Linux Information

Options surrounded by brackets are optional

ls	List all files and directories in current working directory
cd dirname	Move to a new subdirectory, or “room”
	**Use the name of the directory you want to move to
cd ..	Move to parent directory (i.e. /Home/Pictures to /Home)
pwd	Print the path of the current (working) directory
nano file1	Open a file in the text editor
	Press ctrl+x to quit, follow prompts
cat file1	Display the contents of the file (no scrolling)
less file1	Display the contents of the file (allows scrolling)
	Press q to quit when done
cp file1 file2	Copy file1, naming the new copy as file2
mv file1 file2	Rename file1 to file2
mv file1 dir1/	Move file1 into the directory dir1
rm file2	Delete (remove) the file named 'file2'
man rm	View the manual for the 'rm' utility
mkdir dirname	Create a directory with the name 'dirname'
rmdir dirname	Remove the empty directory with the name 'dirname'

All of these utilities have switches to alter their functionality.
Try: ls -l
Hidden files (ones that are preceded with a period) can be viewed by using the -a tag on ls.
Try: ls -al

Note: Using your ls command to list files in the directory, if something is lit up in purple, it is another directory. If it’s in green, it’s a program that you can run that’s been marked as executable. If a program ends in .py, it needs the python interpreter to run. Items ending in .txt, .lst, etc. are files and you must use the less or cat commands to view them.

Terminus

The Terminus game may help you get comfortable with some of these commands
http://web.mit.edu/mprat/Public/web/Terminus/Web/main.html

Encryption

ciphor.py --help
Show cipher help

ciphor.py -c 3 -e “hello world”
Encrypt the message, hello world, with a Caesar Cipher shifting characters 3 to the left. -c designates Caesar Cipher. The 3 designates the size of the shift, and -e is encrypt (shift to the right).

ciphor.py -k “mykey” -e “hello world”
Encrypt the message, hello world, with a key cipher with key “mykey”. -k designates a key cipher, -e is for encrypt and “mykey” is the key used for encryption.

ciphor.py -c 3 -d -f ciphered_message1.txt
Decrypt the file, ciphered_message1.txt, using a Caesar Cipher and a shift of 3. -d designates decryption (shift to the left).

ciphor.py -S -f plain_text1.txt
Generate the signature for plain_text1.txt. -S will show the signature and -f designates an input file.

ciphor.py -s -f ciphered_message1.txt
Generate the letter frequencies for ciphered_message1.txt -s will show the statistics (letter frequencies) for the input text.

ciphor.py -k “mykey” -e -f ciphered_message1.txt -o encrypted_plain_text1.txt
Encrypt plaint_text1.txt and output the encrypted file into the file named encrypted_plain_text1.txt. -o sets the output file.

Password Cracking

View the password file:	**cat /etc/passwd**
View the shadow file:	**sudo cat /etc/shadow**
Crack Passwords	**john myshadowfile**
Show already cracked passwords	**john --show shadowfile**
Specify a hash format	**john --format=md5crypt shadowfile**
List the hash formats that you can specify	**john --list=formats**

All Linux Commands

Command	Description
cd [path]	Move into [path]
cd	Go to home directory
cd ..	Go up a directory (. = current directory | .. = up a directory)
cd ~ cd	Move to the home directory (/home/[username]) Move to the home directory (/home/[username])
ls [options] [path]	List the contents of a directory -l table format -a show everything (including hidden files)
[command1] | [command2]	Take the output of [command1] into [command2] (Use the pipe | character)
pwd	Shows the complete path to the directory you're in
chmod +x [filename]	Make [filename] an executable (runnable)
cat [path]	Display the contents of the file
less	Shows a little bit of the command output or file

man [file/command]	Show how to use a [file/command]
mv [/path/to/oldFile] [/path/to/newFile]	Moves [oldFile] into [newFile path]
mkdir [name] rmdir [name]	Make a directory with a [name] Delete the directory with the [name]
touch [filename]	Creates an empty file called [filename]
ps -aux	Show the processes currently running
xxd [binaryfile]	Shows the binary of the [binaryfile] *Also shows ansi*
cp [/path/to/oldFile] [/path/to/newFile]	Moves [/path/to/oldFile] to [/path/to/newFile]
md5sum [file]	Shows the md5 hash of [file]
rm [file]	Deletes [file]
python3 python3 [filename]	Starts you in the python environment (>>) Runs [filename] with python3
ssh [uname]@[ipaddr]	Remote connect to a system with the username [uname] at the IP [ipaddr]

grep [option] [patters] [file] Ex. grep -E “(ht|f)tp” *.txt Ex. grep -E “[0-9][0-9]” *.txt Ex. grep -P “\d\d-\d\d\d” *.txt	Search for [word] in the directory . any character ? repeat 0 or 1 times * repeat 0 or many times + repeat 1 or many times [a-dh] match any characters listed (a, d, or h) \d finds any number \w finds any word \s find any space characters (needs -P [option]) Looks for http or ftp () = group, | = or Look for 2 numbers between 0 and 9 Look for 2 numbers, a hyphen, then 3 numbers
sudo adduser [use rname]	Add a user with the [user name] *Ignore room number and everything else*
sudo deluser [user name]	Delete the user with the [user name]
su [user] su root	Switch to the [user] Switch to the root user
sudo apt update	Update the system
sudo apt install [package]	Install a [package] on the system
john [file]	Crack the [file]

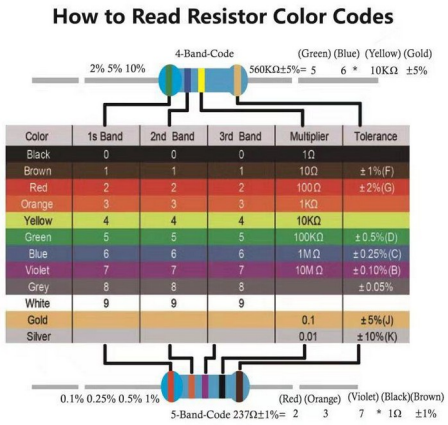
john --wordlist=[wordlist] [file]	Crack the password in the [file] with the list of words in [wordlist] Ex. [wordlist] = /usr/share/john/password.lst
john --show [file]	Show the already cracked password for the [file]
john --format=[format] [file]	Crack the password in the [file] with [format] Ex. [format]=md5crypt
john --list=formats	List formats you can use as a [format]
ciphor.py ciphor.py --help ciphor.py -c [shift] -e [message] ciphor.py -k [key] -e [message] ciphor.py -c [shift] -d -f [file] ciphor.py -S -f [file] ciphor.py -k [k] -e -f [m] -o [e]	Runs ciphor.py -k Designates a key cipher -c Designates a Caeser Cipher w/ a shift -d Designates decrypt -S Designates a signature -f Designates an input file -s Shows the stats (letter frequencies) -o Designates an output file Show ciphor help (shows above options) Decrypts [message] with a shift of [shift] Encrypts [message] with the [key] Decrypt [file] with a shift of [shift] Generate a signature for [file] Encrypt file [m] w/ key [k] and output it

	to file [e]
rsaHex.py [options] [message] rsaHex.py -e [exp] -n [mod] [hx] Ex. rsaHex.py -e 7 -n 33 2	Runs rsaHex.py with [options] on [message] -e exponent -n modulus Encrypting [hx] with a key of ([exp], [mod]) Encrypting a message of 2 w/ a key of (7, 33)

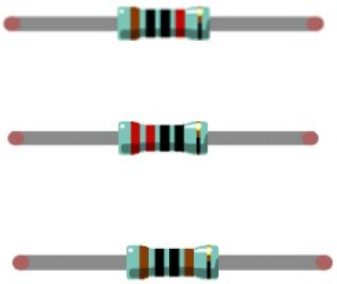
ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

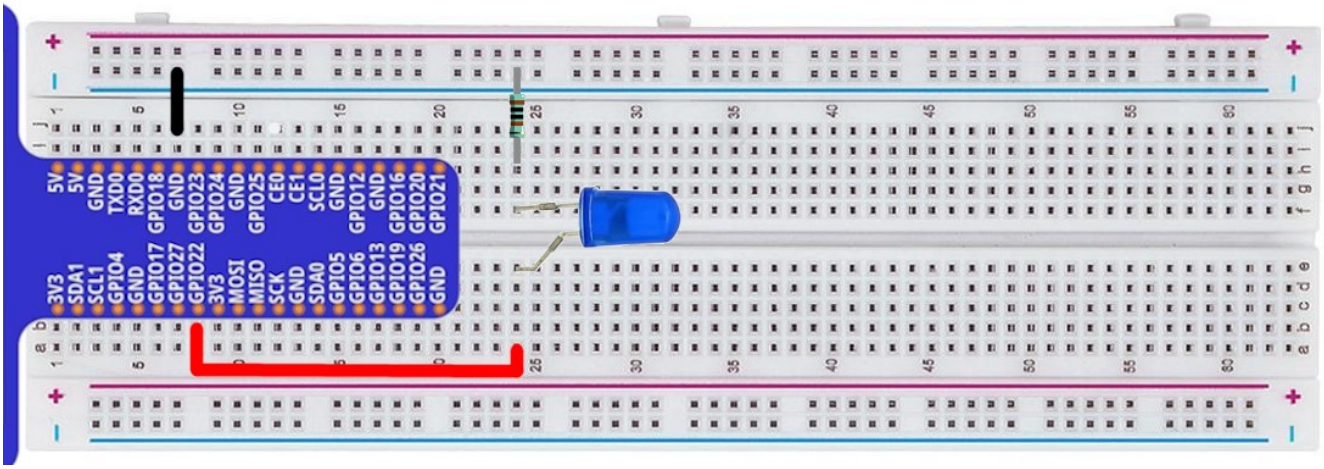
Electronics



Can you determine the resistance of the following resistors?



Connecting an LED to GPIO Port 22



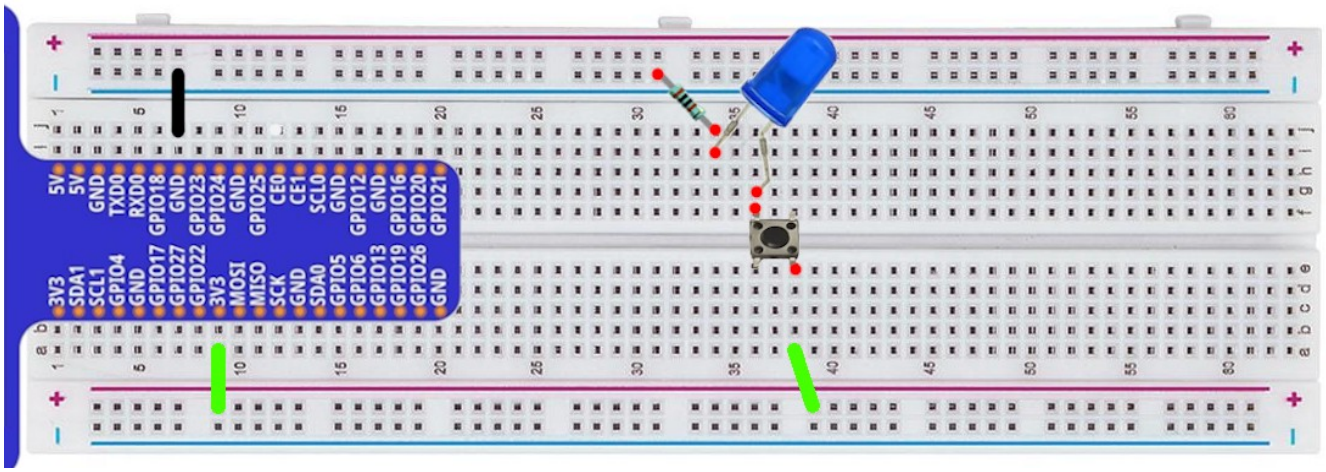
In the terminal start the python interpreter: >>> python3
Note: >>> denotes the interpreter prompt - you don't type these characters, only the prompt that follows.
Note: Anything that follows a '#' is a comment - it and everything after it doesn't need to be typed either.
Note: A new line in the handout/powerpoint indicates a new line of code. If it's on the same line, type it on the same line in your terminal, then hit enter to run the line of code before moving onto the next line.

```
>>> from gpiozero import LED
>>> led1 = LED(22)    # telling python led1 is an LED on GPIO port 22
>>> led1.blink(.4, .4)  #this will cause it to turn on/off every 0.4 seconds
>>> led1.on()                  #this will turn it on
>>> led1.off()                 #this will turn it off
```

Note: Be careful not to mix up your ground (-) and positive (+) connections, this can fry the component attached to your circuit!

Interrupting the circuit with a button

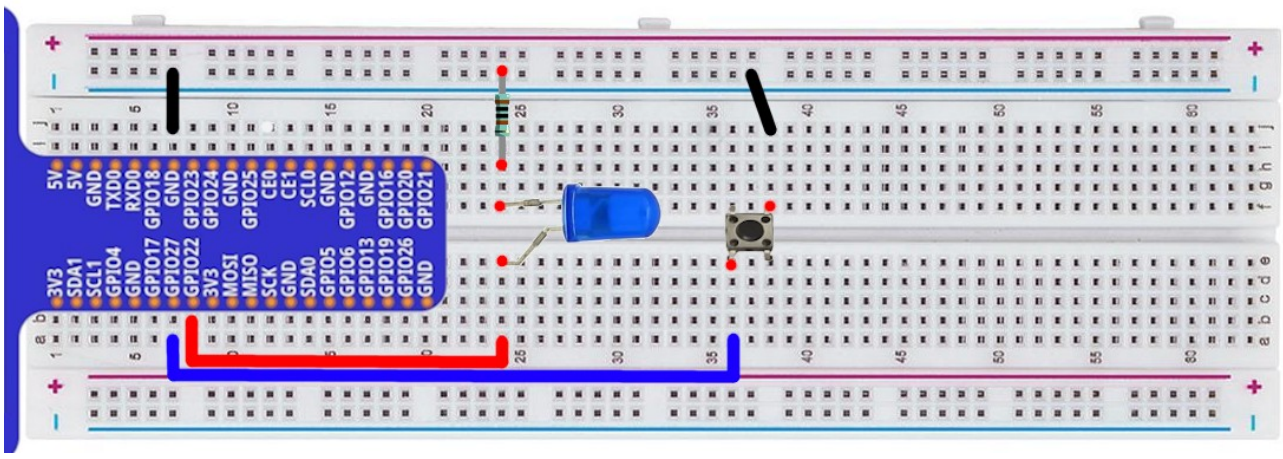
A button can be used to break the circuit
When the button is pressed, the electricity can flow through the circuit to the LED
This is not controlled by code



Connecting a button to a GPIO pin

GPIO 22 controls the LED
GPIO 27 reads the status of the button
Try the following (Remember, you need to have your Python interpreter open)

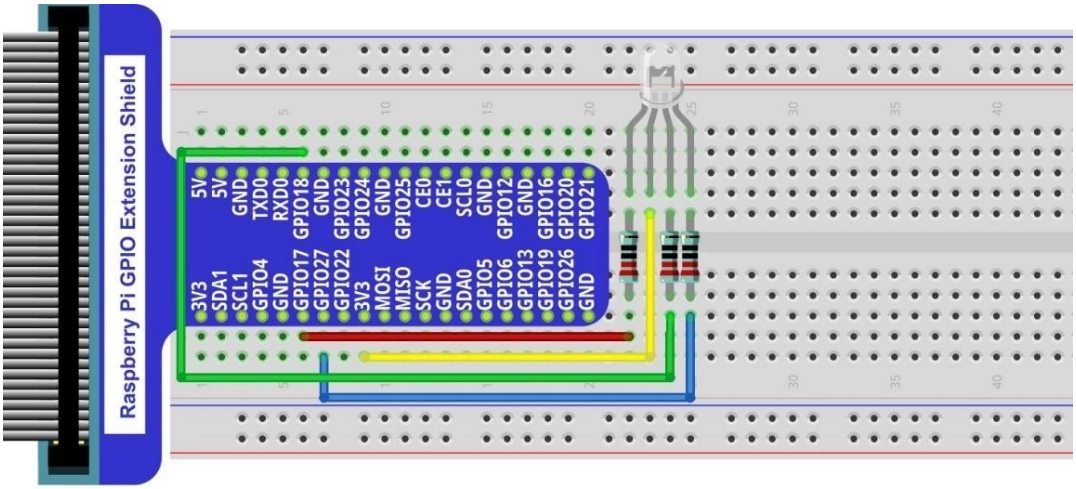
```
>>> from gpiozero import LED
>>> from gpiozero import Button
>>> led1 = LED(22)
>>> button1 = Button(27)
>>> button1.when_pressed = led1.on       # turn the light on when the
                                            button is pressed
>>> button1.when_released = led1.off     # turn the light off when the
                                            button is released
>>> button1.when_pressed = led1.blink  # blink while the
                                            button is pressed
```



RGB Led

The long pin is the anode - it connects to the positive side of the circuit

```
>>> from gpiozero import RGBLED
>>> led = RGBLED(17, 18, 27)              # control red with GPIO 17, etc
>>> led.color = (1, 1, 0)                 # full red and green (yellow light)
>>> help(RGBLED.blink)                    # check out the parameters that
                                                  can be set
>>> led.blink(on_time=.2, on_color=(1,0,0), off_color=(0,1,0)
>>> help(RGBLED.pulse)                    # see if you can get it to pulse red
>>> dir(RGB)                              # see all the functions, then try
                                          putting them in help to explore
```

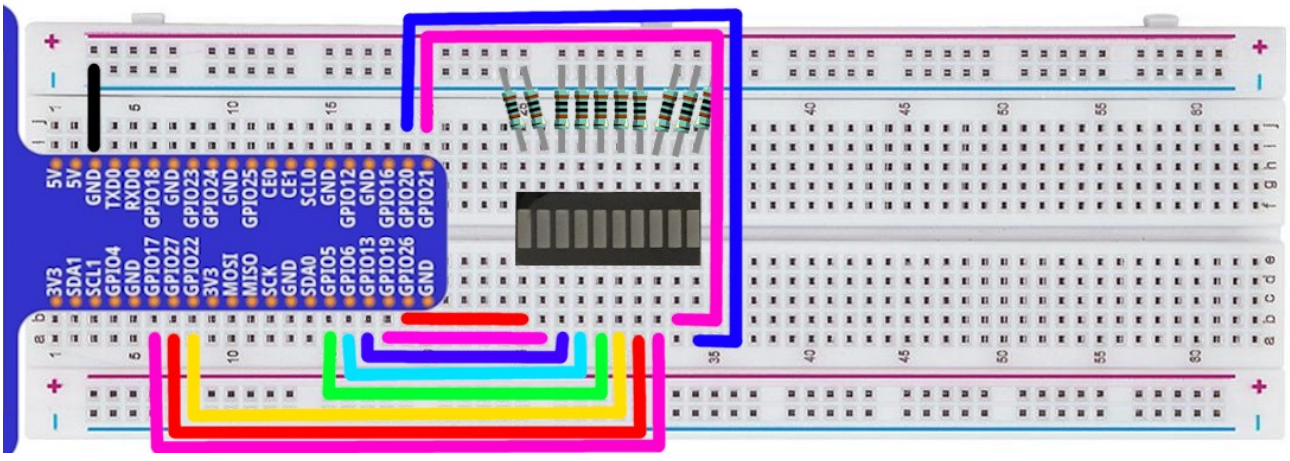


Bar graph

Each LED is controlled individually. 10 LEDs means 20 connections total

There is some writing on the bar graph. It should be oriented toward the bottom of the picture

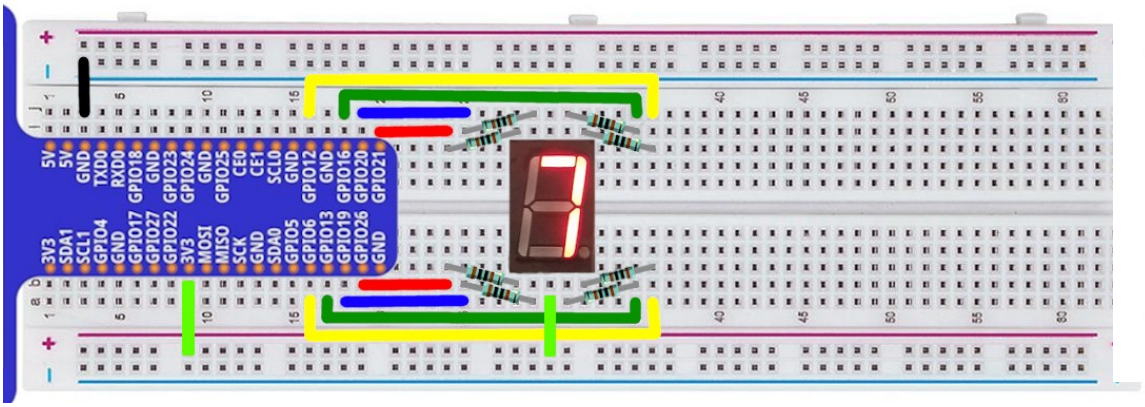
```
>>> from gpiozero import LED
>>> led1 = LED(26)              #  etc
>>> led1.on()                   # you can control each led using on, off, blink
```

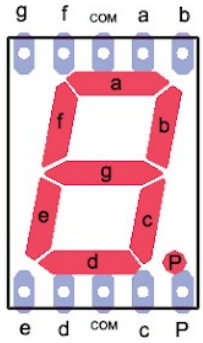


7 Segment display

This reduces the connections by having a common cathode

```
>>> from gpiozero import LED
>>> led_a = LED(16, active_high=False)        # control the top LED (on
                                               when this pin is low)
```





LED Matrix Display

Note the orientation of the matrix (the writing is on the top on this picture)

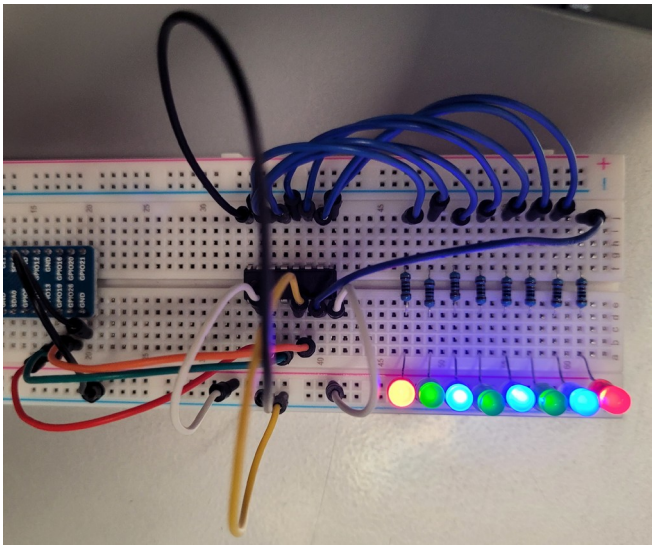
Connect each of the pins as shown, connecting through resistors for those in red

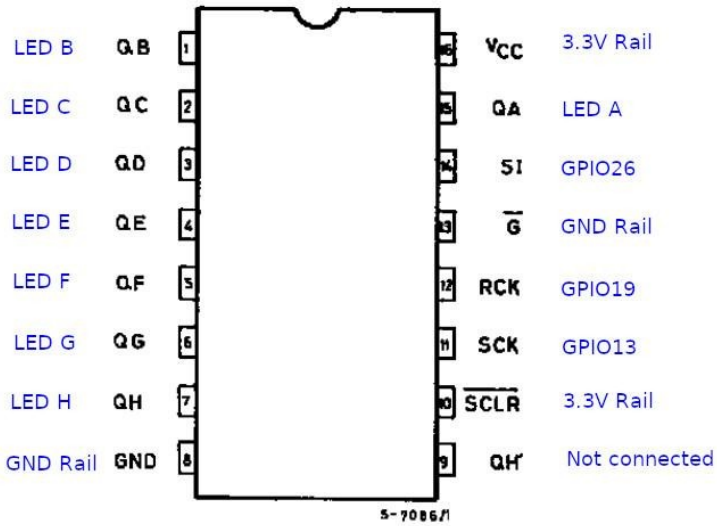
In the dotmatrix directory - try modifying the dots.txt (two blank lines between each frame)

```
>>>python3 testmatrix.py
>>>python3 matrix.py dots.txt
```

Serial to Parallel chip with LEDs (74HC595)

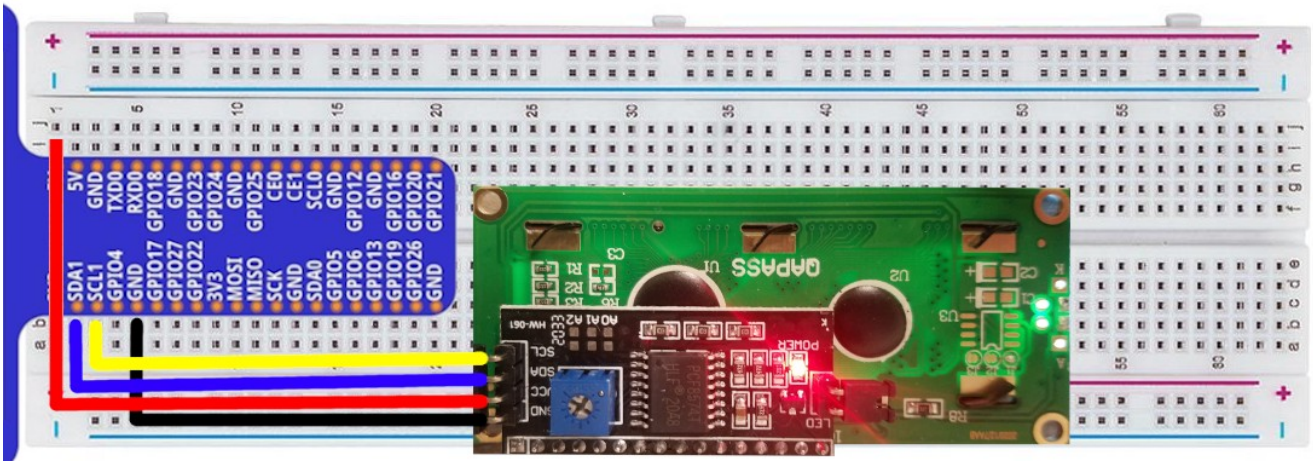
Connect the LEDs through resistors. (I have them all attached to the ground rail in this picture)
In serialChip,
>>>python3 8light.py
Display any byte (by giving it 1s and 0s)





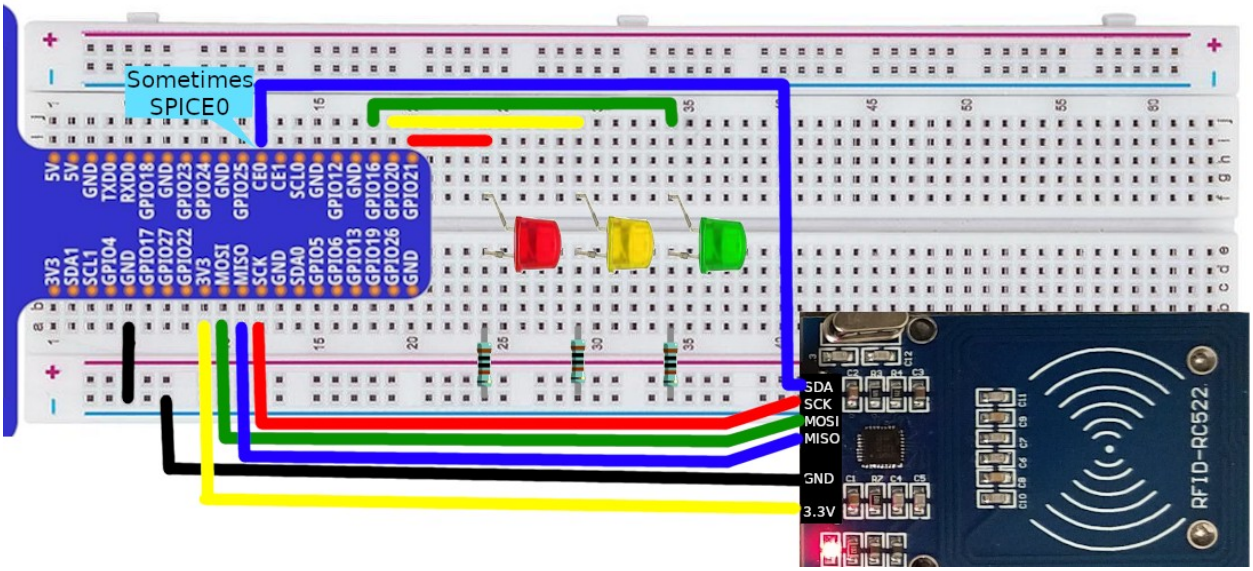
LCD Screen

>>>python3 lcdi2c.py
>>>python3 lcd_display.py

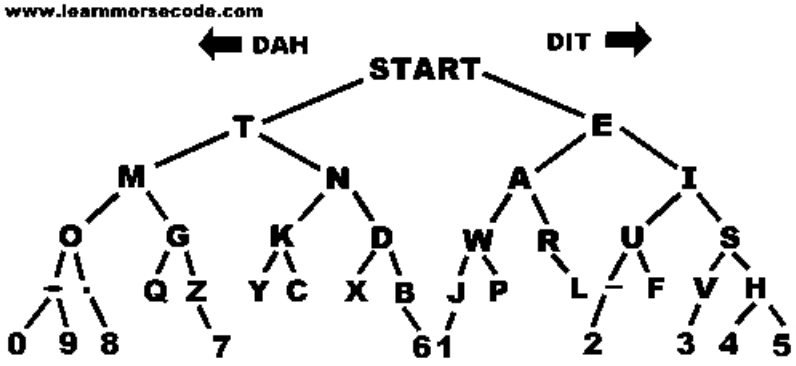


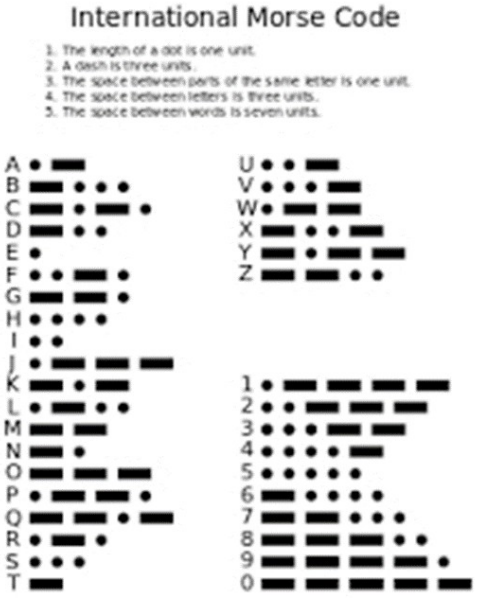
RFID Reader

readrfid.py display_rfid_data.py writeToRFID.py access_card.py



Morse Code

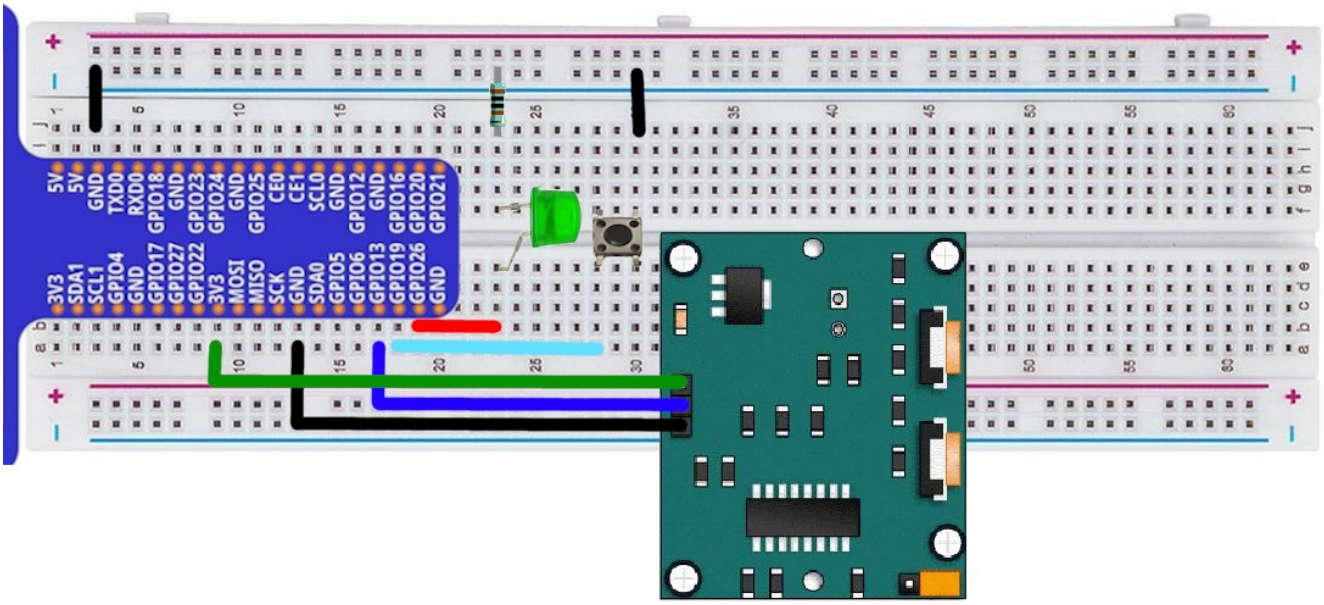




Connect an led to GPIO22 (like we did on page 1)
In the morseCode directory, run:
>>>python3 morseChallenge.py
How high of a score can you get?

PIR Sensor

>>>motiondetect.py



Distance Sensor

>>>distanceSensor.py