CS424 Real-Time Systems
Fall 2017 Lab Setup

Yiran Zhao
University of Illinois at Urbana-Champaign
zhao97@illinois.edu

Lab Assignments (Machine Problems)

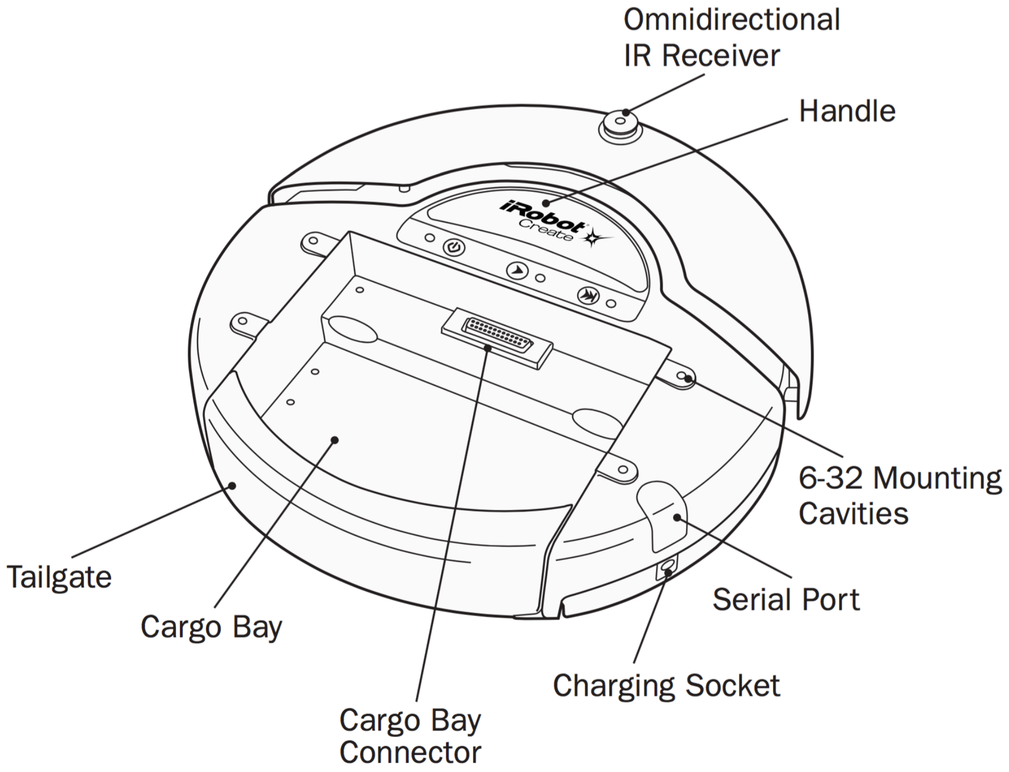
- 4 Lab Assignments
- Form groups of 4 students
- All the lab assignments will use **iRobot create**, **Raspberry Pi**
- Coding Language: C++
- Operating System: Linux (Raspbian)
- You are expected to be comfortable with
  - Linux Terminal, C++, POSIX Threads
- Each group will be given 1 iRobot Create, Interfacing cables, 1 Raspberry Pi, 1 Raspberry Pi Camera, 1 Micro SD card, Power Supply, Chargers, 1 USB-Ethernet adaptor, 1 Ethernet cable
- You are expected to have your own keyboard, mouse, display, HDMI cable (borrow if you don't have)

iRobot Create

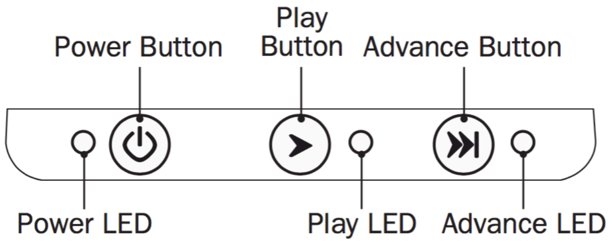
- iRobot Create is a robot development kit
  - Program robot behaviors without worrying about mechanical assembly and low-level code
- Similar to Roomba 400 series Vacuum Cleaner
  - Does not include the vacuum related components
  - Dustbin replaced by cargo-bay
  - Includes Interfacing hardware and additional ports to interface with a computer

iRobot Create

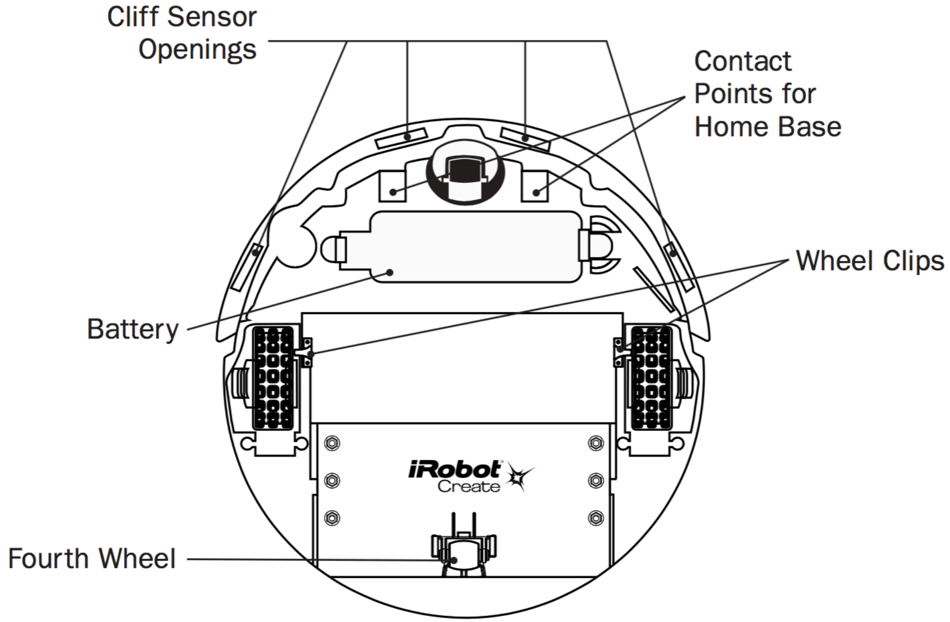
Top View



Buttons and Lights



Bottom View



http://www.irobot.com/filelibrary/create/Create%20Manual_Final.pdf

iRobot Create

- The serial cable plugs into iRobot Create's Mini-Din connector
- iRobot implements a serial protocol called the **Open Interface (OI)**

http://www.irobot.com/filelibrary/create/Create%20Open%20Interface_v2.pdf

- You can open a **serial connection** to the robot, and command it to perform actions, or read sensor values
  - Demo commands
  - Driving commands
  - Song commands
  - Read sensor data
  - Cargo Bay connector commands
  - Scripting commands
- OI Commands look like assembly language. Opcode followed by arguments
  - However, for our assignments, we will be using high level libraries in C++

iRobot Open Interface Modes

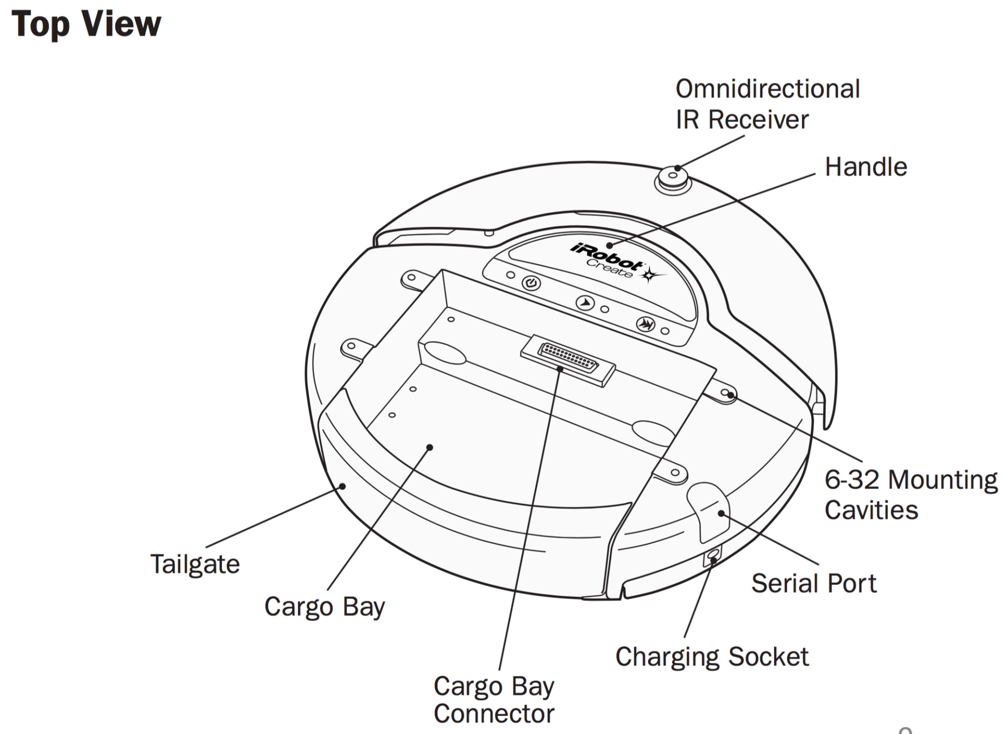
- **Off** – Listens at the default baud rate 57600 for an OI Start command
- **Passive** – Upon sending the **start** command, or the demo command, OI enters into Passive mode.
  - While in Passive mode, you can read sensors, watch Create perform any one of its ten built-in demos, and charge the battery.
- **Safe** – Entered upon sending a **Safe** command to the OI
  - Gives you full control of Create, with the exception of cliff detection, wheel drop, or charger plugged, in which case it reverts back to passive mode.
- **Full** – Entered upon sending a **Full** command to the OI
  - Gives you complete control over Create, all of its actuators
  - All of the safety-related conditions that are restricted are turned off

iRobot Actuators

- Simple Drive Command – 5 bytes
  - 137, 0, 200, 128, 0
  - First: 137 is Drive opcode
  - Second, Third: Velocity High and Low bytes (-500mm/s to 500 mm/s)
    - In this case 0, 200 => 200 mm/s.
    - What velocity 1, 44 correspond to?
  - Fourth, Fifth: Radius High and Low bytes (-2000mm to 2000mm)
    - 128, 0 is a special case, it means Drive straight
    - 1, 244 would mean turn at a radius of 1*256+244 = 500 mm
- Left Wheel Motor and Right Wheel Motor can be independently controlled by “Drive Direct” command (Opcode 145)

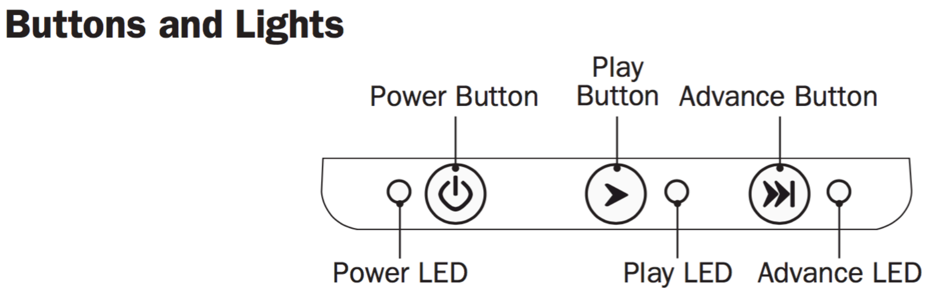
iRobot Actuators (cont)

- LED output
  - Bi-color Power LED, Play LED, Advance LED
- Output on cargo bay connector
  - Digital output
  - PWM and voltage control driver
  - Send IR command through cargo bay
- Speaker (Play songs)



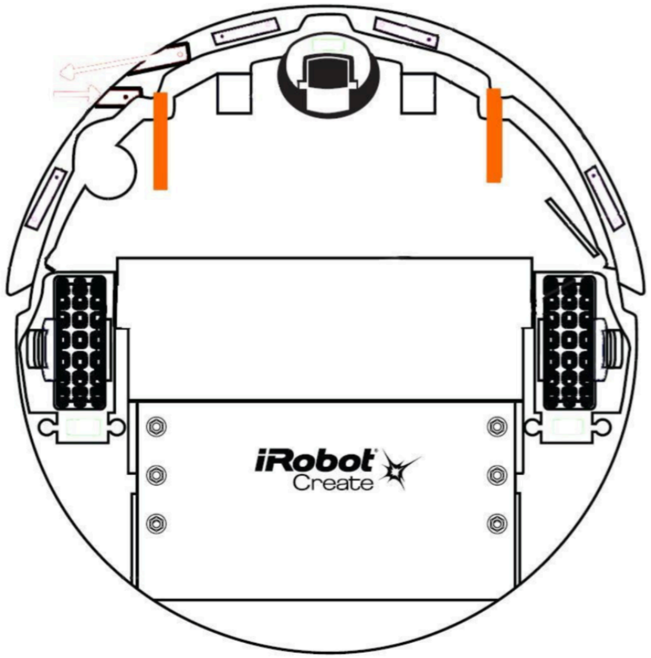
iRobot Sensors

- Play and Advance Buttons
- Omnidirectional IR Sensor
- Left and Right Bumpers
- Three Wheel Drop Sensors
- Four Cliff Sensors
- Wall Sensor
- Distance and Angle Sensor
- Digital and Analog inputs on Cargo bay connector

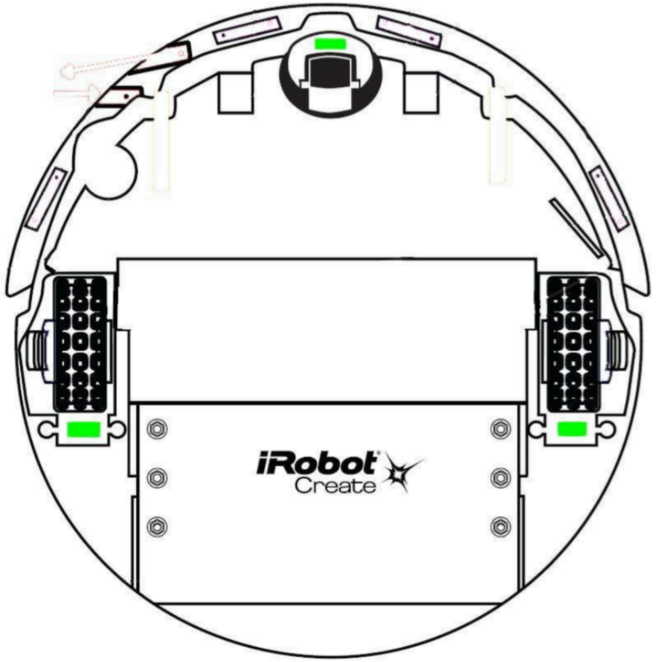


Bump and Wheel Drop

- Left and Right Bumpers



- Front wheel drop
- Left wheel drop
- Right wheel drop



iRobot Sensors Example

- Opcode 142 – Read one sensor
- Opcode 149 – Read multiple sensors
  - 149, 2, 7, 8 reads sensor packet 7 and 8
  - 7 correspond to Bumps and Wheel drops
  - 8 correspond to Wall sensor
- Opcode 148 -- Stream sensor data
- Opcode 150 – Pause / Resume data stream

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

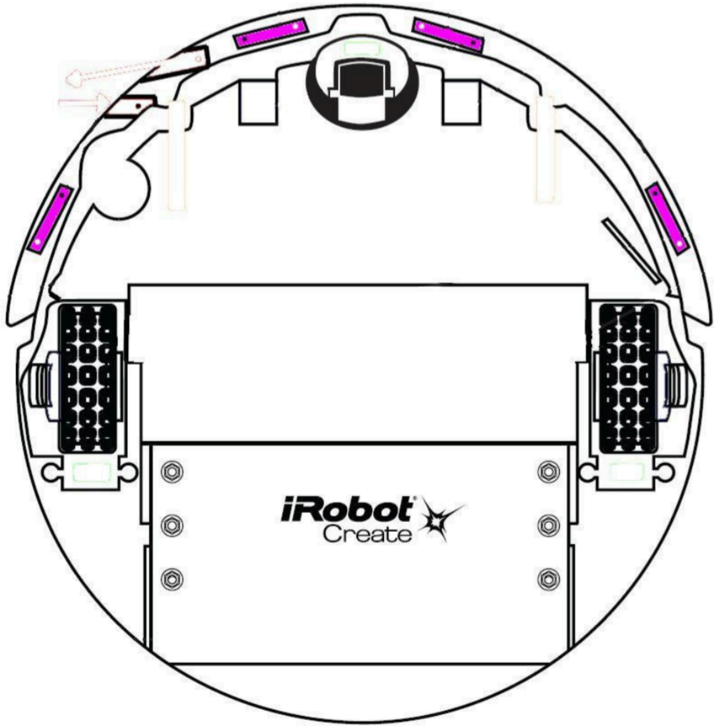
Wall **Packet ID: 8** **Data Bytes: 1 unsigned**

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

Range: 0 – 1

Cliff Sensors

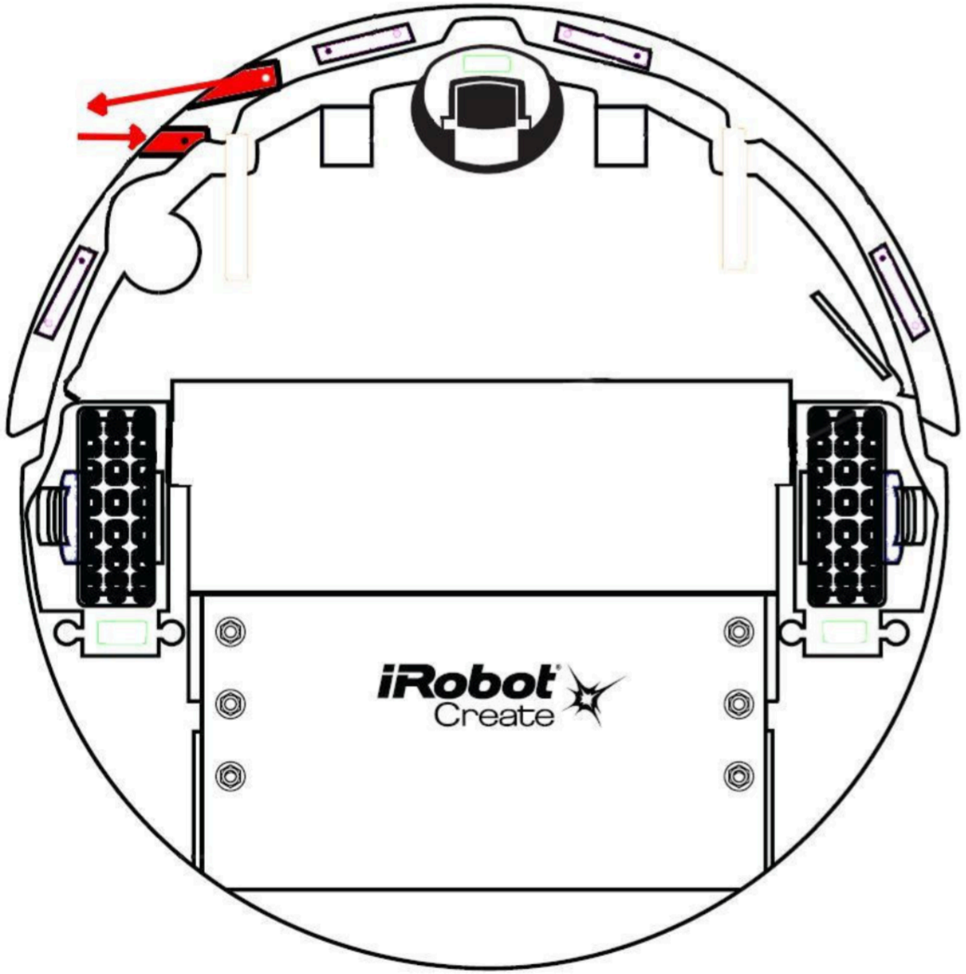
- Cliff sensors detect excessive drops to avoid driving over stairs
- 4 cliff sensors -- Left, Front Left
  Right, Front Right
- Optical cliff sensor - Shines an LED
  onto the ground at an angle and picks it.
  Large drop means reflected light no longer
  detected into the receiver.
- Available as both cliff detection (0-1)
  or sensor values (0-4095)



- http://www.robotappstore.com/Robopedia/Cliff-Sensor

Wall Sensors

- Relates to distance between a wall and iRobot.
- Available as both wall detection (binary) and sensor value (0-4095)



Distance and Angle

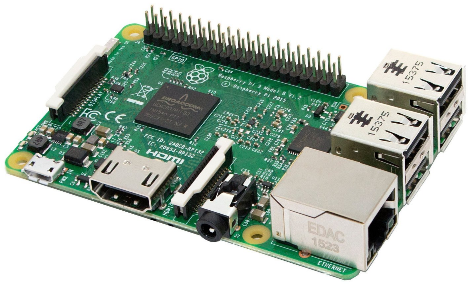
- Distance traveled, angle traveled since last request
- Sum of the distance traveled by both wheels divided by two
- Positive values indicate travel in the forward direction; negative values indicate travel in the reverse direction.
- If the value is not polled frequently enough, it is capped at its minimum or maximum (-32768 – 32767)

Other Sensors

- Other sensors include battery charge, battery temperature, battery voltage, current, infrared byte, requested velocity, requested radius etc
- There are total 43 sensor data packets available
- Please refer to the OI specification for more details

Interfacing with iRobot Create

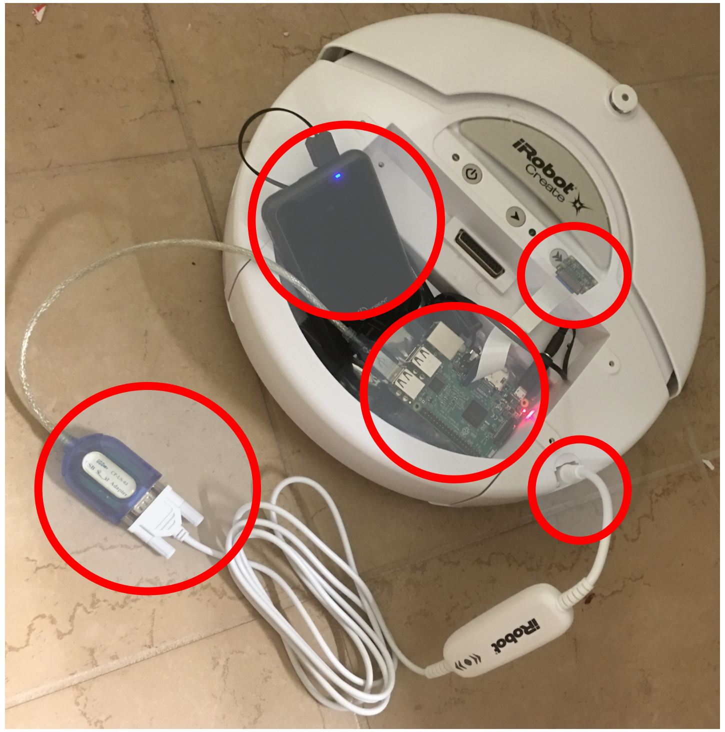
- We can connect a computer, laptop, or microcontroller through the serial port and talk to iRobot using OI protocol.
- Small single board computer
  - We will be using Raspberry Pi 3 Model B
    - 4 cores -- ARM 1.2 GHz, 1 GiB ram, 4 USB ports, HDMI out
    - Includes WiFi, Bluetooth
  - OS: Raspbian – a fork of Debian Linux (Can also install other operating systems)
  - A computer for around $50 (+$30 for camera)





cs424 Lab Setup

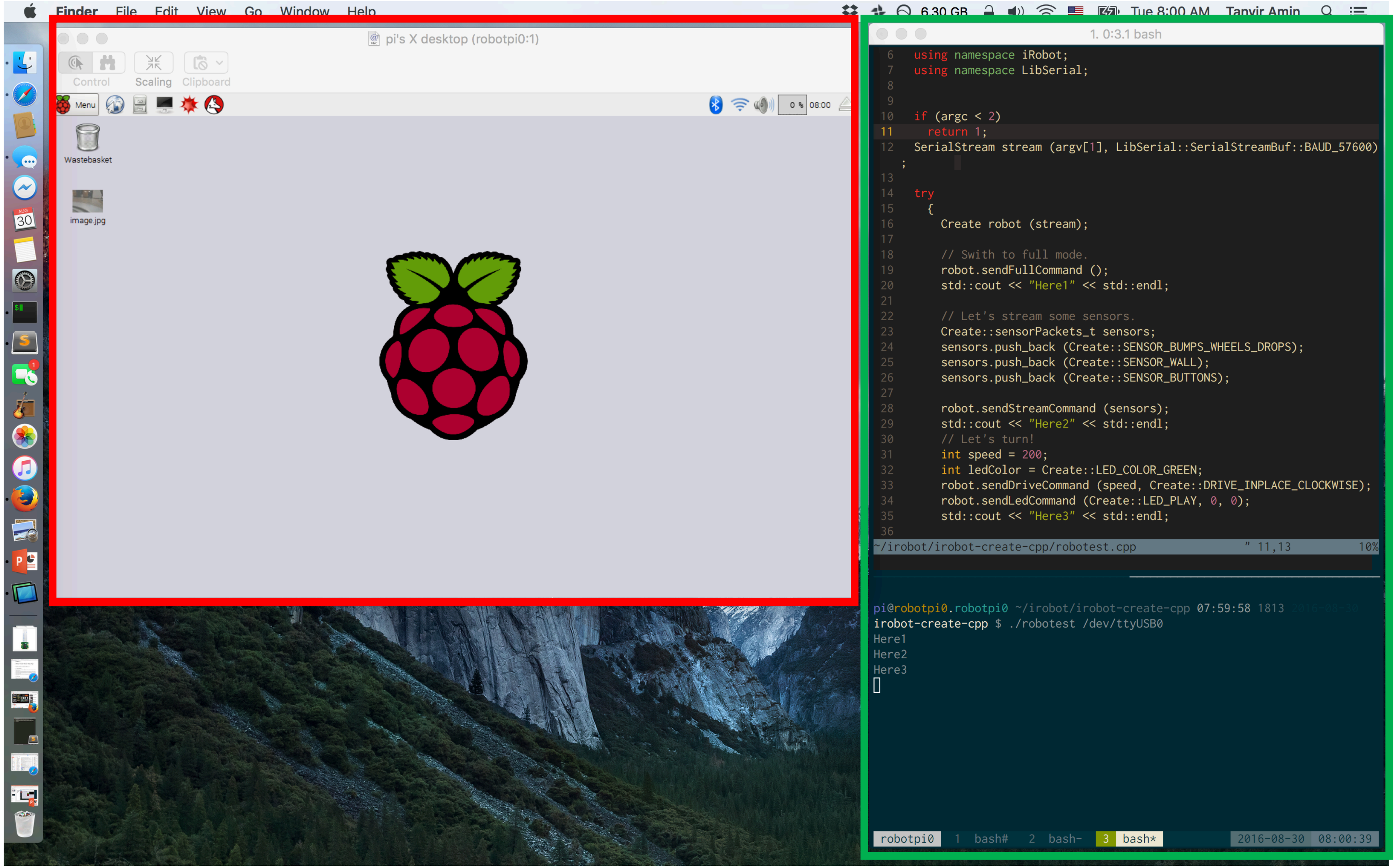
- iRobot Create ➔ Serial Cable ➔ Serial to USB ➔ Raspberry Pi
- Power Raspberry Pi by a battery when testing the iRobot functionality.
  - This gives a “true” robot without any wire.



cs424 Lab Setup

- During code development, it is more convenient to power Raspberry Pi through wall adapter (So that it can stay on).
- Raspberry Pi has WiFi and Ethernet, can also have a camera
- SSH / VNC from your machine to Raspberry Pi and develop there
  - It is assumed that you have an Ethernet Cable or WiFi Router
- (OR) Connect display through HDMI, keyboard, mouse through USB
  - It is assumed that you can access a HDMI supported display, Mouse, Keyboard
- We will provide iRobot Create, Raspberry Pi, Pi Camera, MicroSD Card, Wall Adapter, Interfacing Cables.

VNC and ssh workspace side-by-side



iRobot Create Serial Interface

- Serial Configuration: 7
  - **57600** baud, **8** data bits, **1** stop bit, and **no** flow control.
- We can test with RealTerm (windows), CoolTerm (mac), MiniTerm (linux) to connect through a serial interface and send OI commands.
  - Or we can just use a serial communications library and write programs

Examples

- **To turn on iRobot Create's Play LED only**:
  `128 132 139 2 0 0`
- **To read the state of iRobot Create's left cliff sensor**:
  `128 142 9`
  See how the robot's response changes when you lift it up.
- **To make iRobot Create sing**:
  Send these command sets separately:
  `128 132` (Puts the robot in Full mode)
  `140 0 4 62 12 66 12 69 12 74 36` (Defines the song)
  `141 0` (Plays the song)
- **To make iRobot Create drive forward**:
  CAUTION - place iRobot Create on the ground and be ready to grab it when it reaches the end of its serial cable tether! Send these two command sets separately:
  `128 131`
  `137 0 100 128 0`

iRobot interfacing through Python

- Use pyserial – a serial communications library for python
  - http://pyserial.readthedocs.io/en/latest/
- Simple example
  - Program to drive forward
- Slightly More involved example:
  - Drive but detect bump and rotate accordingly
  - We go through the code in the next slide

Drive with bump detection and rotate

```python
import serial
import time

def send(tty, commands):
    for x in commands:
        tty.write(chr(x))

def main():
    tty = serial.Serial(port='/dev/ttyUSB0', baudrate=57600, timeout=0.01)

    # Full mode
    send(tty, [128, 132])
    time.sleep(1)
    while 1:
        # sleep for 100ms
        time.sleep(100.0/1000.0)
        # Pole for bump,
        # 7: Bump and wheel drop 1 byte
        send(tty, [149, 1, 7])
        inp = tty.read(1)
        if inp:
            bump = ord(inp[0])
            if bump:
                print "Bump, Rotating ..."
                send(tty, [137, 0, 50, 0, 1])
                time.sleep(0.1)
            else:
                # drive straight
                send(tty, [137, 0, 200, 128, 0])

    tty.close()

if __name__ == '__main__':
    main()
```

```
pi@robotpi0.robotpi0 ~/irobot/irobot-create-python 07:53:06 1805
irobot-create-python $ python bump-detection.py
Bump, Rotating ...
Bump, Rotating ...
Bump, Rotating ...
Bump, Rotating ...
Bump, Rotating ...
Bump, Rotating ...
Bump, Rotating ...
Bump, Rotating ...
Bump, Rotating ...
Bump, Rotating ...
Bump, Rotating ...
Bump, Rotating ...
Bump, Rotating ...
Bump, Rotating ...
Bump, Rotating ...
Bump, Rotating ...
Bump, Rotating ...
```

Open serial connection with 57600 baud, 10ms timeout

Initialize and move to full mode

Detect bump every 100 ms

Rotate for 100ms so that we may get rid of the obstacle

If no more bump, then drive straight

Threads

- The robot needs to make decisions in real-time!
- We will be using multithreaded programs for assignments
- Refresher for POSIX threads programming
  - https://computing.llnl.gov/tutorials/pthreads/
- Practice how to create a thread, terminate a thread, join and detach, wait for a thread, set priority, create periodic threads, etc.
- Thread scheduling
  - https://computing.llnl.gov/tutorials/pthreads/man/sched_setscheduler.txt
- We will be working with thread scheduling
  - C++ is the preferred language.

iRobot Interfacing through C++

- Python examples are easy starter, however we need C++ for threading, scheduling, efficient memory usage, and various other reasons.
- libirobot-create
  - Library to talk to iRobot.
  - http://www.nongnu.org/libirobot-create/
  - http://www.nongnu.org/libirobot-create/doc/libirobot-create-0.1.pdf
  - Some code changes needed. We will provide a slightly updated version.
- The library uses libserial
  - Serial communication library
  - http://libserial.sourceforge.net

iRobot Interfacing using C++

```
#include <SerialStream.h>
#include "irobot-create.hh"

int main (int argc, char** argv)
{
  using namespace iRobot;
  using namespace LibSerial;


  if (argc < 2)
    return 1;
  SerialStream stream (argv[1], LibSerial::SerialStreamBuf::BAUD_57600);

  try
    {
      Create robot (stream);

      // Swith to full mode.
      robot.sendFullCommand ();
      std::cout << "Sent Full Command" << std::endl;

      // Let's stream some sensors.
      Create::sensorPackets_t sensors;
      sensors.push_back (Create::SENSOR_BUMPS_WHEELS_DROPS);
      sensors.push_back (Create::SENSOR_WALL);
      sensors.push_back (Create::SENSOR_BUTTONS);

      robot.sendStreamCommand (sensors);
      std::cout << "Sent Stream Command" << std::endl;
      // Let's turn!
      int speed = 200;
      int ledColor = Create::LED_COLOR_GREEN;
      robot.sendDriveCommand (speed, Create::DRIVE_INPLACE_CLOCKWISE);
      robot.sendLedCommand (Create::LED_PLAY, 0, 0);
      std::cout << "Sent Drive Command" << std::endl;

```

The code opens a serial stream to iRobot

Initializes robot object and sets to full mode

Sends stream command to read bump, wheel and button sensors

The robot rotates clockwise (in-place)

iRobot Interfacing using C++

```cpp
      while (!robot.playButton ())
        {
          if (robot.bumpLeft () || robot.bumpRight ())
            std::cout << "Bump !" << std::endl;
          if (robot.wall ())
            std::cout << "Wall !" << std::endl;
          if (robot.advanceButton ())
            {
              speed = -1 * speed;
              ledColor += 10;
              if (ledColor > 255)
                ledColor = 0;

              robot.sendDriveCommand (speed, Create::DRIVE_INPLACE_CLOCKWISE);
              if (speed < 0)
                robot.sendLedCommand (Create::LED_PLAY,
                                      ledColor,
                                      Create::LED_INTENSITY_FULL);
              else
                robot.sendLedCommand (Create::LED_ADVANCE,
                                      ledColor,
                                      Create::LED_INTENSITY_FULL);
            }

          // You can add more commands here.
          usleep(100 * 1000);
        }

      robot.sendDriveCommand (0, Create::DRIVE_STRAIGHT);
    }
  catch (InvalidArgument& e)
    {
      std::cerr << e.what () << std::endl;
      return 3;
    }
  catch (CommandNotAvailable& e)
    {
      std::cerr << e.what () << std::endl;
      return 4;
    }
}
```

Continually scans bump and wall sensors

If the advance button is pressed, changes LED color and reverses the direction of rotation

If play button is pressed, stops the robot

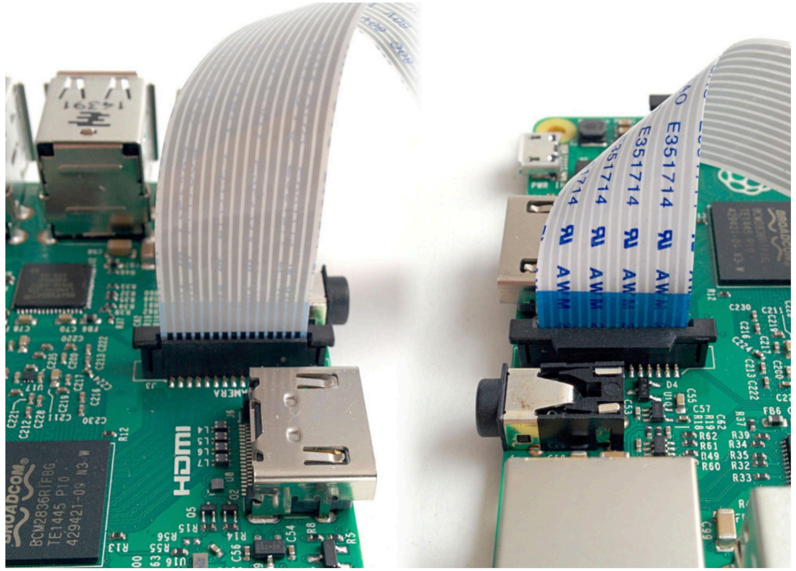
```
g++ -o robotest robotest.cpp irobot-create.cc
-pthread -lserial
```

Our test code

Library implementation

Raspberry Pi Camera

- Getting started with Pi Camera
  - https://www.raspberrypi.org/learning/getting-started-with-picamera/
- PiCamera python library documentation
  - http://picamera.readthedocs.io/en/release-1.12/



Connecting the Pi Camera

1. Lift the lever from camera slot
   - It will not fully come out of the slot,
     but give an opening to insert the cable
2. Insert the blue side facing headphone jack and white side facing HDMI
3. Lift down the lever

RaspiCam and OpenCV

- RaspiCam is a C++ API for using the Raspberry Pi Camera
  - http://www.uco.es/investiga/grupos/ava/node/40
  - https://github.com/cedricve/raspicam
- OpenCV is a Computer Vision Library that we can run even on Raspberry Pi http://opencv.org
- OpenCV Helpful for object detection, facial recognition, etc
- We will use simple code snippets to perform our tasks.
- Helpful instructions on how to install OpenCV on RaspberryPi 3
  - http://www.pyimagesearch.com/2016/04/18/install-guide-raspberry-pi-3-raspbian-jessie-opencv-3/

Links

- iRobot Manual

http://www.irobot.com/filelibrary/create/Create%20Manual_Final.pdf

- iRobot Open Interface Specification

http://www.irobot.com/filelibrary/create/Create%20Open%20Interface_v2.pdf

- Raspberry Pi Setup

http://lifehacker.com/the-always-up-to-date-guide-to-setting-up-your-raspberr-1781419054

- LibSerial C++ http://libserial.sourceforge.net
- Pyserial python serial communications library http://pyserial.readthedocs.io/en/latest/
- Picamera python http://picamera.readthedocs.io/en/release-1.12/index.html
- iRobot-create communications wrapper C++

http://www.nongnu.org/libirobot-create/

http://www.nongnu.org/libirobot-create/doc/libirobot-create-0.1.pdf

- Installing OpenCV on Raspberry Pi

http://www.pyimagesearch.com/2016/04/18/install-guide-raspberry-pi-3-raspbian-jessie-opencv-3/

- RaspiCam C++ http://www.uco.es/investiga/grupos/ava/node/40