

# Setting Up Raspberry Pi, Camera, and iRobot Create

Md Tanvir Al Amin  
maamin2@illinois.edu

Available at <https://courses.engr.illinois.edu/cs424/mp/init.pdf>

## [1. Introduction](#)

### [General Precautions](#)

## [2. Install Raspbian Operating System](#)

### [2.1 Install Raspbian via NOOBS \(New Out Of Box Software\)](#)

#### [2.1.1 Connecting Devices to Raspberry Pi](#)

#### [2.1.2 Installing Raspbian after Powering Up Raspberry Pi](#)

### [2.2 Install Operating System Image without booting up Raspberry Pi](#)

## [3. Set Up Configurations](#)

### [3.1 Setting Up Initial Configurations after Booting Up Raspberry Pi](#)

#### [3.1.1 Change Hostname](#)

#### [3.1.2 Configure WiFi](#)

#### [3.1.3 Configure DHCP Client](#)

#### [3.1.4 Configure Discovery Protocol](#)

### [3.2 Setting Up Initial Configuration through a Wired Network](#)

### [3.3 Setting Up Initial Configurations without Booting Up Raspberry Pi](#)

## [4. Testing the Initial Configuration](#)

## [5. Additional Configurations](#)

### [5.1 Change Password](#)

### [5.2 Setup Timezone and NTP](#)

### [5.3 Install VNC](#)

### [5.4 Adding Periodic Heartbeat](#)

## [6. Interfacing to iRobot Create](#)

### [6.1 Connecting iRobot Create](#)

### [6.2 Install Serial Communications Libraries and Dependencies](#)

## [7. Installing Camera](#)

### [7.1 Install OpenCV](#)

### [7.2 Install RaspiCam](#)

## [8. Testing the System](#)

# 1. Introduction

Raspberry Pi is a small single-board computer with USB, WiFi, Bluetooth, Ethernet, HDMI, Audio, and GPIO connectivity. In cs424, we will be controlling an iRobot create using a Raspberry Pi 3 Model B. This particular model is the most capable compared to the other models. It has 4 cores each clocked to 1.2 GHz, and 1 GB of RAM. The minimum hardware required to run the system is (1) a Raspberry Pi motherboard, (2) a MicroSD memory card, and (3) Power supply. Additionally we will also be using the Raspberry Pi Camera Module v2 as the “vision” for our robot. In this tutorial, we set up Raspberry Pi with an operating system, start it, connect to it, install necessary libraries required for the assignments with the iRobot and the Camera. We will also configure IllinoisNet WiFi, and a simple discovery protocol so that we can develop our software on the Raspberry Pi, and communicate to the robot in a wireless manner.

## General Precautions

Raspberry Pi is a bare motherboard and has the electrical connections exposed. Therefore do not put it on a metallic surface as it may short some terminals. When putting it on the payload bin of the iRobot, note that there are metallic screws there and so take necessary precautions to isolate (for example put it on a paper or a plastic). Try not to touch the pins of the chips by hand whether the pi board is powered on or not. Sometimes the static charge from our body is enough to destroy the chips. Hold the board by the edges and discharge static from your body before setting the board down. When transporting use the anti-static bag that the Raspberry Pi was originally in.

# 2. Install Raspbian Operating System

Before we can boot the pi, we need to install an operating system on the 32 GB MicroSD card. There are many operating systems including different flavors of Linux that can be installed. We will be installing Raspbian operating system, which is a Debian based distribution optimized for Pi hardware (<https://www.raspbian.org>). Because it is Debian based, you can inherit a lot of knowledge from commonly used Ubuntu Linux.

Make sure that you have the MicroSD card and its adapter. Most of the laptops have only the slot for a full sized SD card. The adapter converts the MicroSD card to the form factor of a full-sized card. Connect the card to your Mac, Linux, or Windows laptop. At this point there are two available routes to take.

## 2.1 Install Raspbian via NOOBS (New Out Of Box Software)

This is the easiest method. However, it requires you to have a HDMI display, HDMI Cable, USB Keyboard, and a USB Mouse. You can download NOOBS from the following link. Pick the

"offline and network install" option as it contains the entire contents for the operating system <https://www.raspberrypi.org/downloads/noobs/> Once NOOBS is downloaded, refer to <https://www.raspberrypi.org/documentation/installation/noobs.md> for a description of how to format the SD card and install NOOBS on it. The installation process is basically extracting the zip file and copying it to the card. **After copying NOOBS, properly eject the card from your computer. The card may corrupt if you remove it from the slot without ejecting.** Next step is to boot the Pi.

### 2.1.1 Connecting Devices to Raspberry Pi

We need to connect the devices and power it up. (1) Insert the MicroSD card into the Raspberry Pi. Note that you might need to remove it from the adapter at first as the Pi directly takes the MicroSD form factor. You can find the MicroSD card slot (friction loaded) on the opposite side of the motherboard. (2) Attach one end of an HDMI cable to the Pi and the other end to a display. If that display is an external monitor or a TV, you need to power that up separately. (3) Connect a USB keyboard and a USB Mouse. (4) To keep things simple, ignore the camera for now, (5) Connect the USB 5V power supply. Do not use random USB chargers from your disposal because those may not have sufficient current rating. Use the CanaKit power supply that has been provided for this purpose. Note that there is no "start" or "on" switch / button. Once power supply is connected, it will boot like a computer. If the system has power, it will have a "red" led turned on. The "green" led beside it may blink intermittently, which indicates activity on the MicroSD card.

### 2.1.2 Installing Raspbian after Powering Up Raspberry Pi

Once it boots, you need to follow the instructions displayed on screen and install Raspbian. The following article illustrates the entire process described in this section in more detail.

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

Once you have Raspbian installed, jump to Section 3 to set up the configurations.

## 2.2 Install Operating System Image without booting up Raspberry Pi

This section describes how to install Raspbian directly on a memory card without using NOOBS. It is slightly involved and may require the use of a terminal if you are using a Linux or a Mac computer. But it doesn't require the use of an HDMI display, a keyboard, and a mouse.

Download Raspbian Jessie image from

<https://www.raspberrypi.org/downloads/raspbian/>

Select the one that says "Full desktop image based on Debian Jessie" (i.e. not Lite version)

Follow the link

<https://www.raspberrypi.org/documentation/installation/installing-images/README.md>

It shows how to install Raspbian installed on the MicroSD card using your Mac, Linux, or Windows machine. We also reproduce those links in this section.

If you want to install Raspbian on the SD card using Mac OS X:

<https://www.raspberrypi.org/documentation/installation/installing-images/mac.md>

If you want to install Raspbian on the SD card using Linux:

<https://www.raspberrypi.org/documentation/installation/installing-images/linux.md>

If you want to install Raspbian on the SD card using Windows:

<https://www.raspberrypi.org/documentation/installation/installing-images/windows.md>

Once you follow the steps, you should have Raspbian Jessie installed on the MicroSD card. Jump to Section 3.

## 3. Set Up Configurations

In this section, you will (1) Change the hostname of the Raspberry Pi to a name unique to your group, (2) Set up WiFi configurations for IllinoisNet Enterprise Network and possibly your home network, if you have, (3) Set up a simple discovery protocol for the devices.

If you came here from Section 2.1, it makes sense for you to directly jump to Section 3.1.

If you came here from Section 2.2, you can take either 3.1, 3.2, or 3.3

### 3.1 Setting Up Initial Configurations after Booting Up Raspberry Pi

As you may expect, this method requires having a display, keyboard, and mouse connected to the Raspberry Pi. We will power up the system and edit the configuration files on it. If these devices are not connected or you haven't boot up Pi yet, follow Section 2.1.1 to do so.

#### 3.1.1 Change Hostname

In the following sections, we show the commands required to make the configuration changes. We will be using the `nano` editor. If you are not comfortable with `nano`, you can use any other editor you prefer. Note that `Ctrl + o` saves a file in `nano`, `Ctrl + x` exits `nano`.

Some of the commands require `sudo`. Default user is `pi`, and default password is `raspberrypi`

In this section, we want to set the hostname of your raspberry pi to `robotpiN`. Replace `N` by your assigned group number (For example, hostname will be `robotpi4` for Group 4). To do that, we need to edit two files:

```
sudo nano /etc/hosts
```

There should be a line (most likely the last line) `127.0.1.1 raspberrypi`

Change the term `raspberrypi` to `robotpiN`

**If there is no such line, or if the term after `127.0.1.1` is something else, check if you have opened the correct file.** Use `Ctrl + o` to save the changes, `Ctrl + x` to exit `nano`.

```
sudo nano /etc/hostname
```

Change the term `raspberrypi` to `robotpi`

Use `Ctrl + o` to save your changes, `Ctrl + x` to exit nano

**Note that if you came to this section because you were instructed to follow Section 3.1.1, 3.1.2, 3.1.3, and 3.1.4 when you were in Section 3.3** (which happens if you have mounted the MicroSD card on a linux machine as opposed to attaching it to the Raspberry Pi), **then the files you should be editing are `./etc/hosts` and `./etc/hostname` (remember the leading dot), provided you correctly performed a `cd` (change directory) to the filesystem of the Raspbian on the MicroSD card.** The same rule (i.e there should be a leading dot) also apply for the files edited in Section 3.1.2, 3.1.3, and 3.1.4.

### 3.1.2 Configure WiFi

Execute `sudo nano /etc/wpa_supplicant/wpa_supplicant.conf`

Go to the bottom of the file, and add a section like the following. This setting will work for most of your home WiFi networks, given you are using WPA-PSK scheme (which is default these days). You should put appropriate values for the fields named `your_home_wifi_name` and `your_home_wifi_passphrase`. Note the presence of the " quotation marks that should enclose these values.

```
network={
    ssid="your_home_wifi_name"
    psk="your_home_wifi_passphrase"
    key_mgmt=WPA-PSK
}
```

Using the same mechanism, we now configure IllinoisNet Enterprise network. Enterprise network requires both an identity (Your NetId) and a password (Your NetId password). For security reasons, instead of directly putting the password in plaintext, we will be storing the password hash. Execute the following command to generate the hash.

```
echo -n 'your_netid_password' | iconv -t utf16le | openssl md4
```

Note the single quotation marks around the plaintext password. **You must use the single quotation marks around the plaintext password.** Because passwords generally contain special characters, it may not work if you use double quotation or no quotation. The output of this command will look like `(stdin)= 6602f435f01b917388 9a8d3b9bdcfd0b`

Your output will contain some other hexadecimal string instead of `6602...fd0b` depending on your NetId password. We should now execute `history -cw` to remove terminal history as we typed password in plain text in the terminal, and that should not stay in the history.

Once you have the password hash, add the following block to the `wpa_supplicant.conf` file. Replace `6602f435f01b9173889a8d3b9bdcfd0b` by the actual hash you generated,

and `your_net_id` by your `net_id`. Note the absence of quotation marks (") around `hash:6602...fd0b`. Make sure that there is no space between the keyword `hash:` and the hash itself (i.e. the hexadecimal string that your generated from your NetId password)

```
network={
    ssid="IllinoisNet"
    key_mgmt=WPA-EAP
    proto=WPA2
    eap=PEAP
    ca_cert="/etc/ssl/certs/AddTrust_External_Root.pem"
    identity="your_net_id"
    password=hash: 6602f435f01b9173889a8d3b9bdcfd0b
    phase1="peapver=0"
    phase2="MSCHAPV2"
}
```

Use `Ctrl + o` to save your changes, `Ctrl + x` to exit nano

You can also copy the text from <https://courses.engr.illinois.edu/cs424/mp/wpa-supPLICANT.txt>

### 3.1.3 Configure DHCP Client

Default Raspbian Jessie makes the network interfaces manual. We want to configure the interfaces to take IP Address through DHCP. In this section we edit the file `/etc/network/interfaces`

Open the file by running

```
sudo nano /etc/network/interfaces
```

Find the words "manual" in the file, and change those by "dhcp". There should be three such instances. Finally the file should look like following. We have highlighted the changes:

```
# interfaces(5) file used by ifup(8) and ifdown(8)

# Please note that this file is written to be used with dhcpcd
# For static IP, consult /etc/dhcpcd.conf and 'man dhcpcd.conf'

# Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d

auto lo
iface lo inet loopback

iface eth0 inet dhcp
```

```
allow-hotplug wlan0
iface wlan0 inet dhcp
    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

```
allow-hotplug wlan1
iface wlan1 inet dhcp
    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

Use `Ctrl + o` to save your changes, `Ctrl + x` to exit nano

### 3.1.4 Configure Discovery Protocol

We want to configure a simple discovery protocol, so that we can easily find the IP address of our Raspberry Pi, once connected to IllinoisNet wireless network. We want to send a message to the server `apollo3.cs.illinois.edu` coordinating the protocol, whenever a network interface is up on our Raspberry Pi. **Note that if the changes mentioned in Section 3.1.3 are not made, the discovery protocol configured in this section might not work.**

```
sudo touch /etc/network/if-up.d/robotpi
sudo chmod 755 /etc/network/if-up.d/robotpi
```

Open the file by running

```
sudo nano /etc/network/if-up.d/robotpi
```

Add the following lines

```
#!/bin/sh
curl --data "hostname=`/bin/hostname`&data=`/sbin/ifconfig`" \
http://apollo3.cs.illinois.edu/robotpi/controller.py/send_heartbeat
```

Notice the backtick ( ``` ) and the double quotation ( `"` ) symbols. For your convenience you can copy the text from <https://courses.engr.illinois.edu/cs424/mp/ifupd-robotpi.txt>

Use `Ctrl + o` to save your changes, `Ctrl + x` to exit nano.

At this point, we are done with the initial configuration. Reboot Pi by `sudo reboot now`, and Jump to Section 4 to test our configuration.

## 3.2 Setting Up Initial Configuration through a Wired Network

This method is almost same as Section 3.1. In this method, we still need to boot up Raspberry Pi. Review Section 2.1.1 on how to boot up a Raspberry Pi. Instead of connecting a HDMI

Display, Keyboard, and Mouse, connect an Ethernet cable to the respective port of the Pi. Connect the other end of the Ethernet cable to a Wireless Router that has DHCP running. You can also connect it to your desktop/laptop machine if you do not have a Wireless Router.

After powering it up, wait a minute or two for Raspbian to completely boot. Now we can ssh into it. If you connected the Ethernet cable to a wireless router, you need to visit the web console of the router to find the IP Address of the Pi. Suppose that IP Address is 192.168.xxx.yyy.

You can now ssh to the Pi by using `ssh pi@192.168.xxx.yyy`

It will ask for a password. Default username is `pi`, and the default password is `raspberrypi`

Depending on the settings of your router, you can skip the IP Address step, and might be able directly find the Pi by name `raspberrypi.local` and connect using

```
ssh pi@raspberrypi.local
```

If you do not have a wireless router, you have connected the Pi directly to your machine's Ethernet port. In this case, you have created a private wired network. Your machine is likely to get an autoconfiguration IP of the format 169.254.xxx.yyy. You may be able to directly find the Pi by the name `raspberrypi.local`. If that does not work, you need to find the autoconfiguration IP of the Raspberry Pi. Run the command `sudo arp-scan -l`

This will likely find the IP Address of the Raspberry Pi. You may not have `arp-scan` installed in your machine. In that case you need to install it at first. For Linux, it is available from your package manager. For OS X, install it from homebrew. For Windows, you need to find an ARP scanning software.

Once you have been able to ssh into the Raspberry Pi, follow Section 3.1.1, 3.1.2, 3.1.3, and 3.1.4 to make configuration changes. Once done, jump to Section 4 to test our configuration.

### 3.3 Setting Up Initial Configurations without Booting Up Raspberry Pi

This method doesn't require the use of a display, keyboard, and mouse. You do not even need to boot up the Pi to set the initial configurations. Rather we will be editing the configurations directly on the MicroSD card. **Because Raspbian on the MicroSD card uses ext4 filesystem, you need to have Linux running on your machine. If your machine doesn't have a card reader (common scenario for desktop machines), you need to use a USB Card Reader.** If you do not have Linux installed on your machine, you can use a Ubuntu Live CD/DVD to temporarily boot it up on Linux without actually installing Linux. If your machine is a laptop, it may not have a CD/DVD drive, in which case you can create a Ubuntu Live USB and use that to temporarily boot Linux.

If you are using Mac OS X, the following article shows how to create a bootable USB

<http://business.tutsplus.com/tutorials/how-to-create-a-bootable-ubuntu-usb-drive-for-mac-in-os-x--cms-21253> on Mac. To boot from USB on a Mac, connect the USB drive to it, restart the machine, press and hold Option key immediately upon hearing the startup chime. Release the key after Startup Manager appears and gives you option to boot from the USB. If you are using



a windows computer, you can use the software in <https://rufus.akeo.ie> to create a bootable USB. Note that it might be possible to install drivers for ext4 filesystem on your Windows or OS X operating system, and edit the configuration files on the MicroSD card without using Linux. For OS X, such drivers have been reported to be unstable and cause other side effects including corrupting the filesystem of the MicroSD card. Therefore, we do not take that route.

Once you are running Linux on your machine, insert the MicroSD card (through the MicroSD to SD adapter if required). It will possibly mount it on `/media` or `/media/ubuntu`. Inside the mounted location, there will be two directories. One of those will be named “boot”. We are not interested in the “boot” directory. But we need to edit some files inside the other directory.

That directory can have different names. The name can be either (i) a string of hexadecimal characters and dashes related to the MicroSD card, or (ii) the name can also be “root”. Open a terminal, and change directory like following. Only one of these will work

```
cd /media/ubuntu/hexadecimal_string_related_to_the_memory_card/
(OR)
cd /media/hexadecimal_string_related_to_the_memory_card/
(OR)
cd /media/ubuntu/root
(OR)
cd /media/root
```

This directory corresponds to the filesystem of the Raspbian installed on the MicroSD card. Once we are inside the proper directory as mentioned above, we need to change the following files. Note the leading dot (.) as these paths are relative to the present directory.

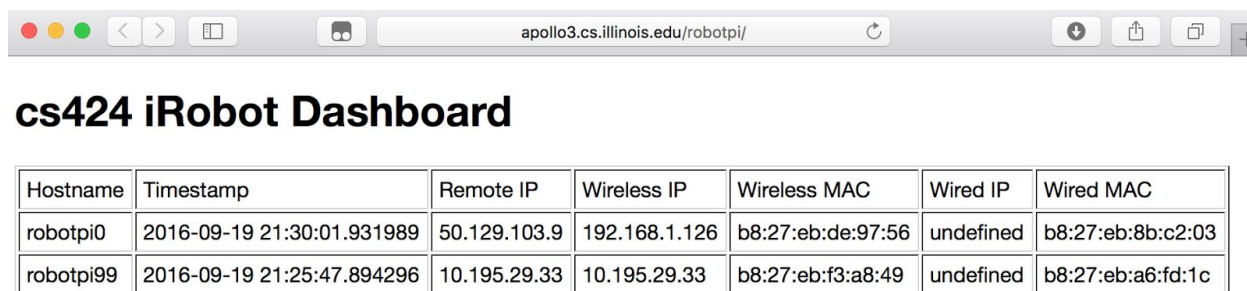
- ⇒ Edit `./etc/hostname` (See Section 3.1.1)
- ⇒ Edit `./etc/hosts` (See Section 3.1.1)
- ⇒ Edit `./etc/wpa_supplicant/wpa_supplicant.conf` (See Section 3.1.2)
- ⇒ Edit `./etc/network/interfaces` (See Section 3.1.3)
- ⇒ Create a file `./etc/network/if-up.d/robotpi` and set appropriate permissions. (See Section 3.1.4)

These files correspond to the files edited in Section 3.1.1, 3.1.2, 3.1.3, and 3.1.4. Follow those sections to make the changes. **Remember the leading dot (.) in the paths here.** We need to edit files on the MicroSD card corresponding to the Raspberry Pi’s filesystem. **If you forget the leading dot, you will be editing files of the host Linux operating system instead, which can result in unexpected problems.** Once done, exit the terminal, make sure no other app is using the MicroSD card, unmount and safely eject the MicroSD card. Jump to Section 4 to test our configuration.

## 4. Testing the Initial Configuration

In this section we test our configuration. If you haven't attached the MicroSD card to the Raspberry Pi yet (which can happen if you followed Section 2.2 and 3.3), now is the time to do so. Review section 2.1.1 on how to connect devices to the Raspberry Pi. In this section, we test whether we can connect to the Raspberry Pi in a wireless setting by connecting only the (1) MicroSD card, (2) CanaKit 5V USB Power Supply.

Wait a few minutes for the Pi to completely boot up. Now open a browser on your computer and visit <http://apollo3.cs.illinois.edu/robotpi>. It will come up with a dashboard page like Figure 4.1. This dashboard is important when working in-campus, as the assigned IP Address through IllinoisNet WiFi is dynamic.



The screenshot shows a web browser window with the address bar containing `apollo3.cs.illinois.edu/robotpi/`. The page title is **cs424 iRobot Dashboard**. Below the title is a table with the following data:

Hostname	Timestamp	Remote IP	Wireless IP	Wireless MAC	Wired IP	Wired MAC
robotpi0	2016-09-19 21:30:01.931989	50.129.103.9	192.168.1.126	b8:27:eb:de:97:56	undefined	b8:27:eb:8b:c2:03
robotpi99	2016-09-19 21:25:47.894296	10.195.29.33	10.195.29.33	b8:27:eb:f3:a8:49	undefined	b8:27:eb:a6:fd:1c

Figure 4.1: Dashboard for the discovery protocol

If you have set up the configurations correctly, the table should contain a row with information related to your Raspberry Pi. That row should have the unique hostname you gave it in Section 3 in the first column. The timestamp should be recent (as you have just boot the machine). Check the wireless IP. This is the present IP Address of your Raspberry Pi when connected through WiFi. Check if you can ssh to it by running the following (replace `10.195.29.33` by the actual IP Address displayed on the dashboard)

```
ssh pi@10.195.29.33
```

If you haven't changed the password, the default password should be `raspberry`. If everything worked up to this point, anytime you need the IP Address dynamically assigned to your Raspberry Pi, you should visit <http://apollo3.cs.illinois.edu/robotpi> to find it.

Note that the command to safely shutdown the Raspberry Pi is `sudo shutdown now`. The command to restart immediately is `sudo reboot now`.

## 5. Additional Configurations

In this section we add additional configurations to make development on the Raspberry Pi and debugging easier. For the rest of this tutorial, it doesn't matter whether you are running the Pi headless through ssh or by connecting a display, keyboard, and mouse.

### 5.1 Change Password

You can change the password of the user `pi` from default `raspberry` to something else by using the command `passwd`

### 5.2 Setup Timezone and NTP

Run `sudo dpkg-reconfigure tzdata` to set the timezone on the Pi (US/Central)

Run `sudo apt-get install ntp` to install network time protocol for synchronization.

### 5.3 Install VNC

For remote access, so far we have connected to and controlled the Pi through `ssh`. Sometimes we need to access the Raspberry Pi desktop GUI. You can install VNC for this purpose. Follow <https://www.raspberrypi.org/documentation/remote-access/vnc/> to install `vnc`.

### 5.4 Adding Periodic Heartbeat

We want the Pi to send periodic heartbeat messages to the coordinator server `apollo3.cs.illinois.edu` so that we can monitor whether it is running and connected.

Log in to user `pi` (Your main working account) on the Raspberry Pi and execute

```
crontab -e
```

It will create a cron job file for the user `pi`. Add the following line at the bottom. This will send heartbeat information every 15 minutes to the coordinator server. **Note that it is a single line.**

```
*/15 * * * * curl --data
"hostname=`/bin/hostname`&data=`/sbin/ifconfig`"
http://apollo3.cs.illinois.edu/robotpi/controller.py/send_heartbeat
```

Now the Timestamp field corresponding to your hostname in Figure 4.1 will update every 15 minutes. The text is also available <https://courses.engr.illinois.edu/cs424/mp/ctab-extra.txt>

## 6. Interfacing to iRobot Create

iRobot create is a robotics platform we will be using for our projects. iRobot defines an API through the serial interface called OI (Open Interface). We will be using OI to read its sensors, and control the actuators. The reference manual for OI is available at

[http://www.irobot.com/filelibrary/create/Create%20Open%20Interface\\_v2.pdf](http://www.irobot.com/filelibrary/create/Create%20Open%20Interface_v2.pdf)

Instead of directly writing code using OI opcodes and arguments, we will be using helper libraries that wrap the API for the sake of easier coding and development. For details on the iRobot platform, review the presentation slides on the cs424 lab setup. The slides are available at <https://courses.engr.illinois.edu/cs424/lectures/cs424-lab-setup.pdf>. In this section, we connect the iRobot to Raspberry Pi, and install python and C++ drivers.

### 6.1 Connecting iRobot Create

iRobot create comes with a yellow colored Roomba battery. The battery compartment is at the bottom of iRobot, which will initially contain a green colored container (Fig 6.1a). This container is used to hold AA batteries that can power iRobot. Note that, we won't be using AA batteries. So eject the green container from the battery compartment (Fig 6.1a), install the yellow battery (Fig 6.1b), and connect the charger to charge it. Refer to the manual of iRobot [http://www.irobot.com/filelibrary/create/Create%20Manual\\_Final.pdf](http://www.irobot.com/filelibrary/create/Create%20Manual_Final.pdf) for details. While iRobot is charging the power light will pulse in orange. Once fully charged, the power light will become green. Charge the battery to full before first using it.



Fig 6.1(a) iRobot create may initially contain a green colored AA battery container.



Fig 6.1(b) The correct battery is the Yellow Roomba battery that you need to install

Note that the robot will not start if the charger is connected. Once the battery is charged, disconnect the charger, connect the RS-232 adapter in the serial port. Connect that serial port to the Raspberry Pi by using the provided Serial to USB adapter. The entire setup will look like Figure 6.2a or 6.2b. Note that in this figure, the Raspberry Pi is powered up from a battery (battery required for mobility, but for now you can use the wall charger), and the camera is also connected (ignore camera for now, we will visit camera in detail in Section 7).



Fig 6.2(a) Connecting iRobot to Pi using the Serial connector and USB-Serial Adapter



Fig 6.2(b) A completely mobile assembly of the iRobot and Raspberry Pi

Once the connection is complete, power up the Raspberry Pi and ssh to it. Run the command `lsusb`. It should include the following line, which means it detected the USB-Serial adapter.

```
Bus xx Device yy: ID zz Prolific Technology, In c. PL2303 Serial Port
```

If you do not have any other USB device connected to the Raspberry Pi, running `ls /dev/tty*` or `ls /dev/ttyUSB*` will include a file `/dev/ttyUSB0`, which corresponds to the iRobot Create. If you have other USB devices connected to the Pi, you need to figure out which one corresponds to the USB-Serial adapter connecting the iRobot.

## 6.2 Install Serial Communications Libraries and Dependencies

ssh to Raspberry Pi and execute the following commands to install the related libraries. Note that some of these will take time, so it is better to execute from a tmux or screen terminal so that even though your ssh connection to Raspberry Pi disconnect, the installation continues.

Table I: Installing Serial Communications Libraries

Package	Installation Command
boost	<code>sudo apt-get install libboost-all-dev</code>
SIP	<code>sudo apt-get install python-sip</code> <code>sudo apt-get install python-sip-dev</code>
pyserial	<code>sudo apt-get install python-pip</code> <code>sudo pip install pyserial</code>
libserial	<code>wget http://downloads.sourceforge.net/project/libserial/\</code> <code>libserial/0.6.0rc2/libserial-0.6.0rc2.tar.gz</code> <code>tar -zxvf libserial-0.6.0rc2.tar.gz</code> <code>cd libserial-0.6.0rc2/</code> <code>./configure</code> <code>make</code> <code>sudo make install</code> <code>sudo ldconfig</code>

## 7. Installing Camera

In this section, we connect the attached camera to the Raspberry Pi. Turn off your Pi, get the camera out from its packaging, and follow one of the links to connect and test it

<https://www.raspberrypi.org/learning/getting-started-with-picamera/worksheet/> (GUI based)

<https://www.raspberrypi.org/documentation/configuration/camera.md> (Command line)

### 7.1 Install OpenCV

OpenCV (<http://opencv.org>) stands for Open Source Computer Vision. It is highly efficient, and known to work well with real-time applications including robotics. Written in C++, OpenCV has C++, C, Python, and Java interface. For our projects, we will use OpenCV to process feed from the Raspberry Pi Camera for obstacle detection and other applications. In this section, we install OpenCV 3 in Raspbian.

```
sudo apt-get update
sudo apt-get upgrade
```

The second command may take more than an hour because it will upgrade everything.

#### Installing some developer tools

```
sudo apt-get install build-essential cmake pkg-config
```

#### Installing image compression libraries

```
sudo apt-get install libjpeg-dev libtiff5-dev
```

```
sudo apt-get install libjasper-dev libpng12-dev
```

### Installing video io libraries

```
sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev
sudo apt-get install libv4l-dev libxvidcore-dev libx264-dev
```

### Installing gui modules

```
sudo apt-get install libgtk2.0-dev
```

### Installing optimization libraries

```
sudo apt-get install libatlas-base-dev gfortran
```

### Install python bindings and dependencies

```
sudo apt-get install python2.7-dev python3-dev
sudo apt-get install python-pip
sudo pip install numpy
```

### Download OpenCV 3 and extra contributions

```
cd ~
wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.1.0.zip
unzip opencv.zip
wget -O opencv_contrib.zip https://github.com/Itseez/opencv_contrib/archive/3.1.0.zip
unzip opencv_contrib.zip
```

### Compile OpenCV 3

```
cd ~/opencv-3.1.0/
mkdir build
cd build
cmake -D CMAKE_BUILD_TYPE=RELEASE \
      -D CMAKE_INSTALL_PREFIX=/usr/local \
      -D INSTALL_PYTHON_EXAMPLES=ON \
      -D OPENCV_EXTRA_MODULES_PATH=~/.opencv_contrib-3.1.0/modules \
      -D BUILD_EXAMPLES=ON ..
make -j4
```

The last `make -j4` command is a time consuming step as it compiles OpenCV. It uses 4 Raspberry Pi cores, and can take from 1~1.5 hours. In case the parallel compile errors out for race conditions, you can clean the build by executing `make clean` and then using `make` to compile using a single core. `make` will take more time as opposed to `make -j4`

### Install OpenCV libraries

```
sudo make install
sudo ldconfig
```

## 7.2 Install RaspiCam

Raspicam (<https://github.com/cedricve/raspicam>) is a C++ API for the Raspberry Pi camera that makes use of OpenCV. We use RaspiCam to control the camera and directly receive OpenCV image matrix from the camera feed without actually storing those to disk at first. In this section we install RaspiCam.

```
git clone https://github.com/cedricve/raspicam
cd raspicam
mkdir build
cd build
cmake ..
```

Check the output of cmake and it should include lines like the following. Make sure `CREATE_OPENCV_MODULE` is configured as 1 (which is highlighted below), which means it has properly detected the OpenCV installation.

```
-- CREATE_OPENCV_MODULE=1
-- CMAKE_INSTALL_PREFIX=/usr/local
...
...
...
-- Configuring done
-- Generating done
-- Build files have been written to: /home/pi/raspicam/trunk/build
```

Now compile and install raspicam

```
make
sudo make install
sudo ldconfig
```

## 8. Testing the System

We have provided an example for you to test the assembly. It will work provided you have completed all the steps mentioned earlier. **At this point, the entire assembly should be mobile. Power Raspberry Pi from the provided external battery. Make sure iRobot's charger is not connected to it, and the system is free to move.** ssh to Pi and execute

```
cd ~
wget https://courses.engr.illinois.edu/cs424/mp/irobot-example.tar.gz
tar -zxvf irobot-example.tar.gz
cd irobot-example
make
```



If the compilation succeeds, run the program `./robotest`

The program will at first initialize camera, robot, etc. Once ready it will send a command to the robot to continuously stream (1) Bump and Wheel Drop sensor, (2) Wall Signal, (3) Buttons. After that, it will command the robot to drive straight at 200 mm/s. Then the robot will continuously look for bumps (with 100ms sleep in between). If the robot has bumped, it will back up straight, rotate, and continue running. At every time, `robotest` will also print at the console what it is doing. If the robot has seen a “wall”, it will take a photo using the raspberry pi camera and save it to a file named `irobot_image.jpg`. If you are using a Mac or a Linux machine, you can copy the photo to your computer using the following command. If you are using windows, you can use WinSCP or similar programs to transfer the photos

```
scp pi@ip_address_of_raspberry_pi:~/irobot-example/irobot_image.jpg .
```

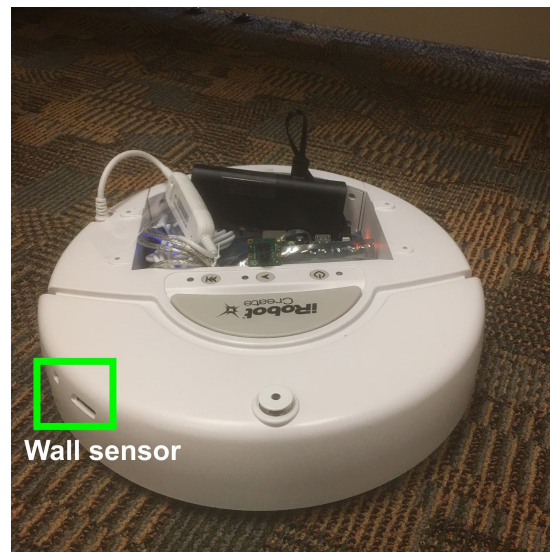


Figure 8.1: Wall sensor

Note the location of the wall sensor in Figure 8.1. The sensor works by transmitting a signal and measuring the strength of the received signal. This type of positioning allows it to detect a wall that is on the side. **You can artificially check the wall sensor by a bringing a dark colored paper near it or taking it away.**

If the “Advance” button is pressed, the robot will start rotating in place. Subsequent presses of the advance button results in reversing the direction of rotation. It will also change the colors of LED. Study the code to make sure you understand it. The program will continue running in the aforementioned manner until the play button is pressed. Once the play button is pressed, the robot will stop and the `robotest` program will exit gracefully. You can also use the power button to turn off the iRobot at any time, but the program `robotest` does not detect such event and as a result it will continue running. But iRobot will not respond as the power has been turned off.