Writing Simple Programs

Getting to know the iRobot Create

Please sit with your choice of a robot partner

Your iRobot Create

- I'll give each pair a locker number and combination.
  - One pair at a time, in parallel with the next set of activities.
- 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.
  - Lockers 21-25 aren't powered, so you are responsible for keeping it charged
- Please have **one** person get your robot from your locker.
- We'll get names from the other partner.

Show Off Some Animations

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

- What other kinds of programs would you like to write?

Q1

Defining a Function

- *Functions*
  - Named sequences of statements—see example below
  - Can *invoke* them—make them run (see next slide)
  - Can take *parameters*—changeable parts (see slide after next)

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

Defining a function called *hello*
Parentheses indicate that it is a function

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

Blank line here 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
```

Q2

Functions with Parameters

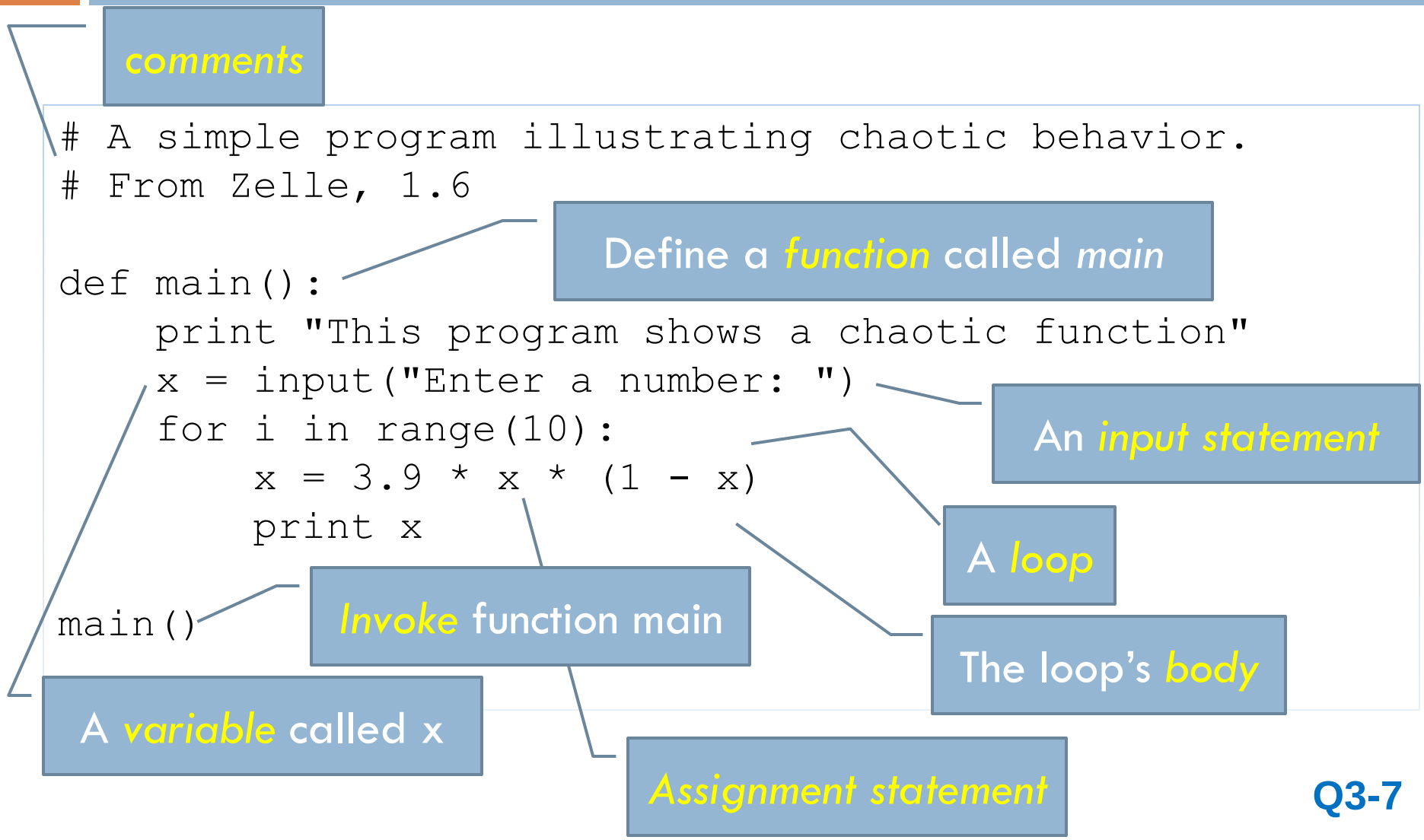
Defining a function called *complain* with a *parameter* named *complaint*

```
>>> def complain(complaint):
        print "Customer: I purchased this parrot not half an hour
   ago from this very boutique"
        print "Owner: Oh yes, the Norwegian Blue.  What's wrong
   with it?"
        print "Customer:", complaint
```

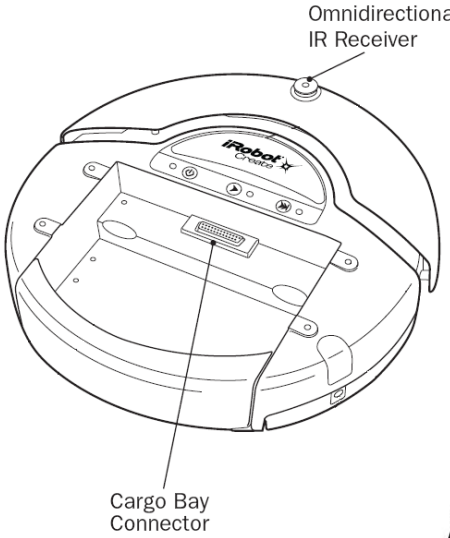
Invoking the *complain* function with the given *argument*

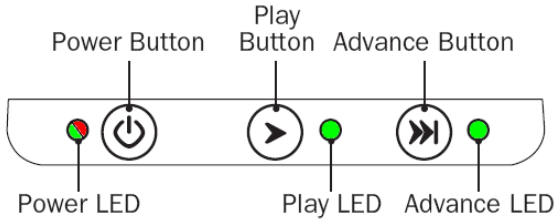
```
>>> complain("He's dead, that's what's wrong with it!")
Customer: I purchased this parrot not half an hour ago from this
   very boutique
Owner: Oh yes, the Norwegian Blue.  What's wrong with it?
Customer: He's dead, that's what's wrong with it!
```

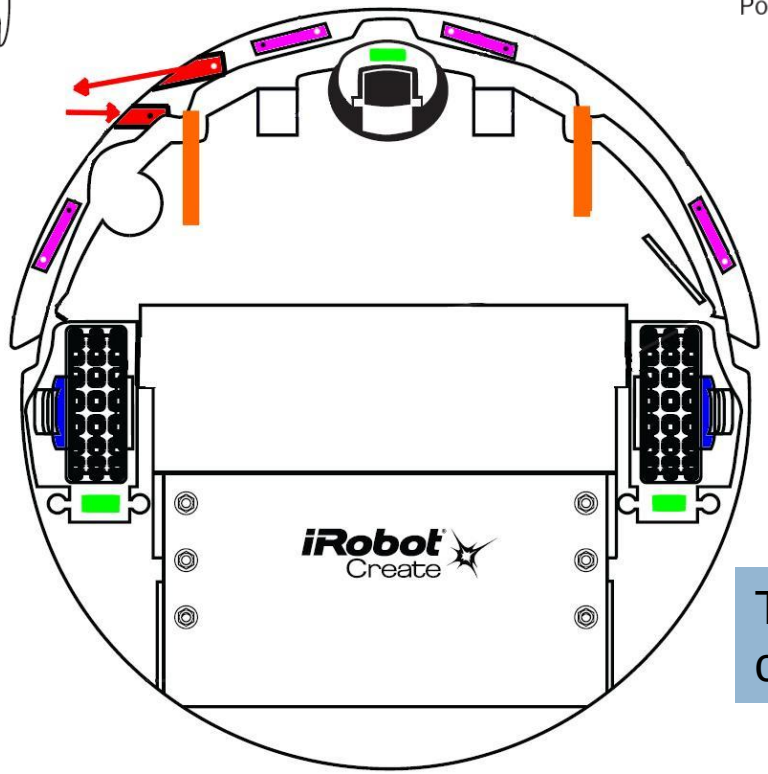
A simple program that defines and invokes a function called *main* – shows input, assignment and a loop



Getting to know the iRobot Create







Turn on your robot (on your table or the floor), pick a program with the Advance Button, and Play the program

Turn OFF your robot as we continue to the next slides

Q8

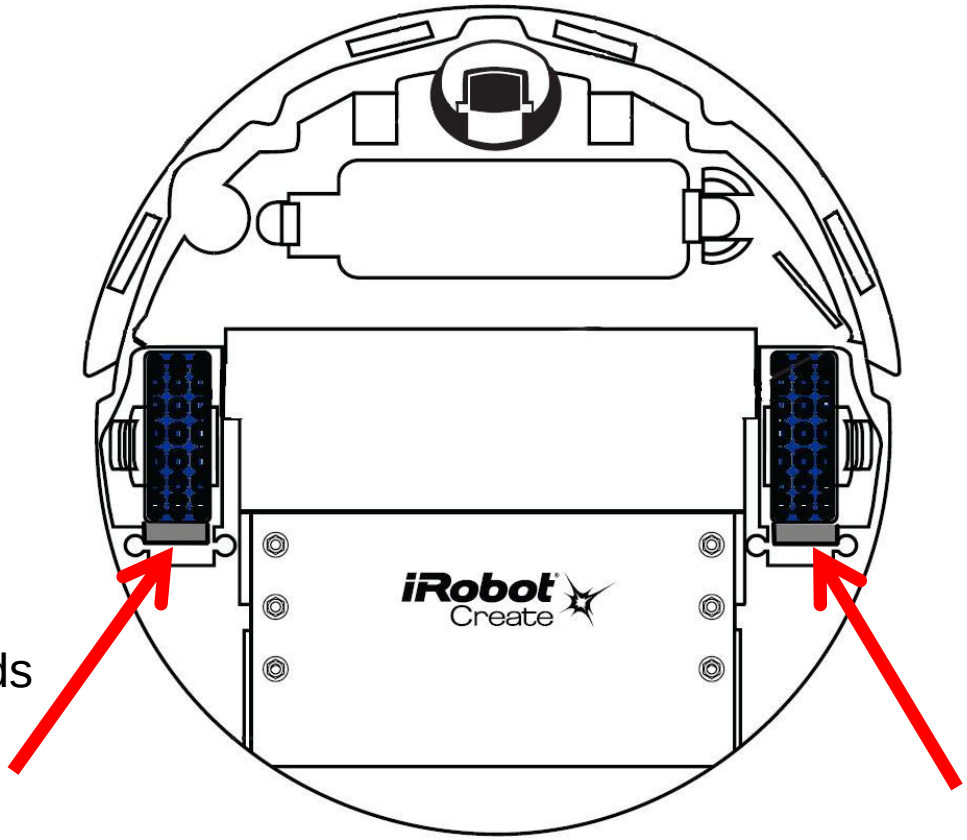
Look at your iRobot Create as we go!

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



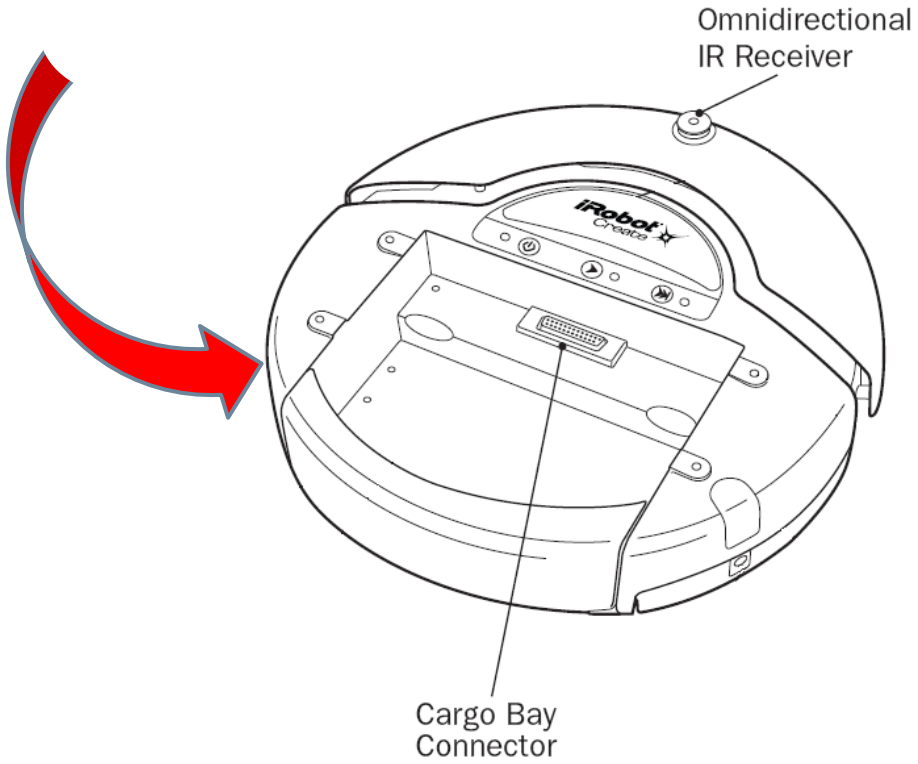
Max speed sets the wheels to 500 mm/s forwards or backwards

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

Q9

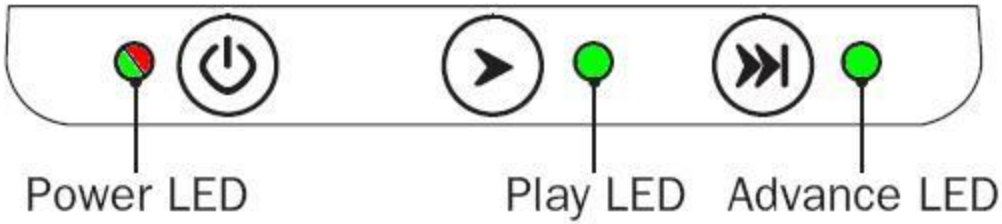
iRobot Actuators – Robot Outputs

- Left Wheel Motor
- Right Wheel Motor
- Speaker



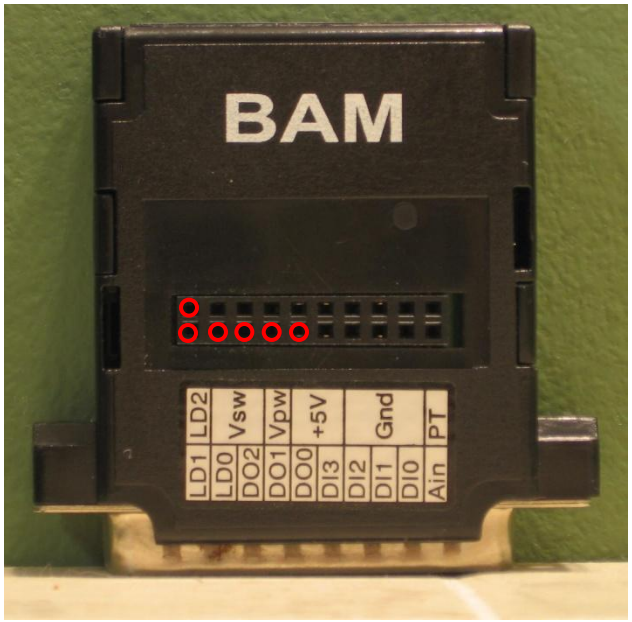
iRobot Actuators – Robot Outputs

- 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)



Q10

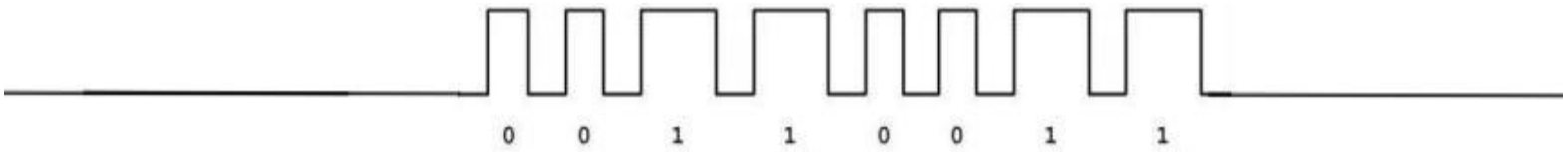
iRobot ***Sensors*** – Robot Inputs

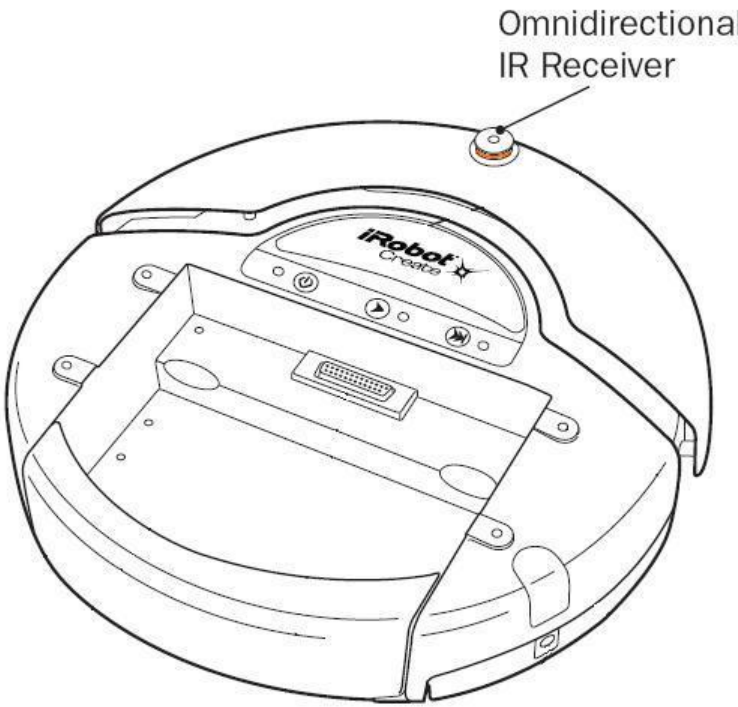
- Omnidirectional IR Sensor
- Play and Advance Buttons
- Left and Right Bumpers
- Three Wheel Drop Sensors
- Four Cliff Sensors
- Wall Sensor
- Encoders
- Four Digital Inputs on the BAM (DI0-DI3)
- Analog Input on the BAM (A_{in})

Omnidirectional IR Receiver

IR Visible →

No IR Light →





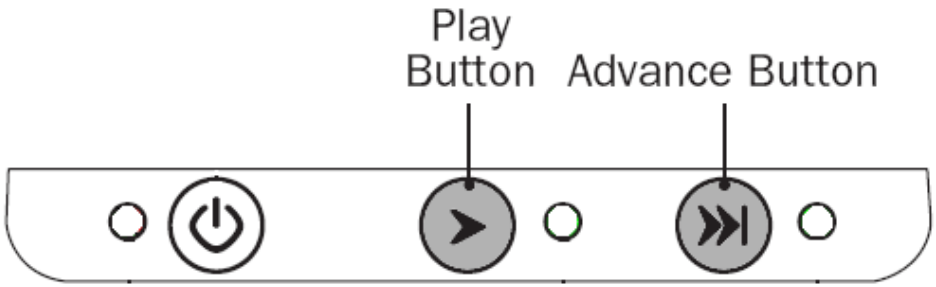
IR receive shown here = 0b00110011
= 0x33
= 51

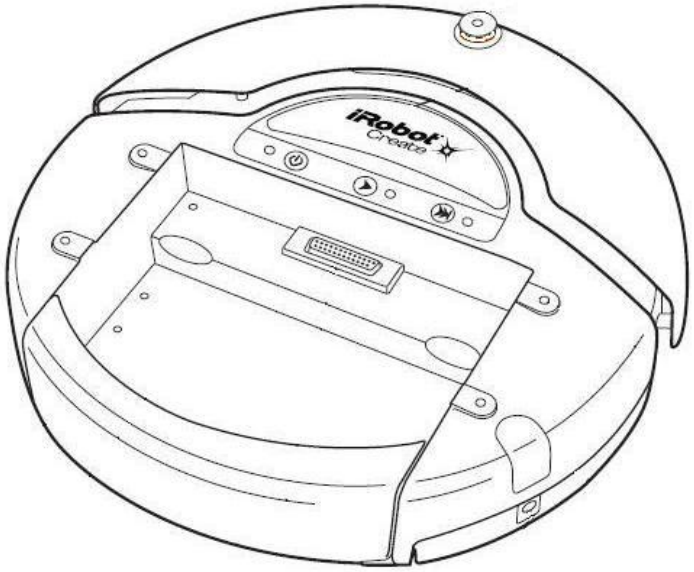
IR transmitters will flash out certain patterns to send 8-bit numbers

Values 0 to 254 (255 is for no signal)

Q11a

Play and Advance Buttons

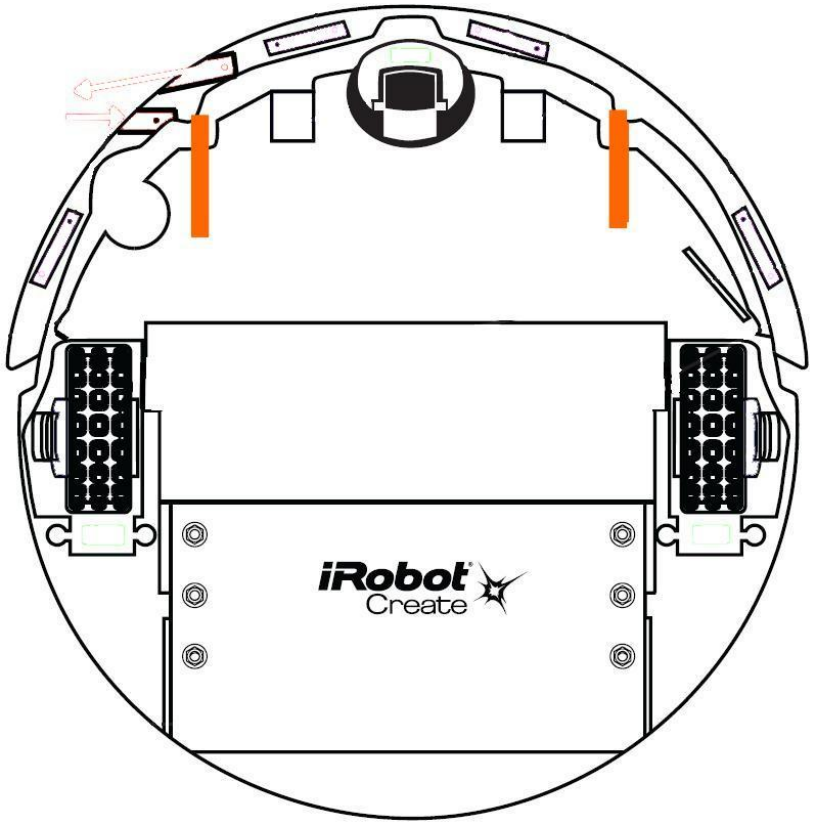




- Digital inputs that you could really use for any function
- They just have symbols on them. Nothing special about that symbol

Q11b

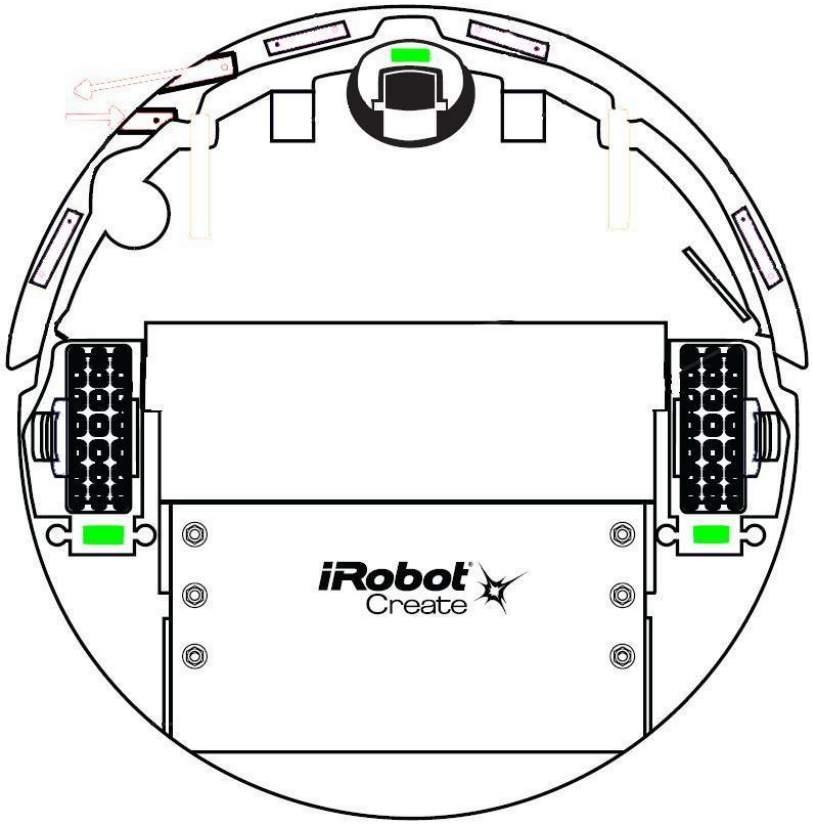
Bump Sensors



- Two digital signals
- Left Bumper
- Right Bumper

Q11c

Wheel Drop Sensors

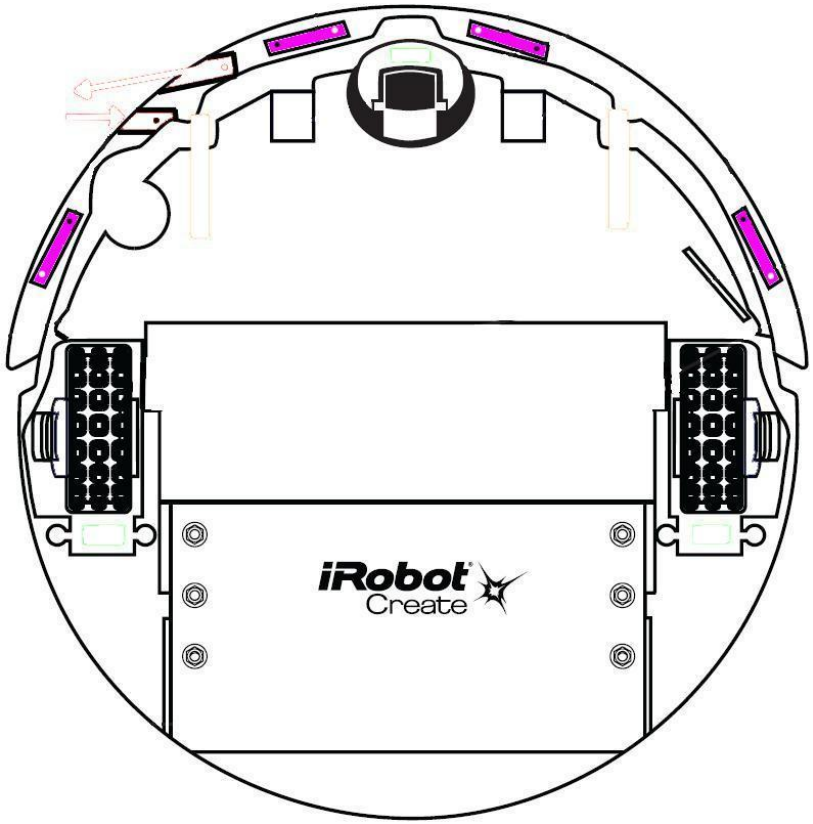


- Three digital inputs

- Front Wheel Drop
- Left Wheel Drop
- Right Wheel Drop

Q11d

Cliff Sensors

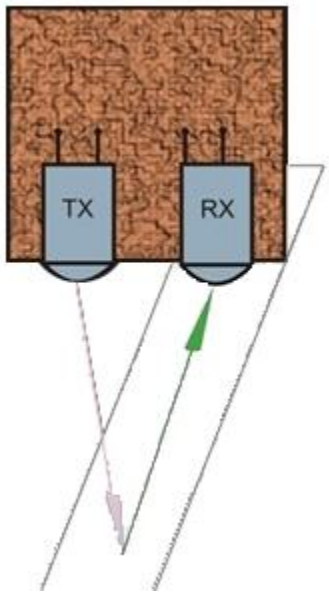
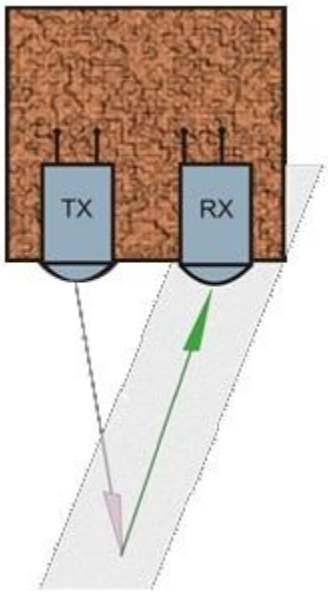
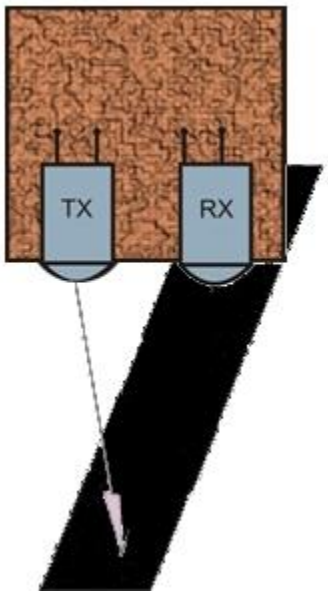
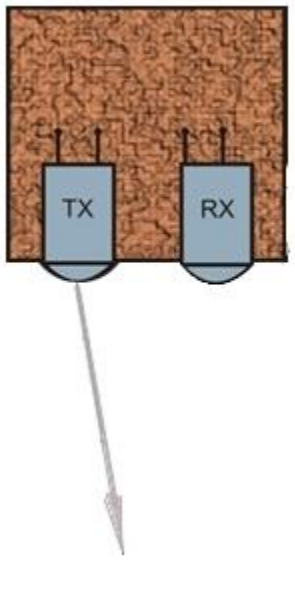


- Four analog inputs

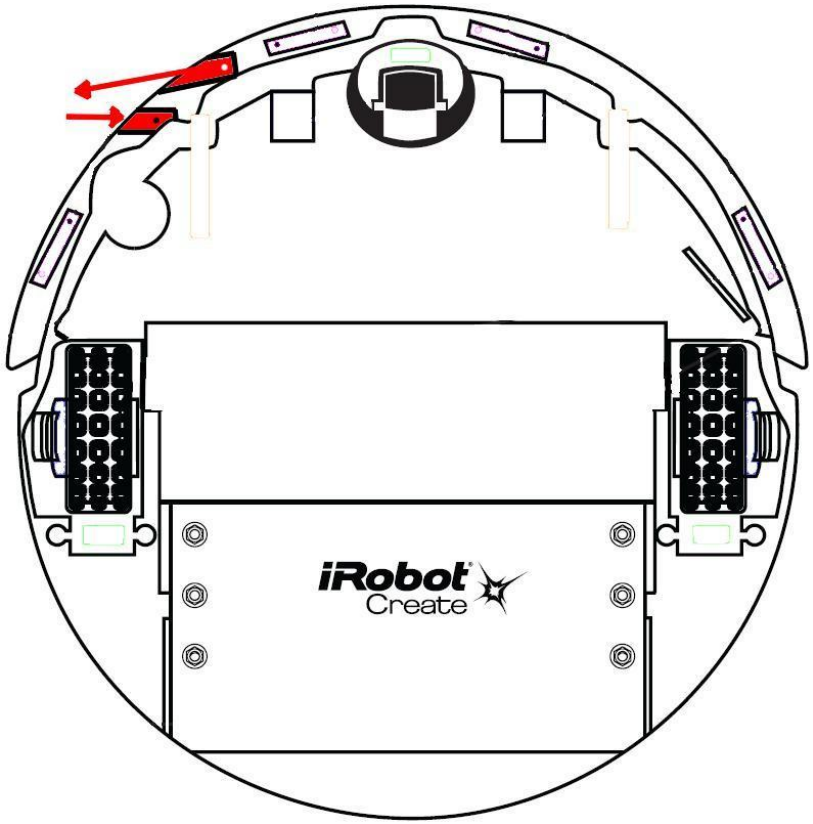
- Cliff Left Signal
- Cliff Front Left Signal
- Cliff Front Right Signal
- Cliff Right Signal

Q11e

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

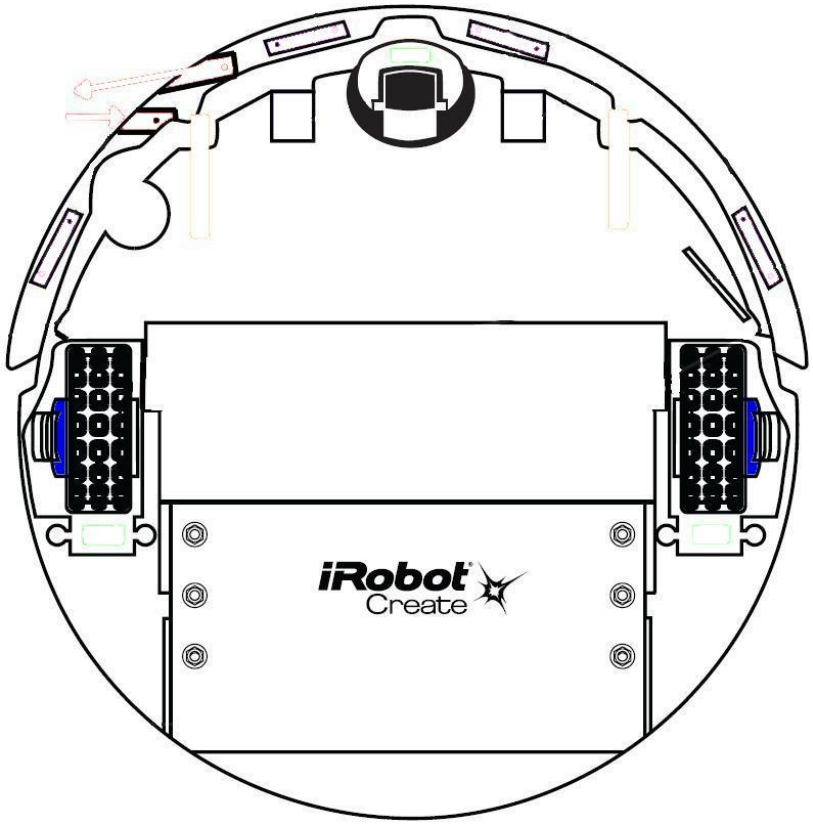


- **One** Analog Sensor
- Value relates to the distance between wall and Create

  0 = No wall seen

Q11f

Wheel Encoders



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

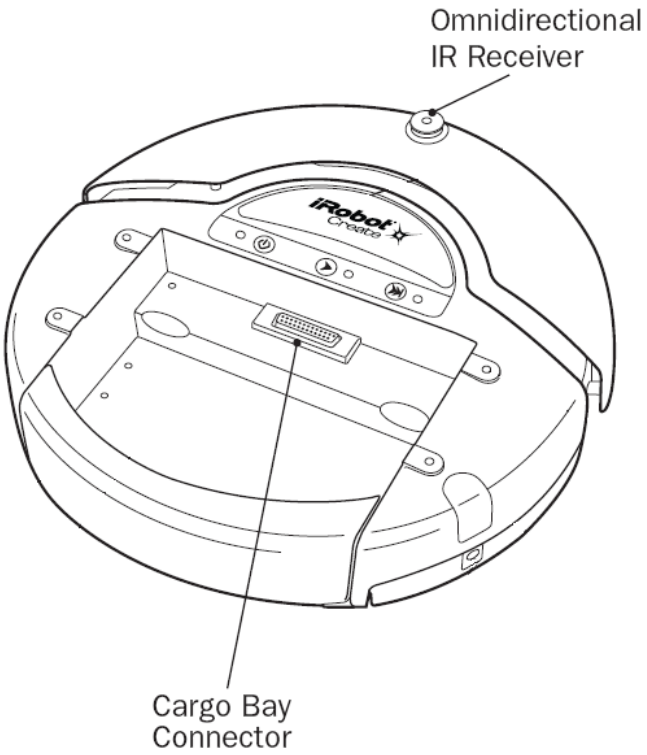
Q11g

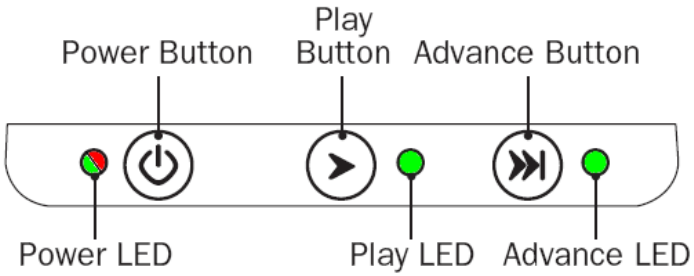
Inputs on the BAM

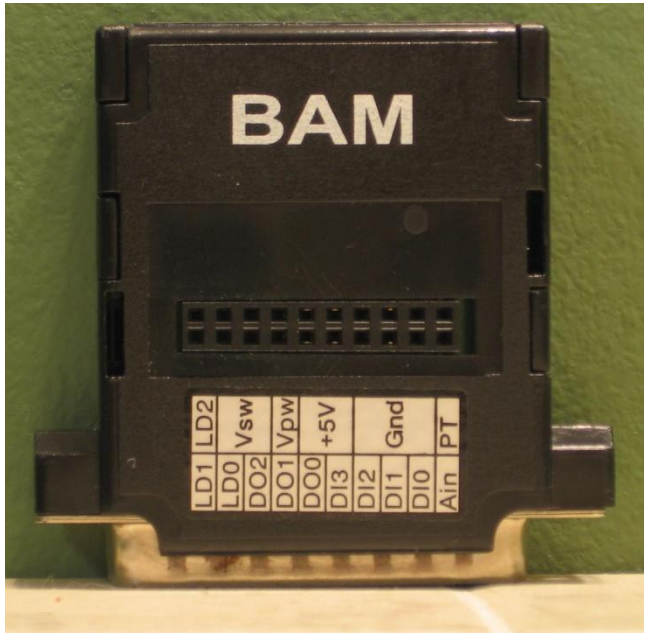
- Four Digital Inputs on the BAM (DI0-DI3)
- Analog Input on the BAM (A_{in})

Q11h

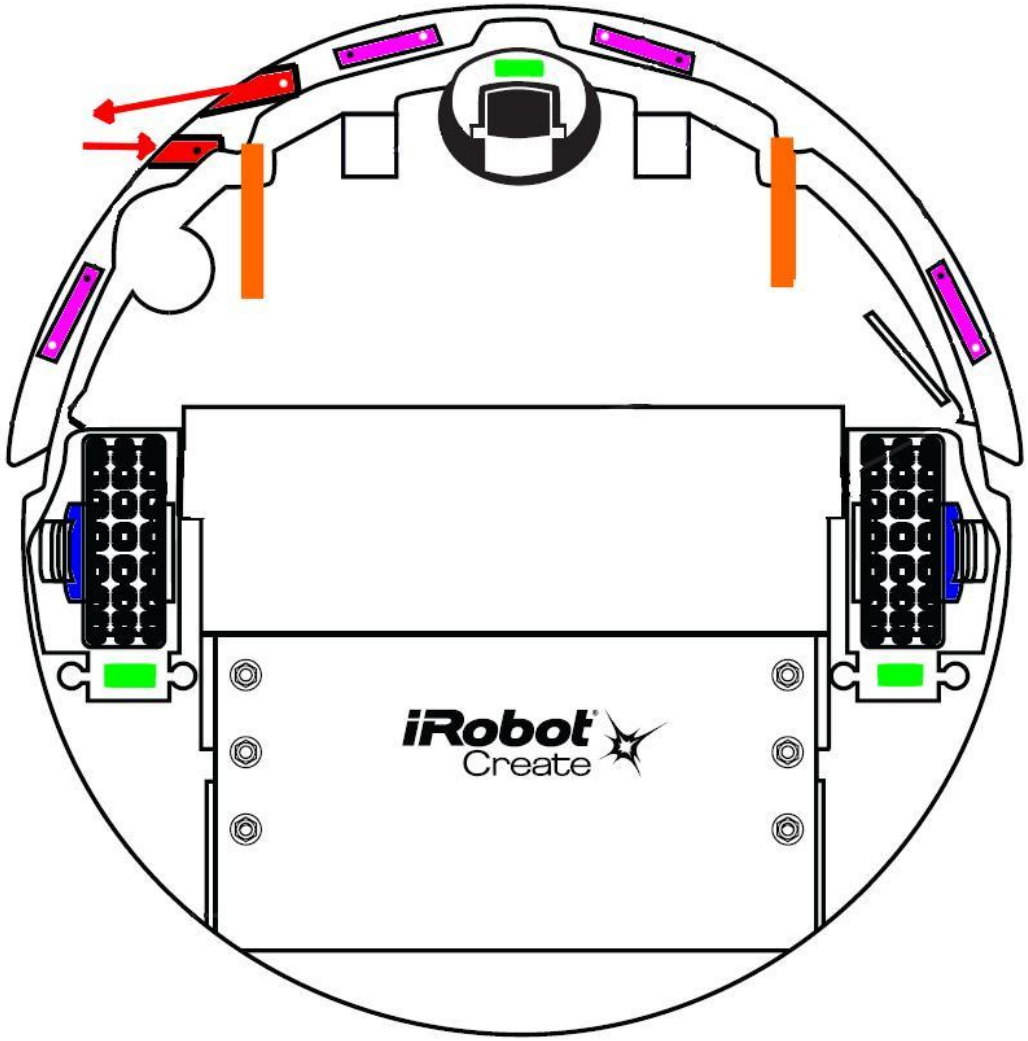
iRobot Create Top View





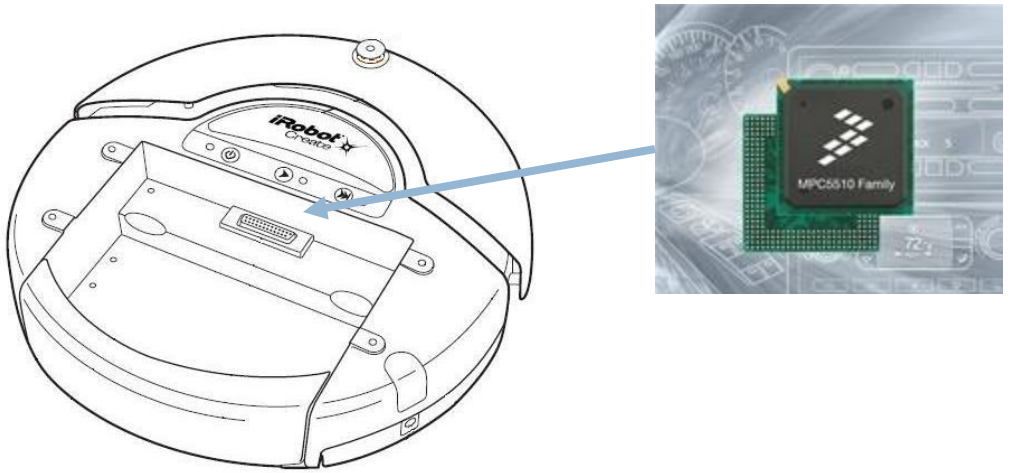


iRobot Create Bottom View

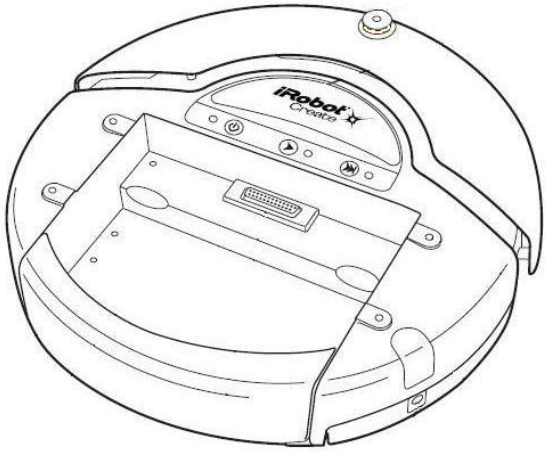


Getting our hands on iRobot Create

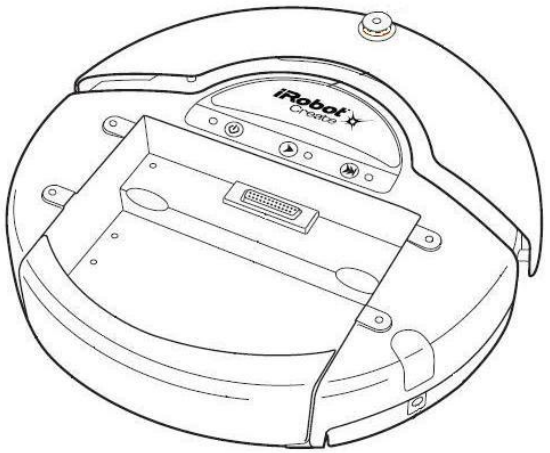
- iRobot Create hardware overview
  - Actuators
  - Sensors
- Sensor signals go to the iRobot microcontroller
- But? The signals need to get to the computer?

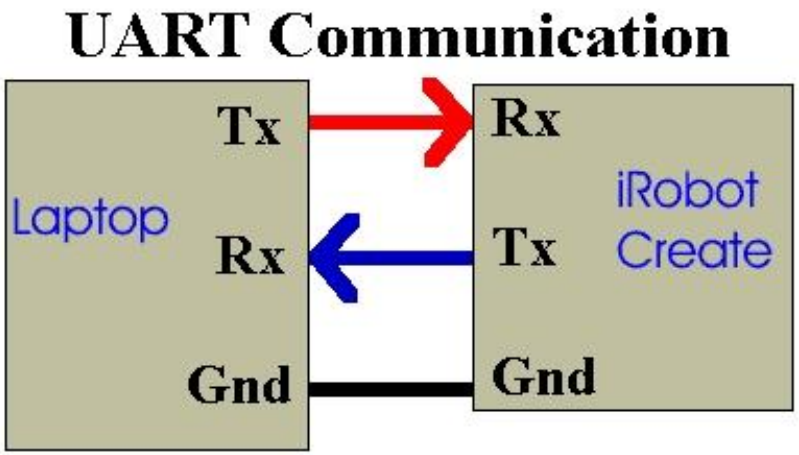


How do we get this information to a PC?



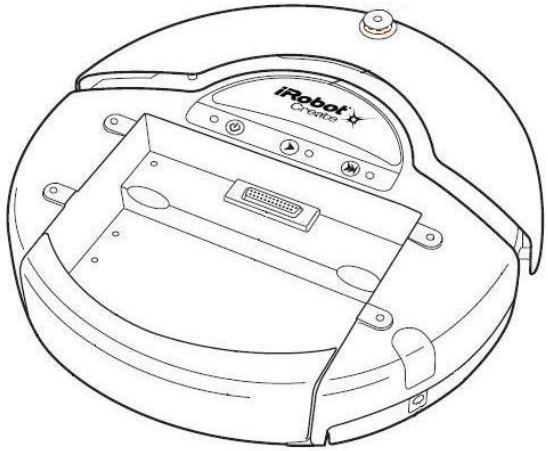
UART Communication

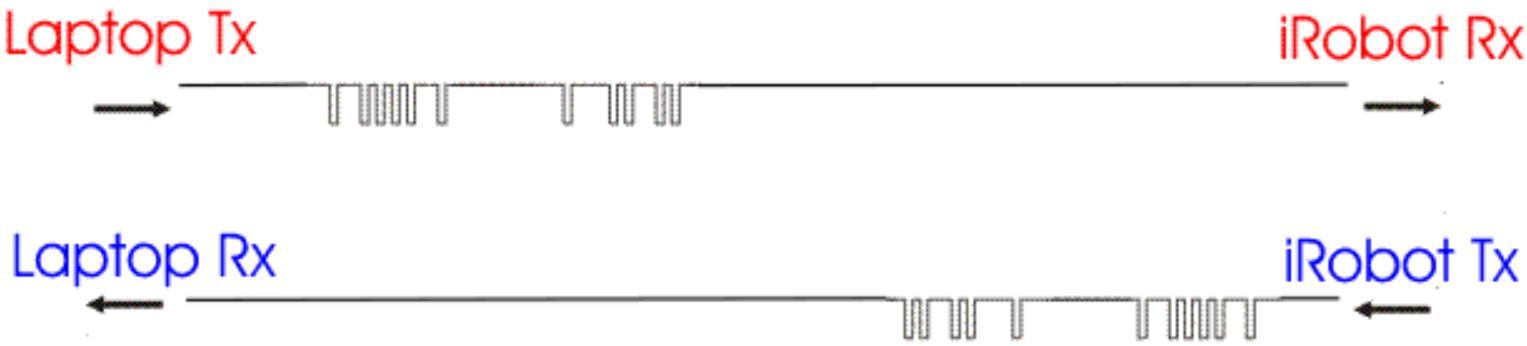




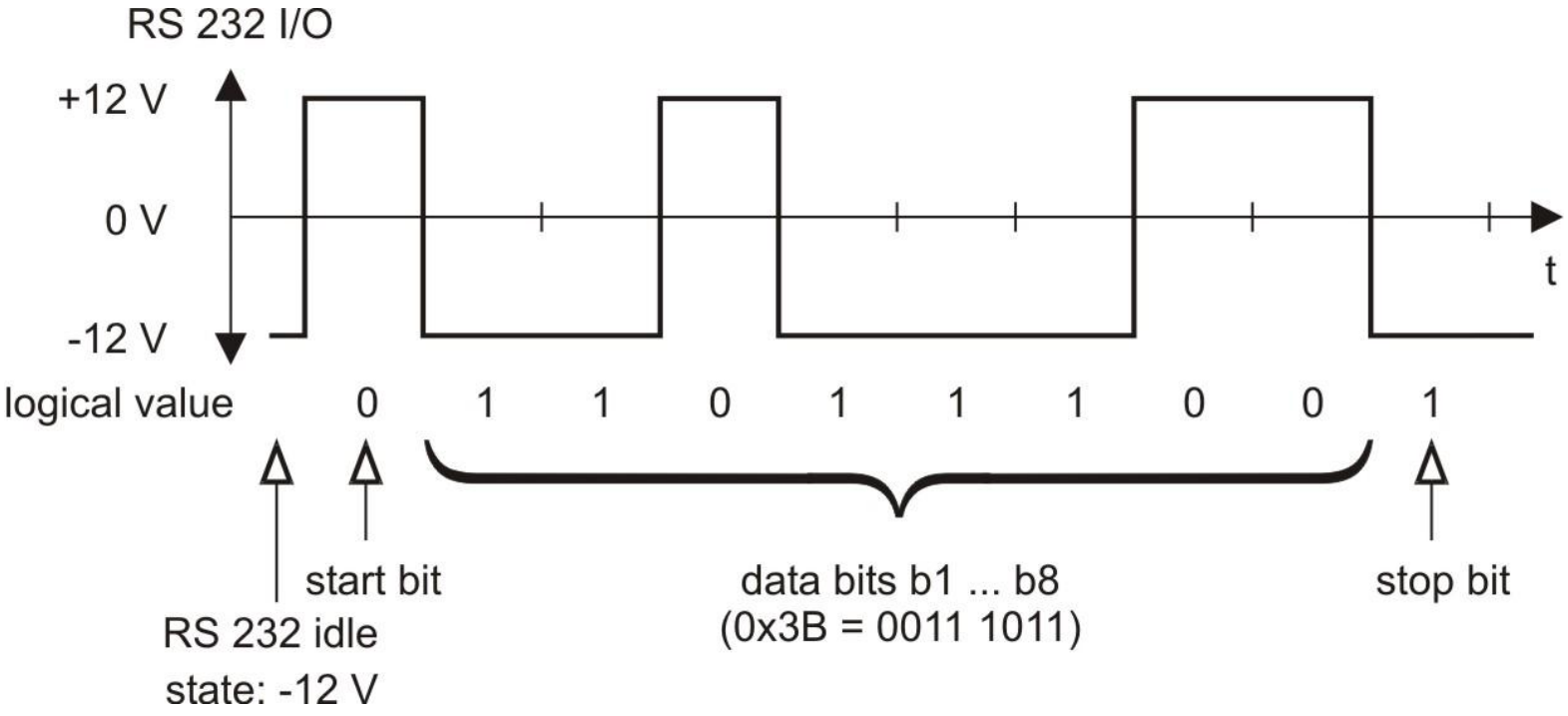
Universal Asynchronous Receiver / Transmitter

Example UART Basics





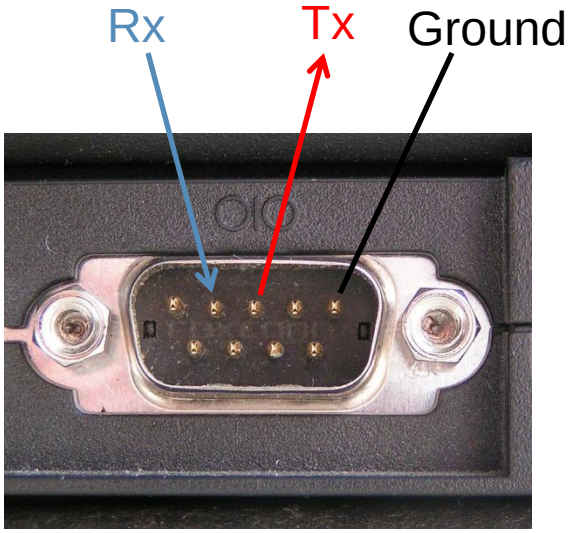
A quick detailed look at UART



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

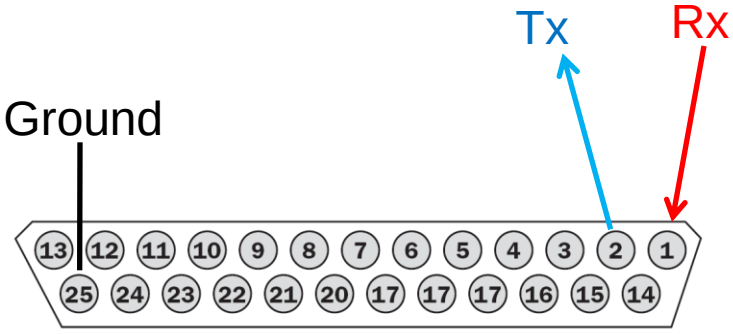
How does UART work?

- 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

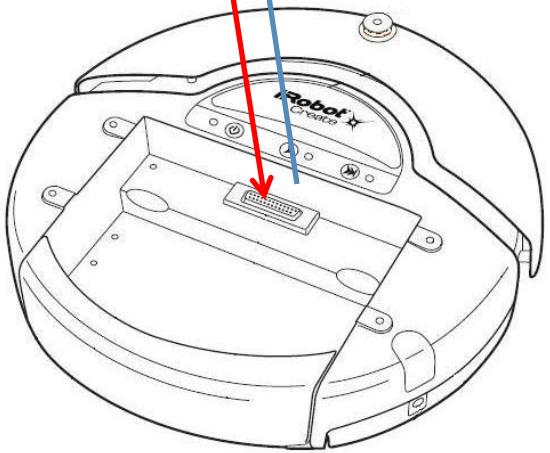
iRobot 25 pin Serial Port

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

How to connect

- ONE partner, go to:

http://www.rose-hulman.edu/class/csse/resources/Robotics/ConnectingToTheCreateRobot.htm

- Follow the directions there ***when your instructor tells you to do so.***
  - ***Once you have connected, shut down your robot***

- Meanwhile, let's talk about how Python can control the robot, then the PyCreate module that it uses to do so

Q12

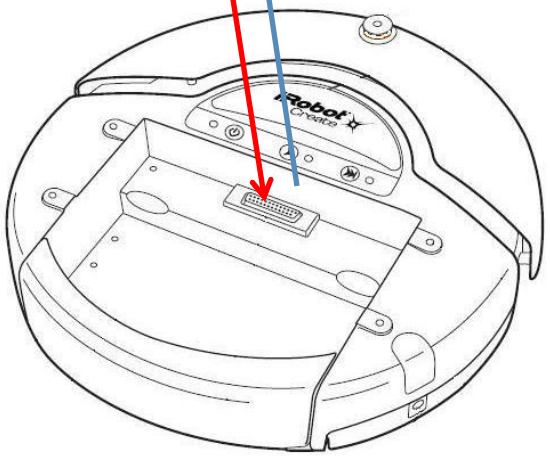
What did we just do?

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

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

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()
>>>
```

While we are getting Bluetooth connections to work, examine the PyCreate handout. Determine how to make the robot:

-- Construct a Create and initiate a connection
-- Close a connection (shutdown)
-- Go forward -- Turn -- Play a song

Your homework will involve these activities!

A PyCreate example

```
from create import *

# Initiate a connection to the robot.
# Use the port for YOUR robot.
robot = Create(9)

for k in range(31, 128, 10):
    print "Note ", k
    robot.playNote(k, 16)
    time.sleep(0.5) # To give the note
                    #  time to play

# Go forward for a couple of seconds
robot.go(10)
time.sleep(2.0)
robot.stop()

# Go forward 20 cm
robot.go(10)
robot.waitDistance(20)
robot.stop()

# Spin 180 degrees
robot.go(0, 30)
robot.waitAngle(180)
robot.stop()

# Disconnect
robot.shutdown()
```