



## Wi-Fi Settings

**SSID:** AcroboticGuest

**Password:** 13meetup37



# Class: Intro to IoT with Raspberry Pi!

# About Makerden

We're an Educational Makerspace where we combine the necessary tools, equipment, and instruction for hands-on learning through building things.

Members of our group vary both widely and wildly in terms of skill level and age range (10 to 68).



# About Makerden

## Events organized at our Educational Makerspace:

**[DIY]** These events include show-and-tells by our members followed by a work-on-your-project(s) and ask-for-help sessions.

**[Workshop]** An instructor leads a hands-on workshop for learning or polishing a practical skill useful for Makers (including schematic-drawing, PCB design, 3D-Printing, soldering).

**[Class]** An instructor leads a formal lecture featuring instructor-led activities that allow participants to learn, develop, or refine a technical skill.



# About ACROBOTIC

## **Makerden's Educational Electronics partner**

Small, bootstrapped Open-Source electronics startup dedicated to the design of hardware and software products for use in education, DIY, hobby, arts, science, and more!

- Online store: [\*\*https://acrobotic.com\*\*](https://acrobotic.com)
- Develops online tutorials: [\*\*http://learn.acrobotic.com\*\*](http://learn.acrobotic.com)
- Supplies all the electronic components for this class!
- Helps Makerden members take their ideas from concept to implementation

# Downloading this presentation

MakerdenIO/Intro\_IoT\_RPi

GitHub, Inc. [US] https://github.com/MakerdenIO/Intro\_IoT\_RPi

**MakerdenIO / Intro\_IoT\_RPi**

**Navigate to:**

**[https://github.com/makerdenio/Intro\\_IoT\\_RPi](https://github.com/makerdenio/Intro_IoT_RPi)**

Code, wiring diagrams, and presentation slides for an IoT Wireless Alarm System using a Raspberry Pi. [Edit](#)

16 commits · 1 branch · 0 releases · 1 contributor

Branch: master

themakerbro re-adding .key presentation file · Latest commit 8acf8b1 22 days ago

| File         | Message                          | Time         |
|--------------|----------------------------------|--------------|
| activity_01  | done                             | 2 months ago |
| activity_02  | done                             | 2 months ago |
| activity_03a | updating                         | a month ago  |
| activity_03b | updating                         | a month ago  |
| activity_04  | adding activities 3 and 4        | 2 months ago |
| activity_05  | activities completed             | 2 months ago |
| activity_06  | activities completed             | 2 months ago |
| activity_07  | activities completed             | 2 months ago |
| activity_08a | updating                         | a month ago  |
| activity_08b | updating                         | a month ago  |
| presentation | re-adding .key presentation file | 22 days ago  |

Code Issues Pull requests Wiki Pulse Graphs Settings

HTTPS clone URL <https://github.com>

You can clone with [HTTPS](#), [SSH](#), or [Subversion](#). [?](#)

Clone in Desktop Download ZIP

Help people interested in this repository understand your project by adding a README.

Add a README

# Class outline

**Duration: 4–5 hrs; Difficulty: Intermediate**

## **Intro and Fundamental Concepts**

Overview of IoT and Raspberry Pi

Project description

## **Setting Up the Hardware**

Access to the GPIO with Python

Controlling an LED, motion sensor, and  
buzzer

Using Python to capture images from a USB  
camera

## **Configuring the Backend**

Raspbian and system administration basics

Running a webserver using Python and Flask

Building an API with Flask for LED control

## **Building a User Interface (Frontend)**

Building a User Interface (UI) with HTML,  
CSS, and JavaScript

## **Piecing Everything Together**

Building a Wireless Alarm System

## **Future Enhancements (discussion)**

Backend communication via WebSockets  
Headless access to your Raspberry  
PiRemote access to the system

# **Intro**

Overview of IoT and Raspberry Pi

Project description

# What is *the* Internet?

The Internet is a global system of interconnected computer networks that communicate with one another to link **billions** of devices worldwide.



*Facebook's Map of World 'Friendship'*

# What devices are found on *the Internet*?

“interconnected computer networks...”



# What is the goal *the* IoT?

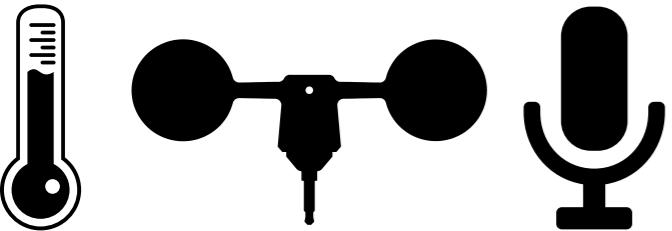
Extending the internet to include “everyday objects”.



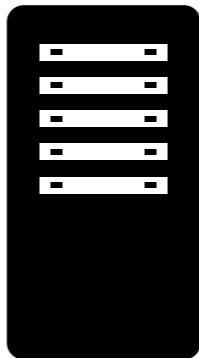
# System Overview

Most IoT devices are comprised by 3 components:

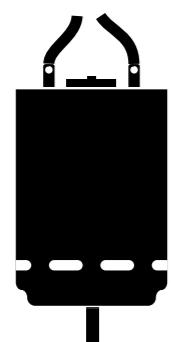
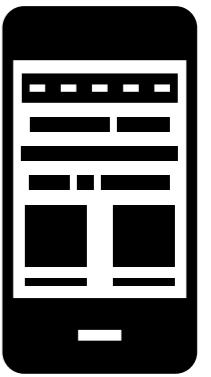
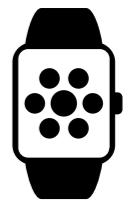
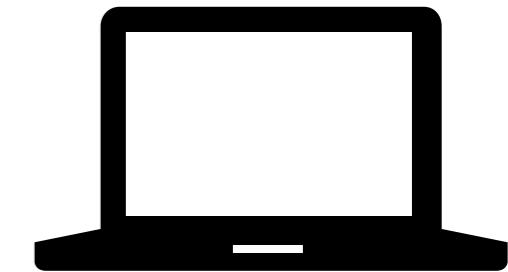
THING



WEB

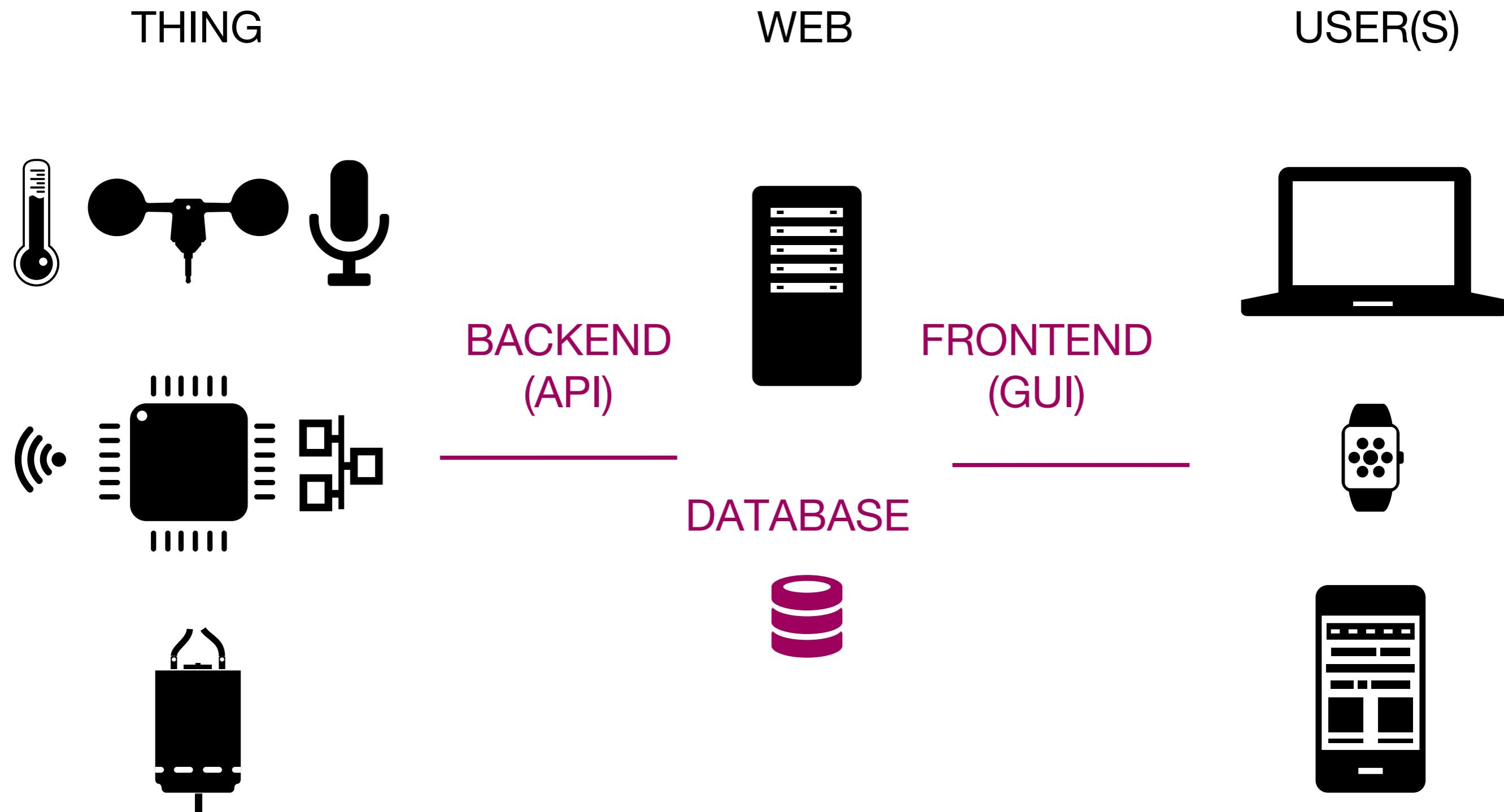


USER(S)



# System Overview

The Web component of an IoT system is comprised by a few sub-components:



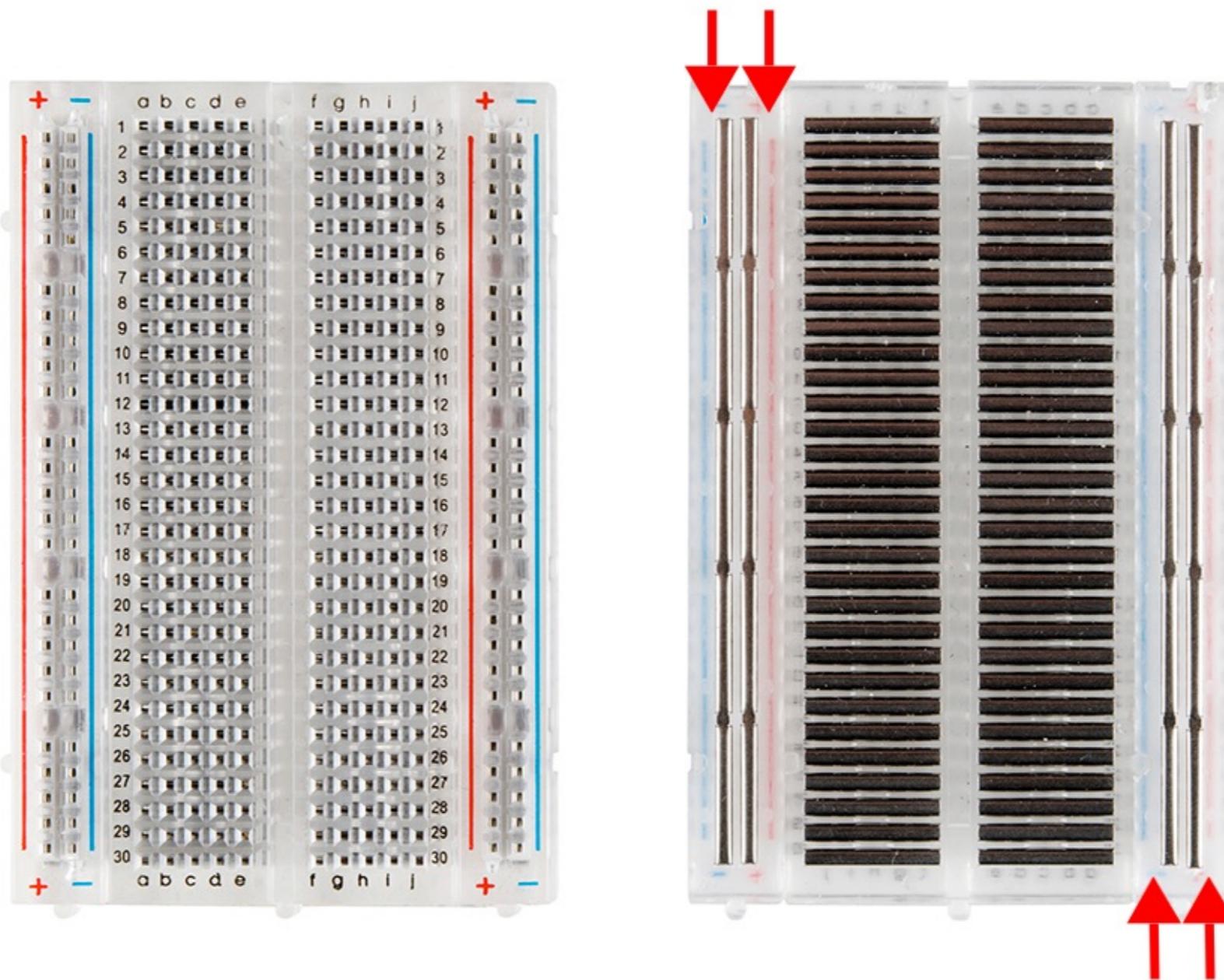
# **Fundamental Concepts**

Hardware prototyping

Linux basic usage

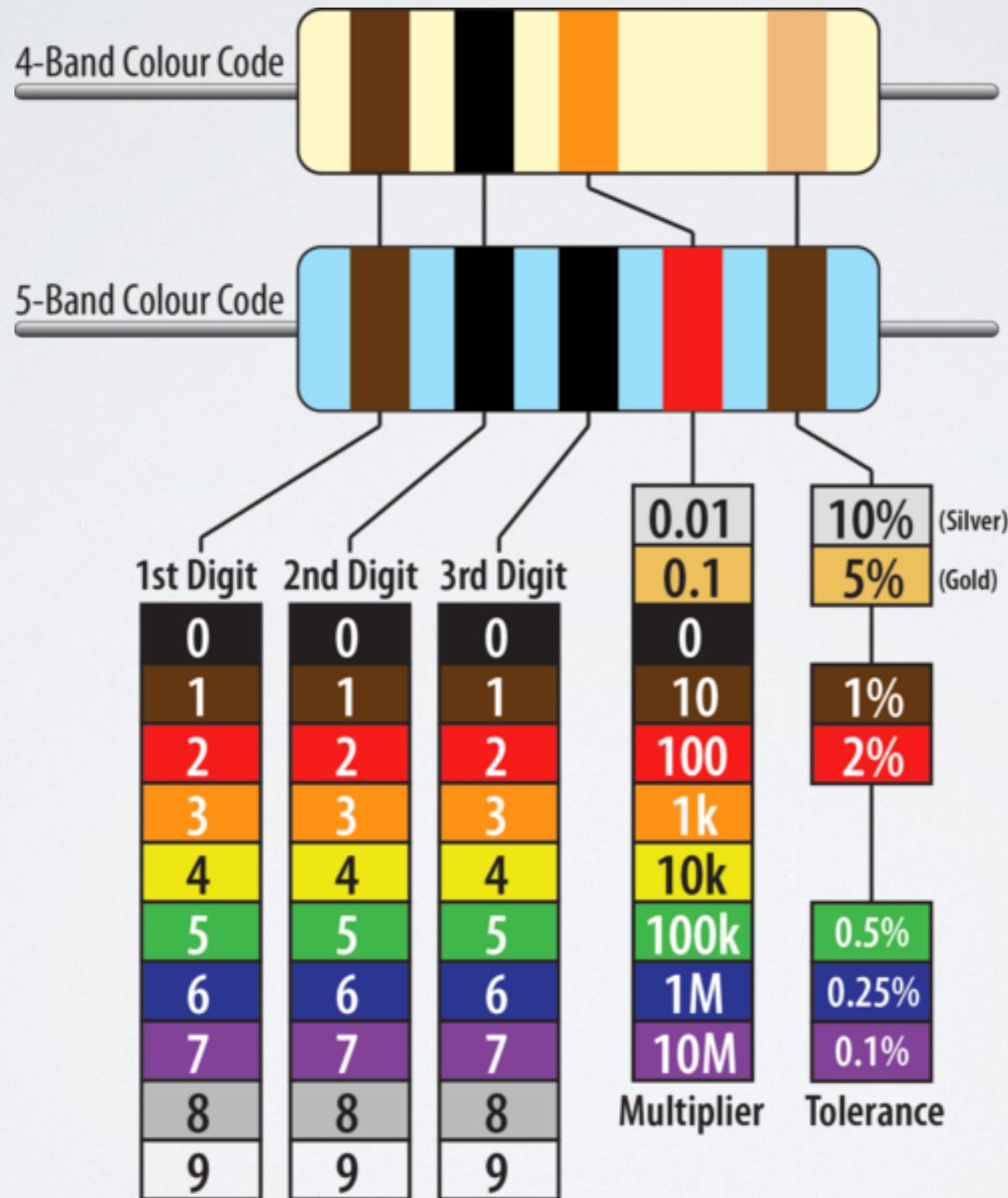
# Electronic Fundamentals

## How the solderless breadboard works:



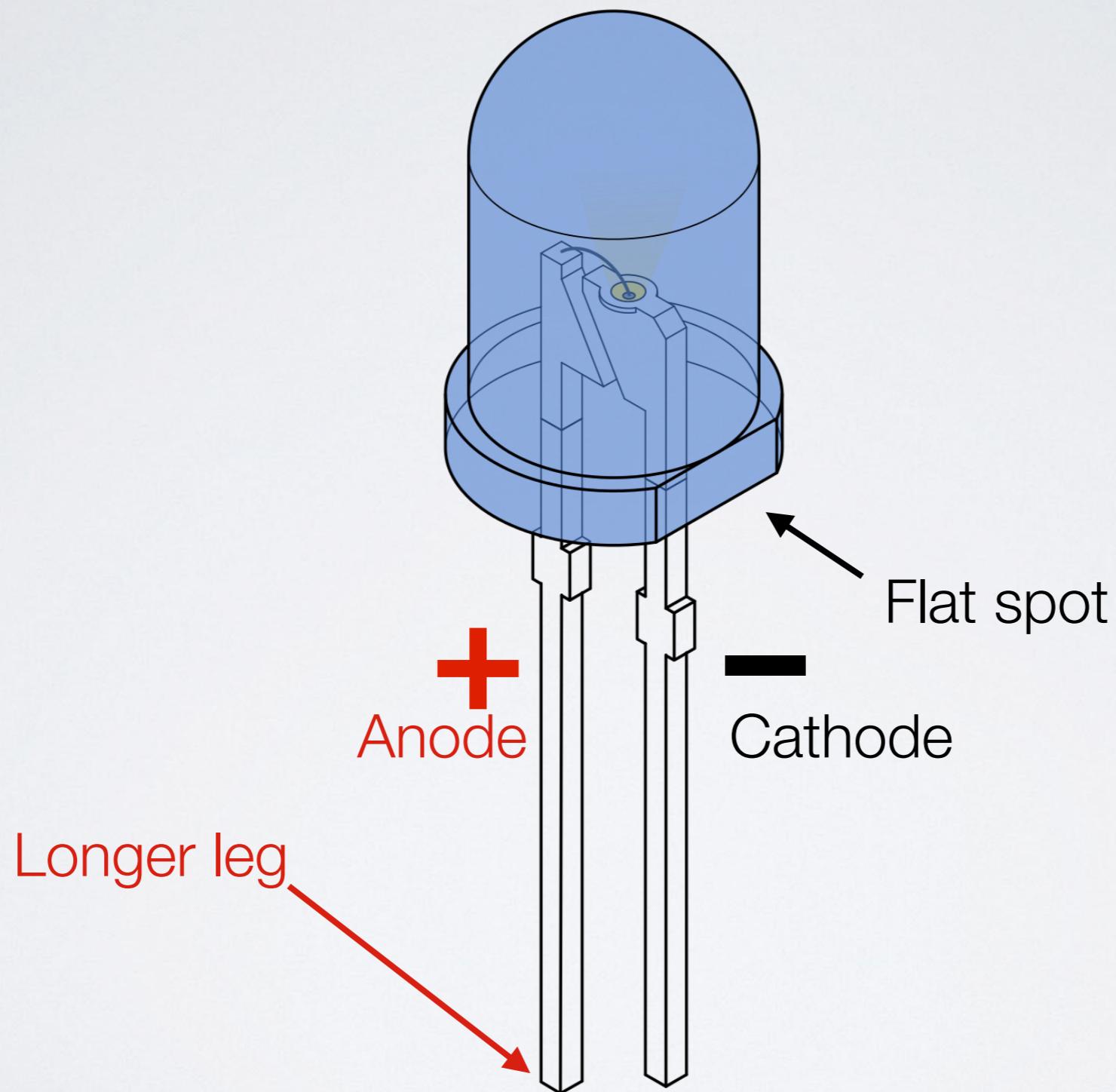
# Electronic Fundamentals

## Reading Resistor Codes:



# Electronic Fundamentals

## Reading LED polarity:



# Raspberry Pi Fundamentals

## Getting Started with Raspberry Pi:

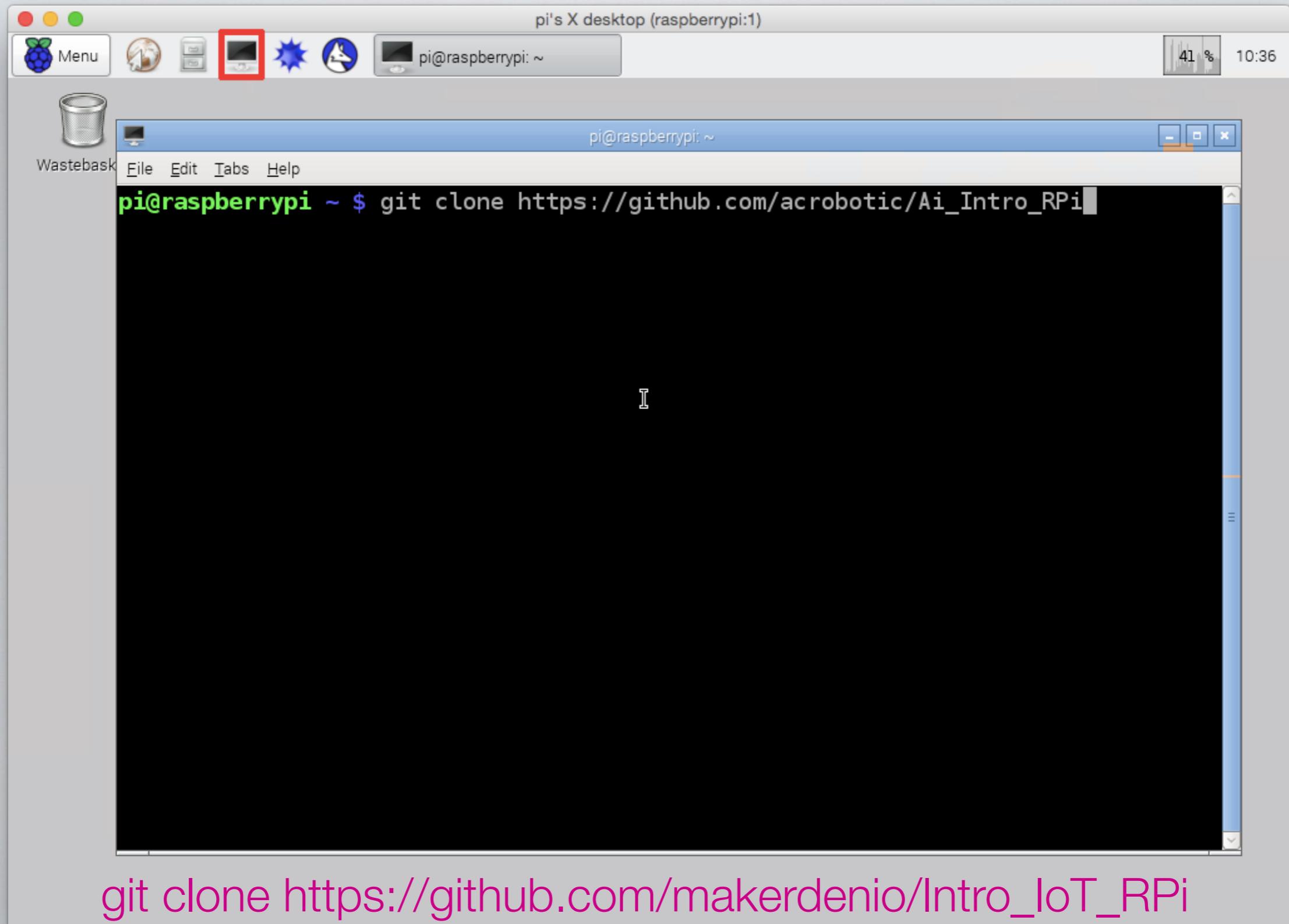


<http://learn.acrobotic.com>

<https://www.youtube.com/watch?v=ZJU7mns3juc>

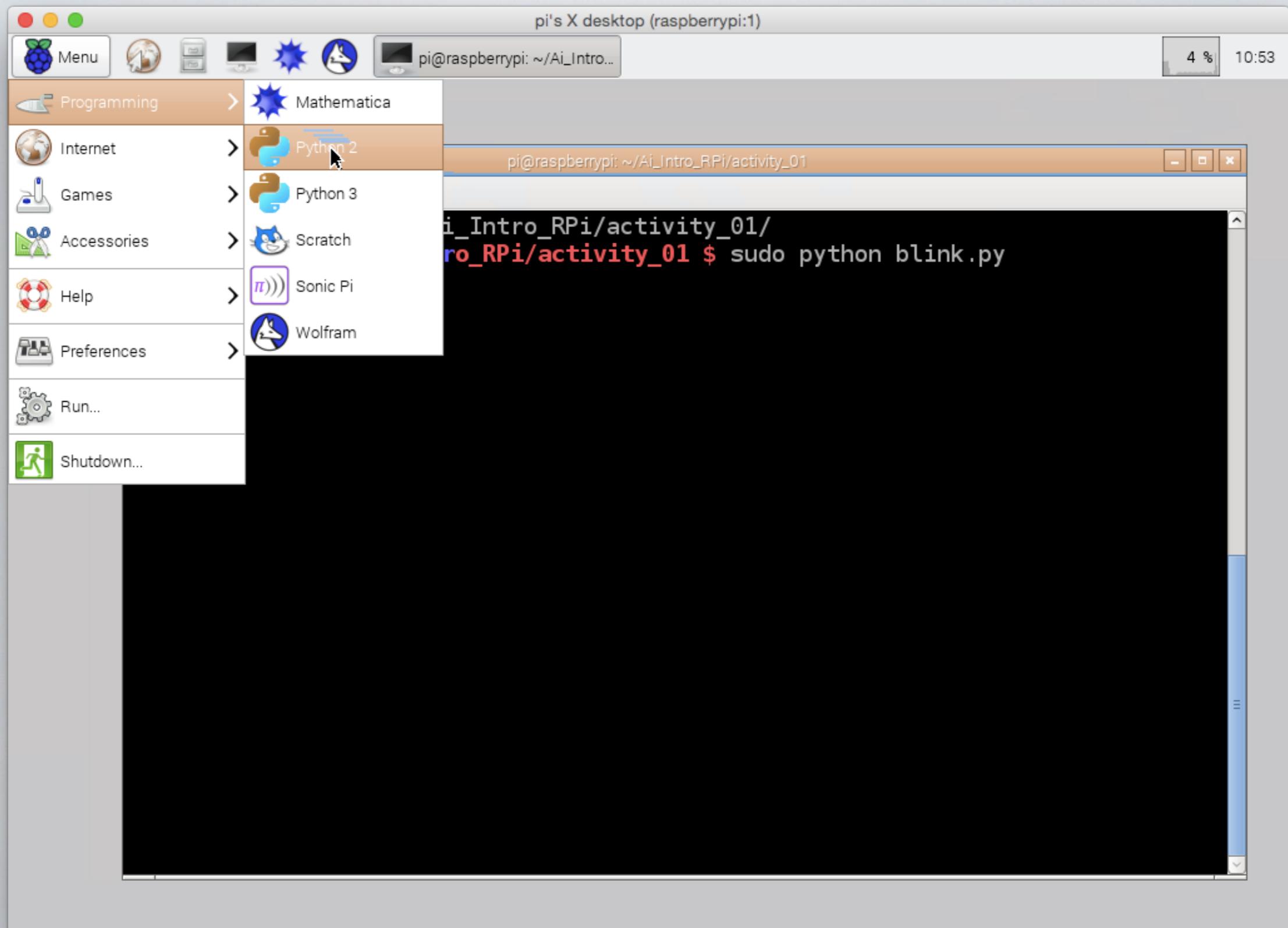
# Linux Fundamentals

## Accessing the Terminal and getting the activities code



# Linux Fundamentals

## Opening scripts in Python's “Integrated Development Environment”



# **Setting Up the Hardware**

Access to the GPIO with Python

Wiring an LED and motion sensor

Using Python to control a USB camera

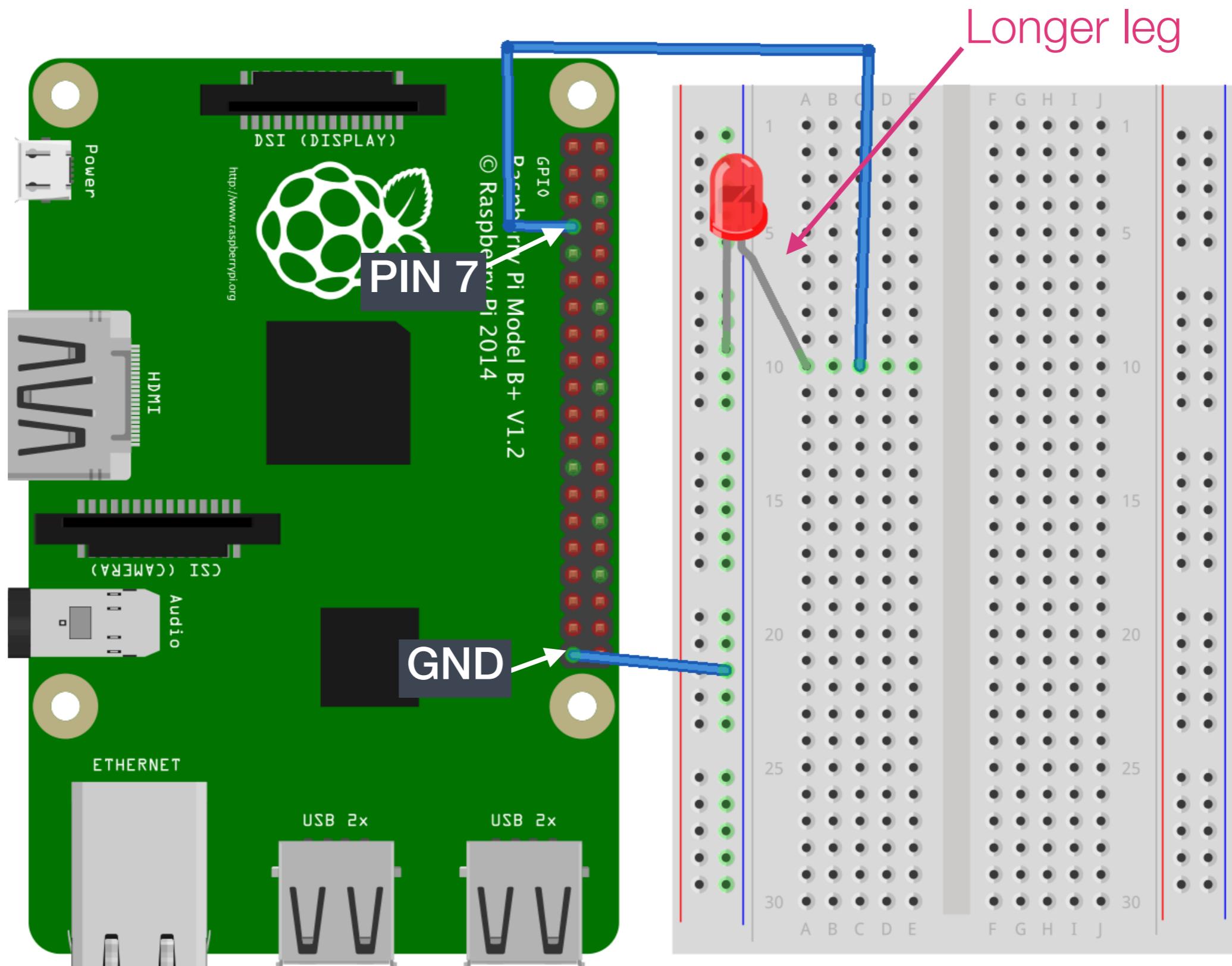
# Completing the Hardware Activities

## Instructions for completing activities 1 through 4

- Follow the diagram on the **Wiring** to connect components to the solderless breadboard
- Run the commands at the top of the **Programming** slide
- Compare the output you get with what's shown on the **Results** slide

# Setting Up the Hardware

## Activity 1: Blinking an LED [Wiring]

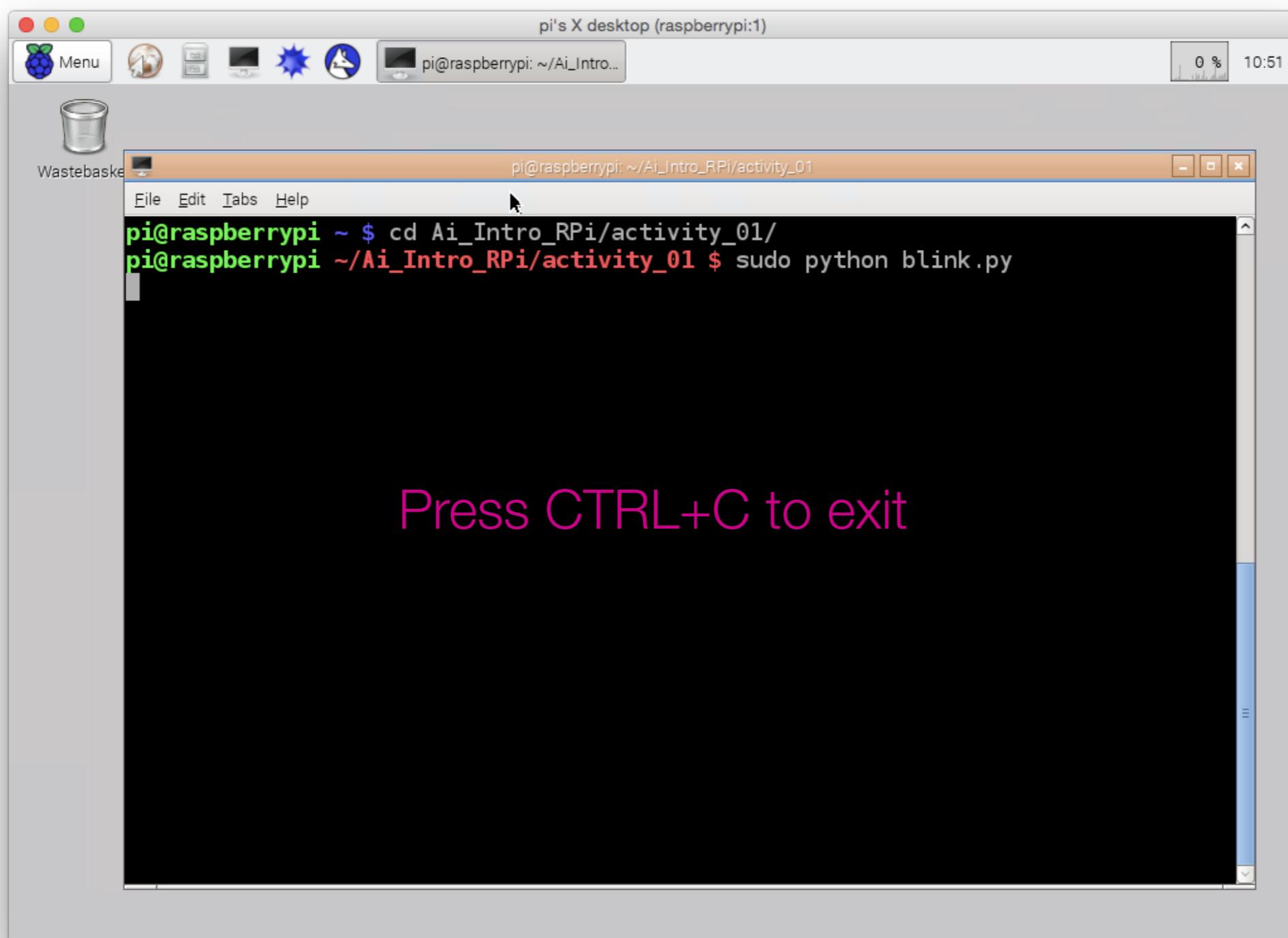


# Setting Up the Hardware

## Activity 1: Blinking an LED [Programming]

```
cd ~/Intro_IoT_RPi/activity_01
```

```
python blink.py
```

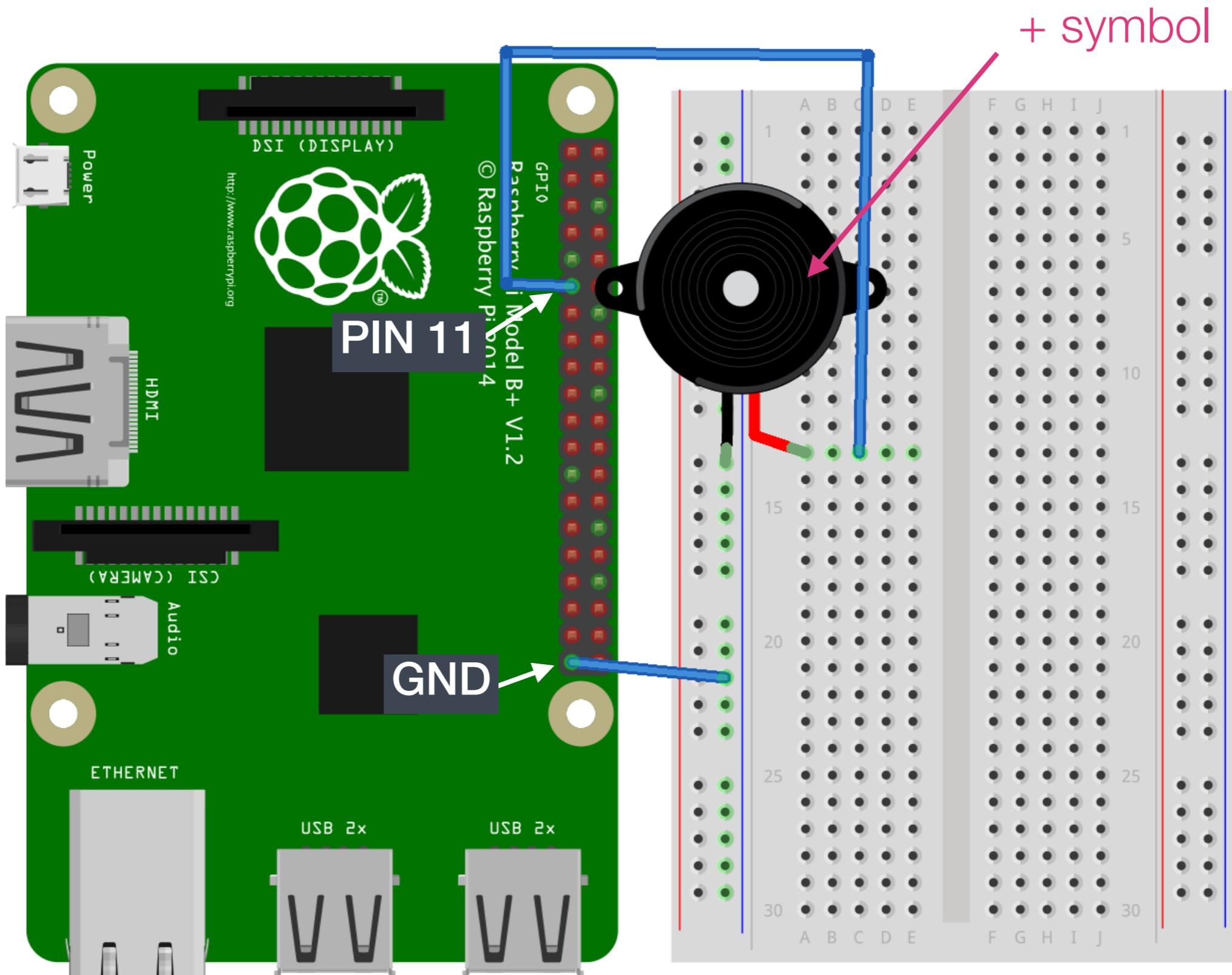


# Setting Up the Hardware

## **Activity 1: Blinking an LED [Results]**

# Setting Up the Hardware

## Activity 2: Generating sound with PWM [sound.py]

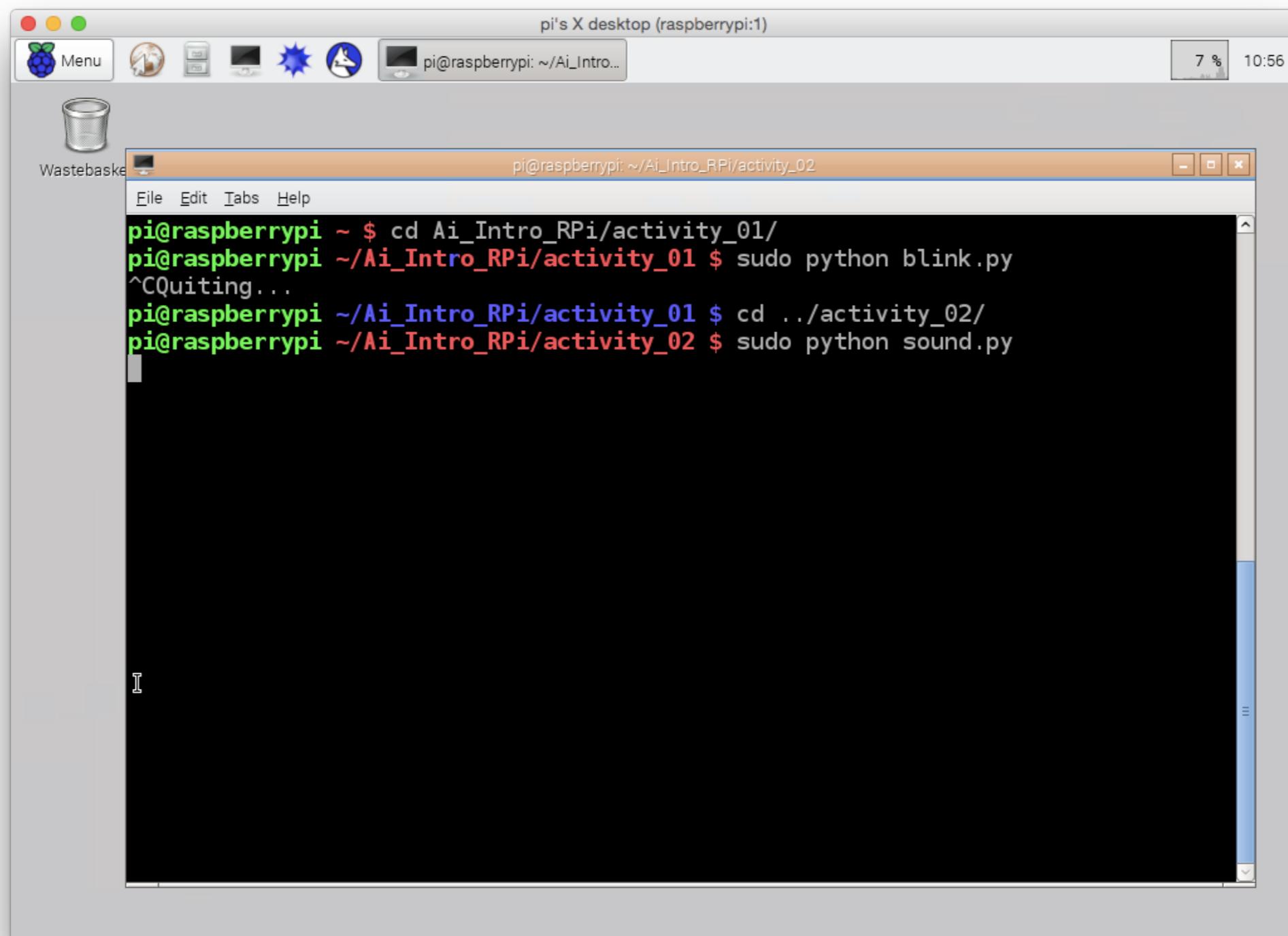


# Running the Code

## Activity 2: Generating sound with PWM [sound.py]

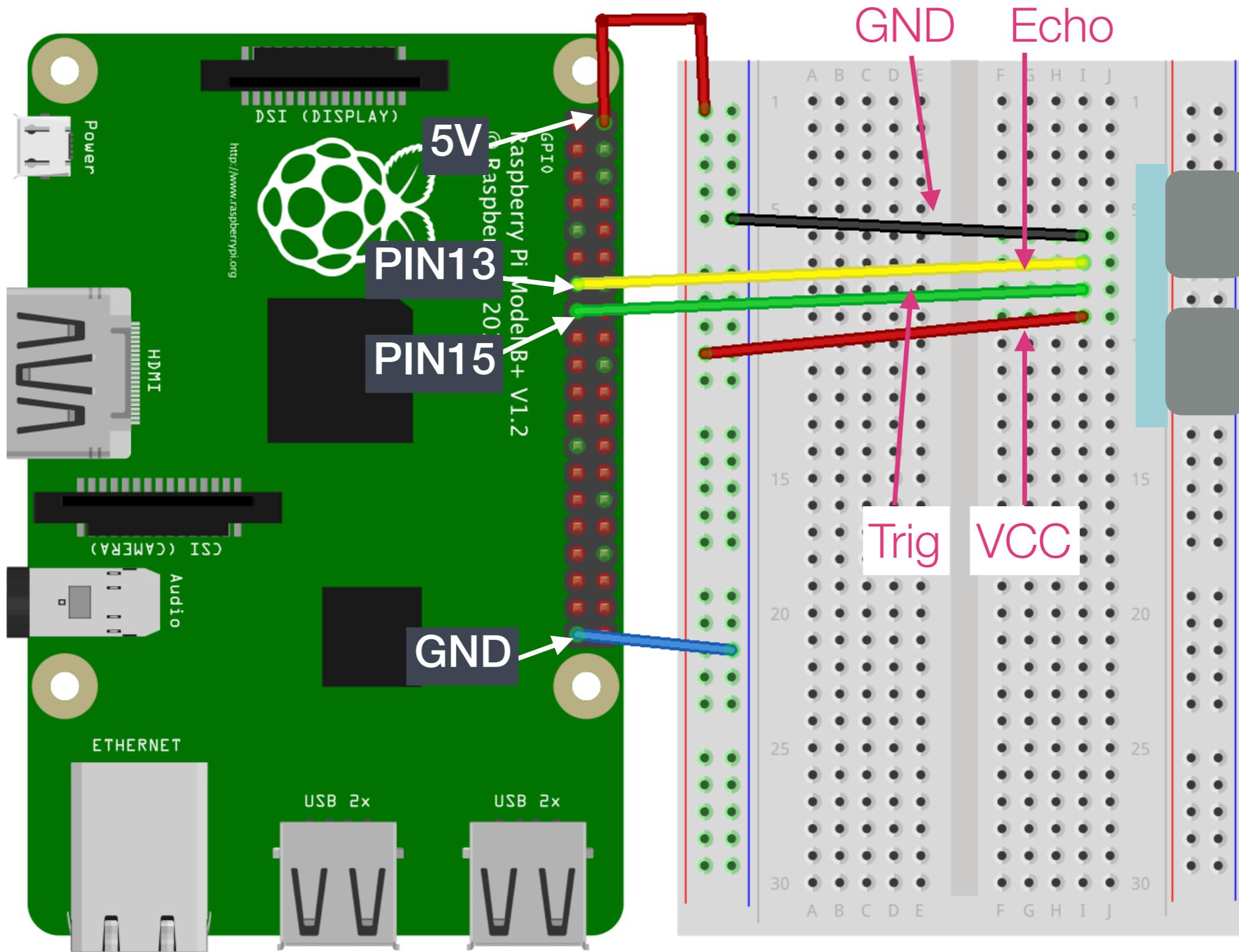
```
cd ../activity_02
```

```
python sound.py
```



# Setting Up the Hardware

## Activity 3a: Detecting Motion with an Ultrasonic sensor [ultrasonic.py]

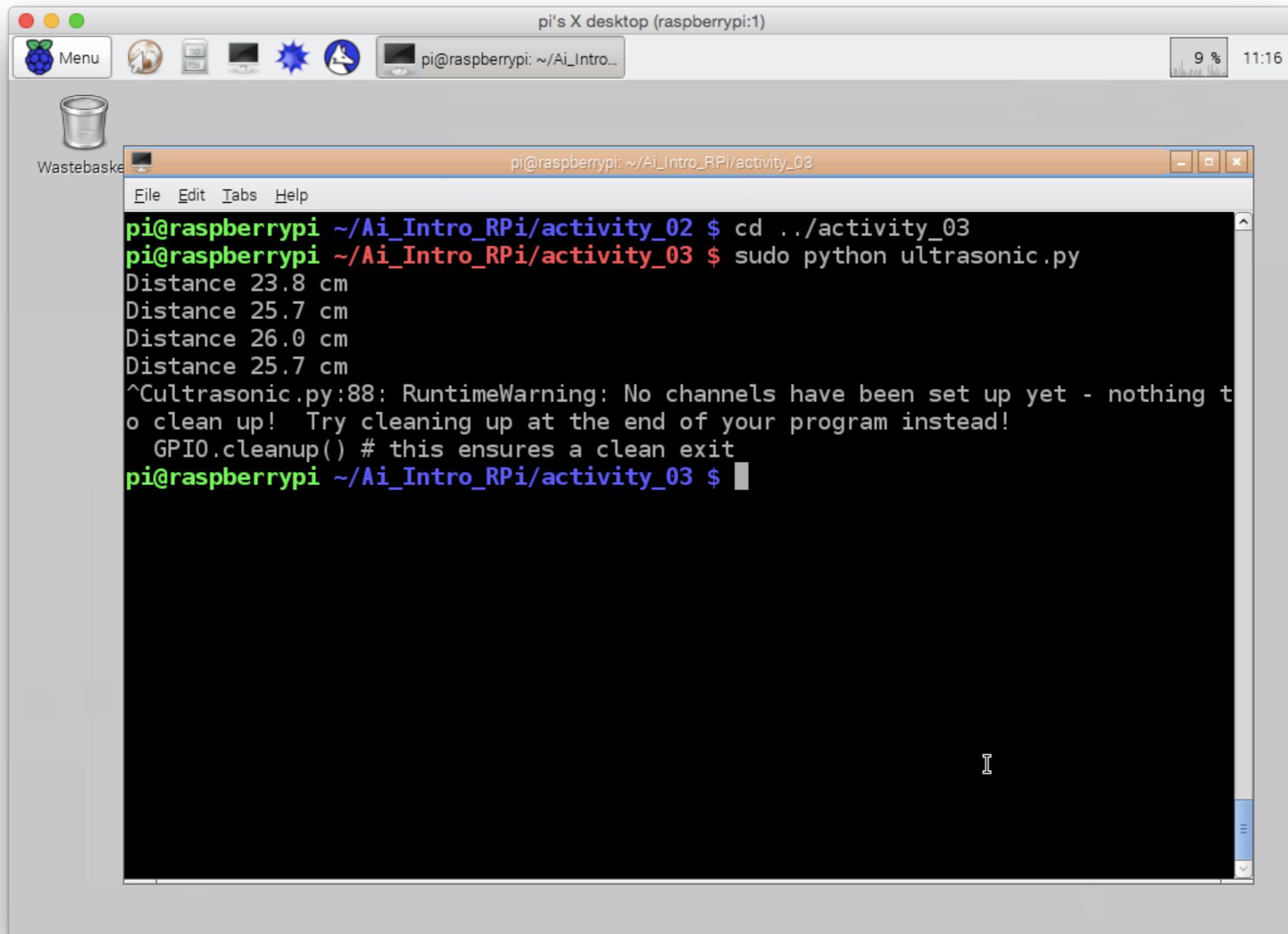


# Running the Code

## Activity 3a: Detecting Motion with an Ultrasonic sensor [ultrasonic.py]

```
cd ../activity_03a
```

```
python ultrasonic.py
```

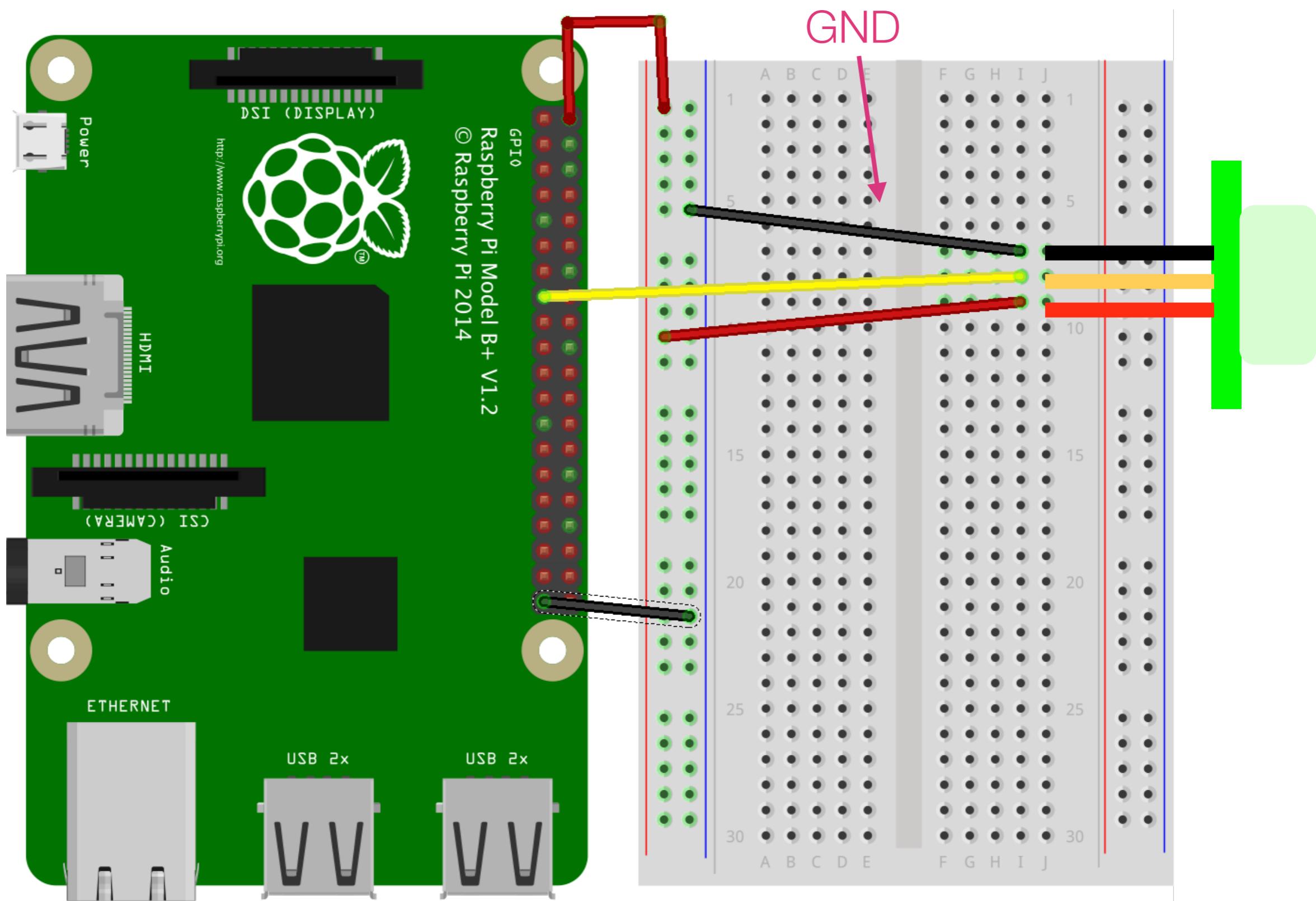


The screenshot shows a Raspberry Pi desktop environment with a terminal window open. The terminal window title is "pi@raspberrypi: ~/" and the command line shows the execution of the ultrasonic.py script. The output of the script is displayed in the terminal, showing distance measurements in centimeters.

```
pi@raspberrypi ~$ cd ../activity_03a
pi@raspberrypi ~$ sudo python ultrasonic.py
Distance 23.8 cm
Distance 25.7 cm
Distance 26.0 cm
Distance 25.7 cm
^Cultrasonic.py:88: RuntimeWarning: No channels have been set up yet - nothing to clean up! Try cleaning up at the end of your program instead!
    GPIO.cleanup() # this ensures a clean exit
pi@raspberrypi ~$
```

# Setting Up the Hardware

## Activity 3b: Detecting Motion with a Motion sensor [ultrasonic.py]

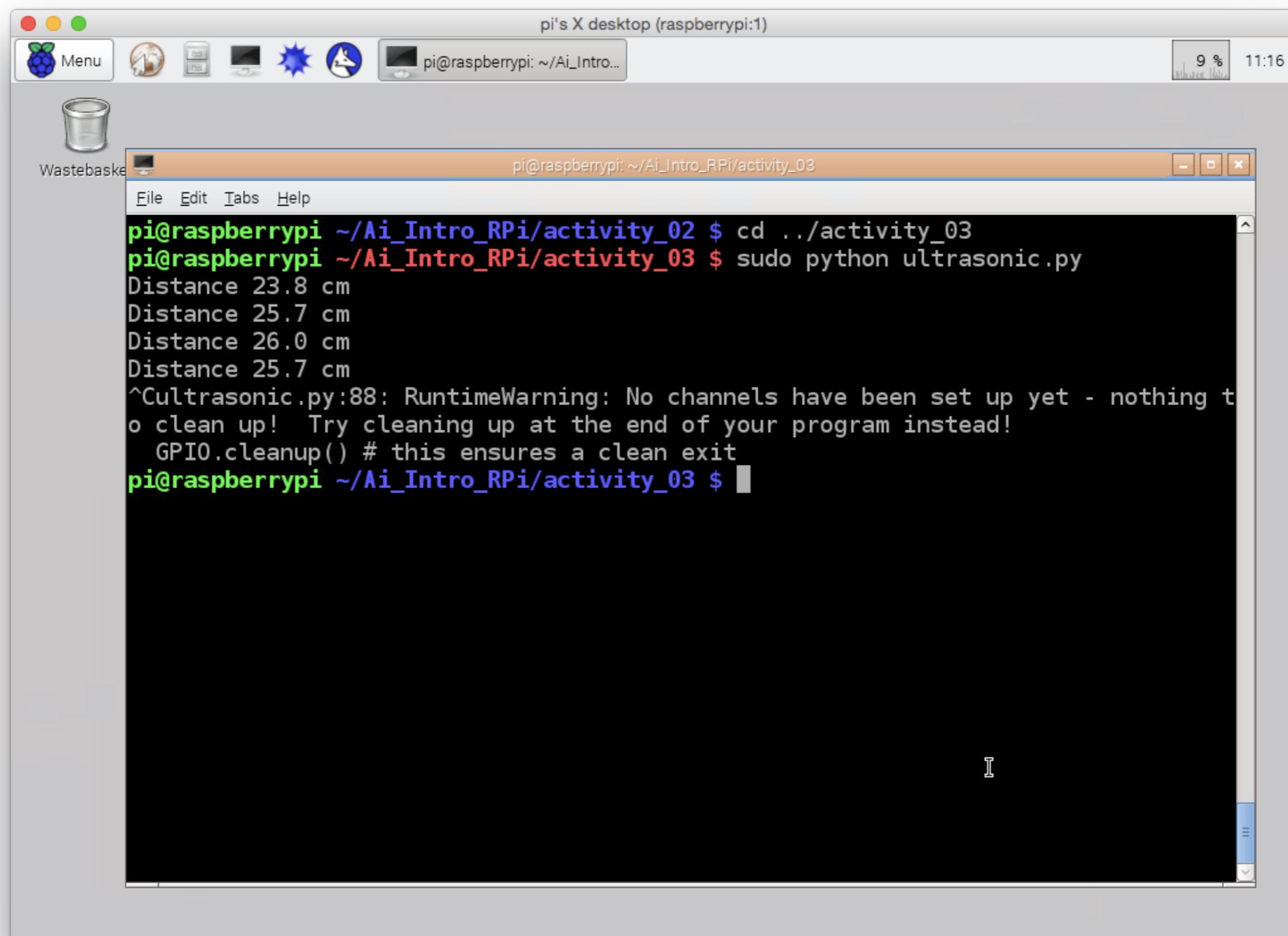


# Running the Code

## Activity 3b: Detecting Motion with an IR Motion sensor [motion.py]

```
cd ../activity_03b
```

```
python motion.py
```



# Setting Up the Hardware

## Activity 4: Capturing an Image from a USB camera [usb\_camera.py]

```
sudo apt-get update
```

```
sudo apt-get install fswebcam python-flask
```

The screenshot shows a desktop environment on a Raspberry Pi. The title bar of the terminal window reads "pi's X desktop (raspberrypi:1)". The terminal window contains the following command history:

```
pi@raspberrypi:~/Ai_Intro_RPi/activity_04 $ cd .. /activity_04
pi@raspberrypi:~/Ai_Intro_RPi/activity_04 $ sudo python usb_camera.py
pi@raspberrypi:~/Ai_Intro_RPi/activity_04 $
```

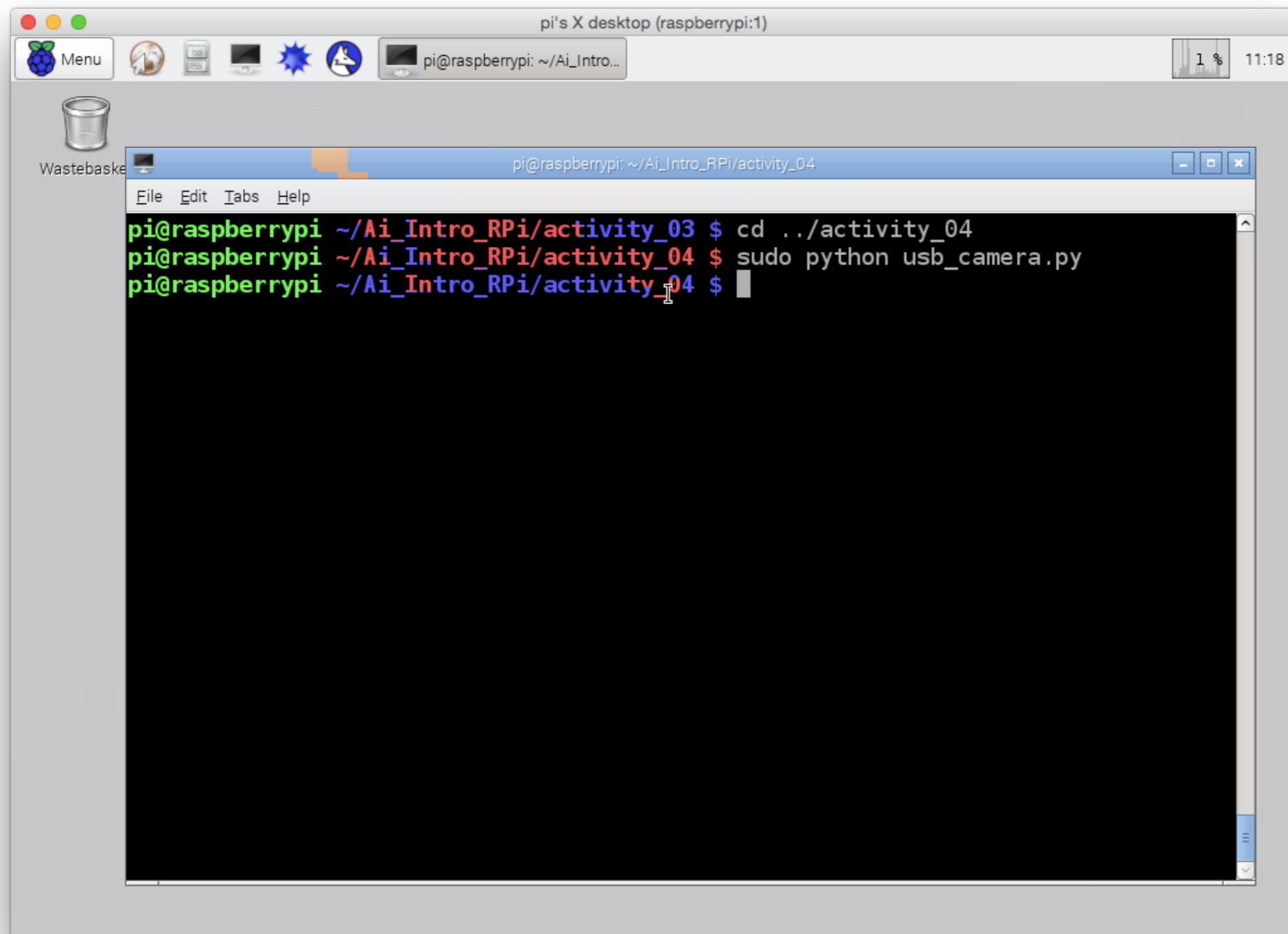
Below the terminal window, there is a large black area with the text "# Plug in usb camera via USB :)" in white. At the bottom of the screen, there is another line of text in magenta: "fswebcam -r 640x480 -d /dev/video0 -q sample\_pic.jpg".

# Running the Code

## Activity 4: Capturing an Image from a USB camera [usb\_camera.py]

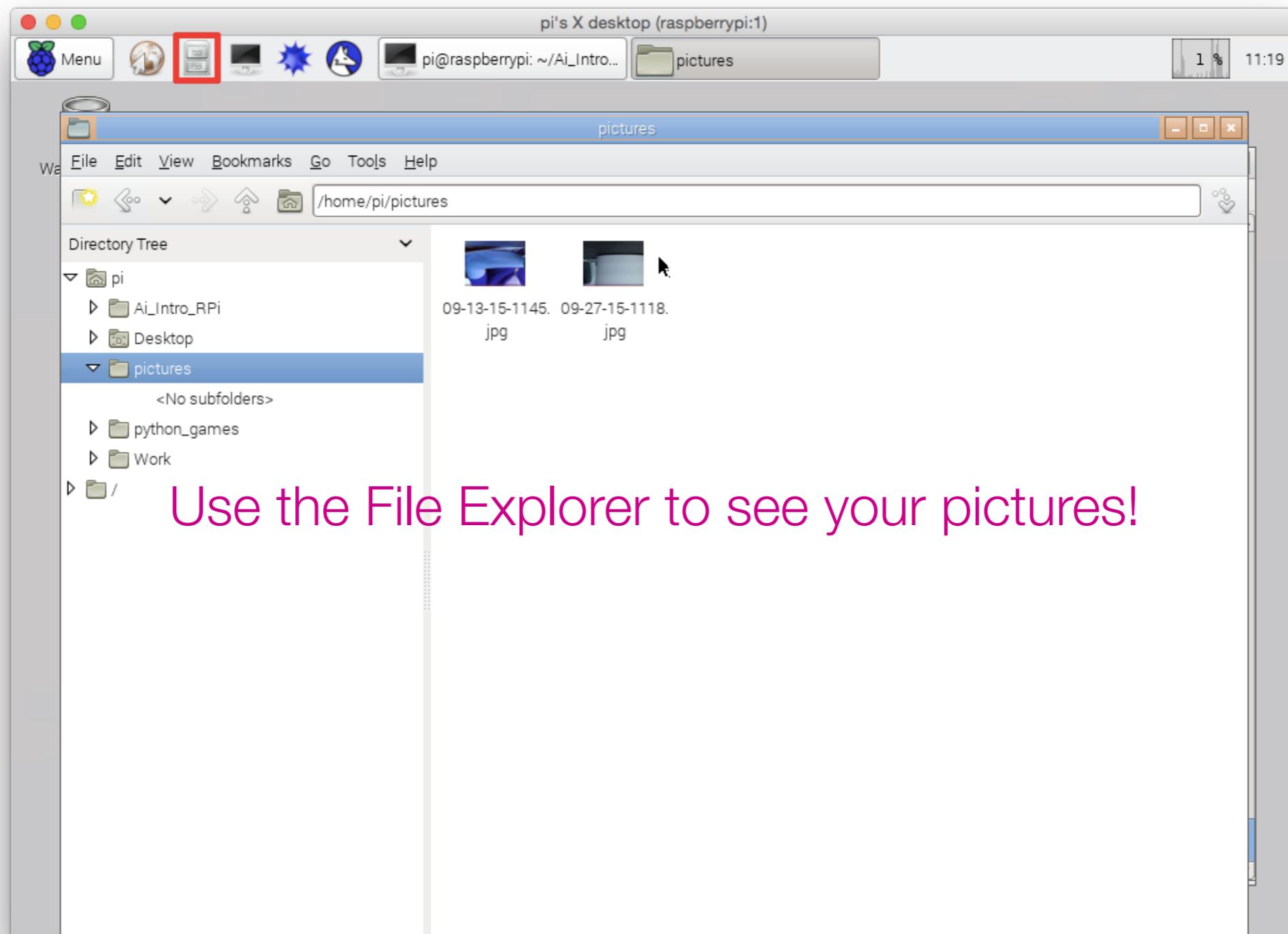
```
cd ../activity_04
```

```
sudo python usb_camera.py
```



# Running the Code

## Activity 4: Capturing an Image from a USB camera [usb\_camera.py]



# **Configuring the Backend**

Raspbian and system administration basics

Running a webserver using Python and Flask

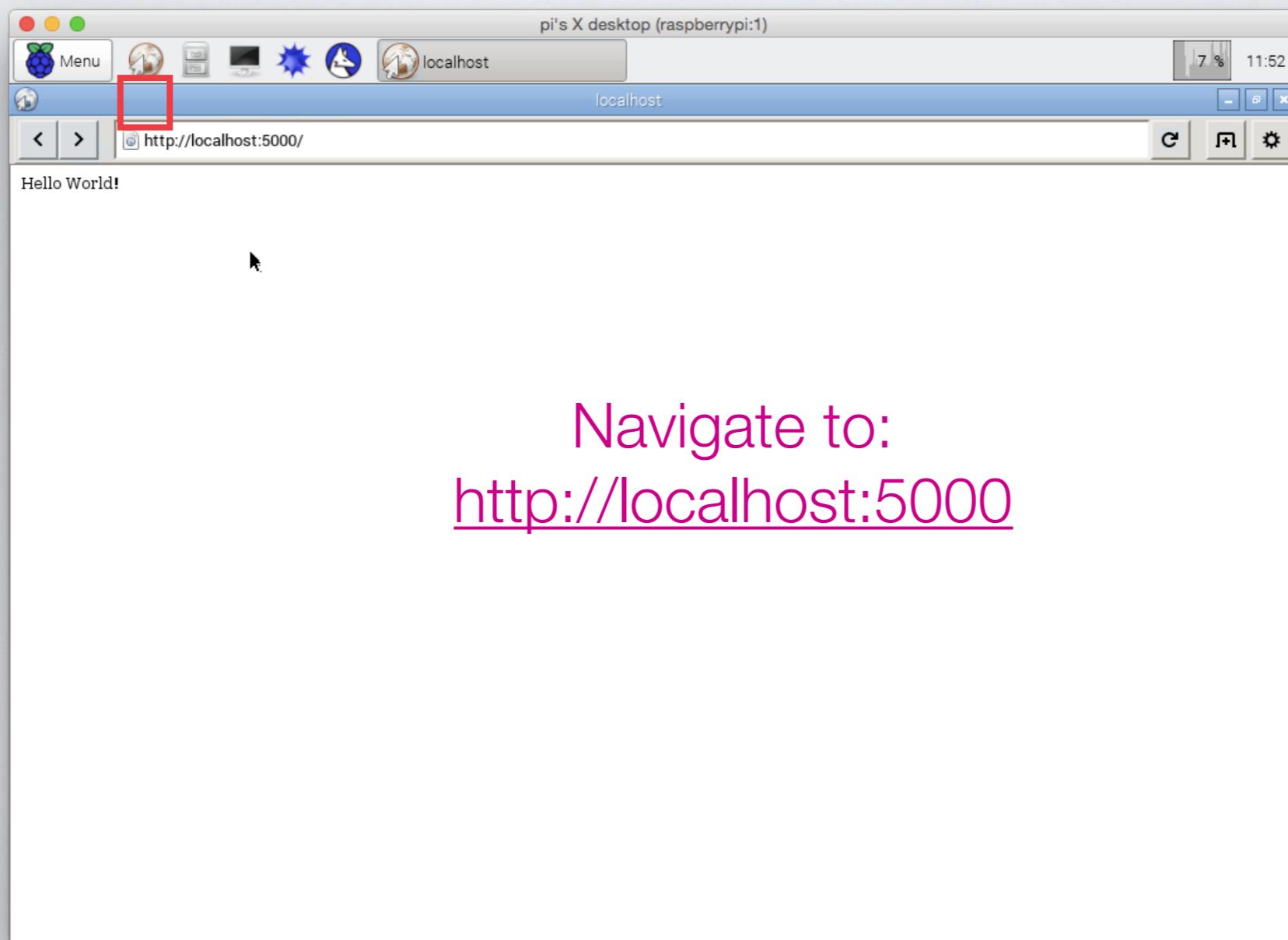
Building an API with Flask for LED control

# Configuring the Backend

## Activity 5: Getting started with web frameworks [application.py]

```
cd ../activity_05/simple
```

```
python application.py
```



# Configuring the Backend

**Activity 5: Getting started with web frameworks [application.py]**

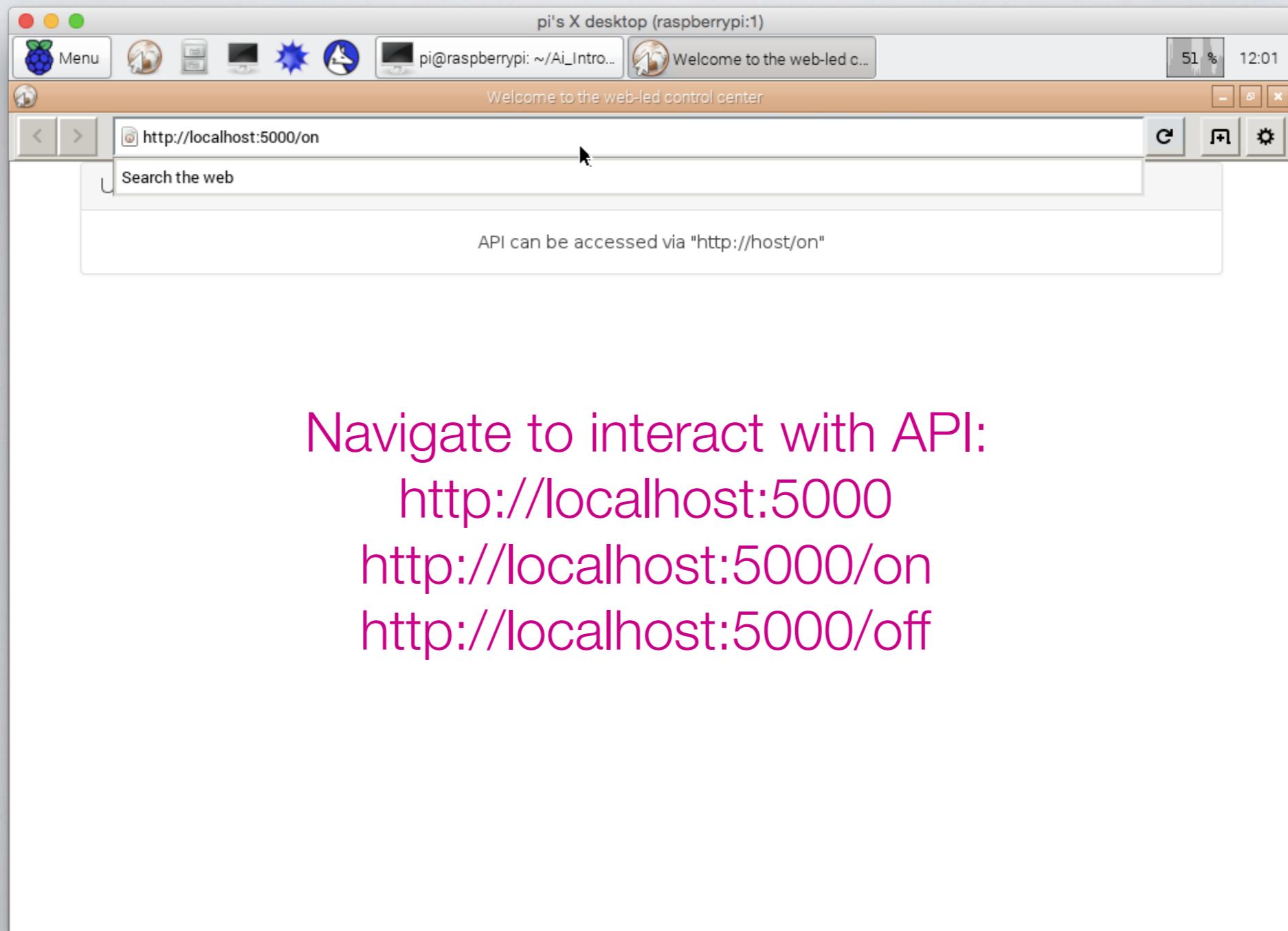
Q&A

# Configuring the Backend

## Activity 6: Building a simple API for LED control [application.py]

```
cd ../../activity_06
```

```
python application.py
```



Navigate to interact with API:

<http://localhost:5000>

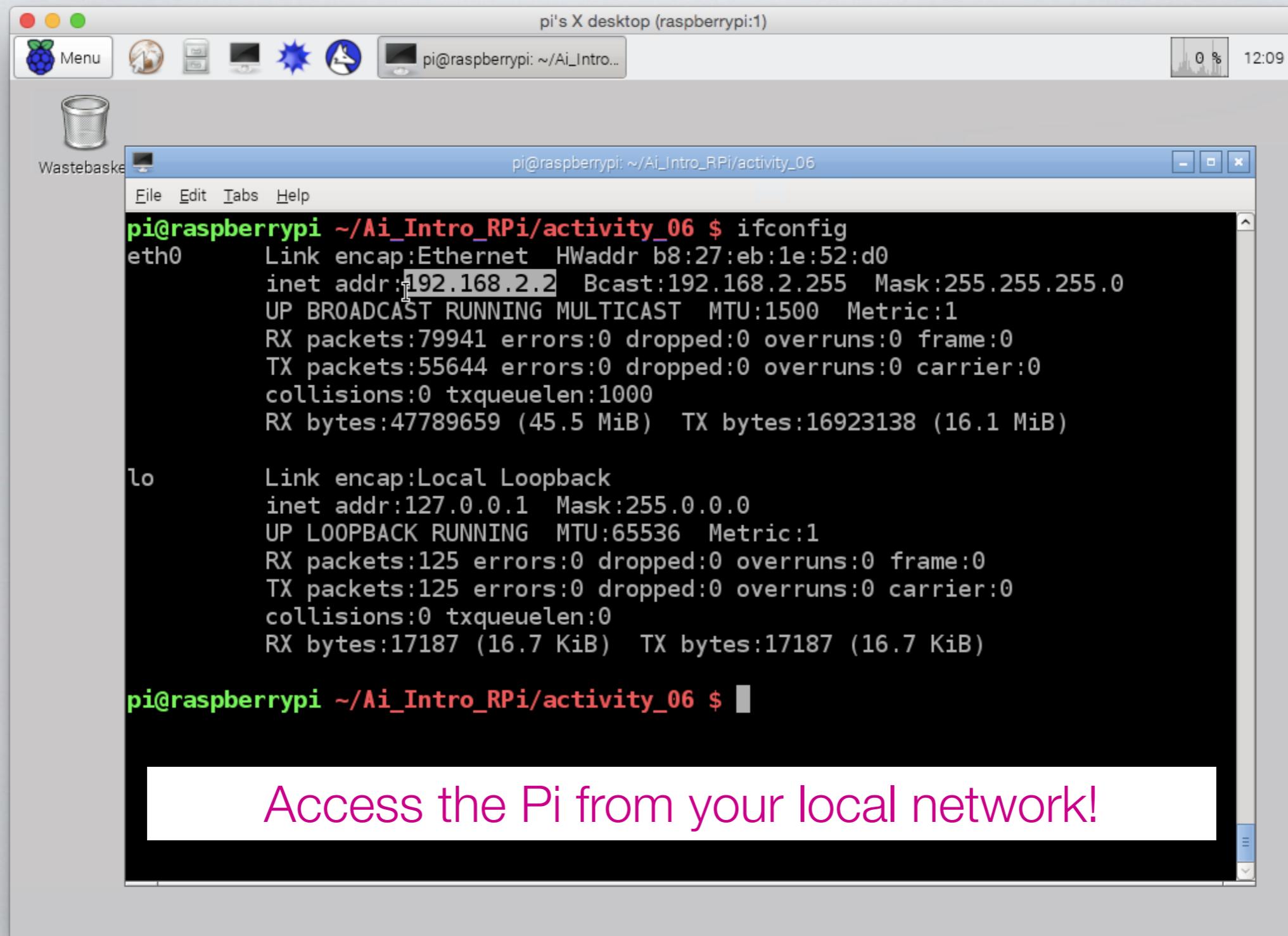
<http://localhost:5000/on>

<http://localhost:5000/off>

# Configuring the Backend

## Activity 6: Building a simple API for LED control [application.py]

`ifconfig`



The screenshot shows a Xfce desktop environment on a Raspberry Pi. A terminal window is open, displaying the output of the `ifconfig` command. The terminal window title is `pi@raspberrypi: ~/Ai_Intro_RPi/activity_06`. The output shows two network interfaces: `eth0` and `lo`. The `eth0` interface has an IP address of `192.168.2.2` and is connected to a local network. The `lo` interface is a loopback interface. At the bottom of the terminal window, there is a pink callout box with the text `Access the Pi from your local network!`.

```
pi@raspberrypi ~/Ai_Intro_RPi/activity_06 $ ifconfig
eth0      Link encap:Ethernet HWaddr b8:27:eb:1e:52:d0
          inet addr:192.168.2.2  Bcast:192.168.2.255  Mask:255.255.255.0
                  UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
                  RX packets:79941 errors:0 dropped:0 overruns:0 frame:0
                  TX packets:55644 errors:0 dropped:0 overruns:0 carrier:0
                  collisions:0 txqueuelen:1000
                  RX bytes:47789659 (45.5 MiB)  TX bytes:16923138 (16.1 MiB)

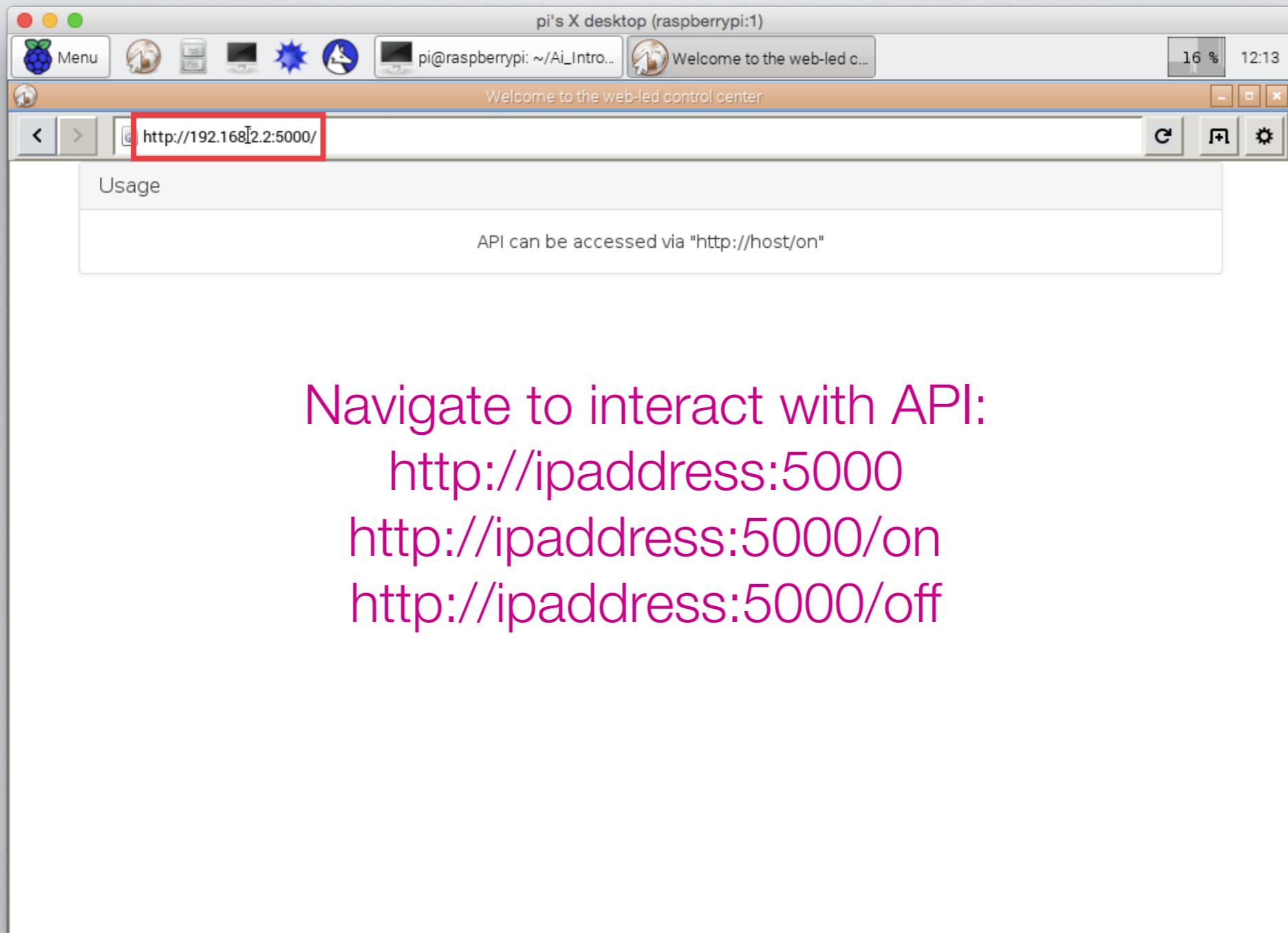
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
                  UP LOOPBACK RUNNING  MTU:65536  Metric:1
                  RX packets:125 errors:0 dropped:0 overruns:0 frame:0
                  TX packets:125 errors:0 dropped:0 overruns:0 carrier:0
                  collisions:0 txqueuelen:0
                  RX bytes:17187 (16.7 KiB)  TX bytes:17187 (16.7 KiB)

pi@raspberrypi ~/Ai_Intro_RPi/activity_06 $
```

Access the Pi from your local network!

# Configuring the Backend

## Activity 6: Building a simple API for LED control [application.py]



Navigate to interact with API:

`http://ipaddress:5000`

`http://ipaddress:5000/on`

`http://ipaddress:5000/off`

# Configuring the Backend

## **Activity 6: Building a simple API for LED control [application.py]**

```
cd ../../activity_06  
python application.py
```

# Q&A

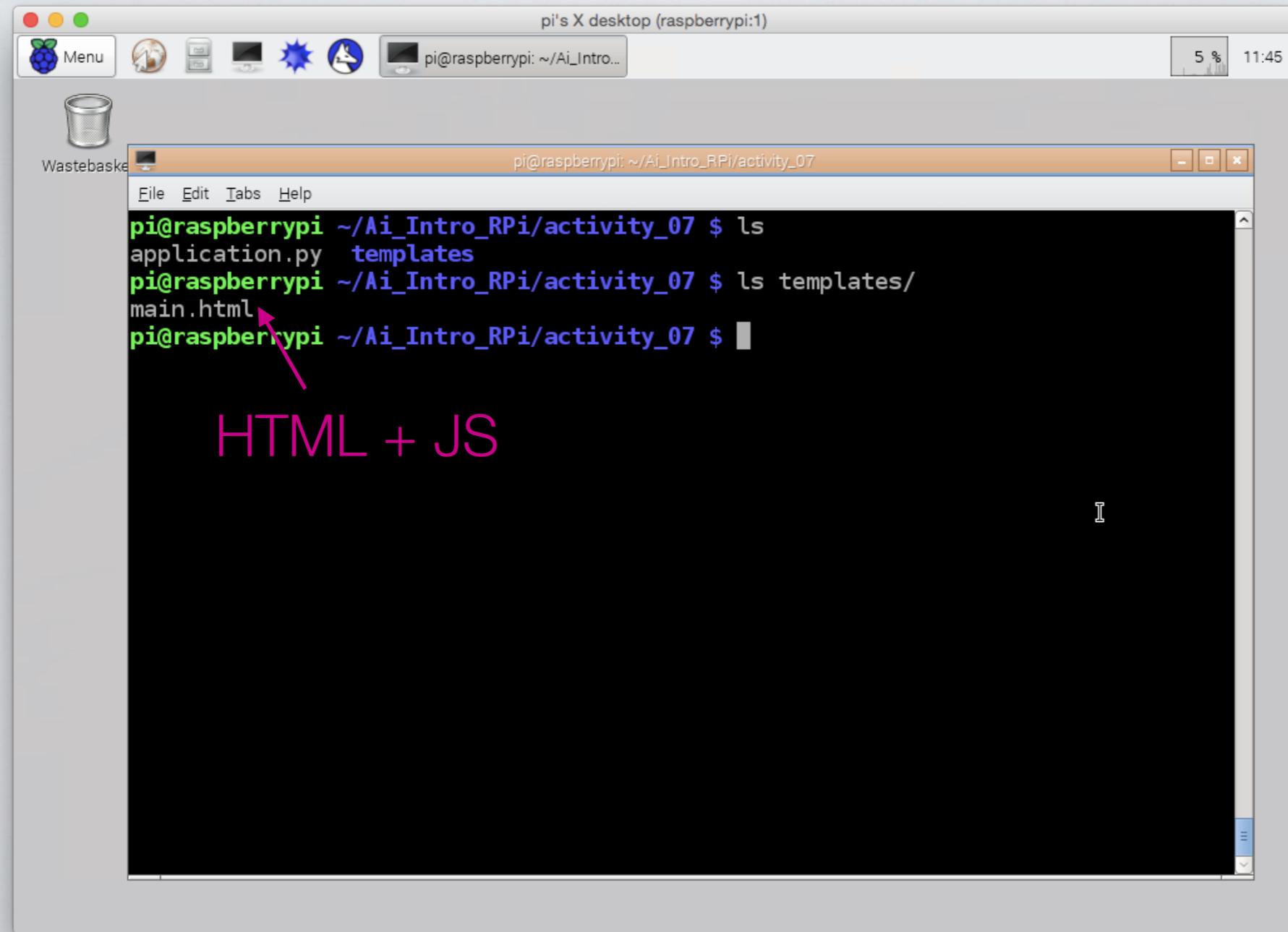
# **Building a User Interface (Frontend)**

Building a User Interface (UI) with HTML, CSS, and JavaScript

# Building a User Interface

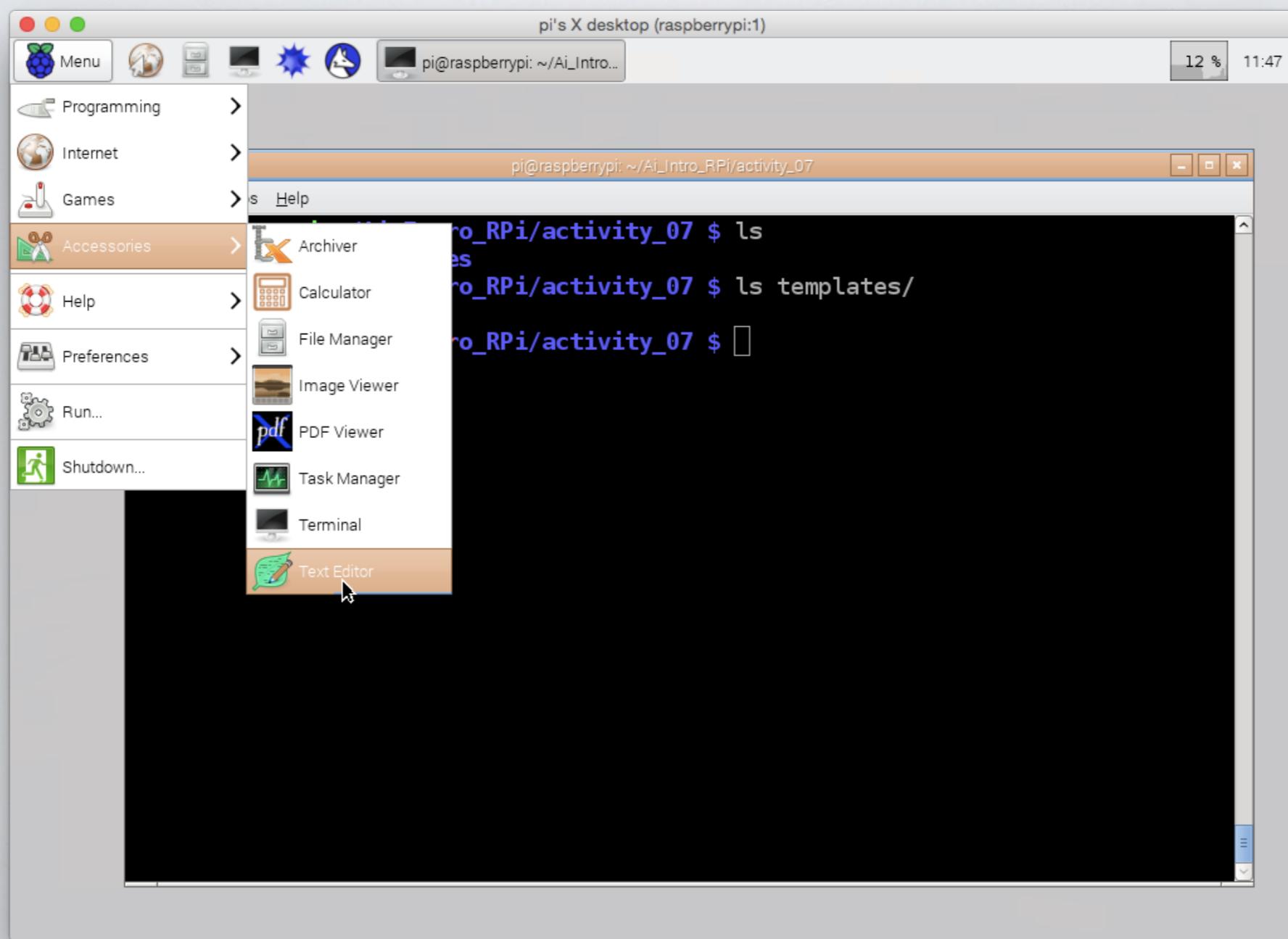
## Activity 7: Creating a simple UI for LED control [application.py]

```
cd ../activity_07
```



# Building a User Interface

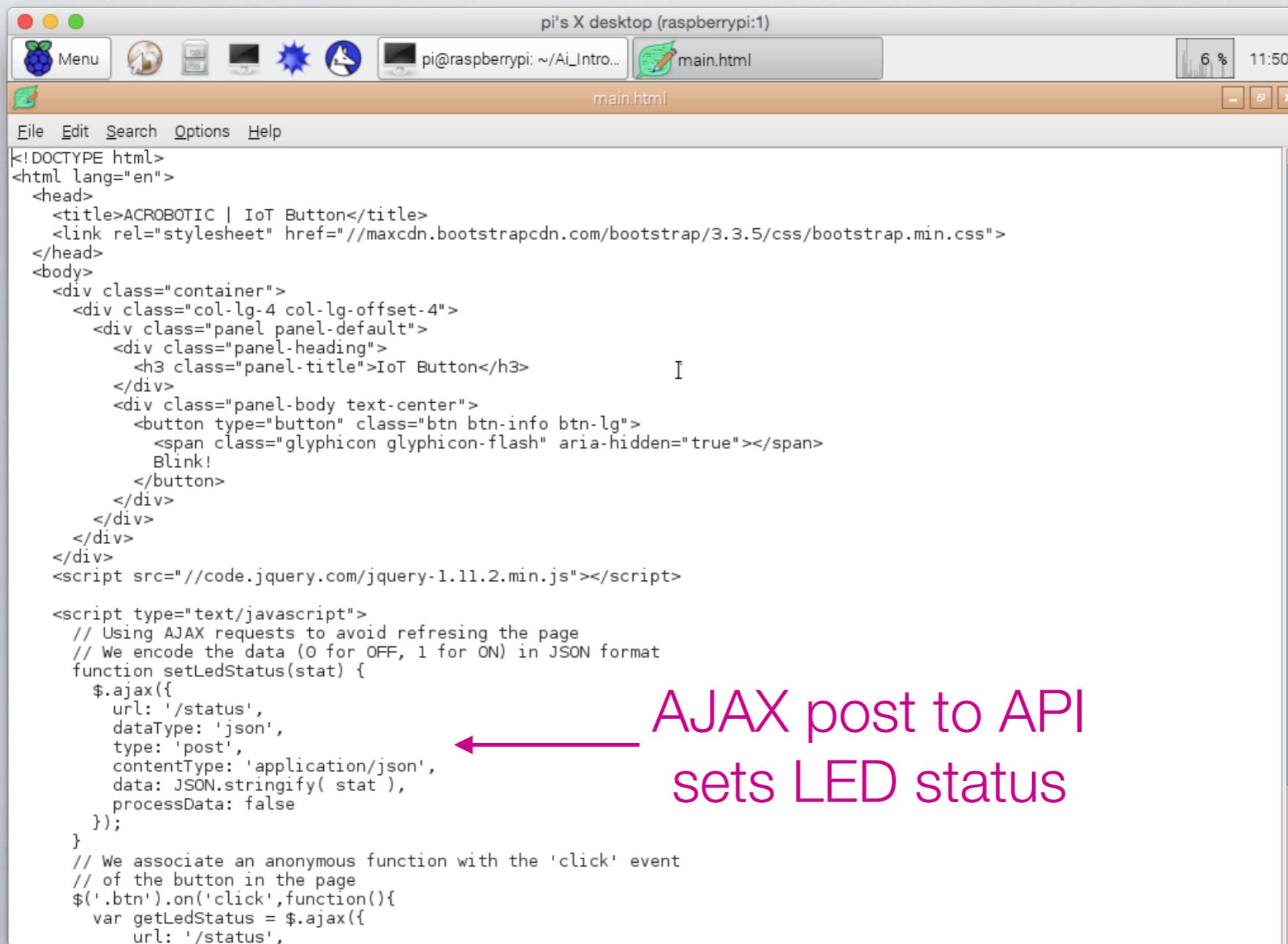
## Activity 7: Creating a simple UI for LED control [application.py]



# Building a User Interface

## Activity 7: Creating a simple UI for LED control [application.py]

### Q&A



```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>ACROBOTIC | IoT Button</title>
    <link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css">
</head>
<body>
    <div class="container">
        <div class="col-lg-4 col-lg-offset-4">
            <div class="panel panel-default">
                <div class="panel-heading">
                    <h3 class="panel-title">IoT Button</h3>
                </div>
                <div class="panel-body text-center">
                    <button type="button" class="btn btn-info btn-lg">
                        <span class="glyphicon glyphicon-flash" aria-hidden="true"></span>
                        Blink!
                    </button>
                </div>
            </div>
        </div>
    </div>
    <script src="//code.jquery.com/jquery-1.11.2.min.js"></script>

    <script type="text/javascript">
        // Using AJAX requests to avoid refresing the page
        // We encode the data (0 for OFF, 1 for ON) in JSON format
        function setLedStatus(stat) {
            $.ajax({
                url: '/status',
                dataType: 'json',
                type: 'post',
                contentType: 'application/json',
                data: JSON.stringify( stat ),
                processData: false
            });
        }
        // We associate an anonymous function with the 'click' event
        // of the button in the page
        $('.btn').on('click',function(){
            var getLedStatus = $.ajax({
                url: '/status',
                type: 'get'
            });
        });
    </script>

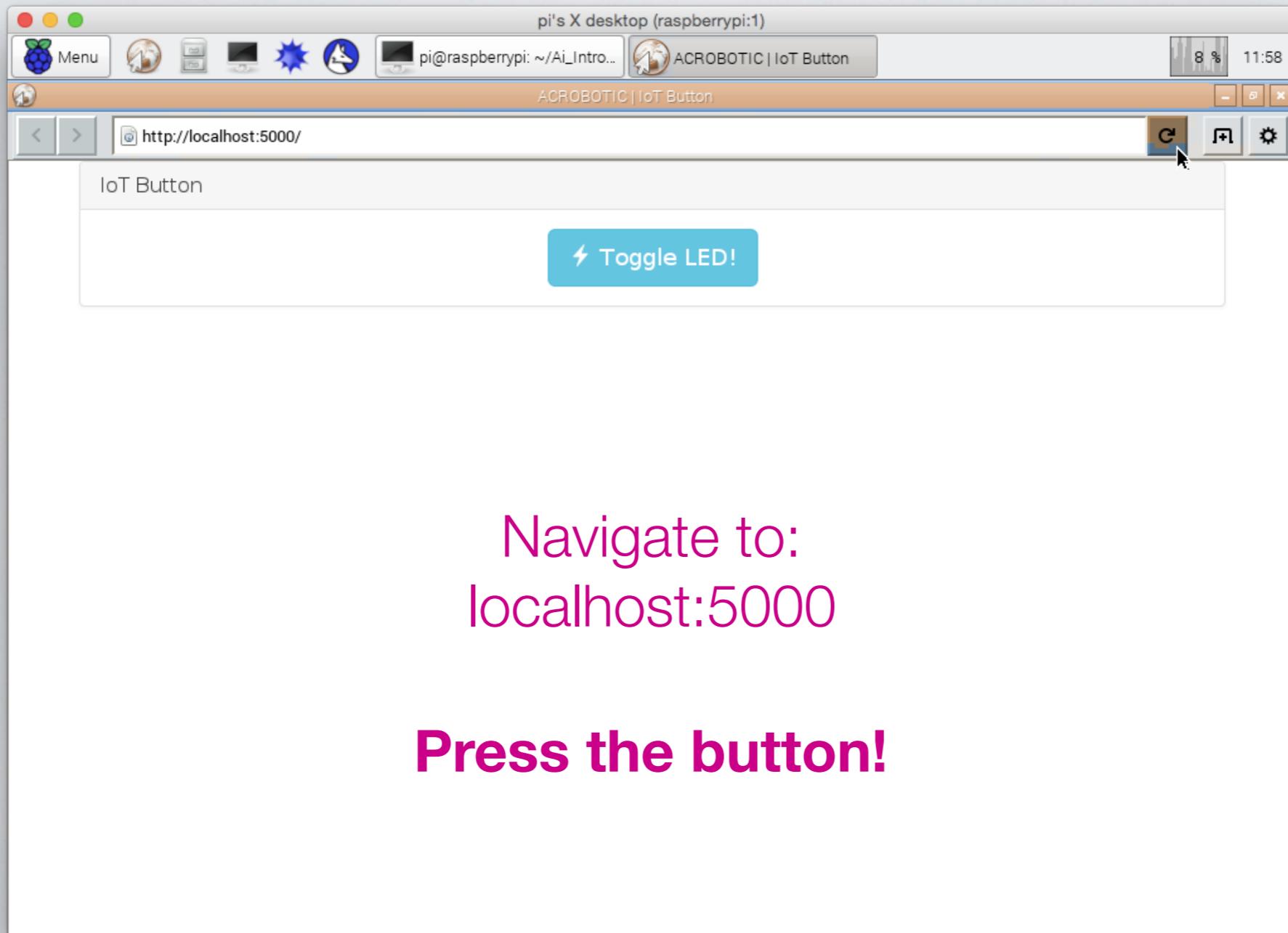
```

AJAX post to API  
sets LED status

# Building a User Interface

## Activity 7: Creating a simple UI for LED control [application.py]

```
sudo python application.py
```

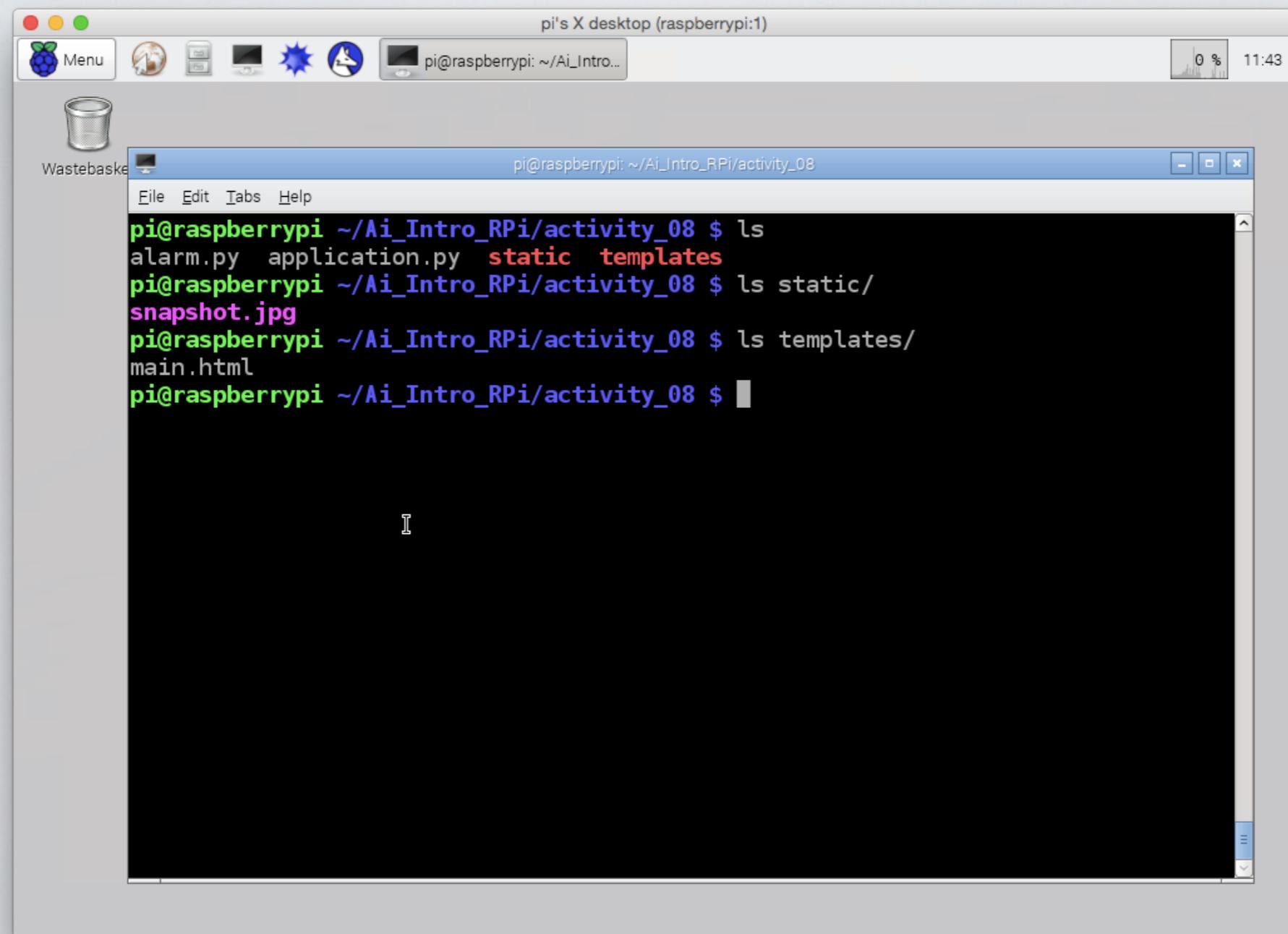


# Building a User Interface

## Activity 8: Creating a simple UI for an Alarm System [application.py]

```
cd ../activity_08
```

```
sudo python application.py
```

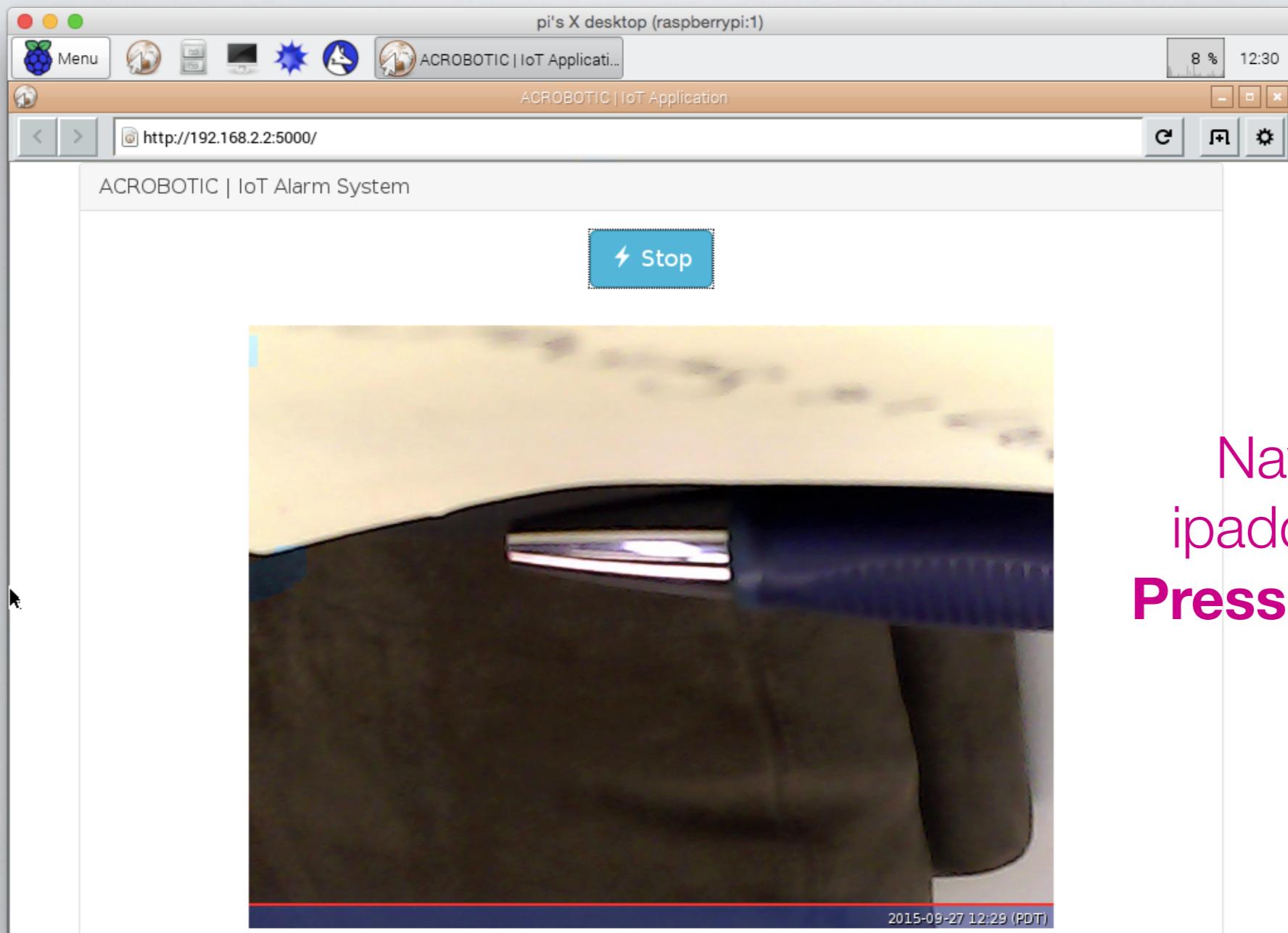


# Building a User Interface

## Activity 8: Creating a simple UI for an Alarm System [application.py]

```
cd ../activity_08
```

```
python application.py
```



Navigate to:  
ipaddress:5000  
**Press the button!**