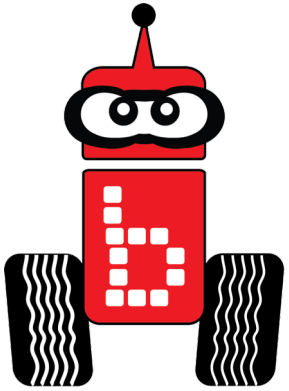
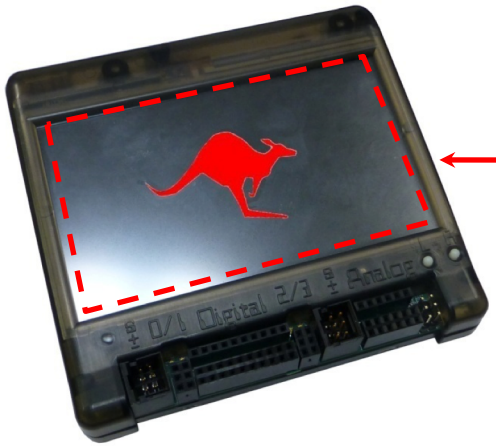
KISS INSTITUTE FOR PRACTICAL ROBOTICS



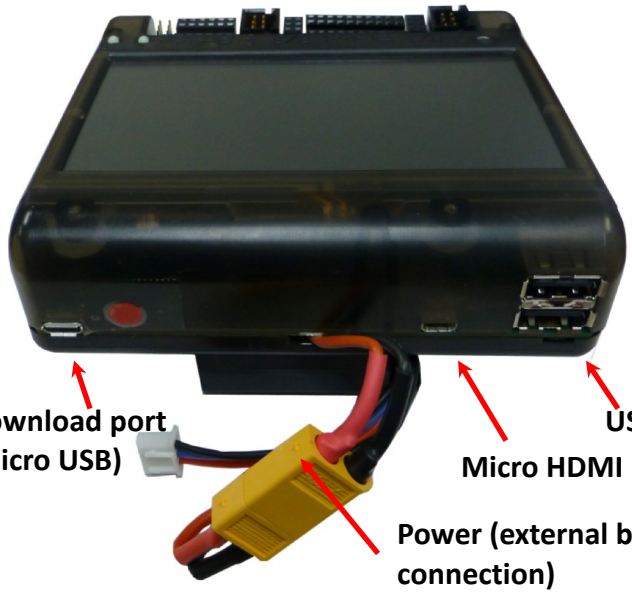
Quick Start

This is a quick start to moving the robot. It will get your students interested and ready to go.

KIPR Wallaby Controller Guide



Color Touch Screen

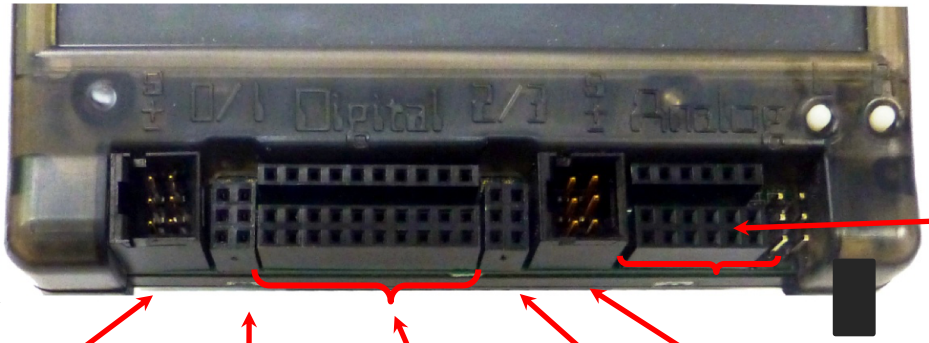


Download port (micro USB)

USB

Micro HDMI

Power (external battery connection)



6 Analog Sensor Ports (Port # 0 - 5)

2 Servo Motor Ports (Port # 0 & 1)

2 Motor Ports (Port # 0 & 1)

10 Digital Sensor Ports (Port # 0 - 9)

2 Motor Ports (Port # 2 & 3)

2 Servo Motor Ports (Port # 2 & 3)

Power Switch

Charging the Controller's Battery

- For charging the controller's battery, use only the power supply which came with your controller.
  - It is possible to damage to the battery from using the wrong charger or from too deep a discharge!

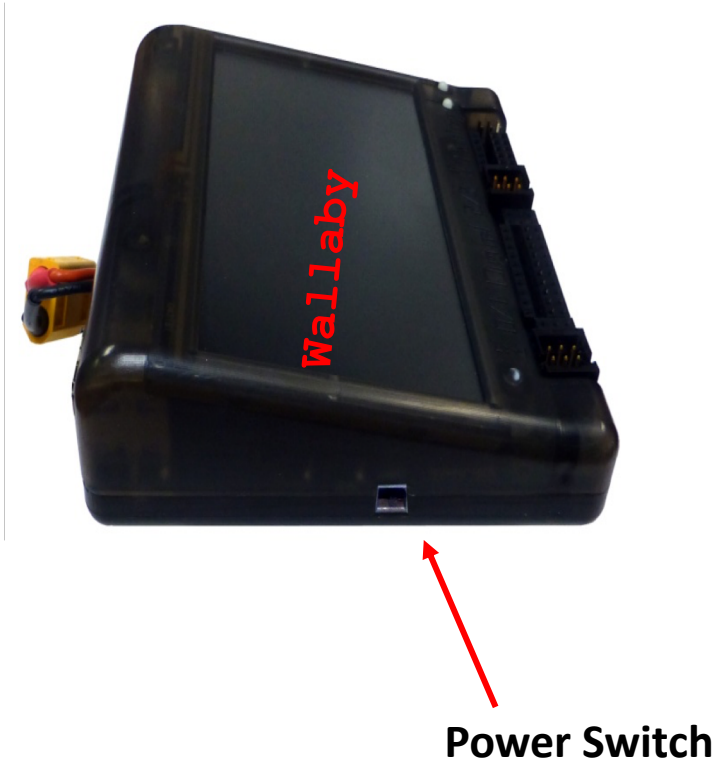
- The standard power pack is a lithium iron phosphate (LiFe) battery, a safer alternative to lithium polymer batteries. The safety rules applicable for recharging any battery still apply:
  - Do NOT leave the unattended while charging.
  - Turn the Wallaby off or unplug it from the battery while charging
  - Charge in a cool, open area away from flammable materials.

Important Information

All connections are as follows:

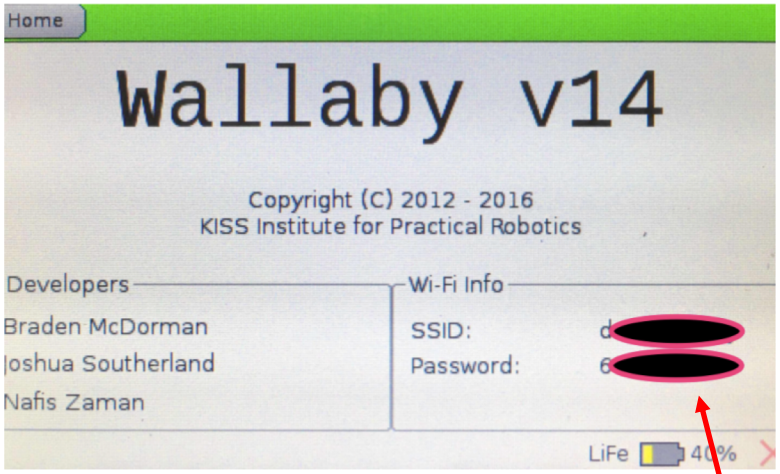
- Yellow to Yellow
- White small to White small (to the charger, may be white to black)
- Black to Black

Turn on your wallaby



Turn on the Wallaby with the **black switch on the side**. Wait 5 minutes before it's ready to use.

Connect to Your Wallaby by Wifi



1. Click **about** on your controller.
2. Every wallaby has a **unique number/name** that will appear in your Wi-Fi list.
3. Look at the Wifi information and find the **SSID**: that is your wallaby number and wallaby **password.**
4. Connect the **Wallaby** to your device via Wi-Fi (go to your Wifi settings on your device).
5. Click on **your** wallaby number, and add your password. . You will see a blue LED

   Light that indicates that you waited 5 minutes and you will hear a voice say "Welcome".

   Your password will may not work if you have not waited the 5 minutes.

- **Wifi not recommended for Challenge Day or Tournament**

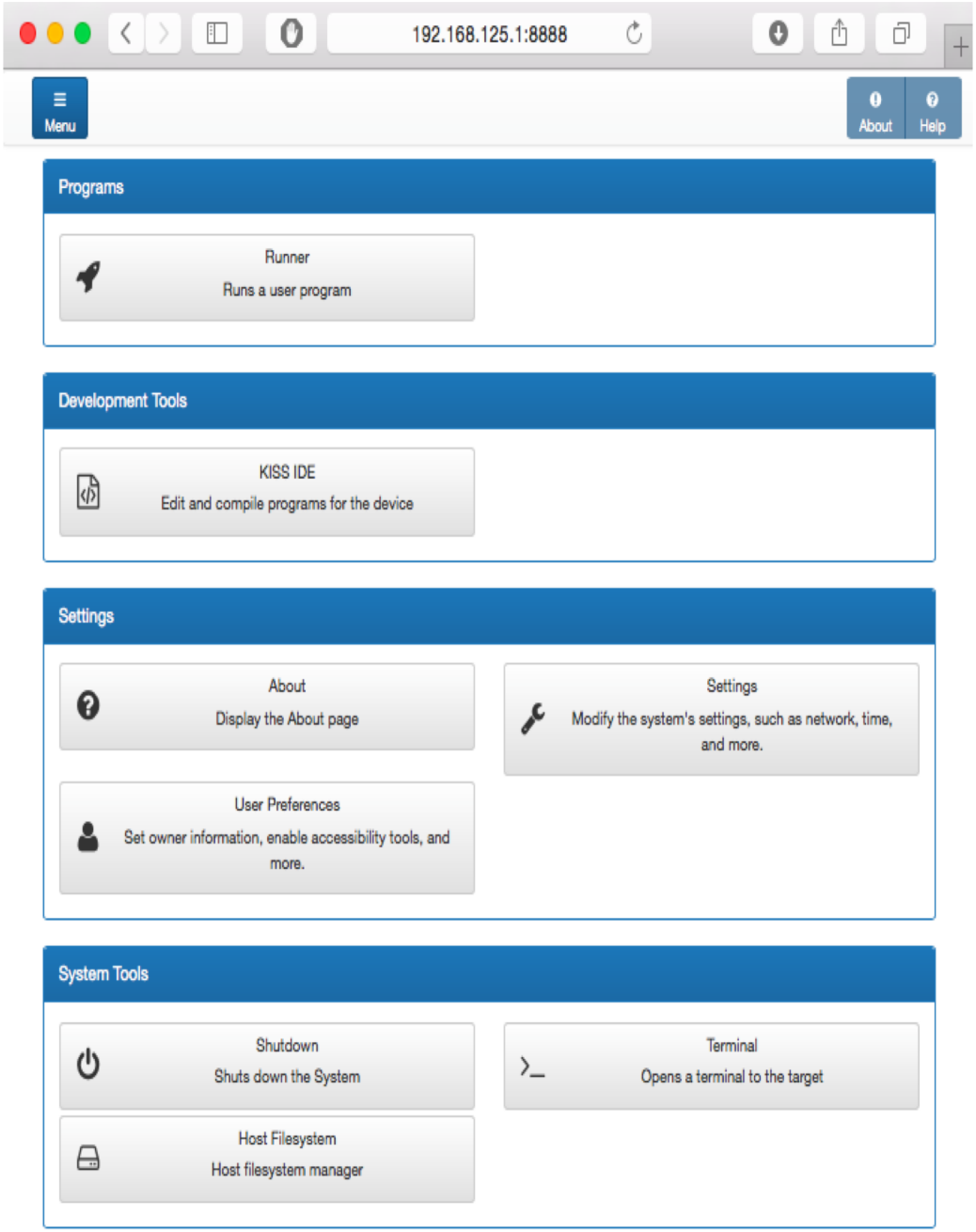
Accessing KIPR Software Suite (using Wi-Fi)

1. Launch your web browser (such as Chrome or Firefox).
2. Copy this IP address into your browser's address:

192.168.125.1:8888

IP address Port #

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



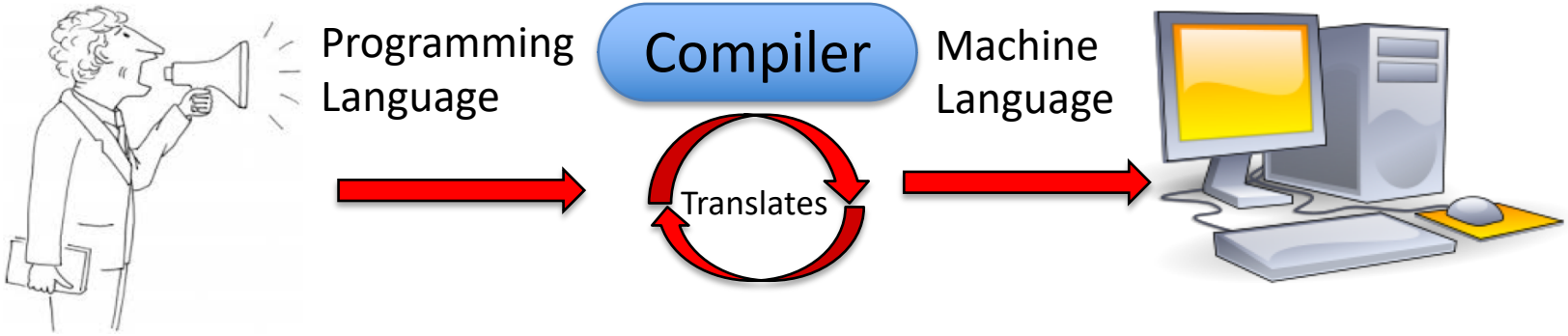
The **KIPR Software Suite** will appear.

Why Do We Need the KIPR Software Suite?

Computers only understand machine language (stream of bytes), which they can then read and execute (run).

Humans on the other hand, don't do well with machine language.

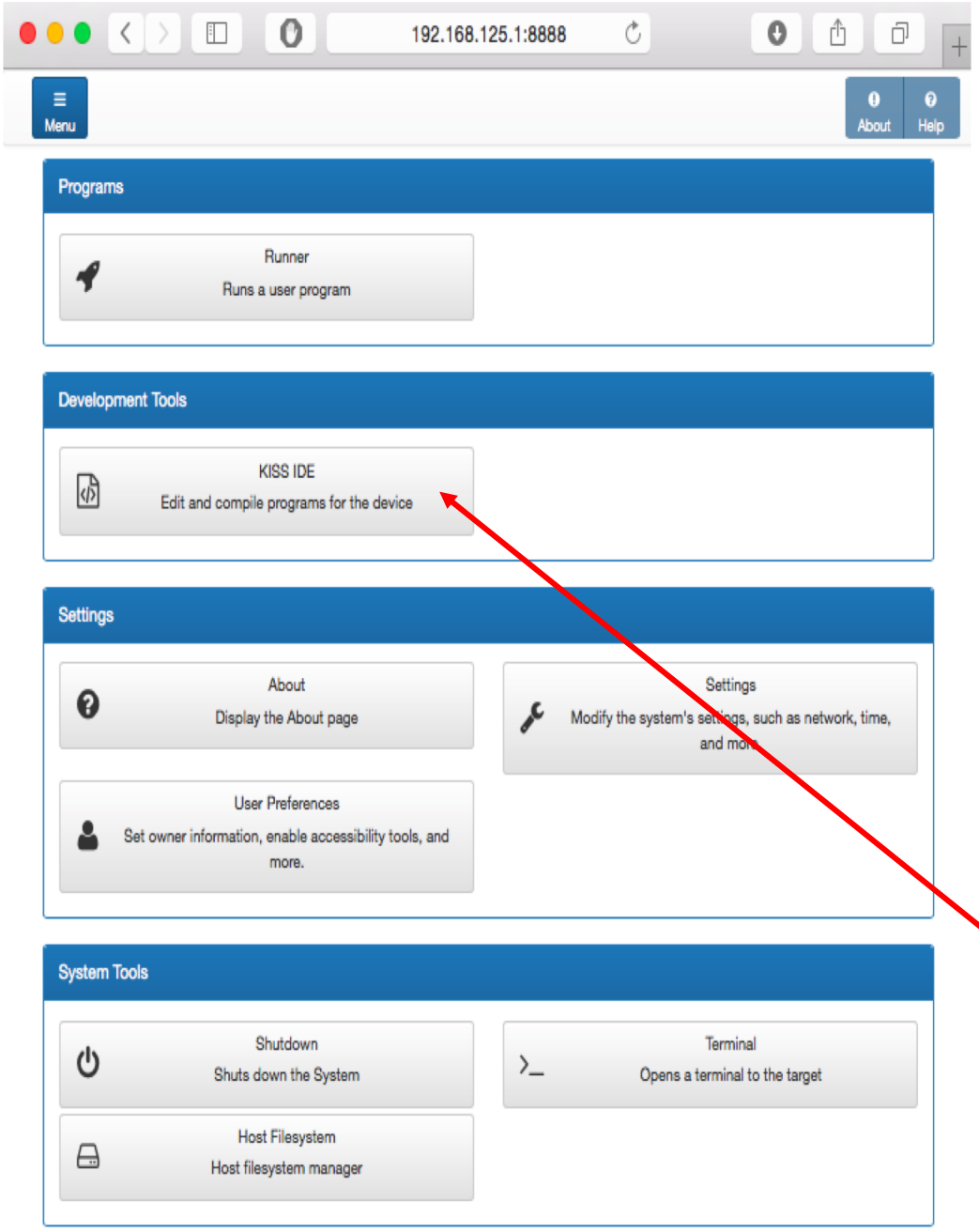
Humans Need an Interpreter



Humans have created languages with funny names like; C, C++, JAVA and Python, that allow them to write “source code” which they can understand and edit.
(Examples of common computer programming languages and their uses.)

This source code is then compiled (translated) into machine language that the computer can understand and

execute (run).



This is the **KIPR Software Suite** is a software application that makes it easy for you to write and edit your source code, debug it (look for and correct mistakes) and compile (translate) it.

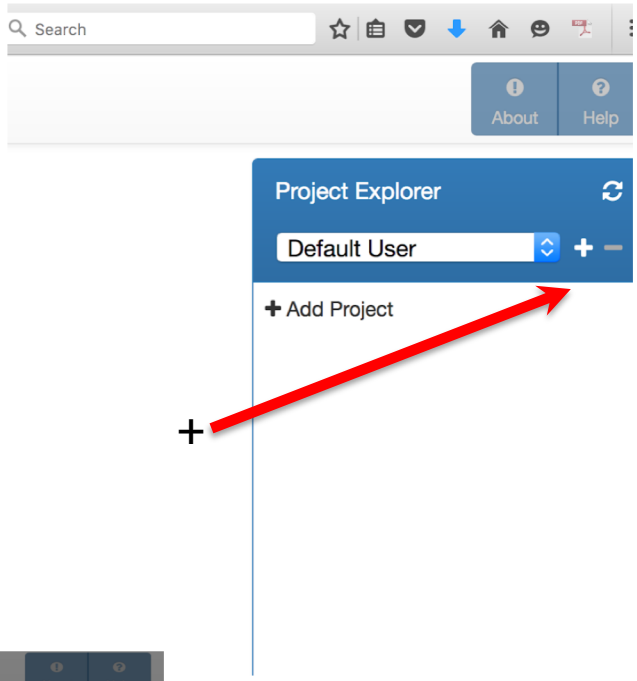
The **KIPR Software Suite** allow you to develop source code with the C **programming language**.

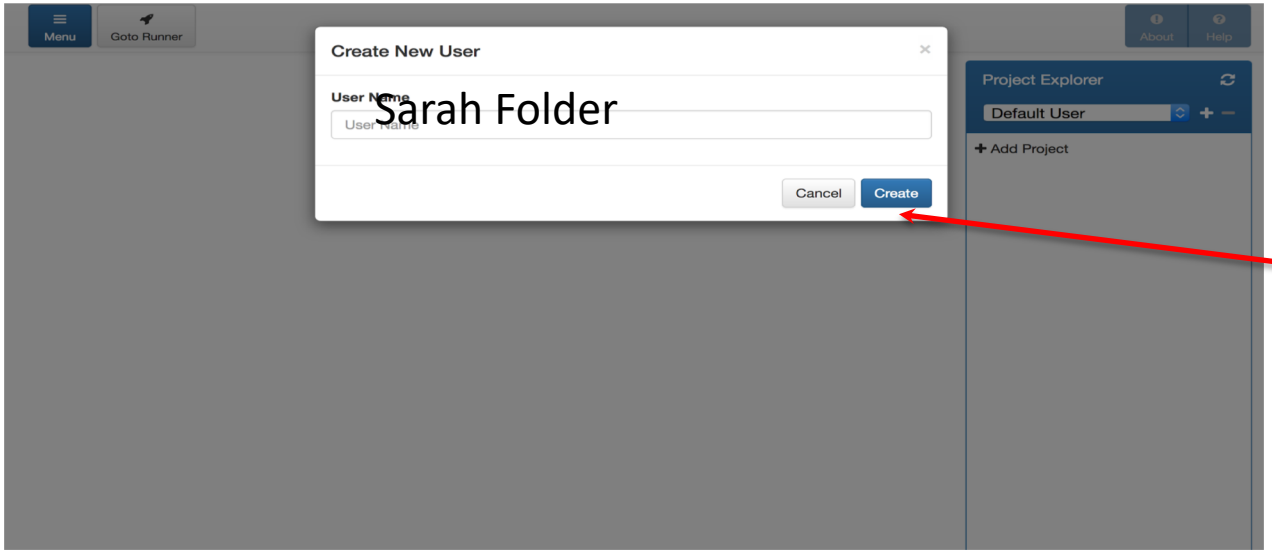
Programming Language = Language humans understand that can be turned into machine language

Click KISS IDE
IDE =Integrated Development Environment

Adding a Project

1. Under **Project Explorer**, you have options to add a user by clicking the + sign, **or** select your folder.
2. Add a **New User (+)**,
3. User name =“Student name and folder” . This is similar to a folder you will put all your different projects into. Not Special characters or “, ‘, .,etc.

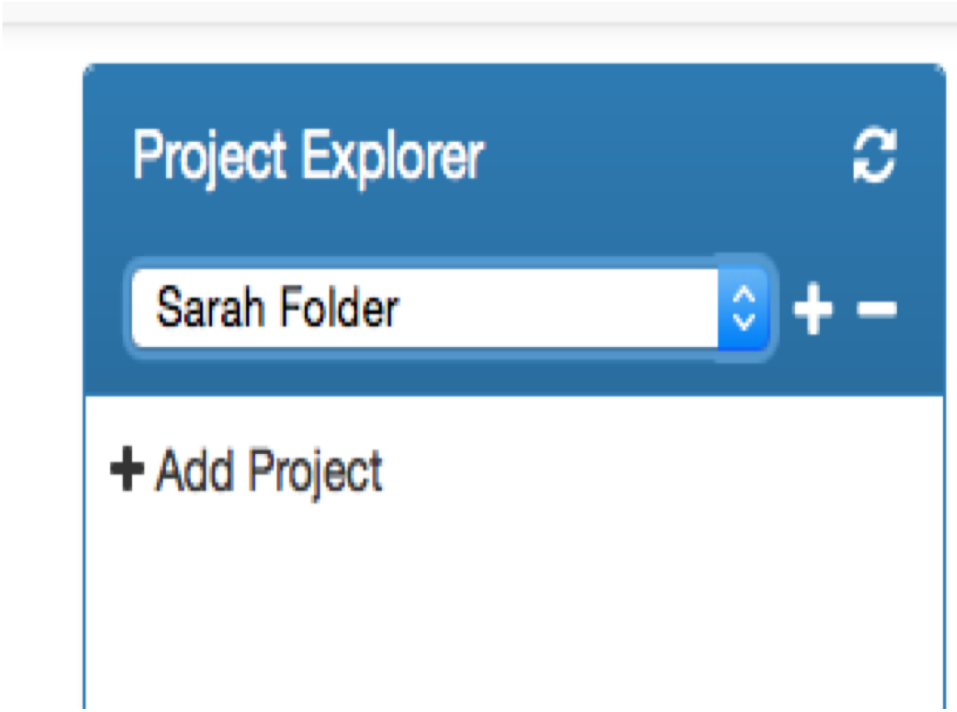




Click Create

Add a Project

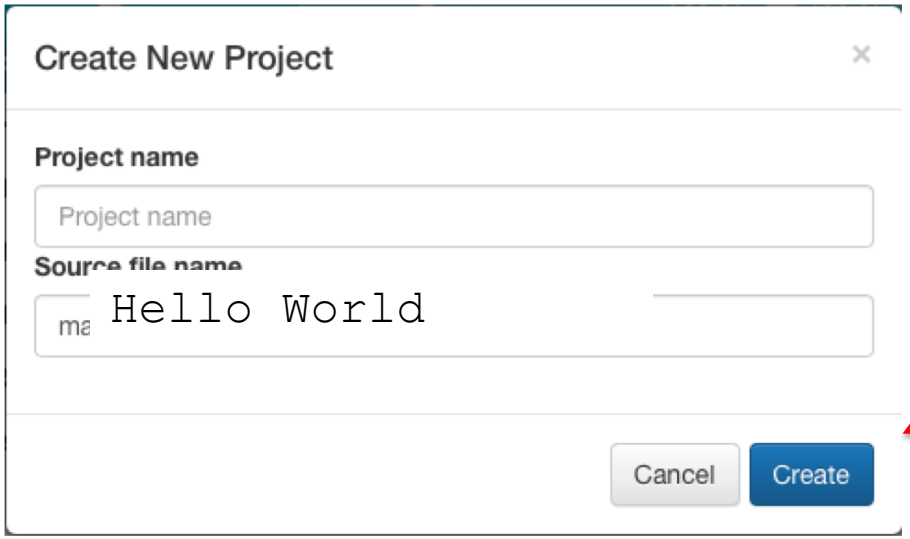
1. Click "**+Add Project**".
2. You are adding a project to your folder.
3. Name the new project



Name your project

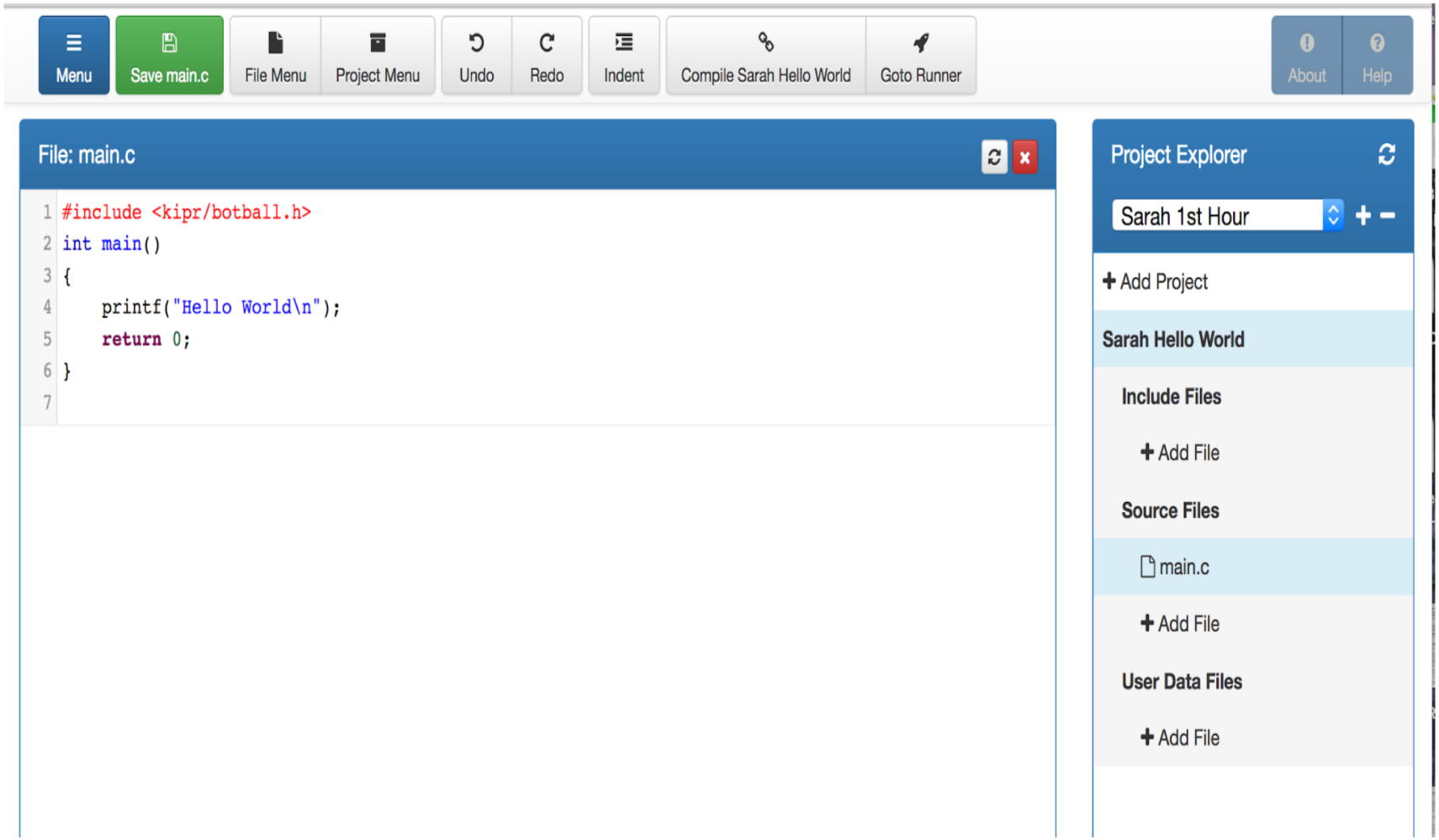
Give your *project* a **descriptive name**

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

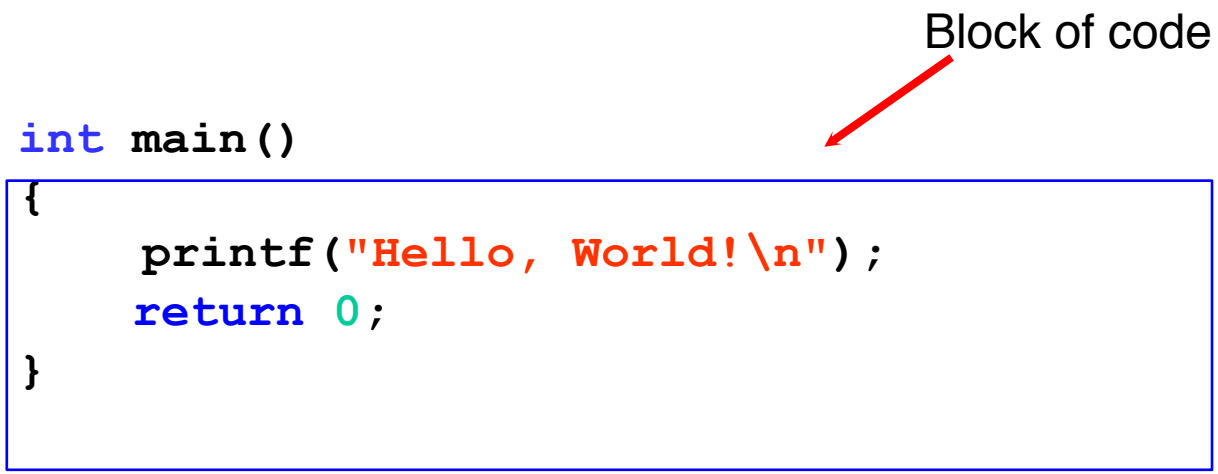


Click Create

This is Your Template



Learning About The C Template:
Hello, World!
Blocks of Code



Learning About the Template

The **KISS IDE contains a large library of functions** you can use to program your robots.
A function is like a title to an instruction book. When you call the function it does all of the commands in the book.

A "**function**" defines a list of actions to take.
A function is like a **recipe**. When you "**call**" (use) the function,
the program follows the instructions/recipe.

A `clean_house()` function could mean vacuum, dust, mop, change the linens, wash the windows, etc… all the commands specified in the function are executed.

KIPR Library

```
#include <kipr/botball.h>

int main()
{
  printf("Hello, World!\n");
  return 0;
}
```

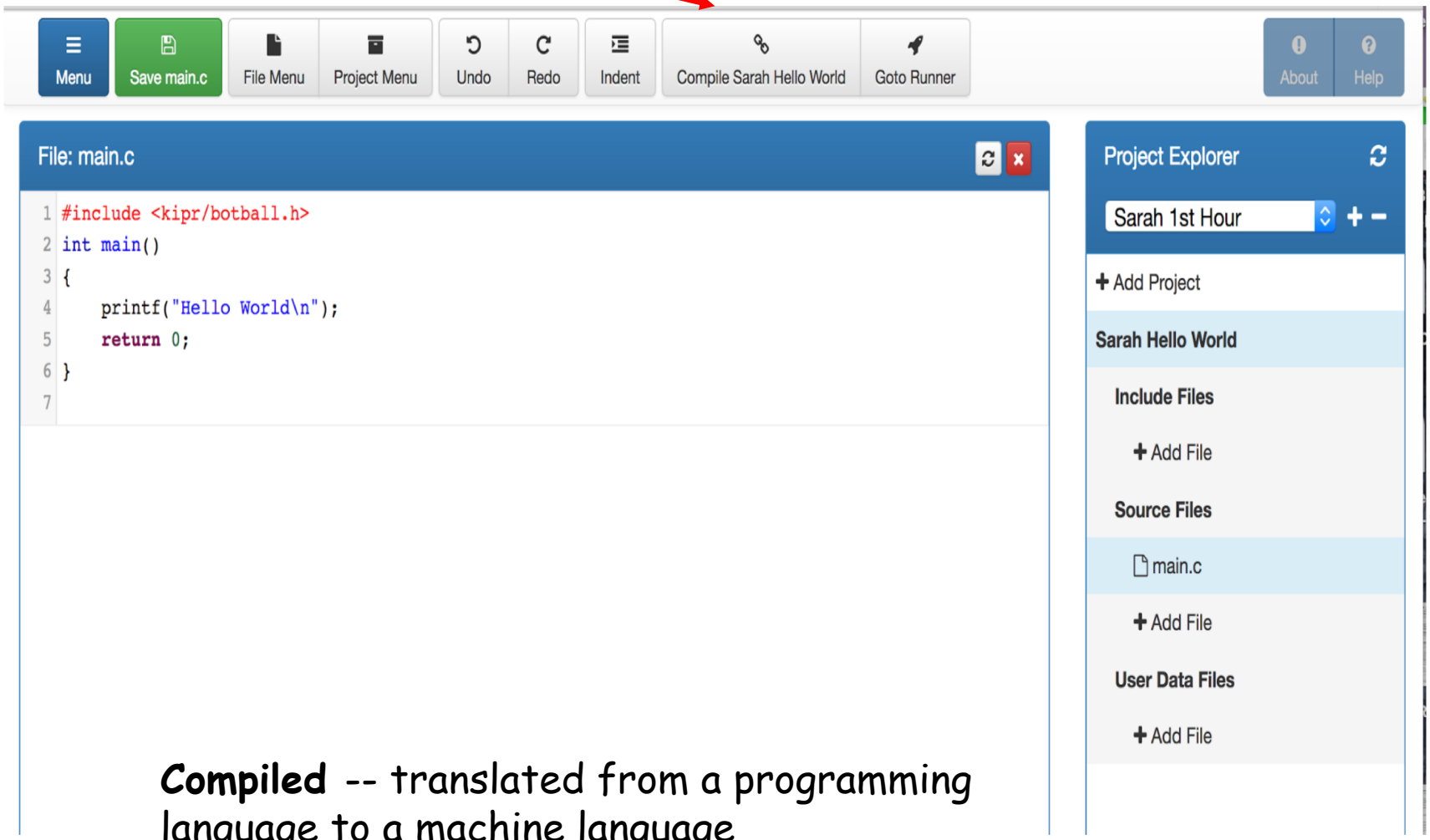
This is the "**main function**".
When you run your program,
the **main function** is executed.

Functions

Until you are familiar with the functions that you will be using while programming , use your “Function Tent” for easy reference. Copy and paste is also very helpful.

```
printf(“text\n”); //Prints text to display
motor(port#, % power); //Turns motor on at % power specified
msleep (# milliseconds);        //Program waits specified number of
milliseconds
ao();//All off, turns all motor ports off
enable_servos();//Turns servo ports on
set_servo_position(port#, position);//Moves servo in specified port to a set position
disable_servos (); //Turns off servo ports
digital(port #); //Refers to a specific digital port #
analog (port #); //Refers to a specific analog port #
```

Compile Your Program “Hello World”



Compiled -- translated from a programming language to a machine language

Compilation succeeded will appear if the **source code** is correct!

File: hello world.c

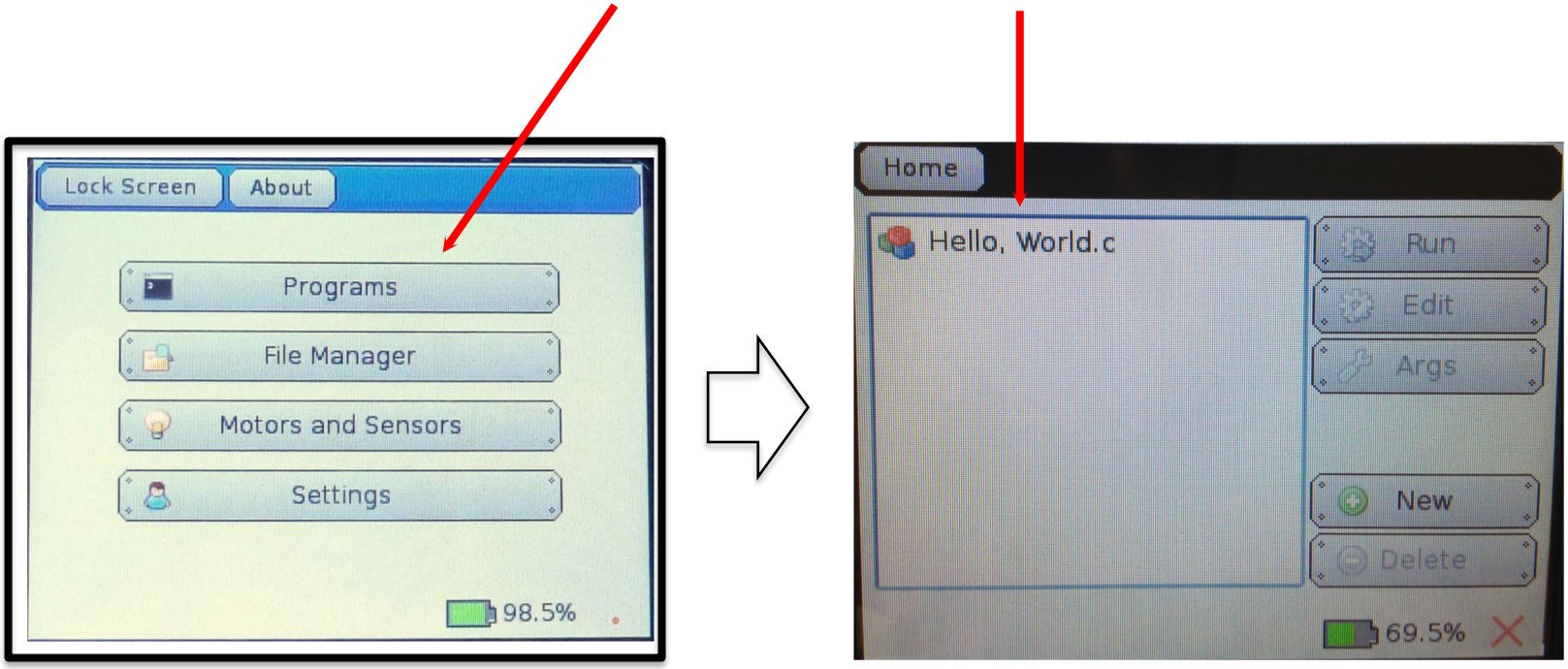
```c
#include <kipr/botball.h>

int main()
{
  printf("Sarah Hello World\n");
  return 0;
}
```

Compilation succeeded

Source Code = name for code written in programming language
C, C++, Java, Python = names of programming languages

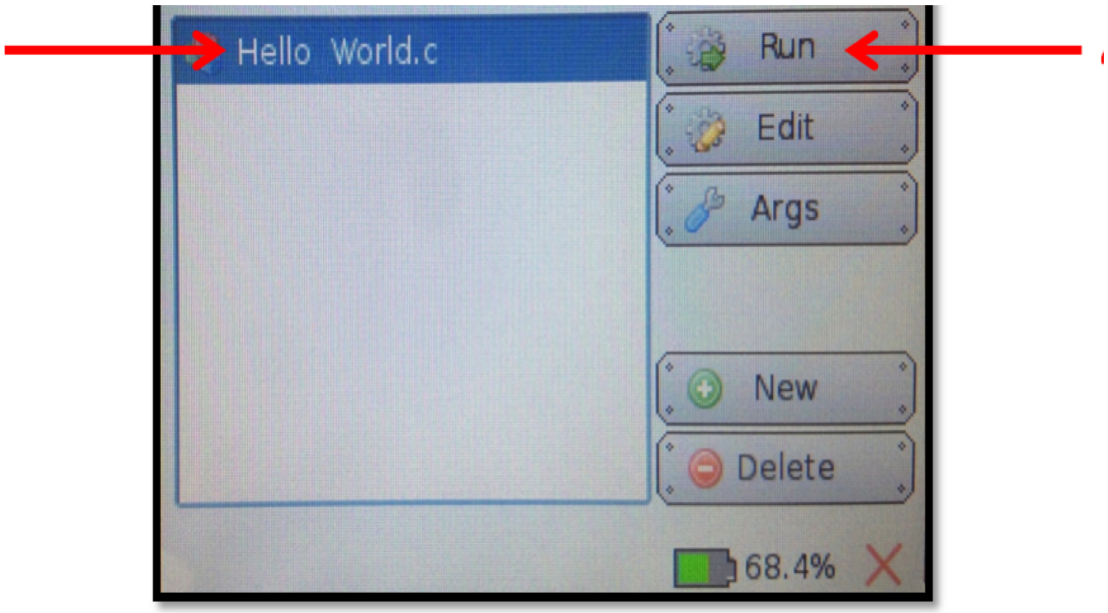
Running your Program on Your Robot

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



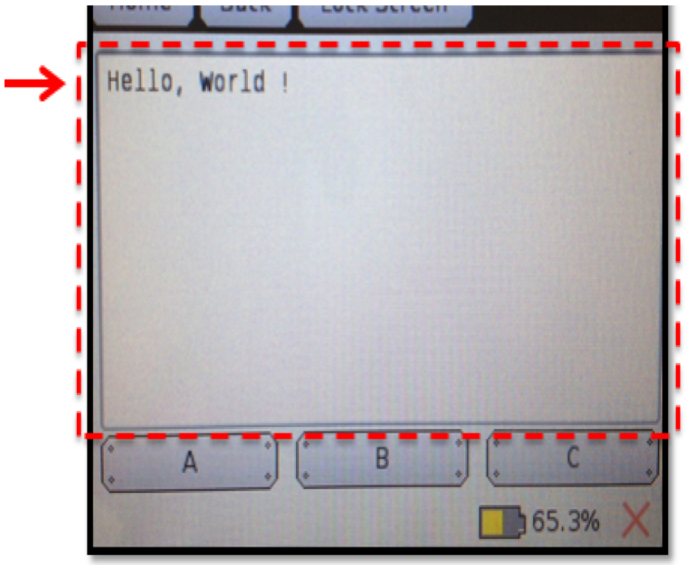
Running your Program on Your Robot

1. Select (touch) the **name of the program** you want to run **Note:** in the example below, this is “Hello, World.c”.
   - Press (touch) the **“Run” button** on the Wallaby.



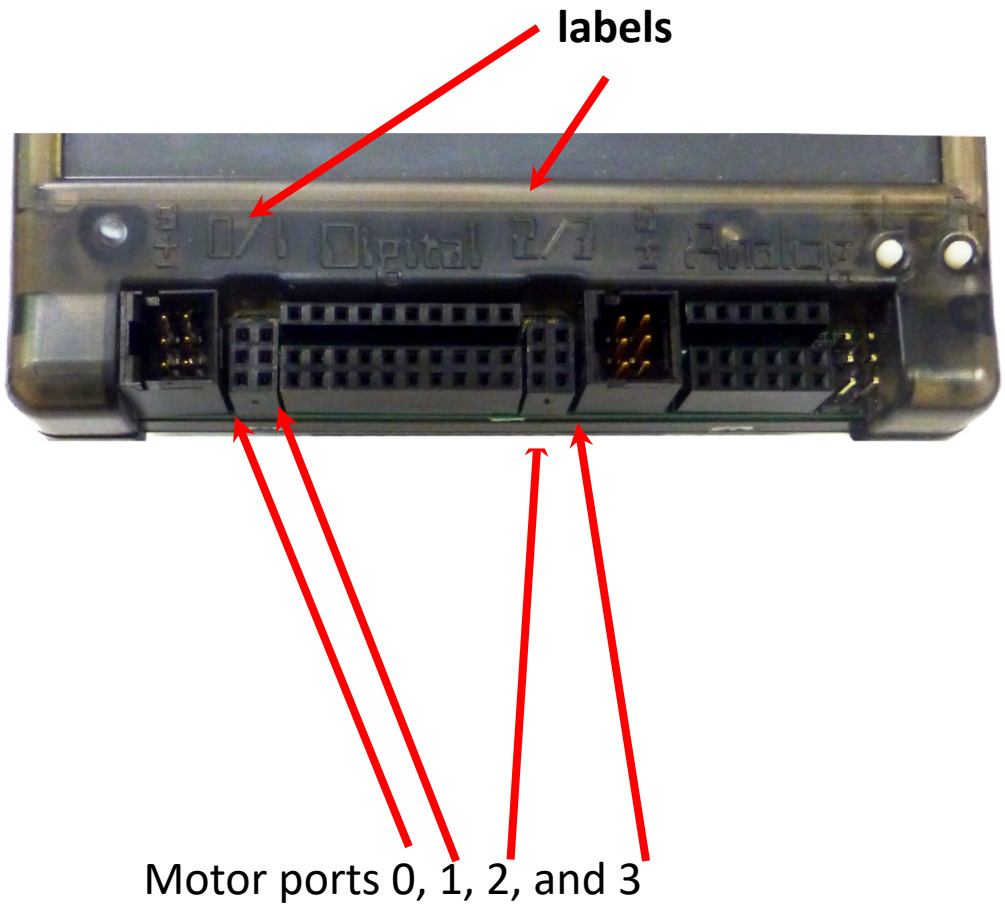
Running your Program on Your Robot

1. Go to the robot (wallaby).
2. The phrase "**Hello, World!**" will appear on your ***Wallaby screen***.

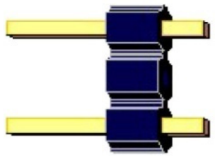


Connecting Your KIPR Wallaby Motor Ports

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



1. Find your motors. The motors have <u>two-prongs</u> on the plug end and have a double wire (2 wires)
2. Plug your motor plugs in port 0 and 3

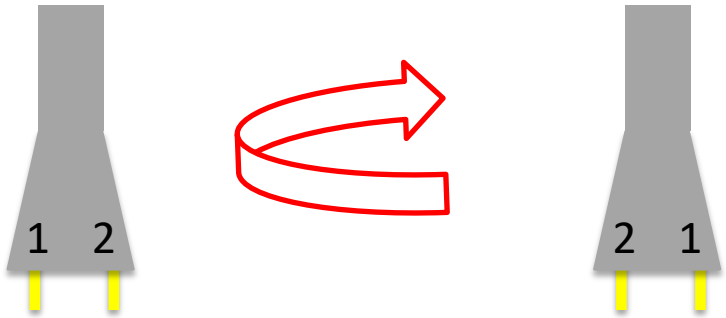


Drive motors have a 2 prong plug

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

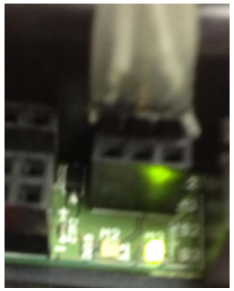


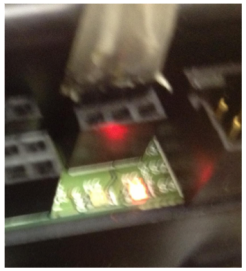
- Motors rotate in the direction that the electricity (electrons) move through them. One direction is clockwise motor rotation and the other direction is counterclockwise motor rotation

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

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

Lesson: Using the `motor()` Function

Goal: Introduction to motor function.

Activity:

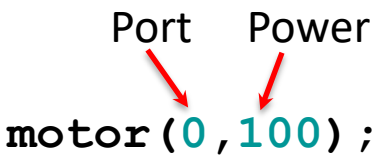
There are several functions for controlling motors, we will begin with motor().

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. Provide three examples of the motor function using different motor ports and powers.



Speed of Executing Code

- The wallaby controller executes the program (code) from top to bottom and goes to the next line faster than a blink of your eye.

- At over 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.

Let's make the Robot Move!

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

1. **Create** a new project in your folder, called **Move.**
2. **Task Analysis:** What is the program supposed to do?

Pseudocode (Task Analysis)

- Drive forward at 100%.
- Wait for 2 seconds.
- Stop motors.
- End the program.

1. **Type your code**, using the **motor(); msleep();** and **ao();** functions
2. Compile your program.
3. Run your program on your wallaby.
   - Did it run as you expected?

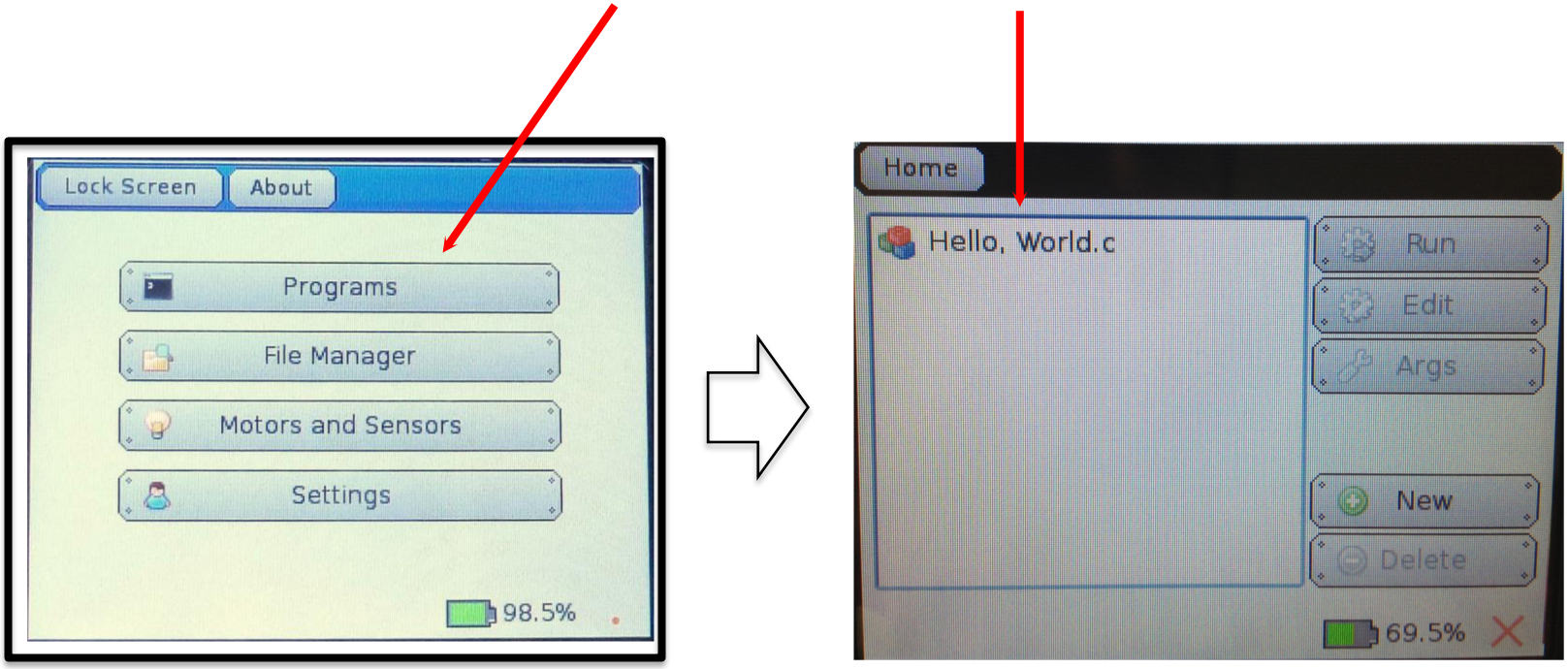
Solution

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

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

  ao();
  return 0;
}
```

What can you change to make it go a longer distance?
Play with the distance. Did it drive straight.

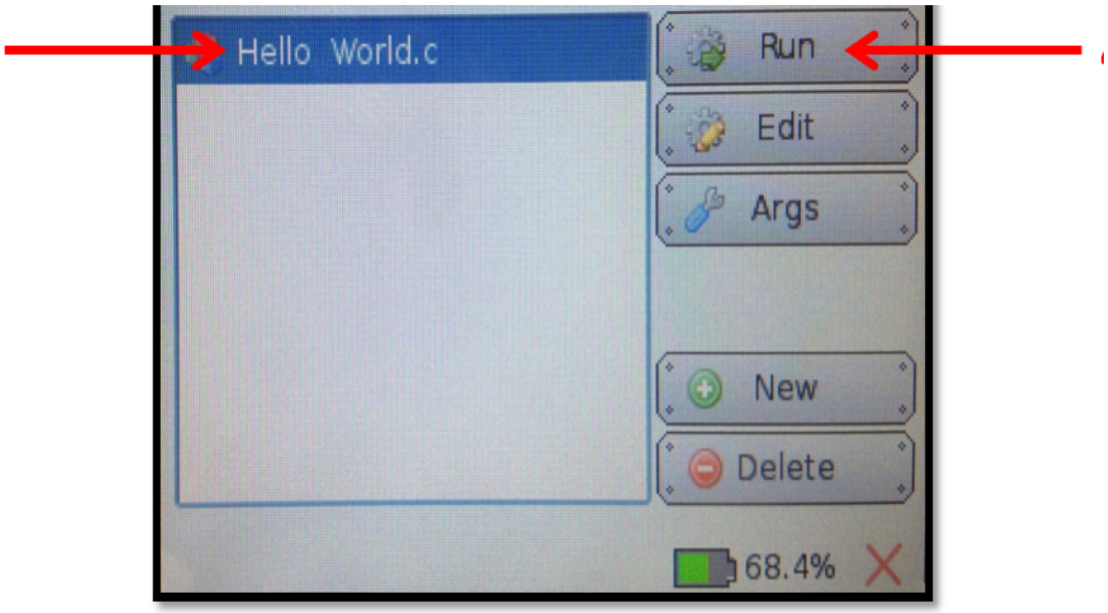
Running your Program on Your Robot

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



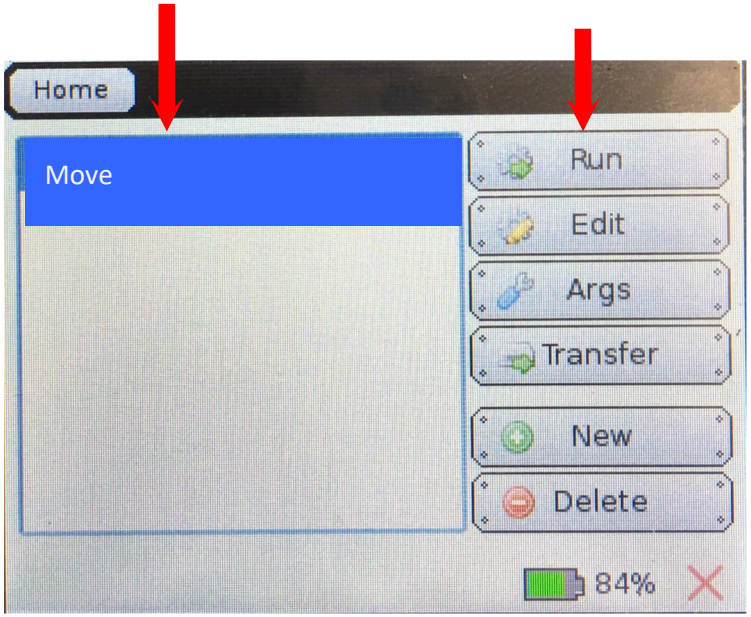
Running your Program on Your Robot

1. Select (touch) the **name of the program** you want to run **Note:** in the example below, this is “Hello, World.c”.
   - Press (touch) the **“Run” button** on the Wallaby.



Running Your Program

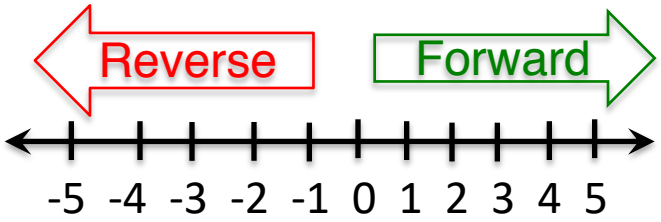
1. Highlight the program you want to run, in this case, *“Sarah drive 1”*.
2. Place your robot on the floor and then push the “Run” button.



Did you robot go straight? .

Driving Straight

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.

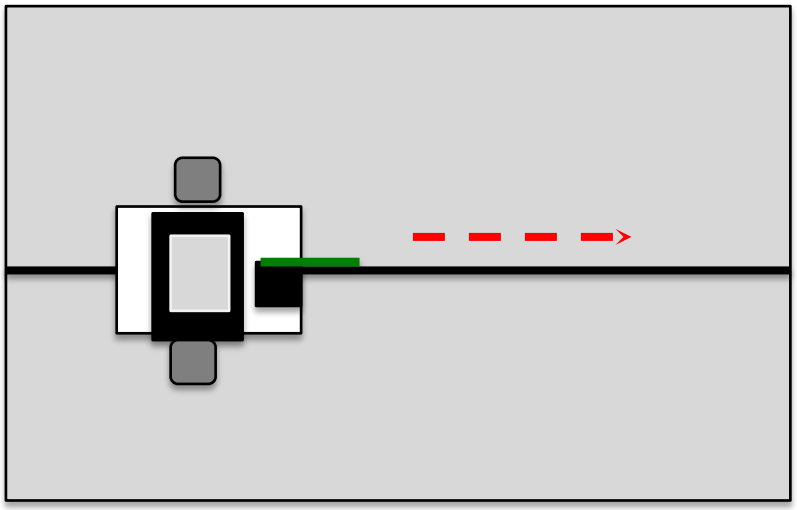
Making Turns

- 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

Drive your robot straight

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

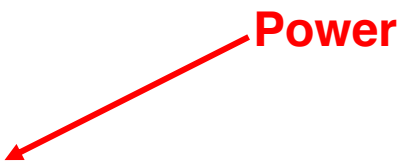
Adjust your wheel powers until straight.



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);
```



Have the students use the: Think Pair Share Strategy

2. Next slide will provide the solution.

Backwards

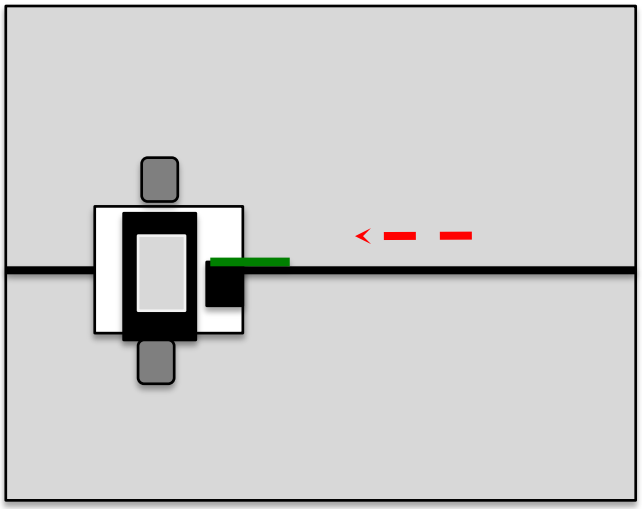
```
motor(0,-100);
motor(3,-100);
msleep(2000);
```

Negative numbers in the power argument will rotate the wheels backwards

Drive your robot backwards and straight

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

Adjust your wheel powers until straight.



Touch the Can on Circle 9

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

Use mat A

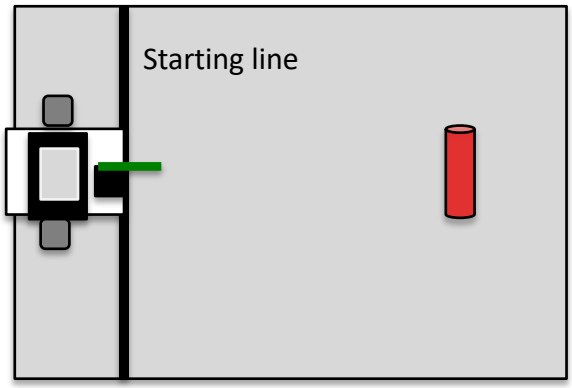
The robot must:

1.Go out and touch the can in circle 9

2. Stop - ao();

3. Pause for 2 seconds (msleep)

4. Go in reverse to start line.



Sample Coding

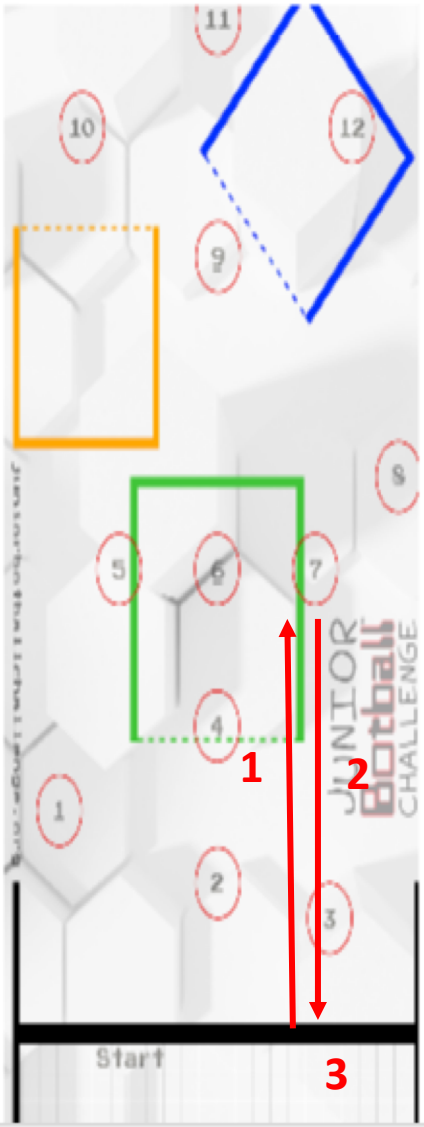
```
int main ()
{
    printf("Sarah Backwards");

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

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

Sample for Learning to Drive 3

Name/Team_____Challenge/Activity drive 3

Goal:

Touch the can on 6 and return to finish line

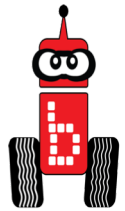


Pseudocode/Steps

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.

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.

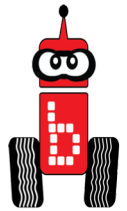
Limitations:

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.

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.

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);
```

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.

   Make observations.

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);
```

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.

Activity to Make Circles
Changing the Power and msleep

1. **Open Your Folder! Create a new project in your folder called "Circles".**
2. **On the template replace "Hello World" with the name of your new project, "Name" Circles. (you can to do this every time you have a new project).**
3. 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.
4. 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.

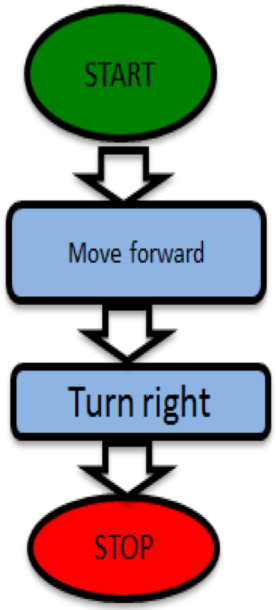
Learning About Comments (Pseudocode)

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?

Learning About Comments (Pseudocode)

- Comments as pseudocode help you keep track of what is going on in the program.
- You can make a flow chart or list and then convert it to pseudocode.
- To make a comment two slash marks must be typed first: `//`

  Complete Comment: `//Prints Hello World to screen`

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

Sample Program

```
int main()
{
   printf("Hello, World!\n"); // Prints Hello World to screen
     return 0;
}
```

Why would you put a comment here?

Add a Comment

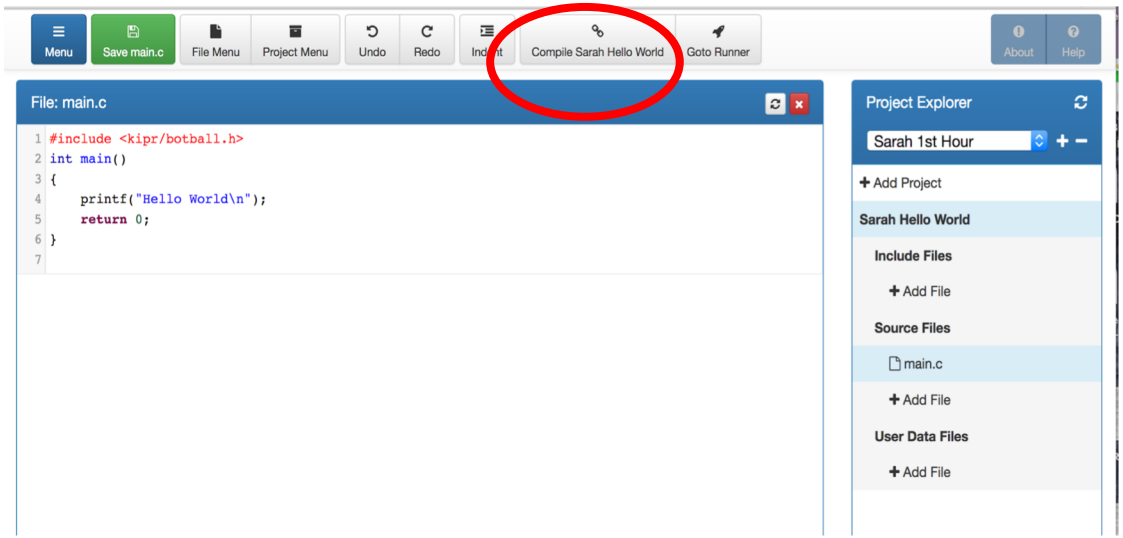
1. Add the //Print Hello world to screen comment to the program
   - Just like using a word processing program you can click to set your cursor and then use return to make space for the comment.
   - Type the comment into your program. The comment can go on the line before the printf function or on the same line as the function.
2. Continue to the next slide.
3. 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

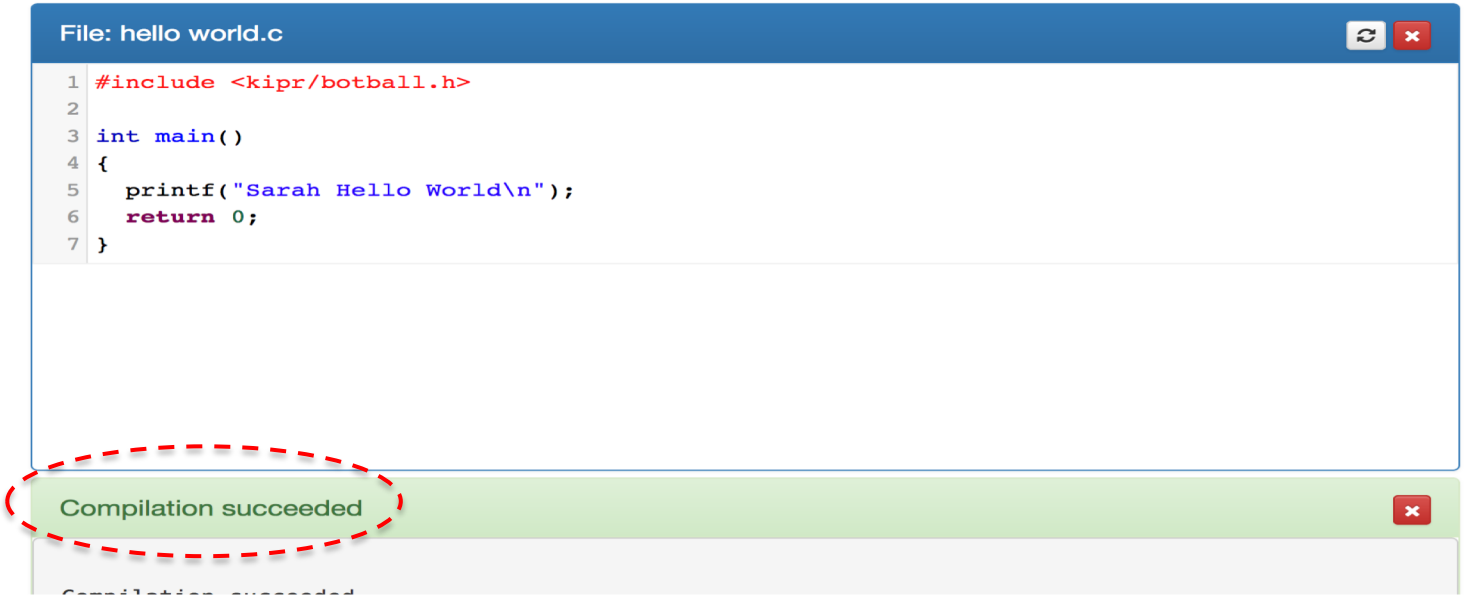
```
int main()
{
    printf("Hello, World!\n"); //Print Hello World to screen
    return 0;
}
```

Go to the next slide

Compile

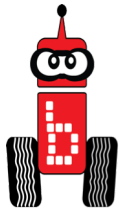


Watch to see if it Succeeded!



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

Terms to Understand



- **Machine Language** -- what the computers understand- Bytes
- **Executes** -- in terms of a computer running or carrying out the instructions
- **Source Code** -- name for code written in programming language
- **Compiled** -- translated from a programming language to a machine language
- **Programming Language** -- Language humans understand that can be turned into machine language
- **C, C++, Java, Python** -- names of programming languages

This was a quick start. You may now go back into the curriculum to **Module 5 Moving Robots** and explore.