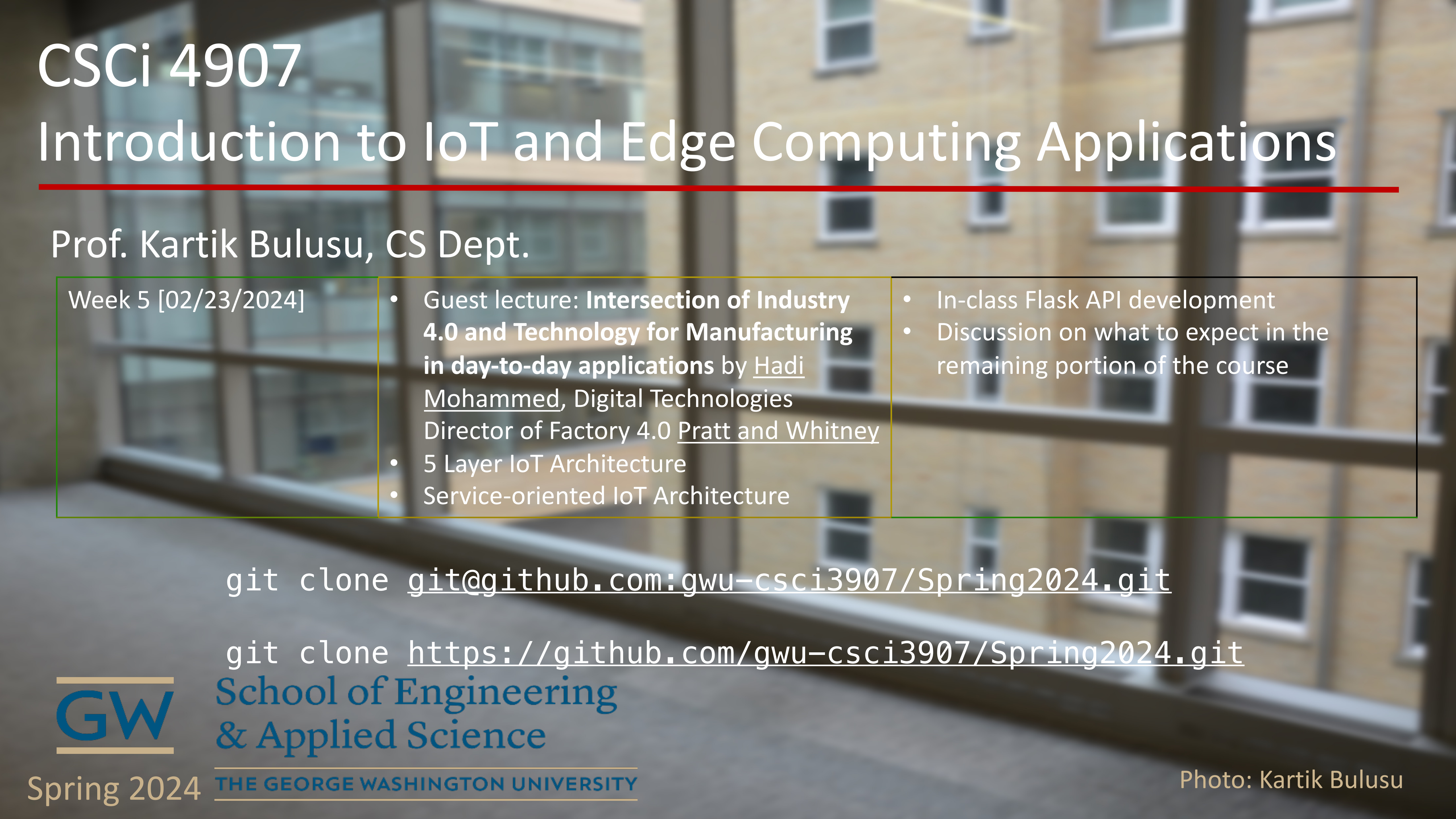


CSCi 4907

Introduction to IoT and Edge Computing Applications

Prof. Kartik Bulusu, CS Dept.

Week 5 [02/23/2024]

- Guest lecture: **Intersection of Industry 4.0 and Technology for Manufacturing in day-to-day applications** by Hadi Mohammed, Digital Technologies Director of Factory 4.0 Pratt and Whitney
- 5 Layer IoT Architecture
- Service-oriented IoT Architecture

- In-class Flask API development
- Discussion on what to expect in the remaining portion of the course

```
git clone git@github.com:gwu-csci3907/Spring2024.git

git clone https://github.com/gwu-csci3907/Spring2024.git
```

GW School of Engineering & Applied Science

THE GEORGE WASHINGTON UNIVERSITY

Spring 2024

Photo: Kartik Bulusu

Midterm projects

Midterm Project Status

Name	Project title	Hardware requirements	Status
Aleks Haskett	*Home Security System*	Pi Camera, Tracking sensor, Passive buzzer	**Approved. Needs to collect sensors**
Gerald Fattah	*Smart Animal Capture System*	IR Sensor, Pyroelectric ("Passive") InfraRed (PIR) module, Pi Camera	**Approved. Needs to collect sensors**
Jonathan Pang	*Proximity Alarm Door System (PADS)*	Pyroelectric ("Passive") InfraRed (PIR) module (HC-SR501), Bluetooth Tranceiver Module	**Approved. Needs to collect sensors**
Oliver Kristeya	*D ungeons & Dragans (D&D) Tower Roller*	Pi Camera, 3.5 in touch screen	**Need more information on 02/23, Approved on 02/26, Needs to collect sensors**
Talia Novack	*Dish Washer Helper*	Touch switch, analog heat sensor	**Approved. Need make and model numbers of the sensors**
Warren Nguyen	*Adaptive Lamp*	SenseHat, Photoresistor, Dimmable light sources	**Approved. Need make and model numbers of the sensors**
Selman Eris	*Food Scanner*	Pi Camera	**Approved. Needs to collect sensors**
Matthew Gouvin	*Plant Lighting Measurement Device*	Light sensors	**Approved. Need make and model numbers of the sensors**
Liza Mozolyuk	*Flight Tracking Interface*	SenseHat	**Need more information on 02/23, Approved on 02/26, Needs to collect sensors**
Bridget Orr	*Ukelele Tuner*	Sound sensor	**Approved. Need make and model numbers of the sensors**
Georgiana Mois	*MediTrack: Smark Medication Management*	Tilt Switch, Vibration Swtich	**Approved. Need make and model numbers of the sensors**
Alicia Ha	*Home Security Camera and Doorbell System*	Ultrasonic sensor, Pi Camera, PIR motion sensor, RGB LED, Passive Buzzer, Button	**Approved. Need make and model numbers of the sensors**
William Mai	*Cat Detector*	Pi NOIR Camera	**Approved. Needs to collect sensors**
Peter Wright	*Smart Cat Feeder*	Weight and Optical Sensor, actuator	**Approved. Need make and model numbers of the sensors**
Abdulrahman Alsaleh	*Camera by sensor detection*	Pi Camera, PIR motion sensor or Ultrasonic sensor,	**Approved. Need make and model numbers of the sensors**
Alvin Isaac	*Water Detection System*	Temperature, humidity and water level sensor	**Need more information on 02/23, Approved on 02/26, Needs to collect sensors**
Kartik Bulusu	*STREAM: **S**ensor s**T**ack fo**R** **E**nvironment**A**l **M**onitoring*	ESP 32, Barometer, GPS, SCD-30 - NDIR CO2 Temperature and Humidity Sensor	**Need more information; Unclear how he's going to pull this off!!**

Resetting the course for the next 5 weeks:

- Topics to be covered
- Weekly deliverables

Hardware:

1. ESP32
2. Cameras
3. SenseHat

App-development:

1. Flask
2. Micropython
3. Flask_Restful
4. WebSockets

Mathematics:

1. Basics of matrices
2. Applications of matrices: filters
3. Basics of Signal processing

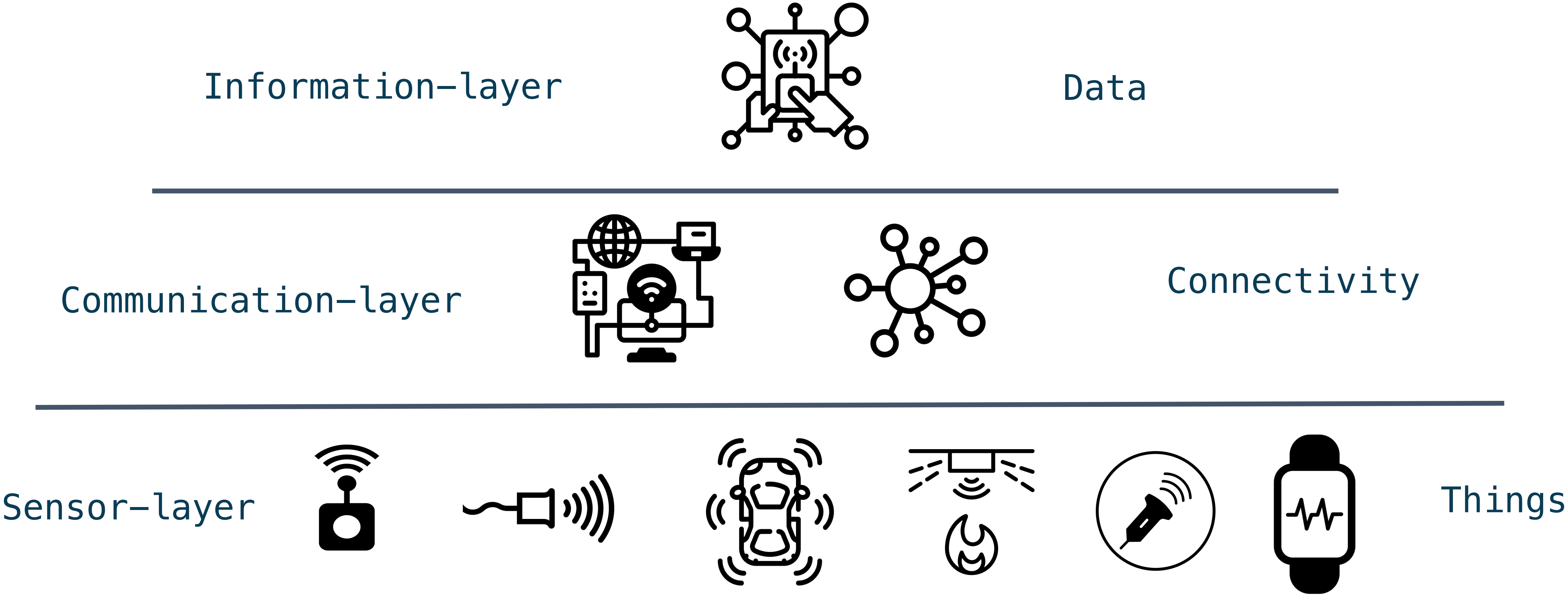
Edge computing on the Pi:
Mathematics + Python + Signal processing

Expectations on student deliverables:

1. Midterm project demo
2. Midterm project presentation
3. Midterm project report in a conference-style template
4. Weekly coding HW
5. Weekly Quizzes
6. Final project proposal
7. Final project presentation
8. Final project demo
9. Final report in a conference-style template

Building up the IoT Architecture and Ecosystem

Icon Sources:
sensor by Carolina Cani:, sensor by Pham Duy Phuong Hung, sensor by Tippawan Sookruay, sensor by Lorenzo: https://thenounproject.com/browse/icons/term/sensor
fire sensor by LAFS : https://thenounproject.com/browse/icons/term/fire-sensor/
Ultrasound by Shocho: https://thenounproject.com/browse/icons/term/ultrasound/
Network by Solikin:, Network by Tippawan: https://thenounproject.com/browse/icons/term/network
application by Chaowalit Koetchuea: https://thenounproject.com/browse/icons/term/application/



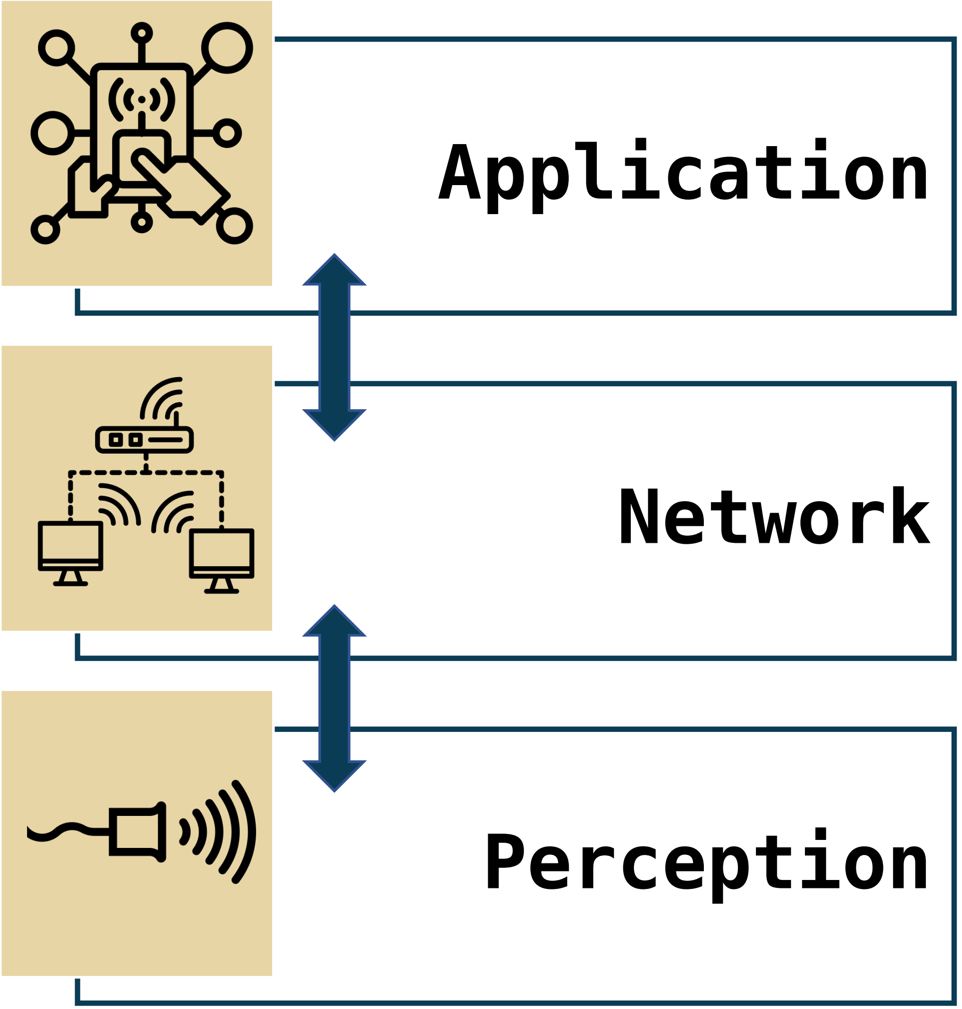
Information–layer — Data

Communication–layer — Connectivity

Sensor–layer — Things

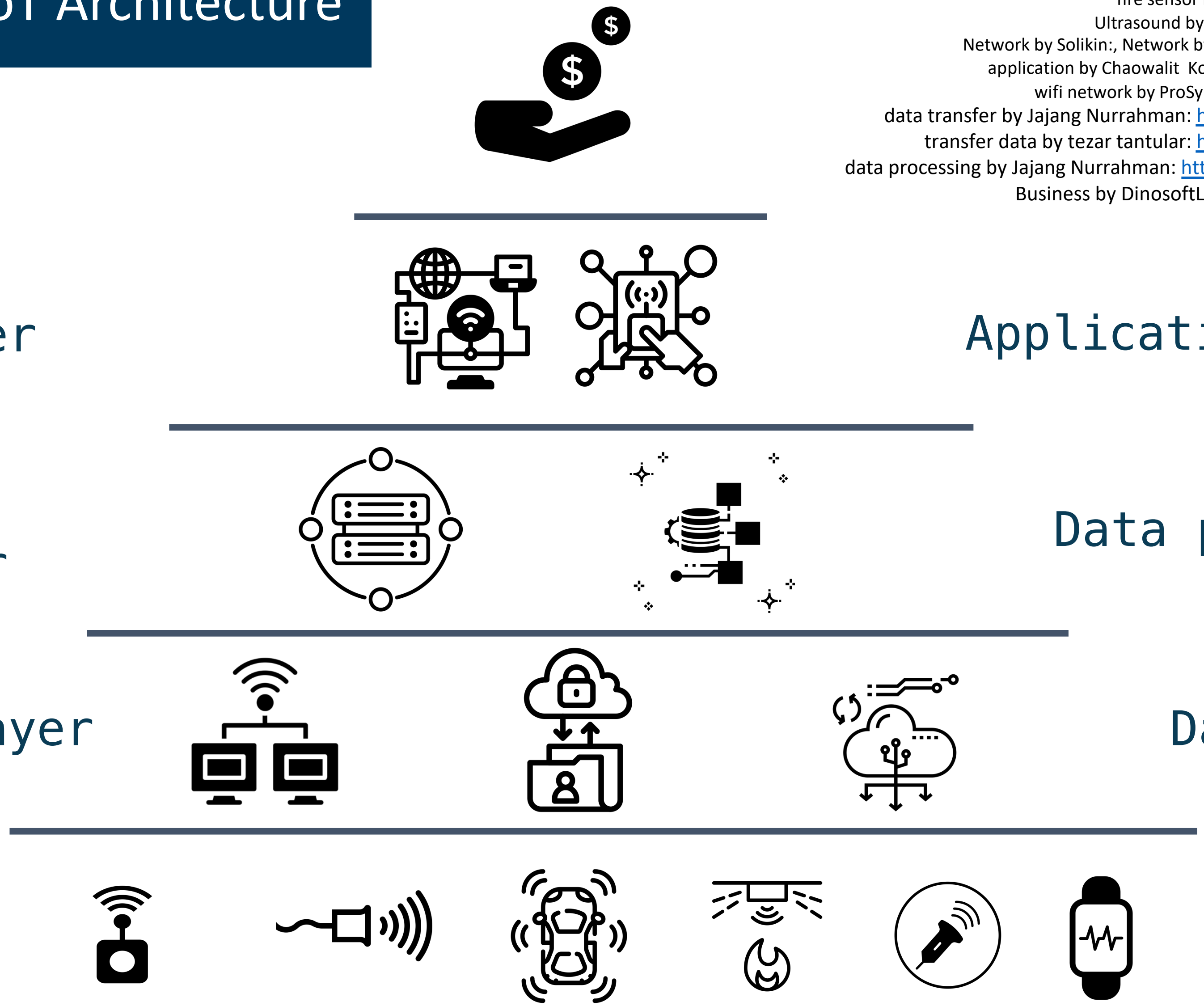
The 3-Layer IoT Architecture

Icon Sources:
sensor by Carolina Cani:, sensor by Pham Duy Phuong Hung, sensor by Tippawan Sookruay, sensor by Lorenzo:
https://thenounproject.com/browse/icons/term/sensor
wifi network by Matthias Hartmann:: https://thenounproject.com/browse/icons/term/wifi-network/
application by Chaowalit Koetchuea: https://thenounproject.com/browse/icons/term/application/



Building the next IoT Architecture

Icon Sources:
sensor by Carolina Cani:, sensor by Pham Duy Phuong Hung, sensor by Tippawan Sookruay, sensor by Lorenzo: https://thenounproject.com/browse/icons/term/sensor
fire sensor by LAFS : https://thenounproject.com/browse/icons/term/fire-sensor/
Ultrasound by Shocho: https://thenounproject.com/browse/icons/term/ultrasound/
Network by Solikin:, Network by Tippawan: https://thenounproject.com/browse/icons/term/network
application by Chaowalit Koetchuea: https://thenounproject.com/browse/icons/term/application
wifi network by ProSymbols: https://thenounproject.com/browse/icons/term/wifi-network/
data transfer by Jajang Nurrahman: https://thenounproject.com/browse/icons/term/data-transfer/
transfer data by tezar tantular: https://thenounproject.com/browse/icons/term/transfer-data/
data processing by Jajang Nurrahman: https://thenounproject.com/browse/icons/term/data-processing
Business by DinosoftLab: https://thenounproject.com/browse/icons/term/business/



Business-layer

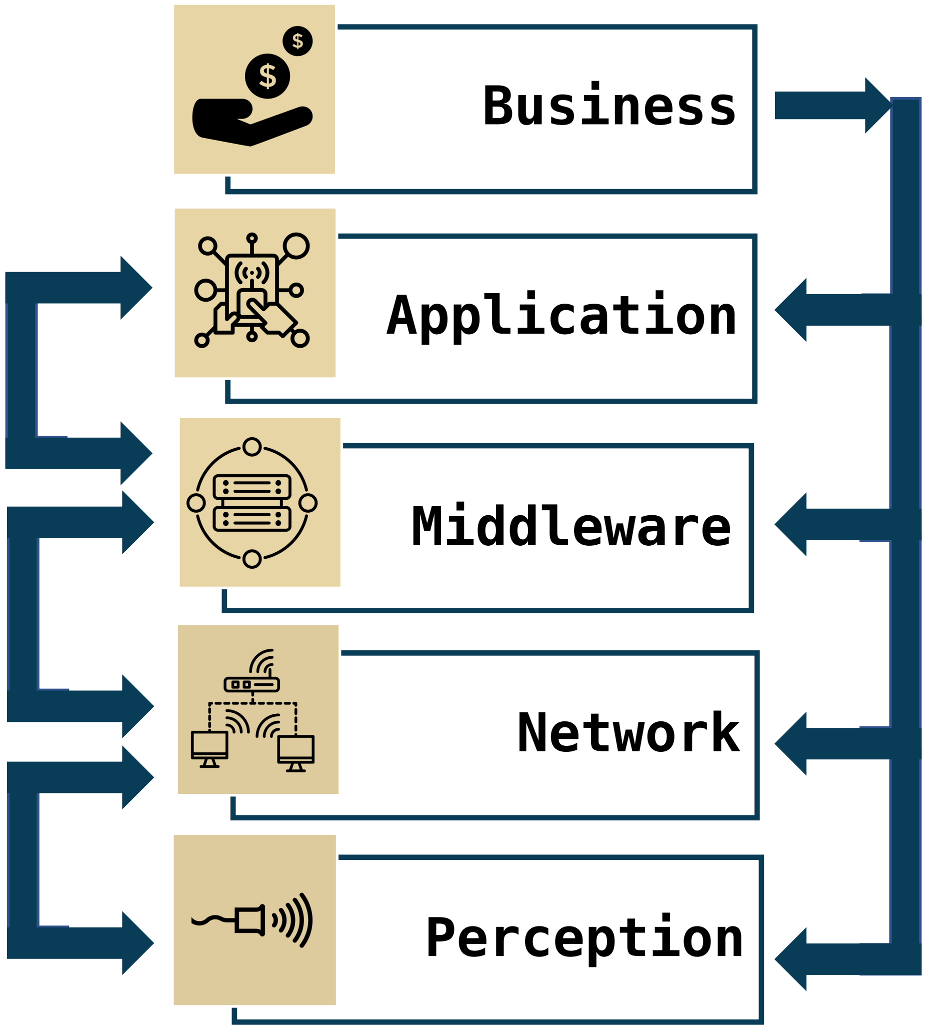
Information-layer — Application

Processing- or Middleware-layer — Data processing

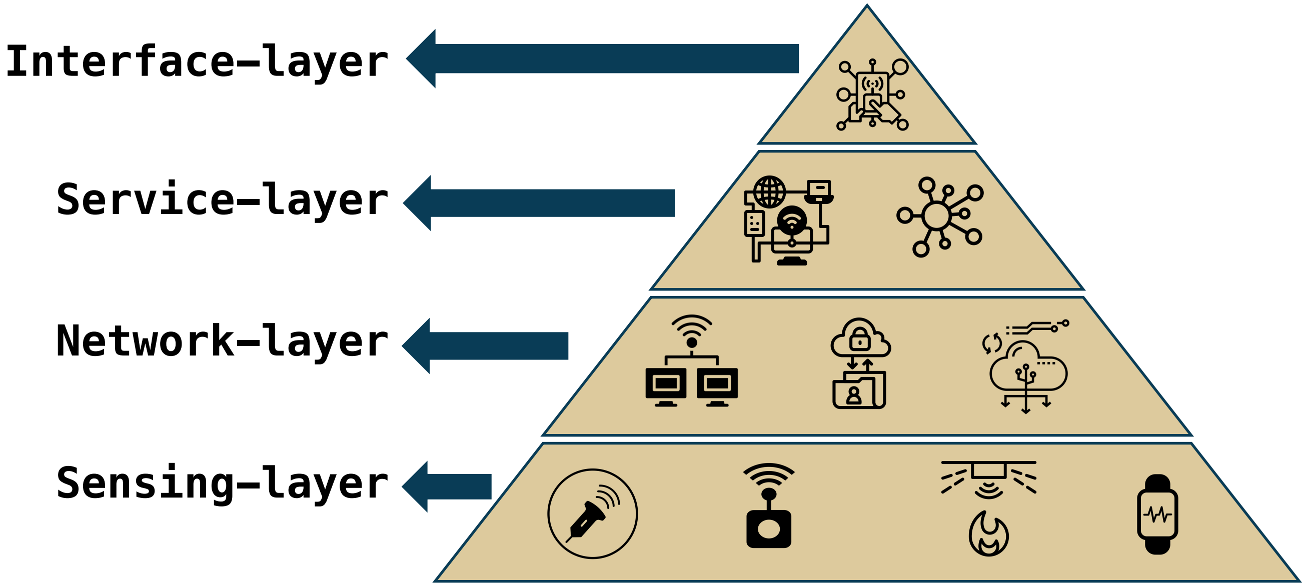
Communication-layer — Data transfer

Sensor-layer — Things

The 5-Layer IoT Architecture



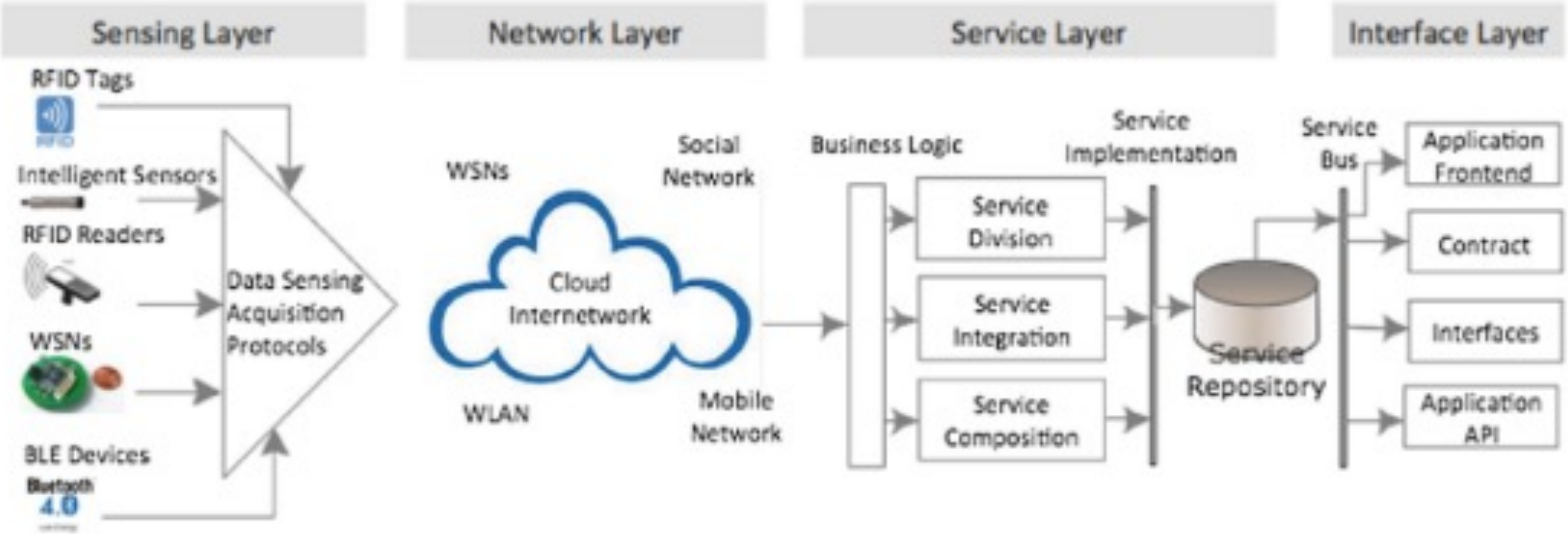
Service-oriented IoT Architecture



Sources:
sensor by Carolina Cani:, sensor by Pham Duy Phuong Hung, sensor by Tippawan Sookruay, sensor by Lorenzo: https://thenounproject.com/browse/icons/term/sensor
wifi network by Matthias Hartmann:: https://thenounproject.com/browse/icons/term/wifi-network/
application by Chaowalit Koetchuea: https://thenounproject.com/browse/icons/term/application/
IoT Architecture layers: https://www.startertutorials.com/blog/iot-architecture-layers.html

Service-oriented IoT Architecture

Sources:
IoT Architecture layers: https://www.startertutorials.com/blog/iot-architecture-layers.html



Source:
https://flask.palletsprojects.com/en/2.2.x/
https://en.wikipedia.org/wiki/Plotly
javascript file by SAM Designs from https://thenounproject.com/browse/icons/term/javascript-file/
html by tezar tantular from https://thenounproject.com/browse/icons/term/html/

Demo project:
Create a Flask API for IoT Applications
[Graded Lab Activity]

1. Python 3 with any IDE or terminal
2. Familiarity with flask API
3. Basic HTML with JavaScript
4. Familiarity with Plotly

Source:
https://flask.palletsprojects.com/en/2.2.x/
https://en.wikipedia.org/wiki/Plotly

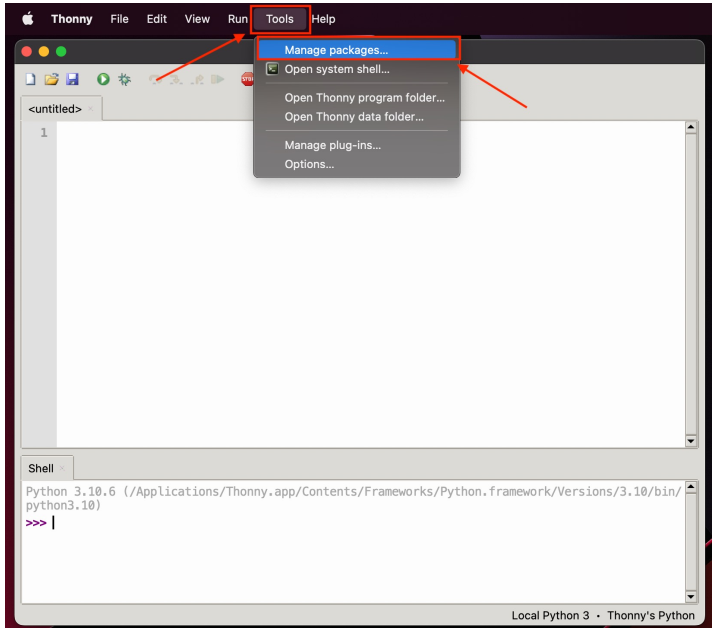
Flask
web development,
one drop at a time

- **Flask** is a micro web framework written in Python.
- **Components:**
  - ***Werkzeug:*** Utility library for Web Server Gateway Interface (WSGI) applications
  - ***Jinja:*** Template engine similar to Django that handles templates in a sandbox
  - ***MarkupSafe:*** String handling library
  - ***ItsDangerous:*** Safe data serialization library

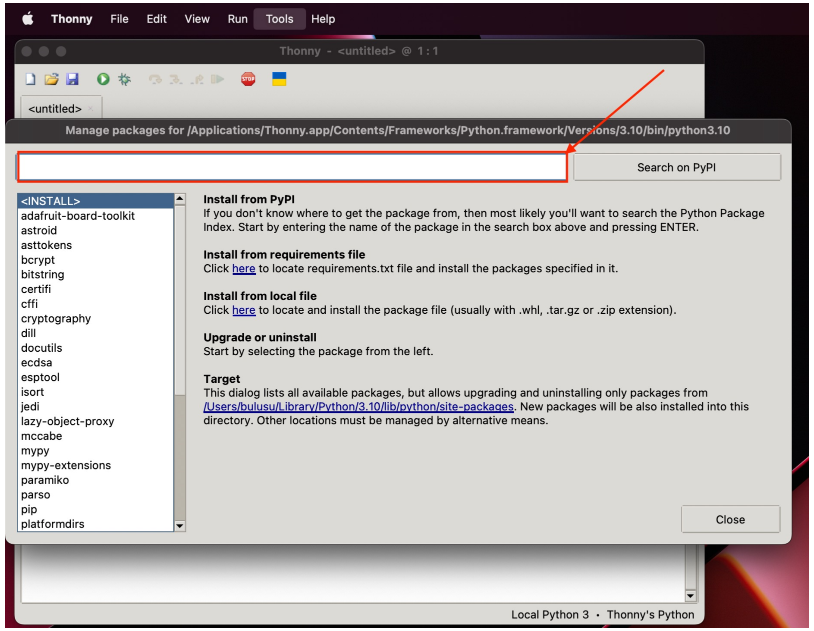
plotly

- **Plotly** provides online graphing, analytics, and statistics tools
  - for individuals and collaboration,
  - as well as scientific graphing libraries
    for Python, R, MATLAB, Perl, Julia, Arduino, and REST.
- **Open-source products:**
  - ***Dash:*** Open-source framework for building web-based analytic applications.
  - ***Chart Studio Cloud:*** Free, online tool for interactive graphics
  - ***Plotly.js data visualization JavaScript library*** for creating graphs and powers Plotly.py for Python, as well as Plotly.R for R, MATLAB, Node.js, Julia, and Arduino and a REST API

Installing packages in Thonny



```
Python 3.10.6 (/Applications/Thonny.app/Contents/Frameworks/Python.framework/Versions/3.10/bin/python3.10)
>>> |
```



Manage packages for /Applications/Thonny.app/Contents/Frameworks/Python.framework/Versions/3.10/bin/python3.10

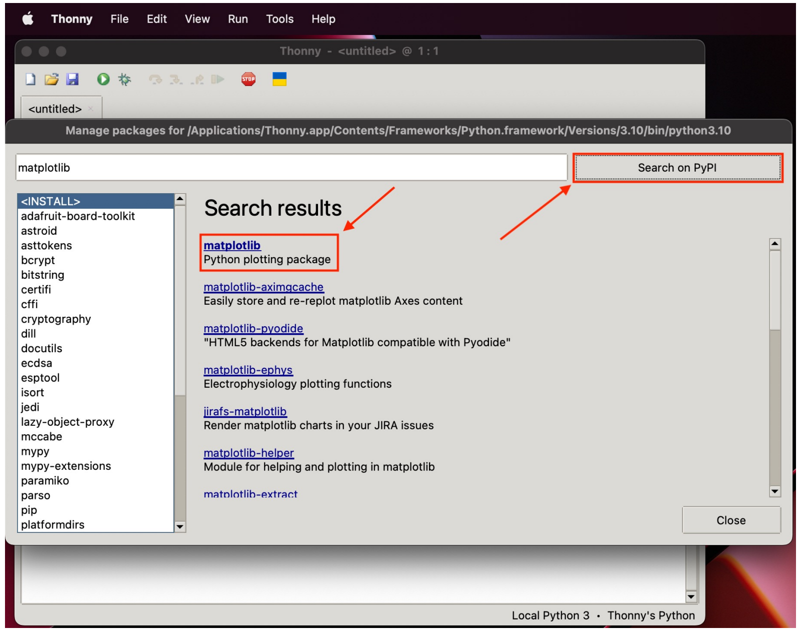
Install from PyPI
If you don't know where to get the package from, then most likely you'll want to search the Python Package Index. Start by entering the name of the package in the search box above and pressing ENTER.

Install from requirements file
Click here to locate requirements.txt file and install the packages specified in it.

Install from local file
Click here to locate and install the package file (usually with .whl, .tar.gz or .zip extension).

Upgrade or uninstall
Start by selecting the package from the left.

Target
This dialog lists all available packages, but allows upgrading and uninstalling only packages from /Users/bulusu/Library/Python/3.10/lib/python/site-packages. New packages will be also installed into this directory. Other locations must be managed by alternative means.



Manage packages for /Applications/Thonny.app/Contents/Frameworks/Python.framework/Versions/3.10/bin/python3.10

matplotlib

Search results

matplotlib
Python plotting package

matplotlib-aximgcache
Easily store and re-replot matplotlib Axes content

matplotlib-pyodide
"HTML5 backends for Matplotlib compatible with Pyodide"

matplotlib-ephys
Electrophysiology plotting functions

jirafs-matplotlib
Render matplotlib charts in your JIRA issues

matplotlib-helper
Module for helping and plotting in matplotlib

matplotlib-extract

Check or install the following libraries in Python 3.10.11: (Note these are the bare minimum versions)

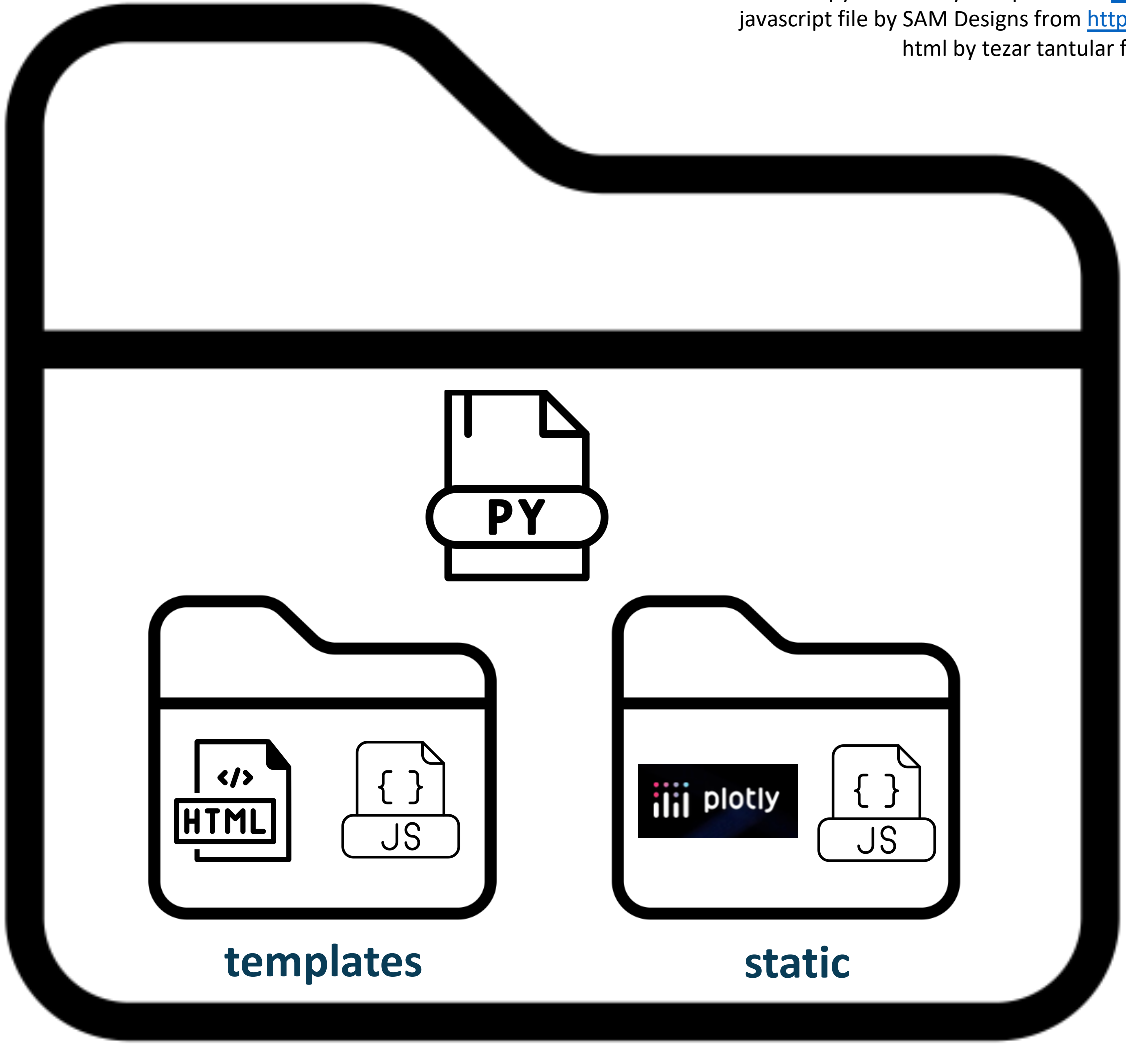
flask	0.12.1
plotly	4.14.3
simplejson	3.10.1
pandas	0.25.3
numpy	1.18.5
matplotlib	3.0.3
datetime	5.0
time	1.0.0

Alternative:

```
>>> pip install <package_name>
>>> # Or install using pip3
>>> # in your virtual environment
```

Source:
Folder by Colourcreatype from https://thenounproject.com/browse/icons/term/folder/
python file by Vectplus from https://thenounproject.com/browse/icons/term/python-file/
javascript file by SAM Designs from https://thenounproject.com/browse/icons/term/javascript-file/
html by tezar tantular from https://thenounproject.com/browse/icons/term/html/
https://en.wikipedia.org/wiki/Plotly

Essentials: Files and folders needed to create the intended APP



templates

static

Skeleton of the Python program for a flask-server on the Raspberry Pi

Source: https://flask.palletsprojects.com/en/2.2.x/

Step-1:
Initialize variables and setup Flask instance

```
from flask import Flask, request, render_template
from flask_restful import Resource, Api, reqparse, inputs
import pandas as pd
import json
import plotly
import plotly.subplots
import plotly.express as px
import random
import numpy as np
import matplotlib.pyplot as plt
import time
import datetime
import logging
import thing_file
```

Import libraries that are relevant for interaction with the Raspberry Pi hardware such as

flask,
json,
plotly and its derivatives
pandas
numpy
matplotlib etc.

Source:
Folder by Colourcreatype from https://thenounproject.com/browse/icons/term/folder/
python file by Vectplus from https://thenounproject.com/browse/icons/term/python-file/
javascript file by SAM Designs from https://thenounproject.com/browse/icons/term/javascript-file/
html by tezar tantular from https://thenounproject.com/browse/icons/term/html/
https://en.wikipedia.org/wiki/Plotly

Step-2: Initialize variables and setup Flask instance

```
# ======== Initialize Logging ==========

# Global logging configuration
logging.basicConfig(level=logging.WARNING)

# Logger for this module
logger = logging.getLogger('main')

# Debugging for this file.
logger.setLevel(logging.INFO)
```

Provides warning on any components that work within flask or other imported libraries

Reference: https://docs.python.org/3/howto/logging.html

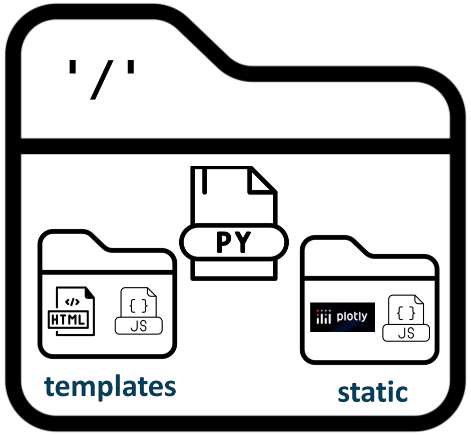
thing_file.py
Logging libraries and a few more are place in this custom library provided to you

```
# ==== Flask & Flask-RESTful instance variables ====

# Core Flask app.
app = Flask(__name__)

# ==== Flask & Flask-Restful Related Functions ====
# @app.route applies to the core Flask instance (app).
# Here we are serving a simple web page.
@app.route('/' + thing_file.thing_name)
```

Initiate flask server-related variables and functions



Step-3: Create a function notdash() to return JSON-formatted data to an html frontend

```
def notdash():
    global data
    data = {
        'timeT': [],
        'Voltage': []
    }
```

Dictionary of empty lists that get appended with data

```
    # Create the graph with subplots
        fig = plotly.tools.make_subplots(rows=1, cols=1, vertical_spacing=0.2)
        fig['layout']['margin'] = {
            'l': 30, 'r': 10, 'b': 30, 't': 10
        }
```

Dictionary of figure layout that is transferred to the html frontend with plotly, JavaScript embedded in it

```
    for i in range(20):
            data['Voltage'].append(random.randint(0, 100))
            data['timeT'].append(timeT)

            fig.append_trace({
                'x': data['timeT'],
                'y': data['Voltage’],
                 'mode': 'lines+markers',
                'type': 'scatter'
            }, 1, 1)
```

Loop to generate random data and plot it in a trace that is transferred to the html frontend with plotly, JavaScript embedded in it

```
graphJSON = json.dumps(fig, cls=plotly.utils.PlotlyJSONEncoder)
return render_template('notdash.html', graphJSON=graphJSON)
```

json.dumps will convert a subset of Python objects into a json
render_template tells Flask to use an HTML template

Step-5
Create a function notdash() to return JSON-formatted data to an html frontend

```
if __name__ == '__main__':

    # If you have debug=True and receive
    # the error "OSError: [Errno 8] Exec format error", then:
    # remove the execuition bit on this file from a Terminal, ie:
    # chmod -x flask_api_server.py
    #
    # Flask GitHub Issue:
    # https://github.com/pallets/flask/issues/3189

     app.run(host="0.0.0.0", debug=True)
```

Creates entry point into the program and executes app.run() in debug mode.

debug=False:
translates to developer mode

app.run() renders the webpage with data plots on a

local host:
127.0.0.1:5000

Port:5000 is a default

The host address can be changed to the IP address of the server.

Skeleton of the the basic HTML code to display data from your flask-app

Sources:
https://plotly.com
API by Vectors Point from https://thenounproject.com/browse/icons/term/api/
json by ME from https://thenounproject.com/browse/icons/term/json/
javascript file by SAM Designs from https://thenounproject.com/browse/icons/term/javascript-file/
html by tezar tantular from https://thenounproject.com/browse/icons/term/html/
https://towardsdatascience.com/web-visualization-with-plotly-and-flask-3660abf9c946

```
<!doctype html>
<html>

<head>
        <meta http-equiv="refresh" content="10">
</head>
```

Will refresh the page every 10 seconds

```
<body>
        <h1>Prof. Kartik Bulusu’s sensor data</h1>
        <div id='chart' class='chart'”></div>
</body>
```

Webpage title etc

```
<!-- <script src='https://cdn.plot.ly/plotly-latest.min.js'></script> -->

<script src='/static/plotly-latest.min.js'></script>
```

Location of plotly-latest.min.js

To download: https://plotly.com/javascript/getting-started/

```
<script type='text/javascript'>
        var graphs = {{graphJSON | safe}};
        Plotly.plot('chart',graphs,{});
</script>
```

{{graphJSON | safe}}: Injects a variable that came from the server directly in the JavaScript code.

Plotly.plot(): Creates a **line chart** drawn into a <div> element on the page, with data from **graphs** with **layout** provided by the server

```
</html>
```