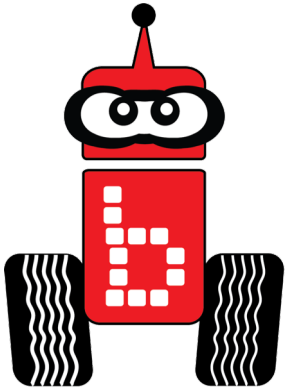


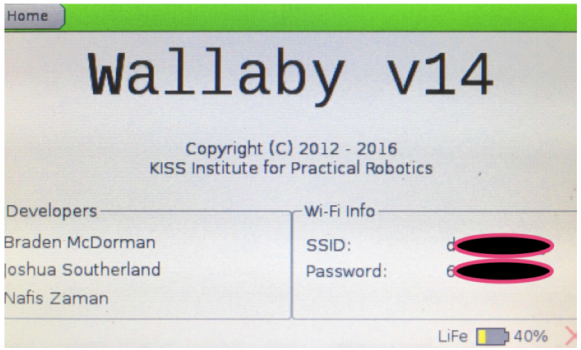
Moving Your Robot Using the `motor()` Function

KIPR Module 5

Connecting to Your Robot

Connect the **Wallaby** to your Browser

- Turn on the Wallaby with the **black switch on the side**.



- Every wallaby has a unique number/name that will appear in your Wi-Fi list.
- To find the number of your wallaby click “about” on your controller.
- Look at the Wi-Fi info and see the SSID: that is your wallaby number and wallaby password.
- Go to the your Wi-Fi icon on your computer and click on your wallaby number,
  and add your password (this may take a few minutes). You will see a blue LED light.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Warning

Do not go to the *Wi-Fi* setting page when starting up your wallaby. Go to the about page only.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Connecting to KIPR Software Suite (using Wi-Fi)

Launch your web browser (such as Chrome or Firefox).

Copy this IP address into your browser's address:

192.168.125.1:8888

IP address Port #

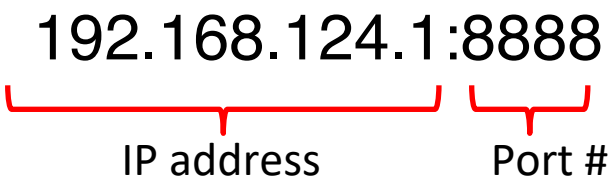
The KISS Software Suite will now come up in your browser.

You may use a computer, tablet or even a smart Phone.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

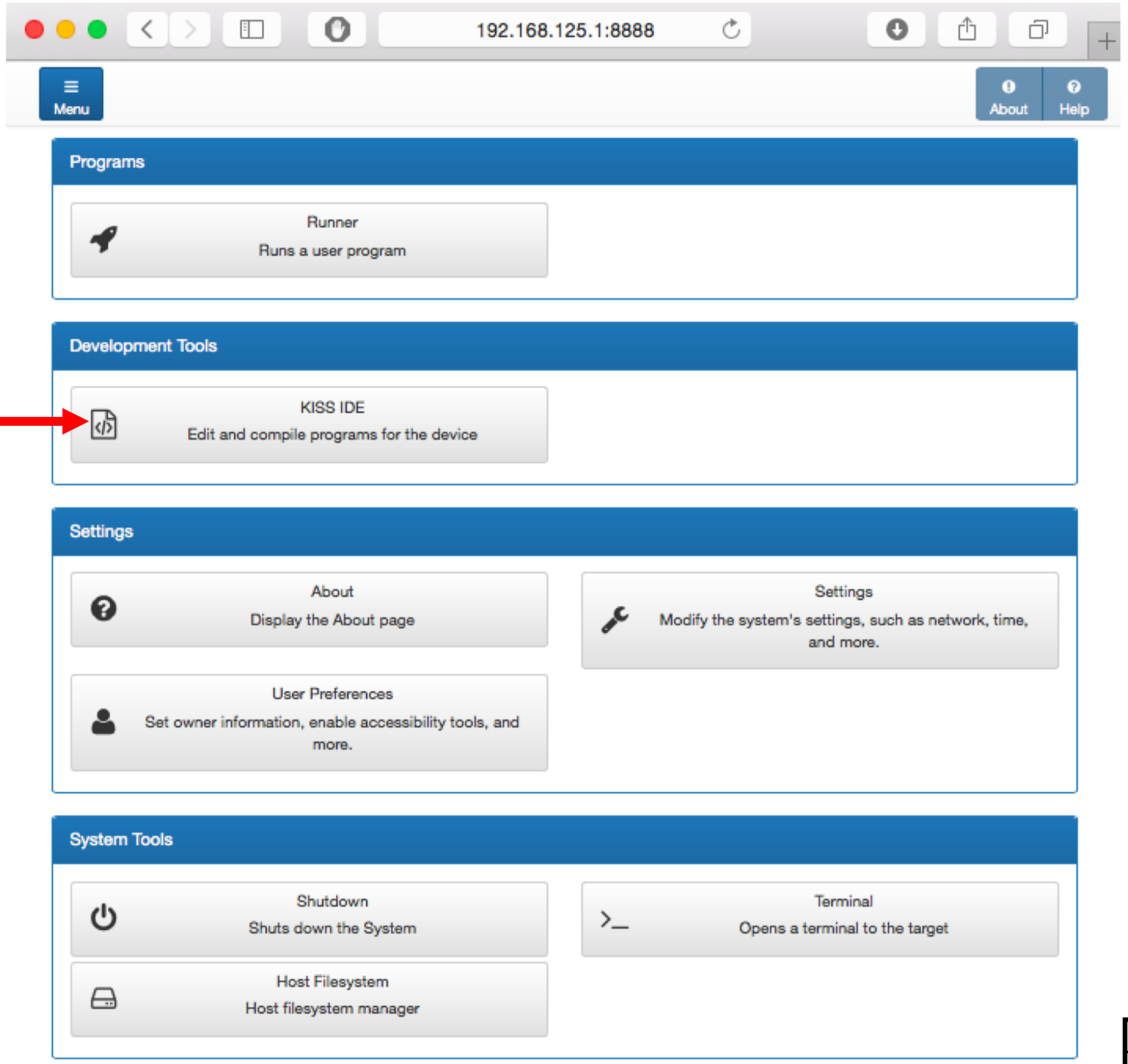
Accessing the KIPR Software Suite Using the USB Cable

1. Launch your web browser (such as Chrome or Firefox).
2. Copy this IP address into your browser's address bar followed by ":" and port number 8888; e.g.,



3. The KISS Software Suite will now come up in your browser.
4. You may use a computer, tablet or even a smart phone as long as they can connect through Wi-Fi.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

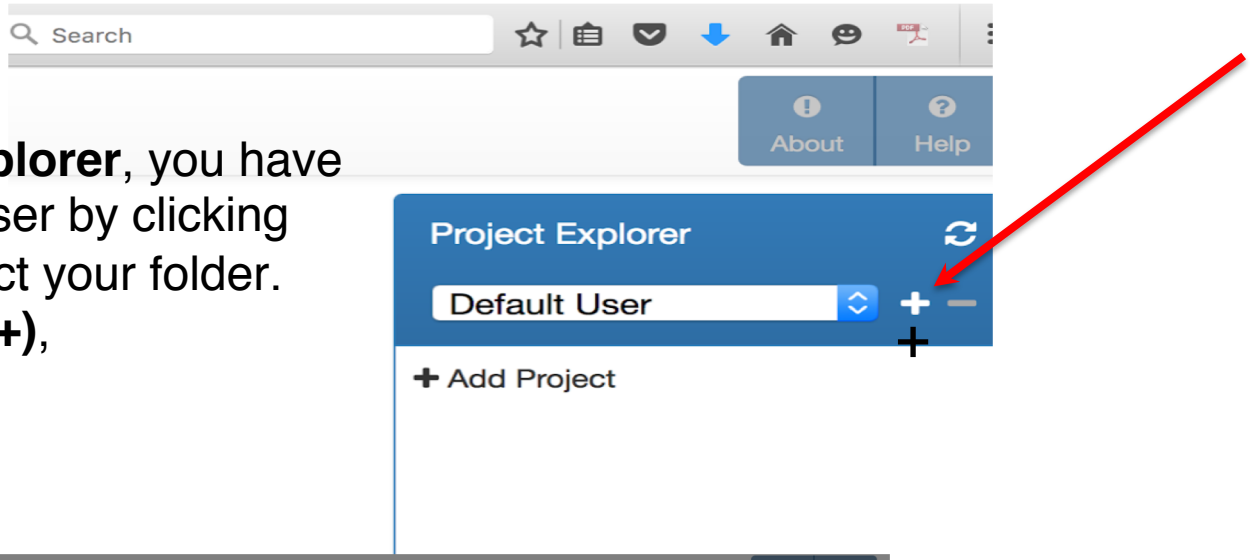


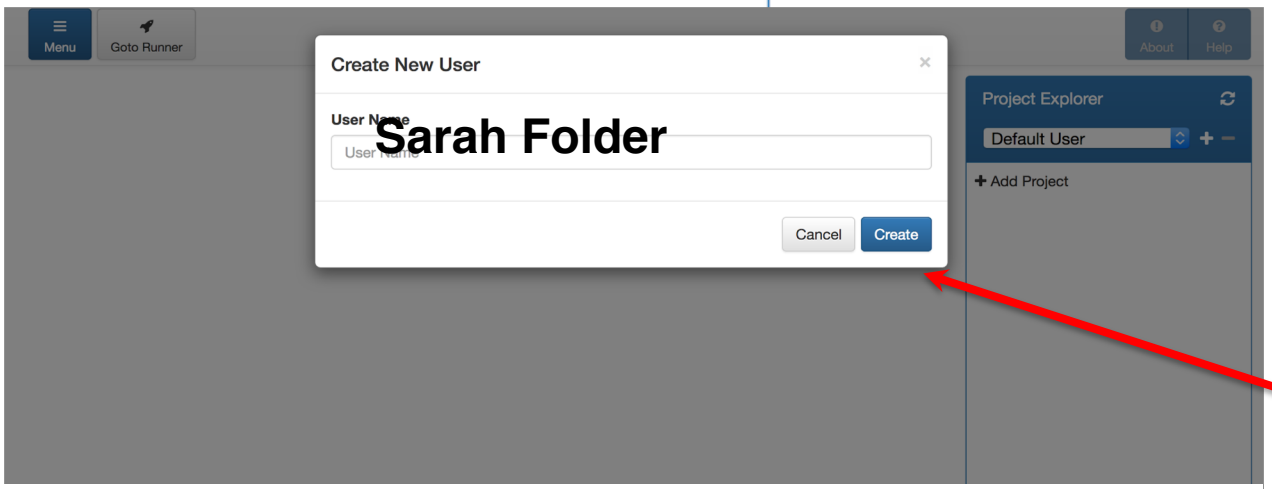
Click KISS IDE

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Launch the KISS IDE

1. Under **Project Explorer**, you have options to add a user by clicking the + sign, **or** select your folder.
2. Add a **New User (+)**,





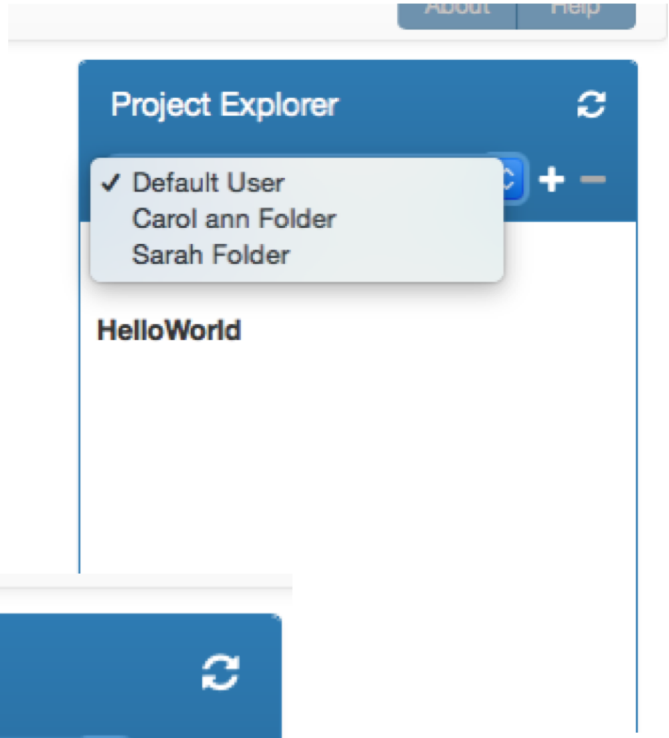
Click Create

User name =“Student name and folder” . This is similar to a folder you will put all your different projects into. No special characters, and no (. , ’ ” ; : _)

KISS INSTITUTE FOR PRACTICAL ROBOTICS

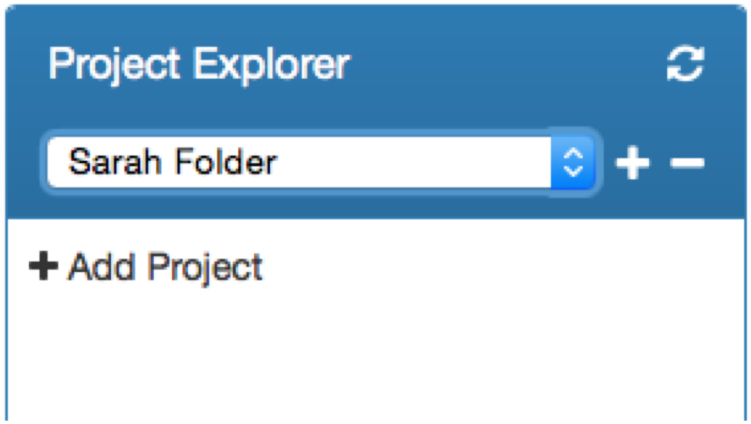
Add a Project

Go back to “Project Explore” and select the **User Name** you created from the drop down. This is the folder you created.



Click “+Add Project”
You are adding a project to your folder.

7. Name your project **“Move”**

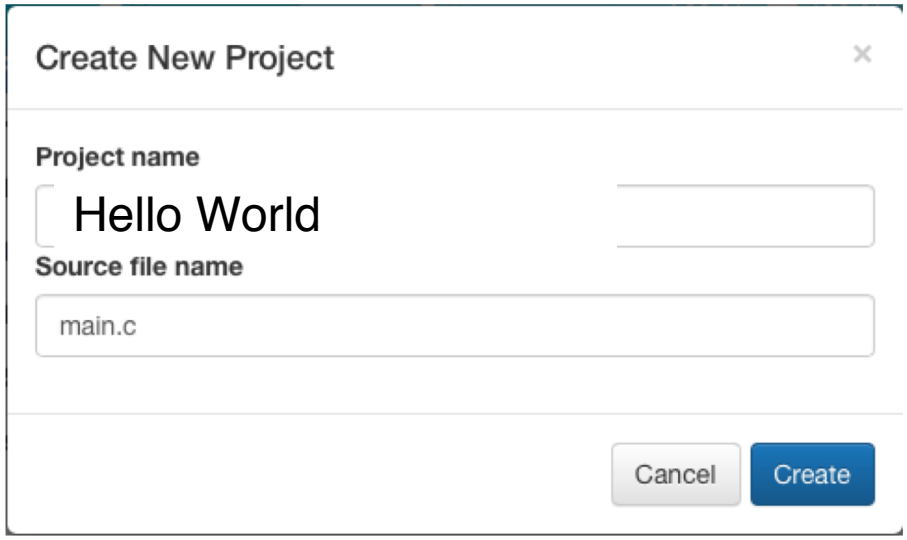


KISS INSTITUTE FOR PRACTICAL ROBOTICS

Name your project

Give your *project* a **descriptive name**

- **Do not** put any special characters or periods, etc. on it.



Click Create

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Lesson: Know Your Motor Ports

5.1 Activity
Objectives: Students will understand about motors and motor ports.

Materials: built robot without motors plugged into ports

Guiding questions:

- What body parts helps us move across the room?
- What will make our robot move?
  - 2 motors with wheels

Our brain tells each leg to move; in the same way the KIPR link must tell each motor to move.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

5.1 Check Your Robot's Motor Ports

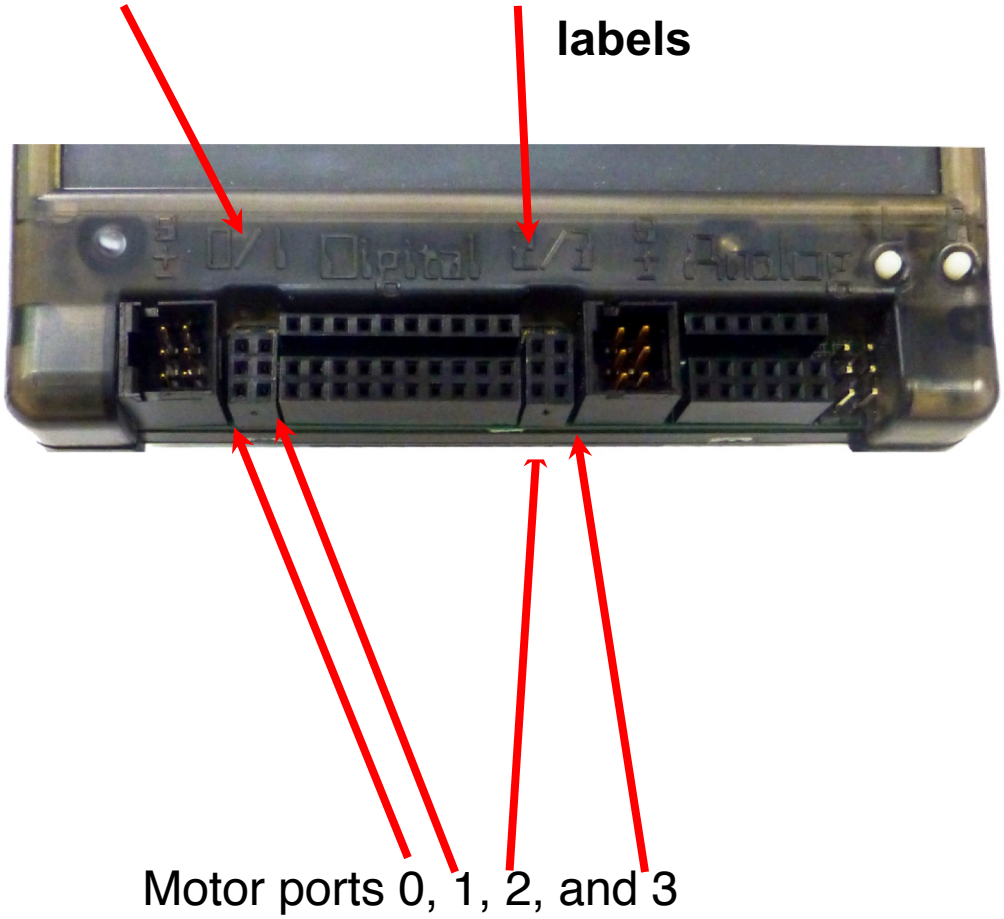
To program your robot, you need to know what motor ports your motors are plugged into (or need to be plugged into)

*** NOTE: Computer scientists start counting at 0 so the motor ports are numbered 0, 1, 2 and 3**

KISS INSTITUTE FOR PRACTICAL ROBOTICS

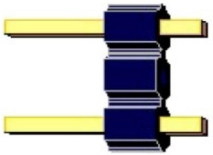
KIPR Wallaby Motor Ports

1. Find the 4 motor ports.
2. Locate labels above the motor ports



3. Find your motors. The motors have two-prongs on the plug end and have a double wire (2 wires)

4. Plug your motor plugs in port 0 and 3



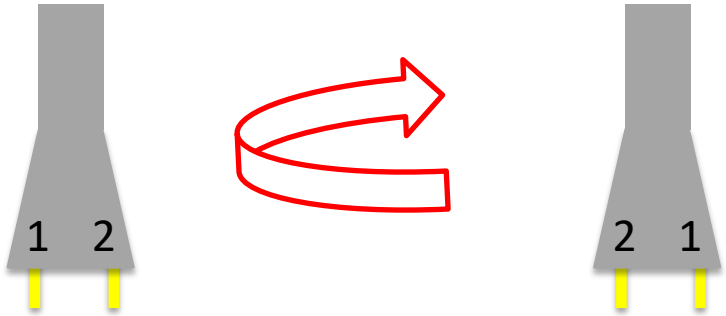
Drive motors have a 2 prong plug

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Motor Direction

Motors have 2 prongs on the plug

- There is no marking for left or right on the motor wire
- You can plug these in two different ways



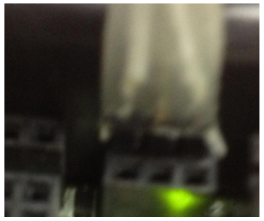
- Motors rotate in the direction that the electricity (electrons) move through them. One direction is clockwise motor rotation and the other direction is counterclockwise motor rotation

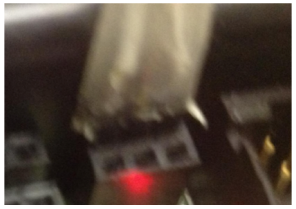
*You want your motors going in the same direction,
otherwise your robot will go in circles!

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Motor Port & Direction Check

- There is an easy way to check this!
  - Manually rotate the tire (turn it with your hand) and you will see a LED light up by the motor port (port # is labeled on the board)
    - If the LED is green it is going forward
    - If the LED is red it is going backwards





- Using the manual tire rotation trick, check the direction and port #'s of your motors
  - If one is red and the other green unplug and turn one motor plug 180° and plug it back in
  - The lights should both be green if the robot is moving forward

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Loading the Starting Web Page

1. Launch your web browser (such as Chrome or Firefox).
2. Copy this IP address into your browser's address bar followed by ":" and port number 8888; e.g.,

3. The KISS Software Suite will now come up in your browser.
4. You may use a computer, tablet or even a smart phone as long as they have a USB port.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Lesson: Using the motor()Function

Goal: Introduction to motor function

Activity:

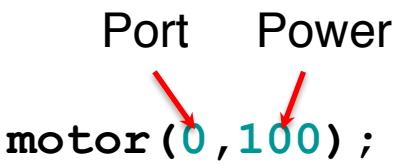
There are several functions for controlling motors, we will begin with motor().

Look at the program below for an explanation of how to use the motor function.

```
motor(0,100);//Turns on motor port 0 at 100% power.
    You can select any power level up to 100%.
motor(3,100); //Turns on motor port 3 at 100% power.
   You can select any power level up to 100%
msleep(2000);// time in milliseconds
ao(); //Turns off all motors
```

Questions:

1. What motor ports are identified in the program?
2. What motor ports are you plugged into?
3. How many different motor ports are available to plug into?
4. What do the motor functions using ports 0 at 100% power, and port 3 at 80% power look like?
5. Provide three examples of the motor function using different motor ports and powers.



KISS INSTITUTE FOR PRACTICAL ROBOTICS

Review

- The controller executes the program (code) from top to bottom and goes to the next line faster than a blink of your eye.
- At 800MHz the controller is executing ~800 Million lines of code/second!
- To control a robot you must give the **function** (command) *TIME* to run on the robot.
- To do this, we use the built-in function called msleep(); refer to learning C language for a review if needed.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Name your project
Review

Notes to help name your project:

- Give your project a unique and descriptive name
- Since there will be multiple students using the same robot, it is recommended to name the project their first name followed by the activity. Example **Sarah Hello World**
- Do not repeat the same name for more than one project. It will be hard to find the one you want!!
- DO NOT USE ANY PUNCTUATION SUCH AS "." (periods) IN YOUR FILE NAME.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Let's make the Robot Move!

- **Open Your Folder!** Create a new project in your folder called, **Drive**.
- On the template replace "Hello World" with the name of your new project, Move. (you can do this every time you have a new project).

Proceed to the next slide.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

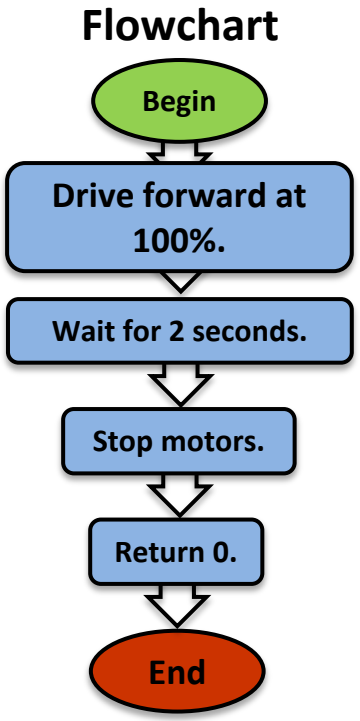
Let's make the Robot Move!

- **Description:** Write a program for the wallaby that drives the Robot forward at 100% power for two seconds, and then stops.

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

- **Pseudocode**

1. Drive forward at 100%.
2. Wait for 2 seconds.
3. Stop motors.
4. End the program.



KISS INSTITUTE FOR PRACTICAL ROBOTICS

Let's make the Robot Move!

1. Type your code, using the **motor(); msleep();** and **ao();** functions.
2. Compile your program. Did it succeed? If not, use your debugging sheet to correct your errors.
3. Run your program on your wallaby. Did it run as you expected?

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Solution

```
int main()
{
  printf("Sarah Robot Move!\n");

  motor(0,100);
  motor(3,100);
  msleep(2000);

  ao();
  return 0;
}
```

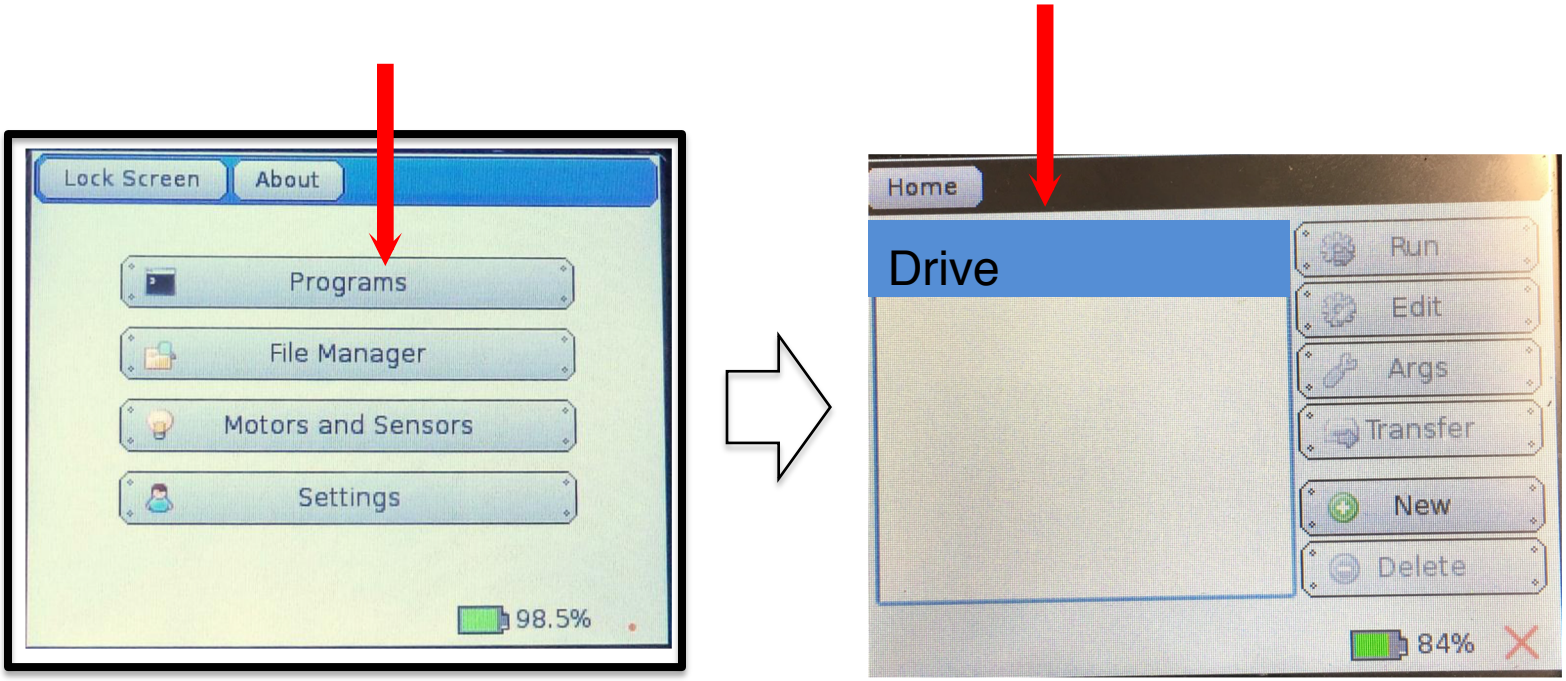
What value would you change to make the robot go a longer distance? Play with the value and observe the distance the robot travels.

Is the robot traveling straight?

KISS INSTITUTE FOR PRACTICAL ROBOTICS

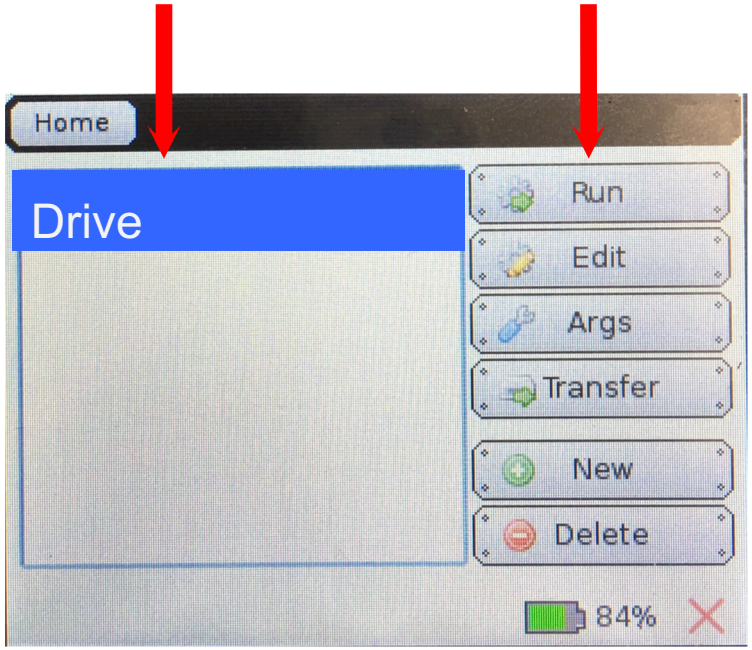
Running Your Program on the Wallaby

- Select the program button that will take you to a list of programs currently on the Wallaby.



Running Your Program on the Wallaby

1. Highlight the program you want to run, in this case, *"Drive"*.
2. Place your robot on the floor and then push the "Run" button.

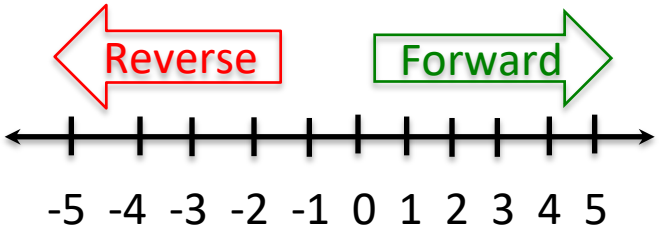


Did you robot go straight? Click on the racecar for help.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Robot Driving Hints

Remember your # line, positive numbers go forward and negative numbers go backwards.



Driving Straight - it is not easy to drive a robot in a straight line.

- Motors are not exactly the same
- The tires may not be aligned well
- One tire has more resistance, etc.

You can adjust this by slowing down and speeding up the motors.

Making Turns

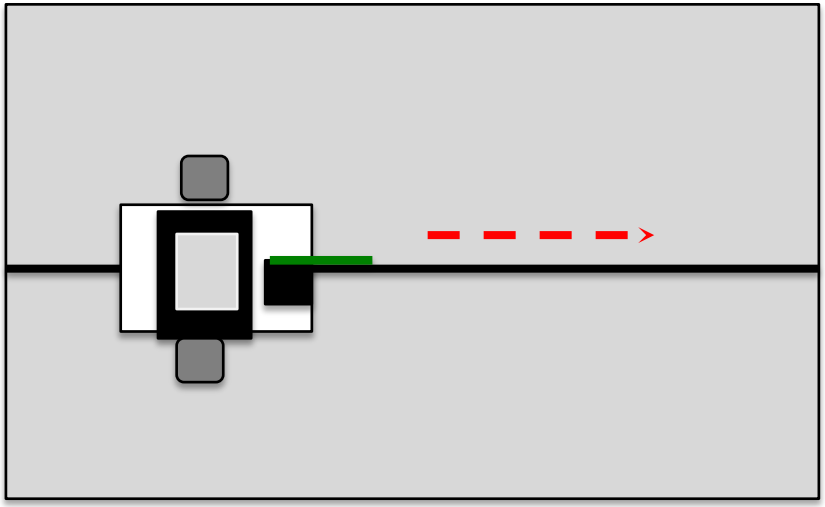
- Have one wheel go faster or slower than the other
- Have one wheel move while the other one is stopped (friction is less of a factor when both wheels are moving)
- Have one wheel move forward while the other is moving backwards

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Drive your robot straight

Use mat B and drive down the center line. Your robot needs to straddle the dotted line.

Adjust your wheel powers until straight.



KISS INSTITUTE FOR PRACTICAL ROBOTICS

Debugging Review

1. Leave off the terminating semicolon and see what happens.
2. Compile.
3. Go to the next slide to understand how to read errors.

```
int main()
{
  printf("Hello, World!\n");

  motor(0,100);
  motor(3,100);
  msleep(2000);

  return 0;
}
```

\n doesn't show up on the printed output it simply tells the display to print to a new line similar to the return key on a keyboard

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Debugging
Compile Failed "Debugging"

Example of an "Error Message" for missing a ";"

- Compile Failed will be the message at the bottom of the window
- ALWAYS IGNORE THE FIRST LINE

Compilation Failed

```
Compilation Failed

/home/root/Documents/KISS/Carol Folder/Carol Hello World/src/main.c: In function 'main':
/home/root/Documents/KISS/Carol Folder/Carol Hello World/src/main.c:6:4: error: expected ';' be
fore 'motor'
    motor (0,100);
    ^
```

1. **<u>Reading the Error</u>**- Ignore the first line, and look at the second line to find the error.
2. This error says that it expected to see a ";" before "motor"
3. Fix one error at a time and then recompile. It might fix all the errors.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Debugging

Compile Failed "Debugging"

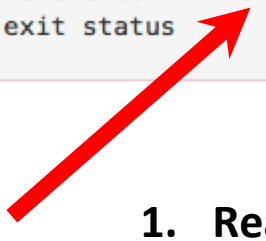
Spell **`msleep`** wrong. Compile and read the error. What does it say?

Example of an "Error Message" for misspelling "msleep"

- Compile Failed will be the message at the bottom of the window

```
/home/root/Documents/KISS/Carol Folder/Carol Hello World/src/main.c:8:4: warning: implicit decl
aration of function 'msleeep' [-Wimplicit-function-declaration]
    msleeep (3000);
    ^
/tmp/cctKBCzM.o: In function `main':
main.c:(.text+0x30): undefined reference to `msleeep'
collect2: error: ld returned 1 exit status
```

undefined reference is always a spelling error.



1. **Reading the Error**- in this case go to the bottom of the errors.
2. This error says undefined reference to "msleep". Spelled wrong.
3. Fix one error at a time and then recompile. It might fix all the errors.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

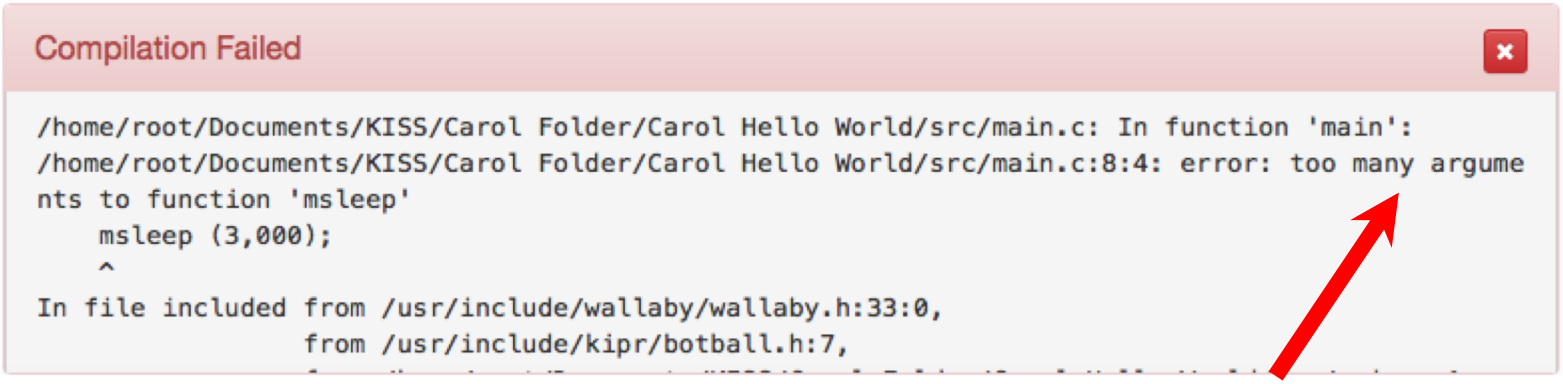
Debugging

Compile Failed "Debugging"

Put a comma in the **`msleep(2,000)`**. Compile and read the error. What does it say?

Example of an "Error Message" for inserting a comma in the time for milliseconds.

- Compile Failed will be the message at the bottom of the window



1. **<u>Reading the Error</u>**- in this case the error is on line 8, "too many argument".
2. msleep has only one argument of time. The comma indicates that there is two argument.
3. Fix one error at a time and then recompile. It might fix all the errors.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

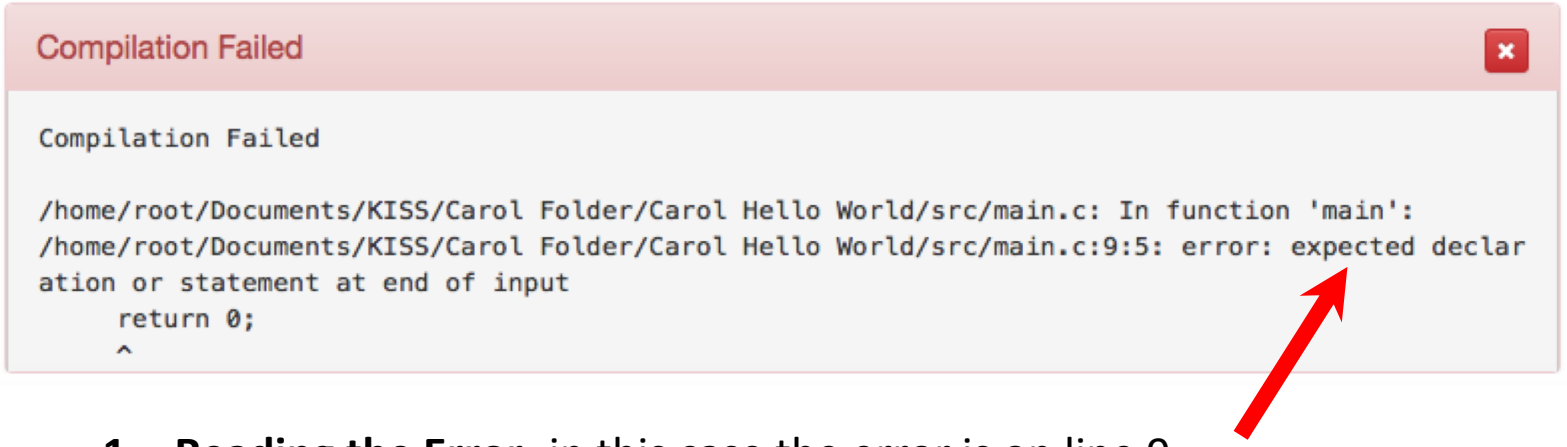
Debugging

Compile Failed “Debugging”

Remove a curly brace, compile, and read the error. What does it say?

Example of an “Error Message”for leaving off a “}”

- Compile Failed will be the message at the bottom of the window



1. **Reading the Error**- in this case the error is on line 9, expected declaration or statement at end of input.
2. Missing a “}” after return 0.
3. Fix one error at a time and then recompile. It might fix all the errors.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Debugging

Compile Failed "Debugging"

Replace a 0 with the letter o, compile, and read the error. What does it say?

Example of an "Error Message"for replace "o" for "0"

- Compile Failed will be the message at the bottom of the window

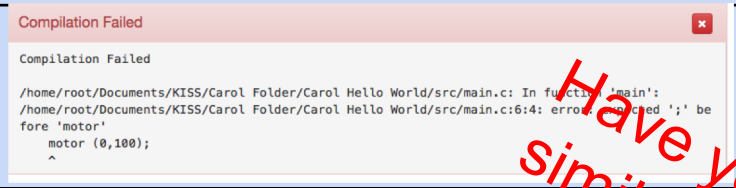
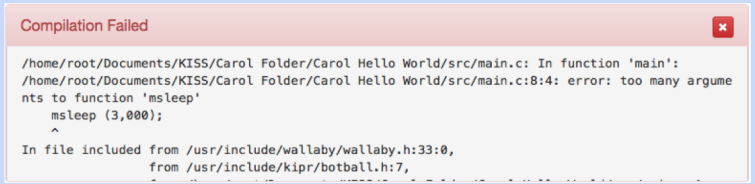
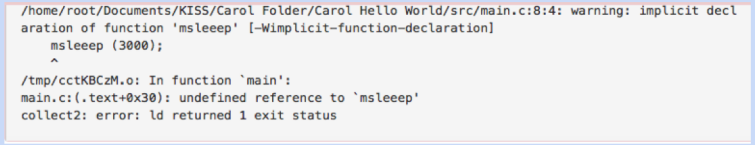
```
Compilation Failed

/home/root/Documents/KISS/Carol Folder/Carol Hello World/src/main.c: In function 'main':
/home/root/Documents/KISS/Carol Folder/Carol Hello World/src/main.c:9:12: error: 'o' undeclared
(first use in this function)
     return o;
            ^
```

1. **<u>Reading the Error</u>**- in this case the error is on line 9, 'o' undeclared (first use in this function)
2. Used a "o" instead of an 0.
3. Fix one error at a time and then recompile. It might fix all the errors.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Common Debugging Errors

<u>Reading the Error-</u> This error says that it expected to see a “;” before line 6. Therefore, line 6 is where the error is.

Error	Reason
	• missing a “;”
	• spelled msleep wrong
	• missing the “int main”
	• `motor(0100);` -is a mistake because it needs a comma to indicate the two arguments of port, and power.
	• `msleep(2,000);`-is a mistake because the comma indicates it has two arguments. This function has only one argument of milliseconds without a comma.
	• a “}” is missing at the end of the program. Can also indicate the “{“ is in the wrong direction if it is there.

Have your students create a similar table or chart in their notebooks.

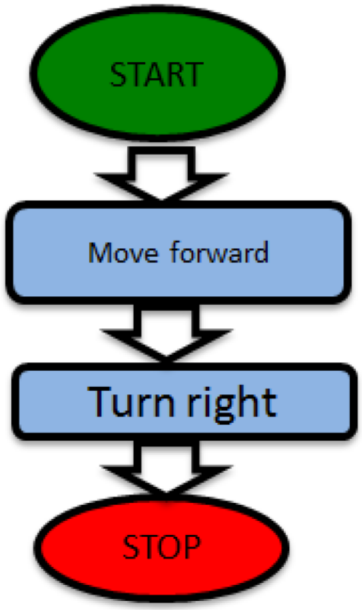
Reviewing Comments (Pseudocode)

1. Read and discuss with a partner this slide and the next slide to understand Pseudocode.

Pseudocode means "false code".

Pseudocode can be used to comment on what you expect your program to cause your robot to do, but that might not be what it will actually do.

```
//1. Move forward
//2. Turn Right
//3. Stop
```



- Why would it be important to create pseudocode?

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Learning About Comments (Pseudocode)

- Comments as pseudocode help you keep track of what is going on in the program.
- To make a comments in your source code two slash marks must be typed first: //

  Complete Comment: //forward

- When you compile the program it will not execute the comment, but you can see it.

Sample Program

```
int main()
{
  printf("Hello, World!\n");

      motor(0,100); // forward
      motor(3,100);
      msleep(2000);

  return 0;
}
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Add a Comment

1. Add the //comments to your program
   - Just like using a word processing program you can click to set your cursor and then use return to make space for the comment.
   - Type the comment into your program. The comment can go on the line before the **printf function** or on the same line as the function.

2. Comments in the program help you keep track of what you are doing. The robot sees the green and is instructed to go to the next line.

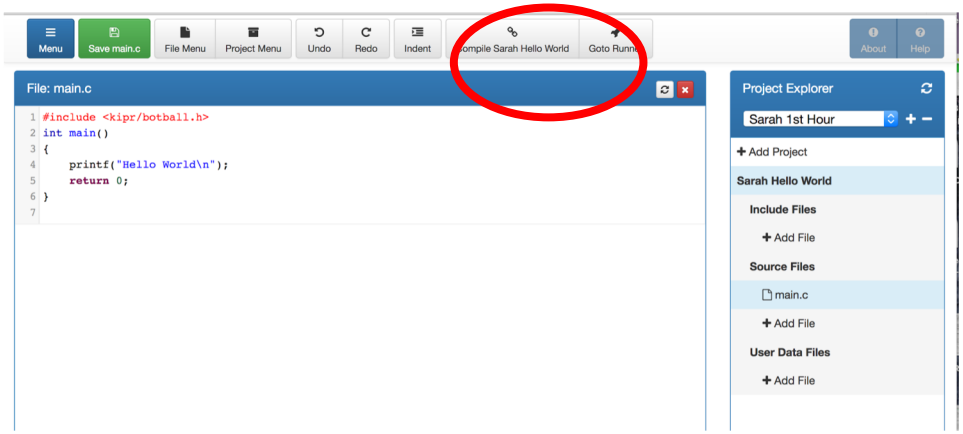
On same line as the function

```
int main()
{
    printf("Hello, World!\n"); //Print Hello World to screen

        motor(0,100); // forward
        motor(3,100);
        msleep(2000);

    return 0;
}
```

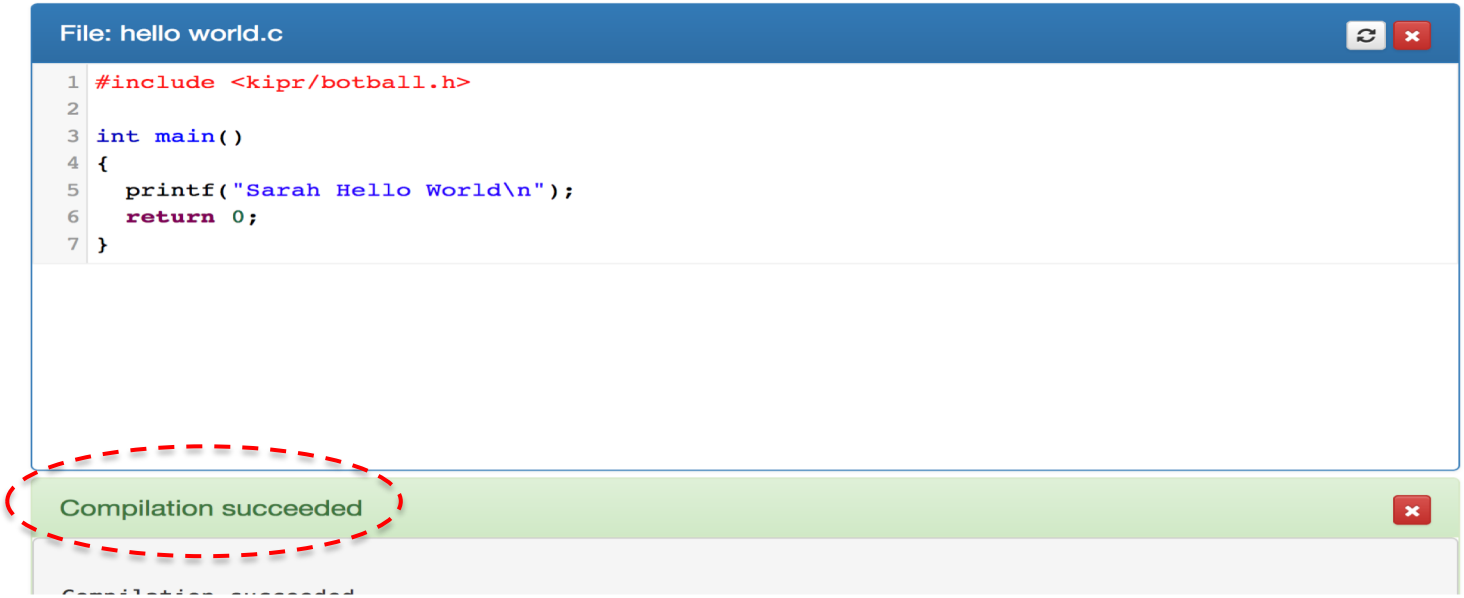
KISS INSTITUTE FOR PRACTICAL ROBOTICS

Add a Comment (cont.)

Compile



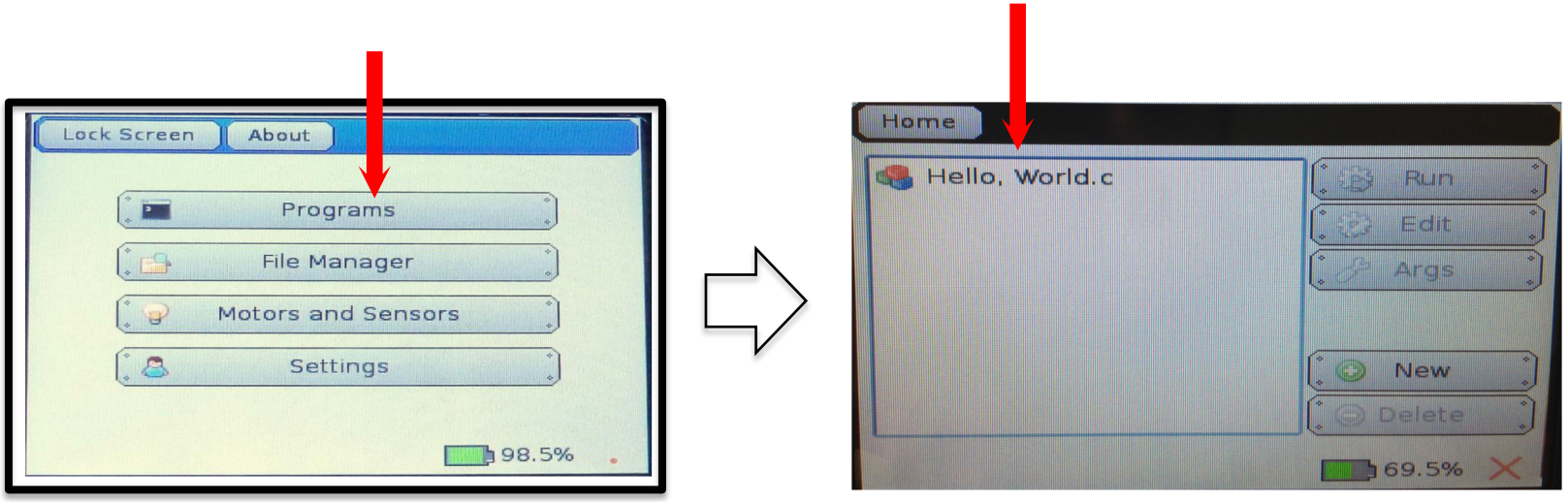
Watch to see if it Succeeded!



Go to your robot and run the program. Did it show? NO!

KISS INSTITUTE FOR PRACTICAL ROBOTICS

1. Press (touch) the **“Programs” button** on the **KIPR Wallaby**.
2. This will take you to a **list of programs** currently on your Wallaby.



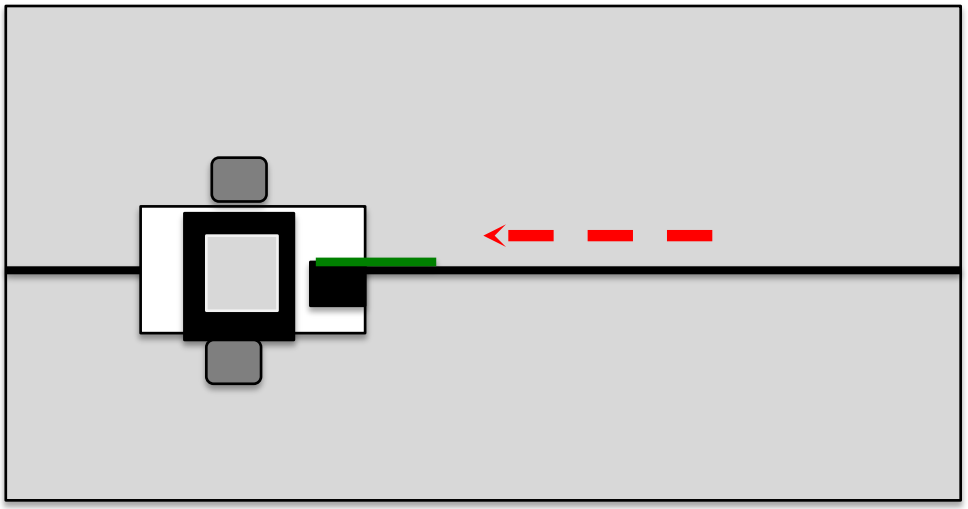
Did your comments show on the screen?

Drive your robot backwards and straight

Use mat B

Goal: drive backwards down the center line. Your robot needs to straddle the dotted line.

Adjust your wheel powers until straight.



KISS INSTITUTE FOR PRACTICAL ROBOTICS

Learning to Move Backwards

1. If positive power moves the robot forward, what do you have to do to make the robot move backwards?

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

Power can be between -100 & 100

Have the students use the: Think Pair Share Strategy

https://k20center.ou.edu/instructional-strategies/

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Backwards

```
motor(0,-100);
motor(3,-100);
msleep(2000);
```

Negative numbers in the power argument will rotate the wheels backwards

KISS INSTITUTE FOR PRACTICAL ROBOTICS

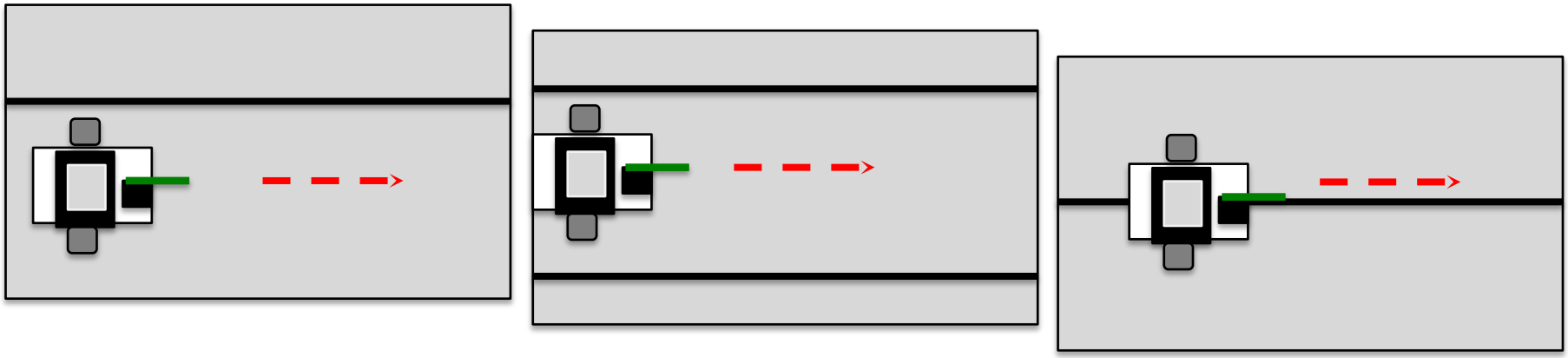
Don’t Touch the Line

1. Open a new project, name it, “Do not touch the line”.

   Robot must move without touching the line (easiest to hardest below)

   - Using one line have the robot move down the side without touching it
     - Make this a time trial-quickest time without touching (faster is harder to control)
   - Use a lane and have the robot drive down it without touching either side.

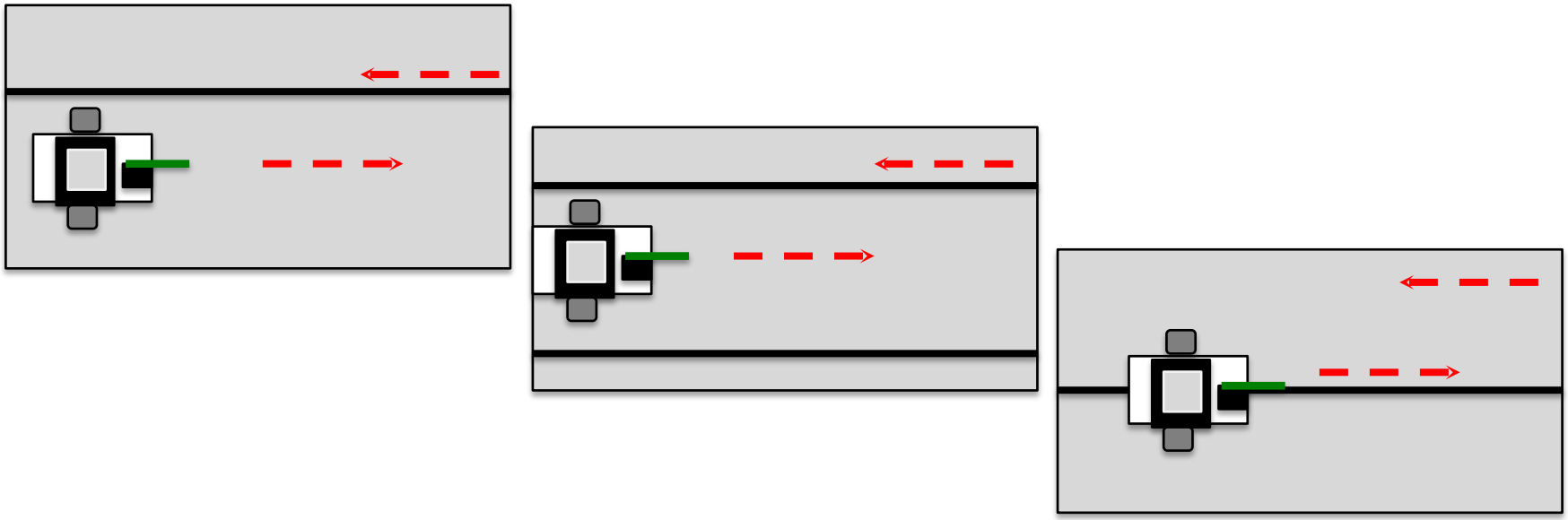
Variations: Increase difficulty by making the lane narrower



KISS INSTITUTE FOR PRACTICAL ROBOTICS

Don’t Touch the Line Variations

Similar except have the robot stop and backup without touching the line when traveling in either direction. Add a starting line to begin and a finish line the robot must touch before backing up.



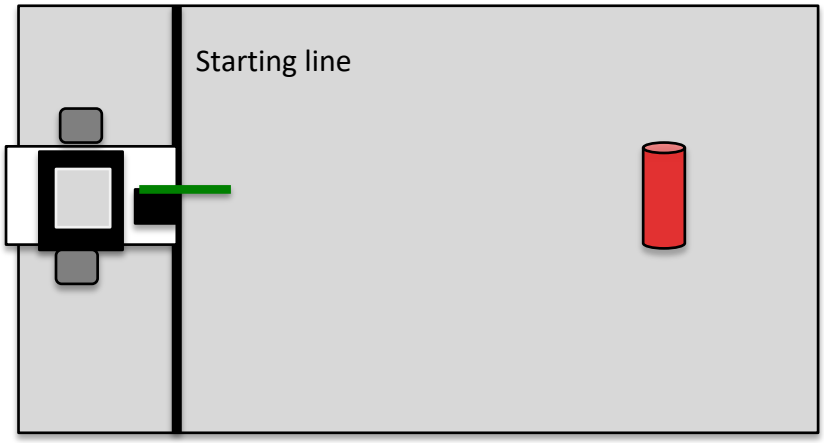
KISS INSTITUTE FOR PRACTICAL ROBOTICS

Touch the Can on Circle 9

Open a new project, name it, “Touch the Can”.

The robot must:

1. Go out and touch the can in circle 6.
2. Stop/pause for 2 seconds.
3. Go in reverse to start line.



KISS INSTITUTE FOR PRACTICAL ROBOTICS

Sample Coding

```
int main ()
{
    printf("Sarah Backwards");

    motor(0,100); // forward
    motor(3,100);
    msleep(3000);

    motor(0,-100); // back
    motor(3,-100);
    msleep(3000);
    return 0;
}
```

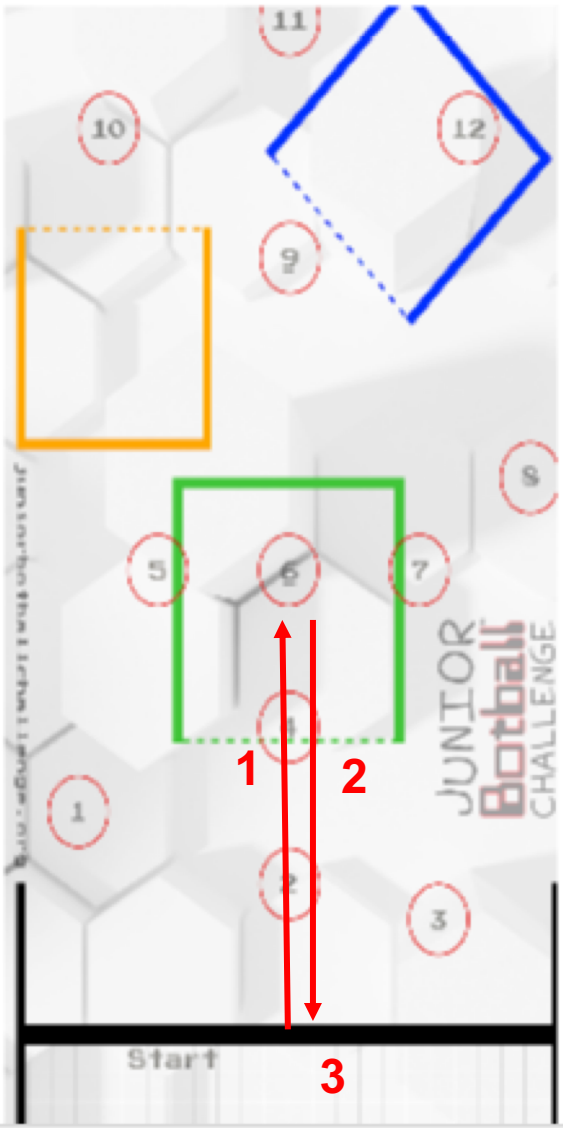
KISS INSTITUTE FOR PRACTICAL ROBOTICS

Sample for Learning to Drive 3

Name/Team_____Challenge/Activity_____ Name, drive 3

Goal:

Touch the can on 6 and return to finish line



Pseudocode/Steps

1. move forward to can 6
2. move backwards until you cross starting line
3. Shut motors down
4.
5.
6.
7.
8.
9.
10.
11.

Notice that the drawing steps match the steps.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Assessment

Assessment 1: Tag – You're It

Setup: Use Surface-A. Place a 12oz empty soda can in circle 9.

Desired Outcome: The robot will drive to the can in circle 9, touch it, and return to the starting area.

Limitations:

1. All robots must be autonomous (no remote control, 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. Must be able to tell that the can was touched by the robot, either visually (the can moved) or audibly (the robot touching the can made a noise).
4. The can must not tip over and some part of the can must remain in the circle, or that team does not complete that run.

Completion: When the robot touches the can and returns behind the starting line.

Extra Optimization: Students try to get lowest time to go out, touch the can and return.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Learning to Turn
by Changing Motor Power Only

1. **Open Your Folder!** Create a new project in your folder called **"Turns"**.
2. On the template replace "Hello World" with the name of your new project, Turns. (you can do this every time you have a new project).
3. Decrease the power on motor port 3 to 70%, compile and run. Make observations.

```
motor (0,100);
motor (3,70);
msleep (3000);
```

change power on this motor

2. Decrease the power on motor port 3 to 10%, compile and run. Make observations. Keep the msleep constant (3000).

4. Repeat above steps using the following power in motor port 3;

- 0%, 20%, 55%, -50%, experiment with different powers.

5. Make observations.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Learning to Turn by Changing msleep Only

1. Decrease the power on motor port 3 to 50%, compile and run. Make observations.

```
motor (0,100);
motor (3,50);
msleep (3000);
```

change msleep time

2. Increase the `msleep` time to 4500. Make observations
3. Decrease the `msleep` time to 1000. Make observations.
4. Increase the `msleep` time to 9000. Make observations.
5. Decrease the `msleep` time to 500. Make observations.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Discoveries on Making Turns

- Did you discover that you can decrease the power on one motor only and it will make the robot turn in the direction of the lower power?

- The greater the **difference** between the two motor powers the sharper the turn.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Activity to Make Circles
Changing the Power and msleep

1. **Open Your Folder!** Create a new project in your folder called "**Circles**".

1. On the template replace "Hello World" with the name of your new project, "Name" Circles. (you can do this every time you have a new project).

1. Place a chair in the middle of the room. Type and compile a program that will move the robot in a large circle around the chair. You can not touch the chair.
   a. You should be able to create the program with only one set of **motor()**, **msleep()**, and **ao()** functions.

2. Place a small object the size of a kleenex box in the middle of the room. Type, compile and run a program that will move the robot in a tight circle around the object.
   a. You should be able to create the program with only one set of **motor()**, **msleep()**, and **ao()** functions.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Learning to Make Right Angle (90º) Turns

1. **Open Your Folder!** Create a new project in your folder called **right angle**.

On the template replace "Hello World" with the name of your new project, Name 90. (you can do this every time you have a new project).

Discover how to write a program to make a right angle turn, compile and run on your robot.

2. Use the JBC Code Planning notebook paper to pre-think how to make:
   - a big rectangle
   - a smaller rectangle
   - squares of different sizes

- When you run your program a good hint is to

*test one step at a time and get it correct before moving onto the next step

This is not an easy task!

KISS INSTITUTE FOR PRACTICAL ROBOTICS

More Activities

1. **Open Your Folder!** Create a new project in your folder called **Fun.**

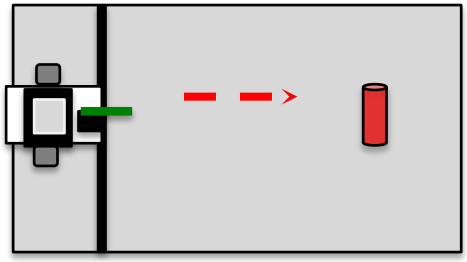
1. Follow the directions below and compile and run your program.

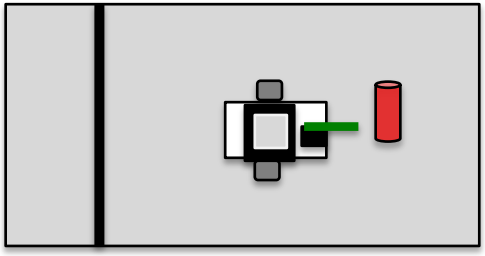
Closest to the Can: Robots must start behind the starting mark and move toward the object with the goal of stopping as close to the can as possible without touching it.

- If they touch the can they must start over at the starting line
- Use rulers to measure the distance stopped from the can- make a data table
- You can use a sheet of paper passed between the robot and can to determine if it is touching
- You can limit the number of attempts and take the best run, have them average several runs or add the distances together for a grand total

Variation:

- Move the can to various distances and locations





KISS INSTITUTE FOR PRACTICAL ROBOTICS

More Learning to Drive Activities

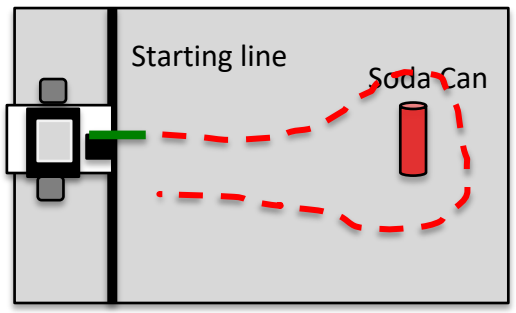
1. Follow the directions below and compile and run your program on your robot.

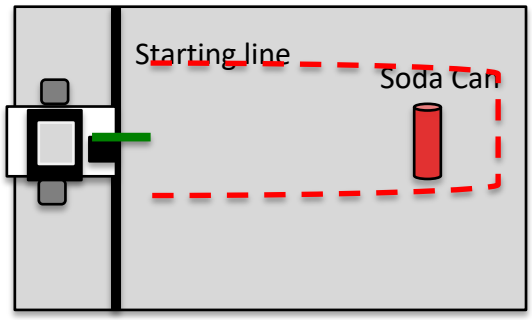
Place a can on circle #6. The robot must go around the can without tipping it over or moving it completely out of the circle. Once the two driving wheels are across the finish line they have completed the activity (remember to comment your code).

Variations:

- Move the can to different circles (4,2,9).
- Make your robot go clockwise and then counter clockwise around the can
- Math- measure the distance to the object and time the robot which you can use to calculate rate/speed, (Speed = Distance/Time).

*Drawing the robot's path in your notebook and doing a step by step analysis will help you complete this task. There is more than one way to get around the can. Use your JBC Code Planning Notebook paper.





KISS INSTITUTE FOR PRACTICAL ROBOTICS

Learning to Drive!
Activity/Mini Contests

Now, it is time to practice using the motor();,msleep(); and ao();functions.

These activities can all be completed using hard coding (“dead reckoning”) and simple motor control functions without the use of any sensors.

Your teacher will decide which activities and assessments you will need to complete.

A great tool to help you with writing pseudocode is the JBC Code Planning Notebook paper.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

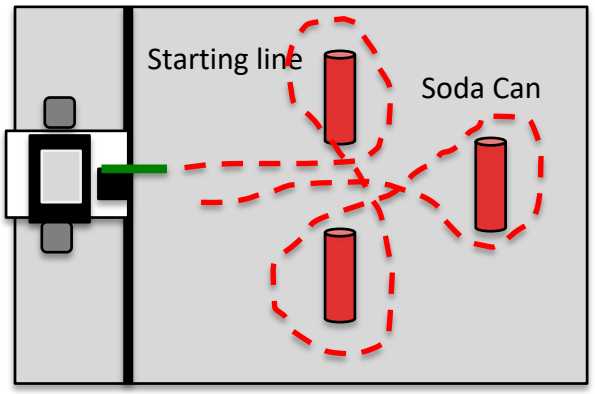
More Learning to Drive Assessment

Variations on Circle the Can:

Activity: The robot must go around the cans without tipping them over or moving them completely out of the circles. Once the two driving wheels are back across the starting line they have completed the activity (remember to comment your code).

2. Barrel Race (have them go around three cans)

*Drawing the robot's path in your notebook and doing a step by step analysis will help you complete this task. There is more than one way to get around the can. Use your JBC Code Planning Notebook paper.



KISS INSTITUTE FOR PRACTICAL ROBOTICS

Assessment

Assessment 2: Ring Around the Can

Setup: Use Surface-A. Place a 12oz empty soda can in circle 6.

Desired Outcome: The robot will drive out and turn around the can in circle 6, and return to the starting area.

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 entire robot must go around the far side of the can.
4. The can must not tip over and some part of the can must remain in the circle, or that team does not complete that run.

Completion: When the robot drives around the can and returns behind the starting line.

Extra Optimization: Students try to get the lowest time to drive out and around the can and return to the starting area.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Parking in a Garage
Activity 6

1. Open a new project, name it, "*Park in the Garage*".

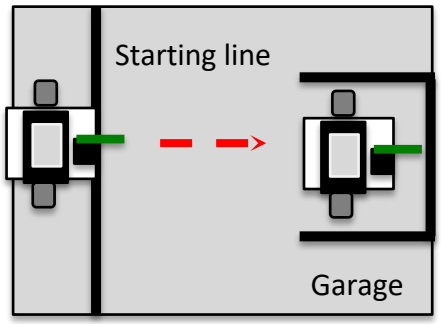
Goal: Robots must start behind the starting line and park in the blue garage (remember to comment your code). Task is complete when the driving wheels are in the garage and are not touching any of the garage walls.

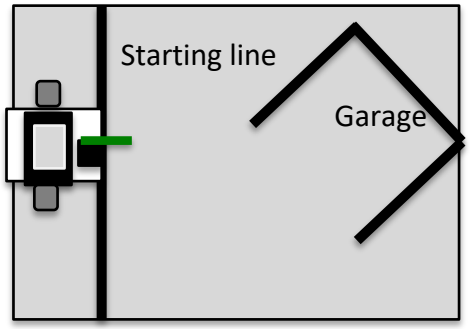
Rules: The robot may not cross over or touch any of the solid lines of the garage they are parking in.

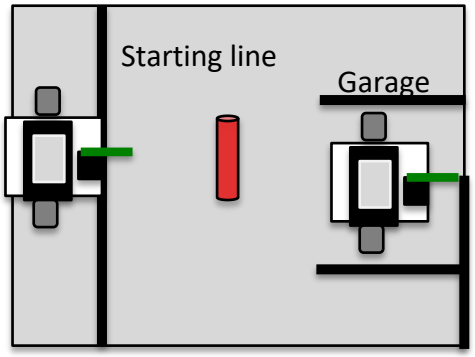
Variations:

- Park in the orange or green garage
- Don't hit the bike: Place a coke can between the start line and the garage. The robot can't hit the can.

*Drawing the robot's path in your notebook and doing a step by step analysis will help you complete this task. There is more than one way to turn into a garage.







KISS INSTITUTE FOR PRACTICAL ROBOTICS

Assessment

Assessment 3: Precision Parking

Setup: Use Surface-A.

Goal: The robot will drive into the smallest colored box (or garage) without touching the solid lines marking the 3 sides of the garage and stop.

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 team must declare which garage they intend to park in before starting a run.
4. The robot may not touch the solid lines marking the 3 sides of the garage the team intends to enter.
5. A robot may pass over (but not touch) the vertical projection of solid line of the selected garage.
6. A robot may drive over the dotted line of the selected garage.
7. All lines from undeclared garages will be ignored.

Completion: When the robot successfully parks in all of the garages.

Extra Optimization: Students try to get the lowest/fastest time to park in all three garages.

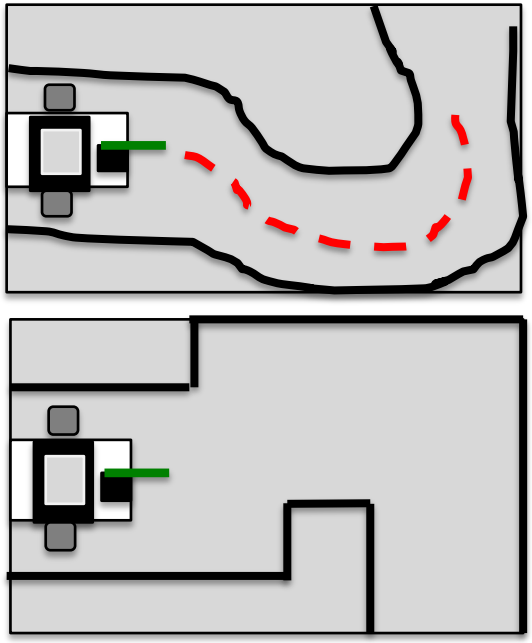
KISS INSTITUTE FOR PRACTICAL ROBOTICS

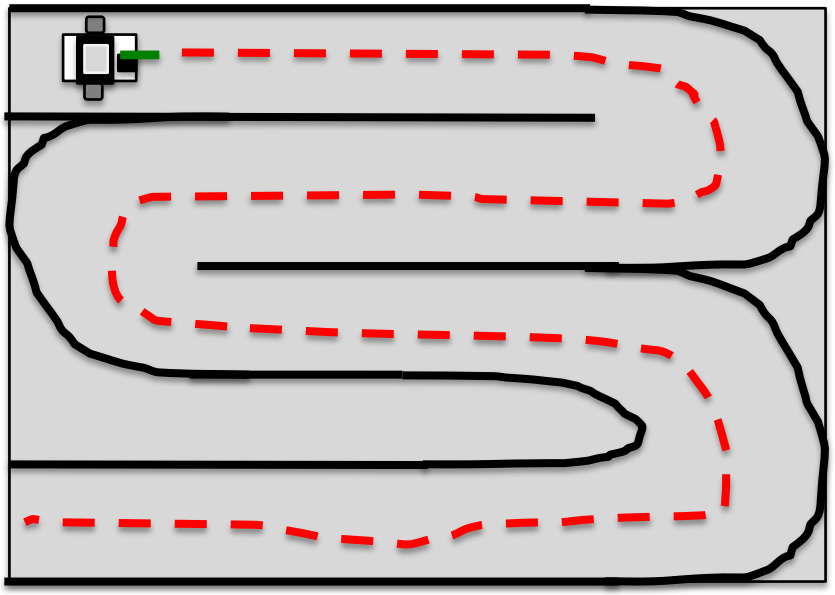
Race Track

Robot must move within the lane completing the course

- Make this a time trial, the fastest to complete the course with no errors
  - If you touch the line, you have to start over and the clock keeps running
- Use a much larger track if desired (taped lanes on the classroom floor work well)
- Use different lane setups
  - The tighter and more numerous the turns the more difficult it is

Extra Optimization: Once finished, make them stop and back up all the way to the start.





KISS INSTITUTE FOR PRACTICAL ROBOTICS

Assessment

Assessment 4: Figure Eight

Setup: Use Surface-A. Place 2 empty 12oz soda cans in circles 4 and 9.

Desired Outcome: The robot will weave in and out of the cans slalom style, around the last can and weave back through the cans to the starting line.

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 entire robot must go around the far side of can 9.
4. The cans must not tip over and some part of each can must remain in the circle, or that team does not complete that run.

Completion: When the robot drives through all the cans and returns behind the starting line.

Extra Optimization: Students try to get the lowest/fastest time to slalom out, around can 9 and slalom back across the start line.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Assessment

Assessment 8: Serpentine

Setup: Use Surface-A.

Desired Outcome: The robot will drive on the surface touching each of the numbered red circles with at least one of the robot's wheels in sequential order (1, 2, 3, etc.).

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 touch each circle with at least one drive wheel in the correct order.

Completion: When the robot drives through (touches with at least one drive wheel) circles 1-8 in the correct order in one run.

Extra Optimization: Students make it all the way to circle 12 in sequential order in one run.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Assessments and Rubrics

Suggestions: *Understanding*, *Group Collaboration*, and *Challenge Assessment* rubrics

KISS INSTITUTE FOR PRACTICAL ROBOTICS