Introduction to StemX

- Benefits of StemX
  - Develop STEM Skills
  - Build Soft Skills
- What are we going to do?
  - Hardware Assembly
  - Engineer Solutions
  - Hands On Builds
  - Learn Python
  - Integrate AI
  - Much More!

Why Should I pursue STEM education?

<u>High demand</u>:

- The demand in government agencies is constantly increasing.
- Government requires large numbers of professionals with technical expertise.
- Approximately 367,000 federal employees classified as having a "STEM" occupational series.
- Wide range of positions, STEM is applicable across ALL industry

SCHOLARSHIPS!

https://www.dodstem.us/participate/opportunities/?type=Scholarship

<u>Competitive government salaries</u>:

- Government agencies offer competitive salaries for STEM professionals.
- Base government salary entry-level positions ranges $40,000 to $80,000 per year.
- The salary for individuals with STEM degrees in government jobs is higher than many other fields, making it a lucrative career path.

<u>Benefits:</u>

- Follow your passion: Individuals with STEM education can find a job that suits their interests and passions!
- Impactful work: Government service provides opportunities to make positive impacts on society. Many focused on addressing critical issues, such as climate change, national security, and public health, safety and defense.
- Job security: Government jobs offer more job security than private sector jobs, with benefits such as retirement plans and health insurance. Pursuing STEM education for government service can provide long-term job stability and security.

"The future of the economy is in STEM," says James Brown, the executive director of the STEM Education Coalition in Washington, D.C. "That's where the jobs of tomorrow will be."

Employment in STEM occupations

Other available formats: (XLSX)

Table 1.11 Employment in STEM occupations, 2021 and projected 2031
(Numbers in thousands)

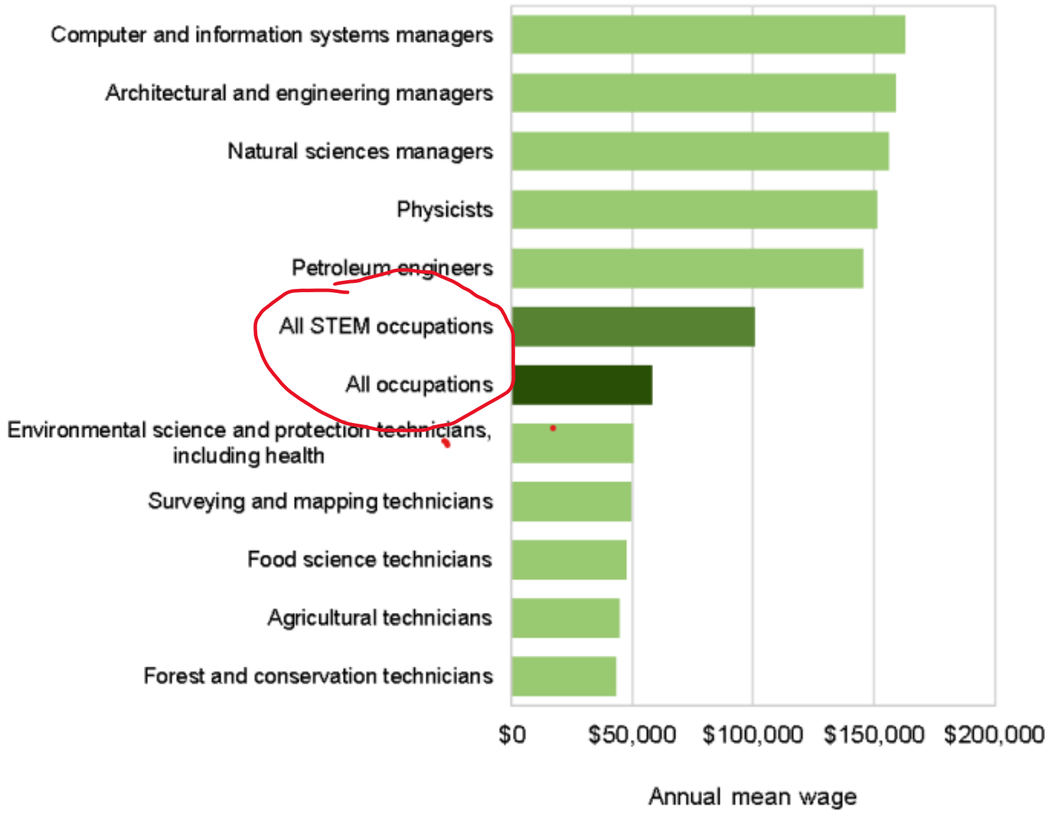
Occupation category	Employment, 2021	Employment, 2031	Employment change, 2021–31	Percent employment change, 2021–31	Median annual wage, 2021(1)
Total, all occupations	158,134.7	166,452.1	8,317.4	5.3	$45,760
STEM occupations(2)	9,880.2	10,944.2	1,064.0	10.8	$95,420
Non-STEM occupations	148,254.5	155,508.0	7,253.5	4.9	$40,120

Note:

- Projected growth at 10%, much better than other occupations at large
- Salary expected twice as much!

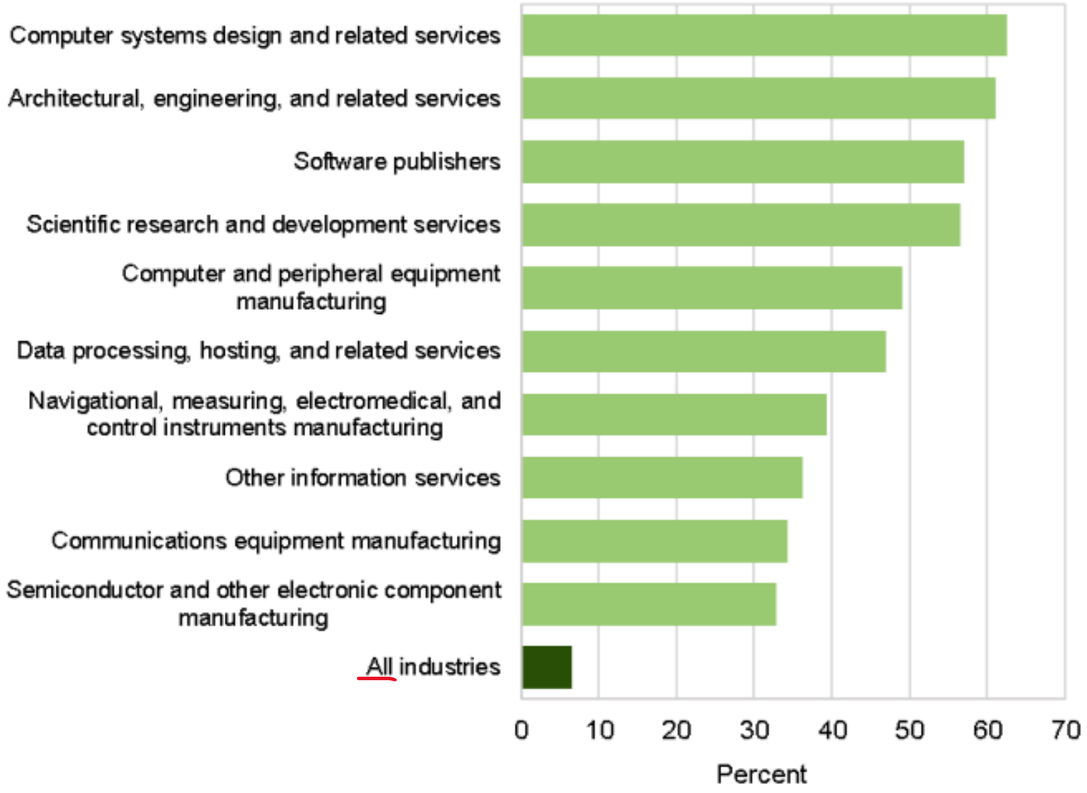
Courtesy: US Bureau of Labor Statistics

Highest and lowest paying STEM occupations, May 2021



Source: U.S. Bureau of Labor Statistics, Occupational Employment and Wage Statistics.

Industries with the highest employment shares of STEM occupations, May 2021



Source: U.S. Bureau of Labor Statistics, Occupational Employment and Wage Statistics.

Government STEM job roles:

DoD Stem career profiles https://dodstem.us/meet/

Dr. Abigail Juhl
Materials Research Engineer
Materials and Manufacturing Directorate, Air Force Research Laboratory

Dr. Bryn Adams
Synthetic Biologist and Environmental Microbiologist
DEVCOM, Army Research Laboratory

Lauren Ejiaga **https://dodstem.us/meet/professionals-and-alumni/**
Student Scientist
2019 DoD STEM Talent Award Winner, Broadcom MASTERS

There are many areas of government that need STEM talent!
DHS Science and Technology Directorate

Brannan Villee, Senior Program Director (~1:54)

https://www.youtube.com/watch?v=Hq8UINLIud8&list=PLPaVPPbYHq_juZ-PC9KHBMulQqBMHzHLi&index=2

Talking points: What is critical infrastructure?

"The Department of Defense needs talented folks like you to solve complex problems that our scientists and engineers are faced with every day to ensure that our nation remains safe."

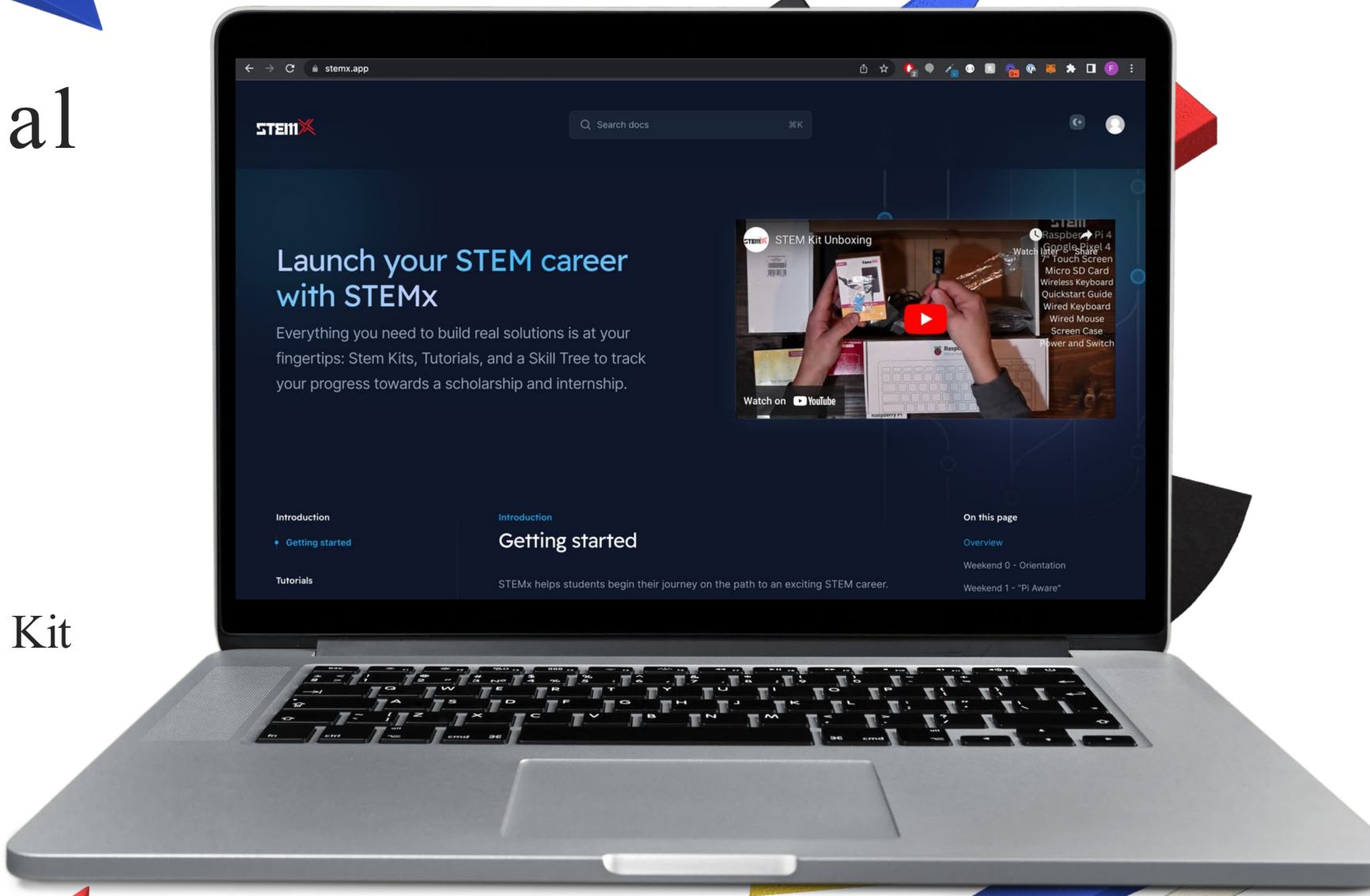
Louie Lopez
Director, DoD STEM

300,000 scientists

The Federal Government is the largest employer of STEM workers in the United States, depending on more than 300,000 scientists and engineers across the Defense Laboratory Enterprise to meet national defense challenges and modernization priorities.

StemX Webportal at stemx.app

- Tutorials
- Modules
- Events and Challenges
- Android Team Awareness Kit
- ATAK Plugin Design and Development



STEMx
What You Need to Know

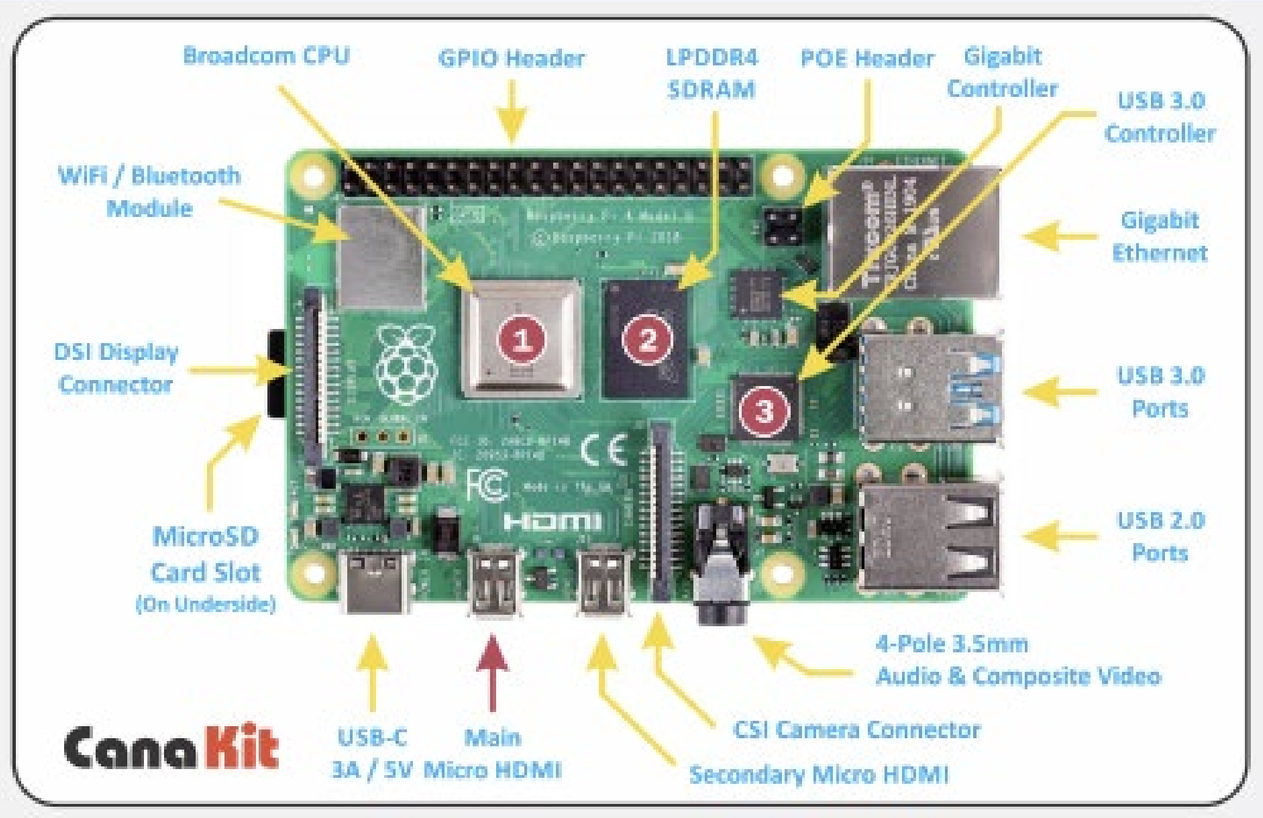
Understand how
your STEMx STEM Kit works

File folder on desktop: Stemx/Python_Basics/

Your STEMx Kit

- Raspberry Pi 4b
- Google Pixel
- 7" Touch Screen
- Micro SD Card
- Wireless Keyboard
- QuickStart Guide
- Wired Mouse
- Screen Case
- Power and Switch
- Solar Battery
- HDMI Cable
- Software Defined Radio

Raspberry PI 4



Raspberry Pi Website:
https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

Download **Raspberry Pi Beginner's Guide 4th Edition**
(bottom of page, PDF)

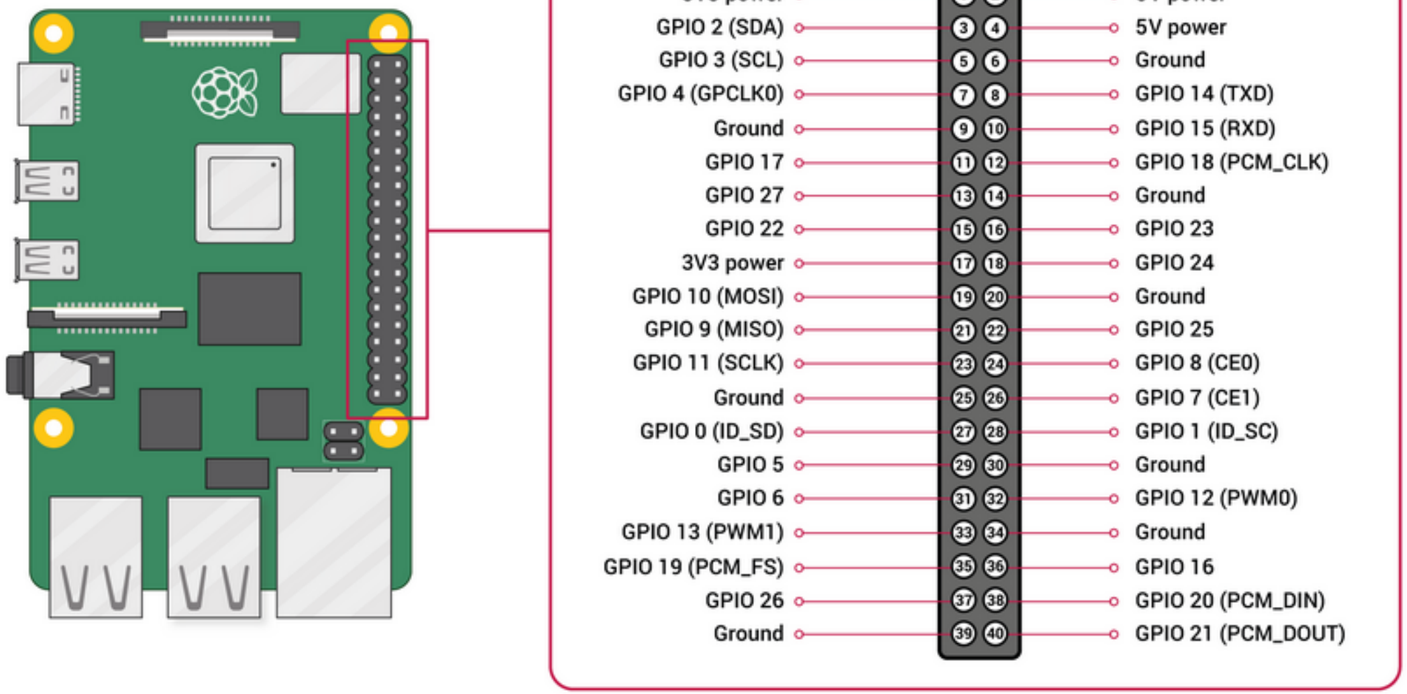
Check out the documentation

Check out the user forums and projects

Lab:
Power on using USB-C and touchscreen
Connect to WiFi (note: To set the country code, open the Raspberry Pi Configuration application from the Preferences Menu, select **Localisation** and set the appropriate code.

Lab:
Attach to lab monitors, keyboard and mouse

Raspberry PI 4

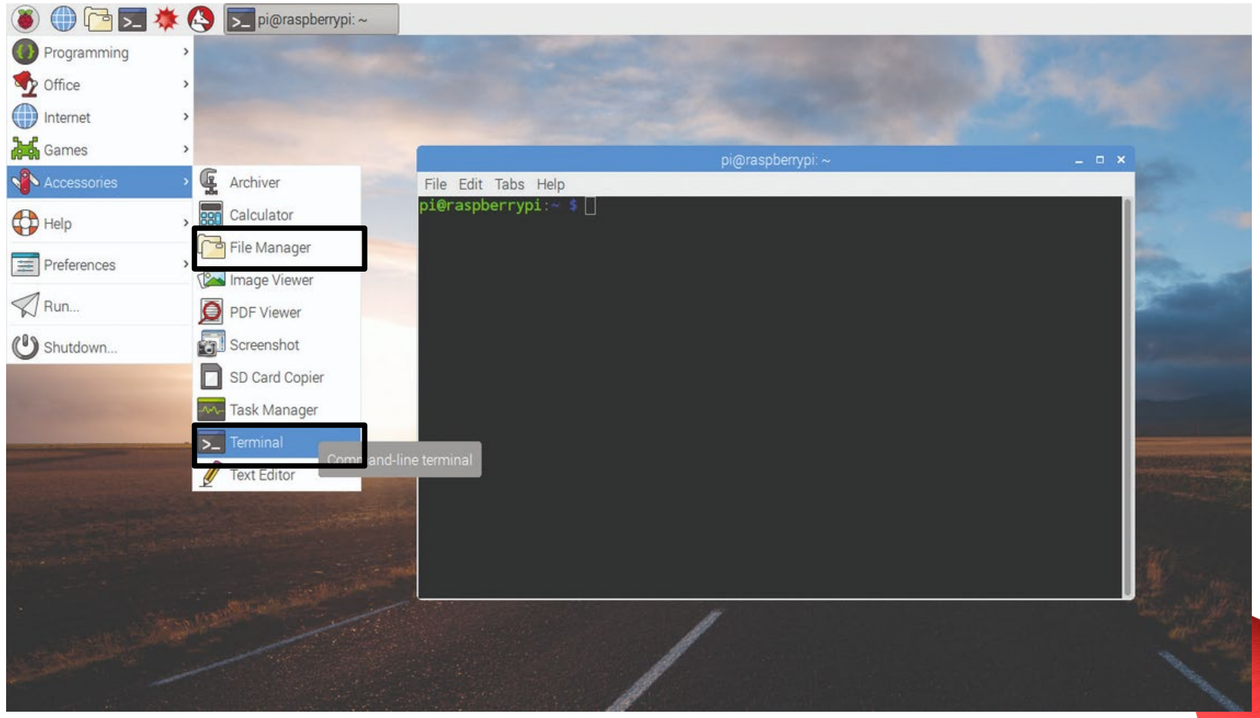


Lab:
Use the GPIO to ‘turn off’ touchscreen
Notice how primary monitor is now desktop computer

We will do a basic circuit after we review IoT

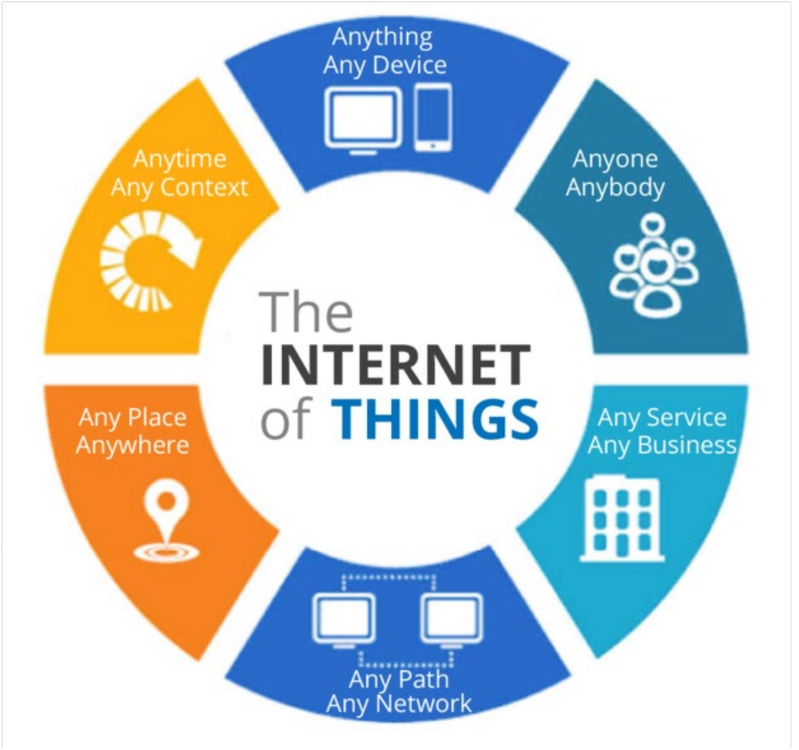
Interact with your Pi

- Graphical User Interface (GUI)
  - File Manager
- Command Line Interface (CLI)
  - Terminal
  - STEMx folder on desktop



What is ‘IoT’

The IoT can be described as an extension of the internet and other network connections to different sensors and devices — or “things” — affording even simple objects, such as lightbulbs, locks, and vents, a higher degree of computing and analytical capabilities.



Internet of Things Uses By Industry

HOME
- Smart Temperature Control
- Optimized Energy Use

INDUSTRIAL
- Machine-to-Machine Communication
- Quality Control

AUTOMOTIVE
- Vehicle Auto-Diagnosis
- Optimized Traffic Flow
- Smart Parking

AGRICULTURE
- Offspring Care
- Crop Management
- Soil Analysis

MILITARY
- Situational Awareness
- Threat Analysis

MEDICAL
- Optimized Patient Care
- Wearable Fitness Devices
- Quality Data Reporting

ENVIRONMENTAL
- Forest Fire Detection
- Species Tracking
- Weather Prediction

RETAIL
- Theft Protection
- Inventory Control
- Focused Marketing

Careers at Fortune 500 Companies…

ThermoFisher SCIENTIFIC

aws

CORNING

Panasonic

DATADOG

IBM

CISCO

AMD

The global IoT market is projected to grow from $478.36 billion in 2022 to $2,465.26 billion by 2029![1]

To mid-cap and small business, too!

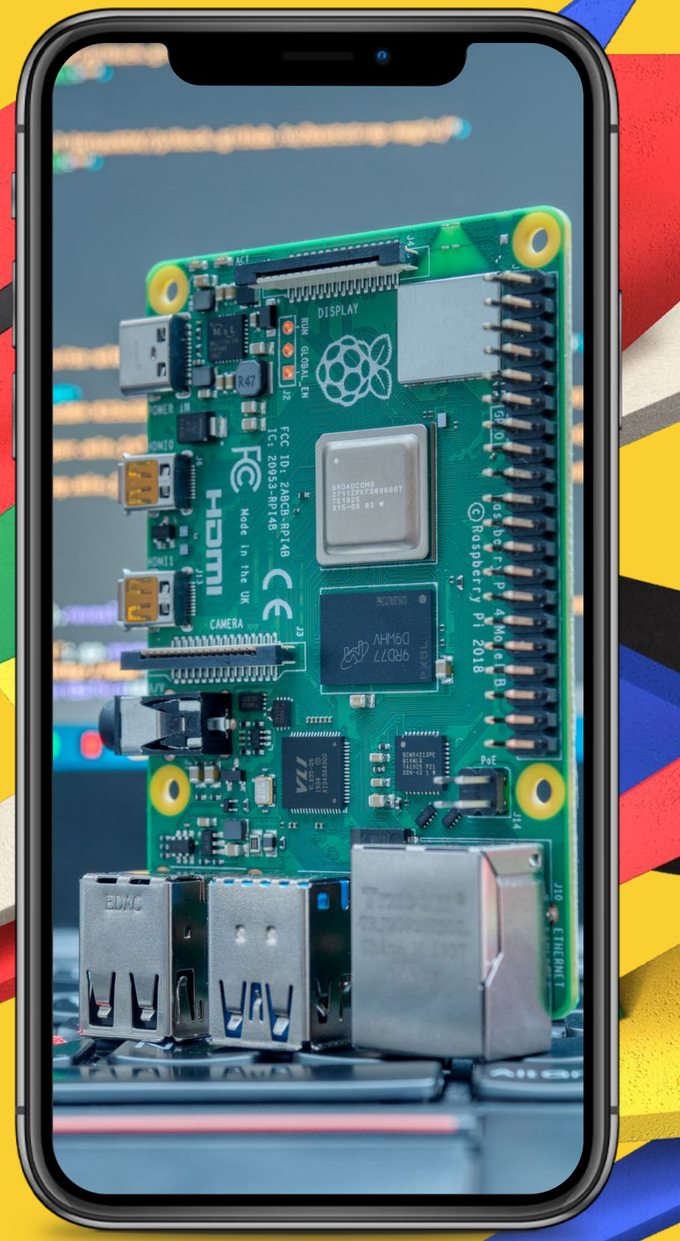
Click on each company logo, to learn more about their IoT offerings and services!

Total IoT installed base worldwide projected to grow from 13.8 billion devices in 2021 to over 30 billion units by 2025![2]

[1] https://www.fortunebusinessinsights.com/industry-reports/internet-of-things-iot-market-100307

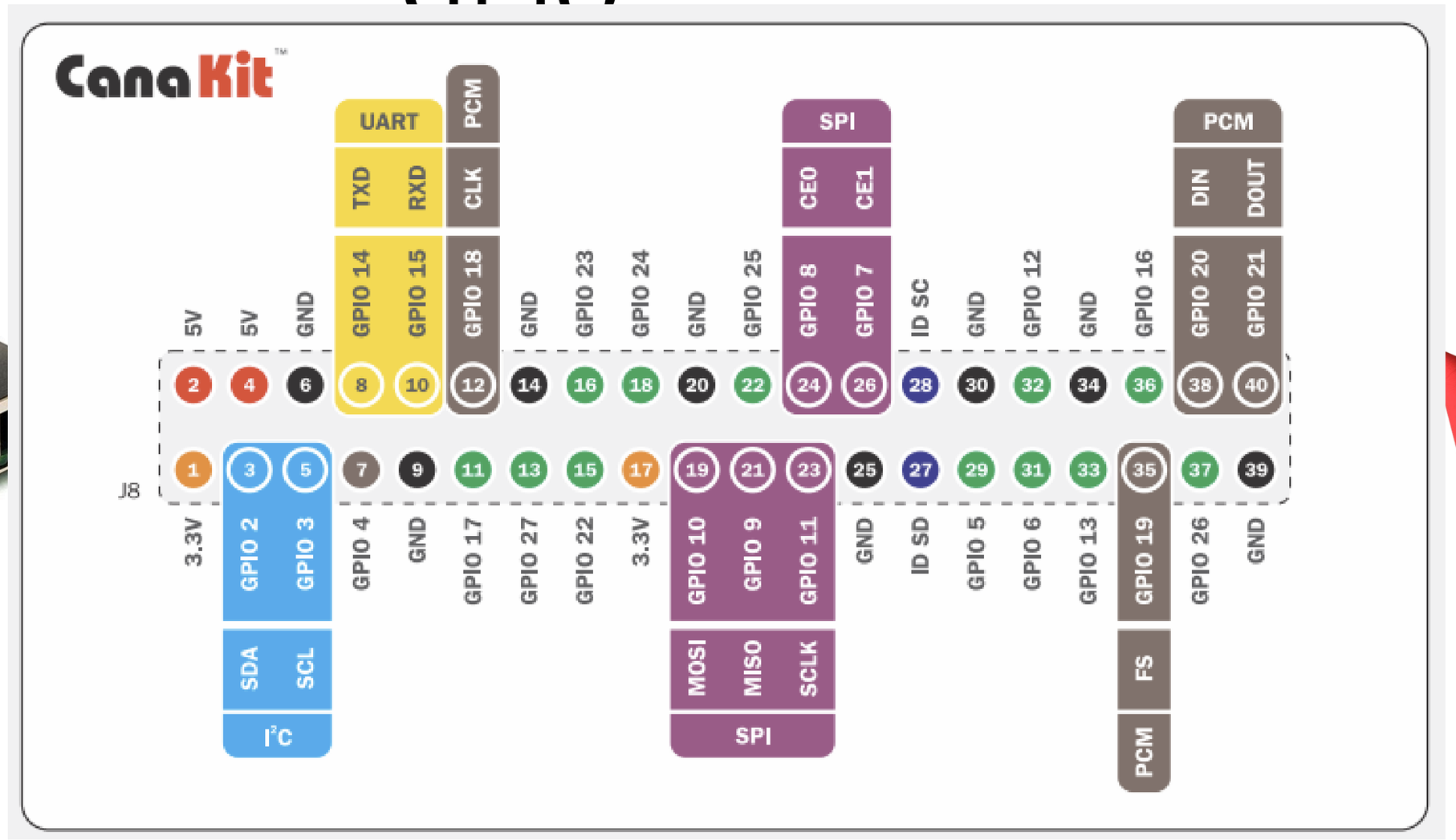
[2] https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide/

General Purpose Input/Output (GPIO) Refresh



STEMX

GPIO


CanaKit
J8
3.3V
5V
GPIO 2
5V
GPIO 3
GND
GPIO 4
GPIO 14
GND
GPIO 15
GPIO 17
GPIO 18
GPIO 27
GND
GPIO 22
GPIO 23
3.3V
GPIO 24
GPIO 10
GND
GPIO 9
GPIO 25
GPIO 11
GPIO 8
GND
GPIO 7
ID SD
ID SC
GPIO 5
GND
GPIO 6
GPIO 12
GPIO 13
GND
GPIO 19
GPIO 16
GPIO 26
GPIO 20
GND
GPIO 21
SDA
SCL
I²C
TXD
RXD
UART
MOSI
MISO
SCLK
SPI
CLK
PCM
CE0
CE1
SPI
FS
PCM
DIN
DOUT
PCM
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40

Blinking LED

File location: Stemx/GPIO/blinking_led.py

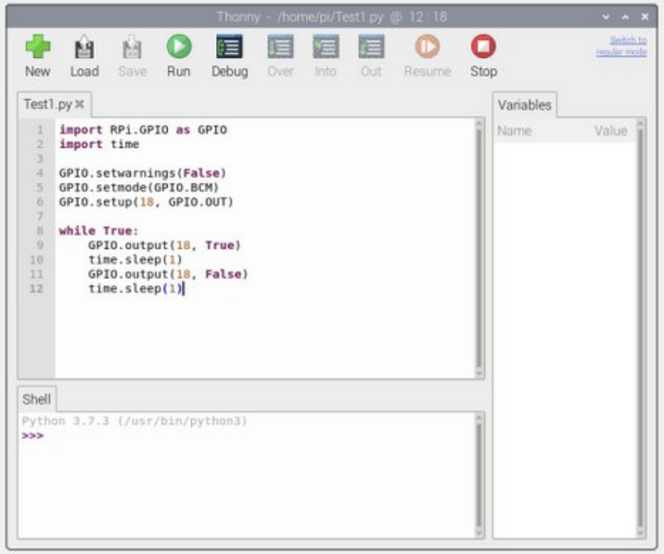
GPIO Port and Python

The General Purpose Input/Output (GPIO) port of the Raspberry Pi can be controlled in various ways but the examples in this guide will use the Python 2 programming language. In order to run the code for each example, follow these steps:

1. From the main menu in Raspbian, choose Programming -> Thonny Python IDE.

2. In the main code area, type the example code exactly as it appears. Note that the Python language is case sensitive, so ensure every character is typed exactly as shown in each example.

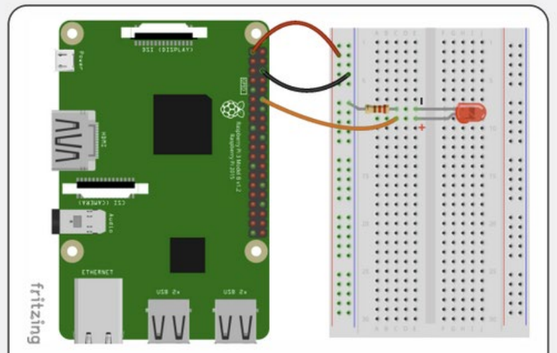
3. Save your file by clicking the "Save" button and finally click "Run" to run your code. If there were no errors in your code, the program will now be executed.



Blinking an LED

To blink an LED, use three male-to-female jumper wires and a 220 Ohm resistor (red, red, brown) to connect the LED to the GPIO port as shown below.

Note that it is important for the LED to be connected with correct polarity or it will not light up and you may damage the LED. The longer leg of an LED is called the Anode (+) and the shorter leg is called the Cathode (–). In this example, the shorter leg (Cathode) is to be connected to the resistor.



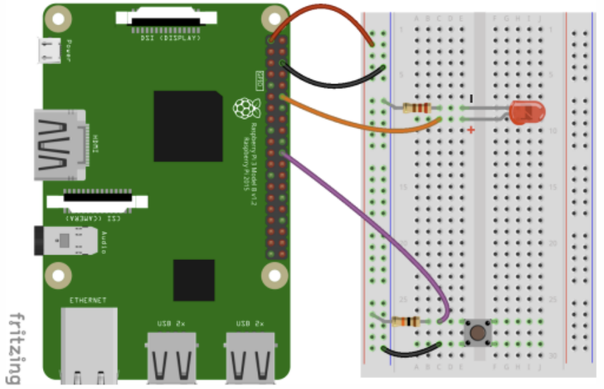

```
import RPi.GPIO as GPIO
import time

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)

while True:
      GPIO.output(18, True)
      time.sleep(1)
      GPIO.output(18, False)
      time.sleep(1)
```


Example available in CanaKit Guide that came with your kit!

LED With A Button

Controlling the LED with a Button

This example builds upon the previous example by adding a push-button switch that will control the LED. Use an additional male-to-female jumper wire, a male-to-male jumper wire, and a 10K Ohm resistor (brown, black, orange) to connect the push-button switch to the GPIO port.




```
import RPi.GPIO as GPIO
import time

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)
GPIO.setup(25, GPIO.IN)

while True:
      if GPIO.input(25):
            GPIO.output(18, False)
      else:
            GPIO.output(18, True)
```

For this challenge, make a copy of the blinking_led.py and name it led_button.py.

Edit the code to fulfill this example and wired your components to test!

Example available in CanaKit Guide that came with your kit!

Python Basics

Variables: A variable is a name given to a memory location that holds a value. Variables in Python can hold any data type, such as integers, floating-point numbers, strings, etc.

Data Types: Python supports several built-in data types such as integers, floating-point numbers, strings, booleans, and more.

Operators: Python supports various types of operators such as arithmetic, comparison, logical, assignment, and more.

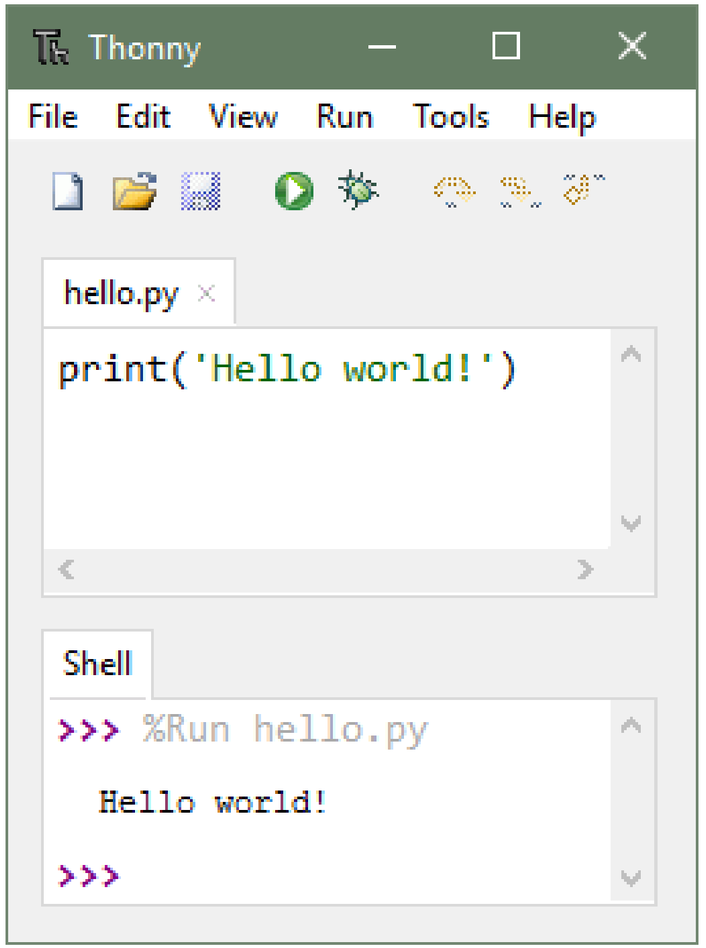
Conditional Statements: Conditional statements are used to execute different code based on whether a condition is true or false. Python has two types of conditional statements, if-else statements and switch statements.

Loops: Loops are used to execute a block of code repeatedly until a certain condition is met. Python supports two types of loops, for loop and while loop.

Functions: Functions are blocks of code that perform a specific task. In Python, functions are defined using the def keyword.

Integrated Development Environment (IDE)

- Click on the Raspberry Pi menu >> Programming >> Thonny to open up the IDE



https://learn.sparkfun.com/tutorials/python-programming-tutorial-getting-started-with-the-raspberry-pi/hello-

Challenge 1: Create a Python file from Scratch

- Objective: Write a simple Python script that prints 1 line of text (ex. “Hello World”) and save it onto the Raspberry Pi.

- Hints:
  - Python files are always recognized by the ending “. py”
  - IDE’s are primarily written to assist developers in coding creation

Command Line

•The command line, also known as terminal, shell, or console, is a text-based interface used to interact with the operating system.
•It allows users to run different commands, navigate through the file system, or control a Linux computer without a graphical interface.

Basic Command Functions:

- `ls`: **l**ist **s** all files in current directory
- `cd`: **c**hanges the current **d**irectory
- `pwd `: **p**rints the **w**orking **d**irectory

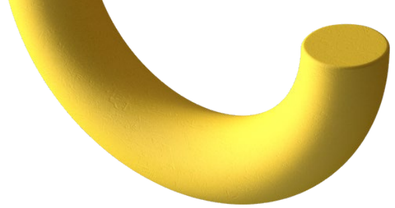
```
colbysawyer@Flagship:~/test$ ls
folder1
colbysawyer@Flagship:~/test$ cd folder1
colbysawyer@Flagship:~/test/folder1$ ls
colbysawyer@Flagship:~/test/folder1$ pwd
/home/colbysawyer/test/folder1
colbysawyer@Flagship:~/test/folder1$
```

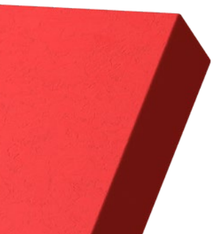
https://learn.sparkfun.com/tutorials/python-programming-tutorial-getting-started-with-the-raspberry-pi/hello-

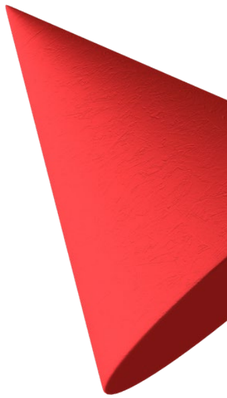
Challenge 2: Run your Python file using the Command Line

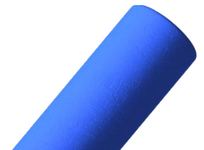
- Objective: Run your previously created Python file using only the command line, NOT THONNY.

- Hints:
  - To change directories (folders) use the “cd” command









Variables

File location: Stemx/Python_Basics/variables.py

In this example, we first assign values to three variables: name, age, and is_student. We then print out the values of these variables using the print() function.

Next, we modify the variables by assigning new values to name and is_student, and incrementing the value of age by 1 using the += operator. We then print out the modified values of the variables.

```
# Assigning values to variables
name = "John"
age = 25
is_student = True

# Printing out the variables
print("Name:", name)
print("Age:", age)
print("Is Student:", is_student)

# Modifying the variables
name = "Jane"
age += 1
is_student = False

# Printing out the modified variables
print("Name:", name)
print("Age:", age)
print("Is Student:", is_student)
```

Notice that in this code there are multiple slightly grey lines preceded by a “#”. This denotes a comment line that will be ignored by the program when running. This is vitally important to help developers understand each others code.

Does anyone know how to comment multiple lines?

Input/Output

In this example, we print a message to the user and than receive their feedback.

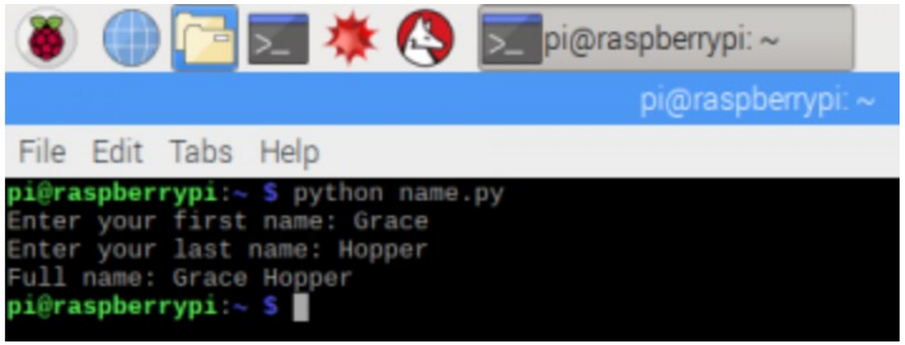
main.py > ...

```python
# Print a statement
print("Please enter a string:")

# Accept a string input
user_input = input()

# Print the received input
print(f"You entered: {user_input}")

```

Challenge 3: Write a Concatenation Script

- Objective: Write a program “name.py” that asks for the user's first name and last name (two separate input() calls) and then prints the user's first and last name on one line

- Hints:
  - Input and Output will be necessary to interact with the user



https://learn.sparkfun.com/tutorials/python-programming-tutorial-getting-started-with-the-raspberry-pi/hello-

Data Types

File location: Stemx/Python_Basics/datatypes.py

In this example, we create variables of different data types and perform operations on them.

We use integers to perform addition, floats to perform multiplication, strings to concatenate with each other, and booleans to store true/false values.

```
# Integers
x = 10
y = 20
sum = x + y
print("Sum of x and y:", sum)

# Floats
a = 2.5
b = 1.5
product = a * b
print("Product of a and b:", product)

# Strings
name = "John"
greeting = "Hello, " + name + "!"
print(greeting)

# Booleans
is_adult = True
is_student = False
print("Is adult:", is_adult)
print("Is student:", is_student)
```

Type Casting

What if we wanted to be SURE that a certain variable was a specific data type? Any guesses on how we could use Python to do so?

Type Casting

Type Functions that come with Python:

- Int(): Converts to Integer
- Float(): Converts to Float
- Str(): Converts to String
- Bool(): Converts to Boolean (0 = False, Any non-zero, non-empty values = True)
- List(): Converts to a List
- Dict(): Converts to a Dictionary
- Bytes(): Converts to Bytes Object

main.py > ...

```python
#Define Variables
species = "Cat"
isVaccinated = 1  # Remeber that 1 is True and 0 is False in Python
age = 3.5
weight = 4

# Print Variables
print(species)
print(isVaccinated)
print(age)
print(weight)

# Typecast Variables

# If we wanted to be sure that the Cat's age was an integer, we could use the int() function to convert it.
ageInteger = int(age)

# If we wanted to be sure that the Cat's age was a string, we could use the str() function to convert it.'
ageString = str(age)

# If we wanted to be sure that the Cat's weight was a float, we could use the float() function to convert it.
weightFloat = float(weight)

# If we wanted to be sure that the isVacinated variable was a boolean, we could use the bool() function to convert it.
isVaccinated = bool(isVaccinated)


# Print Typecast Variables
print(ageInteger)
print(ageString)
print(weightFloat)
print(isVaccinated)
```

STEMX

Operators

File location: Stemx/Python_Basics/operators.py

In this example, we use arithmetic operators to perform mathematical calculations, comparison operators to compare two values, and logical operators to combine boolean expressions.

```python
# Arithmetic Operators
x = 10
y = 3
print("Sum of x and y:", x + y)
print("Difference of x and y:", x - y)
print("Product of x and y:", x * y)
print("Quotient of x and y:", x / y)
print("Remainder of x divided by y:", x % y)
print("x raised to the power of y:", x ** y)

# Comparison Operators
a = 5
b = 10
print("Is a equal to b?", a == b)
print("Is a not equal to b?", a != b)
print("Is a greater than b?", a > b)
print("Is a less than or equal to b?", a <= b)

# Logical Operators
is_adult = True
is_student = False
print("Is the person an adult and not a student?", is_adult and not is_student)
print("Is the person either an adult or a student?", is_adult or is_student)
```

Challenge 4: Write a Voter Eligibility Script

- Objective: Write a program that asks the user to enter their age and checks whether they are eligible to vote in the elections.

- If the user's age is 18 or above, print out a message that says "You are eligible to vote." Otherwise, print out a message that says "You are not eligible to vote."

https://learn.sparkfun.com/tutorials/python-programming-tutorial-getting-started-with-the-raspberry-pi/hello-

Operators

Mathematical operators perform basic math operations on numbers:

Operator	Description	Example
`+`	Adds two numbers	`2 + 3` returns `5`
`-`	Subtracts one number from another	`8 - 5` returns `3`
`*`	Multiplies two numbers together	`4 * 6` returns `24`
`**`	Raises the first number to the power of the second number	`2 ** 4` returns `16`
`/`	Divides the first number by the second number	`5 / 4` returns `1.25`
`//`	Divides the two numbers and rounds down to the nearest integer (divide and floor)	`5 / 4` returns `1`
`%`	Divides the first number by the second number and gives the remainder (modulo)	`19 % 8` returns `3`

Operators

Logical operators compare two numbers and returns one of the *Boolean* values: `True` or `False`.

Operator	Description	Example
`<`	`True` if the first number is less than the second, `False` otherwise	`5 < 3` returns `False`
`>`	`True` if the first number is greater than the second, `False` otherwise	`5 > 3` returns `True`
`<=`	`True` if the first number is equal to or less than the second, `False` otherwise	`2 <= 8` returns `True`
`>=`	`True` if the first number is equal to or greater than the second, `False` otherwise	`2 >=` returns `False`
`==`	`True` if the first number is equal to the second, `False` otherwise	`6 == 6` returns `True`
`!=`	`True` if the first number is not equal to the second, `False` otherwise (not equal)	`6 != 6` returns `False`

Conditional Statements

File location: Stemx/Python_Basics/condtional_statements.py

In this example, we use conditional statements to check whether a variable is positive, not positive, negative, or zero.

```python
# If statement
x = 10
if x > 0:
    print("x is positive")

# If-else statement
y = -5
if y > 0:
    print("y is positive")
else:
    print("y is not positive")

# If-elif-else statement
z = 0
if z > 0:
    print("z is positive")
elif z < 0:
    print("z is negative")
else:
    print("z is zero")
```

Conditional

Statement	Description	Example

`if` `elif` `else`	If a condition is true, execute the block of code underneath the *if statement*. If not, see if the condition is true in one or more *else if* (`elif`) statements. If one of those is true, execute the code block under that. Otherwise, execute the code block underneath the *else statement*. `elif` and `else` statements are optional.	`number = 42`
		`guess = int(input("Guess a number between 1-100: "))`
		`if guess == number:`
		`  print("You win!")`
		`elif guess < number:`
		`  print("Nope")`
		`  print("Too low")`
		`else:`
		`  print("Nope")`
		`  print("Too high")`
		`print("Run the program to try again")`

`while`	A *while loop* executes the block of code underneath it repeatedly as long as the condition is true.	`counter = 15`
		`while counter >= 5:`
		`  print(counter)`
		`  counter = counter - 1`

`for..in`	Iterate over a sequence of numbers or objects. The variable declared in a *for loop* assumes the value of one of the numbers (or objects) during each iteration of the loop.	`for i in range(1, 11):` `  print(i)`

`break`	Use the *break statement* to exit out of a loop.	`while True:`
		`  message = input("Tell me when to stop: ")`
		`  if message == "stop":`
		`  break`
		`  print("OK")`

Loops

File location: Stemx/Python_Basics/loops.py

In this example, we use a while loop to print out numbers from 1 to 10, and a for loop to iterate over a list of fruits and print out each fruit.

```python
# While loop
i = 1
while i <= 10:
    print(i)
    i += 1

# For loop
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
    print(fruit)
```

Challenge 5: Write an Even Loop

- Objective: Write a program that asks the user to enter a positive integer and then prints out all the even numbers from 1 to that integer using a while loop.

- For example, if the user enters 8, the program should print out 2, 4, 6, 8.

Lists

- A list in Python is an ordered collection of items.
- Lists are written with square brackets “[ ]”
- Items in a list are separated by commas “,”

- List items can be accessed directly

- Python is flexible so lists can hold multiple data types

- Common functions are available such as:
  - size()
  - append()

- ***What is the index of the first item in each list?***

main.py > ...

```
1   # Lists
2   fruit = ["apple", "banana", "cherry"]
3   fish = ["cod", "flounder", "tuna"]
4   numbers = [1, 2, 3, 4, 5]
5   mixed = ["apple", 1, "banana", 2, "cherry", 3]
6   floats = [1.1, 2.2, 3.3, 4.4, 5.5]
7
8   # Grab a single item from a list
9   fav_fruit = fruit[0]
10  fav_fish = fish[1]
11
```

Challenge 6: Adding a List to your Even Loop

- Objective: Take your program that asks the user to enter a positive integer and then prints out all the even numbers from 1 to that integer; and now store all the values in a single list. Print the list at the end instead of each number.

- Hints:
  - To make a list longer than its original size you can use the function “append()” to add to its end

Challenge 7: Can lists be made Smaller?

If I wanted to take a list and completely remove the last element, what would I do? Is this possible?

Hint:

- “append()” adds to a list

Functions

File location: Stemx/Python_Basics/functions.py

In this example, we define a function greet that takes a parameter name and prints out a greeting message, and a function add_numbers that takes two parameters x and y and returns their sum. We then call these functions with appropriate arguments.

```python
# Function with parameters
def greet(name):
    print("Hello, " + name + "!")

greet("John")

# Function with return value
def add_numbers(x, y):
    return x + y

result = add_numbers(10, 20)
print("Result of adding two numbers:", result)
```

Functions

- Functions
  - Functions allow you to name a block of code and then reuse that code by calling its name.
  - What does the function below do?

```
def add(x, y):
    sum = x + y
    return sum

print(add(2, 3))
```

Challenge:

Starting with the code below, implement the sumTo() function that takes an integer as a parameter (n), sums all the whole numbers from 1 to n (including n), and returns the sum.
(Hint: use a for loop)

```
def sumTo(n):
    # YOUR CODE GOES HERE

# Should be 1
print(sumTo(1))

# Should be 45
print(sumTo(9))

# Should be 5050
print(sumTo(100))
```

Modules and Libraries

- Modules and Libraries
  - Modules are another way to reuse code and help you organize your program. They are simply files that are imported into your main program.
  - After importing a module, you can use a module in much the same way you would an object: access constants and functions using the dot-notation.

Challenge:

Python comes with several standard modules that you can import into your program. One of them is the math module, which you can use with import math. Use the constants and functions found in the math module to perform the following actions:

```
import math

print(math.ceil(3.456))
print(math.sqrt(9216))
print(math.pi * (2 ** 2))
```

Modules and Libraries

Rasterio: access to geospatial raster data

Geographic information systems use GeoTIFF and other formats to organize and store gridded raster datasets such as satellite imagery and terrain models. Rasterio reads and writes these formats and provides a Python API based on Numpy N-dimensional arrays and GeoJSON.

Here's an example program that extracts the GeoJSON shapes of a raster's valid data footprint.

```
import rasterio
import rasterio.features
import rasterio.warp

with rasterio.open('example.tif') as dataset:

    # Read the dataset's valid data mask as a ndarray.
    mask = dataset.dataset_mask()

    # Extract feature shapes and values from the array.
    for geom, val in rasterio.features.shapes(
            mask, transform=dataset.transform):

        # Transform shapes from the dataset's own coordinate
        # reference system to CRS84 (EPSG:4326).
        geom = rasterio.warp.transform_geom(
            dataset.crs, 'EPSG:4326', geom, precision=6)

        # Print GeoJSON shapes to stdout.
        print(geom)
```

The output of the program:

```
{'type': 'Polygon', 'coordinates': [[(-77.730817, 25.282335), ...]]}
```

Challenge:

Use the Rasterio library to pull coordinates from sattelite imagery.

Download sattelite imagery from stemx.app/x-marks-the-spot-challenge

Use the code same to the left. First, make sure to install Rasterio using the following command:

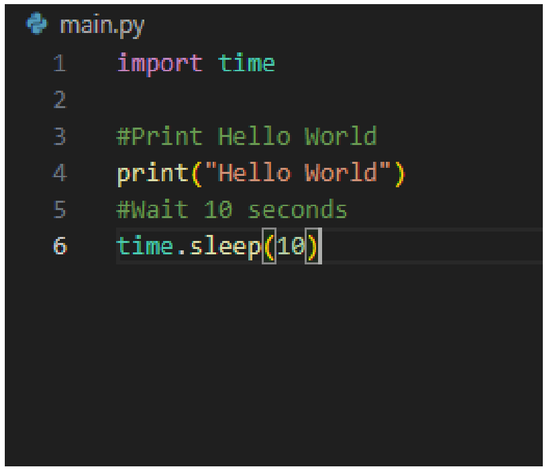
pip install rasterio

STEMX

Extra Challenge: Build a working Stoplight

- Objective: Using the breadboard you created with three lights (green, yellow, red). Write a Python script that will turn on each light in order (green, yellow, red) and keep each light on for 2 seconds before moving.

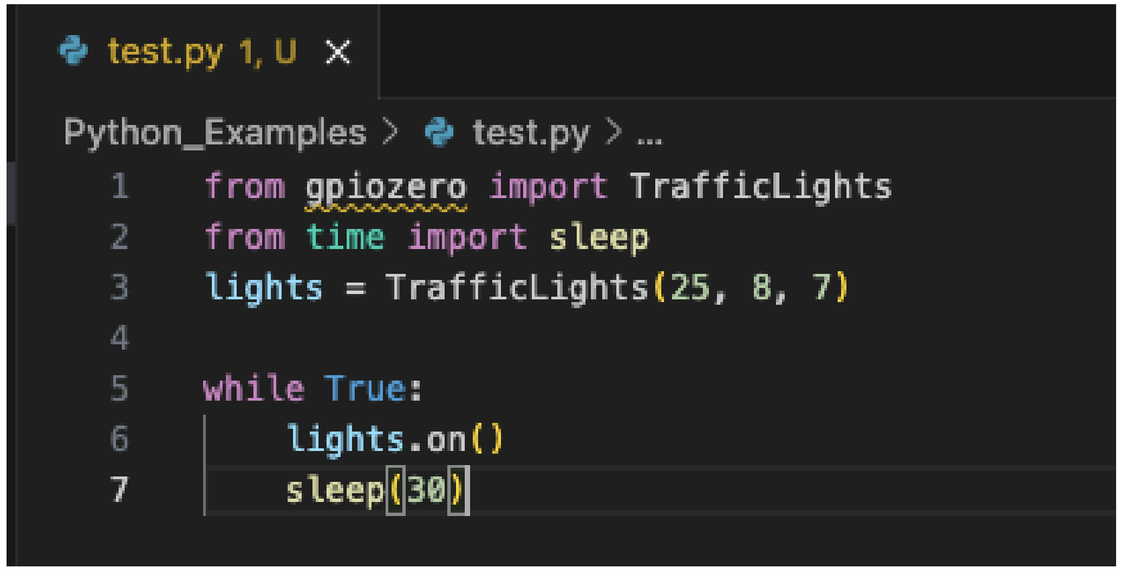
- Hints:
  - Write Psuedo-code first: So just write in plain English the steps you think will need to happen
  - LOOP!
  - Remember “blinking_led.py”
  - Python has a built in “sleep(x)” function that can be used to execute the above code for x seconds.
    - Example: This will print “Hello World” then wait 10 seconds before moving on.



Learning Experience: Never Reinvent the Wheel

- Although building all of the looping code for the Stoplight may be fun and great to learn the concepts of Python, in a real-world developers' environment it is better to find a reliable library!

Check out this alternative solution to our problem! Notice it is only a few lines of code



Git and Github Basics

Git:

• Git is a distributed version control system. It allows multiple people to work on a project at the same time without overwriting each other's changes. It keeps track of all changes made to files in a repository.

• It was created by Linus Torvalds, the creator of Linux, in 2005.

GitHub:

• GitHub is a web -based hosting service for Git repositories. It provides a platform for collaboration, allowing developers to work on projects from anywhere in the world.

• In addition to hosting your code, GitHub offers features like bug tracking, task management, and project wikis.

Git and Github Basics

Key Concepts and Basic Commands:

- Repository (repo): A directory or storage space where your project lives. It can contain folders and files, images, videos, spreadsheets – anything your project needs.
- **Clone** : Creates a local copy of a repository on your machine.
  - Example: git clone https://github.com/user/repo.git
- **Commit** : Saves changes to your local repository. Each commit has a unique ID.
- **Push**: Sends your commits to the remote repository on GitHub.
- **Pull** : Fetches changes from the remote repository and merges them into your local repository.

Challenge 9: Clone a Git repository

Source: https://github.com/ECU-Sensing/stemx-image.git

Screen Detection

File location: Stemx/Python_Basics/screen_detection.py

In this example code, we first load the image that we want to detect using the PIL library. We then get the size of the screen using the pyautogui library's size() function. We then search for the image on the screen using the pyautogui library's locateOnScreen() function, which returns the location of the image on the screen if it's found, or None if it's not found.

We then check if the image was found and if it was, we print the location of the image on the screen and move the mouse cursor to the center of the image location using the pyautogui library's center() and moveTo() functions. If the image was not found, we print a message indicating that it was not found.

Note that you'll need to replace image_to_detect.png with the path to your own image file that you want to detect.

```python
from PIL import Image
import pyautogui

# Load the image to be detected
image_to_detect = Image.open('image_to_detect.png')

# Get the screen size
screen_size = pyautogui.size()

# Search for the image on the screen
location = pyautogui.locateOnScreen(image_to_detect, confidence=0.9)

# Check if the image was found
if location is not None:
    # Print the location of the image on the screen
    print(f"Image found at: {location}")

    # Get the center of the image location
    center = pyautogui.center(location)

    # Move the mouse cursor to the center of the image location
    pyautogui.moveTo(center)
else:
    # Print a message indicating that the image was not found
    print("Image not found on screen")
```

Challenge 10: Screen Detection Automation

Objective: Make changes to the Screen Detection Python program to continuously scan for the image.

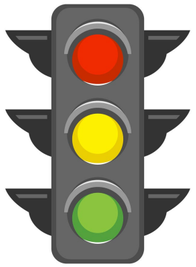
Hints:

- Remember the blinking light code

Stop Light Challenge

File location: Stemx/GPIO/stoplight.py

In this example, the stop light has three states: green, yellow, and red. It has a transition() method that switches the light from one state to the next, and a run() method that loops indefinitely and checks whether it's time to transition to the next state.



Challenge: Wire a RED, GREEN and YELLOW LED from your GPIO to your breadboard.

You will need to add the GPIO code to the stoplight.py to properly send the signals at the correct time to simulate a stop light.

```python
import time

class StopLight:
    def __init__(self):
        self.current_state = 'green'
        self.next_transition_time = time.time() + 10 # Start with 10 seconds of gr

    def set_state(self, state):
        self.current_state = state

    def get_state(self):
        return self.current_state

    def transition(self):
        if self.current_state == 'green':
            self.set_state('yellow')
            self.next_transition_time = time.time() + 2 # 2 seconds of yellow ligh
            print("Yellow light - Prepare to stop!")
        elif self.current_state == 'yellow':
            self.set_state('red')
            self.next_transition_time = time.time() + 10 # 10 seconds of red light
            print("Red light - Stop!")
        elif self.current_state == 'red':
            self.set_state('green')
            self.next_transition_time = time.time() + 10 # 10 seconds of green lig
            print("Green light - Go!")

    def run(self):
        while True:
            if time.time() >= self.next_transition_time:
                self.transition()

stop_light = StopLight()
stop_light.run()
```

Artificial Intelligence

- How is this being applied today?
  - Automation
  - Decision-Making
  - Personalization
  - Predictive Maintenance
  - Customer Service
  - Healthcare

Generative AI

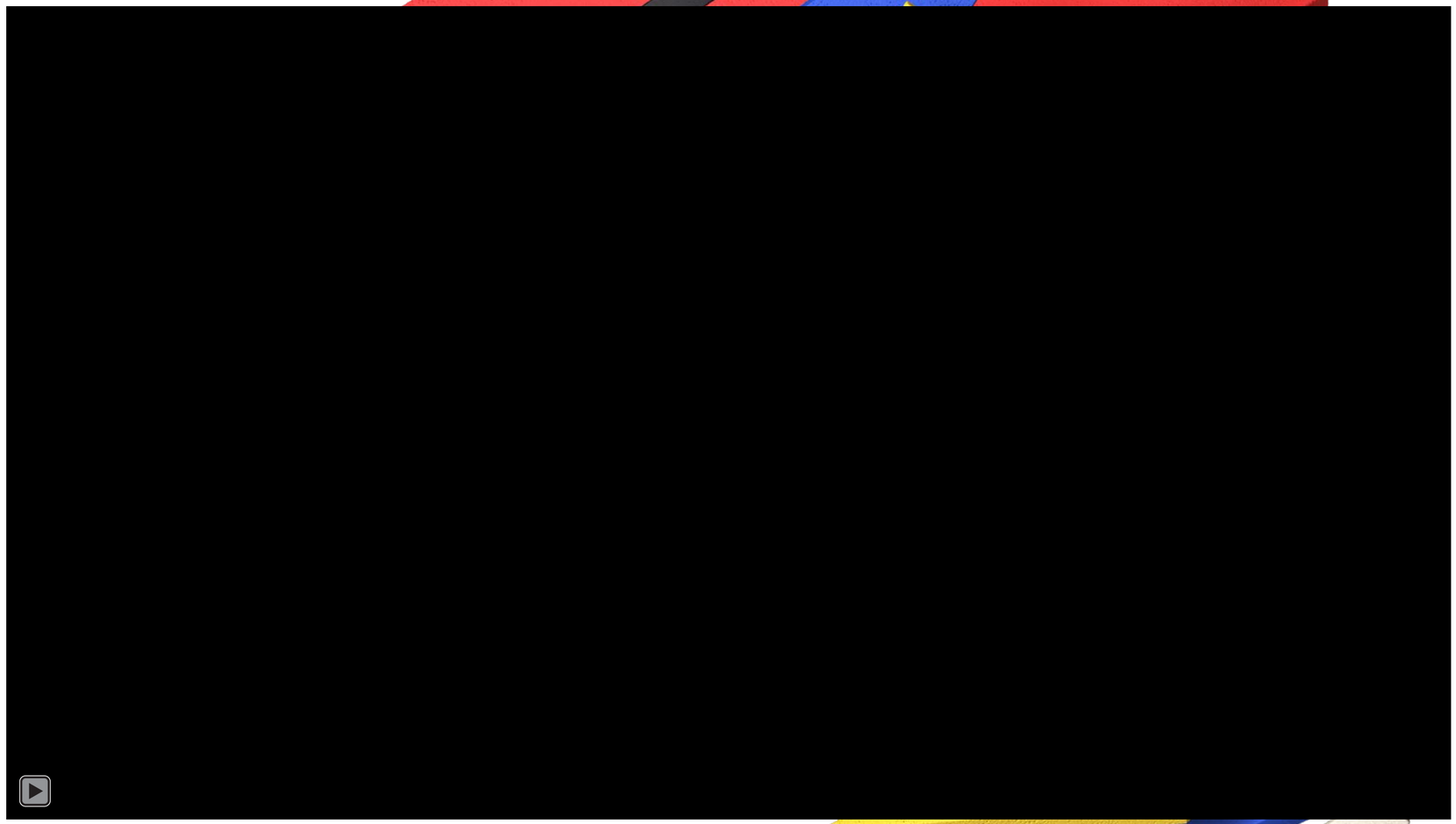
- ChatGPT is a language-based AI developed by OpenAI. It has been trained on vast amounts of text data and can generate human-like text.
- How it Works: It uses a type of neural network called a Transformer, and more specifically an architecture known as GPT (Generative Pretrained Transformer). It takes in a sequence of text and predicts the next word.
- Lets look at some examples!

ChatGPT

- Examples:
  - Write a short adventure story about a dog named Spot who lives in a city.
  - Imagine you are a robot in the future. Describe your day.
  - Write a program that asks the user to enter a positive integer and then prints out all the even numbers from 0 to that integer using a while loop.

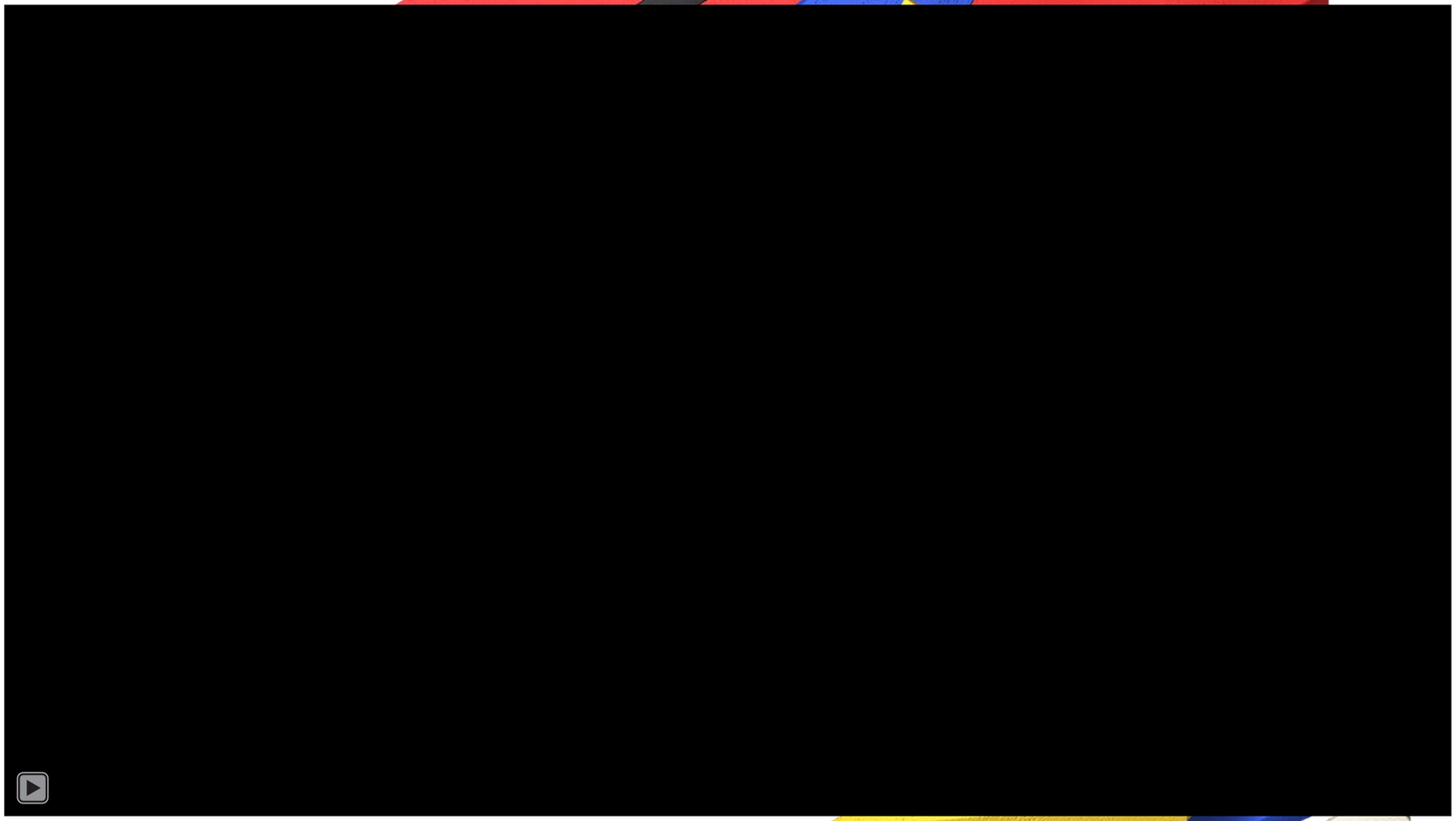
Live AI Detection

- Classification
- Confidence %



Live AI Detection

- Classification
- Confidence %



Who Uses AI?

Google is one of the largest companies in the world that heavily relies on AI. Google uses AI for its search engine algorithms, image recognition, voice recognition, natural language processing, and more.

Amazon uses AI for product recommendations, personalization, and forecasting demand for products, among other things.

Facebook uses AI for content moderation, image recognition, and facial recognition.

Microsoft uses AI for its digital assistant Cortana, speech recognition, and machine translation.

Tesla uses AI for autonomous driving, as well as predictive maintenance of its electric vehicles.

IBM uses AI for its Watson platform, which provides natural language processing and other AI services to businesses.

Apple uses AI for facial recognition in its iPhones, voice recognition with Siri, and other applications.

Netflix uses AI for content recommendations to its subscribers.

Uber uses AI to optimize its ride-hailing services, including route planning and demand prediction.

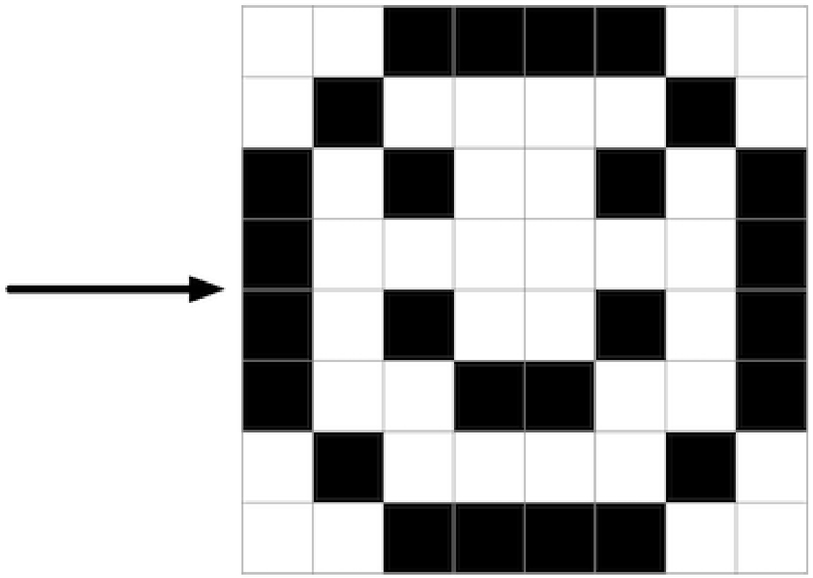
Airbus uses AI to improve aircraft design and performance, as well as for predictive maintenance of its planes.

Computer Vision

How does a computer see?

- Pixels and binary (0,1)
- Features and patterns

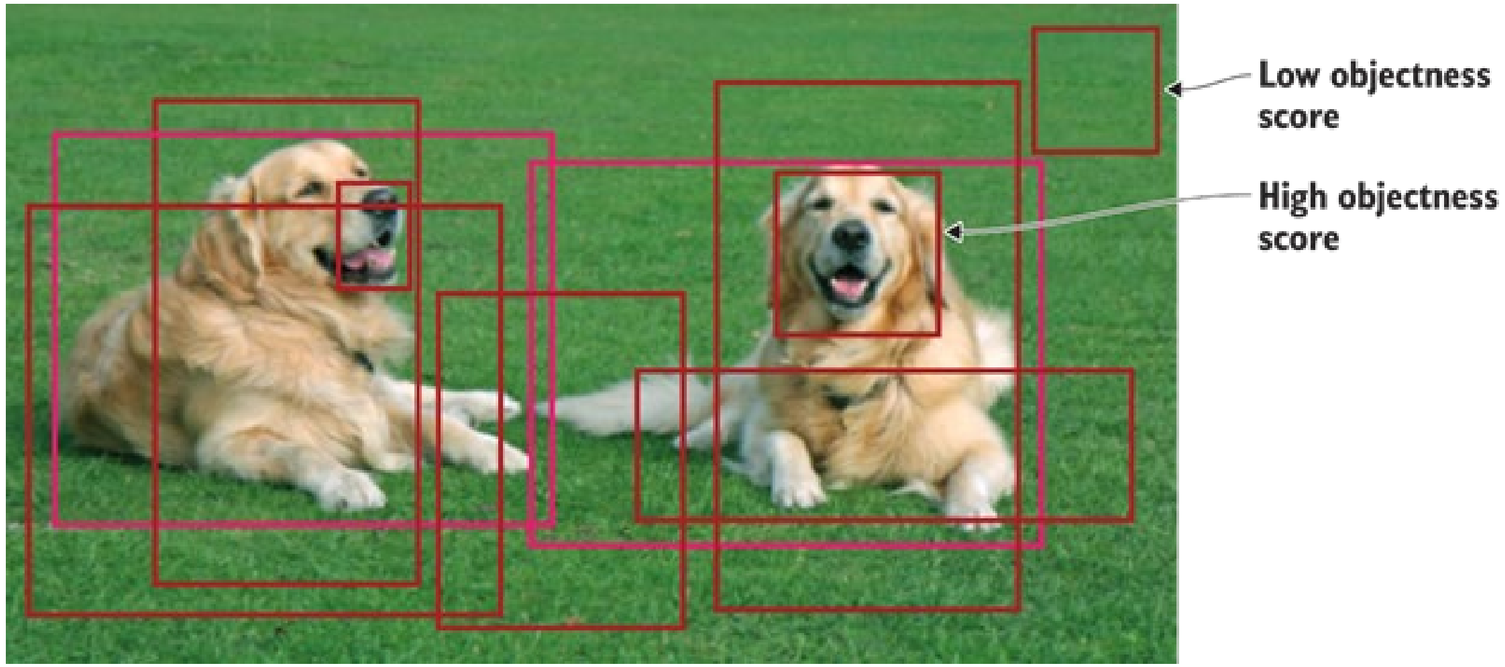
1 1 0 0 0 0 1 1
1 0 1 1 1 1 0 1
0 1 0 1 1 0 1 0
0 1 1 1 1 1 1 0
0 1 0 1 1 0 1 0
0 1 1 0 0 1 1 0
1 0 1 1 1 1 0 1
1 1 0 0 0 0 1 1



Computer Vision

How does a computer recognize objects?

- Where = Localization
  - How? Objectness
  - Feature maps
- What = Classification
  - How? Datasets
  - Machine Learning
- Both = Recognition



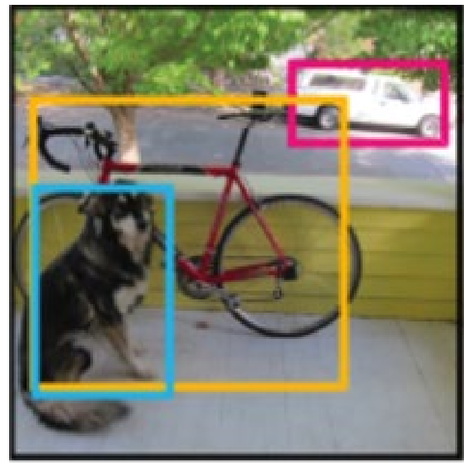
Computer Vision

What is the process that the computer uses to recognize objects?

- YOLO: You Only Look Once (algorithm)
  - Where - Bounding boxes
  - What - Learned Labeled Data Sets

Split the image into grids

Predict bounding boxes and classifications

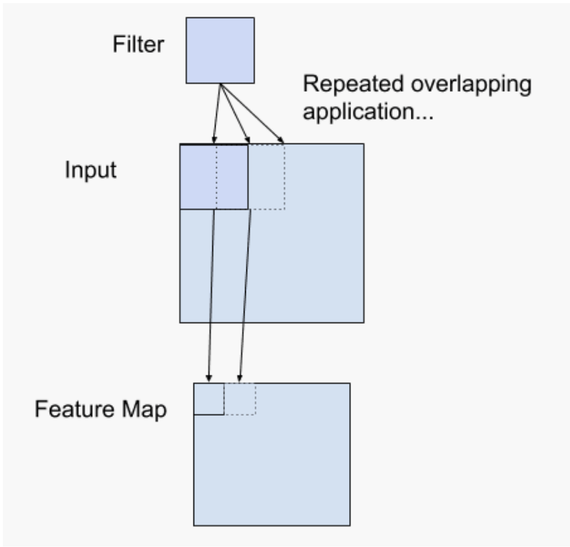


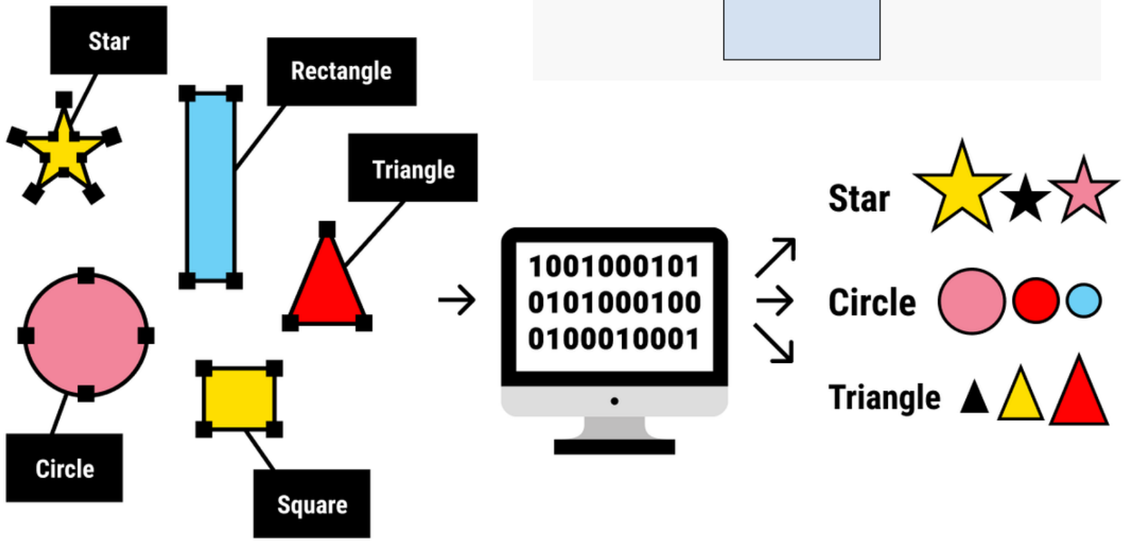
Final predictions after non-maximum suppression

Computer Vision

How does a computer learn?

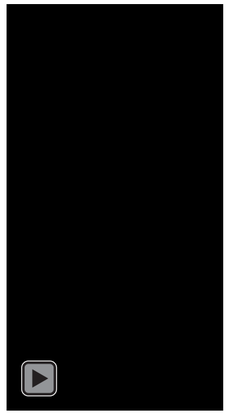
- Labeled Data Sets
  - [Star, Circle, Triangle]
- Feature maps
- Model generation
  - For yolo this is our weights





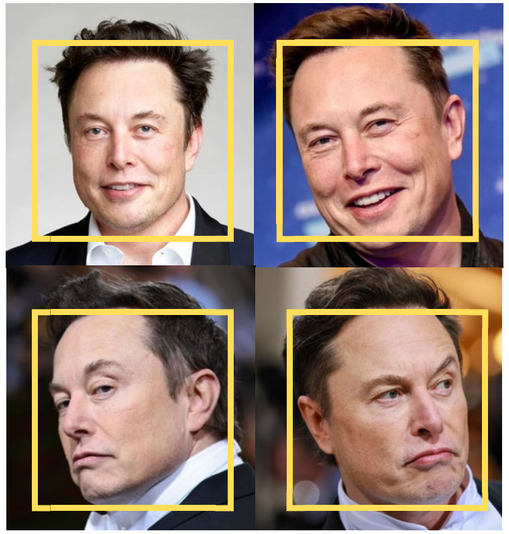
Computer Vision

How does a computer learn to recognize your face?



(1)

Label a Data set by class and set the bounding box



(2)

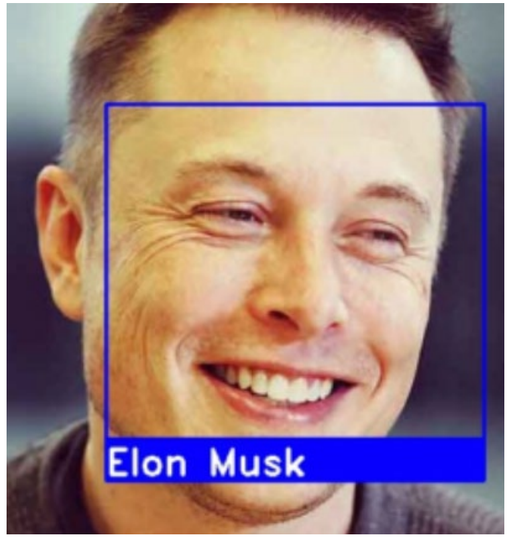
Train you model with preset weights

(3)

Run YOLO Algorithm

(4)

Smile for the camera!



YoloV7 Module

File location: /home/pi/yolov7

Lets test our YoloV7 detection algorithm.

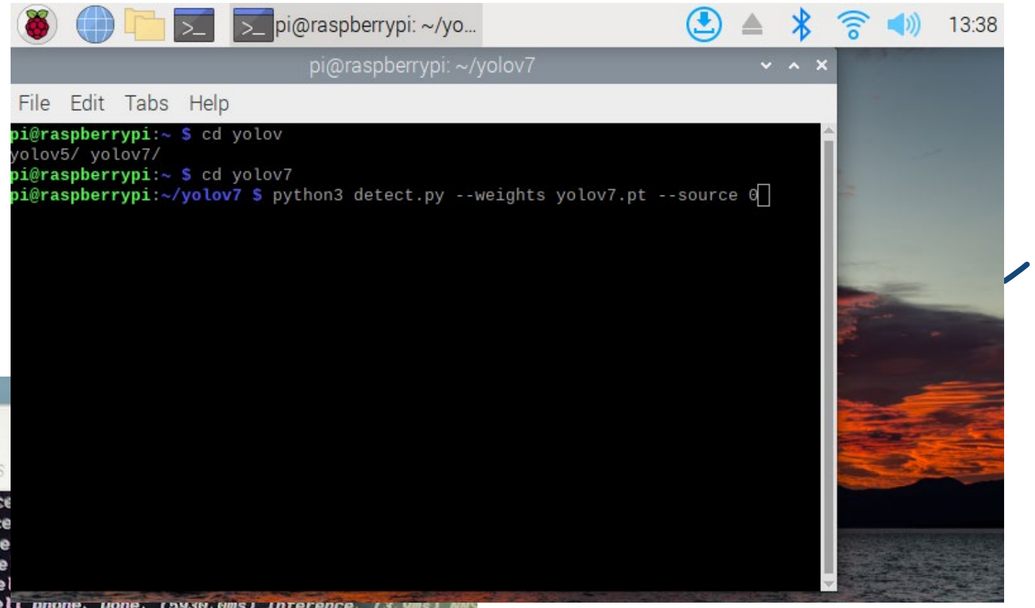
To do this follow the steps below:

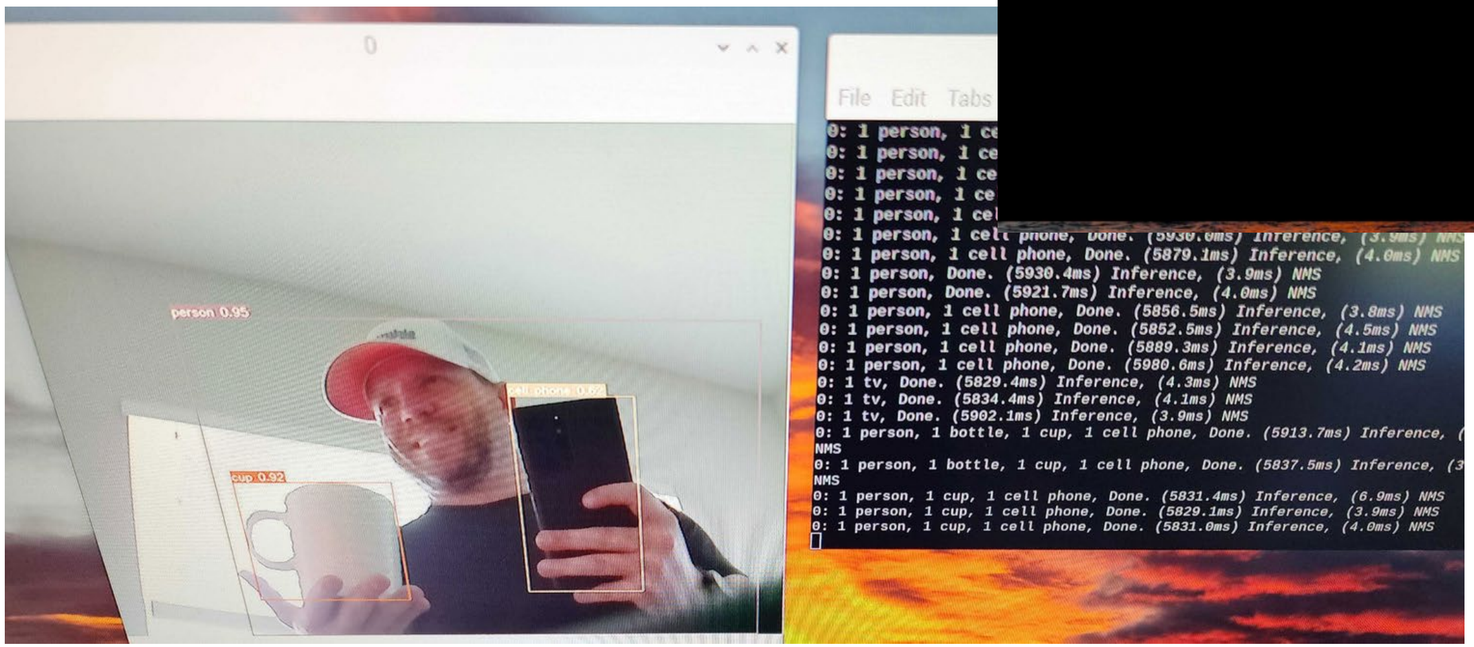
- Open CMD prompt and run

  cd yolov7

  python3 detect.py --weights yolov7-tiny.pt --source 0

Note: You can run with yolov7.pt for weights as in picture but the tiny version will run faster.





YoloV7 Module

File location: /home/pi/yolov7

When we give the command:

```
python3 detect.py --weights yolov7.pt --source 0
```

We are telling python version 3 to run the detect.py script and we are providing two additional arguments, one is the --weights which is the trained model we are passing the data through as well as the --source which is our source of data. The '0' we are suppling in short means 'device 0' which would default to our camera running on the Pi as it's default is device 0.

Challenge: Edit detect.py print statement when an object is detect to output some additional information, such as "You Detected an Object!!!!"

YoloV7 Module

File location: /home/pi/yolov7/detect.py

Challenge: When a person is in view of the camera, trigger an LED to HIGH from the GPIO pins.

For this challenge its recommended to make a copy of your detect.py to edit.

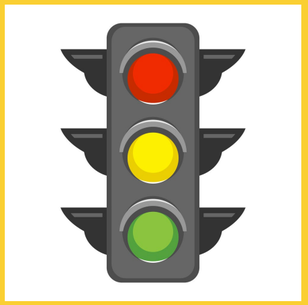
Advanced Challenge: Now that we understand how to edit our detect.py script lets now see if we can incorporate our previous challenge of the from the Stop Light we developed in the GPIO section. Lets pretend our car is driving along and it now detects a person in view. In the case the output of the light must stay RED until no person is in view. Bonus if the classification perecentage of the person detected is less than 40% to trigger the Yellow Light for caution and anything above 40% will stay RED.

Previous Stop Light Challenge:

File location: Stemx/GPIO/stoplight.py

Challenge: Wire a RED, GREEN and YELLOW LED from your GPIO to your breadboard.

You will need to add the GPIO code to the stoplight.py to properly send the signals at the correct time to simulate a stop light.



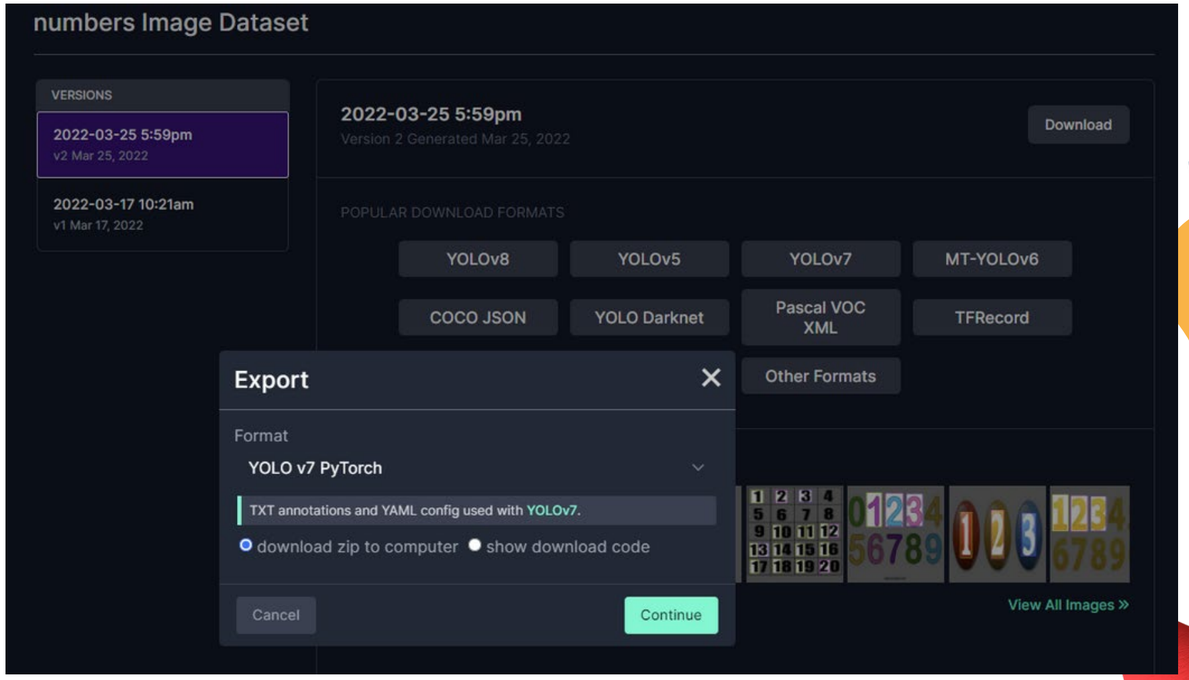
```python
import time

class StopLight:
    def __init__(self):
        self.current_state = 'green'
        self.next_transition_time = time.time() + 10 # Start with 10 seconds of gr

    def set_state(self, state):
        self.current_state = state

    def get_state(self):
        return self.current_state

    def transition(self):
        if self.current_state == 'green':
            self.set_state('yellow')
            self.next_transition_time = time.time() + 2 # 2 seconds of yellow ligh
            print("Yellow light - Prepare to stop!")
        elif self.current_state == 'yellow':
            self.set_state('red')
            self.next_transition_time = time.time() + 10 # 10 seconds of red light
            print("Red light - Stop!")
        elif self.current_state == 'red':
            self.set_state('green')
            self.next_transition_time = time.time() + 10 # 10 seconds of green lig
            print("Green light - Go!")

    def run(self):
        while True:
            if time.time() >= self.next_transition_time:
                self.transition()

stop_light = StopLight()
stop_light.run()
```

Yolo V7 Training

For this section we will look at what labeled data sets look like.
Then we will download a dataset and train our own model.

Large Amount of Datasets:
https://universe.roboflow.com/

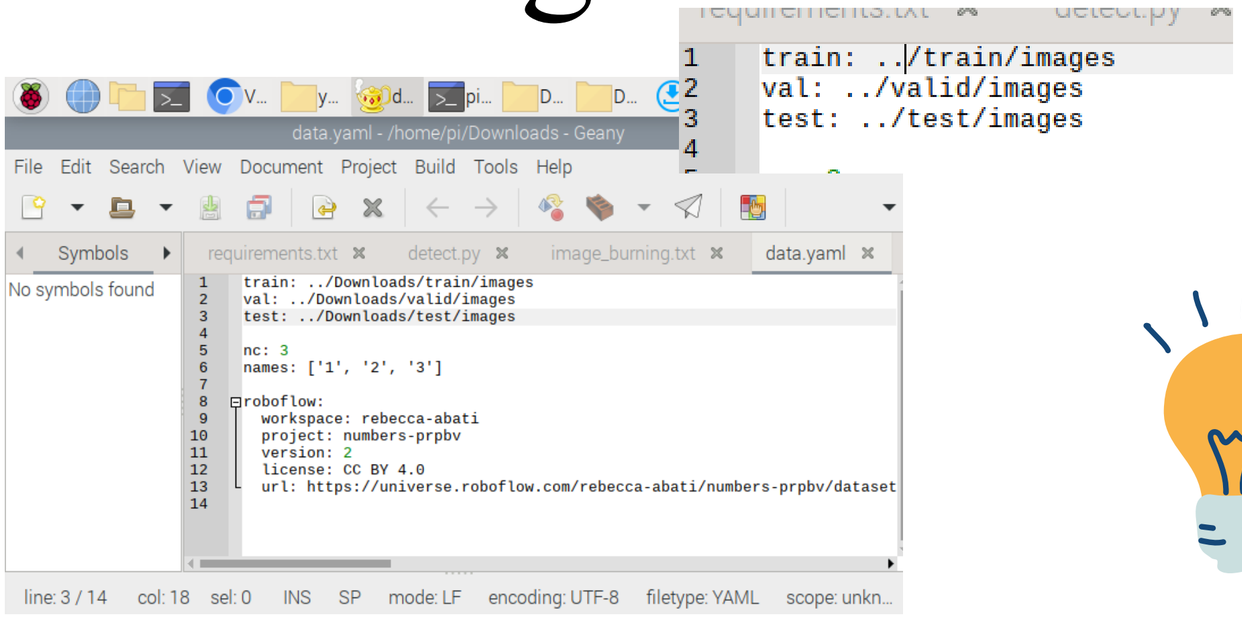
Download Yolov7 Data:
https://universe.roboflow.com/rebecca-abati/numbers-prpbv

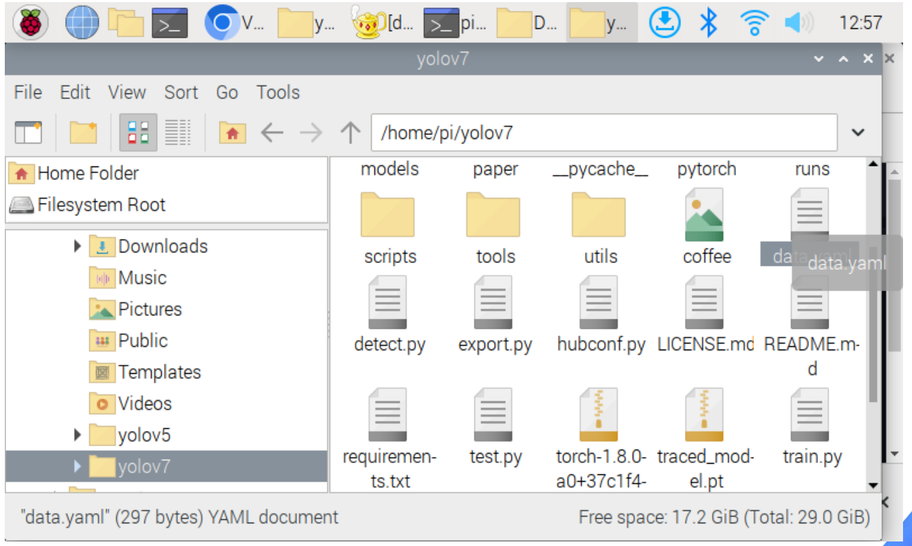


YoloV7 Training

After downloading the yolov7 zip data.

- Open the download directory
- Right click on the zip folder and select Extract all to the current directory.
- Double click to open the data.yaml file
- Change line 1-3 to add "/Downloads/ to each line.
  - ../train/images -----> ../Downloads/train/images
  - ../valid/images -----> ../Downloads/valid/images
  - ../test/images -------> ../Downloads/test/images
- Save file
- Right click on data.yaml and select copy
- Navigate to /home/pi/yolov7 and paste or move the data.yaml file to this directory.

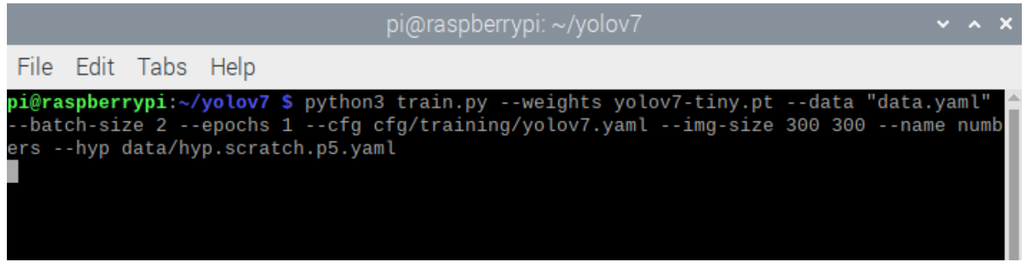




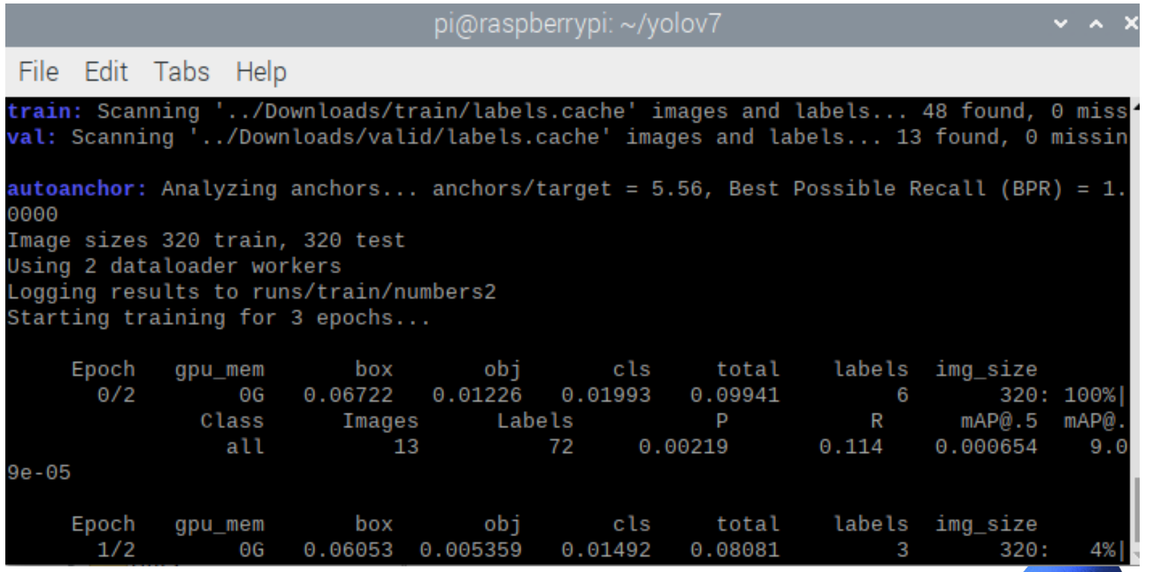
YoloV7 Training

Open a new CMD prompt and cd yolov7

Run:
python3 train.py --weights yolov7-tiny.pt --data "data.yaml" --batch-size 2 --epochs 3 --cfg cfg/training/yolov7.yaml --img-size 300 300 --name numbers --hyp data/hyp.scratch.p5.yaml



You will see the console log output the training of the model.



YoloV7 Training

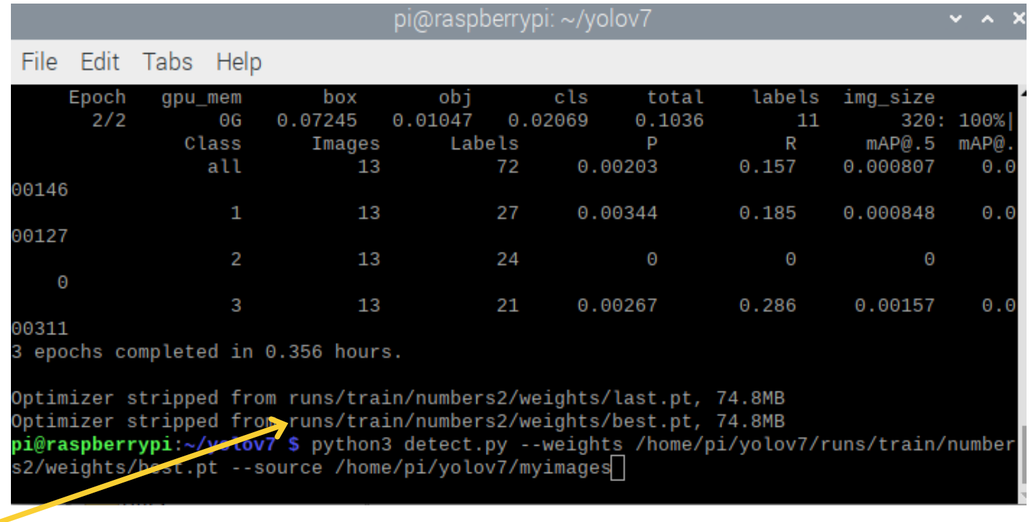
When complete you will see the output of the location of the model weights. Recall that the .pt file is what we feed for our model as the weights.

Make a new folder in your yolov7 directory and name it myimages. Place any images you would like to test in this directory.

Note: If error occurs asking for specific file type such as jpeg. Rename the image and add ".jpeg"

For example if my image was myimage1, I would rename this to myimage1.jpeg

Run:
python3 detect.py --weights /home/pi/yolov7/LOCATION_FROM_OUTPUT_HERE --source /home/pi/yolov7/myimages



Final result will output directory storing the image

```
Done. (4422.1ms) Inference, (1.3ms) NMS
 The image with the result is saved in: runs/detect/exp8/numbers_image.jpeg
Done. (4.472s)
```

Software Defined Radio

SDR

Software Defined Radio

- Who uses SDR?
  - Radio hobbyists and enthusiasts
  - Security researchers
  - Radio professionals
  - Academic researchers
  - Aviation
  - Police
  - Military
  - Emergency Response

https://www.youtube.com/watch?v=n4jUmnaFKc0

What is Software Defined Radio

SDR is a radio communication system where components that have been typically implemeted in hardware (mixers, amplifiers, filters, modulators) are instead implemented by software on personal computer.

Listen to broadcast stations (88-108 mhz)

Listen to Aircraft

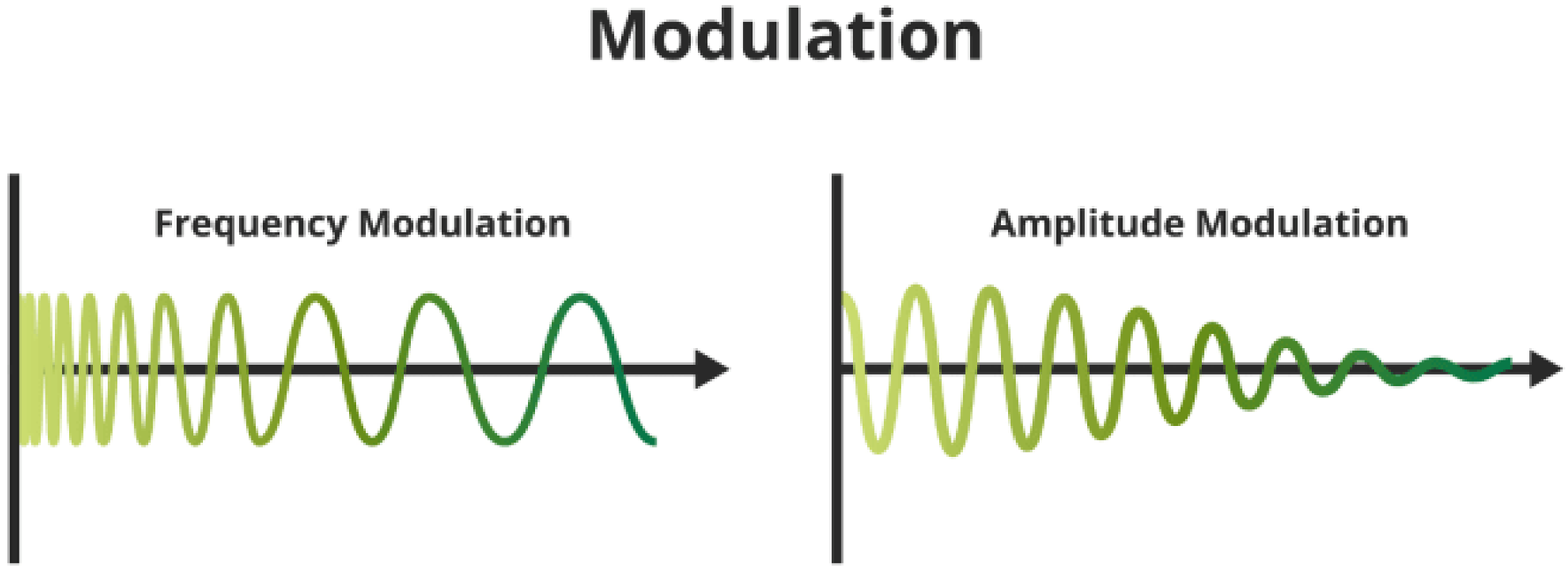
Listen to emergency services

Common Frequencies

In the United States, the most commonly used wireless frequencies for communication are:

- 2.4 GHz: This is the most common frequency band and is used by Wi-Fi, Bluetooth, and many other wireless devices. It is divided into several channels, each 5 MHz apart.
- 5 GHz: This frequency band is also widely used for Wi-Fi and offers more channels than the 2.4 GHz band. It is less crowded and typically provides faster speeds.
- 900 MHz: This band is often used for cordless phones and industrial, scientific, and medical (ISM) applications.
- 800 MHz and 1.9 GHz: These bands are commonly used for cellular communication, including voice calls and mobile data.
- 700 MHz: This band is used for 4G LTE mobile broadband, and is also being adopted for 5G.

AM vs FM



UNITED STATES FREQUENCY ALLOCATIONS

THE RADIO SPECTRUM

RADIO SERVICES COLOR LEGEND

- AERONAUTICAL MOBILE
- AERONAUTICAL MOBILE SATELLITE
- AERONAUTICAL RADIONAVIGATION
- AMATEUR
- AMATEUR SATELLITE
- BROADCASTING
- BROADCASTING SATELLITE
- EARTH EXPLORATION SATELLITE
- FIXED
- FIXED SATELLITE
- INTER-SATELLITE
- LAND MOBILE
- LAND MOBILE SATELLITE
- MARITIME MOBILE
- MARITIME MOBILE SATELLITE
- MARITIME RADIONAVIGATION
- METEOROLOGICAL
- METEOROLOGICAL SATELLITE
- MOBILE
- MOBILE SATELLITE
- RADIO ASTRONOMY
- RADIODETERMINATION SATELLITE
- RADIOLOCATION
- RADIOLOCATION SATELLITE
- RADIONAVIGATION
- RADIONAVIGATION SATELLITE
- SPACE OPERATION
- SPACE RESEARCH
- STANDARD FREQUENCY AND TIME SIGNAL
- STANDARD FREQUENCY AND TIME SIGNAL SATELLITE

ACTIVITY CODE

- FEDERAL EXCLUSIVE
- FEDERAL/NON-FEDERAL SHARED
- NON-FEDERAL EXCLUSIVE

ALLOCATION USAGE DESIGNATION

SERVICE	EXAMPLE	DESCRIPTION
Primary	FIXED	Capital Letters
Secondary	Mobile	1st Capital with lower case letters

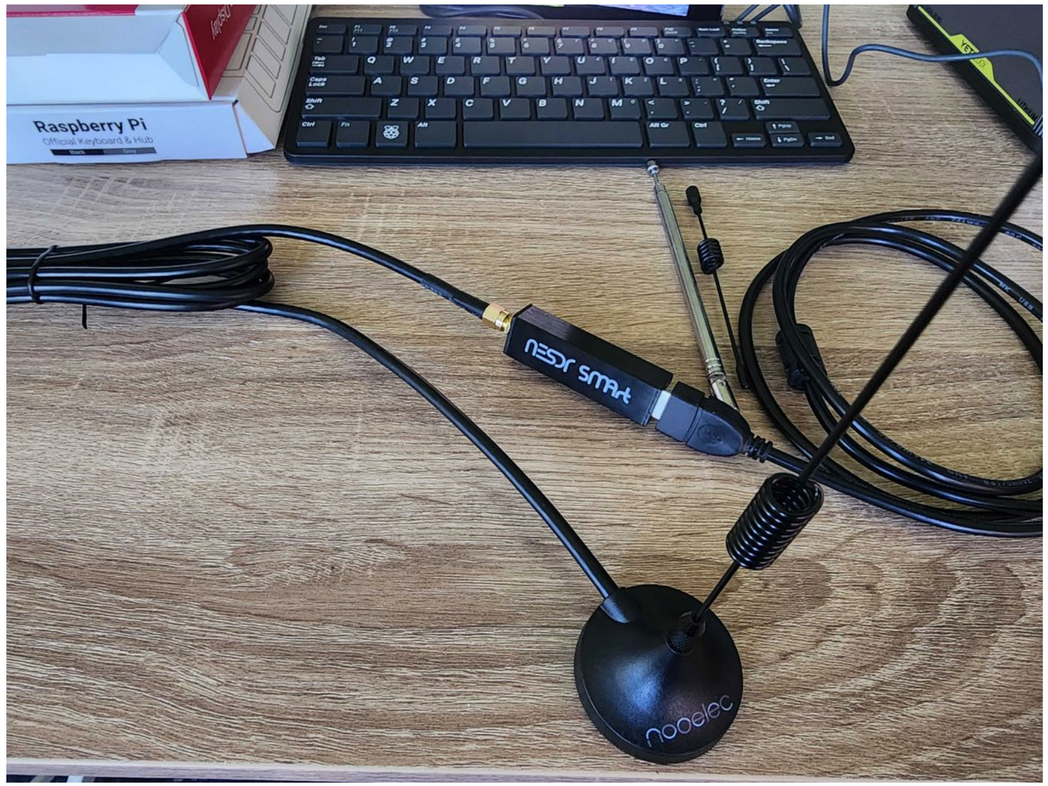
U.S. DEPARTMENT OF COMMERCE
National Telecommunications and Information Administration
Office of Spectrum Management

JANUARY 2016

RTL-SDR Kit

RTL-SDR stands for "Realtek Software-Defined Radio," which refers to a family of USB-based radio receivers that use Realtek RTL2832U chips. These devices can be used to receive and decode a wide range of radio signals, including FM radio, AM radio, DAB digital radio, and various types of digital signals.

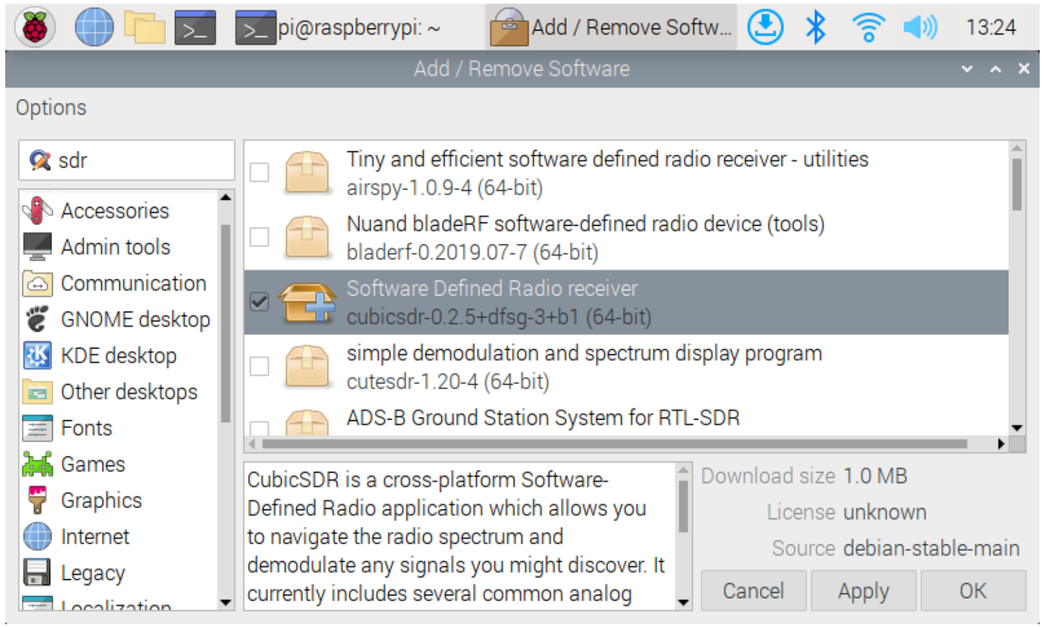
The RTL-SDR devices are popular among hobbyists, hackers, and radio enthusiasts due to their low cost and ease of use. They can be used with various open-source software tools and libraries to perform radio spectrum analysis, signal decoding, and other radio-related tasks.



SDR Setup

Note: This should already be working if loaded with StemX Image on SD Card.

- Navigate under Preferences -> Add/ Remove Software
- Search for SDR
- Select "Software Defined Radio Receiver"
- Apply and Press OK
- Once installed reboot the Raspberry Pi

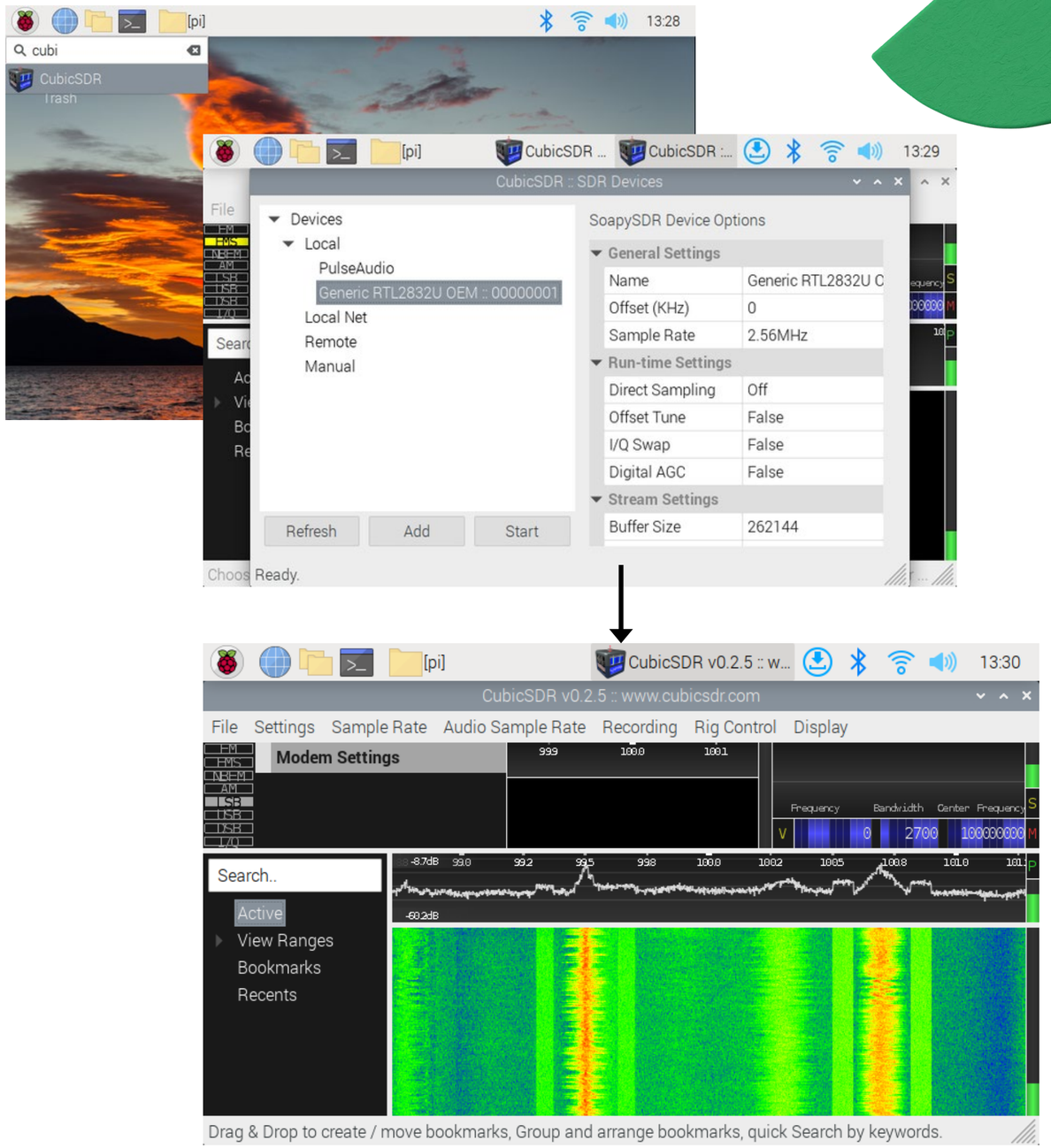


Launch CubicSDR

- Press Top Left Raspberry Icon and search Cubic
  - Open CubicSDR

If Everything was installed and plugged in correctly you should be the Hardware Module listed as pictured under Local. Select that Generic RTL device and press Start.

Note: If it is not being detected, double check that the SDR is properly slotted into the USB port of the pi and reboot.



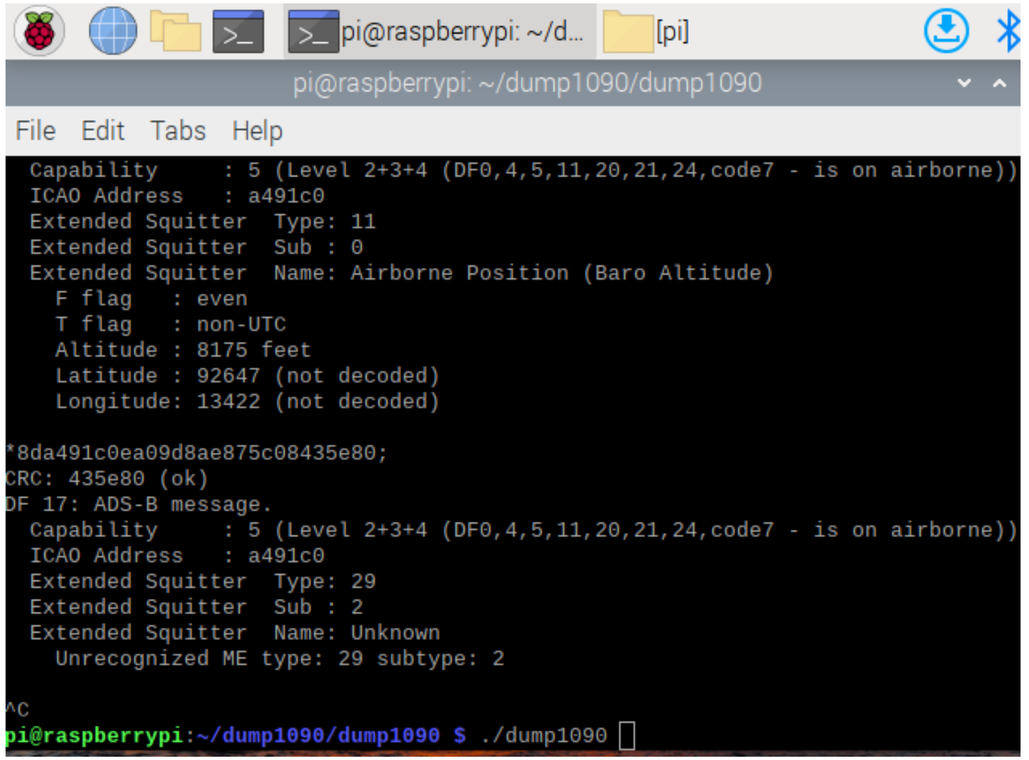
SDR Dump1090

Open CMD and navigate to dump1090/dump1090

cd /dump1090/dump1090
./dump1090

This will outprint the raw JSON to the CMD.

Press CRTL+C to terminate.



SDR Dump1090 Interactive

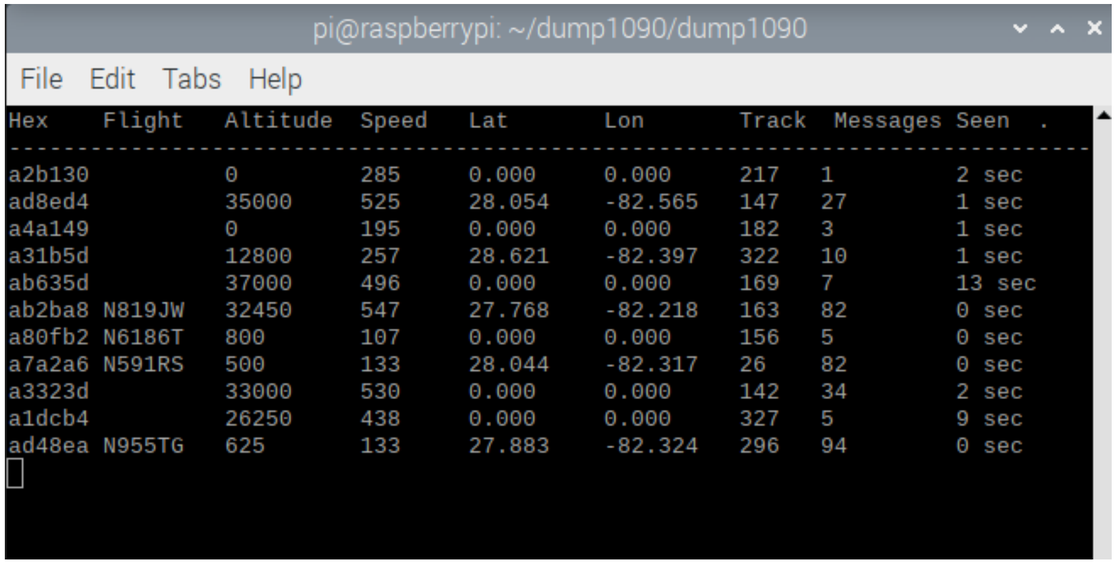
Open CMD and navigate to dump1090/dump1090

```
pi@raspberrypi:~/dump1090/dump1090 $ ./dump1090 --interactive
```

cd /dump1090/dump1090
./dump1090 --interactive

Interactive format displaying flight data.

Press CRTL+C to terminate.



What is an API?

API

File location: Stemx/Python_Basics/api_flights.py

In the above example, we first make a GET request to the hypothetical flight API to retrieve information about flights from New York City to Los Angeles on April 15th, 2022, and then print the response data as a JSON object. We then make a POST request to the same API to book a flight from JFK to LHR on July 1st, 2022, specifying the data as a JSON object and including an authorization header with an API key, and then print the response data.

```python
import requests

# GET request example
response = requests.get("https://flightapi.com/flights?source=NYC&destination=LAX&d
print(response.json())

# POST request example
data = {
    "source": "JFK",
    "destination": "LHR",
    "date": "2022-07-01",
    "passengers": 2,
    "class": "business"
}
headers = {
    "Authorization": "Bearer YOUR_API_KEY"
}
response = requests.post("https://flightapi.com/bookings", json=data, headers=heade
print(response.json())
```

Challenge: Try using the above code to interact with a different flight API that requires authentication and supports additional parameters, such as seat selection or baggage allowance. You'll need to modify the code to include the appropriate credentials and parameters, and ensure that the requests are authorized properly. Good luck!

Link for a large collective list of free APIs to use

SDR Dump1090 Interactive Web Portal

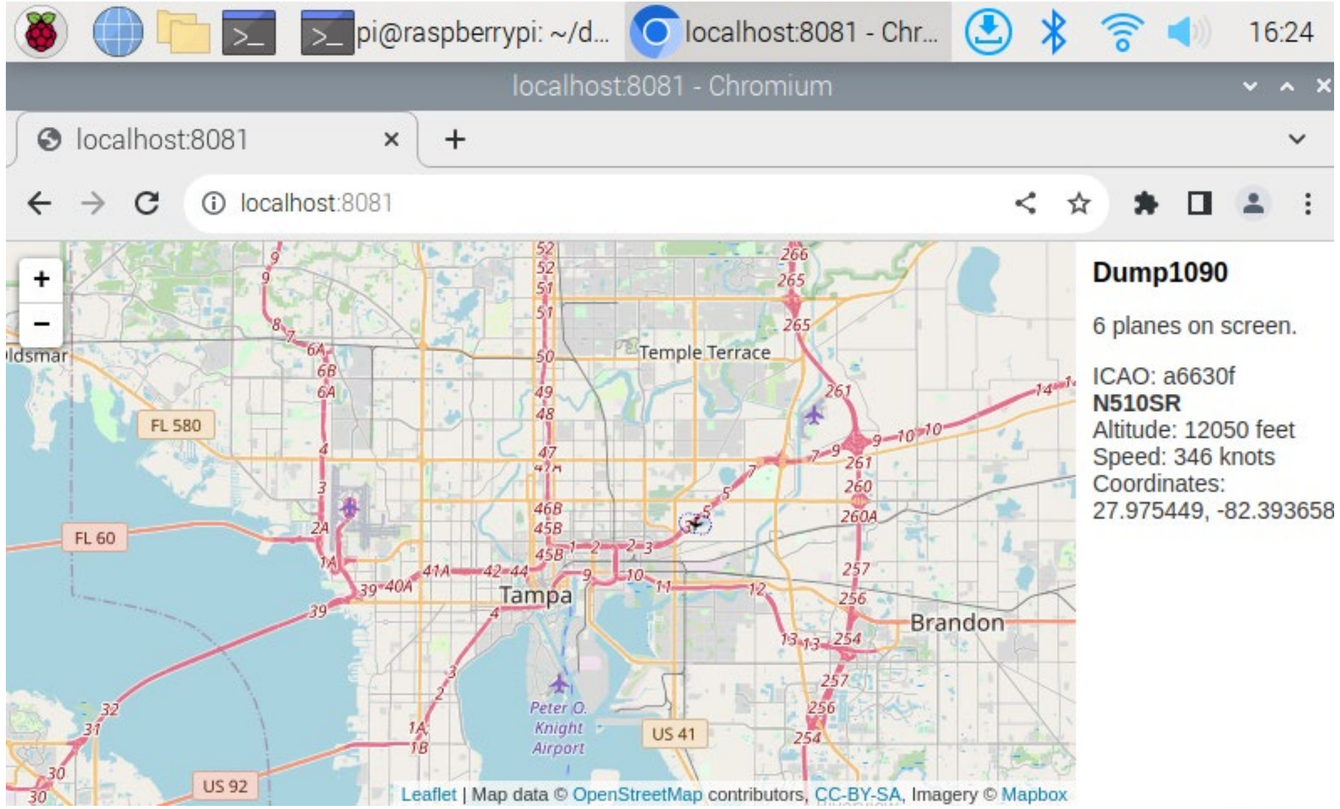
Open CMD and navigate to dump1090/dump1090

```
pi@raspberrypi:~/dump1090/dump1090 $ ./dump1090 --interactive --net --net-http-p
ort 8081
```

cd /dump1090/dump1090
./dump1090 --interactive --net --net-http-port 8081

Open web browser and type in URL:
localhost:8081

Press CRTL+C to terminate.



SDR Dump1090 Interactive Python

The following python code is an example on how to retrieve the data output received from the SDR via dump1090 data API.

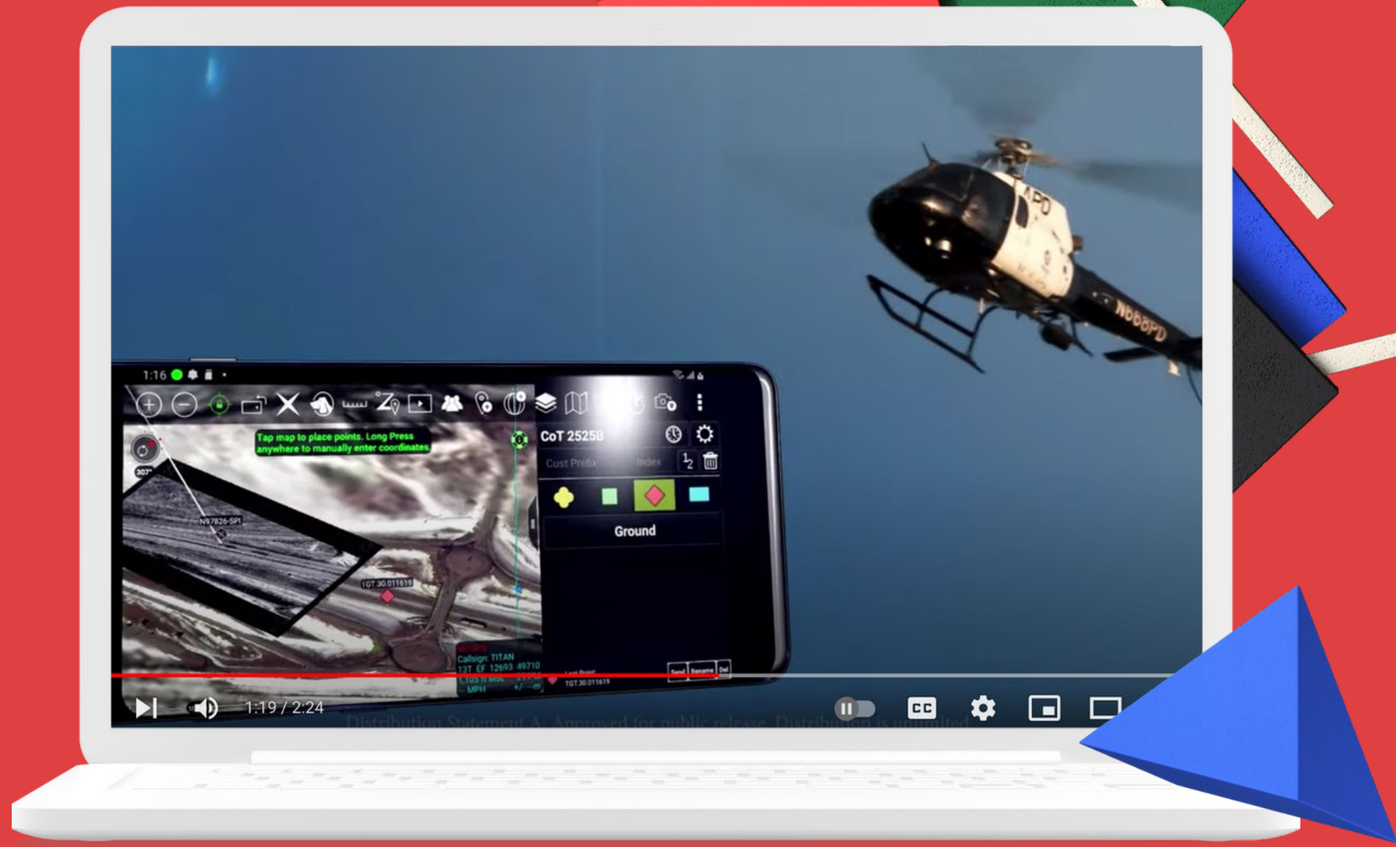
This will be useful when we want to integrate with TAK server.

```python
import urllib.request, json
#import telepot
from time import sleep
import time

dumpurl = "http://localhost:8081/data.json"

while(True):
    with urllib.request.urlopen(dumpurl) as url:
        aircraft_json= json.loads(url.read().decode())

    if None in aircraft_json:
        print("No Plane!")
        time.sleep(5)
    else:
        print(aircraft_json)
        print("------------------------------------------------------------")
        #parse aircraft_json['aircraft']
        #print data...
        #Do Something
        time.sleep(10)
```

TAK

Team Awareness Kit

- Who uses TAK?
- How do they use it?
- How could you use it?



https://youtu.be/fiBt0wEiKh8

TAK Server Launch

File Location : /home/pi/Desktop/Stemx/TAK

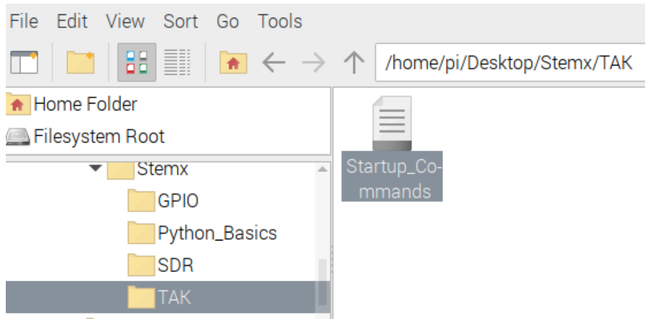
Follow the instructions in file depicted in this screenshot.

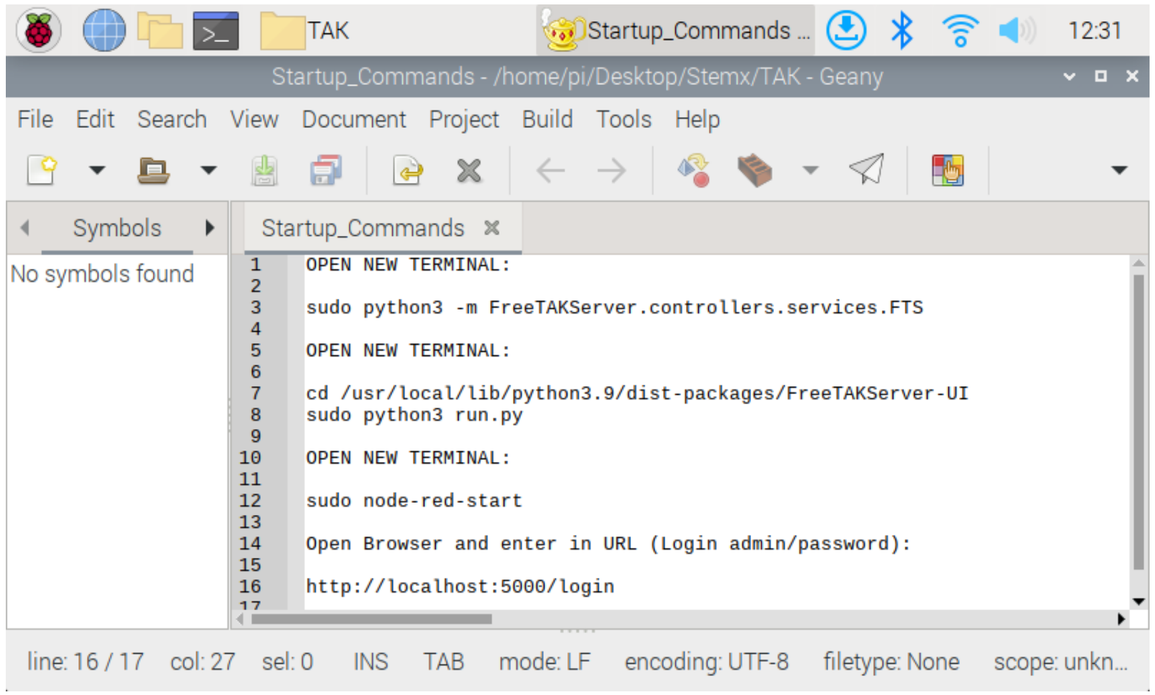
```
pi@raspberrypi:~ $ sudo python3 -m FreeTAKServer.controllers.services.FTS
```

```
pi@raspberrypi:/usr/local/lib/python3.9/dist-packages/FreeTAKServer-UI $ sudo python3 run.py
```

```
pi@raspberrypi:~ $ sudo node-red-start
```

Leave the terminals running.

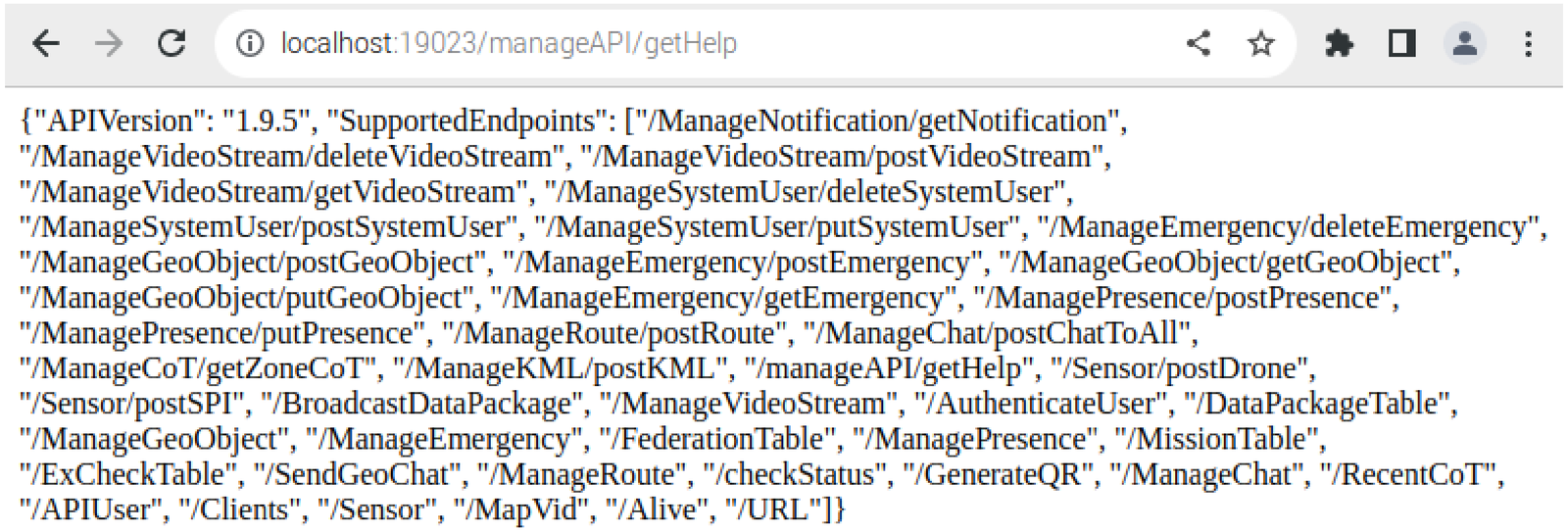




TAK Server APIs

Open Browser and Navigate to localhost:19023/manageAPI/getHelp

API endpoints will be listed as JSON object.

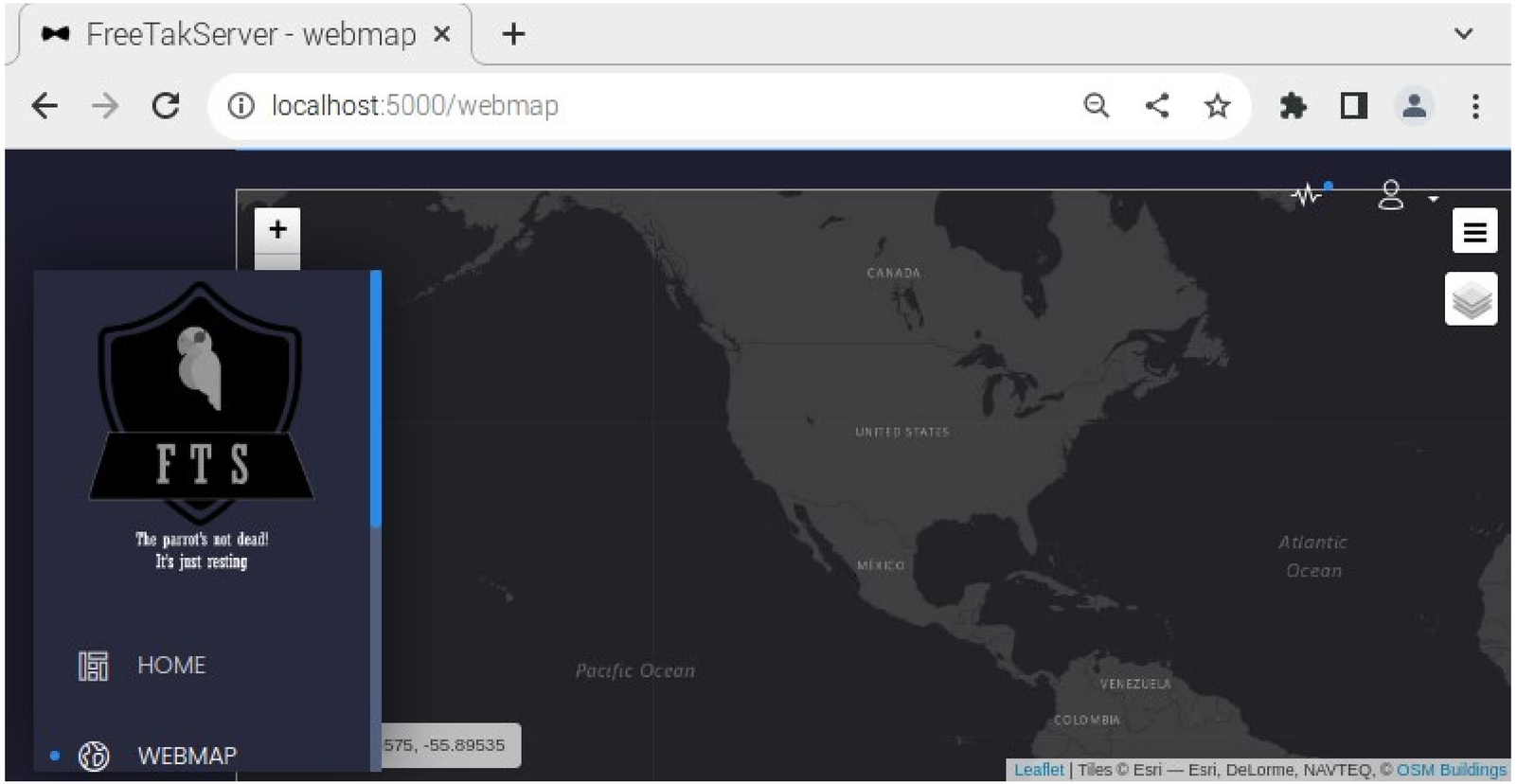


TAK Server Launch

Open Browser and Navigate to localhost:5000

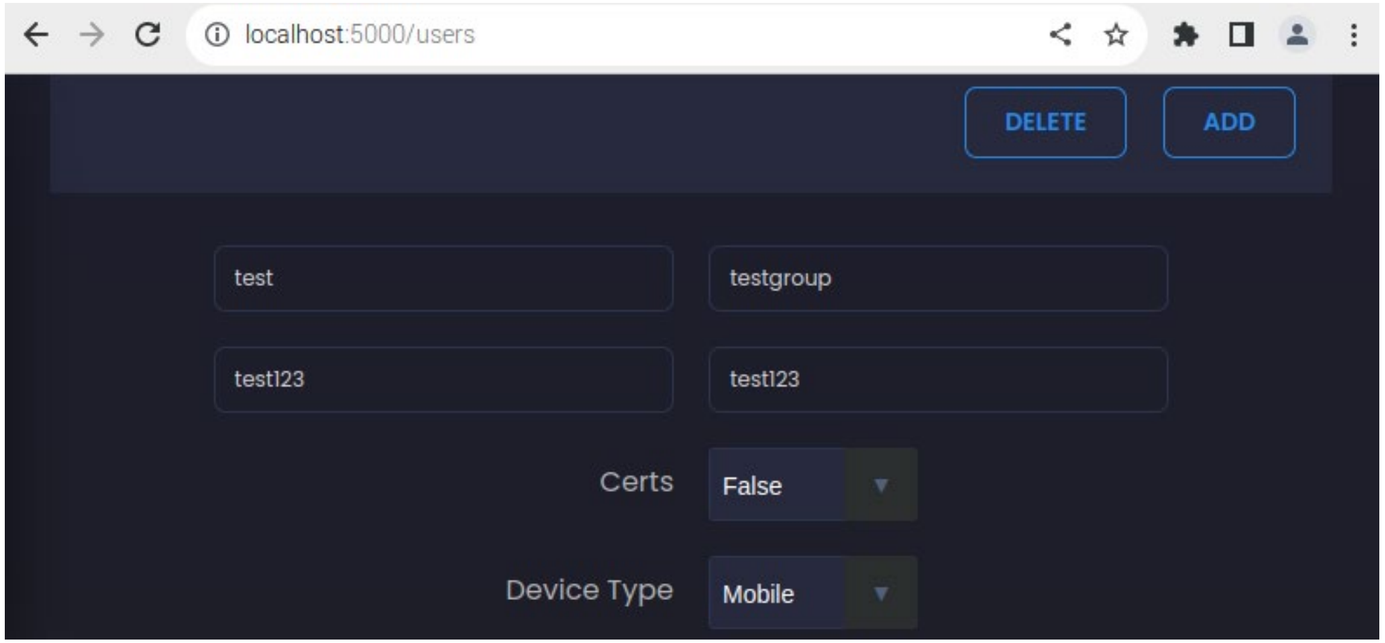
Login with admin/password

Select dropdown top left and open WEBMAP to confirm our Node Red Map is being served. This will be useful to see people connecting to our server!



TAK Create User

Create New Users within the TAK UI and assign passwords in order to make API calls to the server.



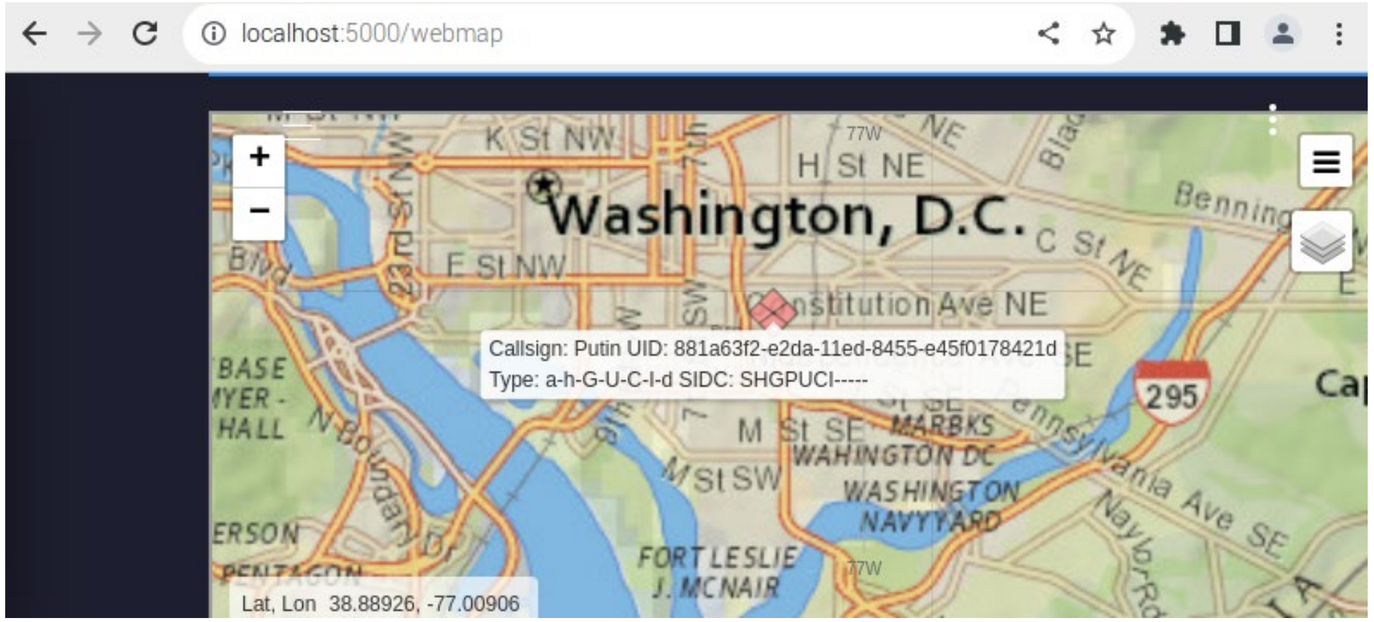
TAK POST API Example

Running this python script will send a POST API call to our server consisting of the data object depicted in the code.

```python
import requests
import json
#http://localhost:19023/manageAPI/getHelp
url = "http://localhost:19023/ManageGeoObject/postGeoObject"

# Define the CoT data
cot_data = {
  "longitude": -77.0104,
  "latitude": 38.889,
  "attitude": "hostile",
  "bearing": 132,
  "distance": 1,
  "geoObject": "Gnd Combat Infantry Sniper",
  "how": "nonCoT",
  "name": "Target",
  "timeout": 600
}

# Convert the data to JSON format
cot_json = json.dumps(cot_data)

# Define the headers
headers = {
    "Content-Type": "application/json",
    "Accept": "application/json",
    "Authorization": "Bearer test123"
}

# Make the API call
response = requests.post(url, headers=headers, data=cot_json)

# Print the response
print(response.text)
```

TAK SDR Challenge

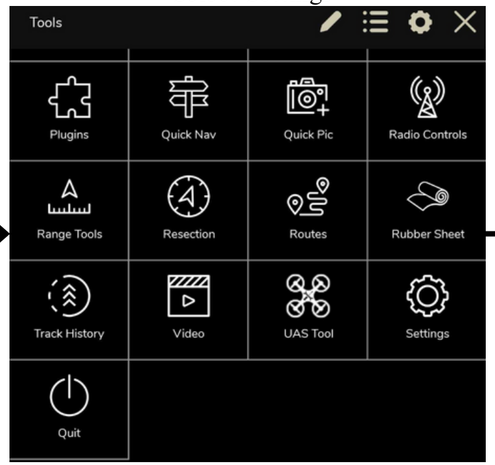
Utilize everything you have learned throughout the modules to take the Flight data output and send that info to the TAK server. You should be able to intercept all local flights and display them as COT locations within your TAK browser as well as on ATAK devices connected to the server. The application should use PUT calls to update those COT messages as new flight data comes in. Users should be able to see each flight and updated positions every 5-10 seconds on their ATAK devices.



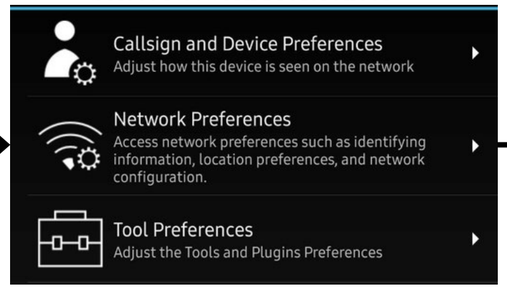
ATAK Connect

Select Hamburger dropdown

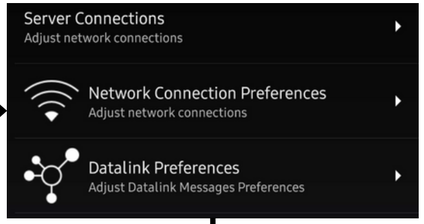
Select Settings



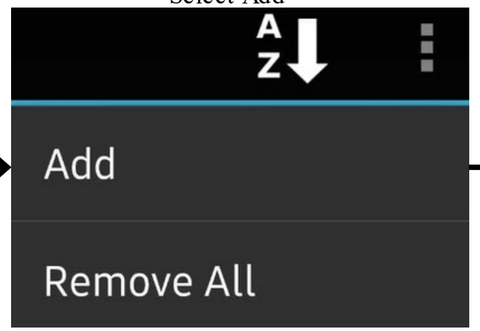
Select Network Preferences



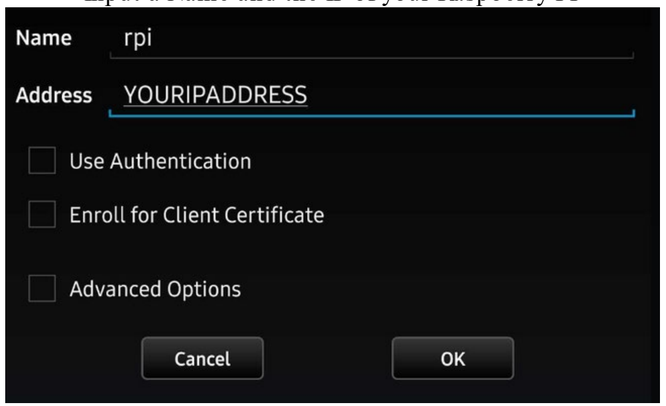
Select Server Connections



Select Add



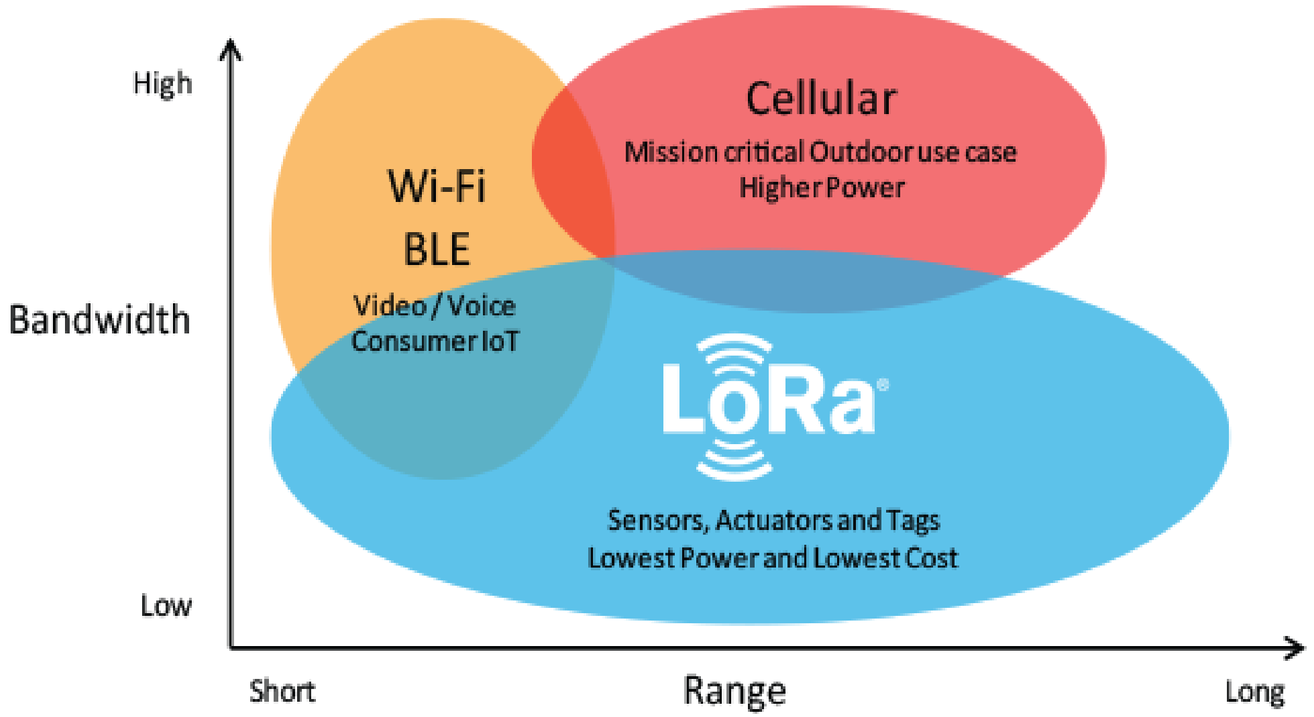
Input a Name and the IP of your Raspberry PI



LoRa Basics

- **LO**ng **Ra**nge
- Low-power wireless communication technology that enables long-range transmissions with low data rates
- Useful for things like collecting data from sensors

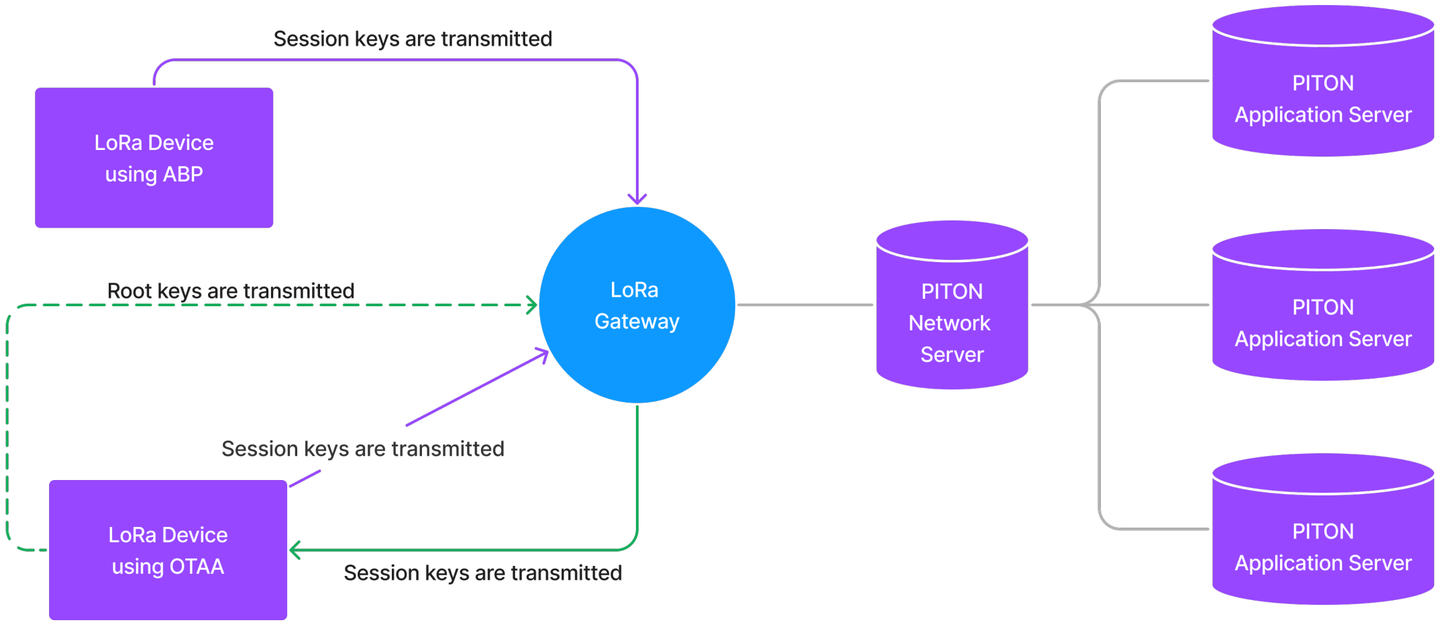
LoRa Basics



- Ultra low power
- Long range
- Deep indoor penetration
- License free spectrum
- Geolocation
- High capacity
- End-to-end security
- Firmware updates over the air
- Roaming
- Low cost

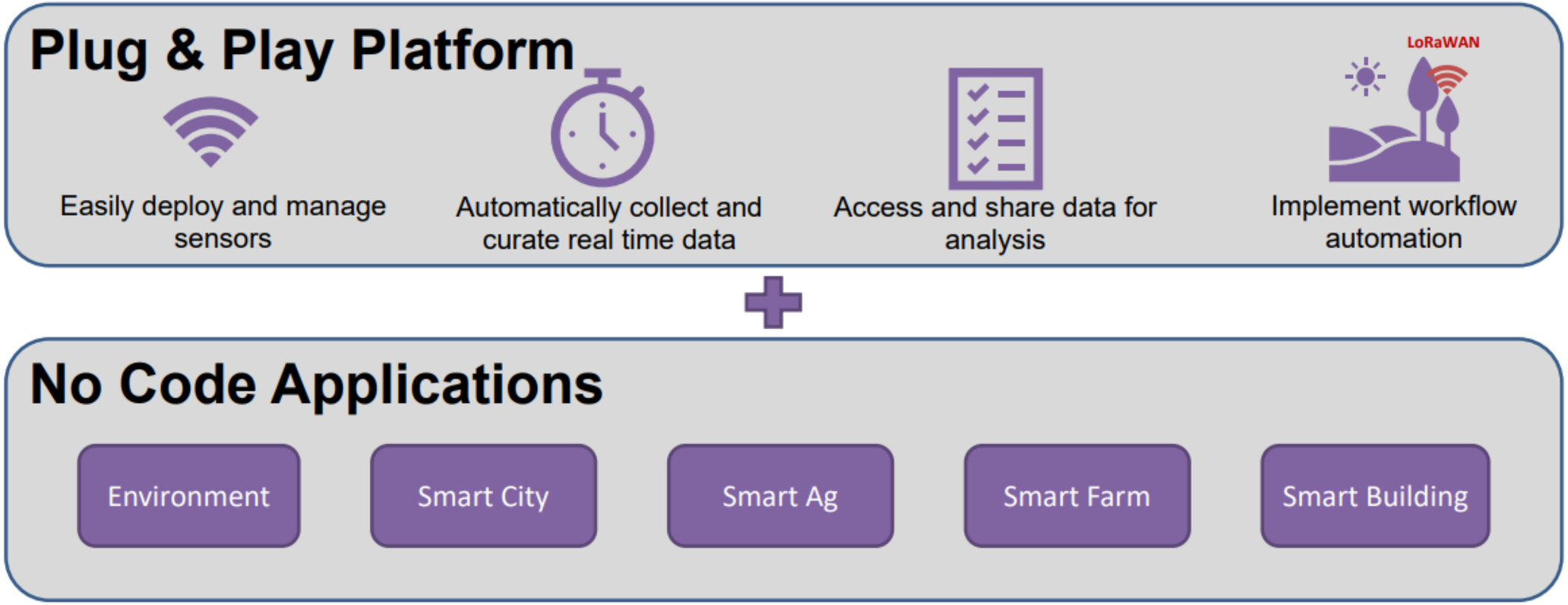
LoRaWAN

- The protocol on top of LoRa
- Has two ways to authenticate devices



PITON

- **P**latform for **I**o**T** **O**pen **N**etworks



STEMX

PITON


RSSI
-70
-80
-90
-100
-110
-120
-130
03/28 18:00
03/28 21:00
03/29 00:00
03/29 03:00
03/29 06:00
03/29 09:00
03/29 12:00
03/29 15:00
03/29 18:00
03/29 21:00
03/30 00:00
03/30 03:00
03/30 06:00
03/30 09:00
03/30 12:00
oyster-7 oyster-5 oyster-8 oyster-9 oyster-6 oyster-1 oyster-2
Signal to Noise
15
10
5
0
-5
-10
-15
-20
03/28 18:00
03/28 21:00
03/29 00:00
03/29 03:00
03/29 06:00
03/29 09:00
03/29 12:00
03/29 15:00
03/29 18:00
03/29 21:00
03/30 00:00
03/30 03:00
03/30 06:00
03/30 09:00
03/30 12:00
oyster-6 oyster-5 oyster-2 oyster-1 oyster-8 oyster-7 oyster-9

STEMX