Arne Six
2 NMD

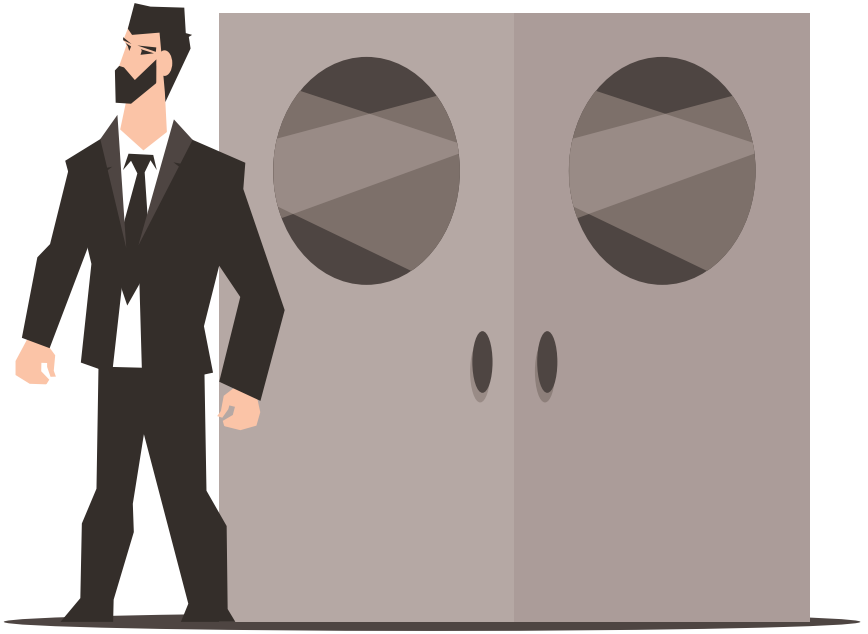
Project SecuCam

Gemaakt in opdracht van ArteveldeHogeschool

Discover

U heeft ongetwijfeld al eens last gehad van verdachte individuën rond uw eigendom, of een nieuwsgierige vriend die graag op je berichten leest. Zelf heb ik hier ook last van, vandaar dat ik op het idee kwam om alles achter een beveiling-slaag te steken en zo ongewenste pottekijkers buiten te houden.

SecuCam zal hiervoor zorgen, dankzij de ingebouwde gezichtsherkenning zal uw apparaat een alarm laten afgaan wanneer er een onbekende gebruiker het apparaat gebruikt



Define

Er bestaan al verschillende beveilingssystemen, de meest voorkomende zijn diegene op je gsm of smart doors. Ik heb daarom ook gekozen voor apparaat te maken dat gebruikt kan worden in hoog beveiligde instellingen.

Noodzakelijke software/hardware

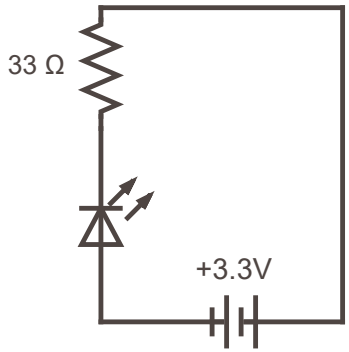
Dit project heeft natuurlijk enkele items nodig, deze zijn een Raspberry PI als main computing system, een camera voor gezichten te detecteren, een speaker voor een alarm af te spelen en een paar LED die aangeven of de gebruiker bekend – en of onbekend is.

Mijn inspiratie komt vooral van Michael Reeves die als schoolproject ook een soort securitylock gemaakt heeft en vaak met IOT bezig is. Vandaar dat ik het idee had om een beveilingsapp te maken.

Design

De SecuCam zal geen gebruik maken van een web interface, of toch niet in de eerste stadia. Er zal ongetwijfeld wel een interface nodig zijn in de toekomst.

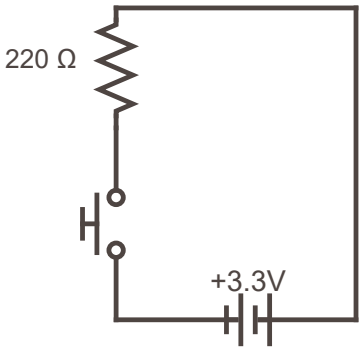
Electronisch Schema



Twee LED's

-Raspberry Pi GPIO bron van 3.3V

- Weerstand van 33 ohm



één Drukknop

-Raspberry Pi GPIO bron van 3.3V

- Weerstand van 220 ohm

Develop

```python
class App:
    def __init__(self):
        self.button_state: bool = False
        self.gpio: GPIO = GPIO()
        self.gpio.bind_to(self.update_button_state)

        self.camera: Camera = Camera(video_device=0)

    def update_button_state(self, button_state):
        self.button_state = button_state

        self.validate()
```

```python
class GPIO:
    def __init__(self):
        self._button_state: bool = False
        self._observers = []
        self.__button = Button(5)

        Process(target=self.run).start()

    @property
    def button_state(self):
        return self._button_state

    @button_state.setter
    def button_state(self, value):
        self._button_state = value
        for callback in self._observers:
            callback(self.button_state)

    def bind_to(self, callback):
        self._observers.append(callback)

    def run(self):
        while True:
            if self.__button.is_pressed:
                self.button_state = True
```

```python
def detect(self):
    """
    Detects the faces and returns a bool if the face is known.
    :return: Detected
    """
    face_encodings = []

    success, frame = self.__cap.read()

    small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)
    rgb_small_frame = small_frame[:, :, ::-1]

    if success:
        self.__face_locations = face_recognition.face_locations(rgb_small_frame)
        face_encodings = face_recognition.face_encodings(rgb_small_frame, self.__face_locations)

        if not self.__face_locations:
            return Detected.EMPTY

        for face_encoding in face_encodings:
            matches = face_recognition.compare_faces(self.__known_face_encodings, face_encoding)

            face_distances = face_recognition.face_distance(self.__known_face_encodings, face_encoding)

            if not face_distances:
                continue

            best_match_index = np.argmin(face_distances)

            if matches[best_match_index]:
                return Detected.VALID

            return Detected.INVALID
```

```python
class Detected(Enum):
    EMPTY = 0
    VALID = 1
    INVALID = 2
```

Deliverables

Om dit project na te maken heb je enkele dingen nodig:

- 1x Raspberry Pi
- 1x Pi Camera
- 2x LED (groen en rood)
- 3x Jumper wires
- 3x weerstanden

Softwarematig heb ik de volgende bibliotheken nodig :

- Taal : Python
- Cmake
- Dlib
- Face-recognition
- Numpy
- Python-opencv