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 togerther 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.

capabilities from the hardware of a laptop with 16gb ddr3 ram and a 4th gen i7 cpu

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.

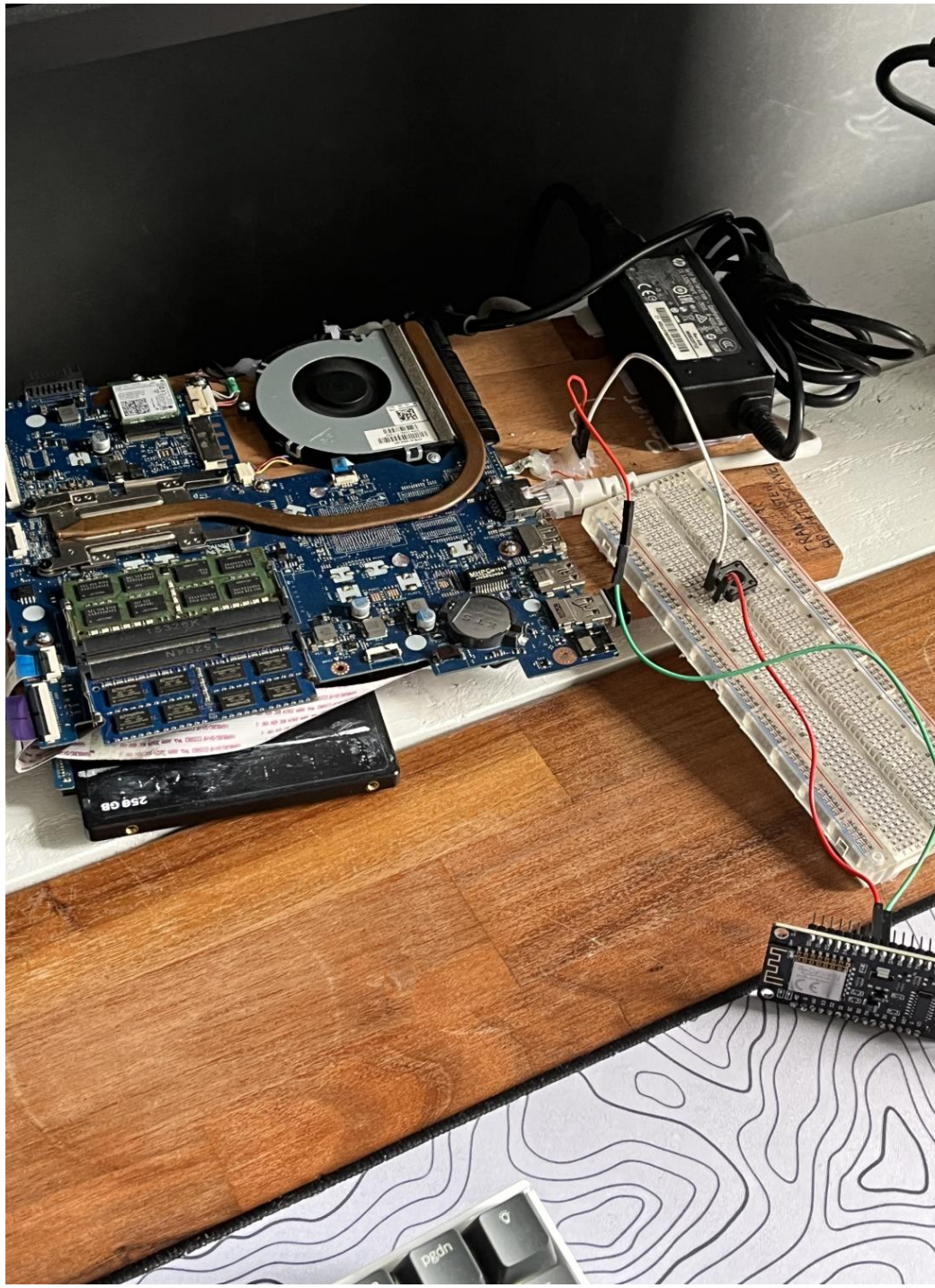
Keep in mind that DIY home security systems may not be as reliable or secure as commercial systems, so it's important to take extra precautions to ensure the security and reliability of your system.

The Server

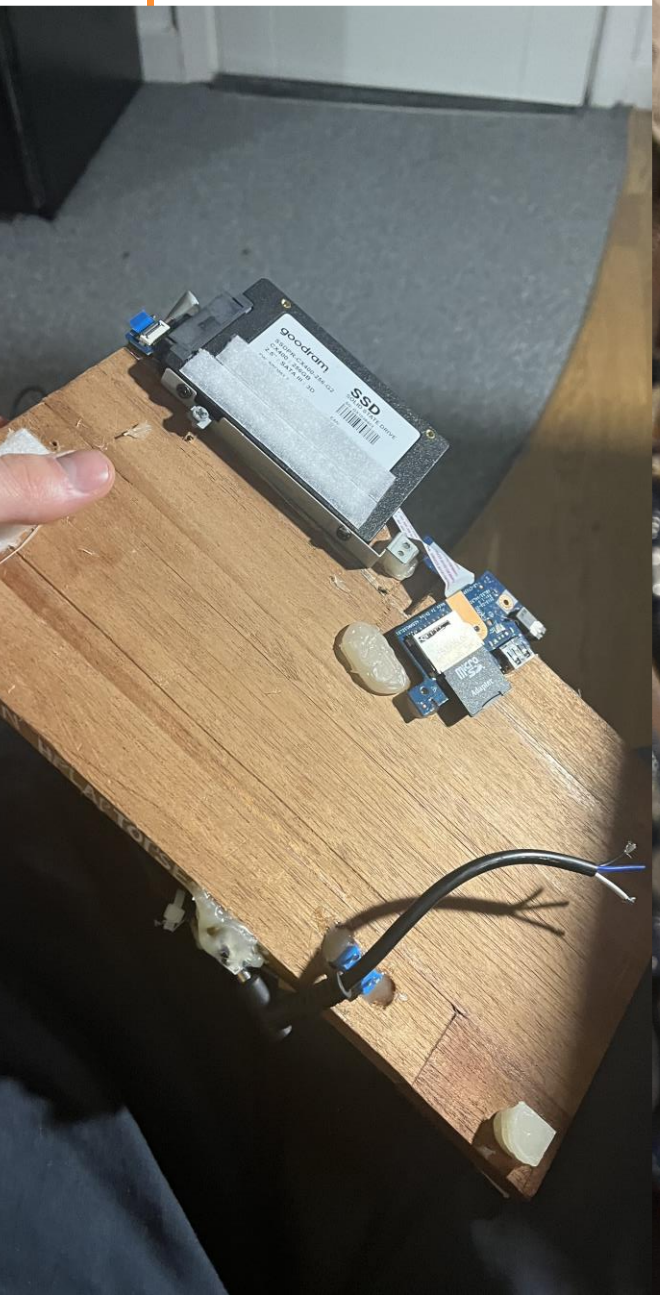
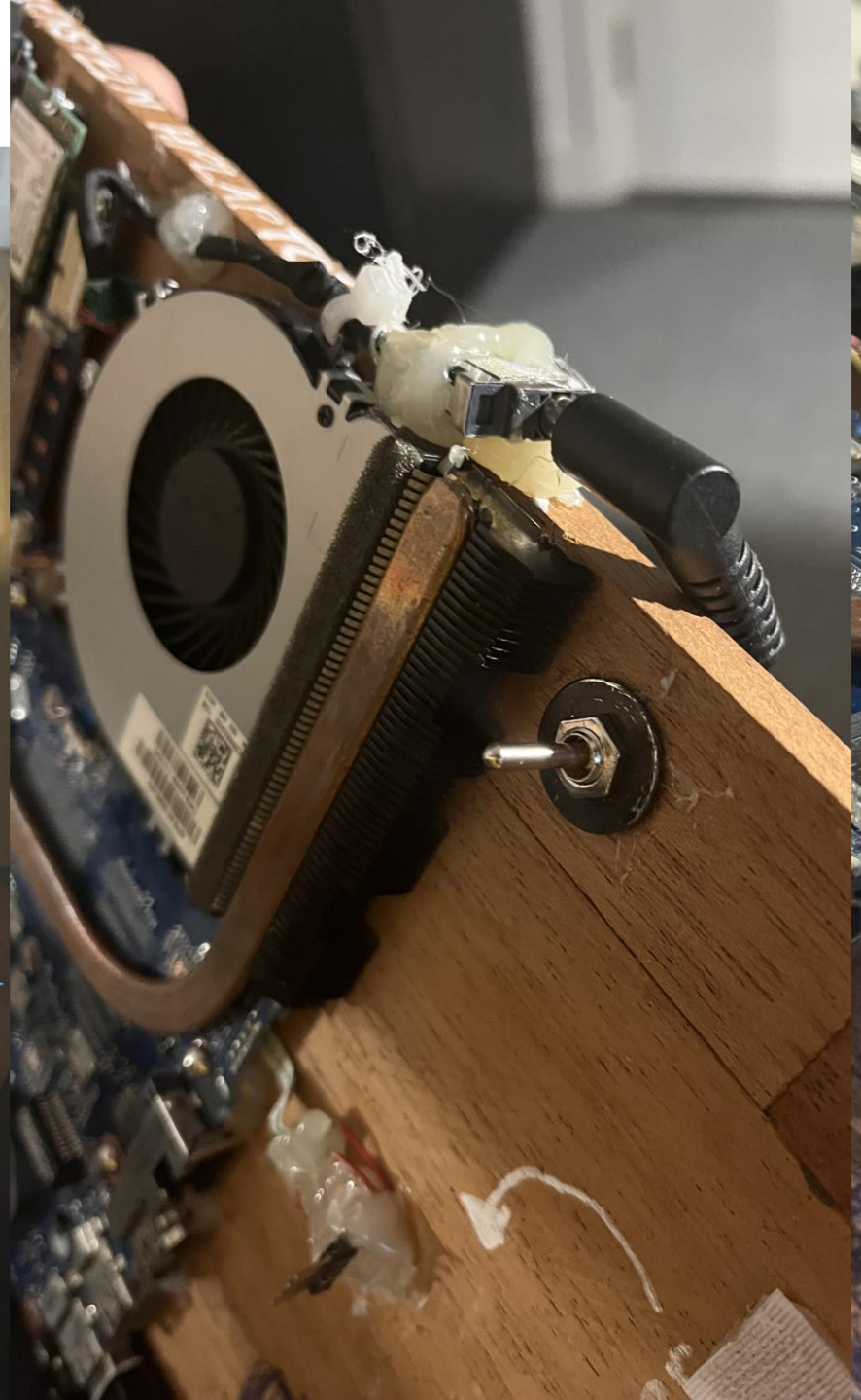
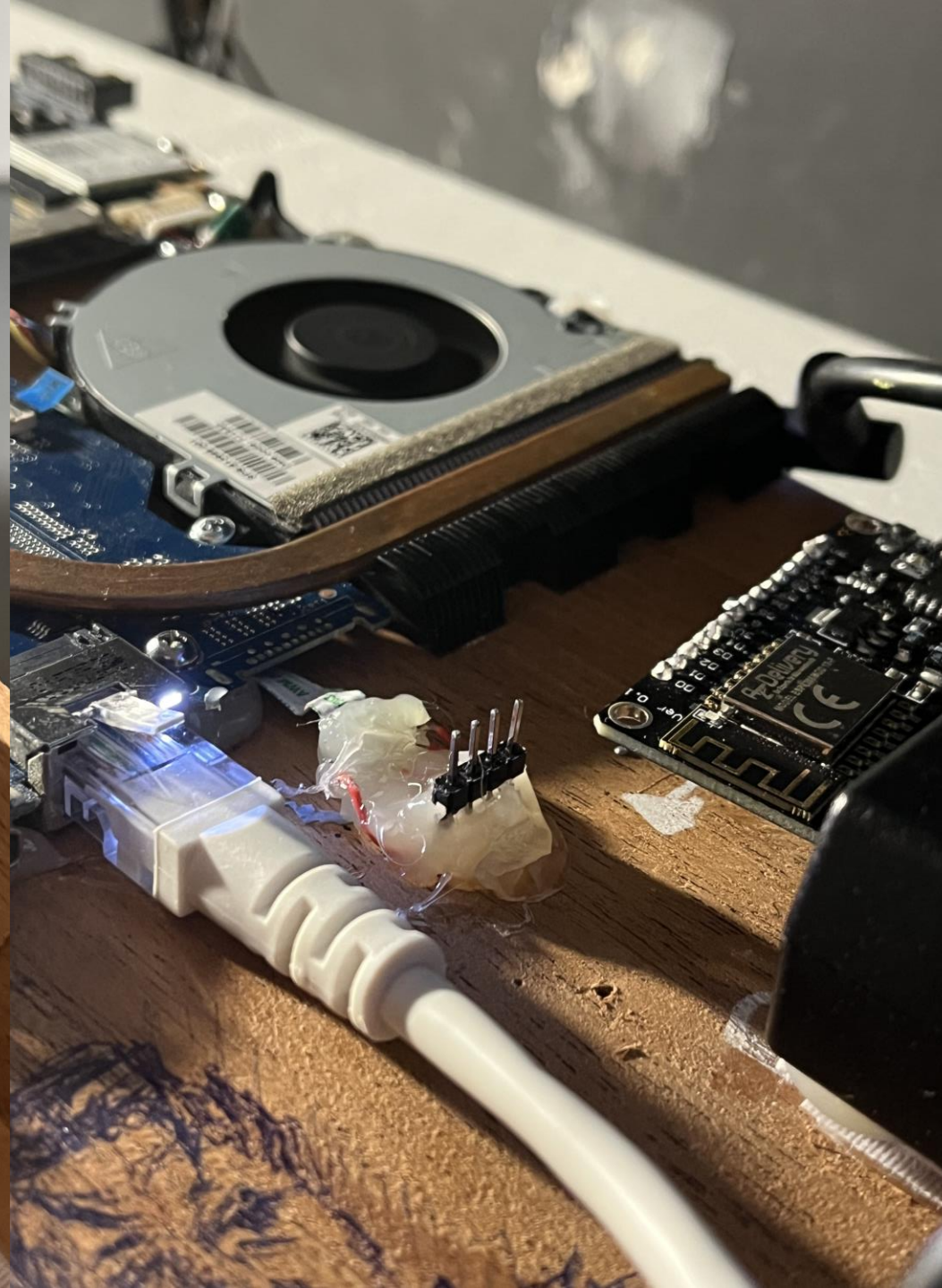
I upgraded my laptop roughly one year ago so my old machine was sitting in a drawer collecting dust. The old machine is a HP office laptop released in 2015 with the following specs (stock):

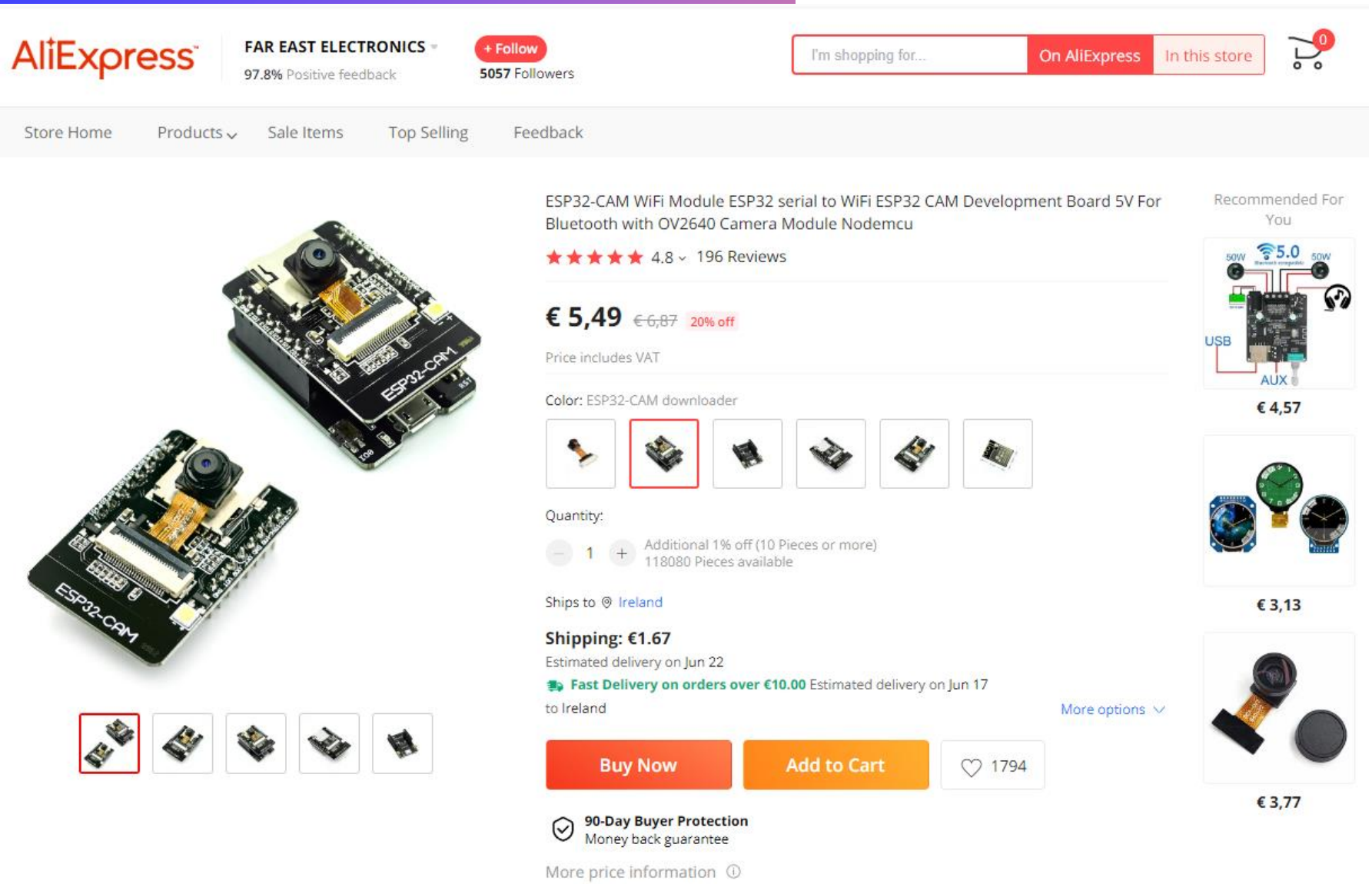
- Intel i7 4510U
- 8gb ddr3
- 1tb Samsung internal hard drive

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.

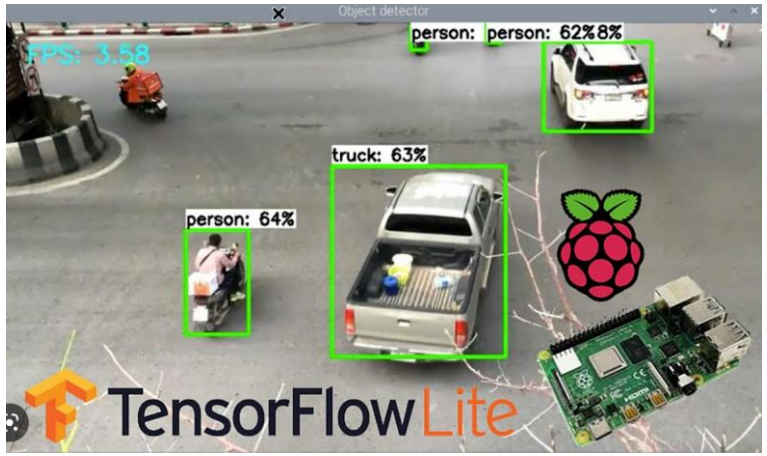

goodram
SSD





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

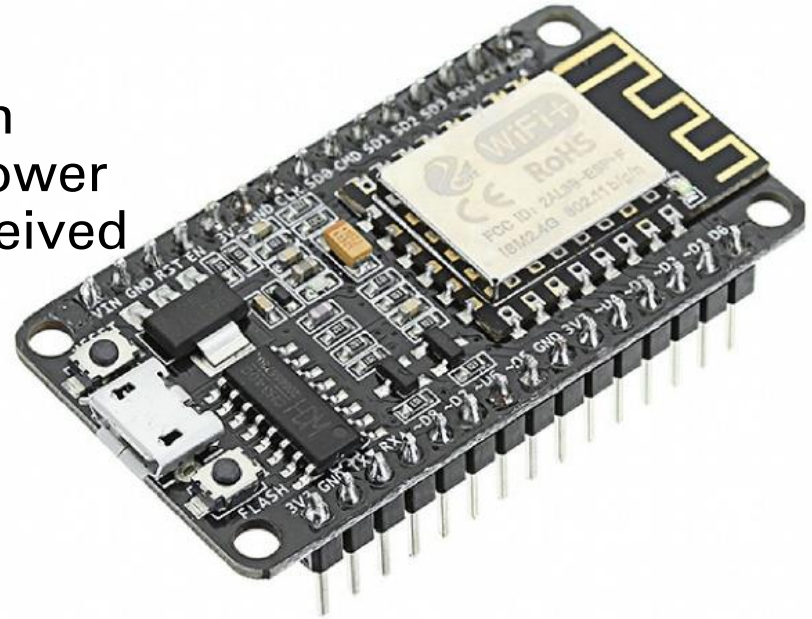
- As I had some experience with website design from coder dojo and previous modules, I was quickly able to create responsive website utilizing the bootstrap framework.
- I used a Python library called flask to run the web server. I decided that it would be best to stick to python for both video processing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

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

MQTT

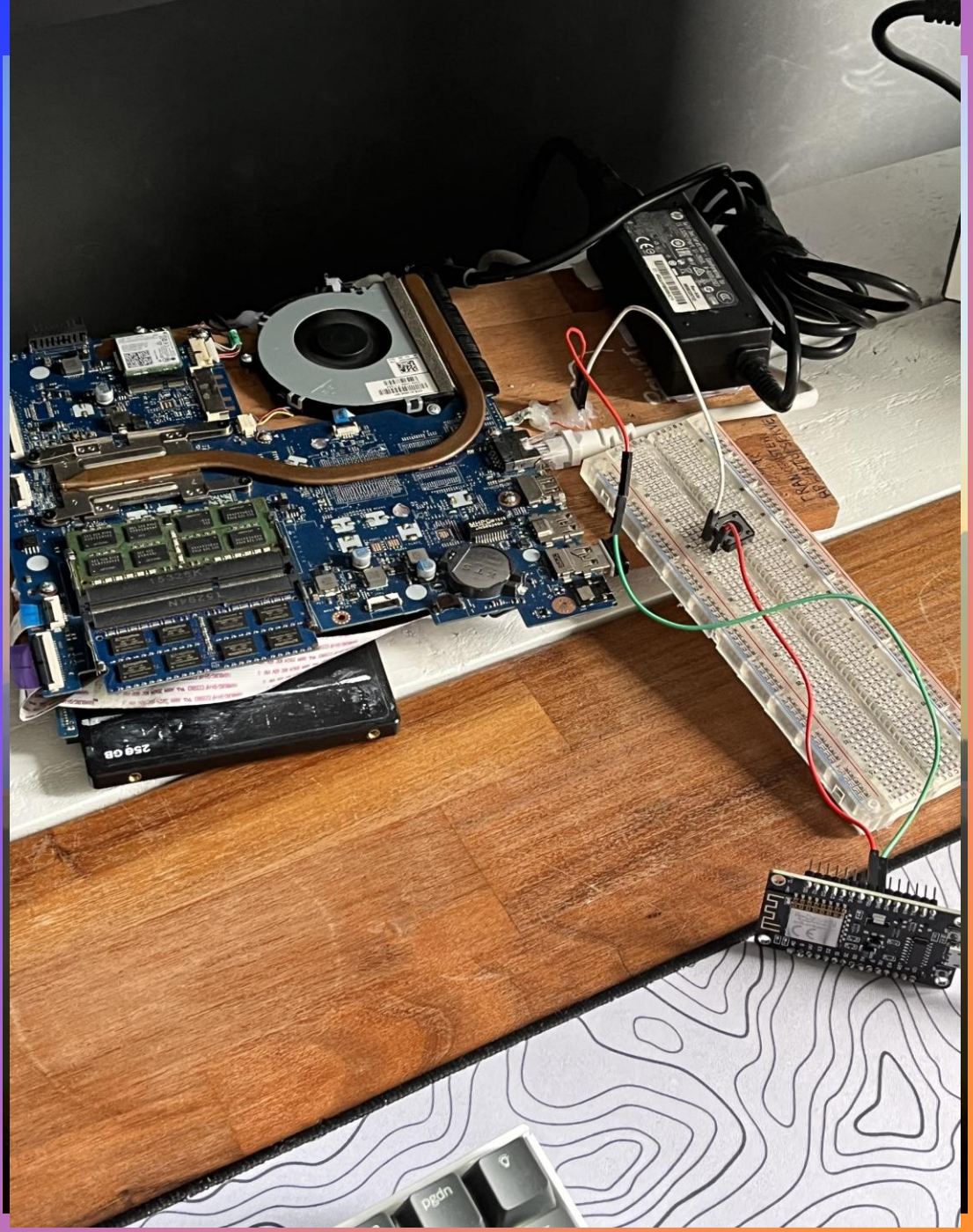
- In the Beginning of my project I had to access the web UI of the router or run an Nmap scan in order to find the local IP address of my server so I could SSH into the machine.
- This proved to be quite cumbersome so I got the idea to write a startup script which finds the machines local IP address 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”}

HIVEMQ



skynet > $ startup.sh

```
#!/bin/bash

# Set your MQTT broker and credentials
# BROKER="mqtt://mqtt.beebotte.com:1883"
BROKER="broker.hivemq.com"
./
API_KEY="hJTMDw8xHLDkduPDtxTKqHgA"
SECRET_KEY="ifFpnRkcglWt9HReE5y6fH6PUBuODnZl"
TOPIC="/test/res"

# Wait for a network connection to be established
echo "Waiting for network connection..."
while ! ping -c1 google.com &>/dev/null; do sleep 1; done

# Get the IP address of the default network interface
IP=$(ip addr show enp7s0 | grep "inet " | awk '{print $2}' | cut -d'/'

# Convert the payload message to a JSON string
PAYLOAD=$(echo '{}' | jq --arg ip "$IP" '.ip=$ip')

# Publish the payload message as a JSON string to the MQTT broker 5 ti
for i in {1..5}; do
  echo "Publishing payload message $PAYLOAD to topic $TOPIC (attempt $
  mosquitto_pub -h $BROKER -p '1883' -t $TOPIC -m "$PAYLOAD"
  sleep 2
done

```


256 GB

AI Object Detection

YoloV4 Model

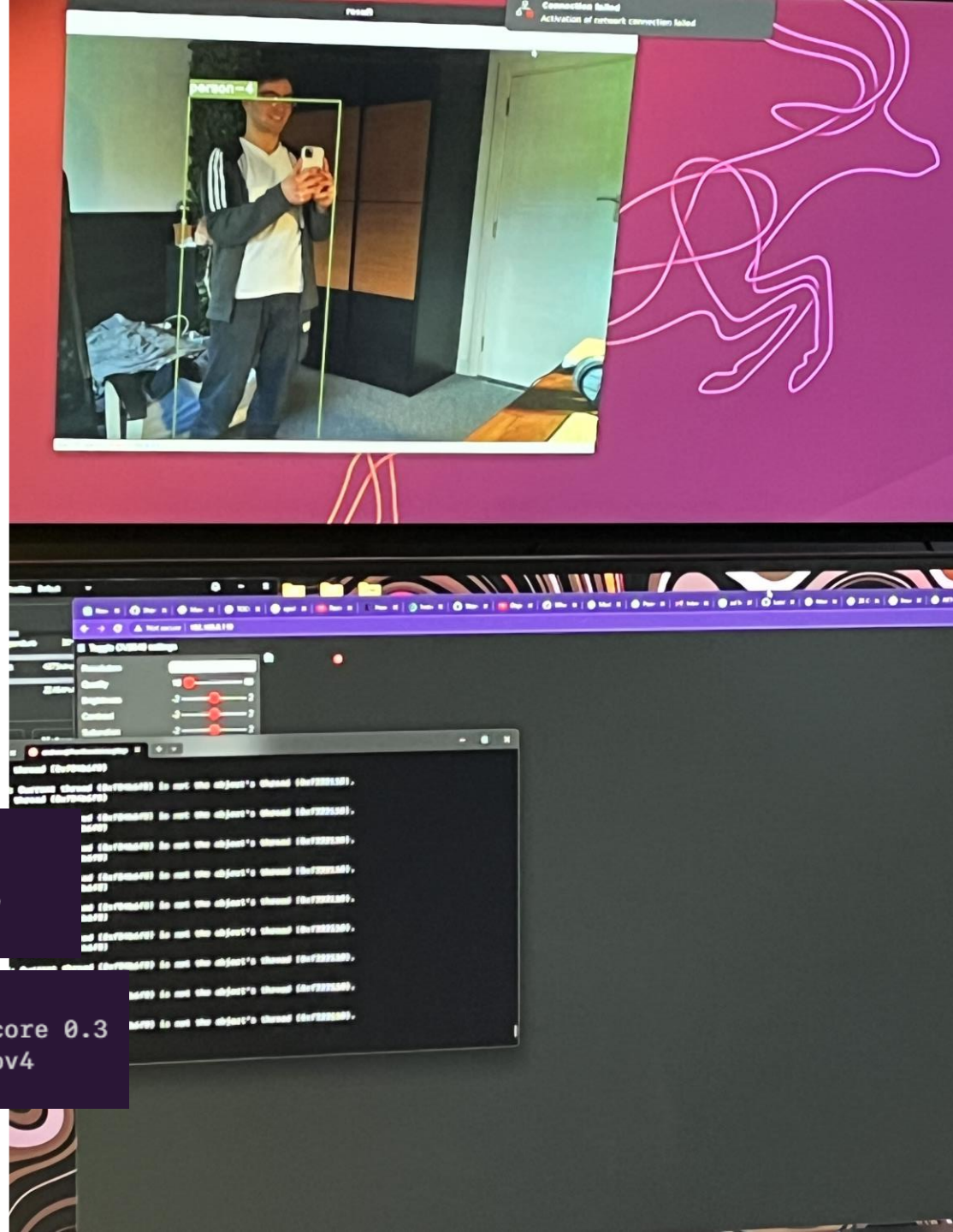
- This was quite a challenging aspect as there was a steep learning curve with Tensorflow.
- I was about to give up on the idea until I found a github repo with a pretrained model called yoloV4.

Using Pip

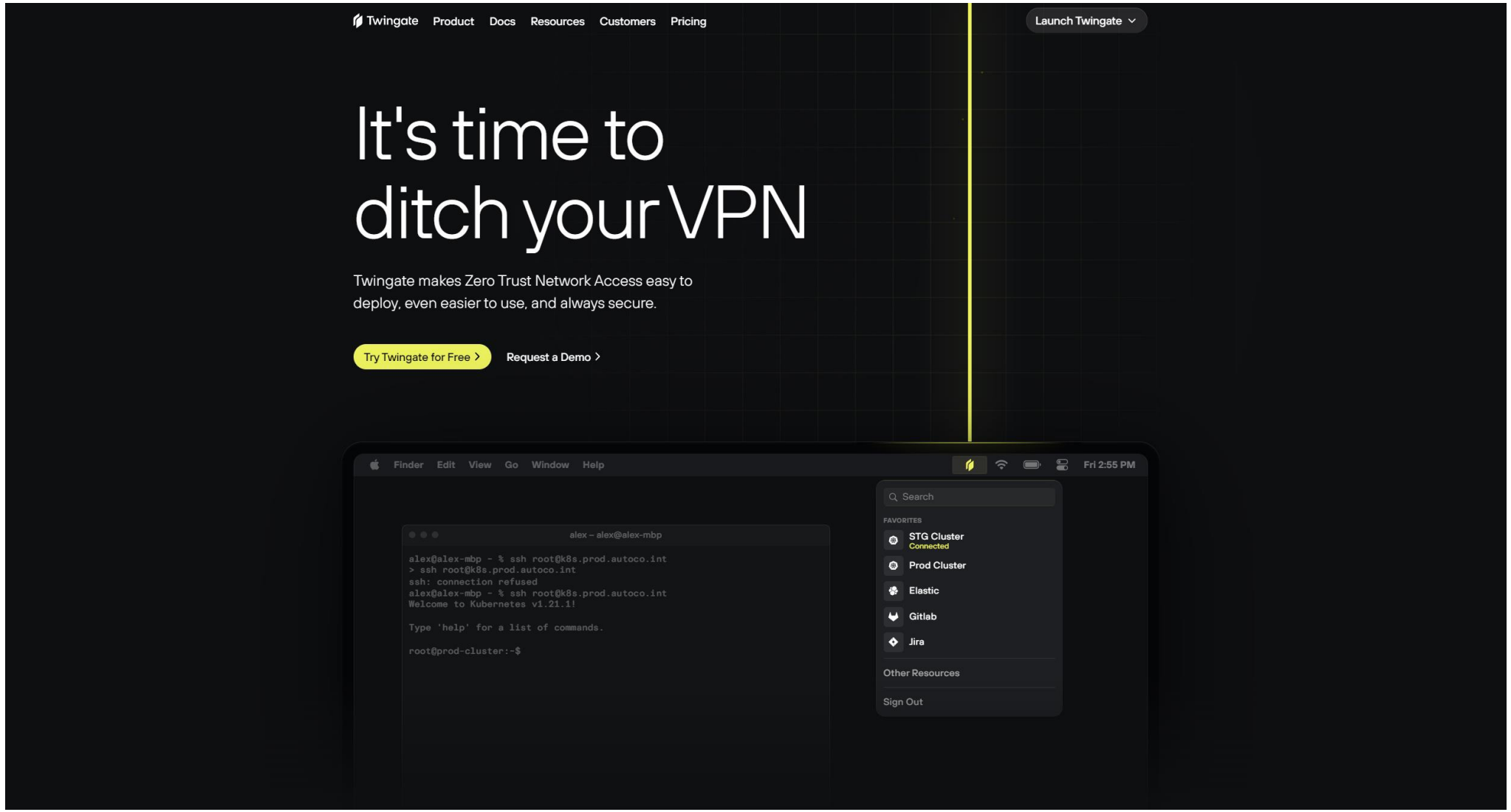
```
# CPU
pip install -r requirements.txt
```

```
## 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



Deploy Twingate for Free

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

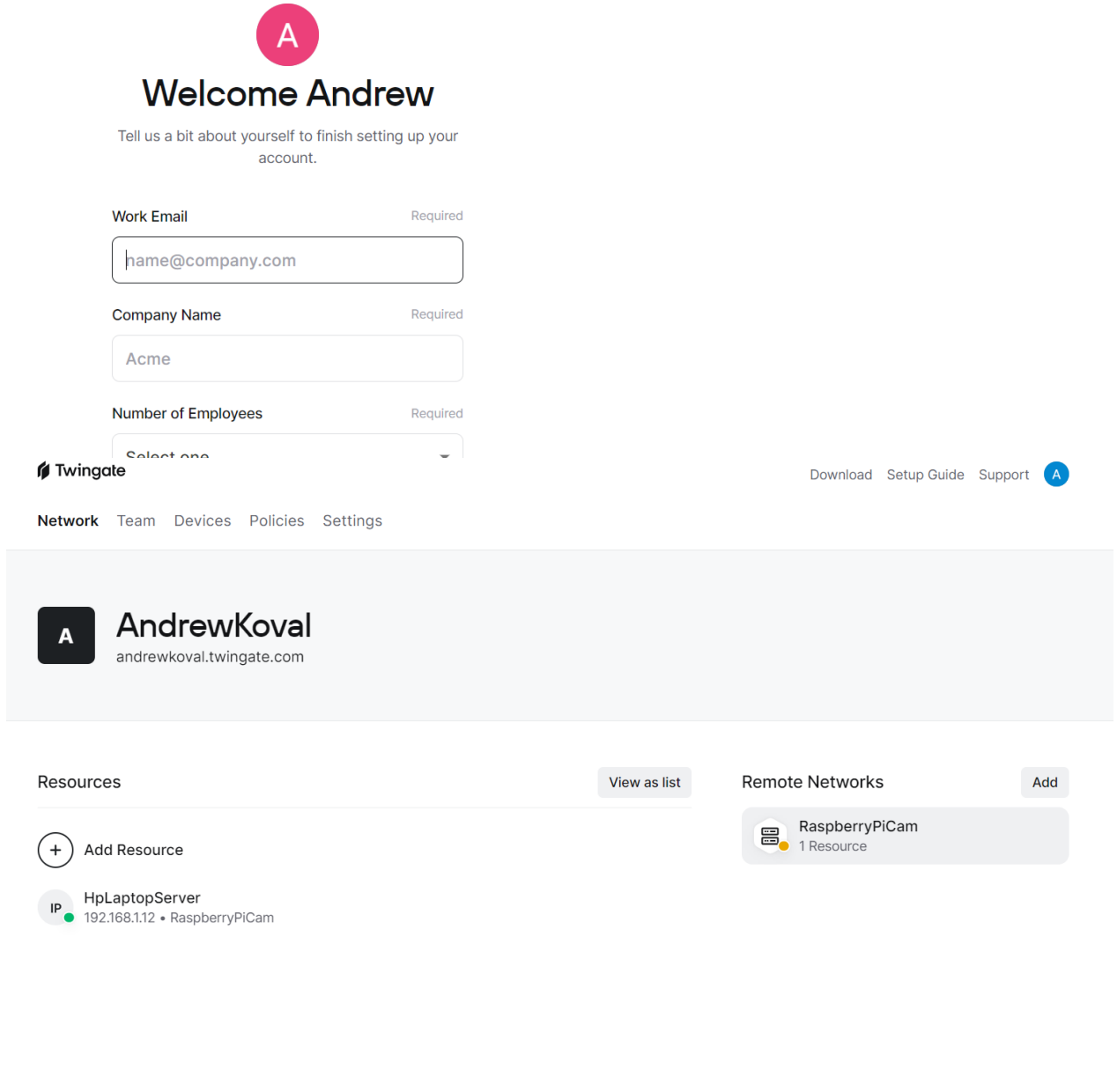
Sign up with Google

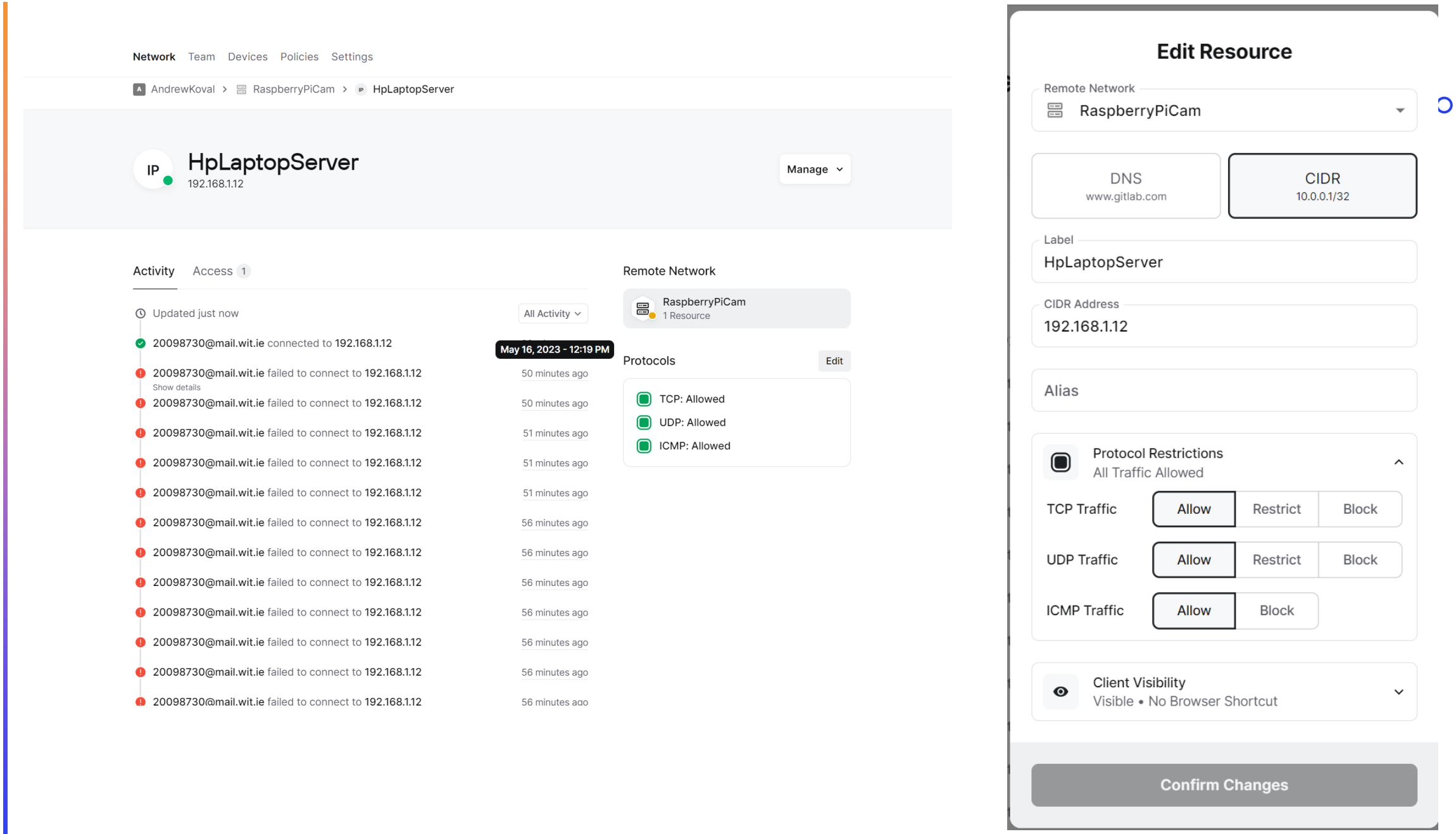
Sign up with Microsoft

Sign up with GitHub

Sign up with LinkedIn

Already using Twingate?
Sign in to an existing Network →




Network
Team
Devices
Policies
Settings
AndrewKoval
RaspberryPiCam
HpLaptopServer
IP
HpLaptopServer
192.168.1.12
Manage
Activity
Access 1
Updated just now
All Activity
20098730@mail.wit.ie connected to 192.168.1.12
May 16, 2023 - 12:19 PM
20098730@mail.wit.ie failed to connect to 192.168.1.12
50 minutes ago
Show details
20098730@mail.wit.ie failed to connect to 192.168.1.12
50 minutes ago
20098730@mail.wit.ie failed to connect to 192.168.1.12
51 minutes ago
20098730@mail.wit.ie failed to connect to 192.168.1.12
51 minutes ago
20098730@mail.wit.ie failed to connect to 192.168.1.12
51 minutes ago
20098730@mail.wit.ie failed to connect to 192.168.1.12
56 minutes ago
20098730@mail.wit.ie failed to connect to 192.168.1.12
56 minutes ago
20098730@mail.wit.ie failed to connect to 192.168.1.12
56 minutes ago
20098730@mail.wit.ie failed to connect to 192.168.1.12
56 minutes ago
20098730@mail.wit.ie failed to connect to 192.168.1.12
56 minutes ago
20098730@mail.wit.ie failed to connect to 192.168.1.12
56 minutes ago
20098730@mail.wit.ie failed to connect to 192.168.1.12
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