Smart Lock Educative Project

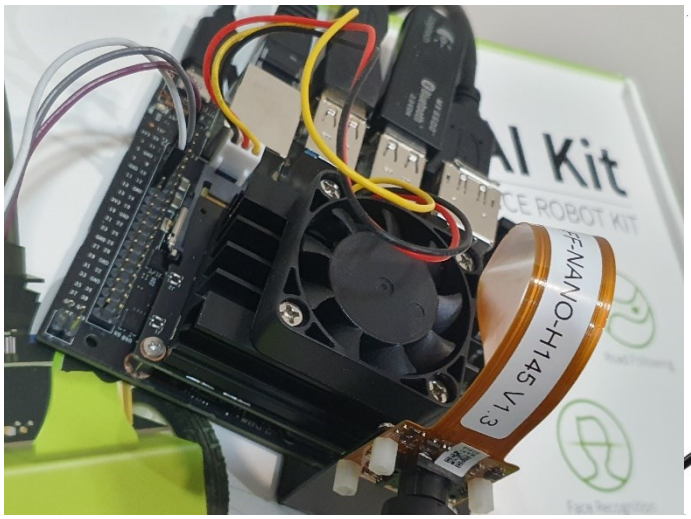
Contents

1. Project Overview 2
   1.1. Hardware Requirements 2
      1.1.1. Jetson Nano Development Kit (4GB) and RPi Cam 2
      1.1.2. Relay + lock power supply + lock 2
      1.1.3. Respeaker + aux speaker 3
   1.2. Preparing the S.O. 4
      1.2.1. openCV 4
      1.2.2. face_recognition and dlib 4
      1.2.3. Jetson Stats for jtop 5
      1.2.4. TTS pyttsx3 5
   1.3. Project Schematics 5
   1.4. Code explanation 6
      1.4.1. Train Code 6
      1.4.2. Live Streaming Code 6
2. Referee 8

1. Project Overview

1.1. Hardware Requirements

1.1.1. Jetson Nano Development Kit (4GB) and RPi Cam



Please take care to ensure a good power supply is being used. Although in this photo the jetson nano is shown outside the case, a protective case has been used, except for the trial setup.

In my case an old GeekPi A01 version was used, with 3-pin fan, camera mount, and power & reset button (jumpers J40 were moved to short circuit the auto power on) a MIPI camera was used. USB prevention charge jumper is also encouraged to be set, and to use a barrel power supply of 5V, 4A is also counselled, especially if you are using a jetBot and not only a development kit card.

Nowadays I will counsel a modern case design with access for the GPIO pins, like this one.

Jetson Nano Metal Case with 4020 5V PWM Cooling Fan,for Jetson Nano Developer Kit and B01(A02&B01),Jetson Nano Case with Camera Holder Reset,Power Button,for Waveshare IMX219 Camera,Wireless-AC8265

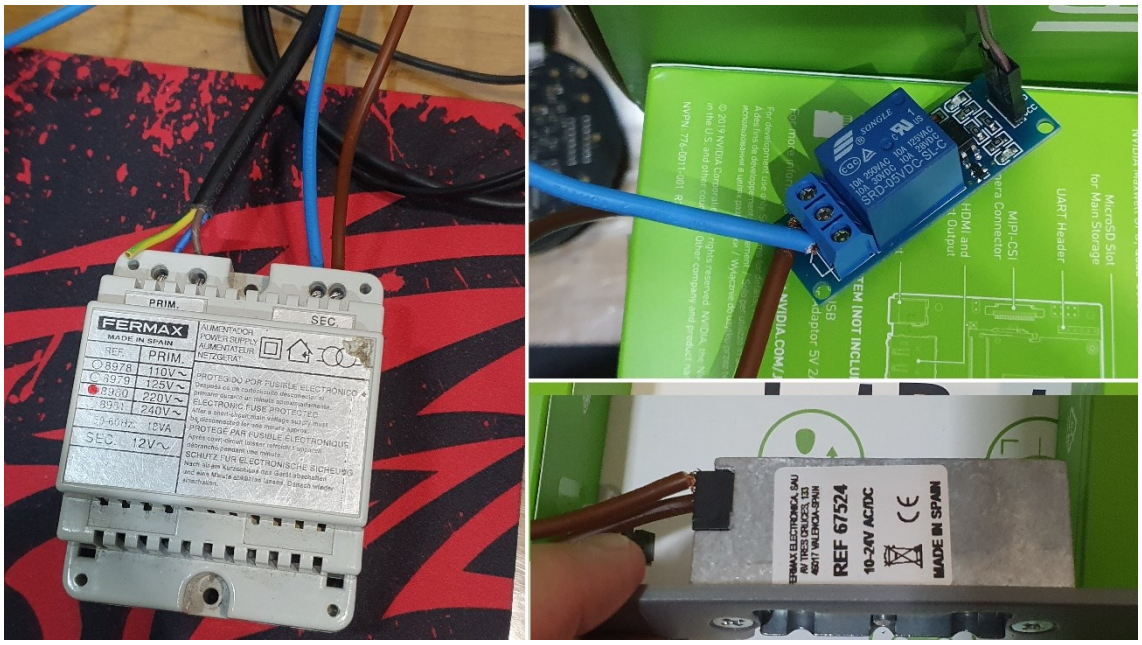
1.1.2. Relay + lock power supply + lock

FERMAX 67524 locking is being used. This locking can be operated thought any 10-24 AC/DC 2 wire signal.

FERMAX 8980 power supply is being used. This power supply can give 18VA power supply with an output of 12V at the secondary coil and an input of 220V 50-60Hz. It has an internal electronic fuse for higher protection.

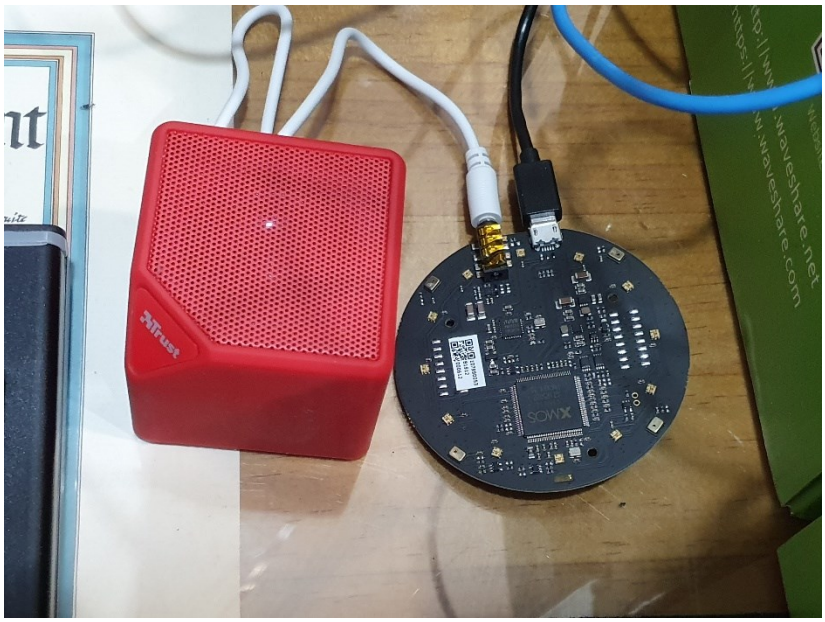
SONGLE SRD-05VDC-SL-C is being used. It is a relay that can be activated with the 3.3 voltage of the GPIO jetson nano pins.

It is important to state that as nearly all the relays, it can be configured as normally closed, or as normally opened (NC/NO) but PLEASE TAKE CARE that it is a LOW LEVEL TRIGGER model, so if you configure as NO, it closes when jetson stops supplying the 3.3 output. This conciguration is quite common in industrial locks, that will need to be disengaged in case of a power loss. It might be also configured to standard residential use, by changind the code or moving the configuration form NC to NO.



1.1.3. Respeaker + aux speaker

A USB 8 micro array respeaker is being used. The aux port is linked to a standard battery powered speaker. Mic is not necessary for this project, but speaker is, as we'd like to hear a confirmation message when the lock is opened.



1.2. Preparing the S.O.

After flashing from the latest nVidia image, several packages shall be installed:

- Visual studio code with python extension or similar. Cmake and pip3.
- OpenCV CUDA compiled, and dlib, for live streaming and deep learning.
- jtop for GPU and CPU usage monotoring
- face recognition (for face recognition)
- pyttsx3 (for text to speech)

1.2.1. openCV

Depending on your system, CUDA might not be enabled though OpenCV. This will not meant that your system wont be using CUDA, but in several rutines OpenCV will not run optimiced. The best way to check if your OpenCV is compiled with CUDA is using jtop (see later). The following links uses version 3, but with that version in my case OpenCV fails to create the living stream, so I had to upgrade to version 4.1

Compiling OpenCV with CUDA support - PyImageSearch

Compile OpenCV with Cuda from the source. | by Basavaraj PN | Medium

1.2.2. face_recognition and dlib

All required documentation is under the following repository:

ageitgey/face_recognition: The world's simplest facial recognition api for Python and the command line (github.com)

And please take note on this:

Installing on an Nvidia Jetson Nano board

- Jetson Nano installation instructions
  - Please follow the instructions in the article carefully. There is current a bug in the CUDA libraries on the Jetson Nano that will cause this library to fail silently if you don't follow the instructions in the article to comment out a line in dlib and recompile it.

1.2.3. Jetson Stats for jtop

Jtop after installed can be run from a terminal, and keep in parallel over the desktop so you can really check if CUDA are working.

rbonghi/jetson_stats: 📊 Simple package to monitoring and control your NVIDIA Jetson [Xavier NX, Nano, AGX Xavier, TX1, TX2] (github.com)

1.2.4. TTS pyttsx3

Sudo apt-get install espeak will install pyttsx3 and all their dependences

1.3. Project Schematics

Connect RPi cam to MIPI (if dev board A2, connect to the first slot)

Connect RJ45 to a router, or add a wifi card, so you can update and install libraries

Connect a USB keyboard and mouse, or configure to use jupyter (8888 port)

Connect the gnd relay pin to a gnd GPIO pin

Connect the vcc relay pin to a 3.3 GPIO pin

Connect the “in” relay pin to jetson GPIO pin 11

Connect the fan, and the jumpers if you are using a case.

Connect the FERMAX 8980 secondary (in serial wired configuration) to the NO and the ground pins of the relay.

Connect the FERMAX 8980 primary to a 220 50 Hz source (for Spain, other countries will require another electric configuration, and instead of model 8980 others like 8978 might be necessary)

Warning: FERMAX 8980 connection shall be done by an authorized professional. Electrical risk are present and health / equipment damages can arise if an improper manipulation is done.

FERMAX 8980 does not use an earth protection conductor, as it is double isolated.

1.4. Code explanation

The following lines give a short explanation about the code.

1.4.1. Train Code

Entrena.py collects a group of photographs from the folder image_dir and encodes it to be later used with face recognition. To skipping doing the training each time you want to use the program, pickle is used to write a file called 'entrenado.pkl' where the face information is saved.

Special care shall be dedicated to image_dir as need to be updated with the path where the student is going to execute the code, considering that the linux user for me is @carlos.

Jpg images shall be used, and an error will halt if too large files are used. The pope, the king, and other few faces were added, and the student shall load their own photograph and later run entrena.py prior executing live stream code.

1.4.2. Live Streaming Code

```
print (dlib.DLIB_USE_CUDA)
print (dlib.cuda.get_num_devices())
print (dlib.__version__)
print (cv2.__version__)
```

First, these prints were added to check dlib and cv2 version, and to check that cudas are being used.

As a RPi camara is being used, the label for managing the camara is as follows:

```
camSet='nvarguscamerasrc !  video/x-raw(memory:NVMM), width=3264, height=2464, format=NV12, framerate=21/1 ! nvvidconv flip-method='+str(flip)+' ! video/x-raw, width='+str(dispW)+', height='+str(dispH)+', format=BGRx ! videoconvert ! video/x-raw, format=BGR ! appsink'
```

The 3264 x 2464 makes reference to the cam objective resolution, but the stream resolution is controlled by dispH and dispW. The current 640 x 480 value is a good threshold, as higher values causes a significative drop in FPS.

- channel=11
- vel=0.25
- hora=time.time()
- repeticion=hora
- invel=int(1/vel)

channel is the pin used with the GPIO relay conexion, vel=0.5 and invel (inverse of vel) is set as counselled by the face_recognition libreary creator to increase FPS, but the quality loss makes necessary a better and closest image to be recognized.

Repeticion is later used to avoid the lock to be energized several times in a row, as it is not good for the item. Please check invel to be an integer

```
engine=pyttsx3.init()
engine.setProperty('rate',150)
engine.setProperty('voice','spanish')

#voices=engine.getProperty('voices')
#print (voices)
engine.setProperty('volume',2.0)
```

engine is the variable used with the TTS library. In this example Spanish voice, speed rate 150% and volume level 2 is used. The #lines enable the student to see the different languages available for them.

```
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
GPIO.setup(channel, GPIO.OUT, initial=GPIO.HIGH)
```

The GPIO needs to be considered initially in 3.3V level (high) as the relay is low level triggered. GPIO.BOARD names are used, as it is the simplest.

```
with open('entrenado.pkl','rb') as f:
    Names=pickle.load(f)
    Encodings=pickle.load(f)
```

We load the already trained images (one shall be of ourselves if we want to try with our face)

```
ret,frame=cam.read()
    frameLR=cv2.resize(frame,(0,0), fx=vel, fy=vel)
    frameRGB=cv2.cvtColor(frameLR,cv2.COLOR_BGR2RGB)
```

We do the vel resize to improve FPS, and we adapt the BGR CV2 standard configuration to the international used RGB

```
facePositions=face_recognition.face_locations(frameRGB, model='cnn')
    allEncodings=face_recognition.face_encodings(frameRGB, facePositions)
```

We identify all faces in each frame, and we see their locations to later encode and compare with the trained ones.

```
        matches=face_recognition.compare_faces(Encodings,face_encoding)
```

Compare_faces return true if a picture contains any face previously encoded and trained (a picture can have many faces, not a single one)

```
  if (repeticion < time.time()):
                GPIO.output(channel, GPIO.LOW)
```

```
                time.sleep(0.5)
                GPIO.output(channel, GPIO.HIGH)
                #engine.say("The door is opened, you are wellcome")
                engine.say("Puerta abierta")
                engine.runAndWait()
                repeticion = time.time()+10
```

If a positive detection happens, we energize the coil of the relay for 0.5 seconds. Also if I have already done this in the previous 10 seconds, I do not energize.

The TTS warns about the fact that the door has been opened.

Engine.runAndWait ensures to not start a new speak till the previous one finished.

```
dt=time.time()-hora
    fps=1/dt
    fpsM=0.80*fpsM+0.20*fps
    hora=time.time()
    cv2.rectangle(frame,(0,0),(100,40), (0,0,255),2)
    cv2.putText(frame, str(round(fpsM,1)) + 'fps',(0,25), font, 0.75, (0,
0,255),2)
```

FPS are calculated, and a basic filter is used to avoid peaks, like a movil mean.

2. Referee

ageitgey/face_recognition: The world's simplest facial recognition api for Python and the command line (github.com)

Technology Tutorials | Making The World a Better Place One High Tech Project at a Time. Enjoy! (toptechboy.com)