PRACTICAL DEV COMPANION

A charrold303 companion guide

Chris Harrold:
@charrold303
www.charrold303.com
bit.ly/charrold303

TABLE OF CONTENTS

WELCOME!

Thanks for attending one of my workshops, An Introduction to Practical Development! I am very excited that you have chosen to invest your time with me today, and in the spirit of respecting that investment, I have prepared this guide to make sure you have as much information as possible at your fingertips. It is broken into three sections:

1. *The glossary of terms lists common terms I will use during the presentation that you should know*
2. *The Useful Things section has some materials to help you during the course of the workshop that we will be using – charts, diagrams, etc.*
3. *Important Links has links to everything we will be using in a shortened format, so you can type them in from a printed copy more easily (they will work automatically from the digital copy of course)*

Thank you again for joining me today, and for investing in this session. I have worked hard to make sure you walk away with some new knowledge and skills that you can use to get started with your own projects!

GLOSSARY

Technology is full of acronyms that are really important and often times poorly or completely misunderstood. This glossary of terms will be helpful for understanding the presentation and materials we will cover in our workshop, and just for general knowledge!

HARDWARE TERMS:

Single Board Computer - a complete computer built on a single circuit board, with microprocessor(s), memory, input/output (I/O) and other features required of a functional computer.

Raspberry Pi – an example of a Single Board Computer that was designed for hobbyists and enthusiasts to offer a low-cost, fully functional computer in a small form-factor.

Sensor – a piece of hardware that detects some sort of condition and provides a signal based on the condition. Our sensor detects sound.

PIN – A part on a circuit board where an external device, sensor, input or output may be connected.

GPIO – an acronym that stands for General Purpose Input/Output. It is used to refer to any pin that is not defined by the pin itself but is defined at runtime (see "runtime" under Software Terms).

Breadboard - A breadboard is a construction base for prototyping of electronics. TRIVIA SIDE NOTE: (Originally it was literally a bread board, a polished piece of wood used for slicing bread. This was popular with early electronics builders because they were cheap, sturdy, and did not conduct electricity!)

Male and Female – this refers to the type of connection that a wire, sensor, or pin can accept. Breadboards are "Female" connections and so require a "Male" connector. The GPIO pins on the Raspberry Pi are "Male" and thus require a "Female" connector.

LED – Light Emitting Diode – a small device that uses a lot of very deep scientific principles to emit light. For our purposes, it is a small, electric light that can operate at VERY low voltage.

Circuit – any complete electrical connection is a circuit. From the lightbulb and light switch you used this morning, to your phone charger, to the circuit you will build in this workshop. Electricity flows from + to – through a circuit. No circuit, no flow of electricity.

Resistance – a property in electronics that allows us to reduce and control the flow of electricity through a circuit.

Resistor - a passive two-terminal electrical component that implements electrical resistance as a circuit element. This is the fancy way of saying that it causes the amount of electricity flowing through it to be reduced. Think of it as an electricity traffic cop for your circuit.

Pinout – this is the map of the Pins on the SBC to their function. Pinouts are critical for wiring a circuit – without it you do not know what the pins do!

Potentiometer a "variable resistor" that allows you to control the resistance of electricity through a circuit. Often embedded on a sensor, but can also be added by itself

SOFTWARE TERMS:

Code – the underlying construct of any computer software, app website, or anything that has to do with computers. There are many, many types of code. Code is written in

different languages. Like all languages, all different code languages have their own syntax, grammar rules, and general flow.

Python – A type of code language. There are many, this one is the one we will use. It is good for things like hardware interfaces because of its support for libraries that make integrations possible with little to no additional code.

Runtime – the time during which your code is running. This is when all the things you are telling the computer through your code to do actually happen.

Bug – an undocumented feature of your code. This is a behavior that you did not expect based on a set of conditions that happen through the use of your cod.

Runtime Error (also debug error, compiler error) – happens because you have something incorrect in the code. Usually this is caused by poor syntax or trying to do things in the code that the code doesn't allow.

Loop – a flow in code that is controlled by executing a test for a condition, and then responding by doing something until the test is passed. Examples of loops are: FOR, WHILE, IF, DO, and INFINITE

Comment – the most important part of any code. Comments are your notes to future generations about your code; why it is what it is, what it does, and how you got it there. Comment early and often for best results! In Python we use the "#" to start a comment.

Comment Out – yes there is another type of comment! This one allows you to try different things without deleting code. By "commenting out" code you can keep it in the program and try different things. Same "#" with a slightly different meaning.

Variable – much like the name implies, it is something that can have a variable definition. Variables are assigned to a value/values in your code and can be of many "types"

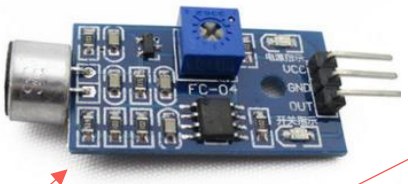
> **String** – just what it sounds like, a string is any string of text
>
> **Number** – numbers can have MANY types of values depending on the language. Python general deals in INT or integers. Our code will also use a Decimal number type

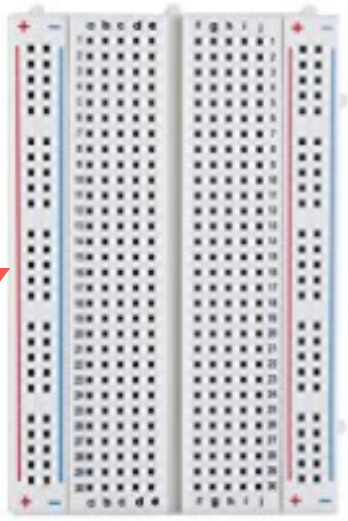
Keyword – keywords are reserved words that the language pre-defines and are key to the language itself. IF, ELSE, TRY, and IMPORT are all examples of keywords in Python

Code Block – a logical area of the code that is one part of the overall program. Blocks in Python are usually started with the "try:" keyword

USEFULL INFORMATION

What's In The Box!?





PARTS LIST:

1 x Sound sensor:

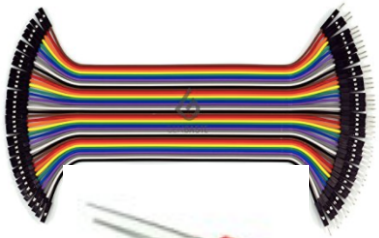
1 x Breadboard:

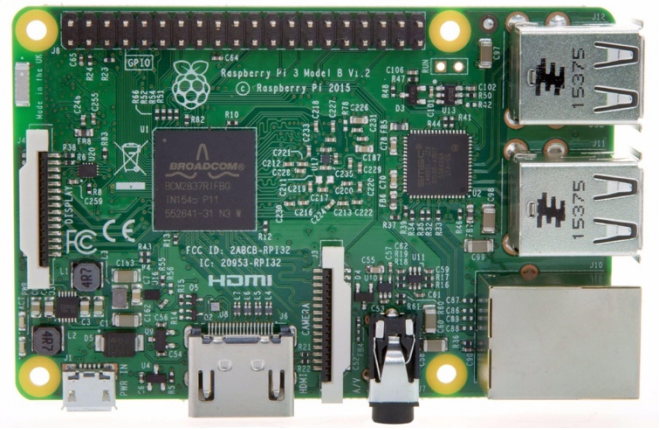
7 x Male to Female Wires:

1 x Red LED:

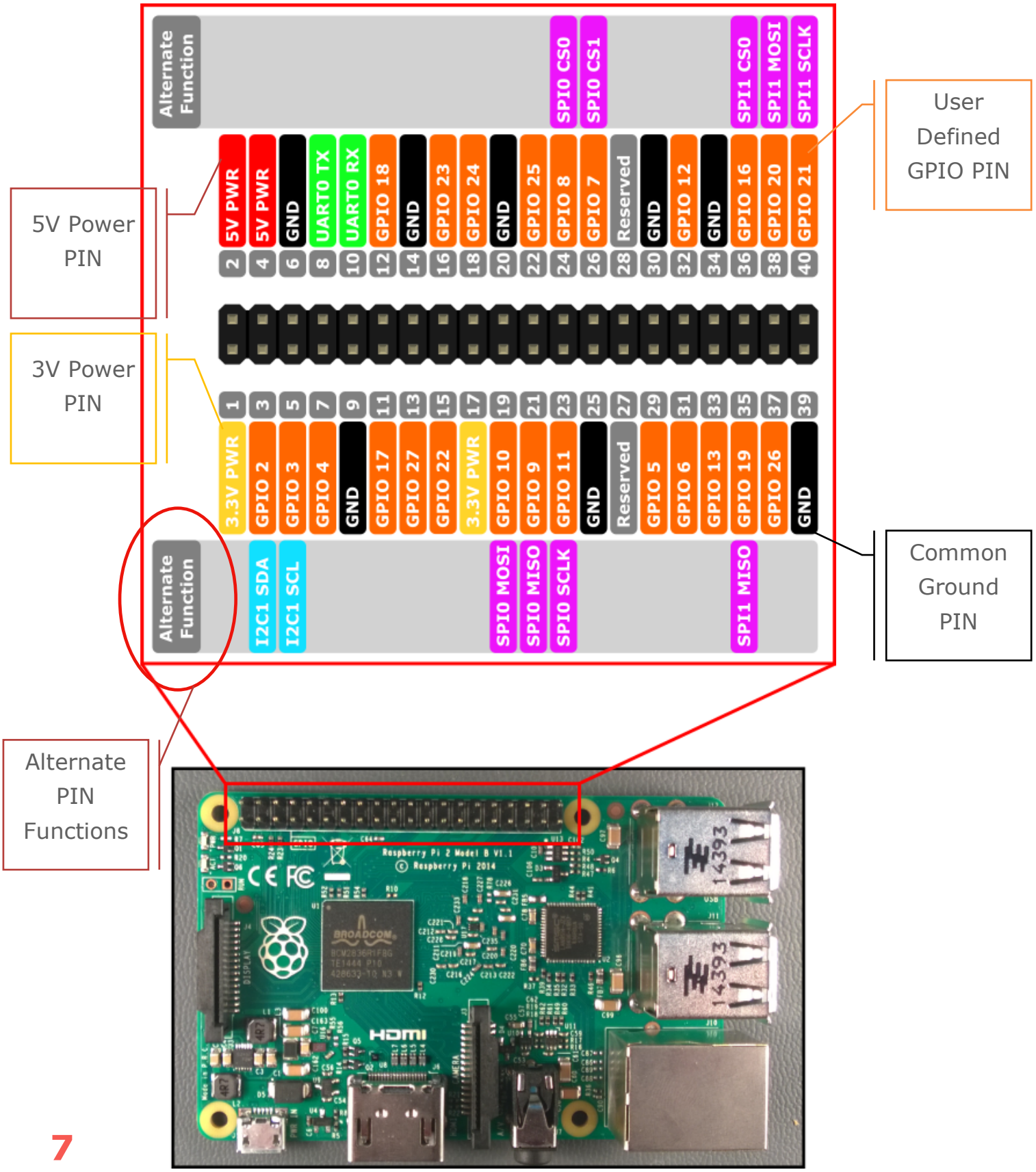
1 x Green LED:

1 x Raspberry Pi with power cord:

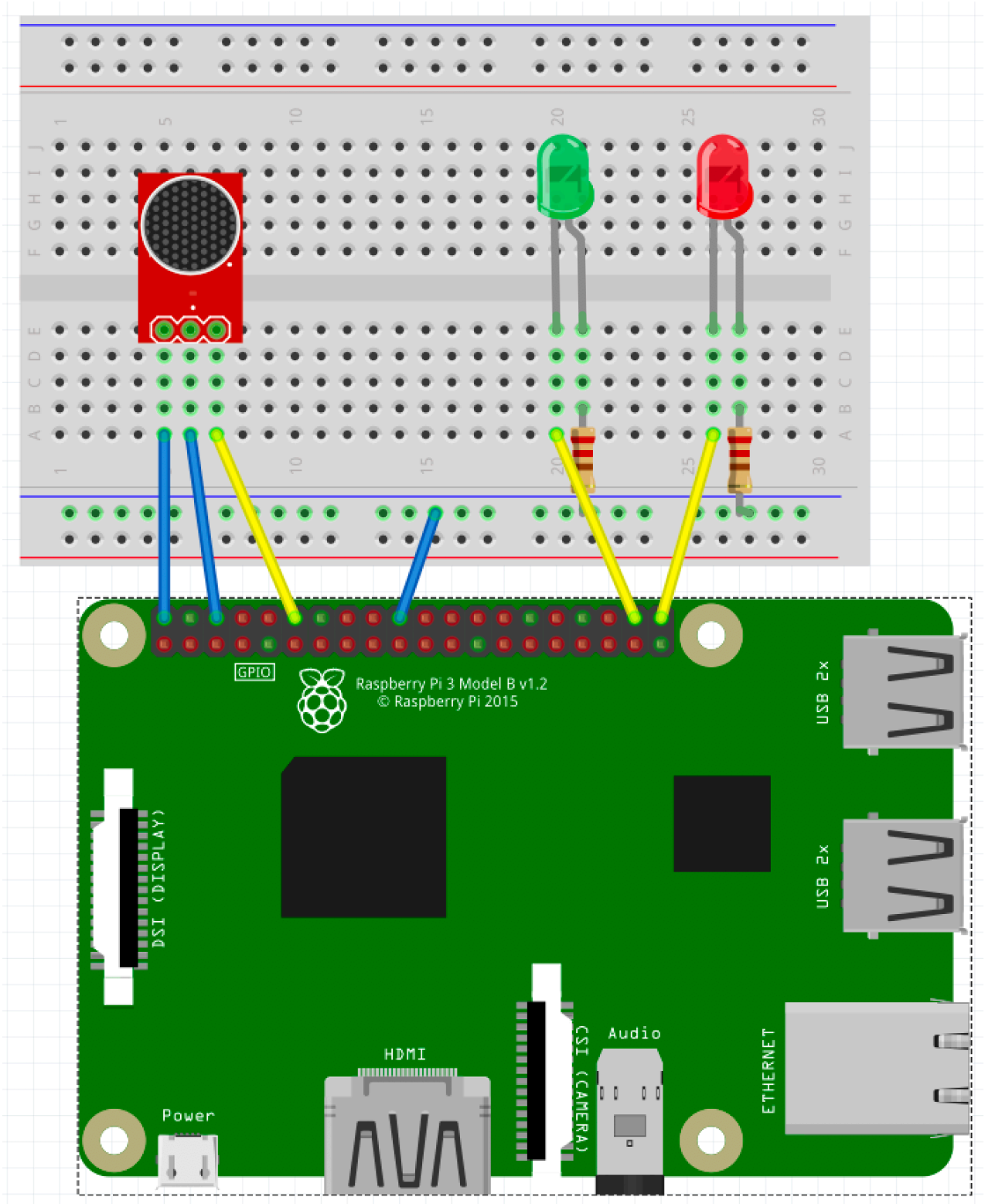


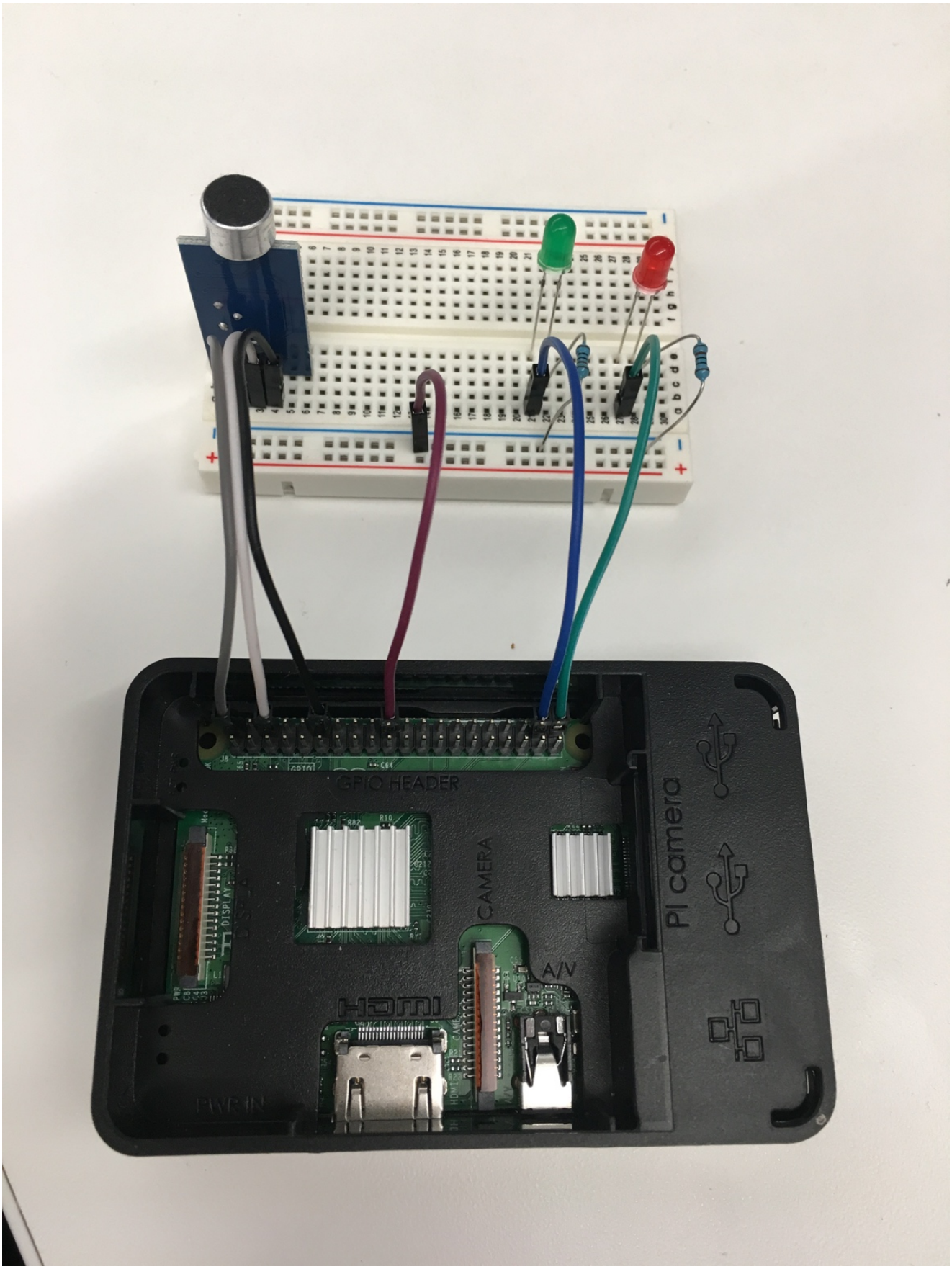


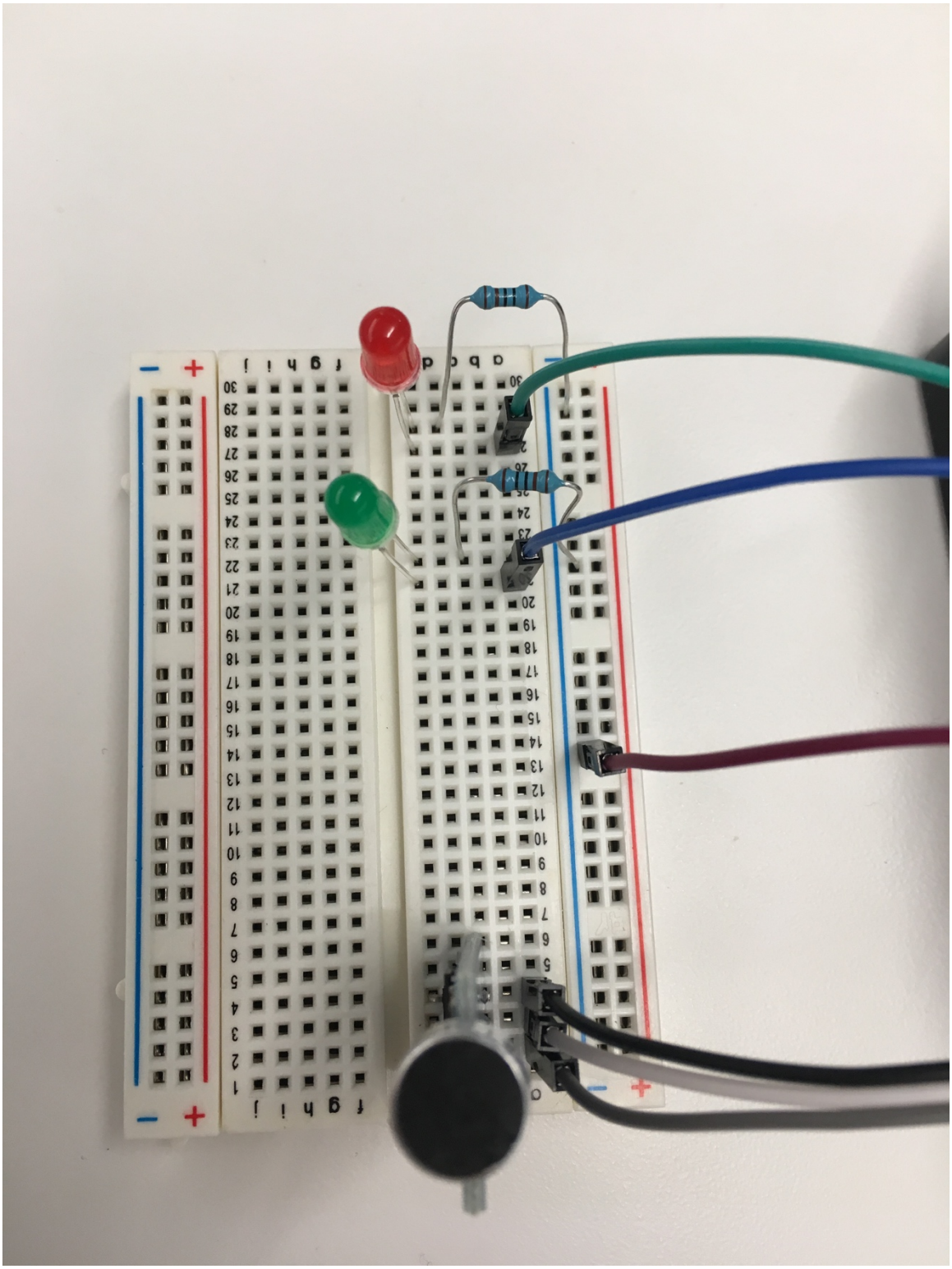
RASPBERRY PI GPIO PINOUT DIAGRAM:



WIRING PICTURES TO HELP YOU GET WIRED:




GPIO HEADER
CAMERA
DISPLAY
HDMI
A/V
PWR IN
PI camera



STEP-BY-STEP REFERENCE

I will be walking the group through the following commands as we progress through the workshop, but this way you can get caught up if you get lost (or sprint ahead if we are going too slow). Also, this will be stored on the github repository, so you can repeat this workshop on your own without me telling you what to do!

To use these steps:

- the line in the `preformatted font` tells you the command string to type.
- After you type in the command, press ENTER or Return on your keyboard
- If you get an error, you probably just had a typo. Pressing the up arrow will repeat the command in the command prompt and you can read it and find the typo and retry the command

COMMANDS

Here are the commands we will perform, in order, with the place the command is performed:

On your laptop - either inside the putty application (windows) or in terminal (Mac):

```
ssh pi@x.x.x.x
```

(in the workshop your IP is written on the box the parts are in)

When prompted for the password, it is: `raspberry`

Your new prompt will be: `charrold303_pixx >` <- where xx is a number. If you see that at the front of your command prompt you are ready to proceed to the next step.

We will be writing the code locally on your own system in whatever editor you wish. (I will suggest a few free ones in our session.) The code is on the next couple of pages, and we will walk through it together while people type it in.

THE CODE:

PLEASE NOTE! The copy-paste below has wrapped lines – you should **NOT wrap the lines** when you type them in. I tried to make the margins as big as I could, but there are still a few wraps so be aware of them (I will call them out during the walkthrough). I left spaces AFTER wrapped lines just for readability, but they do not need to be in your code when done!

You can also skip the lines that start with `print` if you wish to enter it in faster, but your application will not perform any notifications without them. This is OK if you trust it to run clean all the time. You will notice I am not the trusting sort…

You can skip the comments – lines that start with a # they are there to tell you what the actual commands do. The total of uncommented lines is around 50. We will input them together with the class, but you can skip ahead if you wish.

```
001: import RPi.GPIO as GPIO
002: import os
003: import time
004: from decimal import Decimal
005: import math
006:
007: print("Preparing to monitor sound levels")
008: print("You can gracefully exit the program by pressing ctrl-C")
009: print("Readying Web Output File")
010: web_file = "/var/www/html/table.shtml"
011: with open(web_file + '.new', 'w') as f_output:
012:    f_output.write("")
013:    os.rename(web_file + '.new', web_file)
014: Loud_Count = 0
015: louds_per = 0
016: per_detected = 0
017: time_loop = 5
018: stime = time.time()
019: etime = stime + time_loop
020: ptime = time.ctime()
021: Loops_Tot = 0
022: loop_count = 0
023: max_loop = 10000000
024: a_threshold = .00000001
025: sensor_in = 18
026: red_led = 21
027: green_led = 20
028: GPIO.setmode(GPIO.BCM)
029: GPIO.setup(red_led, GPIO.OUT)
030: GPIO.setup(green_led, GPIO.OUT)
031: GPIO.setup(sensor_in, GPIO.IN)
032: GPIO.output(green_led, GPIO.LOW)
033: GPIO.output(red_led, GPIO.LOW)
034: GPIO.output(green_led, GPIO.HIGH)
035: print("GPIO set. Service ready. Initiating Detection Protocol.")
036: GPIO.add_event_detect(sensor_in, GPIO.RISING, bouncetime=300)
037:
038: def dowork(sensor_in):
039:    global Loud_Count, loop_count, per_detected, max_loop, louds_per
040:    if GPIO.input(sensor_in):
041:         GPIO.output(red_led, GPIO.HIGH)
042:         Loud_Count = Loud_Count + 1
043:         louds_per = louds_per + 1
044:         per_detected = Decimal(louds_per) / Decimal(loop_count)
045:         per_detected = round(per_detected, 10)
046:         if per_detected > a_threshold:
047:              print("REALLY PRETTY LOUD! Detect vs Threshold: " +
str(per_detected) + " / " + str(a_threshold))
```

```
048:              print(str(loop_count) + "loops vs " + str(louds_per) + "
events")

049:         else:
050:              print("Meh. Some noise. Detect vs Threshold: " +
str(per_detected) + " / " + str(a_threshold))

051:              print(str(loop_count) + "loops vs " + str(louds_per) + "
events")

052:
053: try:
054:    etime = time.time() + time_loop
055:    while(True): #time.time() < etime:
056:         loop_count = loop_count + 1
057:         Loops_Tot = Loops_Tot + 1
058:         if GPIO.event_detected(sensor_in):
059:              dowork(sensor_in)
060:
061:              GPIO.remove_event_detect(sensor_in)
062:              time.sleep(0.25)
063:              GPIO.add_event_detect(sensor_in, GPIO.RISING, bouncetime=300)
064:         if time.time() > etime:
065:              with open(web_file, 'a') as f_output:
066:                   if louds_per > 5:
067:                        if louds_per > 10:
068:                             f_output.write("<tr><td align=center
bgcolor=red><font color=white>On " + str(ptime) + ", it was Loud!!</td><td
align=center bgcolor=red><font color=white>" + str(louds_per) +
"</font></td></tr>")

069:                        else:
070:                             f_output.write("<tr><td align=center
bgcolor=orange><font color=white>On " + str(ptime) + ", it was a little
loud.</td><td align=center bgcolor=orange><font color=white>" + str(louds_per)
+ "</font></td></tr>")

071:                   else:
072:                        f_output.write("<tr><td align=center
bgcolor=green><font color=white>On " + str(ptime) + ", it was pretty
quiet.</td><td align=center bgcolor=green><font color=white>" + str(louds_per)
+ "</font></td></tr>")

073:
074:              print("Reseting Counters")
075:              loop_count = 0
076:              louds_per = 0
```

```
077:              etime = time.time() + time_loop
078:              ptime = time.ctime(etime)
079:              GPIO.output(red_led, GPIO.LOW)
080:
081:
082: except (KeyboardInterrupt, SystemExit):
083:    print("-------------------------------------------")
084:    print(" ")
085:    print("System Reset on Keyboard Command or SysExit")
086:    print(" ")
087:    print("Total Noises Detected: " + str(Loud_Count))
088:    print(" ")
089:    print("Total loops run: " + str(Loops_Tot))
090:    print(" ")
091:    print("-------------------------------------------")
092:    GPIO.cleanup()
093:
094: else:
095:    print("-------------------------------------------")
096:    print(" ")
097:    print("System Reset for some reason")
098:    print(" ")
099:    print("Total Noises Detected: " + str(Loud_Count))
100:    print(" ")
101:    print("Total loops run: " + str(Loops_Tot))
102:    print(" ")
103:    print("-------------------------------------------")
104:    GPIO.cleanup()
105:
```

COMMANDS CONTINUED:

When you have finished entering your code we need to get it over onto the Raspberry Pi to actually execute it there. If you are a mac user, you will need to open a new terminal window, and execute:

`scp /THE/PATH/TO/yourfile.py pi@x.x.x.x`

(where x.x.x.x is the ip address of your pi)

For Windows:

From the cmd prompt navigate to the directory putty is installed in (Applications/putty most likely). From there use this command (pscp is putty's secure copy command):

`pscp /THE/PATH/TO/yourfile.py pi@x.x.x.x`

You will be prompted on both platforms for the password which is `raspberry`

To run your program, go back to your terminal window that is logged into your pi and run:

`sudo python3 myfilename.py`

At this point, one of two things will happen:

1. *It will run perfectly and work exactly as we expect*
2. *We will get an error and a line number and have to go debug our typos*

Spoiler alert – it's number 2 in almost every case.

Not to worry though, simply edit the offending line, and then repeat the scp/pscp commands and run the application until it runs without error. I will help you debug in the workshop! If we just cannot figure it out, there is a secret shortcut. The code is already on the Pis, and I will help you use that.

To know if it is working correctly, these 3 things happen:

1) The green LED turns on! (if it did not, flip the pins around)

2) *A bunch of logging starts on your terminal screen (if you left the "print" lines in the code*
3) *The red LED should activate when you make noise (flip the pins around if not)*

Now you are ready for the last part. There is a web server running on your pi, and you can see the stats of the sound detection at:

`http:x.x.x.x/sound.html`

where x.x.x.x is your IP address

If you want to do this workshop later at home, and do not want to retype the code from scratch, you can clone my git repository to your RaspberryPi with this command:

git clone https://github.com/ChrisHarrold/PiWorkshop.git

The code we are using is in the "Sounds" directory under the PiWorkshop folder.

IMPORTANT RESOURCES

Some good RPi books for your ongoing education!

Programming the Raspberry Pi, Second Edition: Getting Started with Python – by Simon Monk

Raspberry Pi Cookbook: Software and Hardware Problems and Solutions – also by Simon Monk

Raspberry Pi 3: From Noob to Master; Simple Step By Step Guide to Setting up Your Raspberry Pi 3 and Using It for a Wide Variety of Cool Projects – By Steve Ora

Quick Survey: How did I do? Was this worthwhile? Anything else?
http://bit.ly/charrold

GitHub Repository (code, this doc, other materials):
https://www.github.com/ChrisHarrold/PiWorkshop/Sounds

Pimoroni: A good source for beginner friendly hardware kits and project materials:
https://pimoroni.com

Putty: A windows-based ssh client (if you use a windows machine you will need this for the workshop and should just have it in general!):
http://bit.ly/_putty

SCP: The preeminent file copy tool for Raspberry Pi! Not really, it's a UNIX command that you will want to know for copying your code from your computer to the Pi
http://bit.ly/scp_command

GitHub: The preeminent code repository system for sharing and collaborating on code (yes, really it is)
www.Github.com

ImgFlip: Because memes are life
https://imgflip.com/memegenerator

Fritzing: Great little tool for creating your own very simple and easy to read/share/use wiring and circuit diagrams:
www.fritzing.org

My Website (updates, other projects, blog, etc.):
https://www.charrold303.com