As you arrive:

Sit next to someone DIFFERENT from yesterday.

1. Start up your computer and plug it in.
2. ***Log into Angel*** and go to CSSE 120.
   Do the ***Attendance Widget*** –
   the PIN is on the board.
3. Go to the ***Course Schedule*** web page.
   Open the ***Slides*** for today if you wish.
4. Checkout today's project: `Session02_FunctionsObjectsAndMethods`

Today:

Functions, Objects and Methods

Functions

- Review: the *input-compute-output* pattern
- *Defining* vs. *Calling*
- *Printing* vs. *Returning* values

Objects

- *Constructing*
- *Methods* and *instance variables*

Robots (if time)

- The Create robot – hardware

Outline of today's session

- Introductions: students, assistants and instructor
- Review and practice:
  - The chaos.py program ***and everything about it***
  - The ***input-compute-output*** pattern
  - ***Defining*** a function versus ***calling*** a function
- **Printing** versus **returning** a value
- **Objects**, via zellegraphics
  - ***Constructing*** an object
  - Using objects: dot notation for ***methods*** and ***instance variables***
- Robots – the iRobot Create (as time permits)
  - Demo
  - Introduction to its hardware

Contact Before Work

- Ask your partner:

What foreign country have you visited (or want to visit)?

What did you like about your visit (or what makes you want to visit that country)?

- Why do Contact Before Work?
  - Helps us know our teammates.

    We work better with people we know and like.
  - Helps start the meeting on time:

Roll Call & Introductions

- Name (nickname)
- Hometown
- Where you live on (or off) campus
- Something about you that most people in the room don't know

Syllabus, grading

- Note unusual grading plan
- Will work to your benefit if you work hard!
- Questions?

Checkout today's project:

`Session02_FunctionsObjectsAndMethods`

Are you in the Pydev perspective? If not:

`Window ~ Open Perspective ~ Other` then `Pydev`

Messed up views? If so:

`Window ~ Reset Perspective`

Troubles getting today's project? If so:

No SVN repositories view (tab)? If it is not there:

`Window ~ Show View ~ Other`
then `SVN ~ SVN Repositories`

1. ***In your SVN repositories view (tab), expand your repository*** *(the top-level item) if not already expanded.*
   - If no repository, perhaps you are in the wrong Workspace. Get help.
2. ***Right-click on today's project,*** *then select* ***`Checkout`****.*
   Press ***OK*** *as needed.* The project shows up in the
   `Pydev Package Explorer`
   to the right. Expand and browse the modules under `src` as desired.

Review (this and next set of slides): Your first Python example: *chaos!*

```
def main():
    """ Calls a function (chaos) which shows a chaotic sequence. """
    chaos()
    print('Goodbye!')

def chaos():
    """
    Computes and prints a chaotic sequence of numbers,
    as a function of a number input from the user.
    """
    print('This function illustrates a chaotic function.')
    x = float(input('Enter a number between 0 and 1: '))

    for k in range(20): #@UnusedVariable
        x = 3.9 * x * (1 - x)
        print(x)

    print('Examine the sequence of numbers printed.')
    print('Does it appear chaotic?')
```

Q2

Review: Doc-comments

Your first Python example: *chaos!*

```
def main():
    """ Calls a function (chaos) which shows a chaotic sequence. """
    chaos()
    print('Goodbye!')

def chaos():
    """
    Computes and prints a chaotic sequence of numbers,
    as a function of a number input from the user.
    """
    print('This function illustrates a chaotic function.')
    x = float(input('Enter a number between 0 and 1: '))

    for k in range(20): #@UnusedVariable
        x = 3.9 * x * (1 - x)
        print(x)

    print('Examine the sequence of numbers printed.')
    print('Does it appear chaotic?')
```

Q2

Review: internal comments

Your first Python example: *chaos!*

```
def main():
    """ Calls a function (chaos) which shows a chaotic sequence. """
    chaos()
    print('Goodbye!')

def chaos():
    """
    Computes and prints a chaotic sequence of numbers,
    as a function of a number input from the user.
    """
    print('This function illustrates a chaotic function.')
    x = float(input('Enter a number between 0 and 1: '))

    for k in range(20): #@UnusedVariable
        x = 3.9 * x * (1 - x)
        print(x)

    print('Examine the sequence of numbers printed.')
    print('Does it appear chaotic?')
```

Q2

Review: Execution is sequential except …

Your first Python example: *chaos!*

```
def main():
    """ Calls a function (chaos) which shows a chaotic sequence. """
    chaos()
    print('Goodbye!')

def chaos():
    """
    Computes and prints a chaotic sequence of numbers,
    as a function of a number input from the user.
    """
    print('This function illustrates a chaotic function.')
    x = float(input('Enter a number between 0 and 1: '))

    for k in range(20): #@UnusedVariable
        x = 3.9 * x * (1 - x)
        print(x)

    print('Examine the sequence of numbers printed.')
    print('Does it appear chaotic?')
```

Q2

Review: Defining a function

Your first Python example: *chaos!*

```
def main():
    """ Calls a function (chaos) which shows a chaotic sequence. """
    chaos()
    print('Goodbye!')

def chaos():
    """
    Computes and prints a chaotic sequence of numbers,
    as a function of a number input from the user.
    """
    print('This function illustrates a chaotic function.')
    x = float(input('Enter a number between 0 and 1: '))

    for k in range(20): #@UnusedVariable
        x = 3.9 * x * (1 - x)
        print(x)

    print('Examine the sequence of numbers printed.')
    print('Does it appear chaotic?')
```

Q2

Review: Calling a function

Your first Python example: *chaos!*

```
def main():
    """ Calls a function (chaos) which shows a chaotic sequence. """
    chaos()
    print('Goodbye!')

def chaos():
    """
    Computes and prints a chaotic sequence of numbers,
    as a function of a number input from the user.
    """
    print('This function illustrates a chaotic function.')
    x = float(input('Enter a number between 0 and 1: '))

    for k in range(20): #@UnusedVariable
        x = 3.9 * x * (1 - x)
        print(x)

    print('Examine the sequence of numbers printed.')
    print('Does it appear chaotic?')
```

Q2

Review: Loop (part 1, basic idea)

Your first Python example: *chaos!*

```
def main():
    """ Calls a function (chaos) which shows a chaotic sequence. """
    chaos()
    print('Goodbye!')

def chaos():
    """
    Computes and prints a chaotic sequence of numbers,
    as a function of a number input from the user.
    """
    print('This function illustrates a chaotic function.')
    x = float(input('Enter a number between 0 and 1: '))

    for k in range(20): #@UnusedVariable
        x = 3.9 * x * (1 - x)
        print(x)

    print('Examine the sequence of numbers printed.')
    print('Does it appear chaotic?')
```

Q2

Review: Body of a function/loop

Your first Python example: *chaos!*

```
def main():
    """ Calls a function (chaos) which shows a chaotic sequence. """
    chaos()
    print('Goodbye!')

def chaos():
    """
    Computes and prints a chaotic sequence of numbers,
    as a function of a number input from the user.
    """
    print('This function illustrates a chaotic function.')
    x = float(input('Enter a number between 0 and 1: '))

    for k in range(20): #@UnusedVariable
        x = 3.9 * x * (1 - x)
        print(x)

    print('Examine the sequence of numbers printed.')
    print('Does it appear chaotic?')
```

Q2

Review: Printing (also `print(xx, yy, ...)`)

Your first Python example: *chaos!*

```
def main():
    """ Calls a function (chaos) which shows a chaotic sequence. """
    chaos()
    print('Goodbye!')

def chaos():
    """
    Computes and prints a chaotic sequence of numbers,
    as a function of a number input from the user.
    """
    print('This function illustrates a chaotic function.')
    x = float(input('Enter a number between 0 and 1: '))

    for k in range(20): #@UnusedVariable
        x = 3.9 * x * (1 - x)
        print(x)

    print('Examine the sequence of numbers printed.')
    print('Does it appear chaotic?')
```

Q2

Review: input, variables & assignment

Your first Python example: *chaos!*

```
def main():
    """ Calls a function (chaos) which shows a chaotic sequence. """
    chaos()
    print('Goodbye!')

def chaos():
    """
    Computes and prints a chaotic sequence of numbers,
    as a function of a number input from the user.
    """
    print('This function illustrates a chaotic function.')
    x = float(input('Enter a number between 0 and 1: '))

    for k in range(20): #@UnusedVariable
        x = 3.9 * x * (1 - x)
        print(x)

    print('Examine the sequence of numbers printed.')
    print('Does it appear chaotic?')
```

Q2

Review: Loops, part 2 (notation, range)

Your first Python example: *chaos!*

```
def main():
    """ Calls a function (chaos) which shows a chaotic sequence. """
    chaos()
    print('Goodbye!')

def chaos():
    """
    Computes and prints a chaotic sequence of numbers,
    as a function of a number input from the user.
    """
    print('This function illustrates a chaotic function.')
    x = float(input('Enter a number between 0 and 1: '))

    for k in range(20): #@UnusedVariable
        x = 3.9 * x * (1 - x)
        print(x)

    print('Examine the sequence of numbers printed.')
    print('Does it appear chaotic?')
```

Q2

Review: Accumulation pattern: x = … x …

Your first Python example: *chaos!*

```
def main():
    """ Calls a function (chaos) which shows a chaotic sequence. """
    chaos()
    print('Goodbye!')

def chaos():
    """
    Computes and prints a chaotic sequence of numbers,
    as a function of a number input from the user.
    """
    print('This function illustrates a chaotic function.')
    x = float(input('Enter a number between 0 and 1: '))

    for k in range(20): #@UnusedVariable
        x = 3.9 * x * (1 - x)
        print(x)

    print('Examine the sequence of numbers printed.')
    print('Does it appear chaotic?')
```

Q2

Summary: the *input-compute-output* pattern

```
def celsius_to_fahrenheit():
    celsius = float(input('What is Cel. temperature? '))
    fahrenheit = 9/5 * celsius + 32
    print('Temperature is', fahrenheit, 'degrees Fahr.')
```

- Getting input from the user
  - `input('What is Cel. temperature? ')`
  - `float(...)`
  - `celsius = ...`
- Computing a value using an assignment
  - `fahrenheit = 9/5 * celsius + 32`
- Printing values to the console
  - `print('Tem...', fahrenheit, 'deg...')`

Practice: the ***input-compute-output*** pattern

- In today's project, examine `m1_temperature.py`
- Do `TODO: 1...` And `TODO: 2...`
- Questions on:
  - The doc-comment at the top of the file?
    As the first statement inside the body of a function?
  - The #comments inside the file?
  - Defining a function?
  - Calling a function?
  - The boiler-plate at the bottom of the file that calls *main*?
  - Anything else?

Practice: the ***input-compute-output*** pattern

- In today's project, examine the module (file):

  `m2_distance_from_origin.py`

- The statement

  ```
  import math
  ```

  makes the *math* module (library) available to this module.

  For example, you can then write:

  ```
  math.sqrt(5.8)
  ```

- Do the TODO's (#1, #2).
  - Ask questions as needed!

PRINTING versus RETURNING values

- In today's project, examine the module (file):

  `m3_print_vs_return.py`

- The `print` function prints to the console.
  The `return` expression returns a value to the caller.
  Study the difference by discussing with your instructor:
  - `main`
  - `print_versus_return`
  - `print_sum_of_square_roots`
  - `return_pi_approximation`

- Do the TODO's (#1, #2, #3, ...)
  - Ask questions as needed! Finish as homework.

Zellegraphics

- Check that Zellegraphics is on your computer by:
  - Try to run `m4_face.py module` in `ZellegraphicsAndObjects` in today's project
  - If a window pops us, & if clicking in the window closes it, you're set!
  - If not: Look at your

    `C:\Program Files\Python31\Lib\site-packages`

    folder. If it lacks any of the following:

`zellegraphics.py` `sitecustomize.py` `create.py` `serial`

Then unzip this and put the relevant files/folder into the above.

Also, if the above folder lacks a `win32` folder, then download and run this.

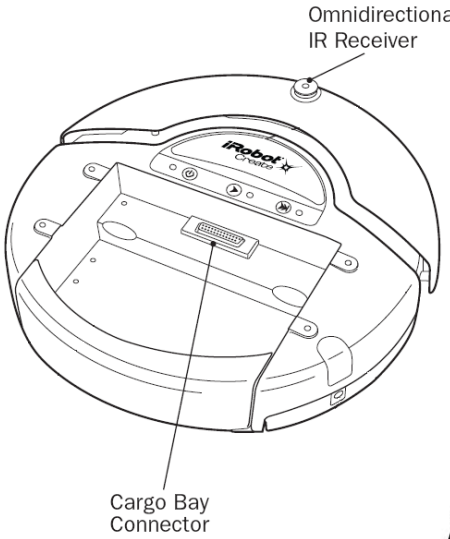
Objects and Zellegraphics

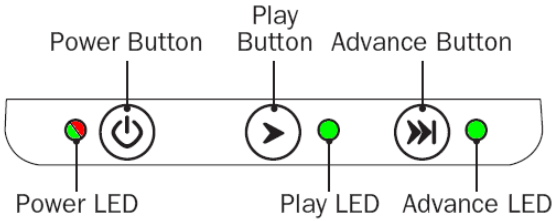
- In today's project, examine the
  `m4_face.py` `module`
- Do the TODO's (#1, #2, #3, …) with your instructor
- Concepts:
  - Importing a module
  - Objects:
    - Know stuff (stored in instance variables)
    - Can do stuff (via methods)
  - Constructing an object
  - Asking the object to do/return something by calling a method that the object has
    - Similar to a function call, but applied to an object

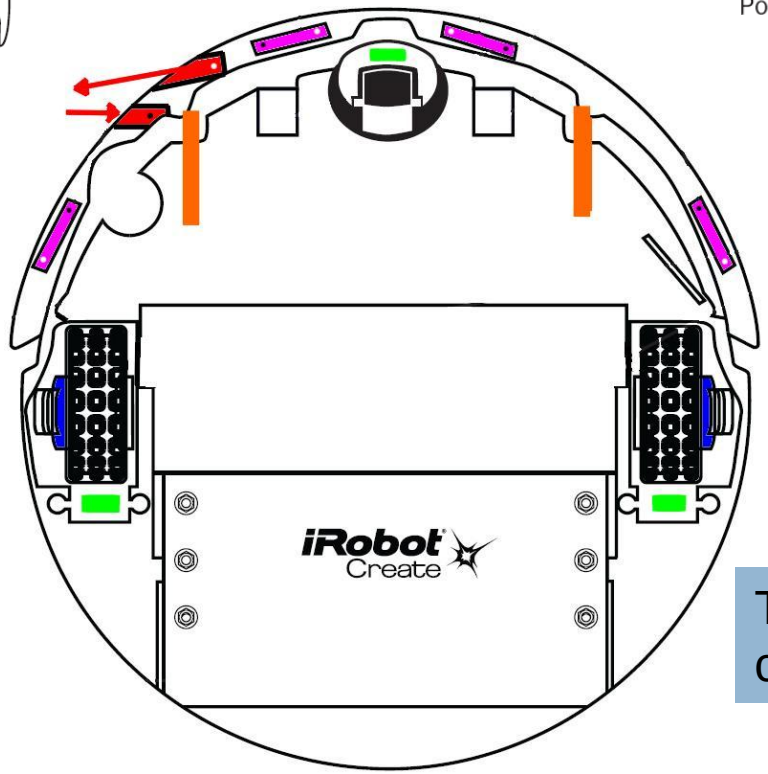
The iRobot Create

- Today:
  - Demo
  - Hardware
- Next session:
  - Writing programs that make them do things!

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

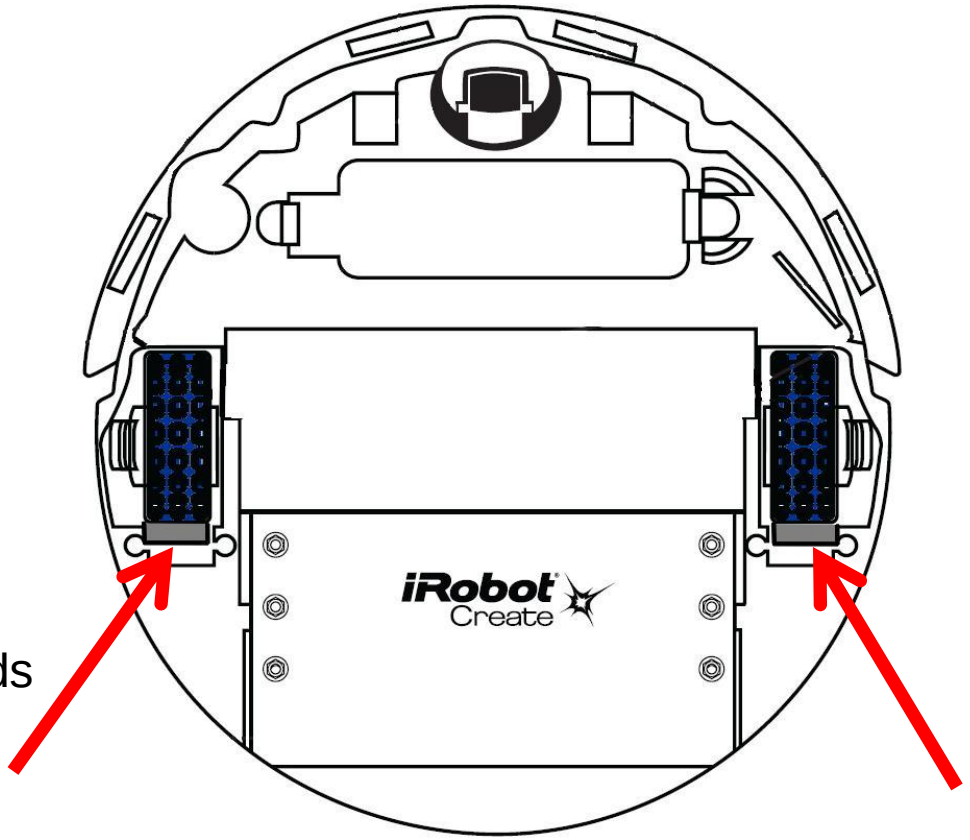
Look at your iRobot Create as we go!

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

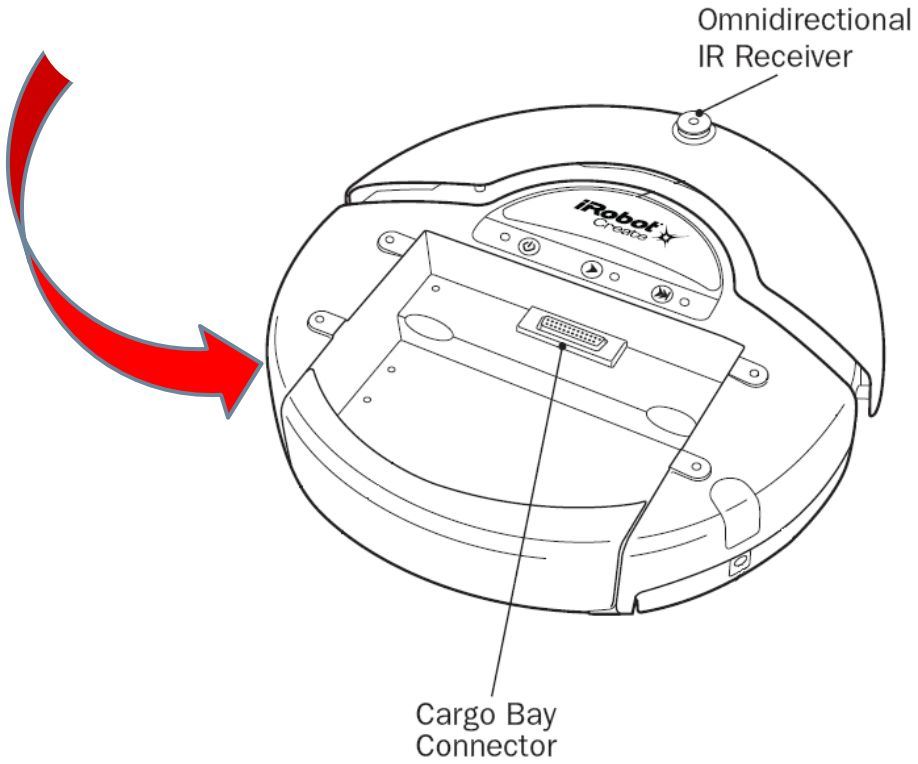


Max speed sets the wheels to 500 mm/s forwards or backwards

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

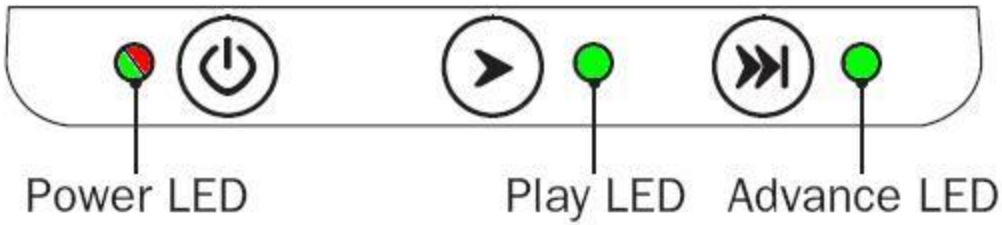
iRobot Actuators – Robot Outputs

- Left Wheel Motor
- Right Wheel Motor
- Speaker



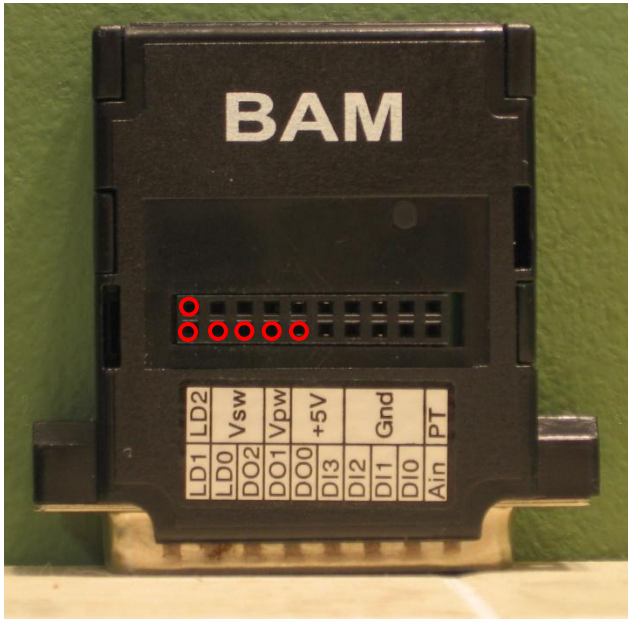
iRobot Actuators – Robot Outputs

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



iRobot Actuators – Robot Outputs

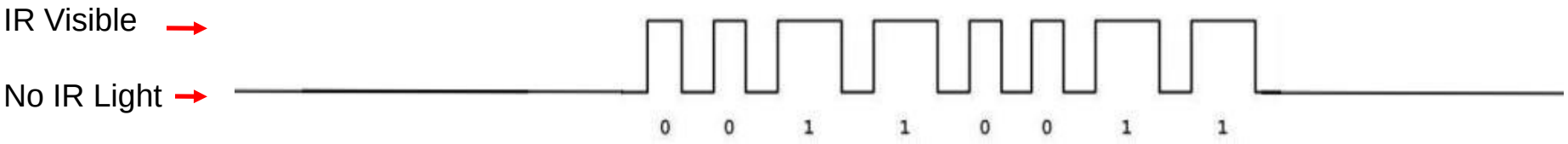
- Left Wheel Motor
- Right Wheel Motor
- Speaker
- Bi-color Power LED
- Play LED
- Advance LED
- Low-side Drivers on the BAM (LD0-LD2)
- Digital Outputs on the BAM (DO0-DO2)

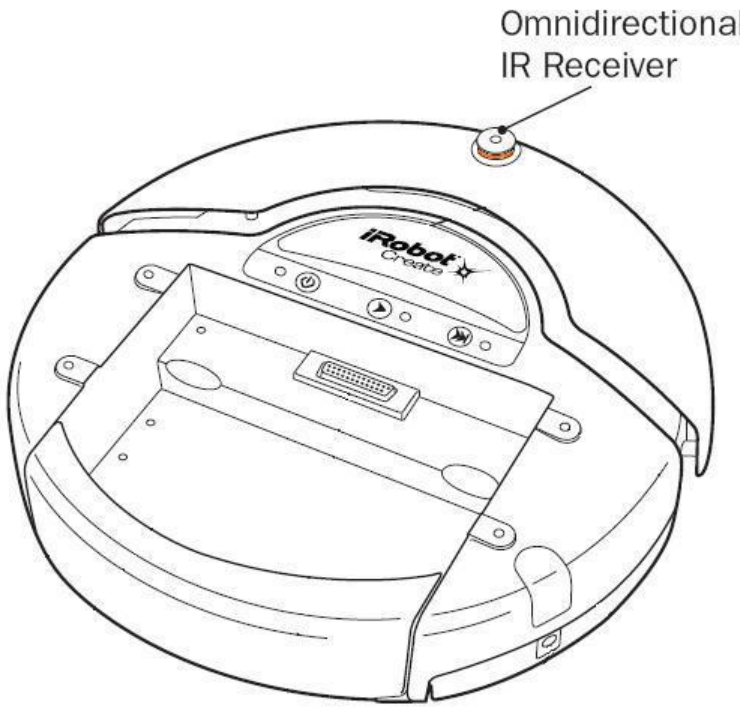


iRobot ***Sensors*** – Robot Inputs

- 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



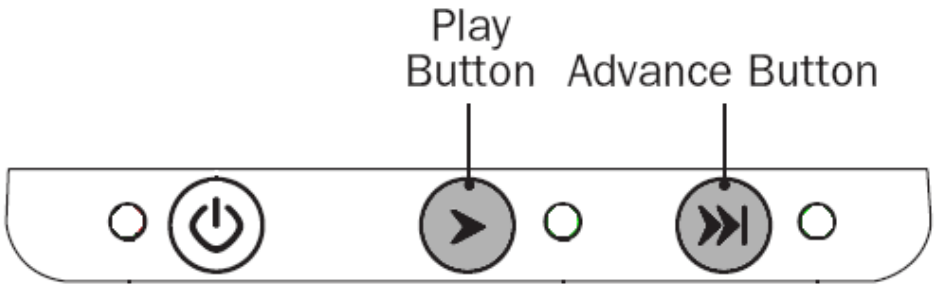


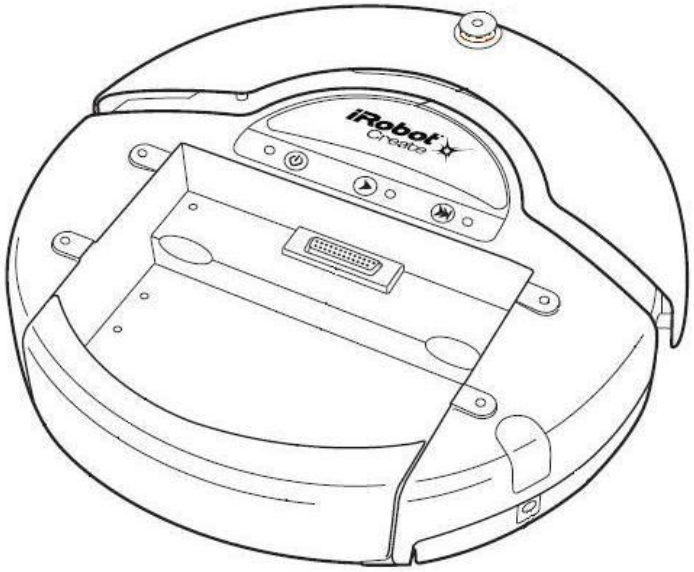
IR receive shown here = 0b00110011
= 0x33
= 51

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

Values 0 to 254 (255 is for no signal)

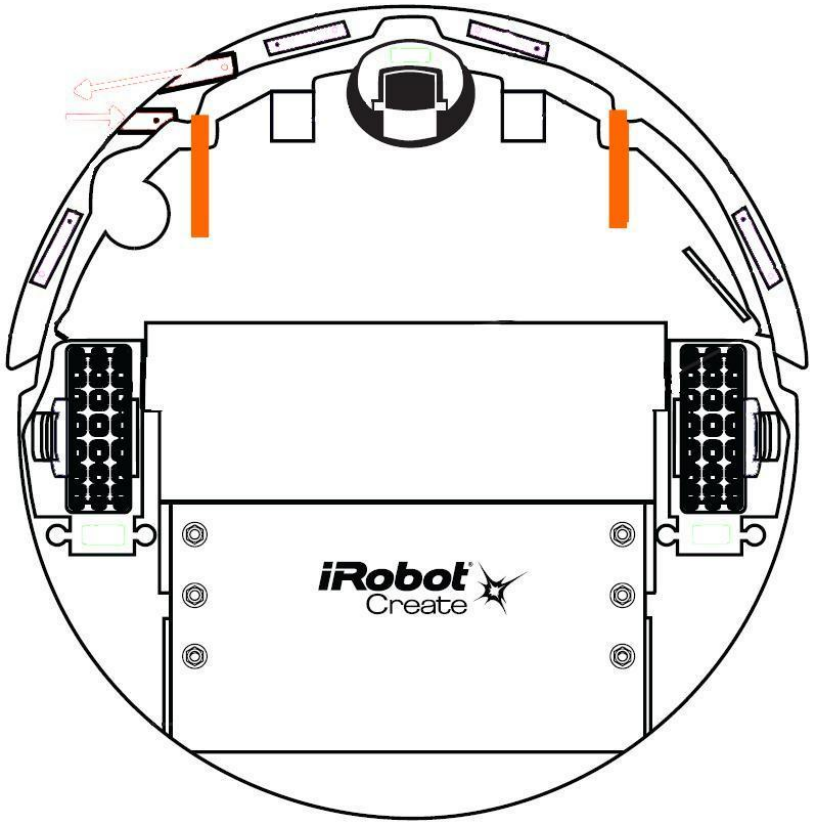
Play and Advance Buttons





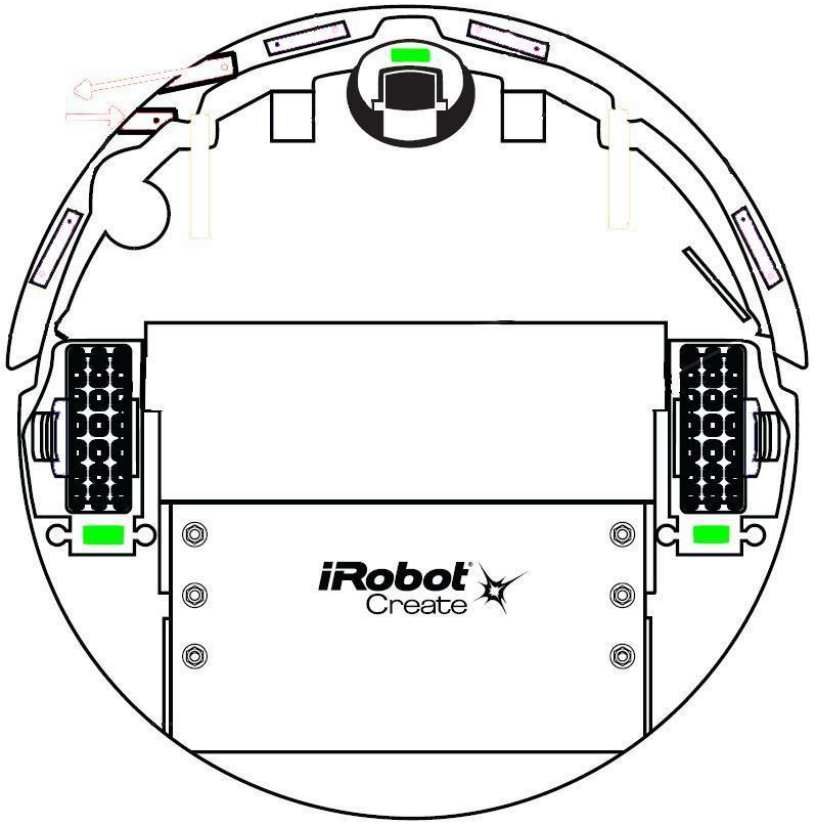
- Digital inputs that you could really use for any function
- They just have symbols on them. Nothing special about that symbol

Bump Sensors



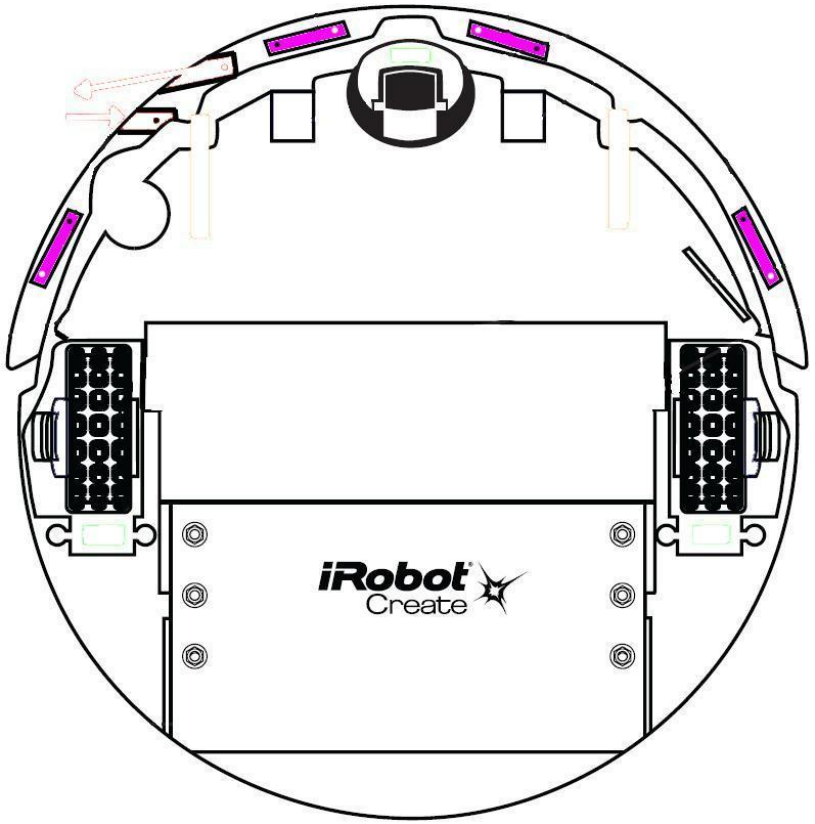
- Two digital signals
- Left Bumper
- Right Bumper

Wheel Drop Sensors



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

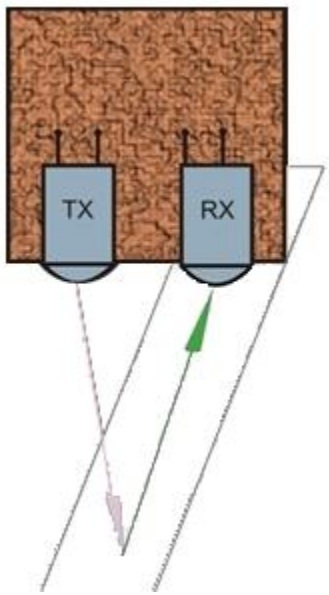
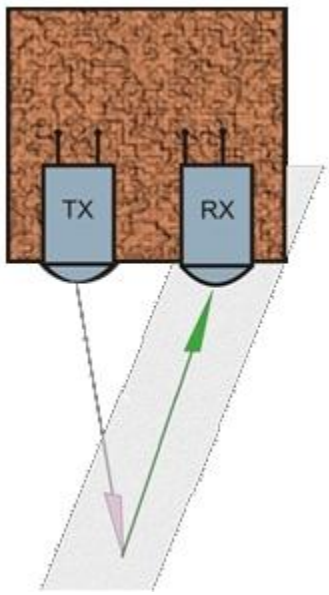
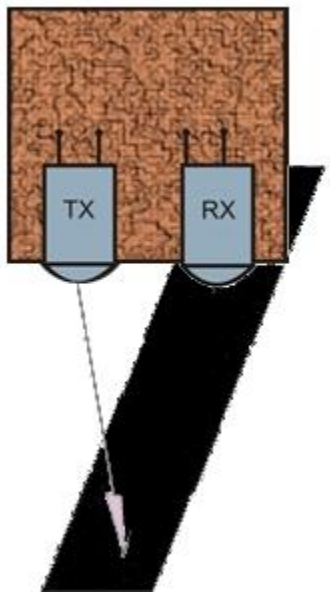
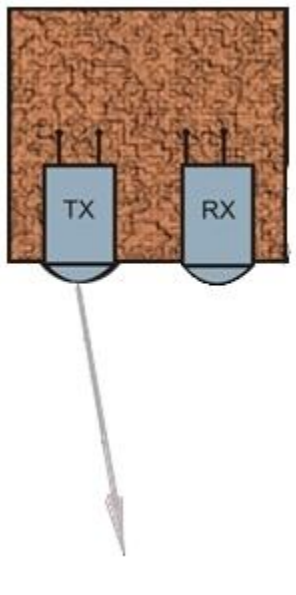
Cliff Sensors



- Four analog inputs

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

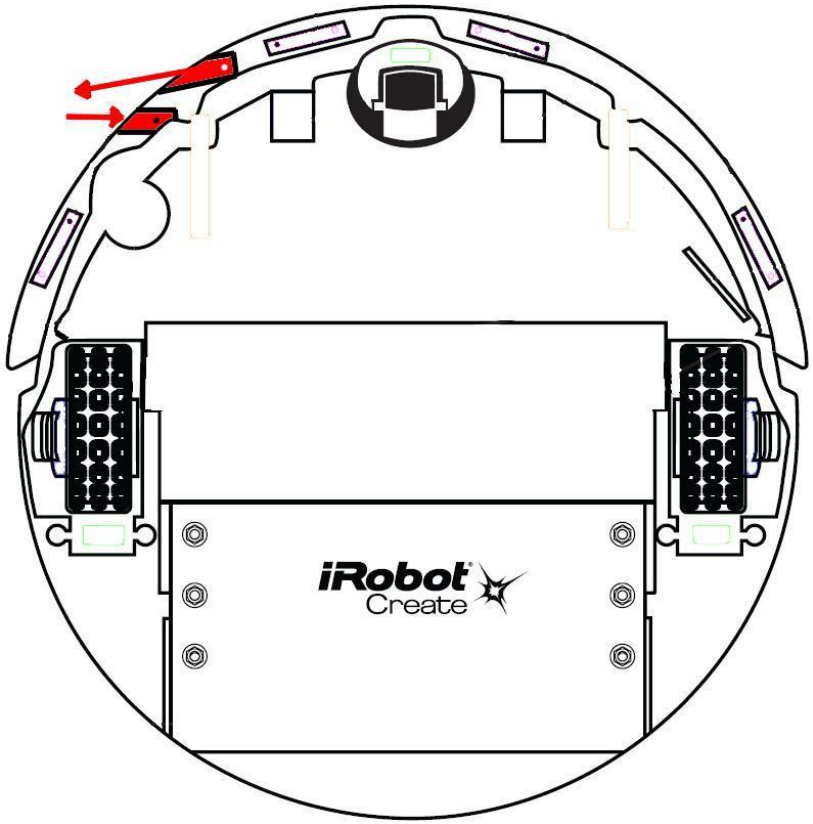
Cliff Sensor Analog Readings

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

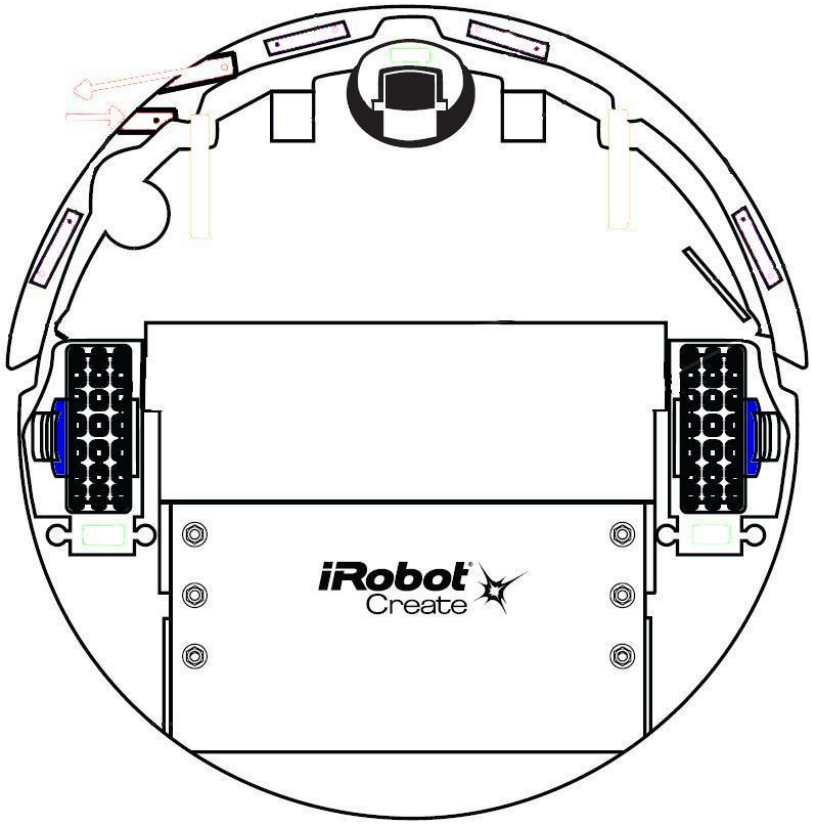
Wall Sensor



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

  0 = No wall seen

Wheel Encoders

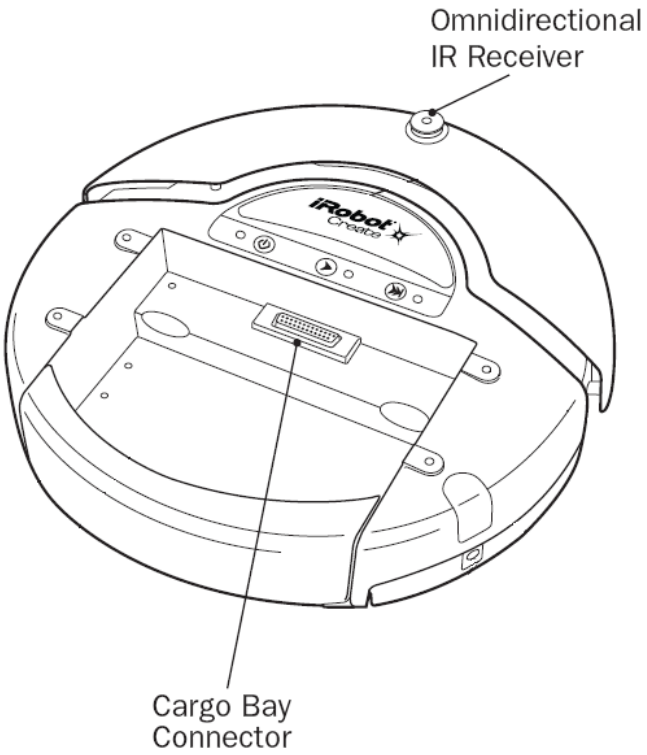


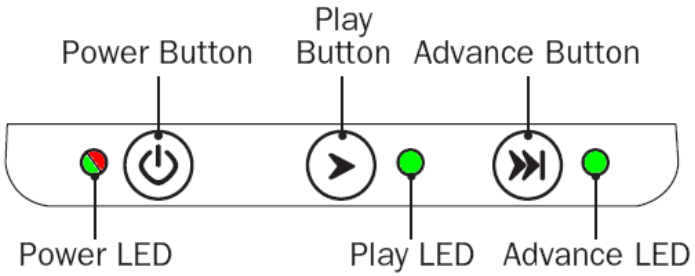
- More complex
- Distance since last request
- Angle since last request
- Used internally to control wheel speed

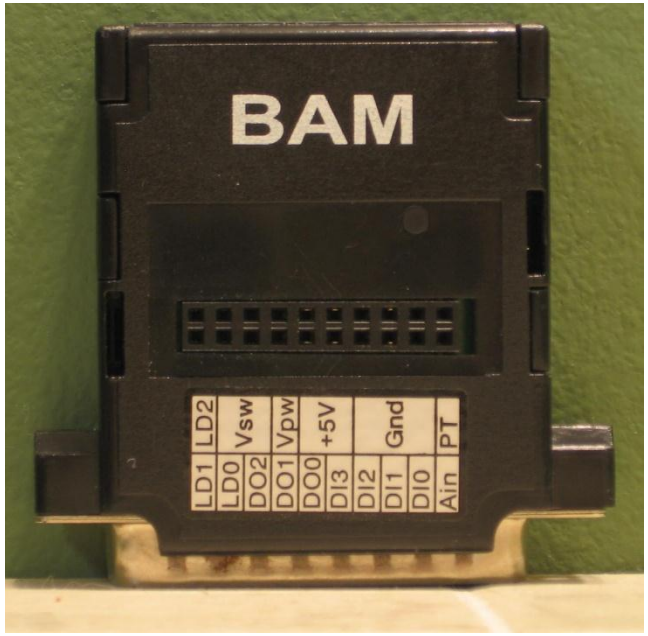
Inputs on the BAM

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

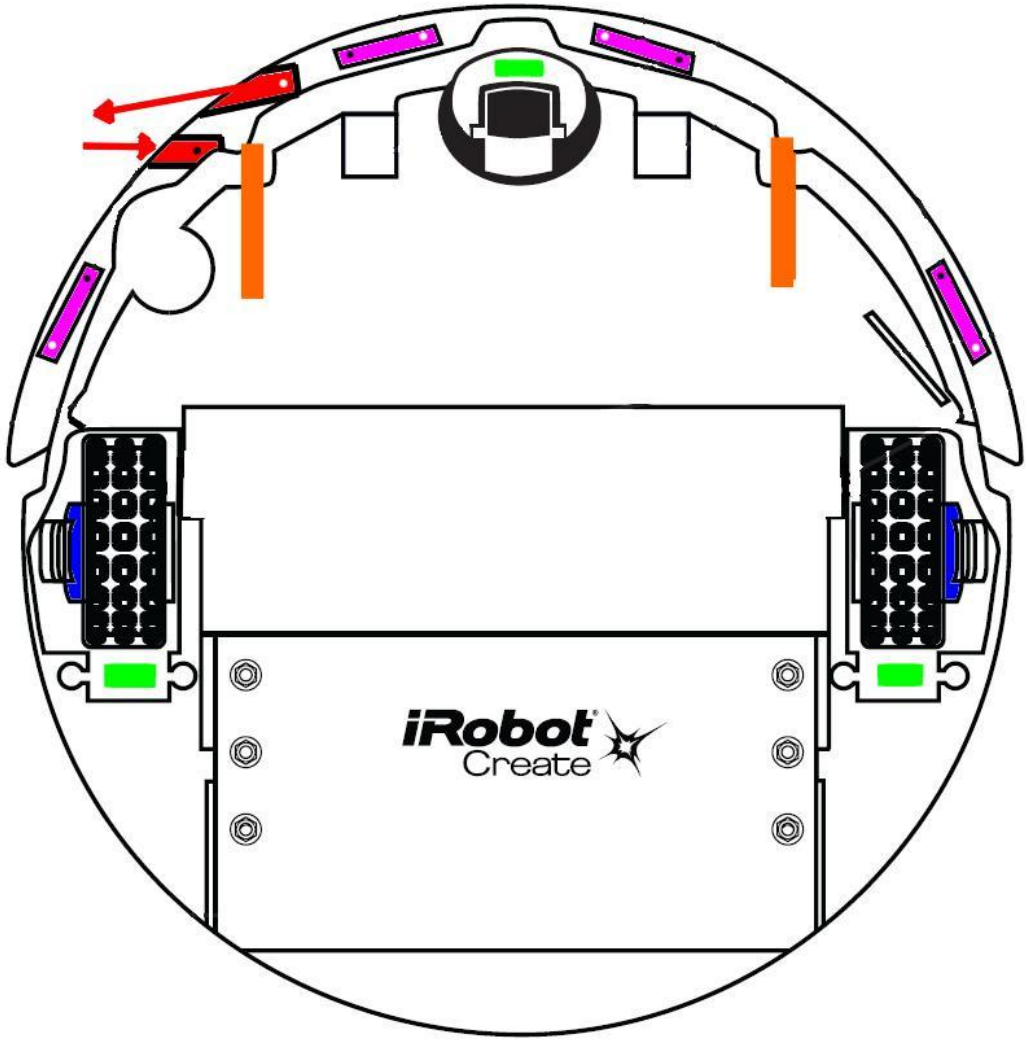
iRobot Create Top View





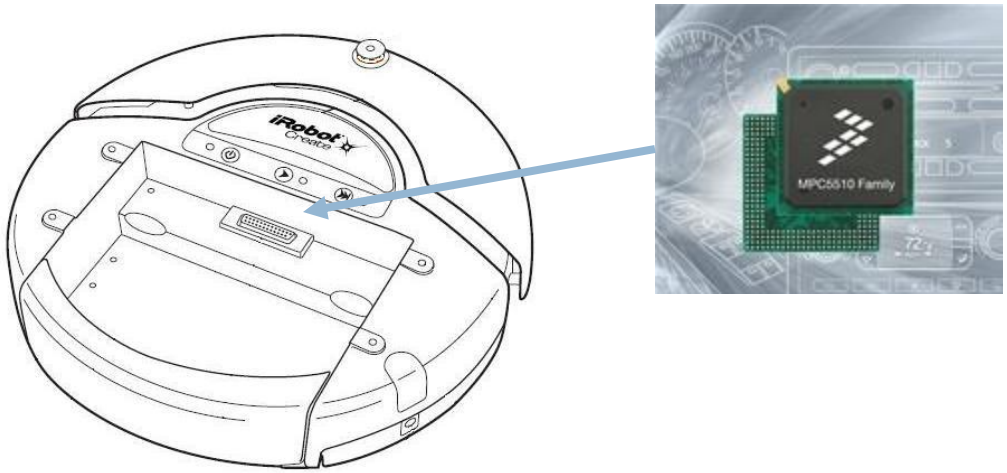


iRobot Create Bottom View

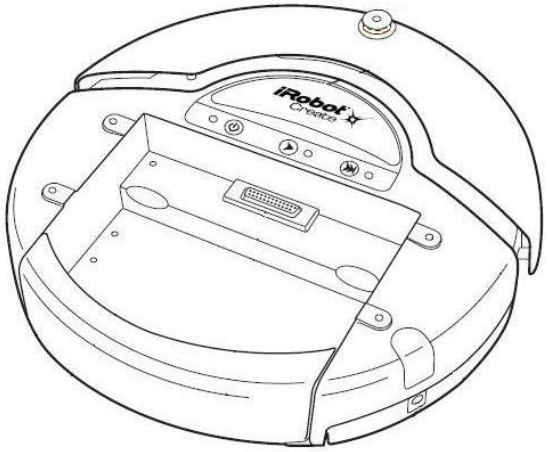


Getting our hands on iRobot Create

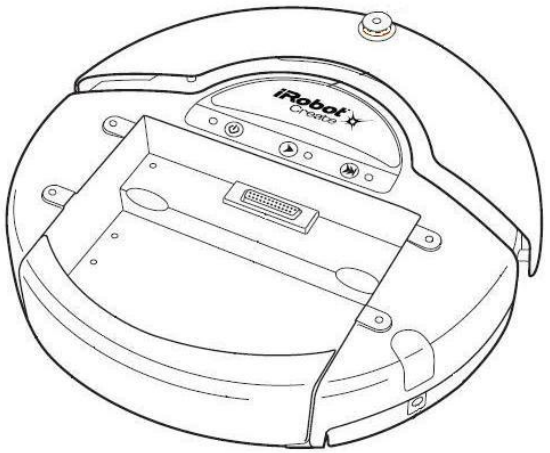
- iRobot Create hardware overview
  - Actuators
  - Sensors
- Sensor signals go to the iRobot microcontroller
- But? The signals need to get to the computer?

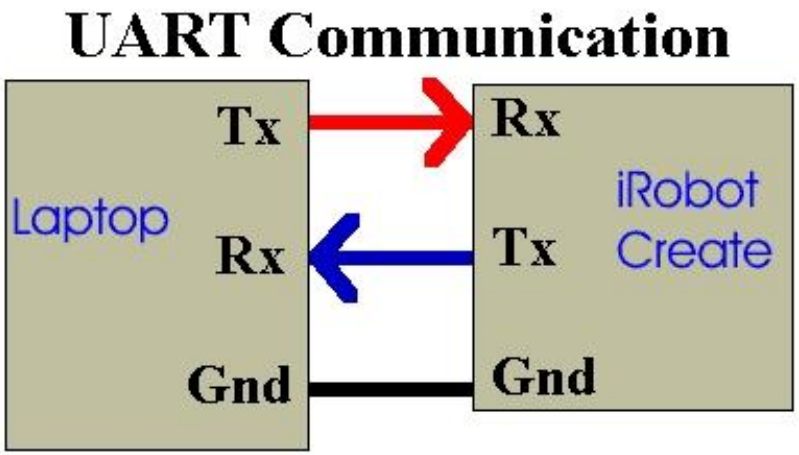


How do we get this information to a PC?



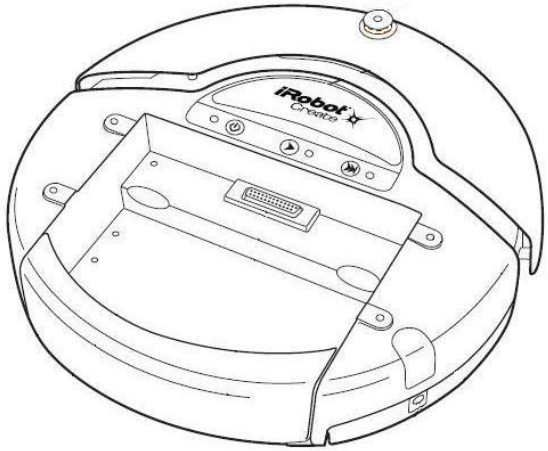
UART Communication

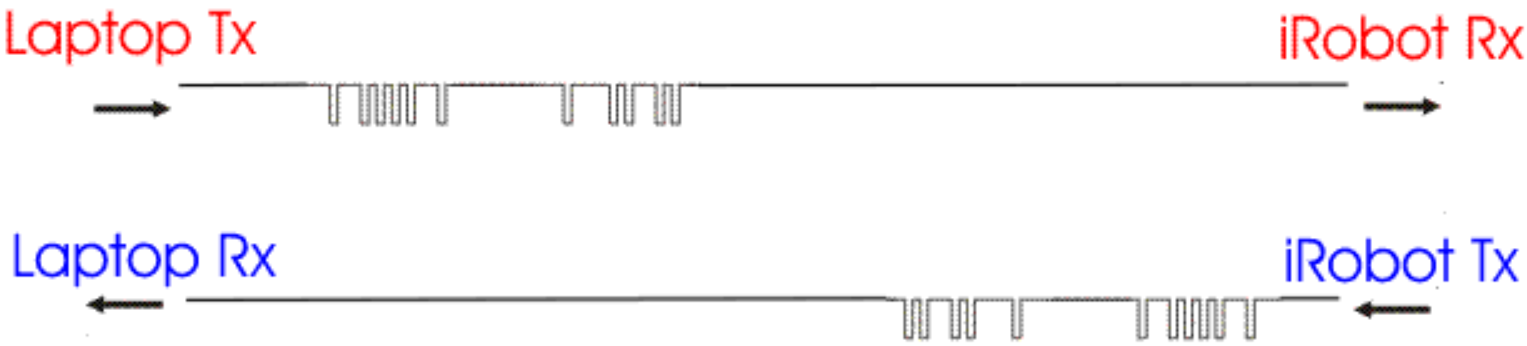




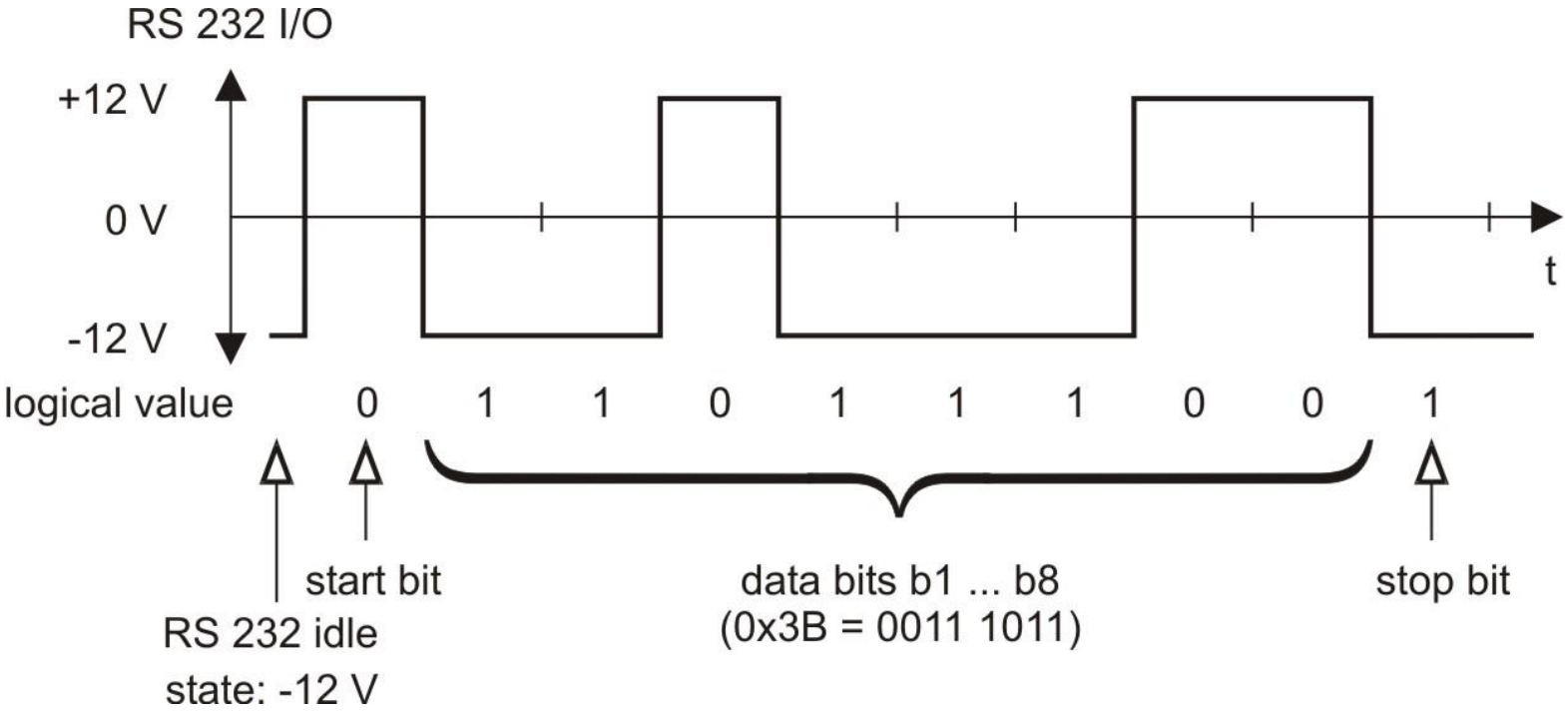
Universal Asynchronous Receiver / Transmitter

Example UART Basics





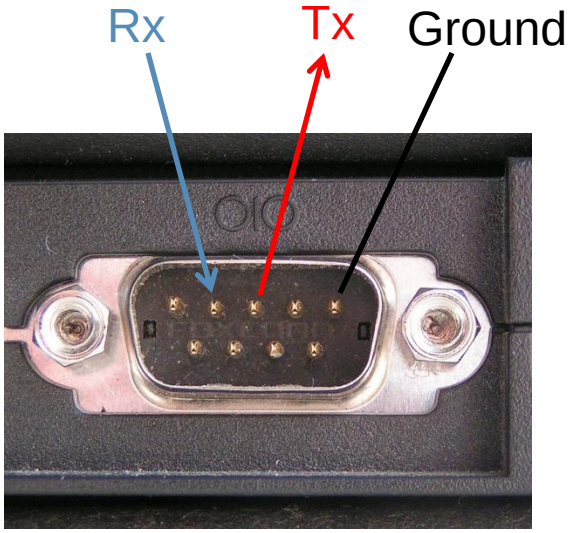
A quick detailed look at UART



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

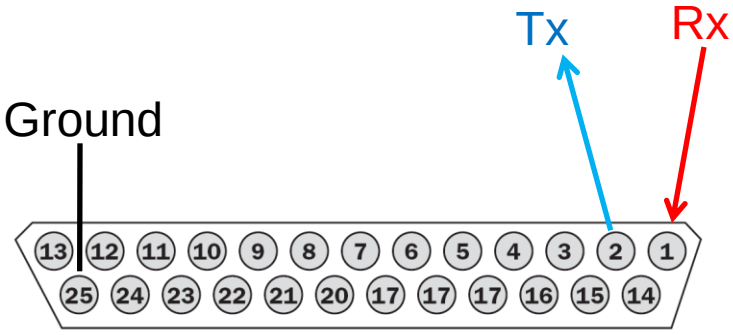
How does UART work?

- 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

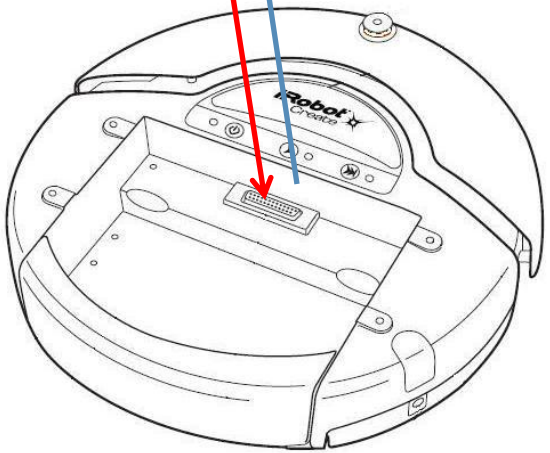
iRobot 25 pin Serial Port

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

How to connect – next time!

Rest of Session

- ***Check your Quiz answers*** versus the solution
  - An assistant may check your Quiz to ensure you are using the Quizzes appropriately
- ***Work on today's homework***
  - Ask questions as needed!
- **Sources of help after class:**
  - **Assistants in the CSSE lab**

```
CSSE lab: Moench F-217
7 to 9 p.m.
Sundays thru Thursdays
```

    - And other times as well (see link on the course home page)
  - **Email** `csse120-staff@rose-hulman.edu`
    - You get faster response from the above than from just your instructor