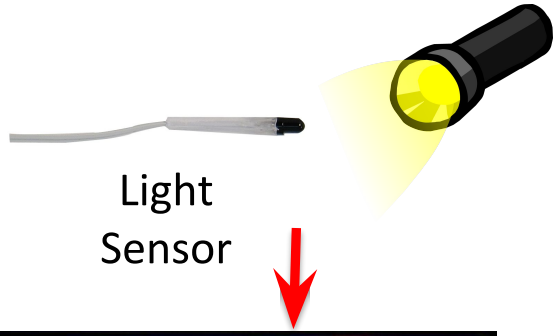
wait_for_light()

1. Start a new project and call it wait_for_light
2. Use this code for your new project

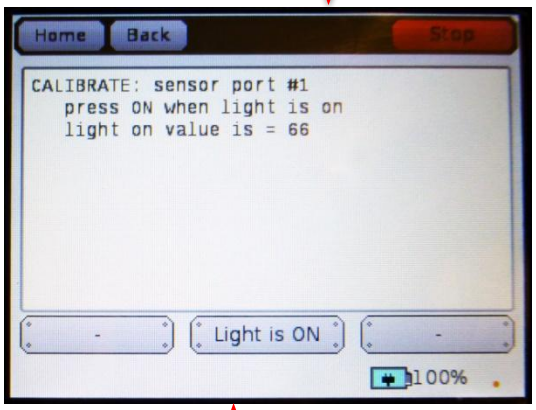
```
int main()
{
   wait_for_light (0);
   printf("Sarah, I see the light!\n");
   return 0;
}
```

3. Compile your program and run it on your robot
4. Go to next slide for calibration routine

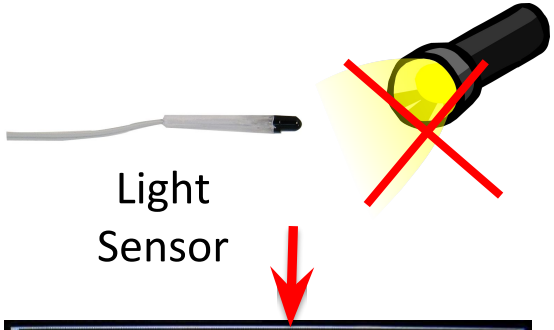
5. Follow the calibration instructions on the robot screen



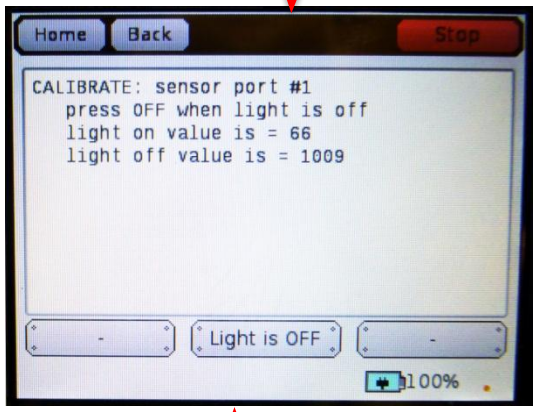
Light Sensor



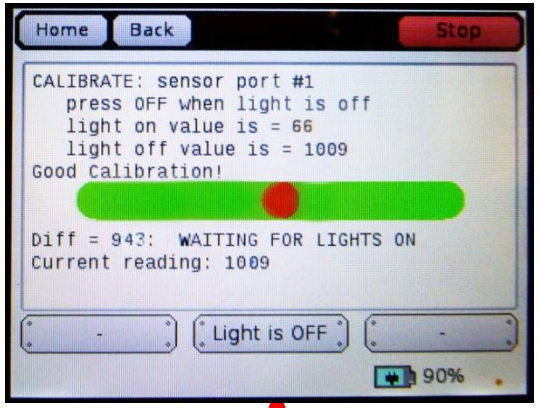
When the light is *on* (low value), press the "Light is On" button.



Light Sensor



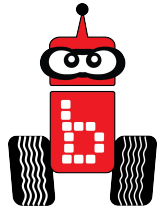
When the light is *off* (high value), press the "Light is Off" button.



You will get a "Good Calibration!" message and moving red dot on green bar when done *correctly*.
You will get a "BAD CALIBRATION" message when not done correctly, and you will need to run through the routine again.

6. Shine the light onto your light sensor to start the program

Wait For Light



Activity :

Write a program for the KIPR Wallaby that waits for a light to come on, drives the robot forward for 3 seconds, and then stops.

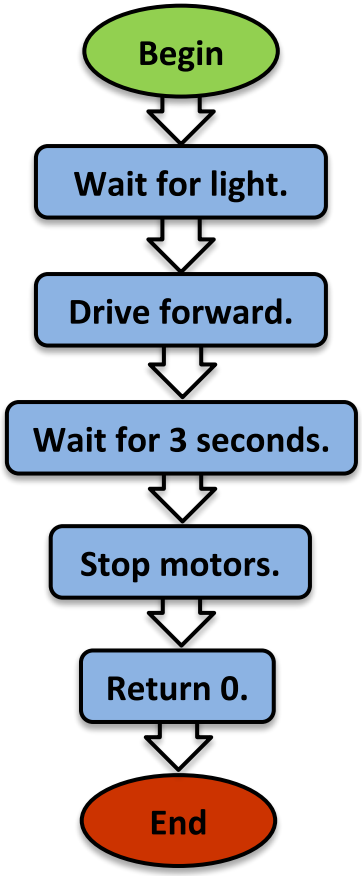
Pseudocode

1. Wait for light.
2. Drive forward.
3. Wait for 3 seconds.
4. Stop motors.
5. End the program.

Comments

```
// 1. Wait for light.
// 2. Drive forward.
// 3. Wait for 3 seconds.
// 4. Stop motors.
// 5. End the program.
```

Flowchart



Check your psuedocode

Assessments and Rubrics

Suggestions: *Understanding* or *Group Collaboration* rubrics

KISS INSTITUTE FOR PRACTICAL ROBOTICS