Intro to Hacking with the Raspberry Pi

Sarah Withee
@geekygirlsarah

Hello!

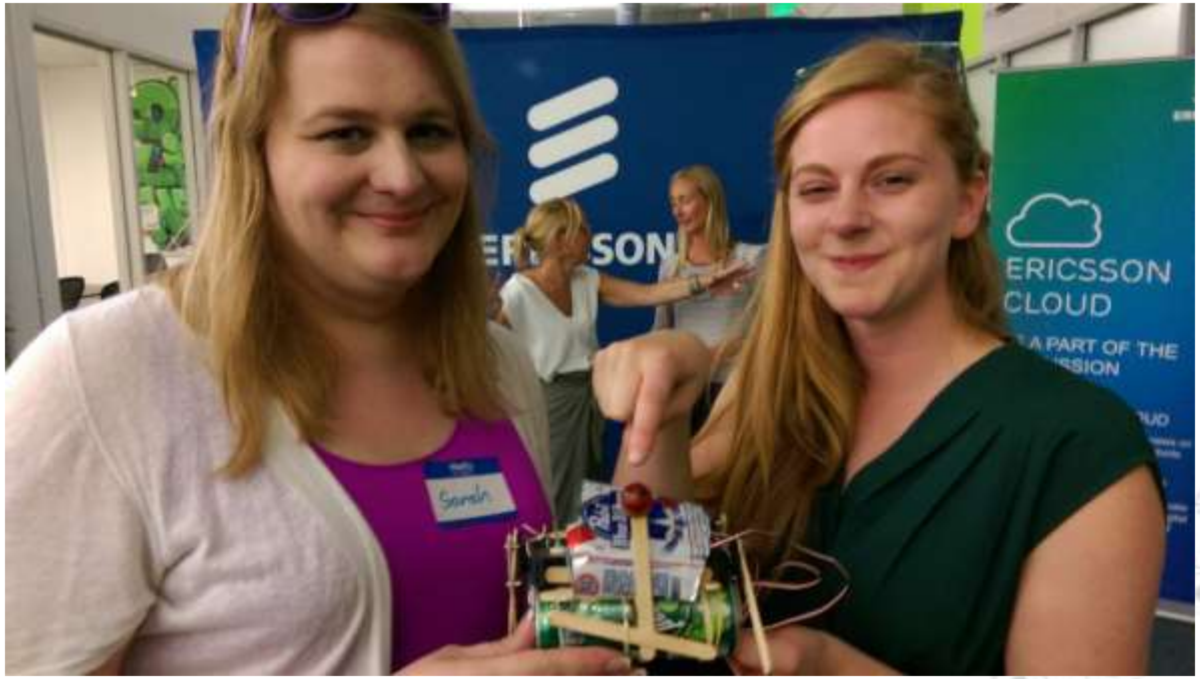
I am Sarah Withee

Friendly polyglot
software engineer

Hardware tinkerer

Robot builder

You can find me at @geekygirlsarah



A Few Questions To Start…

- Who has heard of the Raspberry Pi?
- Who owns at least one?
- Is it sitting in a closet/junk drawer?

- Are you daunted at the idea of tinkering with hardware or electronics?

This talk is probably for you!

A Few Questions To Start…

If you:

- understand circuitry basics,
- are familiar with breadboards and wiring,
- and know how to program GPIO pins on the Pi

This talk *may* not be for you.

(I won’t be offended if you leave)

Breakdown

- Raspberry Pi Information and Models
- Intro to Hardware
- Intro to Programming Hardware
- Project 1 - LEDs
- Project 2 - Sensors
- Project 3 – Push Buttons and LCD screen
- Future Project Inspriation
- Conclusion

“

“The Raspberry Pi is a credit-card sized computer that plugs into your TV and a keyboard. It is a capable little computer which can be used in electronics projects, and for many of the things that your desktop PC does, like spreadsheets, word-processing and games. It also plays high-definition video. We want to see it being used by kids all over the world to learn programming.”

Raspberry Pi Foundation, raspberrypi.org/faqs

“

*“We are **taking back the term ‘Hacking’** which has been soured in the public mind. Hacking is an art form that uses something in a way in which it was not originally intended. This highly creative activity can be highly technical, simply clever, or both. Hackers bask in the glory of building it instead of buying it, repairing it rather than trashing it, and raiding their junk bins for new projects every time they can steal a few moments away.”*

Hackaday, hackaday.com/about/

Resources

Slides, diagrams, videos, and more!

sarahwithee.com/raspberrypi

1.

Raspberry Pi Information and Models

The basics of this delicious computer

History of the Raspberry Pi

- 2006 – Had idea for affordable computer for kids
- 2008 – Processors became cheap enough to make it feasible to build
- 2011 – 50 alpha boards, 25 beta boards, 10,000 production boards
- 2012 – Sold first units
- 2013 – Sold > 2M units
- 2015 – Pi 2 Model B and Pi Zero released
- 2016 – Pi 3 Model b released
- 2017 – Pi Zero W released

Raspberry Pi Models

Pi Model A+

Pi Model B+

Pi 2 Model B

Pi 3 Model B+

Pi Zero

Pi Zero W

Raspberry Pi Model Comparisons

		Processor	Speed	Memory (shared w/GPU)	Expansion	USB Ports	Ethernet/ Wireless/ Bluetooth	Cost	Power Supply Needed
Pi Model A+ +*		ARMv6 (32-bit)	700 MHz	512 MB	Micro SDHC	1	None	$20	700 mA
Pi Model B+ *		ARMv6 (32-bit)	700 MHz	512 MB	Micro SDHC	4	10/100 mbps	$25	1.8 A
Pi 2 Model B		ARMv8 (64-bit)	900 MHz	1 GB	Micro SDHC	4	10/100 mbps	$35	1.8 A
Pi 3 Model B+		ARMv8 (64-bit)	1.4 GHz	1 GB	Micro SDHC/ USB Boot	4	10/100/1000, 802.11ac, BT 4.2	$35	2.5 A
Pi Zero		ARMv6 (32-bit)	1.0 GHz	512 MB	Micro SDHC	1	None	$5	1.2 A
Pi Zero W		ARMv6 (32-bit)	1.0 GHz	512 MB	Micro SDHC	1	801.11n, BT 4.1	$10	1.2 A

2.

Intro to Hardware

Wires and circuits and breadboards, oh my!

Circuits

A complete route that electricity can travel through

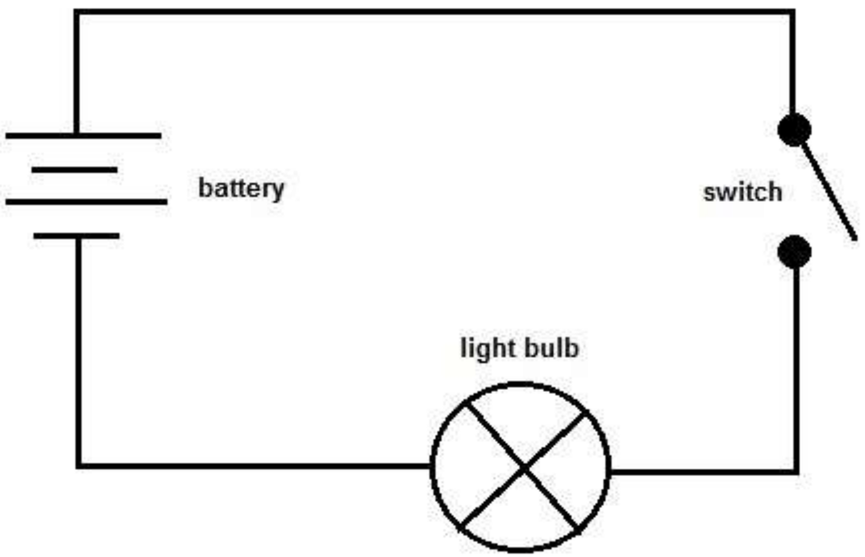
Intro to Hardware – Circuits

Series Circuit

Electricty can flow one direction

Close the switch to make the complete path

Open the switch to stop the electric flow



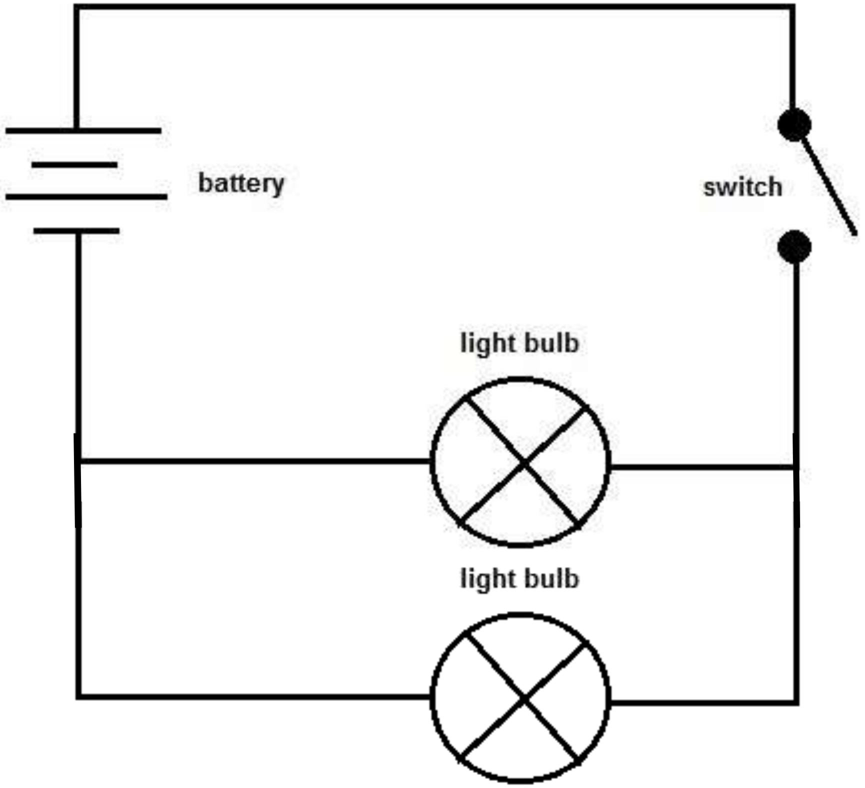
Intro to Hardware – Circuits

Parallel Circuit

Electricty can flow more than one direction

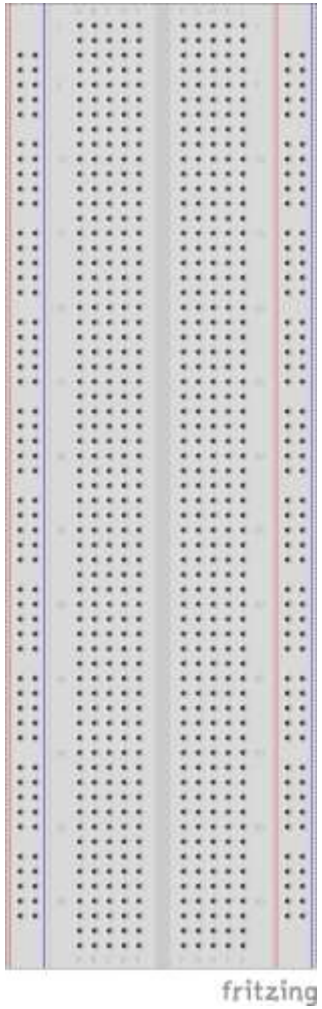
Close the switch to make both paths complete

Open the switch to stop all the flow



Breadboards

Device to quickly prototype circuits without solder

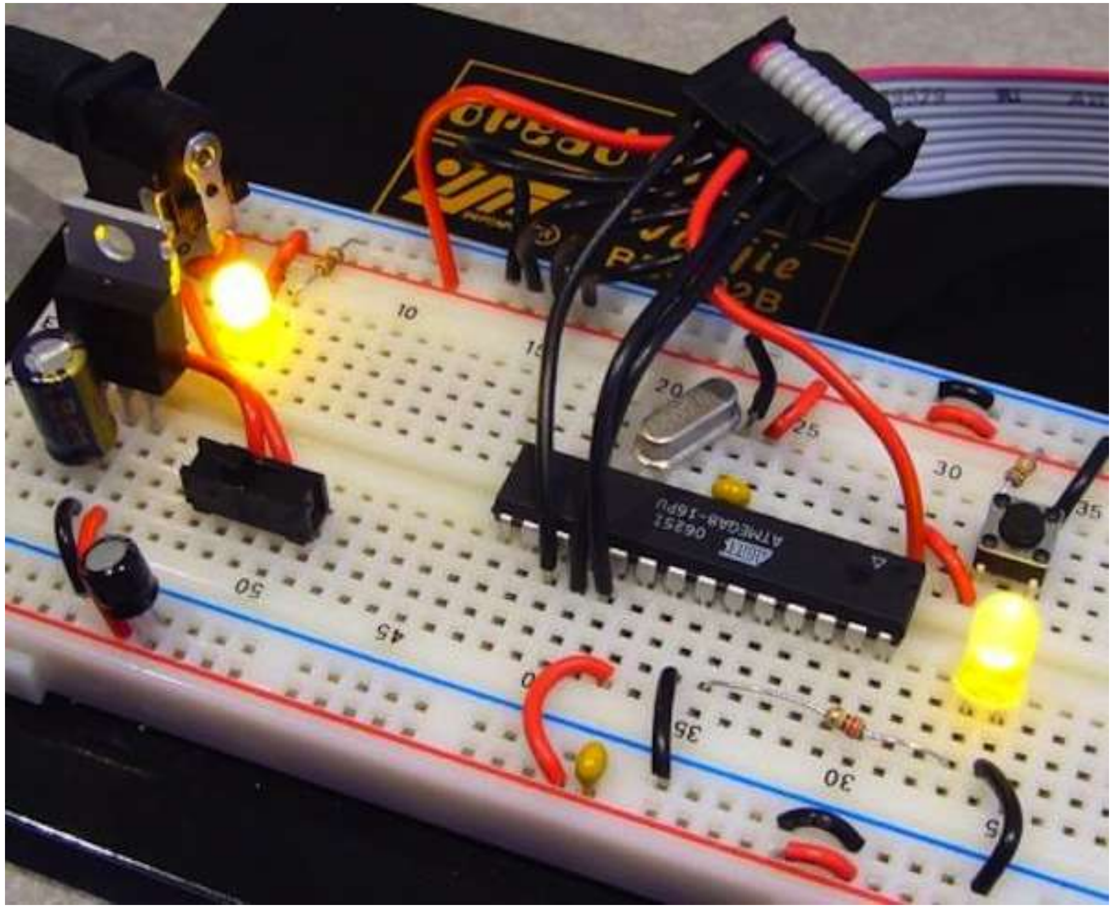


Intro to Hardware - Breadboards

A *literal* breadboard

(http://www.instructables.com/id/Use-a-real-Bread-Board-for-prototyping-your-circui/)

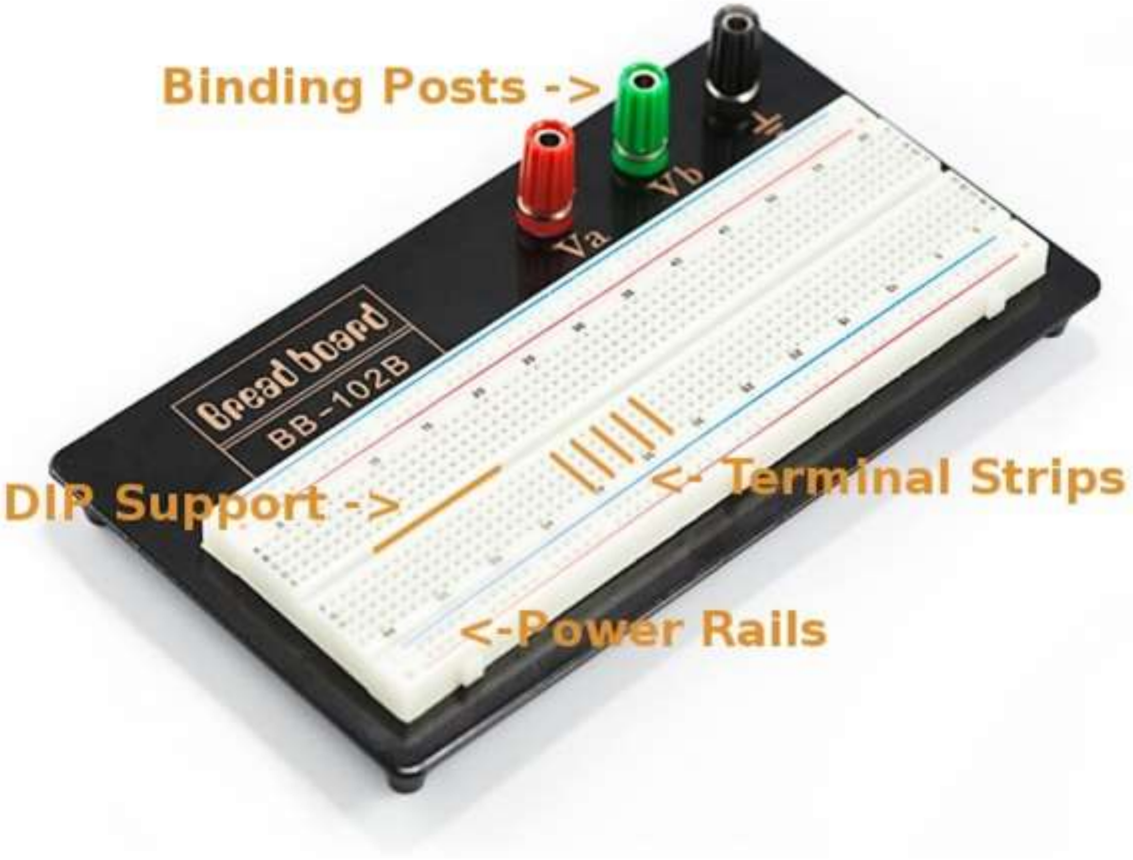
Intro to Hardware - Breadboards



Prototype Circuit

(https://learn.sparkfun.com/tutorials/how-to-use-a-breadboard)

Intro to Hardware - Breadboards

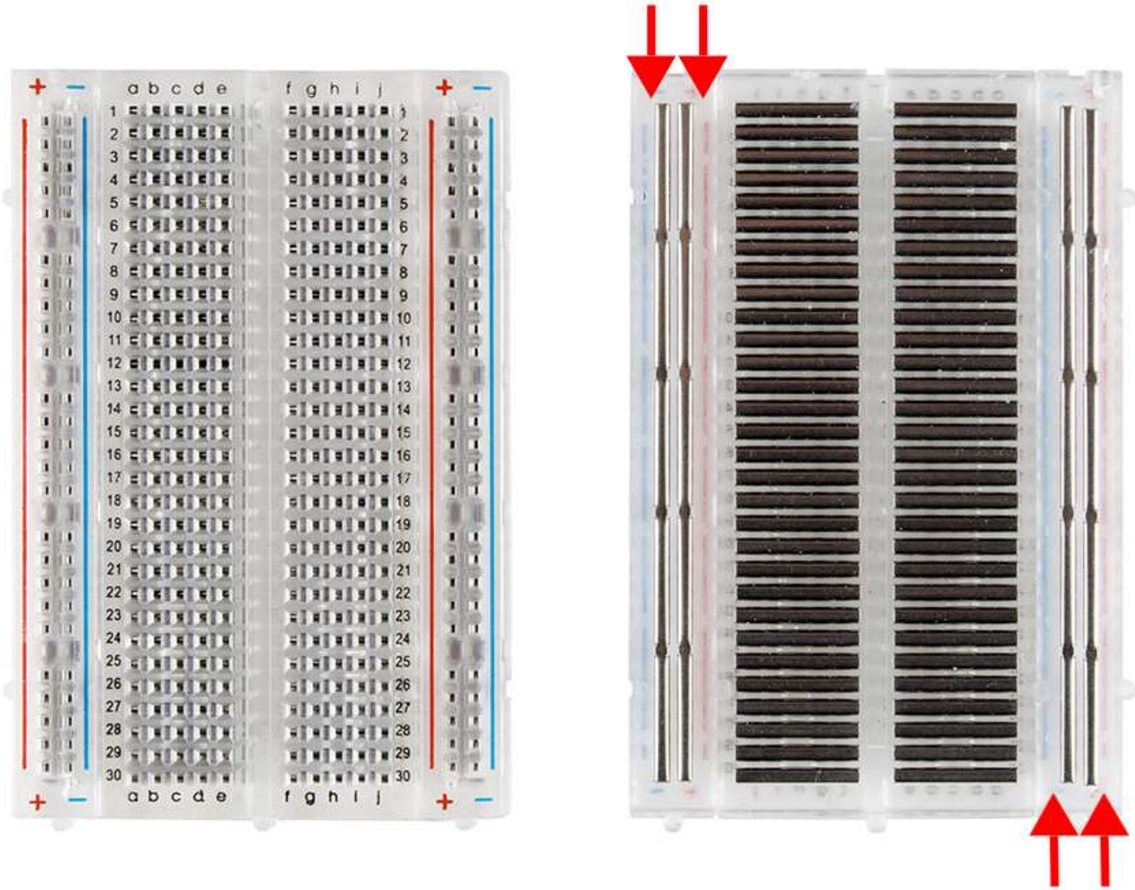


Breadboard features

(https://learn.sparkfun.com/tutorials/how-to-use-a-breadboard)

@geekygirlsarah @codeandsupply

Intro to Hardware - Breadboards



Terminal Strips and Power Rails

(https://learn.sparkfun.com/tutorials/how-to-use-a-breadboard)

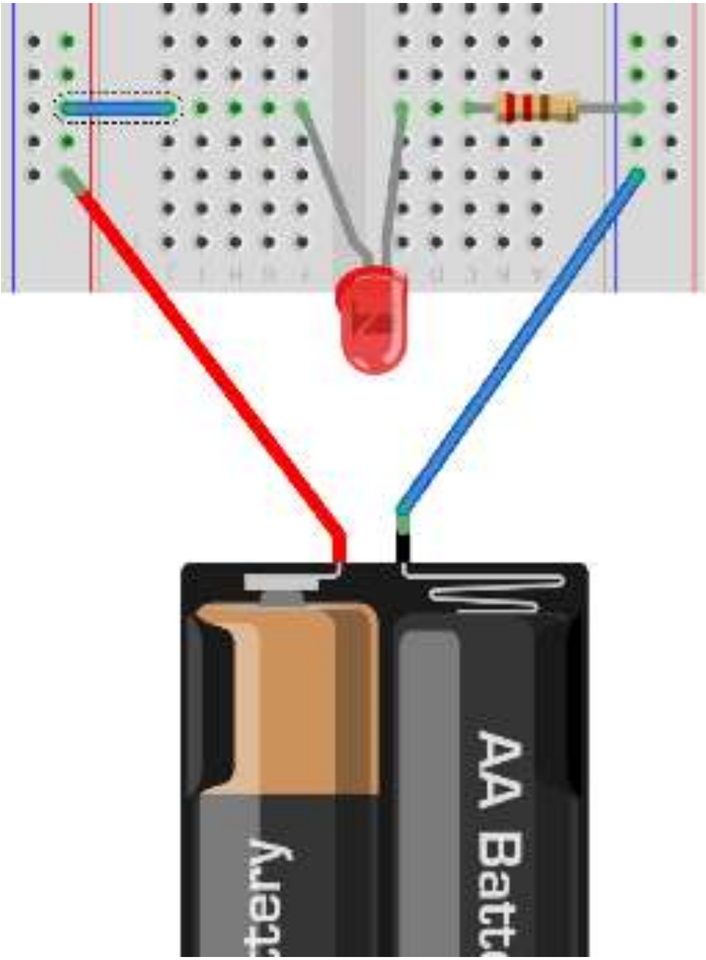
Intro to Hardware – Breadboards

Circuits on Breadboards

Provide power to the power rails

- Red – positive
- Black/blue – negative

Use wires to connect components together

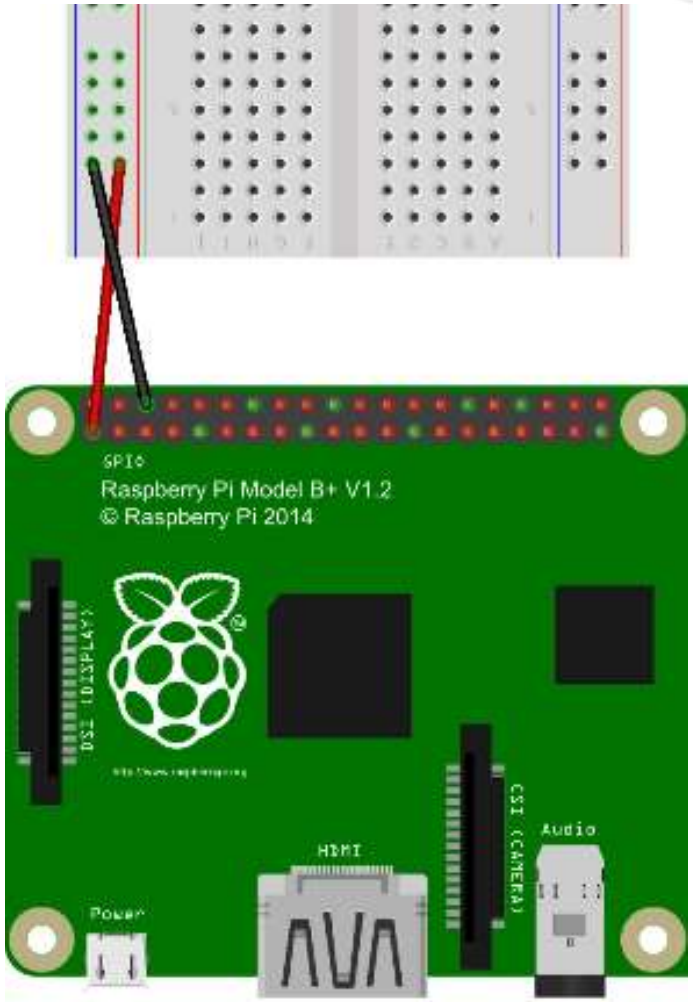


Intro to Hardware – Breadboards

Circuits with Raspberry Pis

Use pin labelled "+3.3V" (volts) for postive

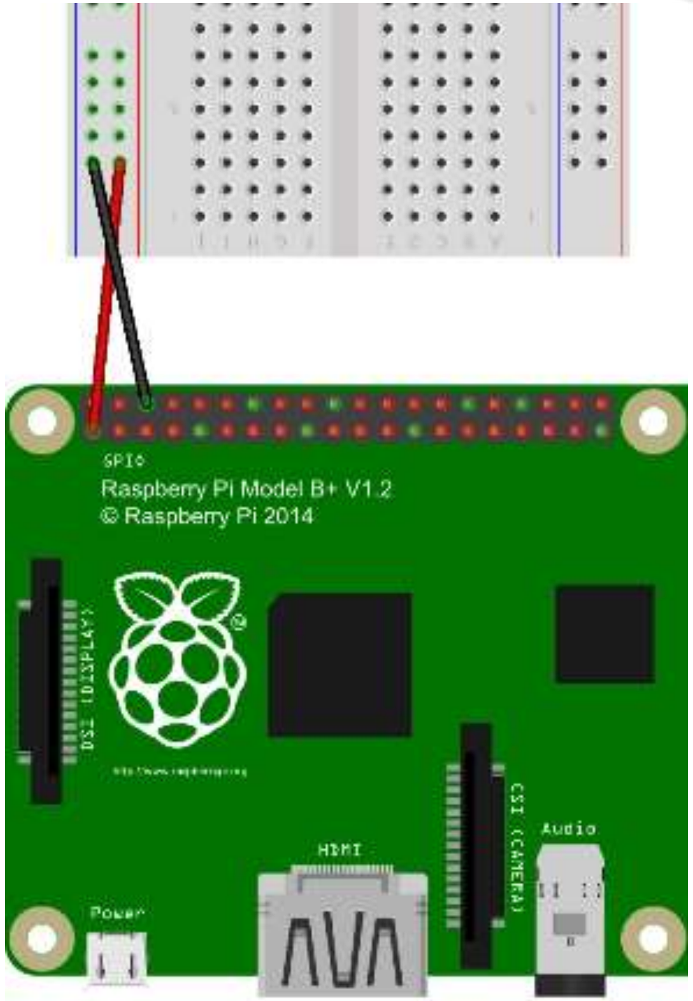
Use pin labelled "Ground" (GND) for negative



Intro to Hardware – Breadboards

Circuits with Raspberry Pis

Note: Arduinos use 5V, be careful not to confuse the two



GPIO

General Purpose
Input/Output

Or how our projects
“talk” to the Pi

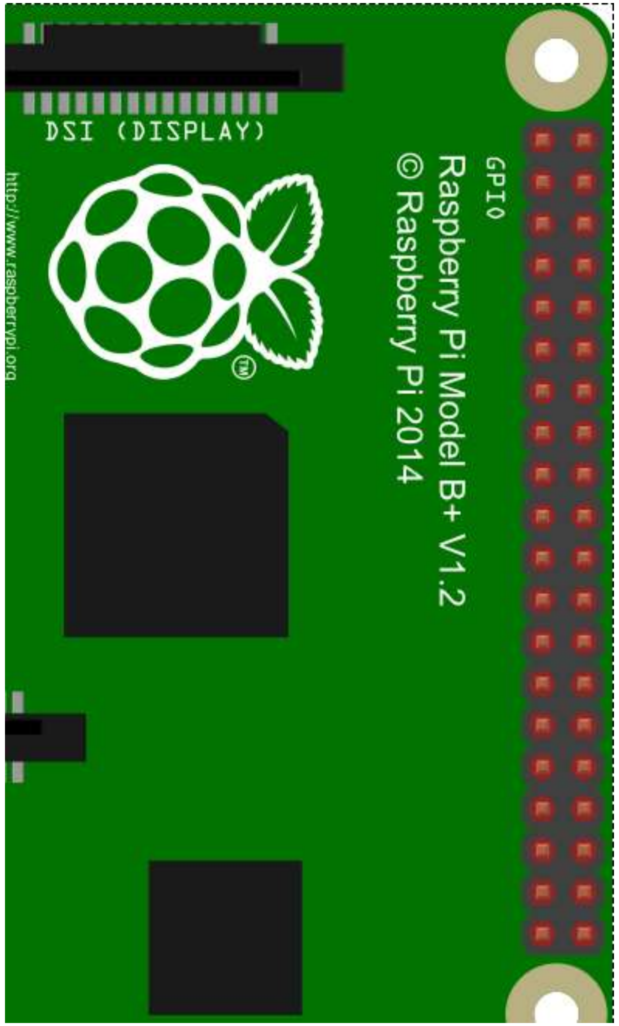
Intro to Hardware – GPIO

Raspberry Pi Pins

40 hardware pins

- 2 +3.3V
- 2 +5V
- 8 Ground
- 20 GPIO
- 2 Miscellaneous

Unfortunately, not labelled ON the Pi



Intro to Hardware – GPIO

Raspberry Pi Pins

Of course, they're not in any distinguishable order

(Pin diagram available at talk website)

Pin#	NAME	NAME	Pin#
01	3.3v DC Power	DC Power 5v	02
03	GPIO02 (SDA1 , I2C)	DC Power 5v	04
05	GPIO03 (SCL1 , I2C)	Ground	06
07	GPIO04 (GPIO_GCLK)	(TXD0) GPIO14	08
09	Ground	(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)	(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)	Ground	14
15	GPIO22 (GPIO_GEN3)	(GPIO_GEN4) GPIO23	16
17	3.3v DC Power	(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)	Ground	20
21	GPIO09 (SPI_MISO)	(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)	(SPI_CE0_N) GPIO08	24
25	Ground	(SPI_CE1_N) GPIO07	26
27	ID_SD (I2C ID EEPROM)	(I2C ID EEPROM) ID_SC	28
29	GPIO05	Ground	30
31	GPIO06	GPIO12	32
33	GPIO13	Ground	34
35	GPIO19	GPIO16	36
37	GPIO26	GPIO20	38
39	Ground	GPIO21	40

3.

Intro to Programming Hardware

The part you’re *really* here for

Intro to Programming Hardware

On the Raspberry Pi, Python is built-in with GPIO libraries installed

Other languages available too

Intro to Programming Hardware – Setup Commands

Pin Mode

First, pick your mode

Board – physical pin #
BCM – GPIO pin #

Pin#	NAME	NAME	Pin#
01	3.3v DC Power	DC Power 5v	02
03	GPIO02 (SDA1 , I2C)	DC Power 5v	04
05	GPIO03 (SCL1 , I2C)	Ground	06
07	GPIO04 (GPIO_GCLK)	(TXD0) GPIO14	08
09	Ground	(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)	(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)	Ground	14
15	GPIO22 (GPIO_GEN3)	(GPIO_GEN4) GPIO23	16
17	3.3v DC Power	(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)	Ground	20
21	GPIO09 (SPI_MISO)	(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)	(SPI_CE0_N) GPIO08	24
25	Ground	(SPI_CE1_N) GPIO07	26
27	ID_SD (I2C ID EEPROM)	(I2C ID EEPROM) ID_SC	28
29	GPIO05	Ground	30
31	GPIO06	GPIO12	32
33	GPIO13	Ground	34
35	GPIO19	GPIO16	36
37	GPIO26	GPIO20	38
39	Ground	GPIO21	40

Intro to Programming Hardware – Setup Commands

Next, import the GPIO library:

```
import Rpi.GPIO as GPIO
```

Then, set the mode (pick one):

```
GPIO.setmode(GPIO.BOARD)
GPIO.setmode(GPIO.BCM)
```

Intro to Programming Hardware – Setup Commands

Pins used should be set to be input or output:

```
GPIO.setup(3, GPIO.OUT)
GPIO.setup(4, GPIO.IN)
```

You can change these later, not but setting them at all causes problems

Intro to Programming Hardware – Output Commands

To write out to the output pins:
(pin number, value)

```
GPIO.output(3, True)
GPIO.output(10, False)
GPIO.output(6, GPIO.HIGH)
GPIO.output(22, GPIO.LOW)
```

True and HIGH are "on" (3.3 V)
False and LOW are "off" (0 V)

Intro to Programming Hardware – Input Commands

To read from the input pins:
(pin number)

```
x = GPIO.input(4)
sensor.setValue(GPIO.input(15))
```

Values will be True/False or 1/0

(Note: Arudinos have analog value inputs. Raspberry Pis to not.)

Intro to Programming Hardware – Cleanup Commands

Finally, when your program ends, you should "clean up" the pins:

```
GPIO.cleanup()
```

This stops reading inputs and stops sending outputs.

Setup recap:

- Import `Rpi.GPIO` library
- Set pin mode (`GPIO.BOARD` or `GPIO.BCM`)
- Set up input/output pins

Runtime recap:

- Output a value to a pin number
- Save input value from a pin to a variable

4.

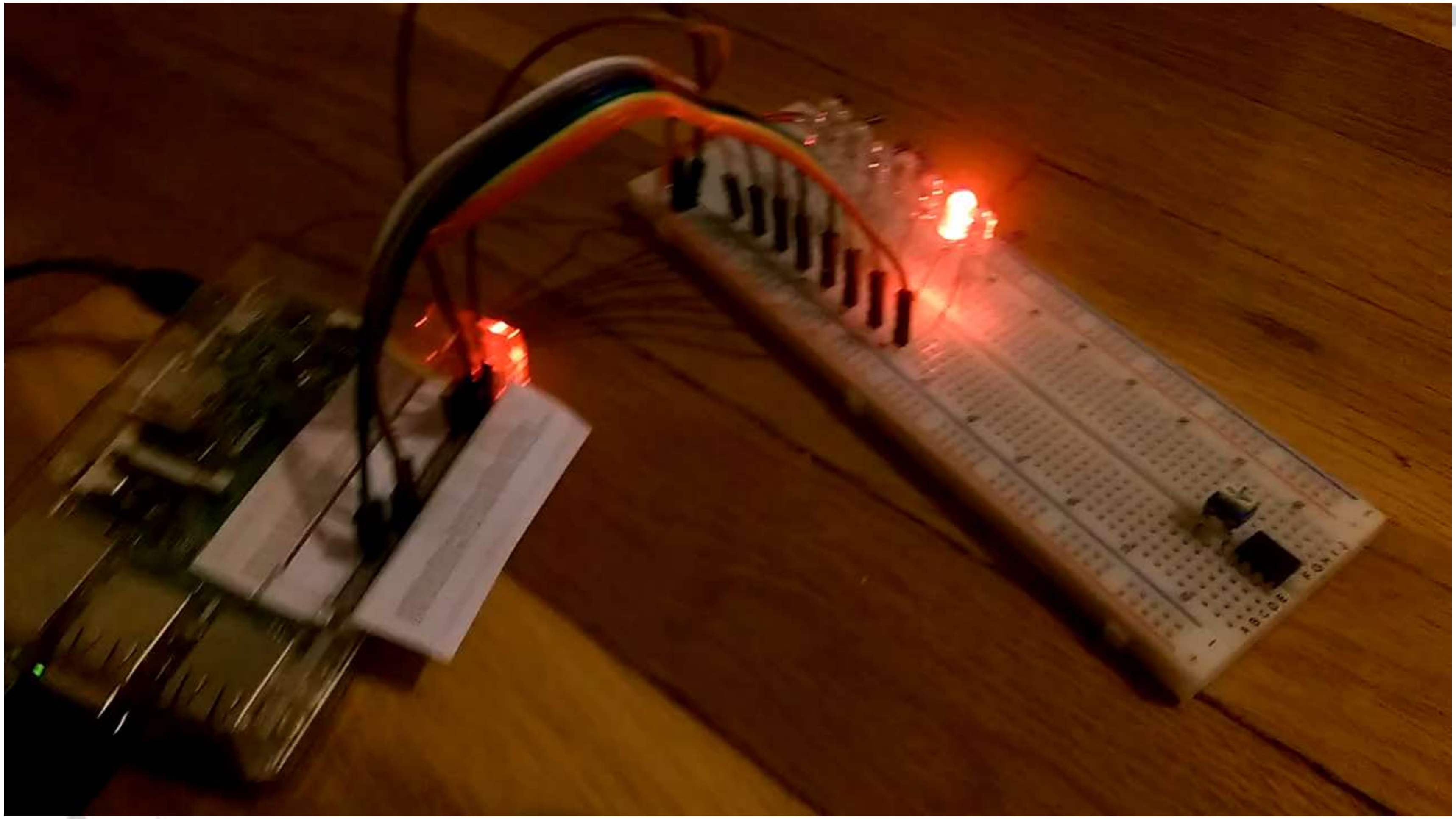
Project 1 - LEDs

The “Hello World” of electronics

Project 1 - LEDs

- 8 LEDs that blink back and forth
- Battlestar Galactica cylon
- Knight Rider KITT car

Project 1 - LEDs



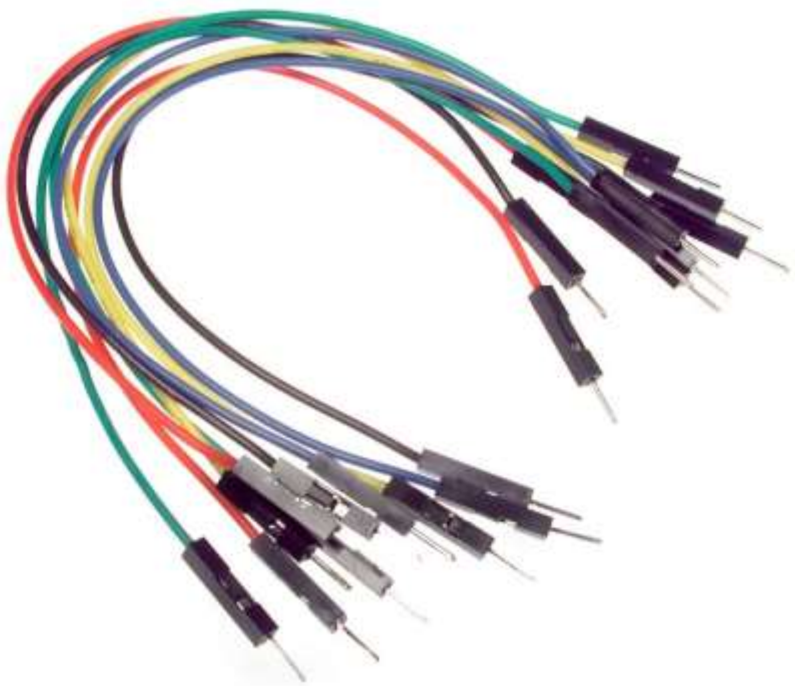
Project 1 - LEDs

Materials:

- Raspberry Pi and power supply
- 1 breadboard
- 8 LEDs
- 8 resistors
- 9 jumper wires

Jumper Wires

Pre-cut wires for prototyping with breadboards



LED

Light-emiting diode

Like a small light bulb

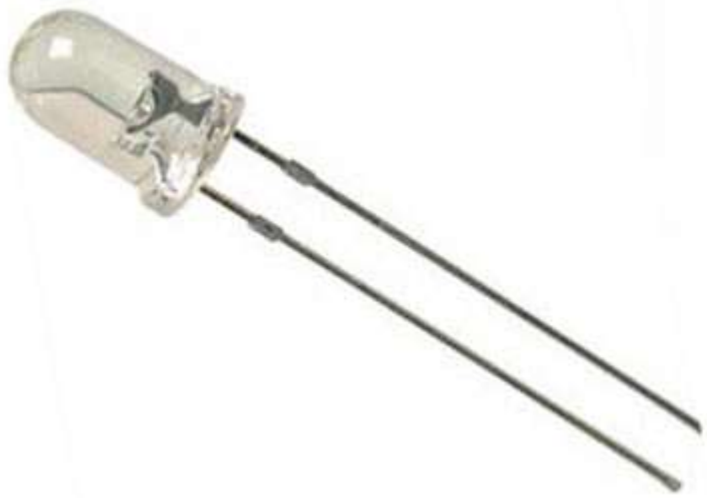
Project 1 - LEDs

LEDs

Diode – ensures electricity only goes one direction
(Wired wrong, it doesn't work)

Anode – longer wire (+)
Cathode – shorter wire (-)

ALWAYS use a resistor



Resistor

Slows down (or resists)
the flow of electricity

Project 1 - LEDs

Resistors

Can wire either direction

Band colors tell resistance value (in Ohms)

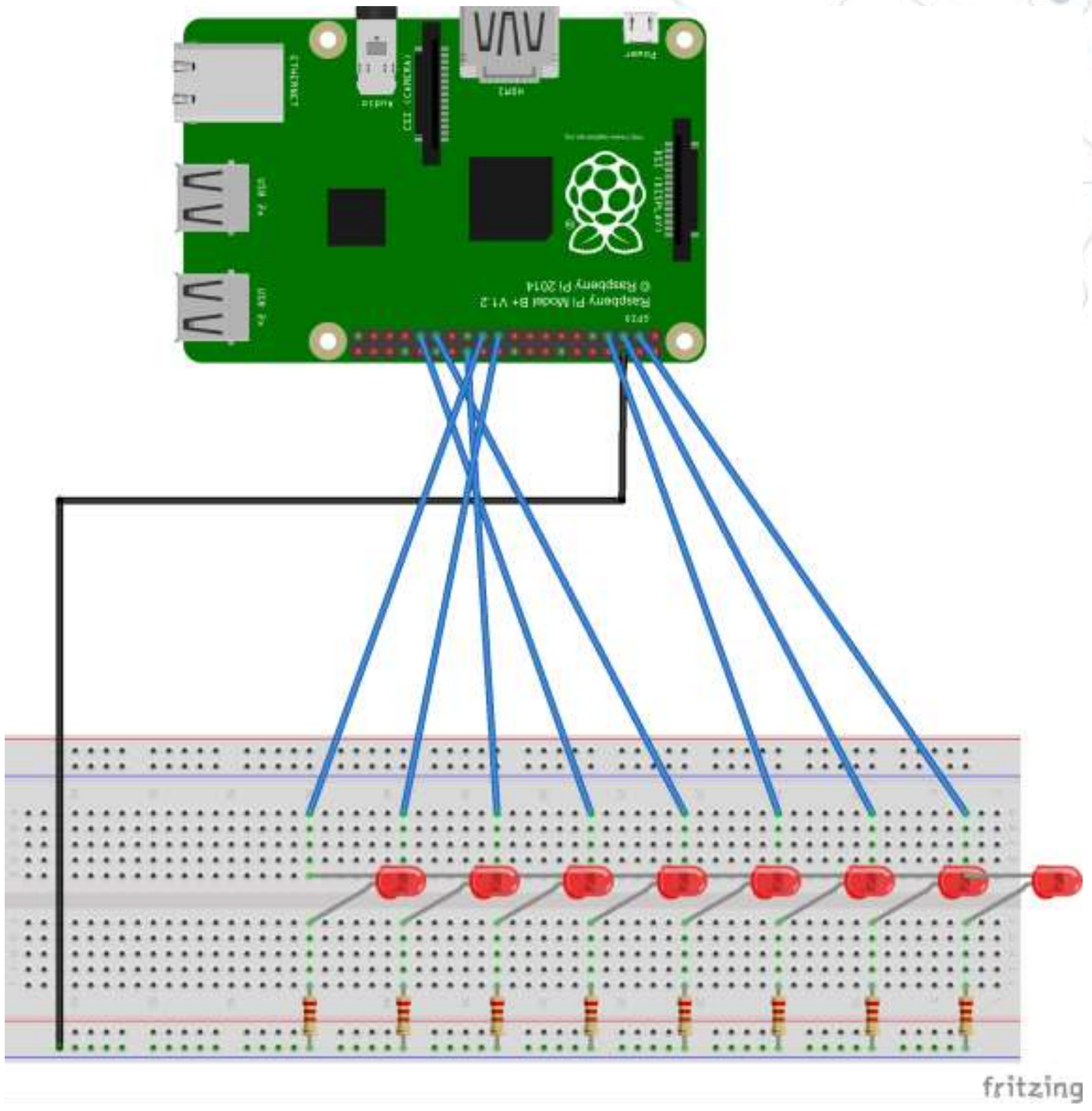
Without these, LEDs glow bright white, then element explodes

Project 1 - LEDs

Wiring

1) Wire Pi’s ground to ground rail (black wire)

2) Plug in LEDs across gap



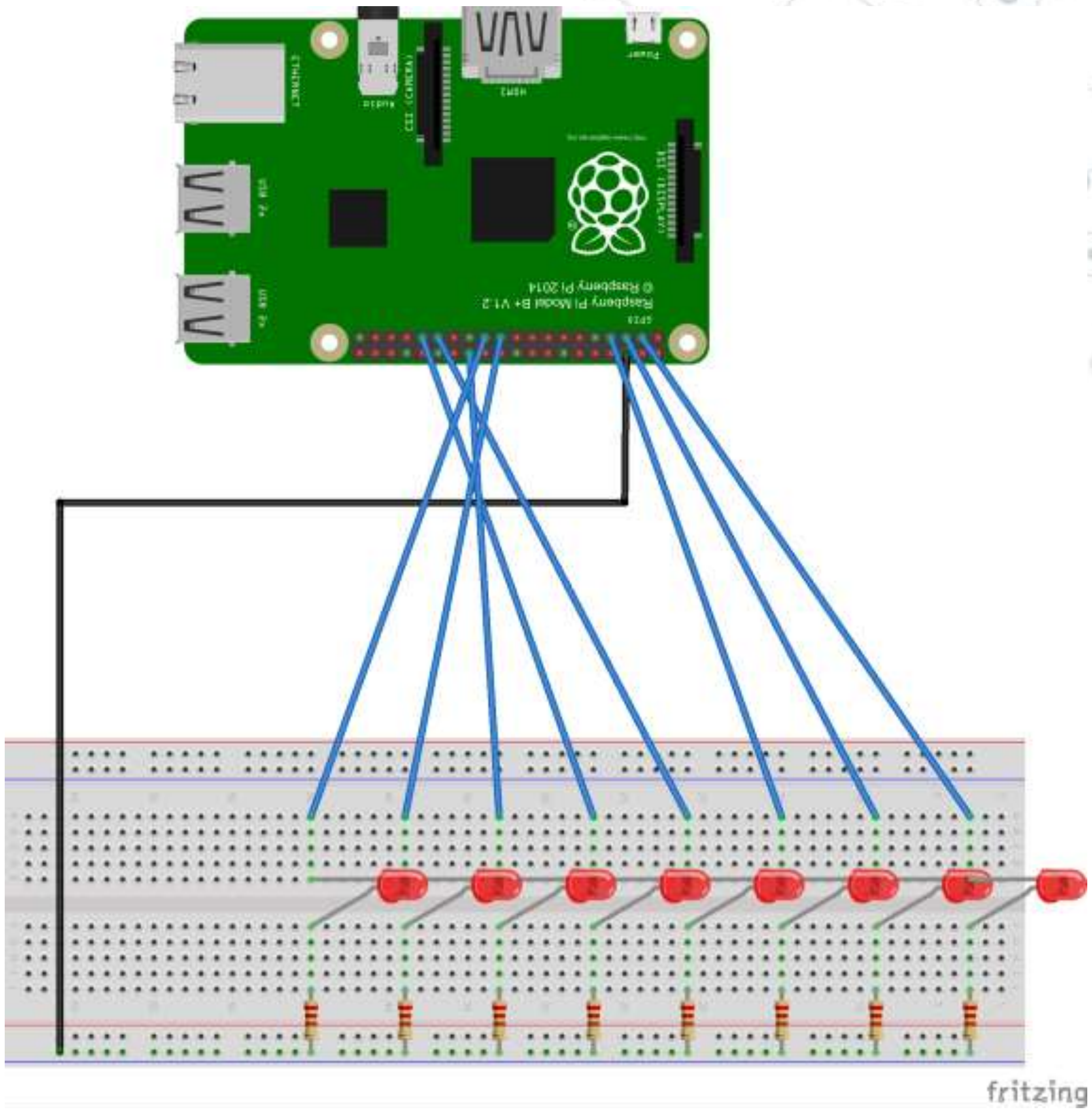
Project 1 - LEDs

Wiring

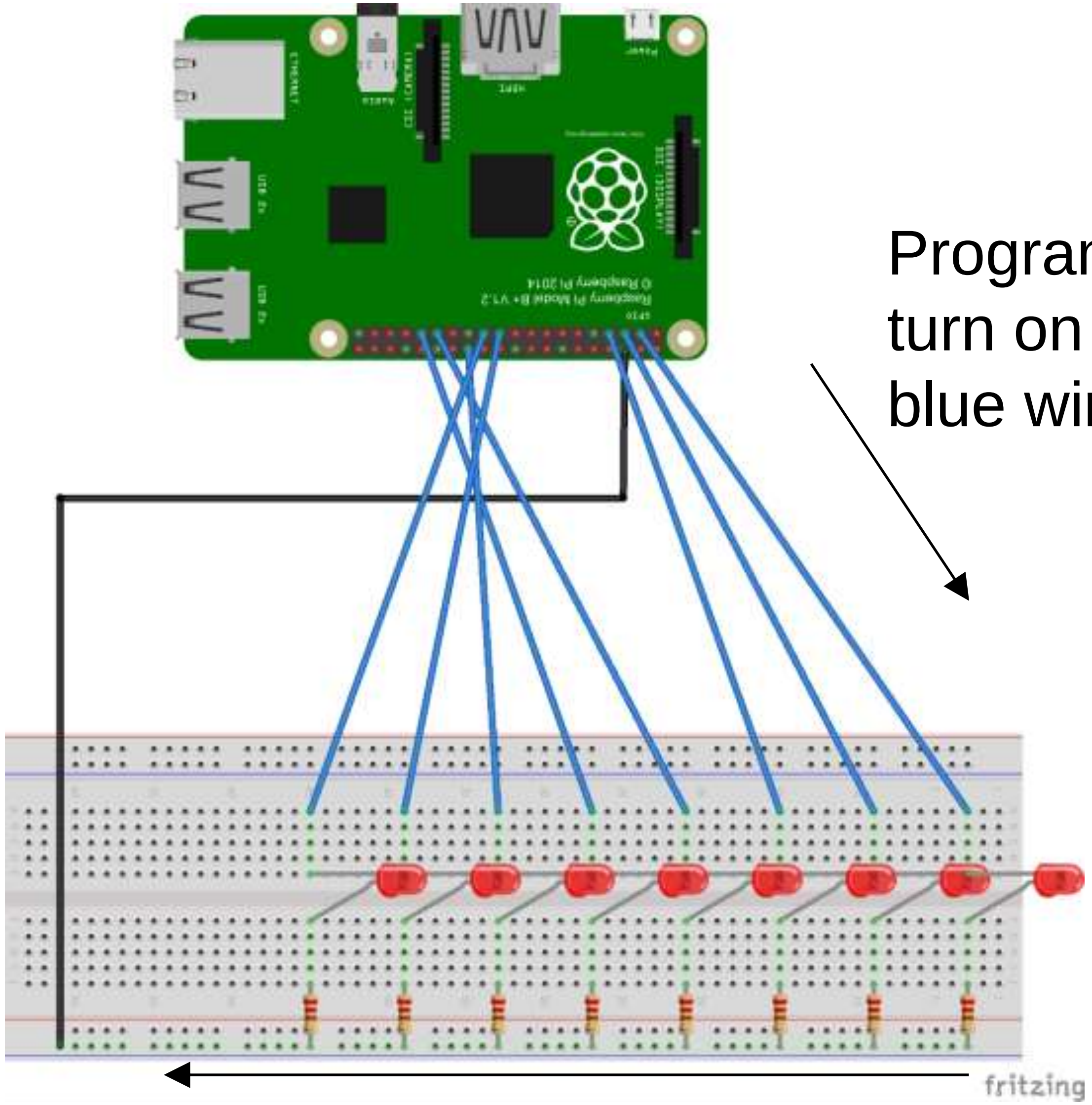
3) Connect resistors from LED cathodes (-) to ground

4) Connect anodes (+) to Pi GPIO pins

GPIO pins *become* power sources



Project 1 - LEDs



Programmatically turn on power to blue wires

Return to Ground

Through LED & Resistor

Through blue GND rail

Project 1 - LEDs

```python
# Import libraries
import RPi.GPIO as GPIO
import time

# Create array of GPIO pins
pins = [3, 5, 7, 29, 31, 26, 24, 21]
# Speed lights will blink (secs)
speed = .2    # Cylon-approved speed

```

Project 1 - LEDs

```python
# Set GPIO pins to board (physical pins) mode
GPIO.setmode(GPIO.BOARD)

# Set up all pins as output pins
for pin in pins:
    GPIO.setup(pin, GPIO.OUT)

```

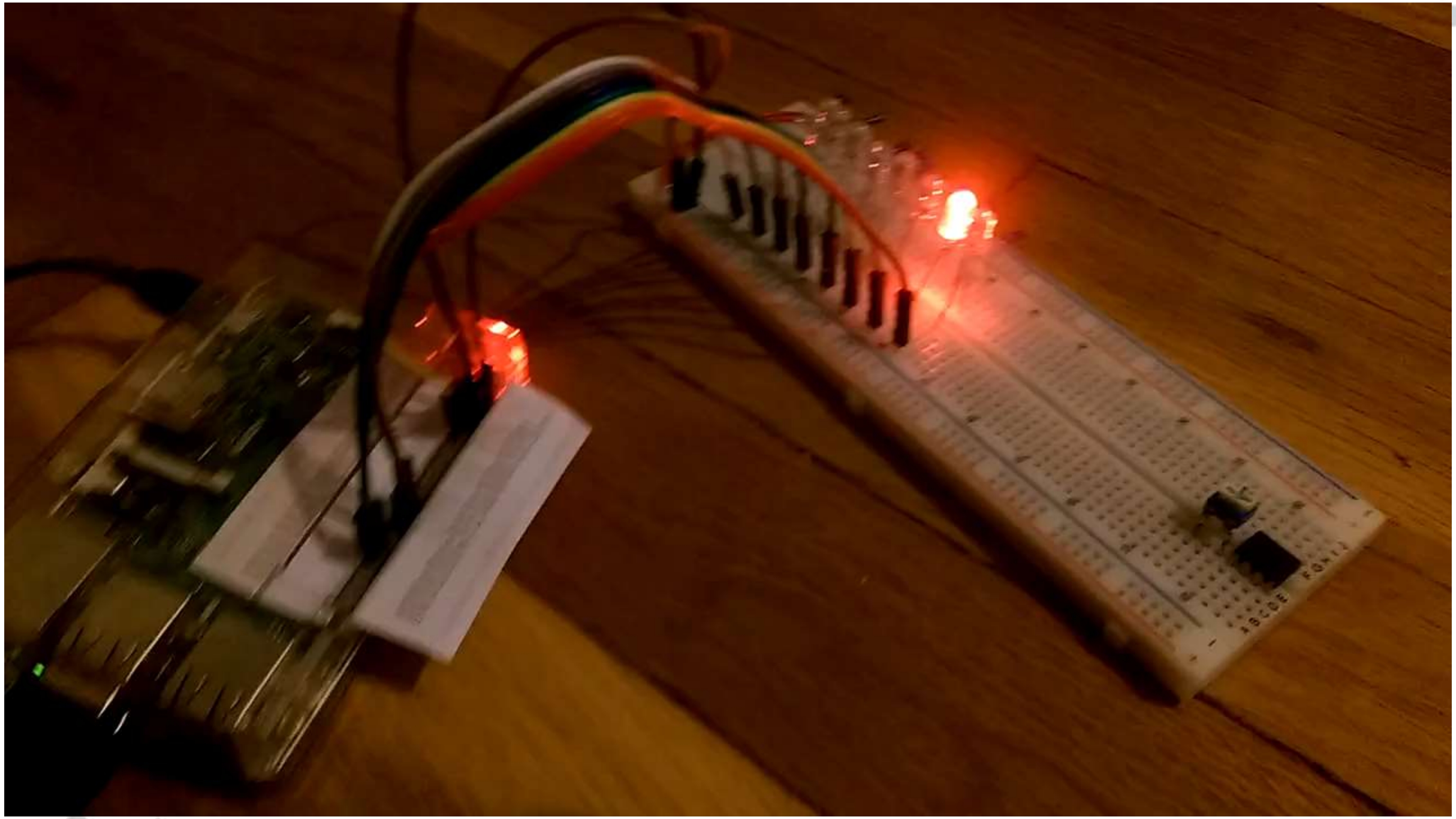
Project 1 - LEDs

```
while True:
   for i in range(len(pins)):
      GPIO.output(pins[i], True)
      time.sleep(speed)
      GPIO.output(pins[i], False)

   for i in range(len(pins)-1, -1, -1):
      GPIO.output(pins[i], True)
      time.sleep(speed)
      GPIO.output(pins[i], False)
```

Project 1 - LEDs

```
while True:
    for i in range(len(pins)-1):
        GPIO.output(pins[i], True)
        time.sleep(speed)
        GPIO.output(pins[i], False)

    for i in range(len(pins)-1, 0, -1):
        GPIO.output(pins[i], True)
        time.sleep(speed)
        GPIO.output(pins[i], False)
```

Project 1 - LEDs



5.

Project 2 – Sensors

Teaching computers to see.
Or something.

Ultrasonic Distance Sensor

Looks like robot, but measures distance with timed sound waves

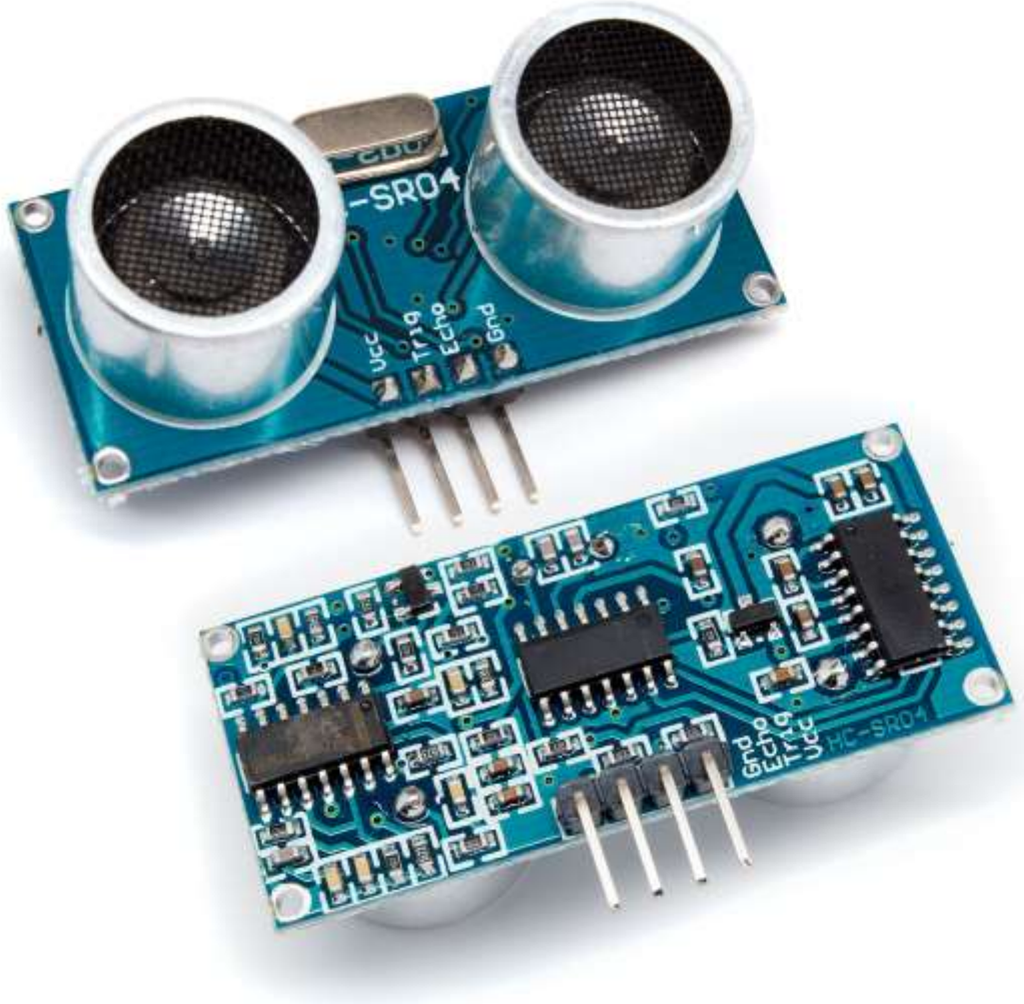
Project 2 - Sensors

Ultrasonic Distance Sensor

Sends ultrasonic (high-pitched 40KHz) sound wave

Detects time it takes for sound to bounce back

Time tells it the distance



Project 2 - Sensors

Ultrasonic Distance Sensor

4 pins:

- VCC (power)
- Trig (trigger)
- Echo
- GND (ground)

One side transmits, one receives

Project 2 - Sensors

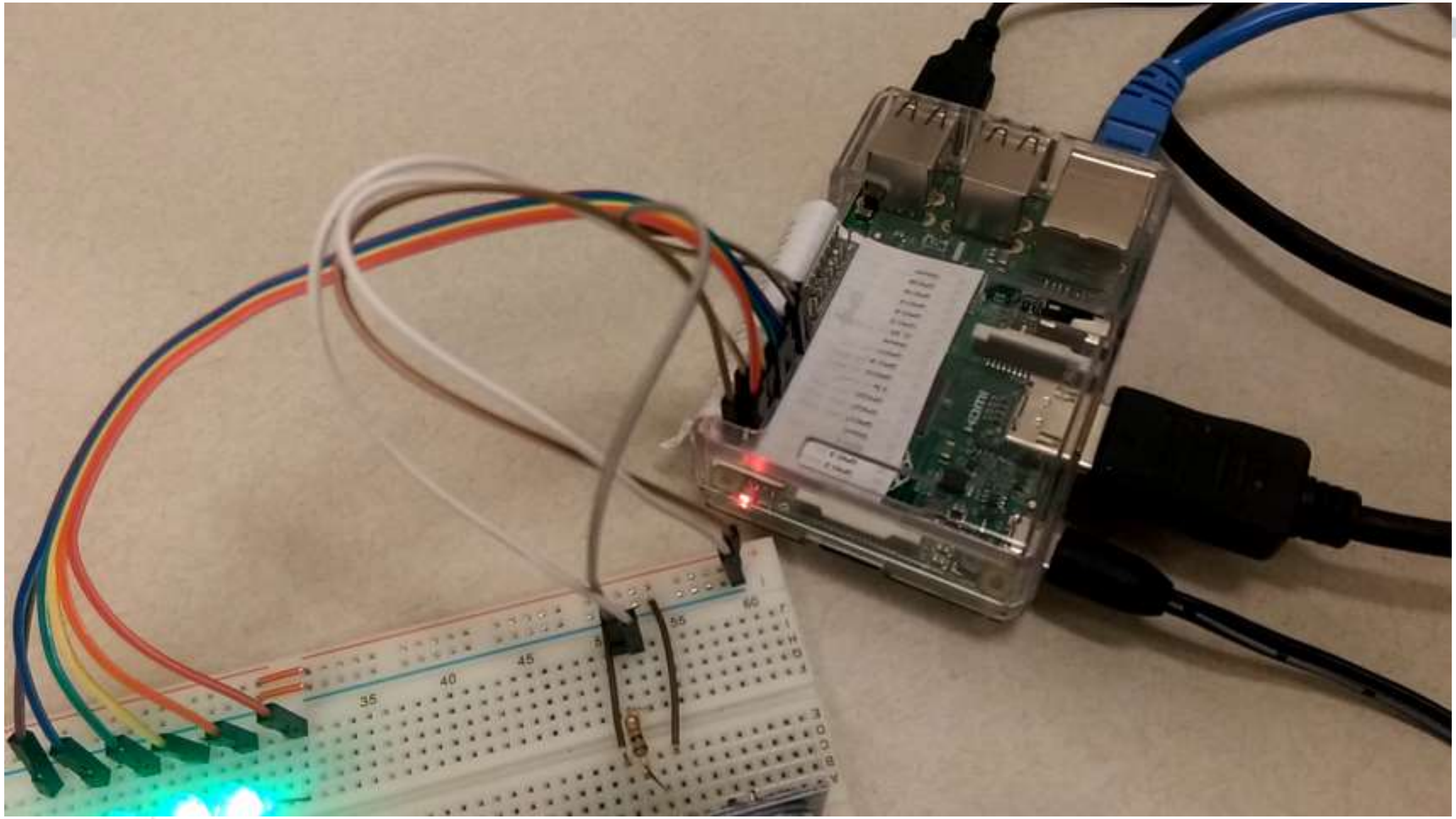
Ultrasonic Distance Sensor

Note: this works on 5V!

Don't mix 3.3V and 5V together! (5V is bad for GPIO pins)

Use 1K resistor on Echo pin

Project 2 - Sensors



Project 2 - Sensors

Materials:

- Raspberry Pi and power supply
- 1 breadboard
- 1 ultrasonic sensor (HC-SR04)
- 6 LEDs
- 6 resistors for LEDs
- 1 1K resistor
- Jumper wires

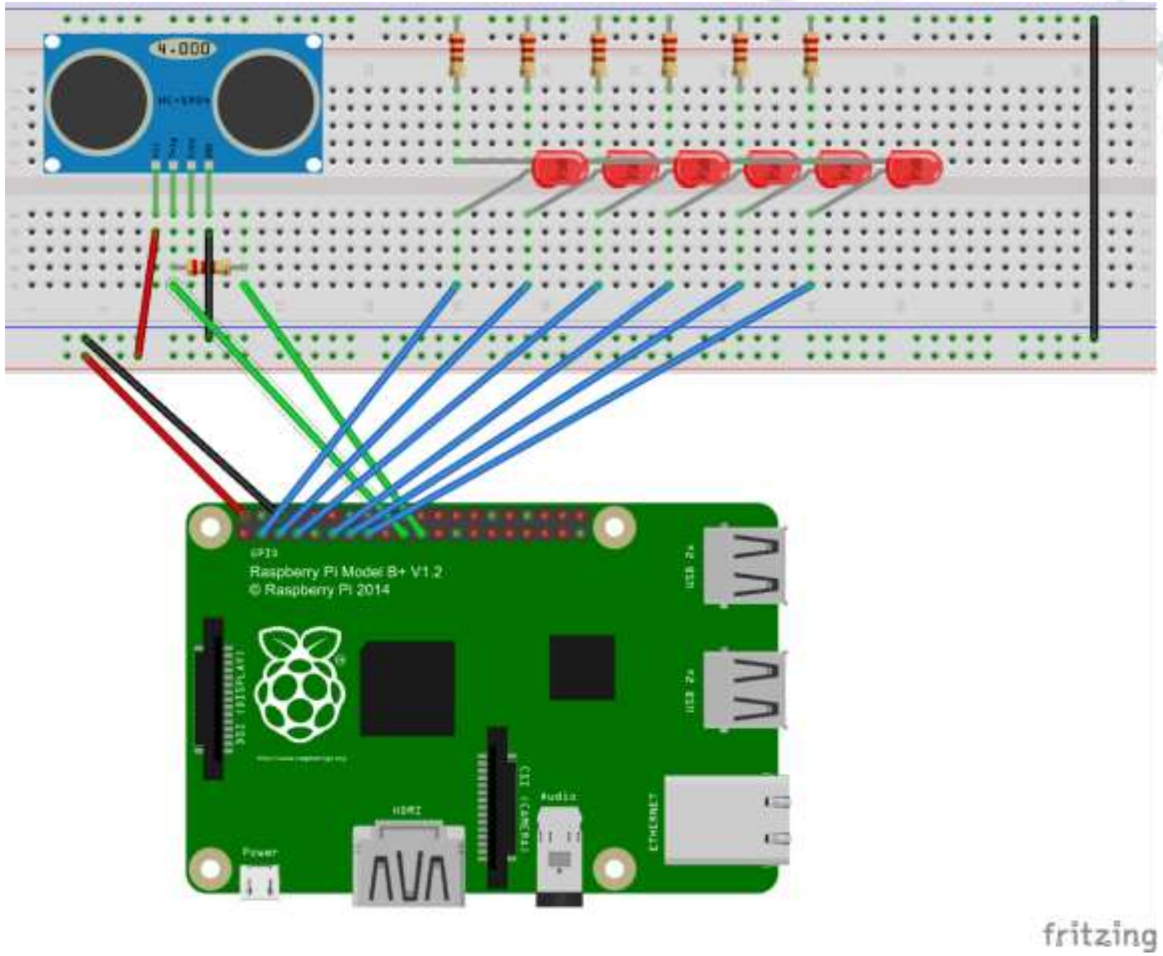
Project 2 - Sensors

Wiring

1) Wire Pi's ground to ground rail (black)

2) Wire Pi's +5V to power rail (red)

3) Wire both ground rails together (black wire)

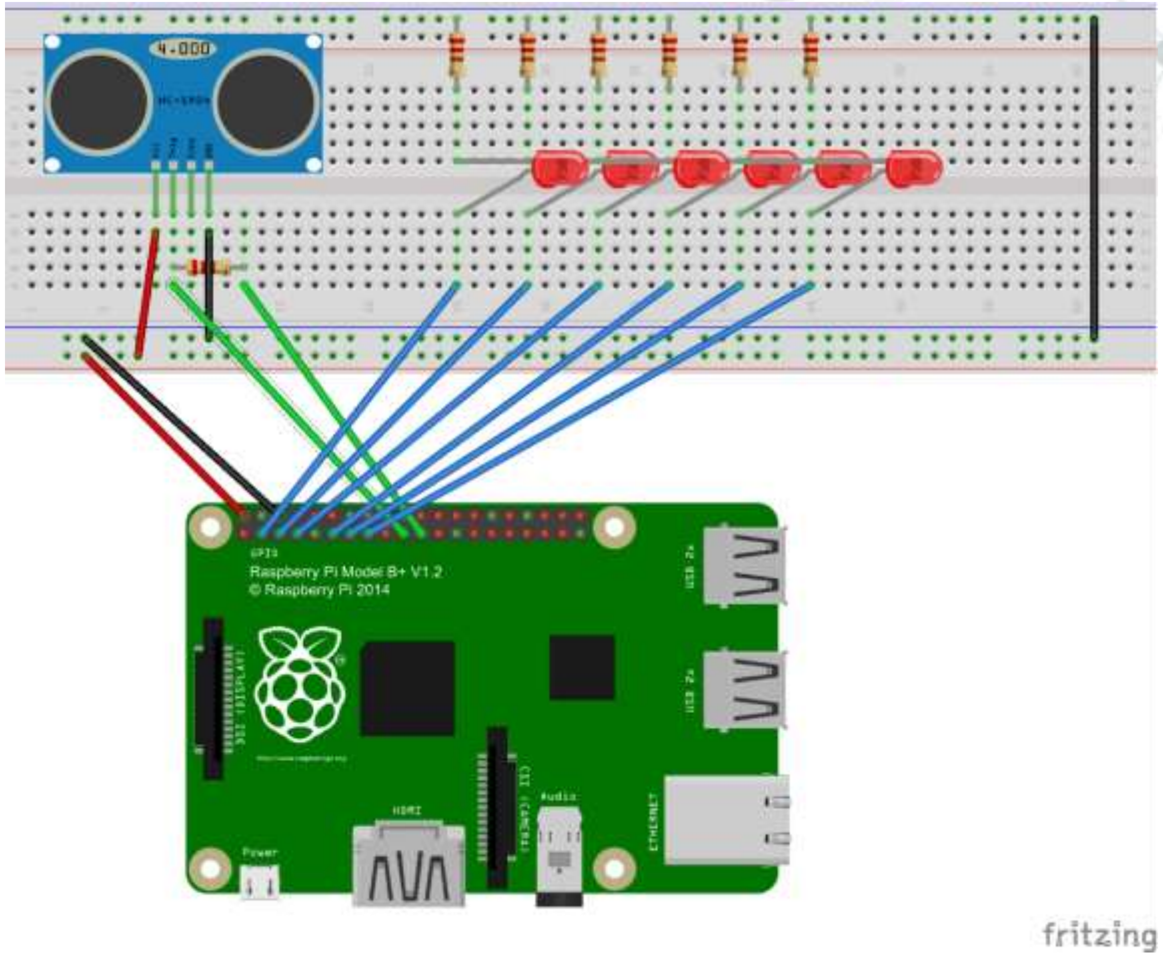


Project 2 - Sensors

Wiring

4) Wire LEDs and resistors to GPIO pins like in Project 1

5) Wire Trigger pin to GPIO

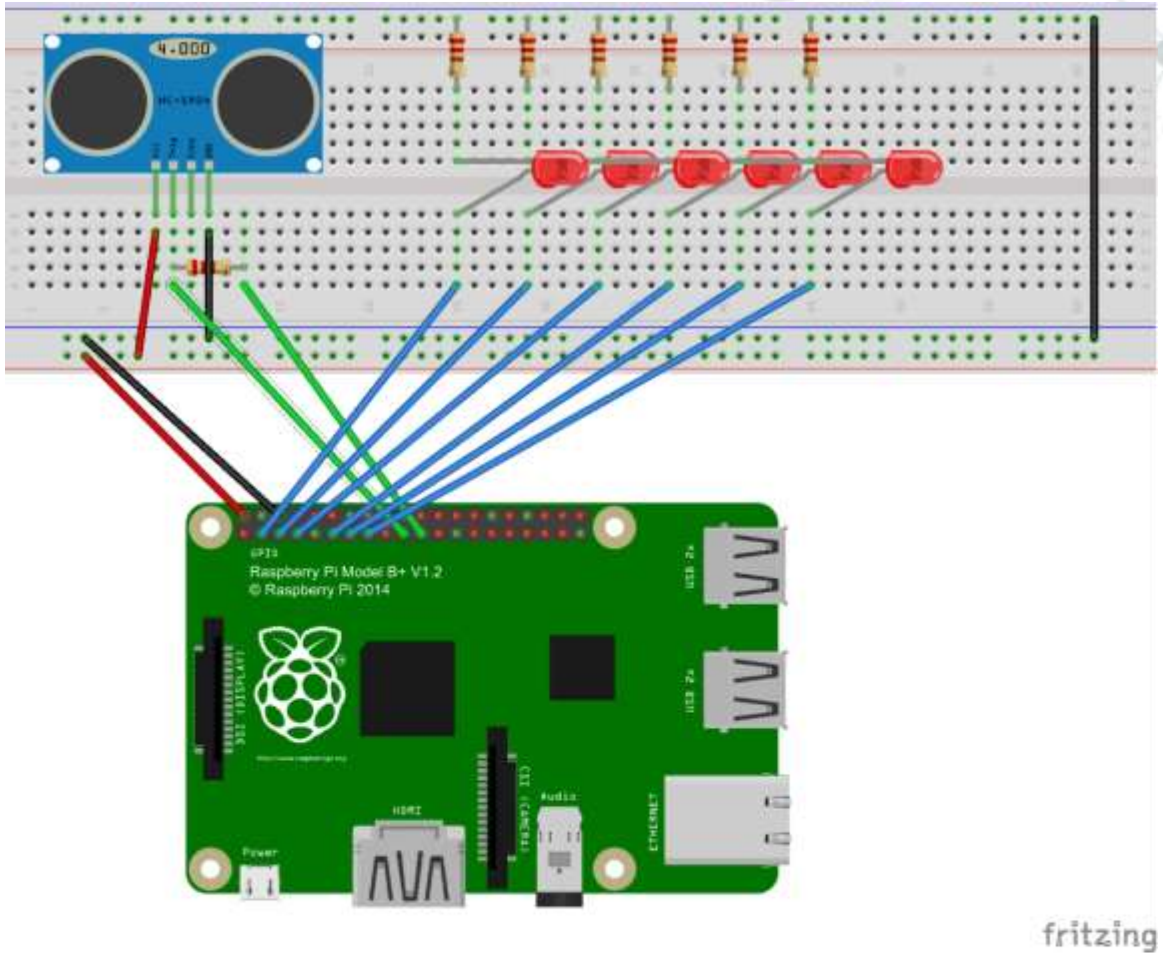


Project 2 - Sensors

Wiring

6) Wire Echo pin to 1K resistor

7) Wire 1K resistor to GPIO



Project 2 - Sensors

```
import time
import RPi.GPIO as GPIO

# Set up the pins
uTrig = 19
uEcho = 21
leds = [3, 5, 7, 11, 13, 15]

GPIO.setup(GPIO.BOARD)
for i in leds:
    GPIO.setup(i, GPIO.OUT)
    GPIO.output(i, GPIO.LOW)   # off
```

Project 2 - Sensors

```
def readDistance(trigger, echo):
    GPIO.setwarnings(False)
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(trigger, GPIO.OUT)
    GPIO.setup(echo ,GPIO.IN)
    GPIO.output(trigger, GPIO.LOW)

    time.sleep(0.3)
```

Project 2 - Sensors

```
GPIO.output(trigger, True)
time.sleep(0.00001)
GPIO.output(trigger, False)

signaloff = 0
while GPIO.input(echo) == 0:
    signaloff = time.time()

```

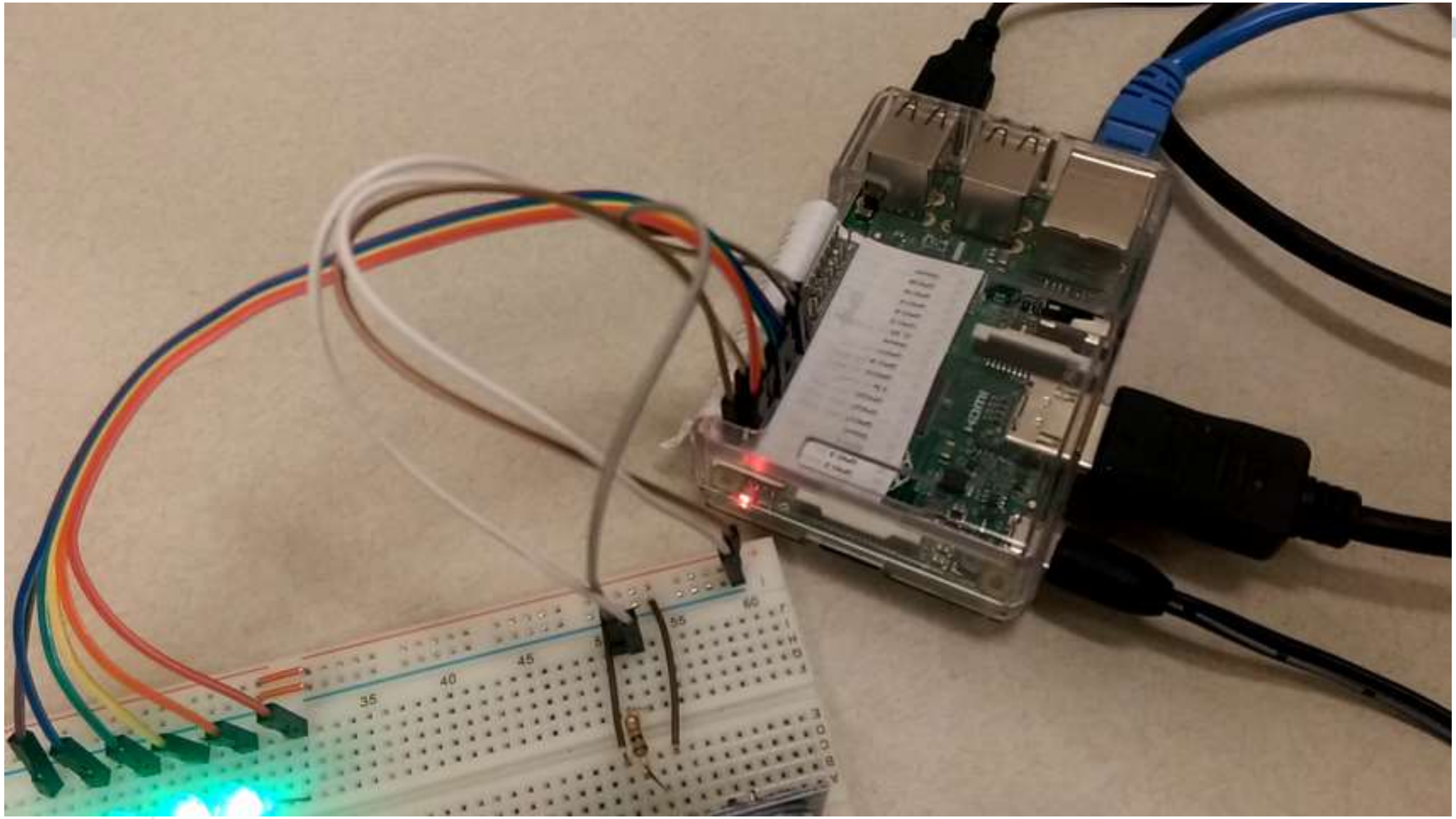
Project 2 - Sensors

```
signalon = 0
while GPIO.input(echo) == 1:
    signalon = time.time()

timepassed = signalon - signaloff

distance = timepassed * 17000

return distance

```

Project 2 - Sensors

```
# Main program
while True:
    # Grab a reading
    read = readDistance (uTrig, uEcho)

    for i in range(0, 6):
        print(i)
        if read > i * 3:    # Every 3 cm
            GPIO.output(leds[i], GPIO.HIGH)
        else:
            GPIO.output(leds[i], GPIO.LOW)

```

Project 2 - Sensors



6.

Project 3 – Push Buttons and LCD screens

Push it, push it real good

Push Button

It's a button you push

Project 3 – Push Buttons and LCD screens

Push Button

Connects two pins when pressed

Fits across the breadboard gap

Project 3 – Push Buttons and LCD screens

These two are connected

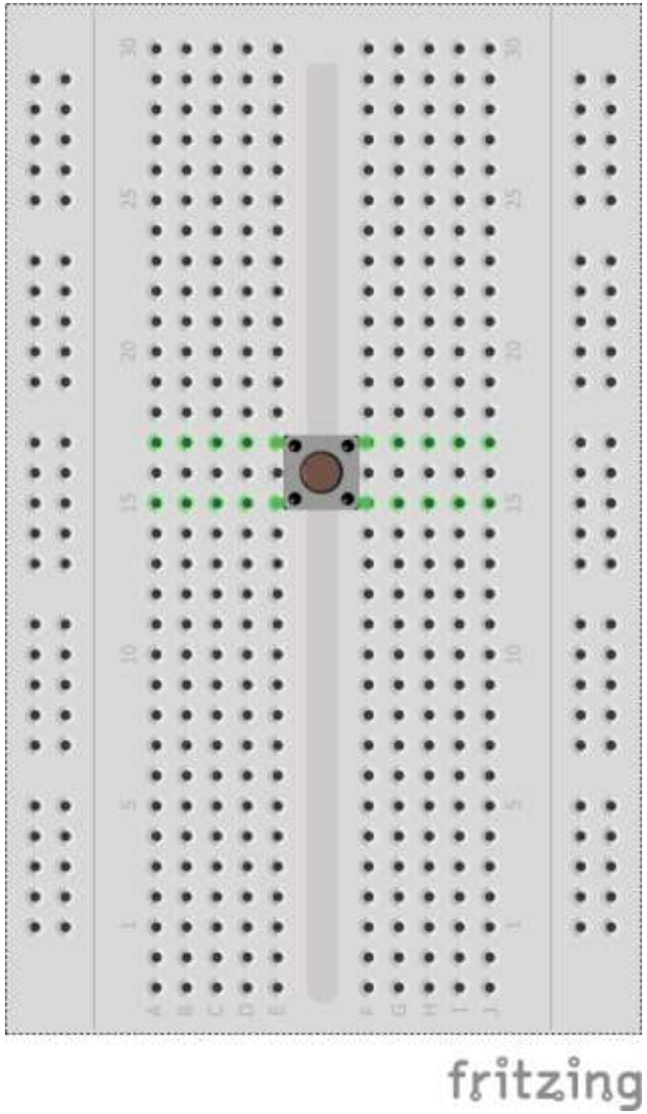
These two are connected

Pressing button connects all 4

Project 3 – Push Buttons and LCD screens

Push Button

The curved pins should "hug" the breadboard gap



LCD Screen

Screen that can display ASCII characters

Project 3 – Push Buttons and LCD screens

LCD Screen

In a variety of row/col configurations

In a variety of pin configurations, but I recommend the I2C type

I²C

Inter-Integrated Circuit,
or “I squared C”

Communication
protocol for electronics

Project 3 – Push Buttons and LCD screens

I^{2}C on the Raspberry Pi

It’s not enabled by default

Run sudo raspi-config

Go to Interfacing Options
Then I2C
Then enable

Then reboot

Project 3 – Push Buttons and LCD screens

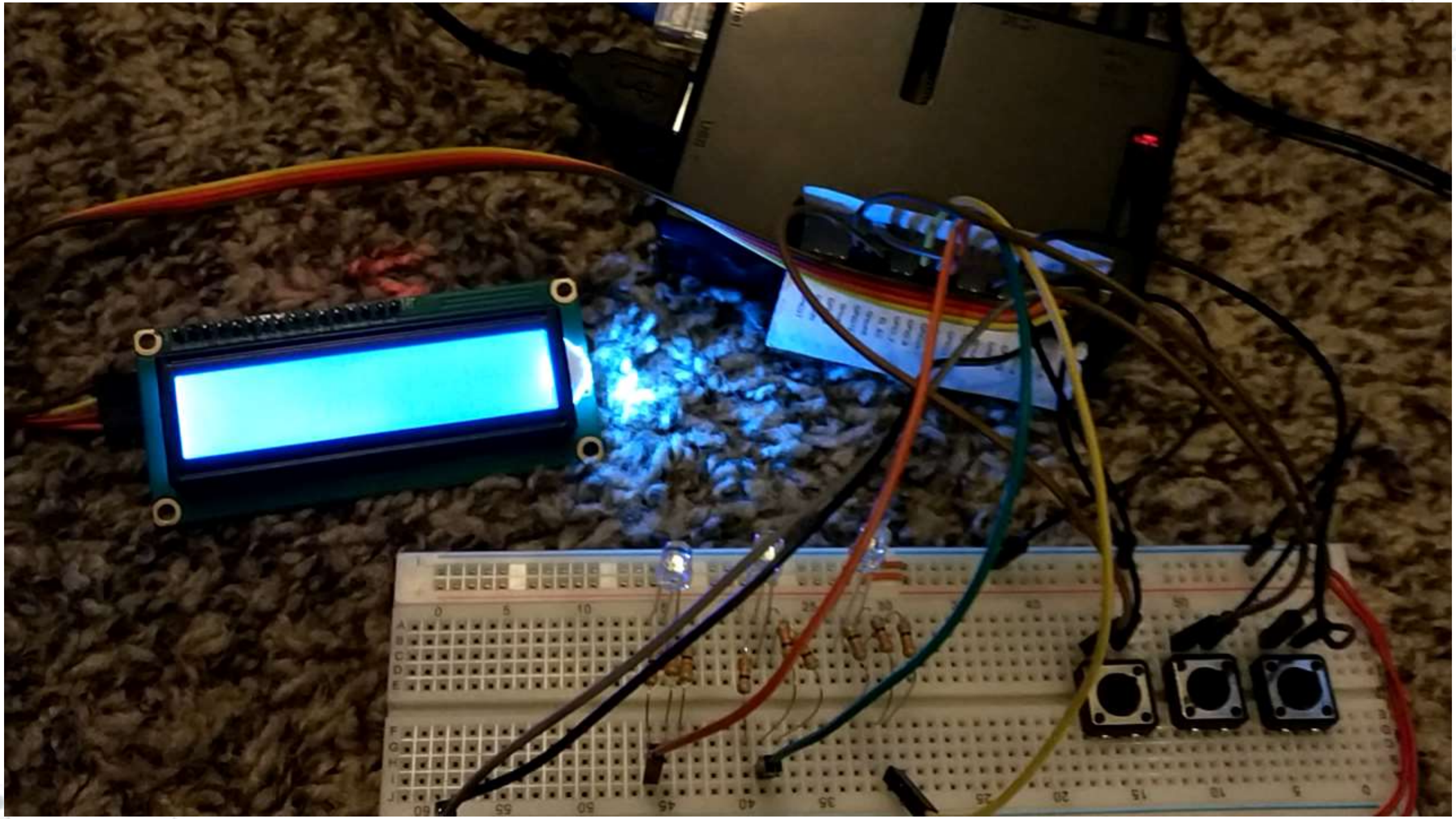
Wiring the LCD Screen

The 4 wires correspond to the 4 pins on the Pi:

Vcc – +5V
Gnd – Ground
SDA – SDA
SCL – SCL

(Yes, this you hook up to 5V)

Project 3 – Push Buttons and LCD screens



Project 3 – Push Buttons and LCD screens

Materials:

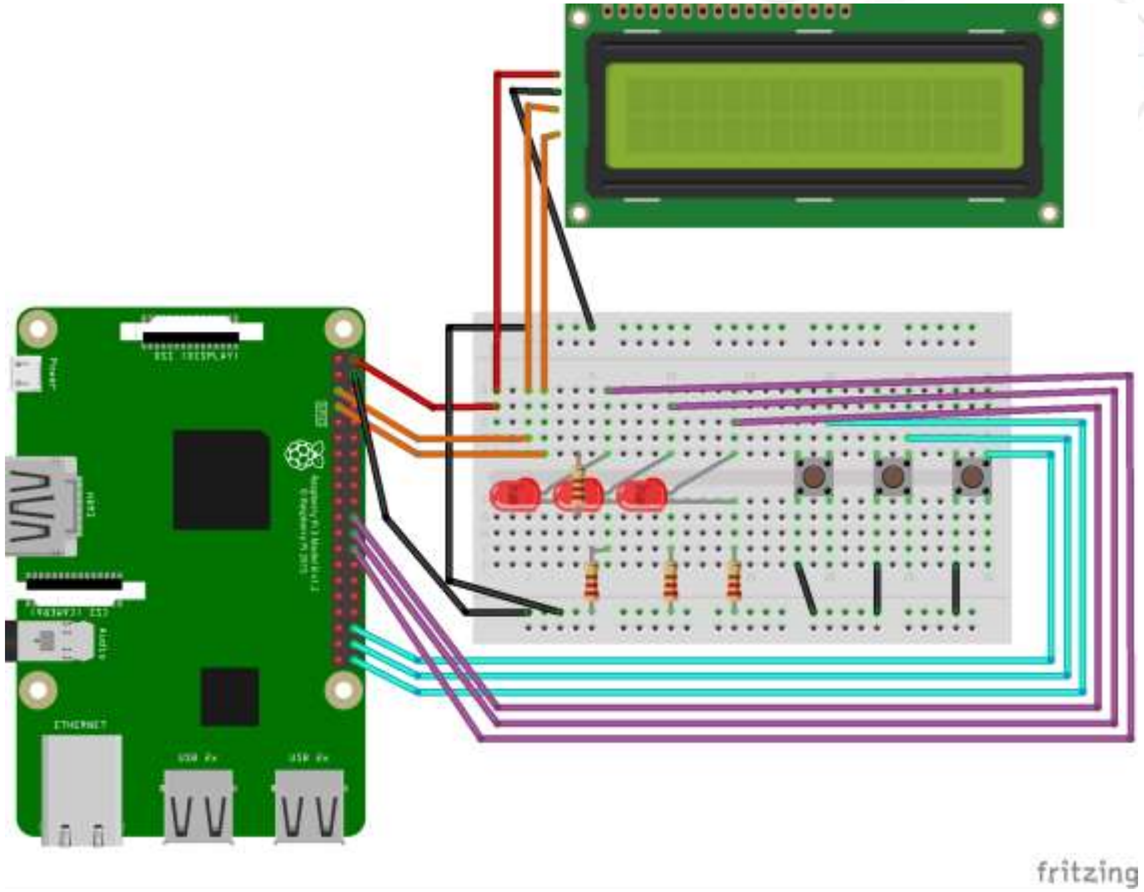
- Raspberry Pi and power supply
- 1 breadboard
- 3 push buttons
- 3 LEDs and resistors
- 1 LCD screen
- Jumper wires

Project 3 – Push Buttons and LCD screens

Wiring

1) Connect ground to black rail

2) Connect LEDs like in project 1 (GPIO->LED->resistor->ground) (purple wires)

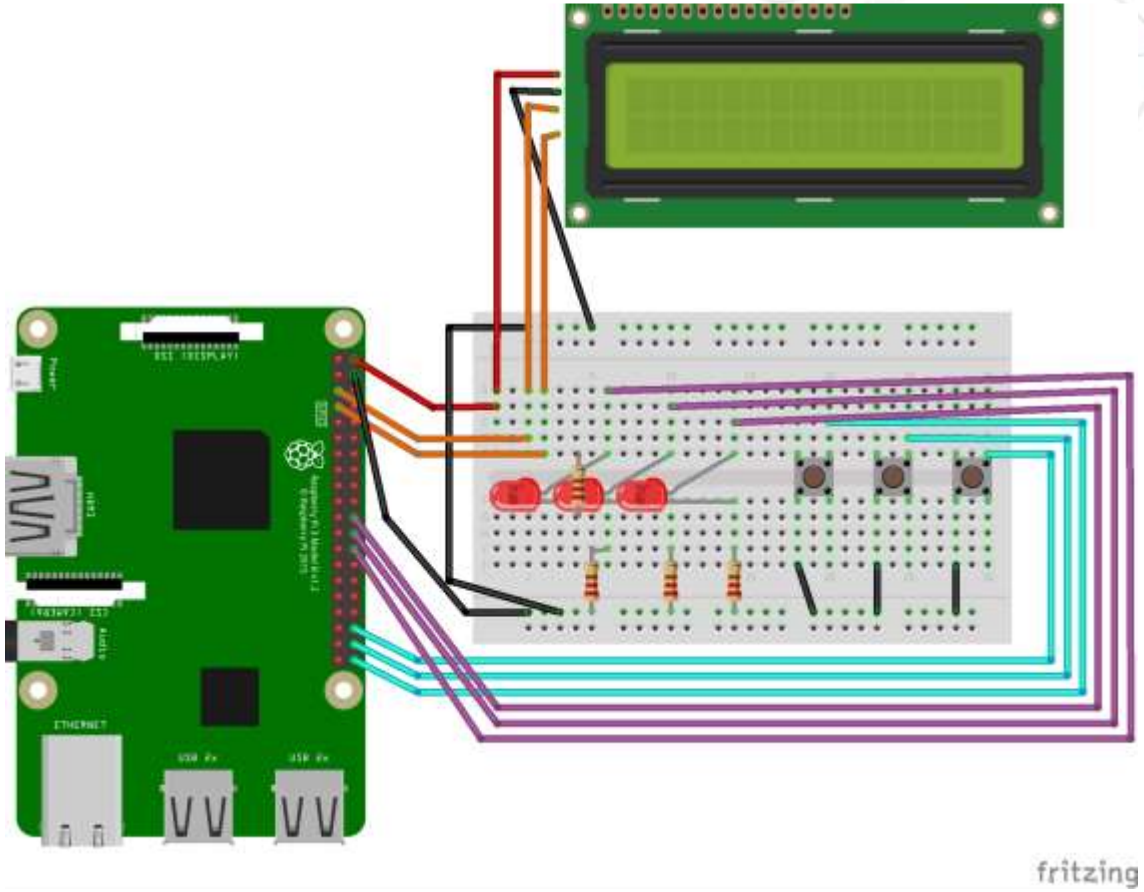


Project 3 – Push Buttons and LCD screens

Wiring

3) Connect one side of each button to Ground (black wires)

4) Connect other side to GPIOs (cyan wires)



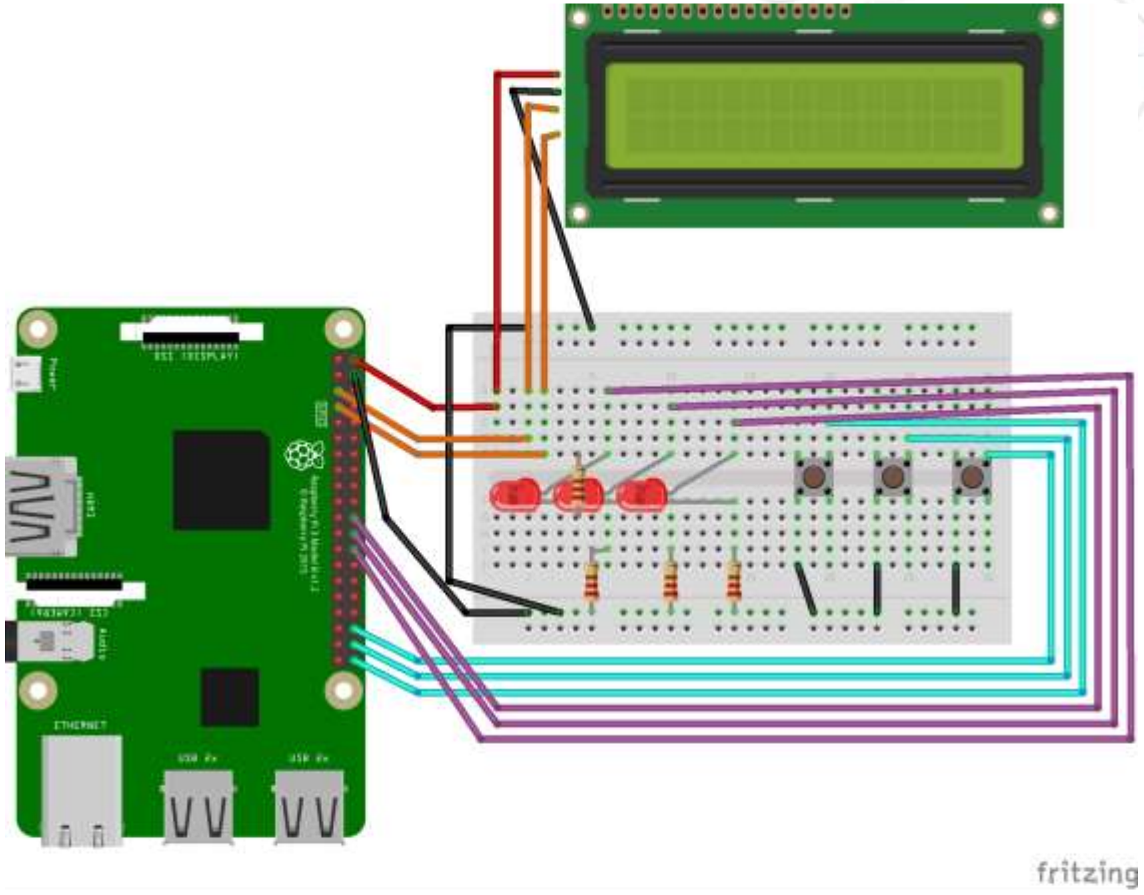
Project 3 – Push Buttons and LCD screens

Wiring

5) Connect LCD's Ground to Ground

6) Connect LCD's Vcc to +5V on pi

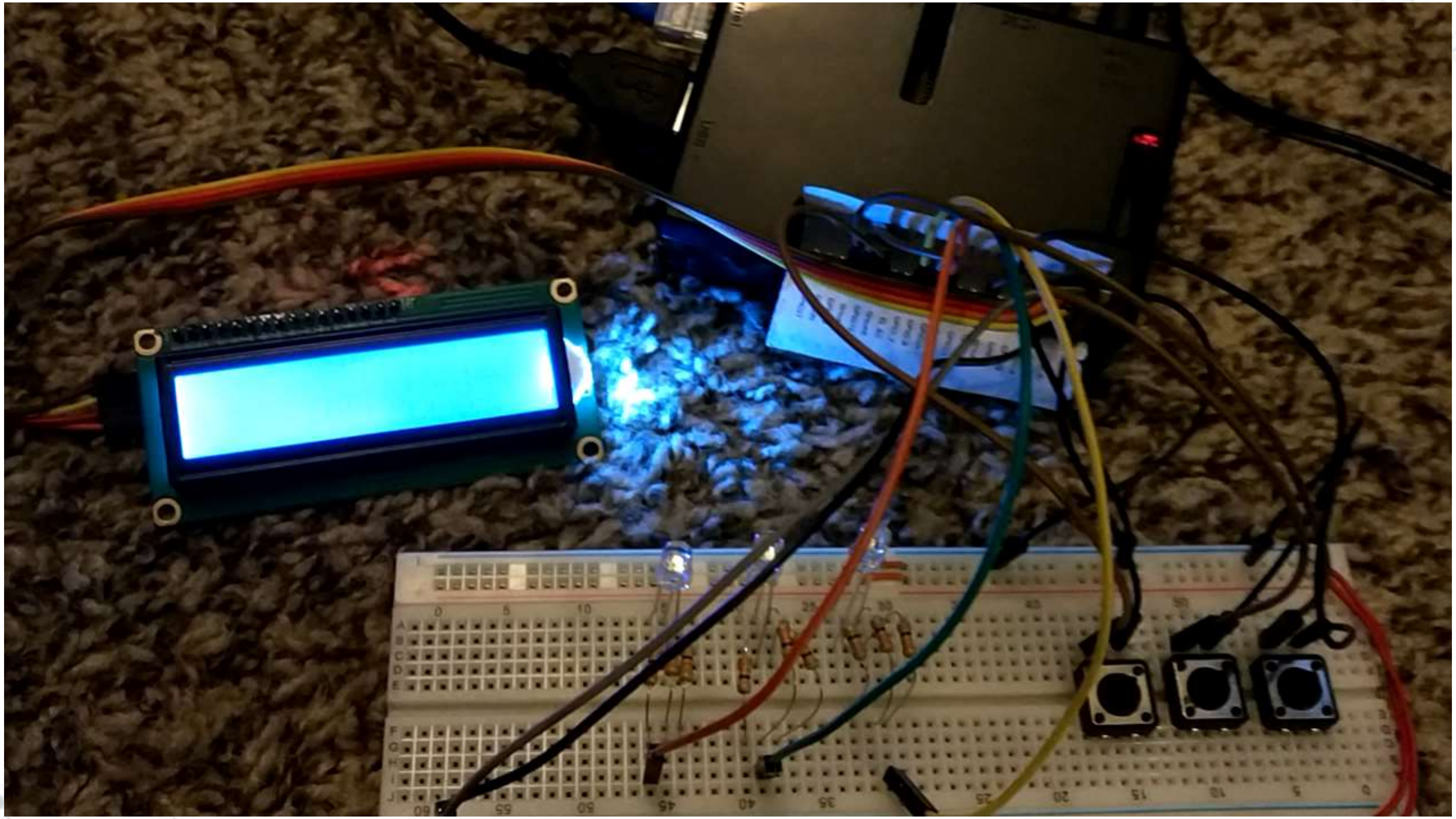
7) Connect SDA/SCL to Pi pins 3/5 (orange wires)



Project 3 – Push Buttons and LCD screens

(Open code in editor)

Project 3 – Push Buttons and LCD screens



7.

Future Project Inspiration

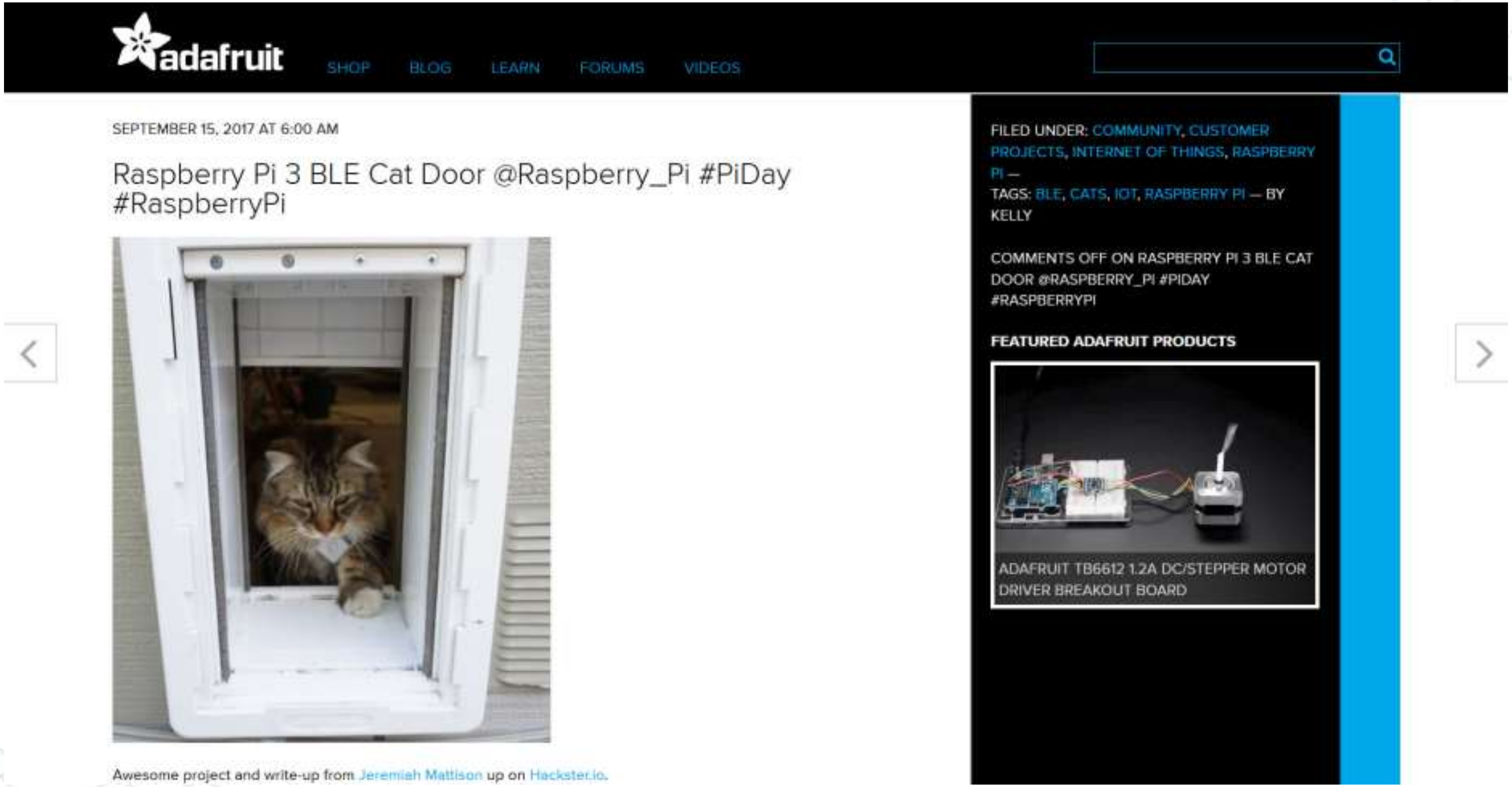
Time to flap those electronic wings and fly!

Future Project Inspiration

Assorted ideas I’ve heard of:

- Customized cookie maker machine
- Show tweets/emails on LCD screen
- AI-programmed remote control cars
- Robots… of any variety…
- Home automation (control heater/AC, control lights when in room, etc.)

Future Project Inspiration



Pi Bluetooth Low Energy Cat Door

https://blog.adafruit.com/2017/09/15/rpi-3-ble-cat-door-raspberry_pi-piday-raspberrypi/

Future Project Inspiration

Christmas Light Controller System

https://chivalrytimberz.wordpress.com/2012/12/03/pi-lights/

Future Project Inspiration

Pi-Powered CNC Oreo customizing machine

https://blog.adafruit.com/2014/03/14/peek-inside-a-pi-powered-cnc-oreo-customizing-machine-raspberry_pi-piday-raspberrypi/, start at 3:40

Future Project Inspiration

Where can I buy parts?

- Pi: MicroCenter, Element14, Adafruit, Pishop.us. (Beware, Amazon costs > $35)
- Power supply: MicroCenter, Amazon, $10-20
- Breadboard: Amazon, MicroCenter, $4-7
- Jumpers: Amazon, $3-7 for setsLEDs: Buy as kit (LEDs + resistors) on Amazon or electronic stores
- Sensors: Amazon has kits for $15-40 (like https://www.amazon.com/OSOYOO-Modules-Mega2560-Raspberry-Learning/dp/B00WQY2704/)

Future Project Inspiration

Final thoughts:

- Development projects always take longer than you think. More so with hardware.
- Hardware sometimes fails randomly. If software quits working, try testing hardware.
- You’re welcome to ask me questions!

8. Conclusion

It all must come to an end…

Conclusion

- Raspberry Pi is a PC, but can interface easily with electronics with GPIO pins
- You’ve seen a variety of sample projects and parts and how to code with them
- Hopefully you’re inspired to hack together your own projects

Thanks!

Slides:
sarahwithee.com/raspberrypi

Share your stories and questions!

Twitter: @geekygirlsarah
Email: hello@sarahwithee.com