What is a Robot

TEACHER'S RESOURCE

Goals

- To help students understand what a robot is
- To understand the basic components needed for a robot
  - Mechanical Structure, Effectors, Power, Computer, Sensors, Computer Program
- To understand the types of tasks and jobs robots perform
- To learn terms and vocabulary related to robots

Preparation

Project slides up on a board or provide print outs for each group

Activity

Follow the slides and complete the activities

KISS INSTITUTE FOR PRACTICAL ROBOTICS

What is a Robot? Activity

Objectives:

- What is a robot?
- What do you need to build and control a robot?
- What types of tasks/jobs do robots perform?
- What terms do we need to know?

STUDENT ACTIVITY

What robots do we use everyday?

- Create a poster/list that describes what robots we use everyday, and why these are robots
- Share out!
- Compile a list of the robots

?

KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

What makes a robot a robot?

List 5 or 6 words or phrases that you think make a robot a robot:

KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Commit and Toss

1. Draw a "+" if you agree
2. Add one more component
3. Toss

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Can we add to it?

- Structure
- Effectors
- Sensors
- Power
- Computation
- Information

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Humans vs. Robot Subsystems

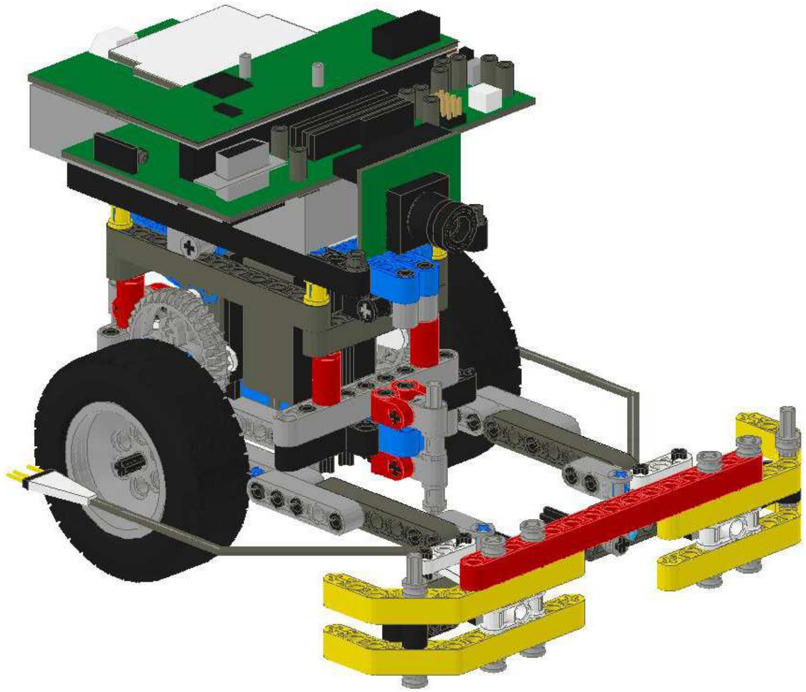
People	Robots
Bones	Mechanical Structure
Muscles	Effectors
Senses	Sensors
Digestion/Respiration	Power
Brain	Computer
Knowledge	Computer Program

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Structure

Robot Structure

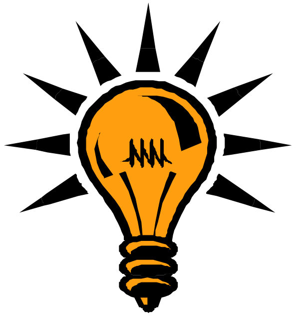
- Provides support to the robot-like your skeleton
- Joints in structure normally have effectors (like muscles) attached
- Holds sensors in position

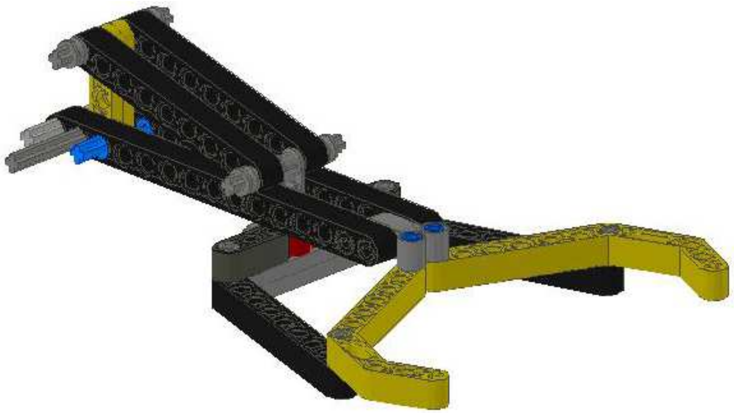


KISS INSTITUTE FOR PRACTICAL ROBOTICS

Effectors

- Used to change the state of the robot itself
- Used to change the state of the world
- *Examples:*
    - o Motors, thrusters, arms, or legs
    - o Voice synthesizers, buzzers, and lights





KISS INSTITUTE FOR PRACTICAL ROBOTICS

Sensors

Proprioceptive sensors

- Report on the current state of the robot- you know you are sitting down even with your eyes closed

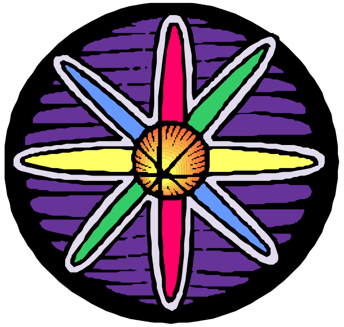
External sensors

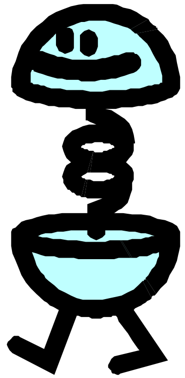
- Report on the current state of the environment the robot is in
  - o Light sensors, range sensors, touch sensors, etc.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

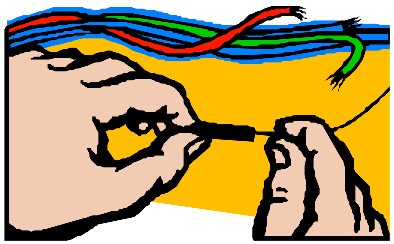
Power (Energy)

- Power Source
  - Batteries, solar panels
  - Springs, hydraulics, pneumatics
  - Nuclear reactor

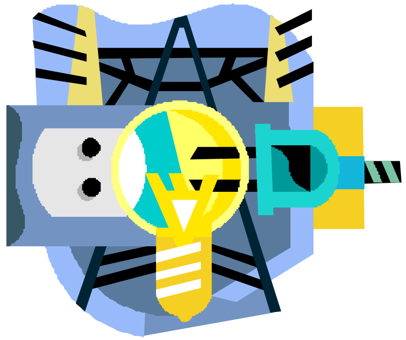




- Power Distribution
  - Wires



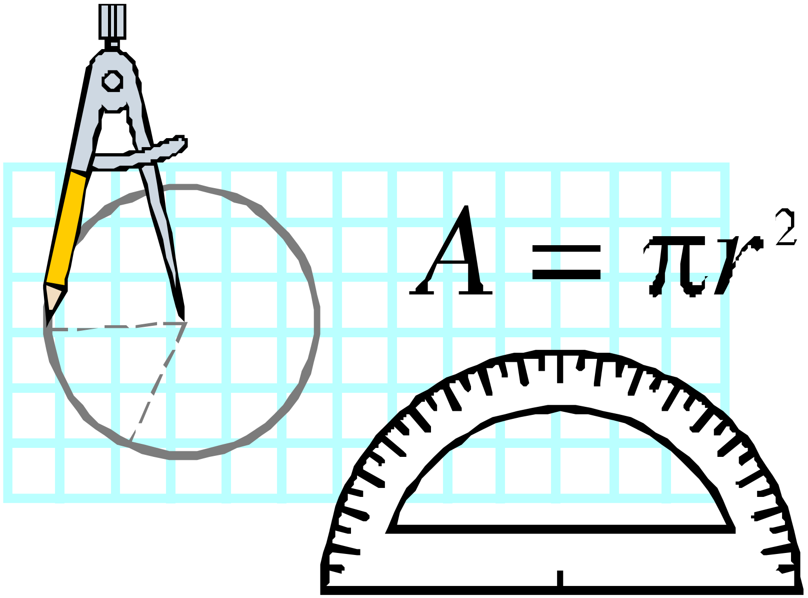
- Power management
  - Regulators
  - Converters



KISS INSTITUTE FOR PRACTICAL ROBOTICS

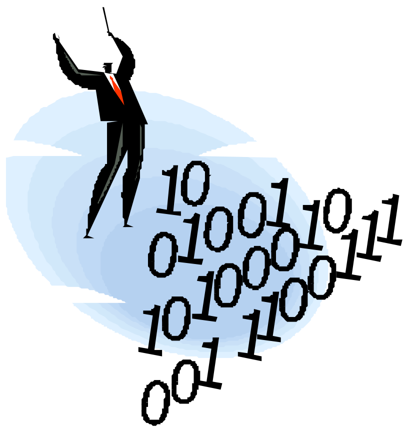
Computation

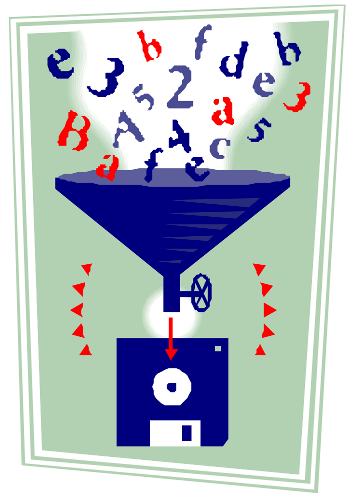
- Used to interpret sensor values; perception
- Used to generate proper effector commands
- Used to project effects and plan actions



Information

- Internal Information
  - How to interpret sensor values
  - How to generate effector commands
  - Internal state & history
- External Information
  - World, user & predictive models
- Program
  - Determines robot actions
  - Forms robot plans
  - Debugging - introspection





KISS INSTITUTE FOR PRACTICAL ROBOTICS

Introductory Kit Overview

TEACHER'S RESOURCE

#1 KIPR Link Controller
#1 KIPR Link Power Adapter
#1 KIPR Link Mini USB Cable
#1 Botball Screwdriver
#1 11/32 Wrench
#1 1/4 Wrench
#1 Introductory LEGO bag
#2 Motors - SG-5010 Standard Motor
#2 SG-5010 Servos
#1 Bag of Screws & Stand-offs
#1 Bag of Push Rivets
#1 Bag of Brass Screws

KIPR Metal Parts to include:

- #1 Metal Ball Caster
- #2 Motor Brackets
- #1 Chassis
- #1 Angle Bracket
- #2 Servo Round Horn, #1 Long Horn

Sensors to include:

- #1 Light Sensor
- #1 Small Touch
- #1 Large Touch
- #1 Long Lever Sensor
- #1 ET Rangefinder
- #2 Small IR

You can access the kit components on our online store:

http://botballstore.org/product/link-Introductory-kit

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Introductory Kit Scavenger Hunt Activity

TEACHER'S RESOURCE

Goals

- Learn the kit components and be able to identify individual parts
- Be able to group parts into categories (KIPR Metal Parts, LEGO, Sensors)
- Distinguish between a motor and a servo motor
- Keep track of parts

Preparation

Place all of the kit components out on a table and have the students use:

1. A checklist (something very helpful in engineering) to identify and account for all parts (printable example on next slide)

2. Have the students use their science/engineering/robotics notebook to make their own checklist to account for all parts

Resources

The online KIPR store is a good resource to help with the identification of specific kit components:
http://botballstore.org/product/link-Introductory-kit
(instructions on resource slide)
*Having a tool box or plastic bins with lids to store kit components is helpful and keeps them from getting lost in the classroom

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Introductory Kit Checklist

TEACHER'S RESOURCE

- [ ] #1 KIPR Link Controller
- [ ] #1 KIPR Link Power Adapter
- [ ] #1 KIPR Link Mini USB Cable
- [ ] #1 Botball Screwdriver
- [ ] #1 11/32 Wrench
- [ ] #1 1/4 Wrench
- [ ] #1 Introductory LEGO bag
- [ ] #2 Motors - SG-5010 Motor
- [ ] #2 SG-5010 Servos
- [ ] #1 Bag of Screws & Stand-offs
- [ ] #1 Bag of Push Rivets
- [ ] #1 Bag of Brass Screws

KIPR Metal Parts to include:

- [ ] #1 Metal Ball Caster
- [ ] #2 Motor Brackets
- [ ] #1 Chassis
- [ ] #1 Angle Bracket
- [ ] #2 Servo Round Horn
- [ ] #1 Long Servo Horn

Sensors to include:

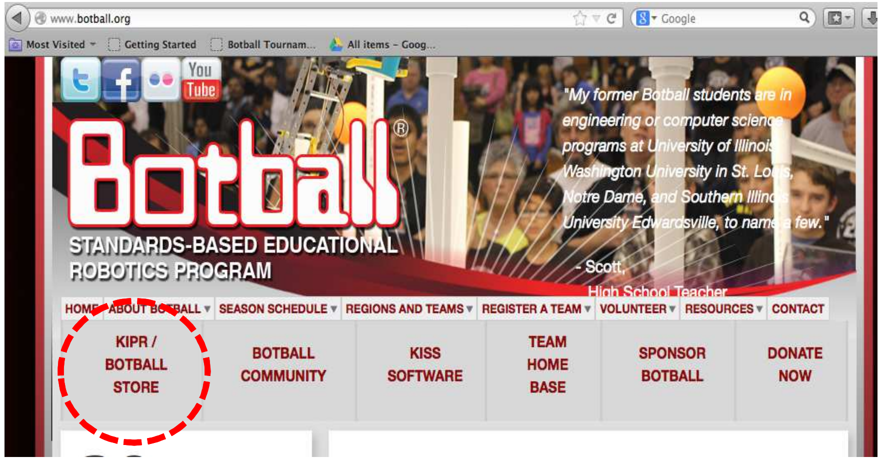
- [ ] #1 Light Sensor
- [ ] #1 Small Touch
- [ ] #1 Large Touch
- [ ] #1 Long Lever Sensor
- [ ] #1 ET Rangefinder
- [ ] #2 Small IR

KISS INSTITUTE FOR PRACTICAL ROBOTICS

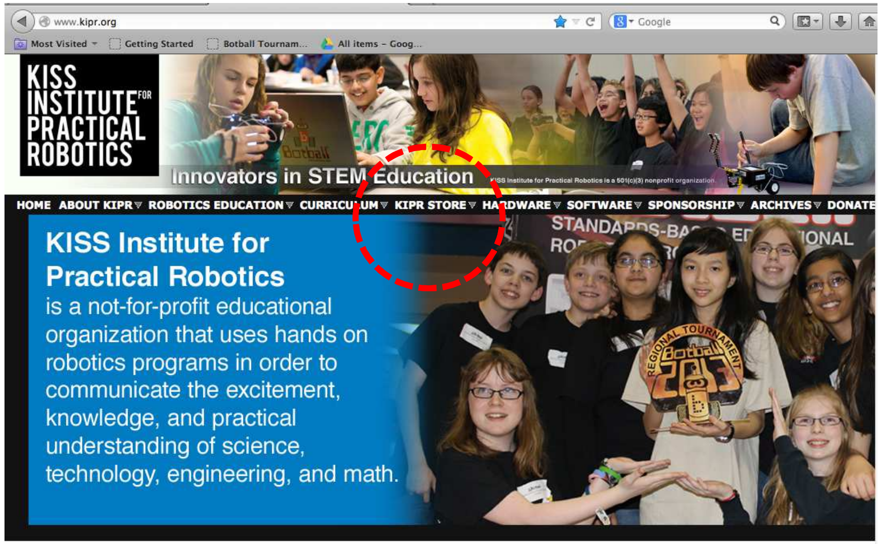
Online Resources-The Botball Store

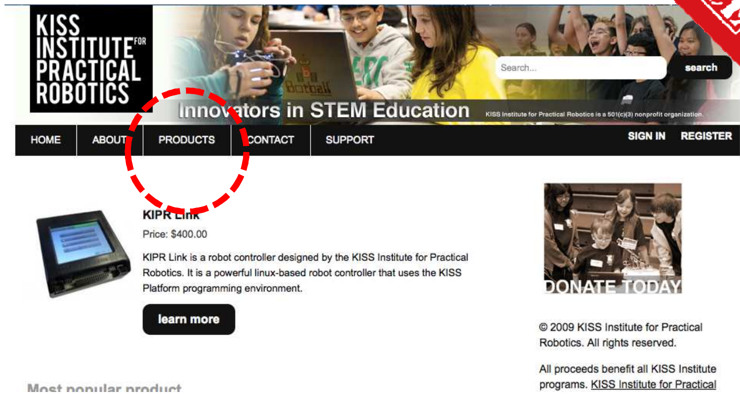
TEACHER'S RESOURCE

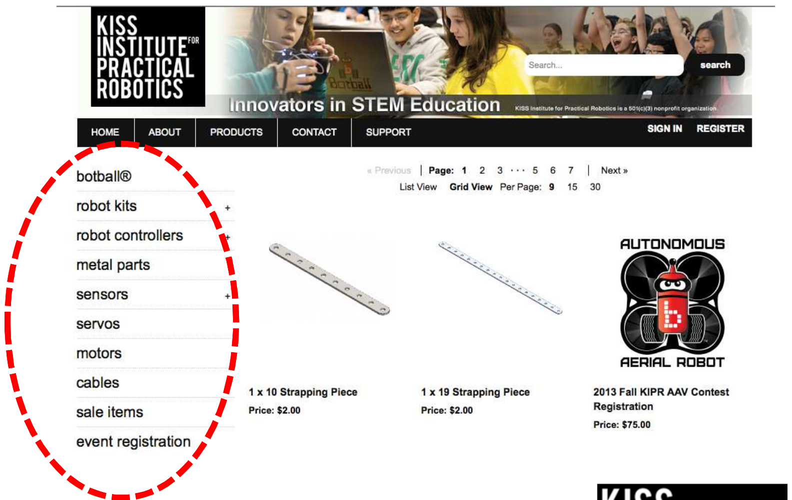
www.botball.org



www.kipr.org







Resources- If you can't get online

KIPR Link Controller

USB – micro USB
download cable

Motor

KIPR Link Wall Charger
(ONLY use this charger with your Link)

Servo Motor

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Resources- If you can't get online continued

Large Touch Sensor

ET- Rangefinder sensor

Small Touch Sensor

Long Lever Sensor

USB Camera

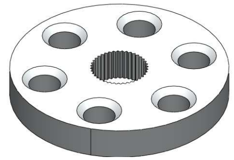
Small IR Reflectance Sensor

Light Sensor

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Resources- If you can't get online continued

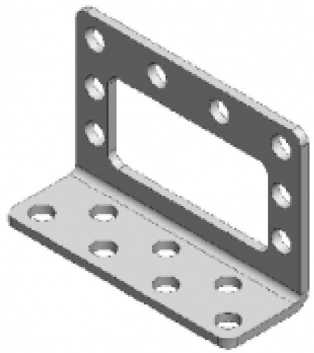
KIPR Metal Part (KMP) Chassis



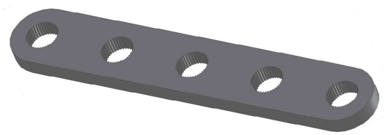
KMP Round Servo Horn

KMP Angle Bracket

LEGO Bag



KMP Motor Bracket



KMP Long Servo Horn

KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Introductory Kit Scavenger Hunt

Task-

- Sort and identify all of the robot kit components and use a checklist to make sure you have everything

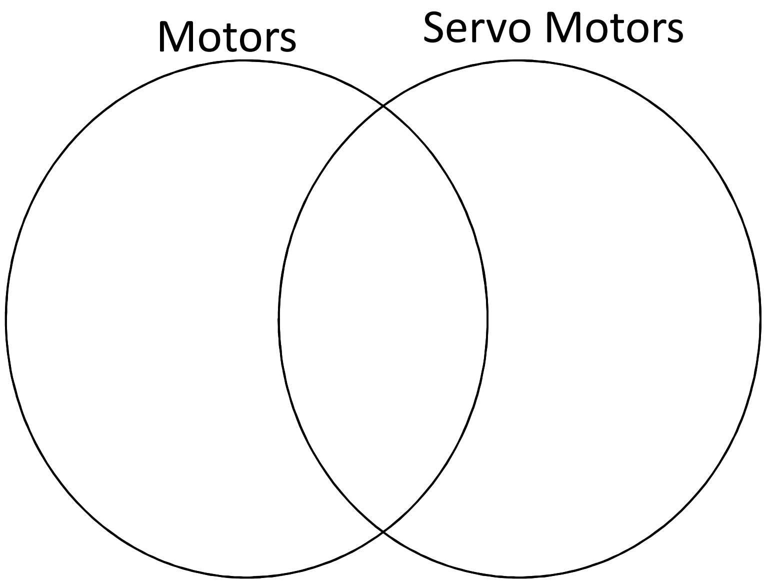
OR

- Generate your own checklist using your notebook and sort and identify all of the robot kit components

THEN

- Use a Venn diagram to compare and contrast motors and servo motors

- Be the expert- each team member must describe and identify a kit component explaining how they identified it until all items have been correctly identified



KISS INSTITUTE FOR PRACTICAL ROBOTICS

Building the DemoBot

TEACHER'S RESOURCE

Goals

- To get a robot built to complete the programming activities
- To learn to follow directions/schematics to construct the robot

Preparation

- Make sure if your robot is NOT already built to complete the Introductory Kit Overview Activity
- Using the guide, have the student build the DemoBot
- The teacher can build the DemoBot ahead of time if desired

*HINT- The hardest part is starting the screws through the metal servo horn to attach the tire. Students may need help starting the screws.

Activity

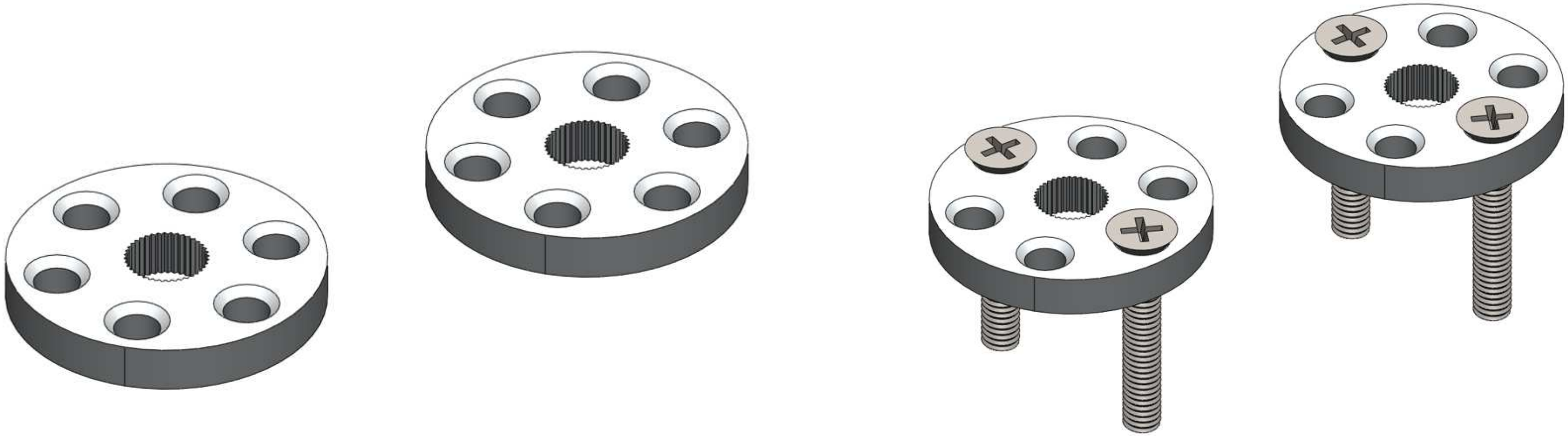
- Using the slides, build the DemoBot

KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

KIPR DemoBot

KISS INSTITUTE FOR PRACTICAL ROBOTICS

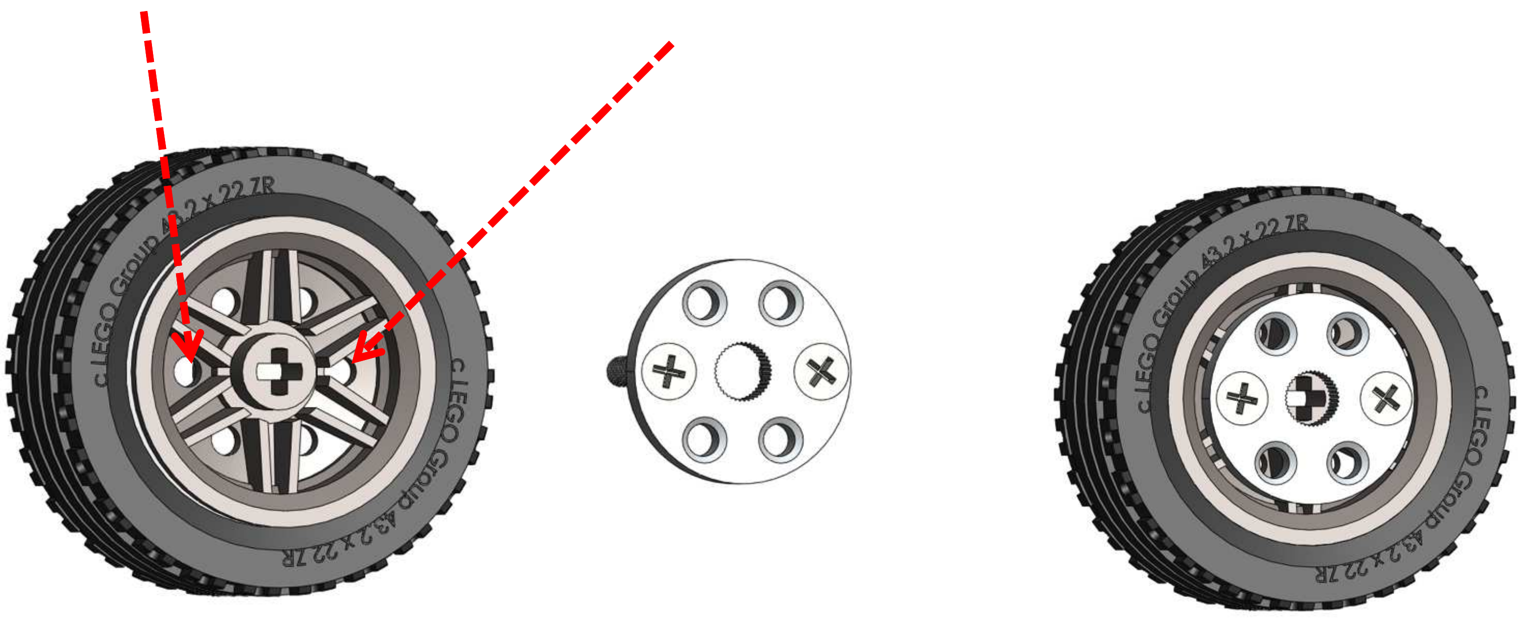


From the bag with two KIPR Metal Pieces (KMP) servo horns and two KMP servo arms remove the two round horns and four machine screws

Using the screwdriver start the machine screws through two opposite holes in the metal round servo horn

(It may be hard to get them started- teacher may need to help)

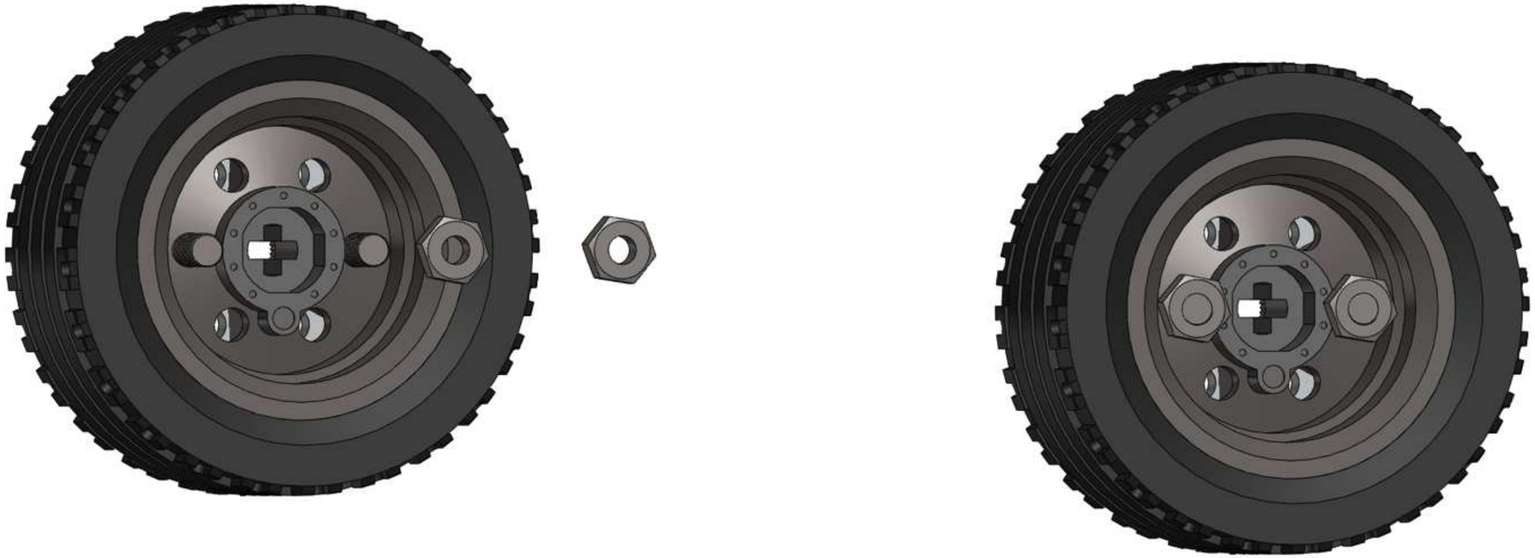
KISS INSTITUTE FOR PRACTICAL ROBOTICS



You will need two tires on rims

The KMP round servo horn with the two screws will go onto the side of the rim that is recessed the least amount

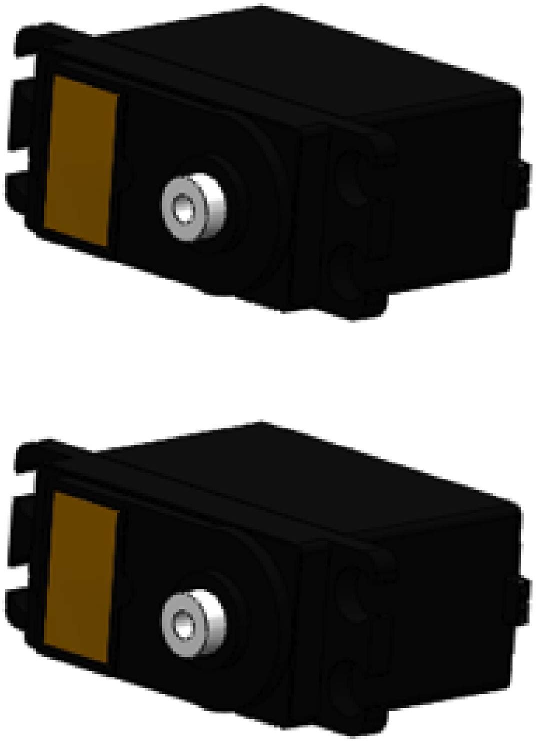
KISS INSTITUTE FOR PRACTICAL ROBOTICS

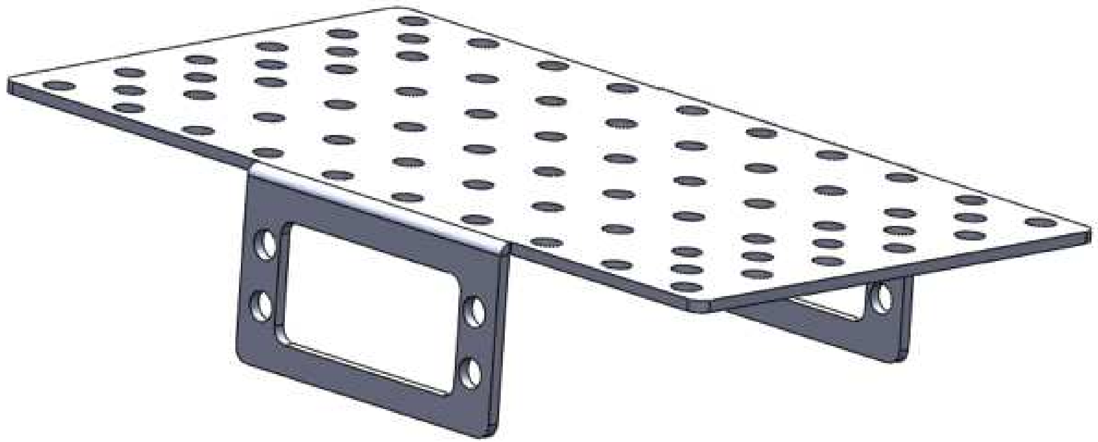


Using the nuts (from the bag with the KMP servo horns) and a screwdriver, attach the KMP round servo horn to the rim

You will attach these to the motors later in the build

KISS INSTITUTE FOR PRACTICAL ROBOTICS

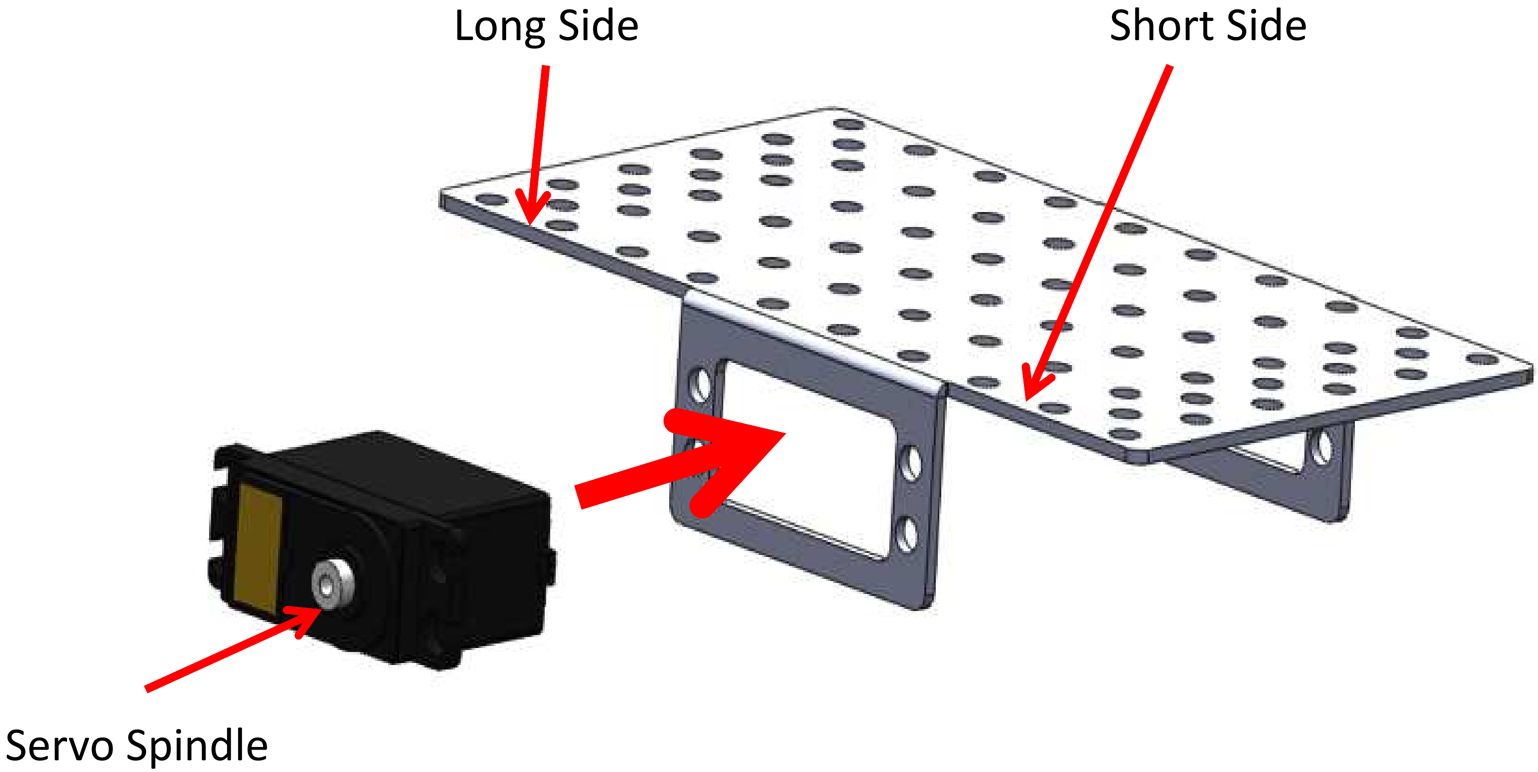




You will need two motors and the KMP Chassis

The kit has two (2) motors and two servos. You can distinguish motors from servos by the wiring. Motors have a double grey wire and servos have a triple red, orange, brown wire.

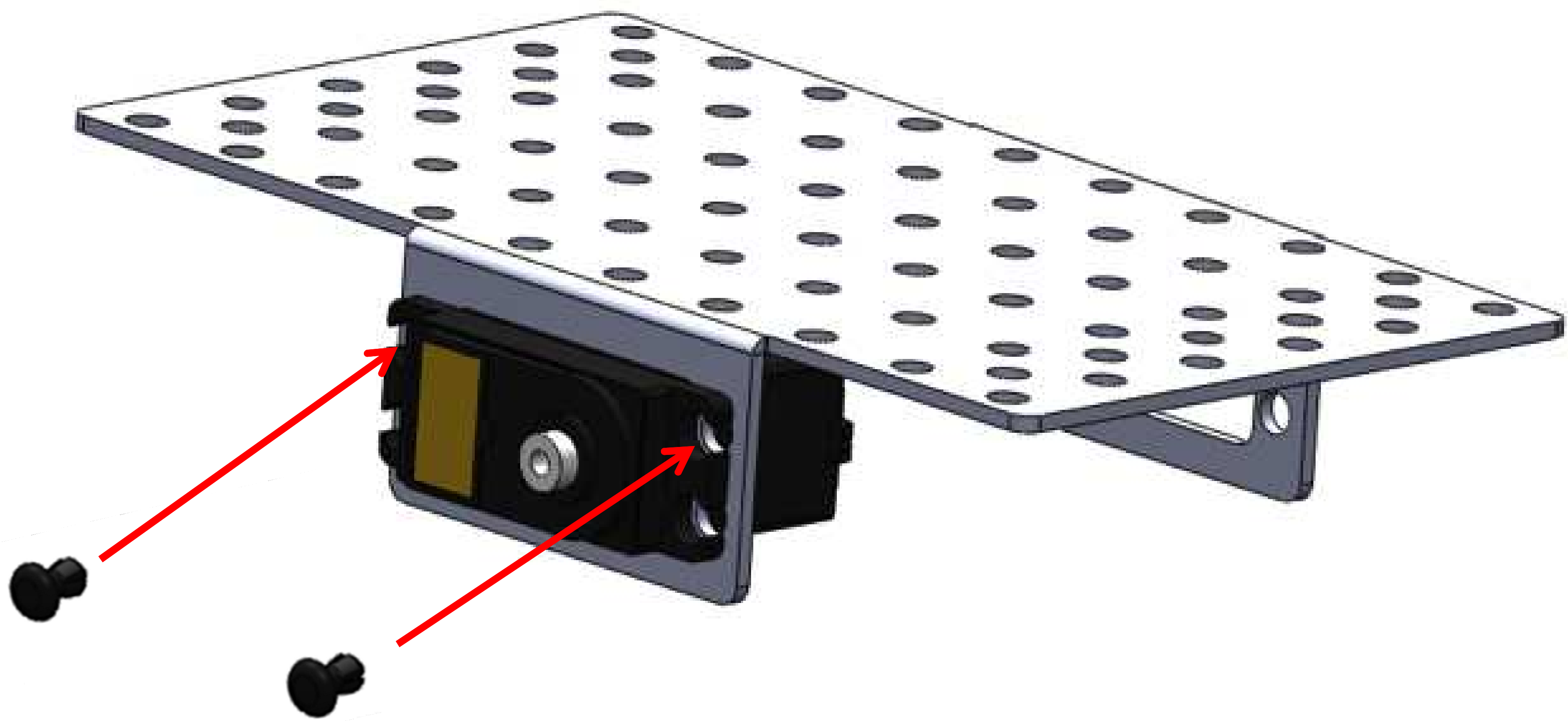
KISS INSTITUTE FOR PRACTICAL ROBOTICS



Insert the motors into the chassis

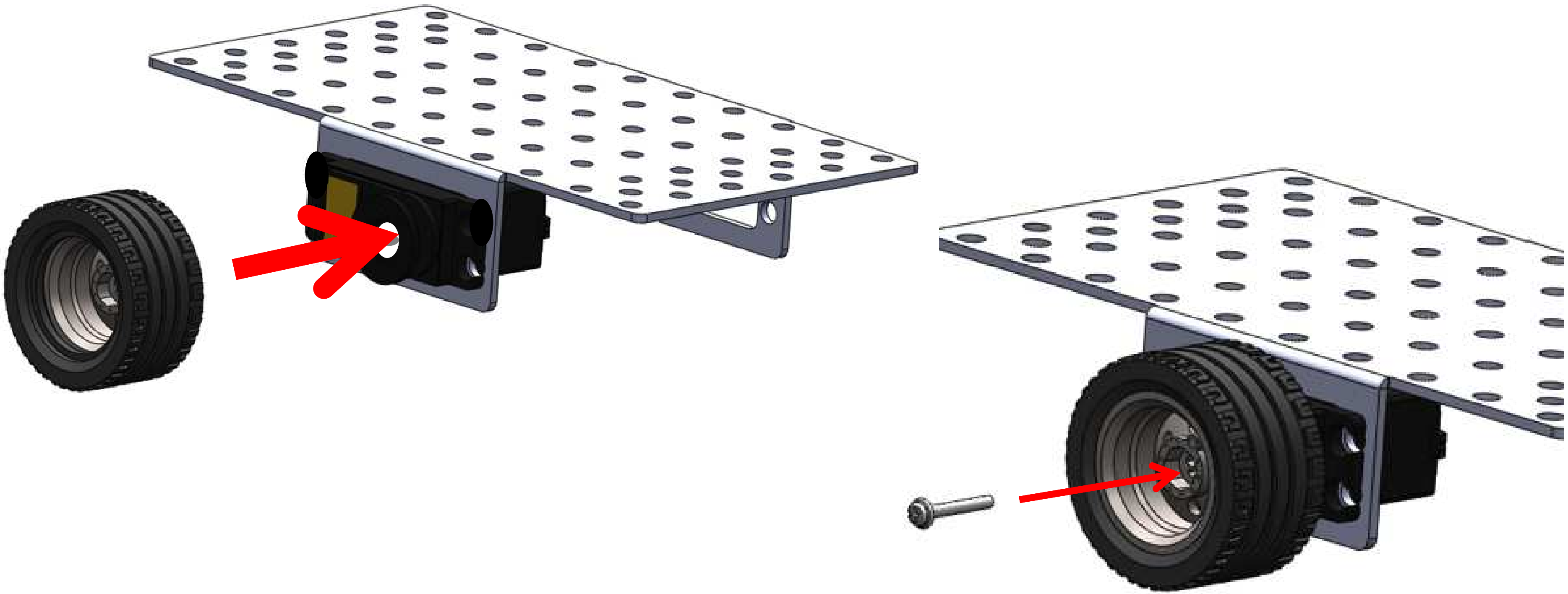
Make sure and place the white servo spindle towards (closer to) the shorter side of the chassis

Repeat for the other side



Using two plastic pop rivets attach the motors to the chassis and repeat the process on the other side with the other motor.
For a more stable connection you can use the short bolts and nuts instead of pop rivets.

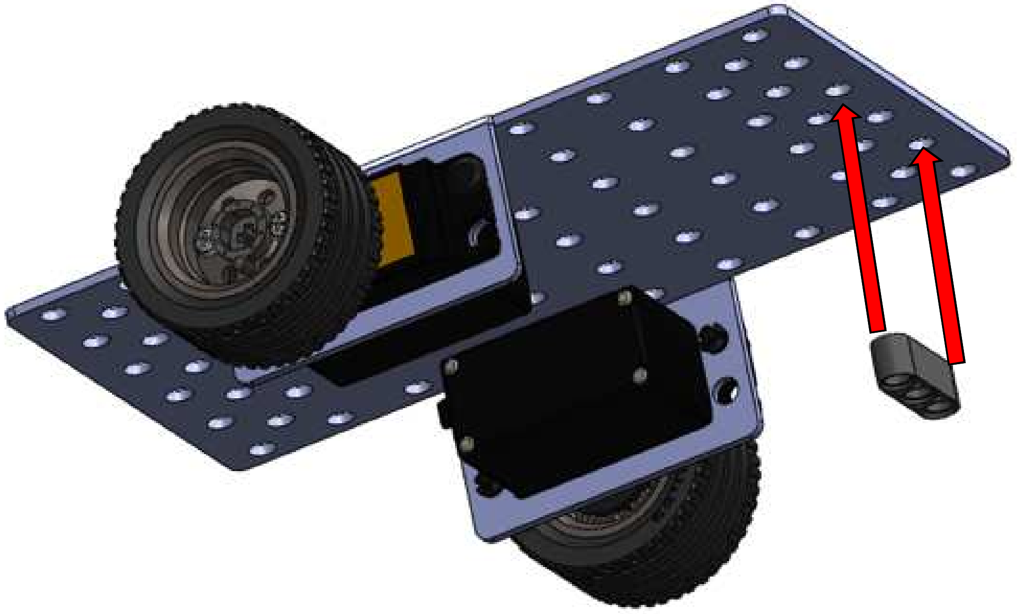
KISS INSTITUTE FOR PRACTICAL ROBOTICS

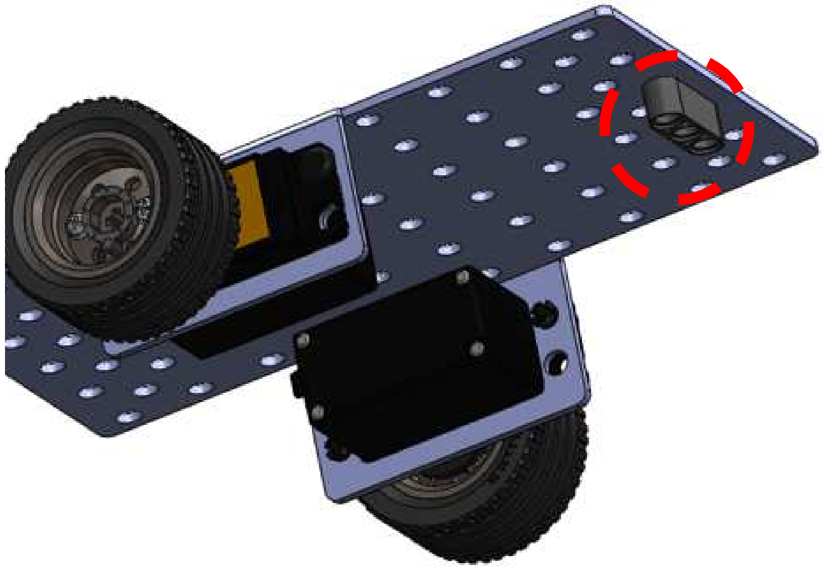


Using the wheels you assembled earlier, push the round metal servo horn onto the servo spindle

Using the screwdriver, secure the wheel in place with a long screw (found in the same bag with the KMP round and arm servo horns)
Repeat for other side

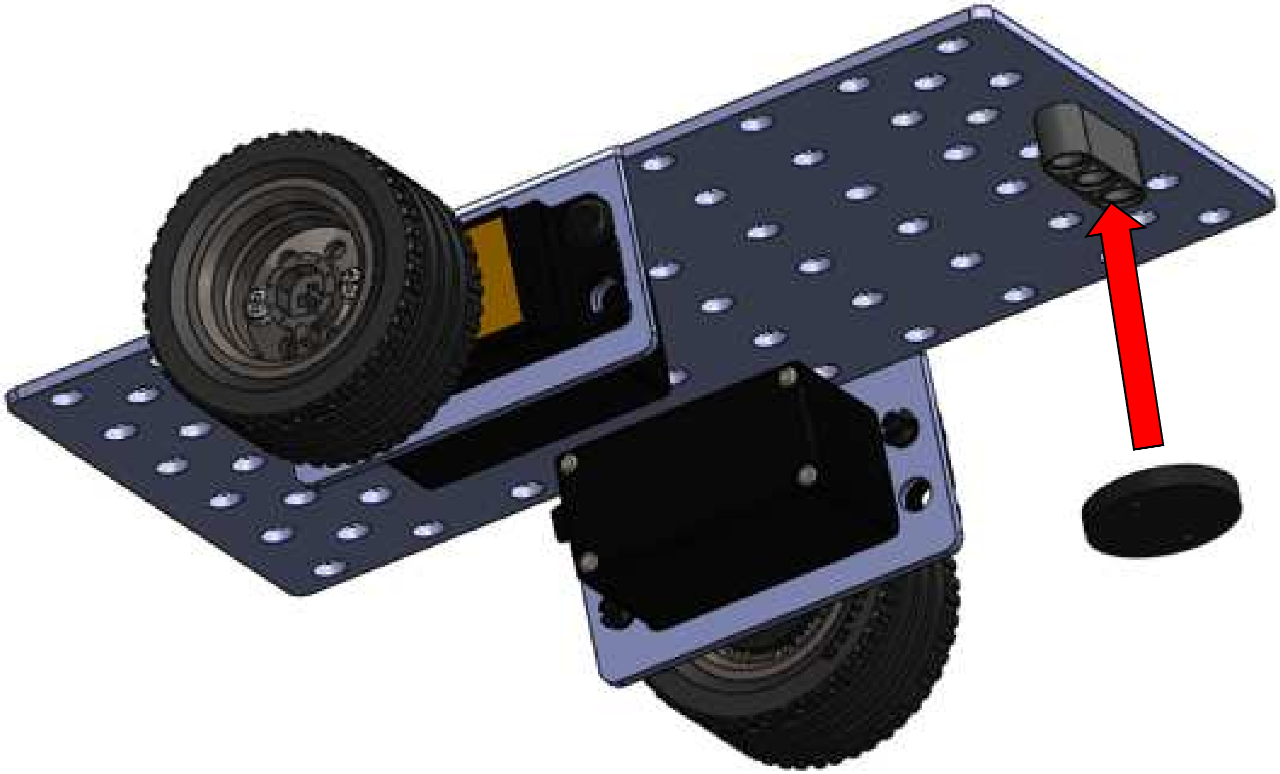
KISS INSTITUTE FOR PRACTICAL ROBOTICS





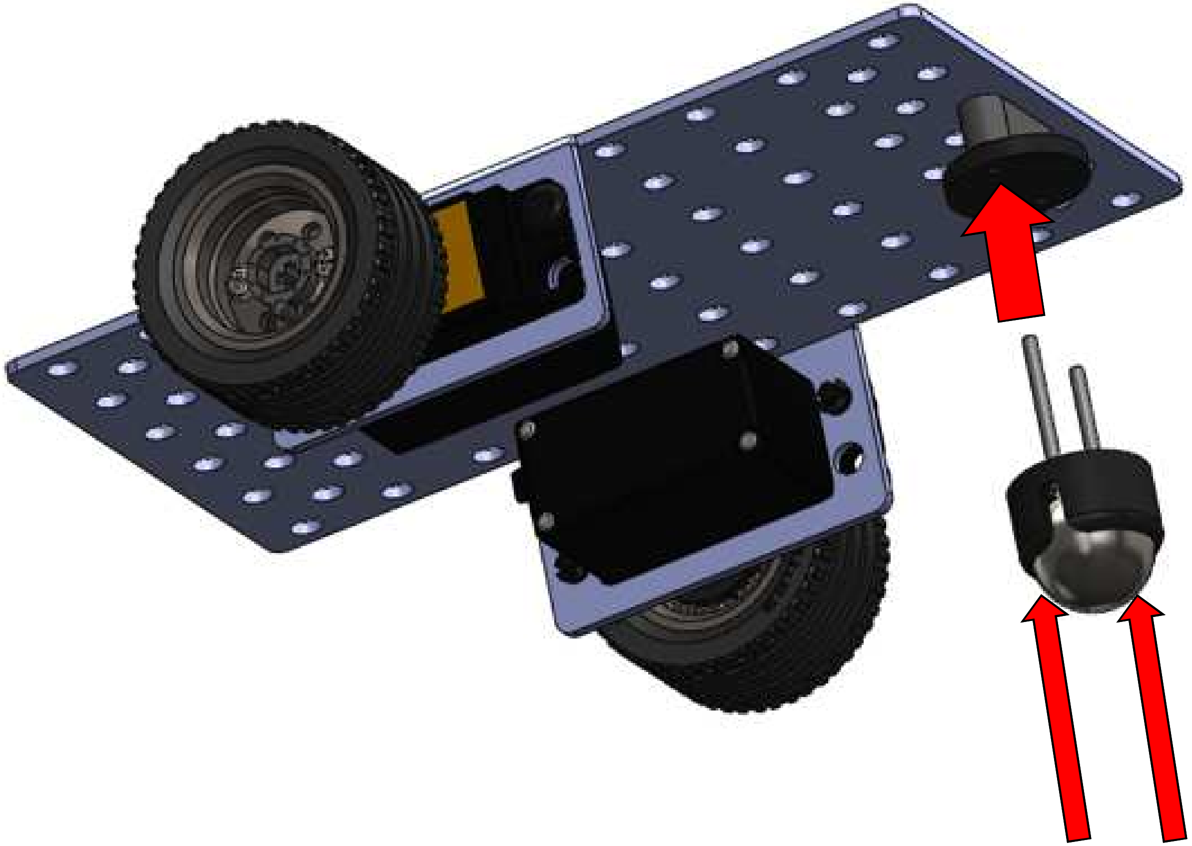
Take one (1) 3-Hole LEGO piece.
This shows the position that it will be mounted in the next few slides.

KISS INSTITUTE FOR PRACTICAL ROBOTICS



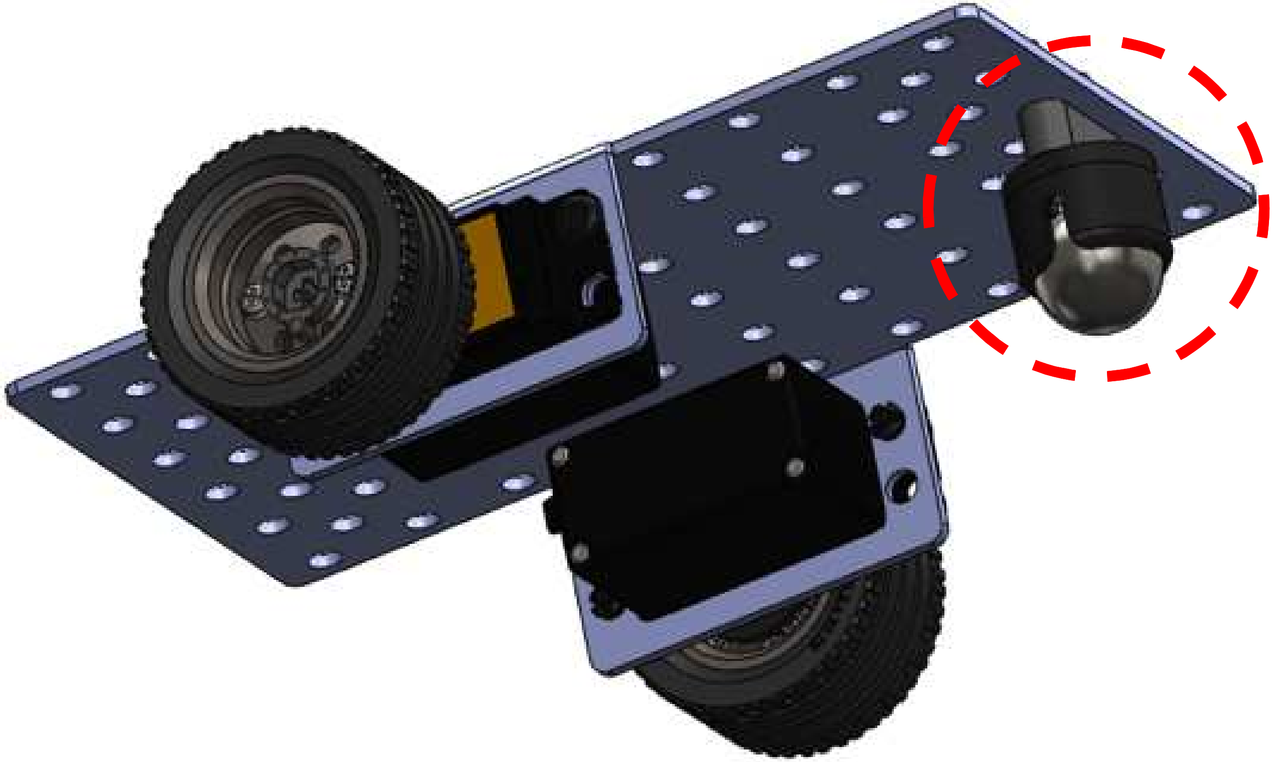
Get the “Pololu Ball Caster”.
The bag will contain a ball bearing caster, 2 short & 2 long bolts with nuts and two plastic washers- one thick and one thin. The washer in this slide is the THICK one and is shown here for position only (it will be attached in later slides).

KISS INSTITUTE FOR PRACTICAL ROBOTICS



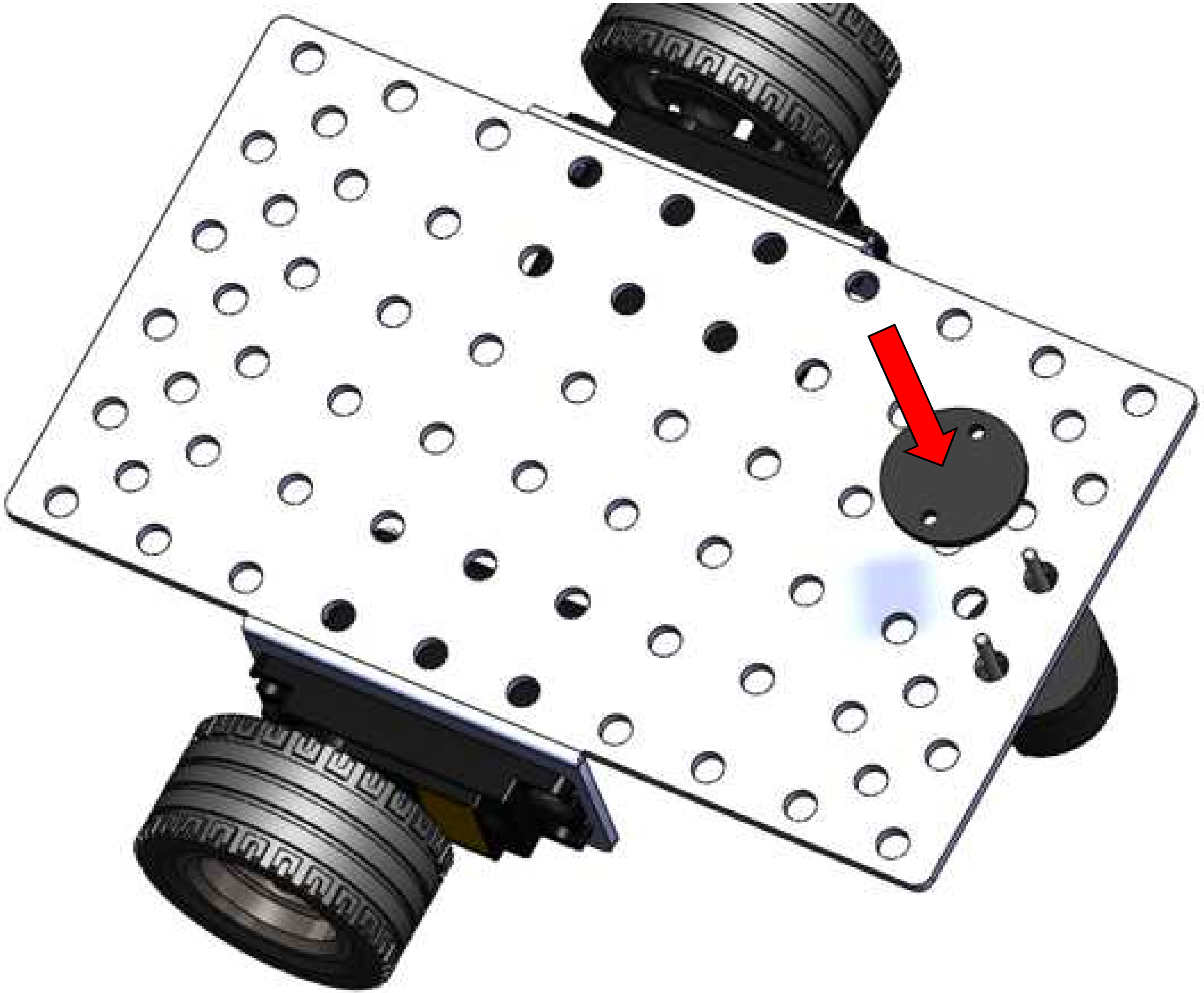
Take the ball bearing caster and the 2 long bolts from the "Pololu Ball Caster" bag. **Pop the ball out of the caster**. Leave the ball out for now. Put the bolts through the caster, the thick washer and the 3 hole LEGO as shown. The assembly is shown here for position only (it will be attached in later slides).

KISS INSTITUTE FOR PRACTICAL ROBOTICS

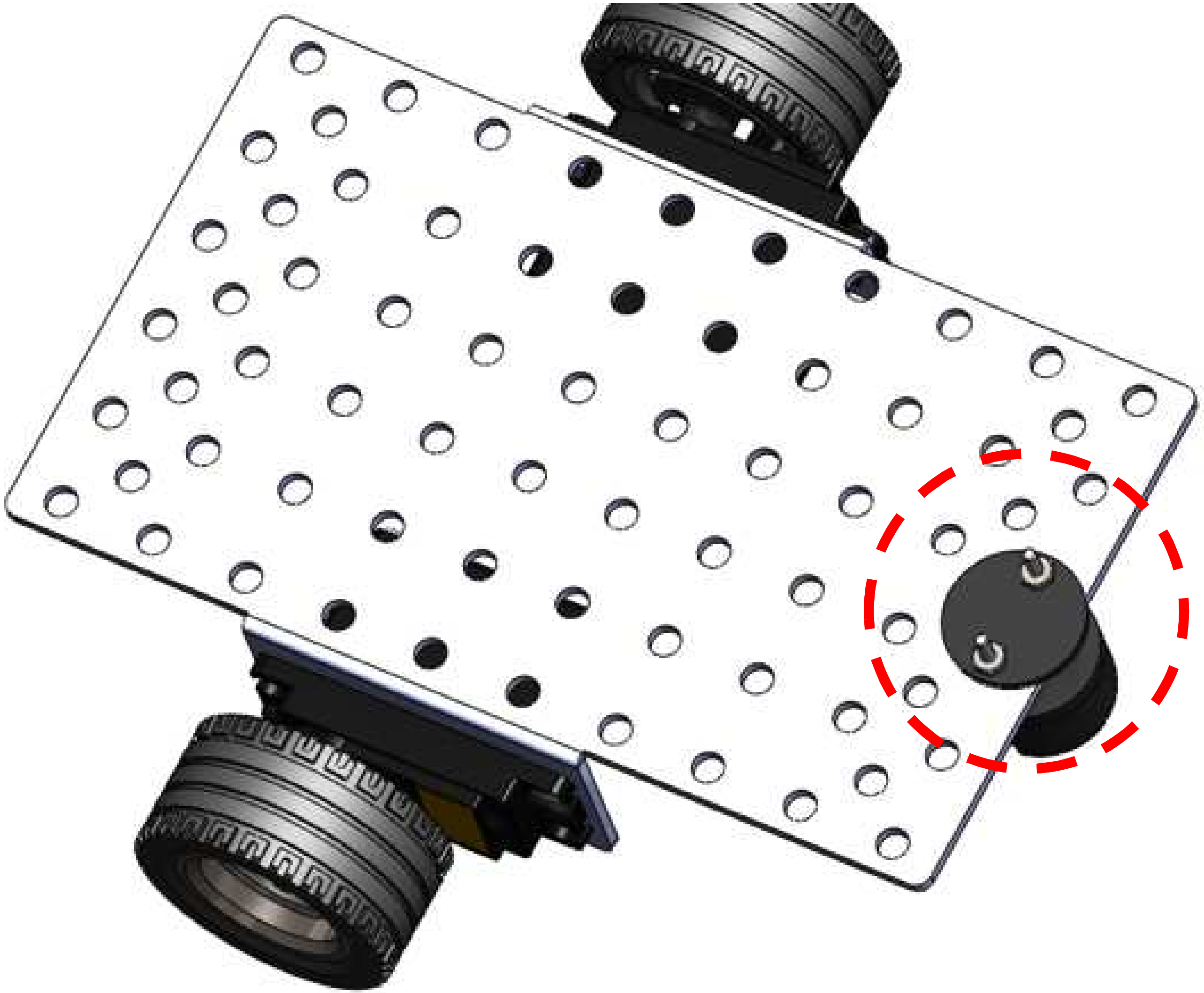


Place everything in place (3 Hole LEGO, Thick Washer, Ball Bearing Caster with 2 long bolts) into position. You will have to hold this in place until it is secured in the next slide. You can have a partner help or you can rest the caster assembly on the table so that it stays in place.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

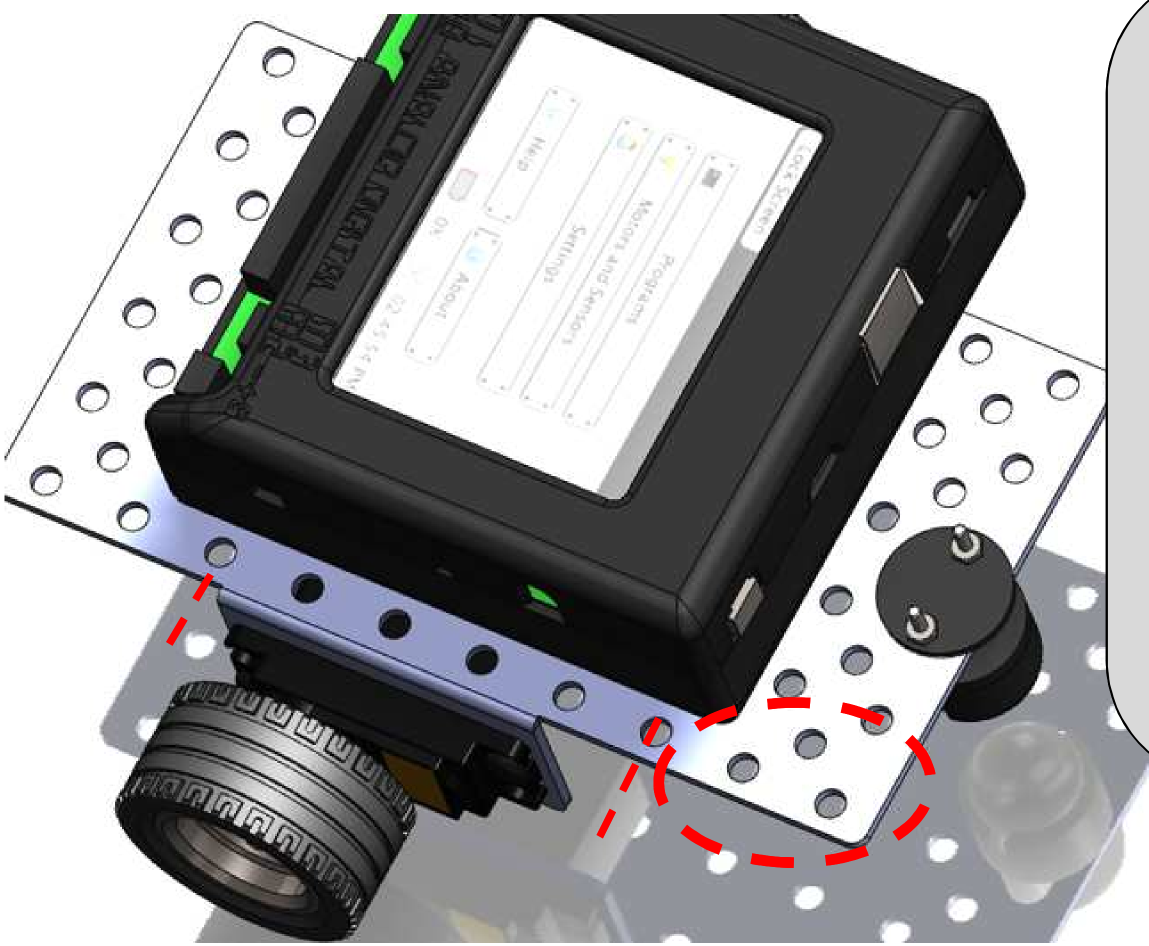


Take the thin washer from “Pololu Ball Caster” bag and place over bolts.



Take nuts out of “Pololu Ball Caster” bag and secure caster assembly to the chassis. Tighten with the screwdriver from the other side.
Pop the ball back into the caster.

KISS INSTITUTE FOR PRACTICAL ROBOTICS



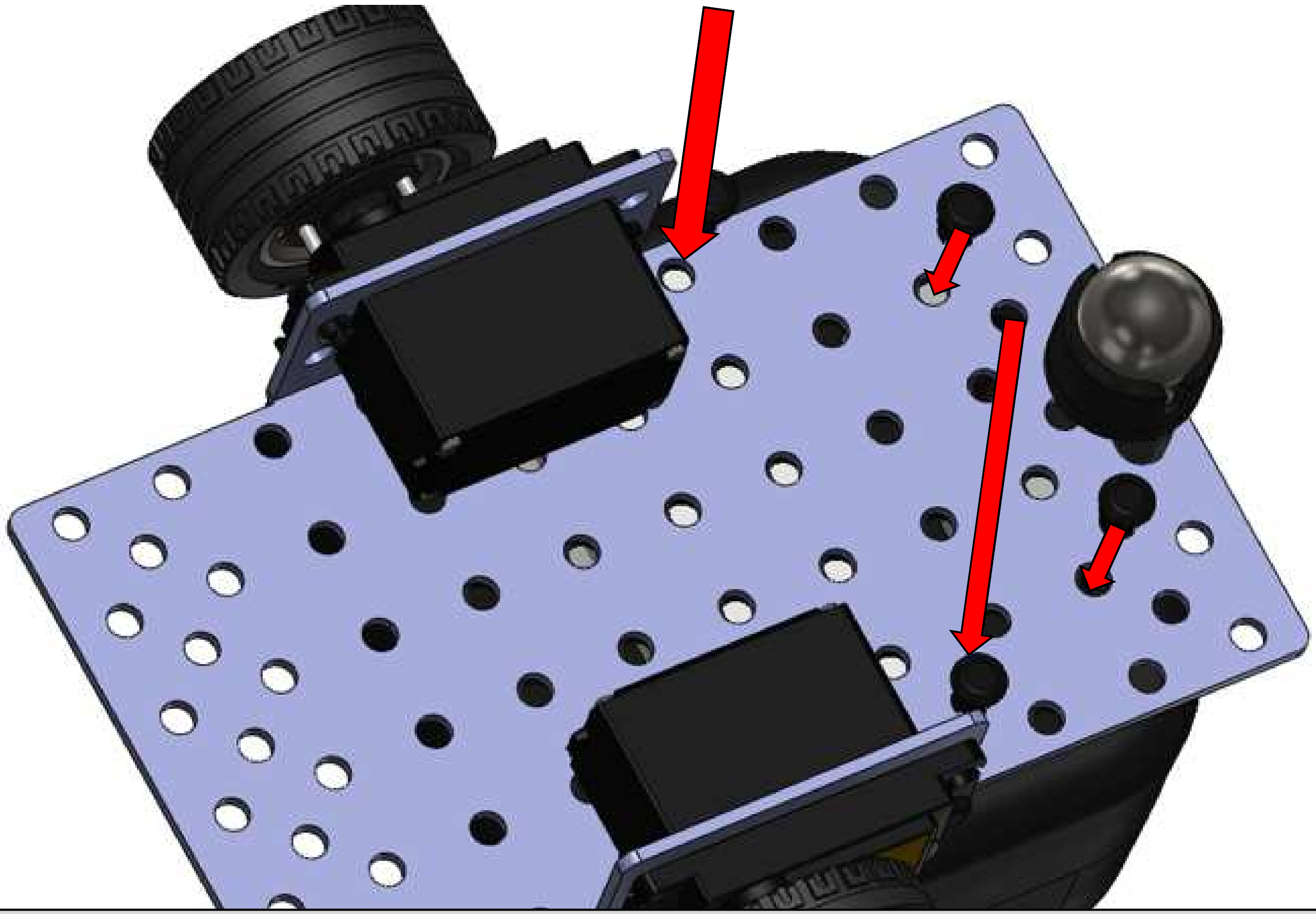
Option one- Set your KIPR Link onto the chassis in the orientation shown. Power switch located opposite from the ball bearing caster. Leave three holes uncovered on the chassis on the caster side.

There are two options for attaching your KIPR Link to the chassis. Option one attaches with push rivets and option two (slides at the end of the presentation) allows you to simply set your KIPR Link on the chassis so that you can quickly change out the KIPR Link if you need to.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

CAREFULLY turn the chassis/KIPR Link upside down. REMEMBER it is NOT attached and will fall. You can hold it with your hand until we attach it in the next step or CAREFULLY set it upside down on the table.

KISS INSTITUTE FOR PRACTICAL ROBOTICS



Take Four (4) plastic pop rivets, identical to the ones you used to attach the motors to the chassis and attach the KIPR Link to the chassis by lining up the holes (as shown) and pushing them in until they lock. Once you have the KIPR Link locked into place you can turn your robot over and let it rest on the wheels and caster.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

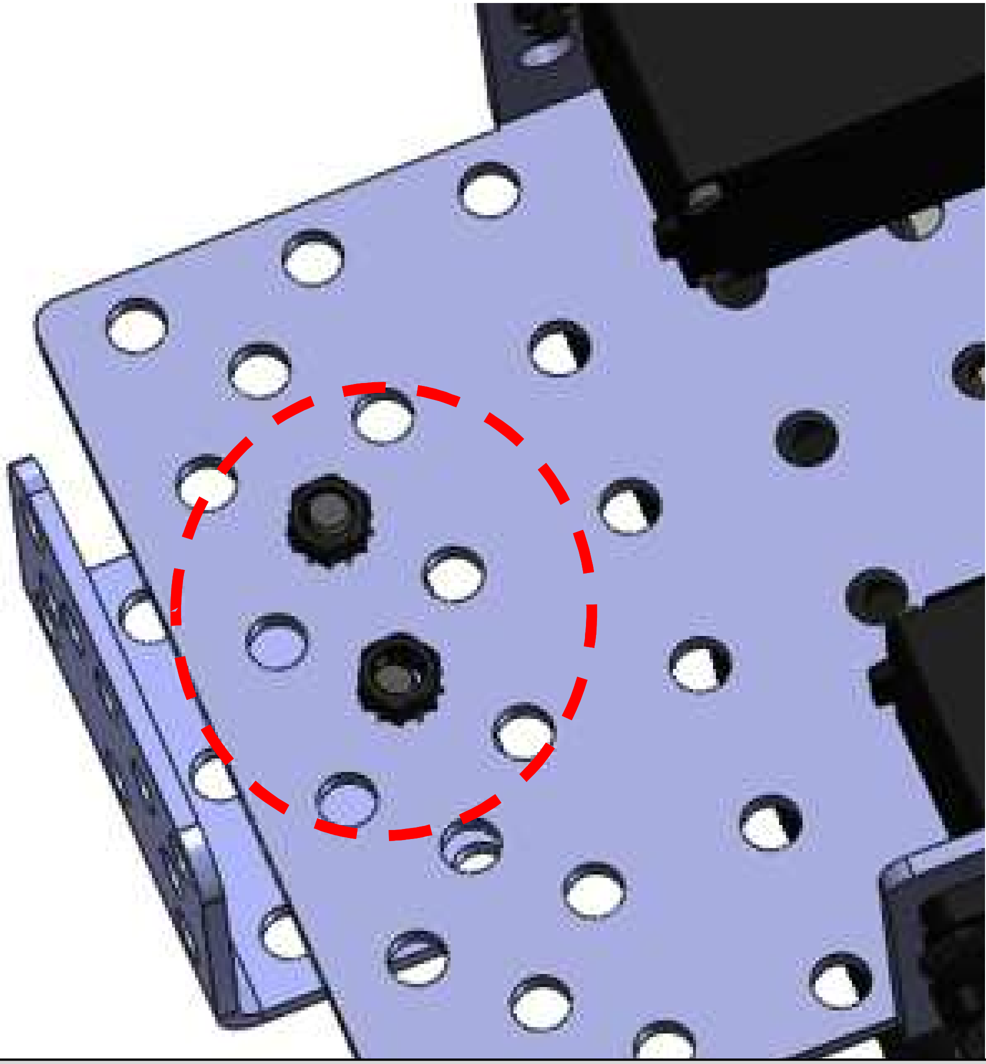
Take a KMP (KIPR Metal Part) Angle Bracket piece out of the "KIPR Metal Pieces Box". You can identify it by the hole spacings that look like a smiley face. It is shown here for identification and will be attached soon.

This is to show position only, it will be attached with bolts and nuts in the next slide. NOTICE the alignment on the chassis and the holes in the angle bracket that will be used. IT WILL NOT be straight when properly aligned, but this is okay.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Place two (2) short black bolts as shown. They are in a ziplock bag with wrenches and offsets in the **GREEN** paper insert labeled “Electronics Kit”. There are 3 lengths of bolts (short, medium and long) and two types of nuts (locking with white nylon and locking with attached lock washer). You will need the locking with lock washer nuts in the next step.

KISS INSTITUTE FOR PRACTICAL ROBOTICS



Using the lock nuts attach the angle bracket to the chassis and tighten using the wrench and screwdriver. This is shown upside down for clarity. You do not have to turn your robot upside down to attach the nuts.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

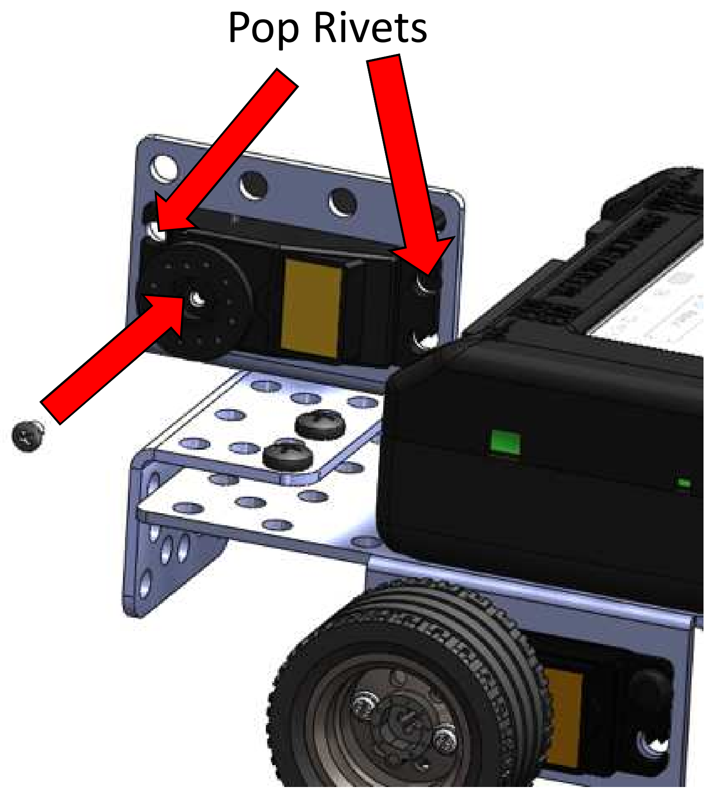
Identify your KMP (KIPR Metal Part) Motor Bracket.
It is shown here for identification and will be attached in the next slide.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Attach the motor bracket to the chassis using the short bolts and lock washer nuts. Tighten with the wrench and screwdriver.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

You will need one (1) servo.
You can distinguish servos from motors by the wiring.
Motors have a double grey wire and servos have a **triple red, orange, brown wire**. You will also need one small round servo horn for the next step.



Place the round servo horn onto the servo just like you placed the servo horn onto the motor and secure with the small silver servo horn screw. Tighten with the screwdriver. You can attach the servo horn to the servo before attaching the servo to the servo bracket with two (2) plastic pop rivets. Secure the servo to the mount by pushing the pop rivets in until they snap into place.

You will need one (1) 11 hole LEGO piece
This will be attached to the servo horn in the next step it is shown here for identification ONLY.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

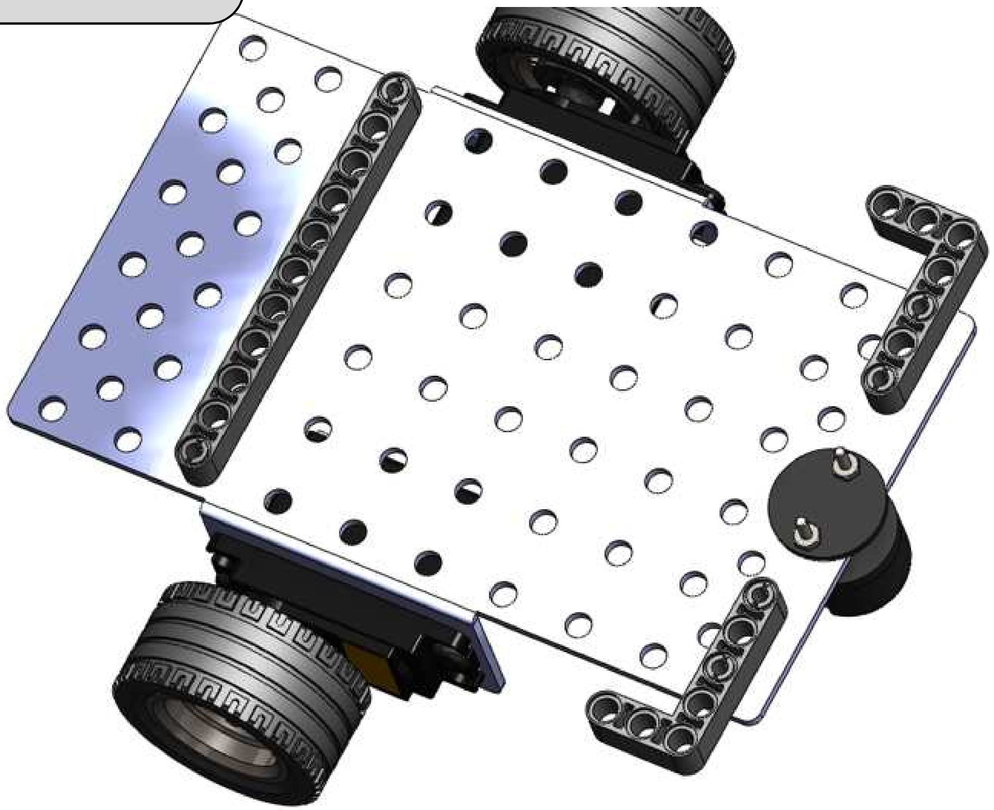
LEGO should be vertical
(Not as depicted here)

BEFORE attaching the LEGO to the servo horn, ROTATE the servo horn clockwise (turn towards KIPR Link) until it stops. Then ORIENT the LEGO piece so that it is pointing straight up/VERTICAL(not shown in this picture).
You will need two (2) of the longer silver screws found in the servo bags and using the screwdriver attach the 11 hole LEGO piece to the servo horn. MAKE SURE the LEGO piece is vertical.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

OPTION 2

Attach the LEGO pieces to the chassis as shown



Option 2 uses ¾ friction pins and the Lego pieces. Remember this just keeps the KIPR Link from sliding off the chassis and allows easy change of the KIPR Link, BUT the KIPR Link will fall off if it is turned upside down.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Get to know your controller

TEACHER'S RESOURCE

KIPR Link Basic Features
GNU/Linux based operating system
Open-source robot control software
Integrated color vision system
800MHz ARMv5te processor
Spartan-6 FPGA
Integrated battery and charge system
Internal speaker
320 x 240 color touch screen

Input and Output
1 - 3 axis 10-bit accelerometer (software selectable 2/4/8g)
8 – Digital I/O ports (hardware selectable 3.3V or 5V)
8 - 3.3V (5V tolerant) 10-bit analog input ports
4 - Servo motor ports
4 - PID motors ports with full 10-bit back EMF and PID motor control
1 - 3.3V (5V tolerant) TTL serial port
2 - USB A host ports for connecting devices
1 - Micro USB port to connect to your computer
1 - Physical button
1 - IR emitter
1 - IR receiver
1 - HDMI port

You have the KIPR LINK Manual on the flash drive provided to you

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Charging the KIPR Link Controller

TEACHER'S RESOURCE

- For charging the KIPR Link, **use only the power supply which came with your Link**
  - **Damage to the Link from using the wrong charger is easily detected and will void your warranty!**
- The KIPR Link power pack is a lithium polymer battery so the rules for charging a lithium battery for any electronic device apply
  - Only an adult should charge the unit
  - You should NOT leave the unit unattended while charging
  - Charge away from any flammable materials and in a cool, open area

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Learning about the Link Controller

TEACHER'S RESOURCE

Goals

- To be able to identify all of the ports on the link controller and what they are used for
- To be able to identify the buttons and their use
- To understand the proper charging procedure (only an adult, only under supervision at all times, not around water or flammable materials)

Preparation

Have a link controller available for students to examine along with a projection of the resource slide with pictures of the controller OR give students a printed sheet of the resource slide

1. Print the table in the resources for students to use to identify and then learn the use of the items

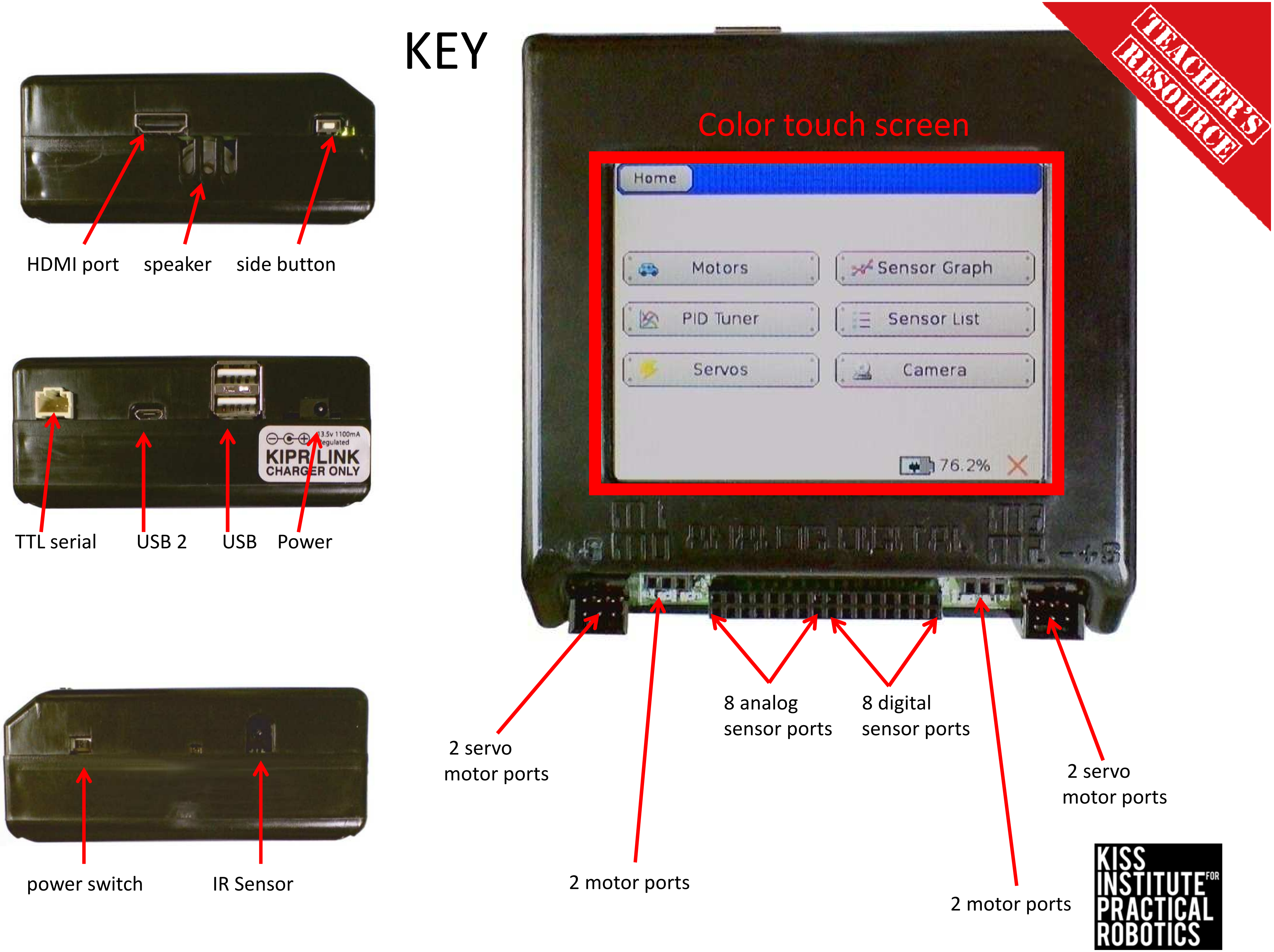
2. Have the students use their science/engineering/robotics notebook to make their own checklist/table to account for all ports and switches

Resources

The KIPR Link Manual (on your flash drive)

KISS INSTITUTE FOR PRACTICAL ROBOTICS

KEY



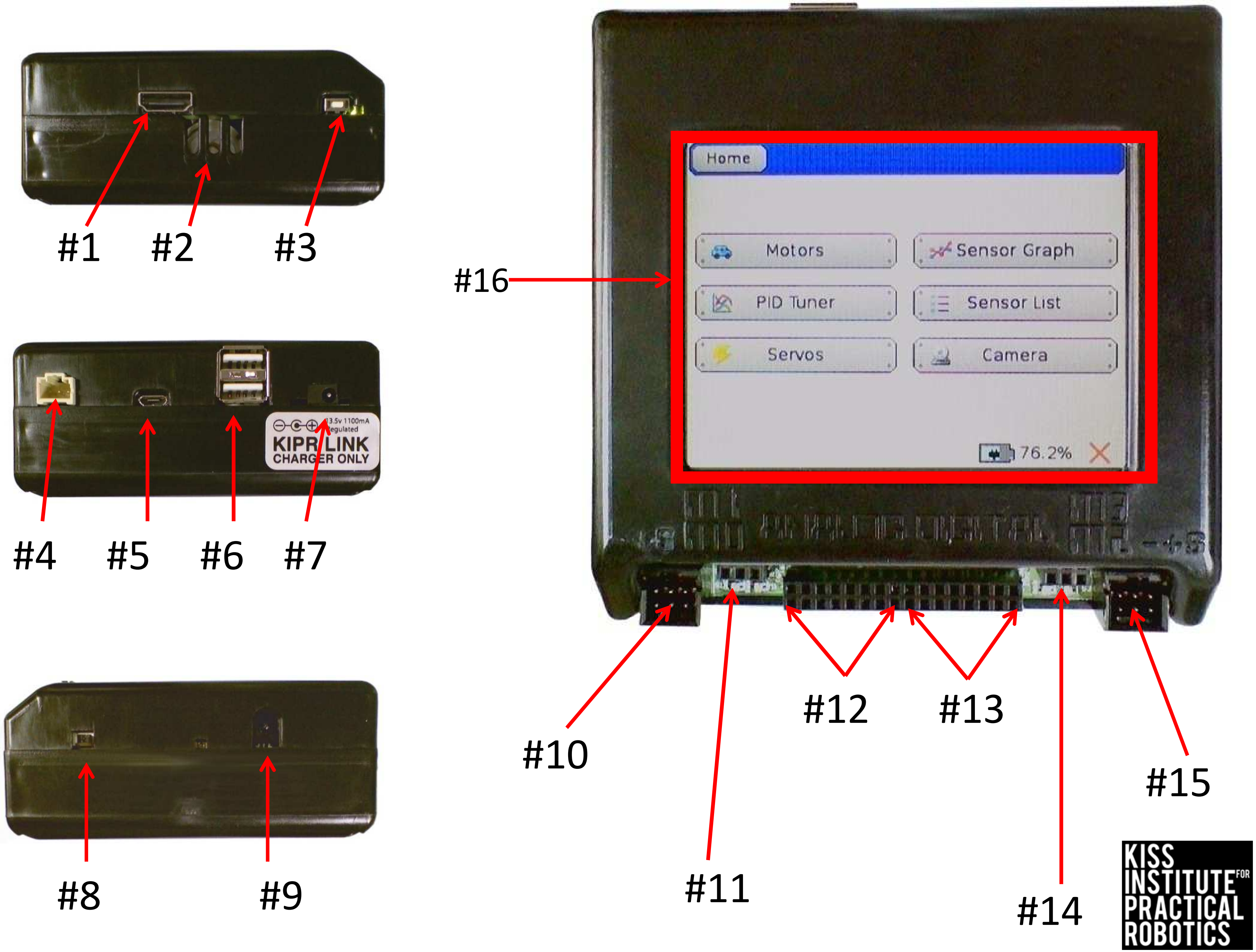
TEACHER'S RESOURCE
HDMI port
speaker
side button
TTL serial
USB 2
USB
Power
KIPR LINK
CHARGER ONLY
power switch
IR Sensor
Color touch screen
Home
Motors
Sensor Graph
PID Tuner
Sensor List
Servos
Camera
76.2%
2 servo motor ports
2 motor ports
8 analog sensor ports
8 digital sensor ports
2 motor ports
2 servo motor ports
KISS INSTITUTE FOR PRACTICAL ROBOTICS

KEY

TEACHER'S RESOURCE

1	A	HDMI Port	J	Where you plug in SERVO motors
2	B	Speaker	K	Where you plug in motors
3	C	Side Button	M	Where you plug in digital sensors
4	D	TTL Serial	L	Where you plug in analog sensors
5	E	USB 2	N	Where you can interact with the controller
6	F	USB	H	Used to turn the Link on and off
7	G	Power	B	Used to play sounds
8	H	Power Switch	I	Used to emit and receive Infrared
9	I	IR Sensor	E	Used to download programs from the computer to the link
10, 15	J	Servo Ports	D	Used to connect to the iRobot Create Platform
11, 14	K	Motor Ports	A	Used to connect to a display
12	L	Analog Sensor Ports	G	Used to charge the Link FOLLOW PROPER PROCEDURES
13	M	Digital Sensor Ports	C	Used for human input (used in porgramming)
16	N	Color Touch Screen	F	Used for flash drives, keyboards, mouse

KISS INSTITUTE FOR PRACTICAL ROBOTICS



#1
#2
#3
#4
#5
#6
#7
3.5v 1100mA
Regulated
KIPR LINK
CHARGER ONLY
#8
#9
#16
Home
Motors
Sensor Graph
PID Tuner
Sensor List
Servos
Camera
76.2%
#10
#11
#12
#13
#14
#15
KISS INSTITUTE FOR PRACTICAL ROBOTICS

Get to know your KIPR Link

Item # goes here ↓

Letter in front of Item ↓

	A	HDMI Port		Where you plug in SERVO motors
	B	Speaker		Where you plug in motors
	C	Side Button		Where you plug in digital sensors
	D	TTL Serial		Where you plug in analog sensors
	E	USB 2		Where you can interact with the controller
	F	USB		Used to turn the Link on and off
	G	Power		Used to play sounds
	H	Power Switch		Used to emit and receive Infrared
	I	IR Sensor		Used to download programs from the computer to the link
	J	Servo Ports		Used to connect to the iRobot Create Platform
	K	Motor Ports		Used to connect to a display
	L	Analog Sensor Ports		Used to charge the Link FOLLOW PROPER PROCEDURES
	M	Digital Sensor Ports		Used for human input (used in porgramming)
	N	Color Touch Screen		Used for flash drives, keyboards, mouse

KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Know your Robot Controller

- Using a Link controller OR the print out OR a Link controller projected on the screen use the table provided by your teacher to identify the items by number
- Now match the items with their proper use (use the letter in front of the item)

Be the Robot Activity

Goals

- To help students understand the importance of specific directions
- To facilitate the student's understanding of pre-thinking the logic of providing directions to the "robot"

Preparation

Set up the room so that a blindfolded "robot" student can move around with out getting hurt.

1. Arrange the room so that there are some open areas and a few obstacles.

Activity

Explain the task to be completed by the blindfolded human robot (must start here go around the desk and stop at the white board, etc.).

- Make sure they must go around some obstacles, make a few turns and end at a specific location, maybe back to where they started
- Blindfold the student robot or simply have the student close their eyes. Put them in the "starting box" and have the other students provide directions to complete the task
  - One student at a time should give only **1 direction at a time**. For example; move forward 3 steps, stop, turn right, stop
- Document the instructions the students provide to the "student robot"
  - Use the documented instructions to write out the steps
  - Using documented steps, they can analyze it for success one step at a time. This is a great whole group activity
- Complete the Flow Chart activity, which introduces the concept of flow charts instead of written steps

TEACHER'S RESOURCE

KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Be the Robot Activity

1. Your teacher will explain the task that your human “robot” must complete.

1. Select a volunteer student to be the human “robot”.

1. Select students who will call out instructions to the robot.
   - Only one student at a time can provide directions, make sure to take turns in the proper order
   - Only one direction at a time may be provided to the “robot”

2. Run the “robot” by providing the directions and see how successful you were at controlling the robot.
   - Discuss was it easy or hard to make the robot complete the task
   - Could you do this with less instructions?

3. After brainstorming ideas to make the instructions better, write the steps out one at a time and in the proper order on the whiteboard, chalkboard or in your notebook.

4. Select another “robot” volunteer and read the written directions one at a time to see if they work any better.

5. Analyze for improvements in the written instructions and repeat.

6. Discuss what made for better instructions.

7. Move on to the next activity, Flowcharts!

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Be the Robot Flowchart Activity

TEACHER'S RESOURCE

Goals

- To help students understand how to construct a flowchart
- To introduce students to the concept of decision making "logic"
- To facilitate the student's understanding of using flowcharts to pre-think and plan the logic of how they program their robot
- To ensure that students can read a flowchart and equate it to actual robot behavior
- To ensure that students can look at robot behavior and equate it to a flow chart
- Use a flow chart to spot errors in logic (it didn't work, where is the problem?)

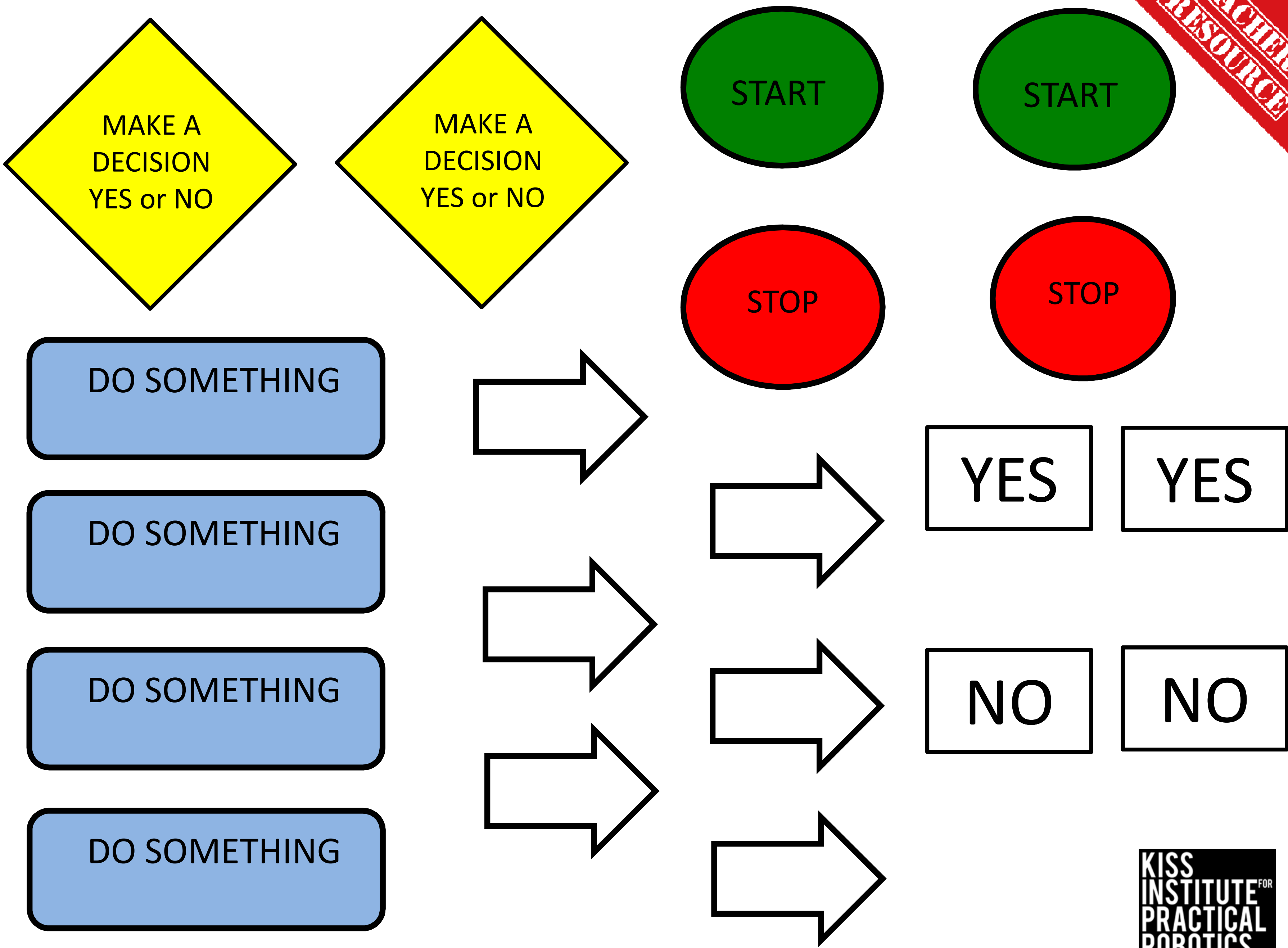
Preparation

1. Print the flowchart symbol (resource slide).
2. Cut out the flowchart symbols ahead of time (one set per group or one for the whole class) or print the sheet and have the students cut them out.
    - A magnet on the back makes them great to use on a whiteboard
    - You can make some oversized symbols if you plan this as a whole group activity

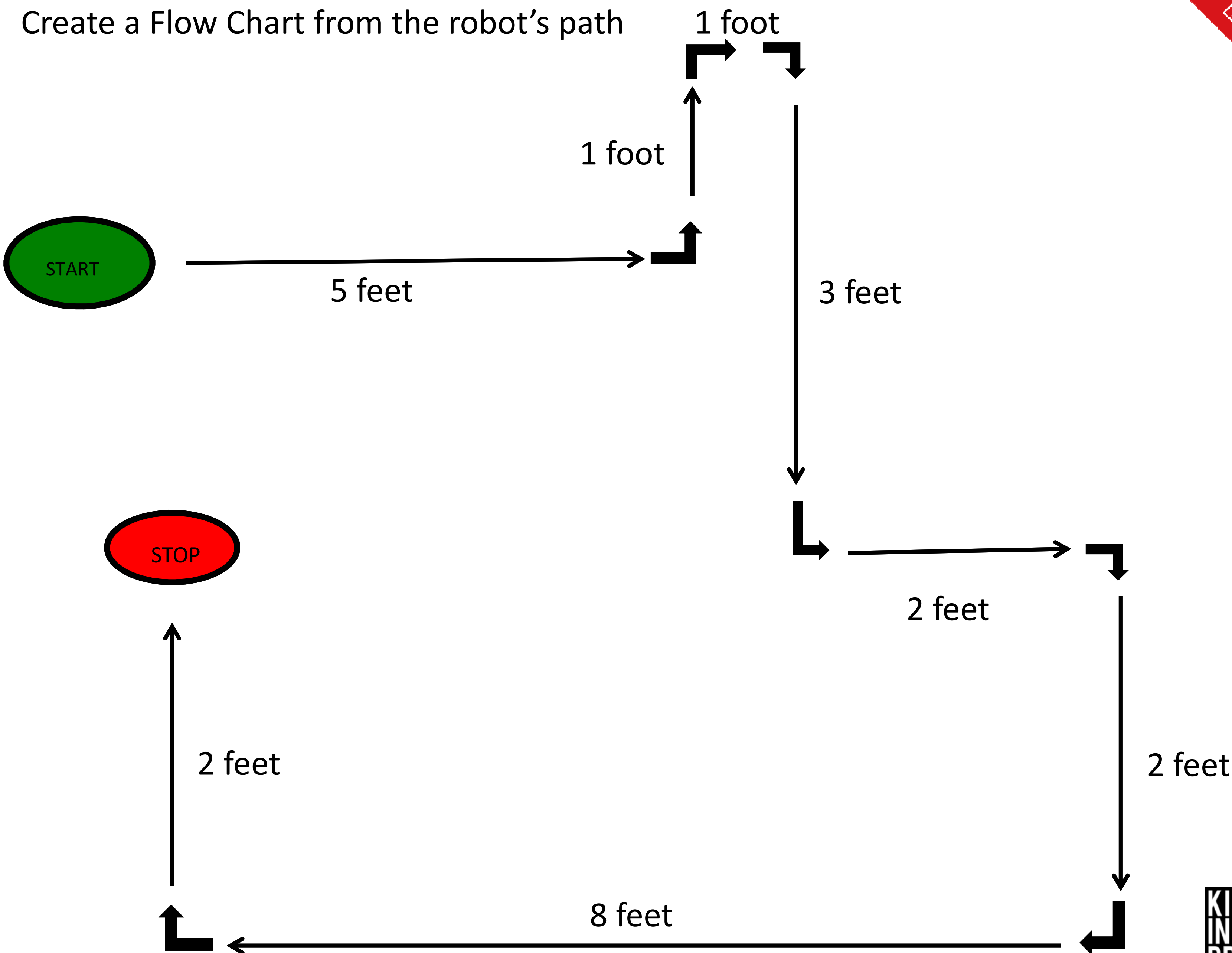
Activity

1. Have the students complete the following flowchart activity.
2. Using the cutouts make a flowchart and have the students draw out what they think the robot will do (pretend the robot leaves a mark on the floor/board with a marker as it moves around).
3. Using a reference robot path (one is provided in resources), print the sheet for each group or draw it on the board and have the students work backwards creating a flowchart from the actual path.

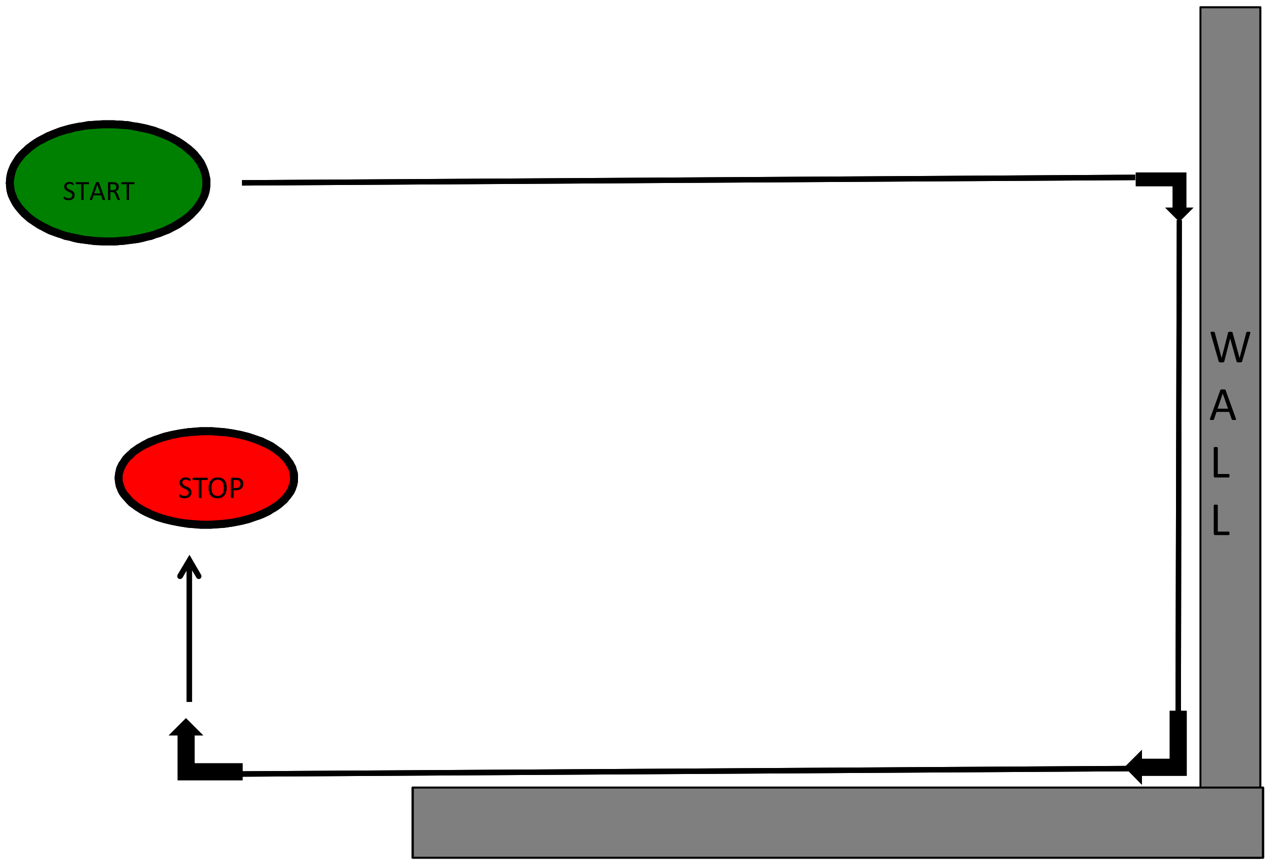
KISS INSTITUTE FOR PRACTICAL ROBOTICS


MAKE A DECISION YES or NO
MAKE A DECISION YES or NO
START
START
STOP
STOP
DO SOMETHING
DO SOMETHING
DO SOMETHING
DO SOMETHING
YES
YES
NO
NO

Create a Flow Chart from the robot's path



Create a Flow Chart from the robot's path (decisions)

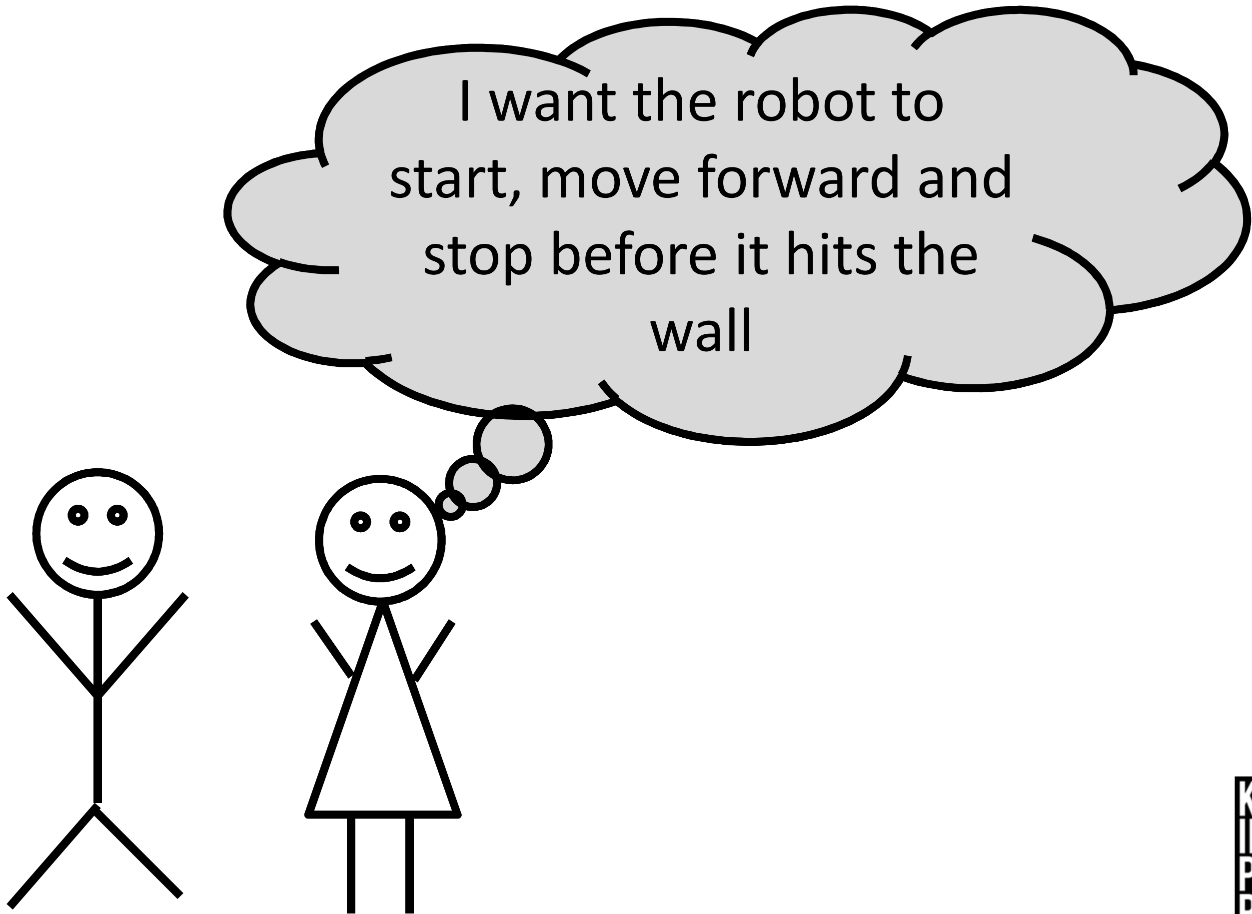


STUDENT ACTIVITY

Flow Chart Activity

Programming (telling your robot what to do)

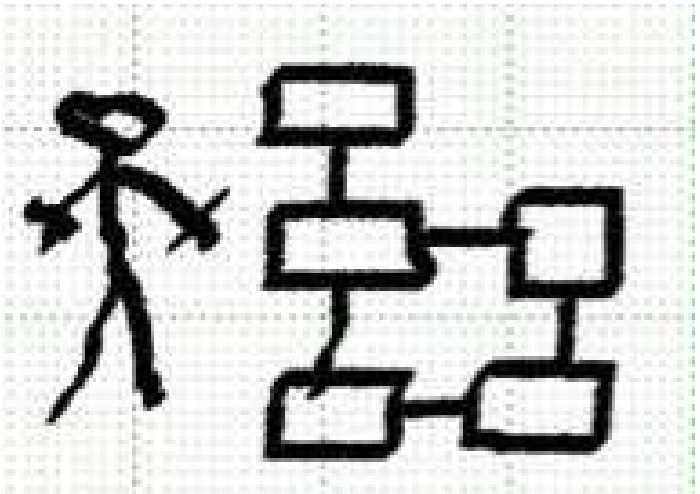
1. Decide what you want the robot to do.



KISS INSTITUTE FOR PRACTICAL ROBOTICS

Programming (telling your robot what to do)

2. Draw a DIAGRAM that shows the steps.

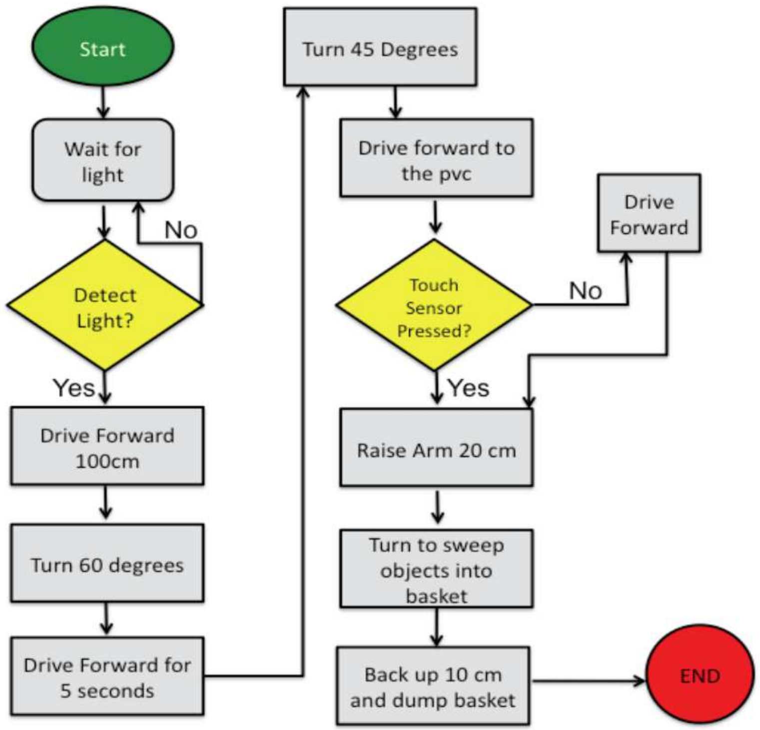


But not just any old DIAGRAM.....

KISS INSTITUTE FOR PRACTICAL ROBOTICS

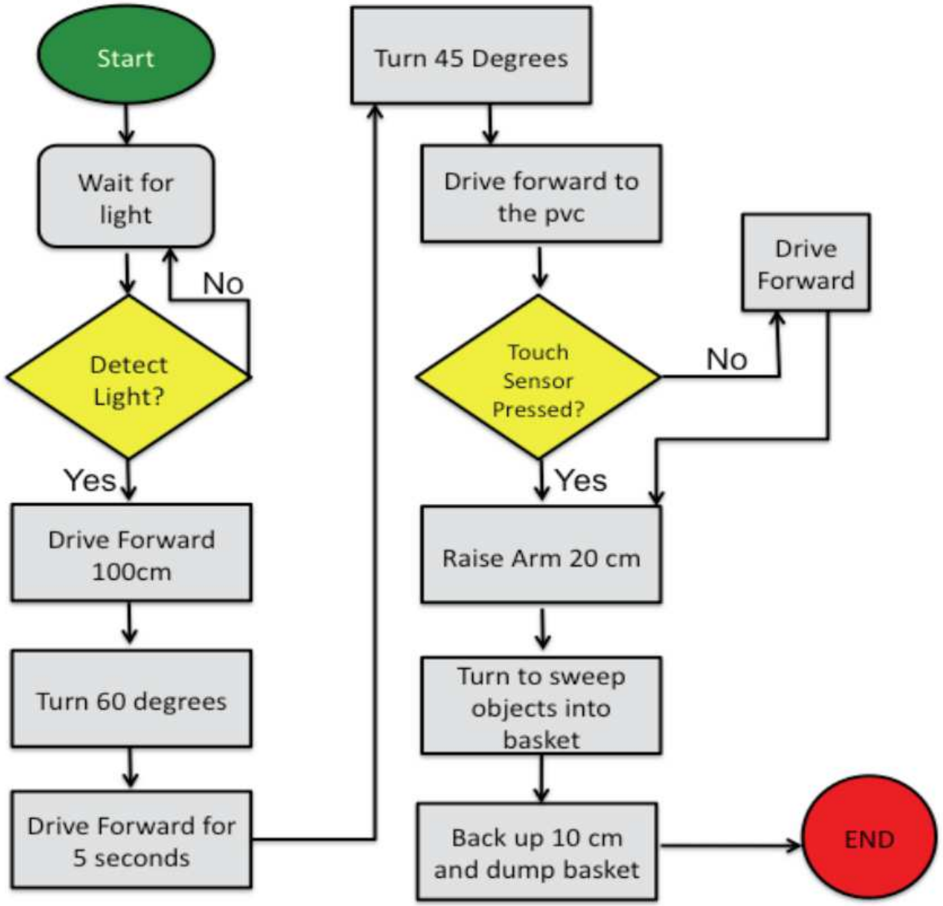
Programming (telling your robot what to do)

Computer Scientists & Engineers use a diagram of their program called a <u>FLOWCHART.</u>



Programming (telling your robot what to do)

Notice how they use different SHAPES and COLORS.



Programming (telling your robot what to do)

So that everyone will understand each other's flow charts everyone uses the same shape and color for certain actions.

Programming (telling your robot what to do)

START

The start and the stop are easy to understand: these are where the instructions "program" starts and stops

STOP

MAKE A DECISION (YES or NO)

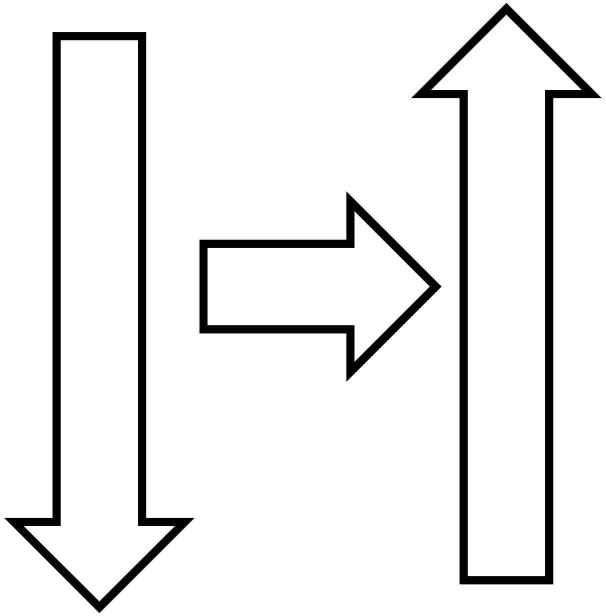
Yellow diamonds are always a choice or decision (you must have at least two choices after a yellow diamond)

DO SOMETHING

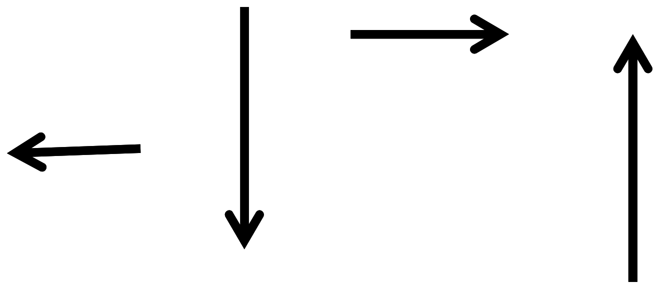
What the robot is programmed to do

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Programming (telling your robot what to do)



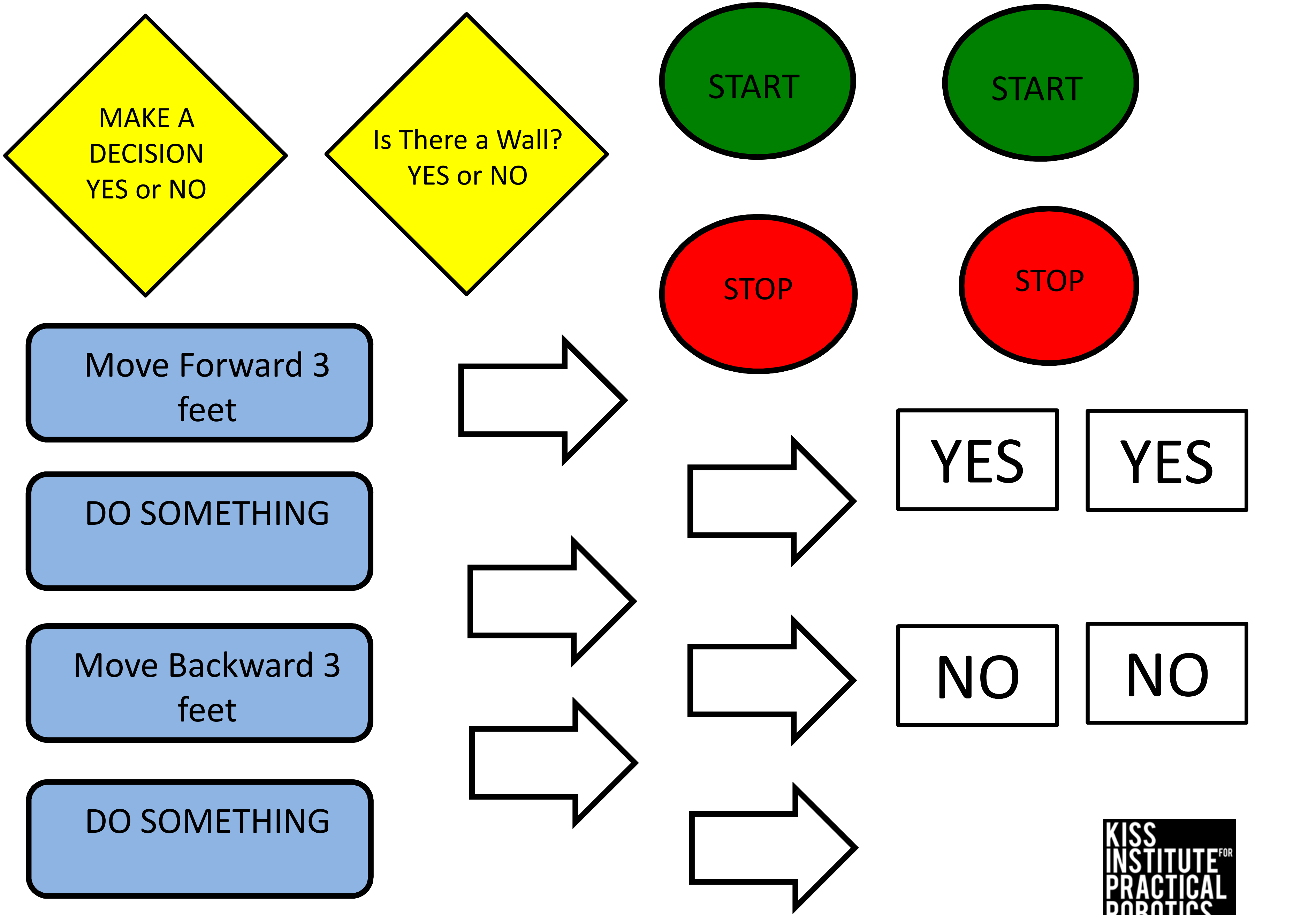
The arrows show the direction or flow of the instructions "program".



That's why it is called a flowchart!

KISS INSTITUTE FOR PRACTICAL ROBOTICS

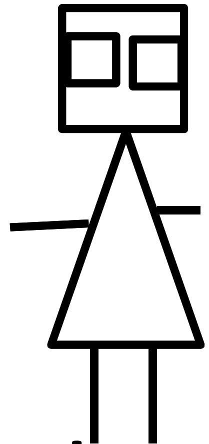
Now, cut out your shapes and sort them



KISS INSTITUTE FOR PRACTICAL ROBOTICS

Programming (telling your robot what to do)

3. Use your cut outs to make a flow chart of a program that:

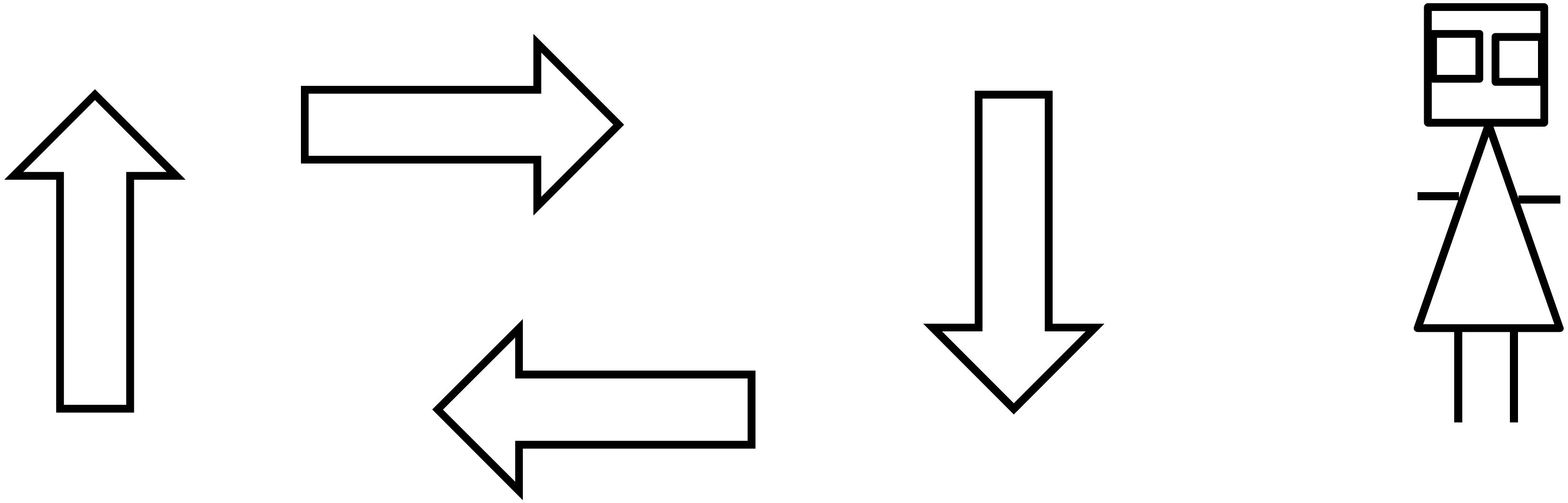


1. Has your robot START.
2. Has your robot move forward 3 feet.
3. Has your robot STOP.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Programming (telling your robot what to do)

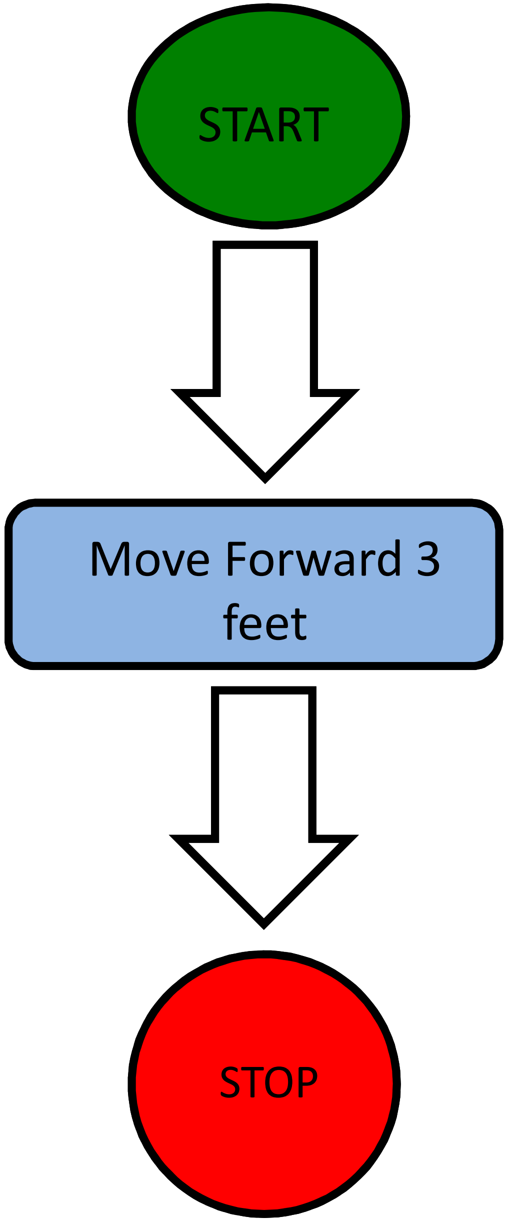
Did you remember the arrows?



Arrows are very important in a FLOWChart.....

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Programming (telling your robot what to do)



Starts the program "instructions"…

…then it…

…moves forward 3 feet…

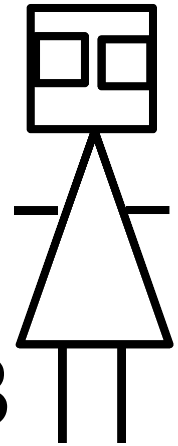
…then it…

…stops the program "instructions"…

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Programming (telling your robot what to do)

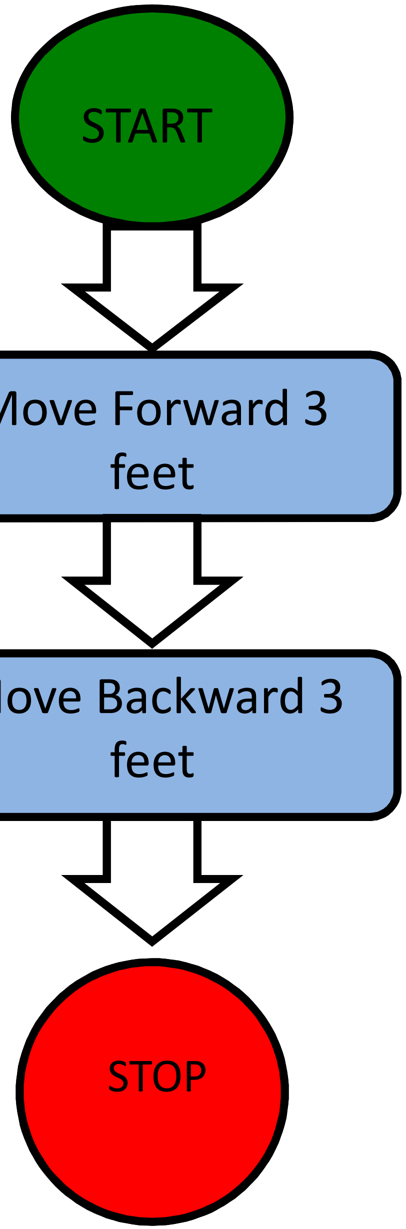
5. Now make a flowchart of a program that:



1. Has your robot START.
2. Has your robot move forward 3 feet.
3. Has your robot move backward 3 feet.
4. Has your robot STOP.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

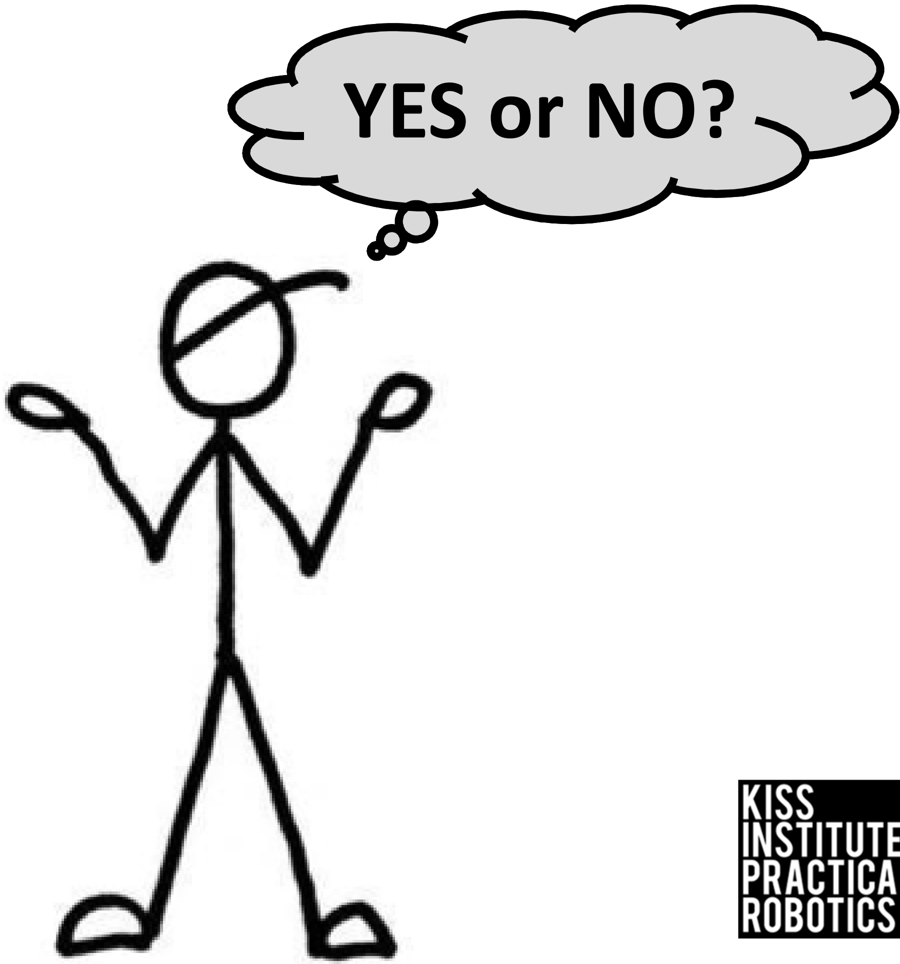
Programming (telling your robot what to do)



KISS INSTITUTE FOR PRACTICAL ROBOTICS

Programming (telling your robot what to do)

4. Time to make a decision.

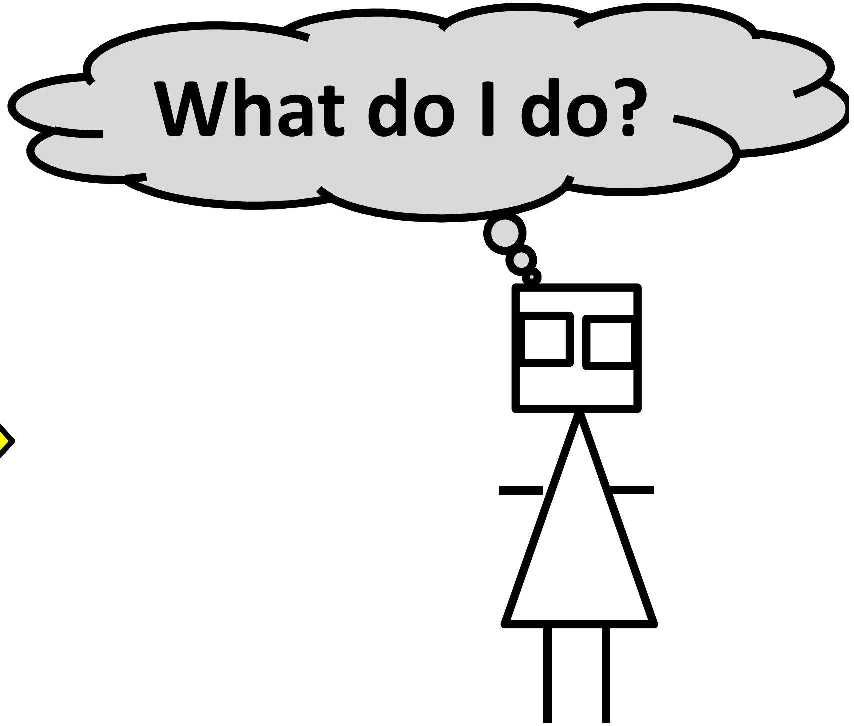


KISS INSTITUTE FOR PRACTICAL ROBOTICS

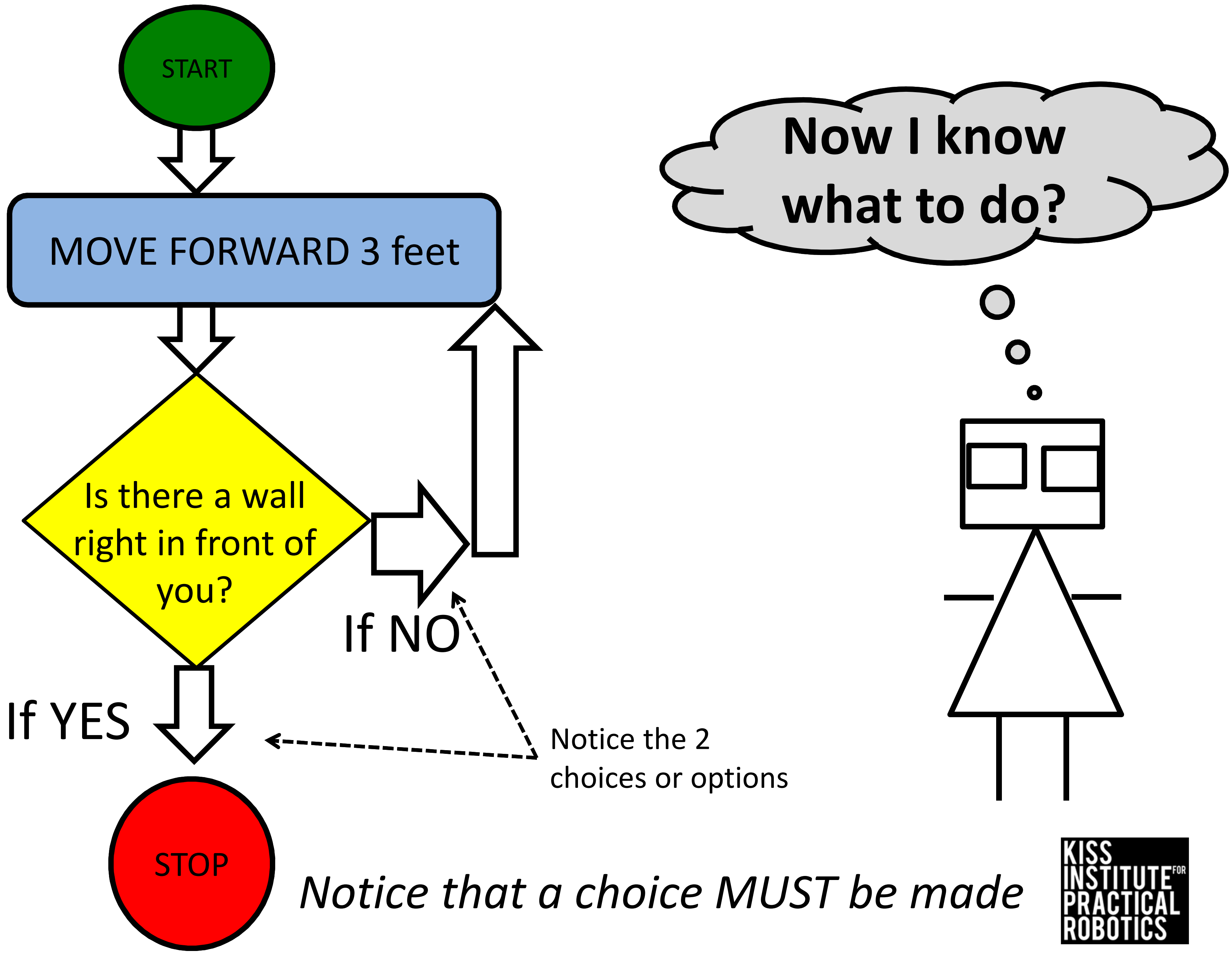
Programming (telling your robot what to do)

The decision will change what the robot does.

Is there a wall right in front of you?



KISS INSTITUTE FOR PRACTICAL ROBOTICS

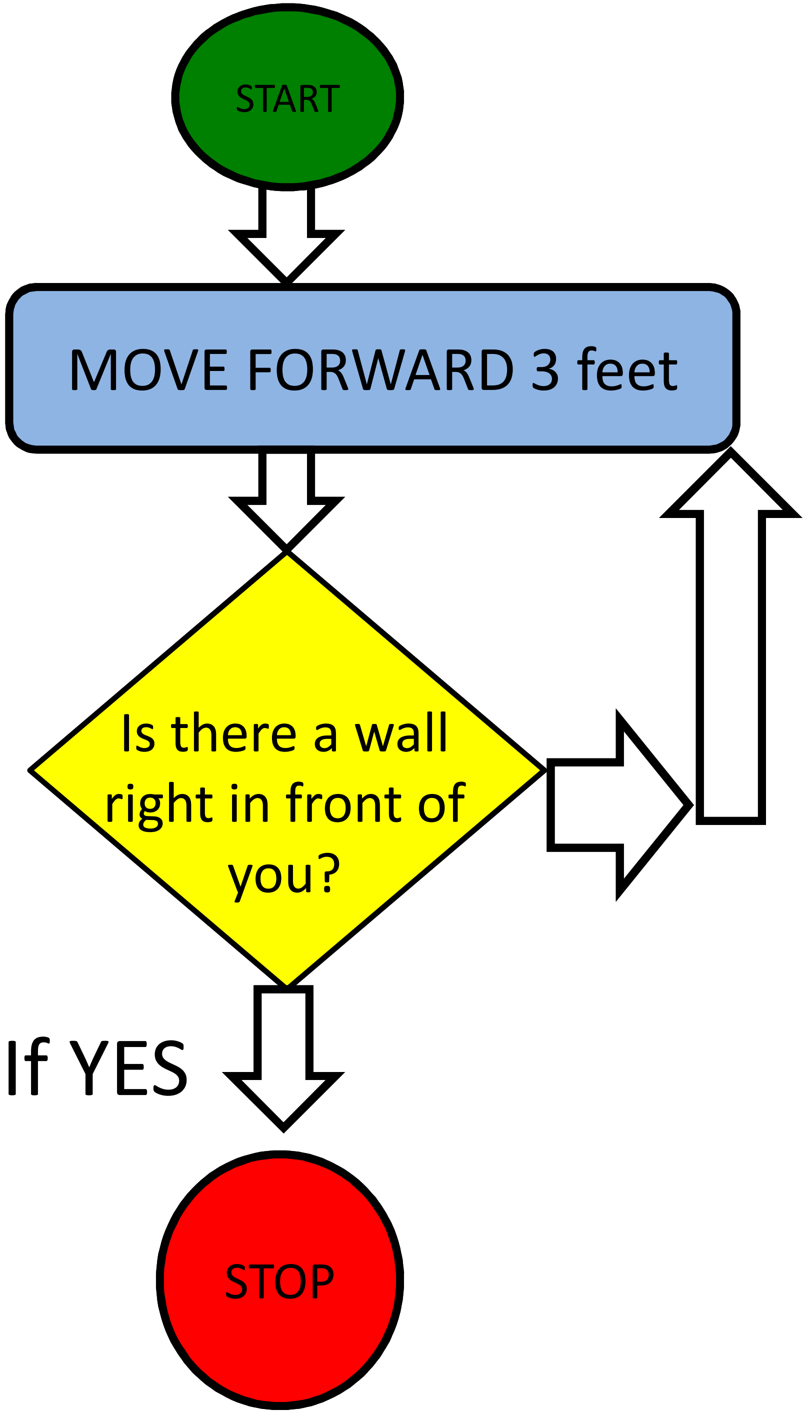

START
MOVE FORWARD 3 feet
Is there a wall right in front of you?
If NO
If YES
STOP
Notice the 2 choices or options
Now I know what to do?
Notice that a choice MUST be made
KISS INSTITUTE FOR PRACTICAL ROBOTICS

Programming (telling your robot what to do)

6. Now make a flow chart of a program that:

1. Has your robot START.
2. Has your robot move forward 3 feet.
1. Has your robot detect a wall and make a decision. (YES or NO)
2. Has two choices for your robot.
3. Has your robot STOP.

KISS INSTITUTE FOR PRACTICAL ROBOTICS



KISS INSTITUTE FOR PRACTICAL ROBOTICS

Programming (telling your robot what to do)

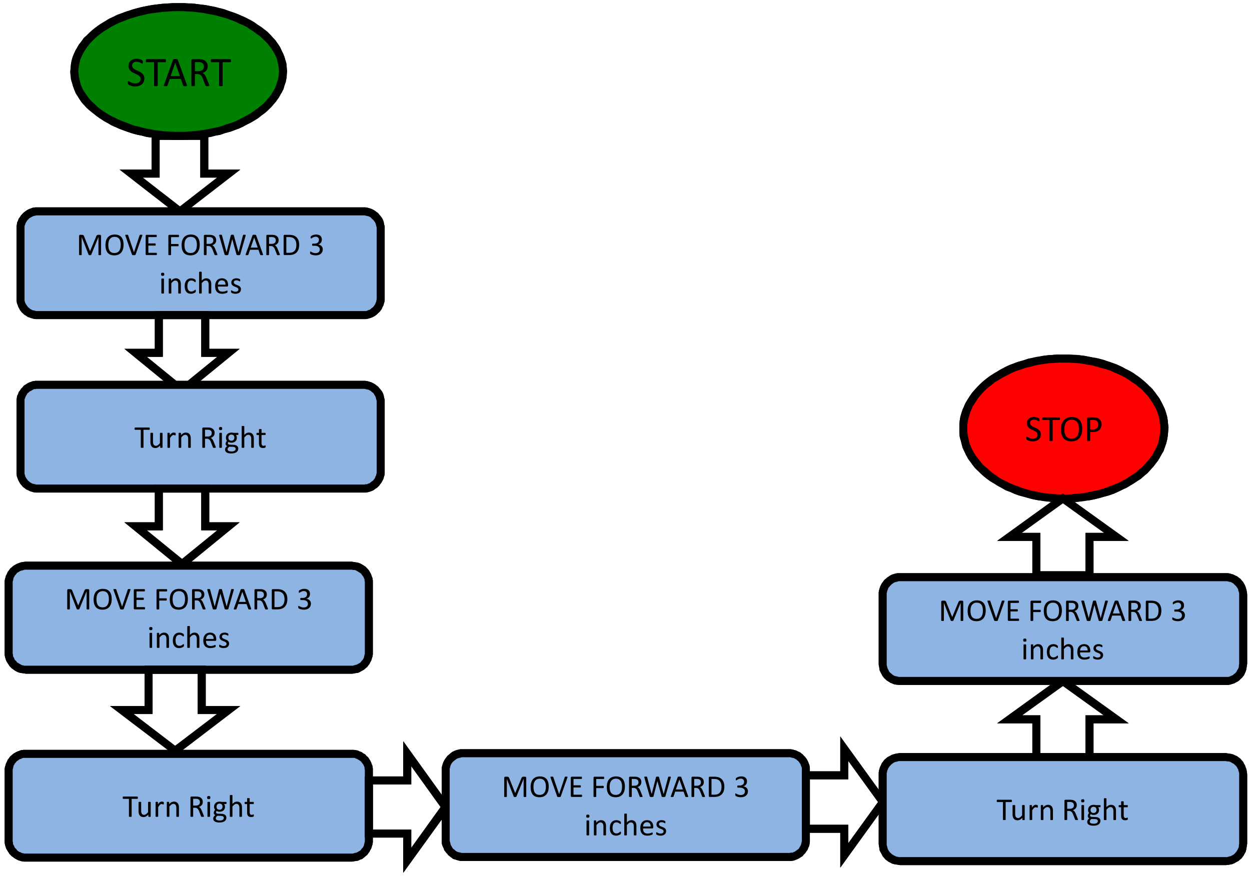
7. Now make a flowchart of a program of your own choice.

Include at least one decision.

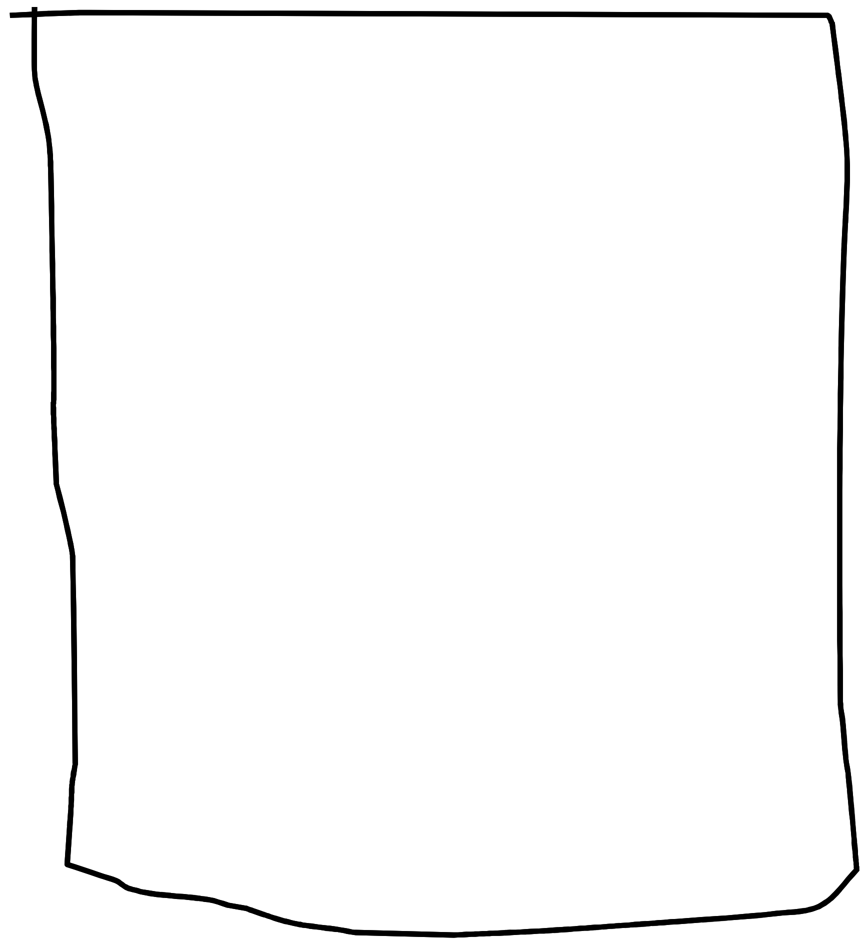
KISS INSTITUTE FOR PRACTICAL ROBOTICS

DRAW WHAT YOU THINK THE FLOW CHART WILL MAKE THE ROBOT DO

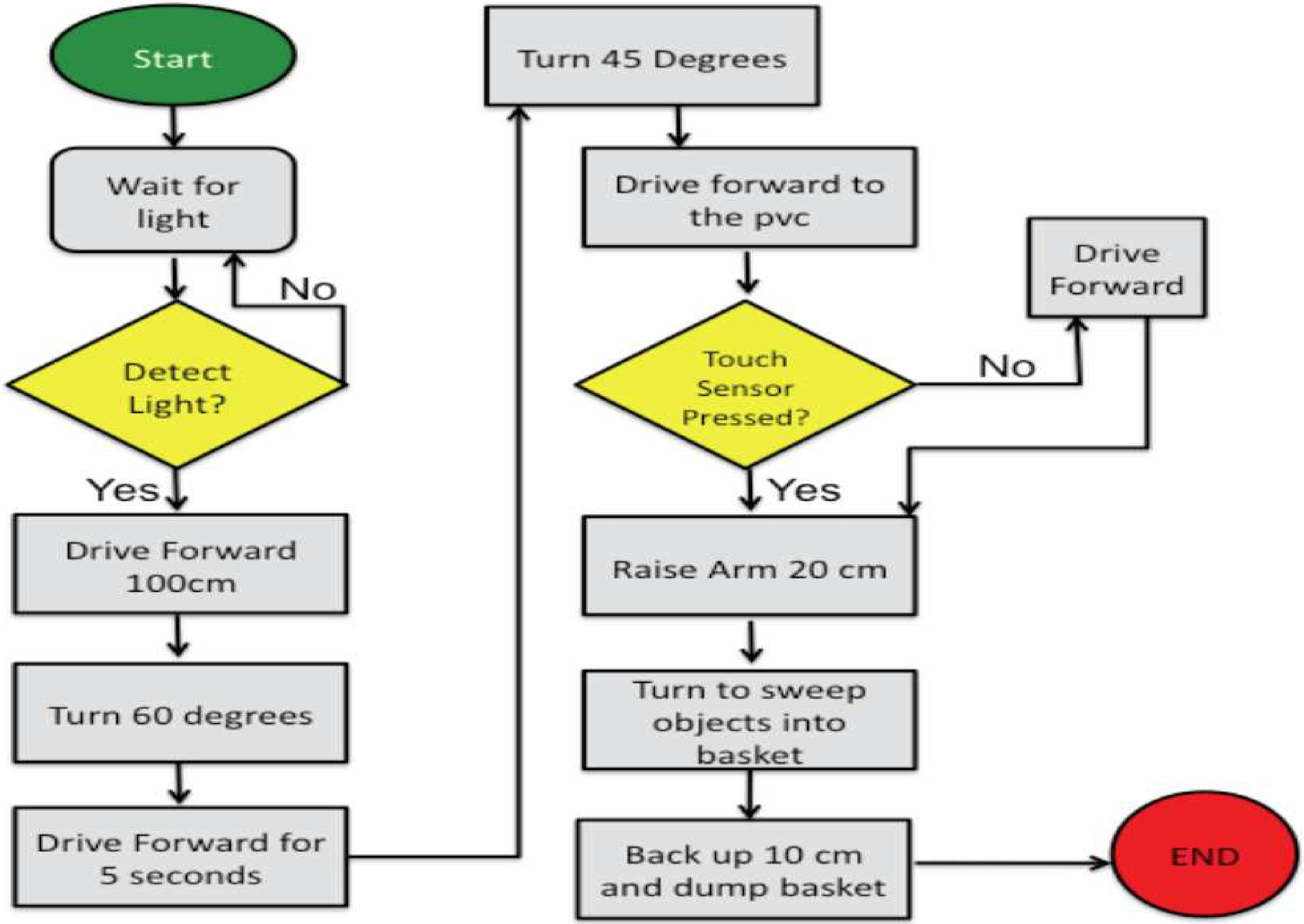
1. In your notebook or on the board draw the path you think your robot will follow from the following flow chart.



Did you get a square?

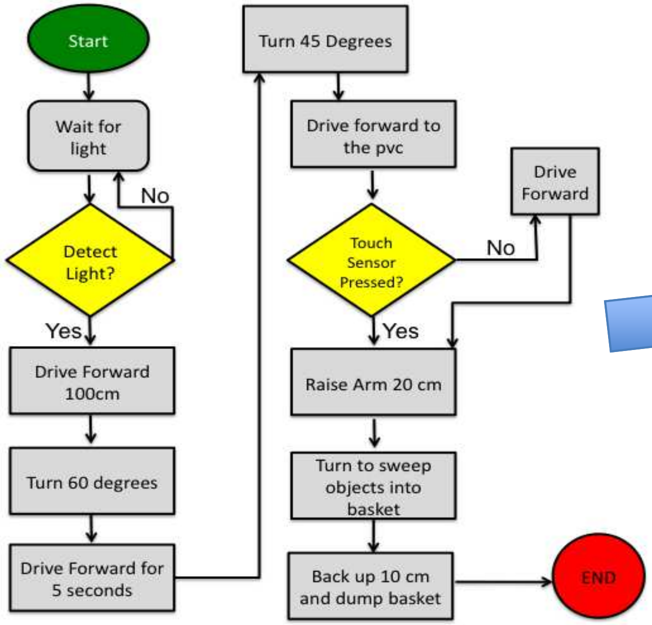


Explain what the robot is doing



KISS INSTITUTE FOR PRACTICAL ROBOTICS

Flowcharts are great for humans, but



Computers don't understand or speak this type of language so we need to write (program) the instructions in a language that computers can use

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Introduction to Programming Languages

TEACHER'S RESOURCE

Goals

- To help students understand the terms; IDE, compiler, source code and programming language

Preparation

- Prepare word/term cards (index cards, etc.) for a word wall (sample in resource slide) or have the students write the terms in their notebooks
  - **Machine Language** (what the computers understand- Bytes)
  - **Executes** (in terms of a computer running or executing 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)

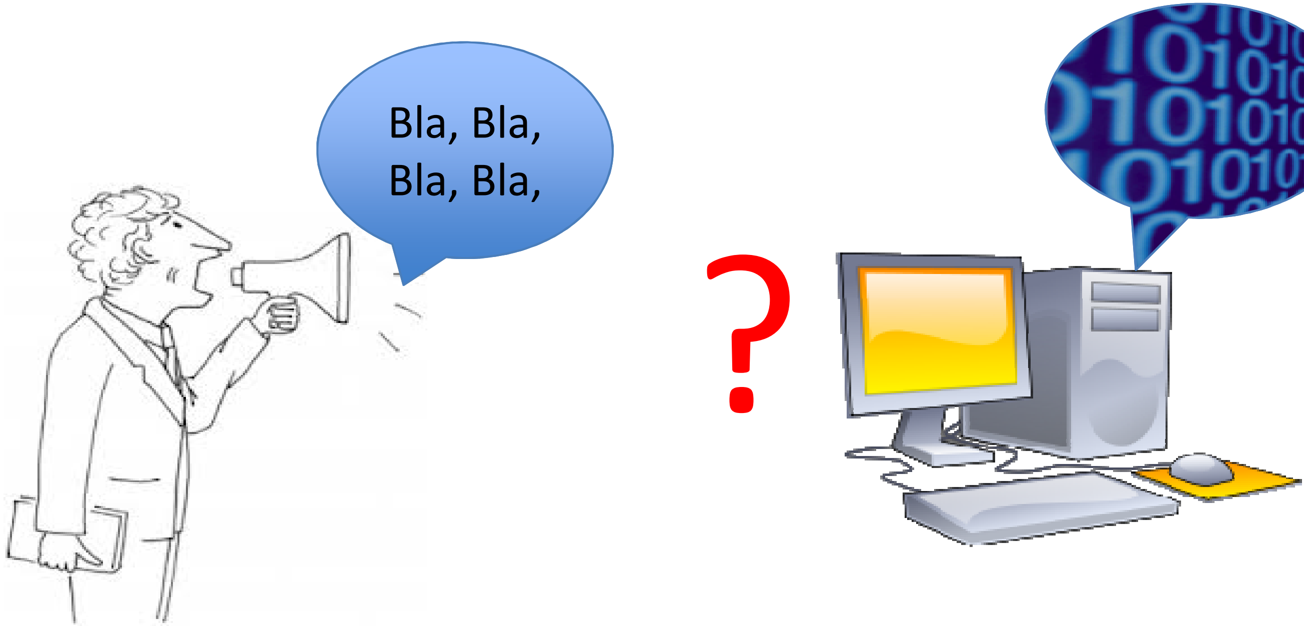
Activity

- After reviewing and discussing the slides have the students generate and agree on the definitions/uses of the term
  - Word Wall
  - Write in their notebooks
  - With a partner match up the word card with the correct definition card

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Machine Language	Programming Language	The language (bytes) that computers understand	
Executes	**C, C++, Java, Python**	A computer ________ a program when it runs the program	
Source Code	Translated from a programming language to a machine language	Language humans can understand that can be turned into machine language	
Compiled	Name for code written in a programming language	Strange names of some of the often used programming languages	

Programming Languages

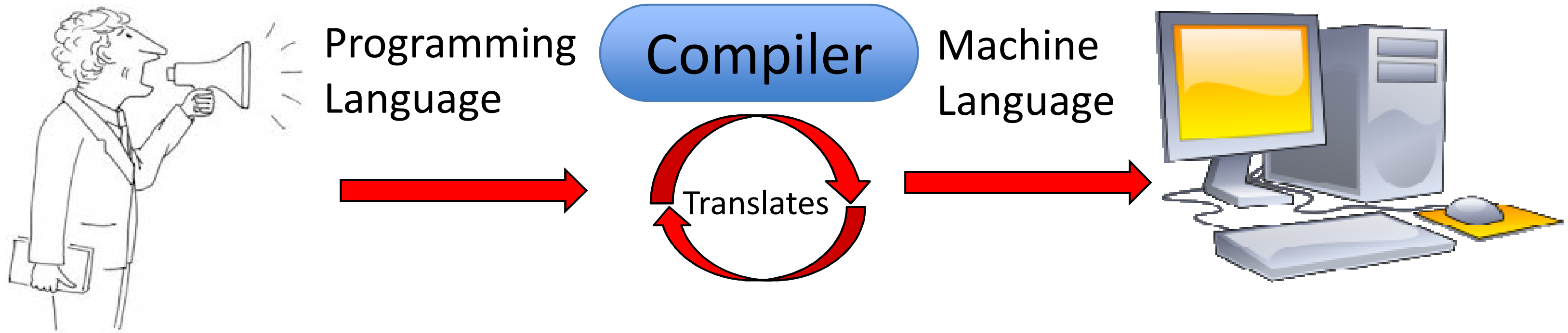


Computers only understand machine language (stream of bytes), which they can then read and execute (run).

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

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Why not use an interpreter?



Humans have created languages with funny names like; C, C++, JAVA, Python, that allow them to write "source code" which they can understand and edit.

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

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Programming Vocabulary Activity

- Using the cards you cut out or your teacher provided for you match up the terms with their definition

Introduction to Programming Languages

TEACHER'S RESOURCE

Goals

- To help students understand that languages have rules/conventions you must follow
  - Sentences start with a capital and end with a punctuation mark, etc.
- Understand how programs such as Microsoft Word or other applications help you follow the rules/conventions of a program
- Understand that the KISS IDE (Integrated Development Environment) is like a word processing program that helps them follow the rules/conventions while they write their source code

- **New Vocabulary word is Debug, meaning checking your program for errors and fixing them much like when Microsoft Word underlines misspelled words or grammatical errors**

Activity

- Have students list things the computer program/application Microsoft Word helps them with when they are writing.
  - In their notebooks
  - Whole class activity on the board

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Rules you use when writing

STUDENT ACTIVITY

- List as many rules as you can think of you must follow when writing a report in school
  - Once you have everything listed go to the next slide and add anything you didn't have on your list

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Rules/Conventions when writing

These are things that are always done in a language.

For example:

- When writing a sentence you always start with a capital letter
- Complete sentences should end with a period.
- Spaces are used to create paragraphs, which are used to separate out ideas
- The order of words make a difference
    - *The ran horse slowly* as compared to *the horse ran slowly*
- Math has conventions as well
    - Order of operations (1+2) X 4

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Programming Languages

Computer languages are like any other language: they take a little time and effort to learn all of the rules/conventions.

The more you practice the better you will get!

KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

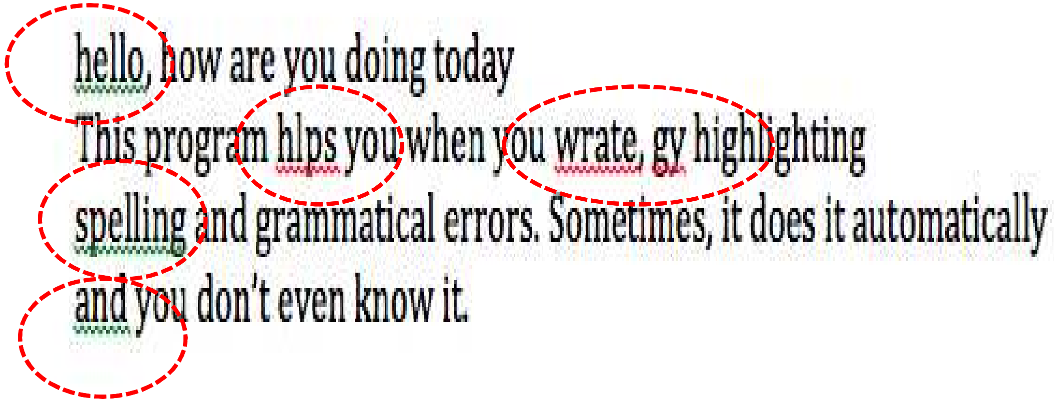
How does the Microsoft Word program help? Activity

Have you ever used a word processing program such as Microsoft Word?

What does it help you do when you are typing something like a report?

- List all of the things the program helps you with

Word Processing programs highlight possible errors



hello, how are you doing today
This program hlps you when you wrate, gy highlighting
spelling and grammatical errors. Sometimes, it does it automatically
and you don't even know it.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Integrated Development Environments (IDE) help you with the rules/conventions of programming languages

These are software applications (Apps) that make it easy for you to write and edit your source code, debug it (look for mistakes) and compile (translate) it.

This is kind of like a word processing program that lets you write text, format it, spell check, etc.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

KISS IDE

To make it easier for you to learn and use the programming language we have the **KISS IDE**, which will allow you to develop source code with the C programming language

Introduction to a simple C program

Goals

- To help students understand the basic components of a C program
    - Main program, function, return, curly braces, terminating statements (;) and comments
- To help students understand what a function is

Preparation

- Students need to be able to see a program
    1. Project the "Hello World" onto a white board or screen so the whole class can see it.
    2. Print out the C program "Hello World" so that each group/student has one (resource on following slide).
    3. You can have them open the KISS IDE and select the "Hello World" template on their computer (instructions on starting KISS IDE in resources).
    4. Print out or project a more complicated program and have students find the main function, curly braces, semicolons and the return.
- You can use vocabulary cards for the new vocabulary words

Activity

Using either the print out, projection or actual KISS IDE on a computer

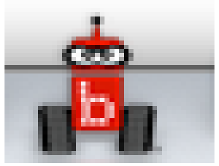
- Run through the program slides and have students identify the different parts of the program (they can circle them on a print out, point to them, etc.) on the simple "Hello World" and the more complicated program as well
- Students can draw a flow chart for each program

TEACHER'S RESOURCE

KISS INSTITUTE FOR PRACTICAL ROBOTICS

TEACHER'S RESOURCE

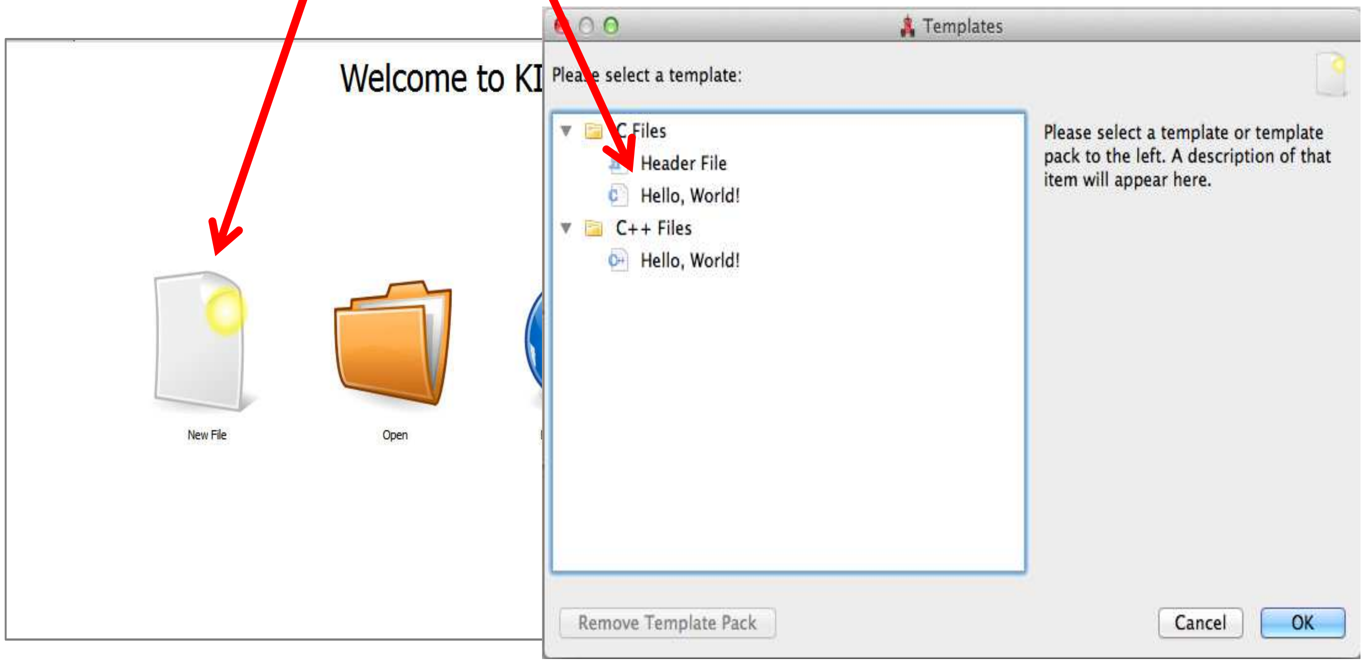
Launch the KISS IDE



KISS IDE icon

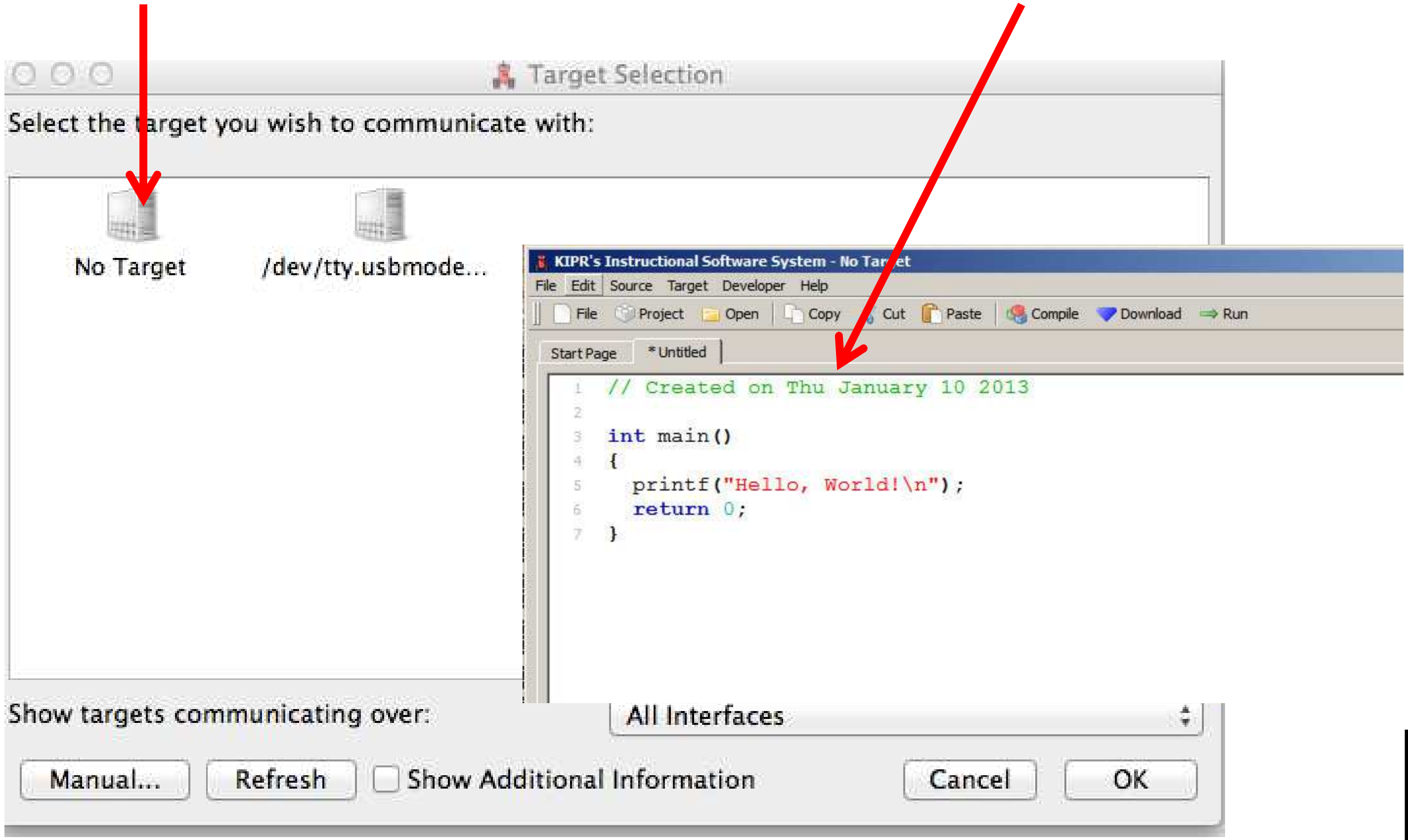
- Start the KISS IDE by clicking on its icon to get the welcome screen
- Click on the *New File* icon and

  and choose the C, *Hello, World!* template



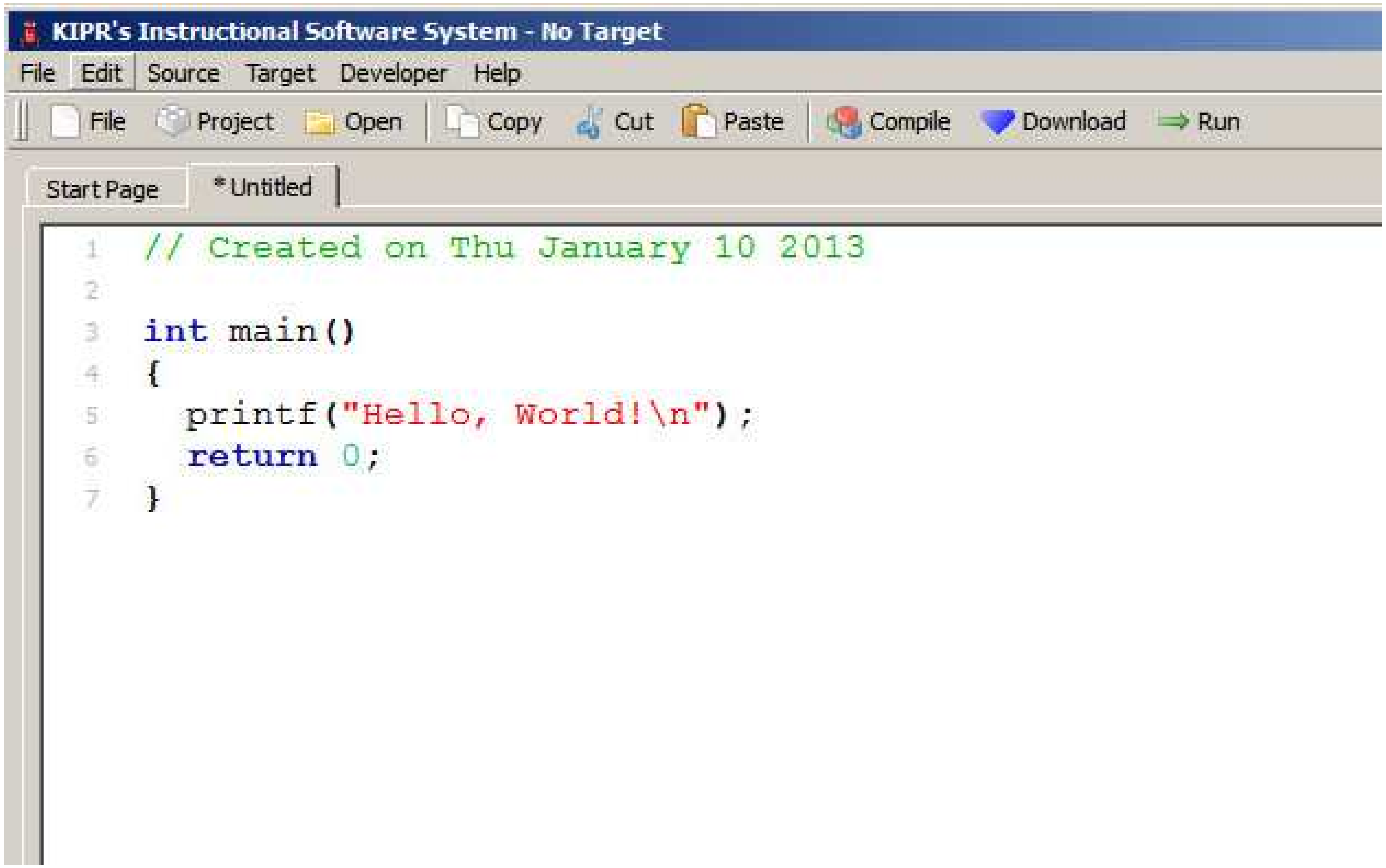
Select Target

- A Target Selection window will appear
- Pick *"No Target"* and the C program template will come up



TEACHER'S RESOURCE

The C Template: Hello, World!



KISS INSTITUTE FOR PRACTICAL ROBOTICS

TEACHER'S RESOURCE

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

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

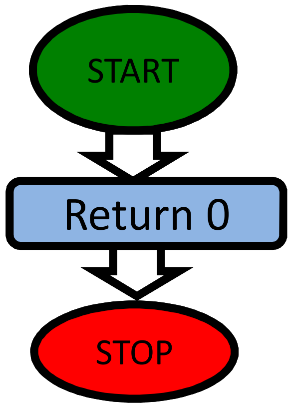
```
int main()
{
    printf("Hello, World!\n"); //print Hello World
    return 0;
}
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

```
int main()
{
  set_analog_pullup(7,0); // change port 7 to floating analog
  while(side_button()==0)//checks status of side button
  {
    if(analog10(7)>525)//read value in analog port 7
      {//move backwards
      motor(0,-1*(analog10(7)+250));
      motor(3,-1*(analog10(7)+250));
      }

    if (analog10(7)<475)//read value in analog port 7
      {//move forward
      motor(0,analog10(7)+250);
      motor(3,analog10(7)+250);
      }
    if(analog10(7)>=475 && analog10(4)<=575) //read value in analog port 7
      { //stop
      ao();
      }
  }
}
```

Computers read a program just like you read a book, they start at the top and go to the bottom. Computers read incredibly fast- 800 MILLION lines per second!

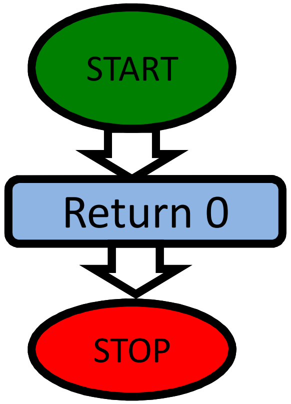
```
int main()
{
   return 0;
}
```



KISS INSTITUTE FOR PRACTICAL ROBOTICS

Programming Languages have Rules/Conventions as well, let's look at a simple C (programming language) program

```c
int main()
{
   return 0;
}
```



This is the main function and where a program starts executing. ALL C programs must have a main() function.

A function specifies what is to be done. It is like a title to a book of instructions.

Functions

Function- ***a function in a program specifies what is to be done. It is like a title to a book of instructions***.

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.

Looking at a program

return type   *name*   *argument list*

```
int main()
{
    return 0;
}
```

- Programs should always return something. In this case the **int** stands for integer (whole number)
- The name is descriptive so you can easily see what it is.
- In between the argument list you can specify details.
  - In this case it is not specifying anything to be returned.

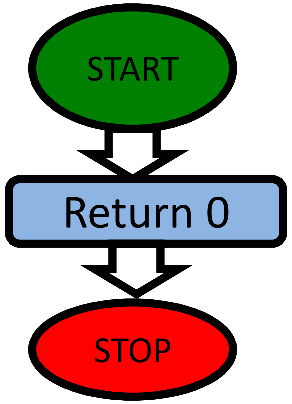
KISS INSTITUTE FOR PRACTICAL ROBOTICS

Looking at a program

Start → `{`

Stop → `}`

```
int main()
{
    return 0;
}
```

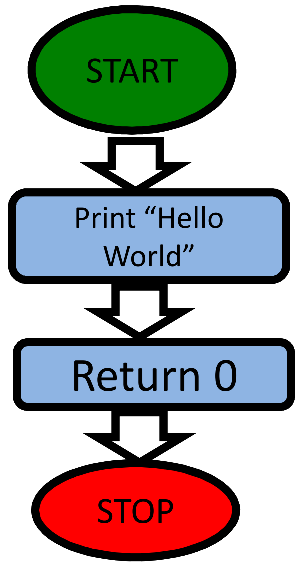


The curly braces organize everything you want the program to do (execute) when the computer comes to the last curly brace it will end the main program.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Looking at a program

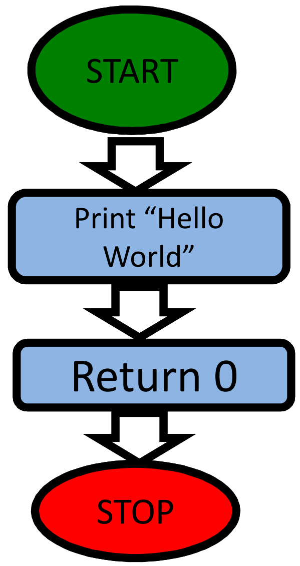
```
int main()
{
    printf("Hello, World!\n");
    return 0;
}
```



This is the code and specifies the things (functions) you want the program to execute.

Looking at a program

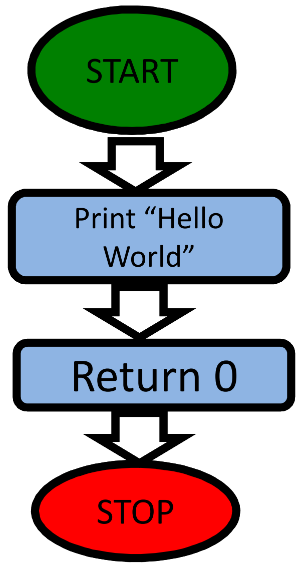
```
int main()
{
    printf("Hello, World!\n");
    return 0;
}
```



Notice the program returns a value even though it is 0.

Looking at a program

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



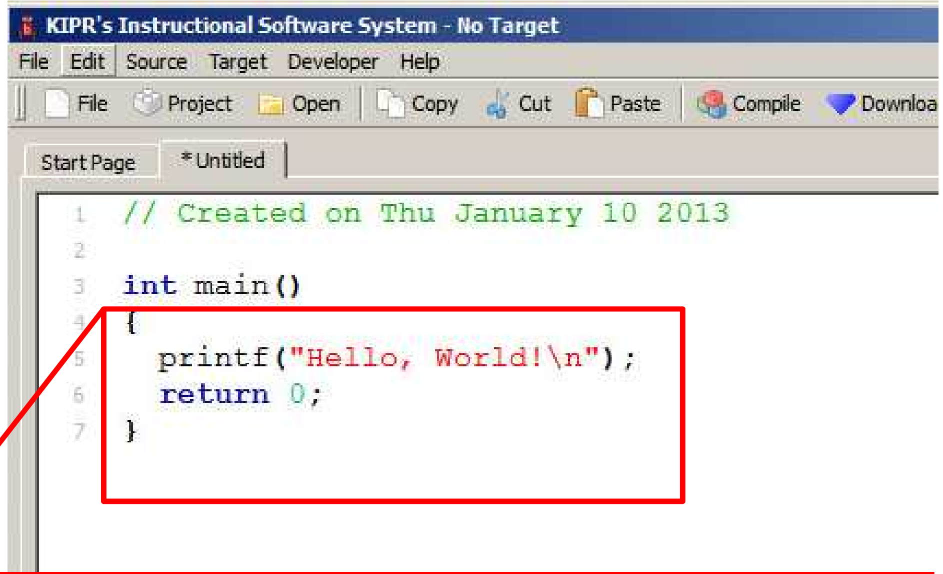
When the program is executing the semicolon terminates the code and says go to the next line. Without it, the code will not compile (be translated so the computer can understand it).

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The KISS IDE highlights parts of a program to make it easier to read

- By default, the KISS IDE colors your code and adds line numbers

- Comments appear in green
- Key words appear in **bold blue**
- Text strings appear in red
- Numbers appear in aqua




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

Functions

TEACHER'S RESOURCE

Goals

- To help students understand functions and how to access the KISS IDE Function Library
- To reinforce the learning of basic functions they will use when programming their robots
  - Function has `descriptive_name (argument);` (terminating statement)

Preparation

- Print cards (found in resource slide) that students can have with them when they start programming their robots
- Have the KISS IDE open and running on a master computer everyone can see OR better yet open on each of the student's computers

Activity

- **Have the student open the KISS IDE (instruction in resource slides)**
  - Access the KISS IDE User Manual
  - Scroll through the function libraries (The libraries also contain functions for the iRobot Create Platform in addition to the DemoBot platform these exercises use)
  - Have students cut out the function cards they will use when programming their robots

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Background Information
Why use **C**?

TEACHER'S RESOURCE

- **C** is a high level programming language developed to support the Unix operating system
  - The KIPR Link controller utilizes a version of Unix called Linux
- **C** is the most widely used language for systems programming
- Botball robots need to be programmed at the systems level to use the features of the KIPR Link
- For the Link controller, the KISS IDE (Integrated Development Environment) provides a user friendly interface to develop programs in **C**, **C++, Java and Python**
- These activities focus on **C**

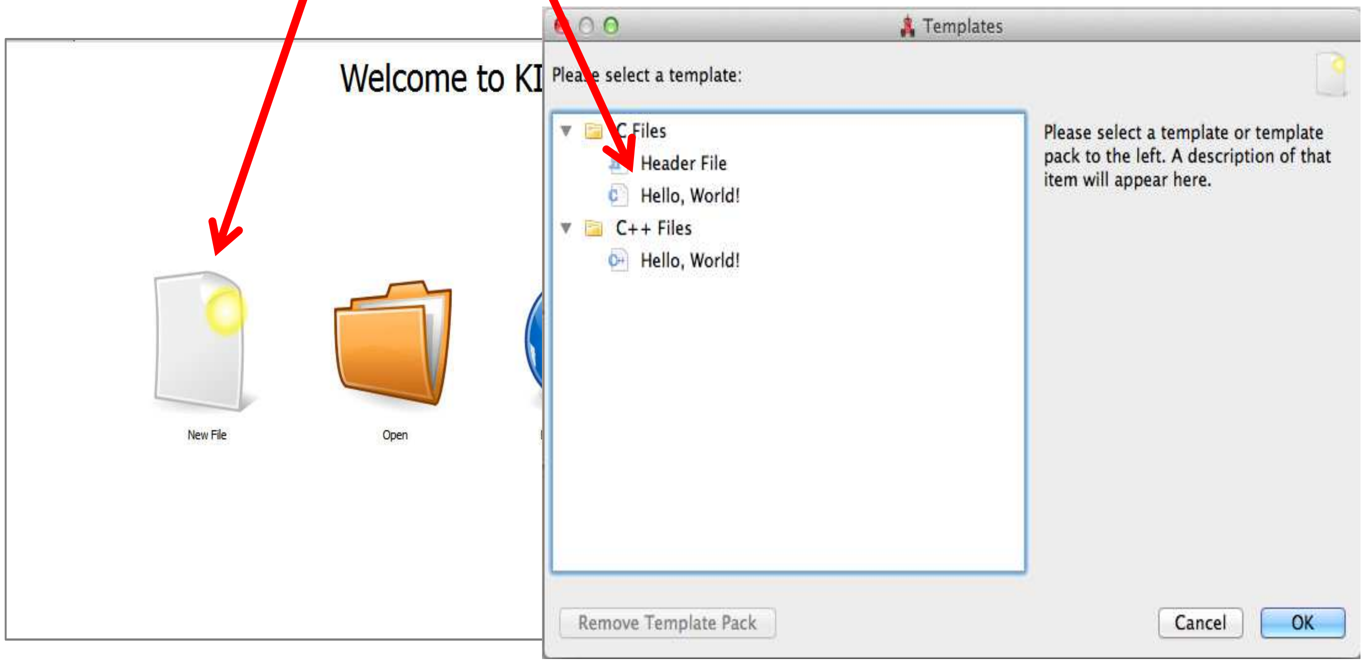
KISS INSTITUTE FOR PRACTICAL ROBOTICS

Launch the KISS IDE

KISS IDE icon

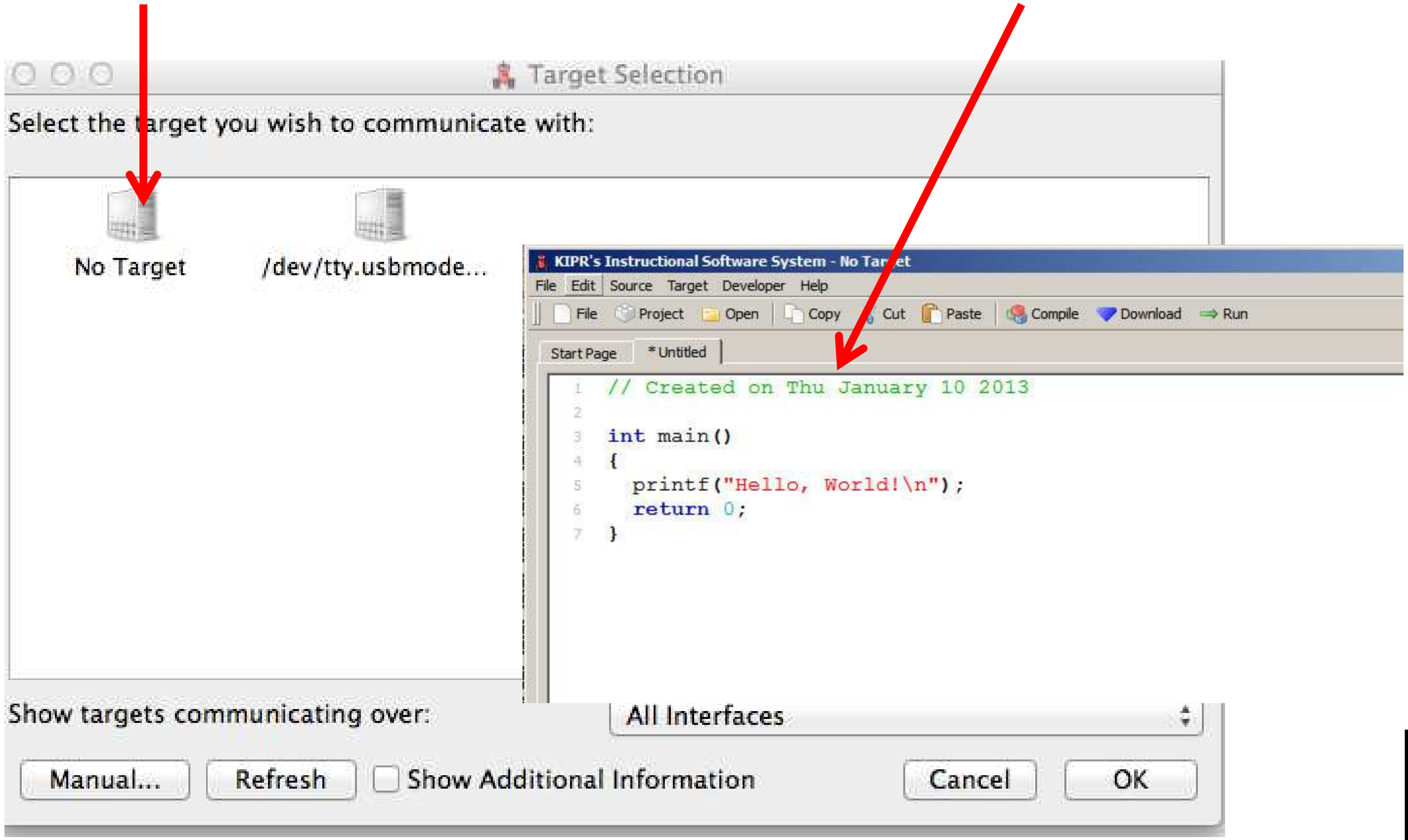
- Start the KISS IDE by clicking on its icon to get the welcome screen
- Click on the *"New File"* icon and

  and choose the C, *"Hello, World!"* template



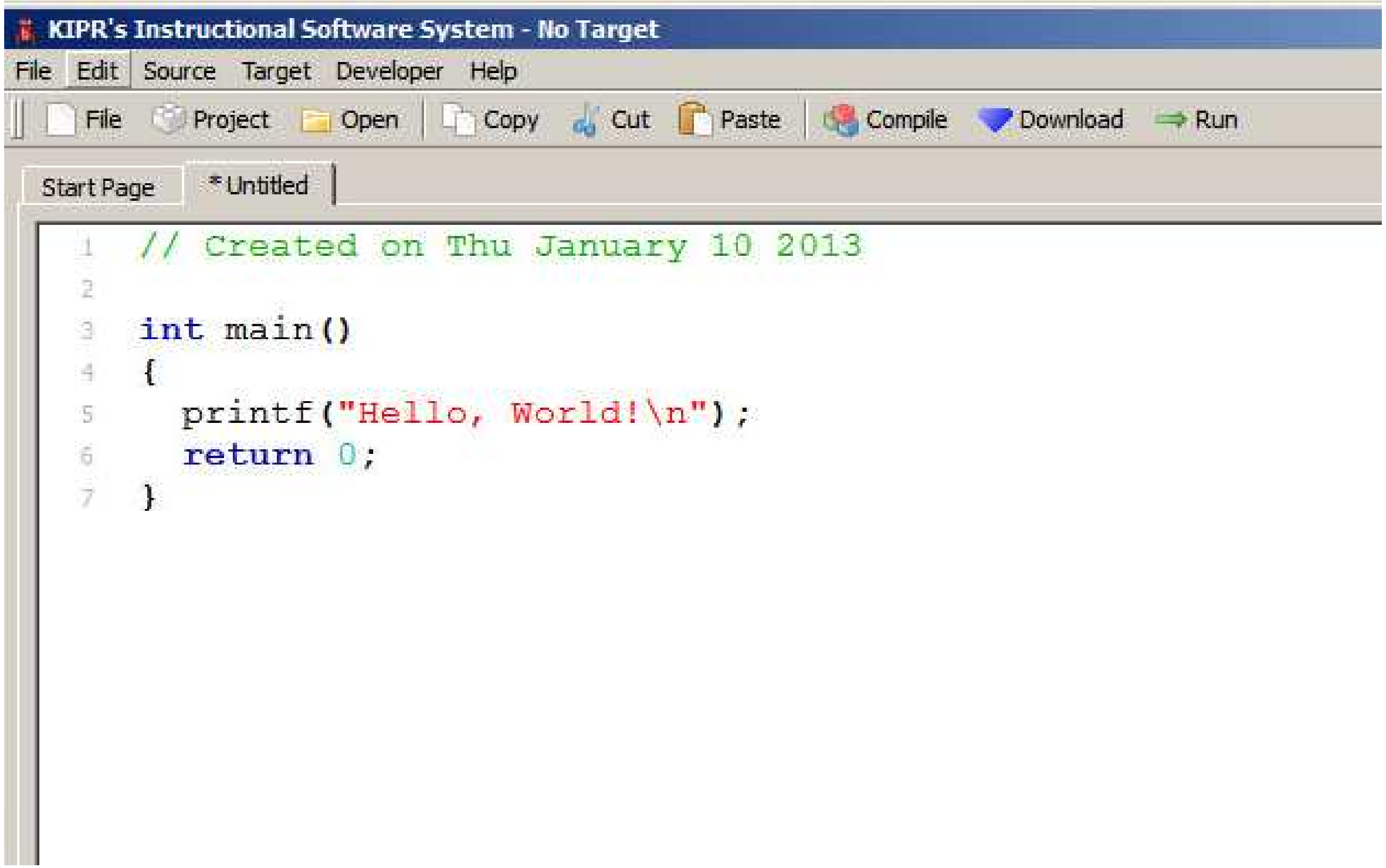
Select Target

- A Target Selection window will appear
- Pick *"No Target"* and the C program template will come up



STUDENT ACTIVITY

The C Template: Hello, World!



KISS INSTITUTE FOR PRACTICAL ROBOTICS

TEACHER'S RESOURCE

motor (port#, % power);	Turns motor on at % power specified
ao ();	All off, turns all motor ports off
digital (port #) ;	Refers to a specific digital port #
analog 10 (port#) ;	Refers to a specific analog port #
mrp (port#, velocity, position) ;	Move to relative position (# ticks)
wait_for_light (port#) ;	Robot waits for light in specified port 3 before starting
shut_down-in (time in seconds);	Shuts down all motors at specified time
msleep (# miliseconds) ;	Program waits specified number of milliseconds
enable_servos () ;	Turns servo ports on
set_servo_position (port#, position);	Moves servo in specified port to a set position
disable_servos ();	Turns all servo ports off

motor (port#, % power);	Turns motor on at % power specified
ao ();	All off, turns all motor ports off
digital (port #) ;	Refers to a specific digital port #
analog 10 (port#) ;	Refers to a specific analog port #
mrp (port#, velocity, position) ;	Move to relative position (# ticks)
wait_for_light (port#) ;	Robot waits for light in specified port 3 before starting
shut_down-in (time in seconds);	Shuts down all motors at specified time
msleep (# miliseconds) ;	Program waits specified number of milliseconds
enable_servos () ;	Turns servo ports on
set_servo_position (port#, position);	Moves servo in specified port to a position
disable_servos ();	Turns all servo ports off

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Functions

A function is like a title to an instruction book. When you call the function it does all of the commands in the book.

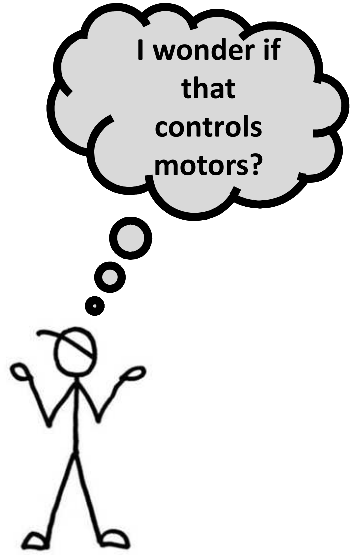
A `clean_house()` function could mean vacuum, dust, mop, change the linens, wash the windows, etc… all the commands specified in the function are executed.

The KISS IDE contains a large library of functions you can use to program your robots.

Function Examples

Name Arguments Terminating Statement

- **motor(0, 100);**
  - o Turns on motor in port 0 at 100% power



- **digital(8);**
  - o Returns the values from the sensor plugged into the #8 digital port. It will be a number, either 1 or 0 (1 = yes/true and 0 = no/false).

- **analog10(3);**
  - o Returns the value of the analog sensor plugged into analog port #3 (analog values are between 0 and 1024).

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Where Can I Get Help?

The KISS IDE has an extensive ***User Manual*** including a brief **C** tutorial

- The ***User Manual*** is found under the *Help* menu
- When using **C** for Botball, the ***User Manual*** is the primary document to consult
- The ***User Manual*** covers the library of functions for accessing the features of the Link controller and for controlling a Create module

The ***Sensors and Motors Manual*** provides additional information about the sensors and motors used with the KIPR Link

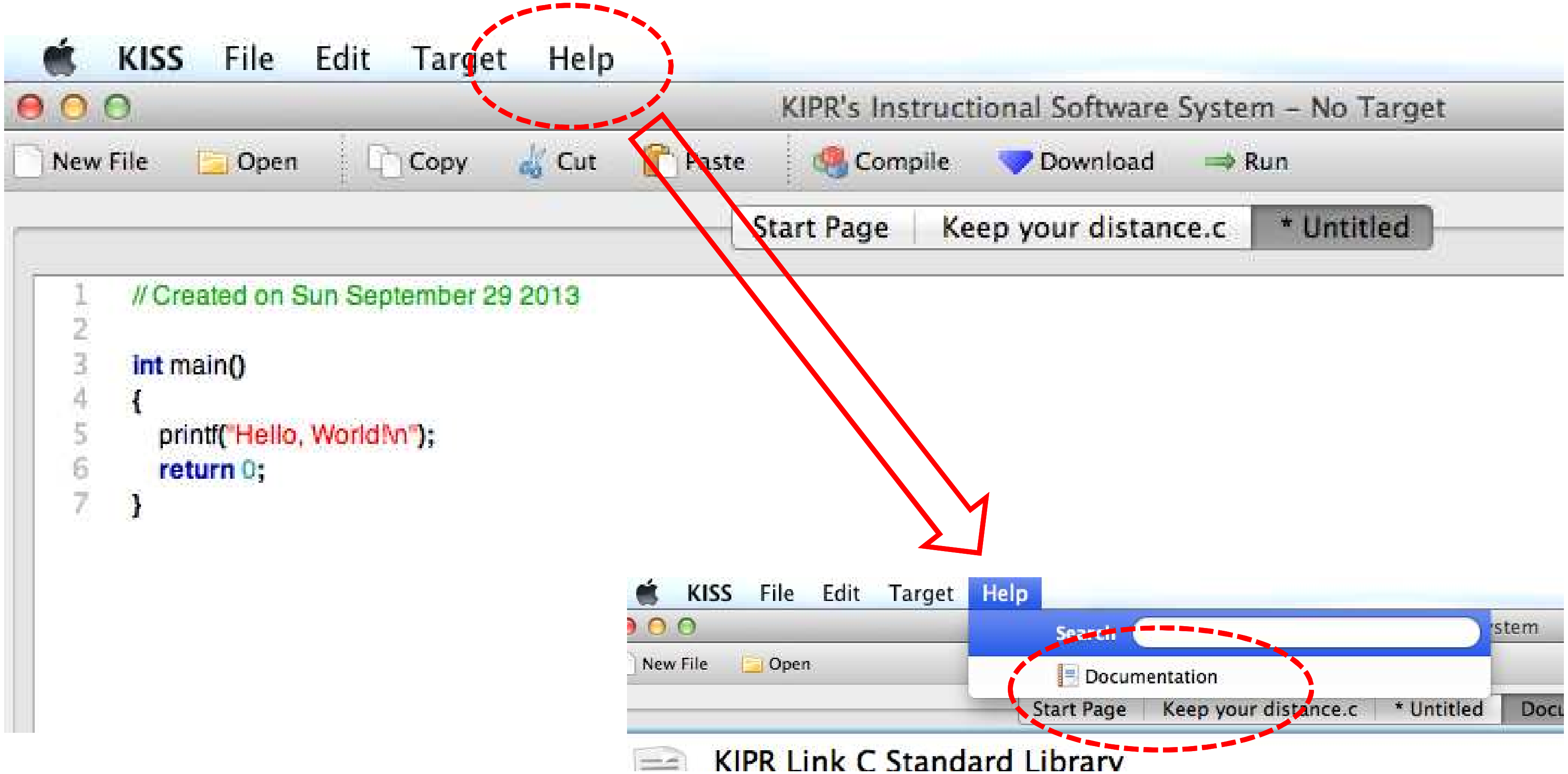
- Accessed from the Team Home Base or on teacher's flash drive

KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Getting Help

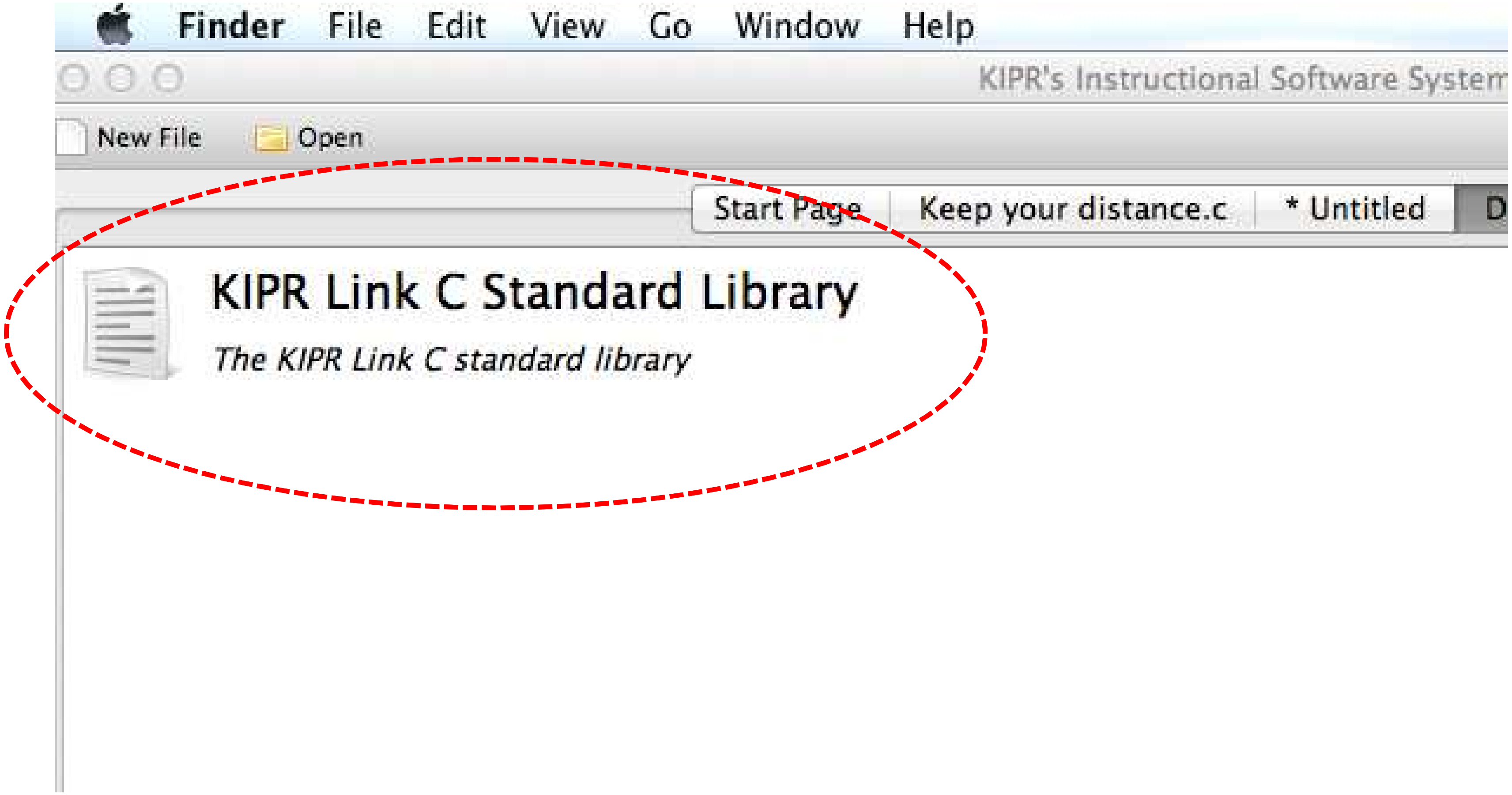
With the KISS IDE open simply select the help tab then documentation



KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Select KIPR Link C Standard Library



KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

KISS IDE User Manual

Select KIPR Link Library

Programmers Manual Index

KISS User Manual for C

Introduction

KIPR's Instructional Software System (KISS for short) is an integrated development environment providing an editor, compilers for multiple programming languages, and a set of libraries and simulator for the LINK Botball Controller. KISS implements the full ANSI C specification. For information about the C programing language, including history and basic syntax, see the Wikipedia article C (programming language). For a more complete tutorial and guide for C Programming visit CPrograming. The Botball community website also has several articles about programming and a user forum where questions can be posted to the botball community. For specific information on Motors and Sensors, see the Sensors and Motors Manual

The primary purpose of this manual is to describe the KIPR Link Botball Controller libraries and simulator, which are extensions to the C programming language. This file also provides a basic introduction to programming in C. To learn more about programming in C, consult one of the many books or websites that provide C references and tutorials.

KISS Interface

Both new (unsaved) and saved files can be opened for editing in KISS. A row of tabs lists the files that have been opened. Clicking a file's tab activates it for editing.

The File menu has standard entries for New, Open, Save, Save As, Print, Close and Exit.

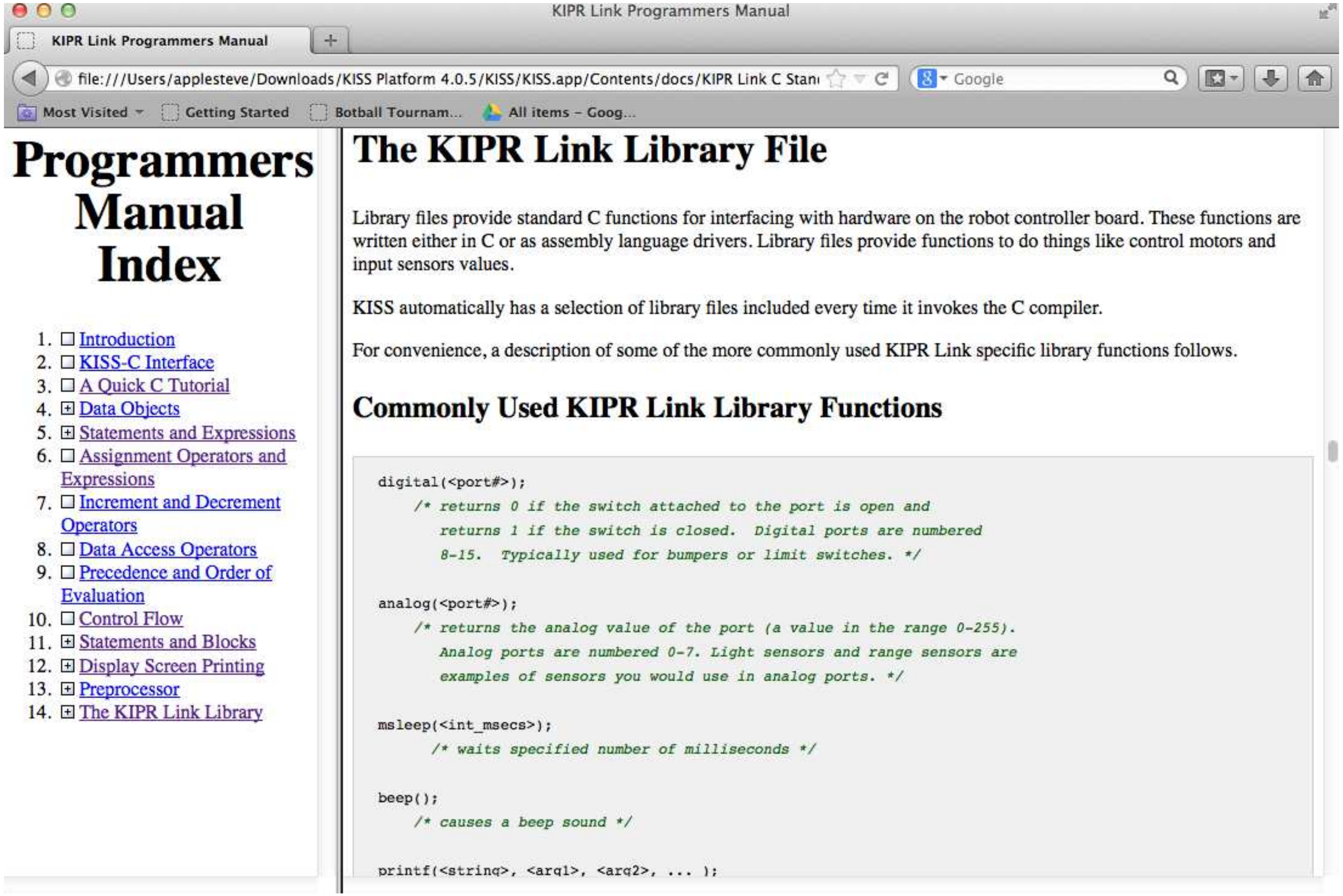
To simulate the active file, simply click the Simulate button. The active file will also be saved, unless it is new, in which case the user is prompted for a "save as" file name. The active file must contain or #include the main function, in order to be simulated.

To download the active file, click on the Download button. If the serial port connecting the KIPR Link to your pc has not already been specified, a dialog

KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

KIPR LINK Library



KIPR Link Programmers Manual

file:///Users/applesteve/Downloads/KISS Platform 4.0.5/KISS/KISS.app/Contents/docs/KIPR Link C Stan

Most Visited | Getting Started | Botball Tournam... | All items – Goog...

Programmers Manual Index

The KIPR Link Library File

Library files provide standard C functions for interfacing with hardware on the robot controller board. These functions are written either in C or as assembly language drivers. Library files provide functions to do things like control motors and input sensors values.

KISS automatically has a selection of library files included every time it invokes the C compiler.

For convenience, a description of some of the more commonly used KIPR Link specific library functions follows.

Commonly Used KIPR Link Library Functions

```
digital(<port#>);
    /* returns 0 if the switch attached to the port is open and
       returns 1 if the switch is closed.  Digital ports are numbered
       8-15.  Typically used for bumpers or limit switches. */

analog(<port#>);
    /* returns the analog value of the port (a value in the range 0-255).
       Analog ports are numbered 0-7. Light sensors and range sensors are
       examples of sensors you would use in analog ports. */

msleep(<int_msecs>);
      /* waits specified number of milliseconds */

beep();
    /* causes a beep sound */

printf(<string>, <arg1>, <arg2>, ... );
```

Hint

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

motor (port#, % power);	Turns motor on at % power specified
ao ();	All off, turns all motor ports off
digital (port #) ;	Refers to a specific digital port #
analog 10 (port#) ;	Refers to a specific analog port #
mrp (port#, velocity, position) ;	Move to relative position (# ticks)
wait_for_light (port#) ;	Robot waits for light in specified port 3 before starting
shut_down-in (time in seconds);	Shuts down all motors at specified time
msleep (# miliseconds) ;	Program waits specified number of milliseconds
enable_servos () ;	Turns servo ports on
set_servo_position (port#, position);	Moves servo in specified port to a set position

Writing Programs- Screen Display

Goals

- To help students understand how to use the KISS IDE to write a program
- To understand how to compile, download and run the program on their Link controller
- To understand how to use the print function to print things to the screen
- To understand how to use the msleep function to give commands time to execute

Preparation

Students will need computers with KISS IDE installed and access to a Link controller

Activity

- **Have the student open the KISS IDE (instruction in resource slides)**

- **Follow the slides to write program to the screen display**

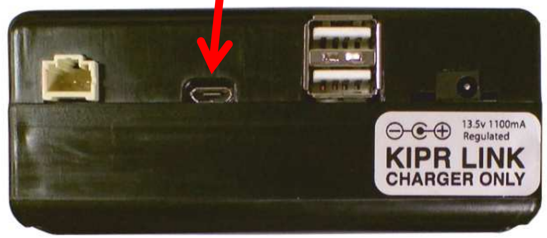
First C Program

Programming Basics and Screen Output

STUDENT ACTIVITY

Connect the Link to your computer

- Using the USB cable connect to the Link (micro usb port)
- Turn the Link on with the black switch on the side



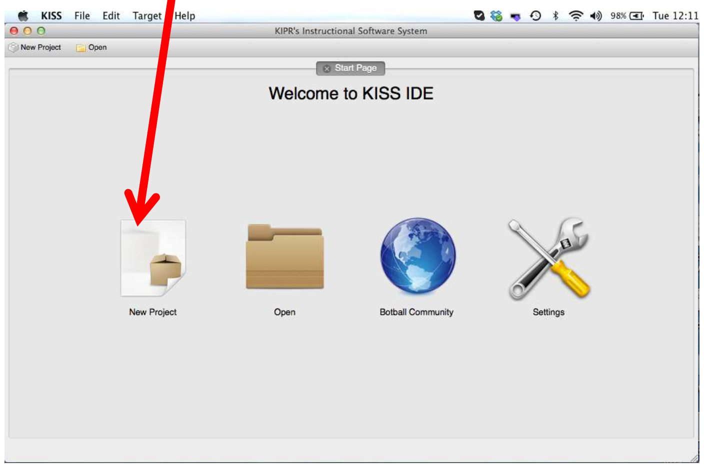
STUDENT ACTIVITY

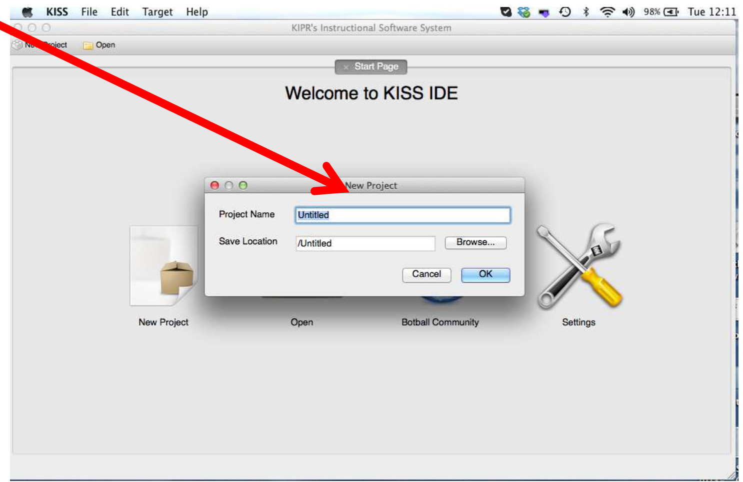
Writing Your First Program

Launch the KISS IDE

- Start the KISS IDE by clicking on its icon to get the welcome screen

- Click on the *"New Project"* icon
- You will have to name and save your project



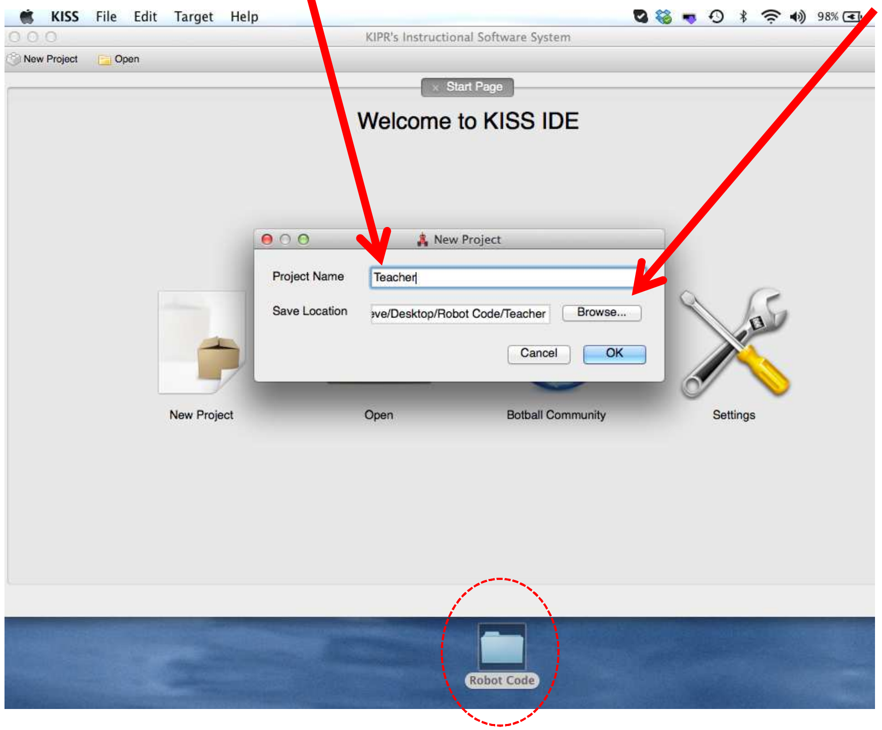


KISS INSTITUTE FOR PRACTICAL ROBOTICS

Writing Your First Program

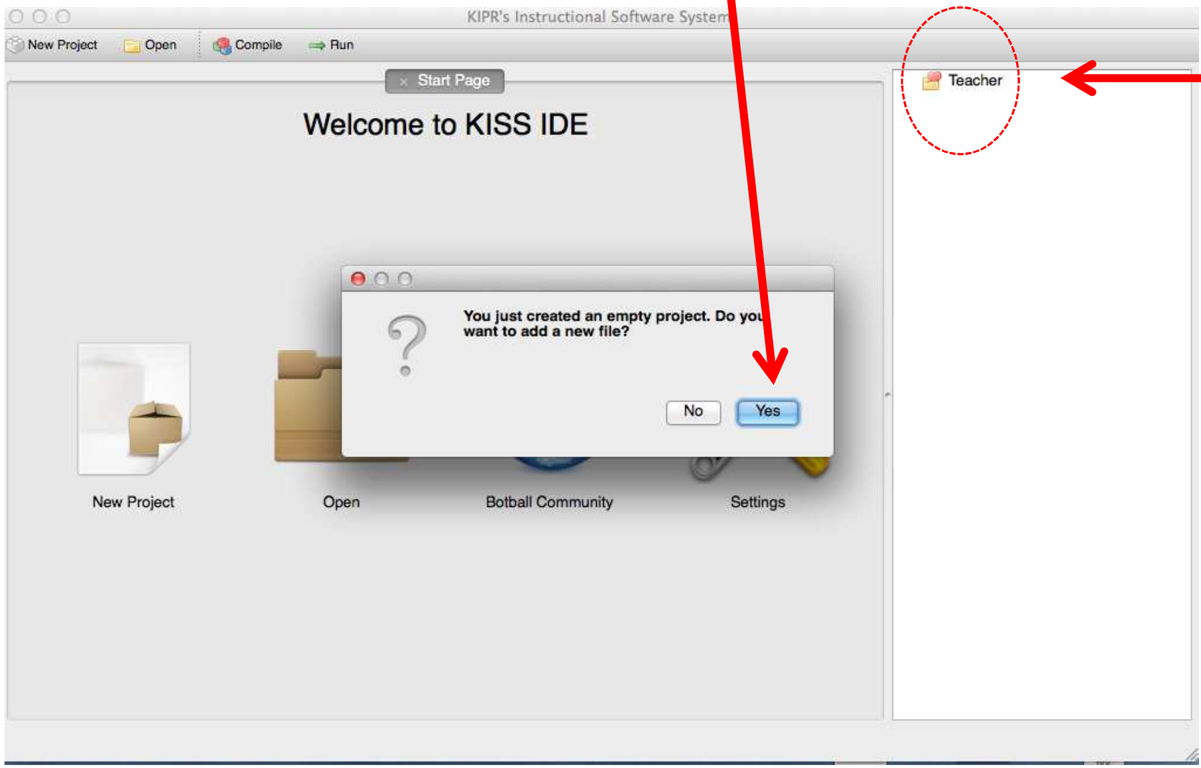
- Create a Robot Code folder on your desktop (you will use this for all of your code)
- Name your new project

Use the Browse button to save the project into your Robot Code folder on your desktop



Writing Your First Program

- You will be asked if you want to add a new file

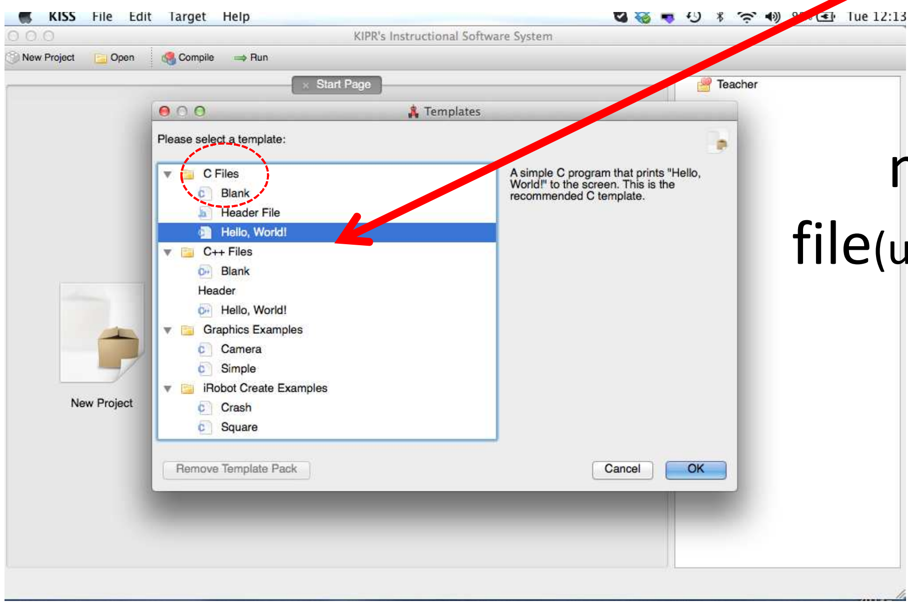


Notice your project "teacher" now appears on the side bar

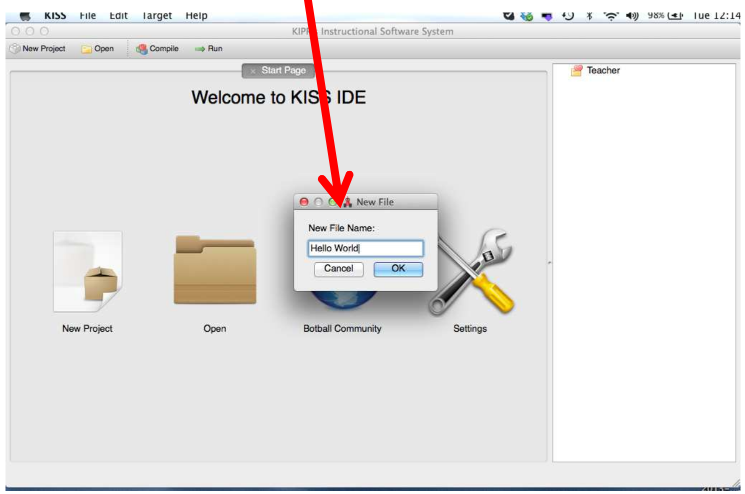
Writing Your First Program

You now need to select a template

Select under the C files folder, *Hello, World!*



You will be prompted to name and save the new file(use something descriptive)



STUDENT ACTIVITY

Writing Your First Program

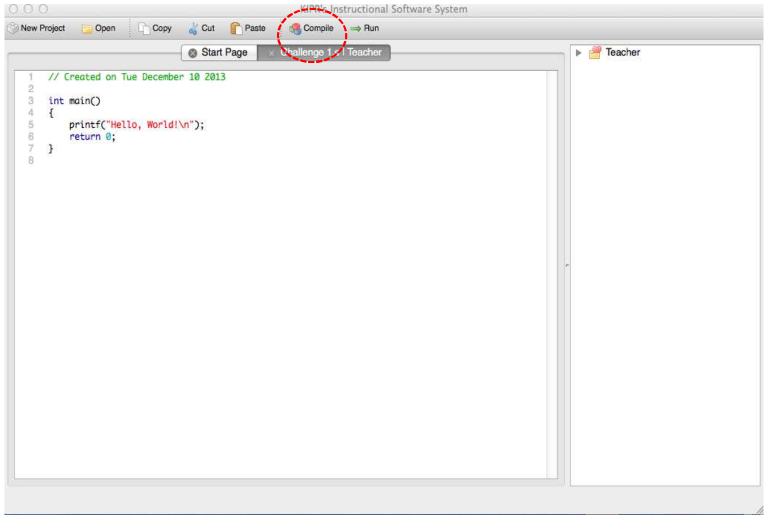
The “Hello, World” template will now appear

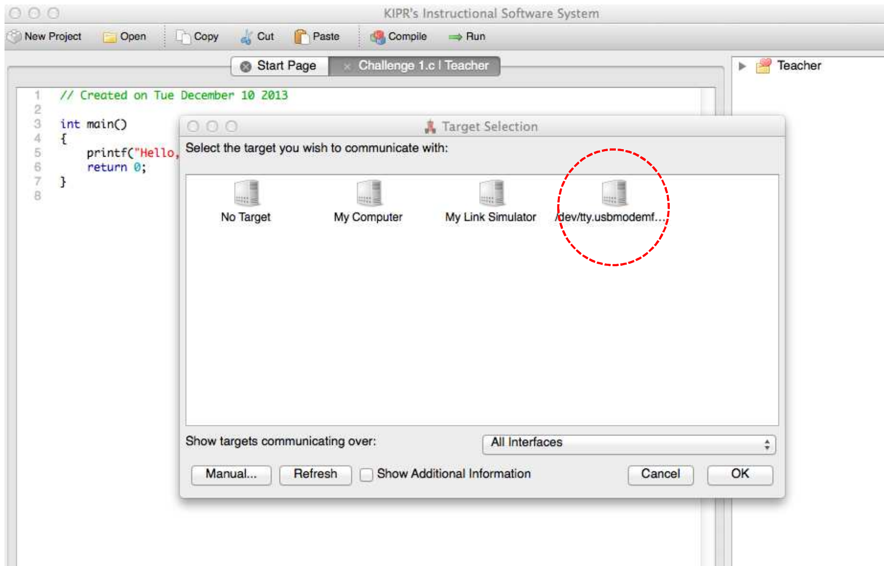
To run the program, you must Compile it

(The compile button sends the program to your target to be compiled)

A *Target Selection* window will appear

Select the usb target (this is your robot)

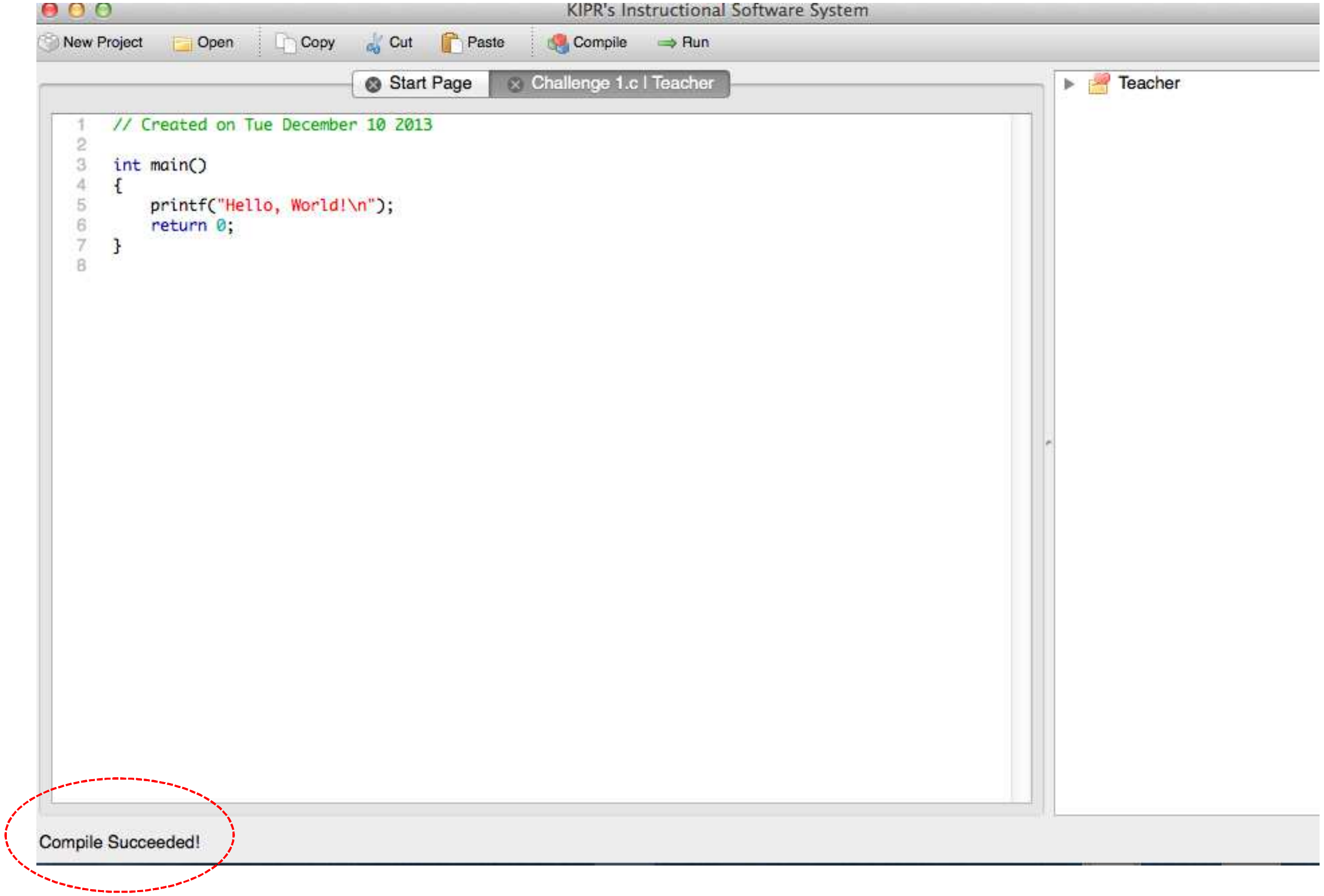




KISS INSTITUTE FOR PRACTICAL ROBOTICS

Writing Your First Program

You will see the *Compile Succeeded*! message



Activity 1 (Task Design)

Programming Basics and Screen Output

Break the objectives down into separate tasks and think about how each might be accomplished; for example, the larger task might be developing a program to operate a robot's claw, which has tasks within for making the claw open or close.

Since this is our first example, the task is pretty simple:

1. Display the text "Hello World!" on the Link screen.

Pseudocode and Comments

Pseudocode- write out what you want the program to do

pseudocode (this means "false code") to help write the real code…

// 1. Display "Hello World!" on the screen.

Comment your code (pseudocode makes great comments) - your comments show what you expect your program to cause your robot to do, but that might not be what it will actually do!

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Comments

Comments as psuedocode are helpful and they help you keep track of what is going on in the program.

You can make a flow chart and then convert it to psuedocode.

The computer will not execute the comment, but you can see it.
There are 2 ways to comment C programs // and /* */

// is a comment for rest of line

or

/* is a comment that goes from
the initial slash-star until
the first star-slash */

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The Program Explained

(it illustrates most C syntax)

return type · *name* · *argument list* · *Comment* · *Function*

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

Syntax is important! Notice the quotation marks and notice the \n at the end?

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Blocks of Code

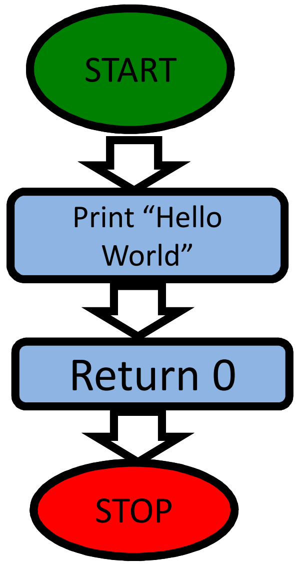
Comment

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

KISS INSTITUTE FOR PRACTICAL ROBOTICS

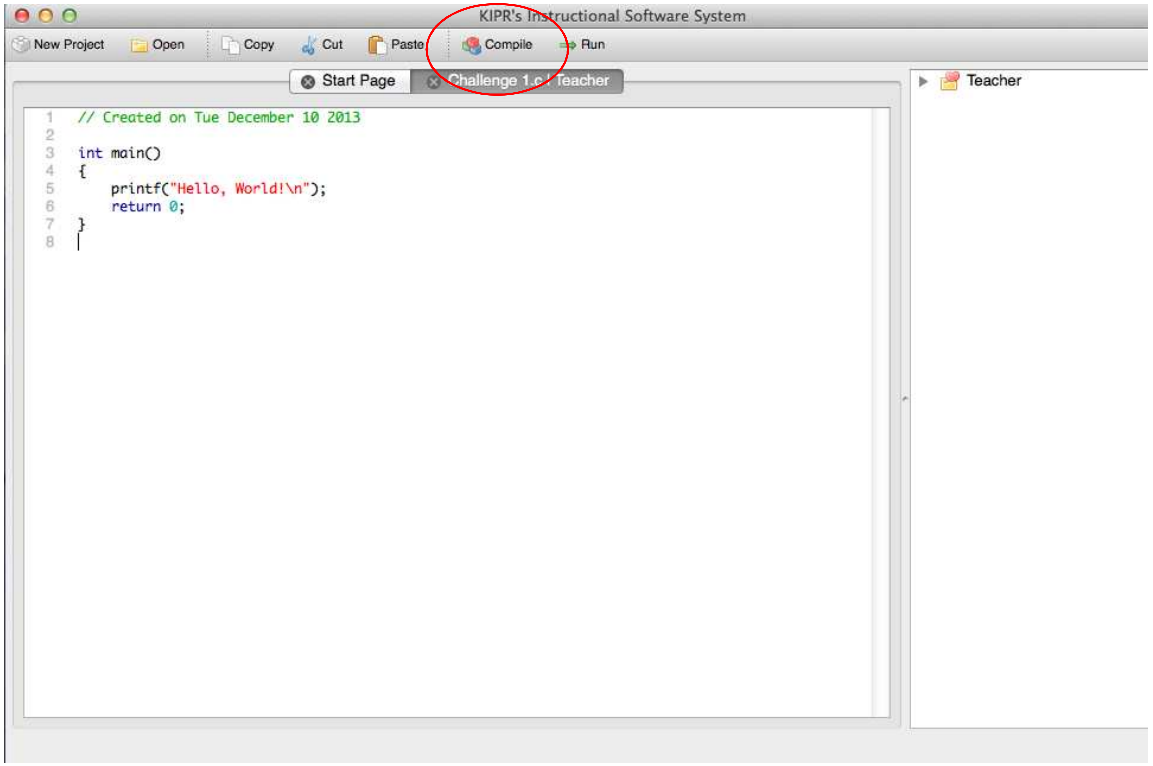
Terminating Statements

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



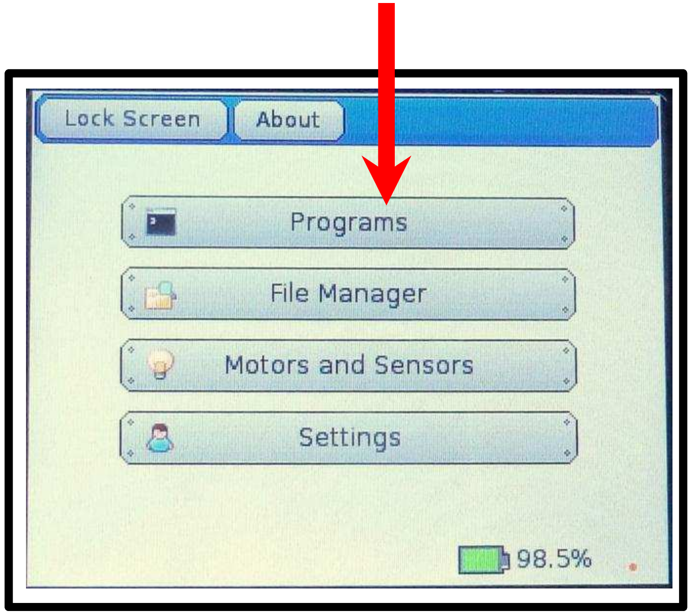
Compile your program

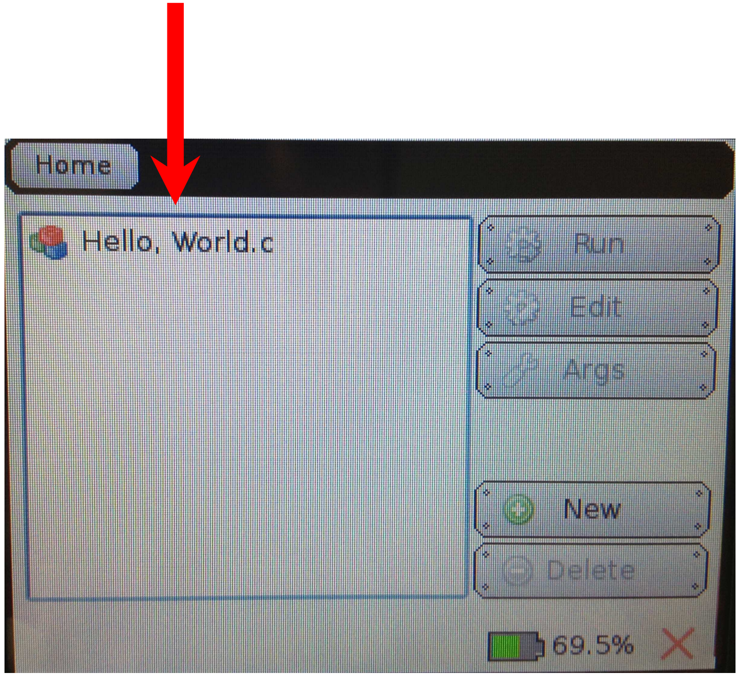
- Compile your program using the compile tab



Running your program on the Link

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



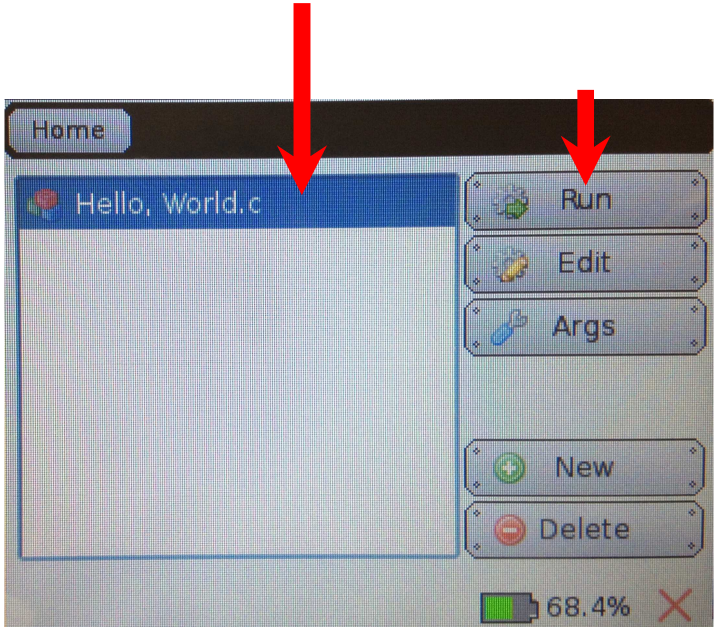


KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Running your program on the Link

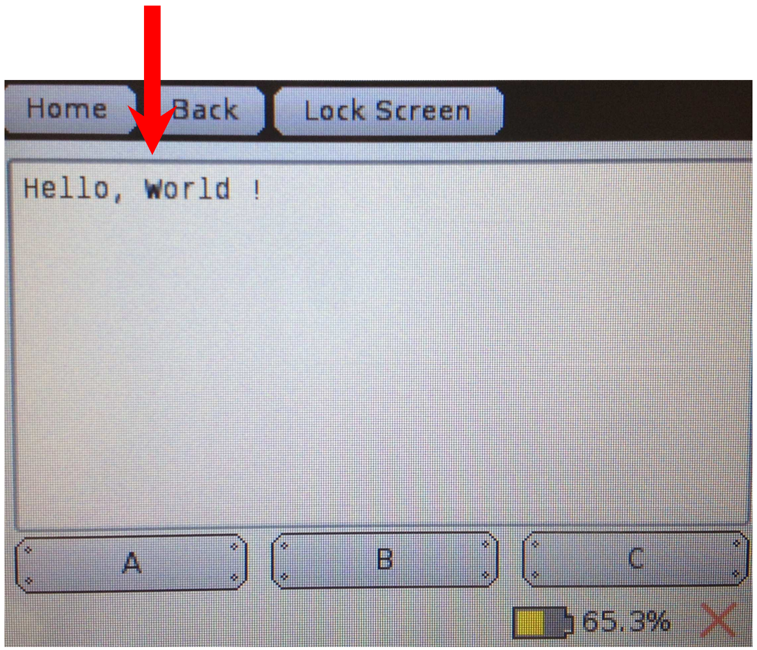
- Highlight the program you want to run, in this case, *"Hello World"*, and then push the "Run" button



KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Running your program on the Link



KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Activity 2
Programming Basics and Screen Output

Write a program for the KIPR Link that displays "Hello World!” and then displays your “name”, compile, download and run it on your *Link*.

Psuedocode (Task Analysis)

```
// 1. Display "Hello World!" on the screen.
     printf (“Hello World\n”);
// 2. Display your name on the screen.
     printf (“ Botguy\n”);
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

What did you notice when you ran the program?

The controller reads the code 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.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

msleep()

Like **printf()**, **msleep()** is a built-in (library) function.

msleep(3000) causes the KIPR Link to pause for 3 seconds (the m stands for milliseconds or 1/1000 of a second).

- Example:

```
printf("slow");
msleep(3000);
printf("reader\n");
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Activity 3

Programming Basics and Screen Output

Write a program for the KIPR Link that displays "Hello World!" to the screen, delays two seconds, and then displays your name on the screen.

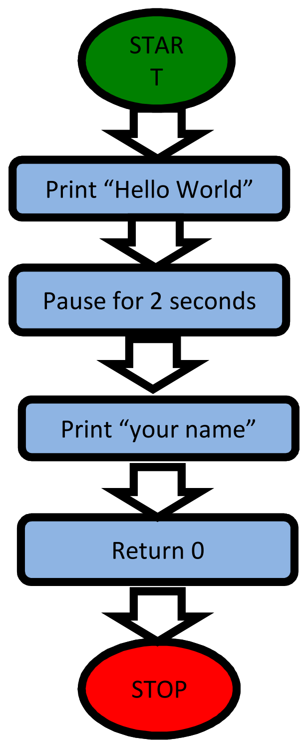
Psuedocode (Task Analysis)

```
// 1. Display "Hello World!" on the screen.
// 2. Pause for 2 seconds.
// 3. Display your name on the screen.
```

Activity 3 Solution

Programming Basics and Screen Output

```c
#/*****************************************************************
*********  Activity 2
*****************************************************************/
int main()
  {
    // 1. Display "Hello World!" on the screen
    printf("Hello World!\n");

    // 2. Delay for 2 seconds
    msleep(2000);    //2000ms = 2sec

    // 3. Display your name to the screen
    printf("Botguy.\n");

    return 0;
}
```



KISS INSTITUTE FOR PRACTICAL ROBOTICS

Debugging

TEACHER'S RESOURCE

Goals

- To help students understand how to use the KISS IDE to debug a program

Preparation

Students will need computers with KISS IDE installed and access to a Link controller

Activity

- Have the student make intentional errors to learn how to debug their program

KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Run the Hello World program on the *Link* again, but this time …

You can download it again to your Link OR simply hit the “BACK” Button on the touch screen and reselect the “Hello World” Program

- The Link will keep the program you run on it in its program files
  - Make sure you name programs so you know which ones to select

Leave off the terminating semicolon and see what happens

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

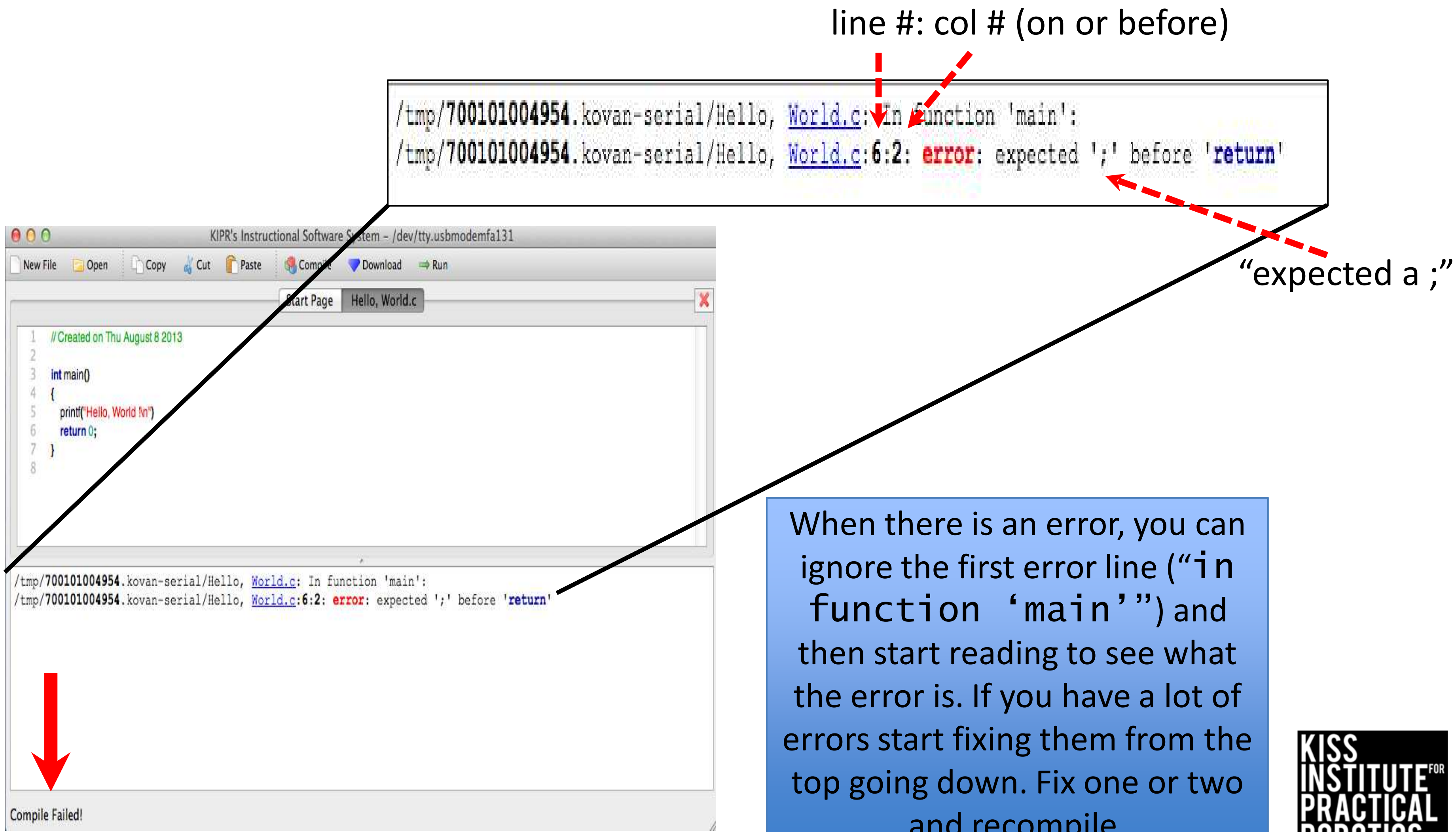
\n doesn't show up on the printed output it simply tells the display to print to a new line similar to the return key on a keyboard

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Compile Failed "Debugging"

Example "Error Message"

- Compile Failed message at the bottom of the window



When there is an error, you can ignore the first error line ("`in function 'main'`") and then start reading to see what the error is. If you have a lot of errors start fixing them from the top going down. Fix one or two and recompile.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Activity Extensions

Programming Basics and Screen Output

TEACHER'S RESOURCE

- Try adding more **printf()** statements to your program (pay close attention to the syntax, particularly the terminating semi-colon needed by each statement)
- Have the program print out a haiku about robotics
- What does \n and \t do?
- What happens if you leave off the quotation marks?
- Try adding the command **`display_clear();`**
- Can you print out more lines than can show on the screen at one time?
- What happens when the screen fills up?

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Moving your robot with the motor() function

TEACHER'S RESOURCE

Goals

- To reinforce the concept of a function
- **To use the motor function to move their robot**

Preparation

- You will need the DemoBot built and ready to go
- You will need computers with the KISS IDE
- You will need the USB download cable

Activity

Follow the slides to make the robot move

KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Activity 3

Lets make a robot move!

Use the provided robot or build your robot using the Demo Robot building guide.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Connect the Link to your computer

- Using the USB cable connect to the Link (micro usb port)
- Turn the Link on with the black switch on the side

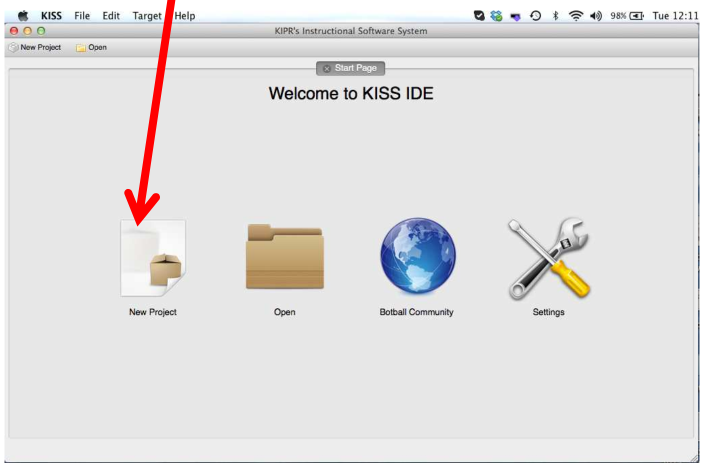
KISS INSTITUTE FOR PRACTICAL ROBOTICS

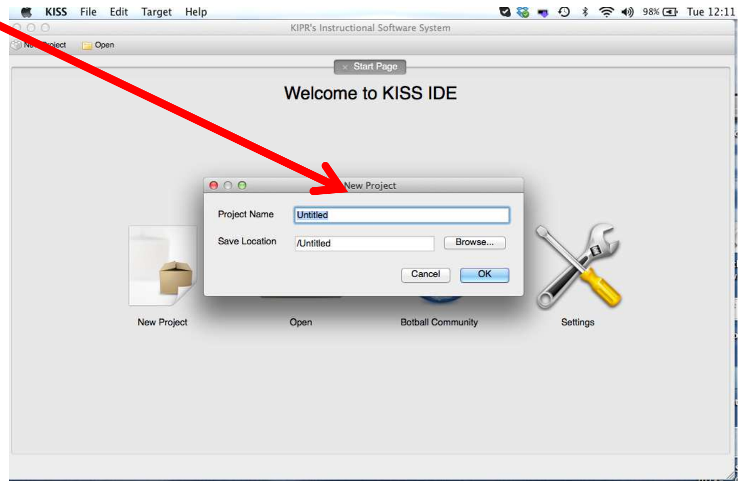
STUDENT ACTIVITY

Launch the KISS IDE

If already running just select new project

- Start the KISS IDE by clicking on its icon to get the welcome screen
- Click on the *"New Project"* icon
- You will have to name and save your project





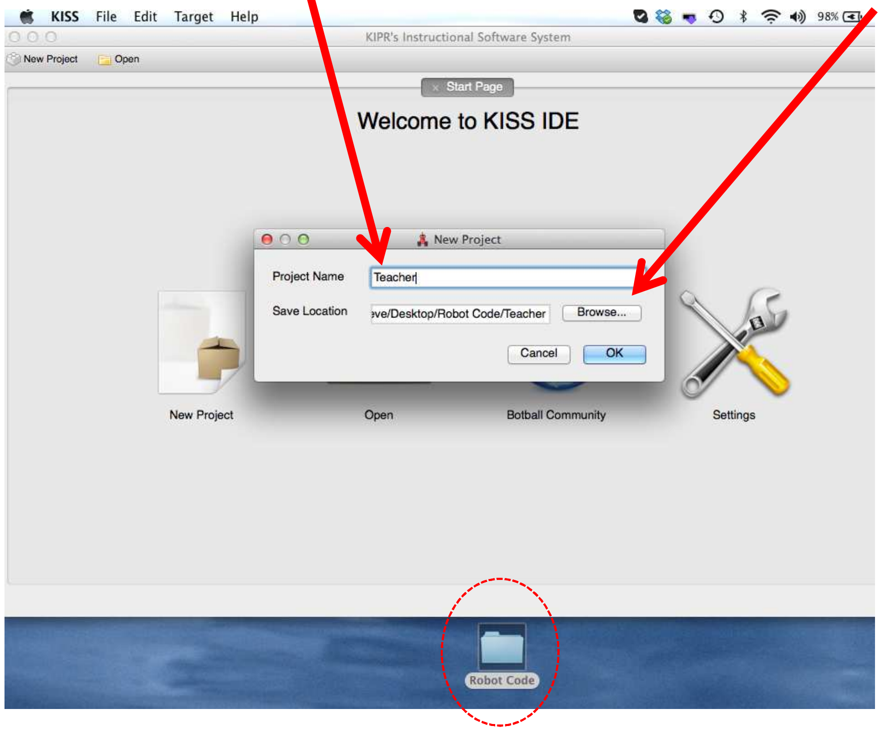
KISS INSTITUTE FOR PRACTICAL ROBOTICS

Writing Your First Program

STUDENT ACTIVITY

- Create a Robot Code folder on your desktop (you will use this for all of your code)
- Name your new project "Making the Robot Move"

Use the Browse button to save the project into your Robot Code folder on your desktop

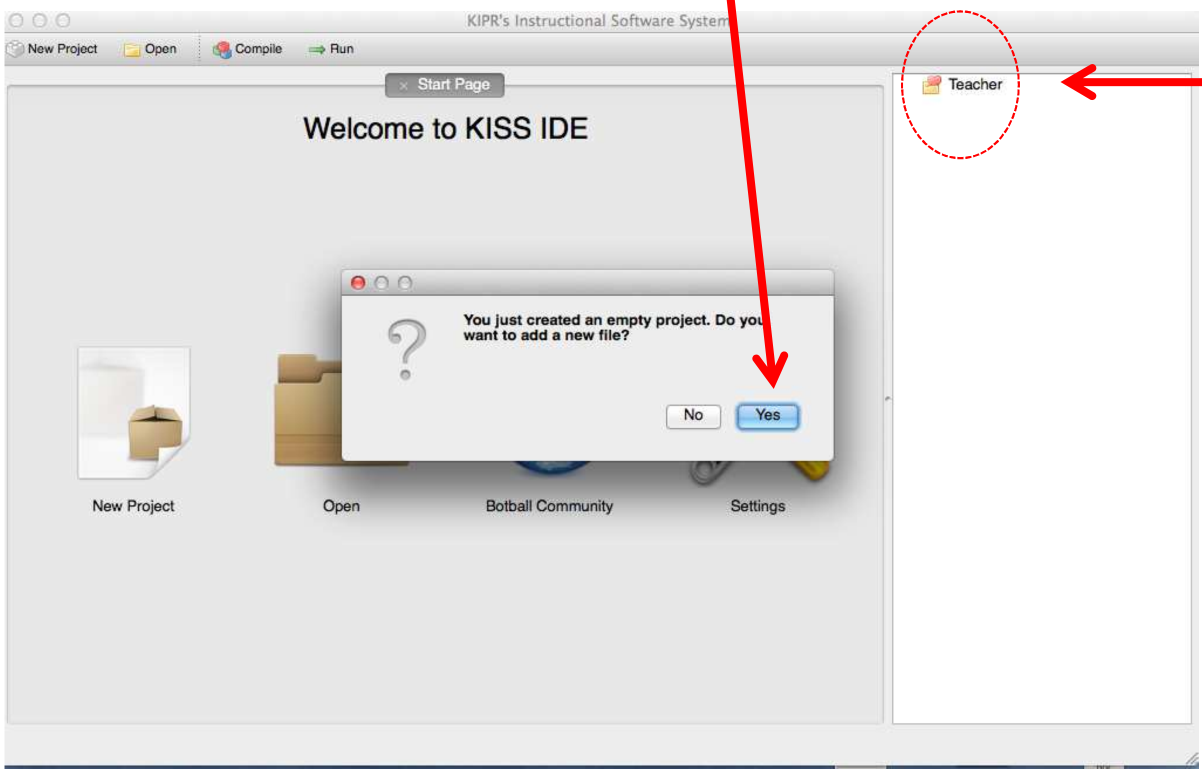


KISS INSTITUTE FOR PRACTICAL ROBOTICS

Writing Your First Program

STUDENT ACTIVITY

- You will be asked if you want to add a new file



Notice your project "teacher" now appears on the side bar

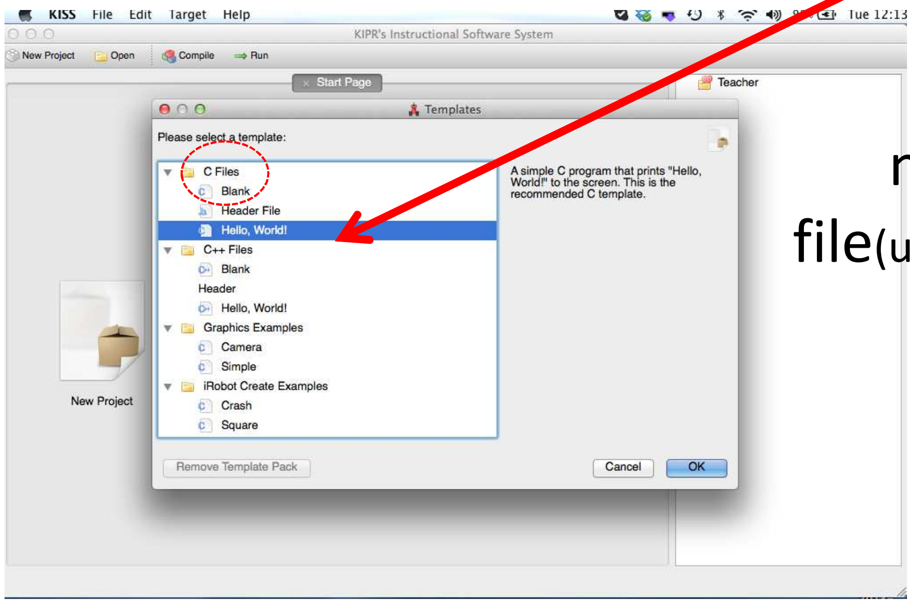
KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

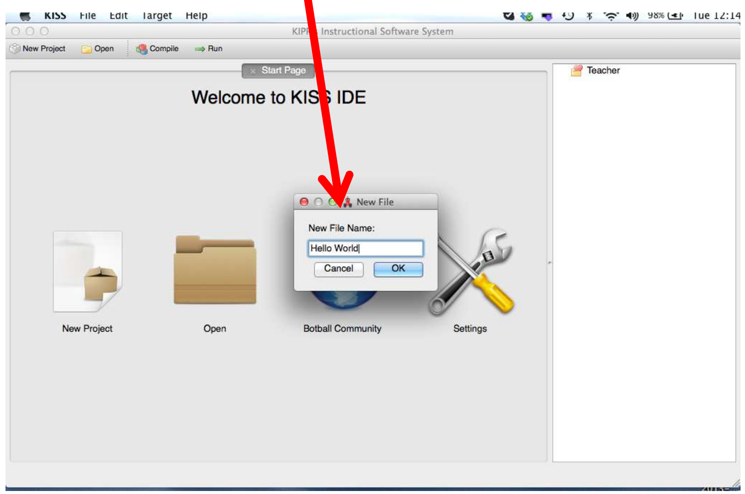
Writing Your First Program

You now need to select a template

Select under the C files folder, *Hello, World!*



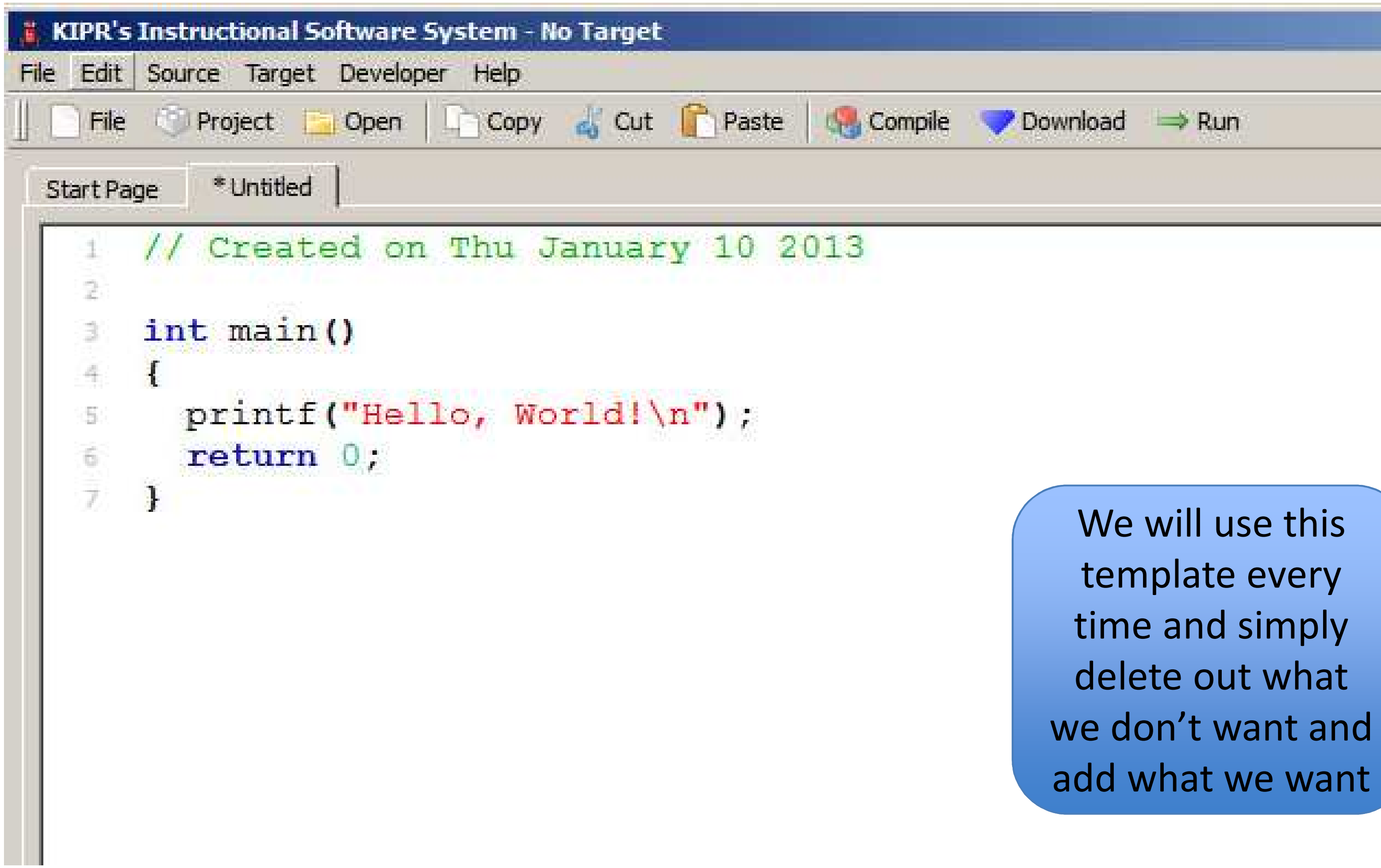
You will be prompted to name and save the new file(use “motor”)



KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

The C Template: Hello, World!



We will use this template every time and simply delete out what we don't want and add what we want

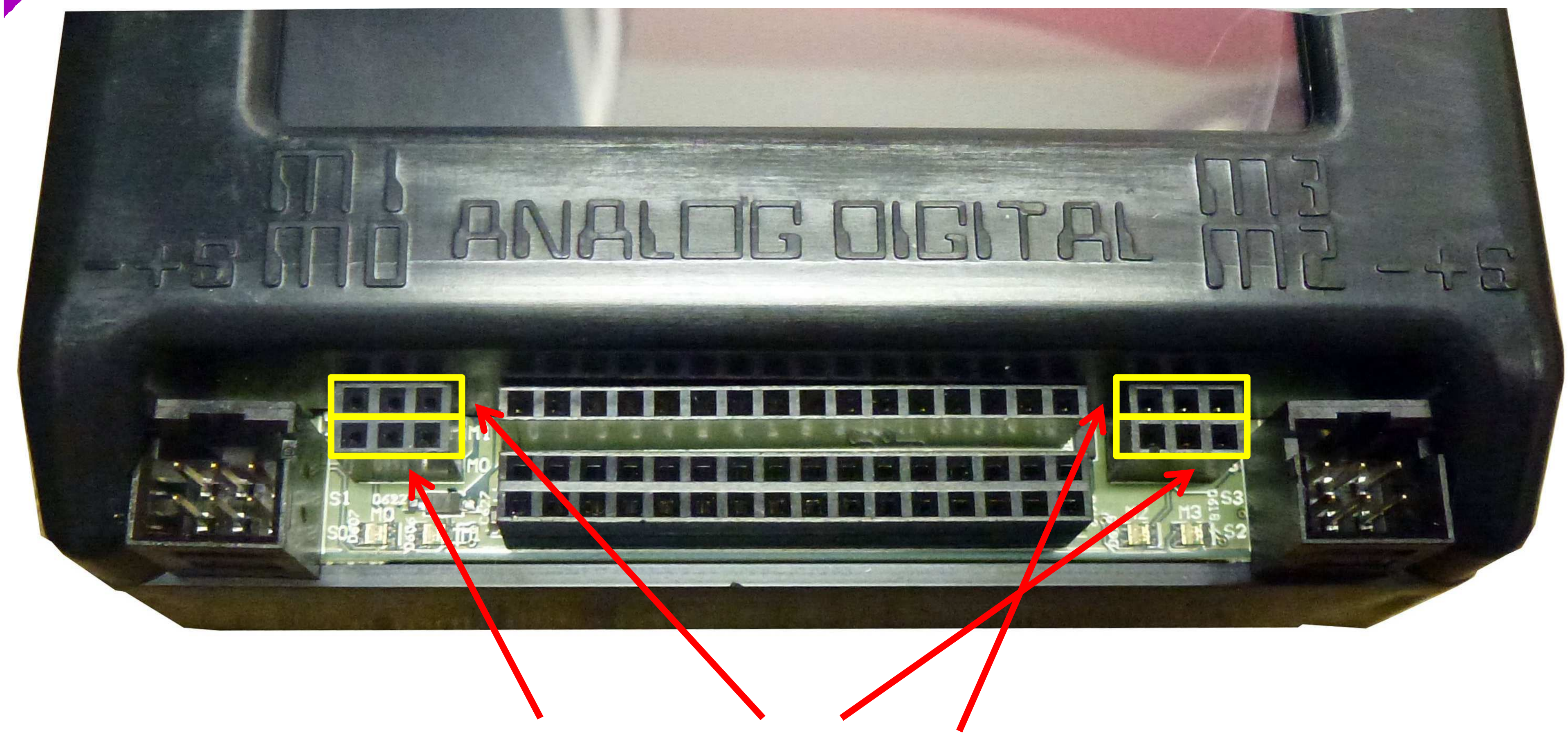
KISS INSTITUTE FOR PRACTICAL ROBOTICS

Check your Robot's Motor Ports

- To program your robot, you need to know what motor ports your motors are plugged into

*** REMEMBER computer scientists start counting at 0 so the motor ports are 0, 1, 2 and 3**

KIPR Link Motor Ports

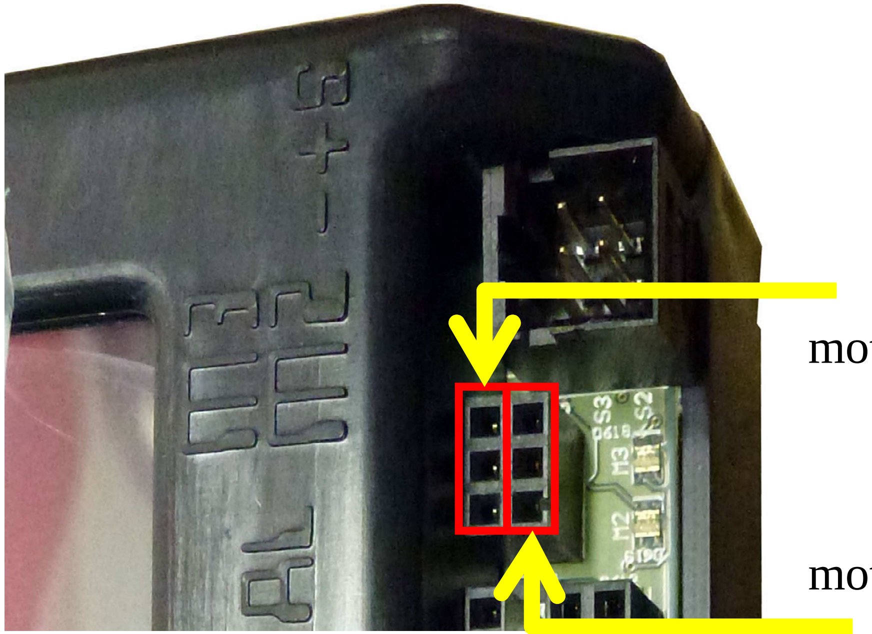


Motor ports 0 (DemoBot), 1, 2, and 3 (DemoBot)

STUDENT ACTIVITY

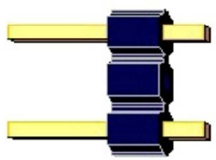
Plugging in Motors

- Motors are the ones with two-prong plugs with 2 gray wires
- The KIPR Link has 4 drive motor ports numbered 0 & 1 on the left and 2 & 3 on the right
- When a port is powered it has a light that glows green for one direction and red for the other
- Plug orientation order determines motor direction, but by convention, green is forward and red reverse



motor port 3

motor port 2



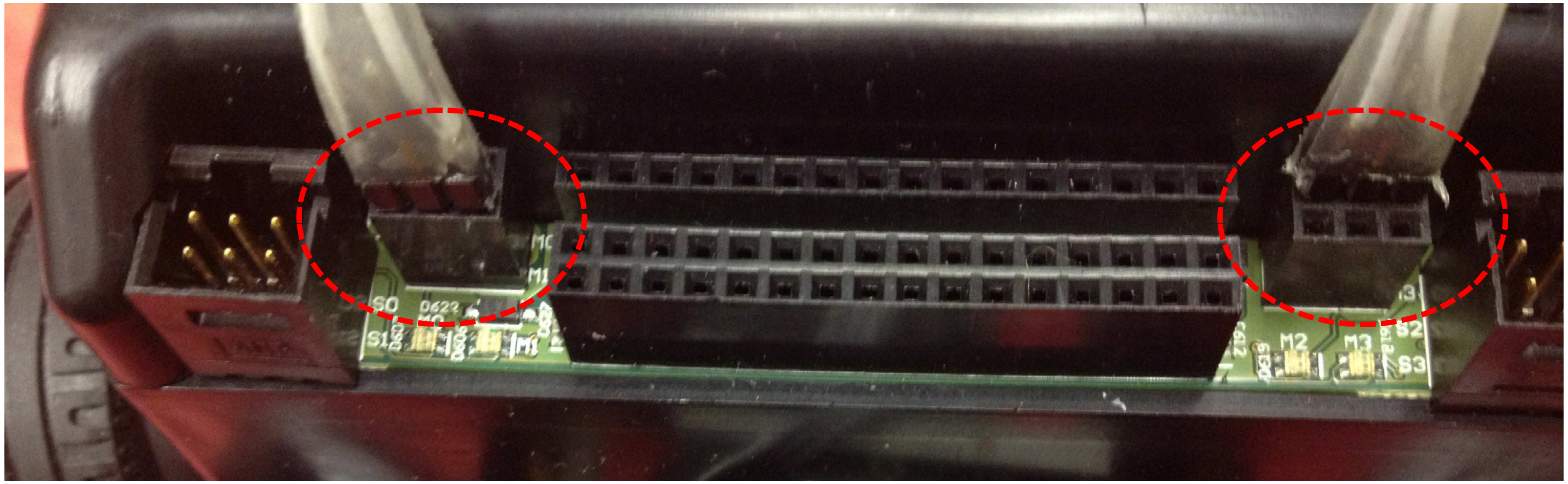
Drive motors have a 2 prong plug

KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Plugged in motors

Motor Ports 0 and 3



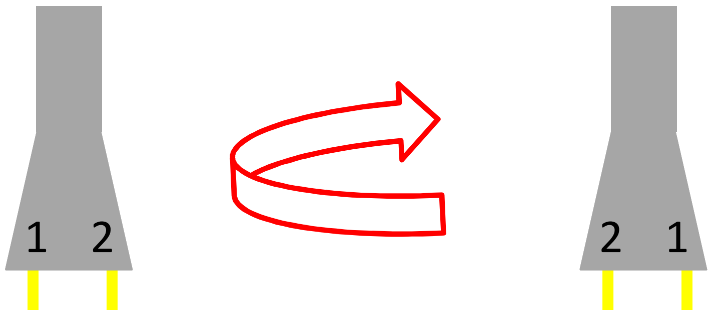
KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Motor Direction

Motors have grey wires with 2 prongs on the plug

- o There is no left or right or colored wire
- o You can plug these in two different ways
  - Motors rotate in the direction that the electricity (electrons) move through them. One direction is clockwise and the other direction is counterclockwise



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

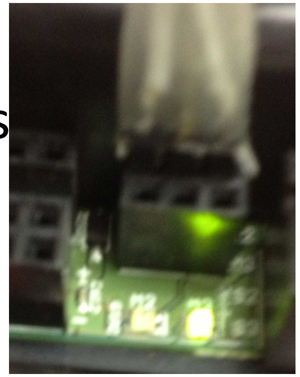
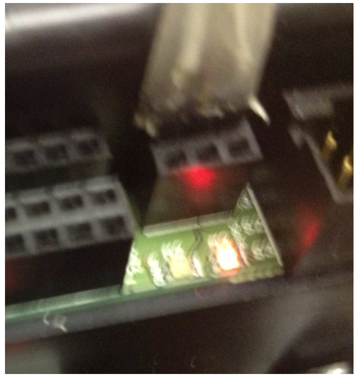
KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Motor Port & Direction Check

There is an easy way to check this!

- Manually rotate the tire 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 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

Functions to Use

STUDENT ACTIVITY

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

Computer scientists start counting at 0 NOT 1

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

Turns on motor port 0 at 100% power. You can select any power level up to 100%

A positive number should drive the motors forward. If not, switch the motor plug 180°.

A negative number will drive the robot in reverse. If the motors are set up opposite one another the robot will go in a circle.

```
msleep(XXXX);
  //Pause
ao();
  //All Off
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Explain using comments

You can use a flow chart and then translate that into comments.

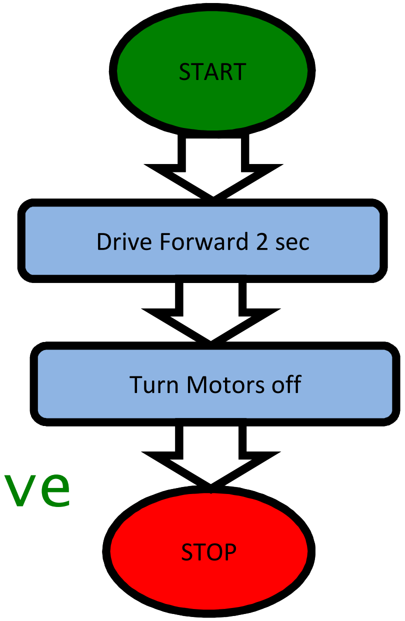
Using //comments as pseudocode is a great way to start.

If you forget which functions to use, look at your cheat sheet.

Lets make a robot move!

Write a program for your robot to move forward for 2 seconds and then stop.

- Use motor ports 0 and 3
  - Check the LEDs to make sure you are in the right ports and going in the right direction



Psuedocode (Task Analysis)

```
// 1. Drive forward

// 2. Pause program for 2 seconds to give
the motors time to move

// 3. Turn everything off
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

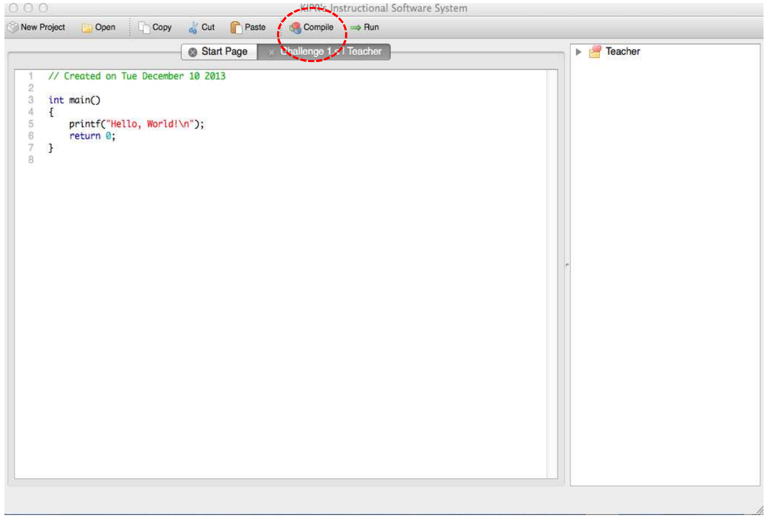
Lets make a robot move!

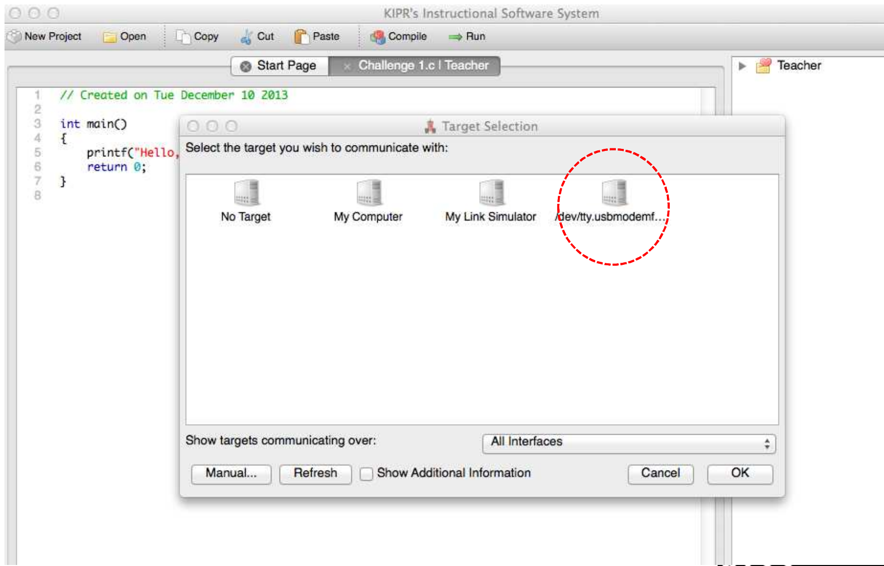
Now that you have written your program, you must Compile it

(The compile button sends the program to your target to be compiled)

A *Target Selection* window will appear

Select the usb target (this is your robot)

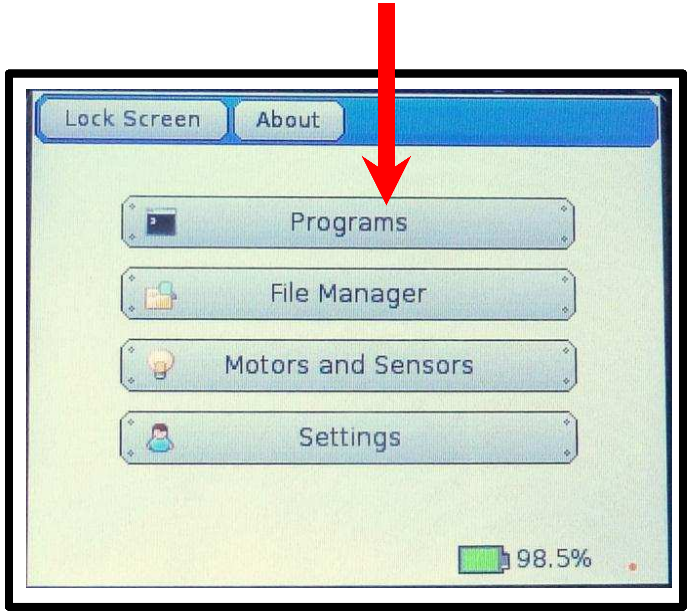


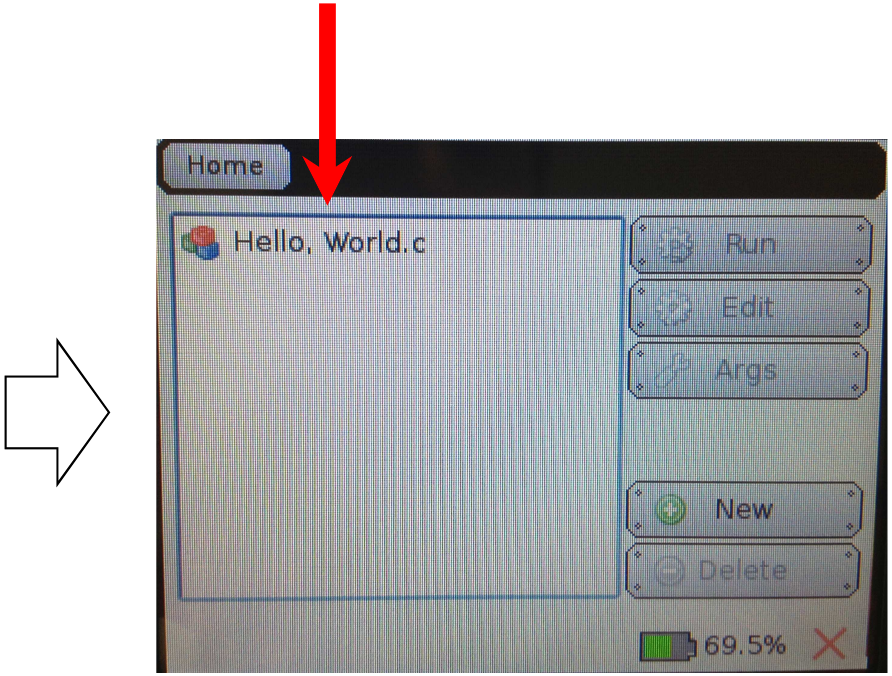


KISS INSTITUTE FOR PRACTICAL ROBOTICS

Running your program on the Link

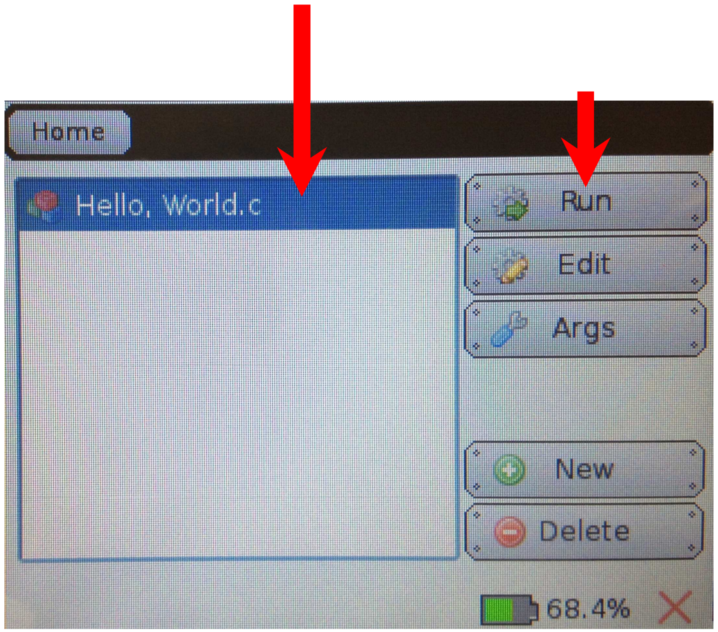
- Select the program button that will take you to a list of programs currently on the Link controller.





Running your program on the Link

- Highlight the program you want to run, in this case, *"Hello World"*, and then push the "Run" button



Activity 3 Solution

```
#/************************************************************
*********  Activity 3
************************************************************/
int main()
  {
    // 1. Drive Forward
    motor(0,80); //Motor in port 0 at 80%
    motor(3,80); //Motor in port 3 at 80%

    // 2. Give time for the motors to move
    msleep(2000);

    // 3. Turn everything off
    ao();
    return 0;
}
```

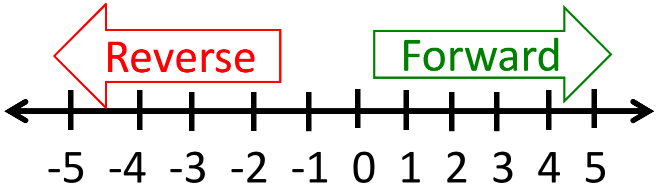
Notice the motor functions are commented

Positive (+) numbers should move the motors in a clockwise direction (forward). If not, Reverse the motor plug 180° where it plugs into the controller. If your robot goes in a circle one motor is either not moving (check the plug) or they are moving in opposite directions.

TEACHER'S RESOURCE

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

LET'S MOVE! Materials/Supplies

TEACHER'S RESOURCE

1. You need a surface to run the robot on

YOU CAN BUY VINYL SURFACES FOR THIS CURRICULUM

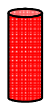
http://botballstore.org/product/elementary-botball-challenge-surfaces

Use the floor, desktop (watch for falling robots), a piece of white or light colored foam or poster board or a vinyl or paper mat as a robot testing track

- o You need an area marked as the starting line (a piece of black tape works well or you can mark it with a black marker)

1. You need an object to navigate to
   - Can of soda, foam block, whiteboard eraser, etc. will work
2. A measuring device and a timer will be useful

Soda Can



KISS INSTITUTE FOR PRACTICAL ROBOTICS

LET'S MOVE!
Activity/mini contests

Using the simple motor function `motor();` and `msleep();` you can have the students work on fun challenges.

These activities can all be completed using hard coding ("dead reckoning") and simple motor control functions without the use of any sensors. This is a good place to start and will teach the students how hard it is to be consistent using dead reckoning.

- This is a good time to bring up controlling variables when they set up their robot- is it the same every time? How could you make it the same (using a jig or ruler to control how they set it at the starting line)

Once they have the skills down of forward, backwards, stop, turn then we can move on and start adding sensors and decision making into the programs.

TEACHER'S RESOURCE

KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

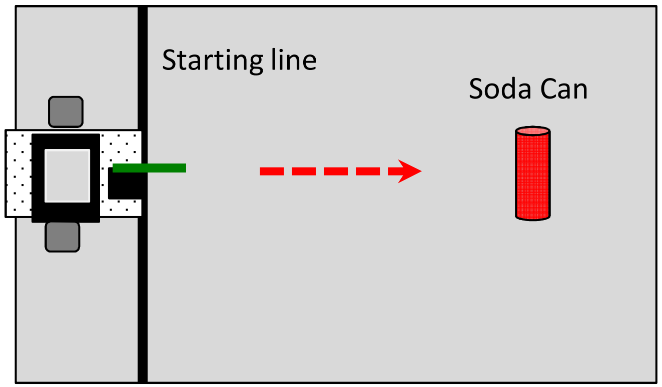
TEACHER'S RESOURCE

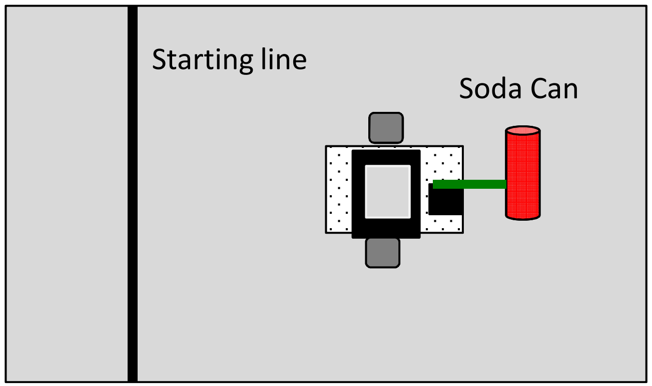
Touch the Can

Robots must start on or behind the starting mark and move to the object with the goal of touching the object in the shortest amount of time

Extensions

- Move the can to various distances
- Make the object smaller and harder to navigate to
- Math- have them measure the distance to the object and time the robot and then calculate rate/speed
    - Speed = Distance/Time





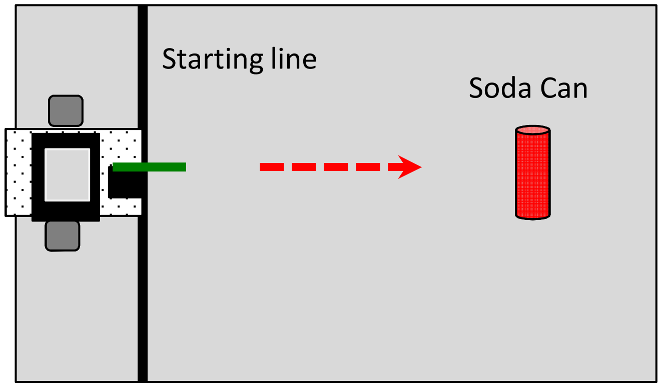
KISS INSTITUTE FOR PRACTICAL ROBOTICS

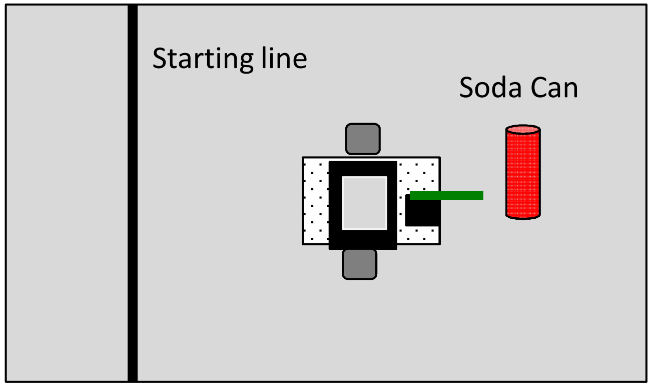
STUDENT ACTIVITY

TEACHER'S RESOURCE

Closest to the Can

1. Robots must start on or behind the starting mark and move to 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 or have them average several runs or add the distances together for a grand total
2. Move the can to various distances and locations





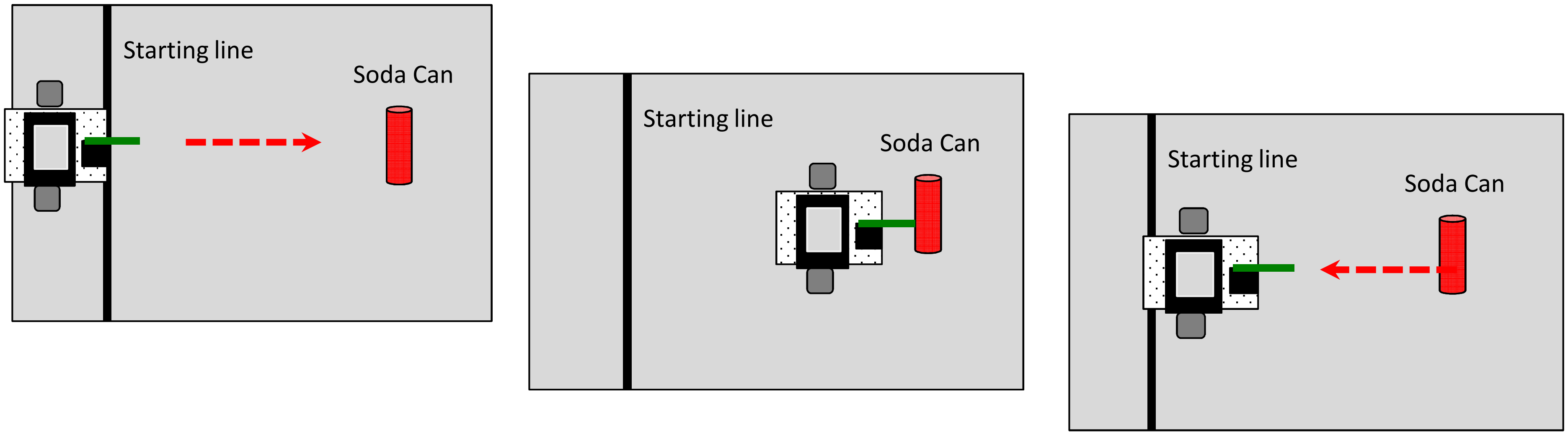
KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

TEACHER'S RESOURCE

Closest to/touch the Can and "Go Home"

1. A variation on touch the can and closest to the can.
2. After stopping closest/touching the can, back the robot up until touching the starting line
    - Move the can to various distances



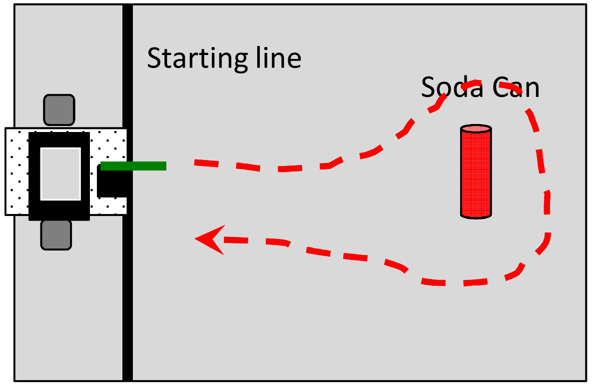
KISS INSTITUTE FOR PRACTICAL ROBOTICS

Circle the Can and "Go Home"

STUDENT ACTIVITY

TEACHER'S RESOURCE

1. Brings in the concept of turning
   - If you touch the can you must start over
   - The quickest trip is the winner
   - Move the can to various distances
   - Make them go clockwise and then counter clockwise

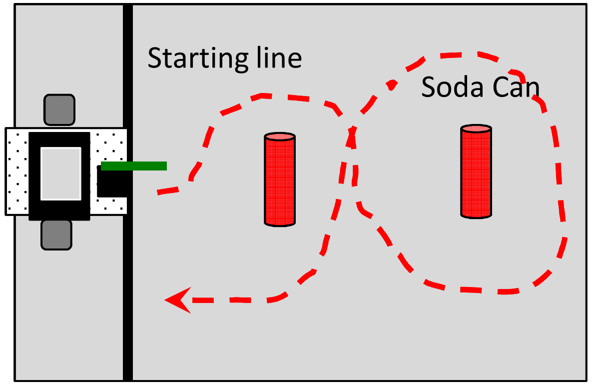


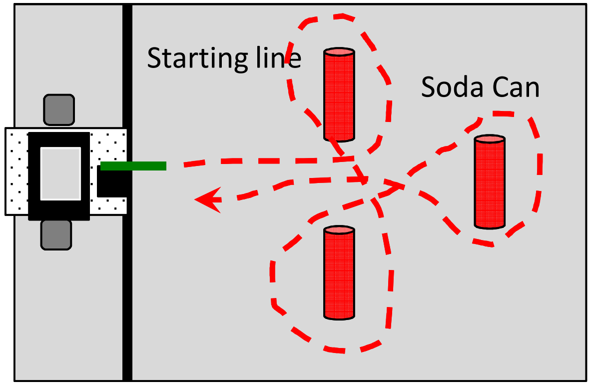
KISS INSTITUTE FOR PRACTICAL ROBOTICS

Circle the Can(s) and "Go Home"

Variation on Circle the Can

1. Have them make a figure 8 around two objects
2. Barrel Race (have them go around three cans)



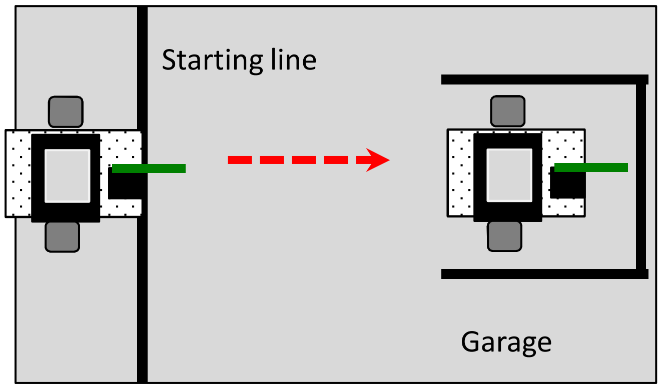


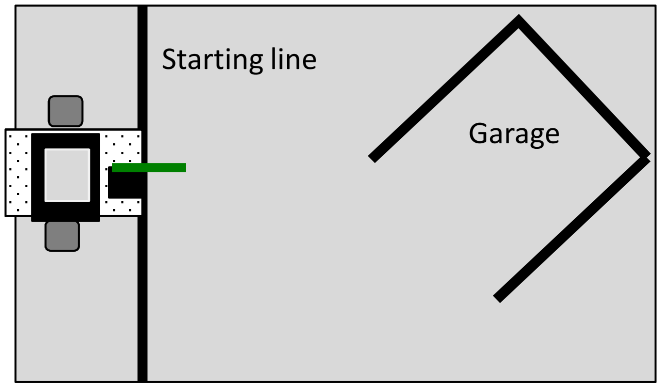
Park in the Garage

STUDENT ACTIVITY

TEACHER'S RESOURCE

1. Robots must start on or behind the starting mark and park in the garage (box or tape outline on board)
   - Start with the garage straight across from the starting line
     - o Garage can be roomy and then make it a tight fit
   - If they touch the garage they must start over at the starting line
   - Move the garage to various distances and locations



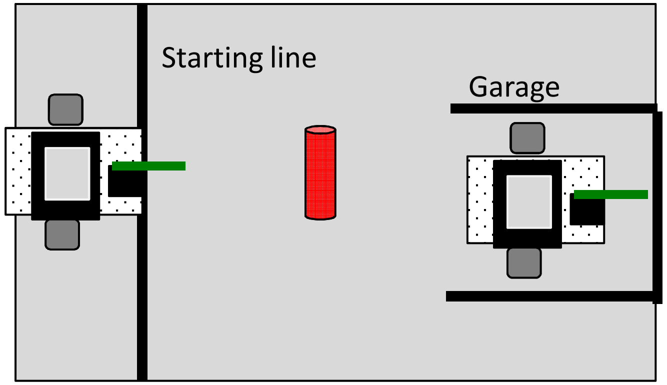


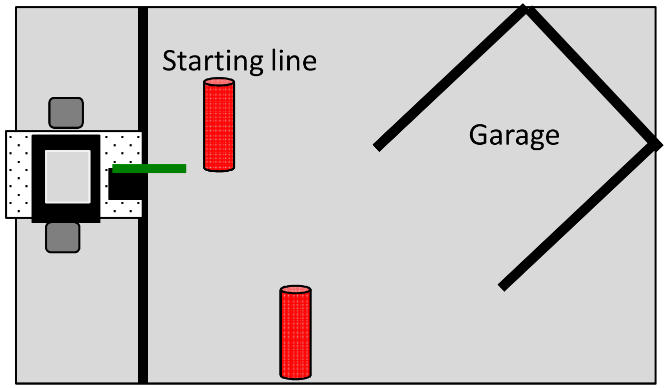
KISS INSTITUTE FOR PRACTICAL ROBOTICS

Park in the Garage and Miss the Bicycle

"Park in the Garage" variation

- Place an object(s) between the starting line and garage

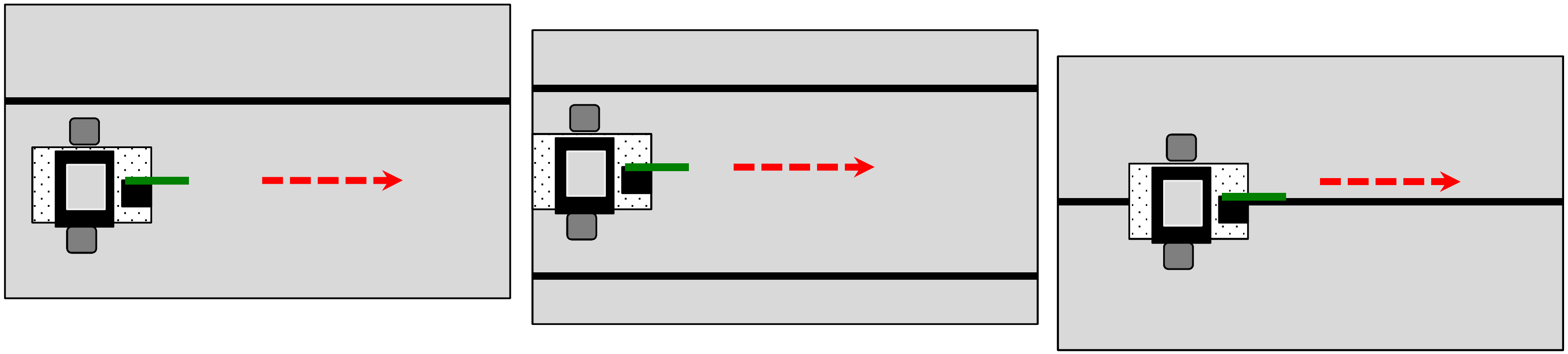




Walk the Line

Brings in the concept of driving in a straight line

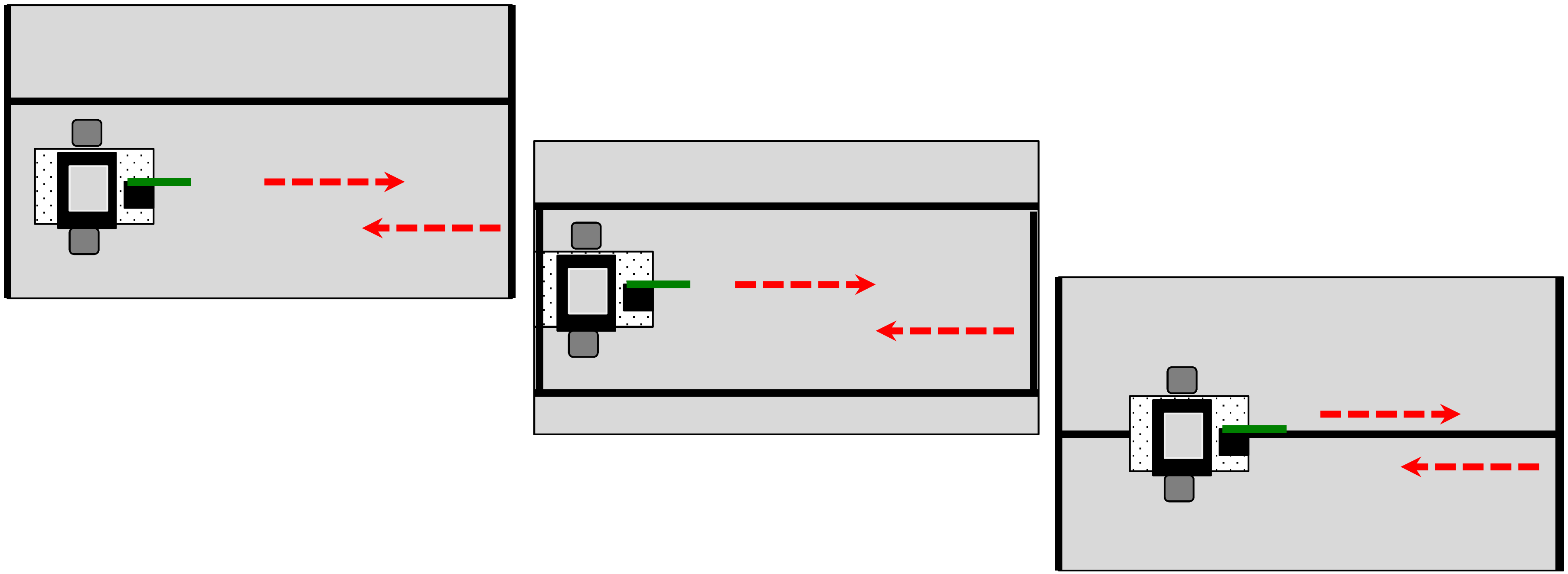
- Robot must move without touching the line (easiest to hardest below)
    - o You can use one line and have the robot move down the side without touching it
        - Make this a time trial-quickest time without touching (faster is harder to control)
    - o You can make a lane and have the robot drive down it without touching either side.
        - Increase difficulty by making the lane narrower
    - o You can use one line and have the robot straddle it with the goal of running the full length without either wheel touching the line



Variations on Walk the Line

Same as before only have them stop and go backwards without touching the line as well

- Add a starting line to begin and a finish line the robot must touch before backing up



STUDENT ACTIVITY

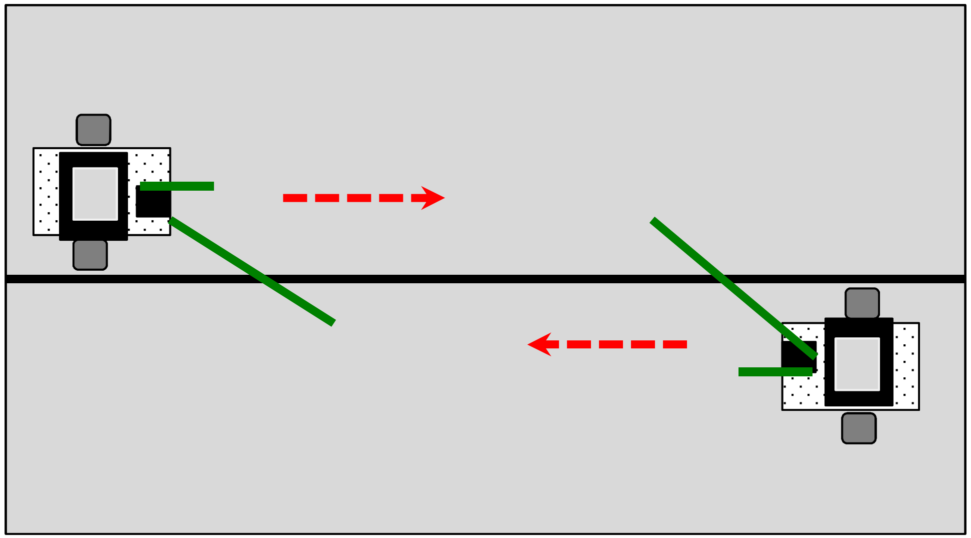
Variations on Walk the Line- Jousting!

TEACHER'S RESOURCE

- Robots on opposite sides of the line move towards each other and try to knock object off of other robot
  - Use whatever object is handy

Engineering Point-

Have the students engineer how they attach their lance (new unsharpened pencils work well) to their robot



KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

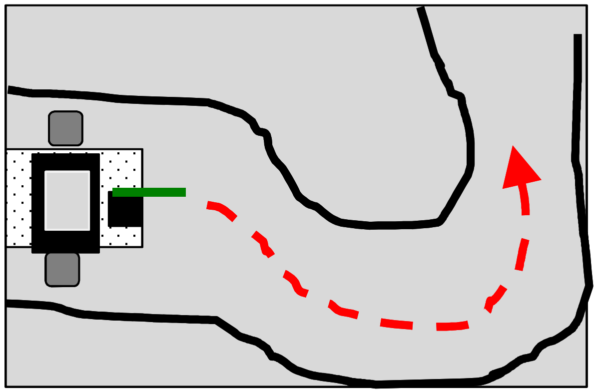
TEACHER'S RESOURCE

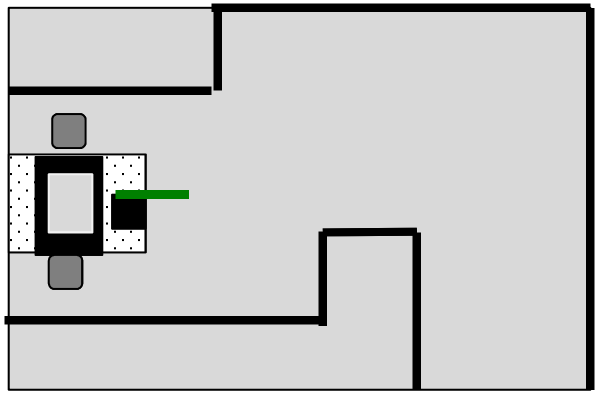
Race Track

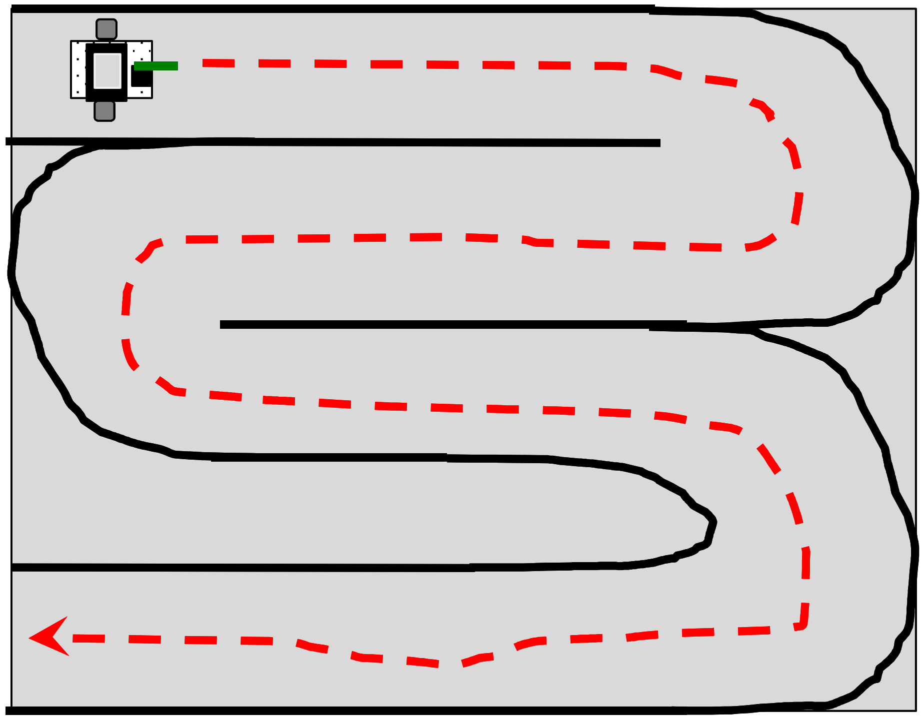
Brings in the concept of controlled driving

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 then you have to start over and the clock keeps running
- You can use a much larger track if desired (taped lanes on the classroom floor work well)
- You can use different lane setups
  - The tighter and more numerous the turns the more difficult it is
- Extension- once finished, make them stop and back up all the way to the start







KISS INSTITUTE FOR PRACTICAL ROBOTICS

Moving your robot with the other functions

TEACHER'S RESOURCE

Goals

- To use the mav() and mrp() functions to move their robot

Preparation

- You will need the DemoBot built and ready to go
- You will need computers with the KISS IDE
- You will need the USB download cable

Activity

Follow the slides to make the robot move using mav() and mrp()

KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Let’s make a robot move using `mav()` and `mrp()`

Use the provided robot or build your robot using the DemoBot building guide.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Connect the Link to your computer

- Using the USB cable connect to the Link (micro usb port)
- Turn the Link on with the black switch on the side

KISS INSTITUTE FOR PRACTICAL ROBOTICS

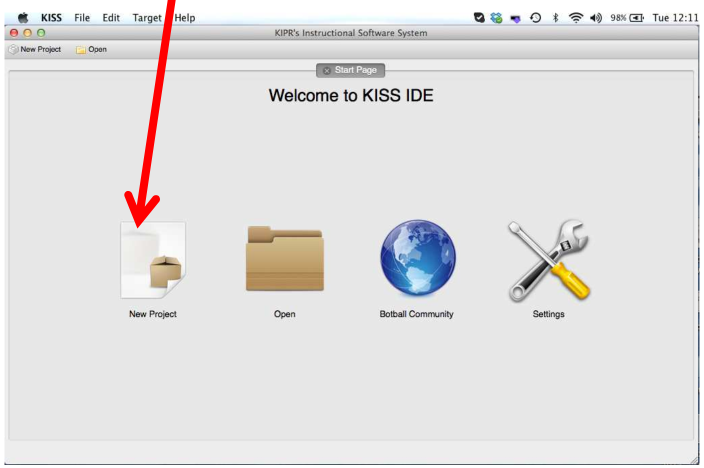
STUDENT ACTIVITY

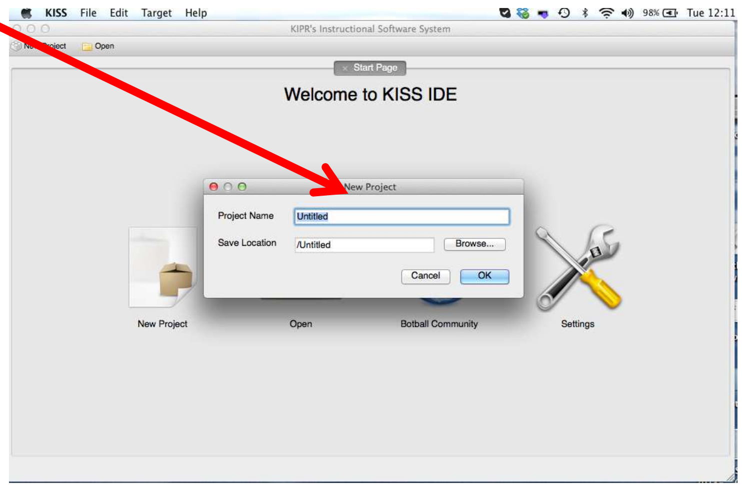
Launch the KISS IDE

(if not already running) If running just select new project

- Start the KISS IDE by clicking on its icon to get the welcome screen

- Click on the *"New Project"* icon
- You will have to name and save your project





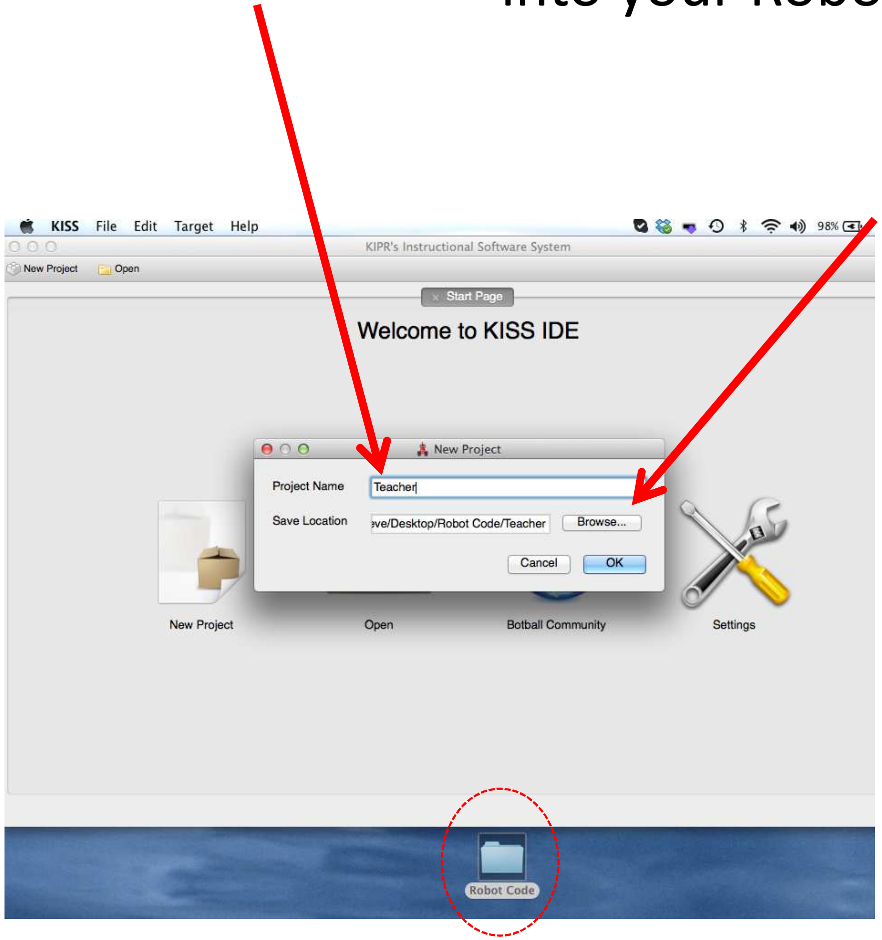
KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Writing Your Program

- Name your new project "Something Descriptive"

Use the Browse button to save the project into your Robot Code folder on your desktop

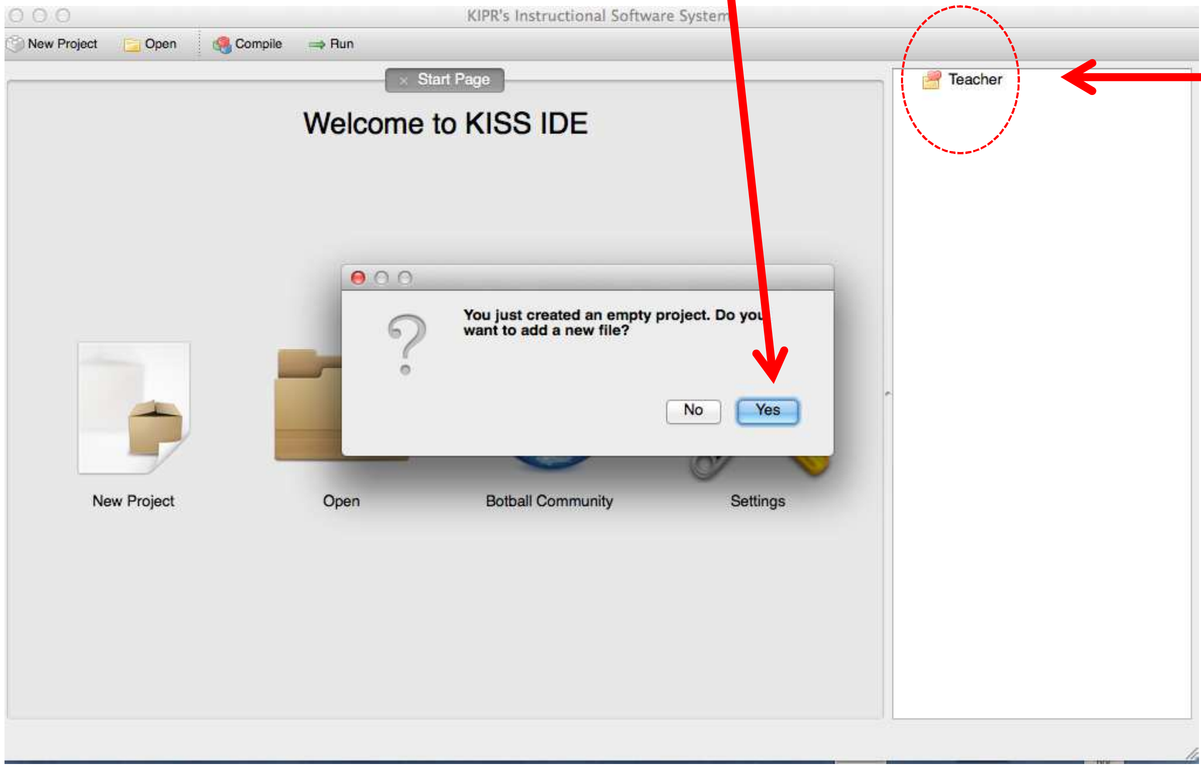


KISS INSTITUTE FOR PRACTICAL ROBOTICS

Writing Your First Program

STUDENT ACTIVITY

- You will be asked if you want to add a new file



Notice your project "teacher" now appears on the side bar

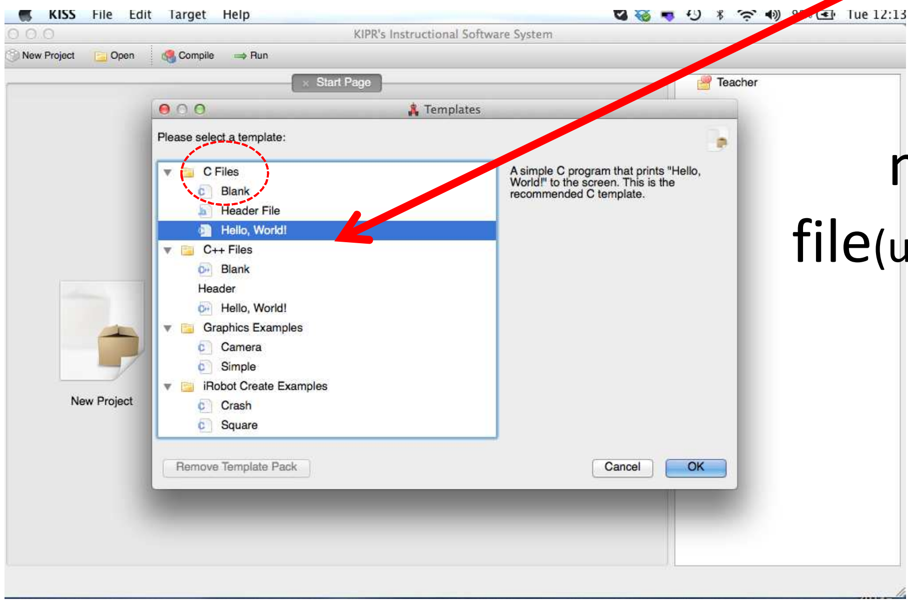
KISS INSTITUTE FOR PRACTICAL ROBOTICS

Writing Your First Program

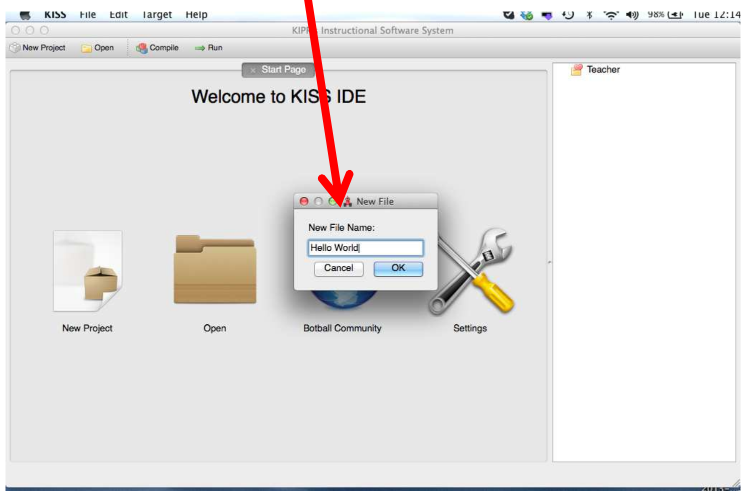
STUDENT ACTIVITY

You now need to select a template

Select under the C files folder, *Hello, World!*



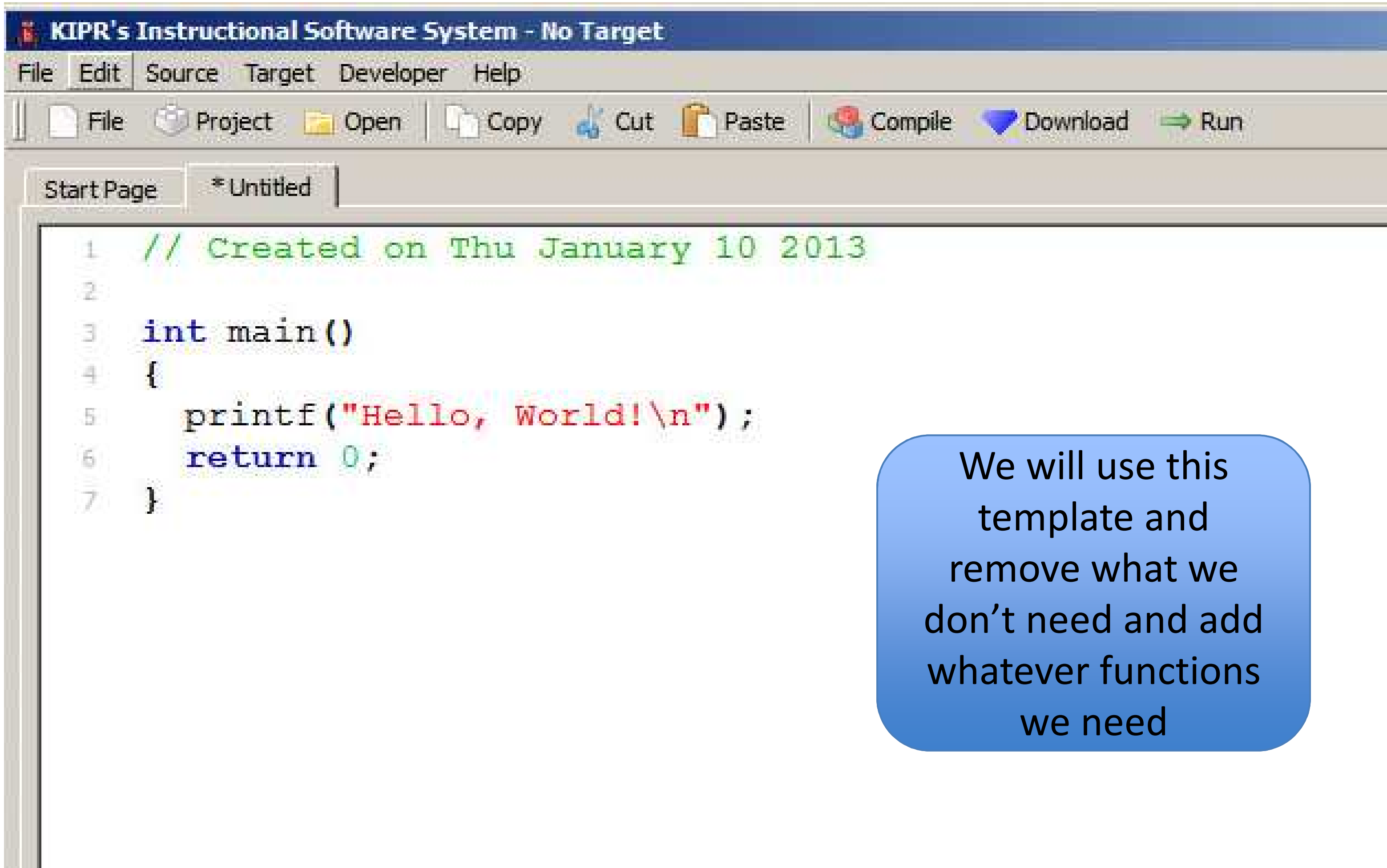
You will be prompted to name and save the new file(use "something descriptive")



KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

The C Template: Hello, World!



We will use this template and remove what we don’t need and add whatever functions we need

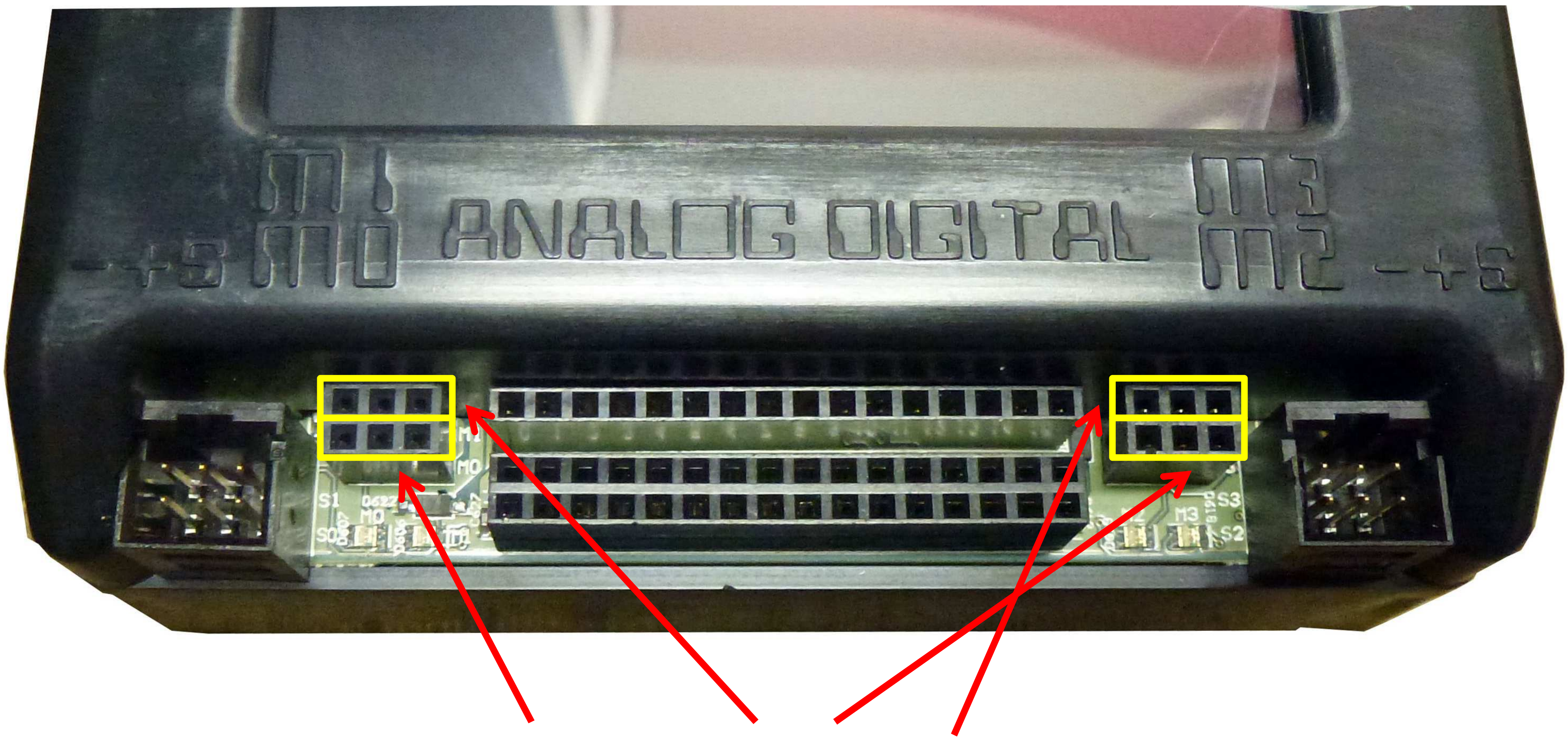
INSTITUTE FOR PRACTICAL ROBOTICS

Check your Robot's Motor Ports

- To program your robot, you need to know what motor ports your motors are plugged into

* REMEMBER computer scientists start counting at 0 so the motor ports are 0, 1, 2 and 3

KIPR Link Motor Ports

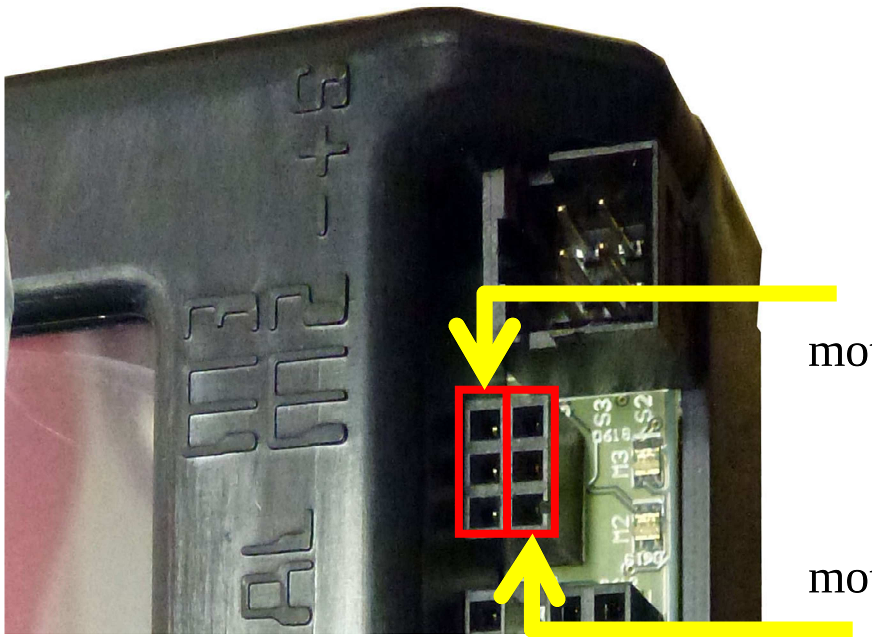


Motor ports 0 (Demobot), 1, 2, and 3 (Demobot)

KISS INSTITUTE FOR PRACTICAL ROBOTICS

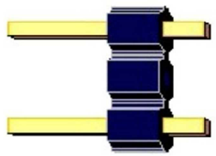
Plugging in Motors

- Motors are the ones with <u>two-prong</u> plugs with <u>2 gray wires</u>
- The KIPR Link has 4 drive motor ports numbered 0 & 1 on the left and 2 & 3 on the right
- When a port is powered it has a light that glows green for one direction and red for the other
- Plug orientation order determines motor direction, but by convention, green is forward and red reverse



motor port 3

motor port 2

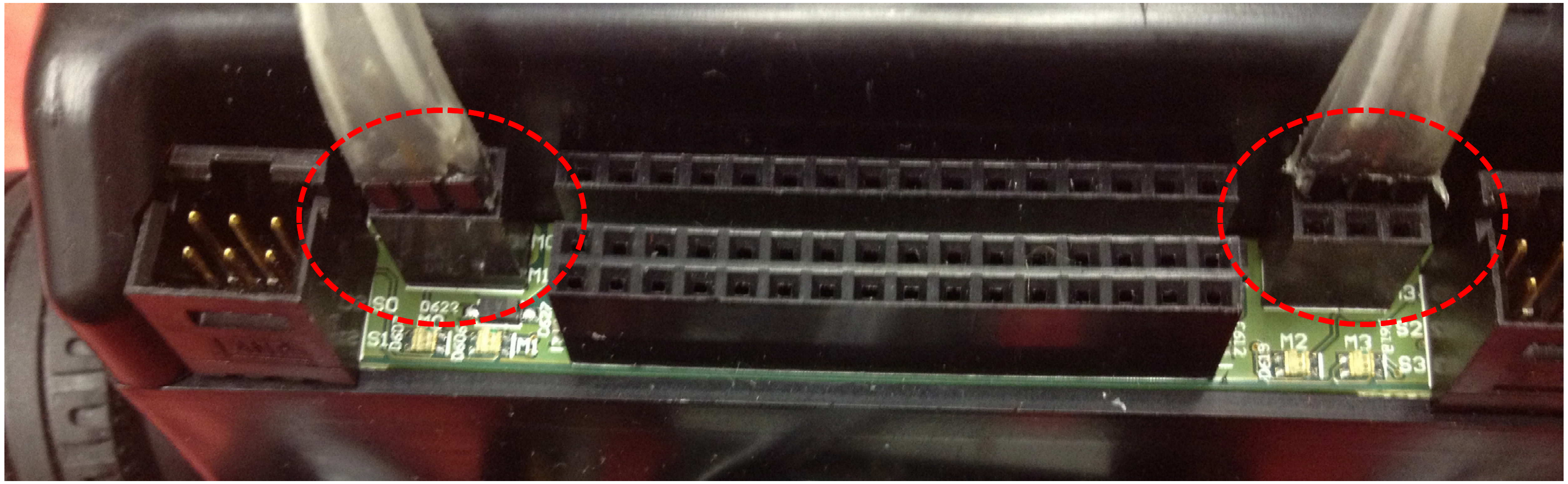


Drive motors have a 2 prong plug

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Plugged in motors

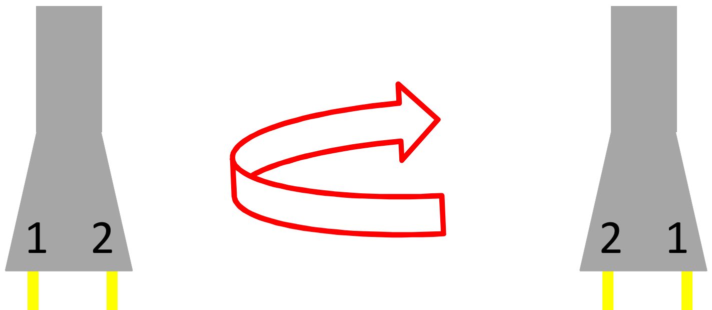
Motor Ports 0 and 3



KISS INSTITUTE FOR PRACTICAL ROBOTICS

Motor Direction

- Motors have grey wires with 2 prongs on the plug
  - o There is no left or right or colored wire
  - o You can plug these in two different ways
    - Motors rotate in the direction that the electricity (electrons) move through them one direction is clockwise and the other direction is counterclockwise

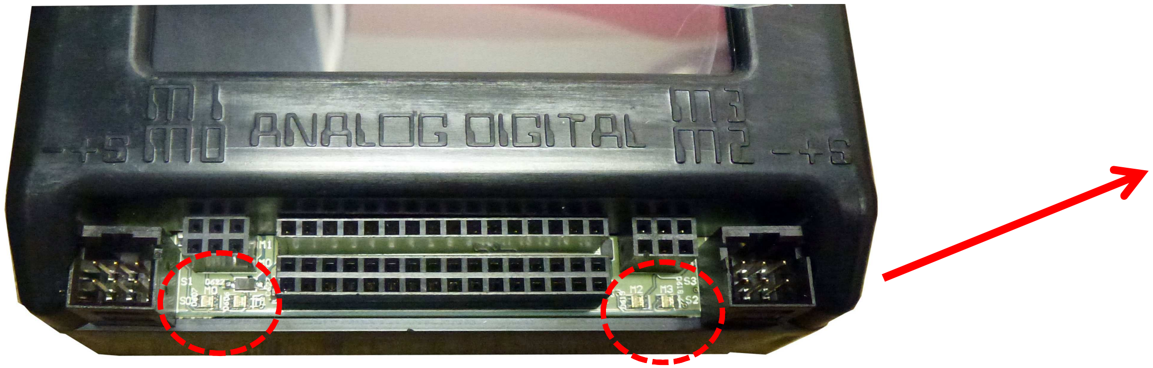


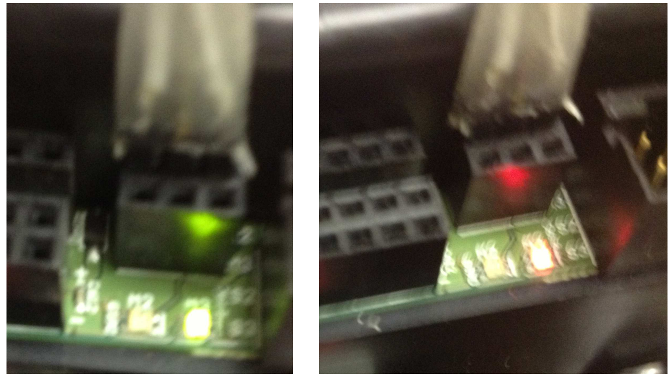
*You want your motors going in the same direction, otherwise your robot will go in circles!

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Motor Port & Direction Check

- There is an easy way to check this!
  - Manually rotate the tire 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 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

Other Motor Control Functions

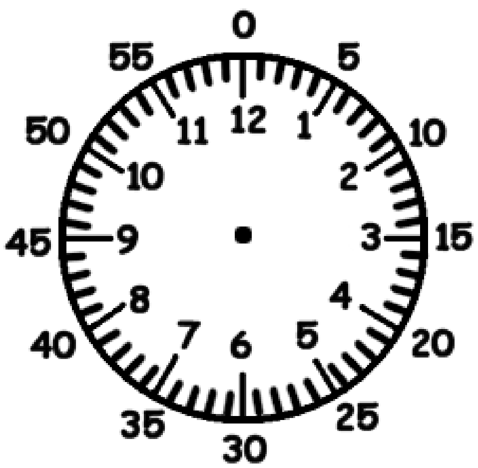
`motor(0, 100);`

- The `motor()` function is not always the best way to move your robot because it is based on a % of power (battery charge)
  - As your battery runs down the power decreases and your robot will not go as far in the same time period
  - In competition when precision is required this is not acceptable

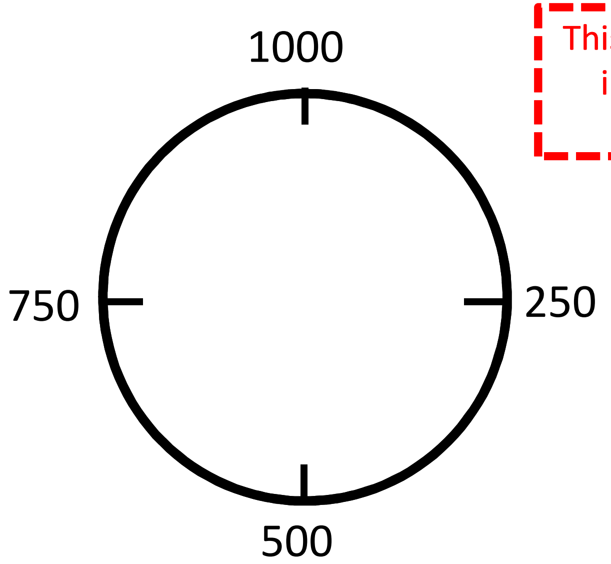
```
mav (0,1000); // Move at velocity ticks/sec

mrp (0,1000,3000);// Move to relative position in ticks
```

Ticks

- A “tick” is a unit of measurement used when talking about the rotation of a motor
- Botball motors have ~1000 ticks in one revolution
  - Great math applications doing unit conversions
    - Circumference in cm or inches = 1 revolution = ~1000 ticks



This clock is divided into 60 second increments



This wheel is divided into 1000 “tick” increments

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Other Motor Control Functions

Move At Velocity

- mav()
- mav (0, 1000)

Motor Port #

Velocity -1000 to +1000 ticks/second
-/+ indicates direction

Motor Position in Ticks
(~1000 ticks/tire revolution

Move Relative Position

- mrp (0, 1000, 3000)

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Explain using comments

You can use a flow chart and then translate that into comments.

Using //comments as pseudocode is a great way to start.

If you forget which functions to use, look at your cheat sheet.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

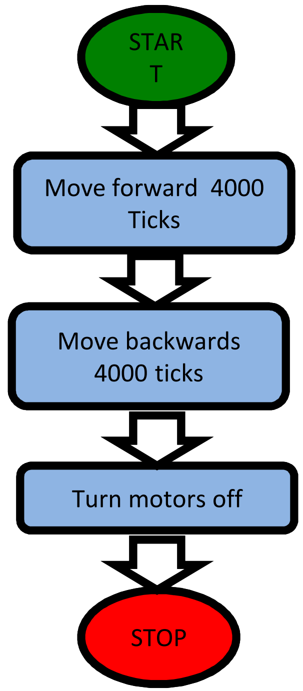
Activity 4
Move to Relative Position

STUDENT ACTIVITY

Write a program for your robot to:

Psuedocode (Task Analysis)

1. //Move motor 0 forward @1000 ticks/second to a position of 4000 ticks
2. //Move motor 3 forward @ 1000 ticks/second to position of 4000 ticks
3. //Allow 6 seconds to complete moving to position
4. //Move motor 0 backward @ 1000 ticks/second to position of -4000 ticks
5. //Move motor 3 backward @ 1000 ticks/second to position of -4000 ticks
6. //Allow 6 seconds to complete moving backwards
7. //Shut everything off



KISS INSTITUTE FOR PRACTICAL ROBOTICS

Activity 4 Solution

```
int main()
{
	mrp (0,1000,4000);	//motor 0 @ 1000 ticks/second to position of 4000
ticks
	mrp(3,1000,4000);	//motor 3 @ 1000 ticks/second to position of 4000
ticks
	msleep (6000); //Allow 6 seconds to complete, should have a 2 second
pause
	mrp (0,1000,-4000); //motor 0 @ 1000 ticks/second to position of -4000
ticks
	mrp(3,1000,-4000);	//motor 3 @ 1000 ticks/second to position of -4000
ticks
	msleep (6000); //Allow 6 seconds to complete, should stop in 4 seconds
	ao (); //All off
	return 0;
}
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Motor Commands

`motor (0,100);` Is great for turning gears or winding up string on a pulley

Not so much for driving robots as it is dependent on the battery charge

`mav (0, 1500);` Is great for driving robots and not as dependent on battery charge

Greater precision of control

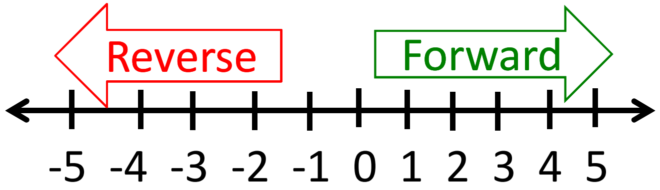
Must use msleep (); correctly

`mrp (0,1000,4000);` Provides the most precise level of control

Most complicated to use

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

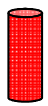
- Have one wheel go faster or slower than the other
- Have one wheel move while the other ones is stopped (friction is less of a factor when both wheels are moving)
- Have one wheel move forward while the other is moving backwards

LET'S MOVE! Materials/Supplies

TEACHER'S RESOURCE

1. You need a surface to run the robot on
   - Use the floor, desktop (watch for falling robots), a piece of white or light colored foam or poster board or a vinyl or paper mat as a robot testing track
     - You need an area marked as the starting line (a piece of black tape works well or you can mark it with a black marker)
2. You need an object to navigate to
   - Can of soda, foam block, whiteboard eraser, etc. will work
3. A measuring device and a timer will be useful

Soda Can



KISS INSTITUTE FOR PRACTICAL ROBOTICS

LET'S MOVE!

Activity/mini contests

Using the simple motor function mav();, mrp() and msleep(); you can have the students work on fun challenges.

These activities can all be completed using hard coding ("dead reckoning") and simple motor control functions without the use of any sensors. This is a good place to start and will teach the students how hard it is to be consistent using dead reckoning.

- This is a good time to bring up controlling variables when they set up their robot- is it the same every time? How could you make it the same (using a jig or ruler to control how they set it at the starting line)

Once they have the skills down of forward, backwards, stop, turn then we can move on and start adding sensors and decision making into the programs.

STUDENT ACTIVITY

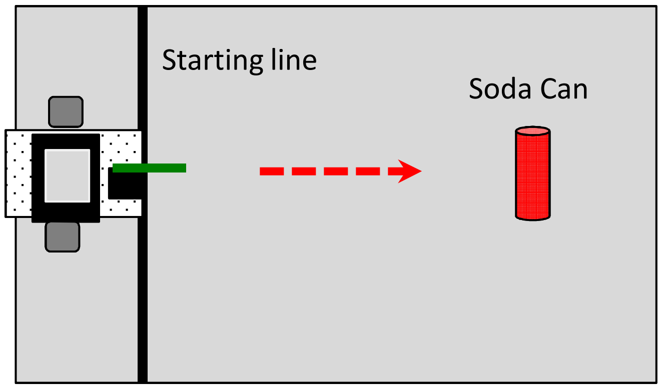
TEACHER'S RESOURCE

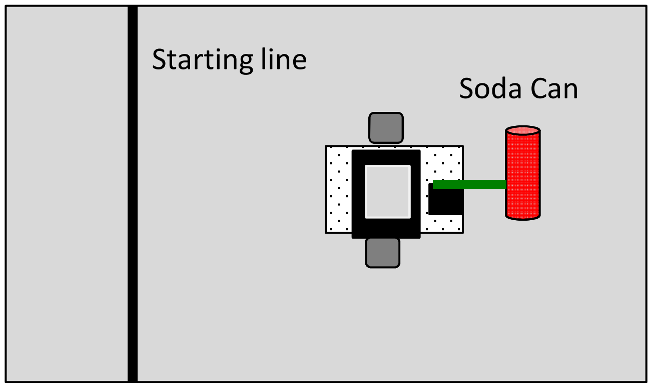
Touch the Can

Robots must start on or behind the starting mark and move to the object with the goal of touching the object in the shortest amount of time

Extensions

- Move the can to various distances
- Make the object smaller and harder to navigate to
- Math- have them measure the distance to the object and time the robot and then calculate rate/speed
    - Speed = Distance/Time





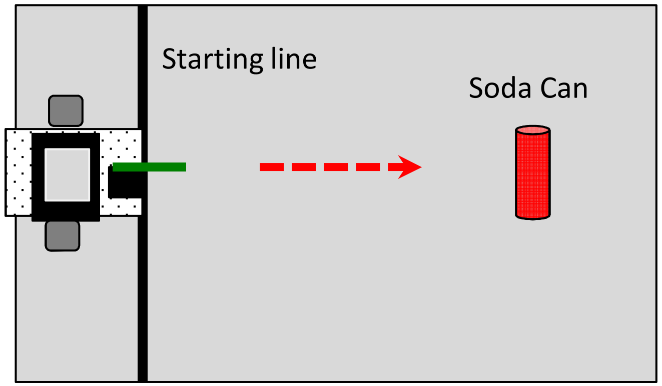
KISS INSTITUTE FOR PRACTICAL ROBOTICS

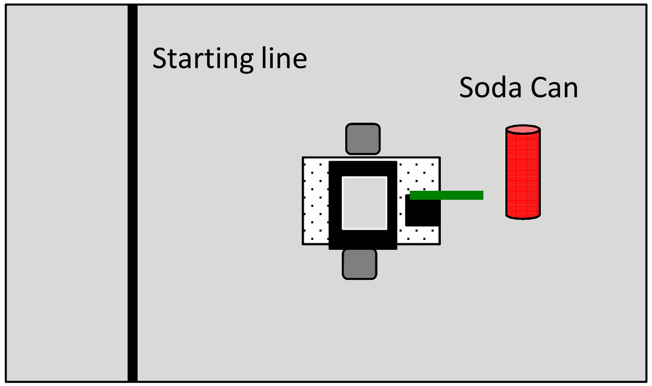
STUDENT ACTIVITY

TEACHER'S RESOURCE

Closest to the Can

1. Robots must start on or behind the starting mark and move to 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 or have them average several runs or add the distances together for a grand total
2. Move the can to various distances and locations





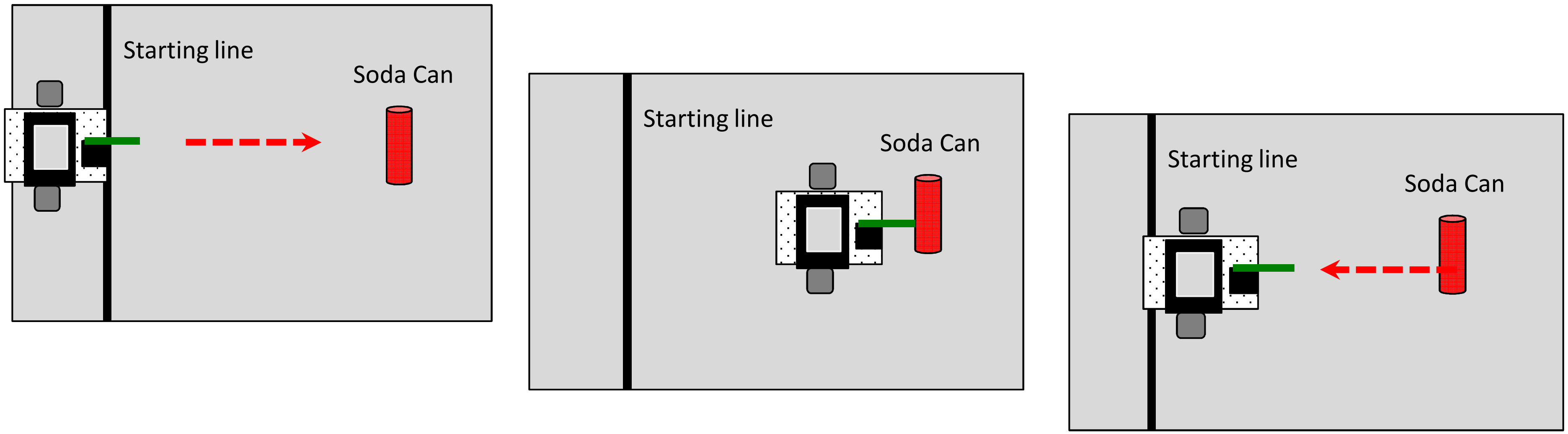
KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

TEACHER'S RESOURCE

Closest to/touch the Can and "Go Home"

1. A variation on touch the can and closest to the can.
2. After stopping closest/touching the can, back the robot up until touching the starting line
   - Move the can to various distances



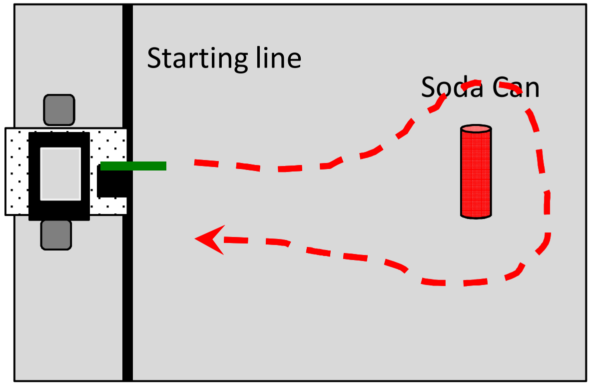
KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

TEACHER'S RESOURCE

Circle the Can and "Go Home"

1. Brings in the concept of turning
   - If you touch the can you must start over
   - The quickest trip is the winner
   - Move the can to various distances
   - Make them go clockwise and then counter clockwise



KISS INSTITUTE FOR PRACTICAL ROBOTICS

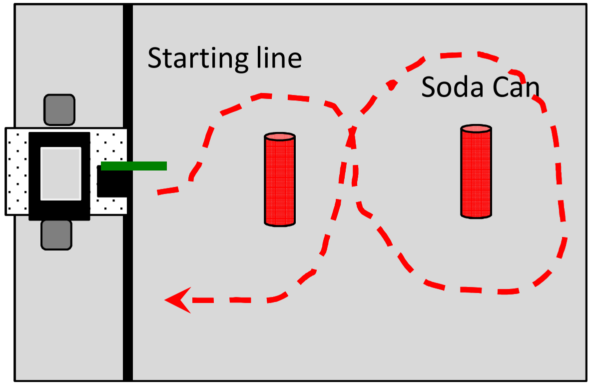
STUDENT ACTIVITY

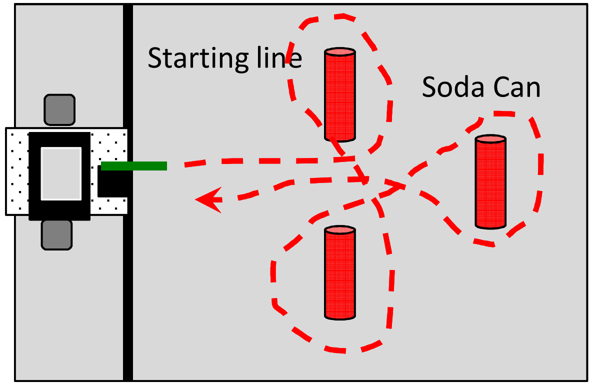
TEACHER'S RESOURCE

Circle the Can(s) and "Go Home"

Variation on Circle the Can

1. Have them make a figure 8 around two objects
2. Barrel Race (have them go around three cans)





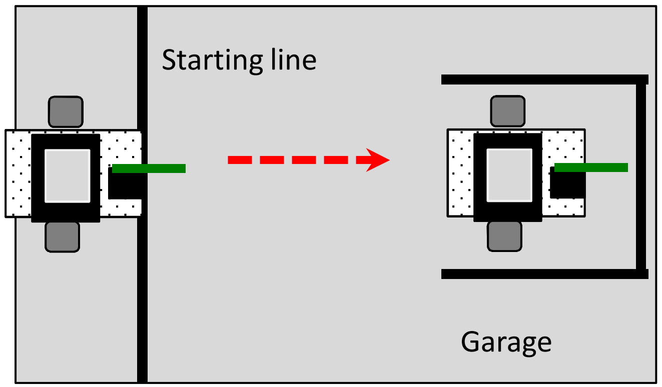
KISS INSTITUTE FOR PRACTICAL ROBOTICS

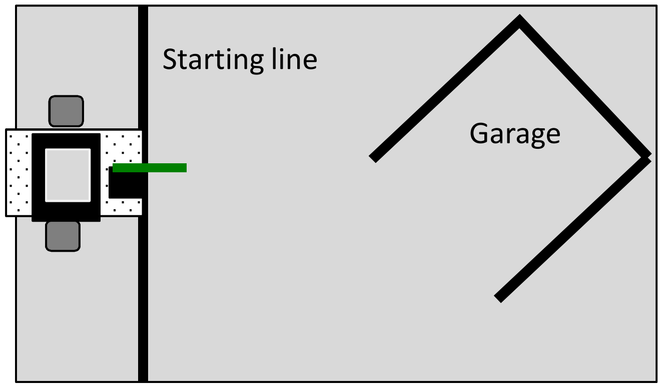
STUDENT ACTIVITY

TEACHER'S RESOURCE

Park in the Garage

1. Robots must start on or behind the starting mark and park in the garage (box or tape outline on board)
   - Start with the garage straight across from the starting line
     - Garage can be roomy and then make it a tight fit
     - If they touch the garage they must start over at the starting line
   - If they touch the garage they must start over at the starting line
   - Move the garage to various distances and locations



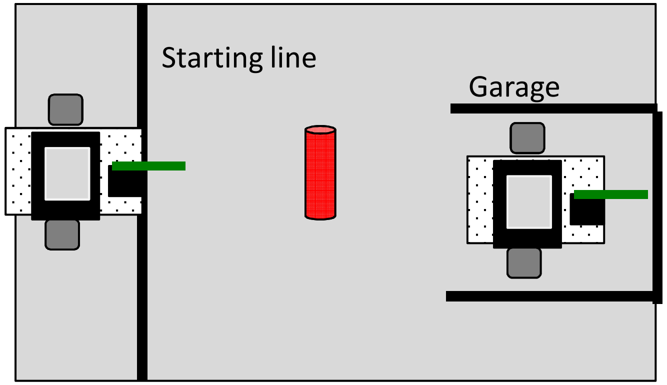


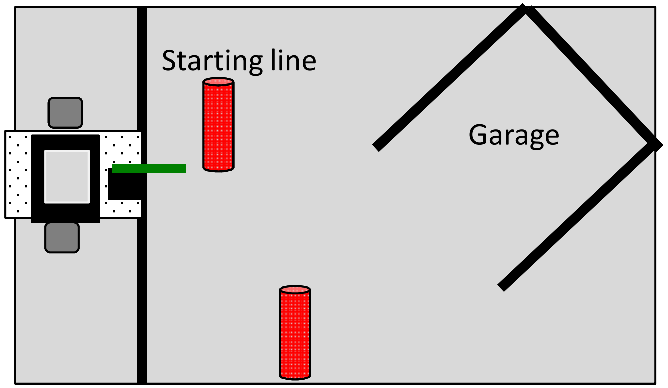
KISS INSTITUTE FOR PRACTICAL ROBOTICS

Park in the garage and Miss the Bicycle

"Park in the Garage" variation

- Place an object(s) between the starting line and garage

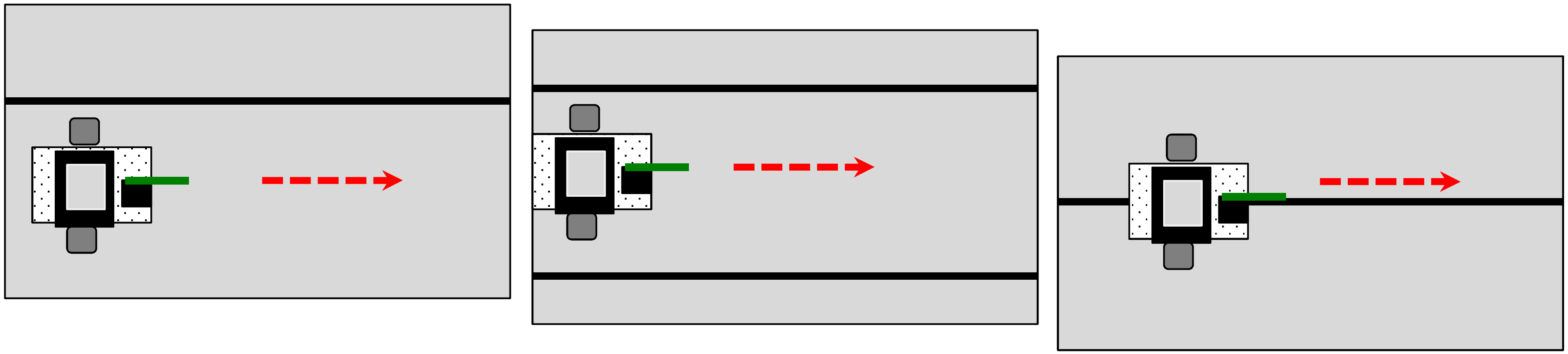




Walk the Line

Brings in the concept of driving in a straight line

- Robot must move without touching the line (easiest to hardest below)
  - You can use one line and have the robot move down the side without touching it
    - Make this a time trial-quickest time without touching (faster is harder to control)
  - You can make a lane and have the robot drive down it without touching either side.
    - Increase difficulty by making the lane narrower
  - You can use one line and have the robot straddle it with the goal of running the full length without either wheel touching the line



KISS INSTITUTE FOR PRACTICAL ROBOTICS

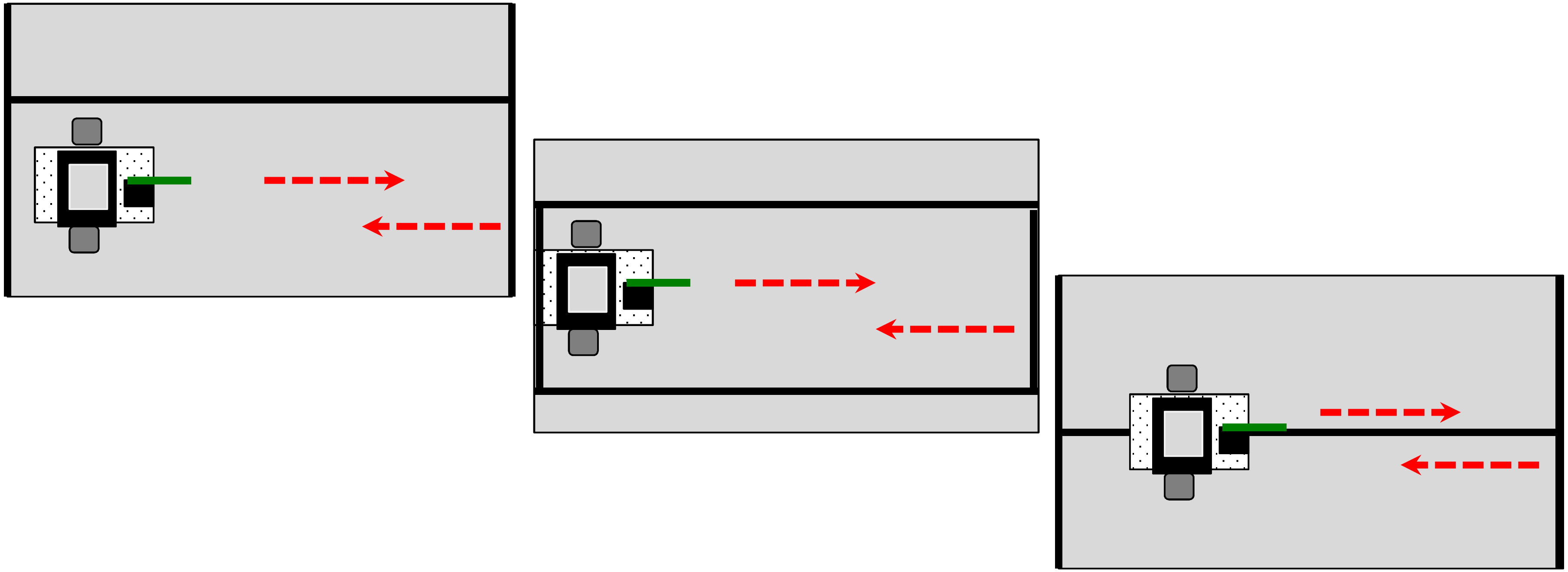
Variations on Walk the Line

STUDENT ACTIVITY

TEACHER'S RESOURCE

Same as before only have them stop and go backwards without touching the line as well

- Add a starting line to begin and a finish line the robot must touch before backing up



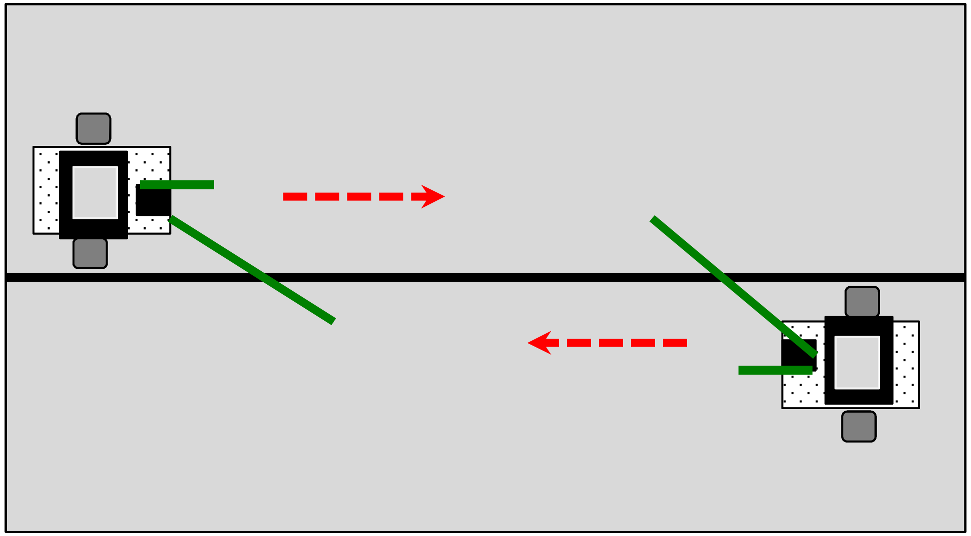
KISS INSTITUTE FOR PRACTICAL ROBOTICS

Variations on Walk the Line- Jousting!

- Robots on opposite sides of the line move towards each other and try to knock object off of other robot
    - Use whatever object is handy

Engineering Point-

Have the students engineer how they attach their lance (new unsharpened pencils work well) to their robot



STUDENT ACTIVITY

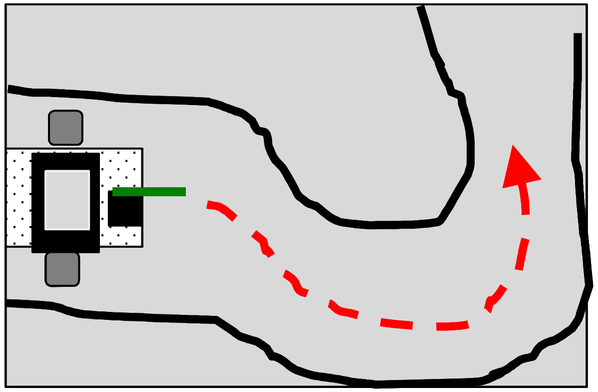
TEACHER'S RESOURCE

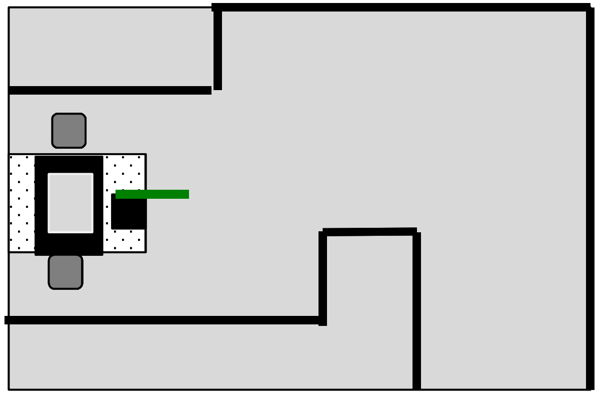
Race Track

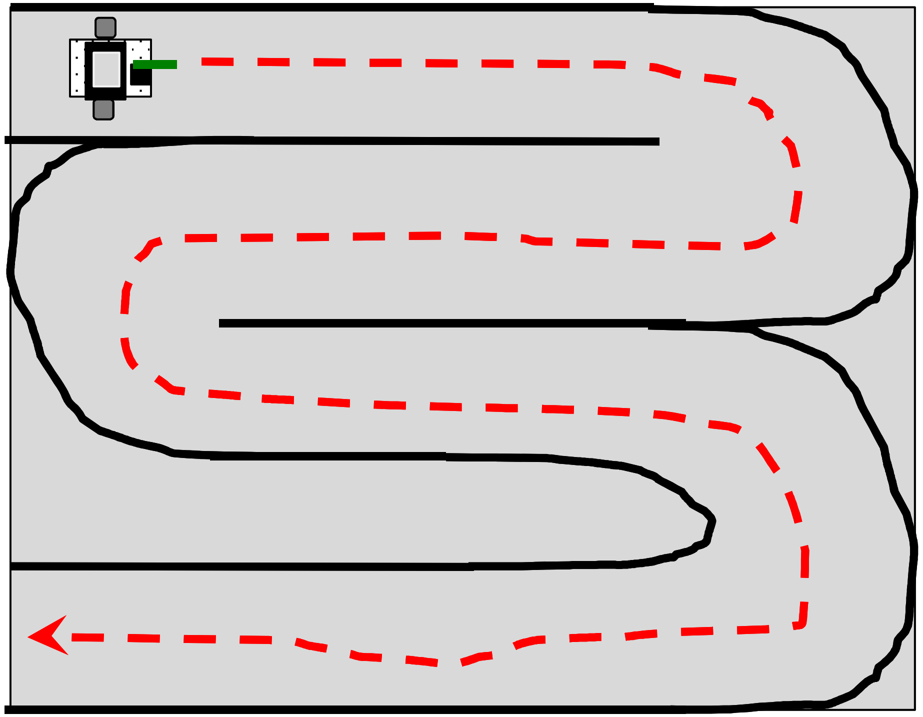
Brings in the concept of controlled driving

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 then you have to start over and the clock keeps running
- You can use a much larger track if desired (taped lanes on the classroom floor work well)
- You can use different lane setups
  - The tighter and more numerous the turns the more difficult it is
- Extension- once finished, make them stop and back up all the way to the start







KISS INSTITUTE FOR PRACTICAL ROBOTICS

Functions

TEACHER'S RESOURCE

Goals

- To help students understand functions and how write their own functions for repetitive tasks
- To understand that functions have two parts, a prototype and a definition
- To understand how to write a function prototype and definition
- To put a function prototype and definition into their code

Preparation

- Have KISS IDE up and running
- Have a robot ready to go
- Have markers if you choose to mark the path of the robots

Activity

- Have the students program their robots to drive a square (you can set the dimensions however you would like)
- After being successful work through the "How to Write a Function" activities
- Have students complete the geometric activities using functions they write themselves

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The Importance of Functions

TEACHER'S RESOURCE

- Now that you have the hard coding down with the robots let's move on to something that teaches you how to write your own functions
  - Start with the simple draw a square activity

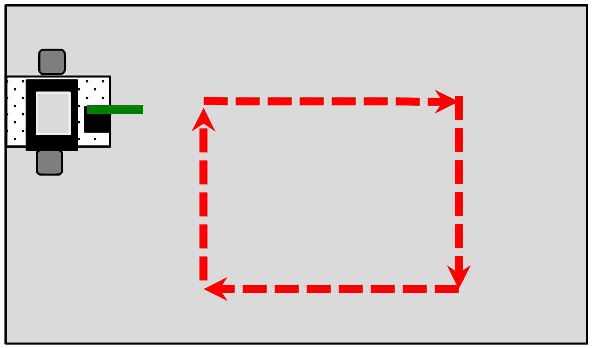
KISS INSTITUTE FOR PRACTICAL ROBOTICS

Get your Robot to Draw a Square

STUDENT ACTIVITY

Program your robot to draw geometric patterns

1. Start with having the robot make a 90° turn (both directions)
2. Now have the robot make a box
   - You will have to remember the path your robot is taking OR
     - Tape a marker to the back to mark on a piece of paper while the robot runs
     - Once you have mastered using a servo the robot can raise a marker up and down and actually draw on a piece of paper



KISS INSTITUTE FOR PRACTICAL ROBOTICS

Drawing a Square

```
// Created on Fri August 30 2013

int main()
{
    printf("lets drive in a square\n"); //print to screen
    msleep(2000); // pause for 2 seconds so you can read the screen
    motor (0, 100);
    motor (3, 100);
    msleep(4000); // drive forward for 4 seconds
    motor (0, 0);
    motor (3, 0);//stop motors
    motor (0, 100);
    motor (3, 20);
    msleep(2000); // turn to the right for 2 seconds
    motor (0, 0);
    motor (3, 0);//stop motors
    motor (0, 100);
    motor (3, 100);
    msleep(4000); // drive forward for 4 seconds
    motor (0, 0);
    motor (3, 0);//stop motors
    motor (0, 100);
    motor (3, 20);
    msleep(2000); // turn to the right for 2 seconds
    motor (0, 0);
    motor (3, 0);//stop motors
    motor (0, 100);
    motor (3, 100);
    msleep(4000); // drive forward for 4 seconds
    motor (0, 0);
    motor (3, 0);//stop motors
    motor (0, 100);
    motor (3, 20);
    msleep(2000); // turn to the right for 2 seconds
    motor (0, 0);
    motor (3, 0);//stop motors
    motor (0, 100);
    motor (3, 100);
    msleep(4000); // drive forward for 4 seconds
    motor (0, 0);
    motor (3, 0);//stop motors
    ao();
    return 0;
}
```

Here is some code that uses the `motor();` and `msleep();` functions to drive the robot in a square

KISS INSTITUTE FOR PRACTICAL ROBOTICS

```c
// Created on Fri August 30 2013

int main()
{
   printf("lets drive in a square\n"); //print to screen
   msleep(2000); // pause for 2 seconds so you can read the screen
   motor (0, 100);
   motor (3, 100);
   msleep(4000); // drive forward for 4 seconds
   motor (0, 0);
   motor (3, 0);//stop motors
   motor (0, 100);
   motor (3, 20);
   msleep(2000); // turn to the right for 2 seconds
   motor (0, 0);
   motor (3, 0);//stop motors
   motor (0, 100);
   motor (3, 100);
   msleep(4000); // drive forward for 4 seconds
   motor (0, 0);
   motor (3, 0);//stop motors
   motor (0, 100);
   motor (3, 20);
   msleep(2000); // turn to the right for 2 seconds
   motor (0, 0);
   motor (3, 0);//stop motors
   motor (0, 100);
   motor (3, 100);
   msleep(4000); // drive forward for 4 seconds
   motor (0, 0);
   motor (3, 0);//stop motors
   motor (0, 100);
   motor (3, 20);
   msleep(2000); // turn to the right for 2 seconds
   motor (0, 0);
   motor (3, 0);//stop motors
   motor (0, 100);
   motor (3, 100);
   msleep(4000); // drive forward for 4 seconds
   motor (0, 0);
   motor (3, 0);//stop motors
   ao();
   return 0;
}
```

Drawing a Square

Notice there are many repeated steps. For example:

```c
//drive forward for 4
  seconds
 motor(0,100);
 motor(3,100);
 msleep(4000);
```

is repeated 4 times in this program.

- And so is turn right for 2 seconds
- As well as stop motors

You will quickly learn to use copy and paste over and over again, but there is a better and easier way.

Learning to write your own functions allows you to repeat code easily.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Writing Your Own Functions

- Remember, a function is like a title to an instruction book. When you call the function it does all of the commands in the book.
  - This can be very helpful if you are doing repetitive actions such as making a 90° turn, moving straight, turning 180°, moving an arm up and closing a claw.
  - It makes it easier to read the main program and to simply change a value if needed

Remember a function has a name and arguments

```
name(arguments); = motor (0,90);
```

Variables Explained

TEACHER'S RESOURCE

Since variables in **C** have differing types, you have to specify the data type for each of your function's arguments, and the type of data returned by the function (which can be `void` if nothing is being returned).

Many of the functions in the KIPR Library like `motor ();` have this hidden.

Most of the time your students will only be dealing with `void` (no data returned) and `int` (arguments).

Data types you may use:

- `Void` Nothing is returned
- `Int` Returns an Integer (whole number such as 5)
- `Double` Returns a fraction of a whole (decimal such as 5.0)

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Function Prototypes

Take some functions you are familiar with:

`motor (0,100);` and `mrp (0,1000,5000);`

- The <u>prototype</u> or formats/name for them are:

```
void  motor (int m, int p);
```

Data type returned | Function name | Data type for arguments

```
void mrp (int motor port, int velocity, int pos);
```

You can find the prototype (format) for every function in the KIPR Help Manual “KIPR Link Library”

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Function Prototype & Definition

IF //drive forward for 4 seconds =

```
motor(0,100);
motor(3,100);
msleep(4000);
```

A prototype is the name for your function that you will use when programming
In this case the function prototype would be:

```
void drive_forward();
```

And the function definition is what the function actually does, in this case:

```
void drive_forward()
{ //definition start
motor(0,100); //runs motor 0 at 100%
motor(3,100); //runs motor 3 at 100%
msleep(4000); //turns off after 4 seconds
} //definition close (end)
```

Notice there is no terminating semicolon after the function name, because the robot needs to look for the definition

KISS INSTITUTE FOR PRACTICAL ROBOTICS

```
// Created on Fri August 30 2013
void drive_forward();
int main()
{
    printf("lets drive in a square\n"); //print to screen
    msleep(2000); // pause for 2 seconds so you can read the screen
    drive_forward();
    motor (0, 0);
    motor (3, 0);//stop motors
    motor (0, 100);
    motor (3, 20);
    msleep(2000); // turn to the right for 2 seconds
    motor (0, 0);
    motor (3, 0);//stop motors
    drive_forward();
    motor (0, 0);
    motor (3, 0);//stop motors
    motor (0, 100);
    motor (3, 20);
    msleep(2000); // turn to the right for 2 seconds
    motor (0, 0);
    motor (3, 0);//stop motors
    drive_forward();
    motor (0, 0);
    motor (3, 0);//stop motors
    motor (0, 100);
    motor (3, 20);
    msleep(2000); // turn to the right for 2 seconds
    motor (0, 0);
    motor (3, 0);//stop motors
    drive_forward();
    motor (0, 0);
    motor (3, 0);//stop motors
    ao();
    return 0;
}
    void drive_forward()
    {
    motor (0, 100);
    motor (3, 100);
    msleep(4000);
    }
```

Notice how the function <u>prototype</u> is BEFORE the `int main()`

```
void drive_forward();
```

And the function <u>definition</u> is provided AFTER the main program

- Note there is no semicolon after the function in the definition

```
void drive_forward()
{
motor(0,100);
motor(3,100);
msleep(4000);
}
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

```c
// Created on Fri August 30 2013
void drive_forward();
int main()
{
   printf("lets drive in a square\n"); //print to screen
   msleep(2000); // pause for 2 seconds so you can read the screen
   drive_forward();
   motor (0, 0);
   motor (3, 0);//stop motors
   motor (0, 100);
   motor (3, 20);
   msleep(2000); // turn to the right for 2 seconds
   motor (0, 0);
   motor (3, 0);//stop motors
   drive_forward();
   motor (0, 0);
   motor (3, 0);//stop motors
   motor (0, 100);
   motor (3, 20);
   msleep(2000); // turn to the right for 2 seconds
   motor (0, 0);
   motor (3, 0);//stop motors
   drive_forward();
   motor (0, 0);
   motor (3, 0);//stop motors
   motor (0, 100);
   motor (3, 20);
   msleep(2000); // turn to the right for 2 seconds
   motor (0, 0);
   motor (3, 0);//stop motors
   drive_forward();
   motor (0, 0);
   motor (3, 0);//stop motors
   ao();
   return 0;
}
   void drive_forward()
   {
   motor (0, 100);
   motor (3, 100);
   msleep(4000);
   }
```

Function Prototype (before the main)

Notice the function calls

Function definition (after the main)

Function Prototype

Now that you have your drive forward function written you can write a right turn function and put it into your program

```
// turn to the right for 2 seconds

void right_turn();      Prototype (goes before the main)

void right_turn()       Definition (goes after the main)
{
motor(0,100);
motor(3,20);
msleep(2000);
}
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Function Prototype

Now that you have your right turn function written you can write a stop motor function

```
//Stop motors

void stop_motors();
```

Prototype (goes before the main)

```
void stop_motors()
{
motor(0,0);
motor(3,0);
}
```

Definition (goes after the main)

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Code without writing your own functions

```c
// Created on Fri August 30 2013
void drive_forward();
int main()
{
    printf("lets drive in a square\n"); //print to screen
    msleep(2000); // pause for 2 seconds so you can read the screen
    drive_forward();
    motor (0, 0);
    motor (3, 0);//stop motors
    motor (0, 100);
    motor (3, 20);
    msleep(2000); // turn to the right for 2 seconds
    motor (0, 0);
    motor (3, 0);//stop motors
    drive_forward();
    motor (0, 0);
    motor (3, 0);//stop motors
    motor (0, 100);
    motor (3, 20);
    msleep(2000); // turn to the right for 2 seconds
    motor (0, 0);
    motor (3, 0);//stop motors
    drive_forward();
    motor (0, 0);
    motor (3, 0);//stop motors
    motor (0, 100);
    motor (3, 20);
    msleep(2000); // turn to the right for 2 seconds
    motor (0, 0);
    motor (3, 0);//stop motors
    drive_forward();
    motor (0, 0);
    motor (3, 0);//stop motors
    ao();
    return 0;
}
    void drive_forward()
    {
    motor (0, 100);
    motor (3, 100);
    msleep(4000);
    }
```

Main is shorter and easier to read

Code with writing your own functions

```c
// Created on Fri August 30 2013
void drive_forward();
void turn_right();
void stop_motors();
int main()
{
    printf("lets drive in a square\n"); //print to screen
    msleep(2000); // pause for 2 seconds so you can read the screen
    drive_forward();
    stop_motors(); //stop motors
    turn_right(); // turn to the right
    stop_motors(); //stop motors
    drive_forward();
    stop_motors();//stop motors
    turn_right(); // turn to the right
    stop_motors();//stop motors
    drive_forward();
    stop_motors();//stop motors
    turn_right(); // turn to the right
    stop_motors(); //stop motors
    drive_forward();
    stop_motors(); //stop motors
    ao();
    return 0;
}
    void drive_forward()
    {
    motor (0, 100);
    motor (3, 100);
    msleep(4000);
    }
    void turn_right()
    {
    motor (0, 100);
    motor (3, 20);
    msleep(2000);
    }
    void stop_motors()
    {
    motor (0, 0);
    motor (3, 0);
    }
```

```
// Created on Fri August 30 2013
void drive_forward();
void turn_right();
void stop_motors();
int main()
{
   printf("lets drive in a square\n"); //print to screen
   msleep(2000); // pause for 2 seconds so you can read the screen
   drive_forward();
   stop_motors(); //stop motors
   turn_right(); // turn to the right
   stop_motors(); //stop motors
   drive_forward();
   stop_motors();//stop motors
   turn_right(); // turn to the right
   stop_motors();//stop motors
   drive_forward();
   stop_motors();//stop motors
   turn_right(); // turn to the right
   stop_motors(); //stop motors
   drive_forward();
   stop_motors(); //stop motors
   ao();
   return 0;
}
   void drive_forward()
   {
   motor (0, 100);
   motor (3, 100);
   msleep(4000);
   }
   void turn_right()
   {
   motor (0, 100);
   motor (3, 20);
   msleep(2000);
   }
   void stop_motors()
   {
   motor (0, 0);
   motor (3, 0);
   }
```

Advantages

1. It makes the main program easier to read, understand and spotting mistakes is much easier

1. It allows you to change a variable value one time in the function definition for the entire program
   - Let's say you wanted to draw a smaller square
     - Simply change the `msleep()` value in your `drive_forward()` function definition from 4000 to 2000 and the `msleep()` value in your `right_turn()` function definition to 1000.

Tip:

Go to the end of the program and write the definition first (remember no semicolon) and then go to the top and fill in the prototype

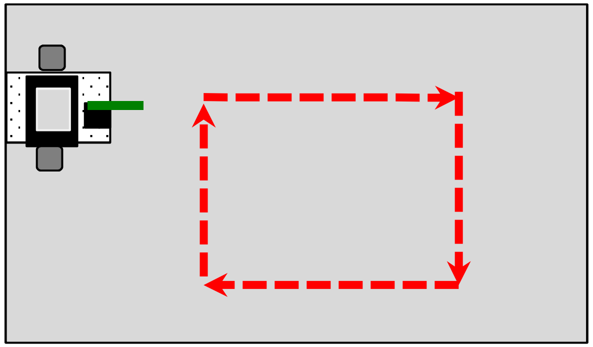
KISS INSTITUTE FOR PRACTICAL ROBOTICS

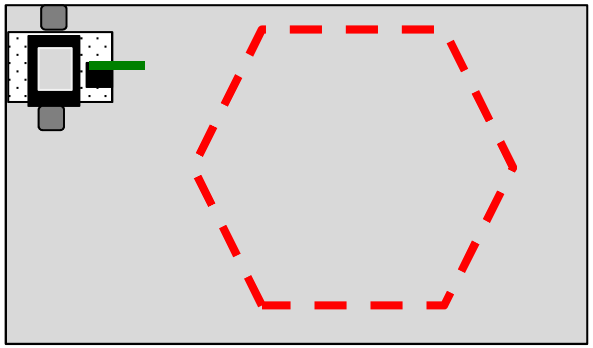
STUDENT ACTIVITY

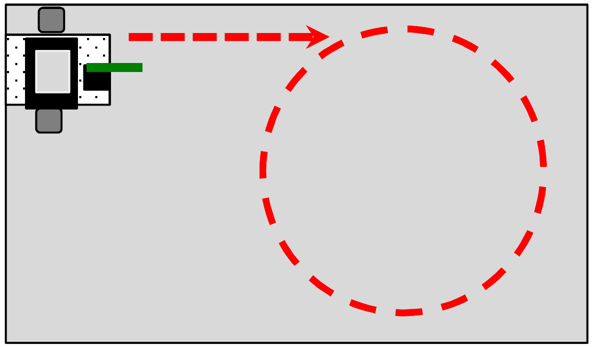
Get your Robot into Shape!

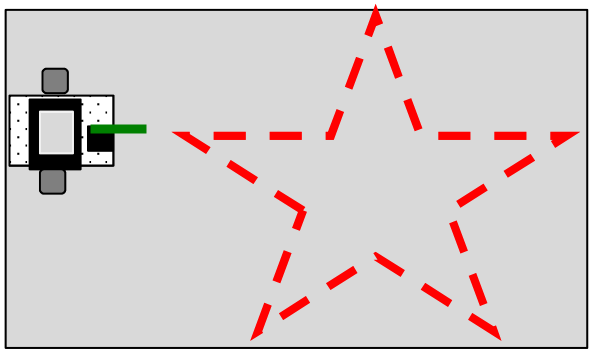
Have the robots draw geometric patterns

1. Have the robot complete a circle
2. Triangle, Star, Pentagon, etc.
3. Make sure you are writing your own functions for repeated actions in the code
4. Great activity for math/geometry extensions









KISS INSTITUTE FOR PRACTICAL ROBOTICS

Programming the robot to run for a set amount of time

Goals

- Learn how to use the `shut_down_in();` function to have the robot shut down after running for a set amount of seconds

*In Botball teams must automatically `shut_down_in(120);`

Preparation

- You will need the DemoBot built and ready to go
- You will need computers with the KISS IDE
- You will need the USB download cable
- You will need materials for “Touch the Can” and “Circle the Can” activities

Activity

Follow the slides to make the robot shut down in XXX seconds

<u>You can put a “Maximum” time limit to complete any of the previous activities</u>

STUDENT ACTIVITY

You have 3 seconds to complete your mission!

Use the provided robot or build your robot using the DemoBot building guide.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Connect the Link to your computer

- Using the USB cable connect to the Link (micro usb port)
- Turn the Link on with the black switch on the side

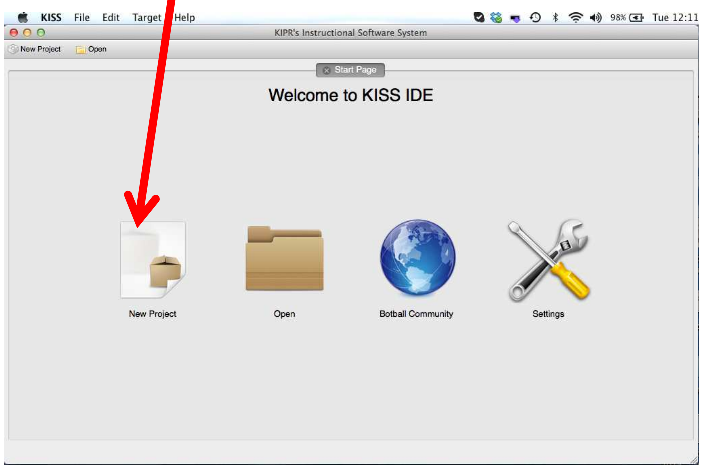
KISS INSTITUTE FOR PRACTICAL ROBOTICS

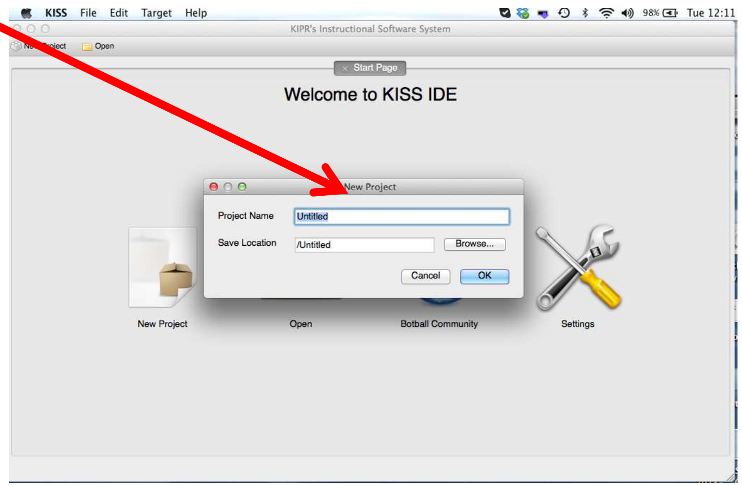
STUDENT ACTIVITY

Launch the KISS IDE

If already running just select new project

- Start the KISS IDE by clicking on its icon to get the welcome screen
- Click on the *"New Project"* icon
- You will have to name and save your project





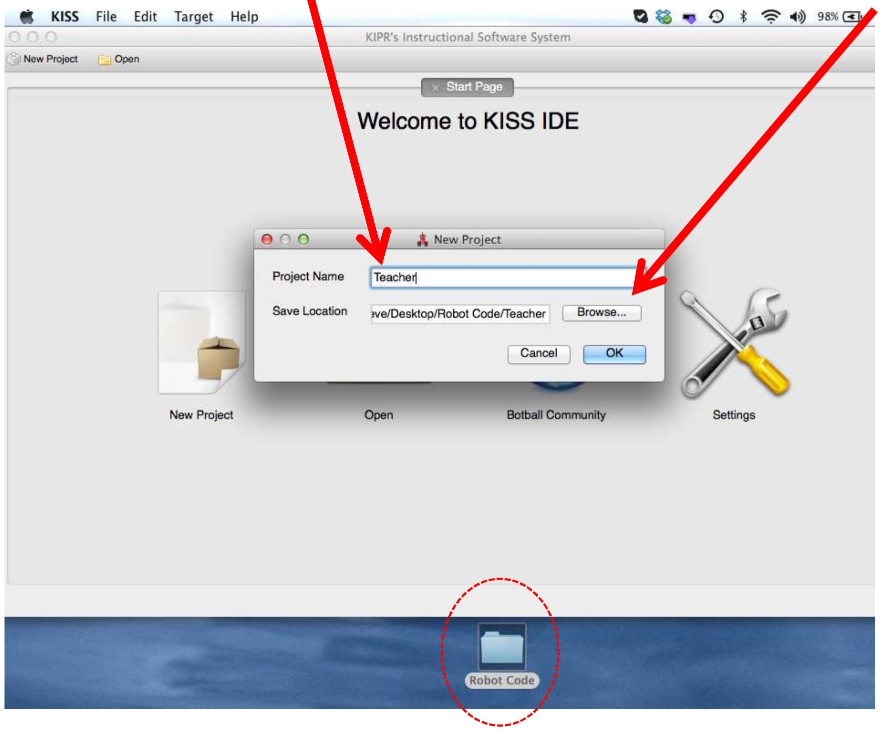
KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Writing a Program

- Create a Robot Code folder on your desktop (you will use this for all of your code)
- Name your new project “something descriptive”

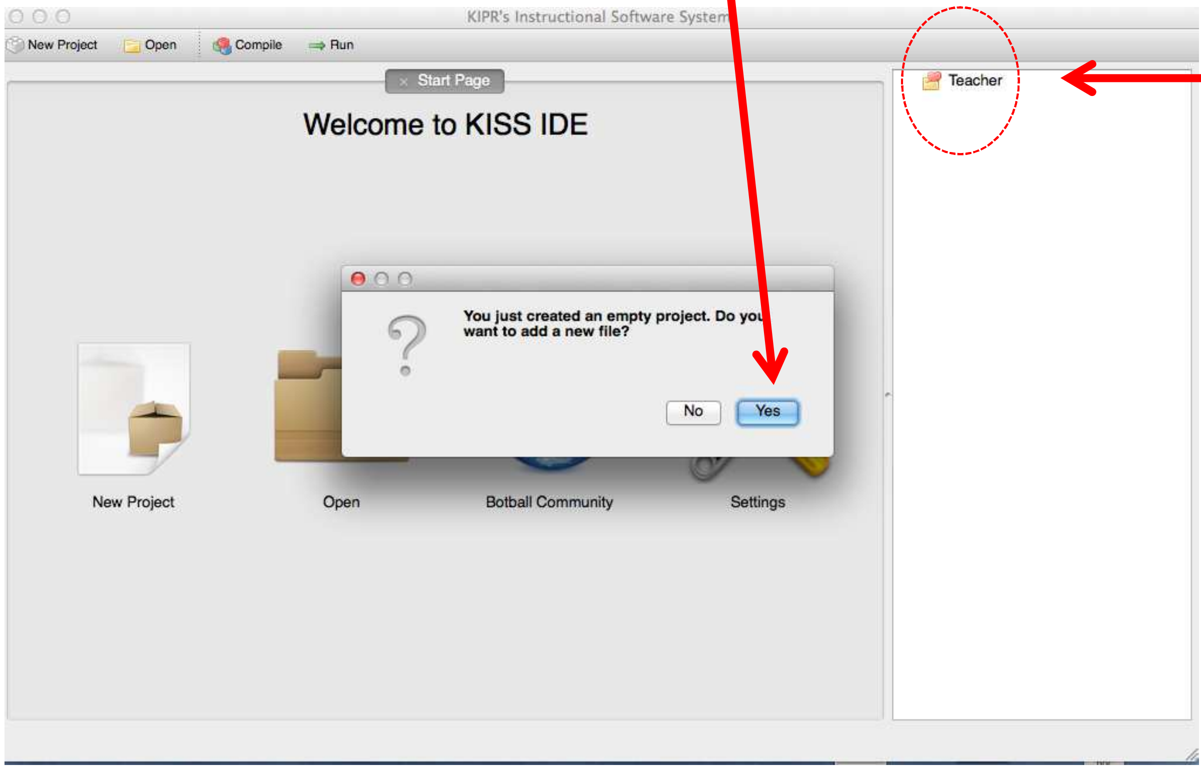
Use the Browse button to save the project into your Robot Code folder on your desktop



KISS INSTITUTE FOR PRACTICAL ROBOTICS

Writing Your First Program

- You will be asked if you want to add a new file



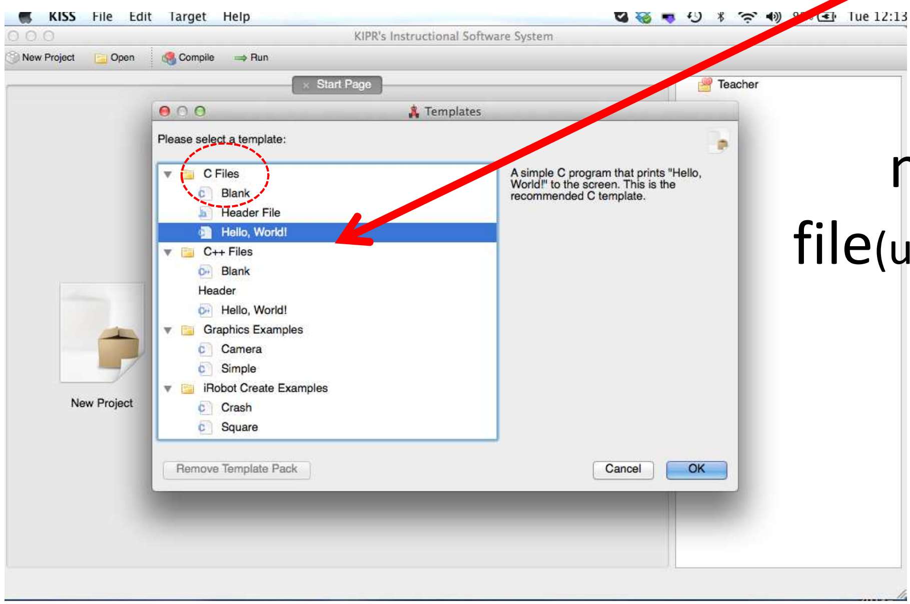
Notice your project "teacher" now appears on the side bar

STUDENT ACTIVITY

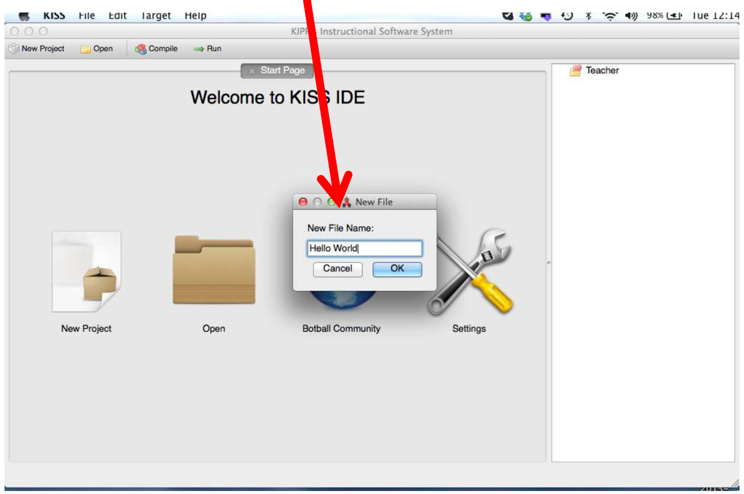
Writing Your First Program

You now need to select a template

Select under the C files folder, *Hello, World!*



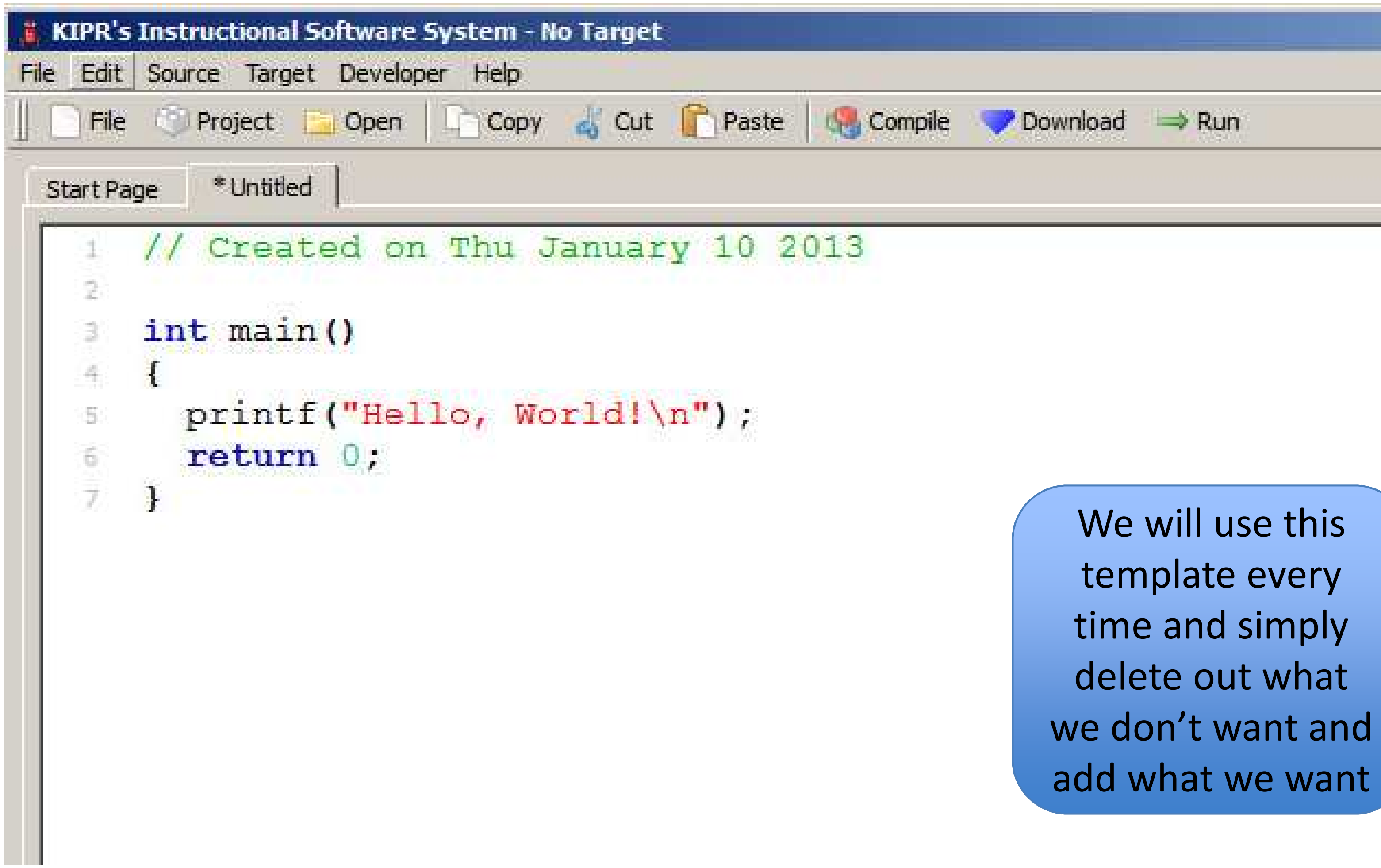
You will be prompted to name and save the new file(use "something descriptive")



KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

The C Template: Hello, World!



We will use this template every time and simply delete out what we don’t want and add what we want

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Programming your robot to run for a set amount of time

The `shut_down_in();` function will end the program after the number of seconds you put into the argument.

```
shut_down_in(3.0);   //3 seconds
shut_down_in(120.0);  //120 seconds
```

Uses

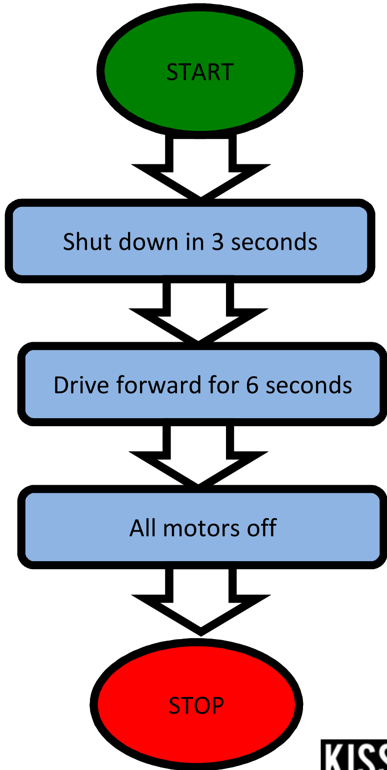
- Botball robots must shut down automatically at the end of the round

You have 3 seconds to complete your activity

Write a program for your robot that has it drive forward for 6 seconds and shut down in 3 seconds using the `shut_down_in(3.0);` function

Psuedocode (Task Analysis)

```
// 1. Shut down in 3 seconds
// 2. Drive Forward for 6 seconds
// 3. Shut off all motors
```



Programming your robot to run for a set amount of time example

```
// Created on Thu September 5 2013
void drive_forward();
int main()
{
   shut_down_in (3.0);
   drive_forward(); // Drive forward
   msleep(6000); // Allow 6 seconds to move forward
   ao(); // Turn everything off
   return 0;
}
void drive_forward()
{
   motor(0,100);
   motor(3,100);
}
```

Notice the function prototype for drive-forward

Program will end in 3 seconds

Even though this says to run for 6 seconds it will be shut down in 3 seconds

Notice the function definition for drive-forward

Programming your robot to run for a set amount of time activities

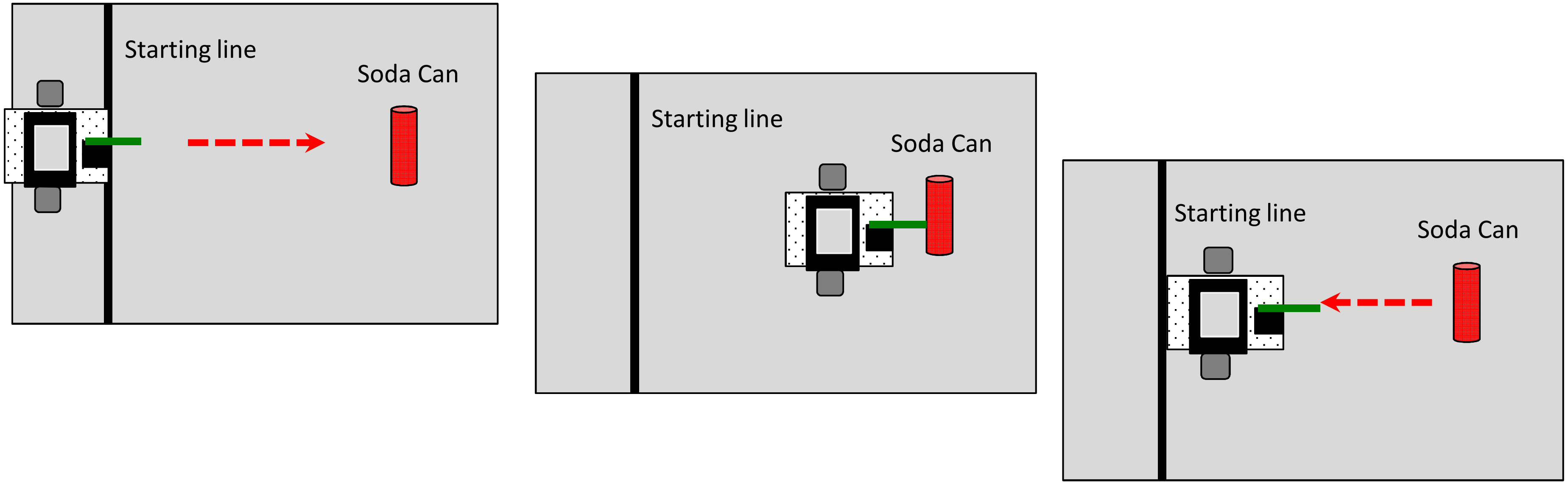
Now complete the following activities again, but this time make the time limit 60 seconds to complete the task

STUDENT ACTIVITY

TEACHER'S RESOURCE

Closest to/Touch the Can and Go Home

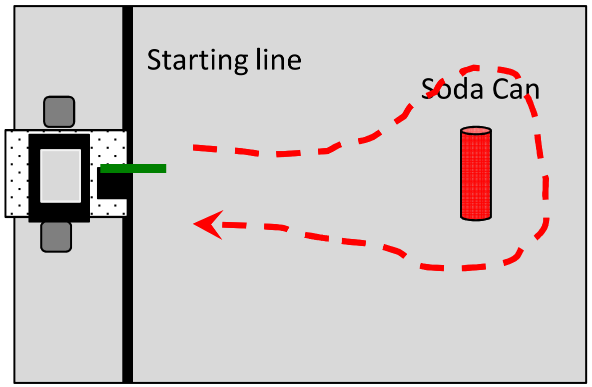
1. A variation on Touch the Can and Closest to the Can.
2. After stopping closest/touching, back up until touching the starting line
3. Using the `shut_down_in();` give this a 30 second time limit



KISS INSTITUTE FOR PRACTICAL ROBOTICS

Circle the Can and Go Home

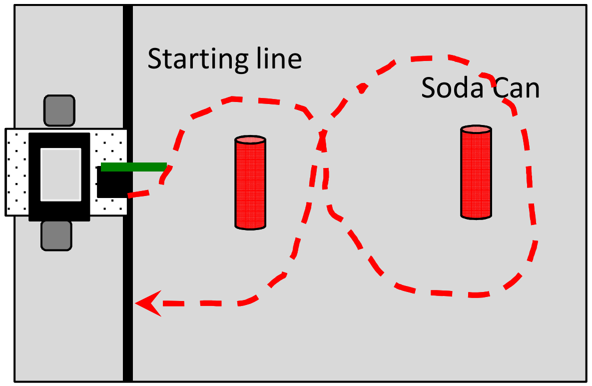
1. Brings in the concept of turning
   - If you touch the can you must start over
   - The quickest trip is the winner
   - Move the can to various distances
   - Make them go clockwise and then counter clockwise
2. Using the `shut_down_in();` give this a 30 second time limit

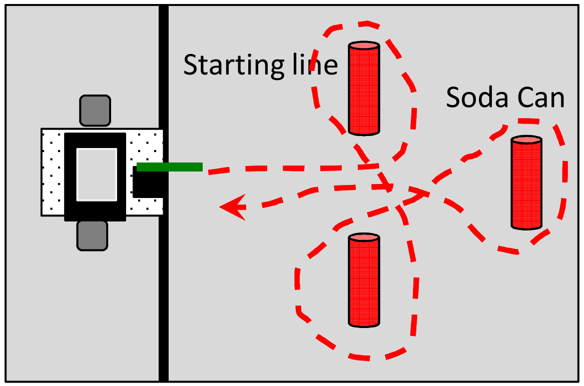


Circle the Can(s) and Go Home

Variation on Circle the Can

1. Have them make a figure 8 around two objects
2. Barrel Race (have them go around three cans)
3. Using the `shut_down_in();` give this a 90 second time limit





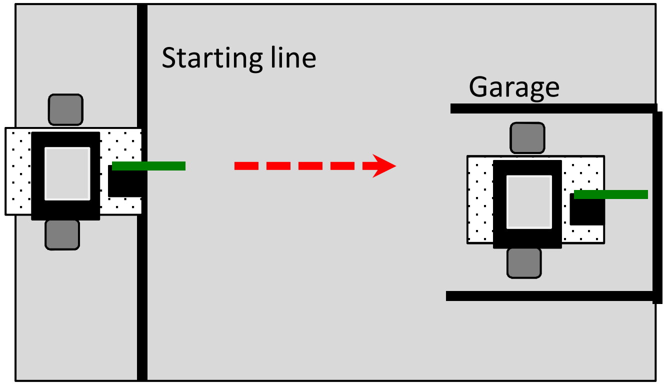
STUDENT ACTIVITY

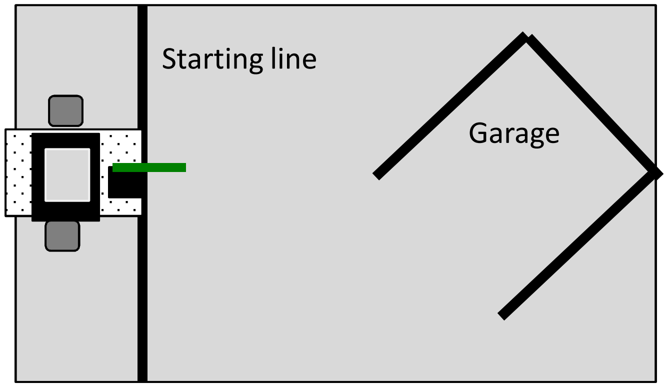
TEACHER'S RESOURCE

Park in the Garage

Robots must start on or behind the starting mark and park in the garage (box or tape outline on board), If they touch the garage they must start over at the starting line

1. Start with the garage straight across from the starting line
   - Garage can be roomy and then make it a tight fit
2. Move the garage to various distances and locations
3. Using the `shut_down_in();` give this a 60 second time limit





KISS INSTITUTE FOR PRACTICAL ROBOTICS

Programming the robot to start automatically when it senses a light

TEACHER'S RESOURCE

Goals

- Learn how to use the `wait_for_light();` function to have the robot sense a light and start
- Students will start working with and becoming familiar with using sensors
- Student will learn how to access and use the sensor list and sensor graph features on the Link

***Autonomous robots need to start automatically when they sense a light**

Preparation

- You will need the DemoBot built and ready to go
- You will need computers with the KISS IDE
- You will need the USB download cable
- You will need a light sensor and something to attach it to the robot (uglu, tape etc)
- The light sensor
- A flashlight *THE SENSOR IS AN INFRARED SENSOR SO MOST LED LIGHTS WILL NOT WORK (YOU NEED AN INCANDESCENT)

Activity

Follow the slides to make the robot start automatically when it senses a light

You can add the `wait_for_light();` to complete any of the previous activities

KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Start your programs with a light

The `wait_for_light();` function allows your program to run when your robot senses a light

- It has a built in calibration routine that will come up on the screen
  (routine is on following slides)

Tip: *The light sensor senses infrared so it must be an incandescent light and not an LED light*

  - You need a flashlight

The more light (infrared) sensed the lower the reported value

Uses

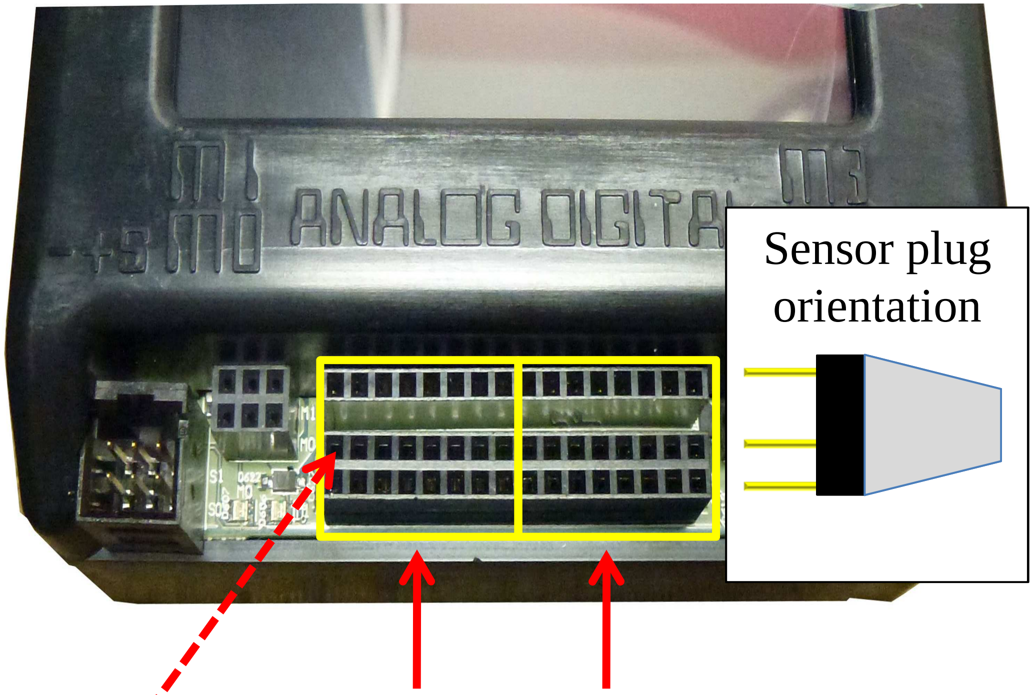
The light sensor is used to start Botball robots at the beginning of the game and it is a cool way to "automatically" start your robot

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Plug in Your Light Sensor

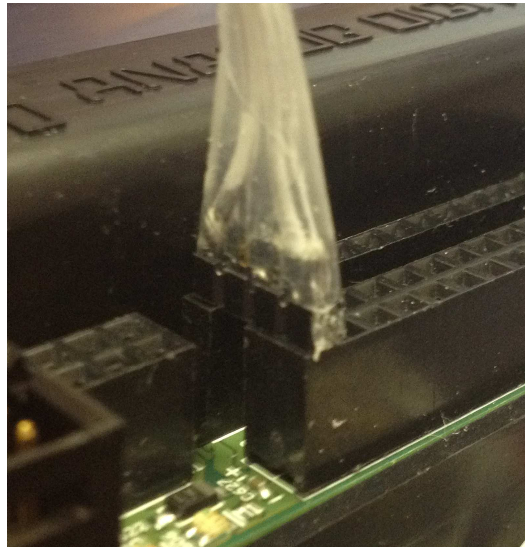
and get your flashlight!

STUDENT ACTIVITY



analog ports (0-7) and digital ports (8-15)

Plug your light sensor into analog port 0



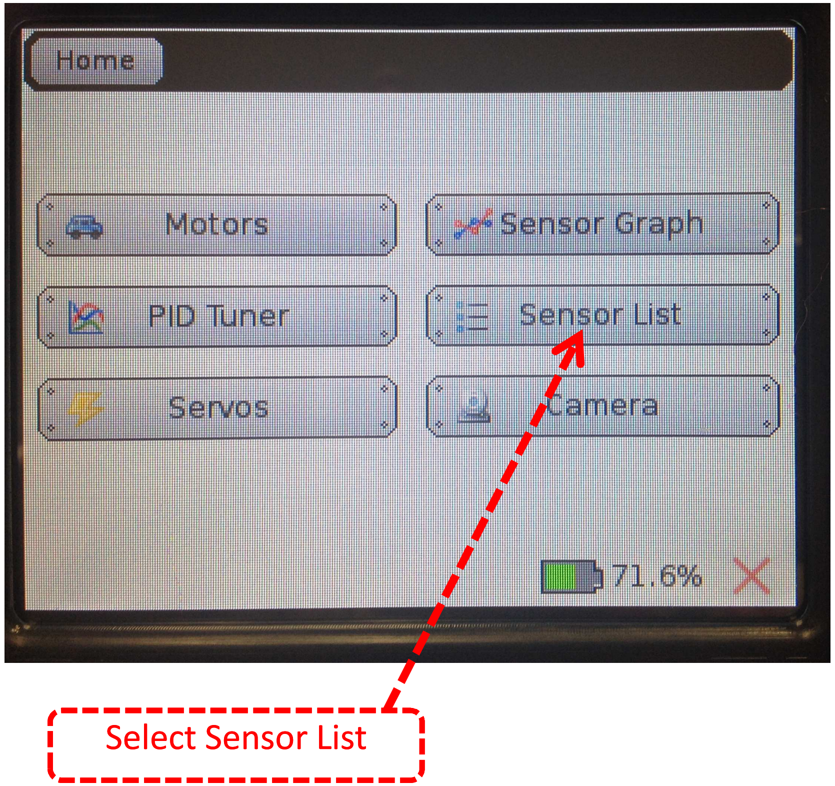
KISS INSTITUTE FOR PRACTICAL ROBOTICS

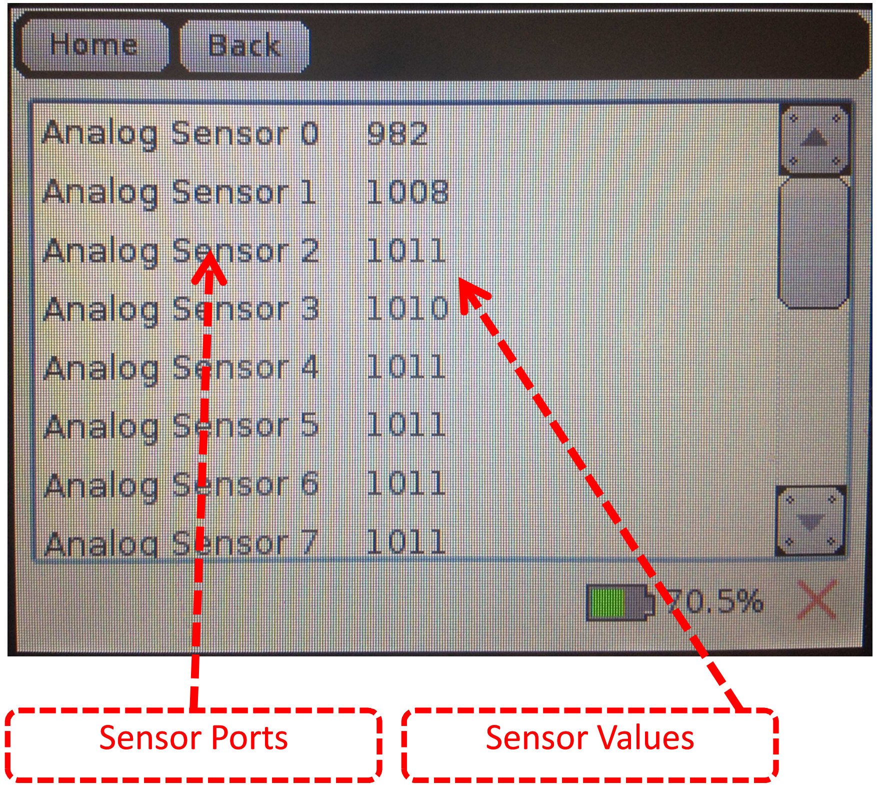
STUDENT ACTIVITY

Reading Sensor Values From the Sensor List

You can access the sensor values from the sensor list on your Link

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





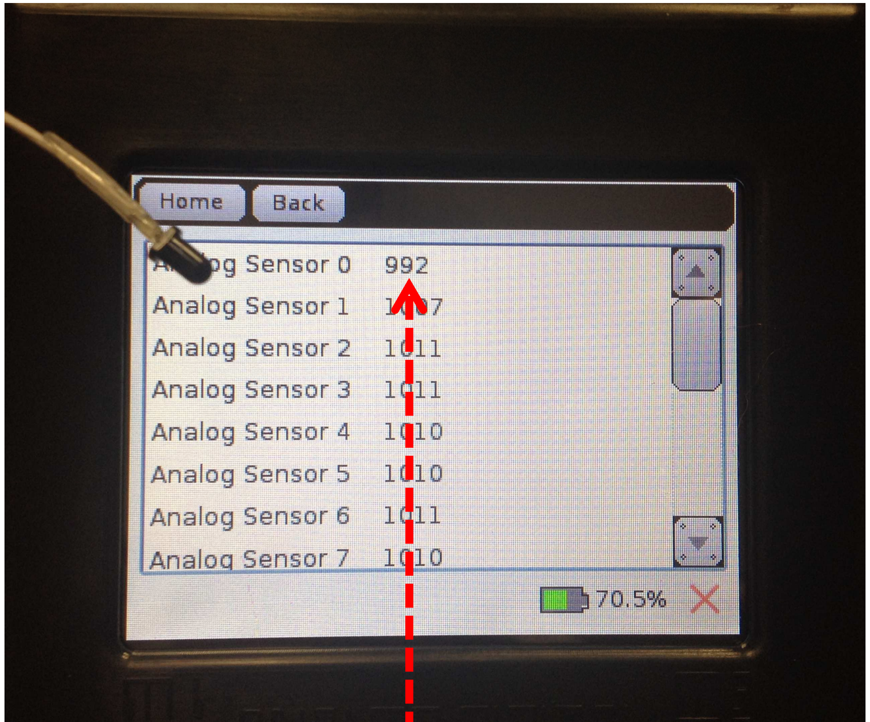
KISS INSTITUTE FOR PRACTICAL ROBOTICS

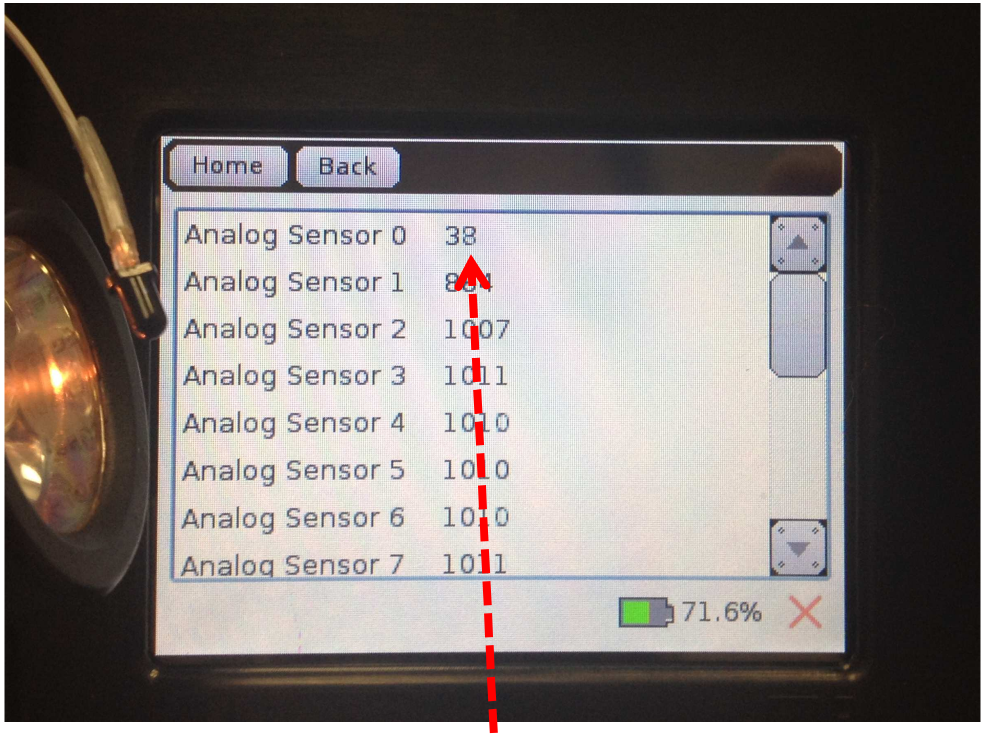
STUDENT ACTIVITY

Reading Sensor Values From the Sensor List

With the light sensor plugged into analog port #0

- With no light sensed the value is (992)
- When the flashlight is on and IR is sensed the value is much lower (38)





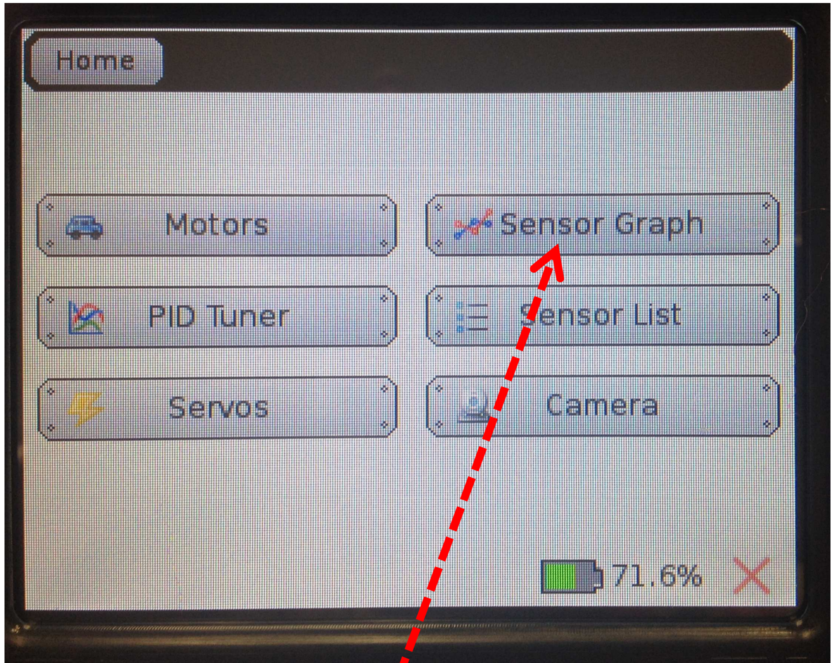
Value of 992 (not sensing light)

Value of 38 (sensing light)

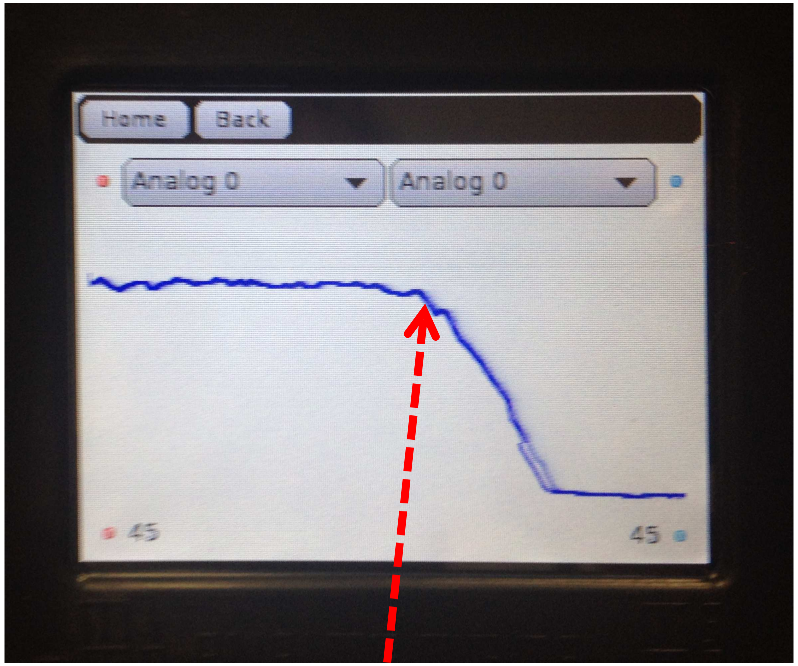
KISS INSTITUTE FOR PRACTICAL ROBOTICS

Watching Sensor values on the Sensor Graph

You can also have a real-time graph of all of the sensor ports. Select the Sensor Graph and then select the sensor port # (in this case, 0)



Select Sensor List

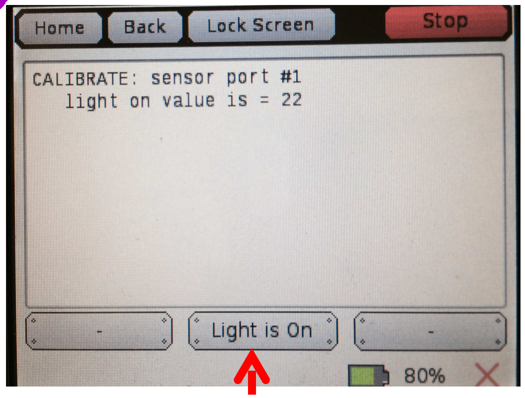


This is graphing Analog port #0 (light sensor) you can easily see when the light was turned on as the value rapidly decreases

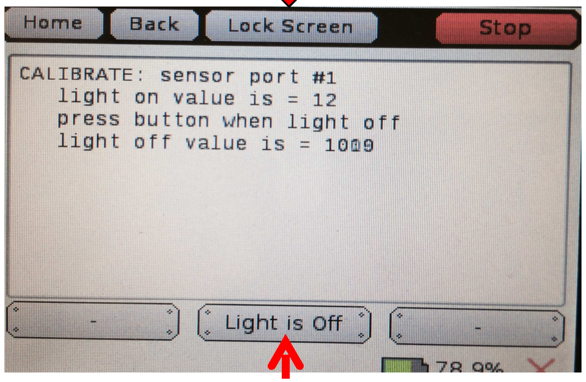
STUDENT ACTIVITY

The light calibration routine

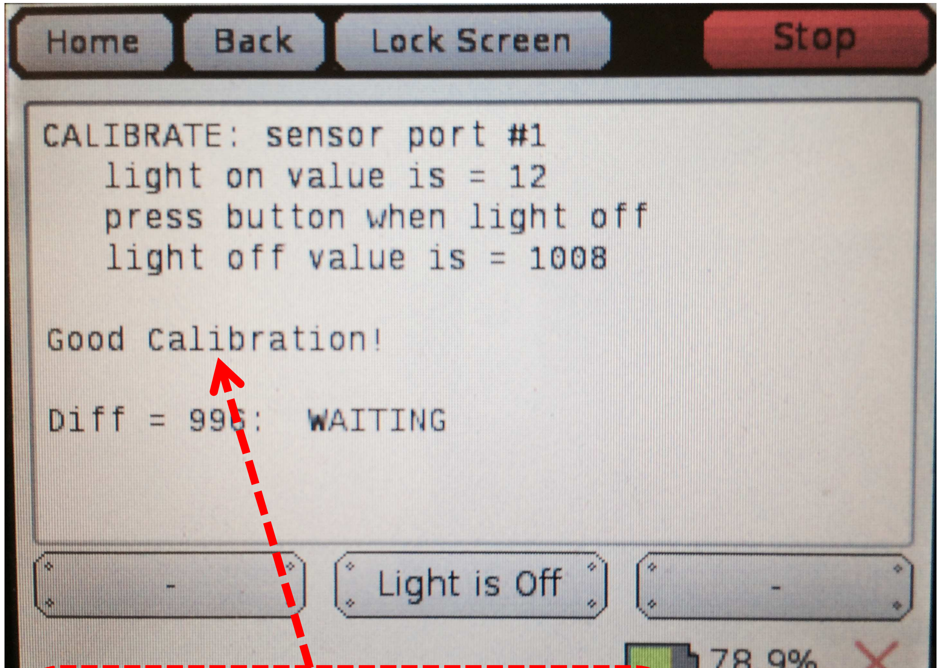
When you use the function in your code the calibration routine will start automatically



When the light is on (low value) select Light is On button



When the light is off (high value) select Light is Off button



You will get a Good Calibration! Message when done correctly IF NOT you will get a BAD CALIBRATION message (you need to run through the routine again)

KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

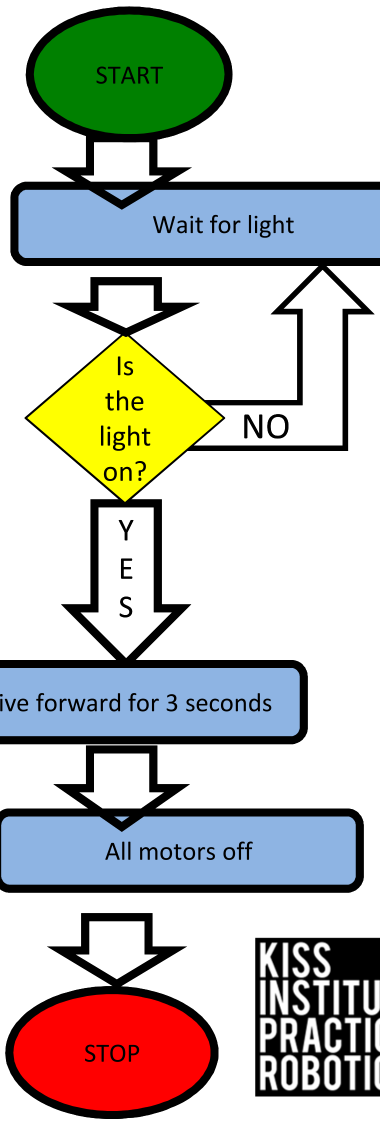
I See the Light!

1) Write a program that uses a light sensor to start your robot
   - You should have a light sensor plugged into analog sensor port #0

2) Have it run forward for 3 seconds and then stop

Psuedocode

```
// Check value of light sensor in analog_port 0

// Drive forward when sensor sees light

// Allow 3 seconds to move forward

// Turn everything off
```



KISS INSTITUTE FOR PRACTICAL ROBOTICS

I See the Light! Solution

```
// Created on Thu September 5 2013
void drive_forward();
int main()
{
   wait_for_light(0); // Check value of light sensor in analog port 0
   drive_forward(); // Drive forward when sensor sees light
   msleep(3000); // Allow 3 seconds to move forward
   ao(); // Turn everything off
   return 0;
}
void drive_forward()
{
   motor(0,100);
   motor(3,100);
}
```

Notice the function prototype for drive-forward

Notice the function definition for drive-forward

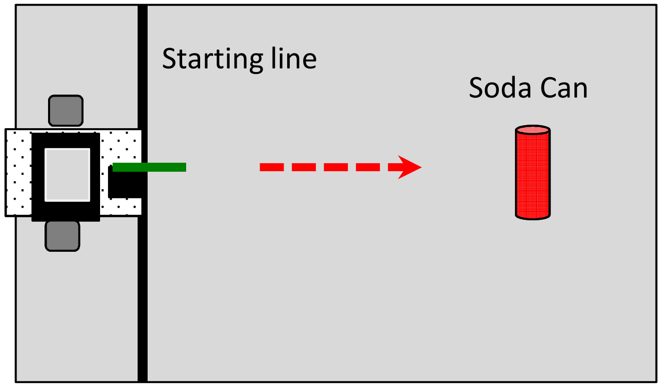
STUDENT ACTIVITY

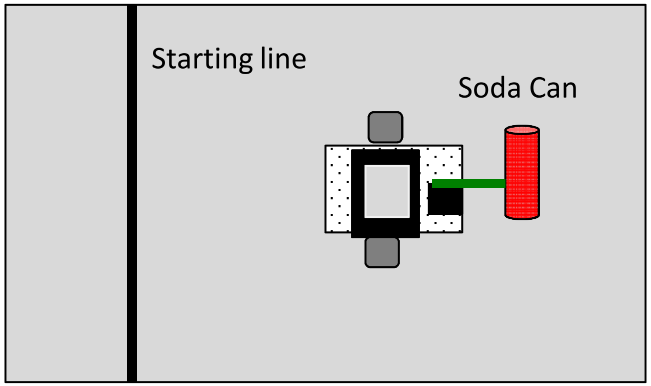
I See the Light!

1) Add the `wait_for_light();` function to the start of any of the previous challenges and activities
   - You cannot touch the robot to start it, it MUST start on its own after sensing the light

Touch the Can

1. Robots must start on or behind the starting mark and move to the object with the goal of touching the object in the shortest amount of time
2. The robot must be started with a flashlight





KISS INSTITUTE FOR PRACTICAL ROBOTICS

Using Servo motors

TEACHER'S RESOURCE

Goals

- To distinguish between motors and servo motors
- To help students understand how to use Servo Motors with their robots
    - Enable, disable, set position and get position functions

Preparation

- Have KISS IDE up and running
- Have a robot ready to go
- Have a servo motor

Activity

Follow slides

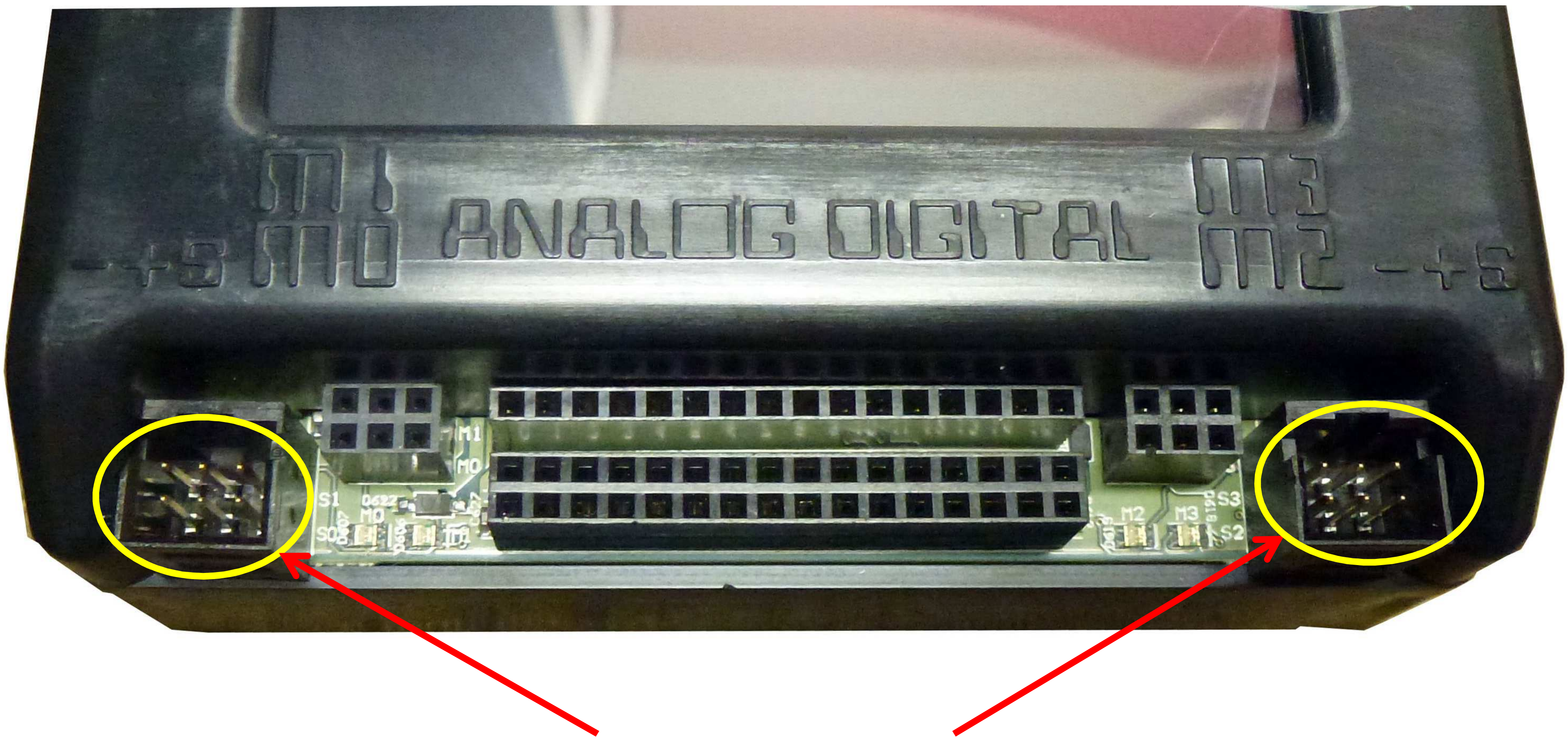
KISS INSTITUTE FOR PRACTICAL ROBOTICS

Servo Motors (Servos)

- A servo is a motor that rotates to a specified position between 0° and 180°
- Servos are great for raising an arm or closing a claw to grab something
- The motors and servos look similar except that a servo has 3 wires (usually colored orange, red, brown) and a plastic plug on the end

KISS INSTITUTE FOR PRACTICAL ROBOTICS

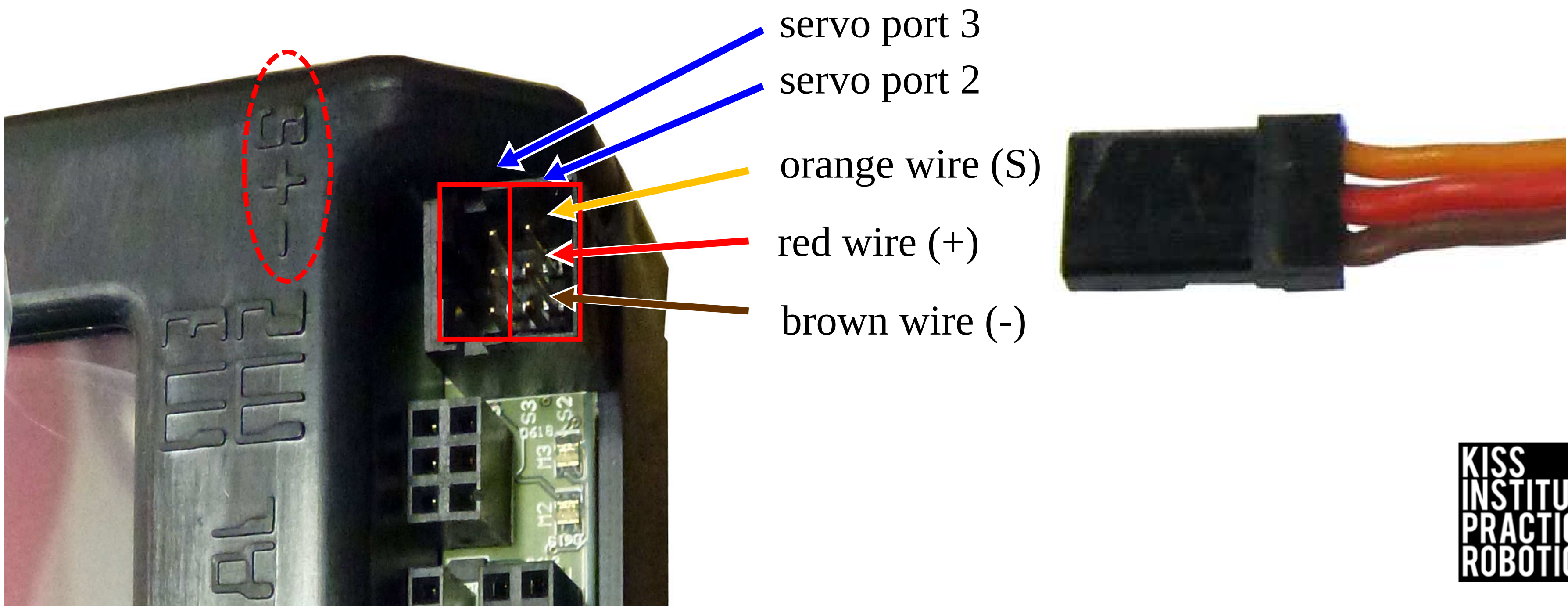
Servo Motor Ports



servo ports 0 and 1; servo ports 2 and 3

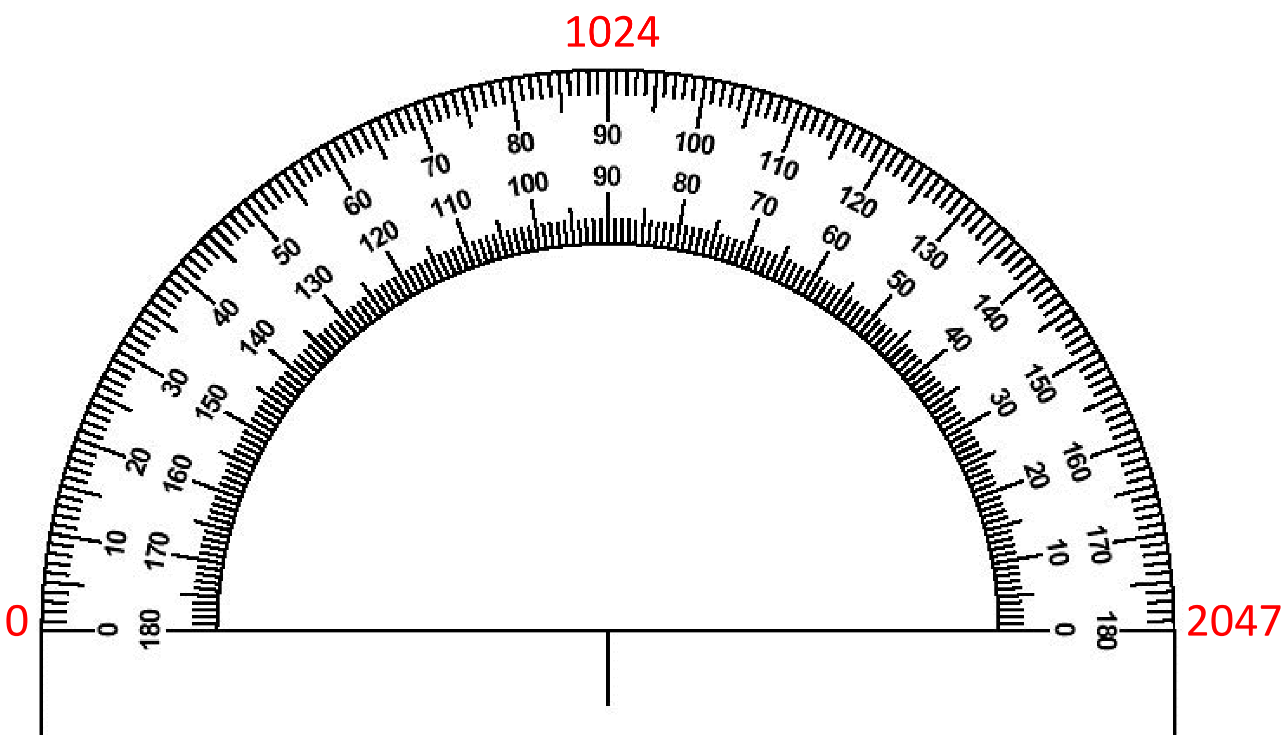
Servo Motors (Servos)

- Notice the case of the link is marked:
  - - for the brown wire
  - + for the red wire (it is in the center)
  - S for the signal wire (regulates the servo position)



Servo Motors (Servos)

- If you think of a servo like a protractor
  - The 180° is divided into 2048 positions (0-2047). Remember we start counting with 0 and not 1
  - This allows for greater precision when setting a position (you have 2048 different settings you can choose)
- The default position is 1024 (centered)



Servo Motors (Servos)

To help save power, servo ports by default are not active until enabled

Functions are provided in the KIPR Link library for enabling (or disabling) all servo ports and for sending them to a position

`enable_servos();` activates all servo ports

`disable_servos();` de-activates all servo ports

`set_servo_position(2,925);` rotates servo 2 to position 925

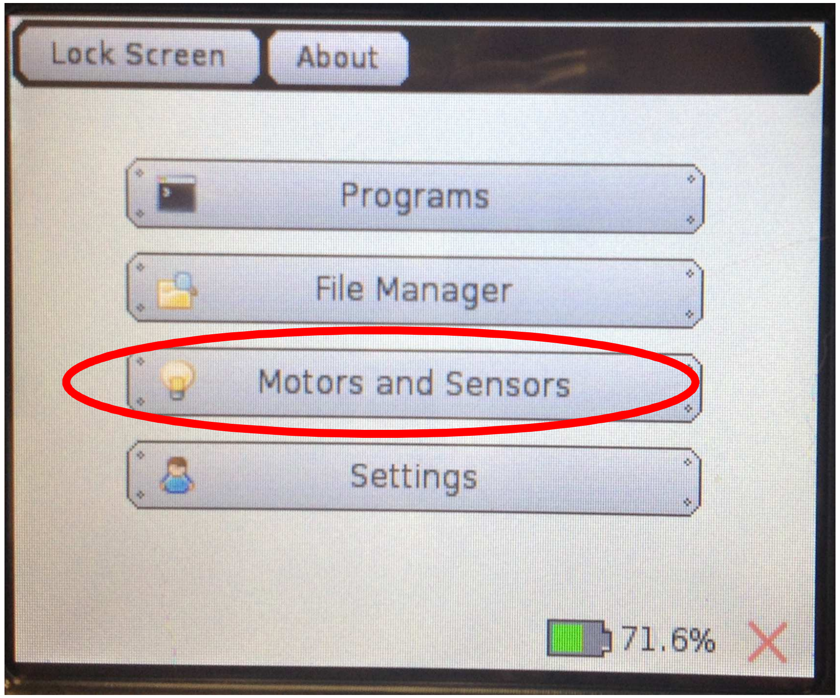
- Remember the range is 0-2047
- Default position when servos are first enabled is 1024**, BUT** You can preset a servo's position before enabling servos so it will immediately move to the position you want when you enable servos

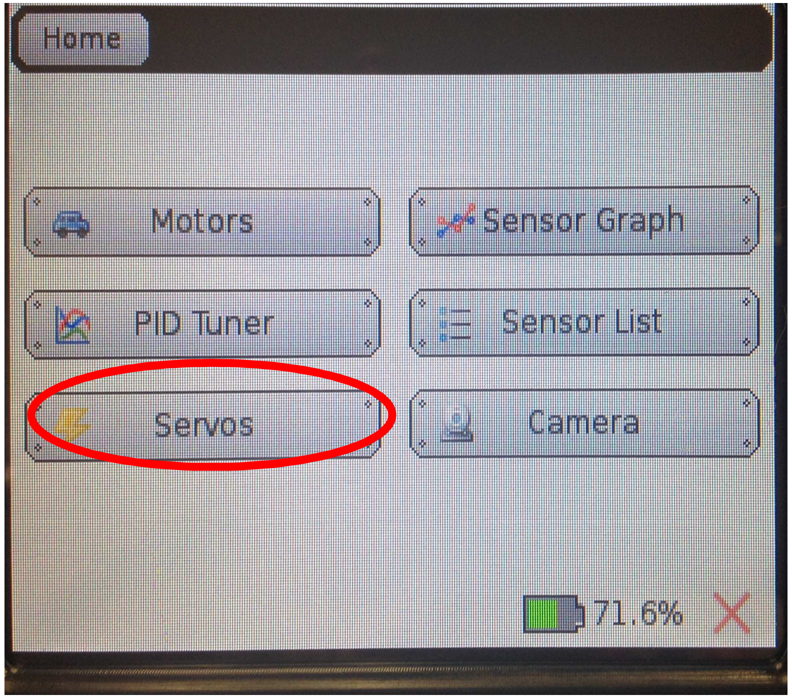
KISS INSTITUTE FOR PRACTICAL ROBOTICS

Servo Activity

STUDENT ACTIVITY

1. Make sure your Link is turned on
2. Plug a servo motor into Servo Port 0
3. Follow the guides to access the Servos Page on the Link



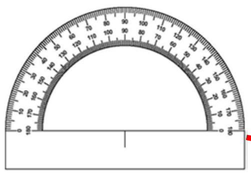


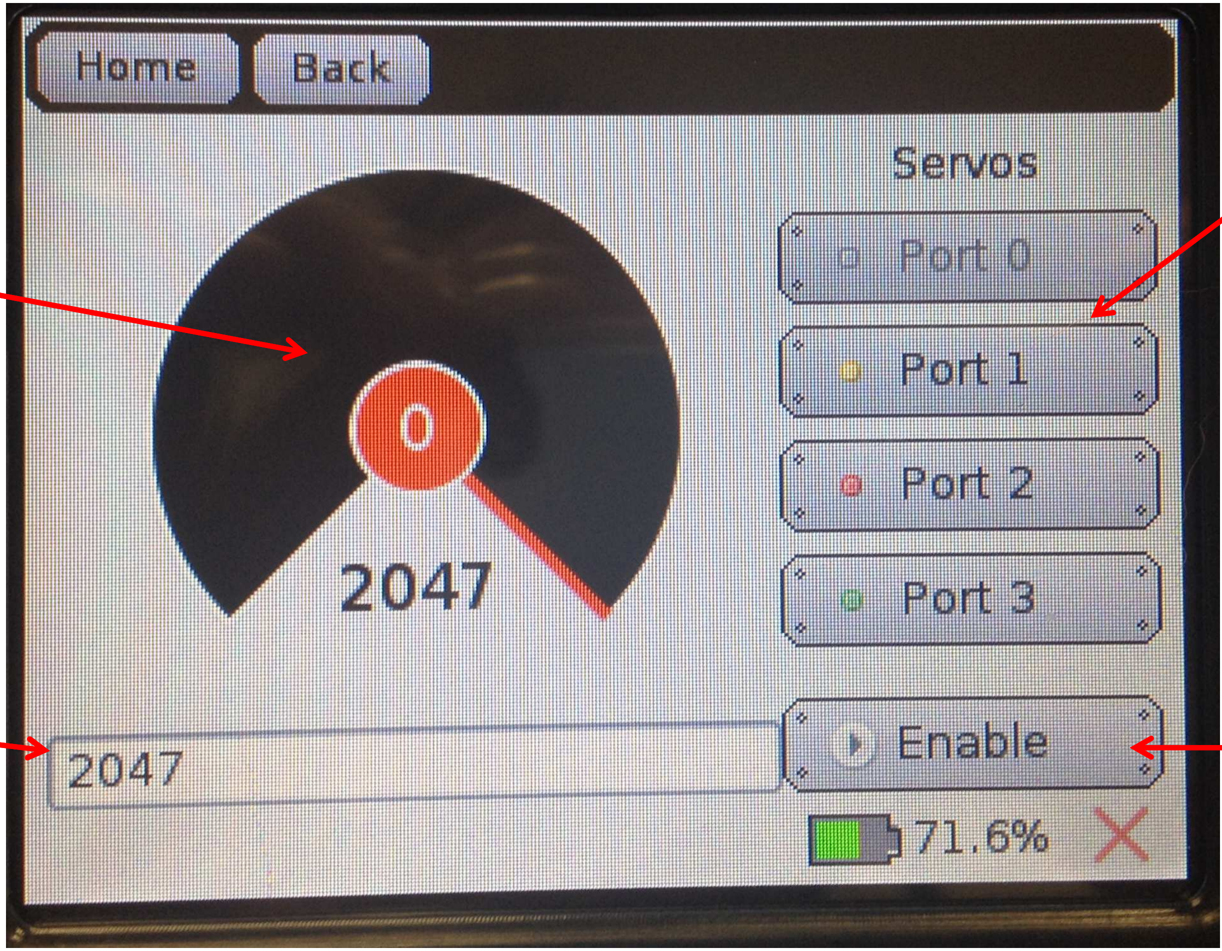
KISS INSTITUTE FOR PRACTICAL ROBOTICS

Servo Activity

STUDENT ACTIVITY

1. Use the Servo Page to test your Servo.





Select the Servo Port

This is the Servo Position

Enables the Servo

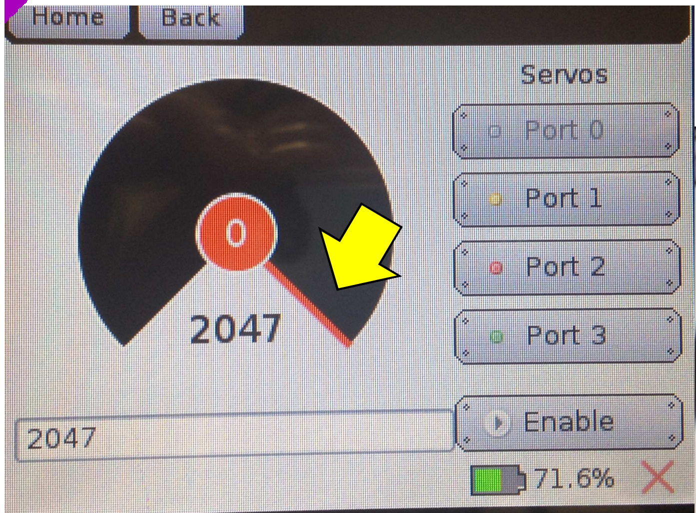
KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

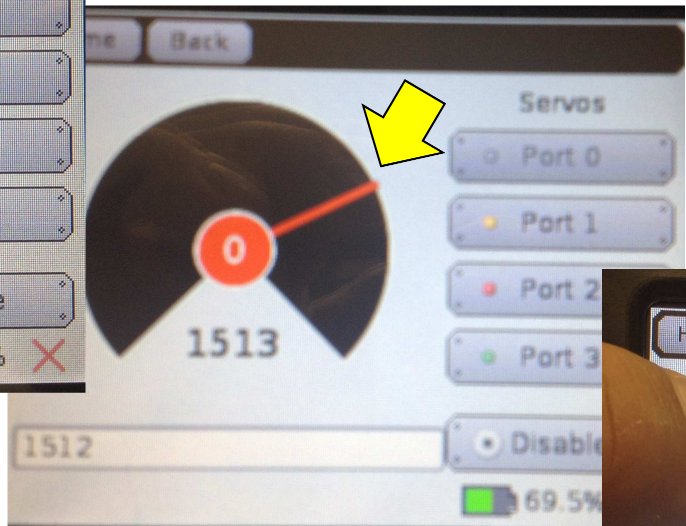
Servo Activity

1. Use the Servo Page to test your Servo.

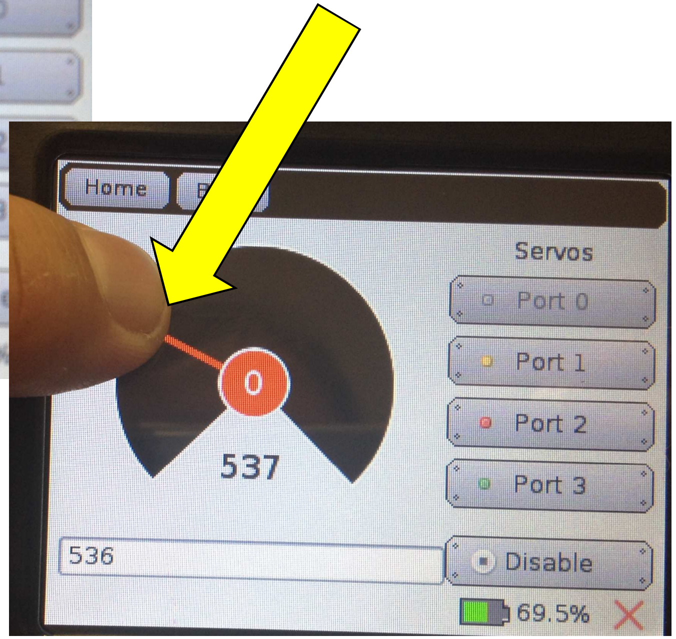
Use your finger to move the dial



Servo Maxed out @ 2047



Servo @ 1513



Servo @537

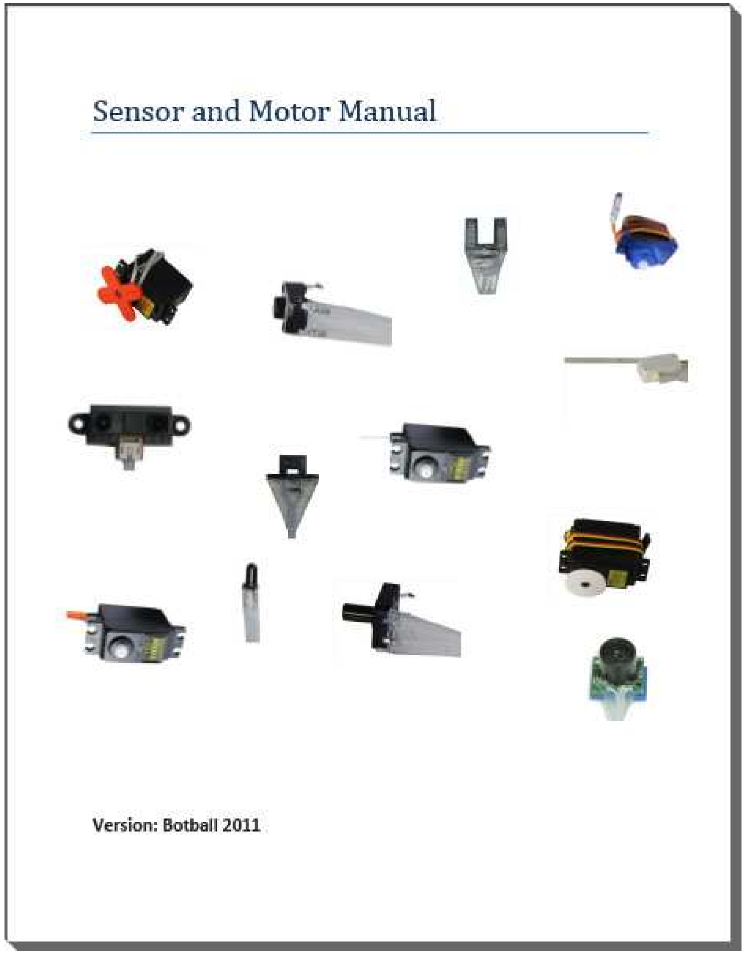
REMEMBER DO NOT KEEP PUSHING A SERVO BEYOND THE 0 OR 2047 POINTS- THIS CAN BURN SERVOS OUT

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Sensor and Motor Manual

TEACHER'S RESOURCE

- For further detail about servos, consult the Sensor and Motor Manual available via KISS IDE help or on your KIPR USB



KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

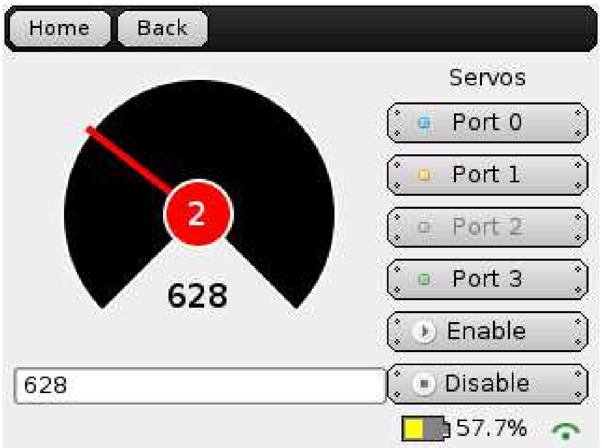
Servo Activity 1

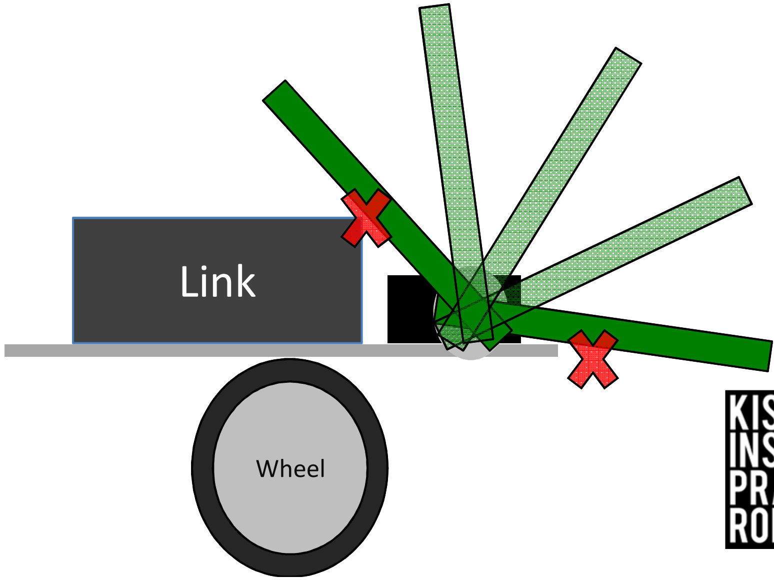
- Wave the pointer
- Using the servo and pointer on your demo-bot

WARNING

The servo mounted on your DemoBot is not free to move to all possible positions because it will run into the chassis and the controller

- DO NOT keep trying to move a servo to a position it cannot reach as this can burn out the servo as well as consume too much power
- Use the KIPR Link servo screen to determine the positions before hitting the chassis and the link and then use them in the code





KISS INSTITUTE FOR PRACTICAL ROBOTICS

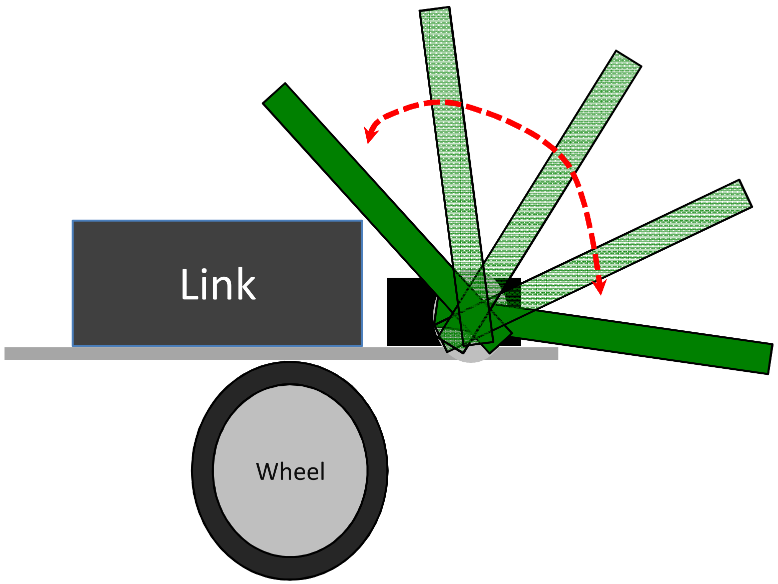
Servo Activity

`set_servo_position();`

Write a program for your robot to:

Psuedocode (Task Analysis)

```
1. //Enable servos
2. //Move servo 0 to 1400 OR # YOU DETERMINED FROM SERVO SCREEN
3. //Allow 1 second to complete moving to position
4. //Move servo 0 to 1024 OR OR # YOU DETERMINED FROM SERVO SCREEN
5. //Allow 1 second to complete moving to position
6. //Shut everything off
```



KISS INSTITUTE FOR PRACTICAL ROBOTICS

Servo Activity

set_servo_position();

TEACHER'S RESOURCE

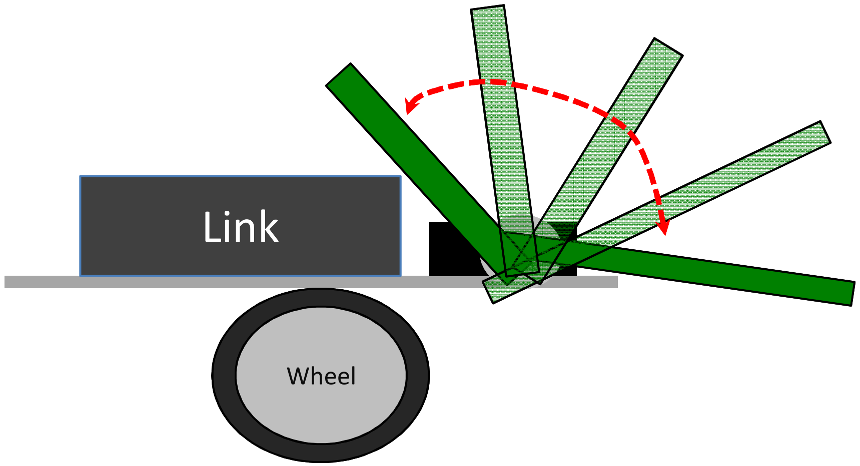
Solution

```
int main()
{
  enable_servos(); //enable servos
  set_servo_position (0,1400); //Move servo 0 to 1400 OR # YOU DETERMINED
  msleep (1000); //Allow 1 second to complete moving to position
  set_servo_position (0,1024); //Move servo 0 to 1024 OR OR # YOU 
DETERMINED
  msleep (1000); //Allow 1 second to complete moving to position
  ao (); //shut everything off
  return 0;
}
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Waving Robot

- Now that you can move the servo to any desired position make the robot wave continually
  - Write a function for the waving behavior and use it



Hokey Pokey (Dancing) Robot

STUDENT ACTIVITY

Have the robots "dance" by moving their servo and their motors to the Hokey Pokey

- Pick other songs and program the robot to dance
  - Make sure and play the music so they have to have some rhythm

You put your right hand in, //Move servo to pointing position
You put your right hand out, //Move servo to vertical position
You put your right hand in, //Move servo to pointing position
And you shake it all about, //Move robot back and forth rapidly

You do the hokey pokey,
and you turn yourself around, //Turn robot in a circle
That what it's all about.

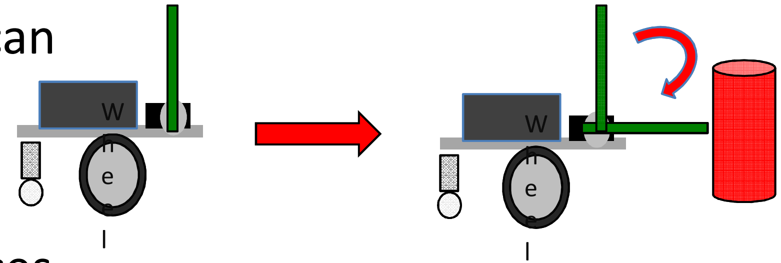
KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Touch the Can with Your Pointer

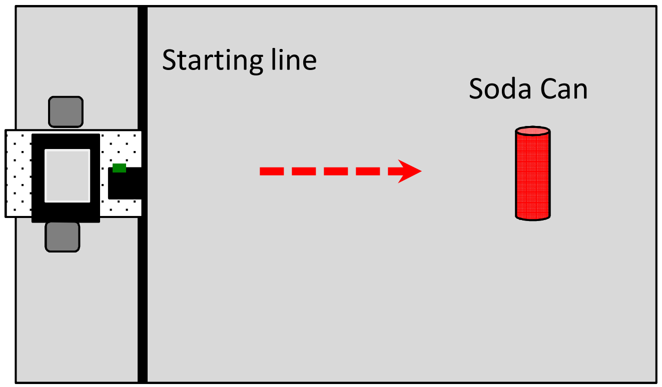
TEACHER'S RESOURCE

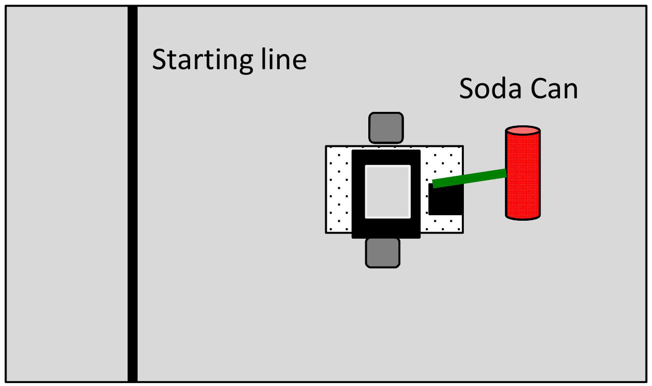
1. Robots must start on or behind the starting mark and move to the object with the goal of touching the object WITH the LEGO attached to the servo in the shortest amount of time
2. The pointer must start in the vertical position and then move to the position required to touch the can



Extensions

- Move the can to various distances
- Make the object smaller and harder to navigate to
- Math- have them measure the distance to the object and time the robot and then calculate rate/speed. Speed = Distance/Time.

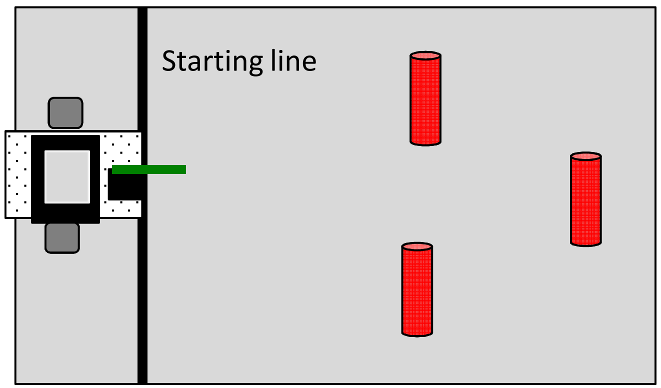


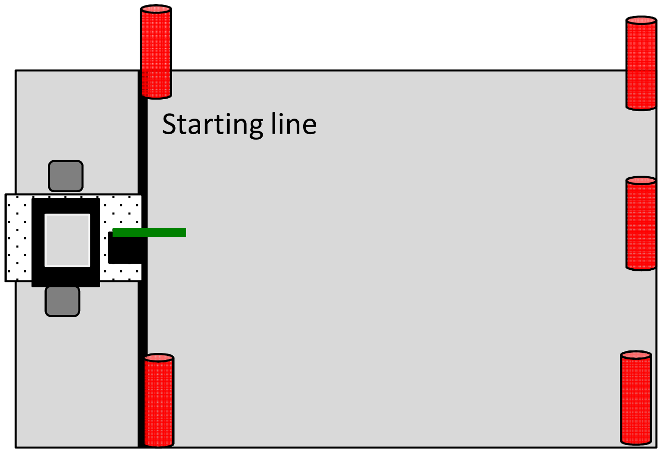


KISS INSTITUTE FOR PRACTICAL ROBOTICS

Tag, Your Out

- "Tag" with your servo pointer the objects that are then removed from the board
  - Must tag with the pointer only- if they touch it with any part of the robot other than the pointer it does not count
  - Pointer <u>has to change position</u> to tag (they can't drive around with the pointer out front all of the time)
- Score points for every item removed from the area
  - Use some tape or a marker to indicate where they should be set up
- <u>Place the items at known or set locations</u>
  - This is because they are still dead reckoning, once we learn more logic and decision making, we will use sensors to locate and find the objects, which can then be tagged and removed





KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

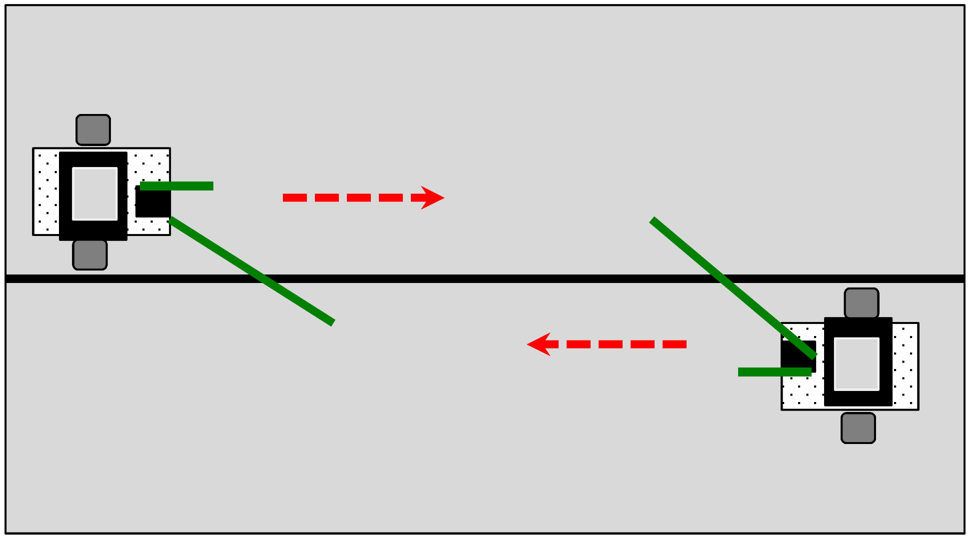
TEACHER'S RESOURCE

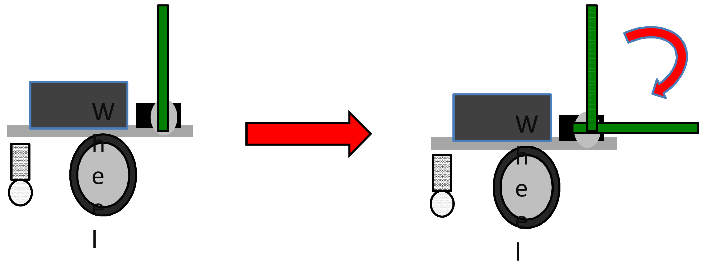
Variations on Walk the Line- Jousting

1. Robots on opposite sides of the line move towards each other and try to knock object off of other robot.
    - Use whatever object is handy

Engineering*

2. Have them use a servo motor to bring the lance from the upright starting position to the striking position before hitting the opponent





KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

TEACHER'S RESOURCE

Moving Objects with your Robot

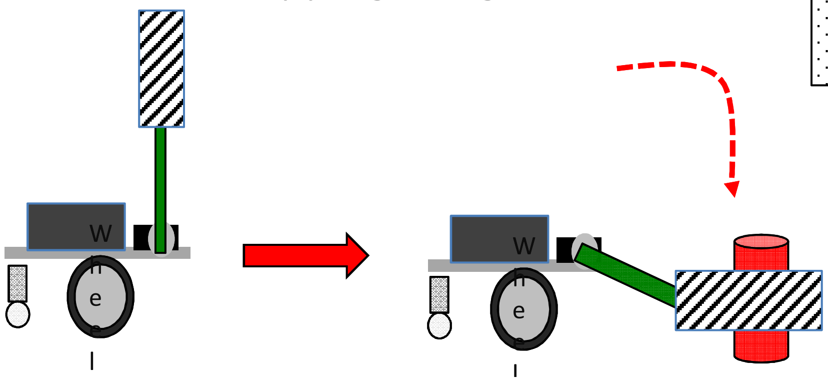
Now that you know how to move a servo you can design structures to collect items and move them around on the game board

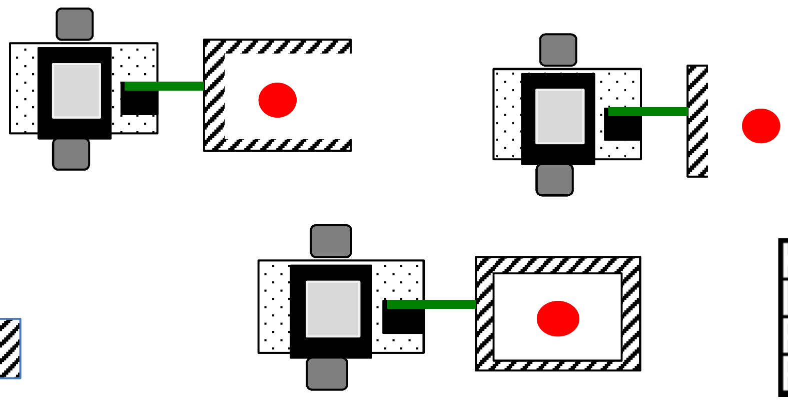
- Grabbing objects and dragging or lifting them to move them around on the game board is very useful in Botball
    - You can use containment structures
    - You can use claws/grippers

Engineering*

A structure can be built onto the servo on your Demo Bot that can be raised and lowered to push an object (bulldozer) or dropped over an object and then keep the object with the robot while it drags it somewhere else on the board (Bulldozers don't work well in reverse)

- You can build this out of LEGO or anything handy, foam board etc.
- REMEMBER your SERVO has a limit to how much weight it can lift without stripping the gears





KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Moving Objects with your Robot

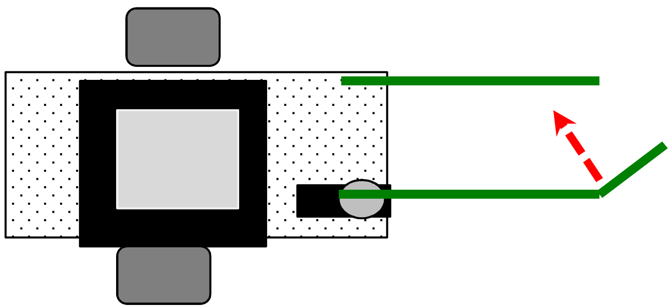
TEACHER'S RESOURCE

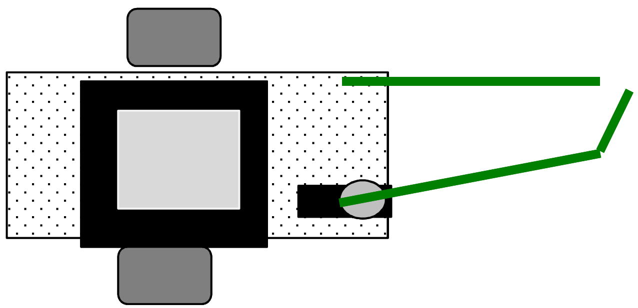
Claws/Grippers

Engineering*

A structure can be built onto the servo(arm) on your demo bot that can be closed and opened to grab an object

- You can build this out of LEGO and KMP
  - There are a lot of photos of claws and grabbers on YouTube, the Botball webpage and the Botball Educational Robotics Facebook page
- The easiest and first grabber to build has a static (unmovable) side and a side with a servo that closes
  - Write a function for opening and closing the servo
- You can use two servos, one to raise and lower the claw/gripper and one to open and close the claw/gripper





KISS INSTITUTE FOR PRACTICAL ROBOTICS

Recycle the Can

STUDENT ACTIVITY

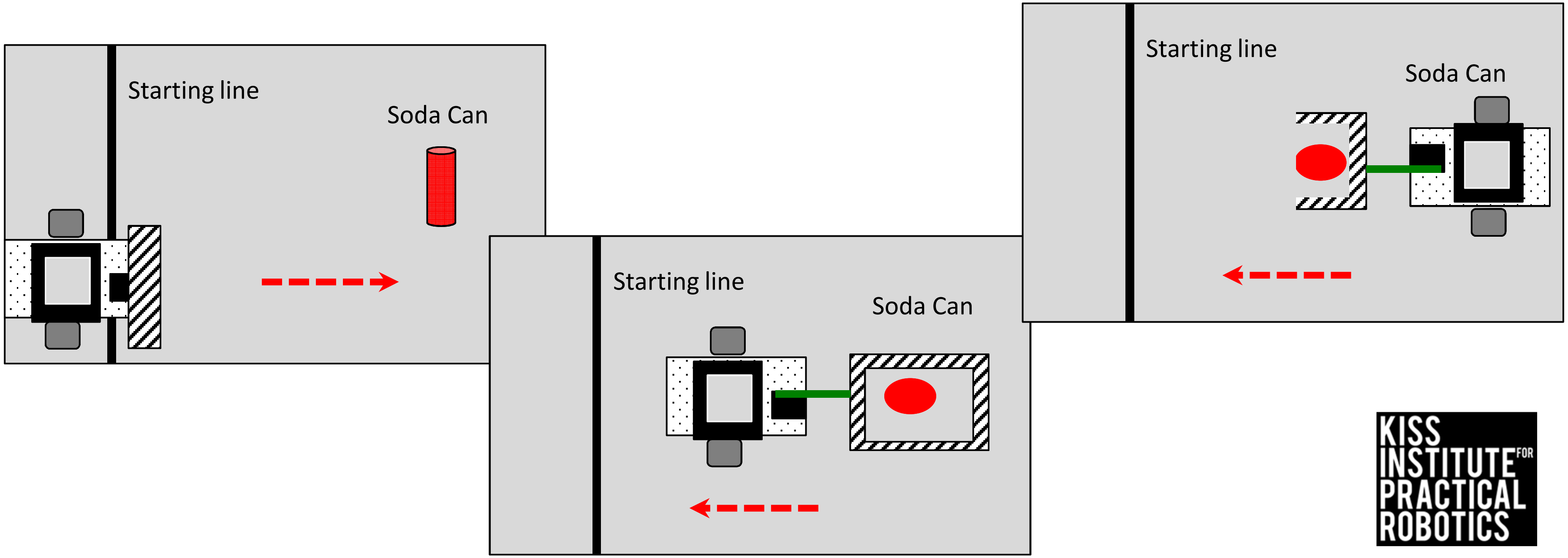
TEACHER'S RESOURCE

Robots must start on or behind the starting mark and move to the object with the goal of bringing the can back to the starting line.

Make the arm/claw/grabber start in the upright position and then lower itself after starting or approaching the object.

Extensions

- Move the can to various distances
- Make the object smaller and harder to navigate to
- Math- have them measure the distance to the object and time the robot and then calculate rate/speed

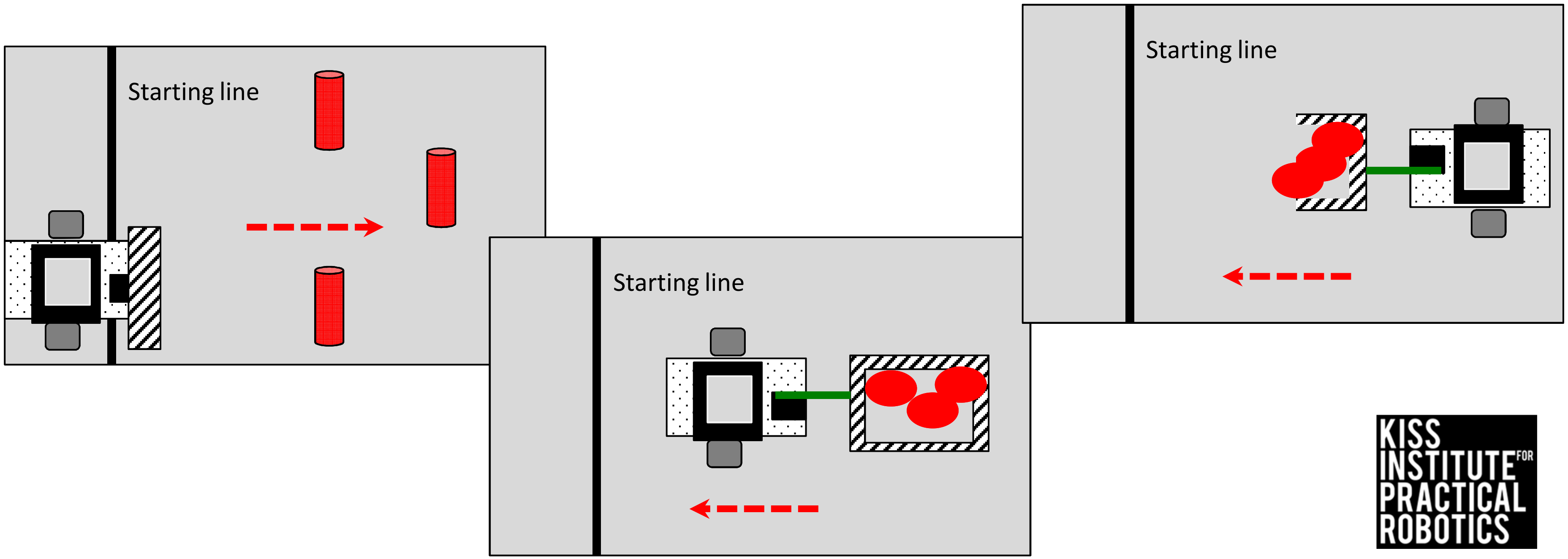


KISS INSTITUTE FOR PRACTICAL ROBOTICS

Recycle the Can(s)

Same as recycle the can only with more objects

- <u>Place the items at known or set locations</u>
    - This is because you are still "dead reckoning", once we learn more logic and decision making, we can program smarter robots that will use sensors to locate and find the objects, which can then be tagged and removed



Engineering Design

TEACHER'S RESOURCE

Goals

- To help students understand how to use the engineering design when building their robots
- Give students practice building with LEGO
- To compare and contrast different types of effectors
- To analyze a task first and then think about the design of an effector

Preparation

- Have a supply of LEGO available for students to build with

Activity

- Have students build mystery structures with their LEGO
    - This gets them familiar with the LEGO pieces and how they work
    - Student tend to over build and make effectors TOO HEAVY for the task or the motor
        - Have students look at pictures of towers and bridges to see how they are constructed
            - Point out that triangles are very strong and are often a good way to go
- Have students build structures for the tasks

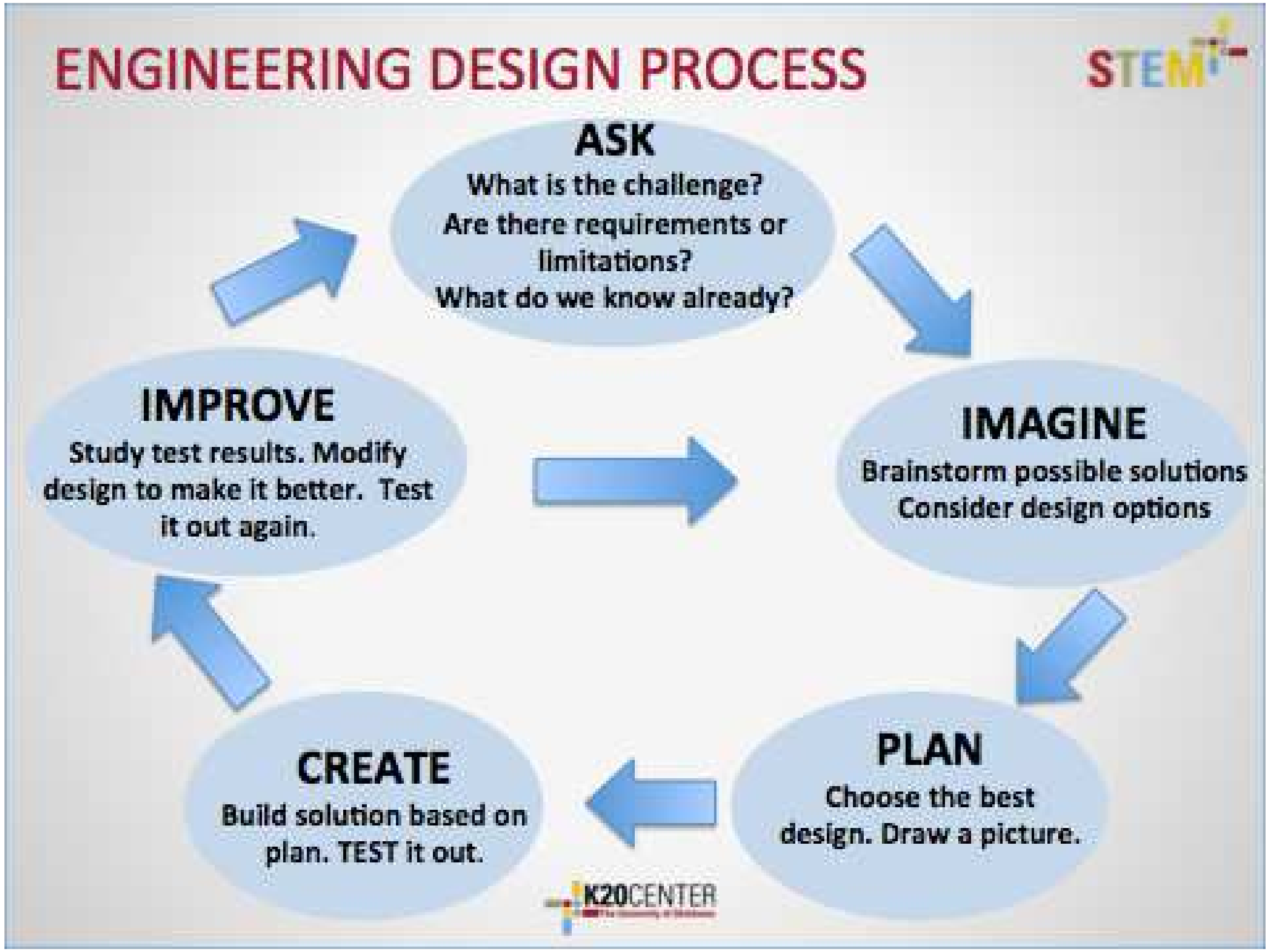
*A great reference is the Art of Building with LEGO included in your flash drive.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Engineering Design Process

Engineers use this process to design, test and produce products.



KISS INSTITUTE FOR PRACTICAL ROBOTICS

Build with LEGO

STUDENT ACTIVITY

- Using the LEGO provided by your Teacher build a structure that:
  - Is the highest free standing tower
  - Is the longest cantilevered bridge
  - Can support the weight of a can of soda the highest off of a table

USE THE ENGINEERING PROCESS AS YOU COMPLETE THESE ACTIVITIES

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Bulldozers

Have you seen a bulldozer working before?

What job does it complete?

Thinking about the blade on the front of the bulldozer

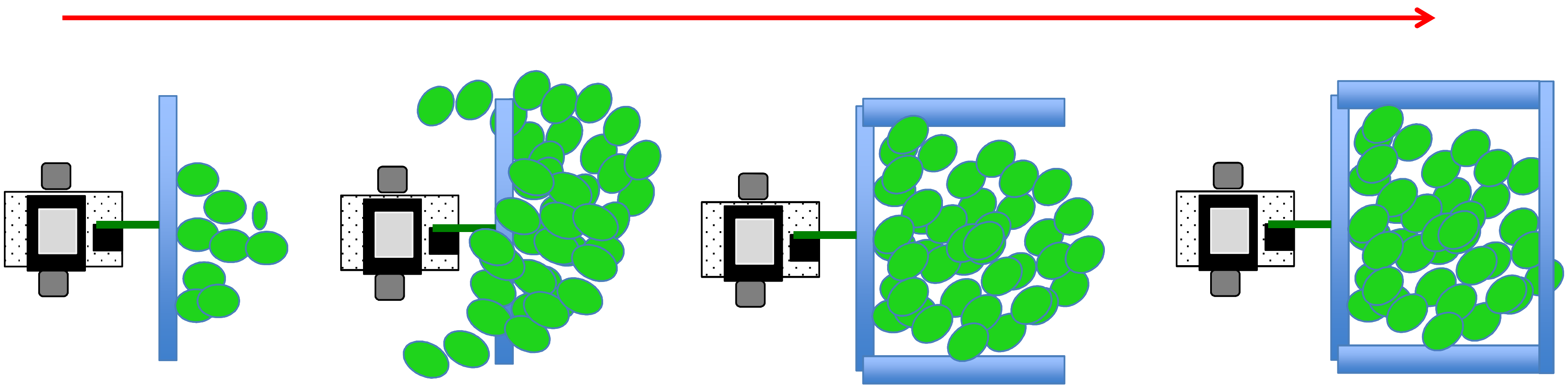
- It is great for PUSHING objects
- Not good at pulling objects
- Not good at picking objects up

KISS INSTITUTE FOR PRACTICAL ROBOTICS

If your task is pushing something

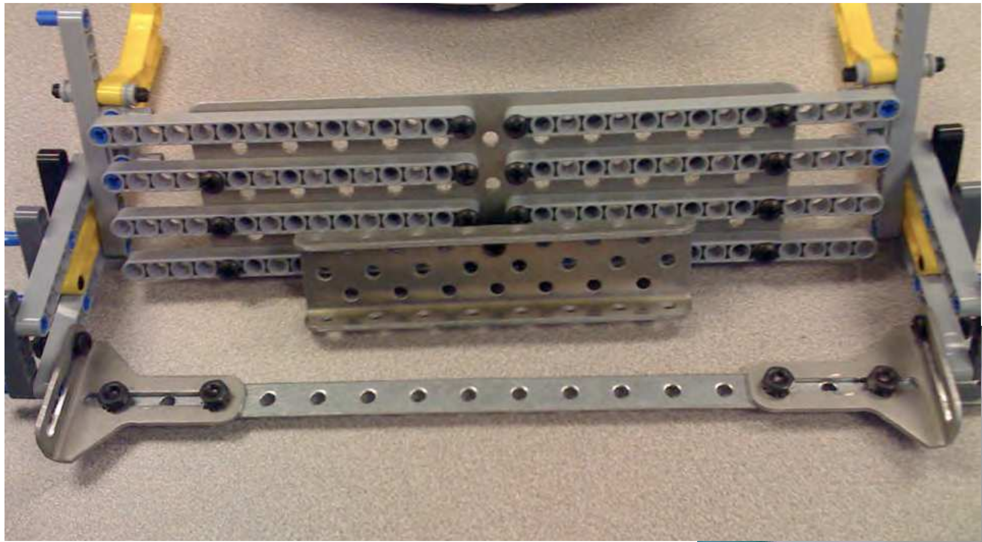
A flat front blade like a bull dozer will work

- Unless there is too much stuff
  - Sides will help
- If you have to turn or back up, sides and a front will help
- Now the front has to be lowered over the objects



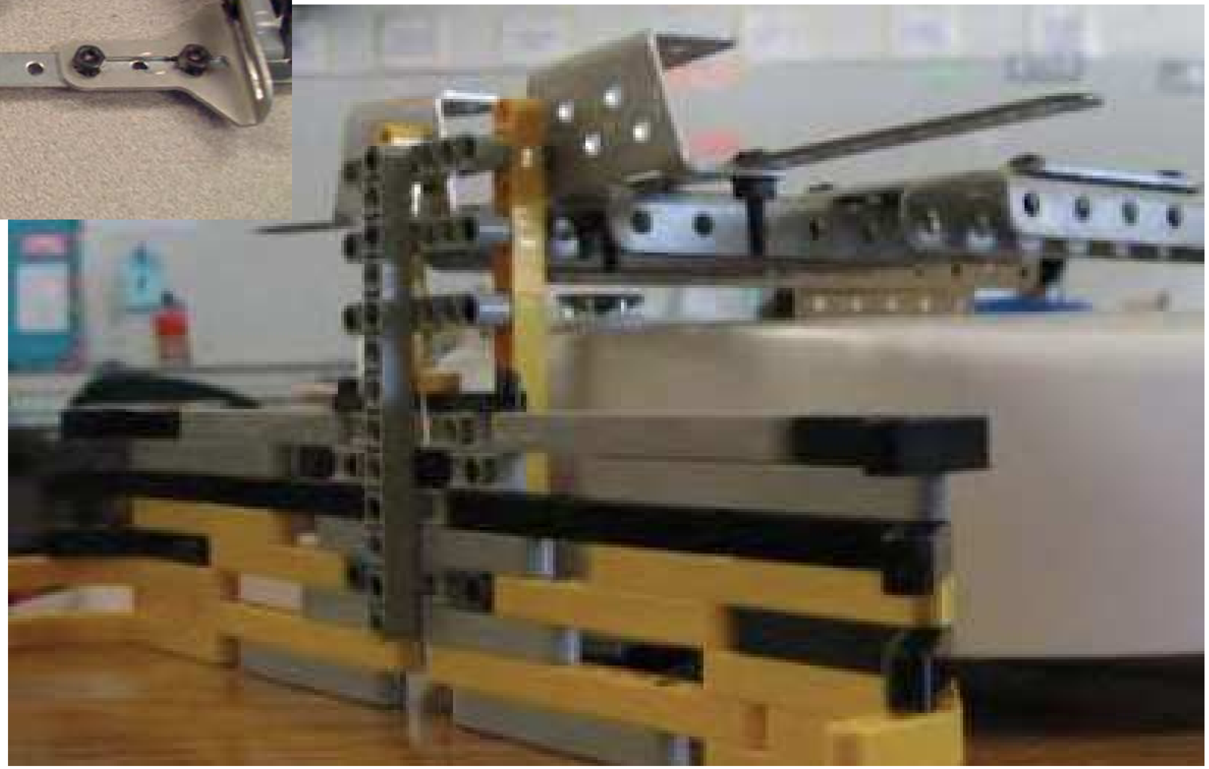
KISS INSTITUTE FOR PRACTICAL ROBOTICS

Some bulldozer blade designs on robots



What task are these designed for?

What are the advantages of these designs?

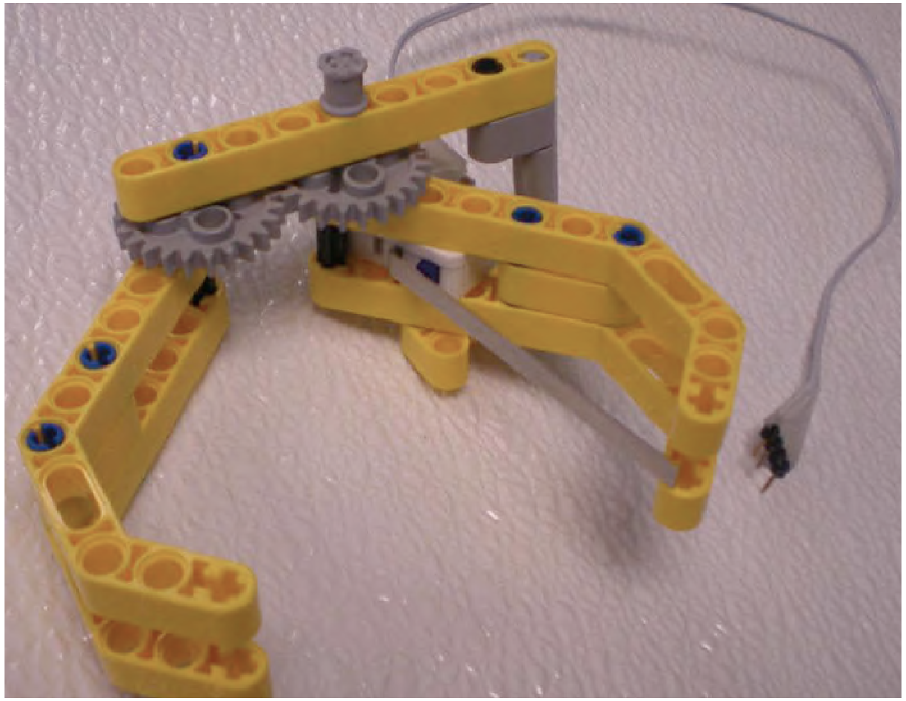


What tasks wouldn’t these designs work well for?

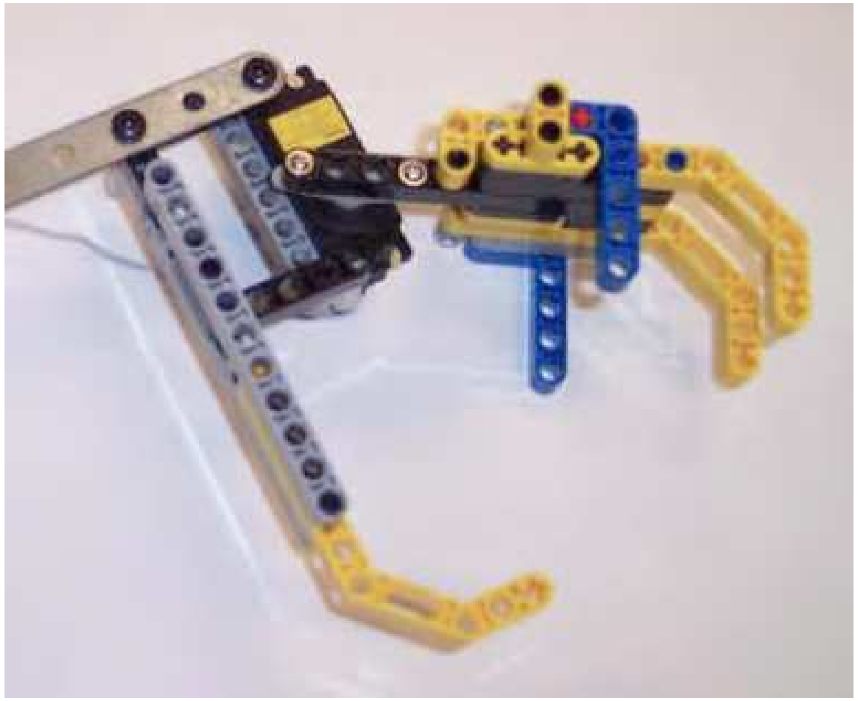
What are the disadvantages of these designs?

KISS INSTITUTE FOR PRACTICAL ROBOTICS

If you have to grab something and pick it up a claw will work well



Notice the long lever sensor to tell when something is in the claw



Notice one side is fixed and the other is moved by the servo

Let the Game Begin- Again

Complete the following activities

- Use the engineering design process to engineer your effectors

Bulldozer Mania

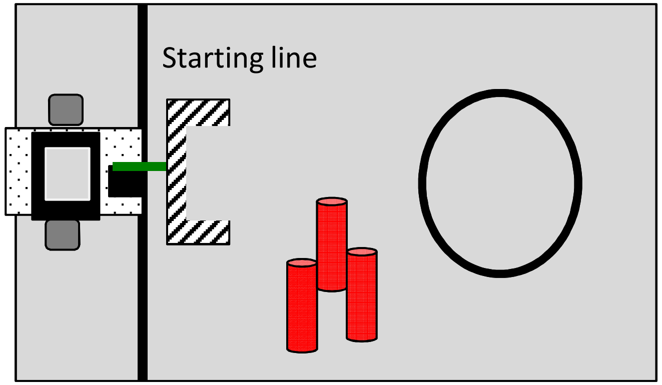
Push as many objects as possible into the designated area

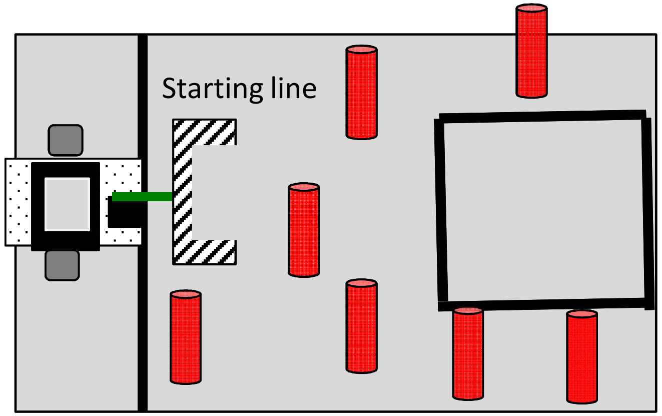
Engineering*

You will need to engineer some kind of a pushing device for the front of the robot (use LEGO, KMP or any type of construction material)

  - Think about what a bulldozer looks like

- Objects can be anything as long as they are relatively easy to push
- Score points for every item in the area
  - Items "off" the official track are lost (no points)
- Make large piles that are easier to get
- Spread the items out to make it harder
- Place the items at random on the board





STUDENT ACTIVITY

TEACHER'S RESOURCE

KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

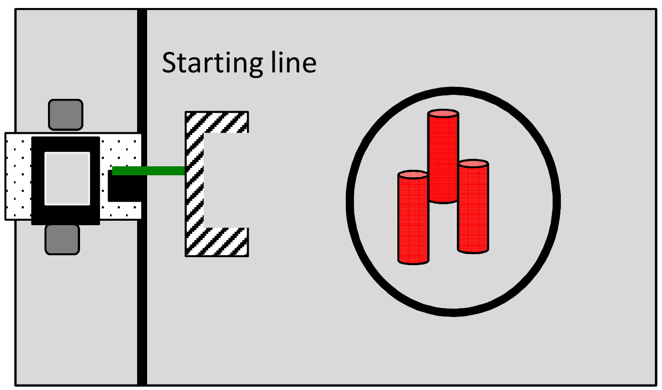
Bulldozer Mania

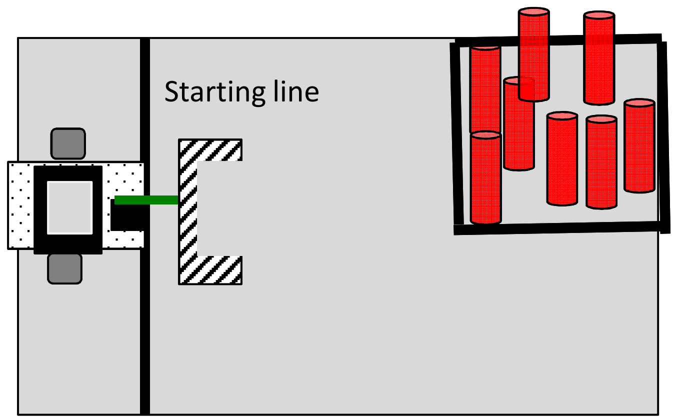
TEACHER'S RESOURCE

Variation

Push as many objects as possible out of the designated area

- Score points for every item NOT in the area
  - Items “off” the official track are lost (no points)
- Make the objects harder to move, use a full can of soda





KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

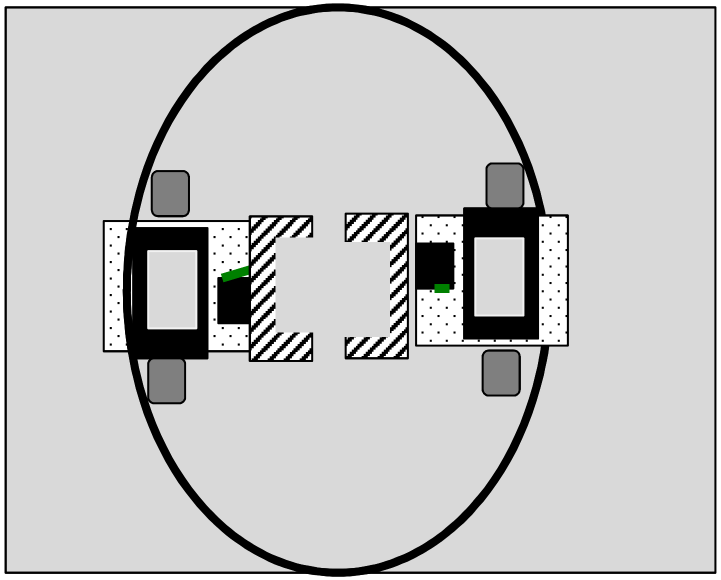
TEACHER'S RESOURCE

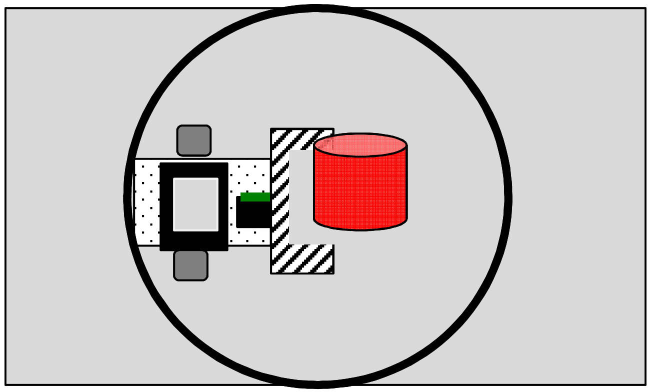
Sumo

Push the other robot or object out of the designated area

Engineering*

- You will need something on the front of the robot to help push the object or other robot
- Win a round by pushing the other robot or object out of the designated area
- Make the object harder to move, use a larger can of soup, etc.





KISS INSTITUTE FOR PRACTICAL ROBOTICS

Decision Making and Sensors

TEACHER'S RESOURCE

Goals

- To help students understand how to use sensors with their robots
- To understand the logic of programming with sensors
- To understand how to write and use a `while` loop
    - Understand that the `while` loop doesn't use a semicolon terminating statement as the program keeps looping
- To use a digital lever sensor to sense when something is touched
- To distinguish between analog and digital sensors

Preparation

- Have KISS IDE up and running
- Have a robot ready to go
- Students will need a long lever touch sensor
- Print out the Boolean Logic Card for each student (on next resource slide) OR
    - Put it on the wall, project it, etc.
- Print out the Boolean operator table (in resource slide to follow)

Activity

- Have students complete the Boolean operator table (true or false)

KISS INSTITUTE FOR PRACTICAL ROBOTICS

> Greater than
< Less than
>= Greater than or equal
<= Less than or equal
== Equal to
!= Not equal to
&& And - used to put several together
|| Or - used to select both options
! Not

KISS INSTITUTE FOR PRACTICAL ROBOTICS

> Greater than
< Less than
>= Greater than or equal
<= Less than or equal
== Equal to
!= Not equal to
&& And - used to put several together
|| Or - used to select both options
! Not

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Statement	TRUE	FALSE
Example 13< 10		X
5 == 4		
5 != 4		
2 <= 6		
500 > 499		
23 < 300		
4 == 4		
32!= 32		

Condition	Write the Statement
five is less than or equal to nine	**5 <= 9**
Three is equal to Three	
four is equal to four	
Five is not equal to four	
five hundred is greater than two	
thirty is greater than or equal to 5	

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Statement	TRUE	FALSE
Example 13< 10		X
5 == 4		
5 != 4		
2 <= 6		
500 > 499		
23 < 300		
4 == 4		
32!= 32		

Condition	Write the Statement
five is less than or equal to nine	**5 <= 9**
Three is equal to Three	
four is equal to four	
Five is not equal to four	
five hundred is greater than two	
thirty is greater than or equal to 5	

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

KISS INSTITUTE FOR PRACTICAL ROBOTICS

What is a sensor?

Sensor are detectors that measure a parameter and convert it into a signal that provides information (value) to your controller

- Proprioceptive sensors
  - Report on the current state of the robot itself
    - Much like you know if you are sitting down or standing up even if you are blindfolded
    - Examples: encoders, gyros, low-voltage sensors
- External sensors
  - Report on the current state of the world
    - Much like you can see if the light is on or feel when the temperature outside gets colder
    - Examples: light sensors, range sensors, touch sensors

KISS INSTITUTE FOR PRACTICAL ROBOTICS

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 code runs and 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:
    - **Analog**- Return whole number values between 0-1023 (10bit analog = 2^{10} or 1024- remember we start counting at 0)
      - Light, small top-hat, ET
    - **Digital**- Return a value of 0 or 1 (true of false)
      - Small touch, large touch, lever

*You can find sensor information in the sensor and motor manual on the KISS IDE help

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Smarter Robots

Sensor Functions

You call for the analog sensor value with a function

- You have 8 analog ports (0-7)

```
Analog10(Port#);        Analog10(1);
```

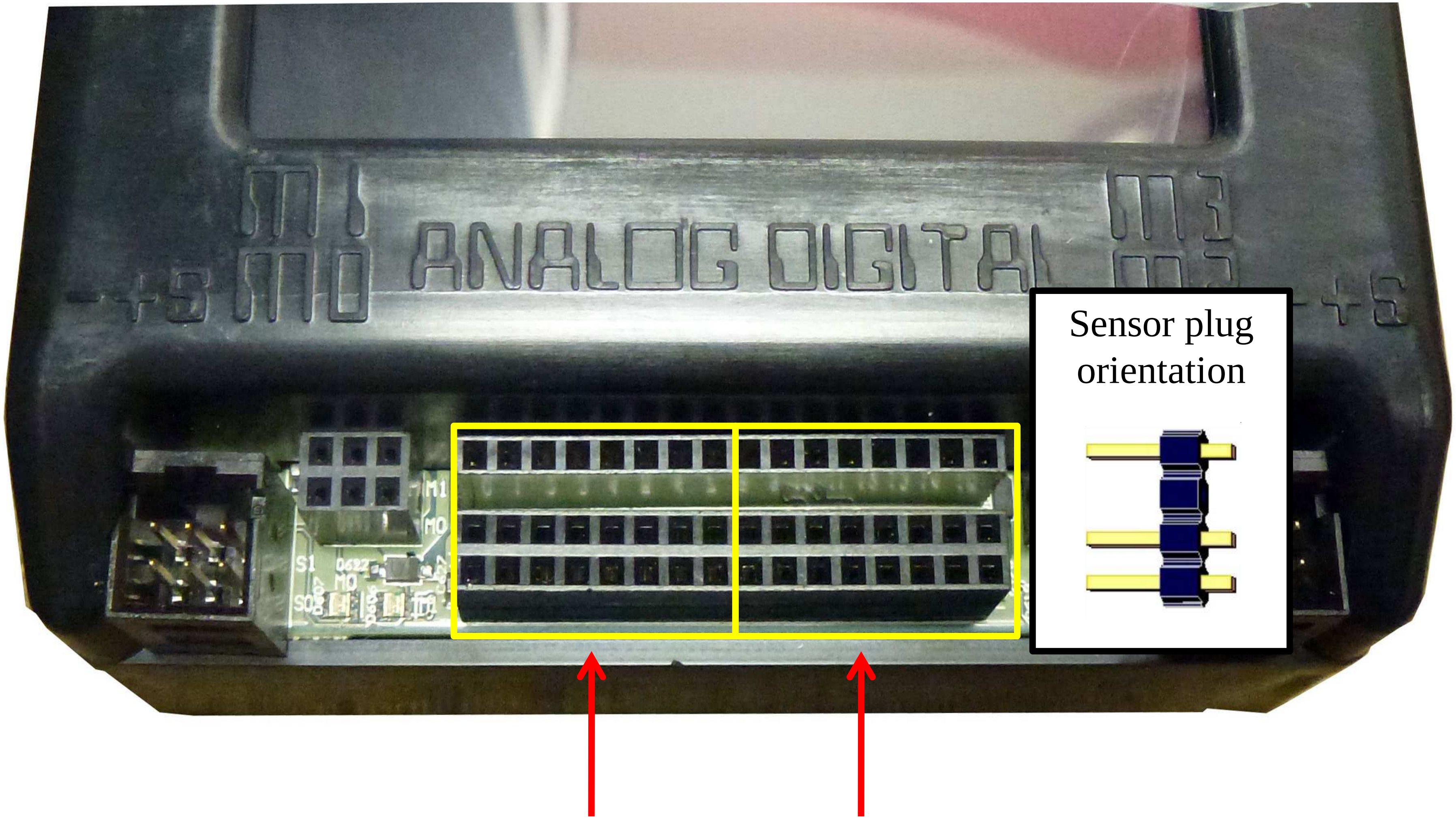
You call for the digital sensor value with a function

- You have 8 digital ports (8-15)

```
Digital(Port#);     Digital(8);
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Sensor Ports



analog ports (0-7) and digital ports (8-15)

KISS INSTITUTE FOR PRACTICAL ROBOTICS

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

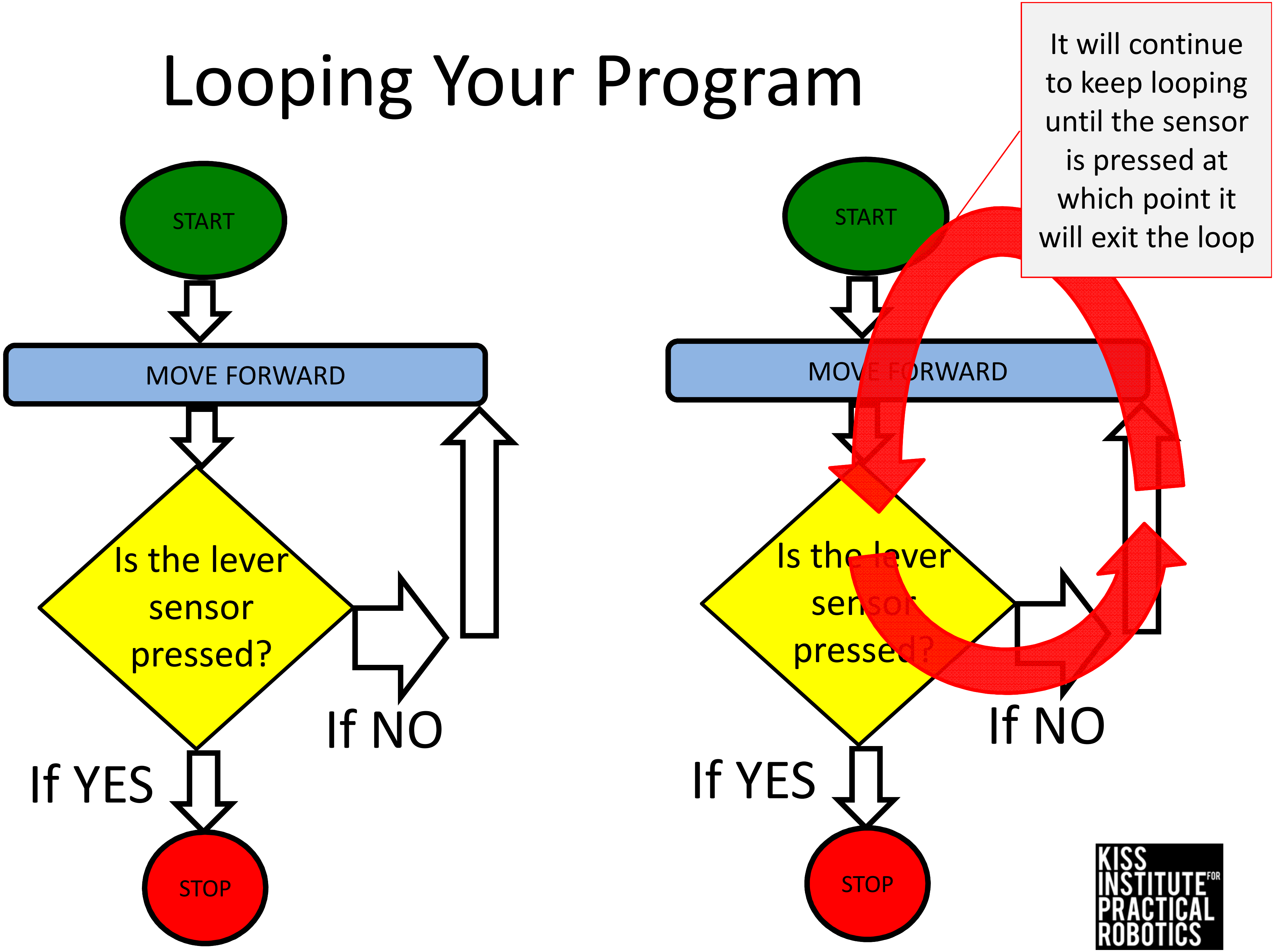
*Until you are familiar with the Operators that you will be using, you can use the “cheat sheet” for easy reference.

The Problem With Reading 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 read the sensor values

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Looping Your Program



KISS INSTITUTE FOR PRACTICAL ROBOTICS

We accomplish this loop with a `while` statement

Keep the block of code running (looping) until 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
}
```

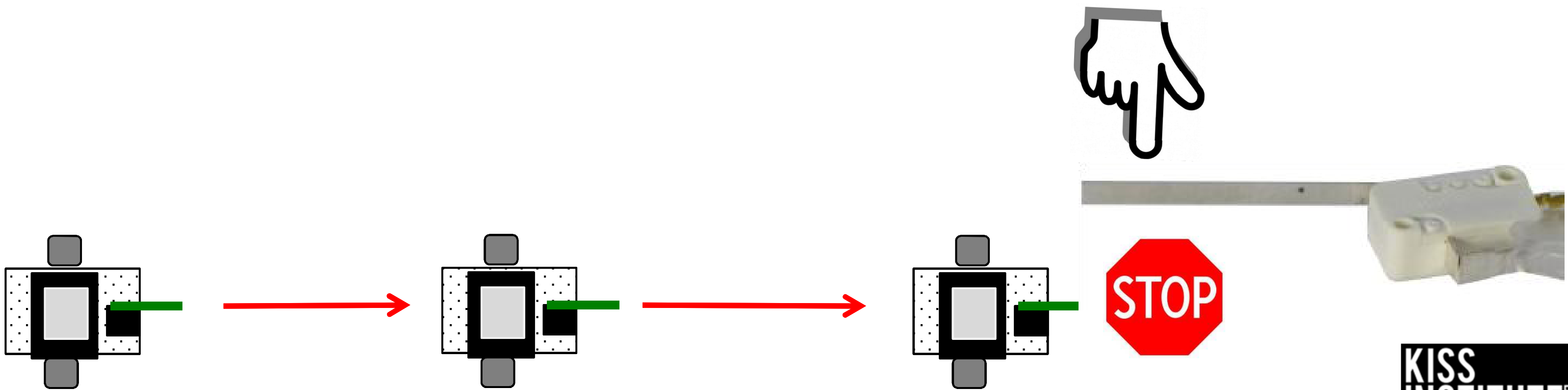
Notice there is no terminating semicolon after the while statement

STUDENT ACTIVITY

Drive Until Bump Activity

- Robot will drive forward until the long touch sensor is pressed

  - You can hold the sensor while the robot is moving and manually trigger it
- You will need a long lever touch sensor
- Plug it into any of the digital ports (8-15)
  - Write a program using a `while` statement that drives the robot forward until the lever sensor is activated

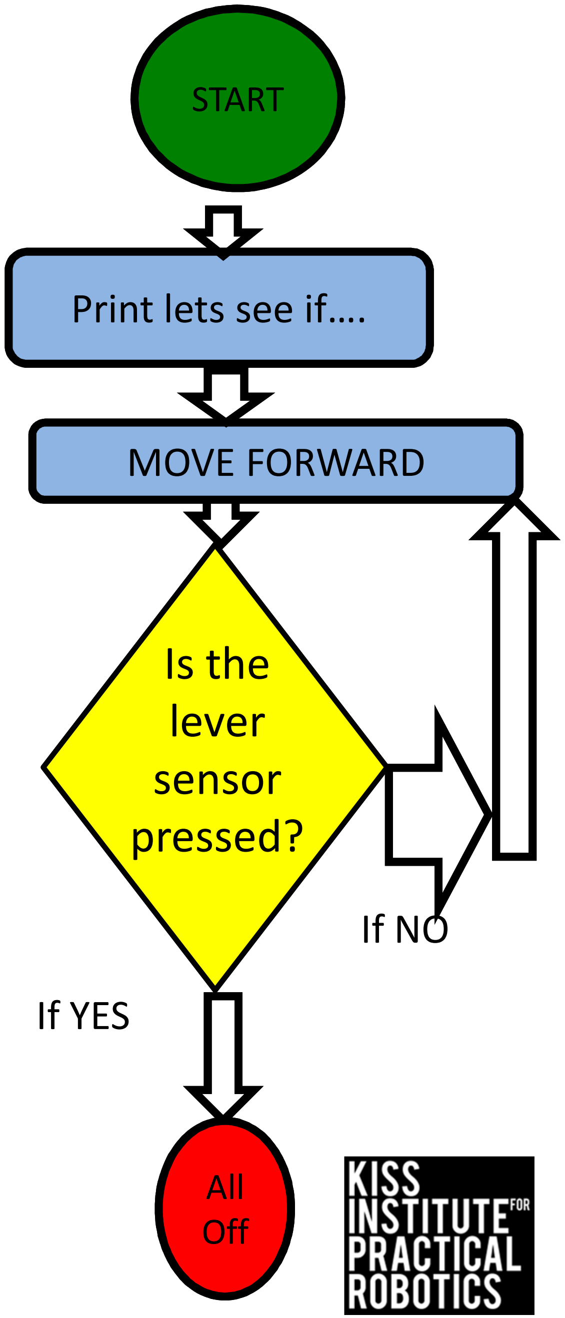


KISS INSTITUTE FOR PRACTICAL ROBOTICS

Drive Until Bump

Psuedocode (Task Analysis)

1. //Print let's see if we can stop with a touch sensor
2. //Pause for 1 second so you can read the screen
3. //Check the sensor value in digital port 15 and when not pressed == 1 (aka true) keep checking and drive forward
4. //Exit loop when sensor value in digital port is pressed == 0 or !=1 (aka NOT true)
5. //Shut everything off



KISS INSTITUTE FOR PRACTICAL ROBOTICS

Drive Until Bump

TEACHER'S RESOURCE

Solution

```c
// Created on Wed September 4 2013

void drive_forward();
int main()
{
   printf("lets see if we can stop with a touch sensor\n"); //print to screen
   msleep(1000); // pause for 1 second so you can read the screen
   while (digital (15) == 0) //check the sensor value in digital port 15 and when not pressed keep checking
      {
      drive_forward(); // drive forward
      } //exit loop when sensor value digital port 15 is pressed
   ao(); // all off
   return 0;
}
   void drive_forward()
   {
   motor (0, 100);
   motor (3, 100);
   }

```

Notice the function prototype for drive-forward

Notice no semicolon after the `while` statement

This is what the robot does while it is looping

Notice the function definition for drive-forward

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Drive Until Bump

TEACHER'S RESOURCE

Solution

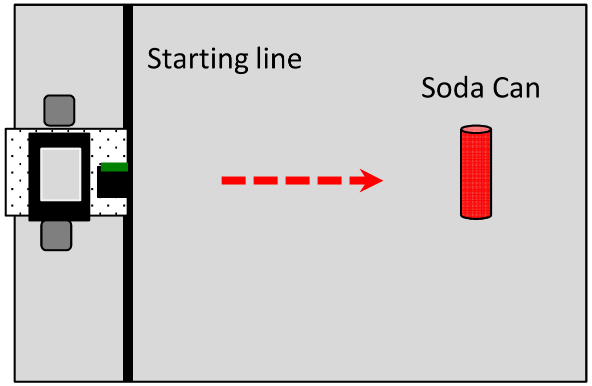
```
// Created on Wed September 4 2013

void drive_forward();
int main()
{
   printf("lets see if we can stop with a touch sensor\n"); //print to screen
   msleep(1000); // pause for 1 second so you can read the screen
   while (digital (15) == 0) // check the sensor value in digital port 15 and when not pressed keep checking
      {
      drive_forward(); // drive forward
      } // exit loop when sensor value digital port 15 is pressed
   ao(); // all off
   return 0;
}
   void drive_forward()
   {
   motor (0, 100);
   motor (3, 100);
   }
```

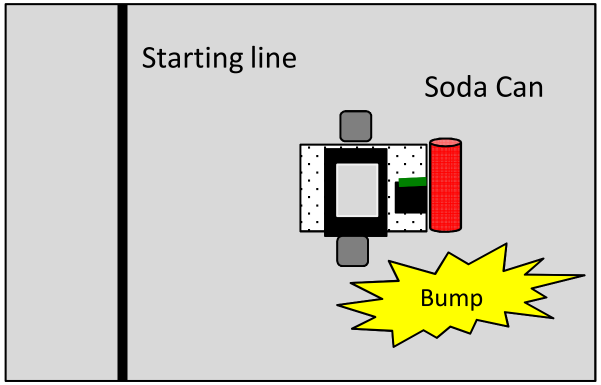
Bump the Can and Go Home

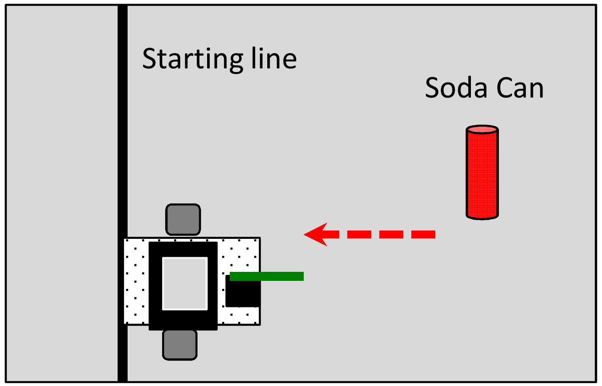
A variation on Touch, Closest to and Recycle the Can.

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







KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

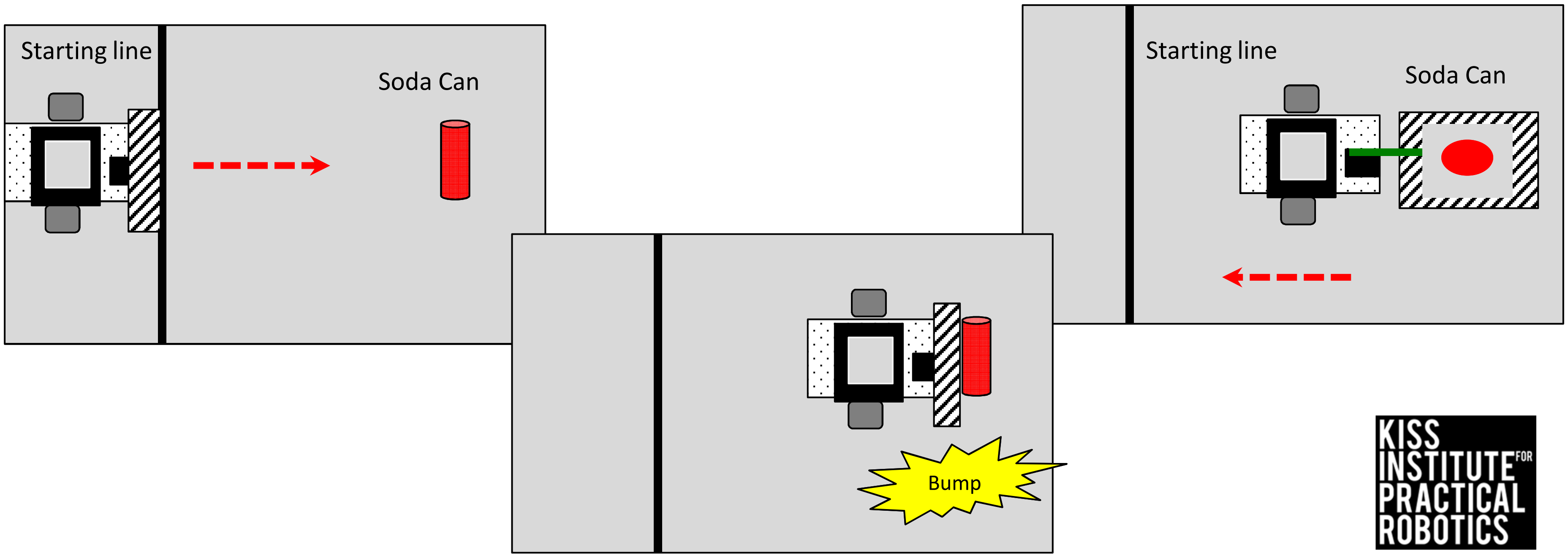
TEACHER'S RESOURCE

Capture the Can/Flag

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 and then have an arm with a Grabber/Claw that is lowered/closed around the can
- Use the long lever sensor to detect when you have touched the can and then lower your arm/claw/grabber to get the can
    - Many claw/grabber designs have a touch sensor that triggers them to close on an object
- Return the can to the starting line
- Move the can to various distances and locations



KISS INSTITUTE FOR PRACTICAL ROBOTICS

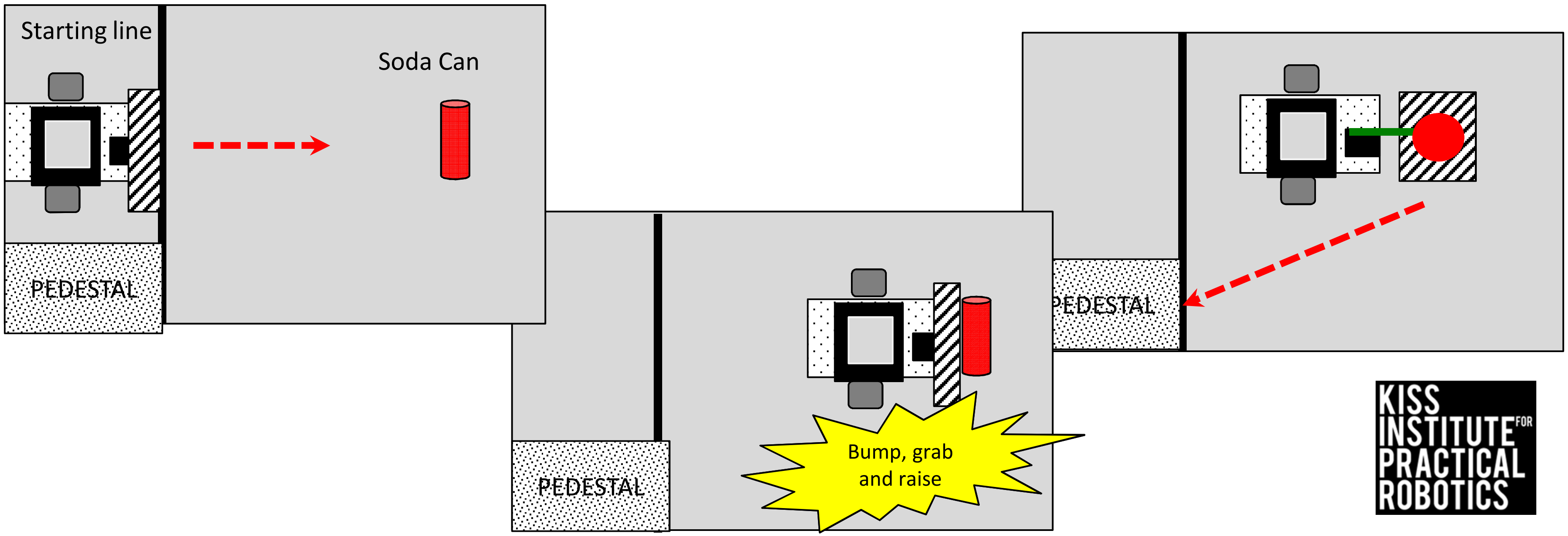
Can on a Pedestal

A variation on Touch, Closest to and Recycle the Can

You will need a thick book (2-3 inches), dictionary, etc.

Engineering*

- Students will need to engineer a grabber/claw that will grip the object so that it can be raised and lowered (simple bulldozing will not work) at least high enough to put on the pedestal
- Use the long lever or other touch sensor to detect when you have the can within your open claw so that you can grab it and raise it off the ground
- Place the can/object on top of the book (pedestal)
- Move the can and the pedestal to various distances and locations



KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

while Loop Operating a Servo

Suppose we want to have a servo move from position 200 to position 1800 in steps of 100

- We could do this by writing 16 separate **set_servo_position** commands
- With less effort and far better efficiency, this can be done by using a **while** loop

```
// Created on Wed September 4 2013


int main()
{
   enable_servos (); // turn power on to the servos
   set_servo_position (2, 200); // move servo 2 to position 200
   msleep(100); // give servo time to move
   while (get_servo_position (2) < 1800)
   {
   set_servo_position (2, get_servo_position (2) + 100); // move servo 2 in steps of 100
   msleep(100); // give it time to move
   }
   ao();
   return 0;
}
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

while Loops continued

We can use successive while loops if needed to get the desired behavior

Write a program that:

```
//Announces the program
//Starts with a light
//Drives forward until large lever
sensor bumps
//Stops the motors
//Prints all done
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

while Loops continued

TEACHER'S RESOURCE

```
// Created on Wed September 4 2013

void drive_forward();
int main()
{
    printf ("start with the light and drive straight until bumped\n"); //announce program
    while (analog10(3) > 500) // check value of light sensor in analog port 3 when value is more than 500 ao
    {
        ao();
    }
    while (digital (9) ==0) // check digital port 9 and when touch sensor isnt pressed drive forward
    {
        drive_forward();
    } //when digital 9!= 0 move on to the next line
    ao(); // stop motors
    printf ("all done\n");
    return 0;
}
void drive_forward()
    {
    motor (0,100);
    motor (3,100);
    }
```

Notice the function prototype for drive-forward

Notice no semicolon after the `while` statement

Notice no semicolon after the `while` statement

Notice the function definition for drive-forward

KISS INSTITUTE FOR PRACTICAL ROBOTICS

IF Statements and Following Lines

TEACHER'S RESOURCE

Goals

- To help students understand how to use sensors with their robots
- To understand the logic of programming with sensors
- To understand how to write and use an `if` statement
- To understand how to use the hard and soft buttons on the link
- To understand how to rename the soft buttons on the Link
- To use an IR reflectance sensor to follow a black line

Preparation

- Have KISS IDE up and running
- Have a robot ready to go
- Students will need a small reflectance sensor

Activity

Follow the slides and complete the line following activity

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Buttons

STUDENT ACTIVITY

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

On the KIPR Link there is 1 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
- All have ***name*_button_clicked()** functions which pause if the button is being pressed and then returns 1 when it is released or returns 0 otherwise
- Soft buttons can have their display changed by using **set_*name*_button_text("*display text* ");**
- By default only a, b and c are displayed. The 3 extra buttons can be shown using:

```
extra_buttons_show();
extra_buttons_hide();
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Name Your Buttons Activity

STUDENT ACTIVITY

Psuedocode (Task Analysis)

```
1.//Announce program
2.//Change button a to “start”
3.//Change button b to “stop”
```

```
// Created on Thu September 12 2013

int main()
{
    printf("Button Test \n"); //announce program
    set_a_button_text ("start\n");// change button a text to start
    set_b_button_text ("stop\n");//change button b text to stop
    return 0;
}
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

`if` Statements

`if` statements allow the code being run by the program to be changed (If the bump sensor is pressed, do this)

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

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

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

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Line Following Activity

Using `while` and `if`

You will need a Small Top Hat Sensor

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.

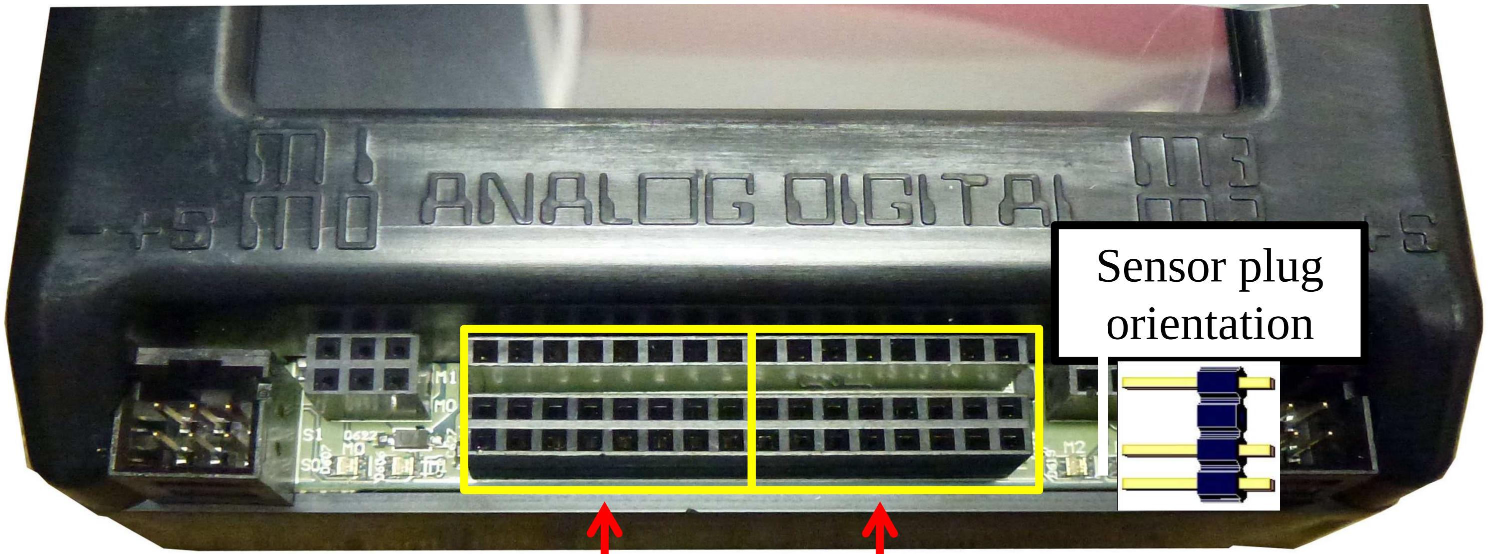
- Amount of IR reflected back depends on surface texture, color and distance to surface

This sensor is excellent for line following

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

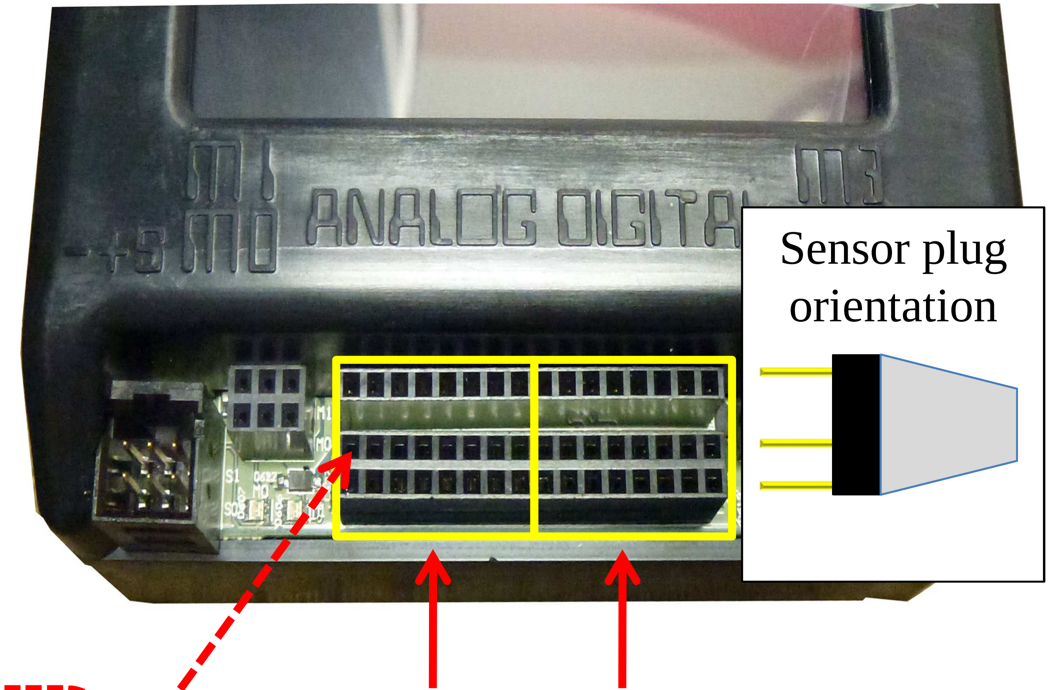
Reflectance Sensor

1. This is an `analog(10)` sensor so plug it into any of your analog ports
   - Values will be between 0-1023
2. Mount the sensor on the front of your robot so that it is pointing to the ground and ~1/8" from the surface



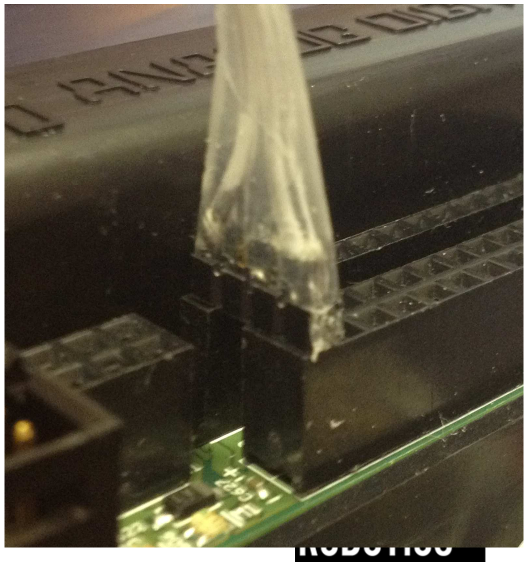
analog ports (0-7) and digital ports (8-15)

Plug in Your Reflectance Sensor



analog ports (0-7) and digital ports (8-15)

Plug your IR sensor into analog port 0

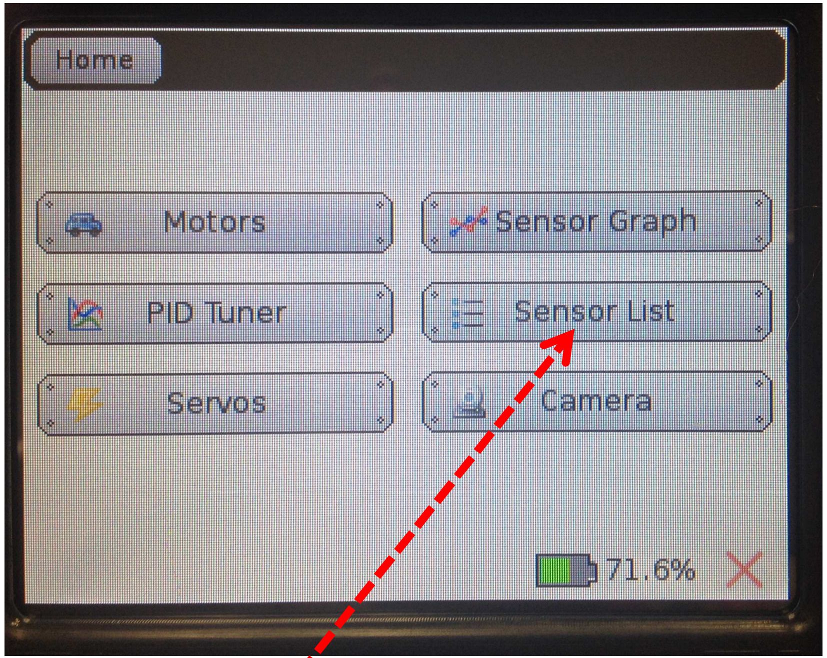


STUDENT ACTIVITY

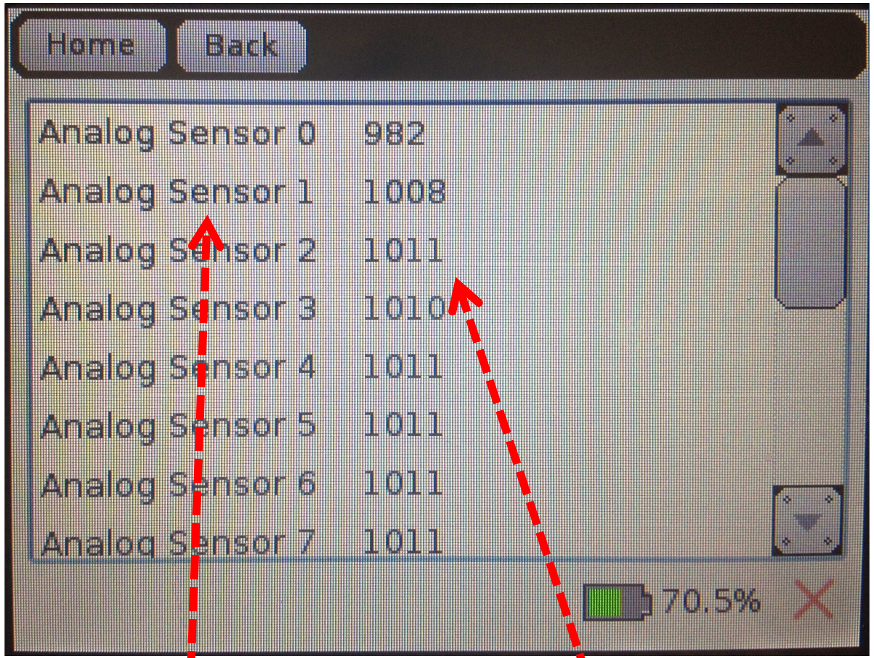
Reading Sensor Values From the Sensor List

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

- 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



Select Sensor List



Sensor Ports

Sensor Values

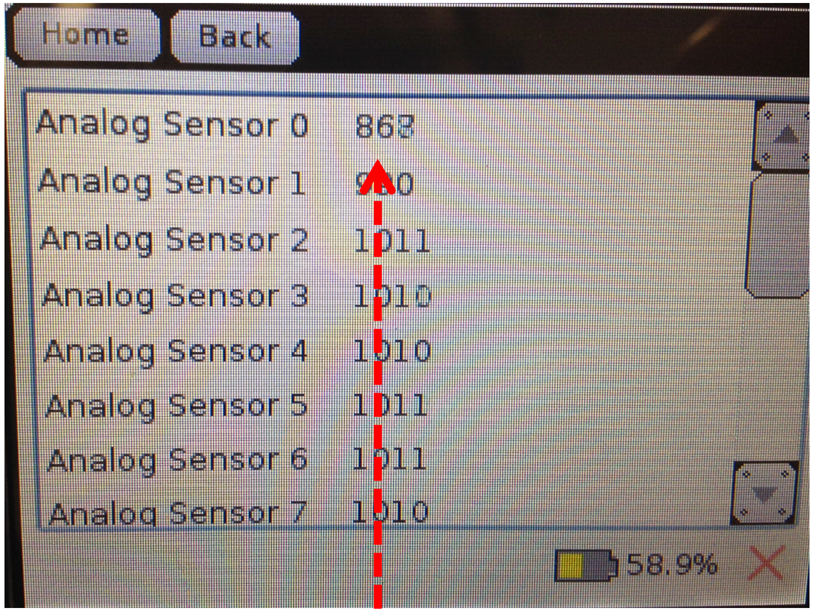
KISS INSTITUTE FOR PRACTICAL ROBOTICS

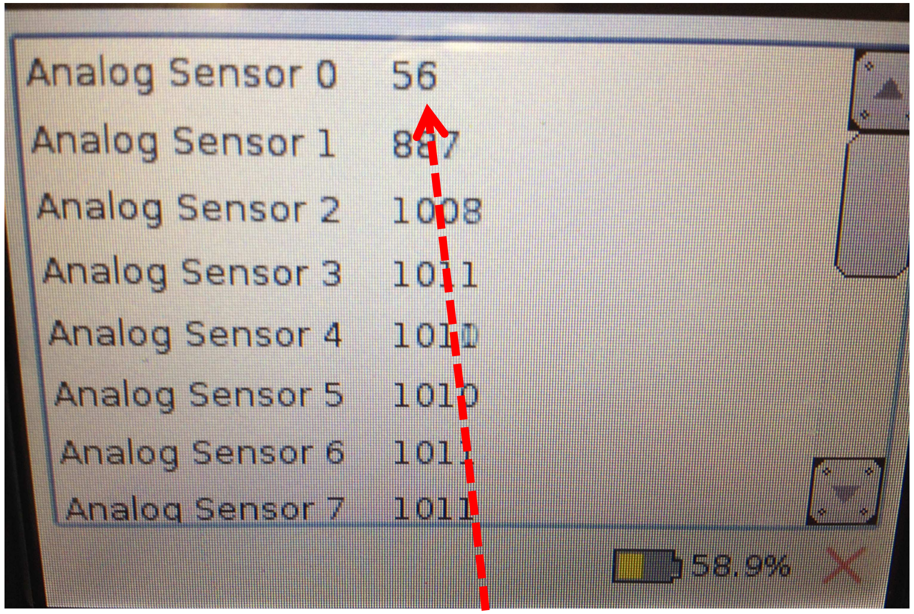
STUDENT ACTIVITY

Reading Sensor Values From the Sensor List

With the IR sensor plugged into analog port #0

- Over a white surface the value is (56)
- Over a black surface the value is (863)





Value of 863 (Black Surface)

Value of 56 (White Surface)

Line Following Activity
Using `while` and `if`

Write a program for your robot that:

Psuedocode (Task Analysis)

```
1. //Announces program
2. //Checks the status of the a button
3. //Checks the value from the reflectance sensor
4. //Turns left if value is >= 512
5. //Turns right if value is < 512
```

Line Following Activity Solution
Using while and if

TEACHER'S RESOURCE

```
// Created on Fri September 13 2013

int main()
{
     printf ("line following program\n"); //annonces program
   while(a_button() ==0)// checks status of the a_button
   {

     If(analog10(5)>=512) //checks value from reflectance sensor in port #5
       { // turn left
       motor(0,90);
       motor(3,10);
       }
     If(analog10(5)<512) // check value from reflectance sensor in port # 5
       { //turn right
       motor(0,10);
       motor(3,90);
       }
   }
}
```

Notice the use of the a button for the while loop. This lets the program run until the button is triggered.

Notice NO semicolon after the if statements

The value of 512 or the "threshold" value is ½ way between the 1024 possible values. Remember black reflects less IR than white so the value is lower.

Notice the Boolean operators >= 512 or < 512

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Line Following Activity Solution

TEACHER'S RESOURCE

Notice the misspelling of “announces”. Comments can be misspelled or even shorthand, as long as your team understands what it says. The computer will not execute any comments, you can even write a poem in the comments!

```
// Created on Fri September 13 2013

int main()
{
      printf ("line following program\n"); //annonces program
   while(a_button() ==0)// checks status of the a_button
   {

      If(analog10(5)>=512) //checks value from reflectance sensor in port #5
         { // turn left
         motor(0,90);
         motor(3,10);
         }
      If(analog10(5)<512) // check value from reflectance sensor in port # 5
         { //turn right
         motor(0,10);
         motor(3,90);
         }
   }
}
```

The `if` statements are all within the `while` loop. This lets the program run until the while statement is not true (a_button pressed)

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Line Following Activity Solution Tip

TEACHER'S RESOURCE

```
// Created on Fri September 13 2013

int main()
{
    printf ("line following program\n"); //annonces program
  while(a_button() ==0)// checks status of the a_button
  {

    if(analog10(5)>=512) //checks value from reflectance sensor in port #5
      { // turn left
      motor(0,90);
      motor(3,10);
      }
    if(analog10(5)<512) // check value from reflectance sensor in port # 5
      { //turn right
      motor(0,10);
      motor(3,90);
      }
  }
}
```

The program can get hard to read. One way to make it easier is to make sure your curly braces { } are lined up

KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Follow Me

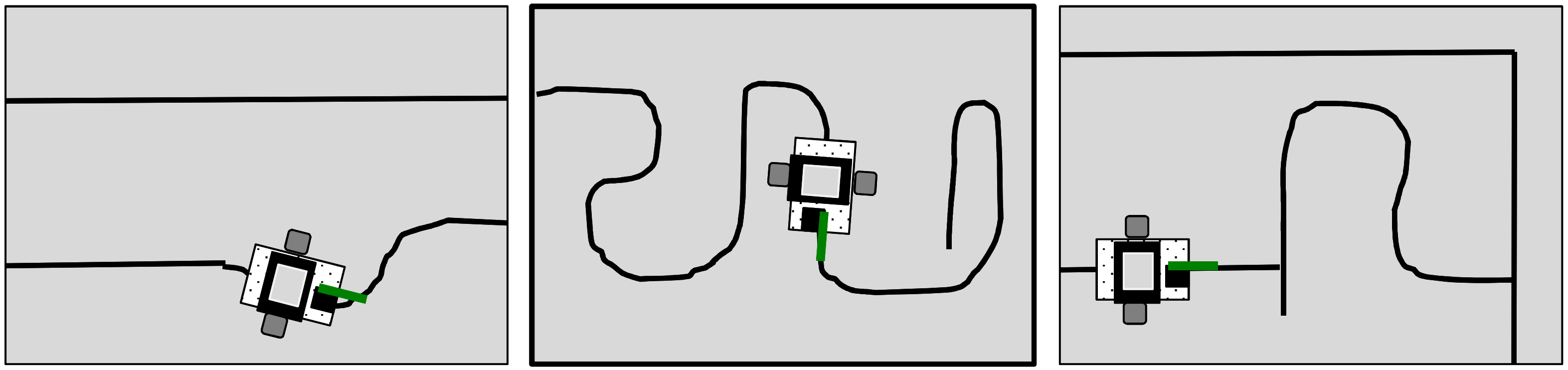
TEACHER'S RESOURCE

Using the reflectance sensor(s) have your robot follow the line

- You can make this a time trial
- Start with a straight line and then move on to curved lines
    - The tighter the turn the harder it is to follow
- Have the line come to a T intersection

Engineering*

- Students need to attach the reflectance sensor(s) to the front of their robot
- Have the students use a sensor on each side of the line to see if it improves performance
- Is it better to have the sensor(s) in the front or the back of the robot?
- How far apart should they be?



KISS INSTITUTE FOR PRACTICAL ROBOTICS

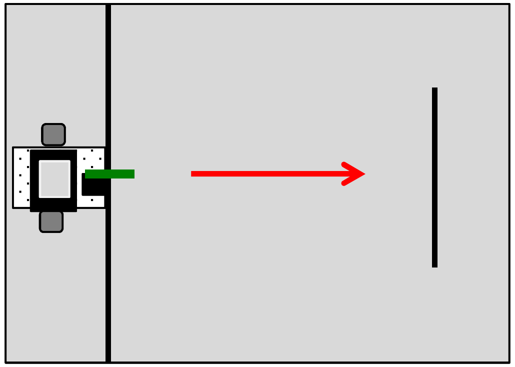
STUDENT ACTIVITY

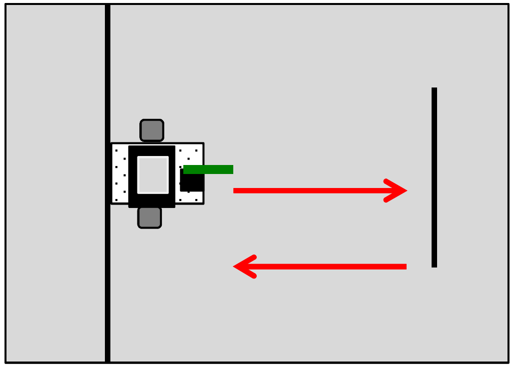
Find Black and STOP

TEACHER'S RESOURCE

Using the reflectance sensor(s) have your robot drive forward until it senses a black line at which point it stops

- Move the line to various distances
- Make the robot find the line, stop and then back up to the starting line





KISS INSTITUTE FOR PRACTICAL ROBOTICS

Measuring Distance Using the ET

TEACHER'S RESOURCE

Goals

- To help students understand how to use sensors with their robots
- To understand the logic of programming with sensors
- To write a program to print a sensor value to the screen
- To use a range finder sensor (ET) to measure a distance

Preparation

- Have KISS IDE up and running
- Have a robot ready to go
- Students will need a range finder sensor

Activity

Follow the slides and complete the activity

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Measuring Distances

You will need the ET Sensor

The "ET" sensor gets its name from the shape of the sensor resembling a famous movie Extra Terrestrial.

This sensor works by sending out an IR beam and measures the angle the reflected IR light returns at and triangulates the distance to an object.

- Maximum detection distance: 80cm

This sensor makes a great medium range distance sensor

- The sensor reads the highest value when it detects an object at 5cm, and value decreases if your object gets closer or farther away
  - One way to fix that is to mount the sensor in such a way that nothing can get closer than 5cm

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Reflectance Sensor

1. This is an `analog(10)` sensor so plug it into your analog ports
2. **!FLOATING PORT!**
3. For this sensor to work properly you must always change the analog port that you plug it into to a floating point!
   - This is the only sensor that you have to do this for
     - Don't worry, we have a function for this

```
set_analog_pullup(7,0);
```

Port # 7

The 0 sets it to floating

Put this in your code right after the

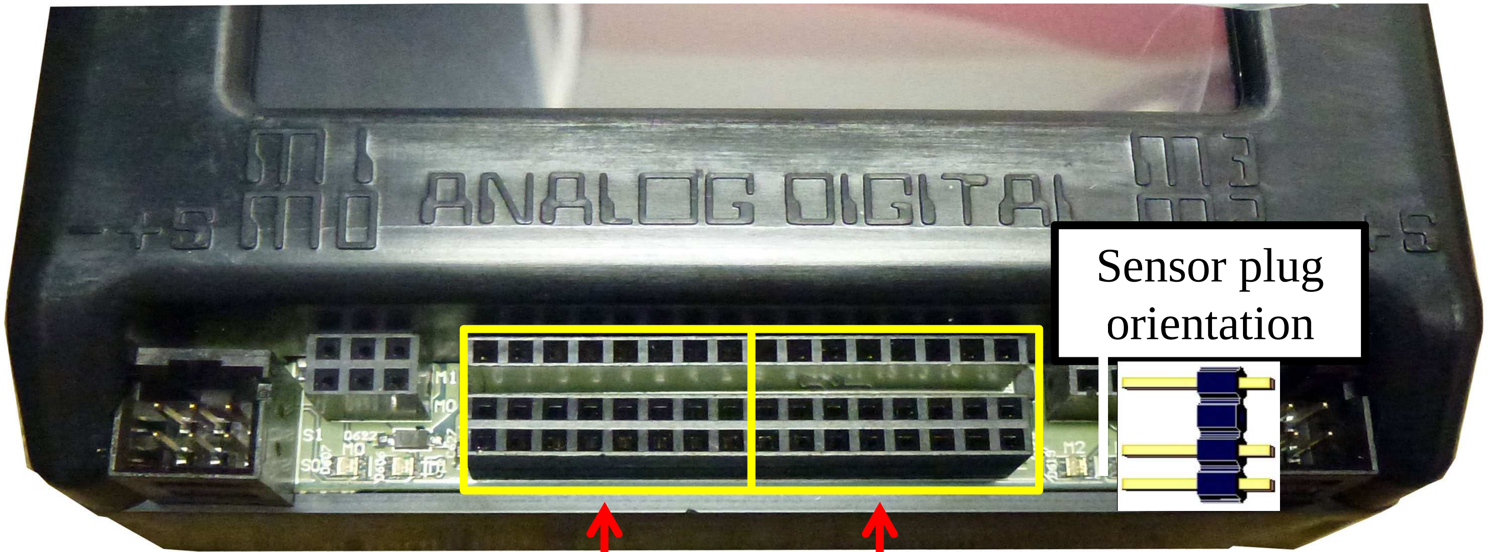
```
int main()
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Reflectance Sensor

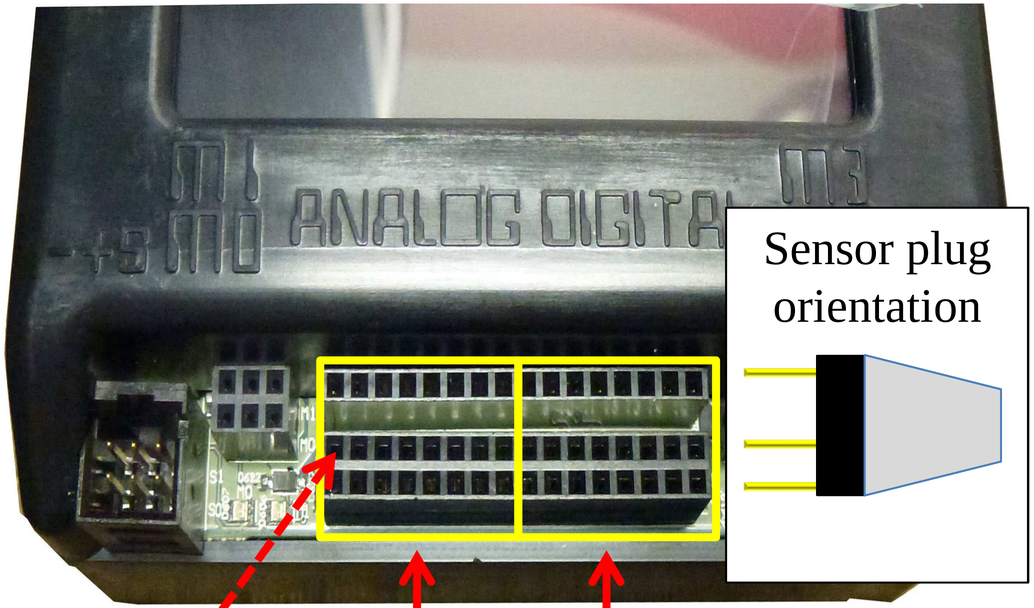
1. Mount the sensor on the front of your robot so that it is pointing forward
2. Plug the ET into one of your analog ports and remember the port #



analog ports (0-7) and digital ports (8-15)

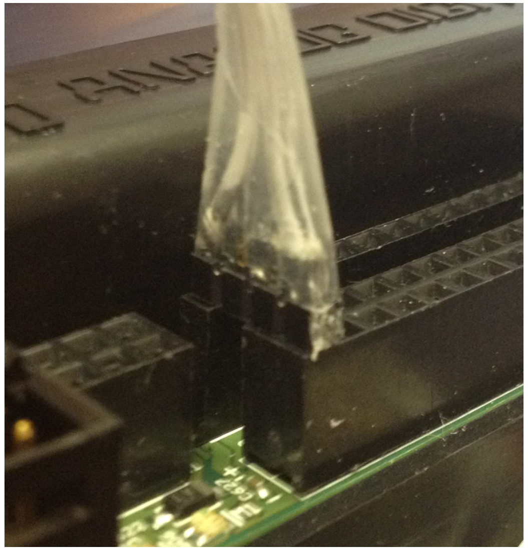
KISS INSTITUTE FOR PRACTICAL ROBOTICS

Plug in Your Reflectance Sensor



analog ports (0-7) and digital ports (8-15)

Plug your IR sensor
into analog port 0



KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Because you need a floating point you cannot use the sensor list to read values

Use the following code to print the value to the Link screen

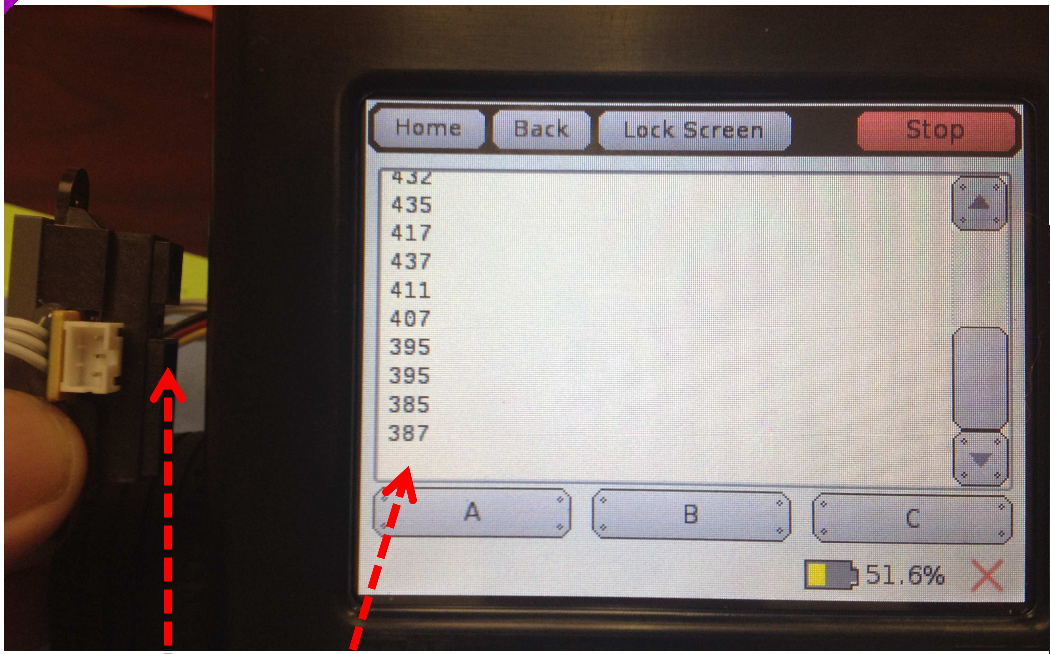
```
// Created on Thu October 3 2013

int main()
{
   set_analog_pullup (0, 0); //set analog port 0 to floating
   while (a_button() != 1) // while a button is not pressed
   {
      printf ("%i\n", analog10 (0)); // %i is a place holder on the screen for the value, in this case an integer returned by analog 0
      msleep (200);// prints the value at 5 readings per second to give you time to read them
   }
   return 0;
}
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

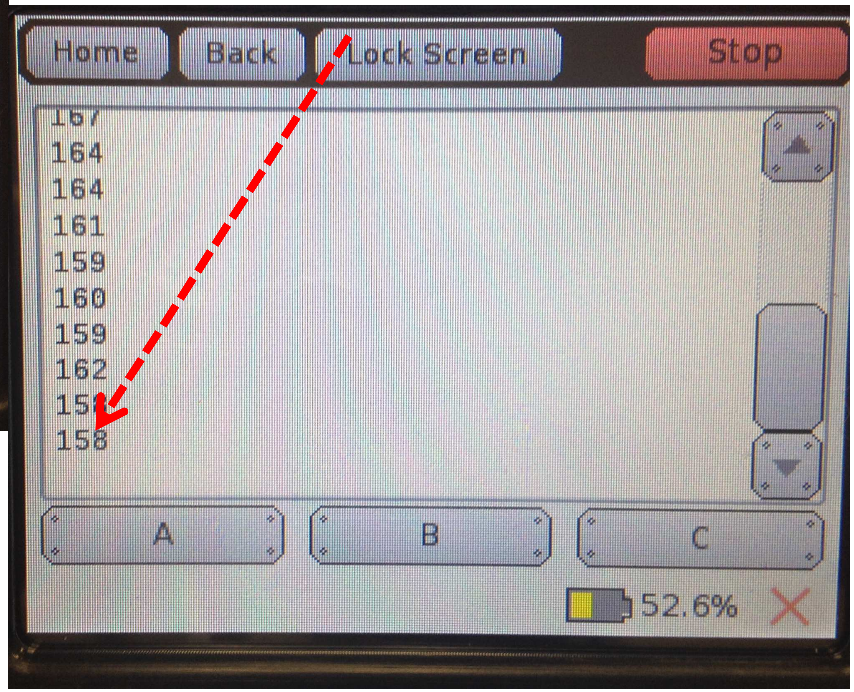
Reading the ET Sensor Values

While running the program hold an object in front of the sensor at different distances to read the corresponding value



ET ~1 inch from Link
Value is 387

ET ~12 inchs from Link
Value is 158



KISS INSTITUTE FOR PRACTICAL ROBOTICS

ET Sensor Activity
Using `while` and `if`

Now that you have some values to work with, write a program for your robot that uses the ET sensor to maintain the same distance from an object

- Too close- backup
- Too far away- move forward
- Just right- stop

Psuedocode (Task Analysis)

```
//Announces program
//Checks the status of the a button
//Checks the value from the ET sensor
//Moves backwards if the value is > 525
//Move forward if the value is < 475
//Stops if value is >= 475 and <= 525
```

ET Activity Solution
Using while and if (x3)

TEACHER'S RESOURCE

```
// Created on Fri September 13 2013

int main()
{
   set_analog_pullup(7,0); // change port 7 to floating analog
   while(side_button()==0)//checks status of side button
   {
      if(analog10(7)>525)//read value in analog port 7
        {//move backwards
        motor(0,-100);
        motor(3,-100);
        }

      if (analog10(7)<475)//read value in analog port 7
        {//move forward
        motor(0, 100);
        motor(3, 100);
        }
      if(analog10(7)>=475 && analog10(4)< 525)  ad value in analog port 7
        { //stop
        ao();
        }
   }
}
```

Remember you have to set the port the ET is using to floating or it won't work

Notice the use of the side button for the while loop. This lets the program run until the button is triggered

Students can use the sensor screen to read the values the ET sensor returns at different distances so they can figure out what value goes with each particular distance.
* Remember the minimum distance is ~5cm (mount the ET sensor at least 5cm back from the front of your robot)

KISS INSTITUTE FOR PRACTICAL ROBOTICS

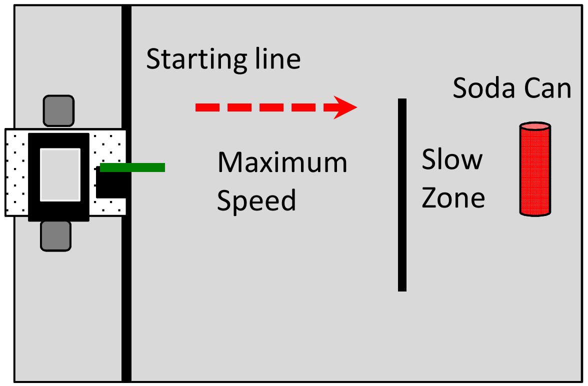
STUDENT ACTIVITY

TEACHER'S RESOURCE

Touch the Can with the ET

Robots must start on or behind the starting mark and move to the object at MAXIMUM SPEED with the goal of slowing down when they are a set distance from the can before they touch it

- This will teach students how to slow down when approaching an object
- 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 or have them average several runs or add the distances together for a grand total
- Change the "slow down" distance
  - o A short slow down distance will teach students about momentum



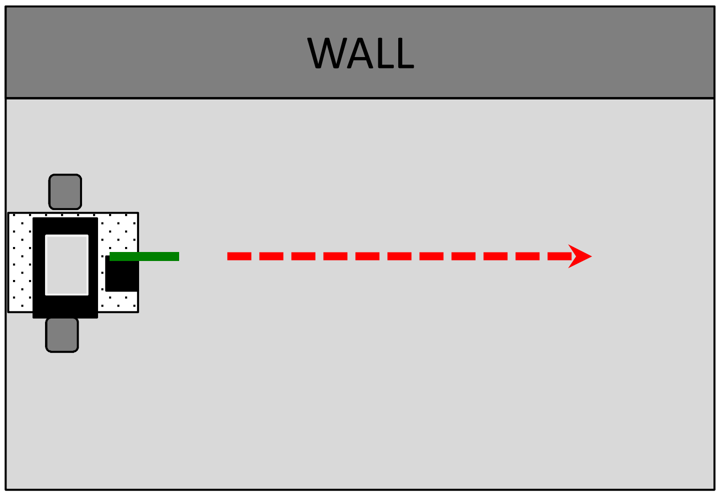
KISS INSTITUTE FOR PRACTICAL ROBOTICS

Follow the Wall

Using the ET sensor have your robot follow a wall maintaining a set distance from the wall

- The robot goes straight IF the value is….
- The robot turns toward the wall IF the value is….
- The robot turns away from the wall IF the value is….

You can use foam board or some other solid object for the side wall



Using the Camera to Track Objects

TEACHER'S RESOURCE

Goals

- To understand how to designate a channel and set the color model
- To help students understand how to use the camera with their robots
- To understand the logic of programming with the camera
- To write a program using the camera to follow an object

Preparation

- Have KISS IDE up and running
- Have a robot ready to go
- Students will need a camera mounted onto their robot
- You will need a colored object to track

Activity

Follow the slides and complete the activity

KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Using the Camera

You will need the USB Camera

- The USB camera plugs into one of the USB (type A) ports on the back of the KIPR Link
- Unplugging the camera while it is being accessed will usually freeze the system, requiring a reboot

USB
Ports

KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Setting the Color Tracking Channels

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

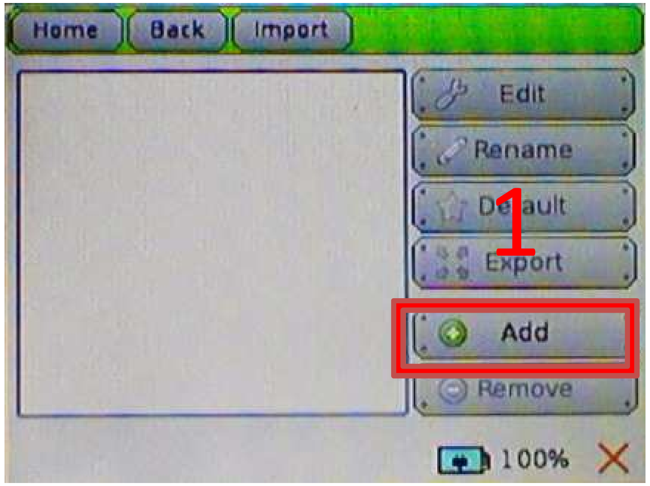
KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Setting the Color Tracking Channels

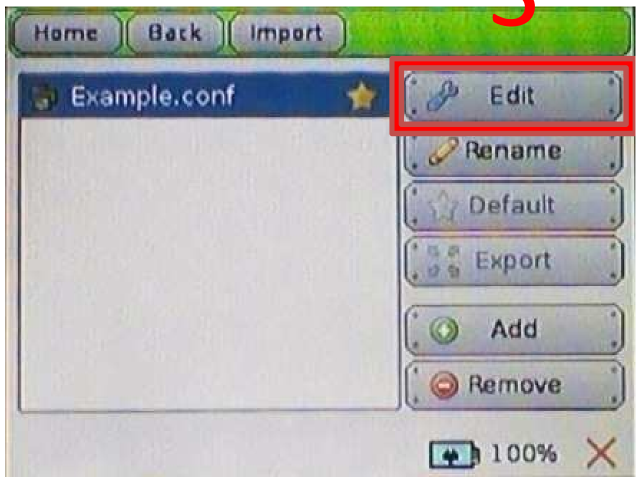
1. To specify a camera configuration select *"Add"*
2. Enter a configuration name such as "find_green" then press enter
3. Highlight the new configuration and press the *"Edit"* button

1



2

3

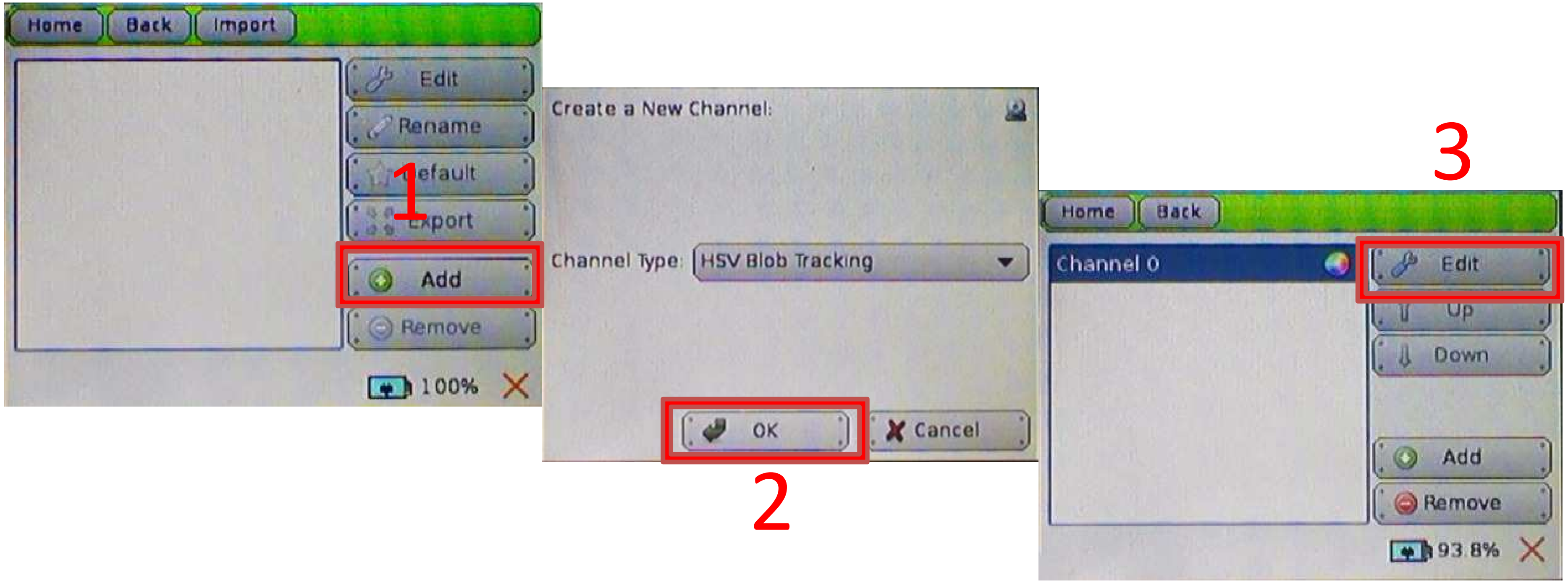


KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Setting the Color Tracking Channels

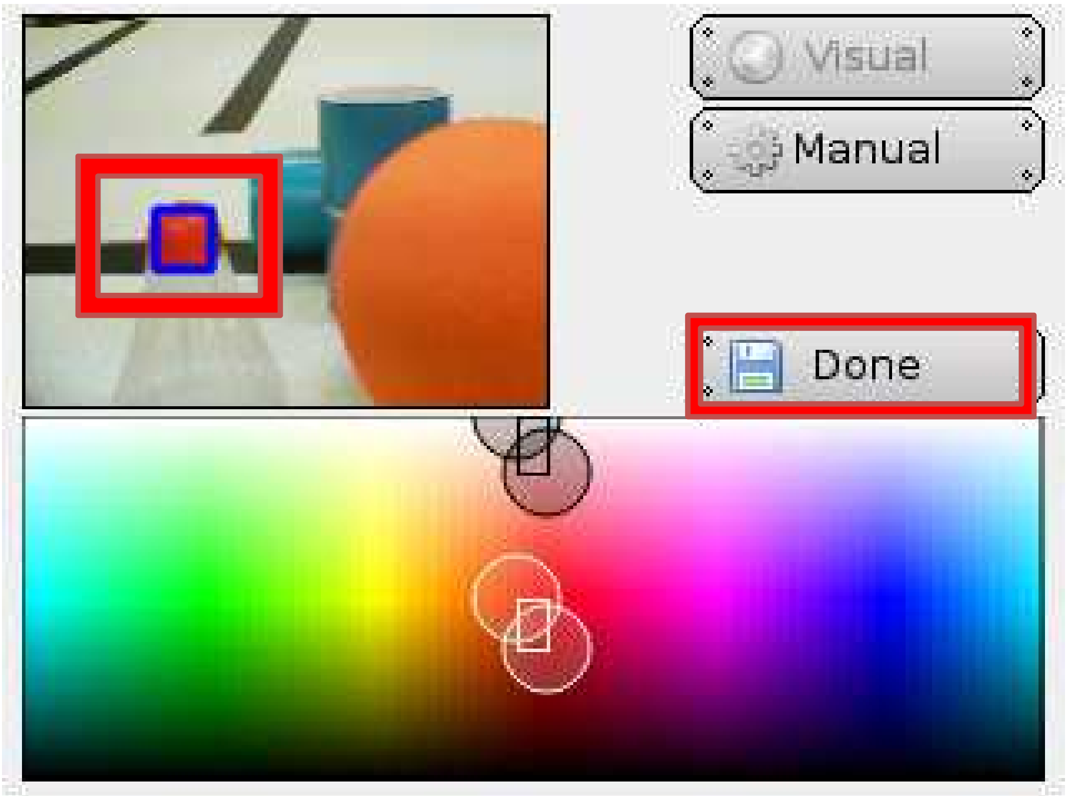
1. Press the *"Add"* button to add a channel to the configuration
2. Select *"HSV Blob Tracking"* then *"OK"* to make this track color
3. Highlight the channel and press the *"Edit"* button to edit settings
    - First channel is 0 by default you can add three more 0,1,2,3



KISS INSTITUTE FOR PRACTICAL ROBOTICS

Setting the Color Tracking Channels

1. Place the colored object you want to track in front of the camera and touch it on the touch screen
   - The program will put a bounding box (dark blue) around the selected object then hit *"Done"*



KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Setting the Color Tracking Channels

Verify the channel is working

1. From the main screen, select *"Motors and Servos"*
2. Select *"Camera"*
   - Objects specified in the configuration should have a bounding box

KISS INSTITUTE FOR PRACTICAL ROBOTICS

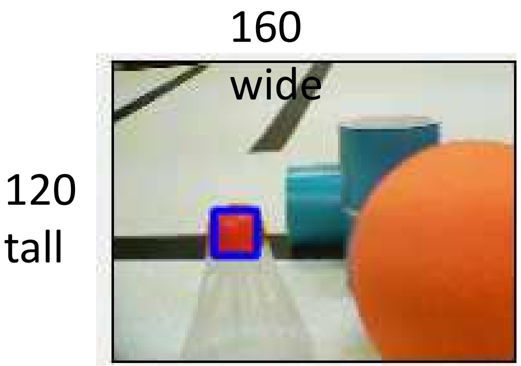
About Color Vision Tracking

For color vision tracking, images are processed by the KIPR Link to identify "blobs" matching the color specification you set in the channel configuration.

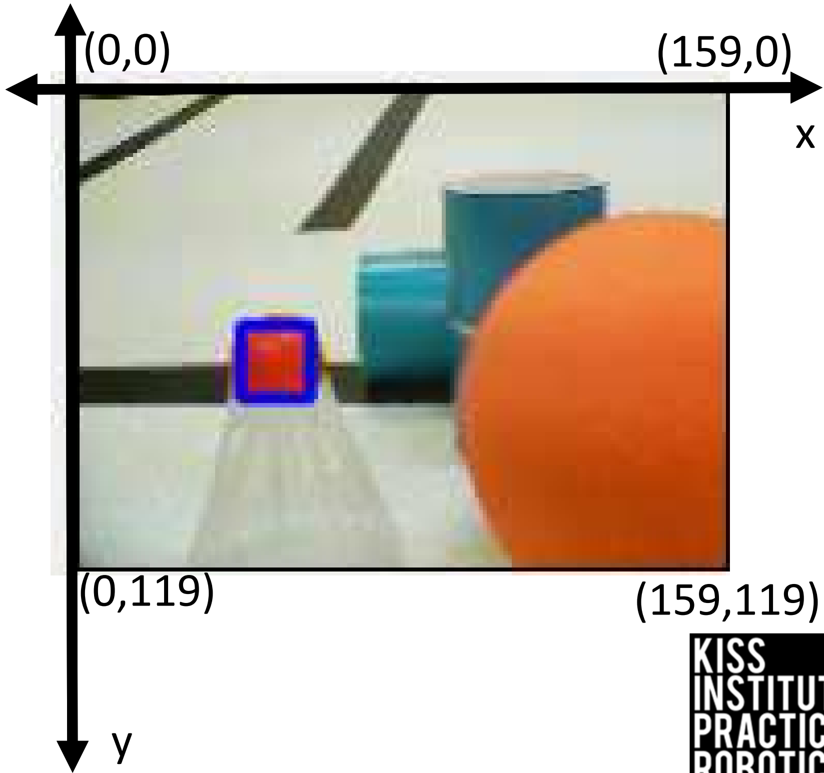
- A blob is a set of contiguous pixels in the image matching your channel color specification

The camera image size is in pixels 160 X 120

- Remember we start counting at 0



Imagine an x and y coordinate system. We start in the upper left corner and read the image like a person reads text. Top to bottom and left to right.



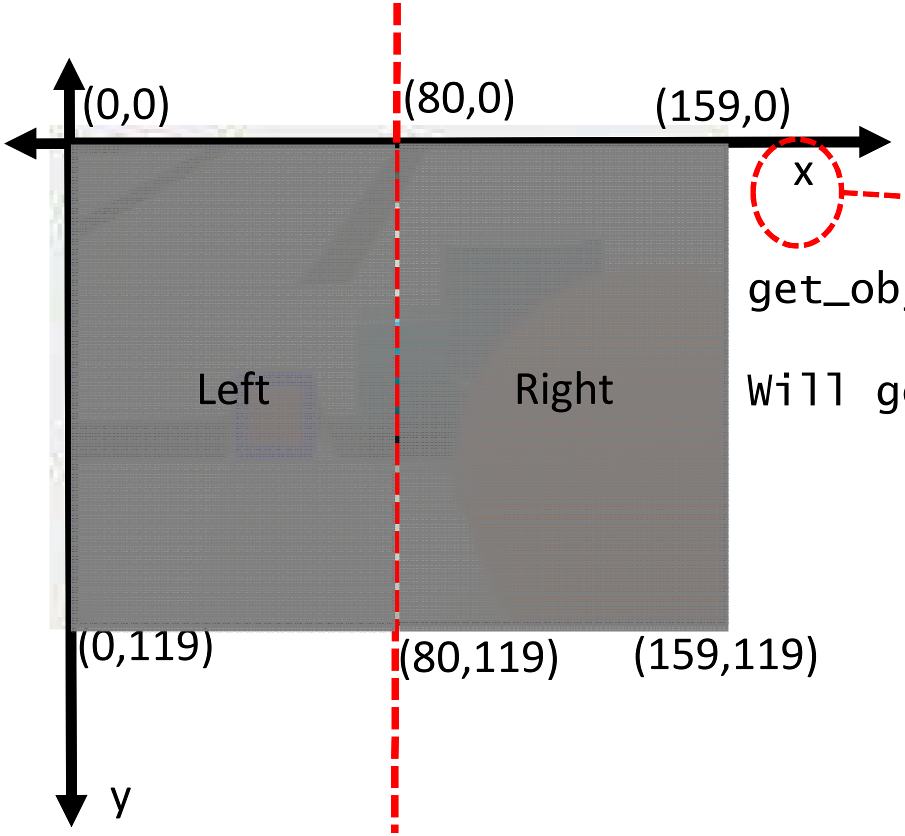
KISS INSTITUTE FOR PRACTICAL ROBOTICS

About Color Vision Tracking

You can use the position of the object in relation to the center of the image to tell if it is to the left or right

- And if you know that the image is 160 wide, then the center is 80
  - o Between 0 and 79 is to the left
  - o Between 81 to 160 is to the right
  - o 80 is straight ahead

*You can also use the position of the object in relation to the y axis to tell how far away it is




```
get_object_center (channel, #).x;
```

```
Will generate a number between 0 and 159
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Camera Functions

```
camera_open(LOW_RES); //sets resolution
```

- Choices include LOW, MED and HIGH. <u>LOW is best for most applications</u>

```
camera_close(); //closes camera

camera_update (); //retrieves current image

get_object_count (); //retrieves number of objects specified by the channel settings with 0 being the largest object specified in the area

get_object_center (channel, #); //retrieves x and y coordinate of the object
    get_object_center (channel, #).x;
    get_object_center (channel, #).y;
```

Camera Activity Using `while` and `if` and `else`

STUDENT ACTIVITY

Psuedocode (Task Analysis)

1. //Prints Move towards object and Press B button when ready
2. //Checks the status of the b button
3. //Checks the status of the side button
4. //Updates camera image
5. //Turns left toward object
6. //Turns right toward object
7. //Stops if no object in sight
8. //Stops when side button is pressed
9. //Prints done

*This is the same type program as the line follow activity, but instead of the reflectance sensor it is using the camera. Because it knows that 80 is the center of the image anything <80 is to the left, so turn left, anything >= 80 is to the right, so turn right, if it doesn't see anything then it stops.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Solution

TEACHER'S RESOURCE

```
// Created on Wed  Oct 9 2013

/* Move the robot towards the largest object on channel 0.
Robots stops if no object is detected*/
int main()
{

   camera_open(LOW_RES);
   printf("Move towards object\n");
   printf("Press B button when ready\n\nPress side button to stop\n");
   while(b_button()==0)
   {
   ao();
   } // wait for button press
   while(side_button()==0) // while side button is not pressed
   {
      camera_update(); //get a new image to analyze
      if(get_object_count(0)>0)
         {
            if(get_object_center(0,0).x < 80)
               {// if object is on left...
               motor(0,100);
               motor(3,10); // turn left
               }
            if(get_object_center(0,0).x >= 80)
               {// if object is on right...
               motor(0,10);
               motor(3,100); //turn right
               }
         }
      else
         {
         ao(); //no object detected
         }
   }
   ao(); // stop because button pressed
   printf("done\n");
   return 0;
}
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Camera Activity Solution

TEACHER'S RESOURCE

```
// Created on Wed  Oct 9 2013

/* Move the robot towards the largest object on channel 0.
Robots stops if no object is detected*/
int main()
{

   camera_open(LOW_RES);
   printf("Move towards object\n");
   printf("Press B button when ready\n\nPress side button to stop\n");
   while(b_button()==0)
   {
   ao();
   } // wait for button press
   while(side_button()==0) // while side button is not pressed
   {
      camera_update(); //get a new image to analyze
      if(get_object_count(0)>0)
         {
            if(get_object_center(0,0).x < 80)
               {// if object is on left...
               motor(0,100);
               motor(3,10); // turn left
               }
            if(get_object_center(0,0).x >= 80)
               {// if object is on right...
               motor(0,10);
               motor(3,100); //turn right
               }
         }
      else
         {
         ao(); //no object detected
         }
   }
   ao(); // stop because button pressed
   printf("done\n");
   return 0;
}
```

This program uses 2 `while` loops

- Remember no semicolon after the `while` statement
- The first `while` is simple
- The second `while` contains 3 `if` statements and 1 `else` statement
  - `If` this is true do this `else` (if it is not true) do this

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Camera Activity Solution Using `while`, `if` and `else`

TEACHER'S RESOURCE

```c
// Created on Wed  Oct 9 2013

/* Move the robot towards the largest object on channel 0.
Robots stops if no object is detected*/
int main()
{

    camera_open(LOW_RES);
    printf("Move towards object\n");
    printf("Press B button when ready\n\nPress side button to stop\n");
    while(b_button()==0)
    {
    ao();
    } // wait for button press
    while(side_button()==0)// while side button is not pressed
    {
        camera_update(); //get a new image to analyze
        if(get_object_count(0)>0)
          {
              if(get_object_center(0,0).x < 80)
                {// if object is on left...
                motor(0,100);
                motor(3,10); // turn left
                }
              if(get_object_center(0,0).x >= 80)
                {// if object is on right...
                motor(0,10);
                motor(3,100); //turn right
                }
          }
        else
          {
          ao(); //no object detected
          }
    }
    ao(); // stop because button pressed
    printf("done\n");
    return 0;
}
```

- Sets camera to LOW RESOLUTION
- First `while` loop waits for b button push to start
- Notice the use of the side button for the `while` loop. This lets the program run until the button is pushed
- Updates to most recent camera image
- Checks channel 0 for the largest object = 0 and if it is >0 (in other words, it sees something) then it…
- Checks to see where the object is in relation to the x axis. To the left then turn left, to the right then turn right (80 is the midpoint of the 160 pixel image
- Checks channel 0 for the largest object = 0 and if it is = 0 (in other words it doesn't see anything) then it turns the motors off

*Make sure the students line up the curly braces otherwise it is easy to get lost

INSTITUTE FOR PRACTICAL ROBOTICS

TEACHER'S RESOURCE

Improved Chase the Object

```
// Created on Wed  Oct 9 2013

/* Move the robot towards the largest object on channel 0.
Robots stops if no object is detected*/
int main()
{

  camera_open(LOW_RES);
  printf("Move towards object\n");
  printf("Press B button when ready\n\nPress side button to stop\n");
  while(b_button()==0)
  {
  ao();
  } // wait for button press
  while(side_button()==0) // while side button is not pressed
  {
    camera_update(); //get a new image to analyze
    if(get_object_count(0)>0)
      {
        if(get_object_center(0,0).x < 65)
          {// if object is on left...
          motor(0,100);
          motor(3,10); // turn left
          }
        if(get_object_center(0,0).x > 95)
          {// if object is on right...
          motor(0,10);
          motor(3,100); //turn right
          }
        if(get_object_center(0,0).x >= 65 && get_object_center(0,0).x <= 95 )
          {// if object is in the middle.
          motor(0,100);
          motor(3,100); //straight ahead
          }
      }
    else
      {
      ao(); //no object detected
      }
  }
  ao(); // stop because button pressed
  printf("done\n");
  return 0;
}
```

You can add another if statement to have the robot go straight if the object is near the middle

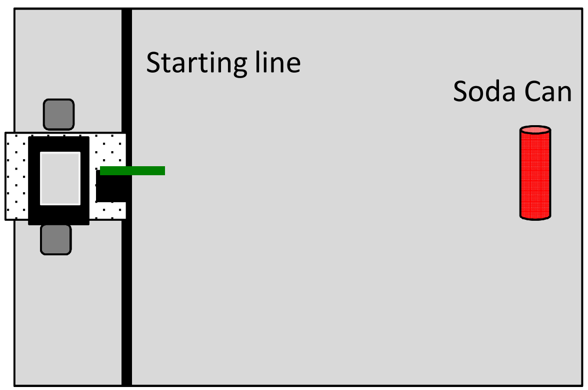
KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

TEACHER'S RESOURCE

Find the Can with the Camera

1. Robots must start on or behind the starting mark then using the camera, find the can and move to it
   - Move the can to random locations



KISS INSTITUTE FOR PRACTICAL ROBOTICS

Get to know your Create©

TEACHER'S RESOURCE

***The *Create* platform comes with the intermediate and advanced starter kits as well as with a Botball kit**

Create

- Is an educational platform from *iRobot* based on the *Roomba* vacuum
- *iRobot* partners with the KISS Institute for Practical Robotics to provide the platform for student use in the Botball Educational Robotics Program
- The platform has built in sensors that can be accessed and read with The KIPR Link robot controller
- More information can be found in the KIPR LINK Manual

You have the KIPR LINK Manual on the flash drive provided to you

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Charging the KIPR Link Controller

TEACHER'S RESOURCE

- For charging the KIPR Link, **use only the power supply which came with your Link**
  - **Damage to the Link from using the wrong charger is easily detected and will void your warranty!**
- The KIPR Link power pack is a lithium polymer battery so the rules for charging a lithium battery for any electronic device apply
  - Only an adult should charge the unit
  - You should NOT leave the unit unattended while charging
  - Charge away from any flammable materials and in a cool, open area

KISS INSTITUTE FOR PRACTICAL ROBOTICS

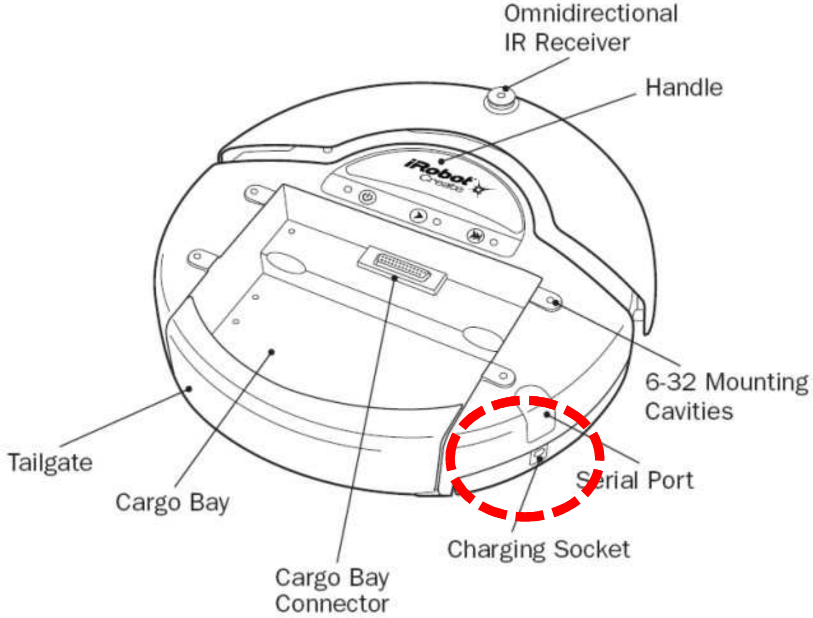
Charging the *Create*

- For charging the *Create*, **use only the power supply which came with your *Create***
  - **Damage to the *Create* from using the wrong charger is easily detected and will void your warranty!**
- The *Create* power pack is a nickel metal hydride battery so the rules for charging a battery for any electronic device apply
  - Only an adult should charge the unit
  - You should NOT leave the unit unattended while charging
  - Charge away from any flammable materials and in a cool, open area

Plugging charger into *Create*

Use only the Create charger provided with your kit

The charger plugs into the power socket



Learning about the *Create*

TEACHER'S RESOURCE

Goals

- To be able to insert the battery into the *Create* properly
- To be able to identify the serial cable used to connect the Link to the *Create*
- To understand how to place the Link inside the *Create* Cargo bay
- To understand the proper charging procedure for the *Create* (only an adult, only under supervision at all times, not around water or flammable materials)

Preparation

Have a *Create* and LINK controller available for students to examine along with a projection of the resource slide with pictures of the controller OR give students a printed sheet of the resource slide

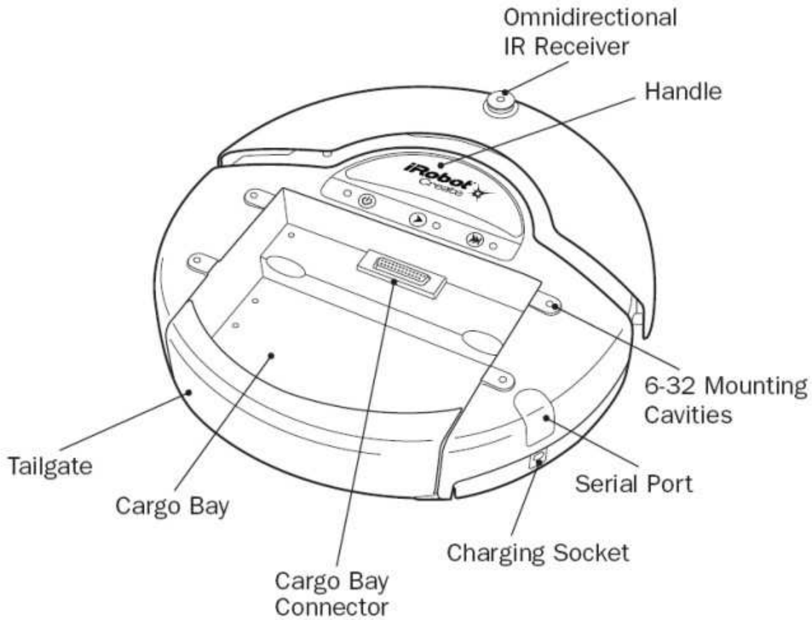
You will need a KIPR Link-*Create* cable

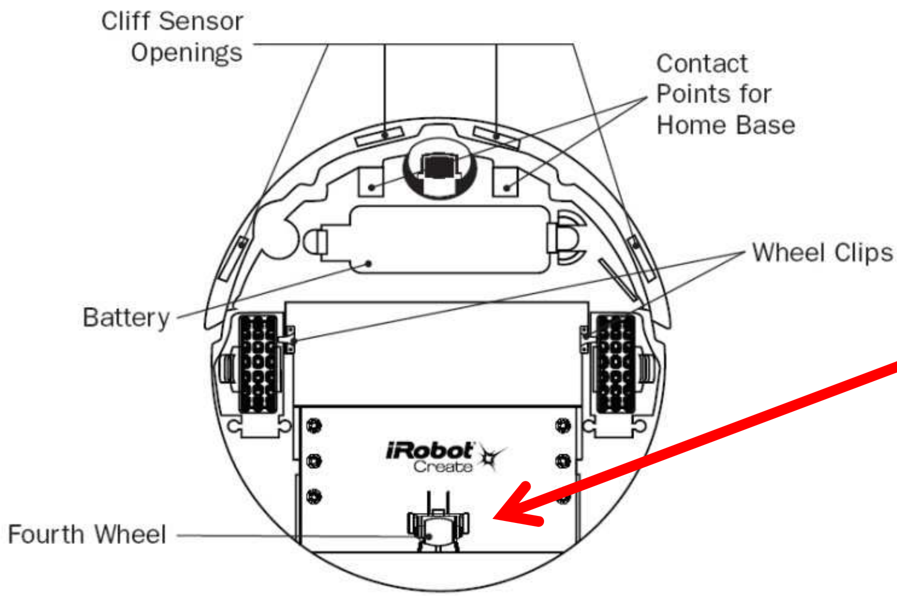
Resources

The KIPR Link Manual (on your flash drive)

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Learning about the *Create*



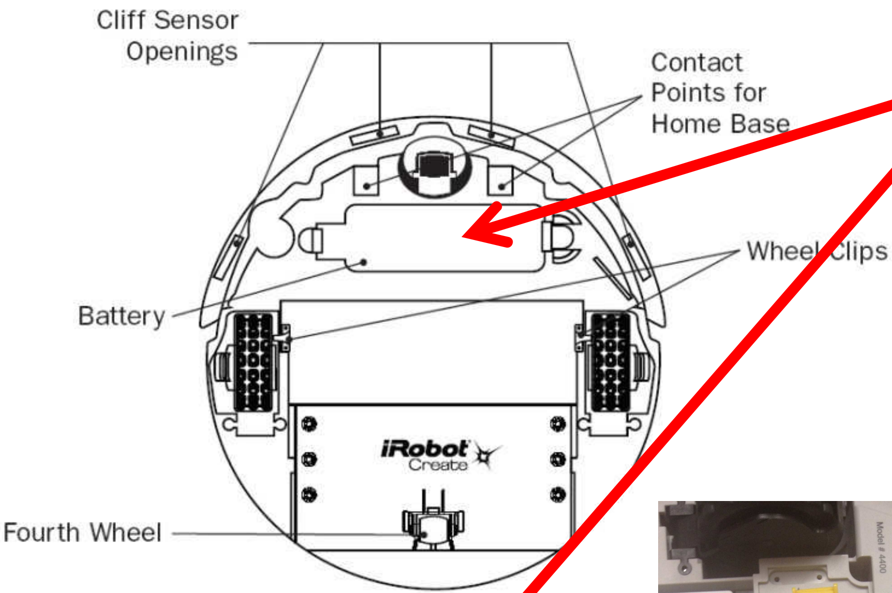


The LINK sits in the cargo bay of the *Create*

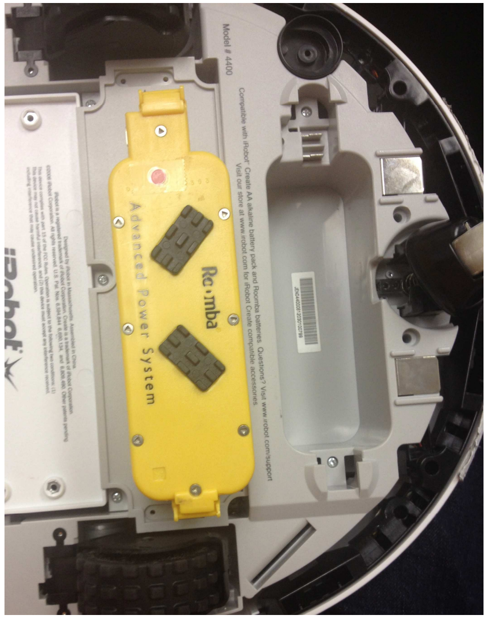
- You will need to attach the fourth wheel (it simply snaps into place)

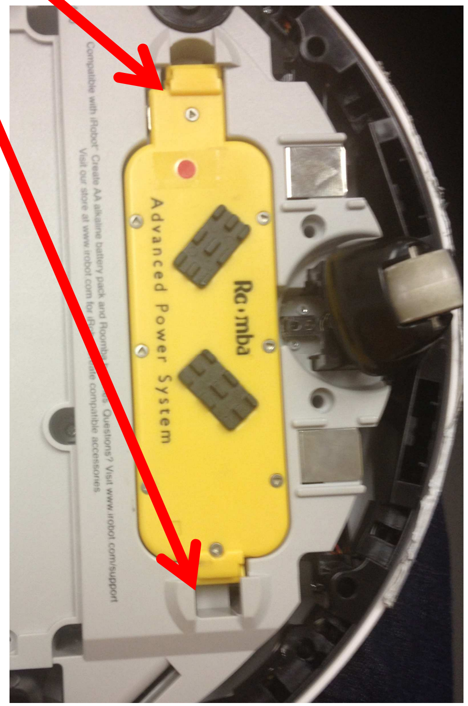
Installing the battery in the *Create*

TEACHER'S RESOURCE



The yellow battery snaps into place on the bottom of the *Create* (make sure both sides snap into place)

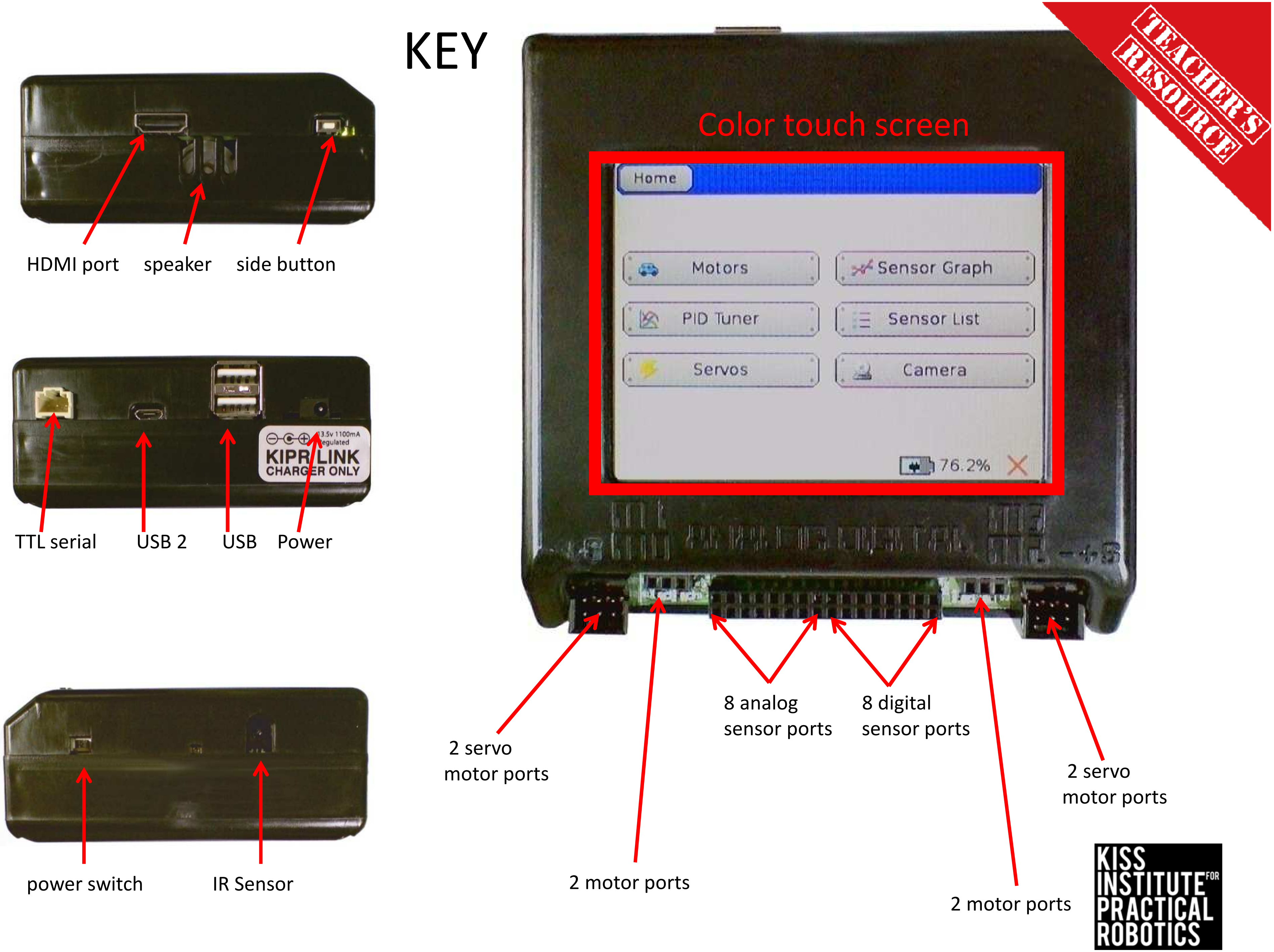




Battery has a long and a short tab that matches the slot in the *Create*

KISS INSTITUTE FOR PRACTICAL ROBOTICS

KEY

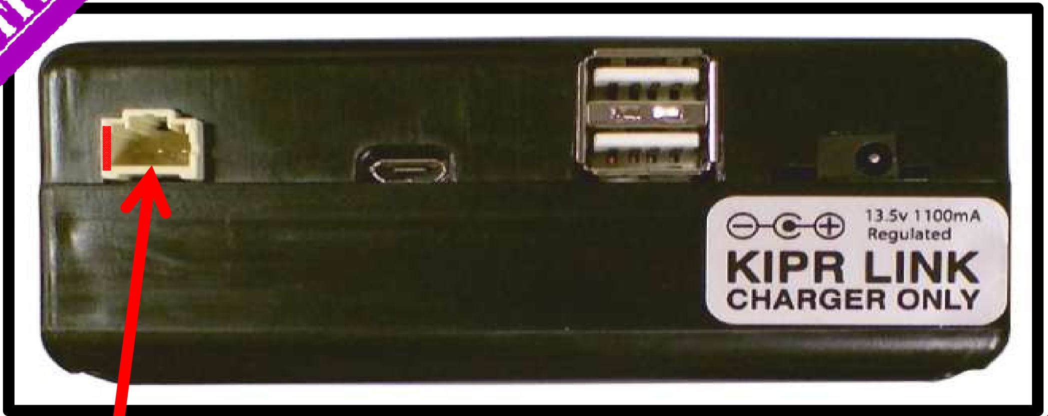


TEACHER'S RESOURCE
HDMI port
speaker
side button
TTL serial
USB 2
USB
Power
KIPR LINK
CHARGER ONLY
power switch
IR Sensor
Color touch screen
Home
Motors
Sensor Graph
PID Tuner
Sensor List
Servos
Camera
76.2%
2 servo motor ports
2 motor ports
8 analog sensor ports
8 digital sensor ports
2 servo motor ports
2 motor ports
KISS INSTITUTE FOR PRACTICAL ROBOTICS

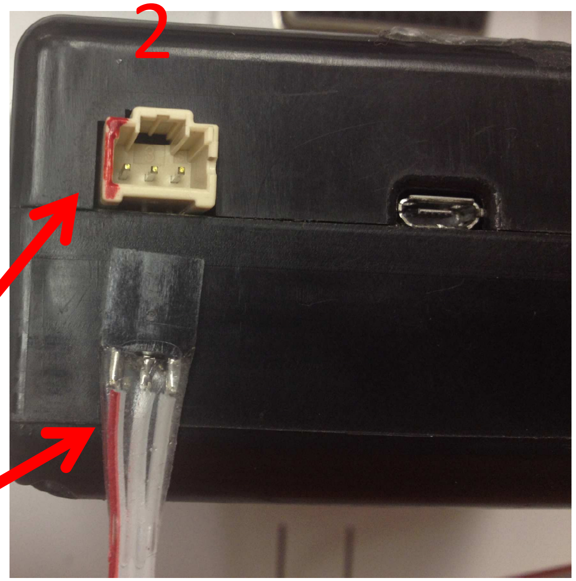
STUDENT ACTIVITY

Plugging serial cable into Link

1



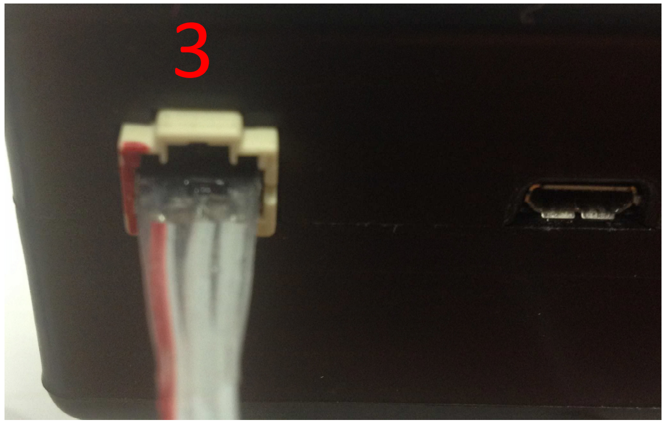
2

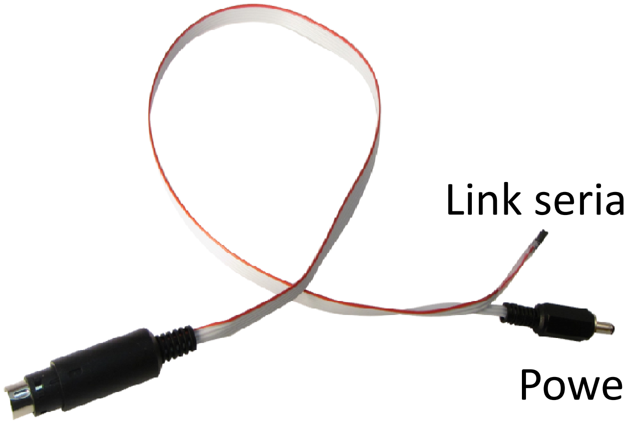


We use the serial cable to plug into the Link TTL Serial plug

NOTICE the red mark on the plug (left side) this corresponds to the red wire in the serial cable

3





Link serial plug

Power connector

Create connector

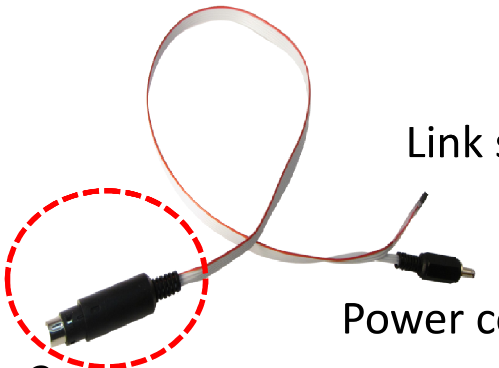
Correctly plugged in

KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Plugging serial cable into *Create*

We use the Create connector (round) end



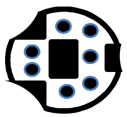
Link serial plug

Power connector

Create connector

Receptacle may have a cover that you can pop off to access

The plug is keyed, make sure you line it up correctly before plugging it in



KISS INSTITUTE FOR PRACTICAL ROBOTICS

Know your *Create*

Have a show and tell describing, explaining and pointing out:

- The serial cable
- The serial port on the Create
- The serial port on the Link
- The correct orientation for the serial cable to be plugged into the Link

Moving the *Create*

Goals

- To reinforce the concept of a function
- To learn and use the functions for connecting to and moving the Create

Preparation

- You will need a charged Create and Link + the serial cable to connect them
- You will need computers with the KISS IDE
- You will need the USB download cable

Activity

Follow the slides to make the robot move

STUDENT ACTIVITY

Activity 3

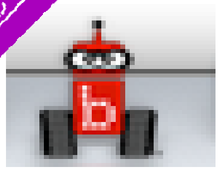
Lets make a robot move!

Use the Create with a Link controller in the cargo bay connected with a serial cable

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Launch the KISS IDE

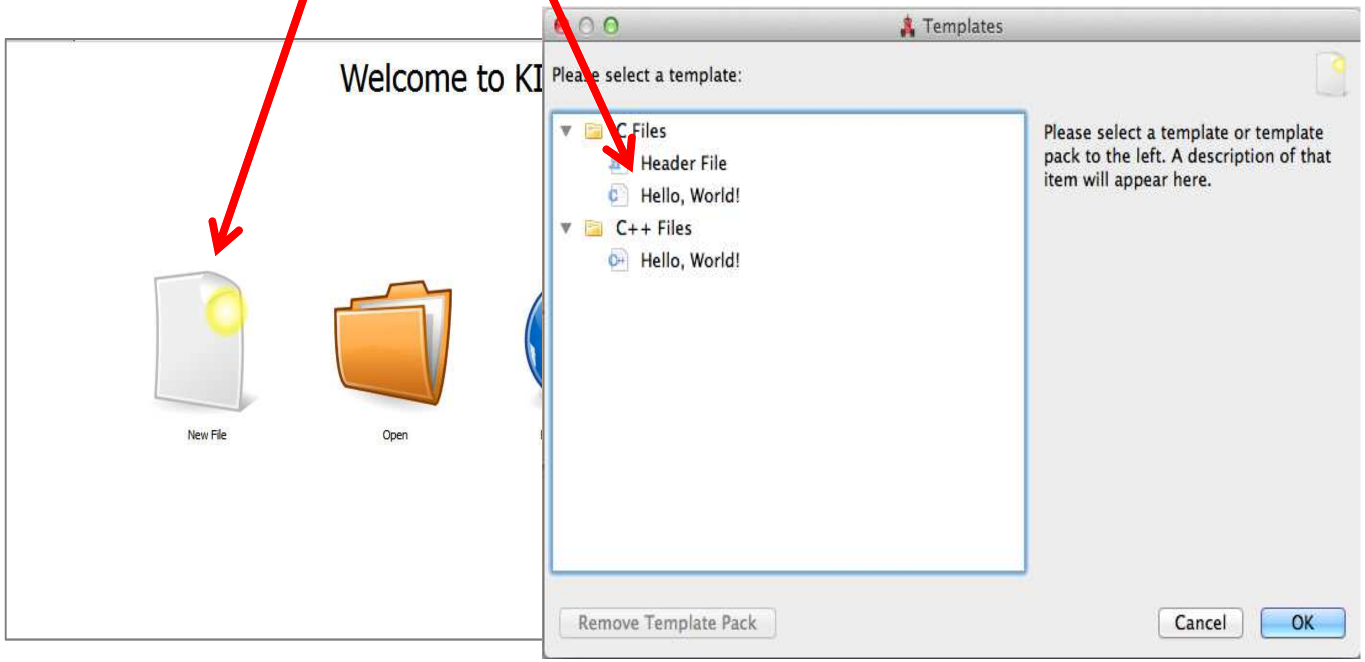
STUDENT ACTIVITY



KISS IDE icon

- Start the KISS IDE by clicking on its icon to get the welcome screen
- Click on the "*New File*" icon and

  and choose the C, "*Hello, World!*" template

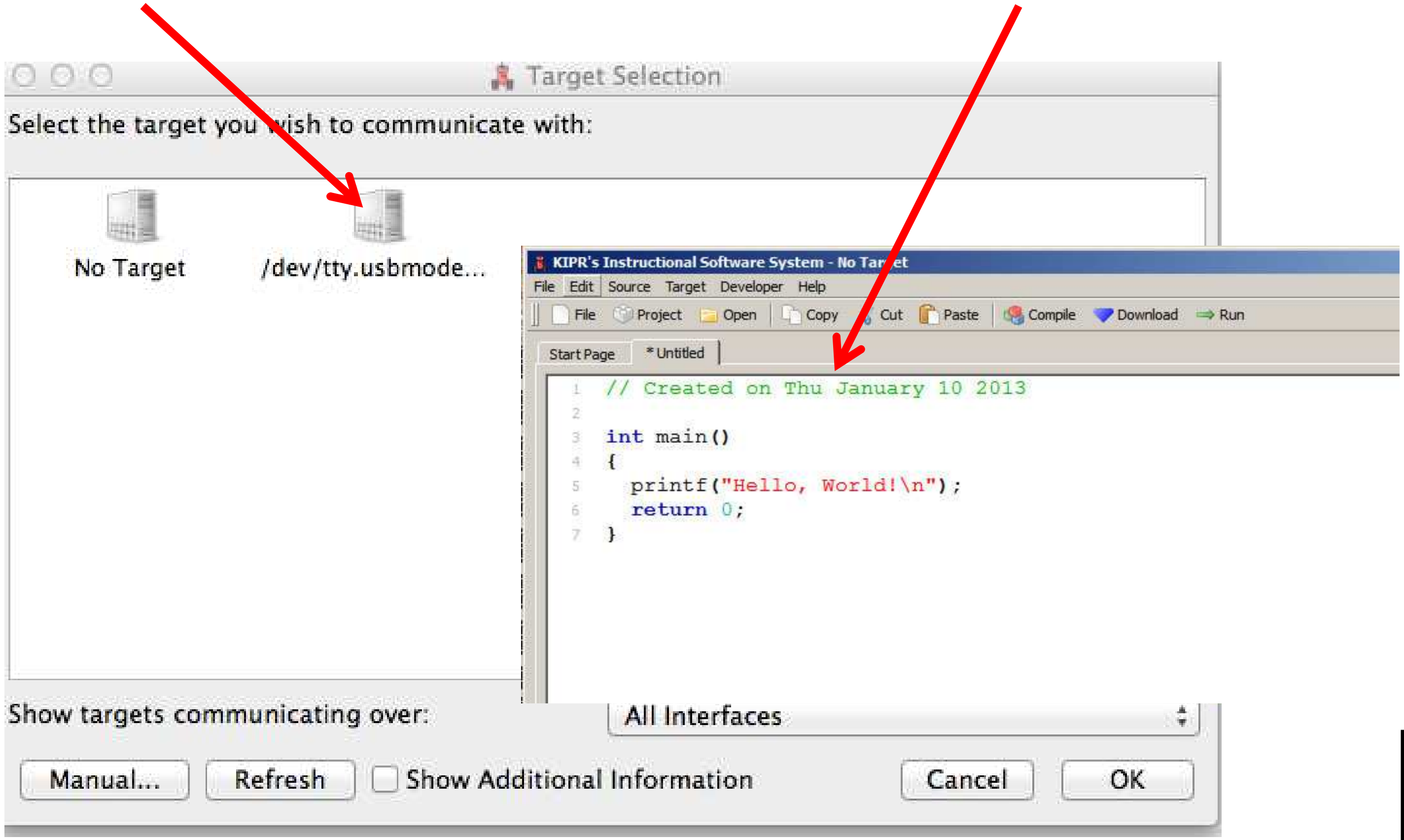


KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Select Target

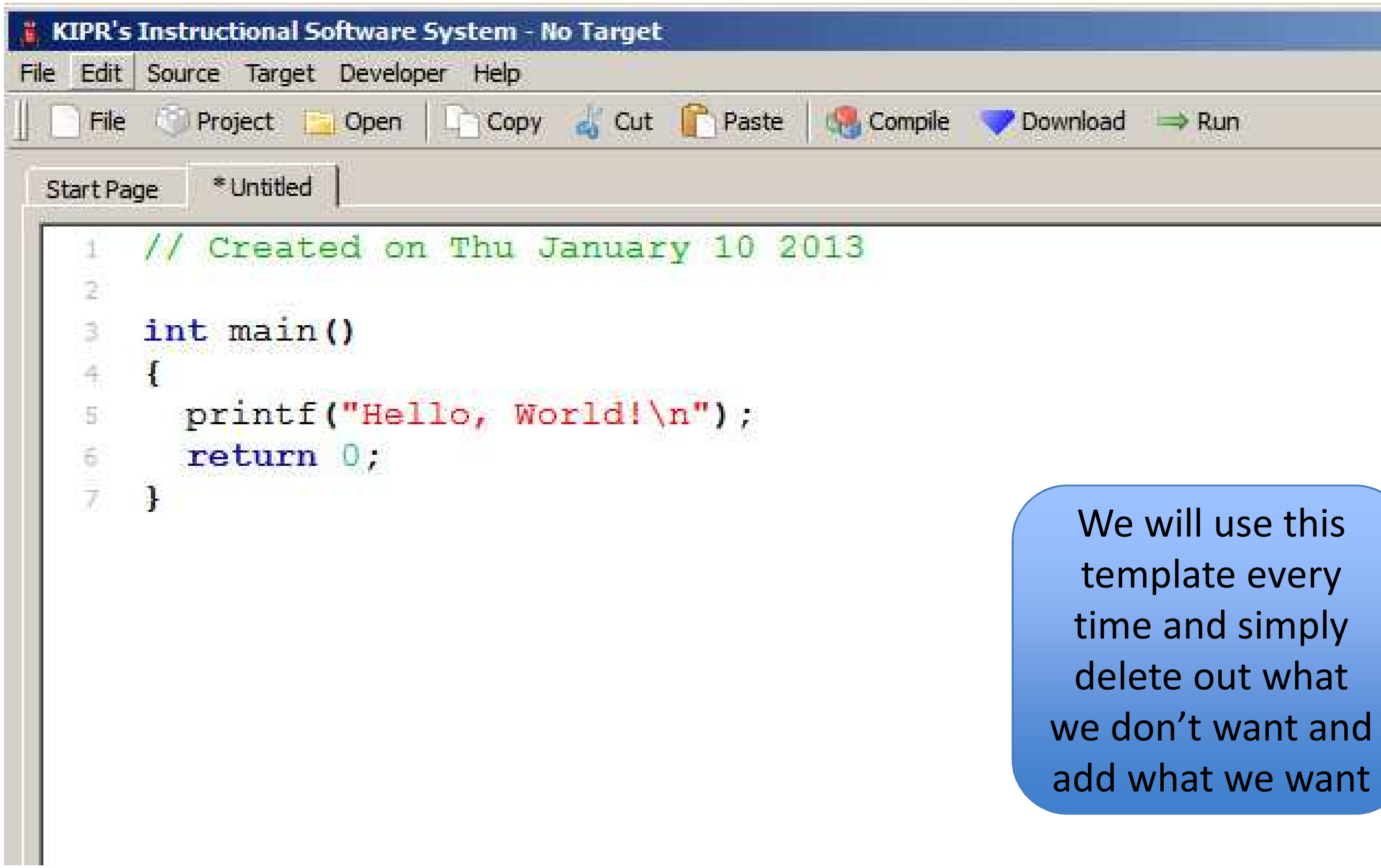
- A Target Selection window will appear
- Pick *"USB Target"* and the C program template will come up



KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

The C Template: Hello, World!



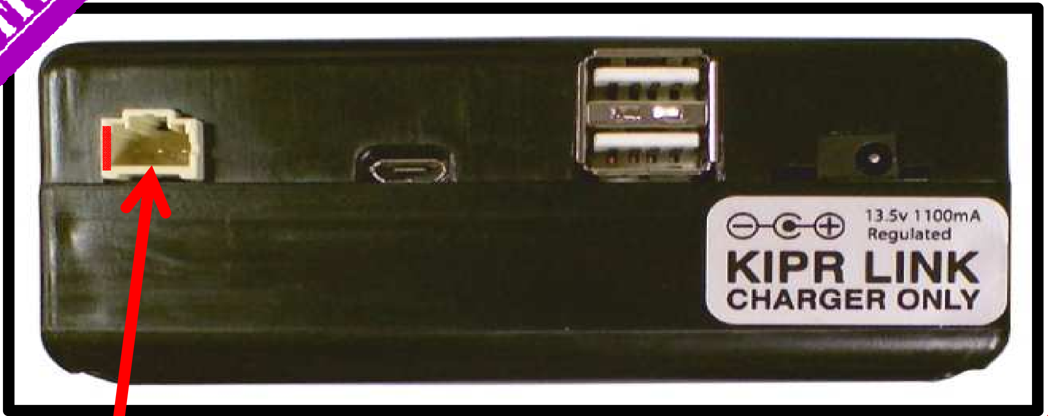
We will use this template every time and simply delete out what we don't want and add what we want

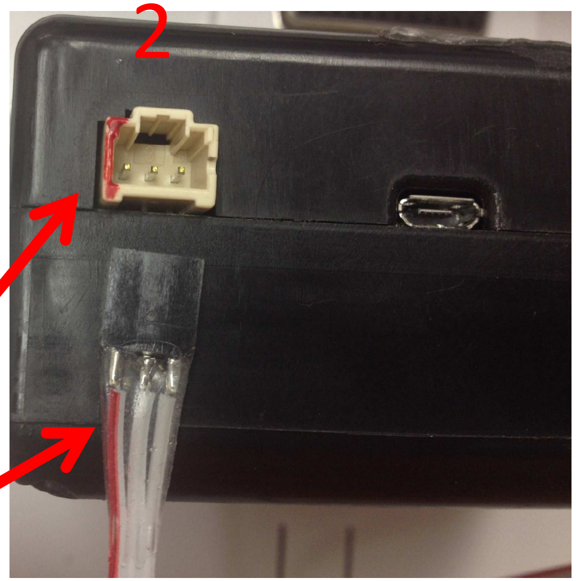
KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Plugging serial cable into Link

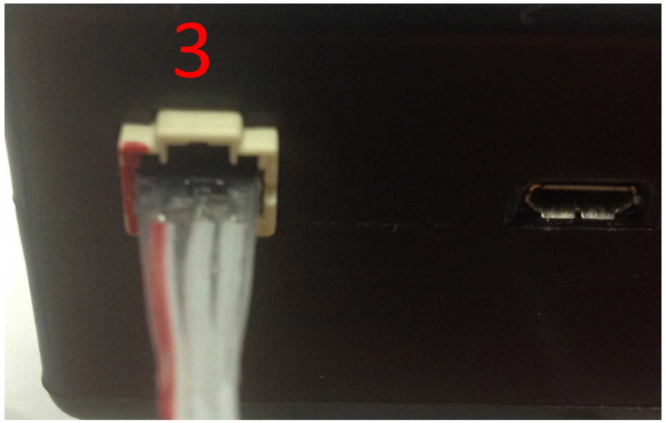
1



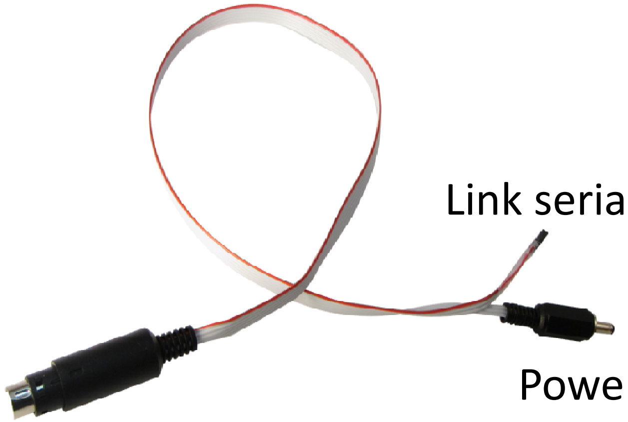


We use the serial cable to plug into the Link TTL Serial plug

NOTICE the red mark on the plug (left side) this corresponds to the red wire in the serial cable



Correctly plugged in



Link serial plug

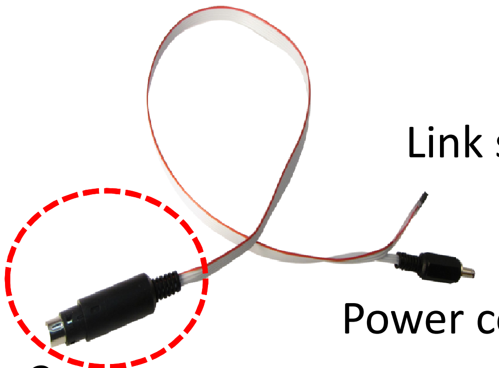
Power connector

Create connector

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Plugging serial cable into *Create*

We use the Create connector (round) end



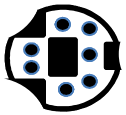
Link serial plug

Power connector

Create connector

Receptacle may have a cover that you can pop off to access

The plug is keyed, make sure you line it up correctly before plugging it in

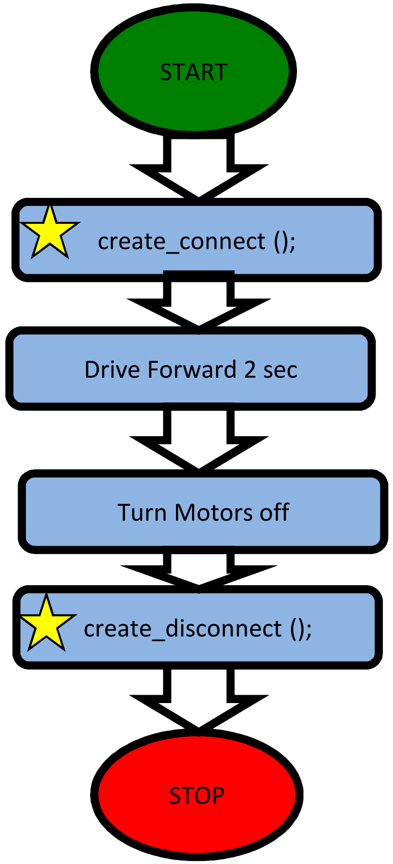


STUDENT ACTIVITY

Functions to Connect & Disconnect

We must tell the controller to use the serial cable to send commands to the *Create*

*The *Create* must be turned on for this to work




```
create_connect();  //tells the Link to use the serial connection to the Create
create_disconnect();  //tells the Link to QUIT using the serial connection to the Create
```

<u>ALL</u> programs used with the *Create* <u>MUST</u> start with `create_connect();` and end with `create_disconnect();`

KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Functions to Move and Stop

Create commands run UNTIL a different motor command is received

```
create_drive_direct (left_speed,right_speed);
```

Left motor/wheel — Speed in mm/second — right motor/wheel — Speed in mm/second

```
create_drive_direct (100,100); //moves forward at 100mm/sec

create_drive_direct (100,200); //create will turn left

create_drive_direct (200,100); //create will turn right
```

***WARNING maximum speed for the Create motors is 1000mm/second = 1 meter (~3feet)/second. It will jump off a table in a second! Use something like 200 for the speed (moderate speed) until teams get the hang of this**

```
create_stop (); // stops the motors
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Explain using comments

STUDENT ACTIVITY

You can use a flow chart and then translate that into comments.

Using //comments as pseudocode is a great way to start.

If you forget which functions to use, look at your cheat sheet.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

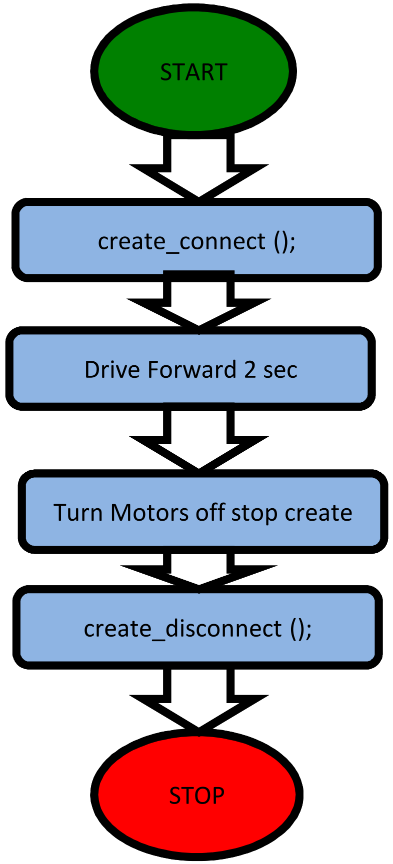
Lets make a robot move!

STUDENT ACTIVITY

Write a program for your robot to move forward for 2 seconds

Psuedocode (Task Analysis)

```
// 1. connect to create
// 2. Drive forward
// 2. Pause program for 2 seconds
to give the robot time to move
// 3. stop the motors/create
// 4. disconnect from create
```



KISS INSTITUTE FOR PRACTICAL ROBOTICS

Activity 3 Solution

TEACHER'S RESOURCE

```
// Created on Thu October 10 2013

int main()
{
    create_connect ();
    create_drive_direct (100, 100);
    msleep (2000);
    create_stop();
    create_disconnect ();
    return 0;
}
```

Notice the create connect first thing right after the int main

Moves both motors forward at 100mm/second (should go straight)

conds giving the robot

Stops the motors

Disconnects create from Link

KISS INSTITUTE FOR PRACTICAL ROBOTICS

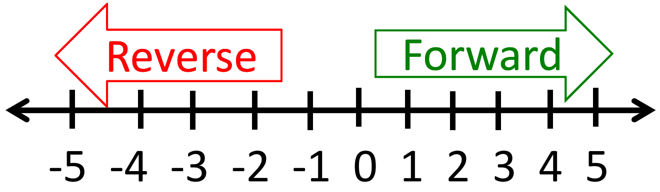
Create Driving Hints

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

The Create is very fast, at 1000mm/sec

It can get away from students quickly

- The Create is heavy and can produce lots of inertia/momentum (keep this in mind while trying to get precise distances)



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.

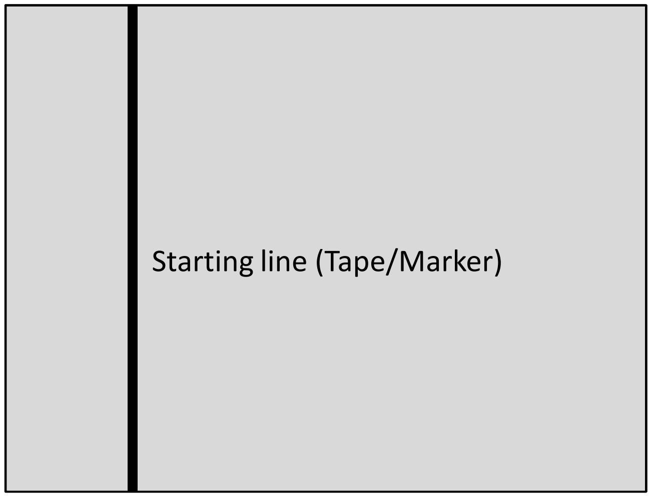
Making Turns

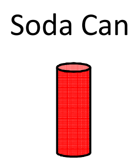
- Have one wheel go faster or slower than the other
- Have one wheel move while the other ones is stopped (friction is less of a factor when both wheels are moving)
- Have one wheel move forward while the other is moving backwards

LET'S MOVE! Materials/Supplies

TEACHER'S RESOURCE

1. You need a large surface to run the robot on
   - Use the floor, a piece of white or light colored foam or poster board or a vinyl or paper mat as a robot testing track
     - You need an area marked as the starting line (a piece of black tape works well or you can mark it with a black marker)
2. You need an object to navigate to
   - Can of soda, foam block, whiteboard eraser, etc. will work
3. A measuring device and a timer will be useful





KISS INSTITUTE FOR PRACTICAL ROBOTICS

LET’S MOVE!

Activity/mini contests

Using the simple motor function `motor();` and `msleep();` you can have the students work on fun challenges.

These activities can all be completed using hard coding (“dead reckoning”) and simple motor control functions without the use of any sensors. This is a good place to start and will teach the students how hard it is to be consistent using dead reckoning.

- This is a good time to bring up controlling variables when they set up their robot- is it the same every time? How could you make it the same (using a jig or ruler to control how they set it at the starting line)

Once they have the skills down of forward, backwards, turn and stop, we can move on and start adding sensors and decision making into the programs.

STUDENT ACTIVITY

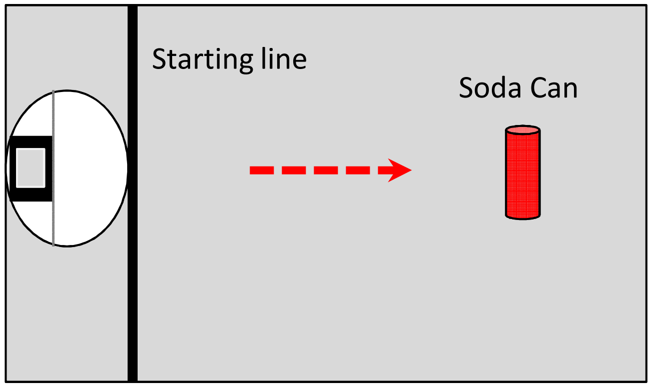
TEACHER'S RESOURCE

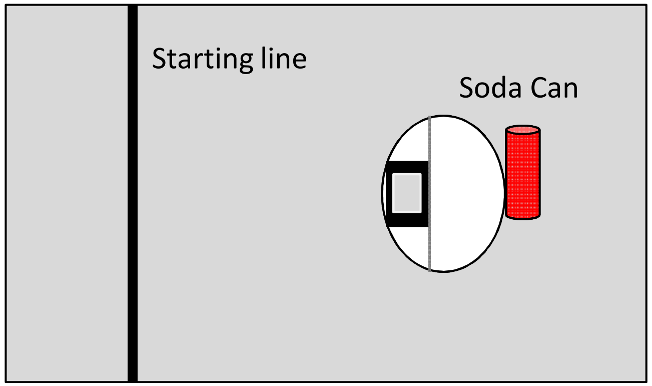
Touch the Can

Robots must start on or behind the starting mark and move to the object with the goal of touching the object in the shortest amount of time

Extensions

- Move the can to various distances
- Make the object smaller and harder to navigate to
- Math- have them measure the distance to the object and time the robot and then calculate rate/speed
    - Speed = Distance/Time





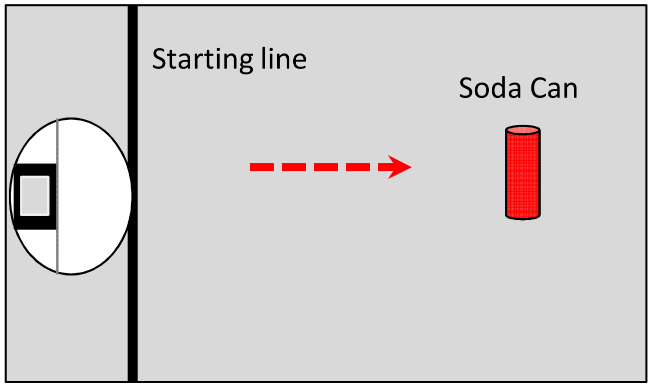
KISS INSTITUTE FOR PRACTICAL ROBOTICS

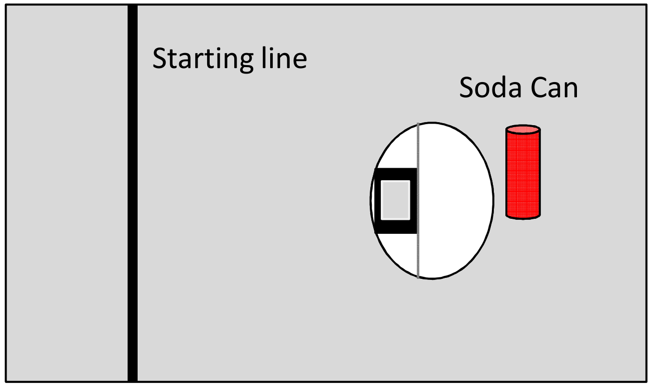
STUDENT ACTIVITY

TEACHER'S RESOURCE

Closest to the Can

1. Robots must start on or behind the starting mark and move to 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 or have them average several runs or add the distances together for a grand total
2. Move the can to various distances and locations





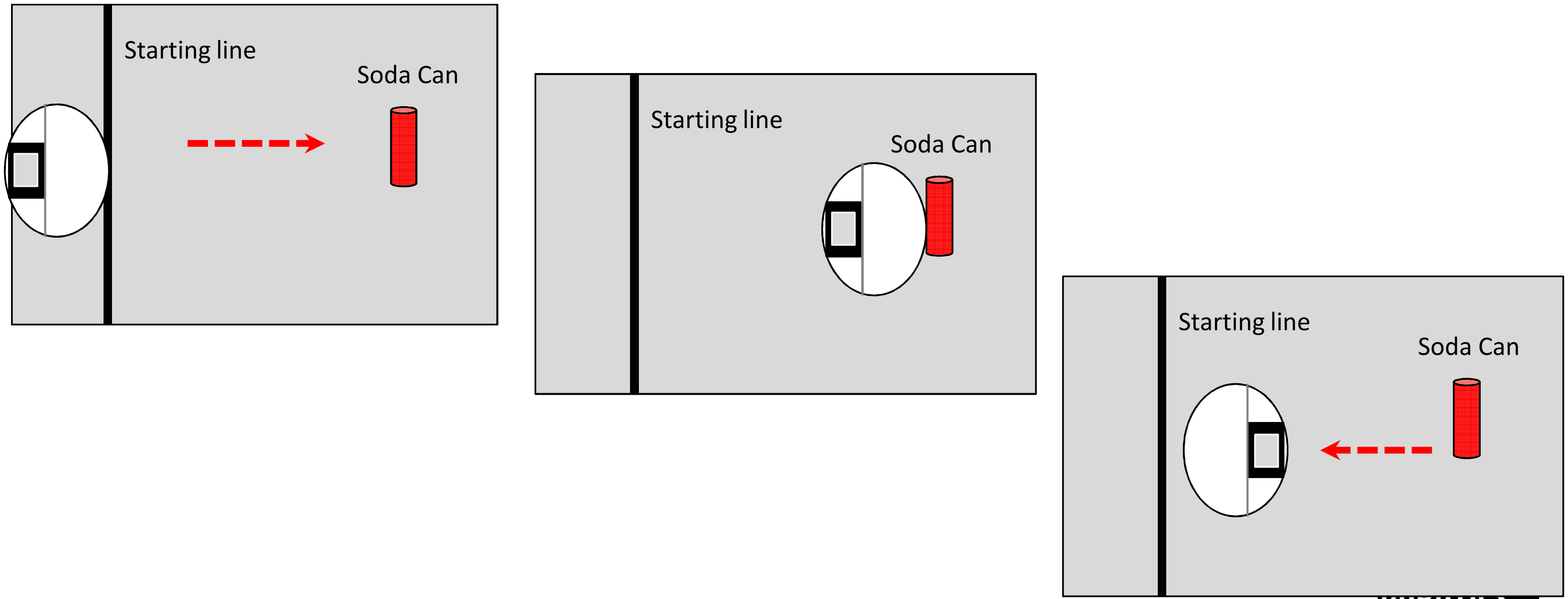
KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

TEACHER'S RESOURCE

Closest to/touch the Can and "Go Home"

1. A variation on touch the can and closest to the can.
2. After stopping closest/touching the can, back the robot up until touching the starting line
   - Move the can to various distances

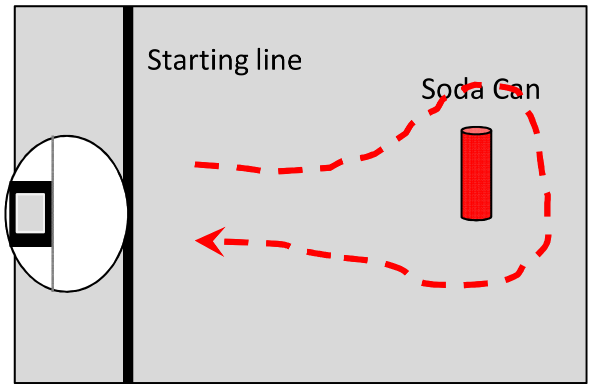


STUDENT ACTIVITY

TEACHER'S RESOURCE

Circle the Can and "Go Home"

1. Brings in the concept of turning
   - If you touch the can you must start over
   - The quickest trip is the winner
   - Move the can to various distances
   - Make them go clockwise and then counter clockwise

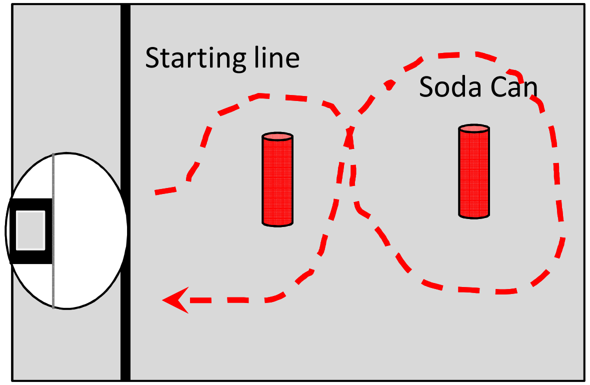


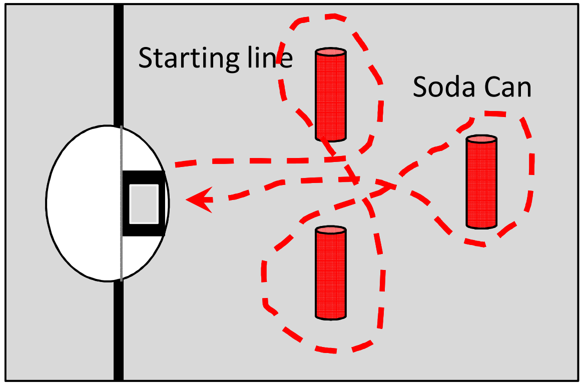
KISS INSTITUTE FOR PRACTICAL ROBOTICS

Circle the Can(s) and "Go Home"

Variation on Circle the Can

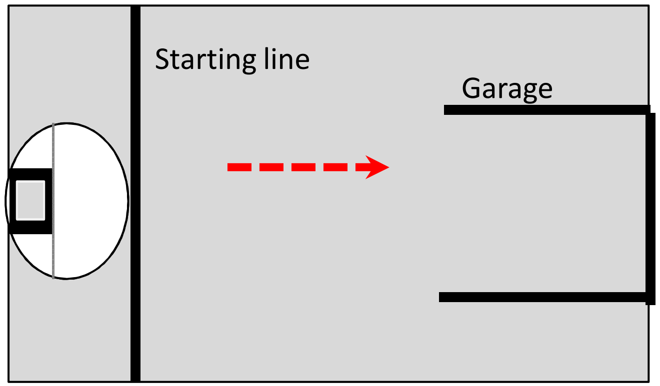
1. Have them make a figure 8 around two objects
2. Barrel Race (have them go around three cans)

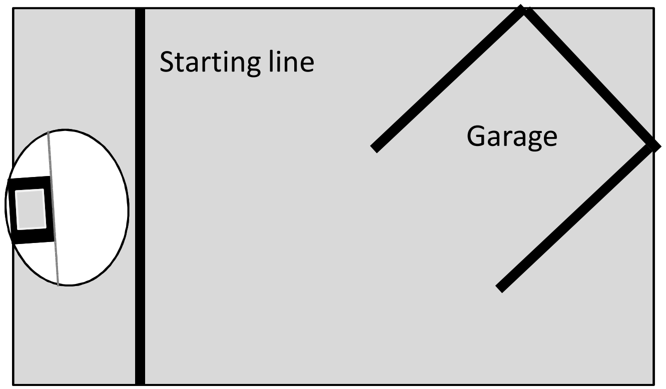




Park in the Garage

1. Robots must start on or behind the starting mark and park in the garage (box or tape outline on board)
   - Start with the garage straight across from the starting line
     - Garage can be roomy and then make it a tight fit
     - If they touch the garage they must start over at the starting line
   - If they touch the garage they must start over at the starting line
   - Move the garage to various distances and locations





KISS INSTITUTE FOR PRACTICAL ROBOTICS

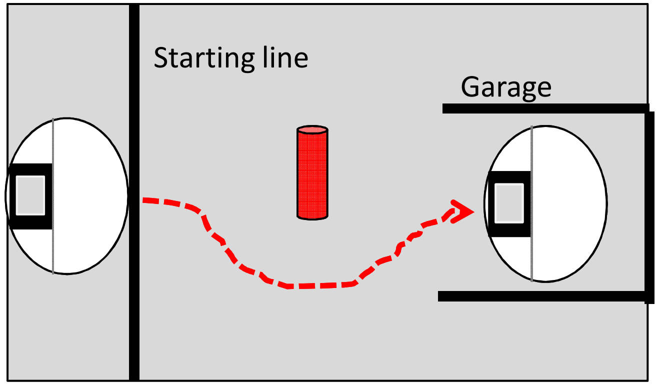
STUDENT ACTIVITY

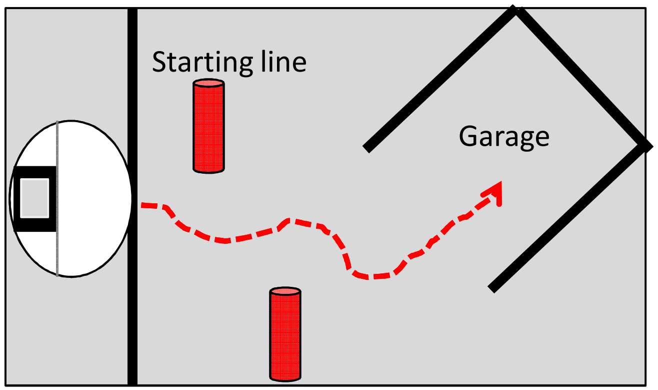
Park in the garage and Miss the Bicycle

TEACHER'S RESOURCE

"Park in the Garage" variation

- Place an object(s) between the starting line and garage





KISS INSTITUTE FOR PRACTICAL ROBOTICS

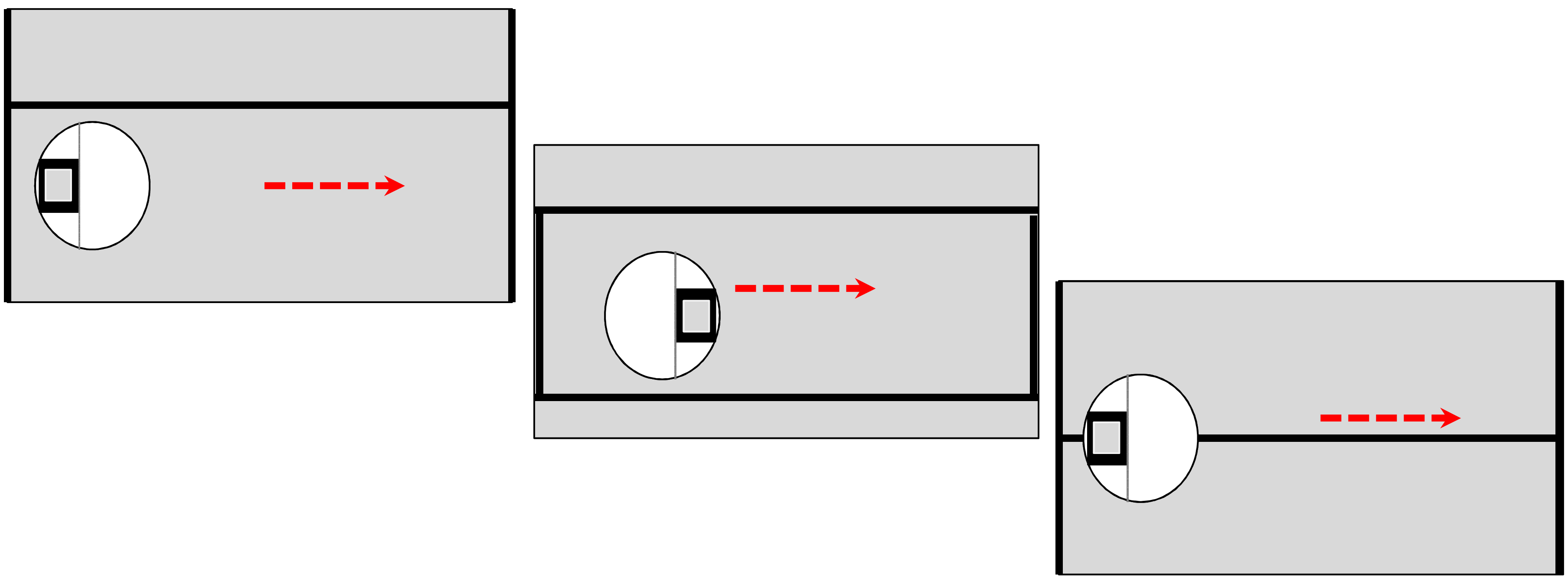
STUDENT ACTIVITY

TEACHER'S RESOURCE

Walk the Line

Brings in the concept of driving in a straight line

- Robot must move without touching the line (easiest to hardest below)
  - You can use one line and have the robot move down the side without touching it
    - Make this a time trial-quickest time without touching (faster is harder to control)
  - You can make a lane and have the robot drive down it without touching either side.
    - Increase difficulty by making the lane narrower
  - You can use one line and have the robot straddle it with the goal of running the full length without either wheel touching the line



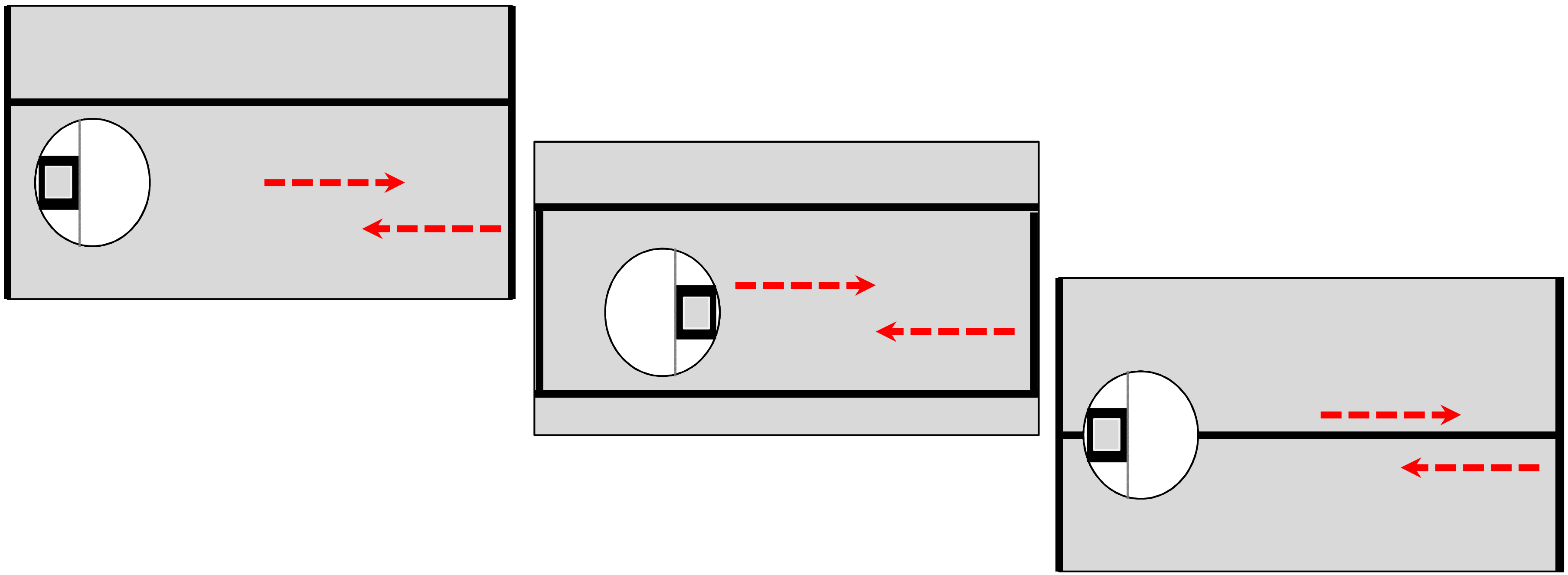
Variations on Walk the Line

STUDENT ACTIVITY

TEACHER'S RESOURCE

Same as before only have them stop and go backwards without touching the line as well

- Add a starting line to begin and a finish line the robot must touch before backing up



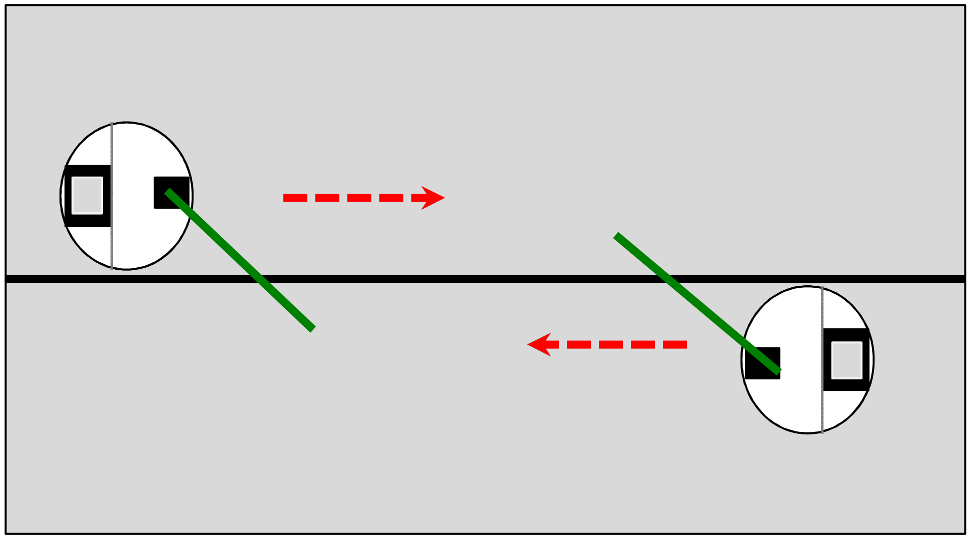
KISS INSTITUTE FOR PRACTICAL ROBOTICS

Variations on Walk the Line- Jousting!

- Robots on opposite sides of the line move towards each other and try to knock object off of other robot
  - Use whatever object is handy

Engineering Point-

Have the students engineer how they attach their lance (new unsharpened pencils work well) to their robot



STUDENT ACTIVITY

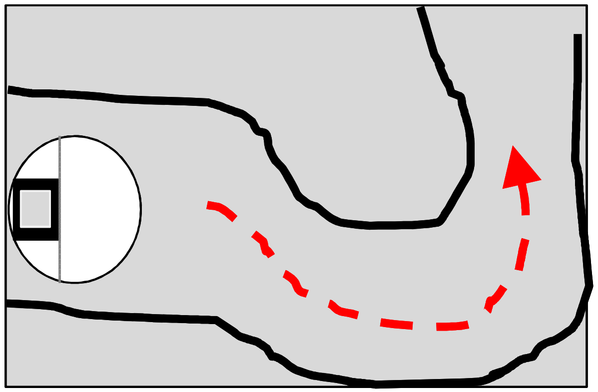
TEACHER'S RESOURCE

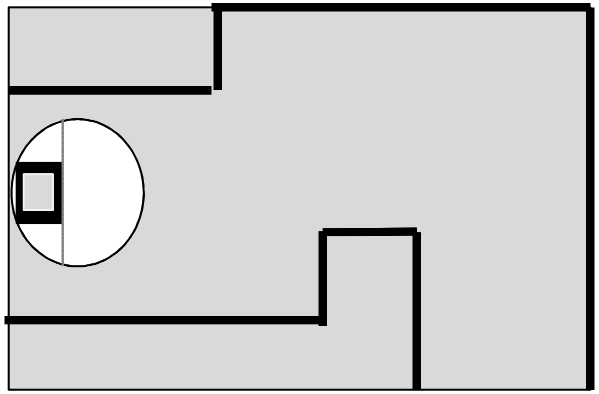
Race Track

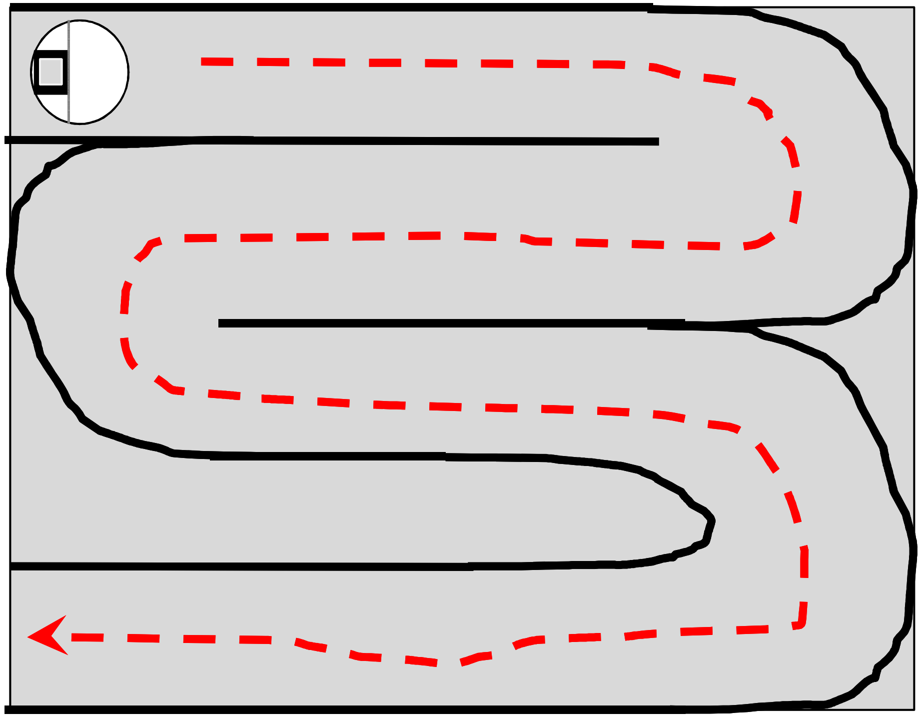
Brings in the concept of controlled driving

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 then you have to start over and the clock keeps running
- You can use a much larger track if desired (taped lanes on the classroom floor work well)
- You can use different lane setups
    - The tighter and more numerous the turns the more difficult it is
- Extension- once finished, make them stop and back up all the way to the start







KISS INSTITUTE FOR PRACTICAL ROBOTICS

Moving the *Create* with

create_drive_straight(); AND create_spin_block();

TEACHER'S RESOURCE

Goals

- To reinforce the concept of a function
- To learn and use the functions for connecting to and moving the Create

Preparation

- You will need a charged Create and Link + the serial cable to connect them
- You will need computers with the KISS IDE
- You will need the USB download cable

Activity

Follow the slides to make the robot move

KISS INSTITUTE FOR PRACTICAL ROBOTICS

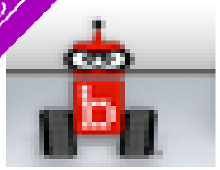
Activity 3

Lets make a robot move!

Use the Create with a Link controller in the cargo bay connected with a serial cable

KISS INSTITUTE FOR PRACTICAL ROBOTICS

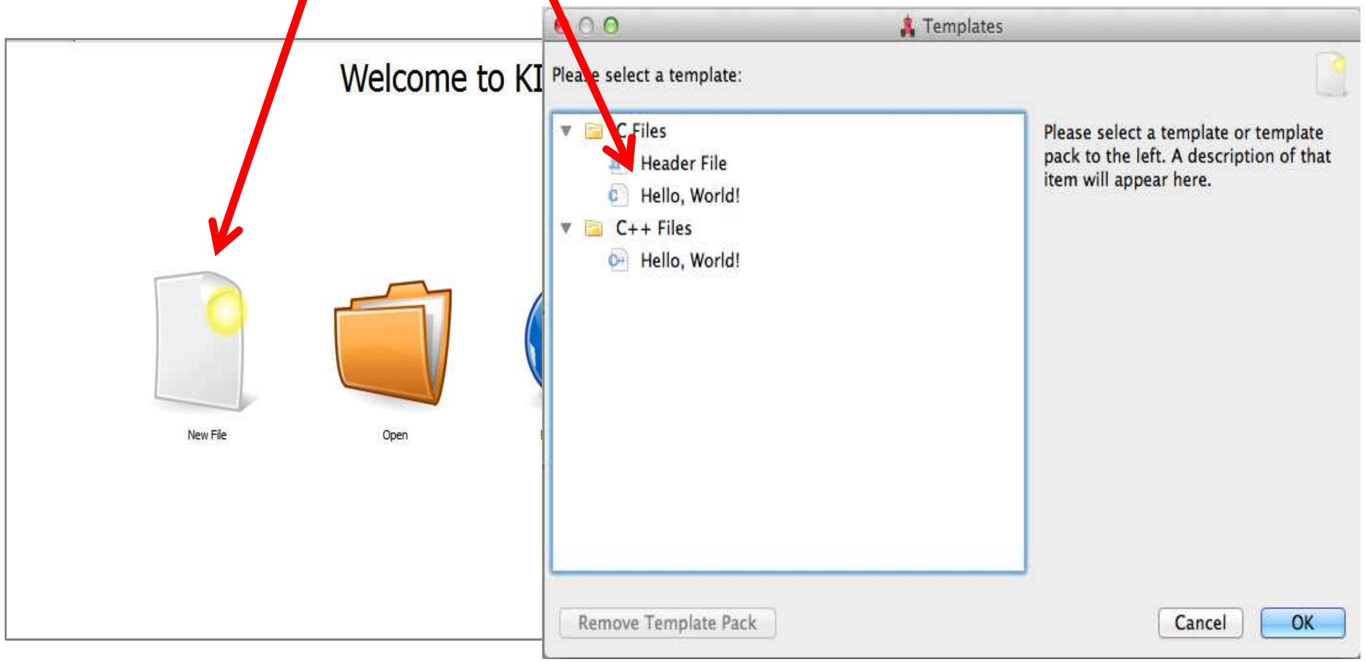
Launch the KISS IDE



KISS IDE icon

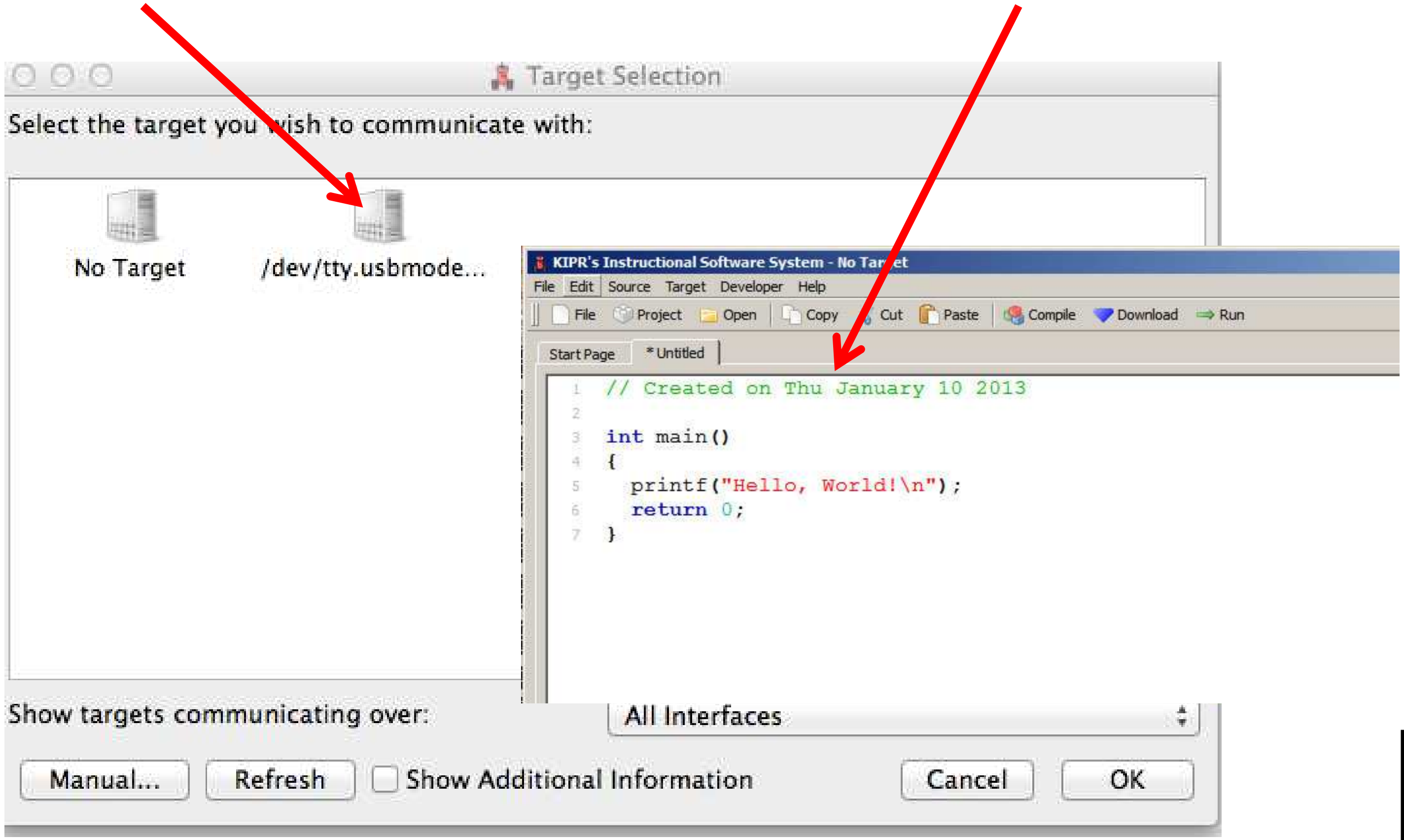
- Start the KISS IDE by clicking on its icon to get the welcome screen
- Click on the *"New File"* icon and

  and choose the C, *"Hello, World!"* template



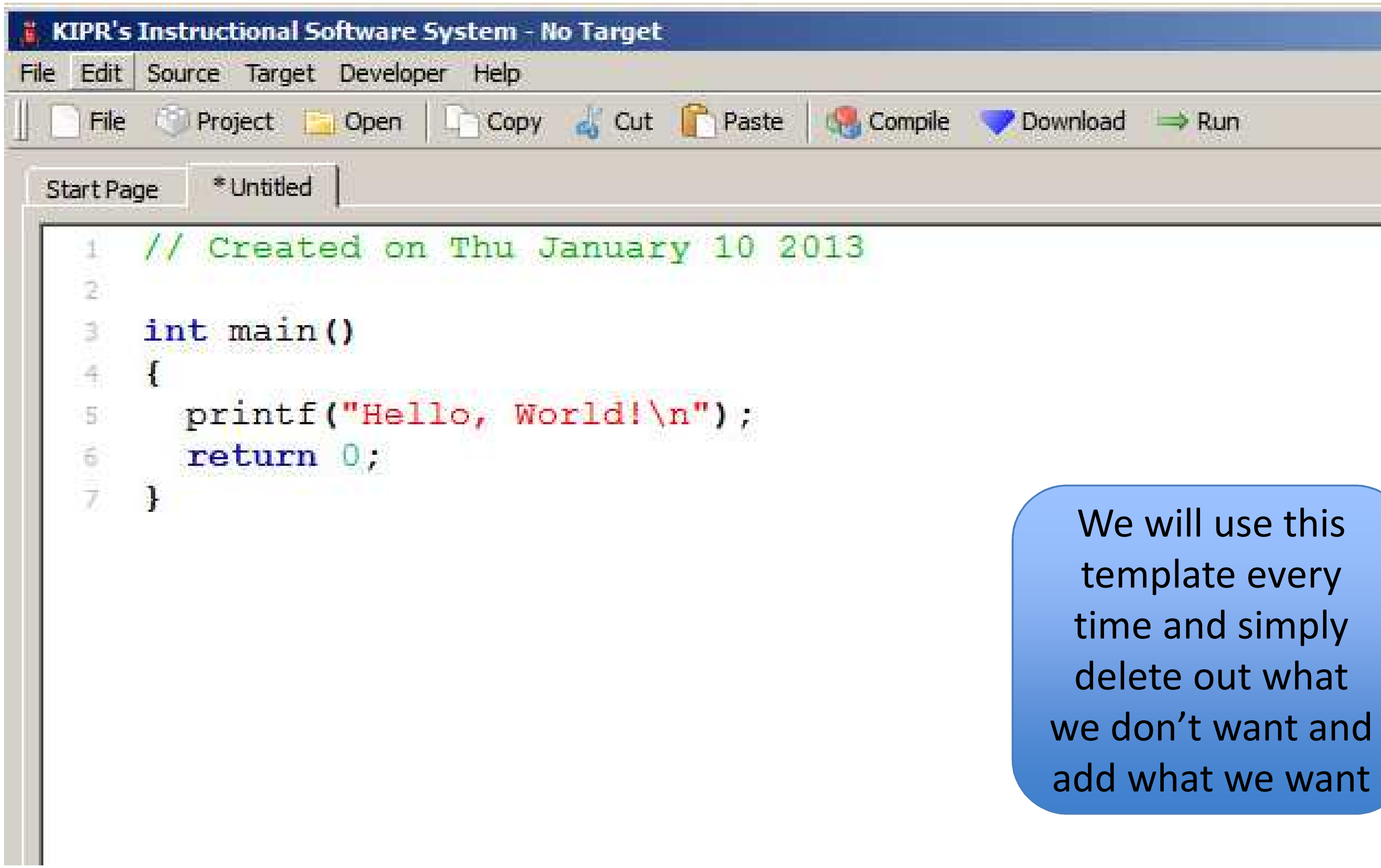
Select Target

- A Target Selection window will appear
- Pick *"USB Target"* and the C program template will come up



STUDENT ACTIVITY

The C Template: Hello, World!



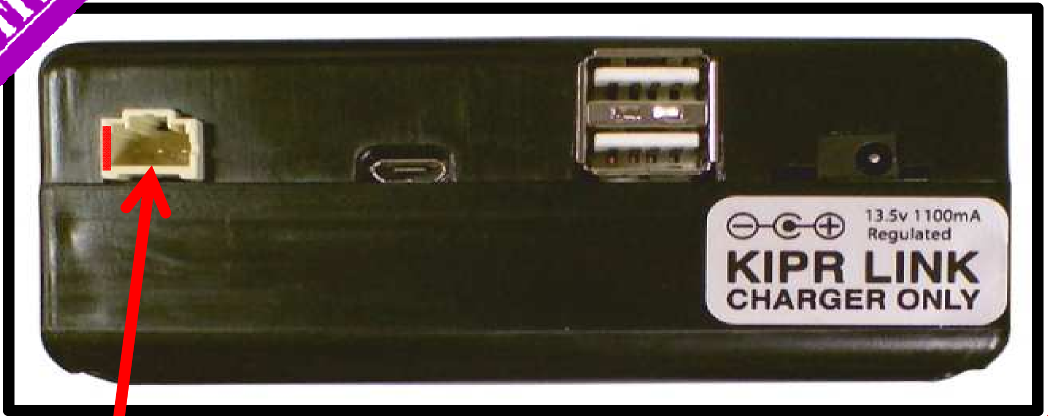
We will use this template every time and simply delete out what we don't want and add what we want

KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

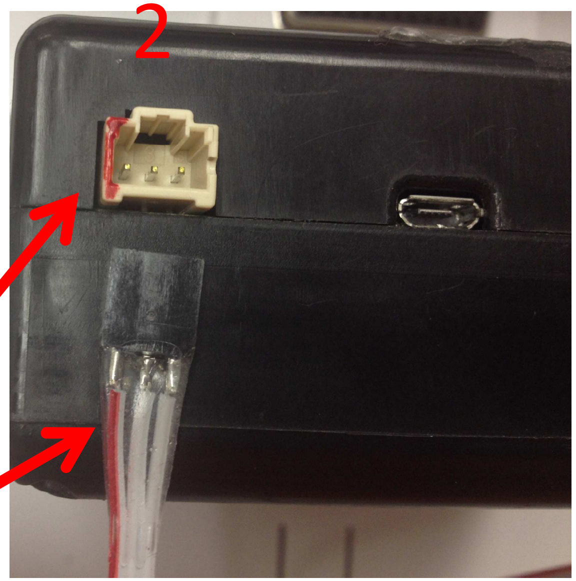
Plugging serial cable into LINK

1

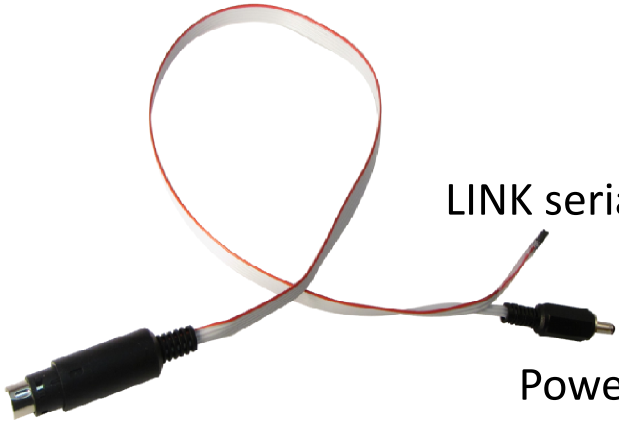


We use the serial cable to plug into the LINK TTL Serial plug

2



NOTICE the red mark on the plug (left side) this corresponds to the red wire in the serial cable

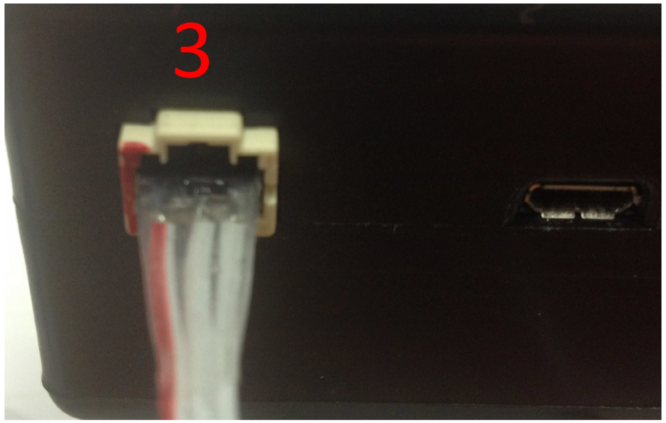


LINK serial plug

Power connector

Create connector

3



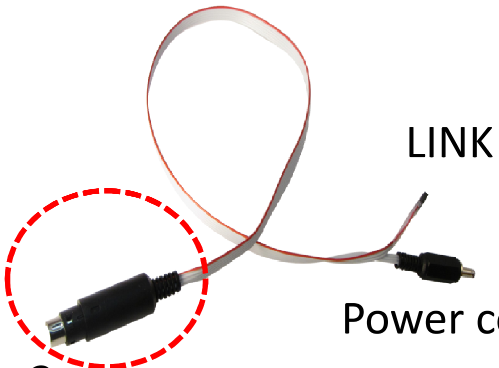
Correctly plugged in

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Plugging serial cable into *Create*

STUDENT ACTIVITY

We use the Create connector (round) end



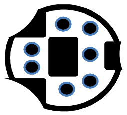
LINK serial plug

Power connector

Create connector

Receptacle may have a cover that you can pop off to access

The plug is keyed, make sure you line it up correctly before plugging it in



KISS INSTITUTE FOR PRACTICAL ROBOTICS

Functions to Move and Stop

STUDENT ACTIVITY

Create commands run UNTIL a different motor command is received

create_drive_straight() (200);

Speed in mm/second for BOTH right and left wheels

```
create_drive_straight (200); //moves forward at 200mm/sec
```

***WARNING maximum speed for the Create motors is 1000mm/second = 1 meter (~3feet)/second. It will jump off a table in a second! Use something like 200 for the speed (moderate speed) until teams get the hang of this**

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Explain using comments

You can use a flow chart and then translate that into comments.

Using //comments as pseudocode is a great way to start.

If you forget which functions to use, look at your cheat sheet.

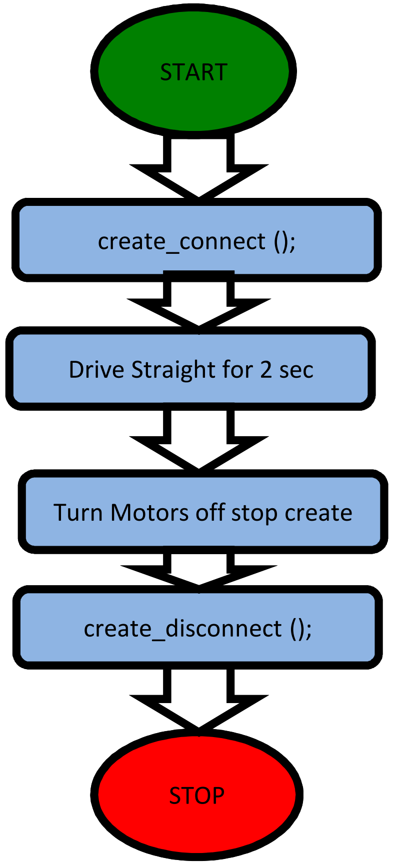
STUDENT ACTIVITY

Lets make a robot move!

Write a program for your robot to move forward for 2 seconds

Psuedocode (Task Analysis)

```
// 1. connect to create
// 2. Drive straight at 500mm/sec
// 3. Pause program for 2 seconds
to give the robot time to move
// 4. stop the motors/create
// 5. disconnect from create
```



KISS INSTITUTE FOR PRACTICAL ROBOTICS

Activity Solution

TEACHER'S RESOURCE

```c
// Created on Wed October 16 2013

int main()
{
    create_connect(); // conect link to create
    create_drive_straight(200);//drive both motors at 500mm/second
    msleep (2000); // Pause program for 2 seconds to give robot time to move
    create_stop(); // stop create
    create_disconnect();//disconnect from create
    return 0;
}
```

Function toTurn/Spin

STUDENT ACTIVITY

Create commands run UNTIL a different motor command is received

Speed in mm/second for BOTH right and left wheels

```
create_drive_straight (200); //moves forward at 200mm/sec
```

***WARNING maximum speed for the Create motors is 1000mm/second = 1 meter (~3feet)/second. It will jump off a table in a second! Use something like 200 for the speed (moderate speed) until teams get the hang of this**

KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Function to Spin your robot

Create commands run UNTIL a different motor command is received

create_spin_block() (200,90);

Degree of spin

Speed in mm/second of the spin (this will turn counterclockwise)

```
create_spin_block() (200,90);//spins 90⁰ at 200mm/sec
```

***WARNING maximum speed for the Create motors is 1000mm/second = 1 meter (~3feet)/second. It will jump off a table in a second! Use something like 200 for the speed (moderate speed) until you get the hang of this**

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Lets make a robot draw a square!

Write a program for your robot to move forward for 2 seconds and then make a 90^0 turn

Psuedocode (Task Analysis)

```
// 1. connect to create
// 2. Drive straight at 200mm/sec
// 3. Pause program for 2 seconds to
      give the robot time to move
// 4. Turn counter clockwise 90⁰
// 5. Stop motors
// 5. disconnect from create
```

Activity Solution

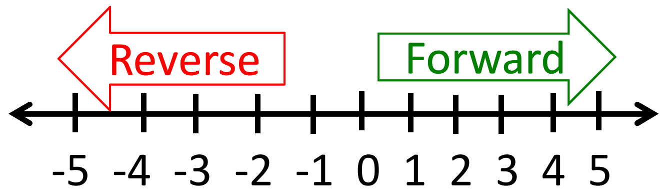
TEACHER'S RESOURCE

```c
// Created on Wed October 16 2013

int main()
{
    create_connect(); // conect link to create
    create_drive_straight(200);//drive both motors at 500mm/second
    msleep (2000); // Pause program for 2 seconds to give robot time to move
    create_spin_block(200, 90); // spin/turn counter clockwise 90 degrees at 200mm/second
    create_stop(); // stop create
    create_disconnect();//disconnect from create
    return 0;
}

```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Create Driving Hints

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



The Create is very fast, at 1000mm/sec

It can get away from students quickly

– The Create is heavy and can produce lots of inertia/momentum (keep this in mind while trying to get precise distances)

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

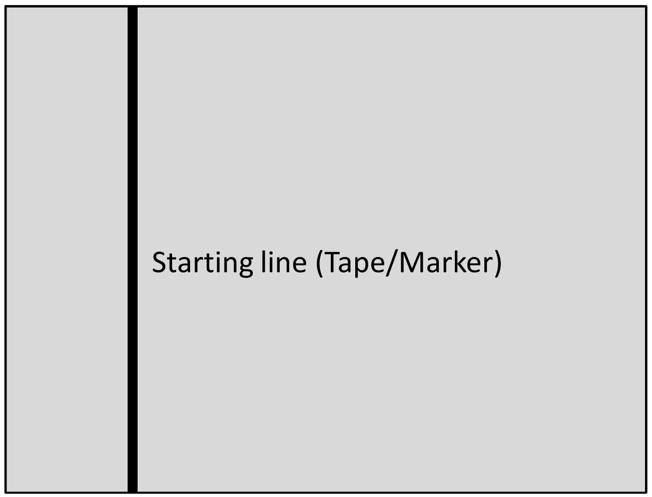
TEACHER'S RESOURCE

KISS INSTITUTE FOR PRACTICAL ROBOTICS

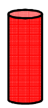
TEACHER'S RESOURCE

LET'S MOVE! Materials/Supplies

1. You need a large surface to run the robot on
   - Use the floor, a piece of white or light colored foam or poster board or a vinyl or paper mat as a robot testing track
     - o You need an area marked as the starting line (a piece of black tape works well or you can mark it with a black marker)
2. You need an object to navigate to
   - Can of soda, foam block, whiteboard eraser, etc. will work
3. A measuring device and a timer will be useful



Soda Can



LET'S MOVE!

Activity/mini contests

Using the simple motor function motor`();` and `msleep();` you can have the students work on fun challenges.

These activities can all be completed using hard coding ("dead reckoning") and simple motor control functions without the use of any sensors. This is a good place to start and will teach the students how hard it is to be consistent using dead reckoning.

- This is a good time to bring up controlling variables when they set up their robot- is it the same every time? How could you make it the same (using a jig or ruler to control how they set it at the starting line)

Once they have the skills down of forward, backwards, stop, turn then we can move on and start adding sensors and decision making into the programs.

STUDENT ACTIVITY

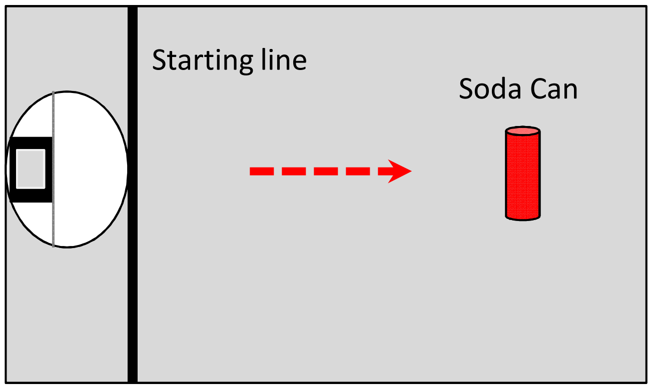
TEACHER'S RESOURCE

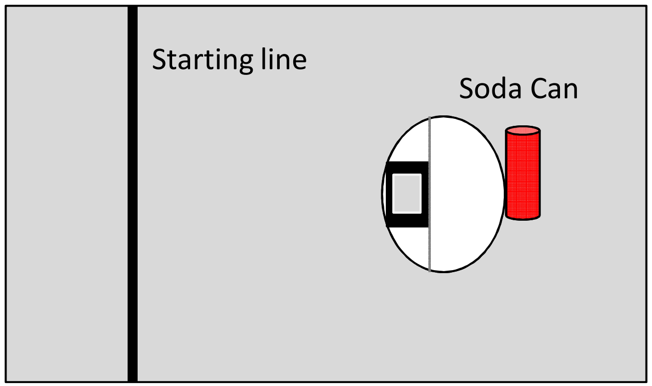
Touch the Can

Robots must start on or behind the starting mark and move to the object with the goal of touching the object in the shortest amount of time

Extensions

- Move the can to various distances
- Make the object smaller and harder to navigate to
- Math- have them measure the distance to the object and time the robot and then calculate rate/speed
    - o Speed = Distance/Time





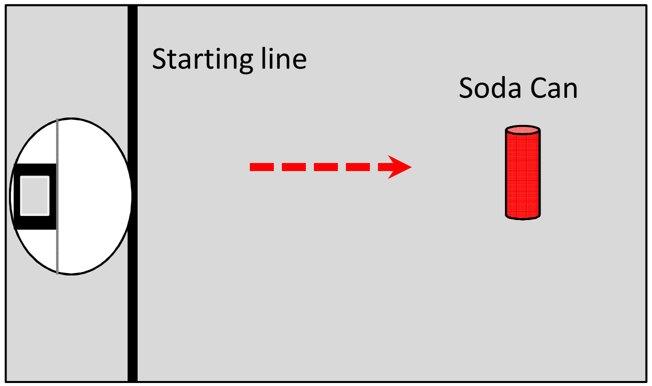
KISS INSTITUTE FOR PRACTICAL ROBOTICS

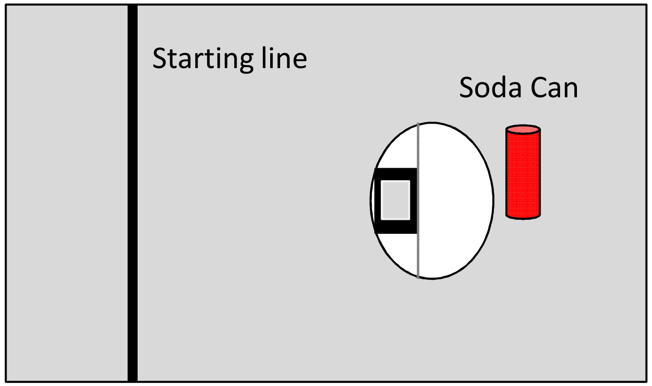
STUDENT ACTIVITY

TEACHER'S RESOURCE

Closest to the Can

1. Robots must start on or behind the starting mark and move to 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 or have them average several runs or add the distances together for a grand total
2. Move the can to various distances and locations





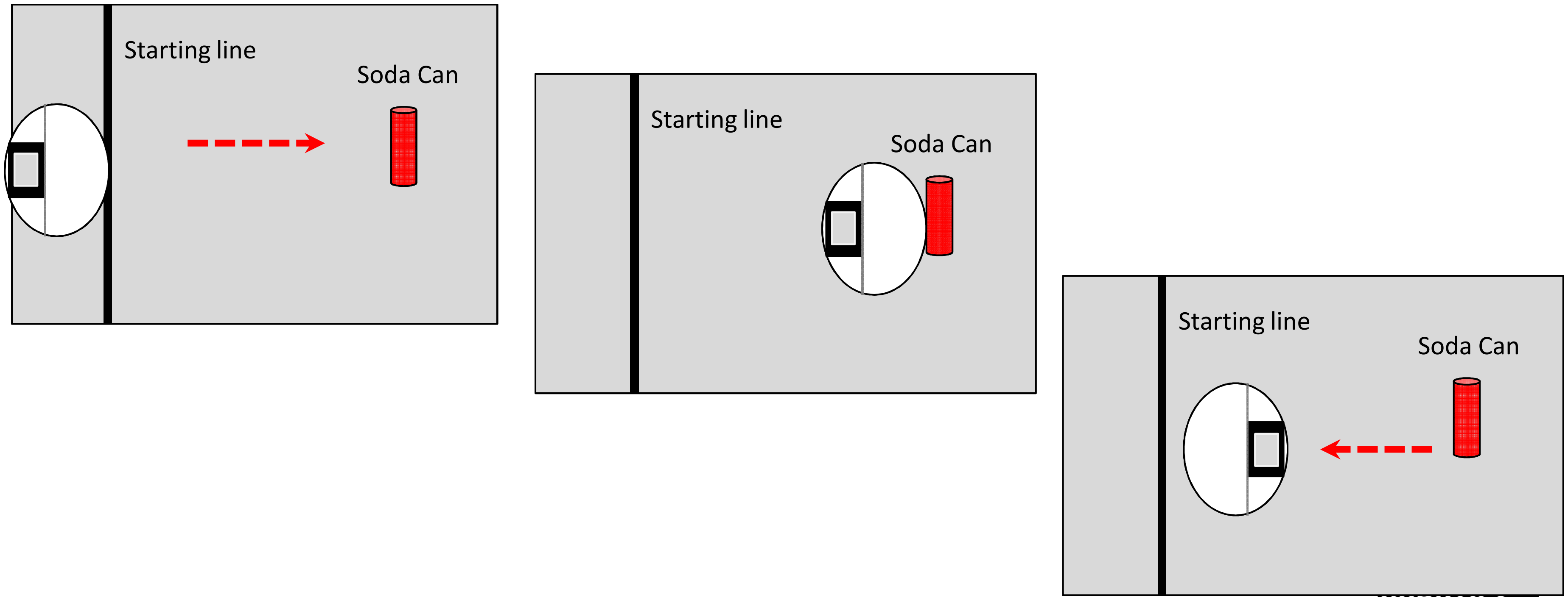
KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

TEACHER'S RESOURCE

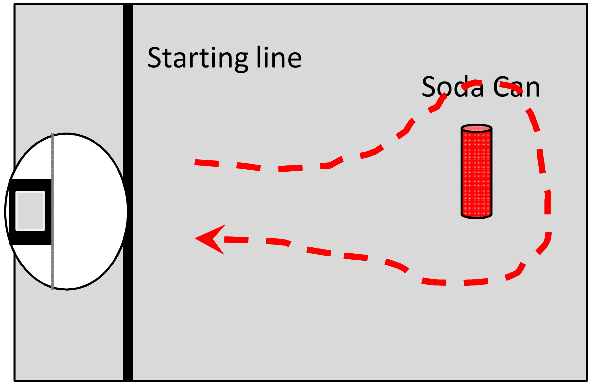
Closest to/touch the Can and “Go Home”

1. A variation on touch the can and closest to the can.
2. After stopping closest/touching the can, back the robot up until touching the starting line
   - Move the can to various distances



Circle the Can and "Go Home"

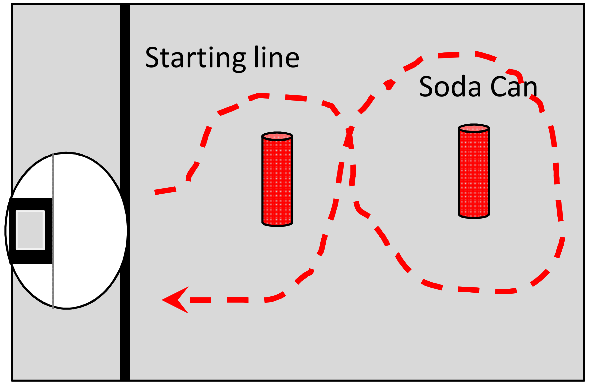
1. Brings in the concept of turning
   - If you touch the can you must start over
   - The quickest trip is the winner
   - Move the can to various distances
   - Make them go clockwise and then counter clockwise

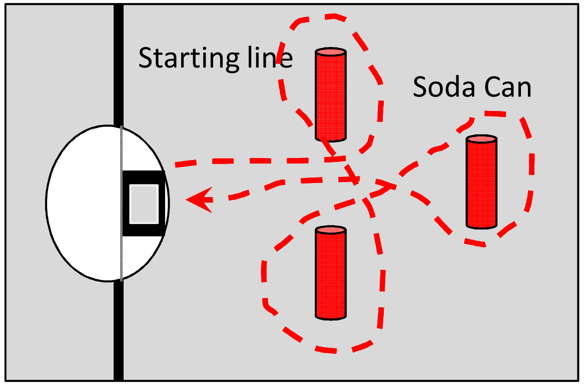


Circle the Can(s) and "Go Home"

Variation on Circle the Can

1. Have them make a figure 8 around two objects
2. Barrel Race (have them go around three cans)



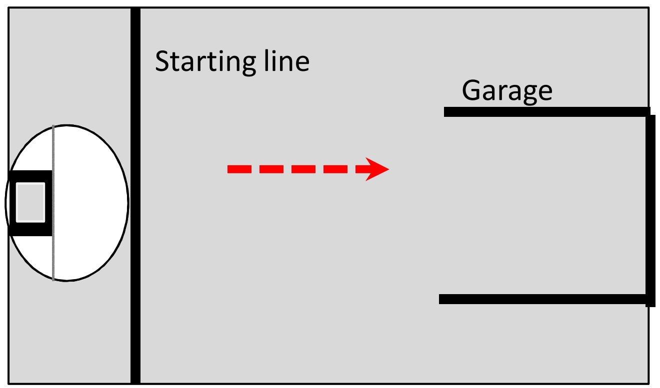


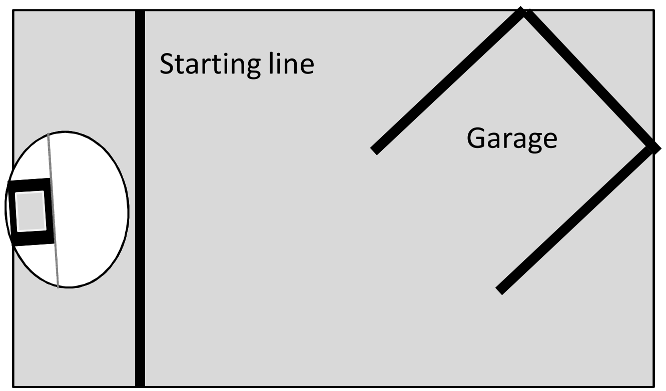
STUDENT ACTIVITY

TEACHER'S RESOURCE

Park in the Garage

1. Robots must start on or behind the starting mark and park in the garage (box or tape outline on board)
   - Start with the garage straight across from the starting line
     - Garage can be roomy and then make it a tight fit
     - If they touch the garage they must start over at the starting line
   - If they touch the garage they must start over at the starting line
   - Move the garage to various distances and locations





KISS INSTITUTE FOR PRACTICAL ROBOTICS

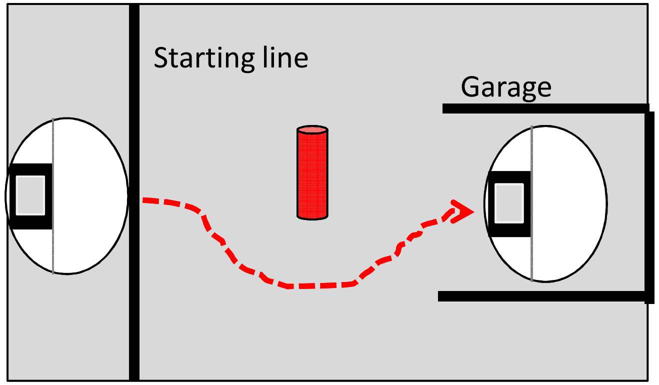
STUDENT ACTIVITY

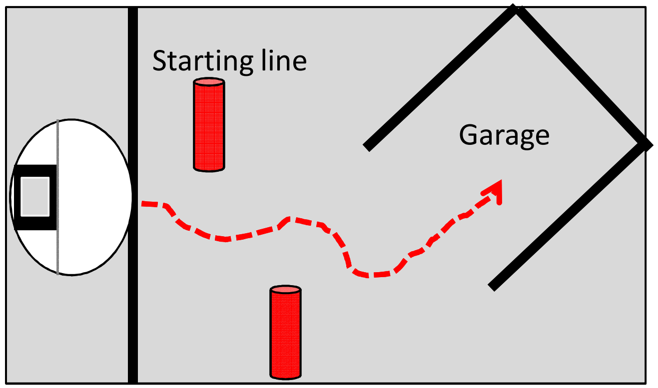
TEACHER'S RESOURCE

Park in the garage and Miss the Bicycle

“Park in the Garage” variation

- Place an object(s) between the starting line and garage





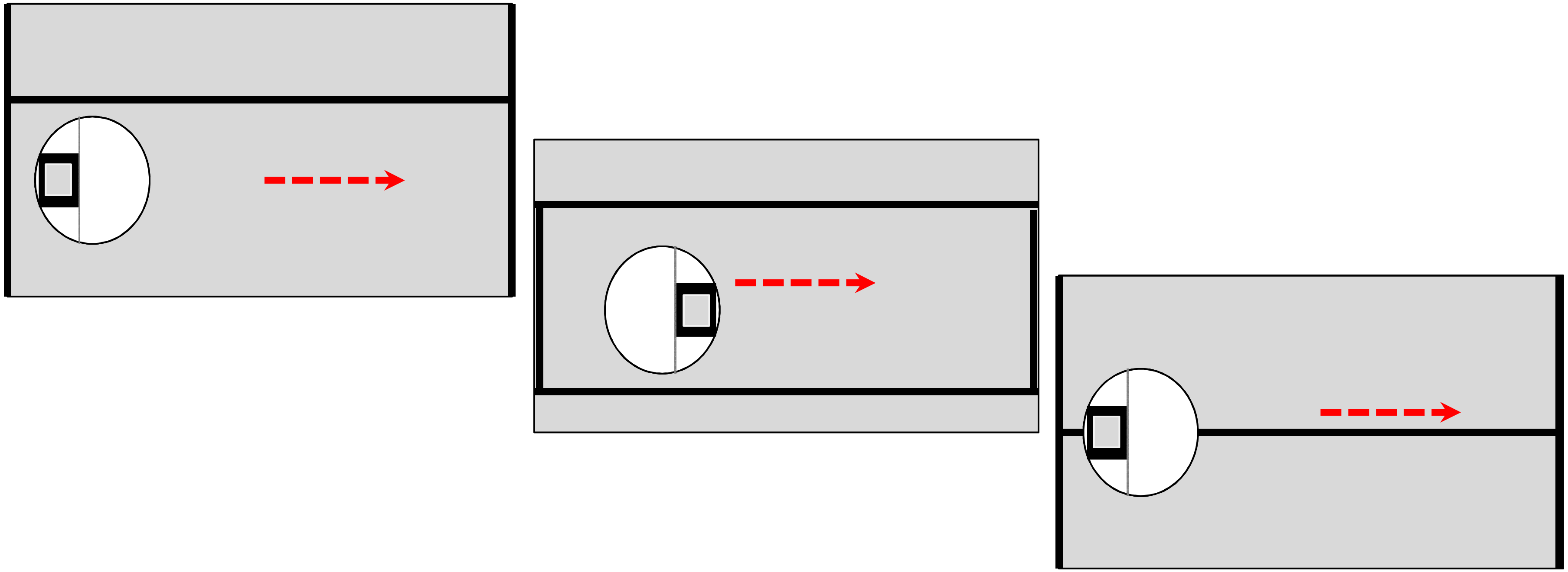
STUDENT ACTIVITY

TEACHER'S RESOURCE

Walk the Line

Brings in the concept of driving in a straight line

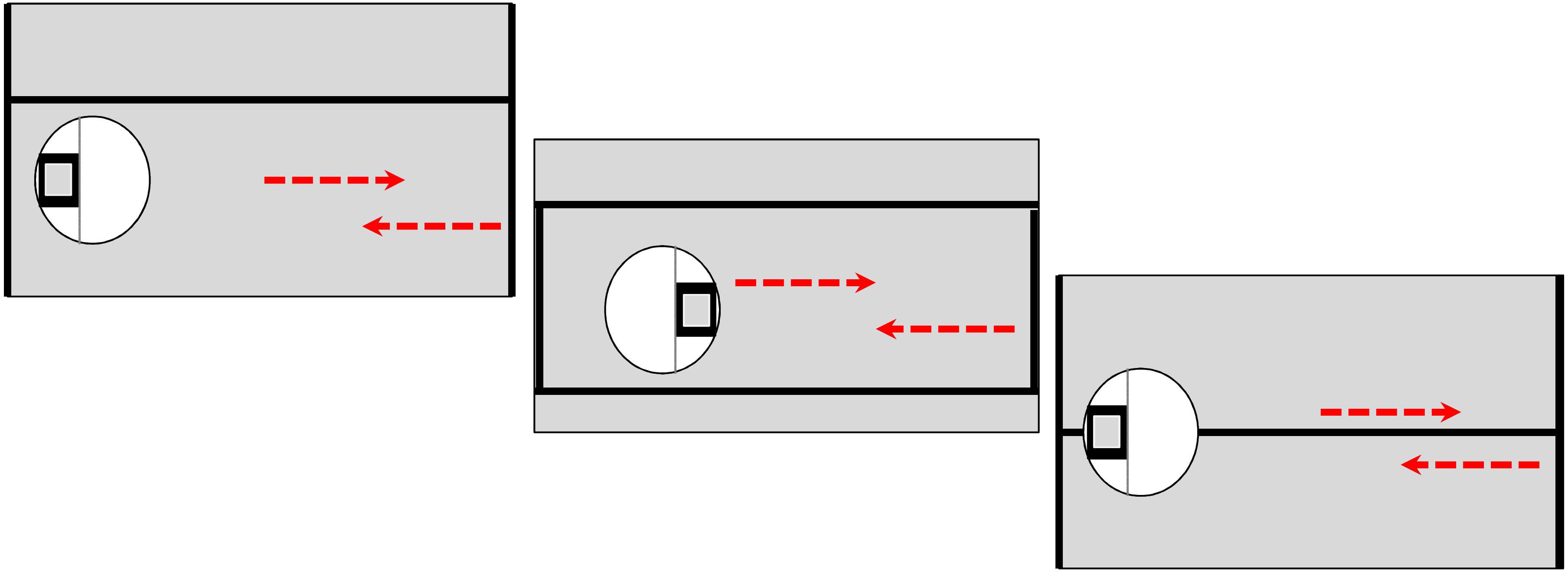
- Robot must move without touching the line (easiest to hardest below)
  - o You can use one line and have the robot move down the side without touching it
    - Make this a time trial-quickest time without touching (faster is harder to control)
  - o You can make a lane and have the robot drive down it without touching either side.
    - Increase difficulty by making the lane narrower
  - o You can use one line and have the robot straddle it with the goal of running the full length without either wheel touching the line



Variations on Walk the Line

Same as before only have them stop and go backwards without touching the line as well

- Add a starting line to begin and a finish line the robot must touch before backing up

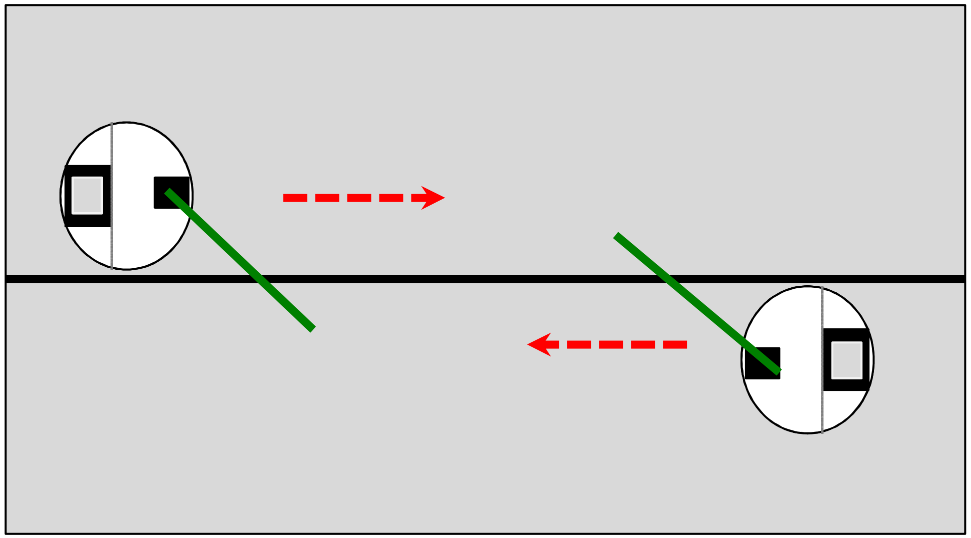


Variations on Walk the Line- Jousting!

- Robots on opposite sides of the line move towards each other and try to knock object off of other robot
  - Use whatever object is handy

Engineering Point-

Have the students engineer how they attach their lance (new unsharpened pencils work well) to their robot



STUDENT ACTIVITY

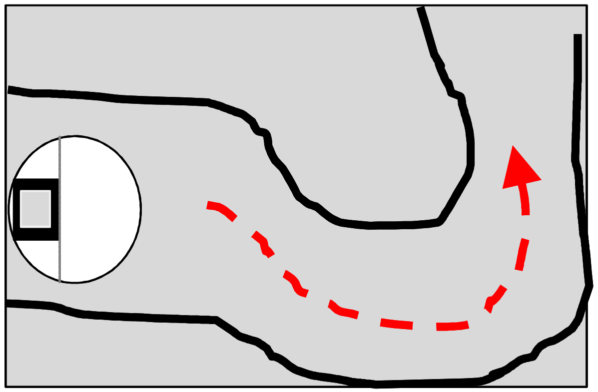
TEACHER'S RESOURCE

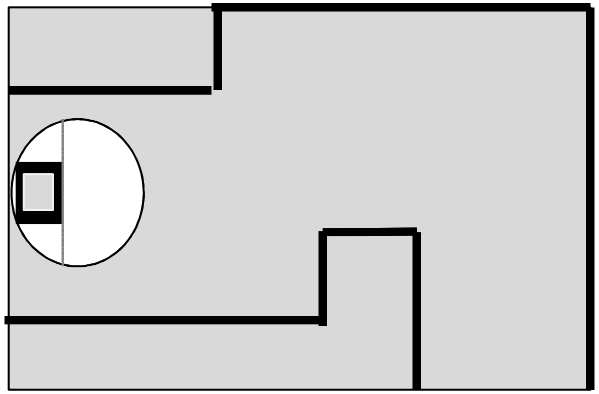
Race Track

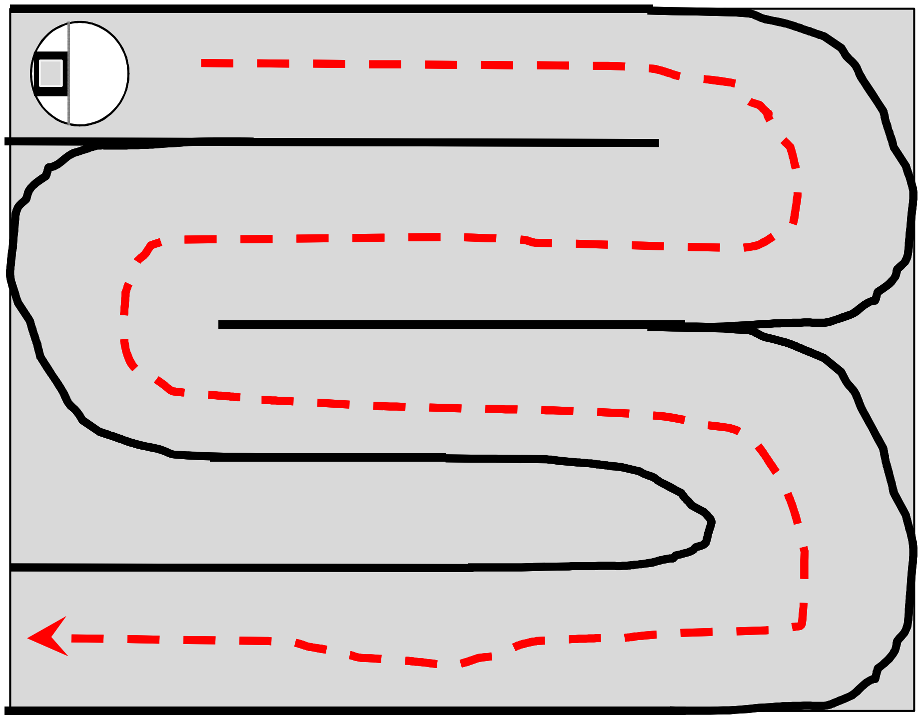
Brings in the concept of controlled driving

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 then you have to start over and the clock keeps running
- You can use a much larger track if desired (taped lanes on the classroom floor work well)
- You can use different lane setups
  - The tighter and more numerous the turns the more difficult it is
- Extension- once finished, make them stop and back up all the way to the start







KISS INSTITUTE FOR PRACTICAL ROBOTICS

Using sensors with the *Create*

Goals

- To learn and use the functions for connecting to and moving the Create
- To learn the functions used to access sensors built into the Create
- To learn how to record distance traveled and angle turned and get the value
- To use real distance measurement to compare to sensor measurement for distance traveled and angle turned.

Preparation

- You will need a charged Create and Link + the serial cable to connect them
- You will need computers with the KISS IDE
- You will need the USB download cable
- A meter stick and protractor or other measuring device

Activity

Follow the slides

Activity 3

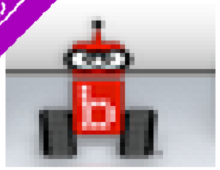
Lets use the Create's built-in sensors!

Use the Create with a Link controller in the cargo bay connected with a serial cable

KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

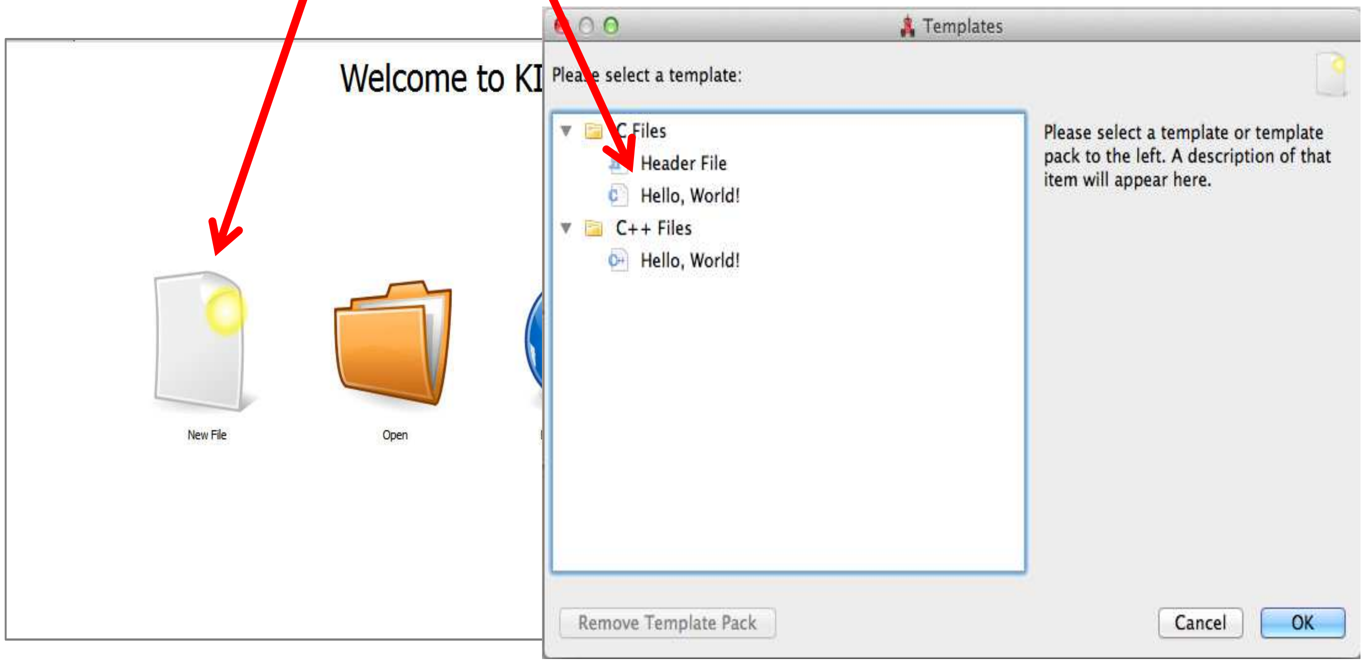
Launch the KISS IDE



KISS IDE icon

- Start the KISS IDE by clicking on its icon to get the welcome screen
- Click on the *"New File"* icon and

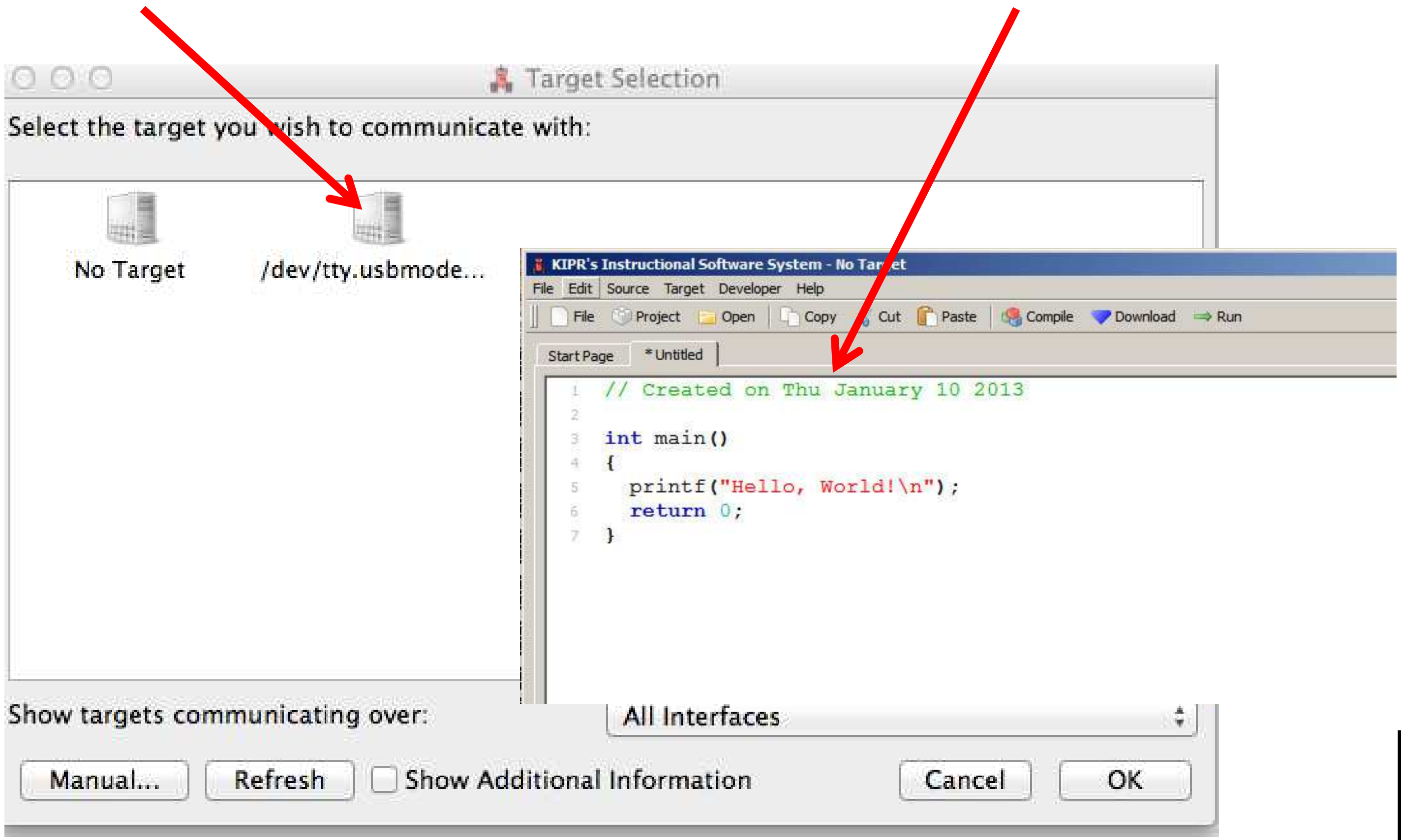
  and choose the C, *"Hello, World!"* template



STUDENT ACTIVITY

Select Target

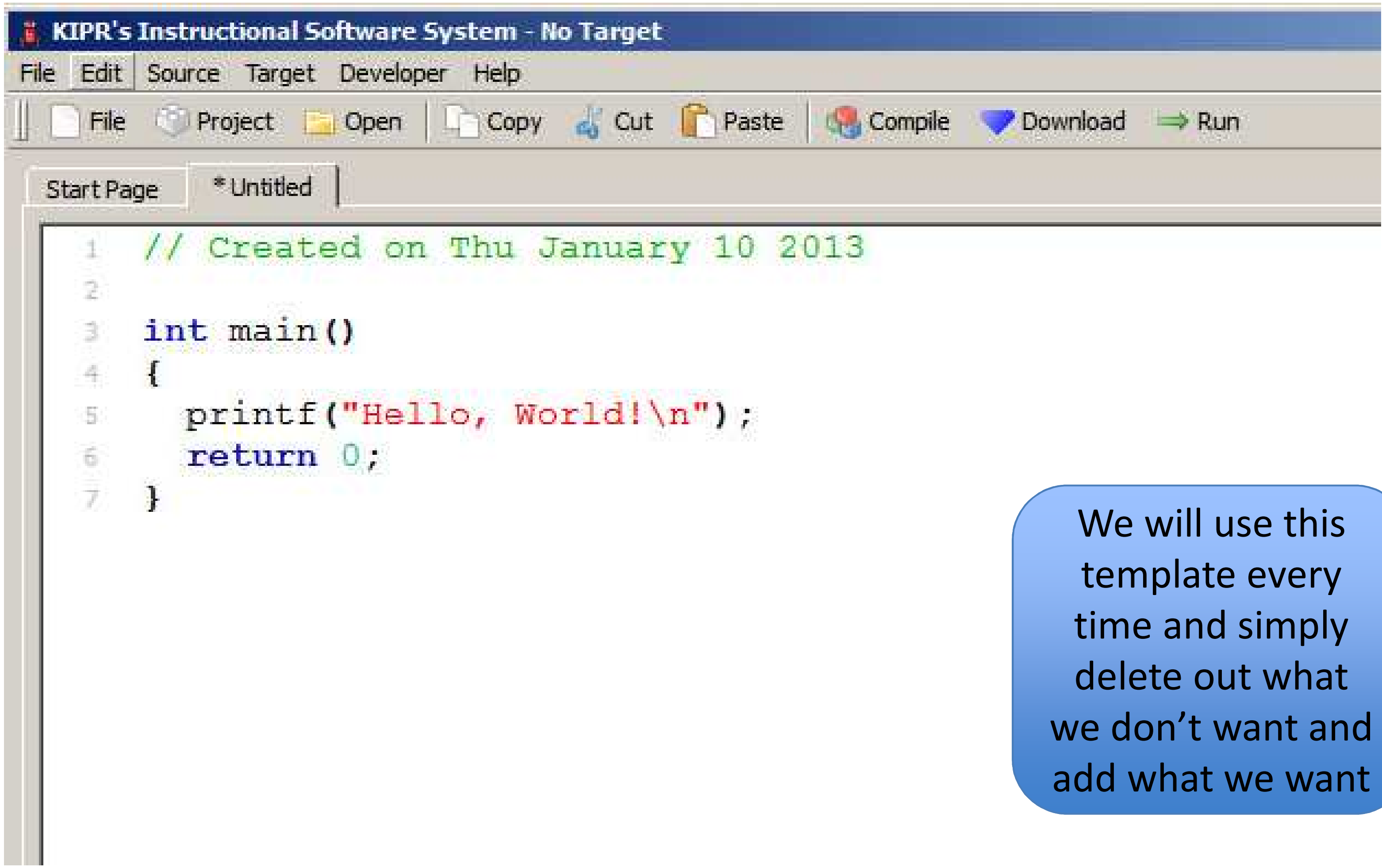
- A Target Selection window will appear
- Pick *"USB Target"* and the C program template will come up



KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

The C Template: Hello, World!

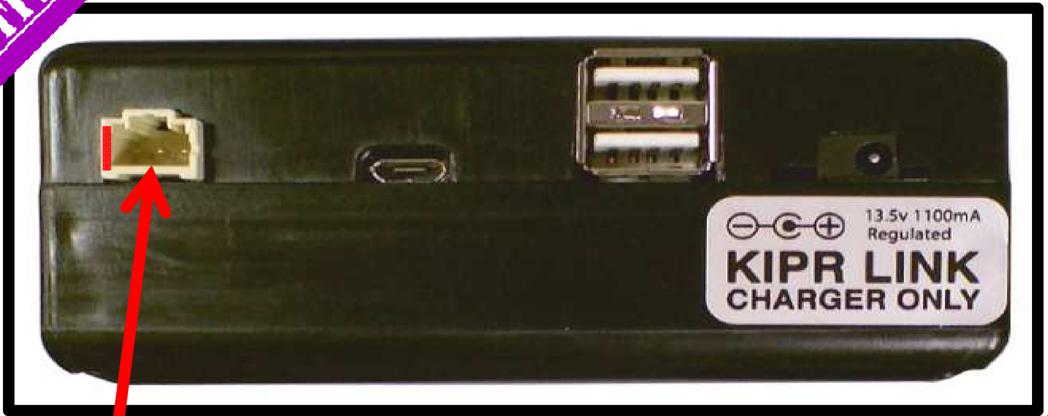


We will use this template every time and simply delete out what we don't want and add what we want

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Plugging serial cable into Link

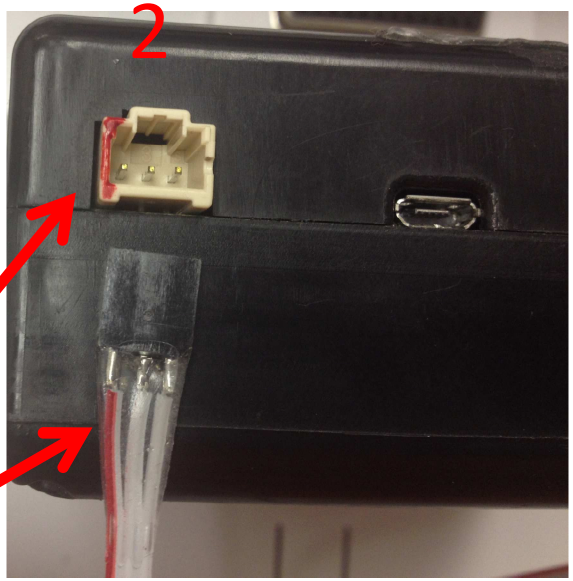
1

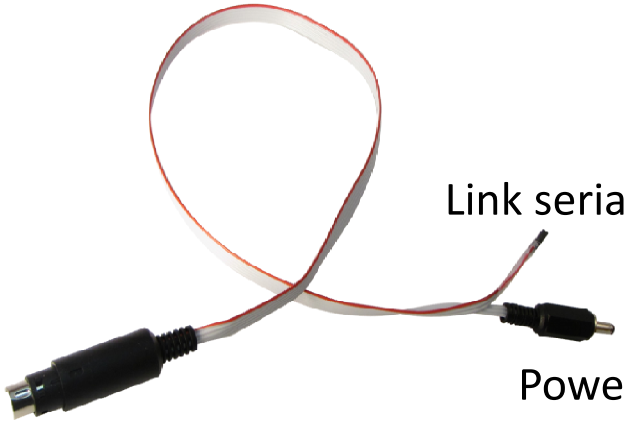


We use the serial cable to plug into the Link TTL Serial plug

NOTICE the red mark on the plug (left side) this corresponds to the red wire in the serial cable

2



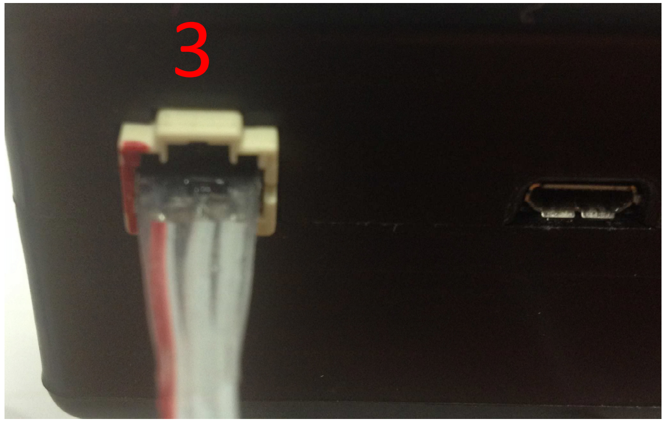


Link serial plug

Power connector

Create connector

3



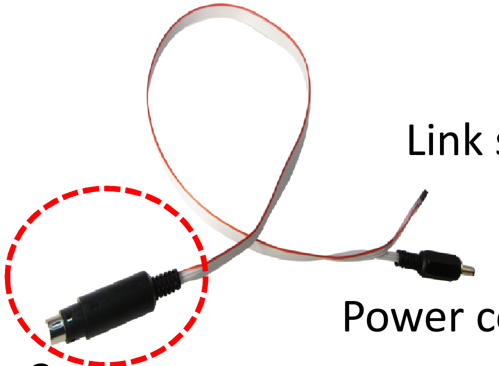
Correctly plugged in

KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Plugging serial cable into *Create*

We use the Create connector (round) end



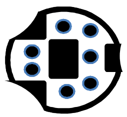
Link serial plug

Power connector

Create connector

Receptacle may have a cover that you can pop off to access

The plug is keyed, make sure you line it up correctly before plugging it in



KISS INSTITUTE FOR PRACTICAL ROBOTICS

Function to get distance traveled

STUDENT ACTIVITY

The Create has a built-in sensor that measures distance traveled in mm

- The `set_create_distance ();` function allows you to reset the counter

The function `get_create_distance ();` returns the recorded value (distance traveled in mm)

```
set_create_distance(0);//tells the Link to reset the distance to 0

get_create_distance();// gets the distance traveled in mm
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

How accurate is the get_create_distance (); ?

Write a program for your robot to move forward for 1m = 100cm = 1000 mm

Psuedocode (Task Analysis)

```
// 1. connect to create
// 2. Reset create distance traveled to 0
// 3. Drive forward @ 500mm/second
// 4. Pause program for 2 seconds to give
      the robot time to move 1000mm
// 5. stop the motors/create
// 6. print get_create_distance (); to
      screen
// 7. disconnect from create
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Activity Solution

```c
// Created on Thu October 10 2013

int main()
{
  create_connect ();
  set_create_distance (0); // resets distance traveled to start at 0
  create_drive_direct (500, 500); //drive forward at 500 mm per second
  msleep (2000); // sleep for 2 seconds to let the robot drive 2 sec X 500mm/sec = 1000mm or 1meter
  create_stop(); //stop the create motors
  printf("%i\n", get_create_distance ()); // prints the distance traveled to the screen
  create_disconnect (); //disconnects the link from the create
  return 0;
}
```

TEACHER'S RESOURCE

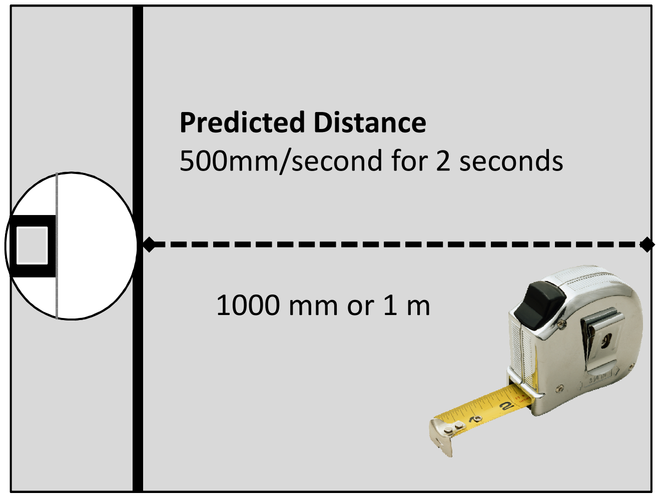
Checking the distance traveled value returned with an actual measurement

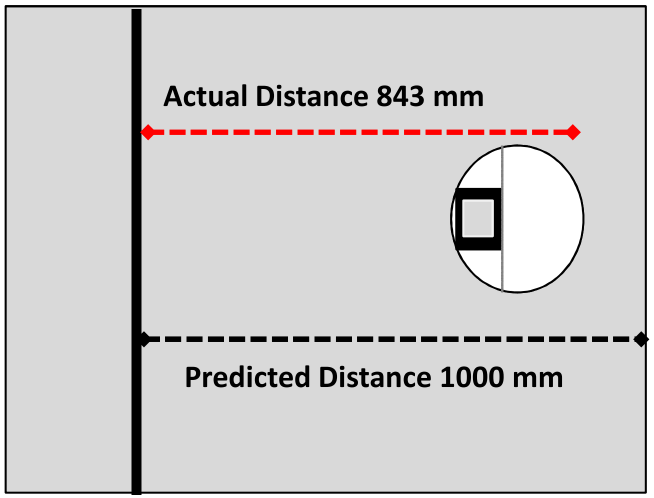
If you place the Create at a starting line and run the program in a perfect world it should go 1000mm or 1 m, BUT this isn't a perfect world

In reality it will go something less (friction, motors are different etc)

- In the previous example our Create printed out it had gone 848mm
- The actual physical measurement with a meter stick was 870 mm

Both were ~ 140mm less than the predicted 1000mm, but you can figure out the differences and account for this when programming the robot (a data table with conversions would be helpful)





KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Function to get angle turned

The Create has a built-in sensor that measures angle turned in degrees

- The `set_create_total_angle ();` function allows you to reset the counter

The function `get_create_total_angle ();` returns the recorded value (angle turned in degrees)

```
set_create_total_angle(0); //tells the Link to reset the angle turned to 0
```

```
get_create_total_angle(); // gets the angle turned in degrees
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

How accurate is the get_create_total_angle (); ?

Write a program for your robot to turn 180 degrees

Psuedocode (Task Analysis)

```
// 1. connect to create
// 2. Reset create total angle turned to 0
// 3. Turn left or counter clockwise
// 4. Pause program for 2 seconds to give
      the robot time to turn
// 5. stop the motors/create
// 6. print get_create_total_angle (); to
      screen
// 7. disconnect from create
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

TEACHER'S RESOURCE

Activity Solution

```c
// Created on Thu October 10 2013

int main()
{
   create_connect ();
   set_create_total_angle (0); // resets distance traveled to start at 0
   create_drive_direct (-200, 200); //turn left-counter clockwise
   msleep (2000); // sleep for 2 seconds to give the robot time to turn
   create_stop(); //stop the create motors
   printf("%i\n", get_create_total_angle ()); // prints the total angle turned to the screen
   create_disconnect (); //disconnects the link from the create
   return 0;
}
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Checking the total angle turned value returned with an actual measurement

TEACHER'S RESOURCE

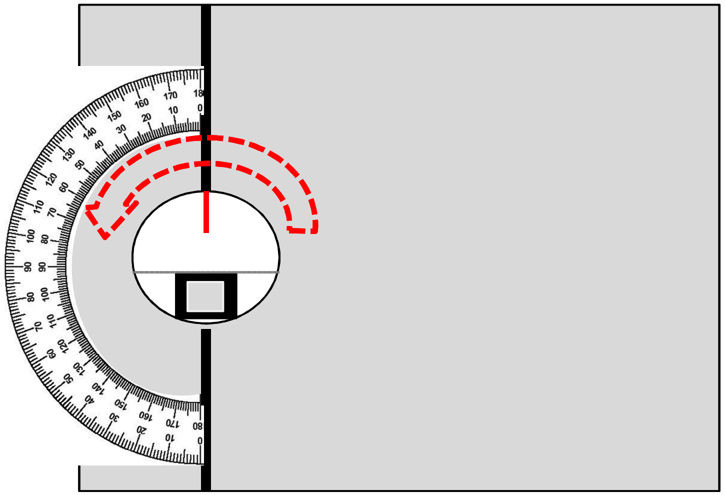
If you place the Create on a starting line and run the program in a perfect world it should turn ~180 degrees BUT this often isn't a perfect world

In reality it may go something less (friction, motors are different etc) or more

- In the previous example our Create printed out it had gone 180 degrees
- The actual physical measurement with a protractor was 182 degrees

With a little work, you can figure out the differences and account for this when programming the robot (a data table with conversions would be helpful)

* You many need to put a mark on the create so you know where to start (it is hard to keep this consistent without a mark)



KISS INSTITUTE FOR PRACTICAL ROBOTICS

Using the *Create* bump sensors

TEACHER'S RESOURCE

Goals

- To learn and use the functions for connecting to and moving the Create
- To learn the functions used to access sensors built into the Create
- To learn how to use the Create's bump sensors

Preparation

- You will need a charged Create and Link + the serial cable to connect them
- You will need computers with the KISS IDE
- You will need the USB download cable
- You need a solid object for the create to bump into
  - In Botball teams use the pvc around the edge of the game board to trigger the Create's bump sensors

Activity

Follow the slides

KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Activity 3

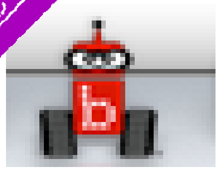
Lets use the Create's built-in sensors!

Use the Create with a Link controller in the cargo bay connected with a serial cable

KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

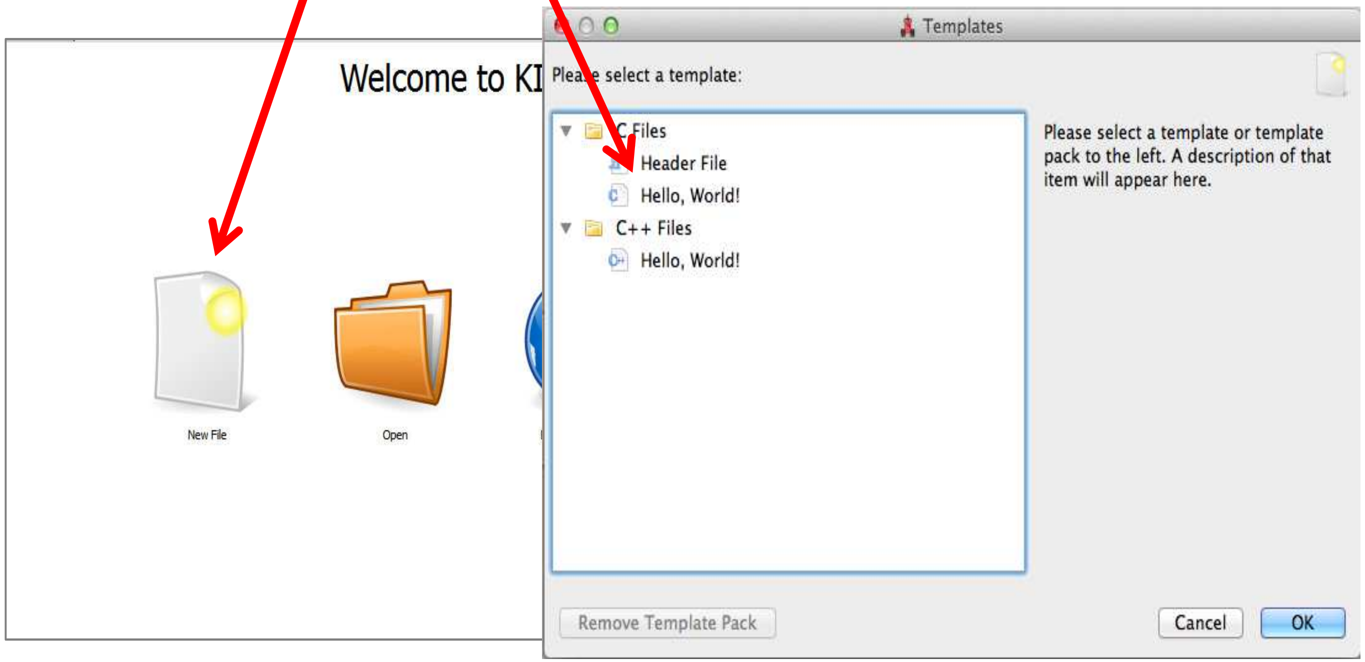
Launch the KISS IDE



KISS IDE icon

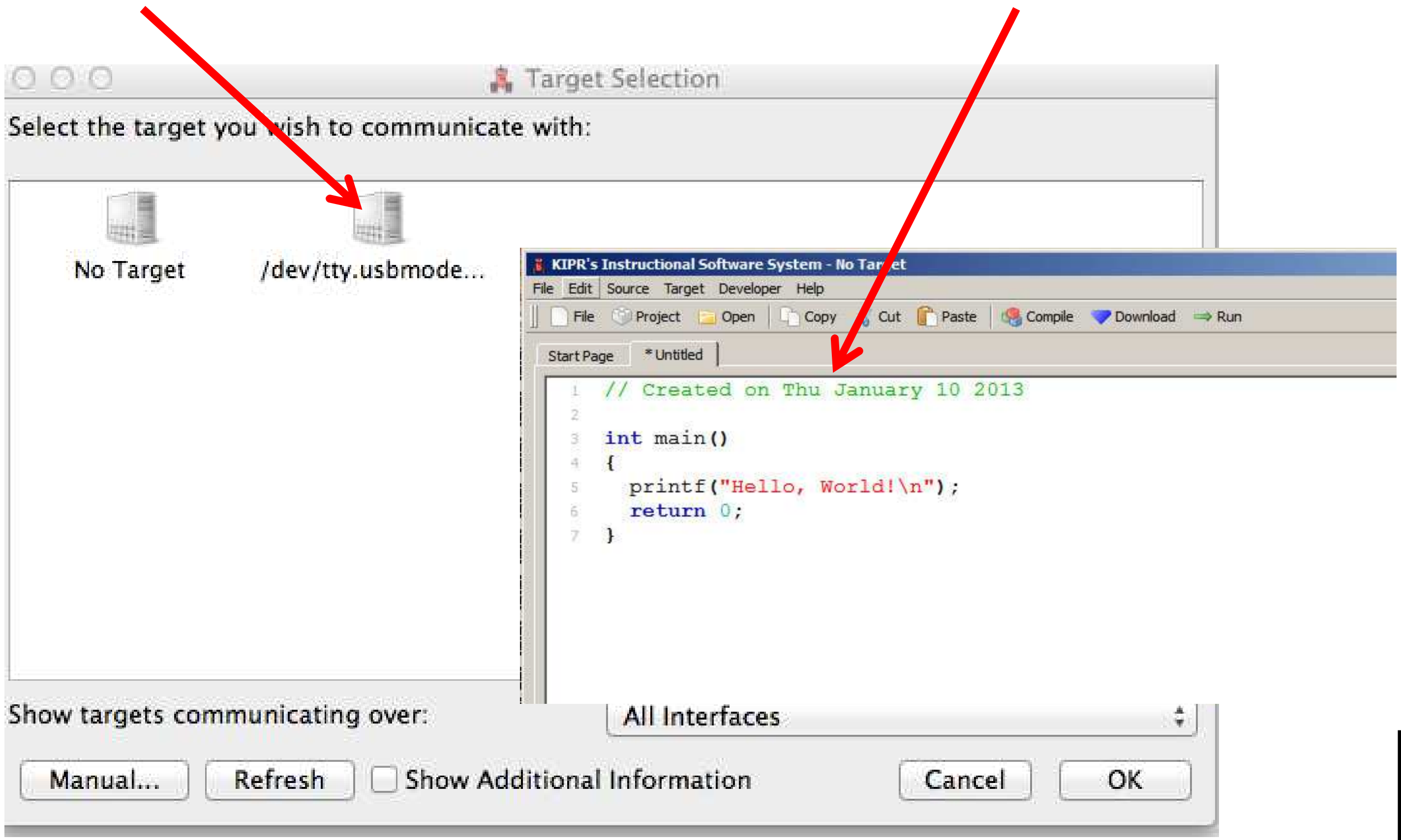
- Start the KISS IDE by clicking on its icon to get the welcome screen
- Click on the "*New File*" icon and

  and choose the C, "*Hello, World!*" template



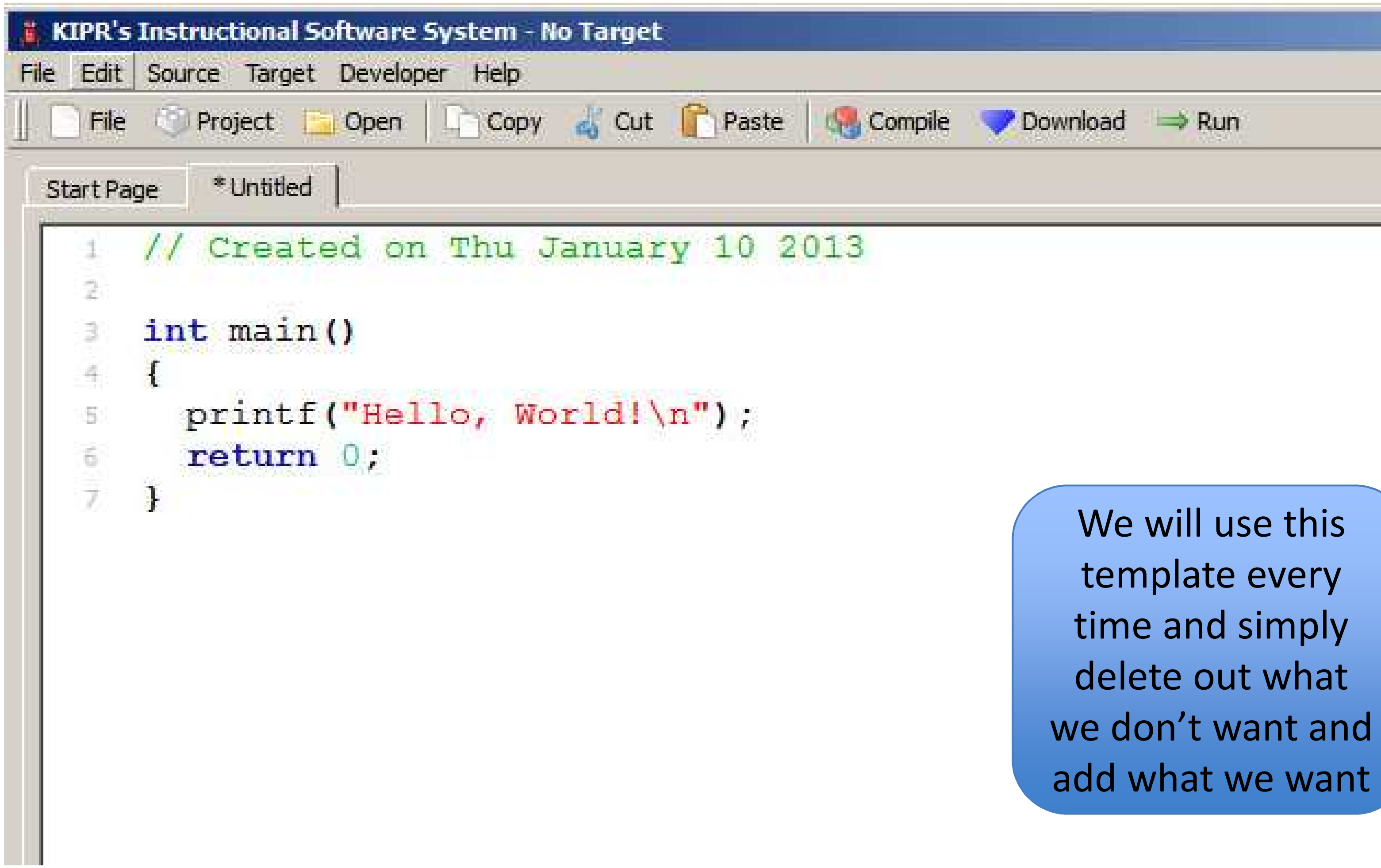
Select Target

- A Target Selection window will appear
- Pick *"USB Target"* and the C program template will come up



STUDENT ACTIVITY

The C Template: Hello, World!



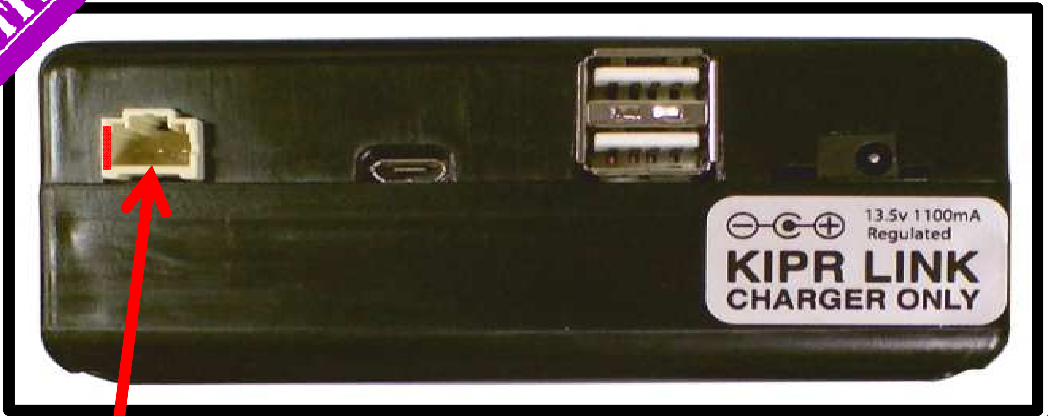
We will use this template every time and simply delete out what we don't want and add what we want

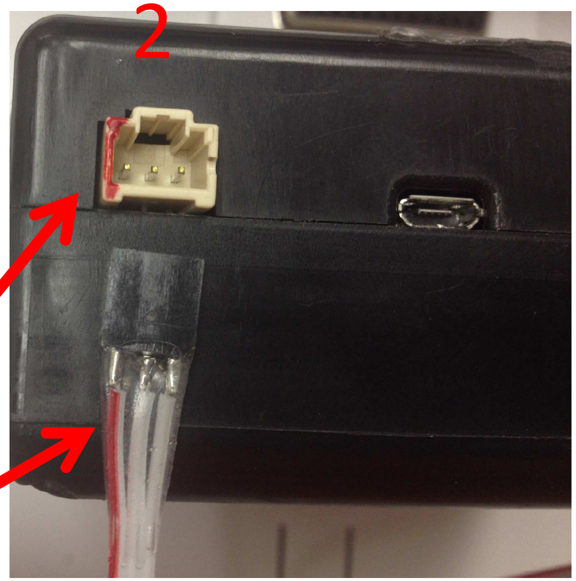
KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Plugging serial cable into Link

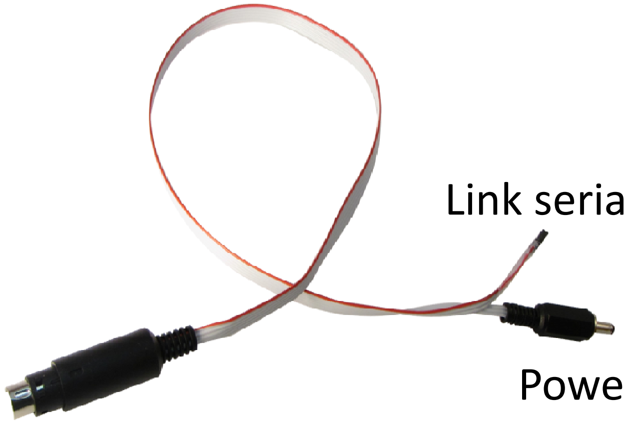
1

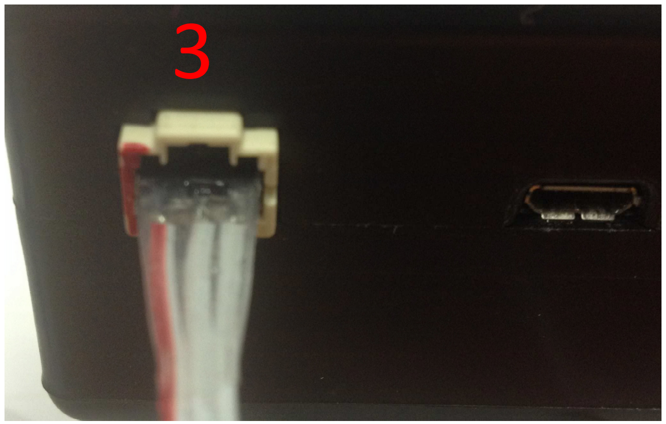




We use the serial cable to plug into the Link TTL Serial plug

NOTICE the red mark on the plug (left side) this corresponds to the red wire in the serial cable





Link serial plug

Power connector

Create connector

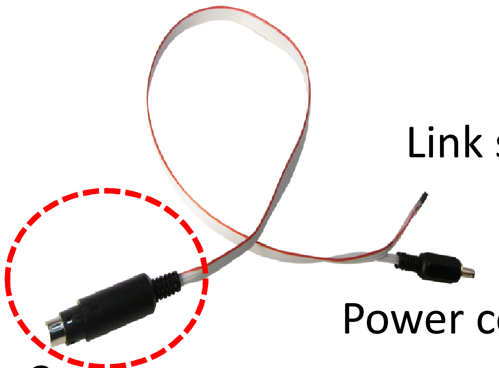
Correctly plugged in

KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Plugging serial cable into *Create*

We use the Create connector (round) end



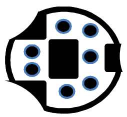
Link serial plug

Power connector

Create connector

Receptacle may have a cover that you can pop off to access

The plug is keyed, make sure you line it up correctly before plugging it in



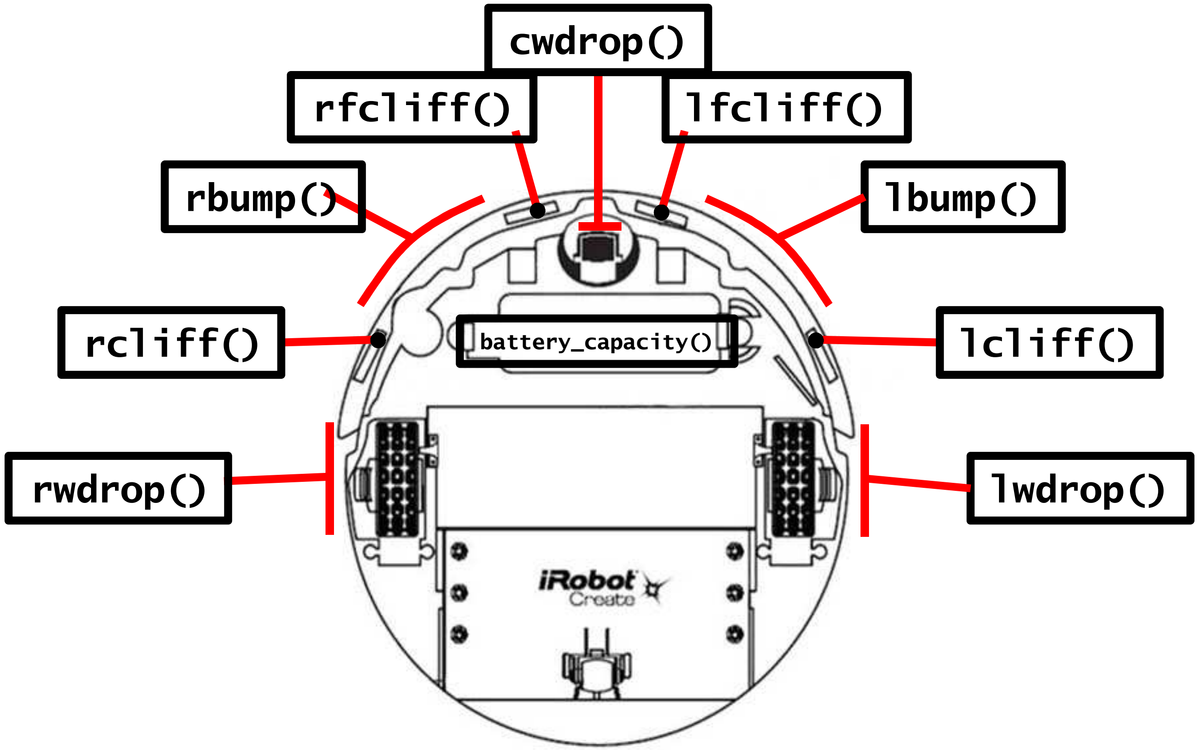
KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Function to *get* Create sensor values

By now you should be familiar with the
`get_create_total_angle ();` and
`get_create_distance ();` functions

The Create has other built-in sensors you can access with the **get_create_....();**



KISS INSTITUTE FOR PRACTICAL ROBOTICS

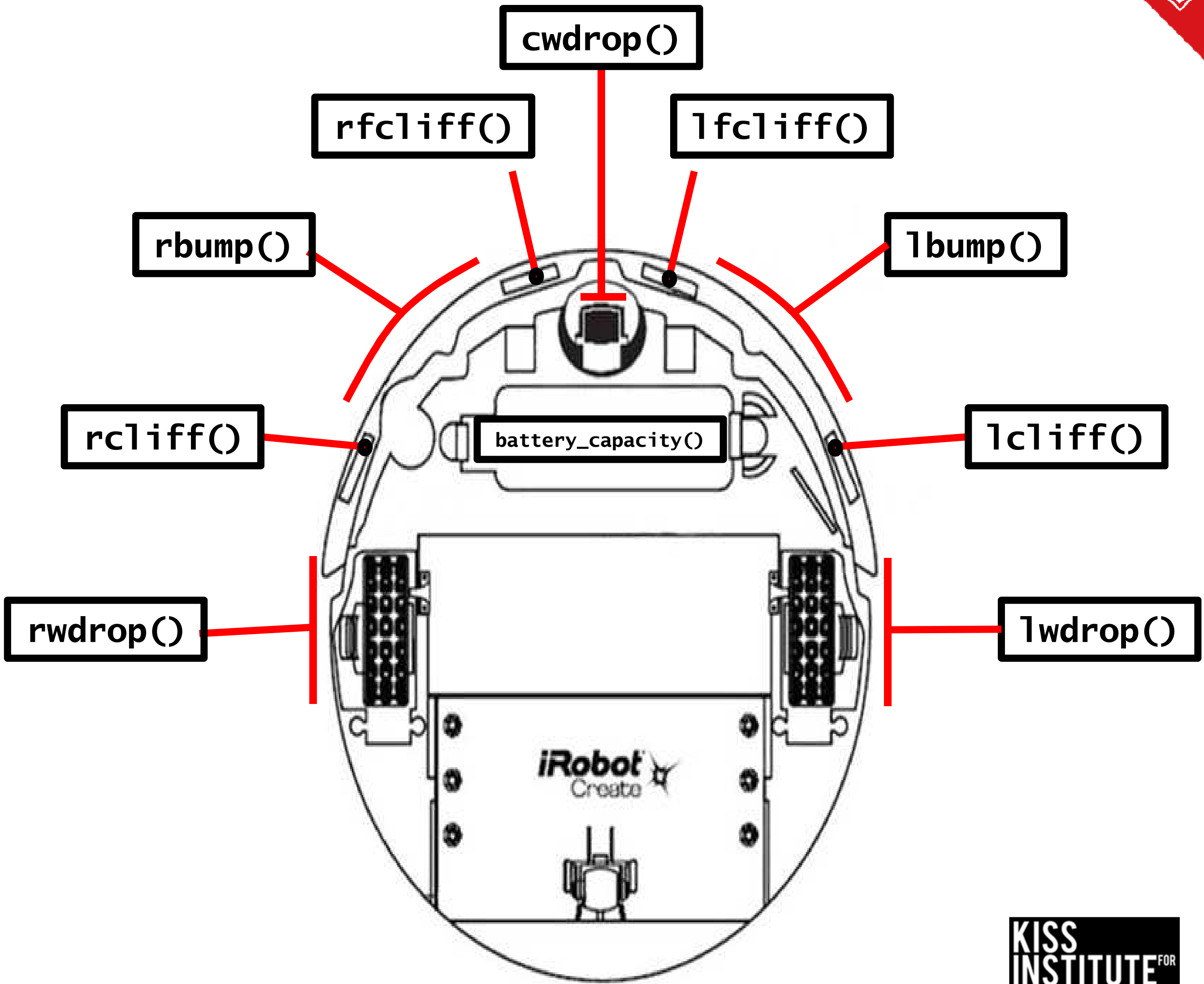
STUDENT ACTIVITY

TEACHER'S RESOURCE

right and left bump
(digital touch sensor)

4 cliff sensors
(IR reflectance)

3 drop sensors
(let you know when a wheel has dropped)



KISS INSTITUTE FOR PRACTICAL ROBOTICS

STUDENT ACTIVITY

Using a bump sensor?

Write a program for your robot to move forward until the right bump sensor is pressed

Psuedocode (Task Analysis)

```
// 1. connect to create
// 2. Check right bump sensor
// 3. Drive forward @ 300mm/second
// 4. Stop the motors/create when bumped
// 6. print “I hit something”
// 7. disconnect from create
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

Activity Solution

TEACHER'S RESOURCE

```
// Created on Thu October 10 2013

int main()
{
  create_connect (); // connect the Link to the Create
  while (get_create_rbump () == 0) // check the right bump sensor
  {// If not hit
  create_drive_direct (300, 300); //move forward
  }//If hit
  create_stop(); //stop the create motors
  printf("I hit something\n"); // prints the total angle turned to the screen
  create_disconnect (); //disconnects the link from the create
  return 0;
}
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS