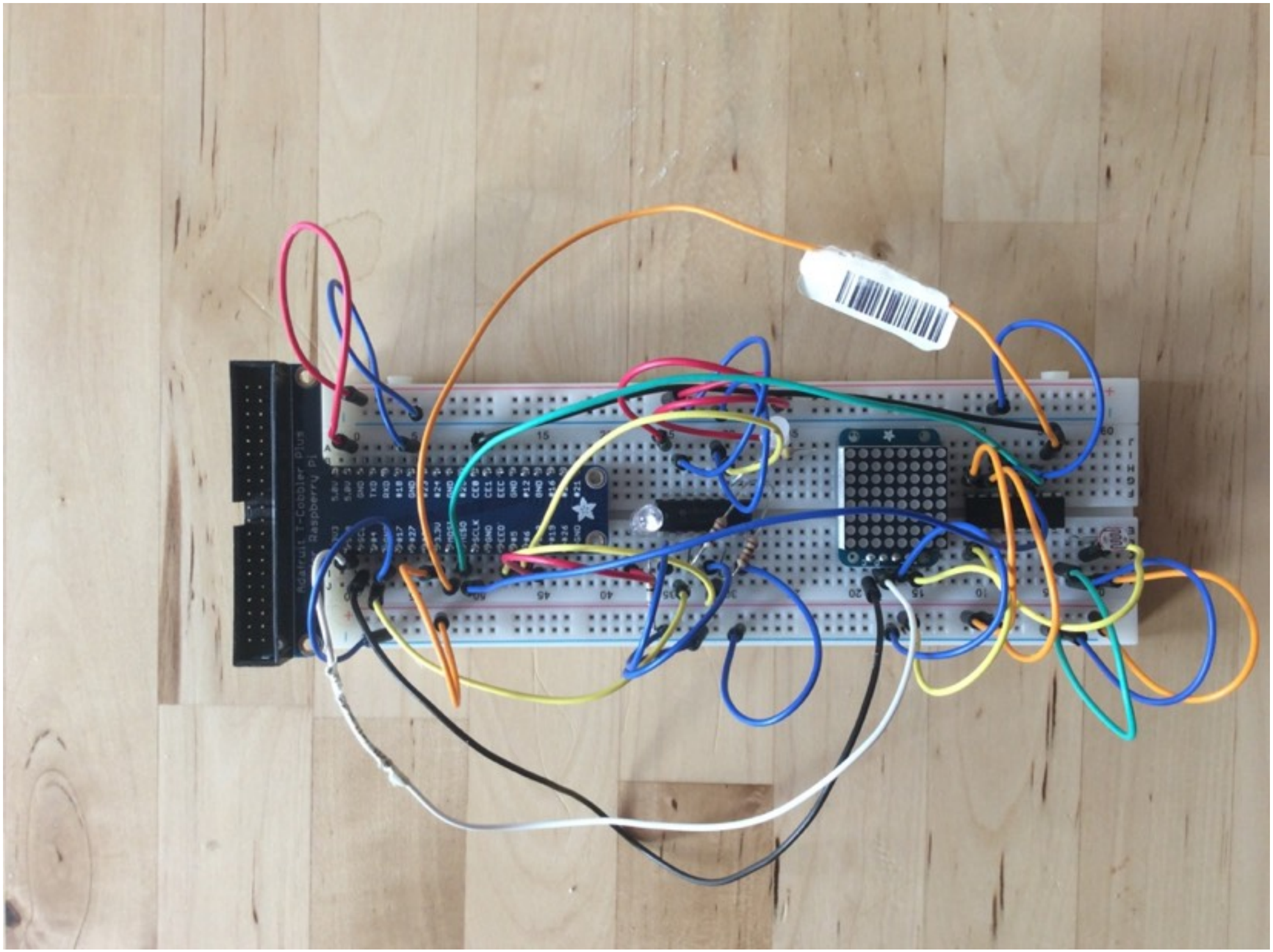
Intro to Raspberry Pi



Intro to Raspberry Pi

Essential Parts

- µSD Card (already installed)
- Network Cable
- µUSB Power Cord

Intro to Raspberry Pi

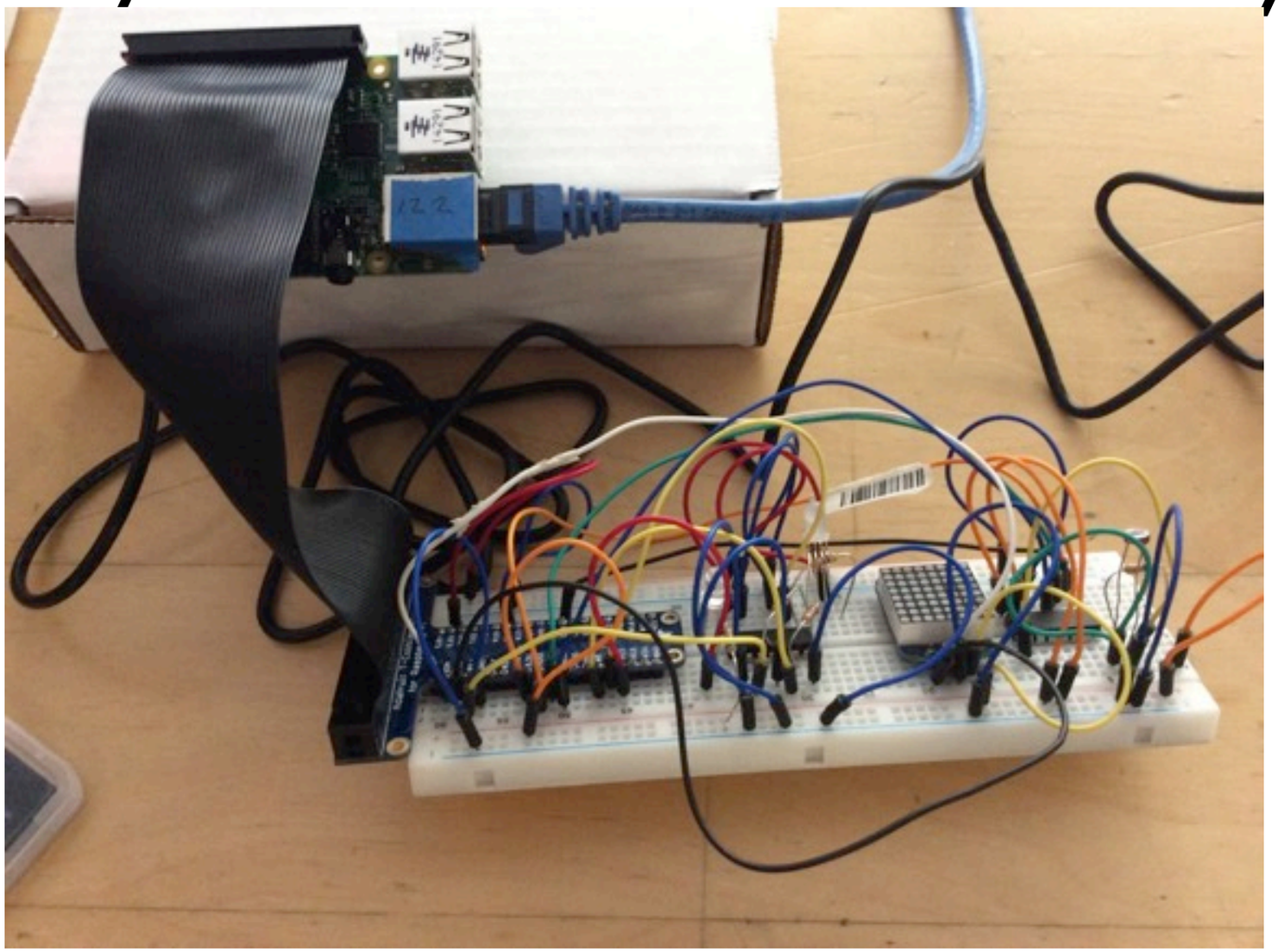
```
dmduser$ ssh pi@172.16.36.XXX
pi@raspberrypi ~ $ sudo python
Python 2.7.3 (default, Mar 18 2014, 05:13:23)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for
more information.
>>>
```

Intro to Raspberry Pi

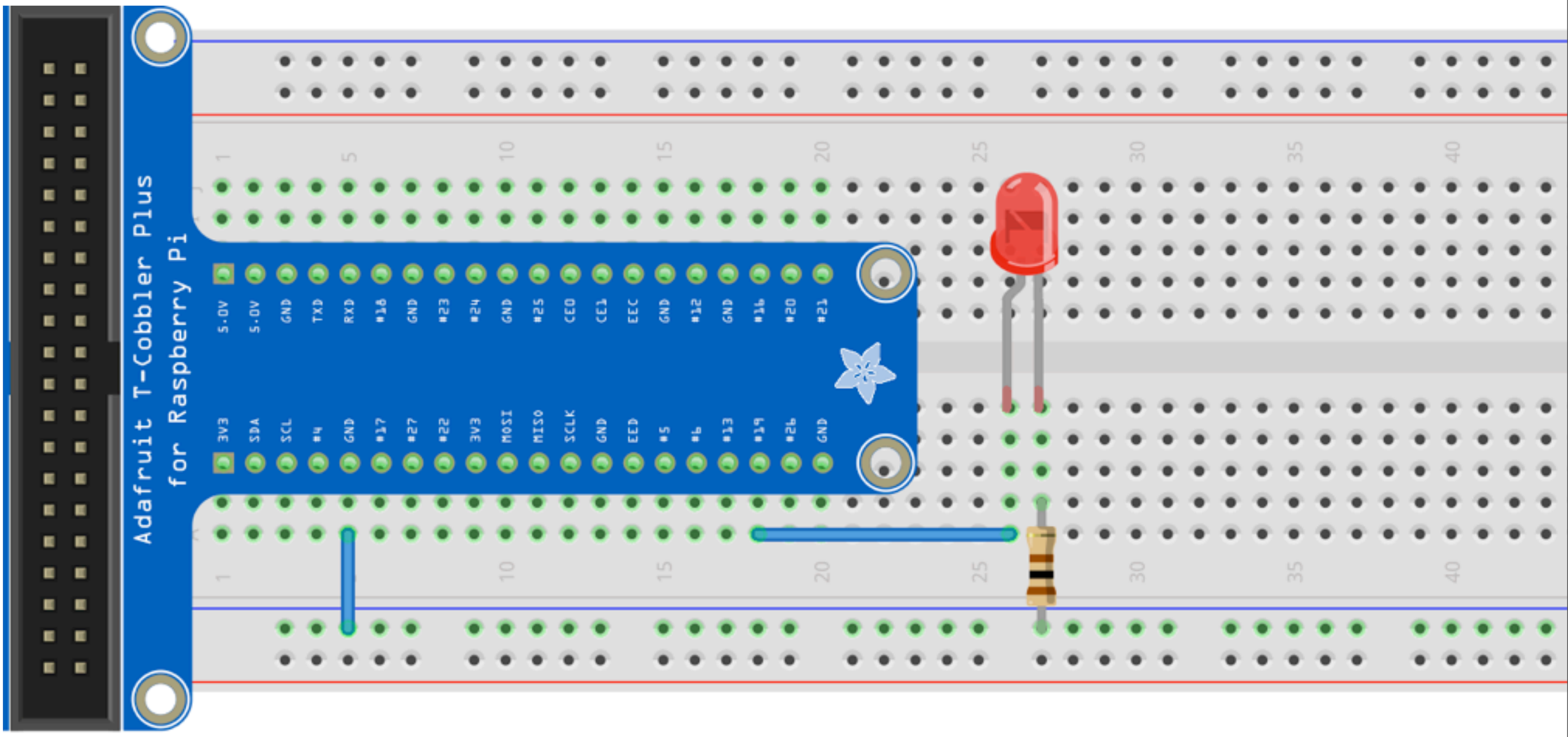
Connect Cable to RasPi

with white stripe towards SD card

Cable only fits into Breadboard 1 way



GPIO & Red LED



- Long leg of LED to Pin 13
- Short leg to resistor, then to Ground

GPIO & Red LED

```
pi@raspberrypi ~ $ sudo python
>>> import RPi.GPIO as GPIO
>>> GPIO.setmode(GPIO.BCM)
>>> GPIO.setup(13, GPIO.OUT)
>>> GPIO.output(13, True)
```

Digital Logic

TRUE	FALSE
On	Off
One	Zero
High	Low

Logic Voltage Levels

What V counts as True?

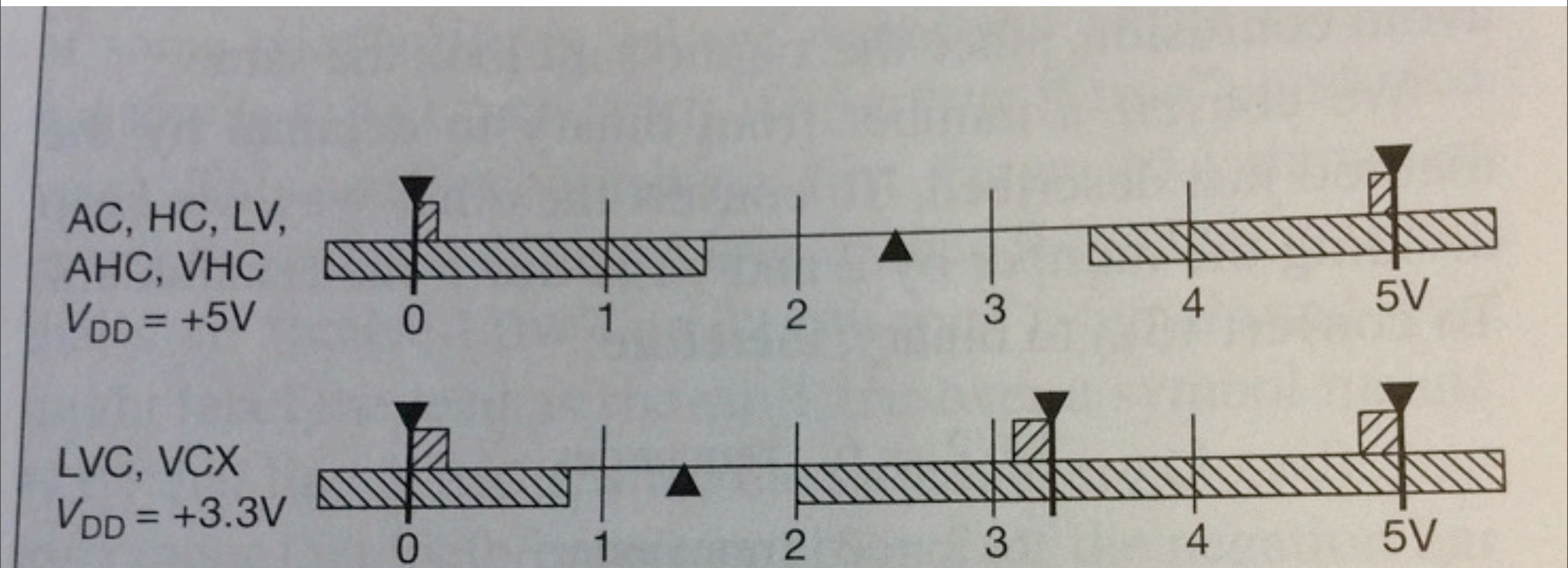
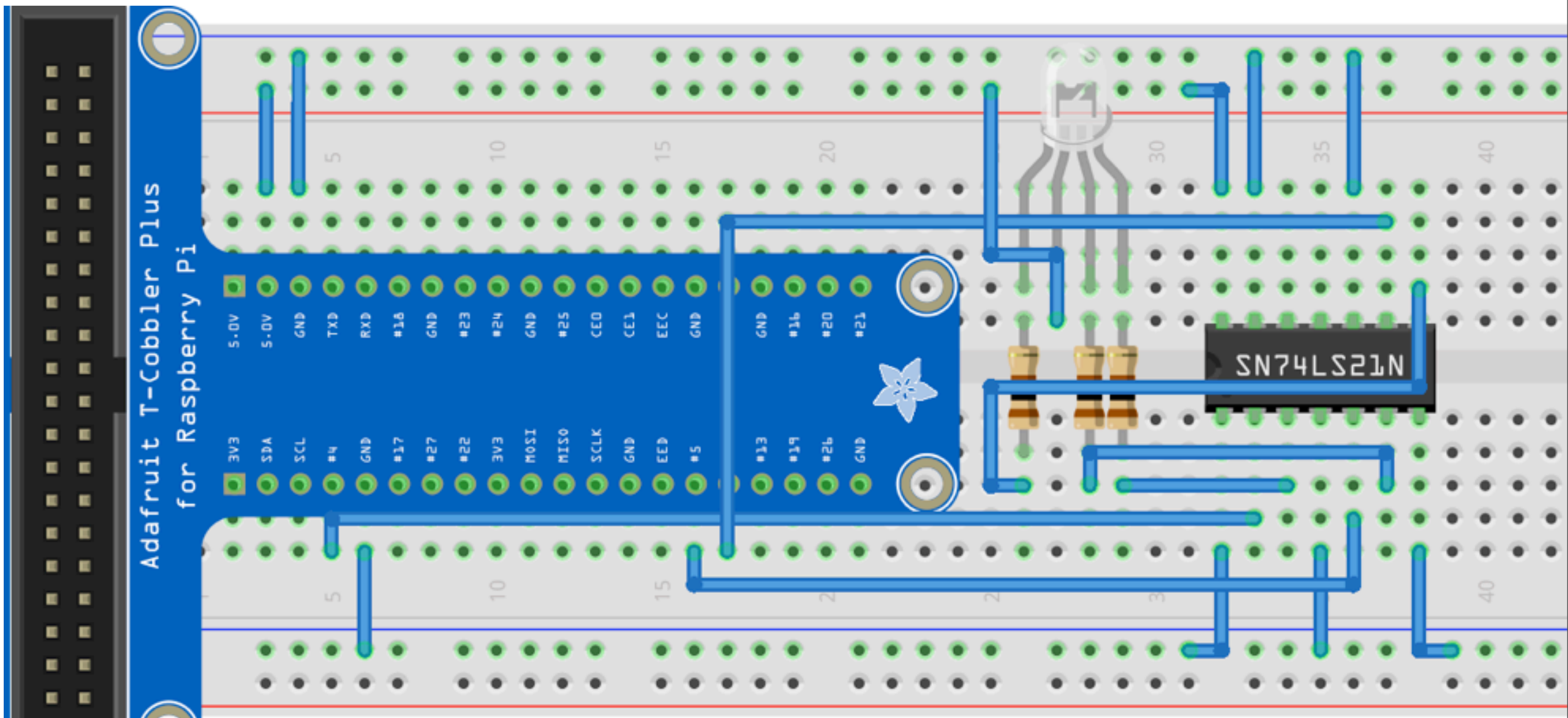


Figure 10.2. Logic levels of some popular logic families.

Level Shifter & RGB LED



- Longest leg of LED to 5V
- Other legs to Resistors, then Level Shifter
- Level Shifter inputs to Pins 4, 5, & 6
- 5 Ground lines on Level Shifter

Level Shifter & RGB LED

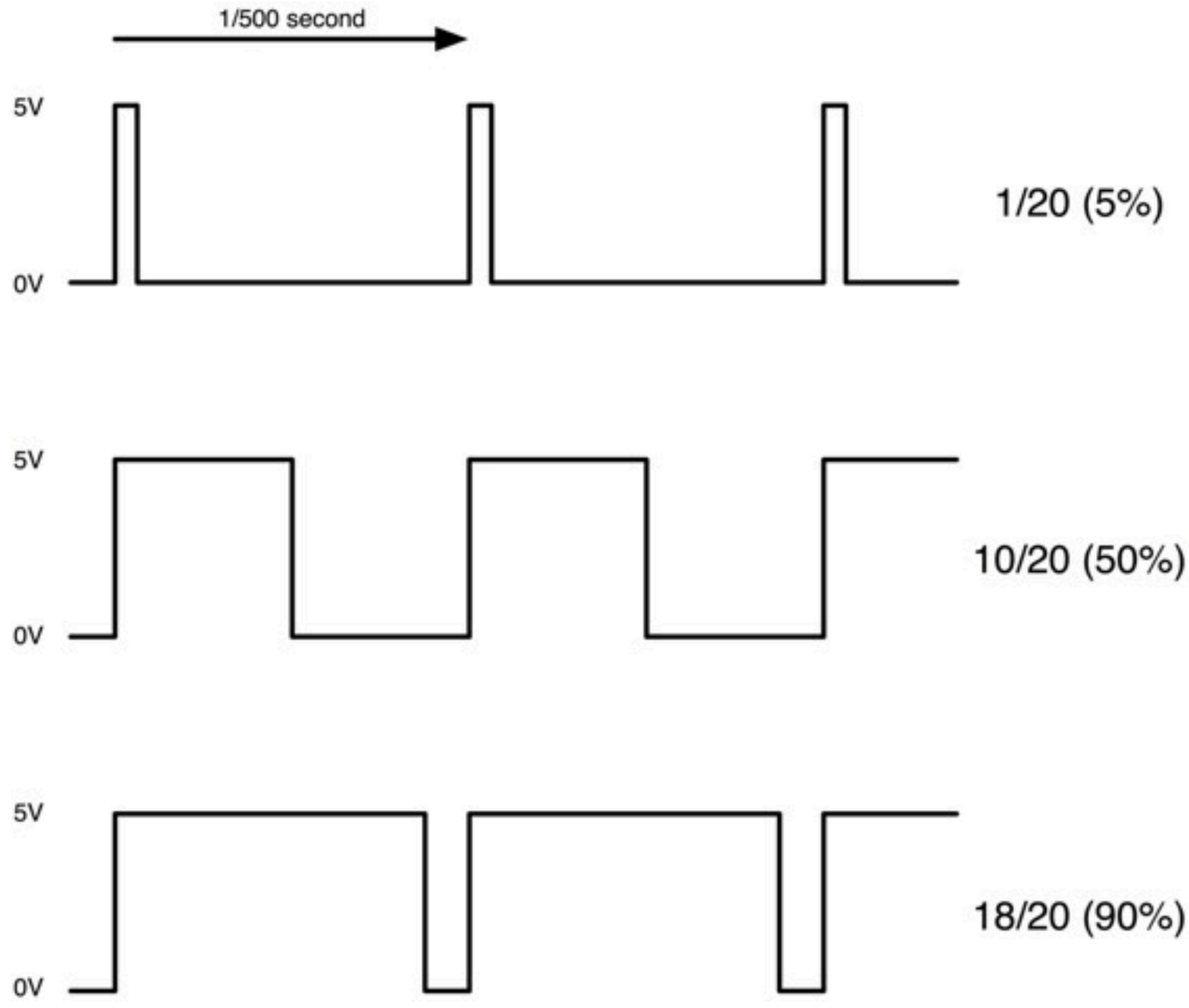
```
>>> GPIO.setup(4, GPIO.OUT)
>>> GPIO.setup(5, GPIO.OUT)
>>> GPIO.setup(6, GPIO.OUT)
>>> GPIO.output(4, True)
>>> GPIO.output(5, True)
```

nano code editor

dmduser$ ssh pi@172.16.36.XXX
pi@raspberrypi ~ $ nano rgb.py
pi@raspberrypi ~ $ python

control-o to save (and return to use the default filename)
control-x to quit

Pulse Width Modulation
(PWM)



© Simon Monk https://learn.adafruit.com/assets/3652

Level Shifter & RGB LED

```
1   import RPi.GPIO as GPIO
2
3   GPIO.setmode(GPIO.BCM)
4   GPIO.setup(4, GPIO.OUT)
5   GPIO.setup(5, GPIO.OUT)
6   GPIO.setup(6, GPIO.OUT)
7
8   b = GPIO.PWM(4, 100)
9   g = GPIO.PWM(5, 100)
10  r = GPIO.PWM(6, 100)
11
12  g.start(50)
13  r.start(100)
14  b.start(0)
```

Level Shifter & RGB LED

```
>>> r = GPIO.PWM(6, 100)
>>> r.start(50)
>>> r.ChangeDutyCycle(100)
>>> GPIO.output(5, False)
>>> g = GPIO.PWM(5, 100)
>>> g.start(50)
>>> b = GPIO.PWM(4, 100)
```

Photoresistor, ADC, & SPI Preliminaries

- sudo aptitude install python-pip python-dev
- sudo pip install spidev
- sudo raspi-config
- Advanced >> turn on SPI, I2C
- Reboot

Photoresistor, ADC, & SPI

Big Ideas

- Take Input from dials, sensors, people, world
- Most sensors are analog (or SPI / I^2C)
- Fewer wires - more logic at each end
- Lots of chips speak SPI

SPI

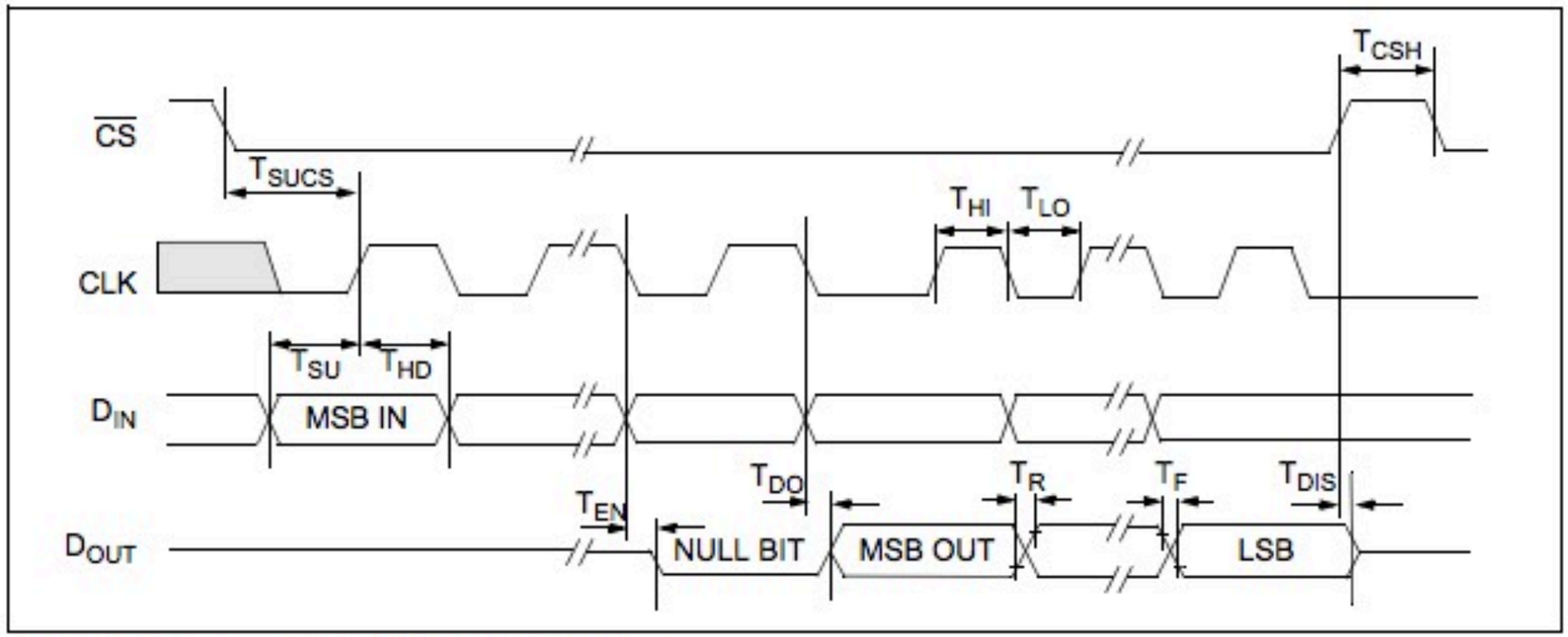
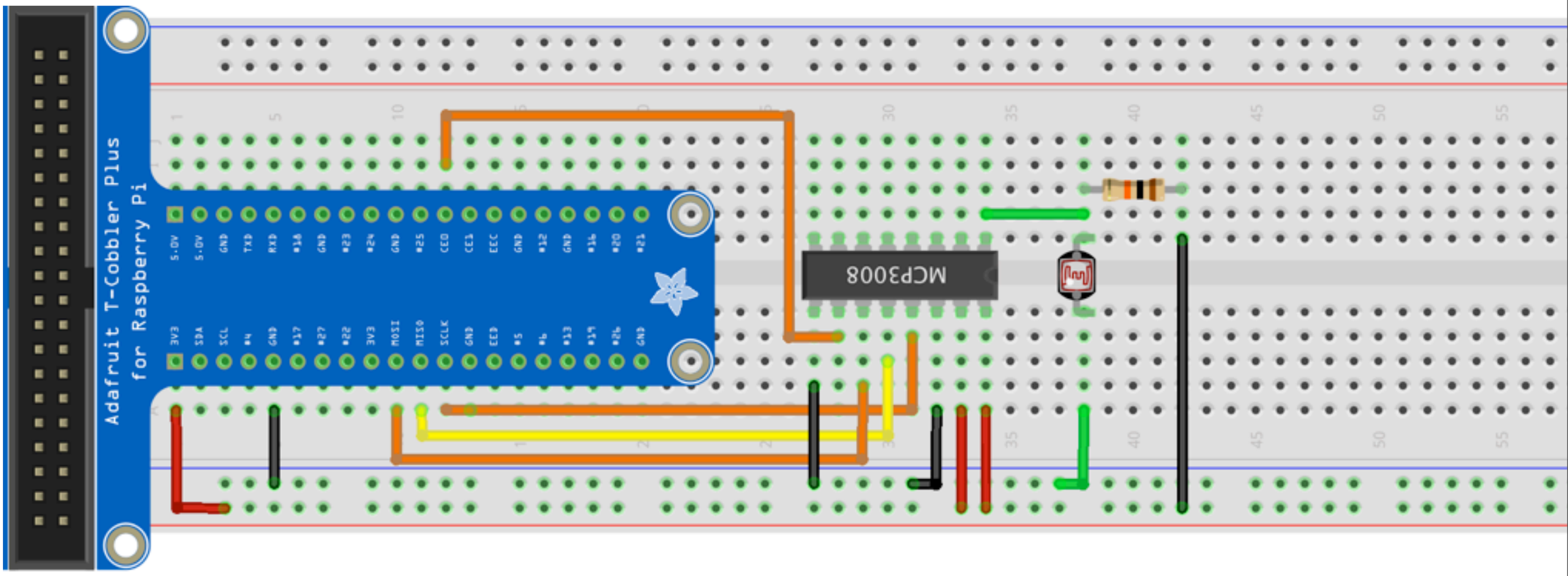


FIGURE 1-1: *Serial Interface Timing.*

- Chip Select Pin - one per chip, other pins shared
- Clock - read data on rising edge of clock
- MOSI - from RasPi to ADC
- MISO - from ADC to RasPi

Photoresistor, ADC, & SPI

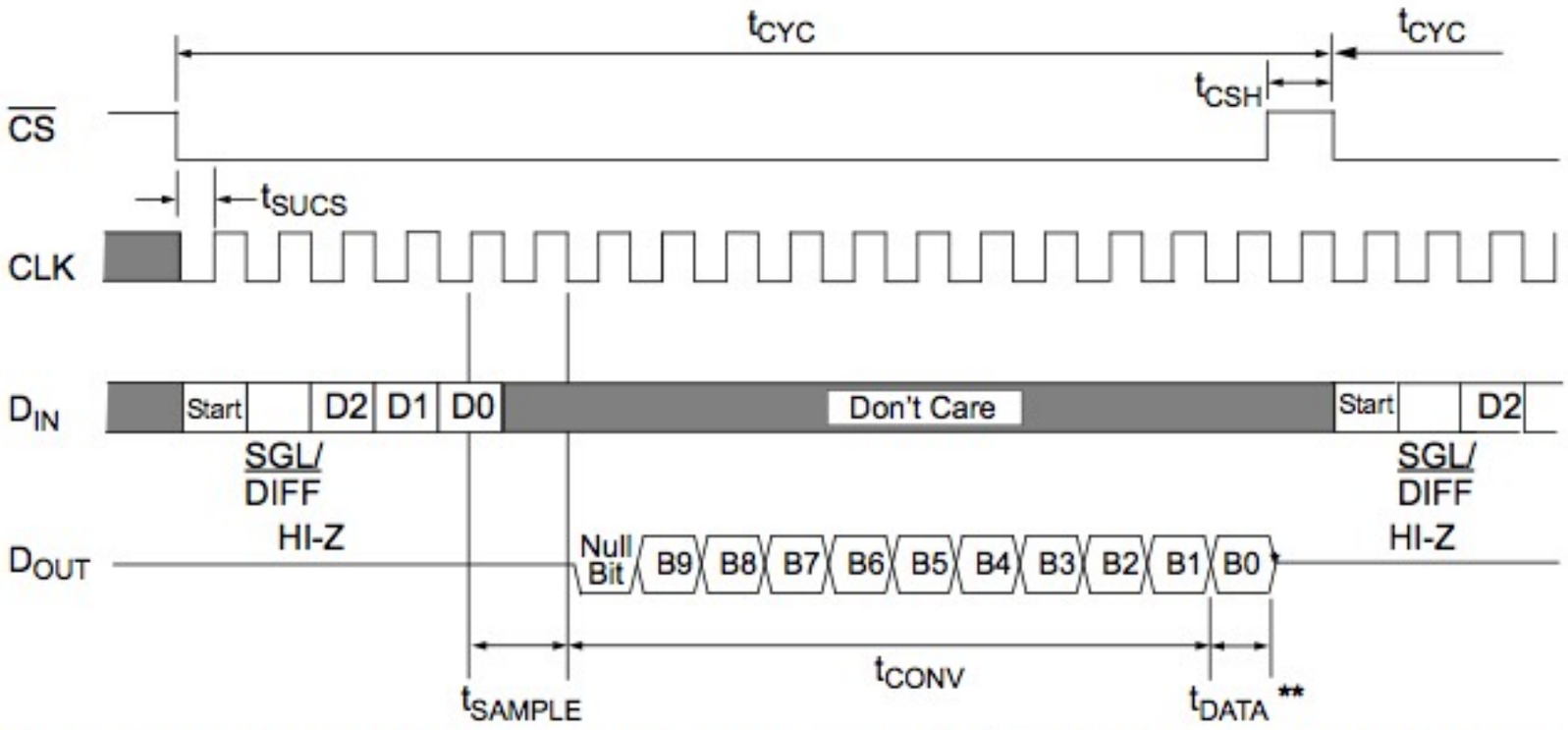


- Ground -> Photoresistor -> Resistor -> 3.3V
- ADC input -> junction Photoresistor & Resistor
- ADC connects to SCLK, MISO, MOSI, CE0
- 8 analog inputs on one side of ADC

MCP3008 ADC

TABLE 5-2: CONFIGURE BITS FOR THE MCP3008

Control Bit Selections				Input Configuration	Channel Selection
Single /Diff	D2	D1	D0		
1	0	0	0	single-ended	CH0
1	0	0	1	single-ended	CH1
1	0	1	0	single-ended	CH2
1	0	1	1	single-ended	CH3
1	1	0	0	single-ended	CH4
1	1	0	1	single-ended	CH5
1	1	1	0	single-ended	CH6
1	1	1	1	single-ended	CH7
0	0	0	0	differential	CH0 = IN+ CH1 = IN-
0	0	0	1	differential	CH0 = IN- CH1 = IN+
0	0	1	0	differential	CH2 = IN+ CH3 = IN-
0	0	1	1	differential	CH2 = IN- CH3 = IN+
0	1	0	0	differential	CH4 = IN+ CH5 = IN-
0	1	0	1	differential	CH4 = IN- CH5 = IN+
0	1	1	0	differential	CH6 = IN+ CH7 = IN-
0	1	1	1	differential	CH6 = IN- CH7 = IN+



* After completing the data transfer, if further clocks are applied with $\overline{CS}$ low, the A/D converter will output LSB first data, then followed with zeros indefinitely. See Figure 5-2 below.

** t_{DATA}: during this time, the bias current and the comparator powers down while the reference input becomes a high-impedance node.

FIGURE 5-1: *Communication with the MCP3004 or MCP3008.*

Datasheet has all the details:

- How many bits per message
- What they all mean

Photoresistor, ADC, & SPI

```
import spidev
import time

adc = spidev.SpiDev()
adc.open(0,0)

def readadc(channel):
    r = adc.xfer2([1, (8+channel)<<4, 0])
    adcout = ((r[1]&3) << 8) + r[2]
    return adcout

while True:
    print(readadc(0))
    time.sleep(1)
```

Bits & Bytes

Many bits in sequence. Base 2 instead of base 10.
Any number of digits in a number.

But most often, we use multiples of 8 bits.
8 bits = 1 byte

Spec sheet says send 4 bits, listen for 11 bits.
Python library sends, receives 1 byte at a time.

Bits & Bytes

#	7	6	5	4	3	2	1	0
	128	**64**	**32**	**16**	**8**	**4**	**2**	**1**
1	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	1	0
3	0	0	0	0	0	0	1	1
255	1	1	1	1	1	1	1	1

Bitwise Operations

```
def readadc(channel):
  r = adc.xfer2([1, (8+channel)<<4, 0])
  adcout = ((r[1]&3) << 8) + r[2]
  return adcout
```

x << y shifts the number x left by 4 places
1 becomes $2^4 = 16$
2 becomes $2^5 = 32$

a & b calculates AND
for each pair of bits

&	0	1
0	0	0
1	0	1

Flask Web Server

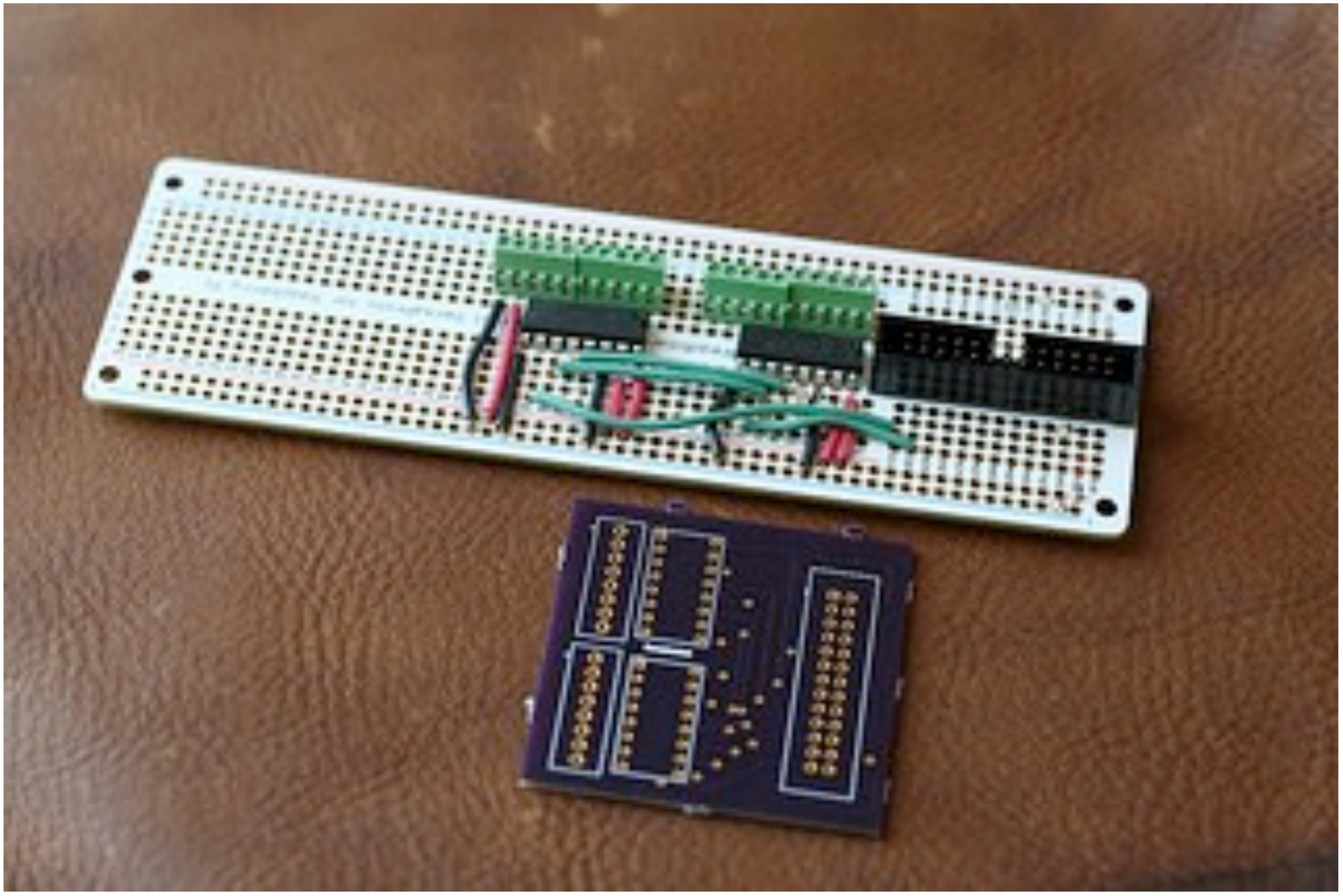
Flask is:

- A Python library
- For writing web applications
- Easy to use
- Very well documented

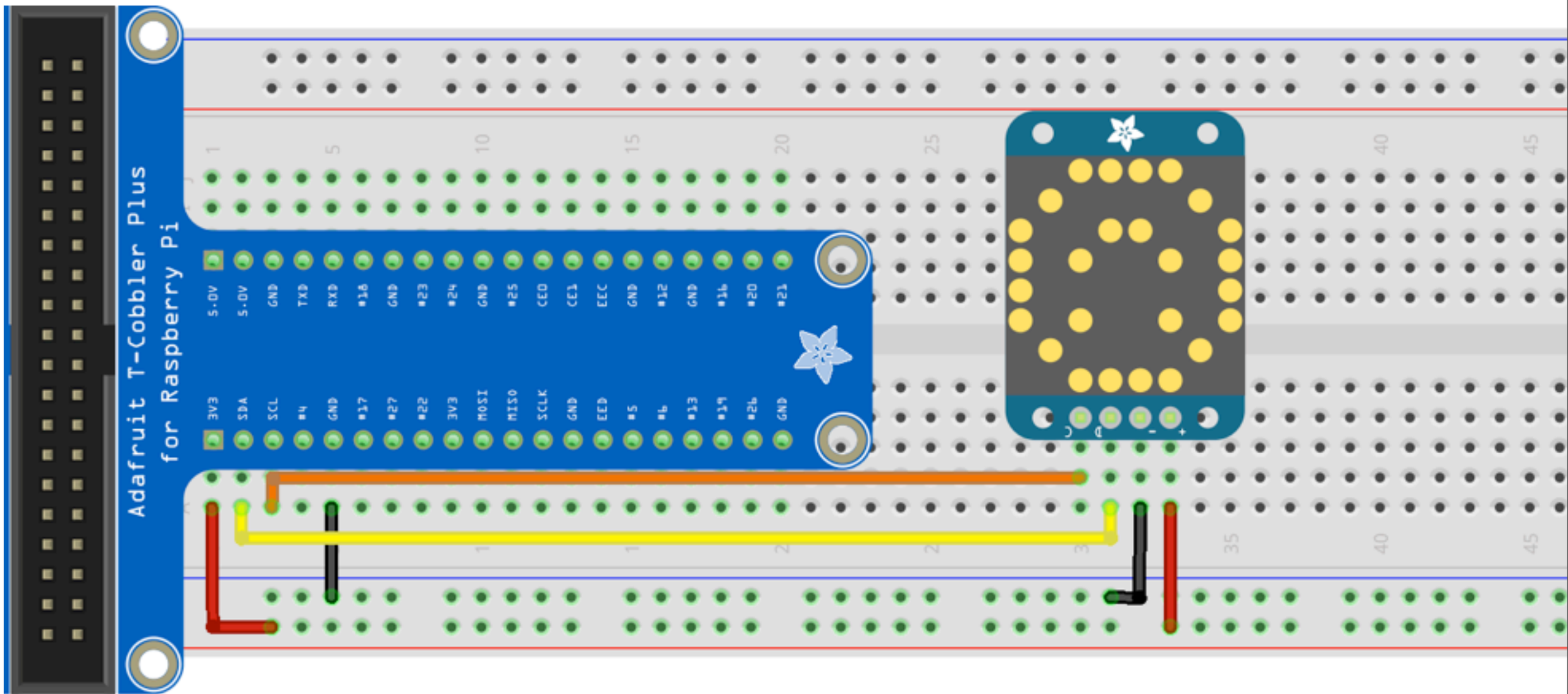
Flask Web Server

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"

if __name__ == "__main__":
    app.run(host=”0.0.0.0”, debug=True)
```

Photoresistor, ADC, & SPI
Breadboard to PCB

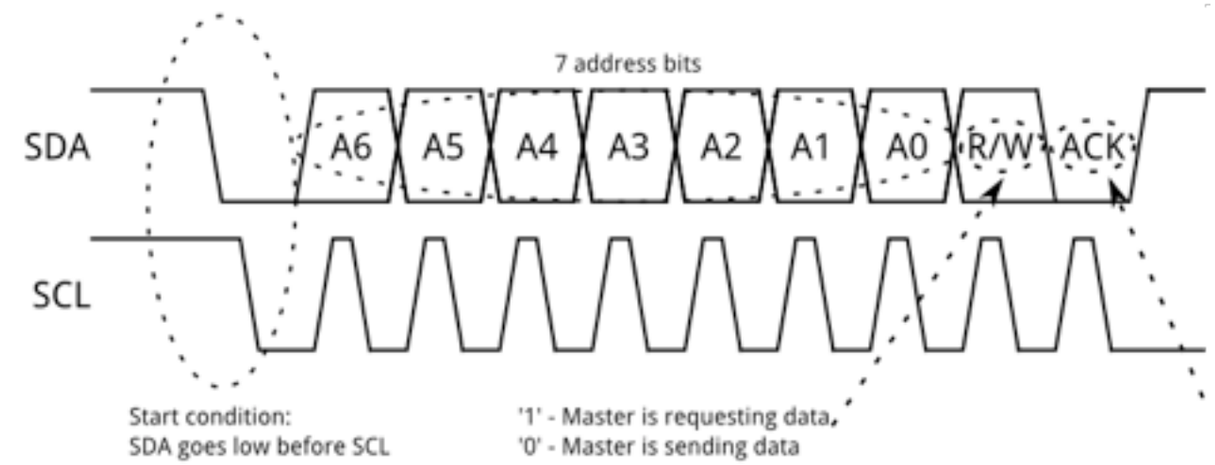


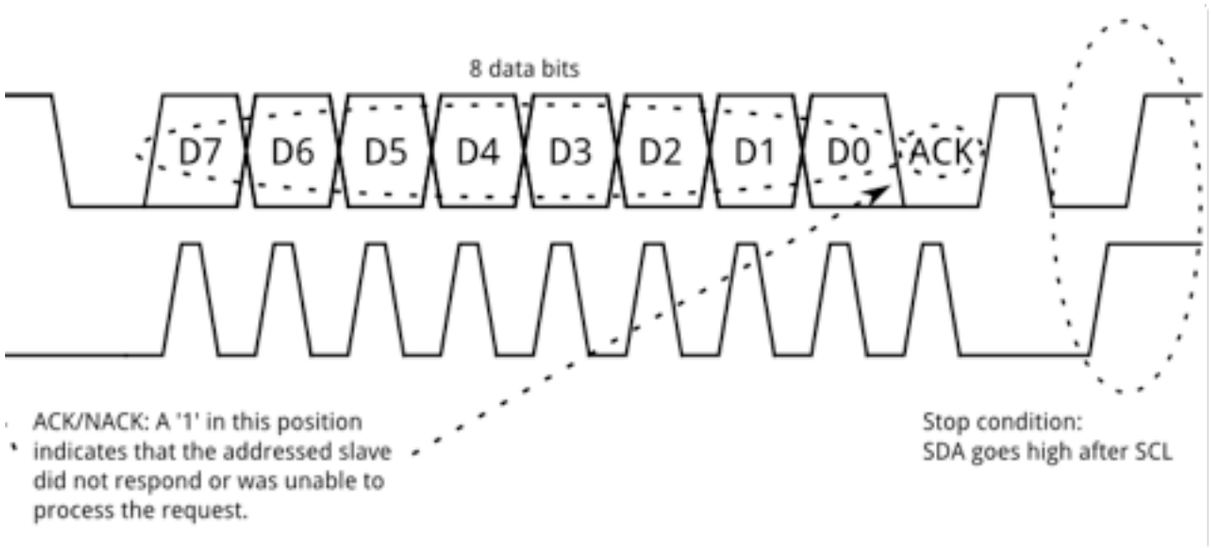
8x8 LED Matrix & I²C



- LED - to Ground
- LED C to RasPi SCL
- LED D to RasPi SDA

8x8 LED Matrix & I²C





- First send the Address (7-bits)
- Then R/W Bit
- Then one byte, in the specified direction

Learn More: https://learn.sparkfun.com/tutorials/i2c

8x8 LED Matrix & I^2C
Why use I^2C?

- Fewer wires
- More devices
- Less time reading spec sheets (more standardized)

8x8 LED Matrix & I²C
Setting up the RasPi

At the Linux prompt ($), on the RasPi, run:

$ sudo aptitude install python-smbus i2c-tools git

$ git clone https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code.git

$ cd Adafruit-Raspberry-Pi-Python-Code

$ cd Adafruit_LEDBackpack

$ sudo python ex_8x8_pixels.py

8x8 LED Matrix & I²C
Setting up the RasPi

dmduser$ ssh pi@172.16.36.XXX

At the Linux prompt ($), on the RasPi, run:

$ sudo python ex_8x8_pixels.py

8x8 LED Matrix & I²C
How does it work?

At the Linux prompt ($), on the RasPi, run:

$ nano ex_8x8_pixels.py

```
#!/usr/bin/python

import time
import datetime
from Adafruit_8x8 import EightByEight

# ===========================================================================
# 8x8 Pixel Example
# ===========================================================================
grid = EightByEight(address=0x70)

print "Press CTRL+Z to exit"

# Continually update the 8x8 display one pixel at a time
while(True):
  for x in range(0, 8):
    for y in range(0, 8):
      grid.setPixel(x, y)
      time.sleep(0.05)
  time.sleep(0.5)
  grid.clear()
  time.sleep(0.5)
```

8x8 LED Matrix & I²C
How does it work?

At the Linux prompt ($), on the RasPi, run:

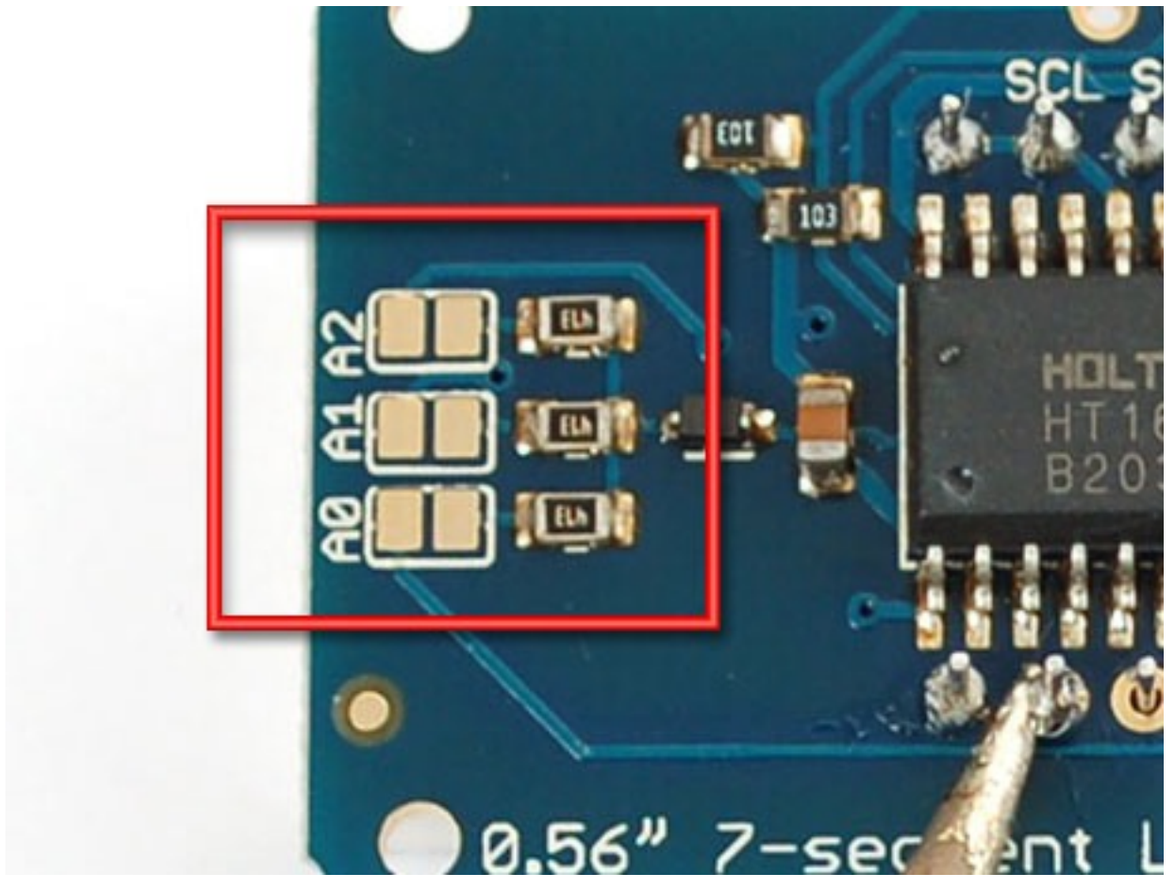
$ nano Adafruit_8x8.py

$ nano Adafruit_LEDBackpack.py

```
11  class EightByEight:
12    disp = None
13
14    # Constructor
15    def __init__(self, address=0x70, debug=False):
16      if (debug):
17        print "Initializing a new instance of LEDBackpack at 0x%02X" % address
18      self.disp = LEDBackpack(address=address, debug=debug)
19
20    def writeRowRaw(self, charNumber, value):
21      "Sets a row of pixels using a raw 16-bit value"
22      if (charNumber > 7):
23        return
24      # Set the appropriate row
25      self.disp.setBufferRow(charNumber, value)
26
27    def clearPixel(self, x, y):
28      "A wrapper function to clear pixels (purely cosmetic)"
29      self.setPixel(x, y, 0)
30
```

8x8 LED Matrix & I²C
Connecting More Devices

Maybe you want 2 LED Matrices.
I said that I2C supports multiple devices, but every matrix has address 0x70
Short across pads on back of LED Matrix
(You will only have two pairs of pads)



Setting up a new RasPi

I did a few things for you before class:

Raspian Linux https://www.raspberrypi.org/downloads/
copy to SD card

Adafruit Occidentalis Bootstrapper
https://github.com/adafruit/Adafruit-Pi-Finder
(IP Address, more hardware drivers)

sudo gem install rmate

Cron

crontab -e

When RasPi starts:
@reboot /usr/bin/python /home/pi/rgb.py

Every Hour (min hr d/m month d/w)
0 * * * * /usr/bin/python /home/pi/awesome.py

Every 15 minutes:
*/15 * * * * /usr/bin/python /home/pi/awesomer.py

git

We'll learn to use git for:

- Following another's project
- Saving changes to your local project
- Syncing changes between your computers
- Contributing upstream
- ~~Managing large projects with dozens of developers~~

Git is powerful, sometimes more than you need. You can learn more when you need it.

git: following

git clone https://github.com/bergey/raspi-classes.git
git pull

git: local

mkdir python
mv *.py python
cd python
git init
git status
git commit rgb.py
make some changes to rgb.py
git status
git commit -a
git log

git: sync

make a github account

make a new repo on github

git remote add origin URL

git push

on another computer

git clone

git pull

git: contributing

https://github.com/bergey/raspi-classes
Fork it!
git clone YOUR_FORK
or in your earlier checkout:
git remote add USERNAME YOUR_FORK
git commit
git push
Make a Pull Request on Github

git: learning more

A good introduction to git:
http://swcarpentry.github.io/git-novice/

Other people use git differently than you do:
https://www.atlassian.com/git/tutorials/comparing-workflows/

All (most of) the gory details:
http://git-scm.com/book/en/v2

More about SPI & GPIO

Adafruit also provides a Python library for the ADC

$ rmate Adafruit-Raspberry-Pi-Python-Code/Adafruit_MCP3008/mcp3008.py

- Doesn't use spidev, uses GPIO
- Works on any GPIO pins, but very slow