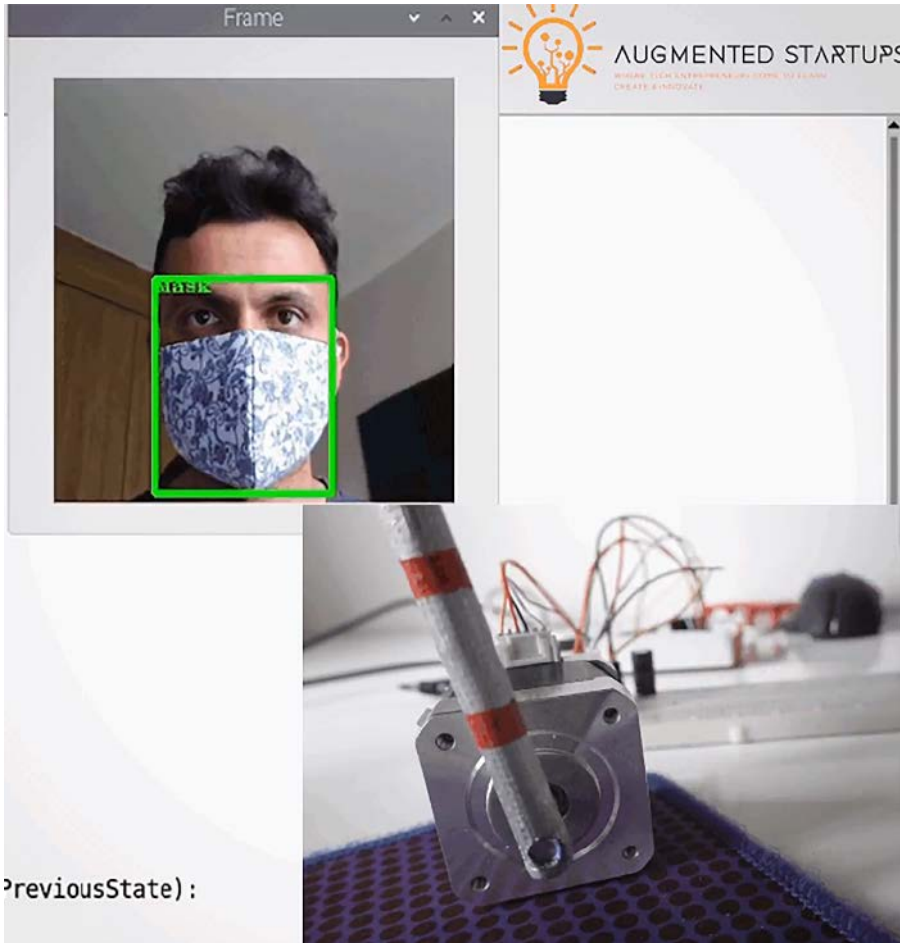
Wykrywacz maseczki

Jeszcze do niedawna maseczki były naszą codziennością i w niektórych okolicznościach ciągle są wymagane (piszemy ten artykuł w czasie, gdy nakaz jeszcze obowiązuje) – w sklepach, centrach handlowych itd. Upominanie nienoszących ich osób jest uporczywe, dlatego też dobrze jest oddać to zadanie w ręce maszyny.



Opisany w artykule prosty system został wyposażony w kamerę, komputer jednopłytkowy Raspberry Pi oraz oprogramowanie oparte na OpenCV. Służy do wykrywania w czasie zbliżonym do rzeczywistego, czy osoba, na którą patrzy system, ma na twarzy założoną maseczkę. System kontroluje pracę prostego szlabanu zbudowanego na silniku krokowym. To tylko demonstracja działania, może on sterować dowolnym innym urządzeniem, np. wyświetlać informację o konieczności noszenia maseczki w danym miejscu.

Zaprezentowany algorytm wykorzystuje detektor obiektów COVID-19 mask/no-mask opracowany przez Google Colab. Jednak nic nie stoi na przeszkodzie, aby zaimplementować inny detektor obiektów lub przygotować własny. W dalszej części artykułu znajduje się skrócony opis trenowania użytej tutaj sieci neuronowej do zastosowań widzenia maszynowego, dzięki czemu możliwe jest samodzielne nauczenie systemu rozpoznawania innego rodzaju obiektów.

Zastosowane elementy

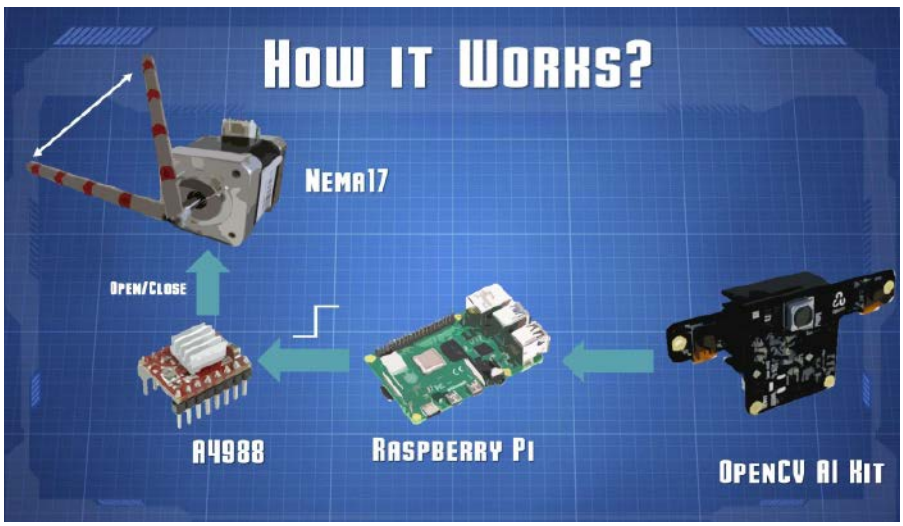
Do zestawienia omawianego urządzenia potrzebne będą następujące komponenty:

- Komputer jednopłytkowy Raspberry Pi z typowymi akcesoriami (zasilacz, karta pamięci itd.);
- Kamera OpenCV AI Kit. Autor zastosował model OAK D, ale można użyć również OAK-1;
- Silnik krokowy. Autor wybtrał silnik w rozmiarze NEMA17, lecz można sterować dowolnym innym silnikiem, kompatybilnym z poniższym driverem lub zastosować własny driver;
- Sterownik silnika krokowego oparty na układzie A4988 lub inny, kompatybilny ze sterowaniem typu STEP/DIR;
- Zasilacz 12 V o dostatecznej wydajności prądowej do zasilenia wybranego silnika i drivera;
- Płytka uniwersalna lub płytka stykowa i drobne elementy, takie jak kondensatory i przewody do poprowadzenia zworek.

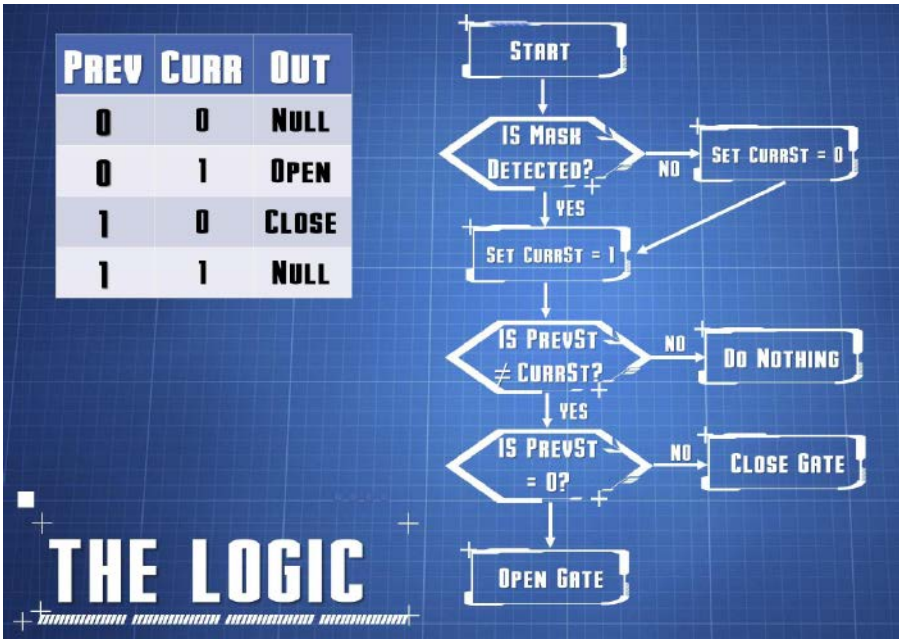
Opcjonalnie przydatna jest jeszcze drukarka 3D, która pozwala na wydrukowanie elementów mechanicznych, jakimi sterować ma silnik krokowy. W przypadku tego systemu jest to prosty szlaban, ale tylko wyobraźnia i moc silnika ogranicza to, co można do niego podłączyć w celu sygnalizacji braku maseczki.

Jak to działa

Obraz osoby z maseczką (lub bez) jest przechwytywany przez zestaw kamery AI OpenCV. Kamera zbiera 30 klatek na sekundę i z tą prędkością obrazy są przekazywane do Raspberry Pi poprzez USB. W samym Raspberry Pi pracuje oprogramowanie, oparte na platformie Roboflow, które analizuje zgromadzone obrazy.



Rysunek 1. Schemat blokowy urządzenia



Rysunek 2. Schemat blokowy ogólnego algorytmu działania urządzenia

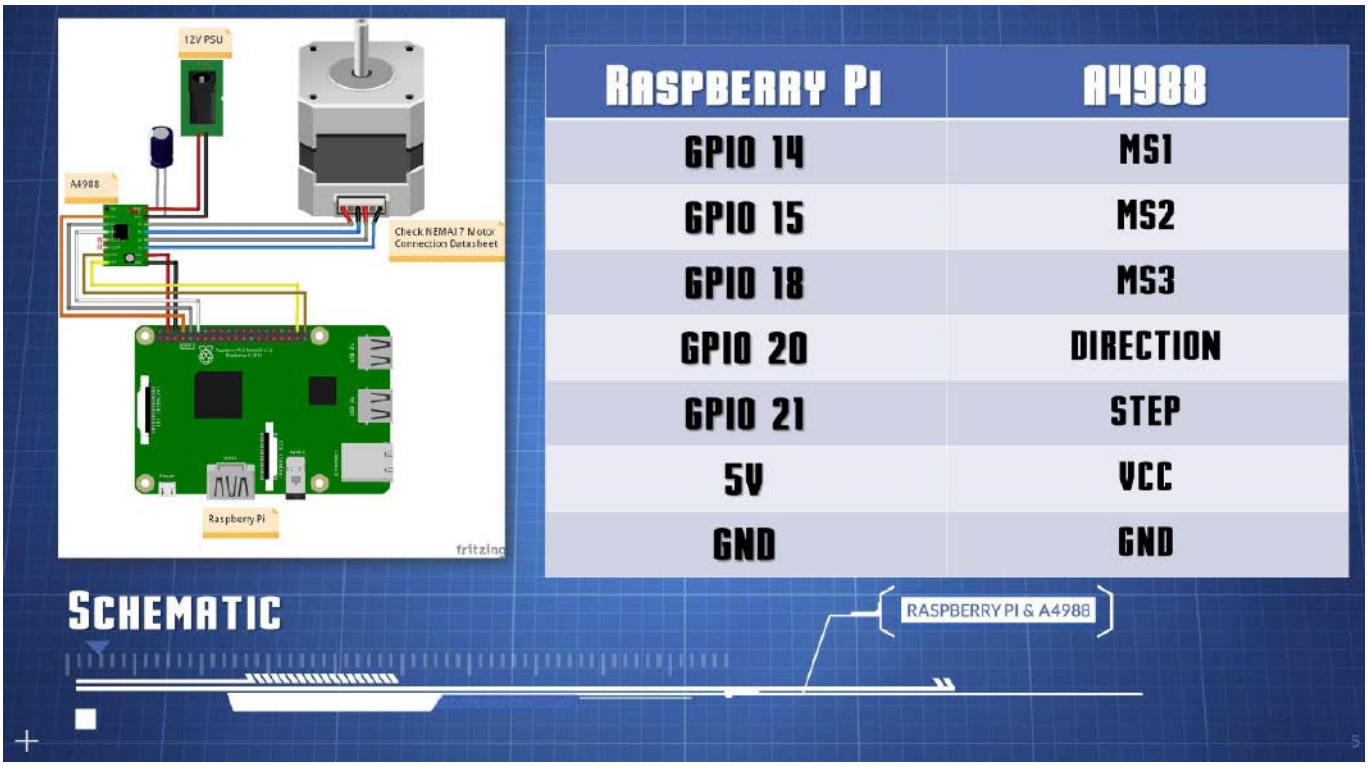
System AI, zaimplementowany w Raspberry Pi, obsługuje pojedyncze wyjście cyfrowe, którego stan odpowiada obecności maseczki na osobie, na którą patrzy kamera. Pojedyncza linia GPIO steruje driverem silnika krokowego i otwiera oraz zamyka szlaban. Algorytm rozpoczyna działanie zaraz po uruchomieniu urządzenia. W przypadku wykrycia twarzy sprawdzane jest, czy dana osoba nosi maseczkę, czy nie. Jeśli nie, to stan linii wyjściowej ustawiany jest na 0, w innym przypadku, jeśli maseczka zostanie wykryta, stan wyjściowy ustawiany jest na 1. W następnym kroku algorytm porównuje, czy poprzedni stan jest taki sam jak aktualny. Zasadniczo poszukuje on przejścia lub zdarzenia wyzwalającego zbocze. Jeśli nie ma takiego przejścia, nic nie robi, ale jeśli wykryte zostanie przejście, to sprawdzany jest poprzedni stan – jeśli był on zero, to układ otwiera bramkę, a w innym przypadku nic się nie dzieje. Jeśli jednak aktualny stan równy jest 1, co oznacza wykrycie maski, to szlaban jest otwierany. Jeśli poprzedni i obecny stan to jedynka, znaczy to, że użytkownik cały czas ma na sobie maseczkę i szlaban pozostaje cały czas otwarty.

Można by zapytać, dlaczego układ działa w tak skomplikowany sposób – monitoruje obecny i aktualny stan oraz porównuje je ze sobą? W tym systemie chcemy, aby brama otwierała się i zamykała. Jeśli nie będziemy kontrolować obecnego stanu, istnieje ryzyko, że system spróbuje otworzyć już raz otwartą bramkę, co będzie oznaczało przekręcenie szlabanu o kolejne 90 stopni i tak dalej.

Opracowany system zapewnia, że silnik nie przekręca nadmiernie szlabanu w żadnym kierunku. Łatwiejszą alternatywą jest to, że można otworzyć bramę po wykryciu maski i zamknąć ją po określonej liczbie sekund, a następnie otworzyć ponownie po wykryciu maski.

Schemat

Schemat jest bardzo prosty, co pokazuje **rysunek 3**. Widać tam diagram połączeń komputera jednopłytkowego Raspberry Pi i sterownika silnika krokowego oraz sposób podłączenia silnika i zasilacza. Po prawej stronie rysunku umieszczono dodatkowo tabelkę, zawierającą listę połączeń poszczególnych linii GPIO Raspberry Pi. Po podłączeniu sterownika do systemu należy ustawić ograniczenie prądu układu A4988, aby było dopasowane do zastosowanego silnika. Za pomocą potencjometru regulowane jest napięcie odniesienia VREF w układzie. Ograniczenie prądowe jest w przybliżeniu równe dwukrotności napięcia VREF, co oznacza, że aby prąd uzwojenia silnika krokowego był równy 1 A, należy ustawić VREF=0,5 V. Warto pamiętać, że dla innych



Rysunek 3. Schemat połączeń Raspberry Pi, sterownika silnika krokowego i samego silnika (po lewej) oraz tabelka podsumowująca połączenia RPi i modułu z układem A4988 (po prawej)

sterowników silników krokowych sposób konfiguracji prądu maksymalnego może być inny – należy zastosować się do zaleceń producenta modułu.

Zestaw kamery AI OpenCV wystarczy podłączyć za pomocą dostarczonego kabla USB C. Aby sprawdzić, czy silnik krokowy jest prawidłowo podłączony (czy dobrze odnaleźliśmy pary uzwojeń), można podłączyć do nich diodę LED i zakręcić osią silnika. Jeśli dioda LED podłączona jest do jednego z uzwojenia, zaświeci się pod wpływem prądu zaindukowanego w uzwojeniu. To samo można powtórzyć z drugim uzwojeniem.

Oprogramowanie

Oprogramowanie do opisywanego modułu dostępne jest na repozytorium autora na GitHubie (https://bit.ly/3yCkp3L). Wystarczy sklonować zawartość wskazanego repozytorium w wygodnym dla nas miejscu. W pierwszej kolejności dobrze jest zrealizować samouczek, zawarty w folderze z aplikacją o numerze jeden. W artykule źródłowym znaleźć można m.in. film, który pokazuje, w jaki sposób należy zrealizować to zadanie.

Dalej należy skupić się na zawartości aplikacji 3. Znajdują się tam cztery pliki, o których należy pamiętać. Bazowe pliki aplikacji to *depthai_utils.py* i *main.py*.

Jeśli chcemy pominąć analizę oprogramowania, wystarczy uruchomić program na Raspberry Pi, wpisując w terminalu następujące polecenie, uruchamiające odpowiedni skrypt:

```
python3 main.py
```

Skrypt powinien się od razu uruchomić i zacząć kontrolować silnik krokowy szlabanu. Jeśli się nie uruchomi, przyczyną może być brak którejś z potrzebnych bibliotek. Aby je dodać, wystarczy wpisać polecenie:

```
python3 -m pip install -r requirements.txt
```

Wykorzystany model ML

Wykorzystana sieć neuronowa pozwala na zaawansowane wykrywanie obiektów. Sieć korzysta z frameworku Tensorflow, w ramach którego uruchamiana jest sieć SSD MobileNet V2. Sieć wytrenowana została w ramach Google Colab z użyciem 700 zdjęć osób w maskach i bez nich, na które naniesiono około 3600 oznaczeń.

Dokładny opis procesu uczenia sieci zawarto w repozytorium, które można znaleźć w linkach na końcu. Pokazany jest tam tzw. notatnik, który zawiera zapis krok po kroku procesu instalacji Tensorflow, MobileNet V2 i wymaganych przez nie bibliotek, a następnie uczenia sieci neuronowej rozpoznawania danych obiektów i generowania plików, wykorzystywanych przez algorytm.

Obrazy są złożone: różnią się znacznie skalą i kompozycją. Niemniej jednak detektor obiektów radzi sobie całkiem nieźle z tym stosunkowo niewielkim zestawem danych do takiego zadania. Szkolenie z zastosowaniem serwerów w chmurze, udostępnianych przez Google Colab, trwa około 2 godzin.

W zależności od tego, jaki GPU Colab przypisuje do instancji danego notatnika, szkolenie z użyciem 10 tysięcy kroków zajmuje od 1,5 do 2,5 godziny. To całkiem krótki czas, wymagany do wygenerowania gotowej do pracy sieci, zważywszy na jakość uzyskanego wyniku.

Następnie wykonywane są kolejne kroki, aby przekonwertować wynik nauki sieci na obiekt typu *blob* z wykorzystaniem OpenVINO jako etapu pośredniego. Pozwala to finalnie uruchomić model widzenia maszynowego na module DepthAI.

Nikodem Czechowski, EP

Źródła:

1. https://bit.ly/2Sob6nw
2. https://bit.ly/2SoDpCj
3. https://bit.ly/3bS8WmJ