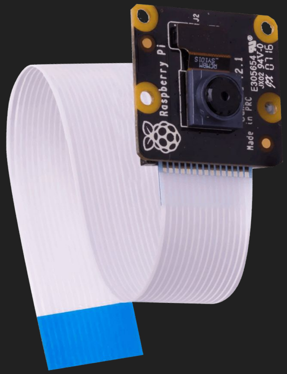
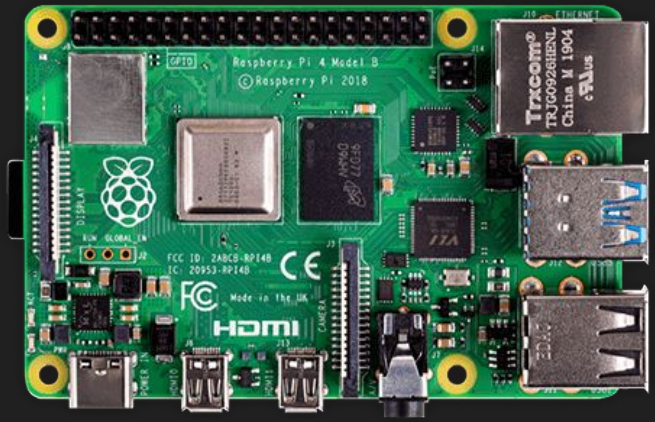
Visitor-Counter



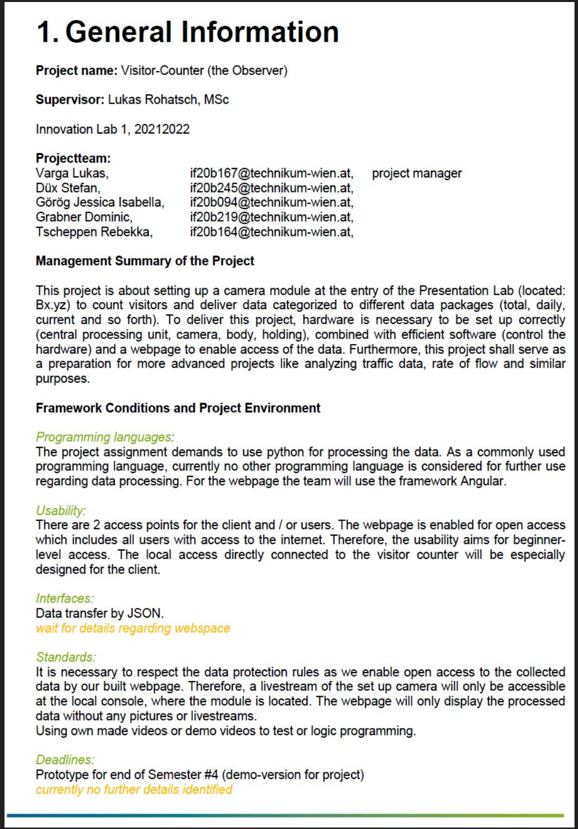


Stefan Düx | Jessica Isabella Görög | Dominic Grabner | Rebekka Tscheppen | Lukas Varga

Visitor-Counter | Group 29 | BIF4 | Innovation Lab 2 | “the Observer”

Description

- data recording + analysis via raspberry pi 4 + camera module
- person recognition to count visitors at Presentation Lab (FH Technikum)
- adapted with interface, web-application and further data processing



1. General Information

Project name: Visitor-Counter (the Observer)

Supervisor: Lukas Rohatsch, MSc

Innovation Lab 1, 20212022

Projectteam:

Varga Lukas,	if20b167@technikum-wien.at,	project manager
Düx Stefan,	if20b245@technikum-wien.at,	
Görög Jessica Isabella,	if20b094@technikum-wien.at,	
Grabner Dominic,	if20b219@technikum-wien.at,	
Tscheppen Rebekka,	if20b164@technikum-wien.at,	

Management Summary of the Project

This project is about setting up a camera module at the entry of the Presentation Lab (located: Bx.yz) to count visitors and deliver data categorized to different data packages (total, daily, current and so forth). To deliver this project, hardware is necessary to be set up correctly (central processing unit, camera, body, holding), combined with efficient software (control the hardware) and a webpage to enable access of the data. Furthermore, this project shall serve as a preparation for more advanced projects like analyzing traffic data, rate of flow and similar purposes.

Framework Conditions and Project Environment

Programming languages:
The project assignment demands to use python for processing the data. As a commonly used programming language, currently no other programming language is considered for further use regarding data processing. For the webpage the team will use the framework Angular.

Usability:
There are 2 access points for the client and / or users. The webpage is enabled for open access which includes all users with access to the internet. Therefore, the usability aims for beginner-level access. The local access directly connected to the visitor counter will be especially designed for the client.

Interfaces:
Data transfer by JSON.
wait for details regarding webspace

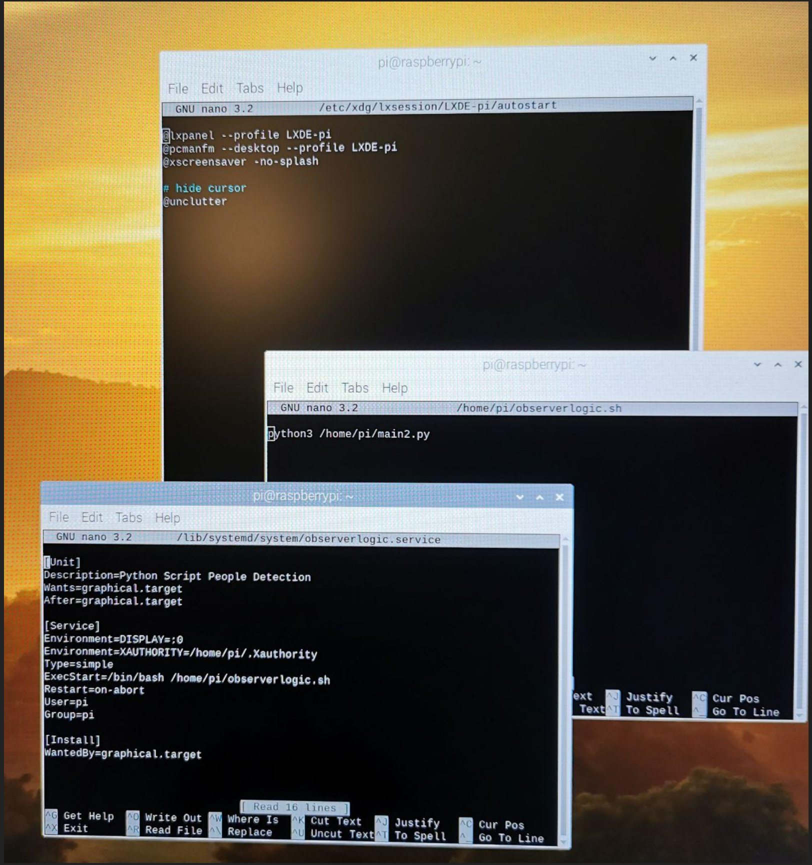
Standards:
It is necessary to respect the data protection rules as we enable open access to the collected data by our built webpage. Therefore, a livestream of the set up camera will only be accessible at the local console, where the module is located. The webpage will only display the processed data without any pictures or livestreams.
Using own made videos or demo videos to test or logic programming.

Deadlines:
Prototype for end of Semester #4 (demo-version for project)
currently no further details identified

Status Quo

- Autostart
- Logic (class based / multiprocessing)
- Interface
- Mqtt
- Open steps

Autostart

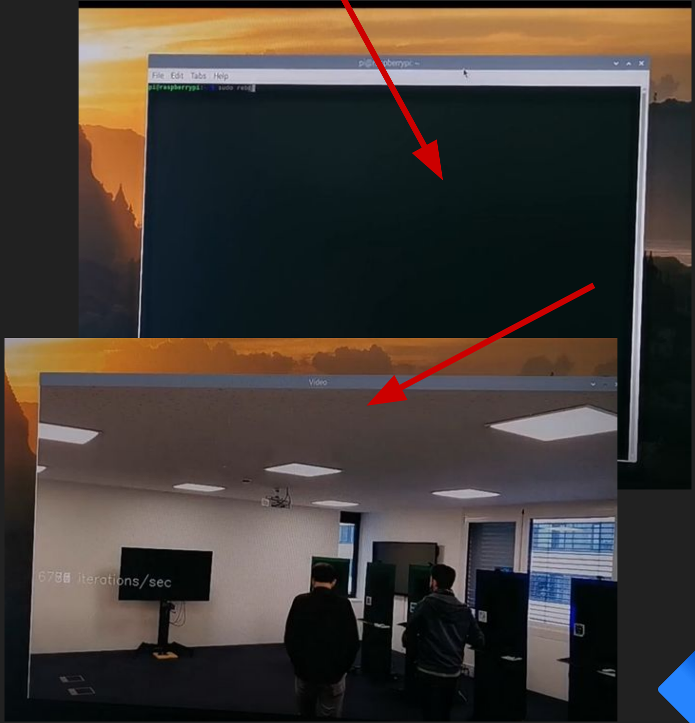


```
pi@raspberrypi: ~
File  Edit  Tabs  Help
  GNU nano 3.2            /etc/xdg/lxsession/LXDE-pi/autostart

@lxpanel --profile LXDE-pi
@pcmanfm --desktop --profile LXDE-pi
@xscreensaver -no-splash

# hide cursor
@unclutter
```

```
pi@raspberrypi: ~
File  Edit  Tabs  Help
  GNU nano 3.2            /home/pi/observerlogic.sh

python3 /home/pi/main2.py
```

```
pi@raspberrypi: ~
File  Edit  Tabs  Help
  GNU nano 3.2       /lib/systemd/system/observerlogic.service

[Unit]
Description=Python Script People Detection
Wants=graphical.target
After=graphical.target

[Service]
Environment=DISPLAY=:0
Environment=XAUTHORITY=/home/pi/.Xauthority
Type=simple
ExecStart=/bin/bash /home/pi/observerlogic.sh
Restart=on-abort
User=pi
Group=pi

[Install]
WantedBy=graphical.target

                         [ Read 16 lines ]
^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text   ^J Justify   ^C Cur Pos
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell  ^_ Go To Line
```

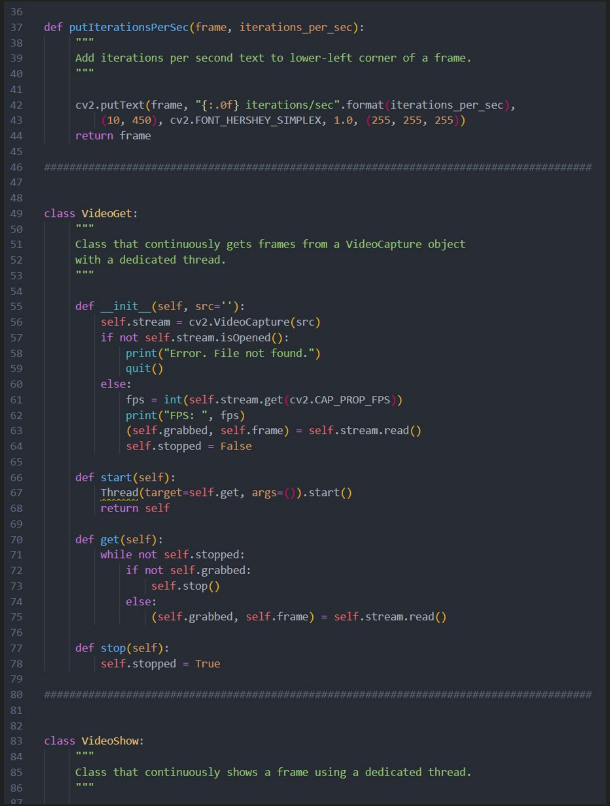
sudo reboot



```
pi@raspberrypi: ~ $ sudo rebo
```

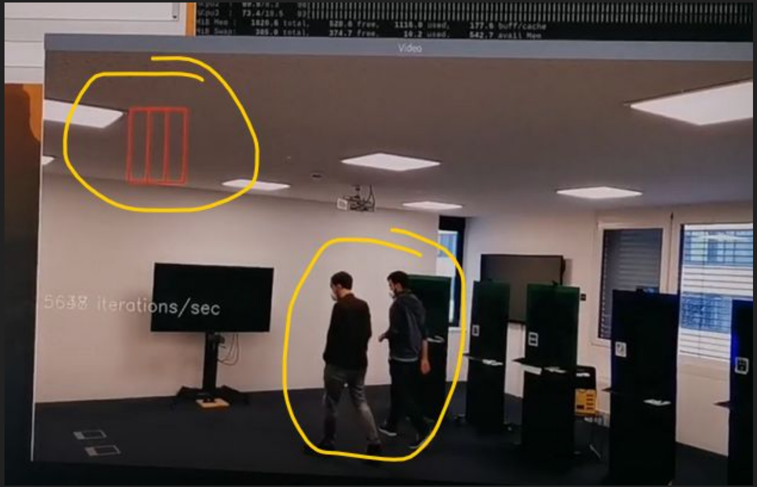
```
6788 iterations/sec
```

Logic (class based)



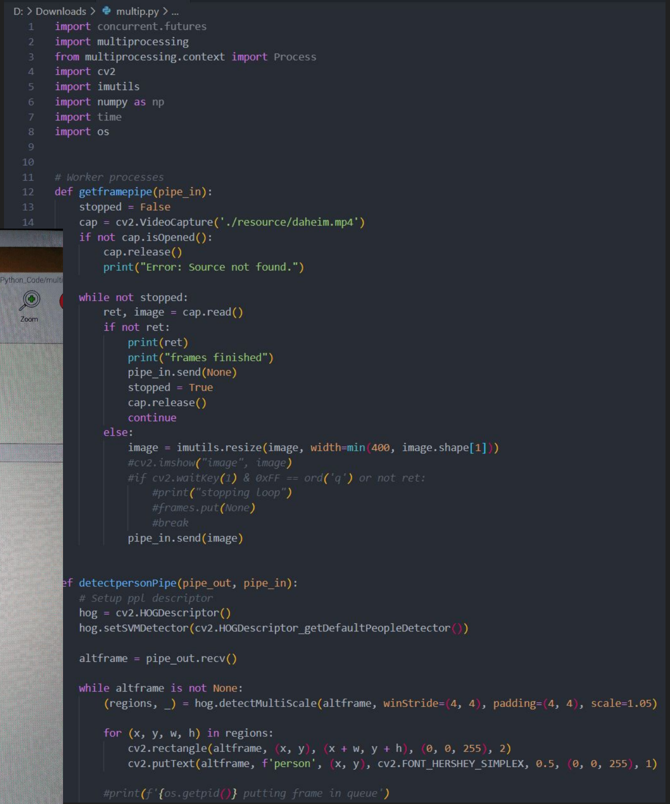
```python

def putIterationsPerSec(frame, iterations_per_sec):
    """
    Add iterations per second text to lower-left corner of a frame.
    """

    cv2.putText(frame, "{:.0f} iterations/sec".format(iterations_per_sec),
        (10, 450), cv2.FONT_HERSHEY_SIMPLEX, 1.0, (255, 255, 255))
    return frame

########################################################################################


class VideoGet:
    """
    Class that continuously gets frames from a VideoCapture object
    with a dedicated thread.
    """

    def __init__(self, src=''):
        self.stream = cv2.VideoCapture(src)
        if not self.stream.isOpened():
            print("Error. File not found.")
            quit()
        else:
            fps = int(self.stream.get(cv2.CAP_PROP_FPS))
            print("FPS: ", fps)
            (self.grabbed, self.frame) = self.stream.read()
            self.stopped = False

    def start(self):
        Thread(target=self.get, args=()).start()
        return self

    def get(self):
        while not self.stopped:
            if not self.grabbed:
                self.stop()
            else:
                (self.grabbed, self.frame) = self.stream.read()

    def stop(self):
        self.stopped = True

########################################################################################


class VideoShow:
    """
    Class that continuously shows a frame using a dedicated thread.
    """
```

- accepting speed for pi
- problem: distorted person recognition drawing/detecting

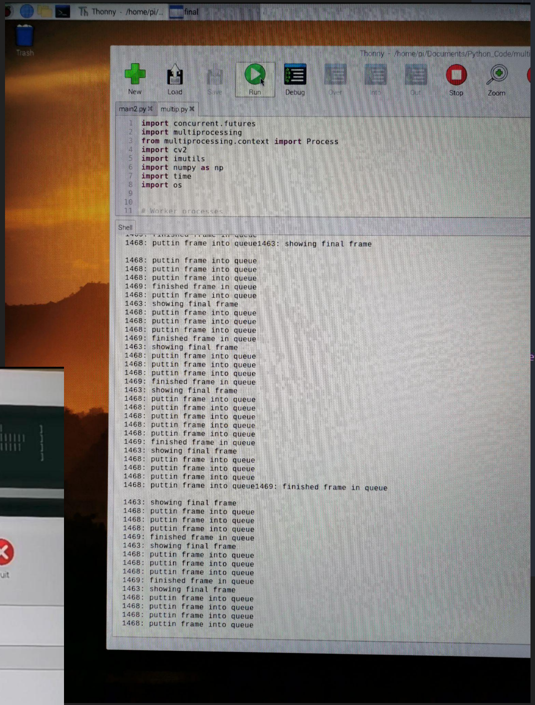


Logic (multiprocessing)

- python multiprocessing module to utilize necessary speed
- problem: low speed but full usage of all cores

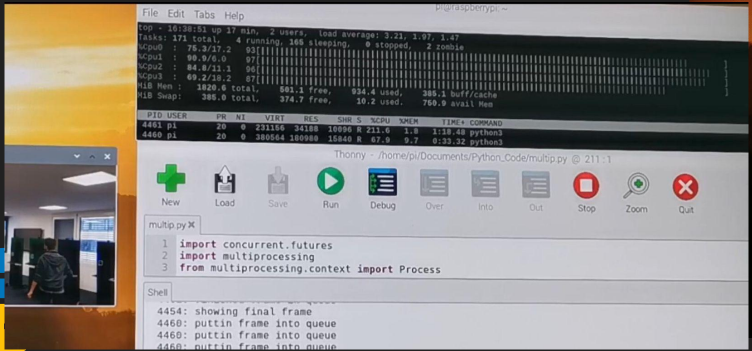

D: > Downloads > multip.py > ...

```python
import concurrent.futures
import multiprocessing
from multiprocessing.context import Process
import cv2
import imutils
import numpy as np
import time
import os


# Worker processes
def getframepipe(pipe_in):
    stopped = False
    cap = cv2.VideoCapture('./resource/daheim.mp4')
    if not cap.isOpened():
        cap.release()
        print("Error: Source not found.")

    while not stopped:
        ret, image = cap.read()
        if not ret:
            print(ret)
            print("frames finished")
            pipe_in.send(None)
            stopped = True
            cap.release()
            continue
        else:
            image = imutils.resize(image, width=min(400, image.shape[1]))
            #cv2.imshow("image", image)
            #if cv2.waitKey(1) & 0xFF == ord('q') or not ret:
                #print("stopping loop")
                #frames.put(None)
                #break
            pipe_in.send(image)


ef detectpersonPipe(pipe_out, pipe_in):
    # Setup ppl descriptor
    hog = cv2.HOGDescriptor()
    hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())

    altframe = pipe_out.recv()

    while altframe is not None:
        (regions, _) = hog.detectMultiScale(altframe, winStride=(4, 4), padding=(4, 4), scale=1.05)

        for (x, y, w, h) in regions:
            cv2.rectangle(altframe, (x, y), (x + w, y + h), (0, 0, 255), 2)
            cv2.putText(altframe, f'person', (x, y), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1)

        #print(f'{os.getpid()} putting frame in queue')
```



```python
import concurrent.futures
import multiprocessing
from multiprocessing.context import Process
import cv2
import imutils
import numpy as np
import time
import os


# Worker processes
```

```
1468: puttin frame into queue1463: showing final frame

1468: puttin frame into queue
1468: puttin frame into queue
1468: puttin frame into queue
1469: finished frame in queue
1468: puttin frame into queue
1463: showing final frame
1468: puttin frame into queue
1468: puttin frame into queue
1468: puttin frame into queue
1469: finished frame in queue
1463: showing final frame
1468: puttin frame into queue
1468: puttin frame into queue
1468: puttin frame into queue
1469: finished frame in queue
1463: showing final frame
1468: puttin frame into queue
1468: puttin frame into queue
1468: puttin frame into queue
1468: puttin frame into queue
1468: puttin frame into queue
1469: finished frame in queue
1463: showing final frame
1468: puttin frame into queue
1468: puttin frame into queue
1468: puttin frame into queue
1468: puttin frame into queue1469: finished frame in queue

1463: showing final frame
1468: puttin frame into queue
1468: puttin frame into queue
1468: puttin frame into queue
1469: finished frame in queue
1463: showing final frame
1468: puttin frame into queue
1468: puttin frame into queue
1468: puttin frame into queue
1469: finished frame in queue
1463: showing final frame
1468: puttin frame into queue
1468: puttin frame into queue
1468: puttin frame into queue
1468: puttin frame into queue
```



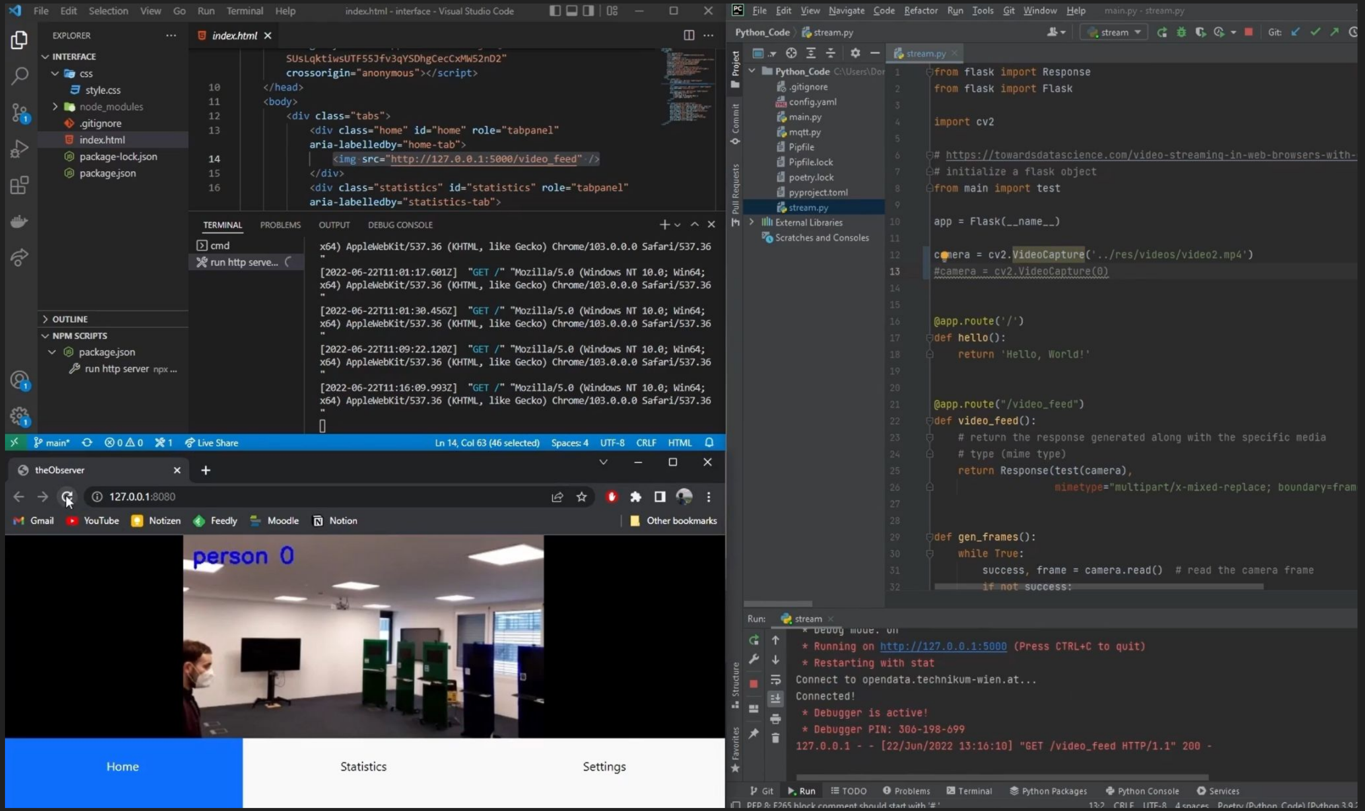
```
top - 16:38:51 up 17 min,  2 users,  load average: 3.21, 1.97, 1.47
Tasks: 171 total,   4 running, 165 sleeping,   0 stopped,   2 zombie
%Cpu0  :  75.3/17.2   93[||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||      ]
%Cpu1  :  90.9/6.0    97[|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||   ]
%Cpu2  :  84.8/11.1   96[||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||   ]
%Cpu3  :  69.2/18.2   87[|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||         ]
MiB Mem :   1820.6 total,    501.1 free,    934.4 used,    385.1 buff/cache
MiB Swap:    385.0 total,    374.7 free,     10.2 used.    760.9 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 4461 pi        20   0  231156  34188  10096 R 211.6   1.8   1:18.48 python3
 4460 pi        20   0  380564 180980  15840 R  67.9   9.7   0:33.32 python3
```

```python
import concurrent.futures
import multiprocessing
from multiprocessing.context import Process
```

```
4454: showing final frame
4460: puttin frame into queue
4460: puttin frame into queue
4460: puttin frame into queue
```

Interface

- get picture via https
- prepared for usage of statistics via influx & grafana environment



```html
        SUsLqktiwsUTF55Jfv3qYSDhgCecCxMW52nD2"
        crossorigin="anonymous"></script>
    </head>
    <body>
        <div class="tabs">
            <div class="home" id="home" role="tabpanel"
            aria-labelledby="home-tab">
                <img src="http://127.0.0.1:5000/video_feed" />
            </div>
            <div class="statistics" id="statistics" role="tabpanel"
            aria-labelledby="statistics-tab">
```

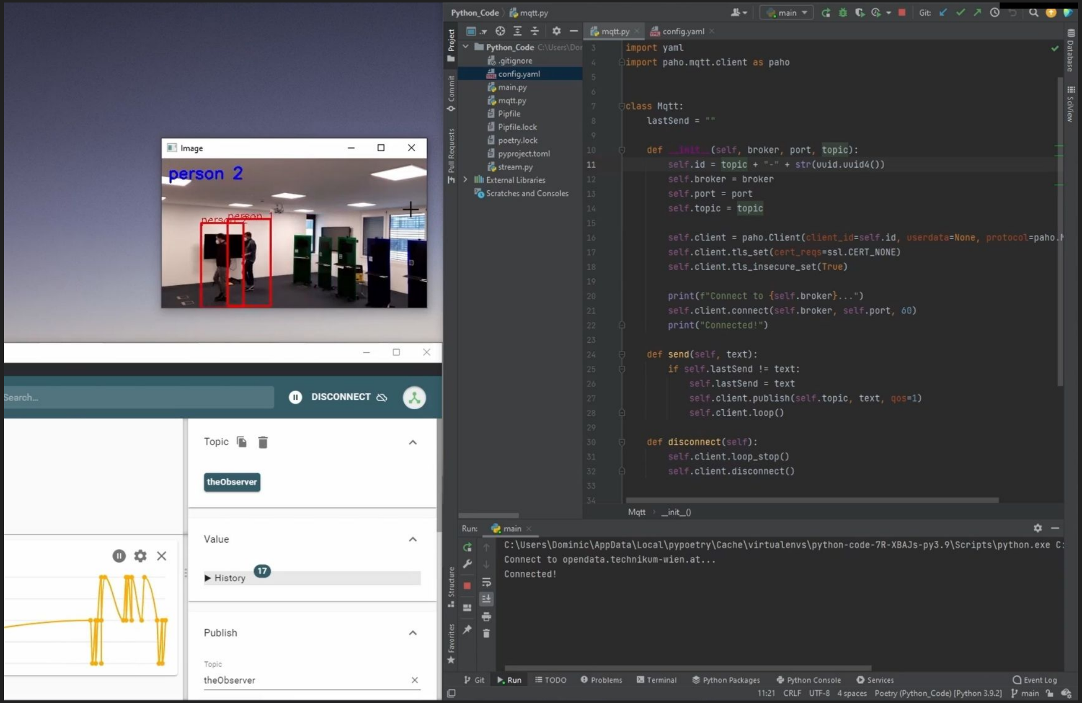
```
x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.0.0 Safari/537.36
"
[2022-06-22T11:01:17.601Z]  "GET /" "Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.0.0 Safari/537.36
"
[2022-06-22T11:01:30.456Z]  "GET /" "Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.0.0 Safari/537.36
"
[2022-06-22T11:09:22.120Z]  "GET /" "Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.0.0 Safari/537.36
"
[2022-06-22T11:16:09.993Z]  "GET /" "Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.0.0 Safari/537.36
"
```

```python
from flask import Response
from flask import Flask

import cv2

# https://towardsdatascience.com/video-streaming-in-web-browsers-with-
# initialize a flask object
from main import test

app = Flask(__name__)

camera = cv2.VideoCapture('../res/videos/video2.mp4')
#camera = cv2.VideoCapture(0)


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


@app.route("/video_feed")
def video_feed():
    # return the response generated along with the specific media
    # type (mime type)
    return Response(test(camera),
                    mimetype="multipart/x-mixed-replace; boundary=fram


def gen_frames():
    while True:
        success, frame = camera.read()  # read the camera frame
        if not success:
```

```
 * Debug mode: on
 * Running on http://127.0.0.1:5000 (Press CTRL+C to quit)
 * Restarting with stat
Connect to opendata.technikum-wien.at...
Connected!
 * Debugger is active!
 * Debugger PIN: 306-198-699
127.0.0.1 - - [22/Jun/2022 13:16:10] "GET /video_feed HTTP/1.1" 200 -
```

MQTT

- record number of visitors
- opendata.technikum-wien.at server used for the mqtt-broker
- detected number of visitors is sent to the mqtt server and ready to be used (statistics)

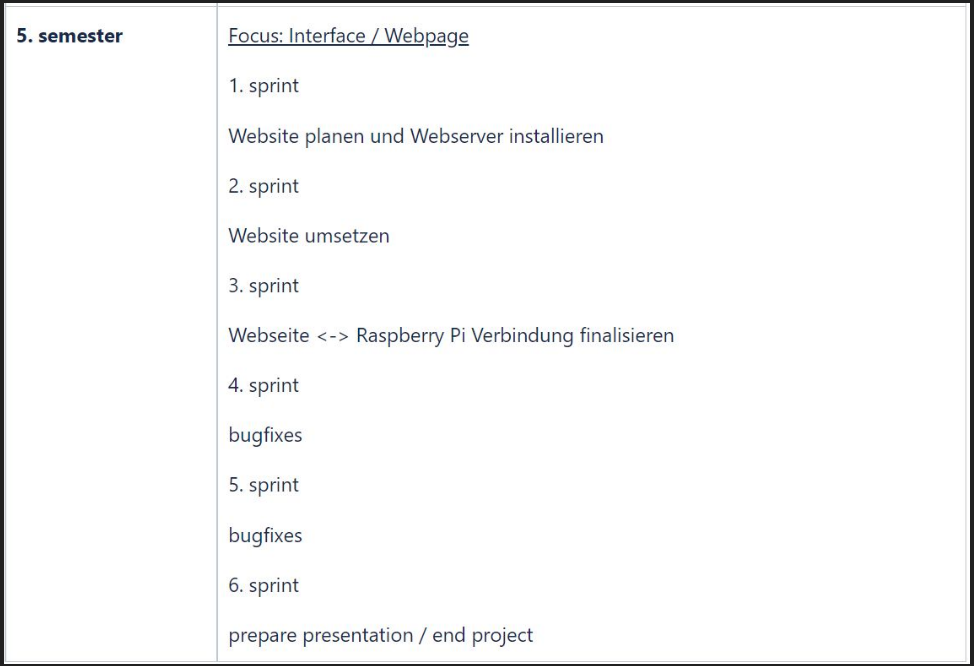


```python
import yaml
import paho.mqtt.client as paho


class Mqtt:
    lastSend = ""

    def __init__(self, broker, port, topic):
        self.id = topic + "-" + str(uuid.uuid4())
        self.broker = broker
        self.port = port
        self.topic = topic

        self.client = paho.Client(client_id=self.id, userdata=None, protocol=paho.
        self.client.tls_set(cert_reqs=ssl.CERT_NONE)
        self.client.tls_insecure_set(True)

        print(f"Connect to {self.broker}...")
        self.client.connect(self.broker, self.port, 60)
        print("Connected!")

    def send(self, text):
        if self.lastSend != text:
            self.lastSend = text
            self.client.publish(self.topic, text, qos=1)
            self.client.loop()

    def disconnect(self):
        self.client.loop_stop()
        self.client.disconnect()
```

```
C:\Users\Dominic\AppData\Local\pypoetry\Cache\virtualenvs\python-code-7R-XBAJs-py3.9\Scripts\python.exe C:
Connect to opendata.technikum-wien.at...
Connected!
```

Open steps

- bug fixes for person detection and multiprocessing
- finalization interface
- implementation webpage
- 3D print
- final connection



5. semester	Focus: Interface / Webpage 1. sprint Website planen und Webserver installieren 2. sprint Website umsetzen 3. sprint Webseite <-> Raspberry Pi Verbindung finalisieren 4. sprint bugfixes 5. sprint bugfixes 6. sprint prepare presentation / end project

End of current phase

- 2 of 3 semesters done
- start of upcoming phase #3 in late september 2022
- same team, final product