Getting to know the iRobot Create

CSSE 120—Rose Hulman Institute of Technology

Show Off Some Animations

- Who would like me to show off their work?
- Otherwise I'll pick some programs at random

Reviewing some concepts you read

- *Functions*
  - Named sequences of statements
  - Can *invoke* them—make them run
  - Can take *parameters*—changeable parts

Parts of a Function Definition

```
>>> def hello():
      print "Hello"
      print "I'd like to complain about this parrot"

```

Defining a function called "hello"

Indenting tells interpreter that these lines are part of the hello function

Blank line tells interpreter that we're done defining the hello function

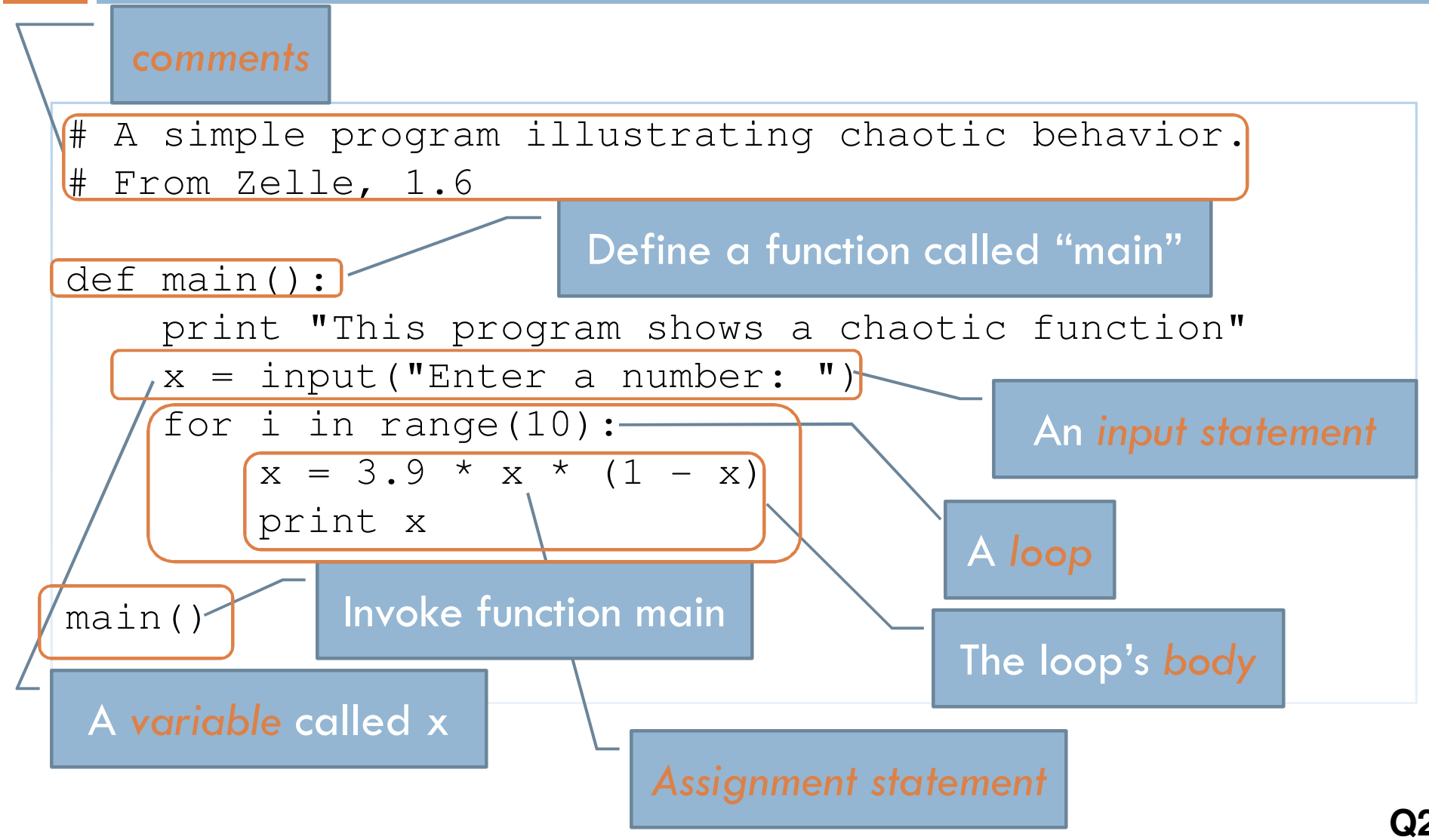
Defining vs. Invoking

- Defining a function says what the function should do
- Invoking a function makes that happen
  - Parentheses tell interpreter to invoke the function

```
>>> hello()
Hello
I'd like to complain about this parrot
```

  - Later we'll define functions with parameters

A simple program that defines and invokes a function called main()



Q2

Questions?

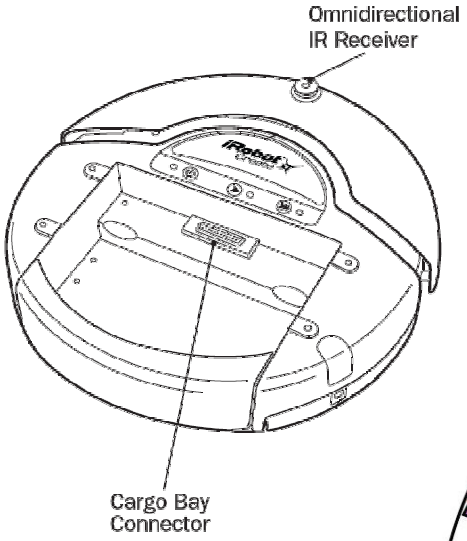
- Your reading should be enough for you to do the homework due Friday:
  - Modifications to chaos program
  - A program to calculate the distance between 2 points
- Questions?

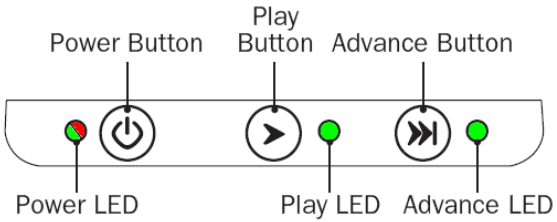
- Up next: Robots!

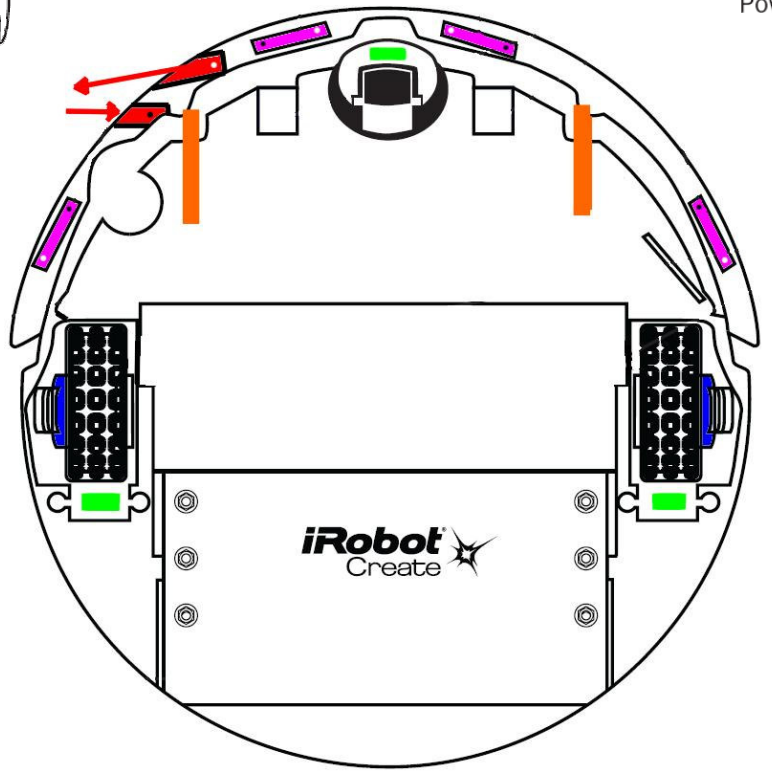
Your iRobot Create

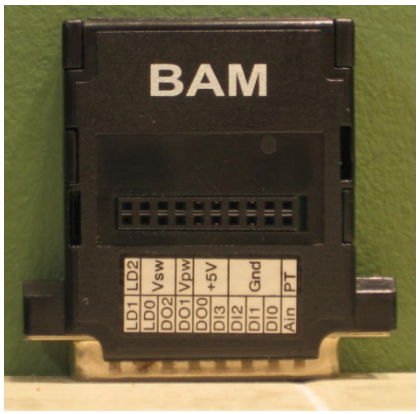
- I'll give each pair a locker number and combination.
- Robots are in the lockers, currently sitting on the dock (2 green lights on dock) and should be returned to the dock at the end of the class to keep them charged.
- Please have **one** person get your robot from your locker.
- We'll get names from the other partner.

Getting to know the iRobot Create









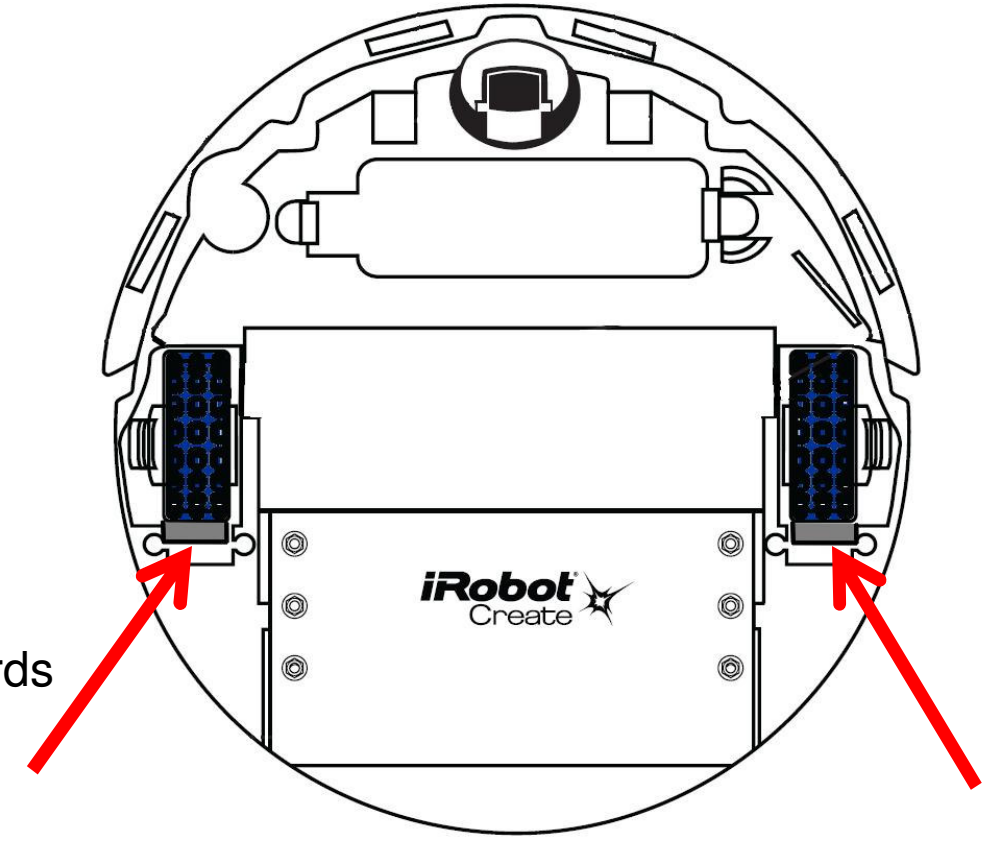
Look at your iRobot Create as we go!

Getting our hands on iRobot Create

- iRobot Create hardware overview
  - Actuators
  - Sensors
- Making a COM port connection over Bluetooth
- iRobot Create's Open Interface Protocol
  - Sending serial commands via RealTerm
  - Sending serial commands via Python
- Using the create.py module!
  - Way Easier! Way Better!

iRobot Actuators – Robot Outputs

- Left Wheel Motor
- Right Wheel Motor

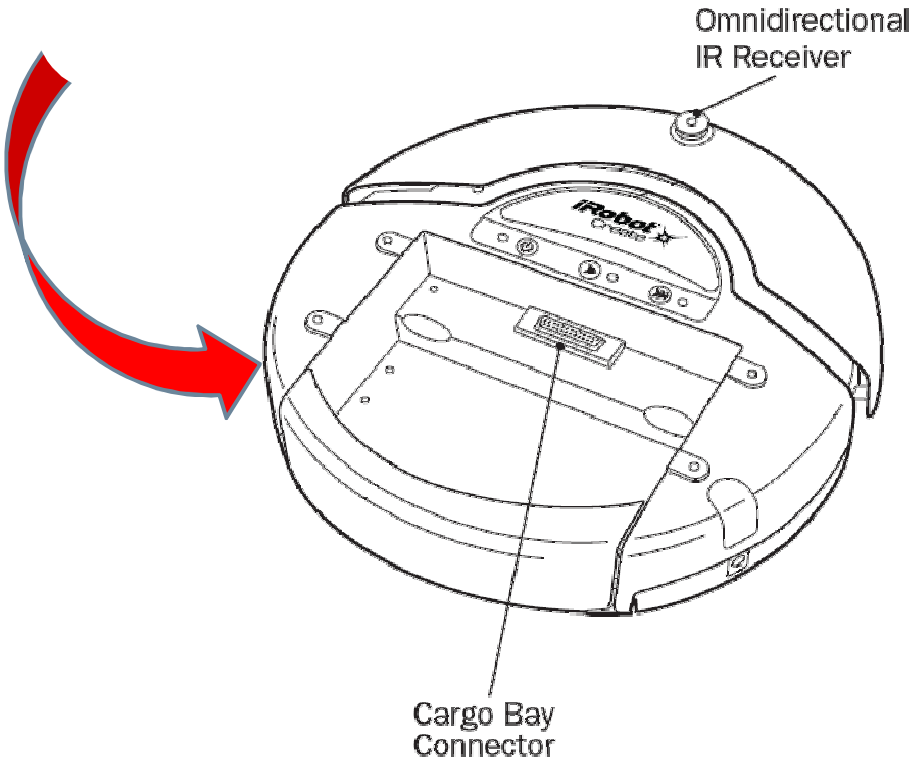


Max speed sets the wheels to 500 mm/s forwards or backwards

That's just over 1 mph so don't get too excited about 500 mm/s

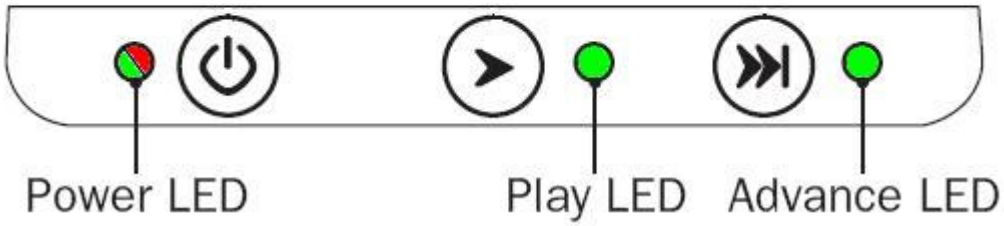
iRobot Actuators – Robot Outputs

- Left Wheel Motor
- Right Wheel Motor
- Speaker



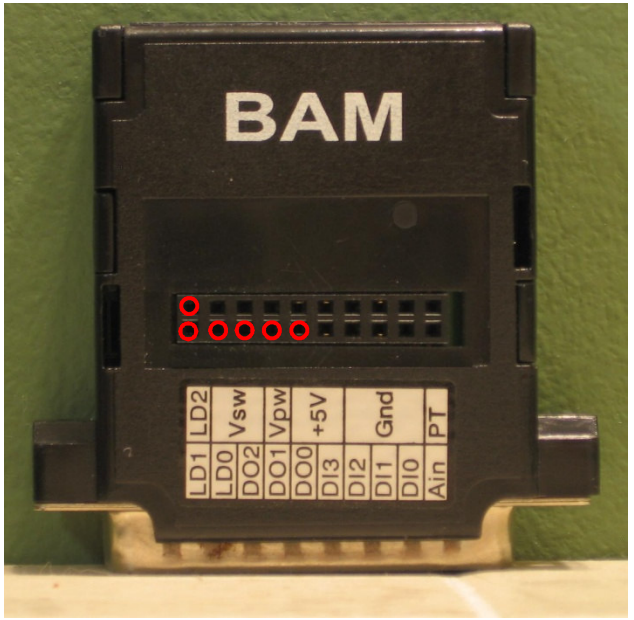
iRobot Actuators – Robot Outputs

- Left Wheel Motor
- Right Wheel Motor
- Speaker
- Bi-color Power LED
- Play LED
- Advance LED



iRobot Actuators – Robot Outputs

- Left Wheel Motor
- Right Wheel Motor
- Speaker
- Bi-color Power LED
- Play LED
- Advance LED

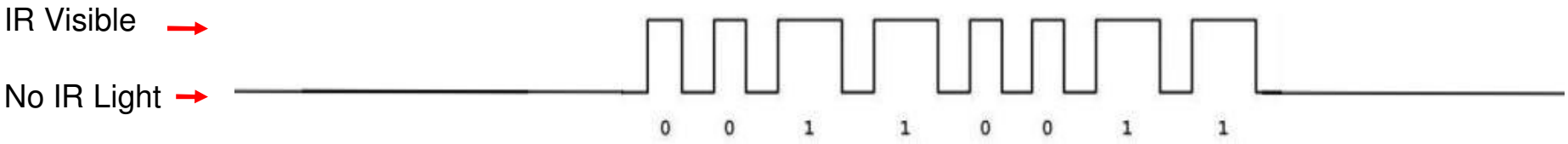


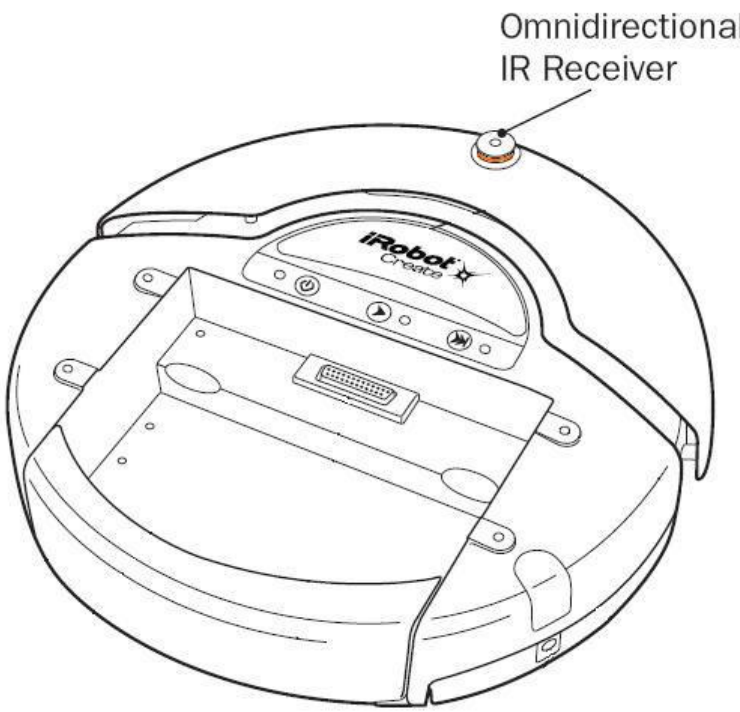
- Low-side Drivers on the BAM (LD0-LD2)
- Digital Outputs on the BAM (DO0-DO2)

iRobot Sensors – Robot Inputs

- Omnidirectional IR Sensor
- Play and Advance Buttons
- Left and Right Bumpers
- Three Wheel Drop Sensors
- Four Cliff Sensors
- Wall Sensor
- Encoders
- Four Digital Inputs on the BAM (DI0-DI3)
- Analog Input on the BAM (A_{in})

Omnidirectional IR Receiver



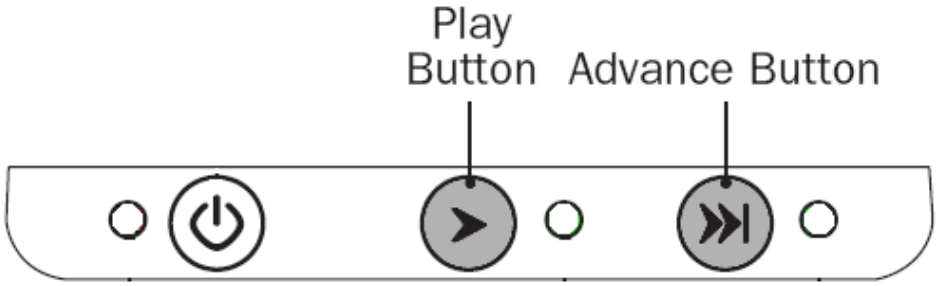


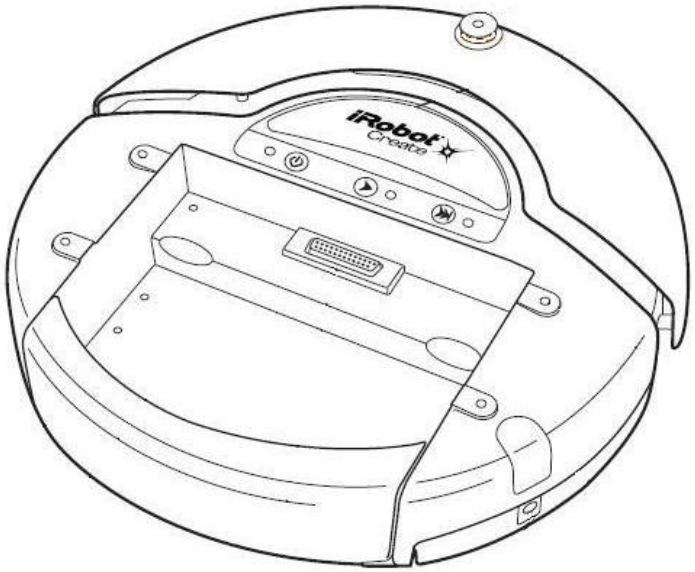
IR receive shown here = 0b00110011
= 0x33
= 51

IR transmitters will flash out certain patterns to send 8-bit numbers

Values 0 to 254 (255 is for no signal)

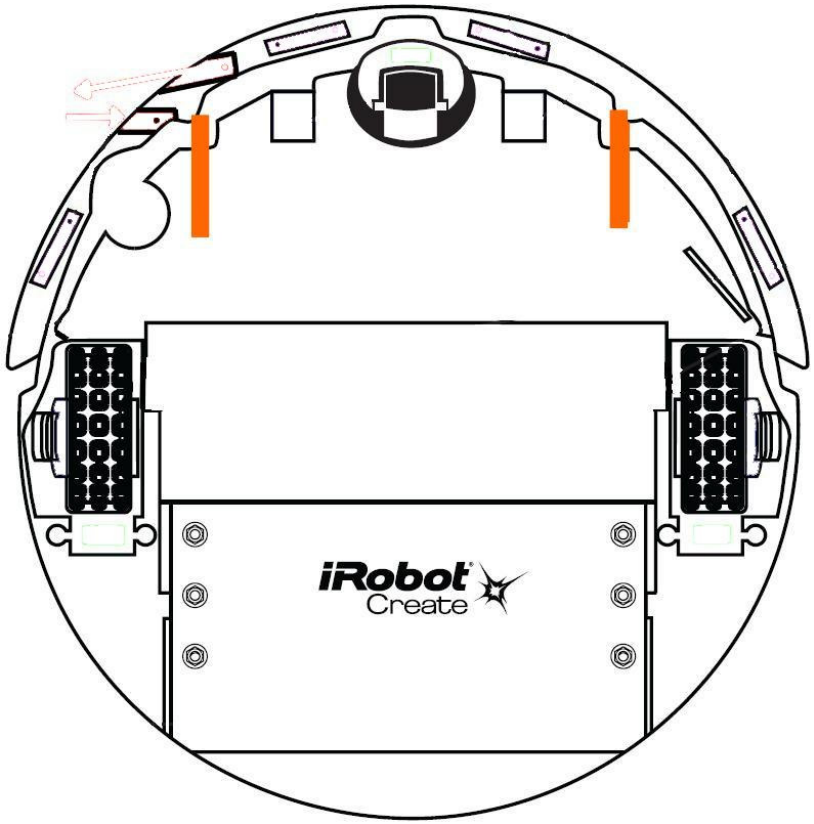
Play and Advance Buttons





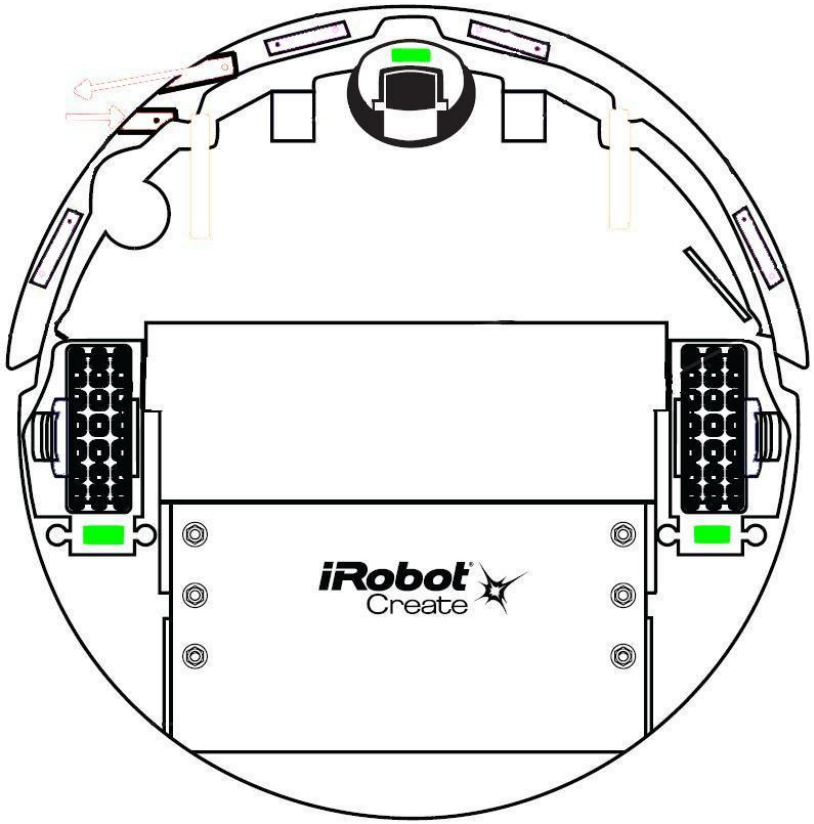
- Digital inputs that you could really use for any function
- They just have symbols on them. Nothing special about that symbol

Bump Sensors



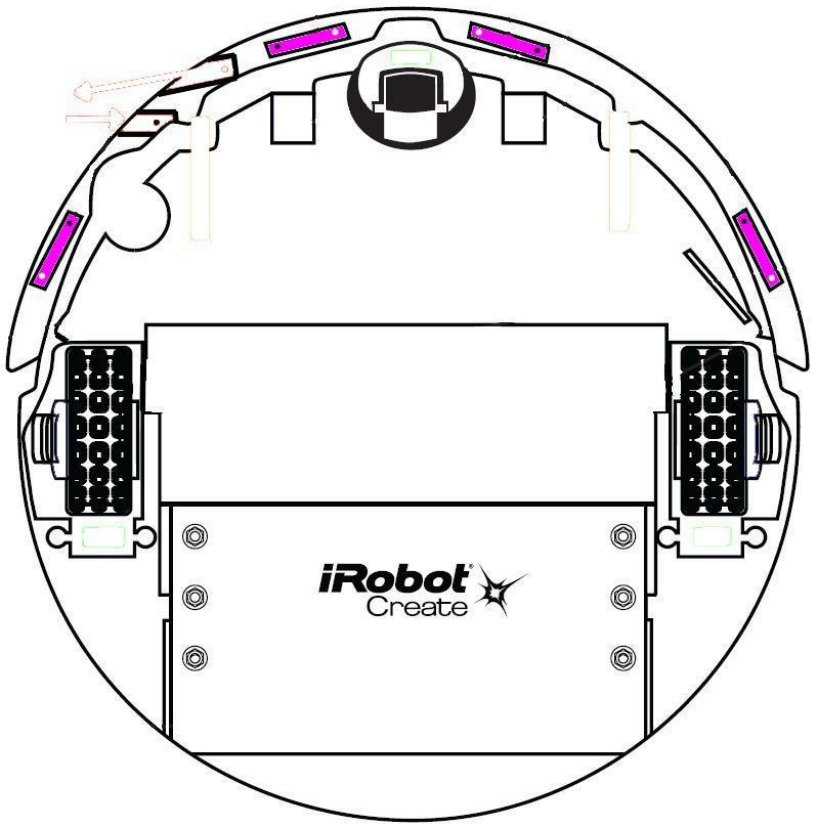
- Two digital signals
- Left Bumper
- Right Bumper

Wheel Drop Sensors



- Three digital inputs
- Front Wheel Drop
- Left Wheel Drop
- Right Wheel Drop

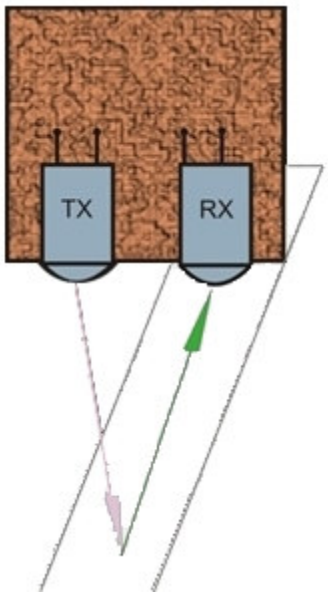
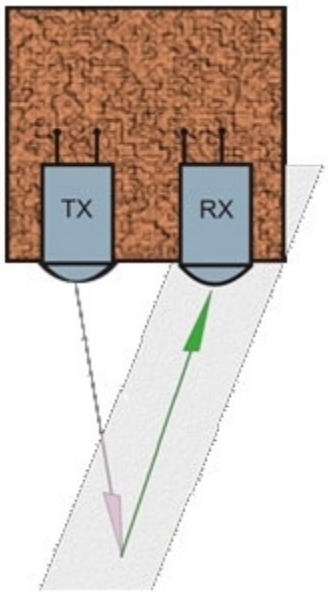
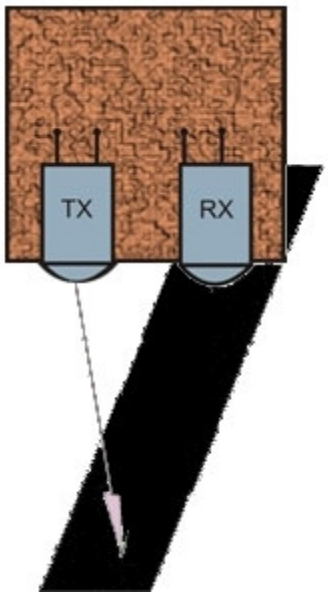
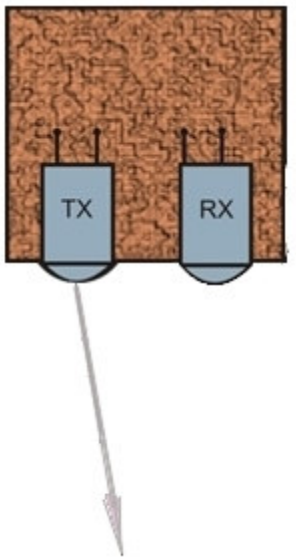
Cliff Sensors



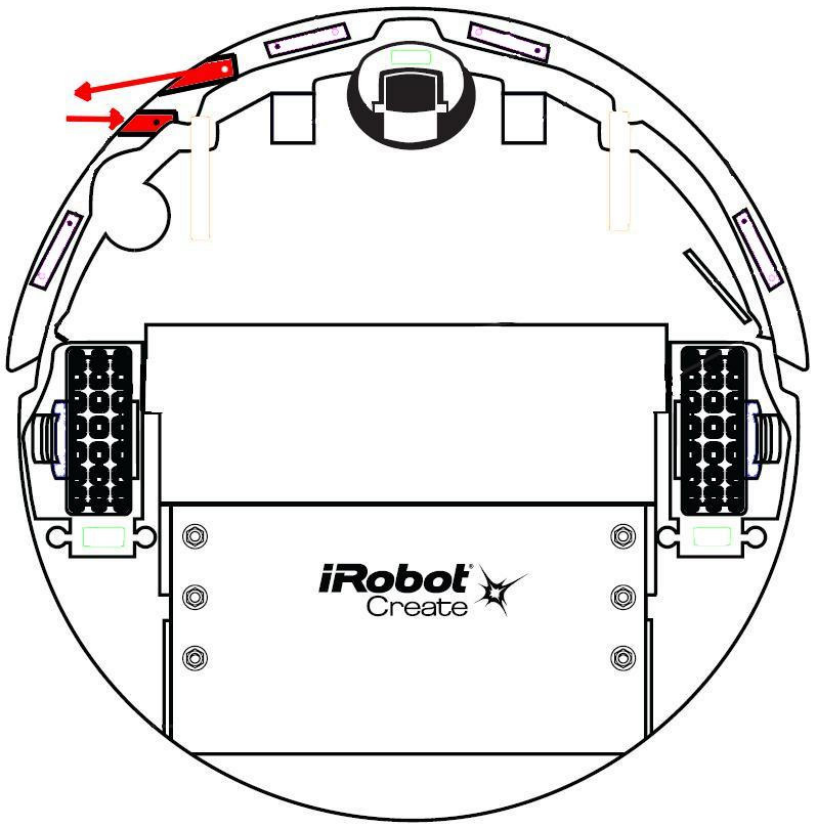
- Four analog inputs

- Cliff Left Signal
- Cliff Front Left Signal
- Cliff Front Right Signal
- Cliff Right Signal

Cliff Sensor Analog Readings

White Surface	Gray Surface	Black Surface	No Surface
			
High value Max = 4095	Medium value	Low value Min = 0	Low value Min = 0
Common real values: 1800	1000	0	0

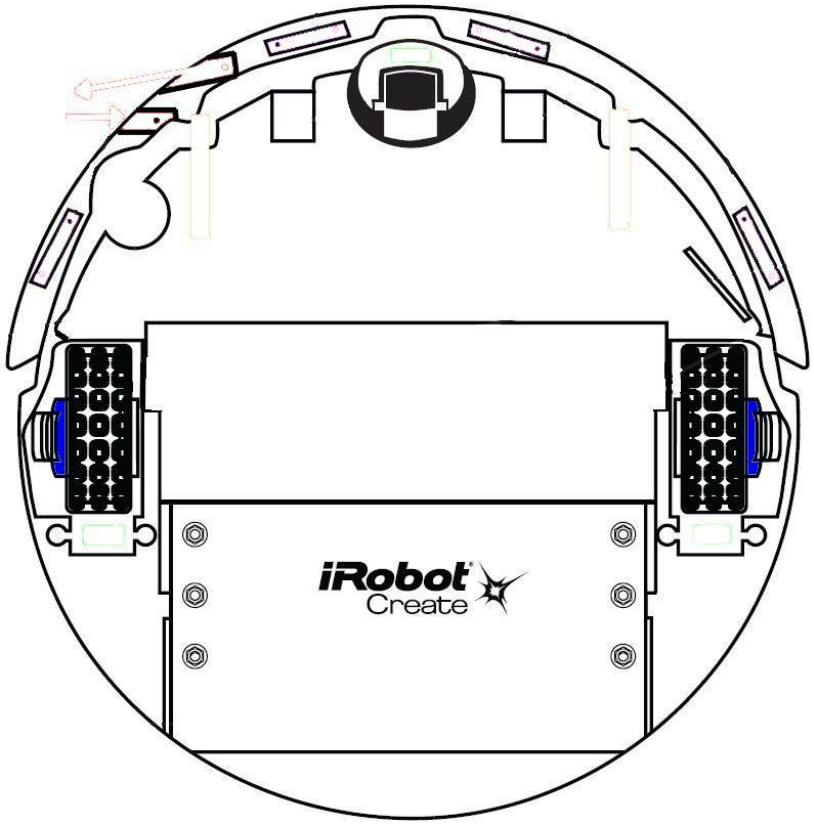
Wall Sensor



- **One** Analog Sensor
- Value relates to the distance between wall and Create

  0 = No wall seen

Wheel Encoders

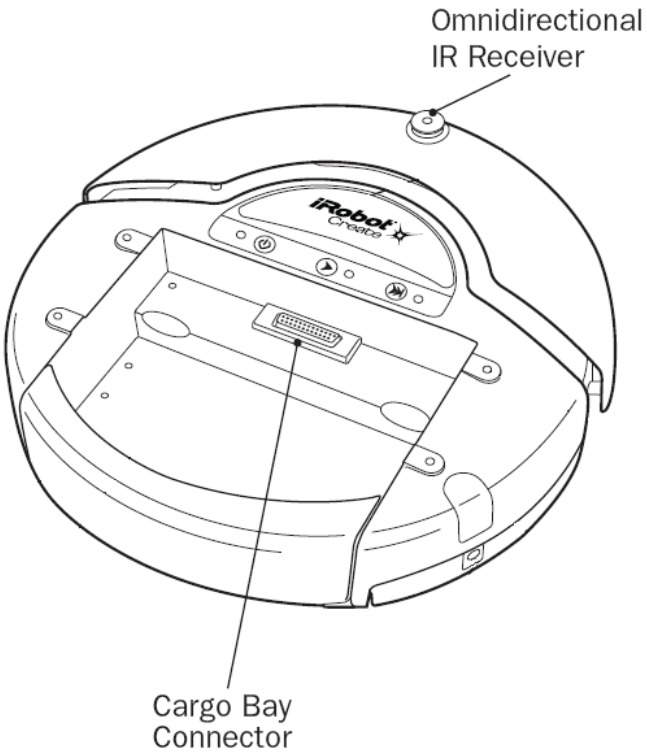


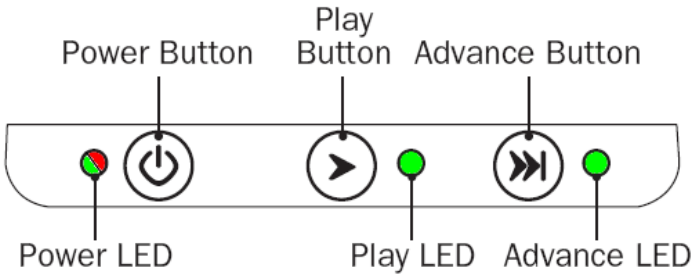
- More complex
- Distance since last request
- Angle since last request
- Used internally to control wheel speed

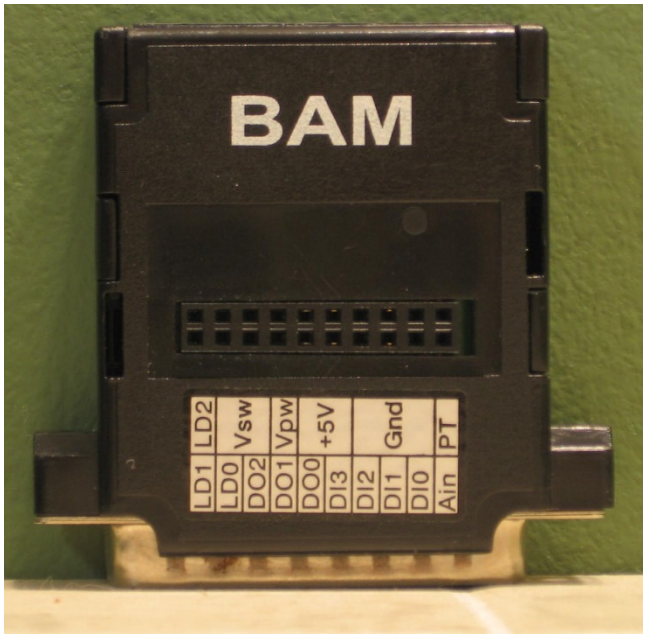
Inputs on the BAM

- Four Digital Inputs on the BAM (DI0-DI3)
- Analog Input on the BAM (A_{in})

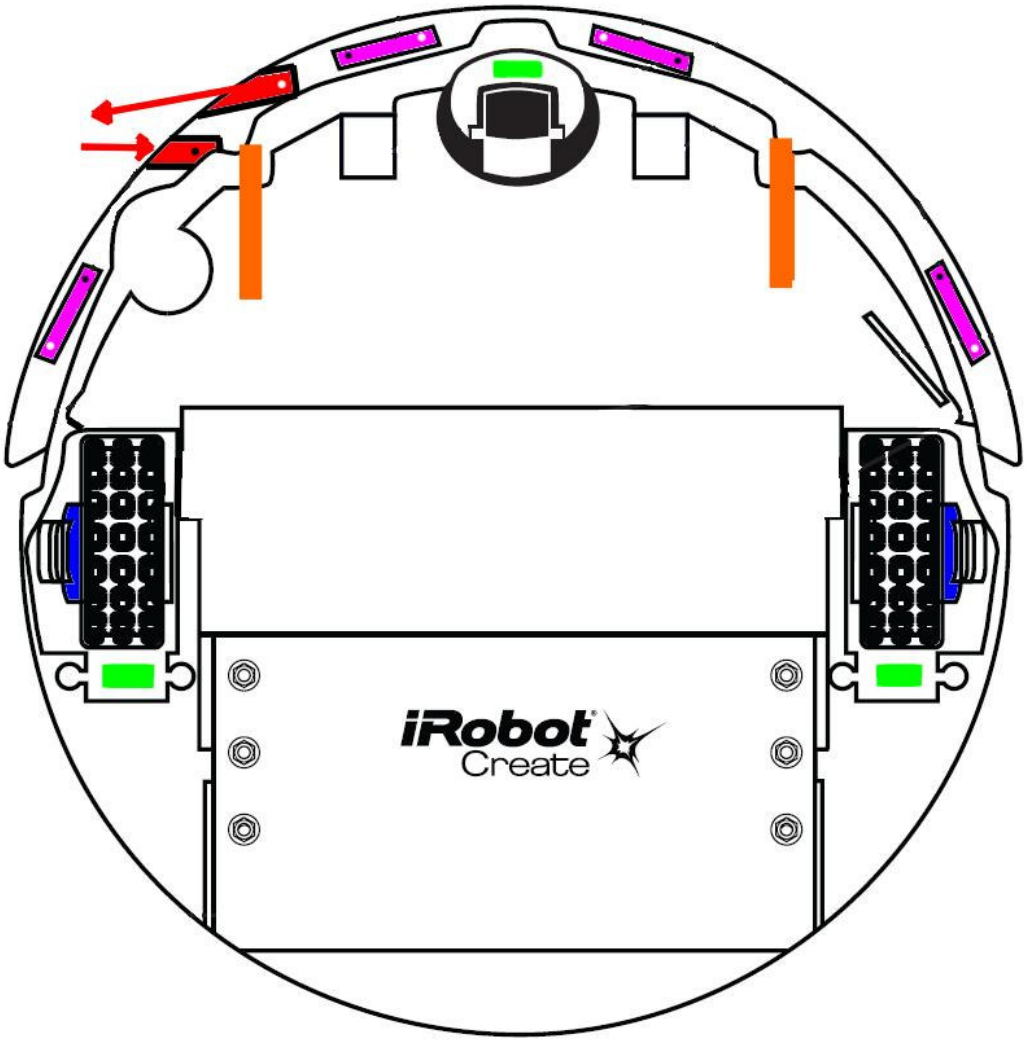
iRobot Create Top View





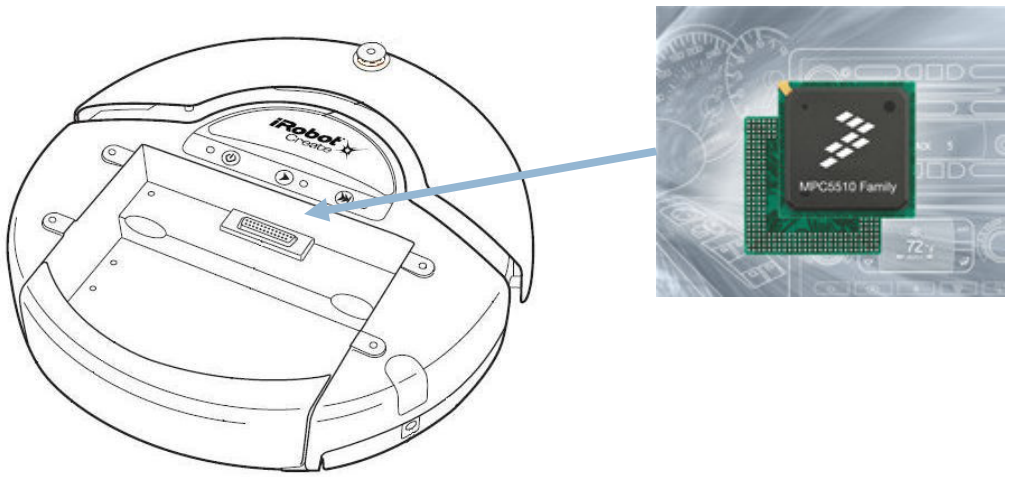


iRobot Create Bottom View

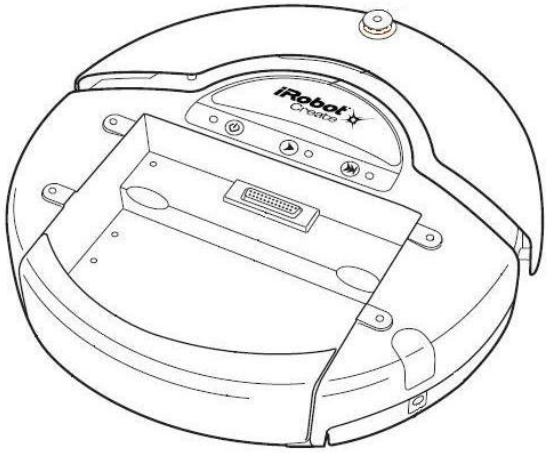


Getting our hands on iRobot Create

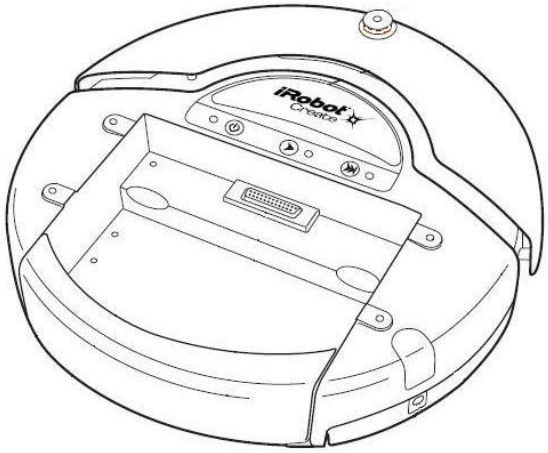
- iRobot Create hardware overview
  - Actuators
  - Sensors
- Sensor signals go to the iRobot microcontroller
- But? The signals need to get to the computer?

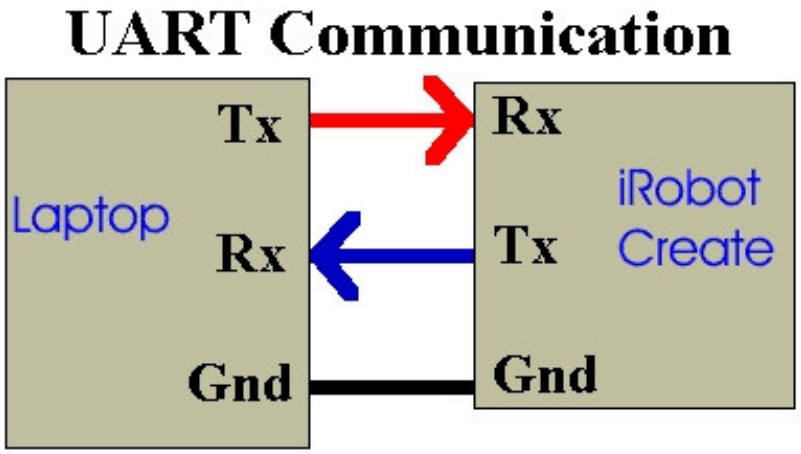


How do we get this information to a PC?



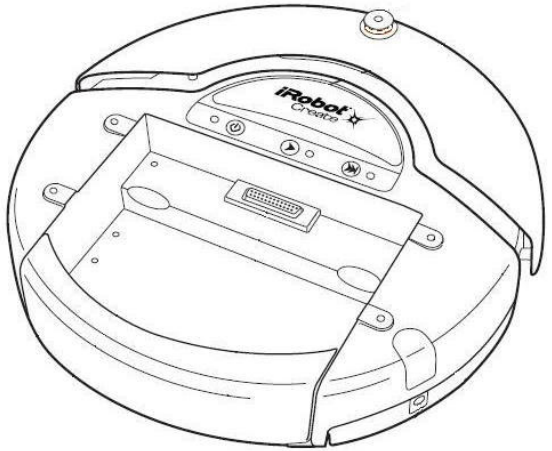
UART Communication



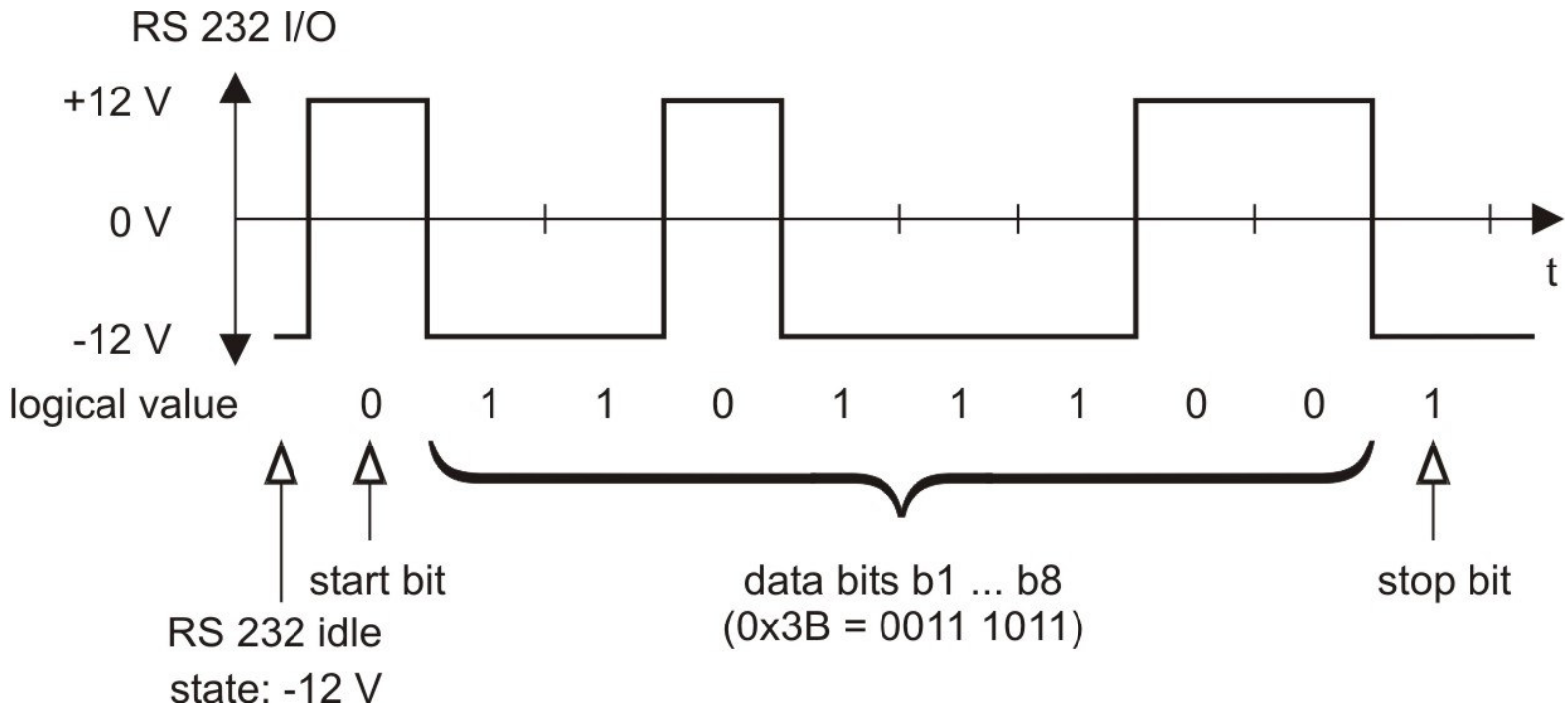


Universal Asynchronous Receiver / Transmitter

Example UART Basics



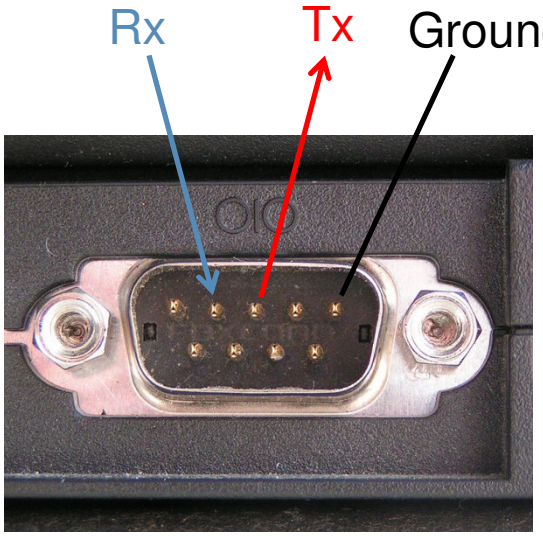
A quick detailed look at UART



Message at predetermined bit rate (baud rate) iRobot uses 57600 bits/second

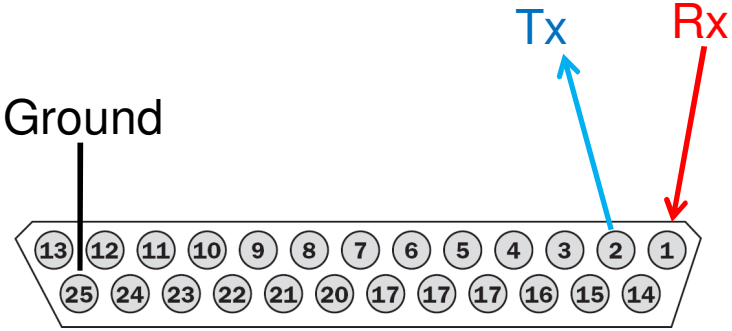
How does UART work?

- Usually (or maybe we should say previously) UART is/was connected via an RS232 port, also known as a DB9 Serial Port, or just called, more simply, a "Serial Port"



Laptop Serial Port


Serial Cable



Pin	Name	Description
1	RXD	0 – 5V Serial input to Create
2	TXD	0 – 5V Serial output from Create
25	GND	Create battery ground

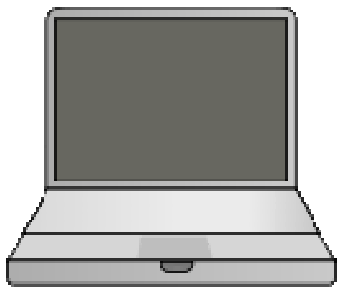
iRobot 25 pin Serial Port

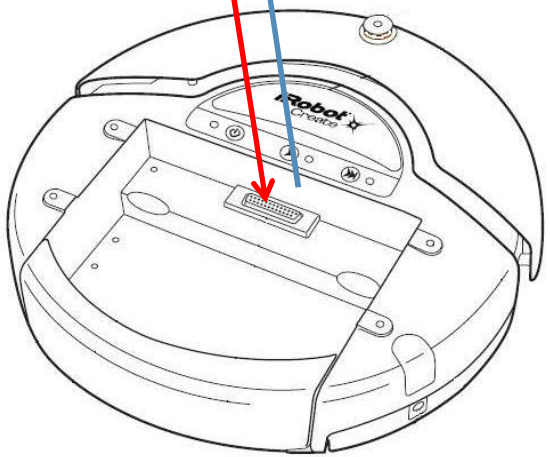
From Society of Robots website – "Let me say this bluntly - no cute girl would ever date you if you have a robot with a long wire dragging behind it. Just that simple."

Wireless Bluetooth using the BAM!

Bluetooth Hardware Inside Laptop Using Bluetooth Driver

Bluetooth "Virtual" Wires





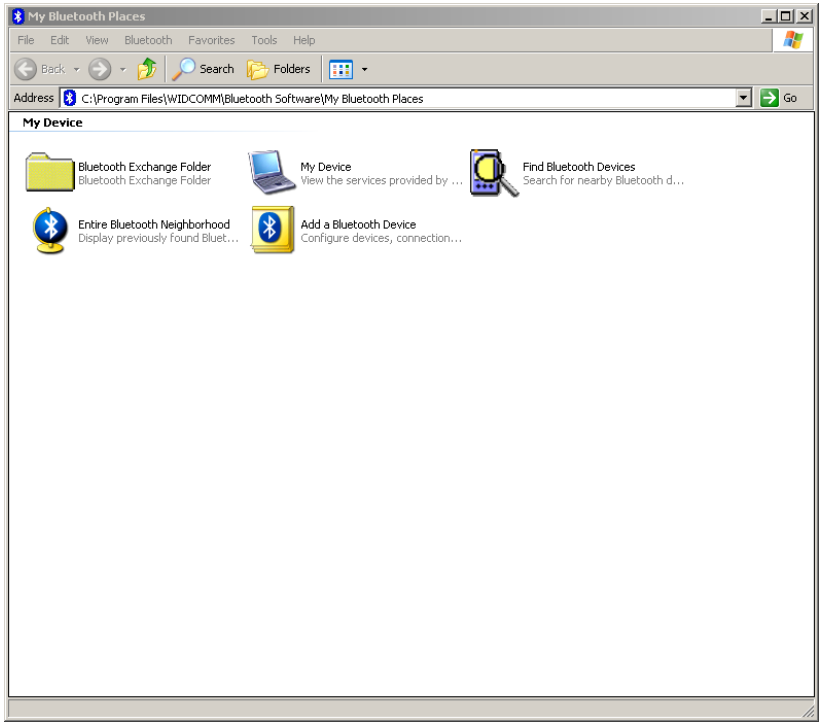
BAM = Bluetooth Access Module

Communication Protocol

- iRobot sets the rules for communication
  - iRobot store website http://store.irobot.com
- Learn and practice the UART commands
  - Click on Educational… then Manuals
    - Owner's Guide
    - Open Interface Specifications
    - RealTerm
- Let's start with RealTerm
  - Sends UART messages over a COM port

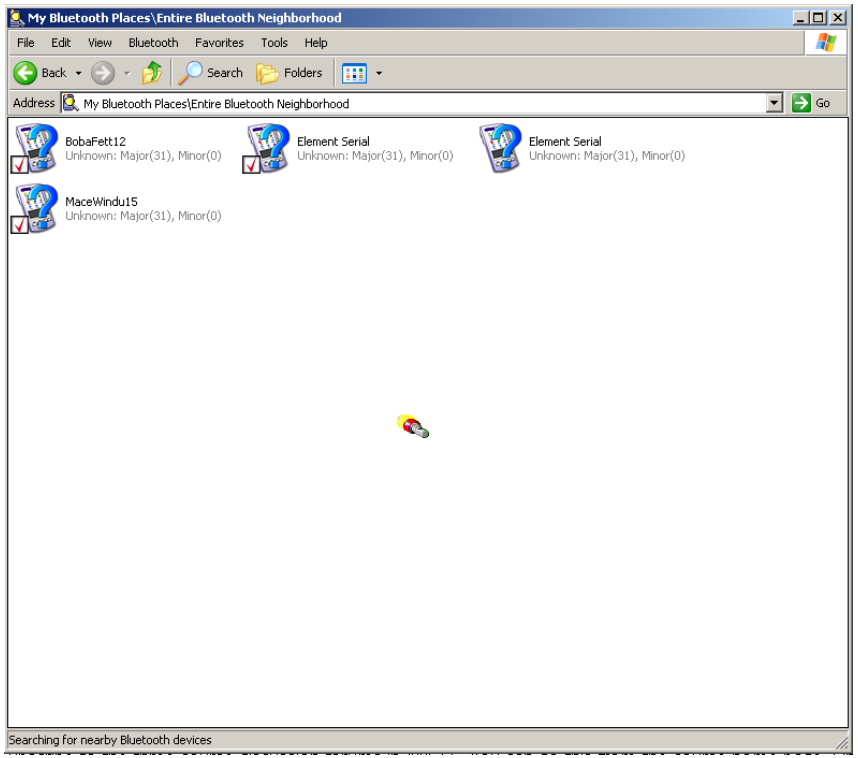
Finding your Bluetooth device

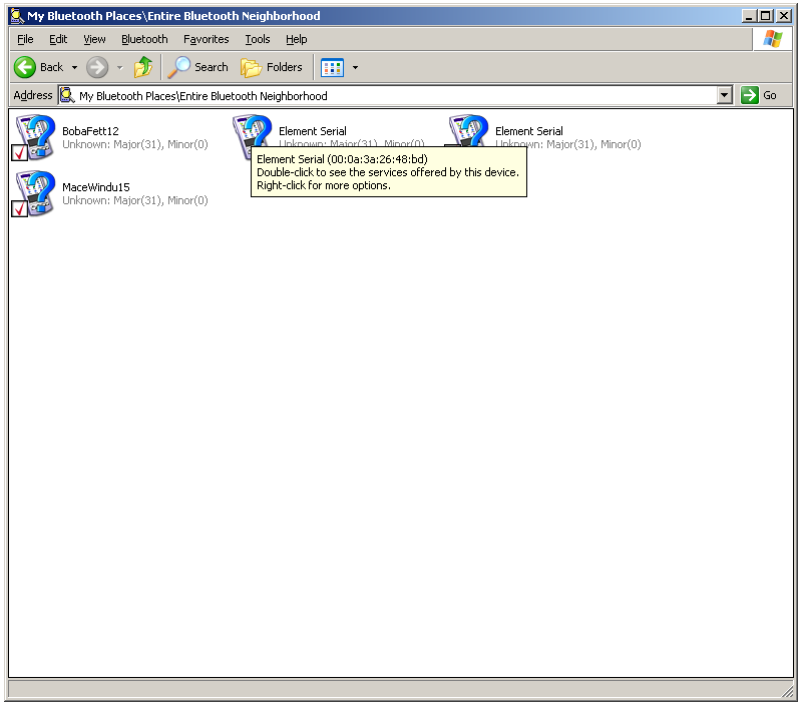
- If you installed the Bluetooth driver from HW1 you should have a Bluetooth icon…
- Double click on that icon to bring up “My Bluetooth Places”
- Turn on Robot
- Double click on “Find Bluetooth Devices"



Finding YOUR iRobot Create BAM

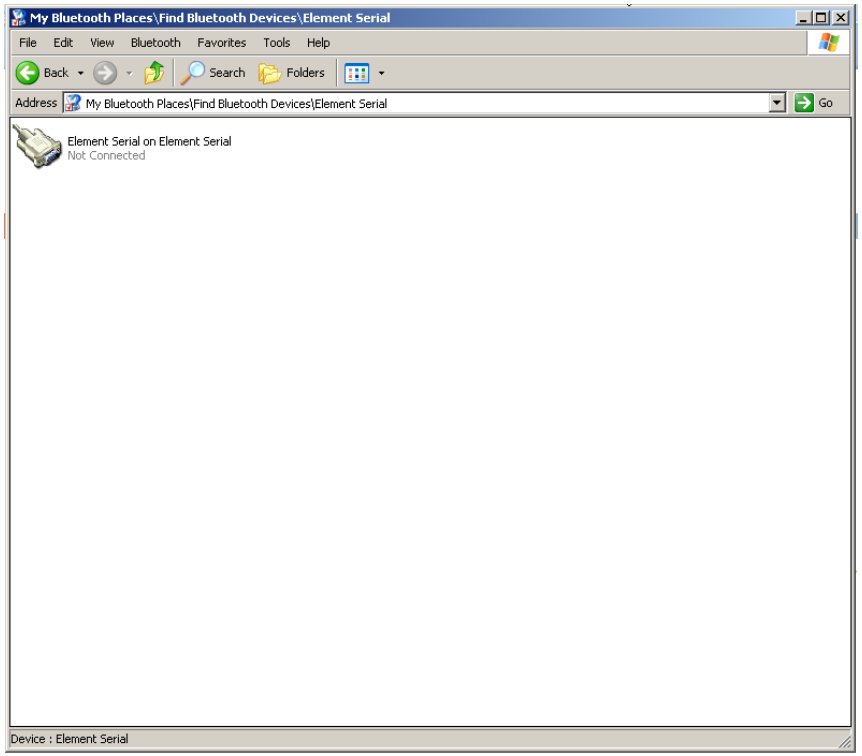
- Let your computer find all devices
- Hover your mouse over an “Element Serial”
- Find the number that matches the one printed on your BAM





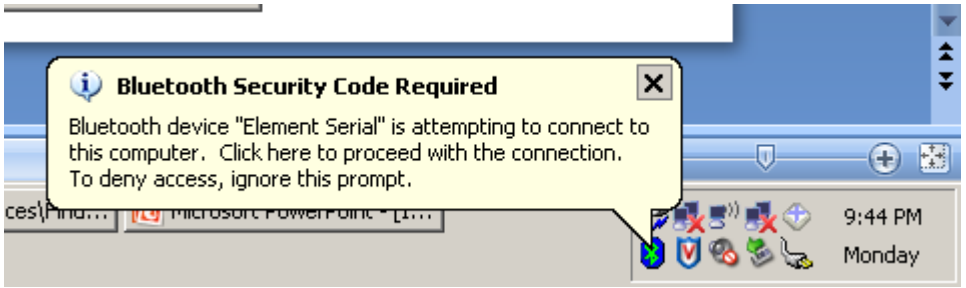
Connect to Element Serial

- Double click on the "Element Serial on Element Serial" to start the Bluetooth pairing process



Entering the Bluetooth security code

- You should have a window pop up that says “You need to enter a super top secret security code for this Bluetooth Device
- Click in that window
- (if you are too slow and the window goes away, click on the, now green, Bluetooth symbol directly)



This slide is Top Secret

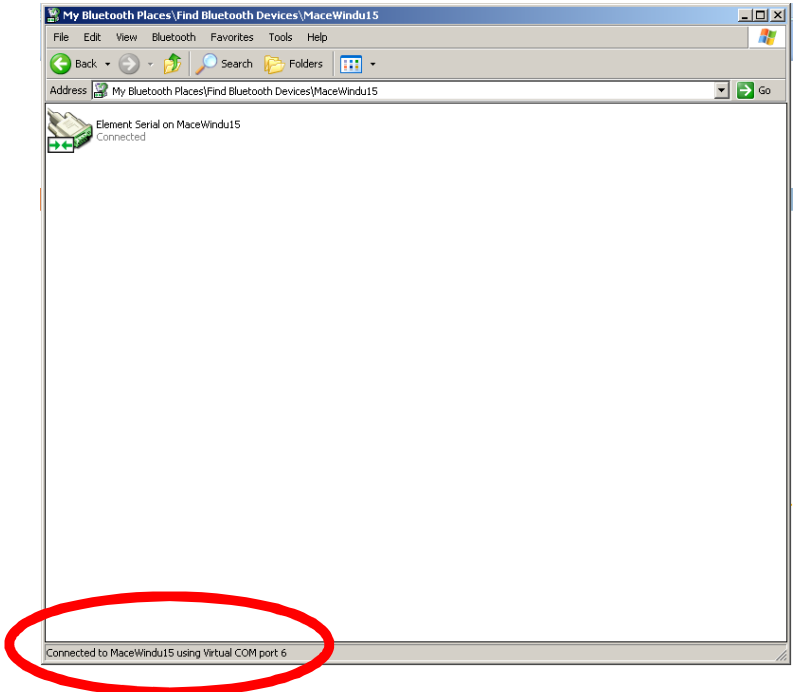
- The top secret code for this Bluetooth device is the number…

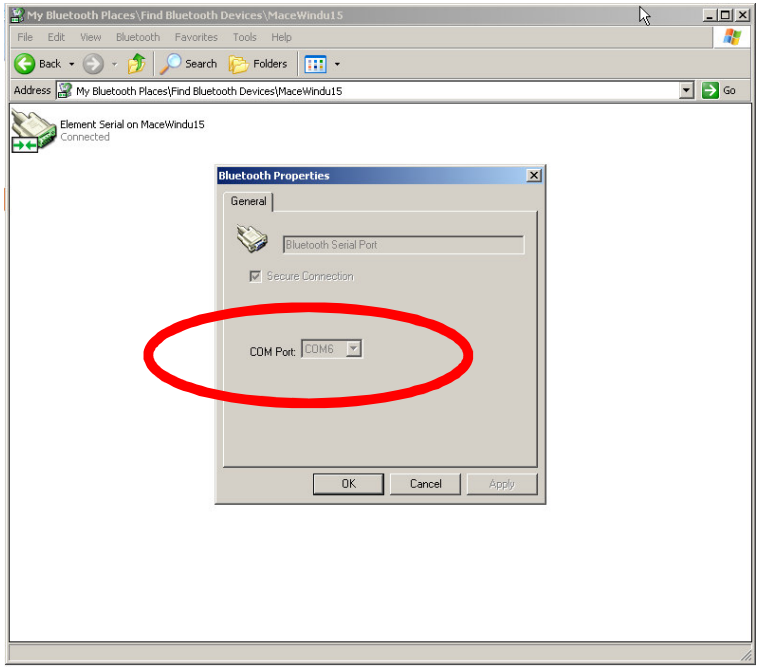
0000

Turns out every Bluetooth device that doesn't care about security is just four zeroes

Remember the COM port

- You should now be connected to the BAM!
  - It's just like two virtual wires Rx and Tx for serial communication to the iRobot Create
- You need to know the COM port #
  - It's at the very bottom of the window (mine is COM port 6)
- (if you are too slow you can always right-click and open the Bluetooth Properties)

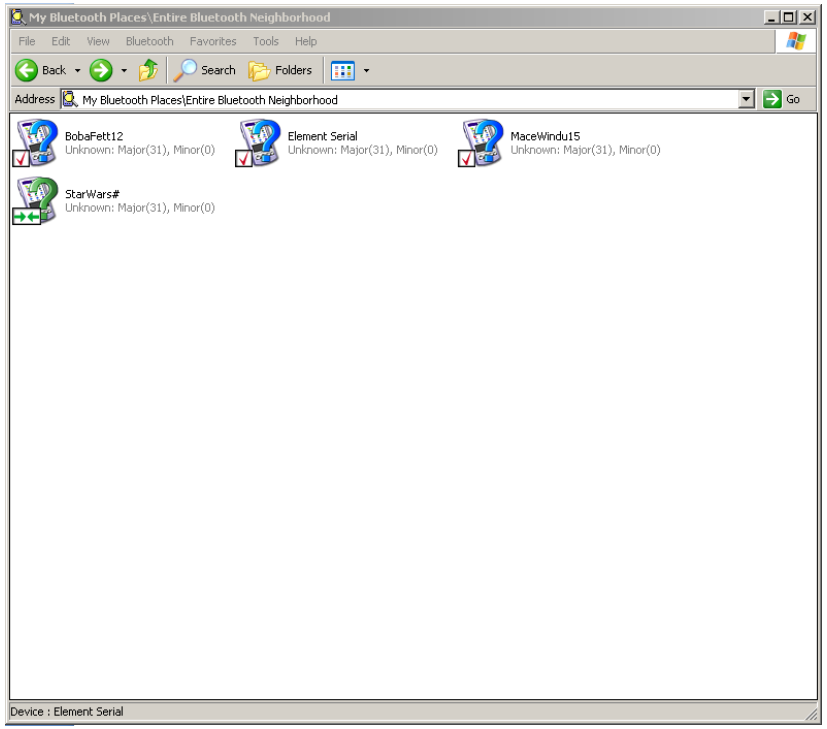


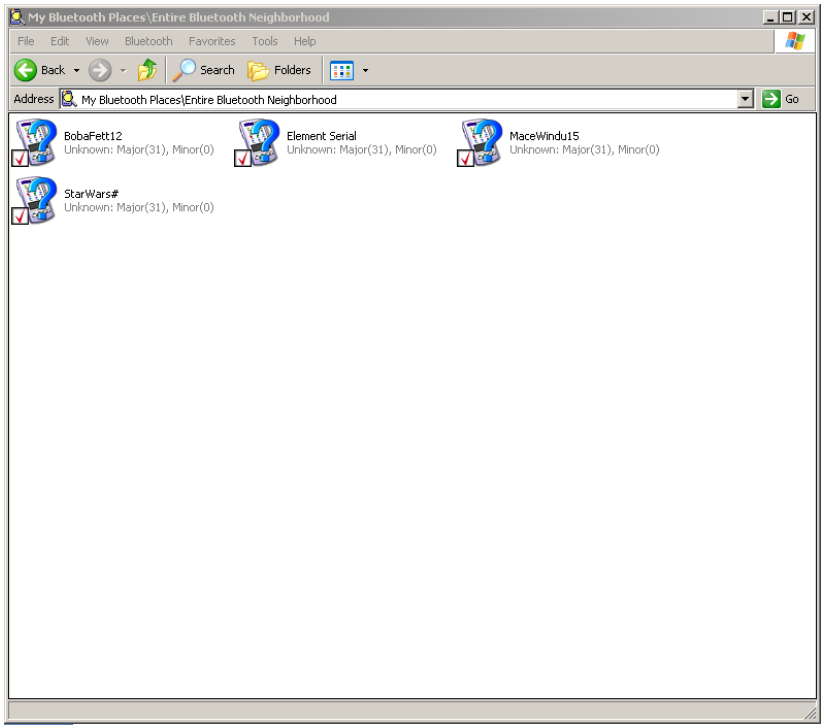


Q6

(optional) Renaming your Element Serial

- If you want you can rename your “Element Serial” to a name that is easier to find in your Bluetooth Neighborhood. I called this one “StarWars#”
- This only effects your computer, it doesn’t change anything on the BAM
- Then next time you “Find Devices” it’s easier to spot in the list





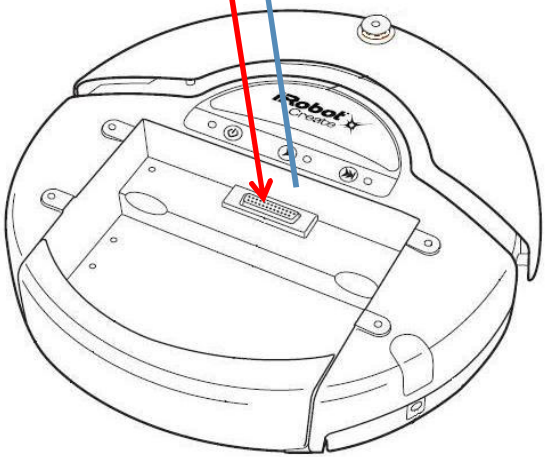
What did we just do?

Bluetooth Hardware Inside Laptop Using Bluetooth Driver

Bluetooth "Virtual" Wires

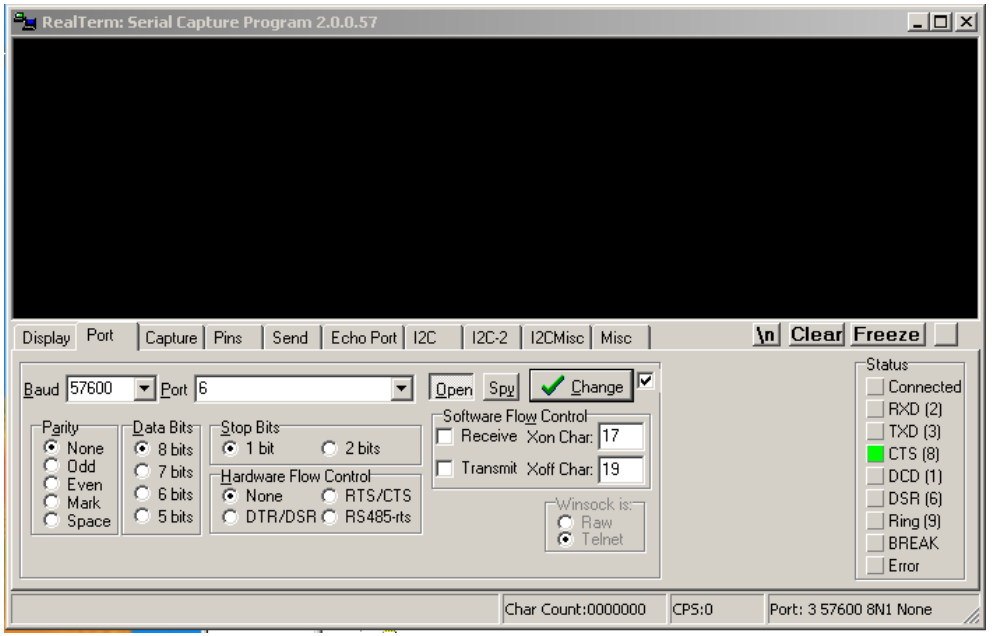
We connected the Rx and Tx wires… wirelessly!

Now we're ready to send data!



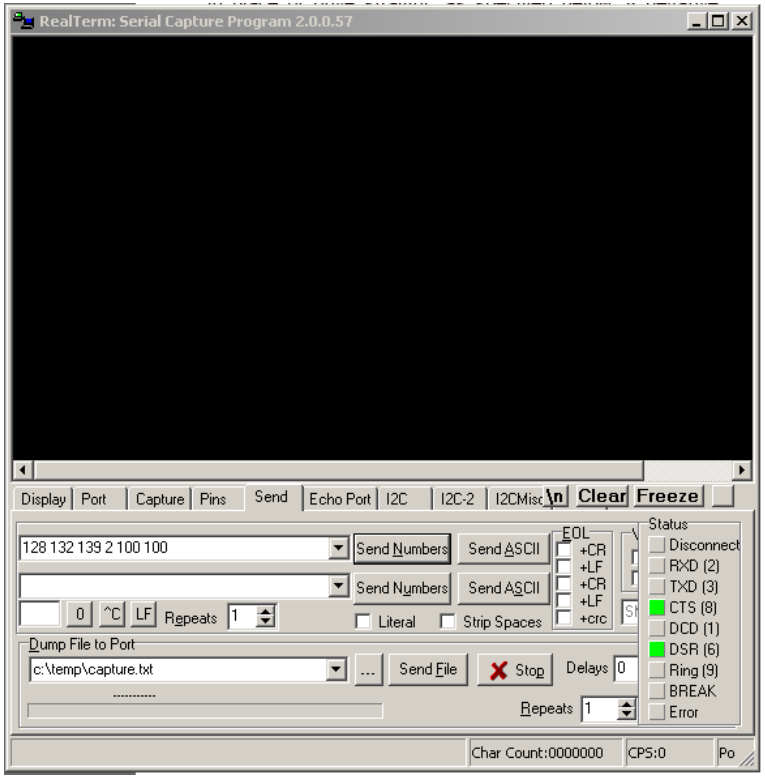
BAM = Bluetooth Access Module

- Open RealTerm
- Port tab
- Set Port to your COM port #
- Set Baud Rate to 57600 bits/second
- Click the Change button to make it actually happen



Send an LED command

- Turn on the Play LED with an amber Power LED
- Type 128 132 139 2 100 100
- Click "Send numbers"
- Watch!



LEDs **Opcode: 139** **Data Bytes: 3**

This command controls the LEDs on Create. The state of the Play and Advance LEDs is specified by two bits in the first data byte. The power LED is specified by two data bytes: one for the color and the other for the intensity.

- Serial sequence: [139] [LED Bits] [Power Color] [Power Intensity]
- Available in modes: Safe or Full
- Changes mode to: No Change
- LEDs data byte 1: LED Bits (0 – 10)

Advance and Play use green LEDs. 0 = off, 1 = on

Bit	7	6	5	4	3	2	1	0
LED	n/a	n/a	n/a	n/a	Advance	n/a	Play	n/a

Power uses a bicolor (red/green) LED. The intensity and color of this LED can be controlled with 8-bit resolution.

- LEDs data byte 2: Power LED Color (0 – 255) 0 = green, 255 = red. Intermediate values are intermediate colors (orange, yellow, etc).
- LEDs data byte 3: Power LED Intensity (0 – 255) 0 = off, 255 = full intensity. Intermediate values are intermediate intensities.

Example:

To turn on the Advance LED and light the Power LED green at half intensity, send the serial byte sequence [139] [8] [0] [128].

Digital Sensor Readings

- Let's request bumper and cliff sensor data

Bumps and Wheel Drops **Packet ID: 7** **Data Bytes: 1**
unsigned

The state of the bumper (0 = no bump, 1 = bump) and wheel drop sensors (0 = wheel raised, 1 = wheel dropped) are sent as individual bits.

Range: 0 - 31

Bit	7	6	5	4	3	2	1	0
Sensor	n/a	n/a	n/a	Wheeldrop Caster	Wheeldrop Left	Wheeldrop Right	Bump Left	Bump Right

Wall **Packet ID: 8** **Data Bytes: 1**
unsigned

The state of the wall sensor is sent as a 1 bit value (0 = no wall, 1 = wall seen).

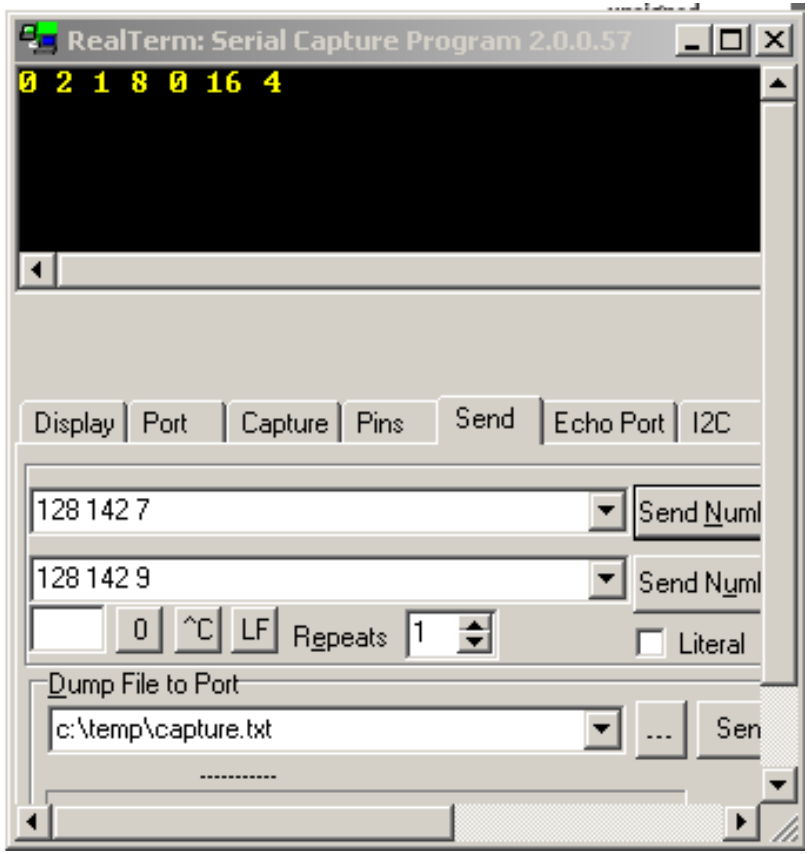
Range: 0 – 1

Cliff Left **Packet ID: 9** **Data Bytes: 1**
unsigned

The state of the cliff sensor on the left side of Create is sent as a 1 bit value (0 = no cliff, 1 = cliff).

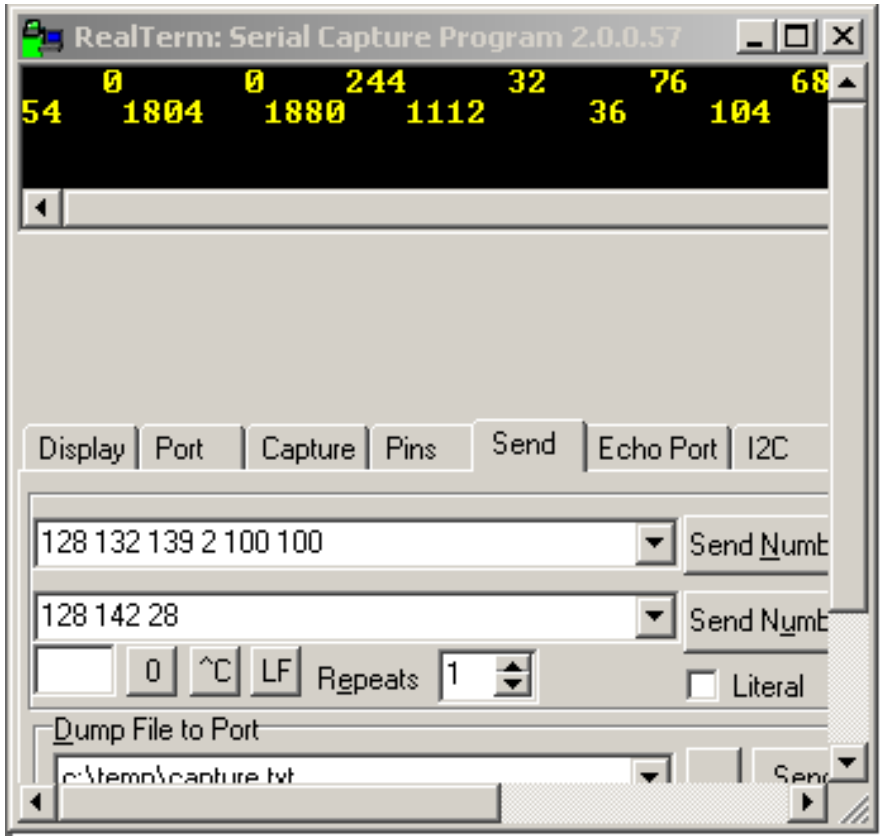
Range: 0 – 1

Digital Version Here



Analog Sensor Readings

- Request an analog value



Wall Signal **Packet ID: 27** **Data Bytes: 2**
unsigned

The strength of the wall sensor's signal is returned as an unsigned 16-bit value, high byte first.

Range: 0-4095

Cliff Left Signal **Packet ID: 28** **Data Bytes: 2**
unsigned

The strength of the left cliff sensor's signal is returned as an unsigned 16-bit value, high byte first.

Range: 0-4095

Cliff Front Left Signal **Packet ID: 29** **Data Bytes: 2**
unsigned

The strength of the front left cliff sensor's signal is returned as an unsigned 16-bit value, high byte first.

Range: 0-4095

Sending commands in Python

- Open IDLE
- Import the serial Python library
- Make sure you are connected via Bluetooth
- Open a Serial port Object called tty

```
>>> from serial import *
>>> dir()
['EIGHTBITS', 'FIVEBITS', 'FileLike', 'MS_CTS_ON', 'MS_DSR_ON', 'MS_RING_ON', 'MS_RLSD_ON', 'PARITY_
EVEN', 'PARITY_MARK', 'PARITY_NAMES', 'PARITY_NONE', 'PARITY_ODD', 'PARITY_SPACE', 'SEVENBITS', 'SIX
BITS', 'STOPBITS_ONE', 'STOPBITS_TWO', 'Serial', 'SerialBase', 'SerialException', 'SerialTimeoutExce
ption', 'VERSION', 'XOFF', 'XON', '__builtins__', '__doc__', '__name__', 'device', 'os', 'portNotOpe
nError', 'serial', 'serialutil', 'serialwin32', 'sys', 'win32con', 'win32event', 'win32file', 'write
TimeoutError']
>>> tty = Serial(port=5,baudrate=57600,timeout=0.01)
```

Note: The serial module is zero based not 1 based so COM 6 is port 5 (sorry)

Sending commands in Python

- Send commands one by one in Python instead of RealTerm (kind of a brute force method)

```
>>> from serial import *
>>> tty = Serial(port=5, baudrate=57600, timeout=0.01)
>>> tty.write(chr(128))
>>> tty.write(chr(132))
>>> tty.write(chr(139))
>>> tty.write(chr(2))
>>> tty.write(chr(100))
>>> tty.write(chr(100))
>>>
```

Note: The serial module is zero based not 1 based so COM 6 is port 5 (sorry)

Make a function to setPlayLED

- We could make an LED function

```
>>> from serial import *
>>> tty = Serial(port=5, baudrate=57600, timeout=0.01)
>>> def setPlayLED(serialObject):
        serialObject.write(chr(128))
        serialObject.write(chr(132))
        serialObject.write(chr(139))
        serialObject.write(chr(2))
        serialObject.write(chr(100))
        serialObject.write(chr(100))


>>> setPlayLED(tty)
>>> tty.close()
```

When you are finished, close the COM port connection.

Using create.py

- So much better! So much easier!

```
>>> from create import *
>>> robot = Create(6)
pycreate version 2.0
PORT is 6
Serial port did open on iRobot Create...
Putting the robot into safe mode...
>>> robot.setLEDs(200,255,1,1)
>>> robot.shutdown()
>>>
```

HANDS-ON PYCREATE