Inteligentna rękawiczka dla rowerzystów

Lato to doskonały okres na jazdę rowerem i warto przy tym zadbać o bezpieczeństwo, szczególnie wieczorem i w nocy. Rowery nie są wyposażane w kierunkowskazy, ale ich dodanie poprawia widoczność pojazdu na drodze, zwiększając tym samym bezpieczeństwo, nie tylko rowerzysty.

Rower, zgodnie z przepisami (rozporządzenie Ministra Transportu z 6.06.2013 w sprawie warunków technicznych pojazdów oraz ich niezbędnego wyposażenia, Dz. Ustaw 2013 poz. 951.) powinien posiadać: co najmniej jedno światło barwy białej lub żółtej, skierowane do przodu; co najmniej jedno światło odblaskowe barwy czerwonej, o kształcie innym niż trójkąt; co najmniej jedno światło pozycyjne barwy czerwonej, skierowane do tyłu. Nie ma ani słowa o kierunkowskazach, które w warunkach drogowych, szczególnie w mieście, są bardzo przydatne.

Rowerzysta, na ogół, sygnalizuje chęć skrętu poprzez wyciągnięcie ręki w odpowiednim kierunku, jednakże gest ten jest słabo widoczny, szczególnie przy niedostatecznym oświetleniu itp. Problem ten rozwiązuje sygnalizacja manewru kierunkowskazami. Autor opisanego poniżej projektu stworzył właśnie taki system oświetlenia. Składa się on ze specjalnych rękawiczek z sensorami gestów oraz panelu LED, montowanego np. na plecaku, bądź plecach, który wyświetla informacje o kierunku jazdy roweru.

Inspiracją do stworzenia projektu był fakt, że autor dojeżdża codziennie do pracy rowerem. Byłaby to duża przyjemność, gdyby nie to, że mieszka on w jednym z najbardziej zatłoczonych miast Francji, gdzie wypadki między samochodami i rowerzystami zdarzają się dość często. Problem pogłębia brak ścieżek rowerowych i konieczność poruszania się ulicami. Projekt ma pomóc użytkownikom dróg w lepszej komunikacji. „Z mojego punktu widzenia większość niedomówień, jakie widzę między użytkownikami dróg, wynika z tego, że niektórzy użytkownicy dróg źle interpretują zachowanie innych" – opisuje autor, mówiąc o projekcie. „Za pomocą tego urządzenia chcę sprawić, aby użytkownicy dróg lepiej się rozumieli: strzałki wskazują kierunek skrętu, a dodatkowo można wyświetlać tekst".

Słowo „inteligentna" w nazwie projektu znalazło się nieprzypadkowo. W systemie zastosowano bowiem techniki uczenia maszynowego – sztucznej inteligencji. A rękawiczka? Projekt powstał zimą, więc naturalne było zintegrowanie detektora gestów z zimowymi rękawiczkami. Autor jednak szybko zdał sobie sprawę, że to nie najlepszy pomysł. W miejscu gdzie mieszka, latem jest dosyć gorąco, więc rękawiczka byłaby niekomfortowa. Dlatego cały system finalnie umieszczono w obudowie, montowanej na dłoni, ale nazwa projektu pozostała taka sama.

Ze względu na długość listingów, są one dostępne na stronie artykułu: ***https://bit.ly/37UN4Uz*** – dla prenumeratorów, jeszcze przed ukazaniem się wydania papierowego. Równolegle do pobrania ze strony ***http://media.avt.pl***.

Zasada działania systemu

Układ rękawiczki bazuje na płytce z rodziny Arduino, która zbiera dane z żyroskopu i akcelerometru. Oprogramowanie Arduino wykorzystuje algorytm uczenia maszynowego, osadzonego na frameworku tinyML, który jest zoptymalizowaną wersją TensorFlow Lite, przeznaczoną do pracy na platformach wbudowanych o ultraniskim poborze energii, takich jak Arduino. Algorytm umożliwia także rozpoznawanie gestów ręki: każdy jej ruch jest analizowany i klasyfikowany (ręka przechylona w lewo, prawo, przód, tył itp.).

Sygnał z Arduino, odpowiedzialny za rozpoznawanie gestów, jest przesyłany przez Bluetooth Low Energy (BLE) do innego mikrokontrolera podłączonego do matrycy diod elektroluminescencyjnych (LED), umieszczonej np. na plecaku. Matryca wyświetla wzory zależne od gestów użytkownika, aby inni użytkownicy drogi wiedzieli co zamierza rowerzysta. Dla przykładu, układ może wyświetlać strzałki w prawo lub w lewo, sygnalizując zamiar skrętu, ale może także wyświetlać tekst.

Projekt inspirowany jest dwoma innymi rozwiązaniami. Jak często bywa z systemami tworzonymi w ekosystemie Arduino, autor

nie zaczynał od zera, ale skorzystał z doświadczeń innych, aby pójść o krok dalej i zrobić coś bardziej zaawansowanego. Pierwszym projektem, jest system rozpoznawania gestów na platformie Arduino Nano 33 BLE SENSE, który opisano tutaj: ***https://bit.ly/2Yod3km***. Drugim nie jest konkretny projekt, ale koncepcja paneli LED dla rowerzystów. Istnieje wiele tego rodzaju projektów. Niektóre są plecakami ze zintegrowaną matrycą LED, inne składają się z matrycy, którą można przymocować w dowolnym miejscu. We wszystkich przypadkach panele sterowane są za pomocą klasycznego pilota.

Wymagane elementy

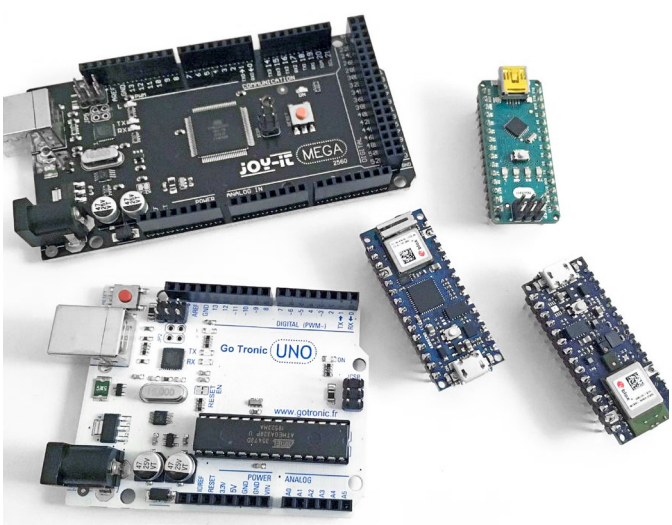
Do wykonania obudów poszczególnych elementów, potrzebny będzie dostęp do drukarki 3D. Wszystkie elementy drukowane są z PLA, więc drukarki dostępne na rynku powinny sobie poradzić. Dodatkowo, do skręcenia i montażu obudowy potrzebne będą śrubki i nakrętki M3 oraz rzep (obie strony).

Spośród komponentów elektronicznych potrzebne będą:

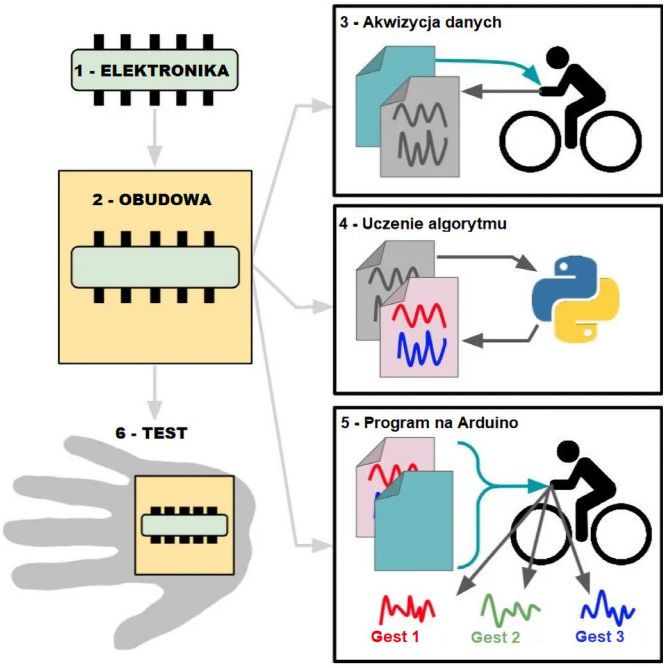
- moduł Arduino Nano 33 BLE SENSE,
- inny moduł z wbudowanym interfejsem BLE (Arduino Nano 33 BLE, Arduino 33 BLE SENSE, Arduino NANO 33 IOT, ESP32, etc.) – autor zdecydował się na zastosowanie ESP32,
- adresowalny pasek LED (np. WS2812B) – autor zastosował w swoim projekcie 160 diod, do budowy macierzy 20×8 LED,
- poczwórny translator poziomów (3,3 V do 5 V) – 74AHCT125 lub analogiczny układ z elementów dyskretnych,
- kondensator 1000 µF.
- trzy przyciski SPDT,
- płytka uniwersalna i cienki kabelek do połączeń (np. kynar),
- bateria 9 V z stosownym klipsem,
- powerbank z wyjściem USB.

Na rynku jest wiele ciekawych modułów Arduino, jak pokazano na **fotografii 1**, ale autor zdecydował się na zastosowanie Arduino Nano 33 BLE SENSE, ponieważ jako jedyna posiada wbudowane, odpowiednie czujniki i obsługuje framework Tensorflow Lite. Co ważne, wszystkie płytki z rodziny Arduino Nano 33 (IOT, BLE oraz BLE SENSE) posiadają interfejs Bluetooth, dzięki czemu mogą sprawdzić się jako kontroler matrycy LED, odbierający sygnały Bluetooth.

Dodatkowo, przed przystąpieniem do realizacji projektu, warto zapoznać się z podstawami uczenia maszynowego na Arduino. Można zacząć od kursu dostępnego tutaj: ***https://bit.ly/2Yod3km***. Znajduje się tam wprowadzenie do klasyfikacji gestów z wykorzystaniem algorytmów AI.



Fotografia 1. Przykładowe moduły z ekosystemu Arduino



Rysunek 2. Plan pracy, jaki przyjął autor projektu

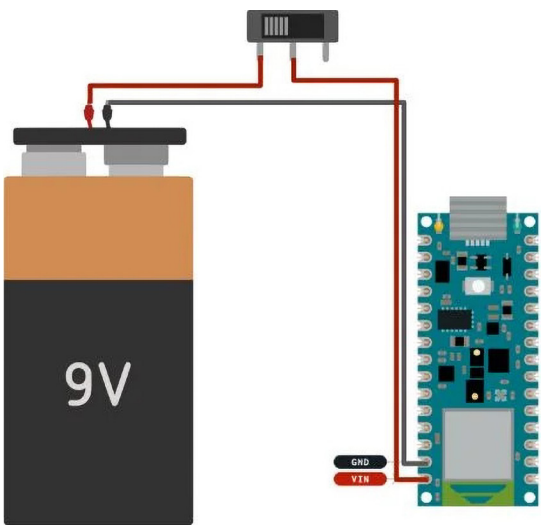
Rękawiczka

Ogólny plan pracy nad urządzeniem, z podziałem na etapy, pokazano na **rysunku 2**. Obejmuje on tak pracę nad sprzętem (elektroniką i jej integracja z rękawiczka oraz obudową), jak i prace nad inteligentnym oprogramowaniem do sterowania światłami.

Układ elektroniczny, który instaluje się w rękawiczce, jest bardzo prosty. Składa się z trzech elementów (**rysunek 3)**: modułu Arduino Nano 33 BLE SENSE, baterii 9 V do zasilania oraz włącznika (autor zastosował hebelkowy, ale dowolny przełącznik w konfiguracji SPDT będzie się tutaj nadawał). Dzięki temu, że wszystkie wymagane sensory (żyroskop i akcelerator) zintegrowane są na płytce Arduino, nie ma potrzeby podłączania dodatkowych czujników do płytki, co upraszcza cały układ.

Obudowa dla wykrywacza gestów wykonana została w całości w druku 3D. Składa się z dwóch plastikowych, wydrukowanych elementów i paska z rzepem, który służy do zamontowania urządzenia na dłoni. Obudowę pokazano na **fotografii 4**.

Pierwsza żółta część zawiera płytkę Arduino, baterię i przełącznik. Autor dodał wycięcie, które pozwala na ładowanie zastosowanego akumulatora (ogniwo w formie baterii 9 V, ładowane poprzez microUSB). Wycięcia pozwalają również na programowanie modułu Arduino bez konieczności rozbierania obudowy. W części czarnej,



Rysunek 3. Schemat elektroniki w rękawiczce

Fotografia 4. Obudowa sensora

przygotowane zostały podobne otwory jak w pierwszej części obudowy. Dodatkowo jest też otwór w miejscu, gdzie znajduje się dioda LED RGB na płytce. Dzięki temu widoczne jest światło diody, mimo że układ zamknięty jest w obudowie.

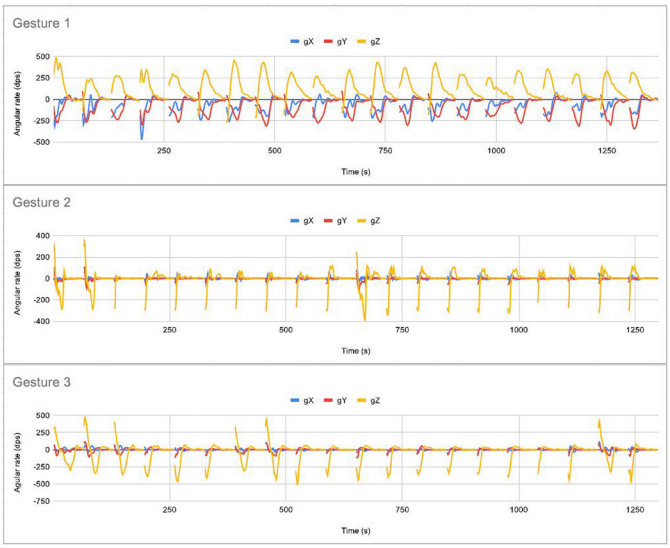
Algorytm

Zbieranie danych

Uczenie algorytmu musi zacząć się od zebrania danych i opracowania tzw. zbioru uczącego. Wykorzystany zostanie sam sensor, z wgranym specjalnym oprogramowaniem, które pokazano na **listingu 1**. Program posiada prekonfigurowany próg wartości żyroskopu. Po jego przekroczeniu, Arduino zaczyna przesyłać pomiary do komputera, celem rejestracji. Szkic zapala diodę LED na inny kolor co 20 ruchów. W ten sposób wiadomo, kiedy zmieniać wykonywany gest (potrzebne jest wiele powtórzeń tego samego gestu, aby opracować zbiór uczący).

W czasie akwizycji nagrywano następujące gesty:

- ramię skierowane w lewo (typowy gest dla rowerzystów wskazujący skręt w lewo),
- naciśnięcie hamulca (ruch palców, sięgających do rączki hamulca na kierownicy),
- pochylenie dłoni do tyłu,
- pochylenie dłoni do przodu,
- pochylenie dłoni w lewo,
- pochylenie dłoni w prawo.



Rysunek 5. Przykładowe dane zebrane dla trzech różnych gestów z żyroskopu

Nic oczywiście nie stoi na przeszkodzie, aby uzupełnić powyższy zestaw o kolejne gesty.

Po zarejestrowaniu wszystkich gestów, ostatnią rzeczą do zrobienia jest skopiowanie danych wyświetlanych na monitorze portu szeregowego i zapisanie ich w plikach z rozszerzeniem .csv (wartości rozdzielone przecinkami). Przykładowy wykres odczytów z żyroskopu dla 20 powtórzeń danego gestu, pokazano na **rysunku 5**.

Trening

Do szkolenia algorytmu wykorzystany został framework tinyML. Opis szkolenia znacznie wykracza poza ramy tego artykułu. Cały kurs znaleźć można tutaj: ***https://bit.ly/3dvwfAJ***. Autor zmodyfikował tylko kilka rzeczy, względem materiałów z poradnika.

W „Graph Data (optional)", zmieniona została nazwa pliku na jeden z przygotowanych wcześniej:

```
filename = "Arm_left.csv"
```

Następnie należy zmodyfikować poniższy wiersz, aby wykreślić tylko dane z żyroskopu:

```
#index = range(1, len (df ['aX']) + 1)
index = range(1, len (df ['gX']) + 1)
```

W kolejnym kroku należy ująć w komentarz następujące wiersze, aby wyglądały następująco (ponownie po to, by nie używać danych z akcelerometru):

```
#plt.plot(index, df['aX'], 'g.', label='x',
linestyle='solid', marker=',')
#plt.plot(index, df['aY'], 'b.', label='y',
linestyle='solid', marker=',')
#plt.plot(index, df['aZ'], 'r.', label='z',
linestyle='solid', marker=',')
#plt.title("Acceleration")
#plt.xlabel("Sample #")
#plt.ylabel("Acceleration (G)")
#plt.legend()
#plt.show()
```

W „Parse and prepare the data" dodać należy wszystkie nazwy, których użyto do opisu zebranych danych, zamiast przykładowych wpisów:

```
#GESTURES = ["punch", "flex",]
GESTURES = ["Arm_left", "Brake", "Hand_back-
tilt", "Hand_front-tilt", "Hand_left-tilt",
"Hand_right-tilt"]
```

Finalnie, zmienić należy ilość próbek na gest, jeśli taką samą zmianę dokonano również w szkicu Arduino:

```
#SAMPLES_PER_GESTURE = 119
SAMPLES_PER_GESTURE = 64
```

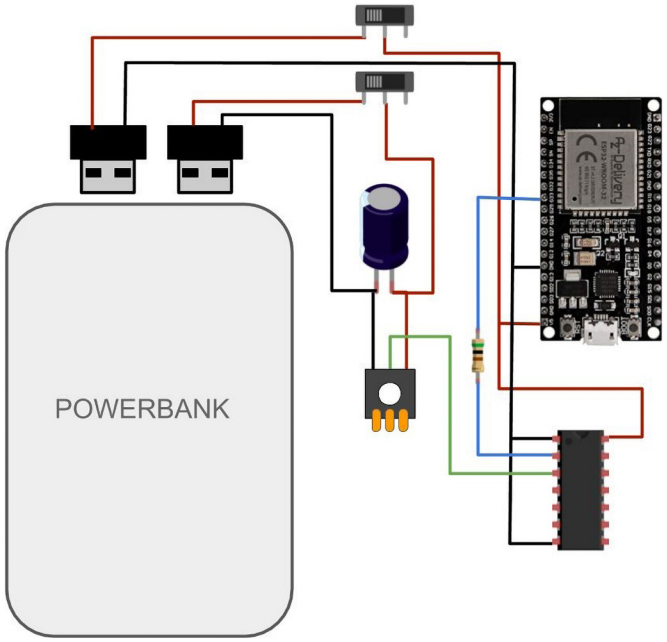
Ostatnią rzeczą do zmiany jest ujęcie w komentarz przyspieszenia w tensorze wejściowym algorytmu do treningu systemu:

```
tensor += [
        #(df['aX'][index] + 4) / 8,
        #(df['aY'][index] + 4) / 8,
        #(df['aZ'][index] + 4) / 8,
        (df['gX'][index] + 2000) / 4000,
        (df['gY'][index] + 2000) / 4000,
        (df['gZ'][index] + 2000) / 4000
]
```

Po wprowadzeniu zmian i uruchomieniu całego skryptu, można pobrać gotowy, wyszkolony model, który będzie działał jako klasyfikator wykonywanych gestów. Jeśli zajrzymy do pobranego pliku nagłówkowego (*model.h*), zobaczymy macierz liczb. Teraz pozostaje tylko zaimplementować model w szkicu Arduino, odpowiedzialnym za sterowanie systemem.

Implementacja Arduino

Ostateczny program kontrolera inteligentnej rękawiczki, to połączenie dwóch innych programów. Pierwszym jest znajdujący się w bibliotece



Rysunek 6. Schemat panelu LED

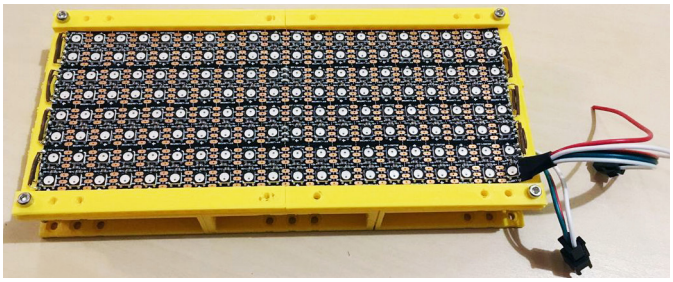
ArduinoBLE przykład, opisany jako LED. Drugi to klasyfikator IMU_Classifier, pochodzacy z przykładów wykorzystania Tensorflow Lite, które obejrzeć można w repozytorium: ***https://bit.ly/3fQIAkC***. Dokładny opis tych programów wykracza poza ramy tego artykułu, ale warto się z nimi zapoznać, czekając aż Arduino IDE skompiluje i załaduje szkic (**listing 2**) do pamięci mikrokontrolera.

Po załadowaniu skompilowanego programu do modułu w rękawiczce, można przejść do konstrukcji części wykonawczej systemu – panelu LED, do zamontowania na plecach rowerzysty.

Panel LED

Elektronika

Autor napotkał problemy podczas przesyłania szkicu – z biblioteki ArduinoBLE do płytki Arduino Nano 33 – obsługującego macierz LED. Zdecydował się zatem na użycie modułu z układem ESP32. Płytka została podłączona jak na **rysunku 6**. Ponieważ zarówno Arduino Nano 33 BLE SENSE, jak i ESP32 komunikują się z poziomem logicznym 3,3 V, do układu dodany został translator poziomów 3 V do 5 V (74AHCT125), aby wysterować diody LED RGB w macierzy. Dodano także kondensator 1000 µF do linii zasilania diod LED. Cały układ zasilany jest z powerbanku USB. Matryca LED (nie pokazana na schemacie) i moduł z mikrokontrolerem zasilane są z osobnych wtyczek USB w powerbanku, z uwagi na wysoki pobór prądu przez macierz LED.



Fotografia 7. Obudowa panelu LED RGB wraz z zainstalowanymi paskami LED-owymi

Obudowa

Obudowa panelu LED jest modułowa, można dopasować jej wielkość do własnego układu diod LED. Dzięki temu można wydrukować ją nawet na małej drukarce 3D. Pliki, potrzebne do druku, są do pobrania ze strony z projektem na portalu ***http://instructables.com***. Gotowa, zmontowana obudowa, wraz z diodami RGB, pokazana jest na **fotografii 7**.

Firmware panelu LED

Podobnie jak w przypadku innych programów w urządzeniu, także ten powstał na podstawie innych, dostępnych skryptów. Autor wzorował się przykładem BLE_Write z biblioteki Arduino dla BLE ESP32 oraz „MatrixGFXDemo64" z biblioteki „FastLED NeoMatrix". Finalny szkic, załadowany do modułu ESP32 pokazano na **listingu 3**.

Podsumowanie

Nadszedł czas, aby przetestować urządzenie. Za każdym razem gdy rozpoznawany jest gest, do panelu LED wysyłany jest sygnał i wyświetlany jest wzór. Dodatkowo, na rękawiczce zapala się dioda LED, informująca o poprawnym rozpoznaniu gestu. Jeśli system nie rozpoznaje poprawnie ruchów dłonią, konieczne może być ponowne wygenerowanie modelu z nowym zbiorem uczącym, szczególnie dla problematycznych ruchów. Jak przyznaje autor, zbiór danych uczących był zbyt mały, by zapewnić w pełni niezawodny model. Nagranie większej ilości ruchów dałoby lepszy model i mniej błędów w rozpoznawaniu gestów.

Projekt jest dobrym wstępem do świata systemów uczenia maszynowego. Dzięki wykorzystaniu otwartych narzędzi i niedrogiej płytki Arduino, każdy może stosować AI w swoim projekcie, nie tylko do klasyfikowania gestów, ale i dowolnych innych sygnałów. Tylko wyobraźnia jest tutaj granicą.

Nikodem Czechowski, EP

Źródło: ***https://bit.ly/3hR3WjH***