Introduction to Raspberry Pi

ROCHESTER MAKERSPACE

2021

Class Objectives

1. Become familiar with Raspberry Pi hardware and software
2. Be aware of the range of Raspberry Pi boards
3. Understand how to create and run a program on an Raspberry Pi
4. Provide basic Python programing syntax
5. Understand how to control a simple circuit from an Raspberry Pi
6. Understand cautions when using General Purpose Input/Output pins (GPIO)
7. Understand proper Shutdown procedure
8. Provide advanced projects for continued learning
9. Provide list of resources for learning more

Class notes and code

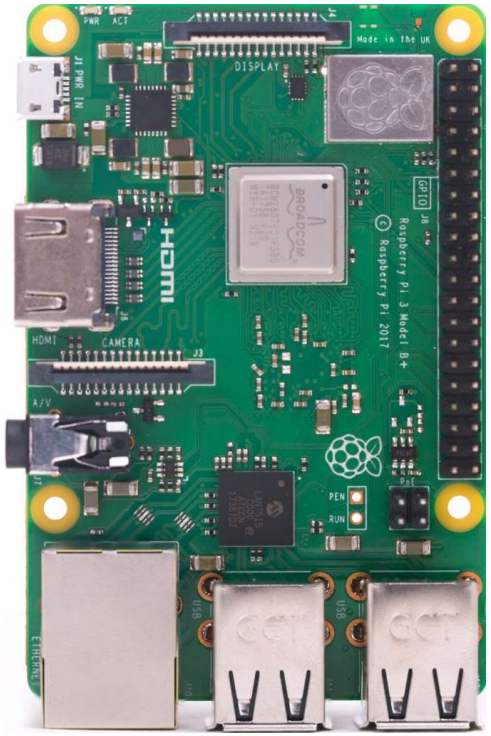
- All the class notes and code can be downloaded from GitHub:
  - https://github.com/RochesterMakerSpace/Raspberry-Pi-class
- Pull down the green "Code" button and select "Download ZIP"

History

- The Raspberry Pi was created in February 2012 by the Raspberry Pi Foundation to promote and teach basic computer science in schools and colleges around the UK.
- The Raspberry Pi Foundation is a UK-based charity that works to put the power of computing and digital making into the hands of people all over the world. We do this so that more people are able to harness the power of computing and digital technologies for work, to solve problems that matter to them, and to express themselves creatively.
- More than 1M units sold in the first year, more than 4.5M by 2014, 36M in late 2020.
- Raspberry Pi 4 B went on sale in June 2019, offering faster processor and ports, as well as options for more RAM.
- Raspberry Pi Pico went on sale in January 2021. It is the first microcontroller offered by the Raspberry Pi Foundation.

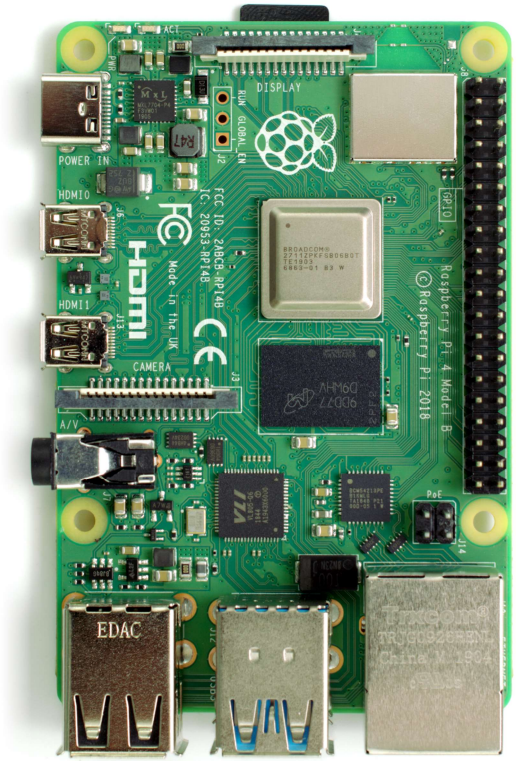
Raspberry Pi 3 B+ ($35)

- Broadcom BCM2837B0, Quad Core Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
- 1GB LPDDR2 SDRAM
- 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE
- Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps)
- Extended 40-pin GPIO header
- Full-size HDMI
- 4 USB 2.0 ports
- CSI camera port for connecting a Raspberry Pi camera
- DSI display port for connecting a Raspberry Pi touchscreen display
- 4-pole stereo output and composite video port
- Micro SD port for loading operating system and storing data
- 5V/2.5A DC power input (micro USB)



Raspberry Pi 4 B ($35–2GB, $75–8 GB)

- Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- 2GB, 4GB or 8GB LPDDR4-2400 SDRAM (depending on model)
- 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE, Gigabit Ethernet
- 2 USB 3.0 ports; 2 USB 2.0 ports
- Raspberry Pi standard 40 pin GPIO header (fully backwards compatible with previous boards)
- 2 × micro-HDMI ports (up to 4kp60 supported), 2-lane MIPI DSI display port
- 2-lane MIPI CSI camera port
- 4-pole stereo audio and composite video port
- H.265 (4kp60 decode), H264 (1080p60 decode, 1080p30 encode), OpenGL ES 3.0 graphics
- Micro-SD card slot for loading operating system and data storage
- 5V DC via USB-C connector (minimum 3A)
- 5V DC via GPIO header (minimum 3A)



Raspberry Pi Zero W ($10)

- Broadcom BCM2835,1 GHz, single-core ARM v7 CPU
- 512 MB RAM
- IEEE 802.11n wireless (2.4 Ghz), Bluetooth 4.1, Bluetooth Low Energy (BLE)
- Micro USB for power
- Micro USB for OTG (On-The-Go)
- Raspberry Pi standard 40 pin GPIO header (fully backwards compatible with previous boards)
- Mini HDMI
- 2-lane MIPI CSI camera port
- Micro-SD card slot for loading operating system and data storage

Raspberry Pi 400 ($70)

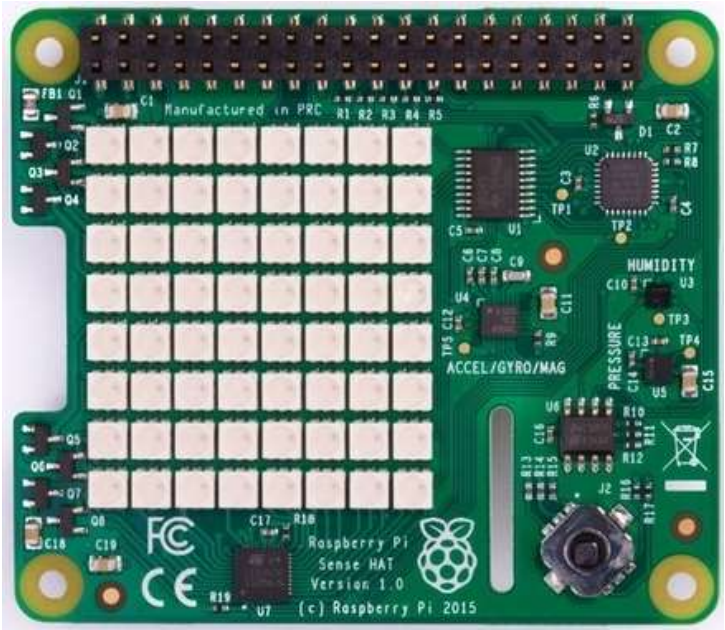
- Broadcom BCM2711 quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.8GHz
- 4GB LPDDR4-3200
- Dual-band (2.4GHz and 5.0GHz) IEEE 802.11b/g/n/ac wireless LAN
- Bluetooth 5.0, BLE
- Gigabit Ethernet
- 2 × USB 3.0 and 1 × USB 2.0 ports
- Horizontal 40-pin GPIO header
- 2 × micro HDMI ports (supports up to 4Kp60)
- H.265 (4Kp60 decode); H.264 (1080p60 decode, 1080p30 encode)
- MicroSD card slot for operating system and data storage
- 78- or 79-key compact keyboard (depending on regional variant)
- 5V DC via USB connector
- Operating temperature: 0°C to +50°C ambient
- Maximum dimensions 286 mm × 122 mm × 23 mm

Raspberry Pi Compute Module 4 ($25)

- Broadcom BCM2711 quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- H.265 (HEVC) (up to 4Kp60 decode), H.264 (up to 1080p60 decode, 1080p30 encode)
- OpenGL ES 3.0 graphics
- Options for 1GB, 2GB, 4GB or 8GB LPDDR4-3200 SDRAM (depending on variant)
- Options for 0GB ("Lite"), 8GB, 16GB or 32GB eMMC Flash memory (depending on variant)
- Option for fully certified radio module:
  2.4 GHz, 5.0 GHz IEEE 802.11 b/g/n/ac wireless;
  Bluetooth 5.0, BLE;
  On-board electronic switch to select either external or PCB trace antenna

Sense HAT ($35)

- The Sense HAT is an add-on board for Raspberry Pi, made especially for the Astro Pi mission – it launched to the International Space Station in December 2015 – and is now available to buy.
- The Sense HAT has an 8×8 RGB LED matrix, a five-button joystick and includes the following sensors:
- Gyroscope
- Accelerometer
- Magnetometer
- Temperature
- Barometric pressure
- Humidity
- We've also created a https://pythonhosted.org/sense-hat/ providing easy access to everything on the board.
- On-line emulator: https://trinket.io/sense-hat

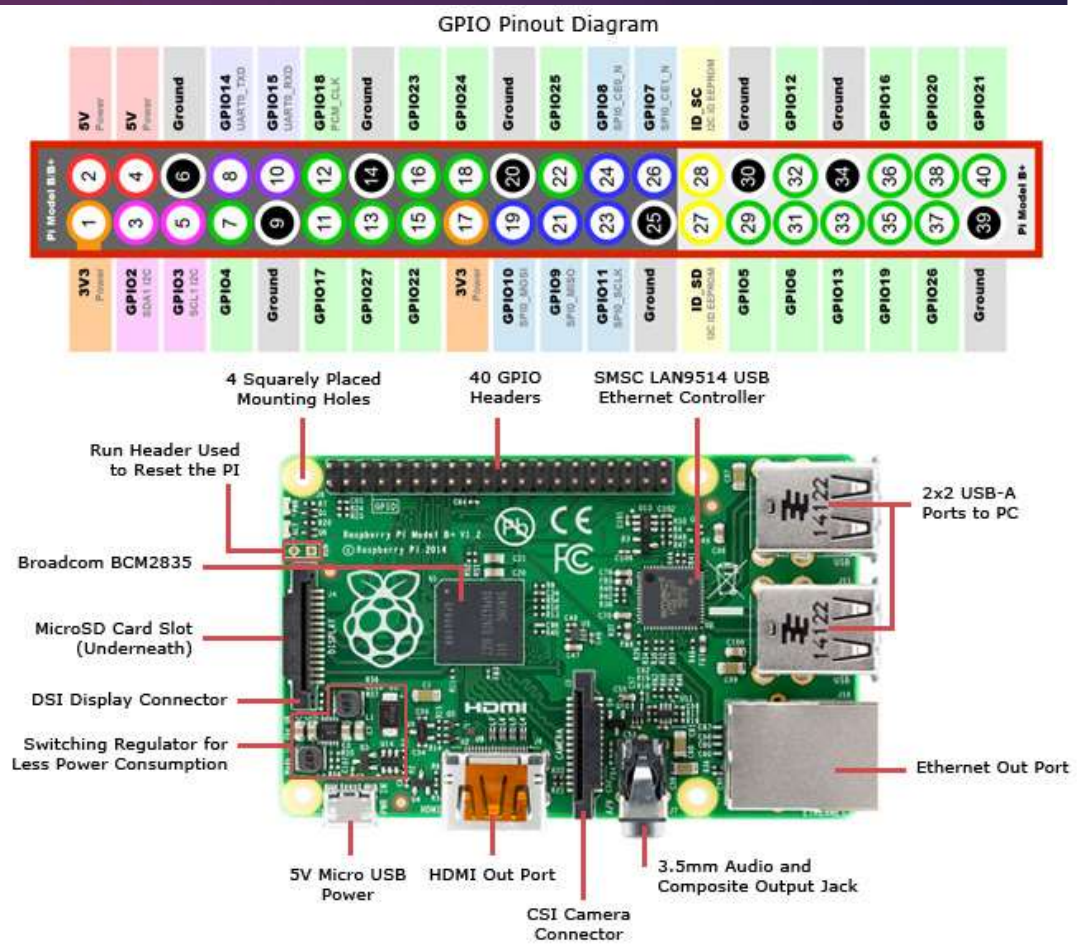


Raspberry Pi Pico ($4)

- RP2040 microcontroller, Dual-core Arm Cortex-M0+, 133 Mhz
- 264 Kb on-chip RAM
- 2 Mb on-board QSPI Flash
- 26 multifunction GPIO pins, including 3 analog inputs
- 2 x UART, 2 x SPI controllers, 2 x I2C controllers, 16 PWM channels
- 1 x USB 1.1 controller and PHY, with host and device support
- 8 x Programmable I/O (PIO) state machine for custom peripheral support
- Drag-and-drop programming using mass storage over USB
- Temperature sensor
- Accelerated integer and floating-point libraries on-chip

Linux with GPIO!

- At 1.4MHz, quad-core and 1GB of RAM, RPi 3 B+ is a capable small Linux system
- Raspberry Pi 4 offers a faster processor, two USB 3.0 ports, two (micro) HDMI ports, options of 2GB, 4GB, or 8GB of RAM
- Both offer GPIO enabling the creation of different projects e.g., robots



Comparison vs. Arduino

- Pros
  - More RAM + Storage
  - More processing power
  - Debug in place
  - Multiple, concurrent programs
  - Built-in HDMI and camera interfaces
  - Built-in WiFi and Bluetooth comms
  - Built-in Ethernet and 4 USB ports
  - Pluggable SD card to expand memory
- Cons
  - Non-real-time OS makes real-time programming impractical
  - Requires some understanding of Linux
  - Much more configuration required
  - More electrical power required
  - No Analog (A/D) inputs
  - Narrow input voltage tolerance
  - More expensive

Bottom Line: Choose the right tool for the job

Remote connections

- VNC (connection to user pi only)
  - Enable VNC in Raspberry Pi Preferences->Raspberry Pi Configuration->Interfaces
  - Install https://www.realvnc.com/en/connect/download/viewer/ or https://www.tightvnc.com/download.php viewer on Windows
- Remote Desktop (connection to all users)
  - Install xrdp on Raspberry Pi: `sudo apt-get install xrdp`
  - Connect from Windows using Remote Desktop
- Secure Shell
  - Enable SSH in Raspberry Pi Preferences->Raspberry Pi Configuration->Interfaces
  - Enable OpenSSH client feature on Windows: Settings->Apps->Apps & Features->Optional features
  - Or https://www.putty.org/ on Windows
- sftp (secure ftp to transfer files)
  - Enable SSH on Raspberry Pi Preferences->Raspberry Pi Configuration->Interfaces
  - Install https://winscp.net/eng/download.php or https://filezilla-project.org/ on Windows
- Transfer files:
  - USB flash drive
  - Cloud storage service

Resources

- Raspberry Pi Foundation: https://www.raspberrypi.org/
- Python Introduction: https://www.raspberrypi.org/documentation/usage/python/
- Embedded Linux Wiki: www.elinux.org
- Adafruit: https://www.adafruit.com/category/105?gclid=Cj0KCQiA962BBhCzARIsAIpWEL1YEmA7k-jbQbsoDKZOApe8oyufKTMf4lNfXgVyIHtExK8ghNguFY8aAmpREALw_wcB
- Dronebot Workshop: https://dronebotworkshop.com/raspberry-pi-microcomputer/
- Explaining Computers: https://www.explainingcomputers.com/sbc.html
- Andreas Spiess: https://www.youtube.com/channel/UCu7_D0o48KbfhpEohoP7YSQ
- Paul McWhorter: https://toptechboy.com/raspberry-pi-with-linux-lessons/
- M Heidenreich: https://www.youtube.com/watch?v=g-tSLJ3oFzE&list=PL68Go3Wa2wAIPmz2KGwEbC7g9E6ZUgchJ

Python Quickstart

- IDE (Integrated Development Environment) for Python:
  - Beginner: Thonny Python IDE. Python 3.7.x in right hand corner.
  - Intermediate: Geany.
  - Advanced: Visual Studio Code
- Key feature: most everything in Python is a object. An object is an instance of a class.
- Key feature: indention is important!
- import statement to import modules into the program to expand functionality.
- Comments: "#" or ''' multiline '''
- Core data types: string, integer, float, Boolean
- Variables used to store values. Must start with a letter or "_". Convention: use "_" between words
  - Ex: hello_world = "Hello World" (called snake case)
- 7 arithmetic operators: + - * / % ** //. * / perform before + /. Use () for clarity.
- f string: new (Python 3.6) way to format strings

Python Quickstart

- Collection types: Lists, Tuples, Dictionaries
  - Lists: define with [ ], access with list_name[0], list_name[start:end:step], append() to add to list
  - Tuples: like list but can't change values (immutable). Use () instead of []. Can convert to list to change
  - Dictionary: key value pair. Use {<key1>:<value1>, <key2>:value2, etc}
- Conditionals: if - elif - else. Logical operators: and, or, not
- Looping: perform an action a set number of times: for <var> in range(0, 10):
- Looping: perform repeated action but don't know when will end: while(<conditional>):
- Functions: reuse code: def fun_name():
- File I/O: write and read to file system
- List module functions from >>> prompt: dir(<module> or <variable>)
- List help on a module function from >>> prompt: help(module.function)

More info: https://www.youtube.com/watch?v=N4mEzFDjqtA&list=WL&index=1&t=1736s

GPIO Libraries

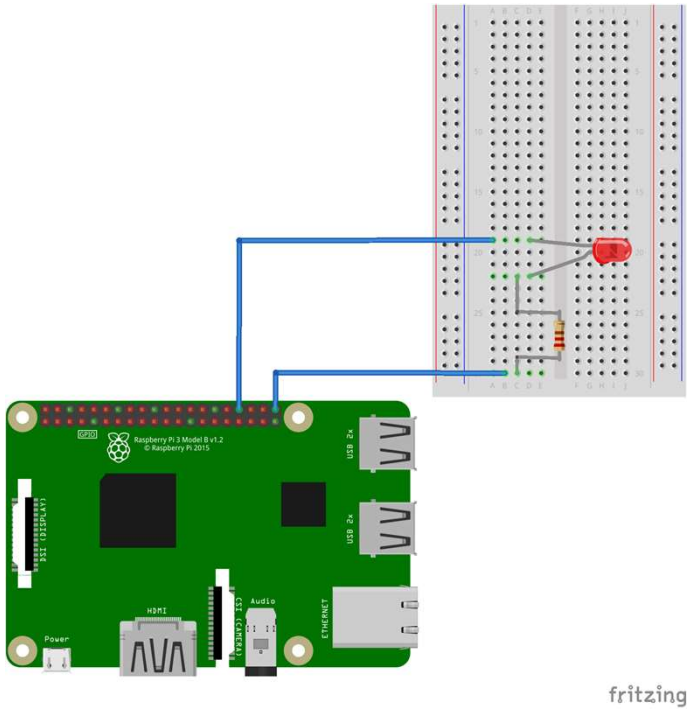
- RPi.GPIO: https://sourceforge.net/p/raspberry-gpio-python/wiki/BasicUsage/
  - First and most popular.
- gpiozero: https://gpiozero.readthedocs.io/en/v1.3.1/
  - Designed for ease of use.
  - Built on RPi.GPIO
  - GPIO cleanup on normal program end
- pigpio: http://abyz.me.uk/rpi/pigpio/python.html
  - Talks to pigpio daemon
  - Remote manipulation of GPIO
- RPIO: https://pythonhosted.org/RPIO/
  - Hardware PWM
  - TCP socket interrupts

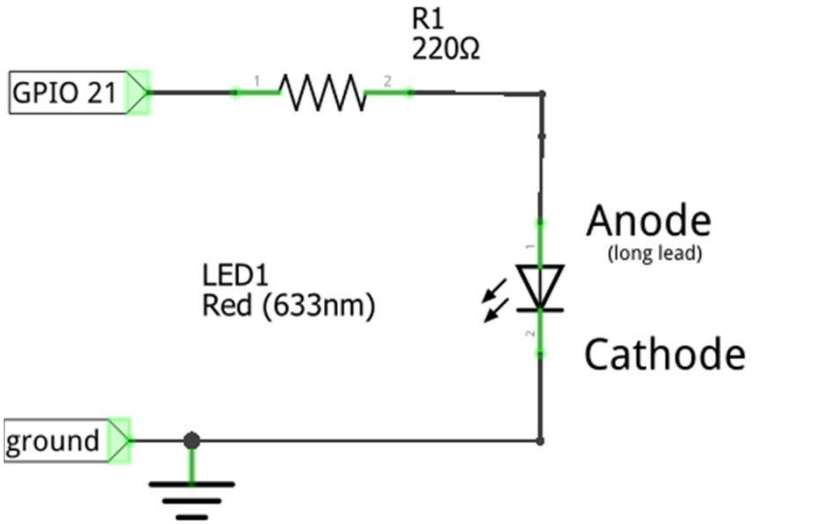
Projects – Blink LED

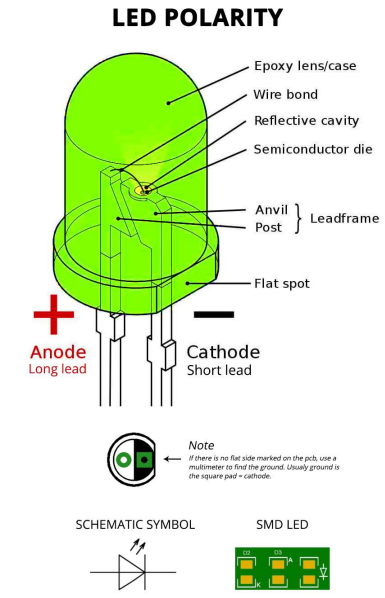
- Blink LED
  - Hook up LED circuit (see subsequent slide)
  - Programming->Thonny Python IDE
  - File->New
  - Type in code found in subsequent slide
  - Thonny will ask for file name and location when running for first time. Save to Maker/Documents/RaspberryPiClass/my_blink.py

This program blinks a led connected to GPIO21 (board pin 40) forever, on for 1 second and off for 1 second. It uses the gpiozero library.

Blink LED Hookup







Blink LED Code

```
from gpiozero import LED
from time import sleep

light = LED(21)

while True:
    light.on()
    sleep(1)
    light.off()
    sleep(1)
```

Projects – Safe Blink

- Safely blink LED
  - Load Thonny and load Maker/Documents/RaspberryPiClass/safe_blink.py

  This program demonstrates GPIO cautions that should be added to each program to put the GPIO into a safe state when the program is exited. It also demonstrates the Rpi.gpio library.
- Shutdown scenarios that should put the GPIO pin in a safe state:
  - Ctrl-C
  - pkill
  - Parent close

Safe Blink Code

```
#! /usr/bin/python3
# specify the python interpreter

import RPi.GPIO as GPIO
from time import sleep
from signal import signal, SIGTERM, SIGHUP

led_pin = 21 # use a variable for the pin number

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)     # Use Broadcom (processor) pin numbering
# Set pin to be an output pin and set initial value to low (off)
GPIO.setup(led_pin,GPIO.OUT,initial=GPIO.LOW)

def safe_exit(signum, frame):
    exit()

try:
    signal(SIGTERM, safe_exit) # handle pkill
    signal(SIGHUP, safe_exit)  # handle parent close

    while True: # Run forever
        GPIO.output(led_pin, GPIO.HIGH) # Turn on
        sleep(2)                        # Sleep for 2 seconds
        GPIO.output(led_pin, GPIO.LOW)  # Turn off
        sleep(1)                        # Sleep for 1 second

# Prevent Traceback warning when using Ctrl-C to exit the program
except KeyboardInterrupt:
    pass

# Cleanup if program closes due to a signal
finally:
    GPIO.cleanup()
```

Projects – Fade LED

- Fade LED
  - Load Thonny and load Maker/Documents/RaspberryPiClass/fade_led.py

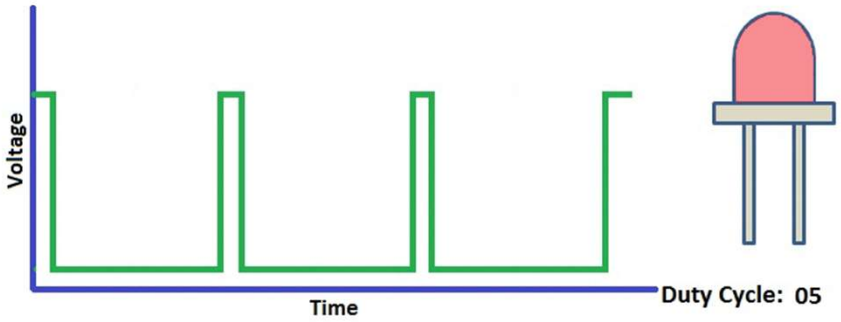
  This program fades a led connected to GPIO21 (board pin 40) forever. The default fade-in time is 1 sec and fade-out time is 1 second. The fade time can be adjusted using parameters fade-in-time=<sec> and fade-out-time=<sec>. It uses the gpiozero library.

  Also shown is the close() function that cleans up the GPIO similar to the GPIO.cleanup() function in the RPi.GPIO libarary

Fade LED Code

```
from signal import pause
from gpiozero import PWMLED

light = PWMLED(21) # create a PWMLED object called "led"

try:
    light.pulse()  # blink the LED with fade-in and fade-out time
    pause          # pause the program

except KeyboardInterrupt:  # capture CTRL-C to prevent warning
    pass                   # no operation

finally:
    light.close()  # clean up the GPIO
```

Fade LED PWM



Cautions

- Shutdown
  - Never just disconnect power from the Raspberry Pi
    - The Linux operating system continuously accesses (reads/writes) the SD card
    - Interrupting power in the middle of a write may corrupt the SD card
  - Shutdown the Raspberry Pi by:
    - Pi->Logout->Shutdown
    - Wait for green LED to stop blinking (about 10 seconds)
    - Remove power
- GPIO
  - The Raspberry Pi GPIO pins work with 3.3V logic levels and are not 5V tolerant. If you apply 5V to a GPIO pin you risk permanently damaging it
  - Use GPIO programming cautions to avoid leaving GPIO pins in a unsafe state after the program terminates
    - RPi.GPIO: GPIO.Cleanup()
    - GPIOZERO: <module object>.close()

Advanced Projects – Log Fade LED

- Fade LED with log algorithm
  - Use same circuit as Blink LED
  - Execute Thonny and load Maker/Documents/RaspberryPiClass/log_fade_led.py

This program will fade the brightness of a LED up and down in 10 steps. The brightness is adjusted using the PWMLED class of the gpiozero library. The PWMLED class sets the value of the LED to a number in the range of 0 to 1. This is called duty cycle with 0 being off and 1 being full on. Three algorithms are used to vary the brightness. Experiment with the algorithms and number of steps and observe the perceived brightness. A complete explanation can be found at: https://www.youtube.com/watch?v=1fKH5PU9eck&t=9s

Log Fade LED Code

```
from time import sleep
from gpiozero import PWMLED
from math import log10

light = PWMLED(21)

loop_delay = 0.2  # delay in seconds between each brightness change

# three brightness change algorithms
algorithms = ["linear", "power_2", "logarithmic"]
algorithm = algorithms[0]

steps = 10     # number of brigntness steps
fade_factor = (steps * log10(2))/(log10(steps))

led_min = 0          # minimum led brightness
led_max = 100         # maximum led brightness
led_step = 100/steps # brightness change

def brightness(level):
    if algorithm == "power_2":
        return pow(2, level/10)/1024
    elif algorithm == "logarithmic":
        return (pow(2, ((level/led_step)/fade_factor))-1)/steps
    else:
        # default to linear
        return level/100

try:
    while True:                        # loop forever
        for level in range (led_min,led_max+1,int(led_step)):  # variable x
incremented from min to max by step
            bright = brightness(level) # calculate new duty cycle
            print("increasing:", level, bright*100) # print for debug
            light.value = bright       # change duty cycle to vary the brightness
of LED.
            sleep(loop_delay)          # sleep in seconds

        for level in range (led_max,led_min-1,int(-led_step)): # variable x
decremented from max to min by step
            bright = brightness(level) # calculate new duty cycle
            print("decreasing:", level, bright*100) # print for debug
            light.value = bright       # change duty cycle to vary the brightness
of LED.
            sleep(loop_delay)          # sleep in seconds

except KeyboardInterrupt:
    pass

finally:
    light.close()
```

Advanced Projects – Blink with Timer

- Blink LED with Timer
  - Execute Thonny and load Maker/Documents/RaspberryPiClass/timer_blink.py

This starts by executing a function blink(). blink() starts a timer with a 1 second timeout and specifies that the same function, blink() should be called when the timer expires. The program then pauses for keyboard input. The timer does all the work in the background. When the program exits a call to sys.exit() is necessary to stop the timer, otherwise the timer will keep running and the LED will continue to blink.

Advanced Projects – Blink Remotely

- Blink LED remotely
  - As user pi:
    - Enable Remote GPIO in Raspberry Pi Preferences->Raspberry Pi Configuration->Interfaces
    - Automate running pigpiod daemon at boot time: `sudo systemctl enable pigpiod`
    - Run the daemon once using systemctl: `sudo systemctl start pigpiod`
    - Or to run the daemon manually: `sudo pigpiod`
  - Install https://thonny.org/ on Windows
  - Install gpiozero using Thonny on Windows
  - Install pigpio using Thonny on Windows
  - Run remote_blink.py

Advanced Projects – Blink via Browser

- Blink LED via Web Browser
  - Execute Thonny and load Maker/Documents/RaspberryPiClass/simple_webserver.py
  - Enter the PI's IP address for the variable host name and run the program
  - Point browser to IP address, port 8000
  - Press On or Off
  - Reference: https://github.com/e-tinkers/simple_httpserver

This is a simple webserver and will probably not scale well. Some other options are at:

- Apache and php: https://www.raspberrypi.org/documentation/remote-access/web-server/apache.md
- Flask: https://projects.raspberrypi.org/en/projects/python-web-server-with-flask/1

Advanced Projects – Blink via Button

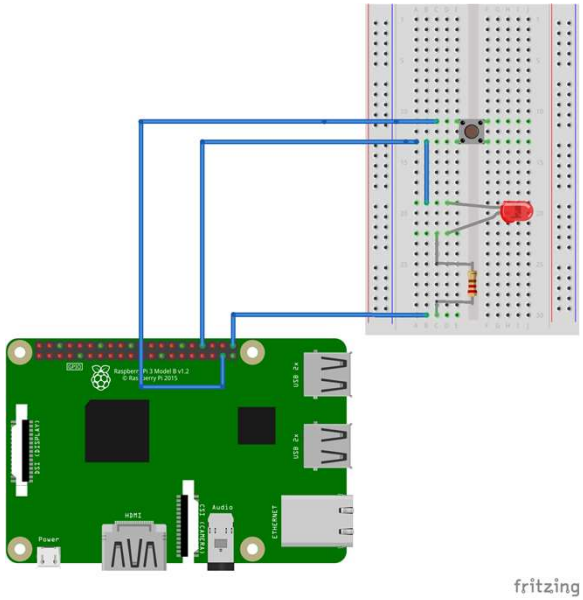
- Blink LED via Button
  - Execute Thonny and load Maker/Documents/RaspberryPiClass/button_blink.py
  - Run button_blink.py

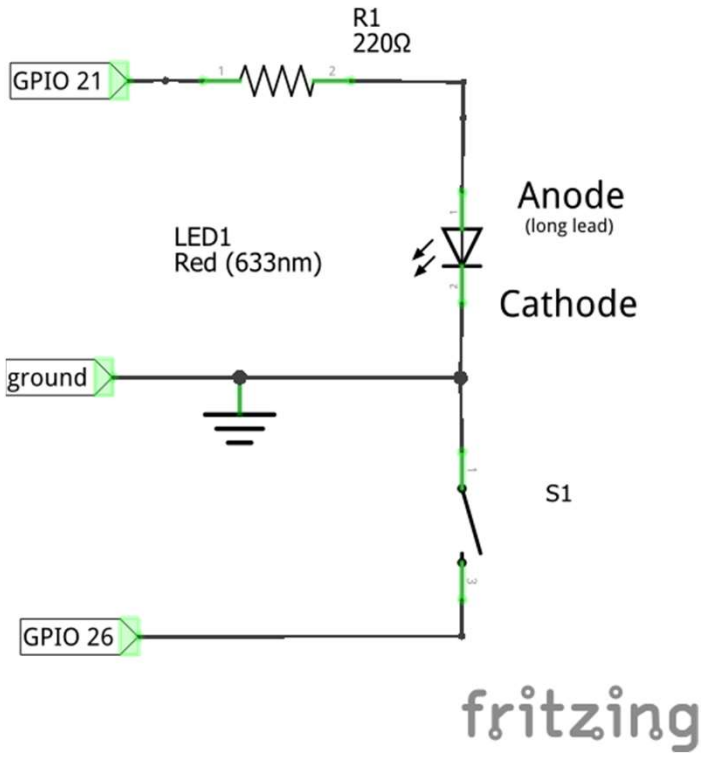
  This program will wait for a button press. When pressed the LED will light for three seconds and then turn off.
- For more information on the gpiozero Button class:

  https://gpiozero.readthedocs.io/en/stable/api_input.html#
- For a simple debounce technique:

  https://roboticadiy.com/how-to-debounce-push-button-in-raspberry-pi-4/

Blink LED via Button Hookup





Advanced Projects – Blink via Blue Dot

- Blue Dot is an Android app and Python library that allows you to wirelessly control your Python projects such as a light switch, remote camera, robot, etc.
- Install the Blue Dot app on your Android device from the Google Play Store: https://play.google.com/store/apps/details?id=com.stuffaboutcode.bluedot
- Install Blue Dot on the Raspberry Pi: sudo pip3 install bluedot
- Pair the Android device with the Raspberry Pi:
  - On your Android phone:
    1. Open Settings
    2. Select Bluetooth and make your phone "discoverable"
  - On your Raspberry Pi:
    1. Click Bluetooth ▸ Turn On Bluetooth (if it's off)
    2. Click Bluetooth ▸ Make Discoverable
    3. Click Bluetooth ▸ Add Device
    4. Your phone will appear in the list, select it and click **Pair**
  - On your Android phone and Raspberry Pi.
    1. Confirm the pairing code matches
    2. Click OK
- On the Raspberry Pi execute Thonny and load and execute the demo program blue_dot_led.py
- On the Android device open the Blue Dot app and connect to the paired device. Press the dot to blink the led.

Handy Linux commands

- Get ip address of Pi: `hostname -I`
- Change permission of a file to executable: `chmod 755 <file>`
- Get the Raspberry Pi OS version: `cat /etc/os-release`
- Get Raspberry Pi GPIO pinout: `pinout`
- Update Raspberry Pi OS
  - Update system's package list: `sudo apt-get update`
  - Upgrade installed packages: `sudo apt-get upgrade`
- Command line configuration: `sudo raspi-config`
- Check disk usage: `df -h`
- Check GPIO status: `gpio readall` (`raspi-gpio get` on Raspberry Pi 4)

Install Raspberry Pi OS

- For full instructions go to: https://www.raspberrypi.org/documentation/installation/installing-images/
  - Download Raspberry Pi Imager and install it: https://www.raspberrypi.org/software/
  - Run Imager and Choose OS and SD Card to write to (make sure you are choosing the correct driver letter for the SD Card.
  - Write, All existing data on 'SDHC Card' will be erased. Yes to continue
  - After a few minutes a message will indicate that the OS is written to the SD card
  - Remove the SD card from the PC and install into the Raspberry Pi.
  - Power up the Raspberry Pi and go through the welcome setup screens
  - The default user is 'pi' and password is 'raspberry'. It is recommend to change the password
  - Setup your wifi network
  - The OS will be updated to latest