

SKYNET SECURITY SYSTEM



By Andrew Koval

Introduction

My goal for this project was to create an affordable solution to home security with on-premise machine learning to detect people passing our home

This project enabled me to learn many new technologies which I previously thought were beyond the scope of my skillset as a second-year student.

I made my proposal extremely ambitious knowing I will not achieve every feature in the list. I wanted to see how far I could push myself







INITIAL VISION

As my proposal was very ambitious, I did not know where to begin with the project so I asked chatGPT to give me some ideas

A

I want to write a program for a linux machine which processes a live video feed from two ESP camera modules and converts that to very low framerate (maybe 1 fps) video and stores one video of the two camera feeds stitched together every 2 hours to a hard drive on the machine. What is the best approach? what language is best suited for this?



To accomplish this task, you can use Python programming language and the OpenCV library   which provides a set of tools for computer vision and image processing. Here's an approach you can take:

1. Set up the two ESP camera modules to stream video over a local network to the Linux machine. You can use the OpenCV VideoCapture function to capture the video streams.
2. Use OpenCV's functions to process the video frames, stitch them together, and create a new video with a low frame rate. You can use OpenCV's VideoWriter function to write the processed video to the hard drive.
3. Use a timer function in Python to trigger the video capture and processing every 2 hours. You can use the time.sleep function to pause the program for the duration of the 2-hour interval.
4. Set up the program to run in the background as a daemon process, so it runs continuously and automatically even if the machine is restarted.

Python is a good choice for this task because it is easy to read and write, has a large number of libraries available, and has good support for asynchronous programming. Additionally, OpenCV is a powerful library for image and video processing and has a Python API.



Sure, here are some ideas on how you could implement a DIY home security system using low-cost ESP-enabled camera modules and various sensors, and how you could leverage edge computing capabilities from a laptop with 16GB DDR3 RAM and a 4th gen i7 CPU:



1. **Use ESP-enabled camera modules:** One way to use low-cost ESP-enabled camera modules is to set up a live video stream of your home's entrance, porch, and other areas you'd like to monitor. You can then use motion detection to alert you when someone enters your property. To set this up, you'll need to install the ESP-enabled camera modules and configure them to connect to your home Wi-Fi network. You can then use software like OpenCV or TensorFlow to detect motion in the video stream.
2. **Install various sensors:** In addition to the ESP-enabled camera modules, you can also install various sensors such as door and window sensors, temperature and humidity sensors, and motion detectors. These sensors can be connected to a central hub (like a Raspberry Pi or an ESP8266 board) which can then be used to trigger alerts when certain events occur. For example, if a door or window is opened when you're not home, the hub can trigger an alert on your phone or send an email.
3. **Set up edge computing:** To enhance the performance of your home security system, you can use edge computing to process data locally on your laptop instead of relying on cloud-based services. Your laptop with a 16GB DDR3 RAM and a 4th gen i7 CPU would be powerful enough to handle the edge computing workload. You can install software like TensorFlow, PyTorch or OpenCV to run machine learning algorithms and computer vision tasks on the camera feeds and sensor data. For example, you could use object detection algorithms to identify and track people in the video feed or analyze the temperature and humidity data to detect anomalies that may indicate a fire or a gas leak.



4. **Use Home Automation Platforms:** Home automation platforms like Home Assistant or Node-RED can be used to integrate all the sensors and cameras into a single interface, allowing you to monitor and control them from a single dashboard. These platforms can also be used to automate tasks based on events detected by the sensors or cameras. For example, you could set up an automation to turn on the lights when motion is detected in a certain area.
5. **Implement Remote Access:** Lastly, you can implement remote access to your home security system by setting up a VPN connection to your home network. This will allow you to securely access the live video feeds and sensor data from anywhere in the world. You can also use this connection to control your home automation devices remotely.

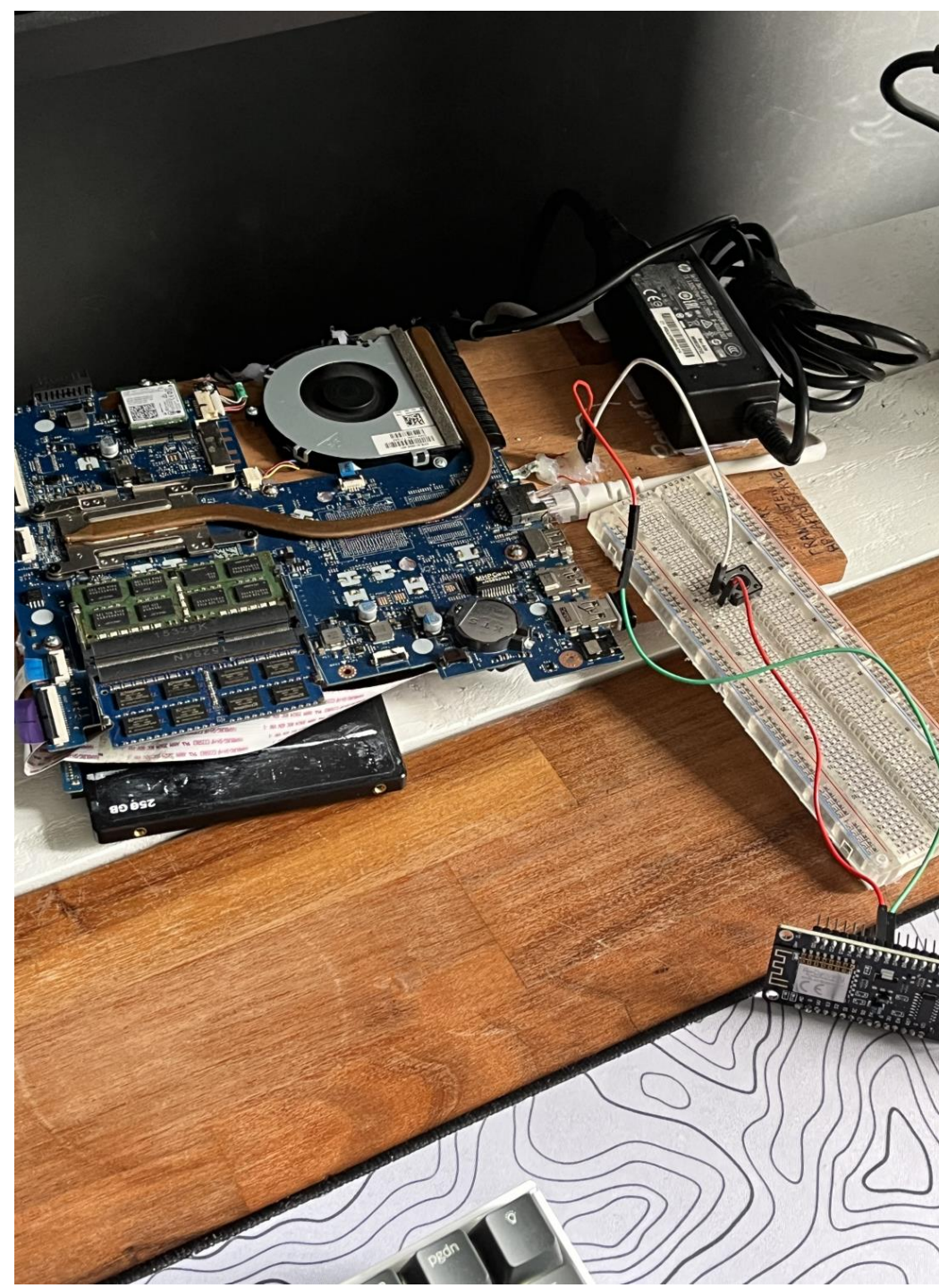
Keep in mind that DIY home security systems may not be as reliable or secure as commercial systems, so it's important to take extra precautions to ensure the security and reliability of your system.

The Server

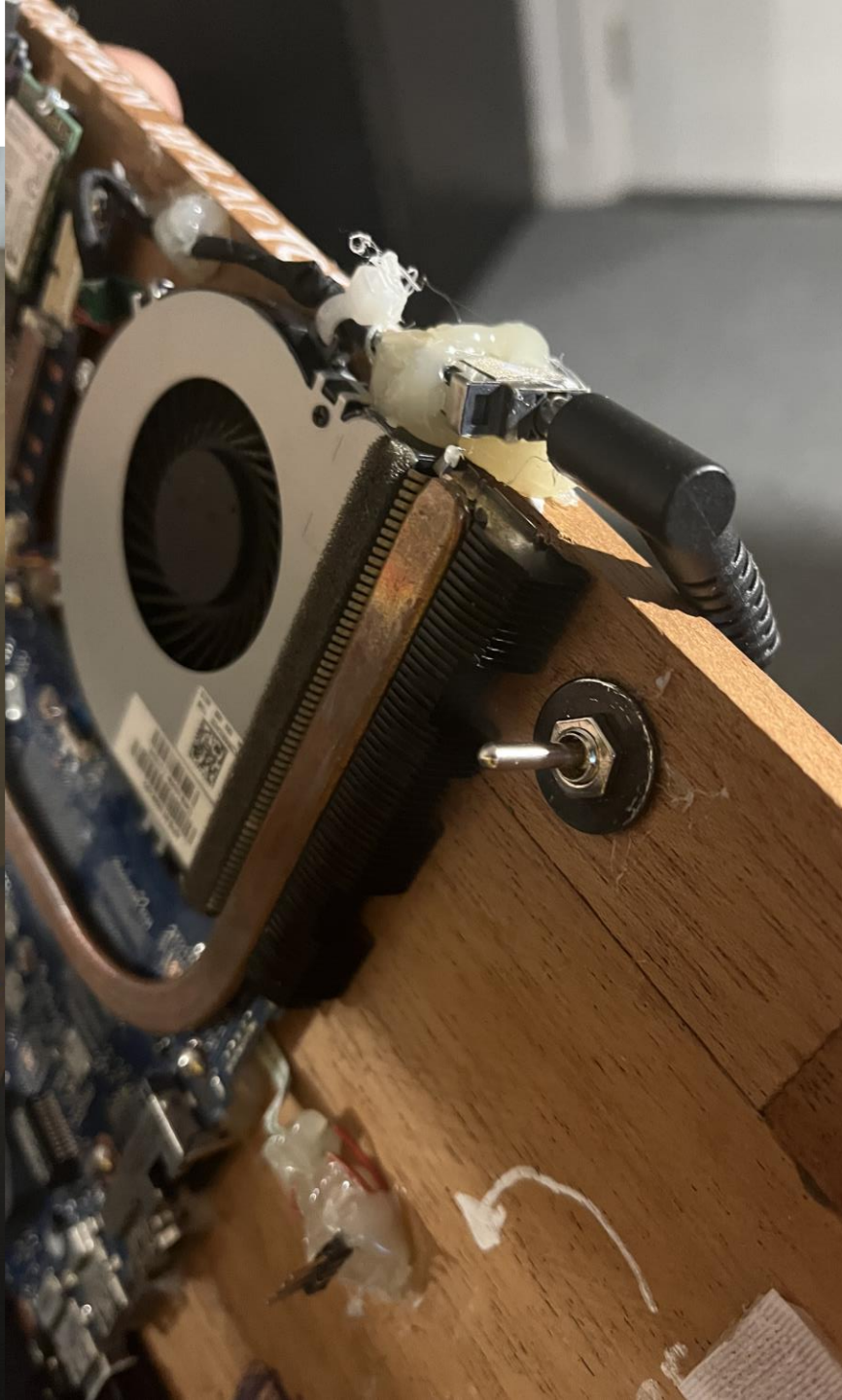
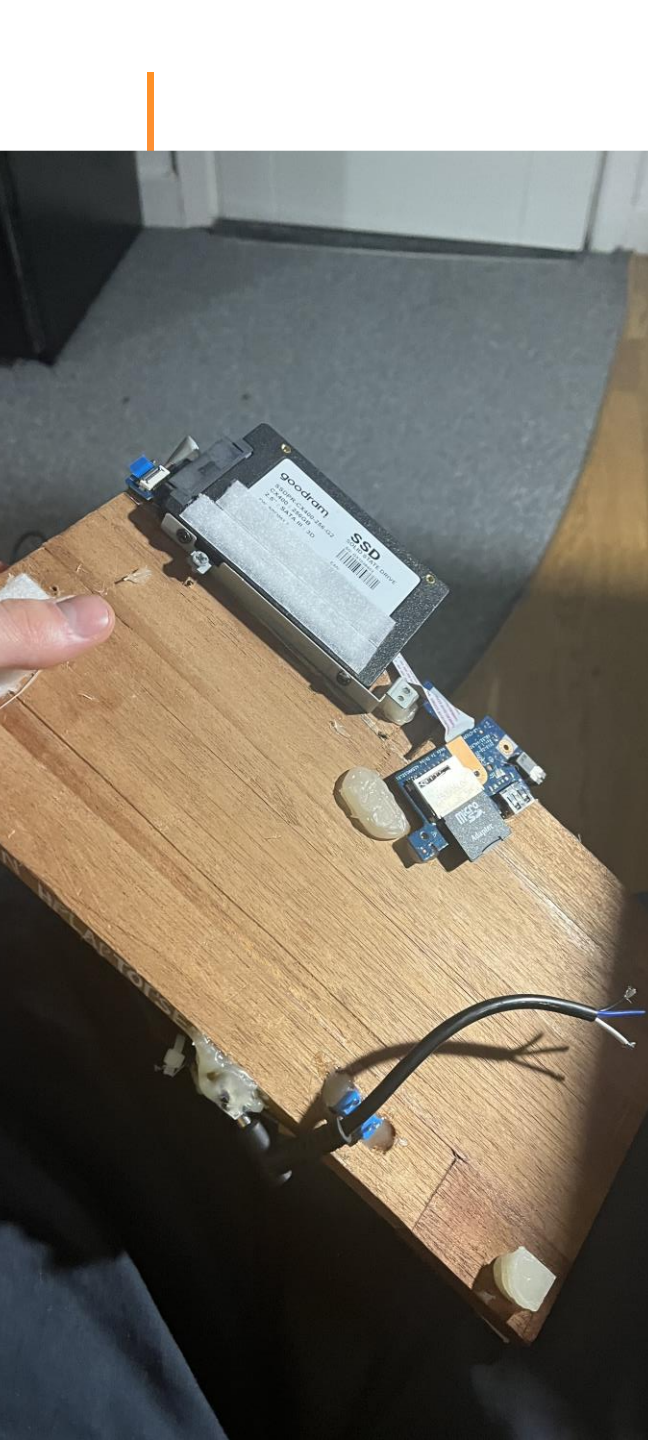
I upgraded my laptop roughly one year ago so my old machine was sitting in a drawer collecting dust. The old machine is a HP office laptop released in 2015 with the following specs (stock):

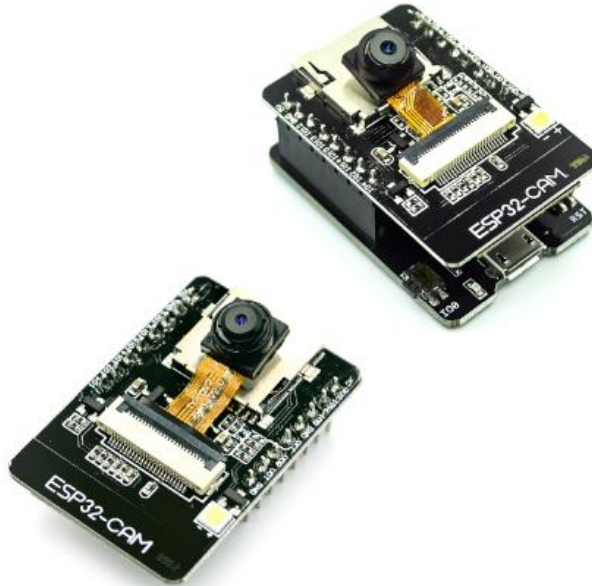
- Intel i7 4510U
- 8gb ddr3
- 1tb Samsung internal hard drive

Throughout the past few years I gathered many old computer parts. By disassembling old laptops I had a spare 8gb stick of ddr3 memory and a 256gb sata3 ssd. This allowed me to upgrade the laptop by swapping the hard drive with an SSD and add the extra RAM. Now that the machine had 16gb of RAM, it was much more capable as a web server. In the beginning I planned to use the Raspberry pi 4 to process and stream live video however this machine proved to be much more capable.



- This upgrade in hardware gave me the idea to run something more demanding alongside the webserver which sparked the idea of running a pretrained Tensorflow model to analyze the video to detect people outside the house.





ESP32-CAM WiFi Module ESP32 serial to WiFi ESP32 CAM Development Board 5V For Bluetooth with OV2640 Camera Module Nodemcu

★★★★★ 4.8 ▾ 196 Reviews

€ 5,49 ~~€ 6,87~~ 20% off

Price includes VAT

Color: ESP32-CAM downloader



Quantity:



Additional 1% off (10 Pieces or more)
118080 Pieces available

Ships to Ireland

Shipping: €1.67

Estimated delivery on Jun 22

Fast Delivery on orders over €10.00 Estimated delivery on Jun 17 to Ireland

Buy Now

Add to Cart

♥ 1794

90-Day Buyer Protection
Money back guarantee

More price information ⓘ

Recommended For You



€ 4,57



€ 3,13



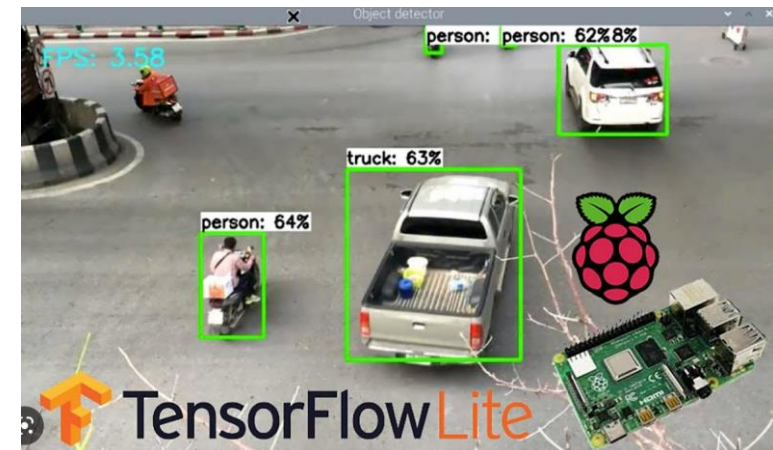
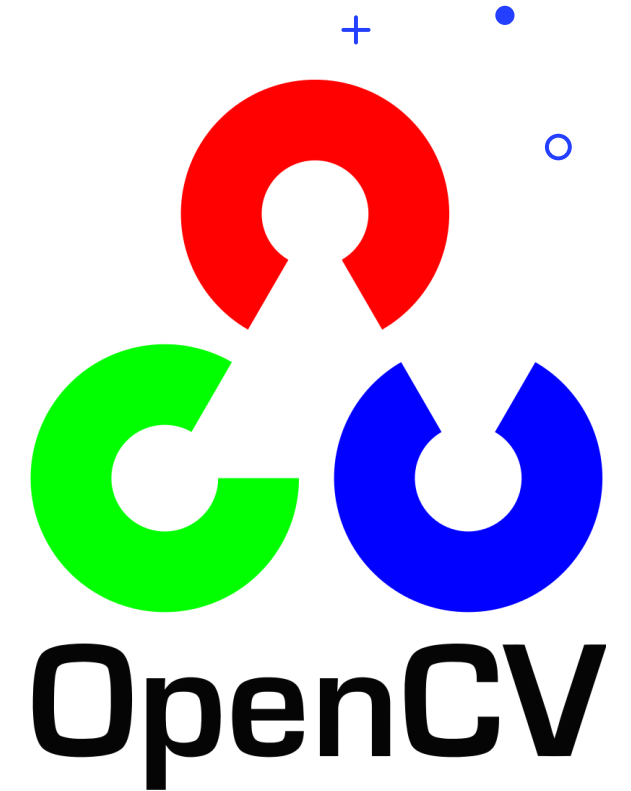
€ 3,77

More options ▾

I was genuinely blown away by how cheap these were

Python

- The Python programming language was an integral part of this project. Using python was like a breath of fresh air when compared to stricter languages like Java.
- OpenCV was a major reason why I decided to use python.
- ChatGPT suggested the idea of using OpenCV in order to take live video streams and display them on a website interface



Creating the website

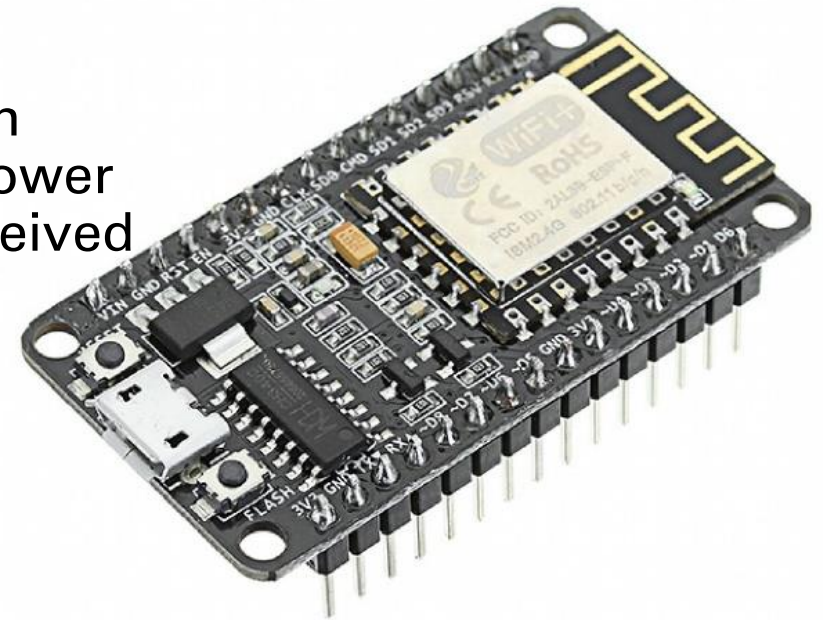
- As I had some experience with website design from coder dojo and previous modules, I was quickly able to create responsive website utilizing the bootstrap framework.
- I used a Python library called flask to run the web server. I decided that it would be best to stick to python for both video processing and webserver integration

I found a very straightforward Flask tutorial on freecodecamp.org/news/learn-flask-for-python-full-tutorial/



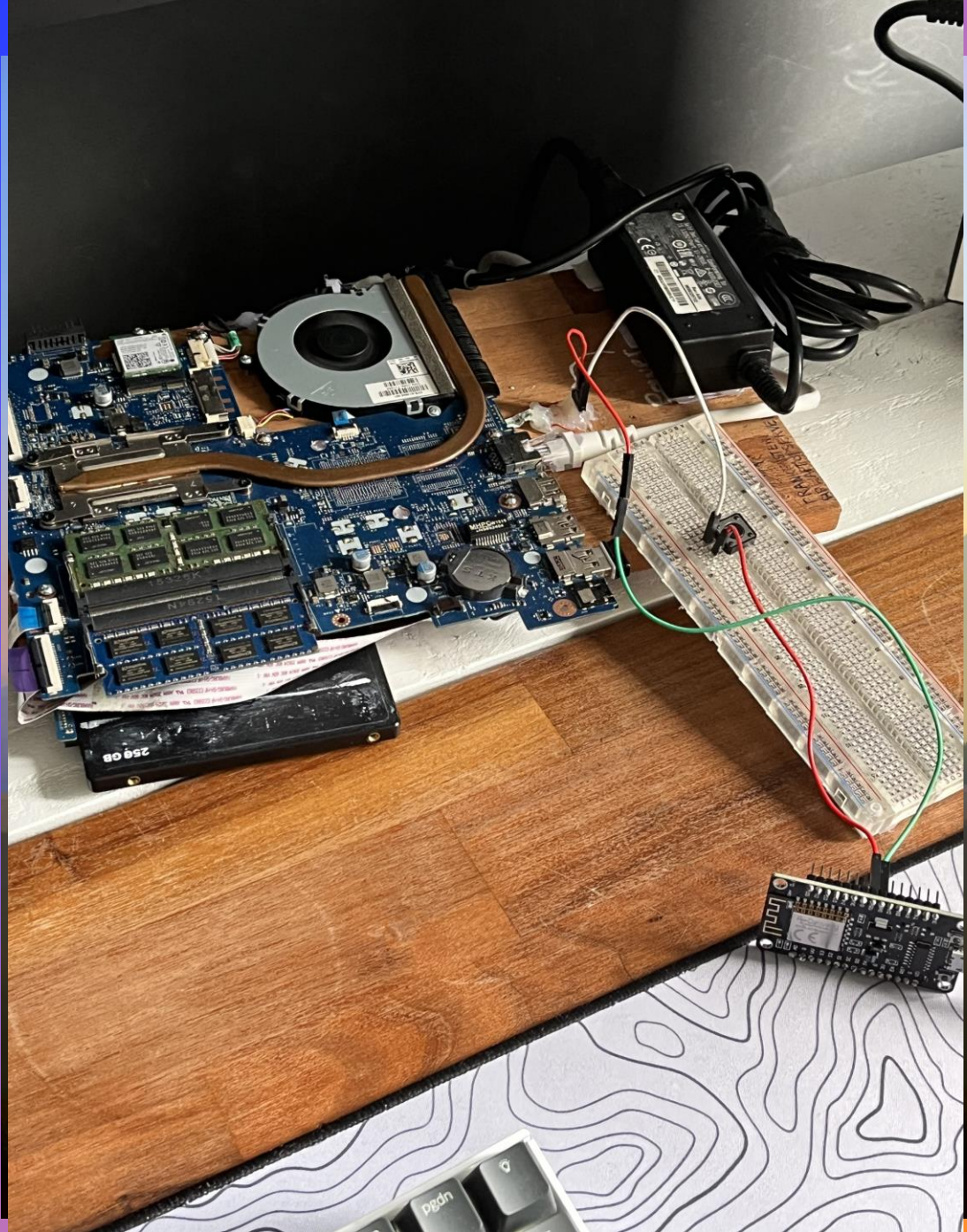
MQTT

- In the Beginning of my project I had to access the web UI of the router or run an Nmap scan in order to find the local IP address of my server so I could SSH into the machine.
- This proved to be quite cumbersome so I got the idea to write a startup script which finds the machines local IP address and sends it to HiveMQ where I can find it from my laptop.
- This was to be paired with a remote power button/switch using an ESP8266 Module which would short the 3.3v power pin to ground –acting as a power button whenever it received the MQTT message: {data: "power"}



skynet > \$ startup.sh

```
1  #!/bin/bash
2
3  # Set your MQTT broker and credentials
4  # BROKER="mqtt://mqtt.beebotte.com:1883"
5  BROKER="broker.hivemq.com"
6  ./
7  API_KEY="hJTMDw8xHLDkduPDtxTKqHgA"
8  SECRET_KEY="ifFpnRkcglWt9HReE5y6fH6PUBuODnZl"
9  TOPIC="/test/res"
10
11 # Wait for a network connection to be established
12 echo "Waiting for network connection..."
13 while ! ping -c1 google.com &>/dev/null; do sleep 1; done
14
15 # Get the IP address of the default network interface
16 IP=$(ip addr show enp7s0 | grep "inet " | awk '{print $2}' | cut -d '/'
17
18 # Convert the payload message to a JSON string
19 PAYLOAD=$(echo '{}' | jq --arg ip "$IP" '.ip=$ip')
20
21 # Publish the payload message as a JSON string to the MQTT broker 5 ti
22 for i in {1..5}; do
23     echo "Publishing payload message $PAYLOAD to topic $TOPIC (attempt $
24     mosquitto_pub -h $BROKER -p '1883' -t $TOPIC -m "$PAYLOAD"
25     sleep 2
26 done
27
```

AI Object Detection

YoloV4 Model

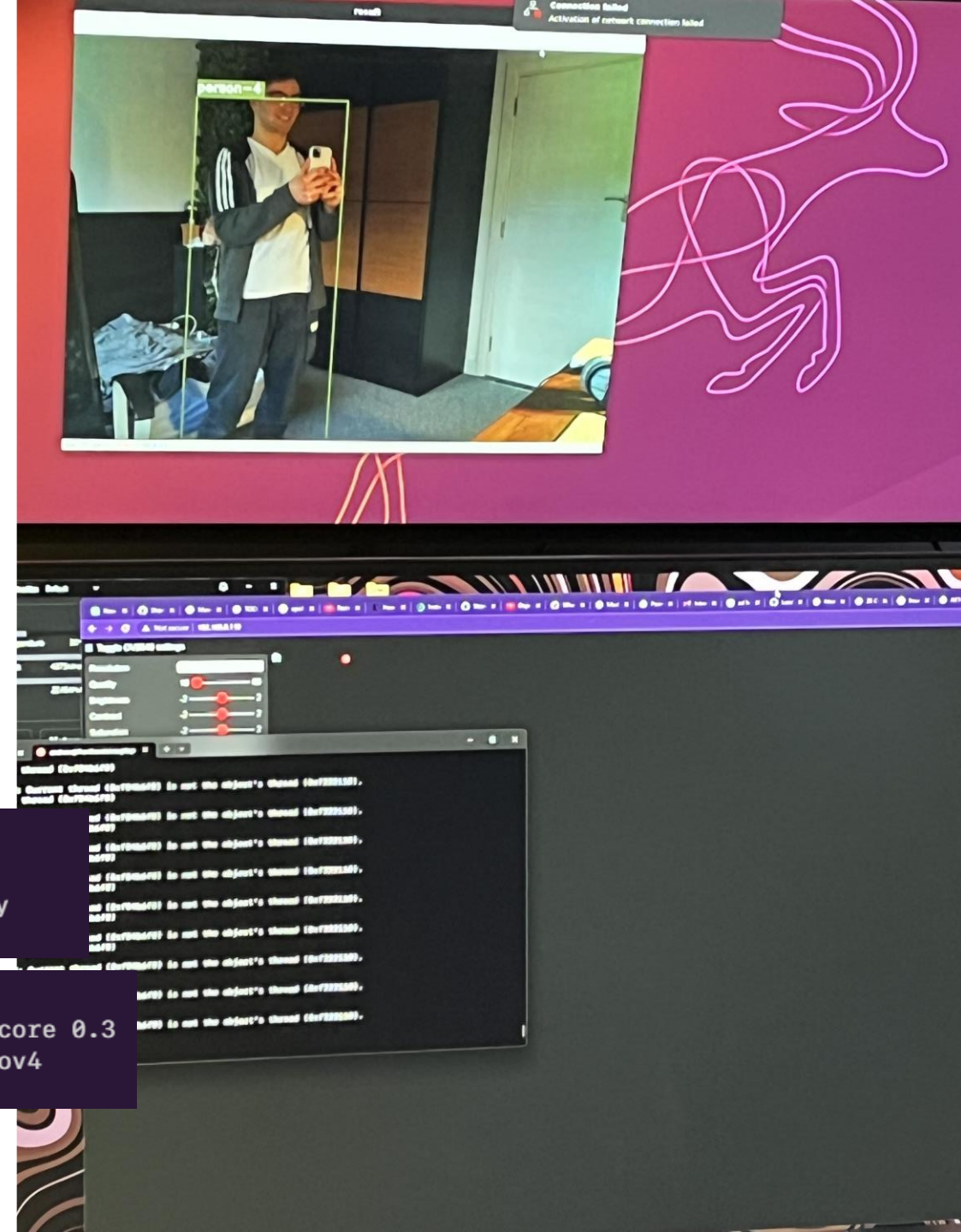
- This was quite a challenging aspect as there was a steep learning curve with Tensorflow.
- I was about to give up on the idea until I found a github repo with a pretrained model called yoloV4.

Using Pip

```
# CPU  
pip install -r requirements.txt
```

```
## yoloV4-tiny  
python save_model.py --weights ./data/yolov4-tiny.weights --output  
./checkpoints/yolov4-tiny-416 --input_size 416 --model yolov4 --tiny
```

```
# Run Tracking on Video With Tiny YoloV4  
python track_objects.py --weights ./checkpoints/yolov4-tiny-416 --score 0.3  
--video ./data/dog.mp4 --output ./results/demo_tiny.avi --model yolov4
```



Twingate VPN

The image shows the Twingate website landing page. The header includes the Twingate logo and navigation links: Product, Docs, Resources, Customers, Pricing. A 'Launch Twingate' button is in the top right. The main headline reads 'It's time to ditch your VPN'. Below it, a sub-headline states 'Twingate makes Zero Trust Network Access easy to deploy, even easier to use, and always secure.' Two buttons are present: 'Try Twingate for Free' and 'Request a Demo'. At the bottom, a screenshot of a macOS desktop is shown. It features a terminal window with the following text: 'alex - alex@alex-mbp', 'alex@alex-mbp - % ssh root@k8s.prod.autoco.int', '> ssh root@k8s.prod.autoco.int', 'ssh: connection refused', 'alex@alex-mbp - % ssh root@k8s.prod.autoco.int', 'Welcome to Kubernetes v1.21.1!', 'Type \'help\' for a list of commands.', and 'root@prod-cluster:~\$'. To the right of the terminal is a sidebar with a search bar and a 'FAVORITES' section listing 'STG Cluster' (Connected), 'Prod Cluster', 'Elastic', 'Gitlab', and 'Jira'. Below this is an 'Other Resources' section and a 'Sign Out' link. The macOS menu bar at the top of the screenshot shows 'Finder', 'Edit', 'View', 'Go', 'Window', 'Help', and the system clock 'Fri 2:55 PM'.

Twingate Product Docs Resources Customers Pricing Launch Twingate

It's time to ditch your VPN

Twingate makes Zero Trust Network Access easy to deploy, even easier to use, and always secure.

[Try Twingate for Free](#) [Request a Demo](#)

alex - alex@alex-mbp

```
alex@alex-mbp - % ssh root@k8s.prod.autoco.int
> ssh root@k8s.prod.autoco.int
ssh: connection refused
alex@alex-mbp - % ssh root@k8s.prod.autoco.int
Welcome to Kubernetes v1.21.1!

Type 'help' for a list of commands.

root@prod-cluster:~$
```

FAVORITES

- STG Cluster Connected
- Prod Cluster
- Elastic
- Gitlab
- Jira

Other Resources

Sign Out

Deploy Twingate for Free

14-day trial of Twingate Business, no credit card required. Twingate Starter is free forever.



Sign up with Google



Sign up with Microsoft



Sign up with GitHub



Sign up with LinkedIn

Already using Twingate?
[Sign in to an existing Network →](#)

Welcome Andrew

Tell us a bit about yourself to finish setting up your account.

Work Email

Required

hame@company.com

Company Name

Required

Acme

Number of Employees

Required

Select one



[Download](#) [Setup Guide](#) [Support](#)

[Network](#) [Team](#) [Devices](#) [Policies](#) [Settings](#)



AndrewKoval

andrewkoval.twingate.com

Resources

[View as list](#)



Add Resource



HpLaptopServer
192.168.1.12 • RaspberryPiCam

Remote Networks

[Add](#)



RaspberryPiCam
1 Resource

IP

HpLaptopServer

192.168.1.12

Manage

Activity Access 1

Updated just now

- 20098730@mail.wit.ie connected to 192.168.1.12

20098730@mail.wit.ie failed to connect to 192.168.1.12

Show details

20098730@mail.wit.ie failed to connect to 192.168.1.12

20098730@mail.wit.ie failed to connect to 192.168.1.12

20098730@mail.wit.ie failed to connect to 192.168.1.12

20098730@mail.wit.ie failed to connect to 192.168.1.12

20098730@mail.wit.ie failed to connect to 192.168.1.12

20098730@mail.wit.ie failed to connect to 192.168.1.12

20098730@mail.wit.ie failed to connect to 192.168.1.12

20098730@mail.wit.ie failed to connect to 192.168.1.12

20098730@mail.wit.ie failed to connect to 192.168.1.12

20098730@mail.wit.ie failed to connect to 192.168.1.12

All Activity

May 16, 2023 - 12:19 PM

50 minutes ago

50 minutes ago

51 minutes ago

51 minutes ago

51 minutes ago

56 minutes ago

56 minutes ago

56 minutes ago

56 minutes ago

56 minutes ago

56 minutes ago

56 minutes ago

Remote Network

RaspberryPiCam

1 Resource

Protocols

Edit

- TCP: Allowed
- UDP: Allowed
- ICMP: Allowed

Edit Resource

Remote Network

RaspberryPiCam

DNS

www.gitlab.com

CIDR

10.0.0.1/32

Label

HpLaptopServer

CIDR Address

192.168.1.12

Alias



Protocol Restrictions

All Traffic Allowed

TCP Traffic

Allow

Restrict

Block

UDP Traffic

Allow

Restrict

Block

ICMP Traffic

Allow

Block



Client Visibility

Visible • No Browser Shortcut

Confirm Changes

Pain and Frustration

CameraWebServer | Arduino IDE 2.1.0

File Edit Sketch Tools Help

ESP32 Wrover Module

BOARDS MANAGER

Filter your search...

Type: All

Arduino AVR Boards by Arduino

1.8.6 installed

Boards included in this package: Arduino Yun, Arduino BT, LilyPad Arduino, Arduino Uno,...

More info

1.8.6 REMOVE

Arduino Mbed OS Edge Boards by Arduino

Boards included in this package: Arduino Edge Control

More info

4.0.2 INSTALL

Arduino Mbed OS Giga Boards by Arduino

Boards included in this package: Arduino Giga

More info

4.0.2 INSTALL

Arduino Mbed OS Nano Boards by...

Boards included in this package: Arduino Nano 33 BLE, Arduino Nano 33 BLE Sense, Arduino...

More info

4.0.2 INSTALL

Arduino Mbed OS Nicla Boards by Arduino

Boards included in this package: Nicla Sense ME, Nicla Voice, Nicla Vision

More info

4.0.2 INSTALL

Arduino Mbed OS Opta Boards by Arduino

CameraWebServer.ino

```
1 #include "esp_camera.h"
2 #include <Wifi.h>
3
4 //
5 // WARNING!!! PSRAM IC required for UXGA resolution and high JPEG quality
6 // Ensure ESP32 Wrover Module or other board with PSRAM is selected
7 // Partial images will be transmitted if image exceeds buffer size
8 //
9
10 // Select camera model
11 // #define CAMERA_MODEL_WROVER_KIT // Has PSRAM
12 // #define CAMERA_MODEL_ESP_EYE // Has PSRAM
13 // #define CAMERA_MODEL_M5STACK_PSRAM // Has PSRAM
14 // #define CAMERA_MODEL_M5STACK_V2_PSRAM // M5Camera version B Has PSRAM
15 // #define CAMERA_MODEL_M5STACK_WIDE // Has PSRAM
16 // #define CAMERA_MODEL_M5STACK_ESP32CAM // No PSRAM
17 #define CAMERA_MODEL_AI_THINKER // Has PSRAM
18 // #define CAMERA_MODEL_TTGO_T_JOURNAL // No PSRAM
19
20 #include "camera_pins.h"
21
22 const char* ssid = "R95 T4A3 (smart devices)";
23 const char* password = "SL667DJVRQVETE";
24
25 // const char* ssid = "iPhone";
26 // const char* password = "vJk3-F0IT-adoe-sG7o";
27
28 // const char* ssid = "LEGIONLAPTOP";
29 // const char* password = "20098730";
30
31 // const char* ssid = "IoT2";
32 // const char* password = "ilikecake";
33
34
```

Output

Sketch uses 2594110 bytes (82%) of program storage space. Maximum is 3145728 bytes.
Global variables use 56272 bytes (17%) of dynamic memory, leaving 271408 bytes for local variables. Maximum is 327680 bytes.
esptool.py v3.0-dev
Serial port COM6
Traceback (most recent call last):
File "esptool.py", line 3682, in <module>
File "esptool.py", line 3675, in _main
File "esptool.py", line 3329, in main
File "esptool.py", line 275, in __init__
File "site-packages\serial\serialutil.py", line 372, in write_timeout
File "site-packages\serial\serialwin32.py", line 222, in _reconfigure_port
serial.serialutil.SerialException: Cannot configure port, something went wrong. Original message: WindowsError(31, 'A device attached to the system is not functioning.')
Failed to execute script esptool
Failed uploading: uploading error: exit status 0xffffffff

Ln 1, Col 1 ESP32 Wrover Module on COM6



+

•

○