

# PAJĄK MECHATRONICZNY

## Spis treści:

1. Opis projektu
2. Wykaz elementów
3. Schemat połączeń elektrycznych
4. Instrukcja obsługi
5. Zdjęcia
  - Modele 3D
  - Symulacji w Python
  - Rozpoznawania obiektów
  - Transmisji wideo online
  - Robot
6. Komunikacja w sieci komputerowej, protokół SSH
7. Magistrala I<sup>2</sup>C
8. Obliczenia oraz opis kinematyki robota
9. Wykrywanie obiektów
10. Zastosowanie w przemyśle
11. Program Arduino
12. Program symulacji w języku Python
13. Program komunikacja Arduino z Raspberry

# 1.Opis projektu

Projekt ten świetnie wykorzystuje połączenie elementów mechanicznych, współpracujących ze sobą komponentów elektronicznych oraz oprogramowania, w tym sensie czyniąc pozycję wartą dalszej lektury.

Przedstawiony robot kroczący wykorzystuje do realizacji ruchu operacje macierzowe dzięki którym może precyzyjnie określić pozycję i orientację każdego z członów robota w przestrzeni trójwymiarowej. Takie rozwiązanie realizacji ruchu jest aktualnie wykorzystywane wśród wielu robotów przemysłowych.

Składający się z dwunastu serwomechanizmów sterowanych popularnym modułem Arduino Mega bazującym na mikrokontrolerze AVR ATmega2560, który z kolei połączony przez magistrale I<sup>2</sup>C z minikomputerem Raspberry Pi Zero WH z zamontowaną kamerą na pokładzie umożliwia nam zarówno sterowanie robotem jak i dostęp do obrazu przed robotem, w czasie rzeczywistym z lekkim opóźnieniem z dowolnego miejsca w zasięgu naszego routera.

Dodatkowo nasz robot potrafi z dostarczanego przez kamerę obrazu wykrywać obiekty i je rozpoznawać. Metoda wykrywania obiektów należy do metod opartych na głębokim uczeniu się. Wykorzystuje do tego celu otwarto źródłową bibliotekę programistyczną napisaną przez Google Brain Team, nazwaną TensorFlow, która to właśnie wykorzystuje sztuczne sieci neuronowe, które mogą wydobywać z danych zawartych w dostarczonym obrazie wzory oraz modele rozpatrywanego przedmiotu.

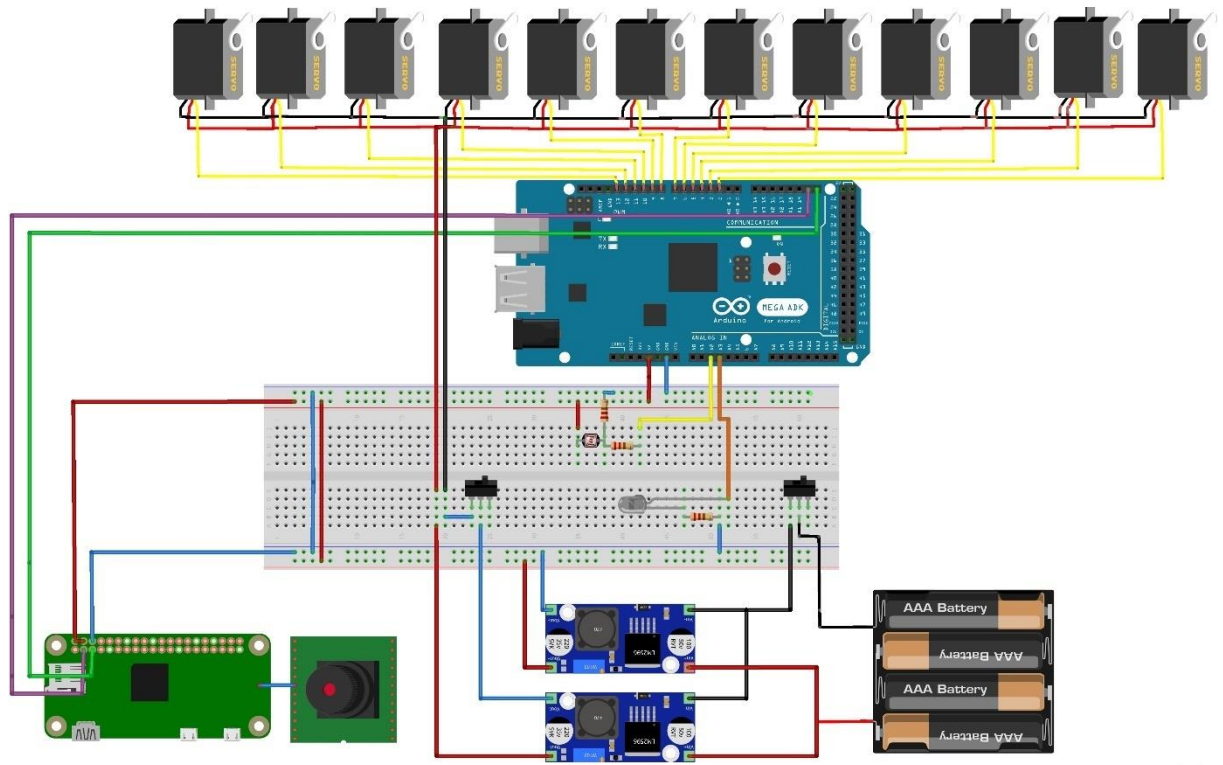
Konstrukcja robota została w całości zaprojektowana w programie Inventor Autodesk i wykonana w technologii druku 3D. Materiał jaki został użyty do tego celu to ABS, znacznie bardziej odporny na wyższą temperaturę, przez co też łatwiejszy w obróbce ręcznej. Jedną z zalet tego materiału jest także dość duża odporność na pęknięcia i stosunkowo duża twardość przez co wybór padł na ten rodzaj materiału.

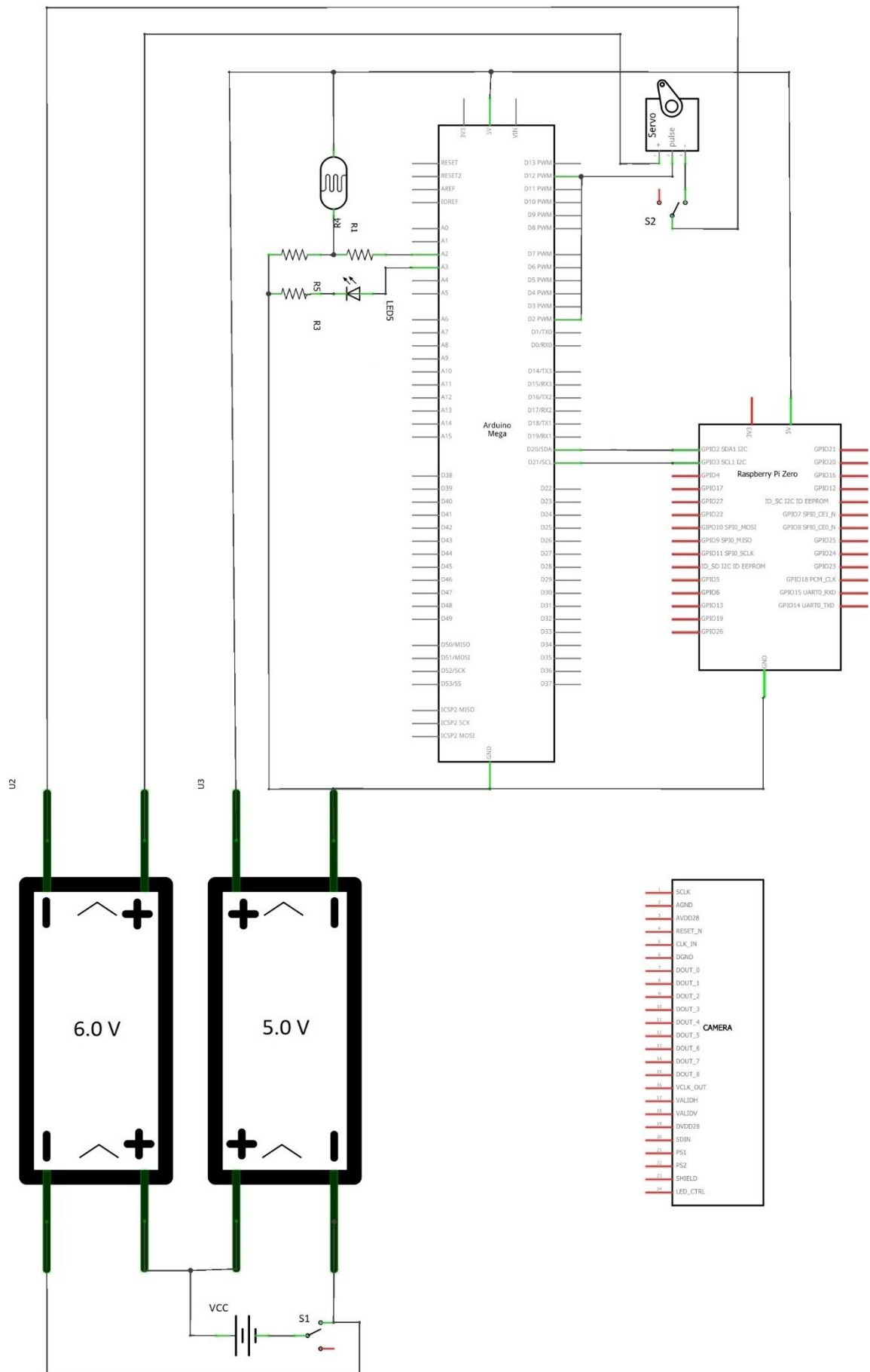
Dodatkiem do kamery jest płytka na której zamieszczone jest sześć diod świecących, dzięki czemu kamera z w przypadku braku światła dalej może wysyłać obraz. Dzięki fotorezystorowi zamieszczonemu nad kamerką można określić graniczną wartość napięcia, które przepuszcza nasz element światłoczuły wprost proporcjonalnie do natężenia oświetlenia w otoczeniu przy której diody zaczynają świecić.

## 2. Wykaz elementów


- Raspberry Pi Zero WH - model tego minikomputera posiada procesor Broadcom BCM2835 1 GHz i 512 MB pamięci RAM dodatkowo został wyposażony w moduł Wi-Fi, Bluetooth, czterdzieści pinów ogólnego przeznaczenia oraz złącze kamery CSI. System operacyjny jaki został użyty to Linux, a wchodząc w szczegóły jedna z dystrybucji Linuxa oparta na Debianie jaką jest Raspbianstretch.
- Arduino Mega 2560 – Jedna z wersji popularnego Arduino płytka posiada mikrokontroler ATmega2560, do wykorzystania 256 kB pamięci Flash, 8 kB RAM, 54 cyfrowych wejść/wyjść z czego 15 można wykorzystać jako kanały PWM, które zostały użyte do sterowania serwomechanizmami, mamy też 16 wejść analogowych.
- Kamera, którą robot posiada to - Raspberry Pi Camera HD v2, która wyposażona jest w matrycę o rozdzielczości 8 Mpx, wspiera tryb nagrywania HD 1080p, 720p oraz 640 x 480p, potrafi wykonywać zdjęcia w jakości 3280 x 2464 px. Raspberry posiada sprzętowe wsparcie dla obsługi tej kamery, dzięki temu urządzenie nie zużywa mocy obliczeniowej procesora. Nie wymaga instalacji sterowników dla systemu Raspbian.
- Serwomechanizmy użyte w projekcie to konkretny model -Serwo PowerHD HD-1800A ich napięcie zasilania jest w zakresie od 4,8 V do 6,0 V. Zakres ruchu: od 0 ° do 180 °. Moment siły oraz prędkość zależy od parametrów zasilania w robocie zasilanie dla serwomechanizmów ustawione jest na 6,0V z tego względu aby uzyskać większy moment, co też daje nammoment siły: 1,3 kg\*cm (0,12 Nm) oraz prędkość: 0,08 s/60°. Robot posiada ich w sumie dwanaście, a na każdą nogę przypada ich trzy.
- Zasilanie. Za źródło zasilania posłużyły mi ogniwa litowo-polimerowe. Połączenie ogni w sposób 2S2P, dzięki temu uzyskuje napięcie około 8,0V i pojemność 3Ah
- Przetwornice napięcia - Arduino a także Raspberry potrzebują do pracy napięcia 5,0V. Serwomechanizmy tak jak już wcześniej wspomniałem pracują przy 6,0V chociaż przy wyższym napięciu nic im się złego nie dzieje to jednak z tego względu, że podczas projektowania mogło dojść do zablokowania którejś z nóg, w wyniku czego przy wyższym napięciu układ sterowniczy takiego serwomechanizmu mógł się uszkodzić, wynikała konieczność użycia dodatkowych dwóch przetwornic step-down, które obniżają napięcie zasilania do poziomu pożądanego.
- Dwa przełączniki, jeden służy do załączenia zasilania na obydwie przetwornice tym samym zasilamy już Arduino i Raspberry, z kolei drugi przepuszcza nam napięcie z przetwornicy 6,0V do zasilania serwomechanizmów, takie rozwiązanie pozwala nam na odłączenie serwomechanizmów w dowolnym momencie bez konieczności wyłączenia Raspberry.
- Fotorezystor 5-10kΩ GL5616

### 3.Schemat połączeń elektrycznych





## 4. Instrukcja obsługi

- a. Aby uruchomić robota musimy przełączyć dwa przełączniki znajdujące się w górnej części korpusu.
- b. Żeby nawiązać połączenie z Raspberry potrzebujemy mieć na swoim komputerze zainstalowany program do komunikacji [PuTTY](#). Putty to standardowy klient SSH na systemy Windows/Linux, służy do terminalowego łączenia się ze zdalnymi komputerami.
- c. Kolejno co musimy zrobić to dowiedzieć się jaki adres IP ma nasz minikomputer, wchodzimy w menu start  i wpisujemy cmd potwierdzamy Enter, kiedy otworzy nam się konsola wpisujemy „arp -a” wtedy wyświetlają nam się adresy IP wszystkich podłączonych urządzeń z naszym routerem. Raspberry rozpoznamy po adresie Mac, który będzie zaczynać się od ciągu „b8-27-eb”.
- d. Kiedy wiemy już jaki adres IP ma Raspberry włączamy program Putty. Domyślnie mamy ustawioną pierwszą z kategorii oraz zaznaczone SSH. W linijce pod nazwą „Host Name” wpisujemy adres IP naszego Raspberry i potwierdzamy Enter.
- e. Gdy Raspberry nawiąże komunikację z naszym komputerem musimy się do niego zalogować. Login „pi” hasło „8054”, po chwili otworzy nam się konsola.
- f. Aby móc sterować robotem wpisujemy w konsoli „python ardu.py” uruchomimy program, który służy do komunikacji między Arduino i Raspberry, który również jest dołączony do dokumentacji.
- g. Uruchomionym programem możemy sterować ruchem robota wpisując, co chcemy żeby robot zrobił i zatwierdzamy Enter.
  - „w” / „s” – idź do przodu / idź do tyłu
  - „a” / „d” – idź w lewo / idź w prawo
  - „z” / „x” – siedź / wstań
  - „c” / „n” – pomachaj
  - „m” / „l” – przód / dół do góry
  - „z” – robot wykonuje przykładową sekwencję ruchów
  - „f” – pozycja początkowa
  - „v” – wyłączamy program
- h. Wpisując w przeglądarce internetowej „IP Raspberry:8081” i potwierdzając Enter ładuje nam się obraz z kamerki robota. Usługa została tak ustawiona, aby uruchamiała się zaraz po starcie systemu. Możemy ją wyłączyć wpisując w konsoli – „sudo service motion stop” aby ponownie uruchomić wpisujemy to samo tylko zamiast „stop” wpisujemy „start” przydatne jest czasem także zrestartowanie usługi więc, żeby to zrobić postępujemy tak samo tylko końcówka „restart”.
- i. Jeśli chcielibyśmy uruchomić rozpoznawanie obiektów musimy zainstalować protokół [VNC](#). Służy on do wirtualnego przekazywania obrazu pulpitu Raspberry. Kolejno

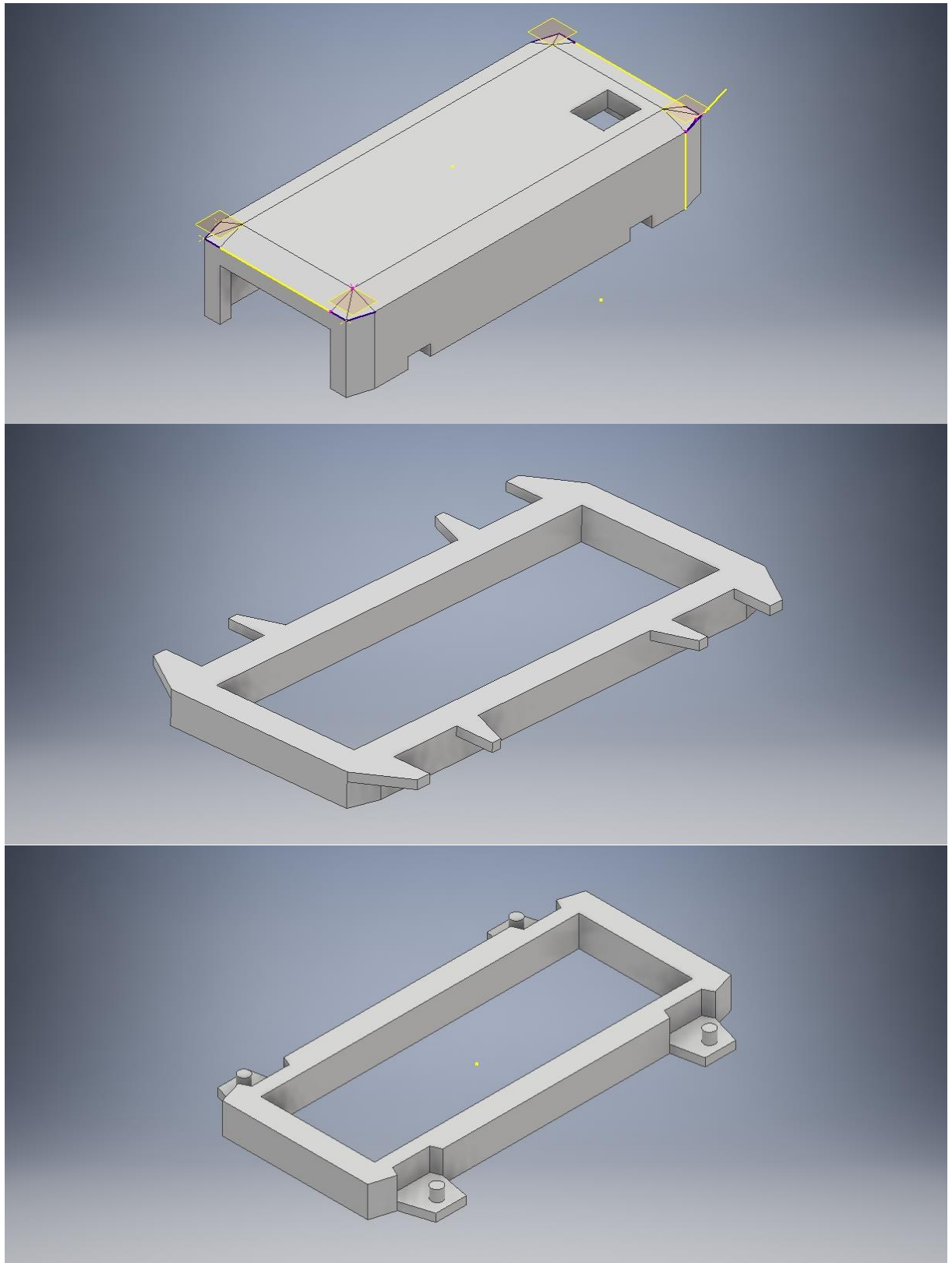


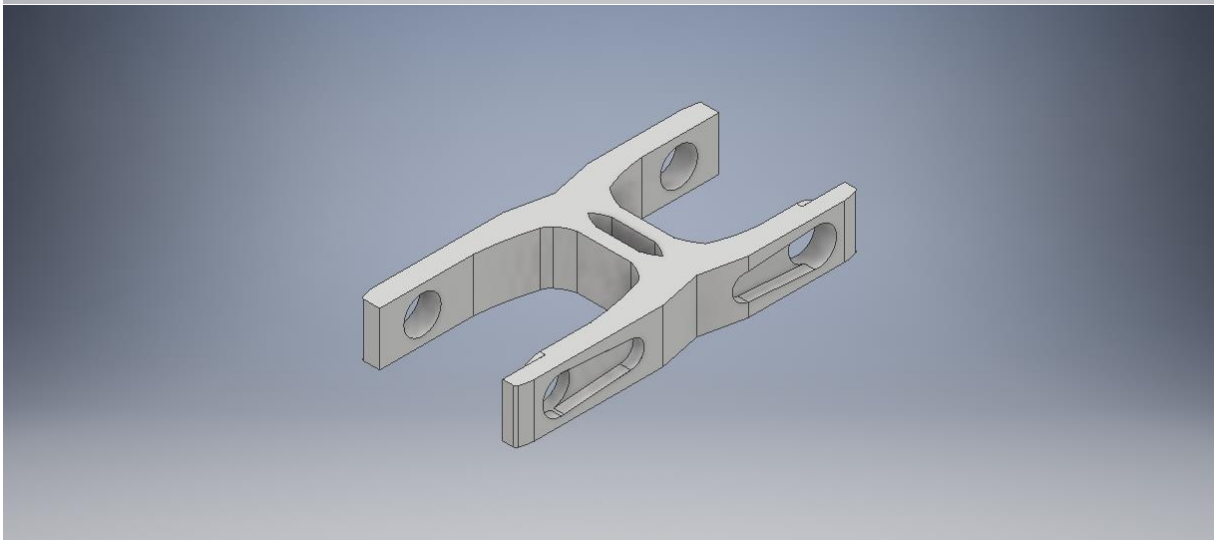
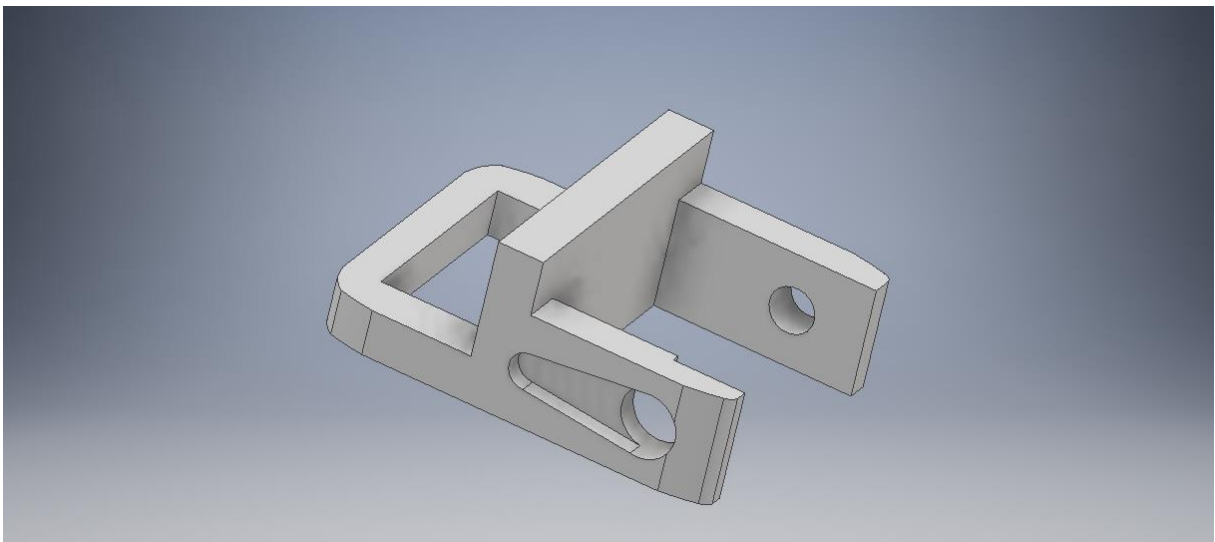
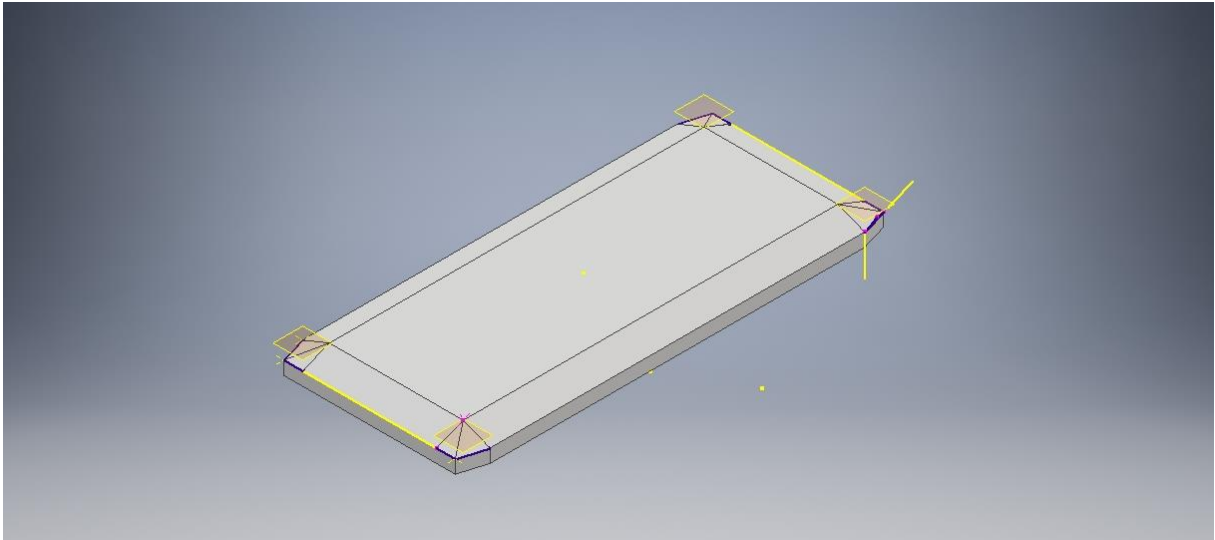
wpisujemy adres IP i logujemy się gdy już otworzy nam się pulpit naszego minikomputera w lewym górnym rogu znajdziemy coś na wzór menu, potwierdzamy, wchodzimy w zakładkę „System tools” i wybieramy „UXterm” w podpunkcie zdjęcia łączy screena.

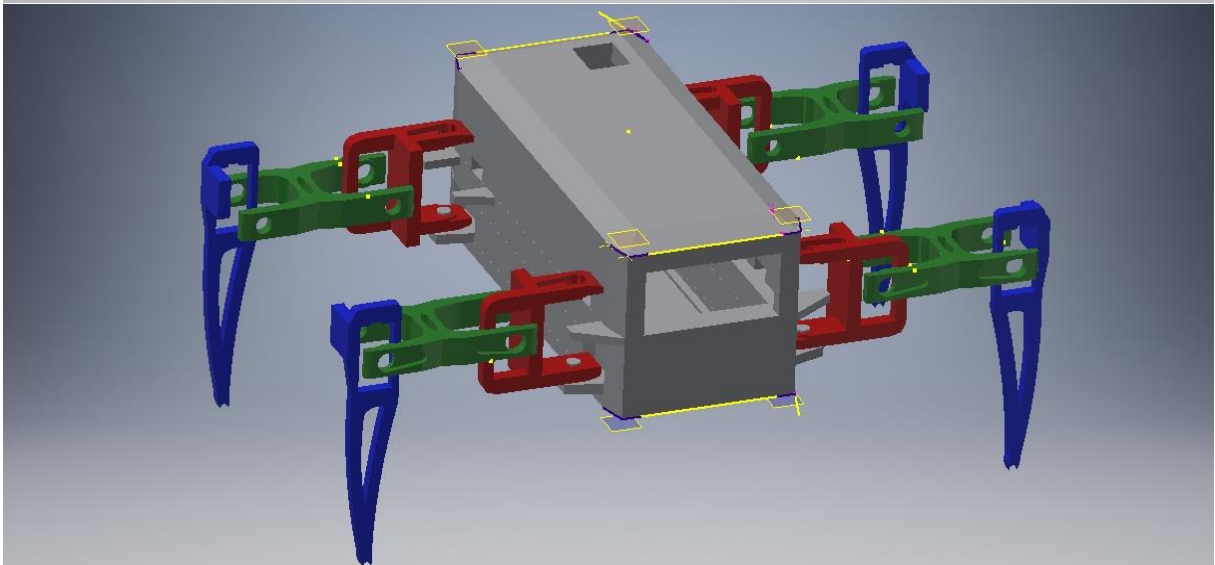
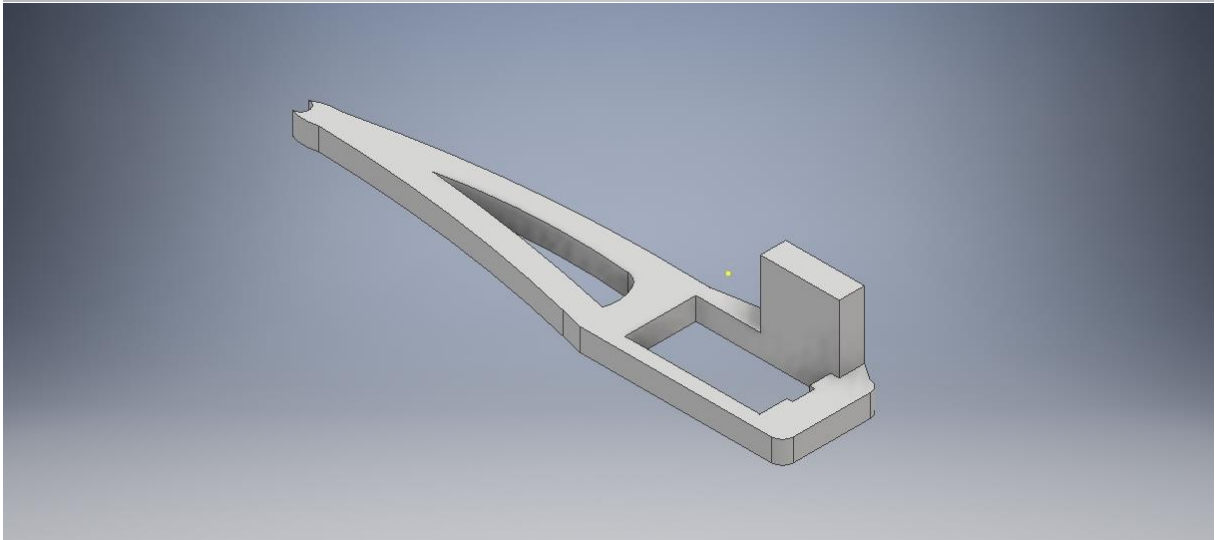
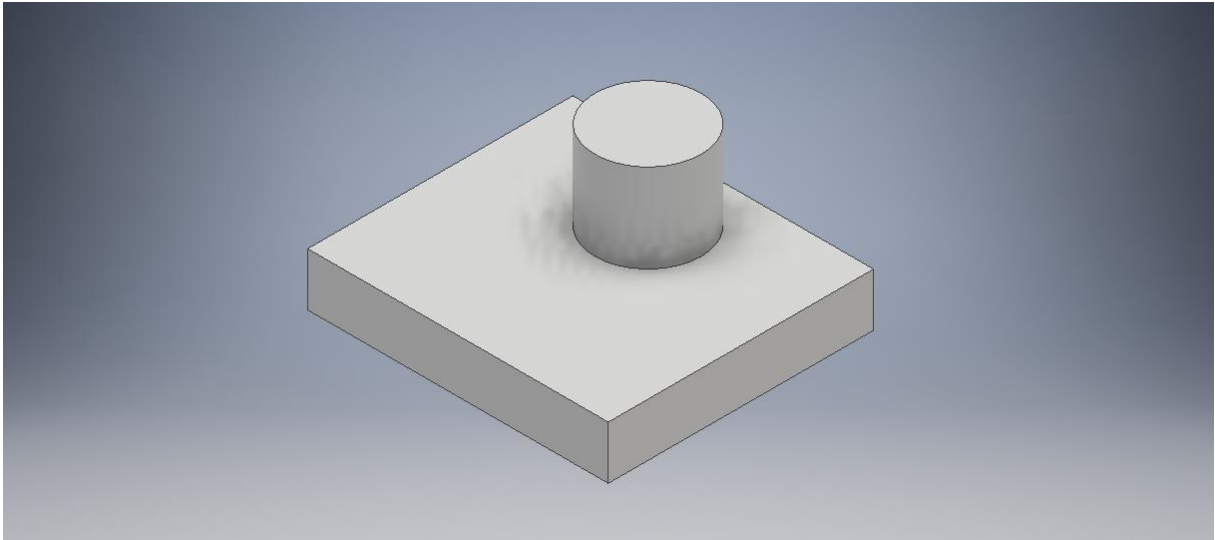
- j. Otworzy nam się terminal. Aby go uruchomić musimy przede wszystkim wyłączyć nasz obraz z kamerki na przeglądarce kamera musi być zwolniona z jakichkolwiek procesów. Tak więc „sudo service motion stop” kolejno wpisujemy dwie komendy – „cd tensorflow1/models/research/object\_detection” wchodzimy tym samym w folder z programem „Object\_detection\_picamera.py”. Jeśli chcemy z tego folderu wyjść wystarczy wpisać „cd”. Gdy jesteśmy już w folderze to prawie identycznie jak programem do sterowania uruchamiamy „python3 Object\_detection\_picamera.py” po chwili otwiera nam się okno z obrazem z przed kamery... tutaj już ciężko jest mówić o sam obraz jest to bardziej to co komputer rozpoznał poprzez swoje algorytmy.
- k. Aby wyłączyć program najeżdżamy na krzyżyk okna programu, następnie klikamy w okno konsoli i wciskamy Ctr+Z.
- l. Raspberry wyłączamy poprzez wpisanie w konsoli „sudoshutdownnow” bądź w menu start na dole znajdziemy „Shutdown”.

## 5. Zdjęcia

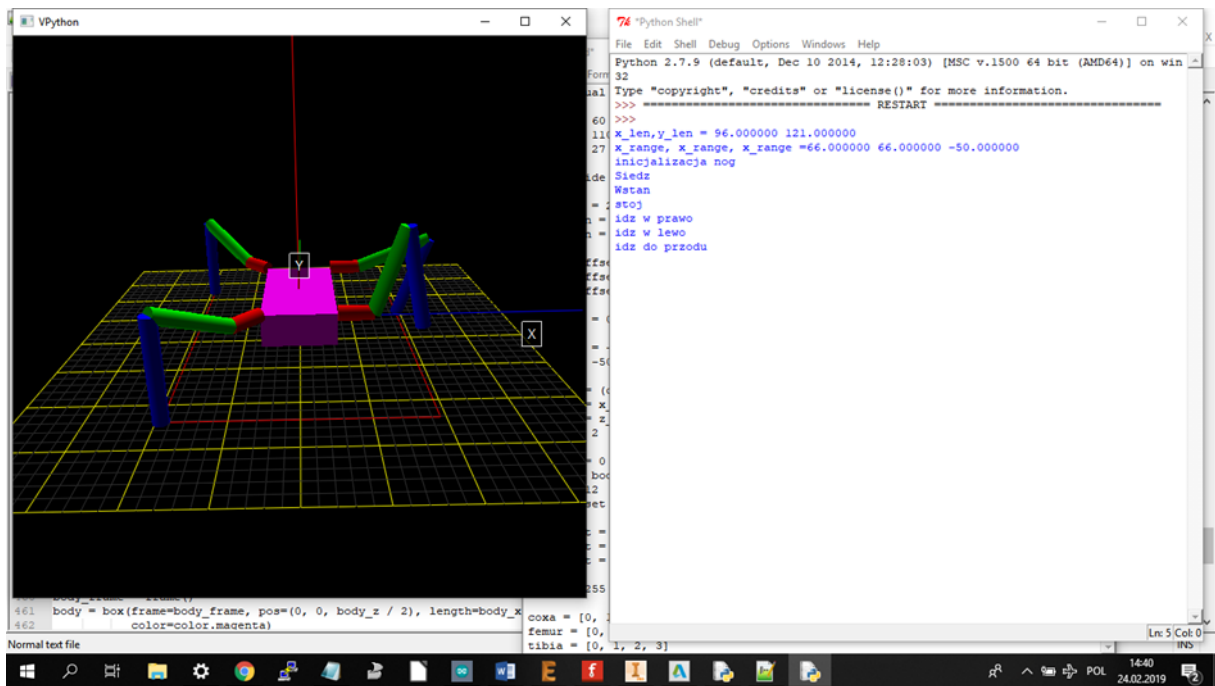
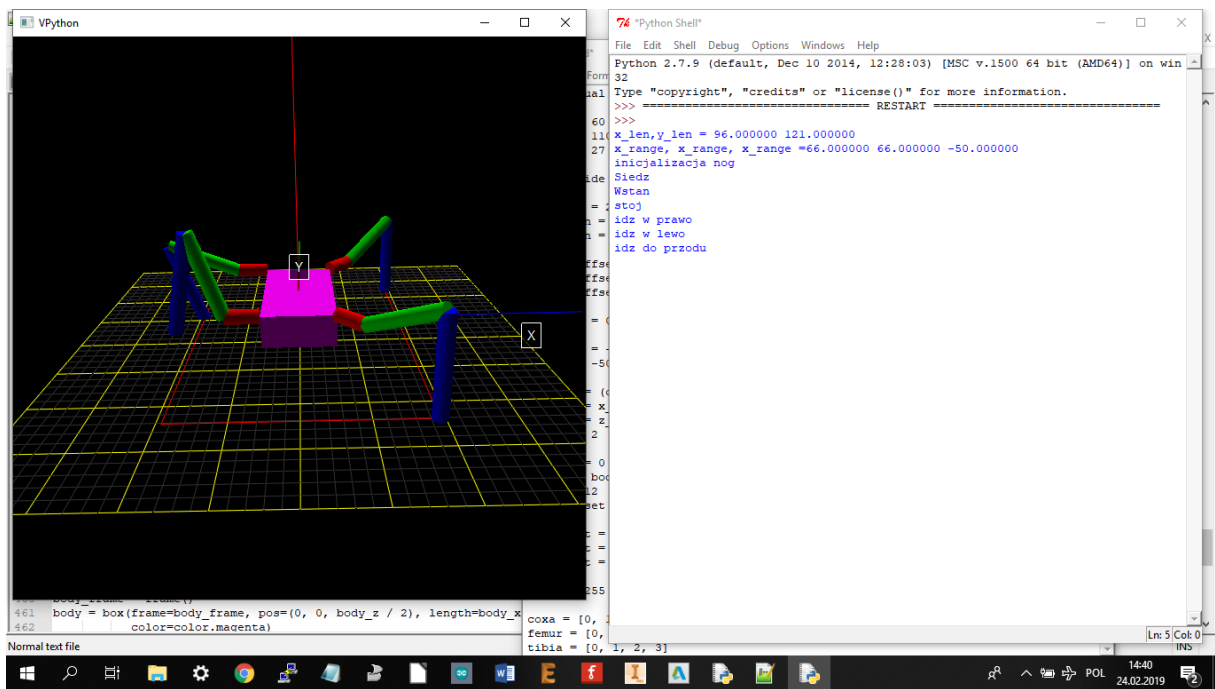
- Modele 3D

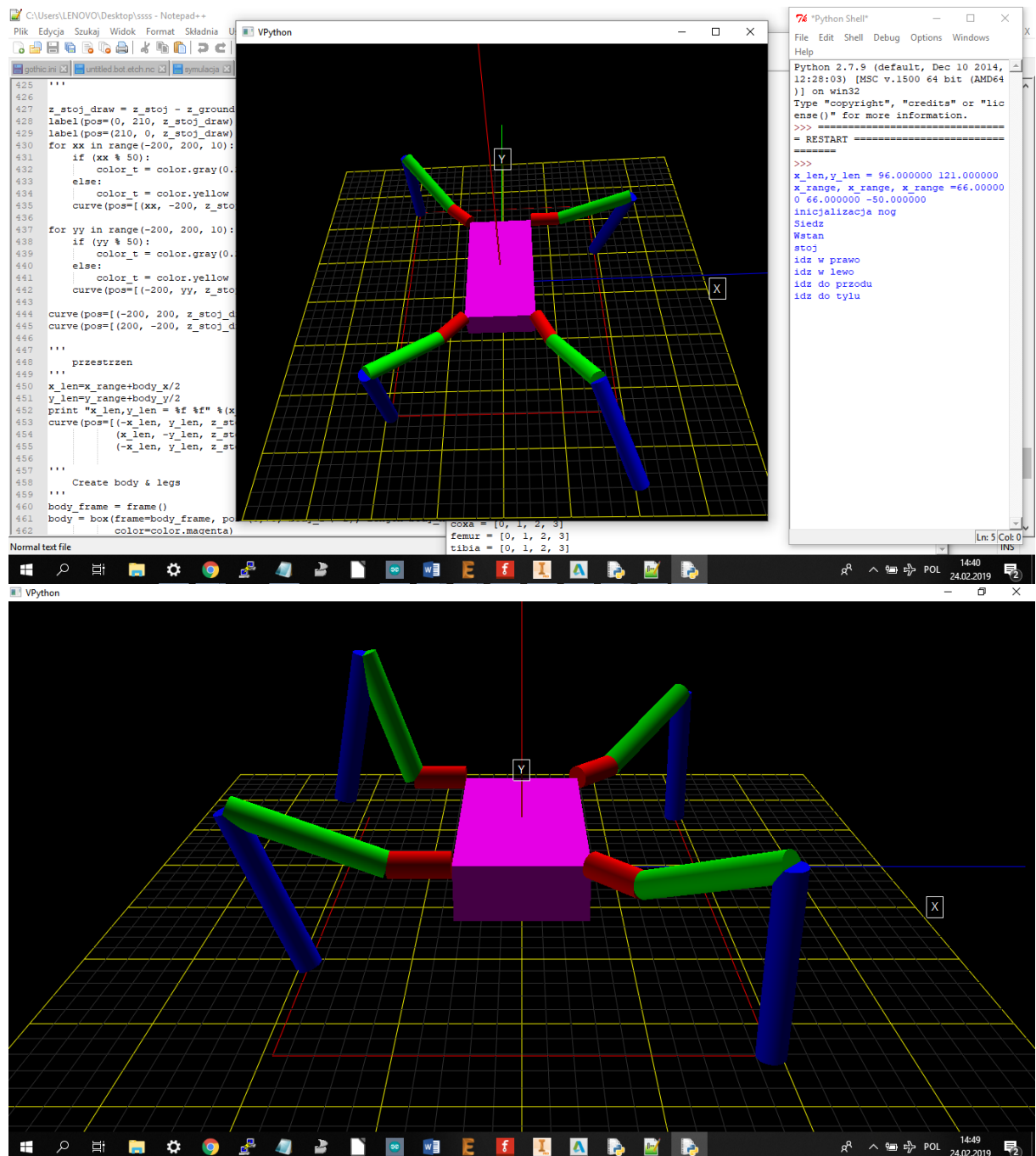






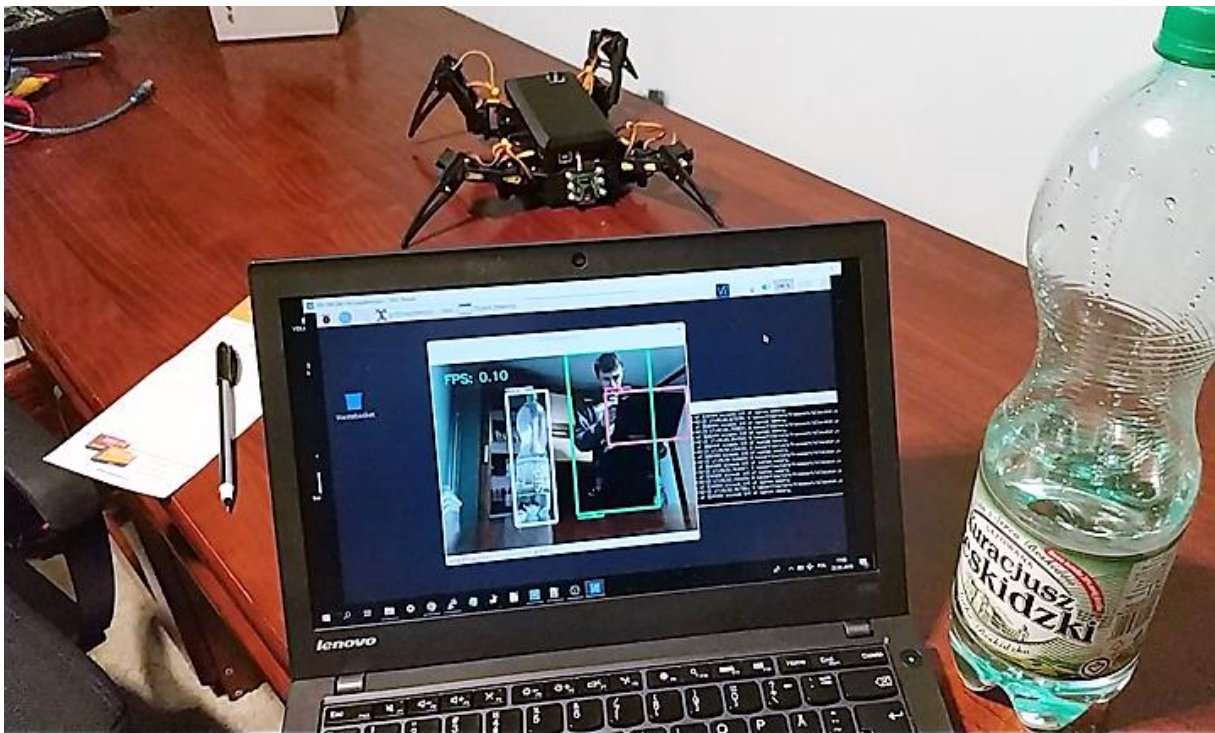
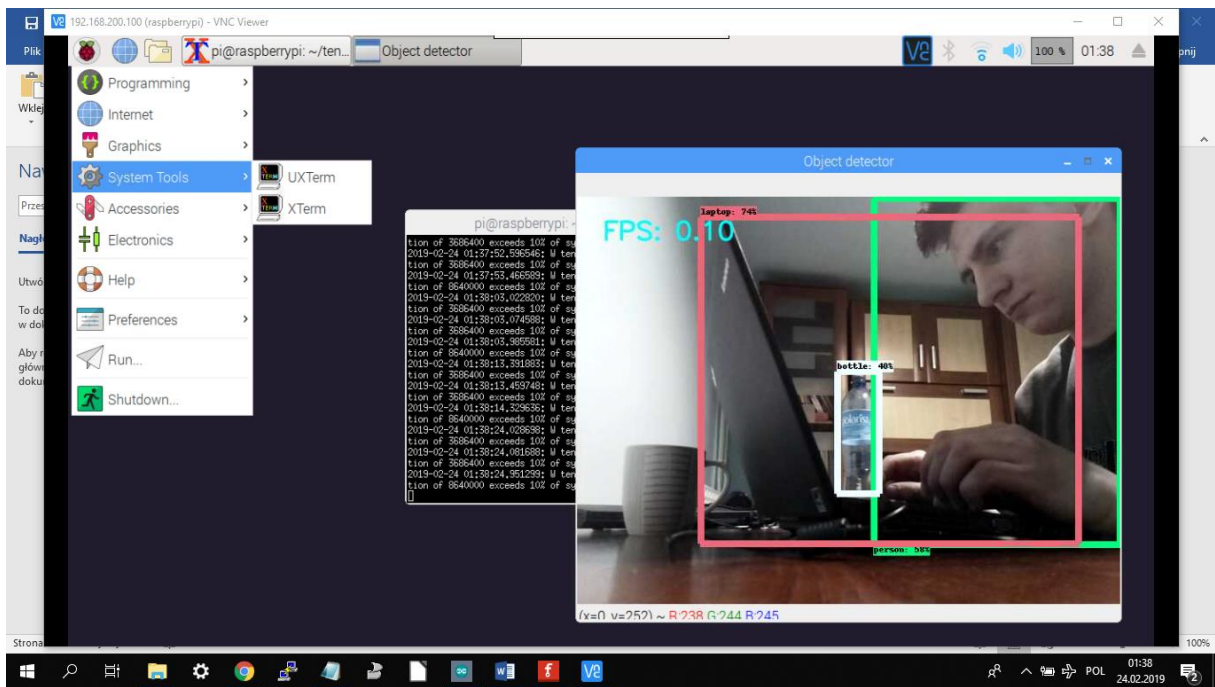
- Symulacja w Python



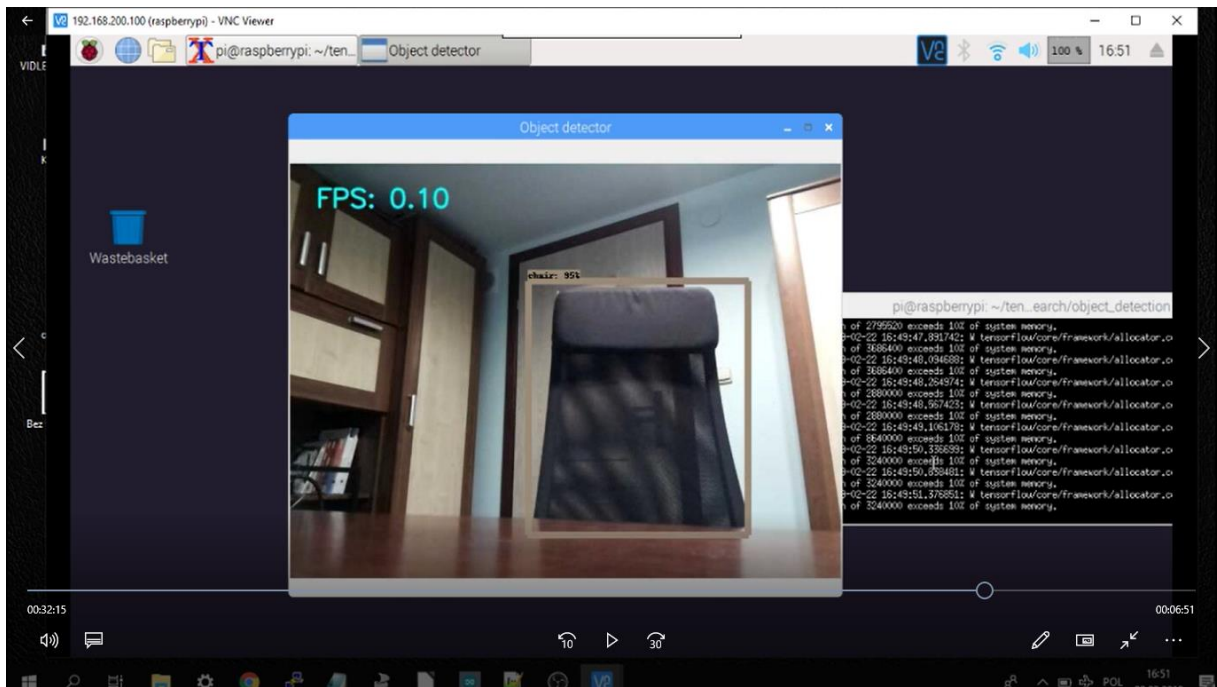
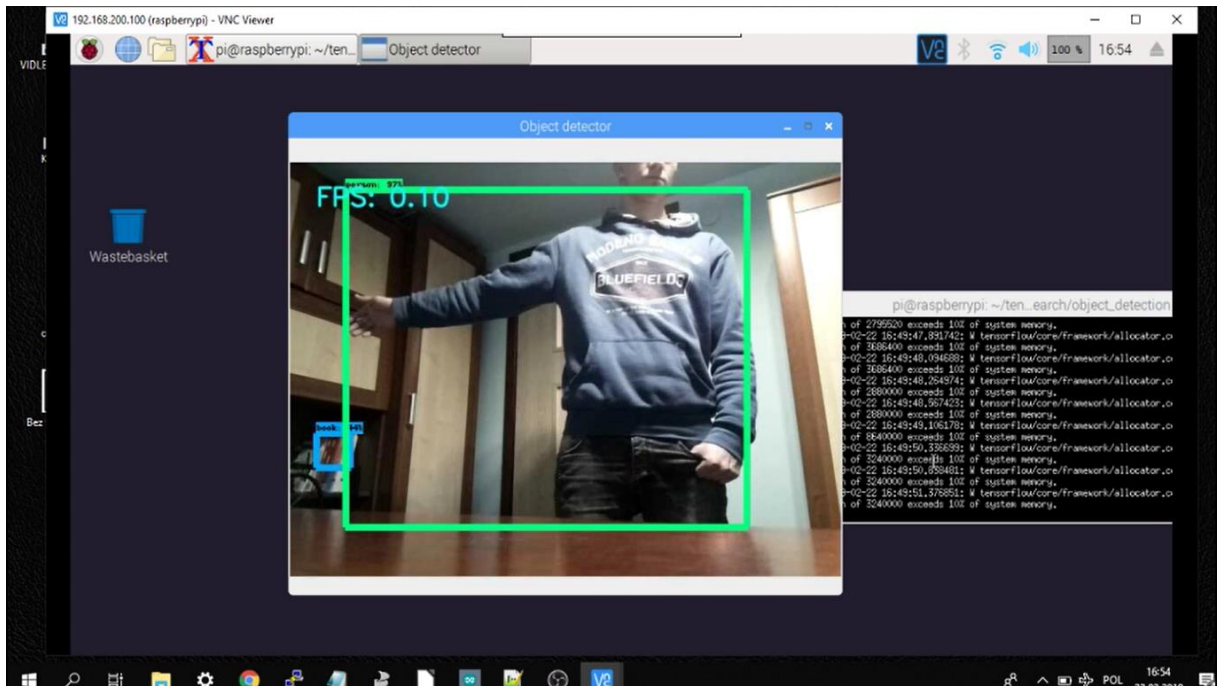




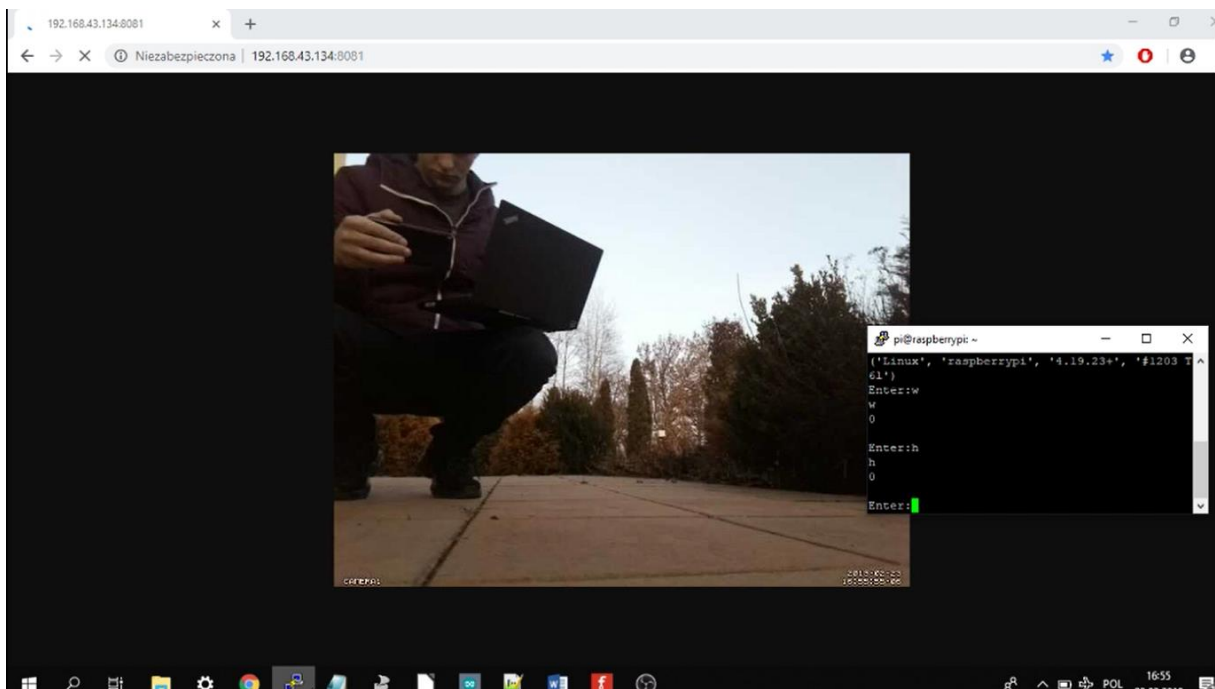
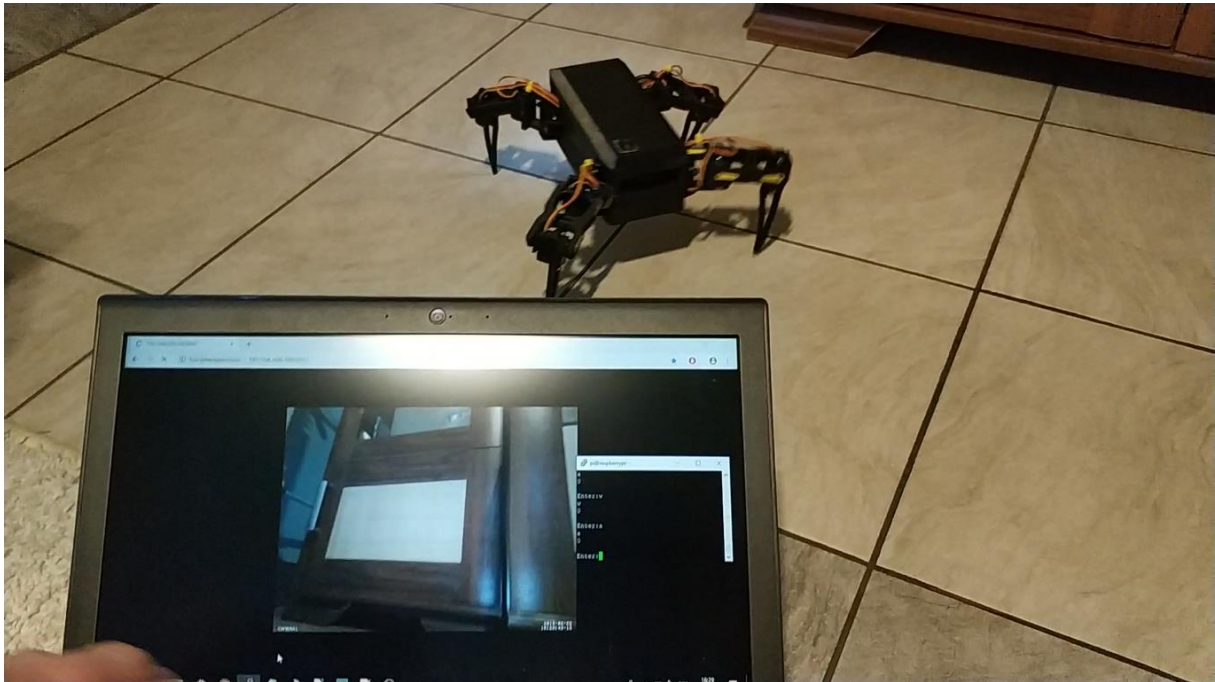
- Rozpoznawanie obiektów

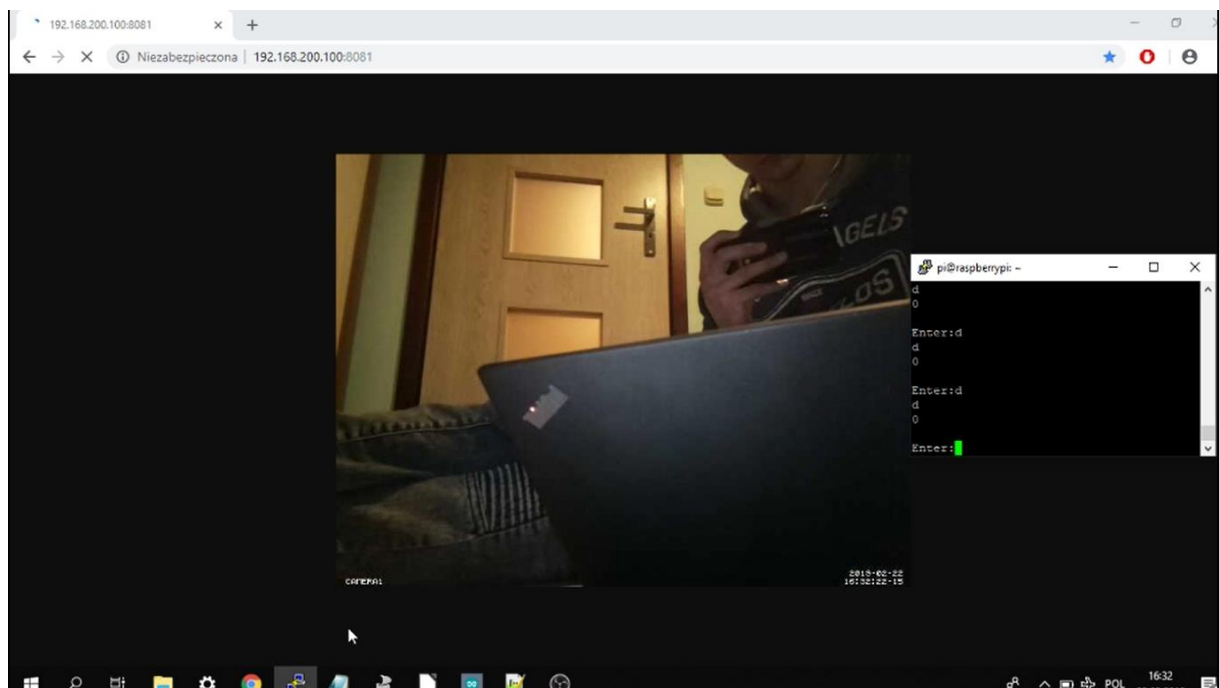
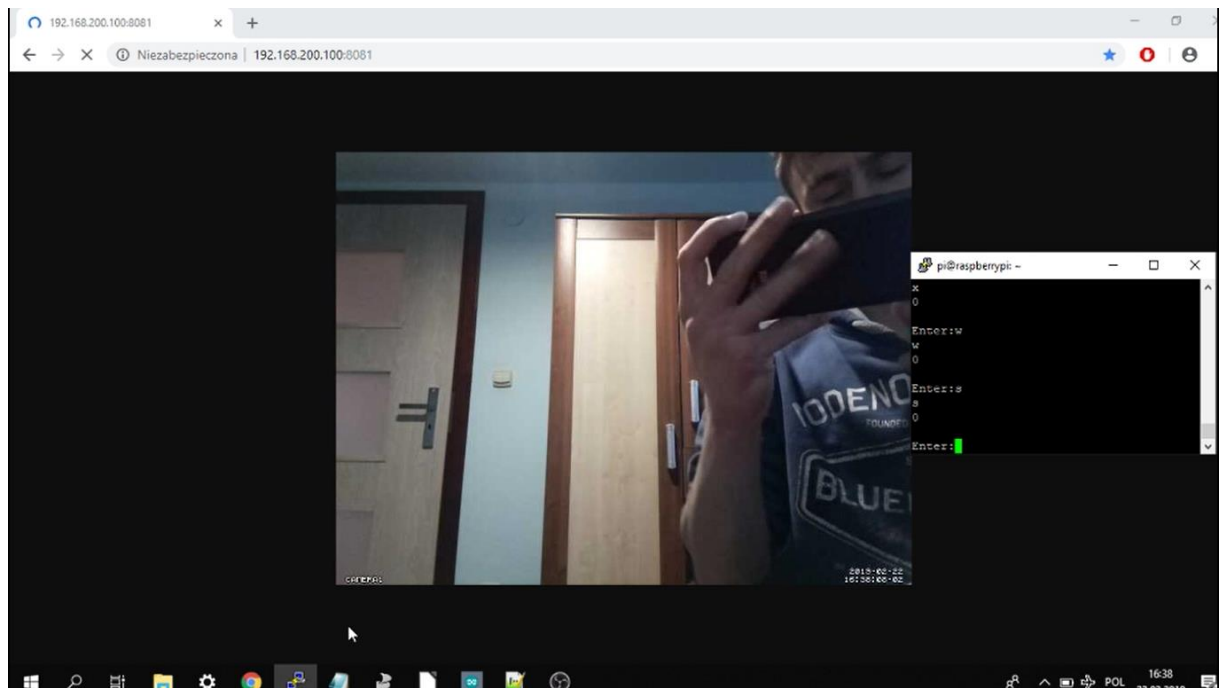






- Transmisja wideo online

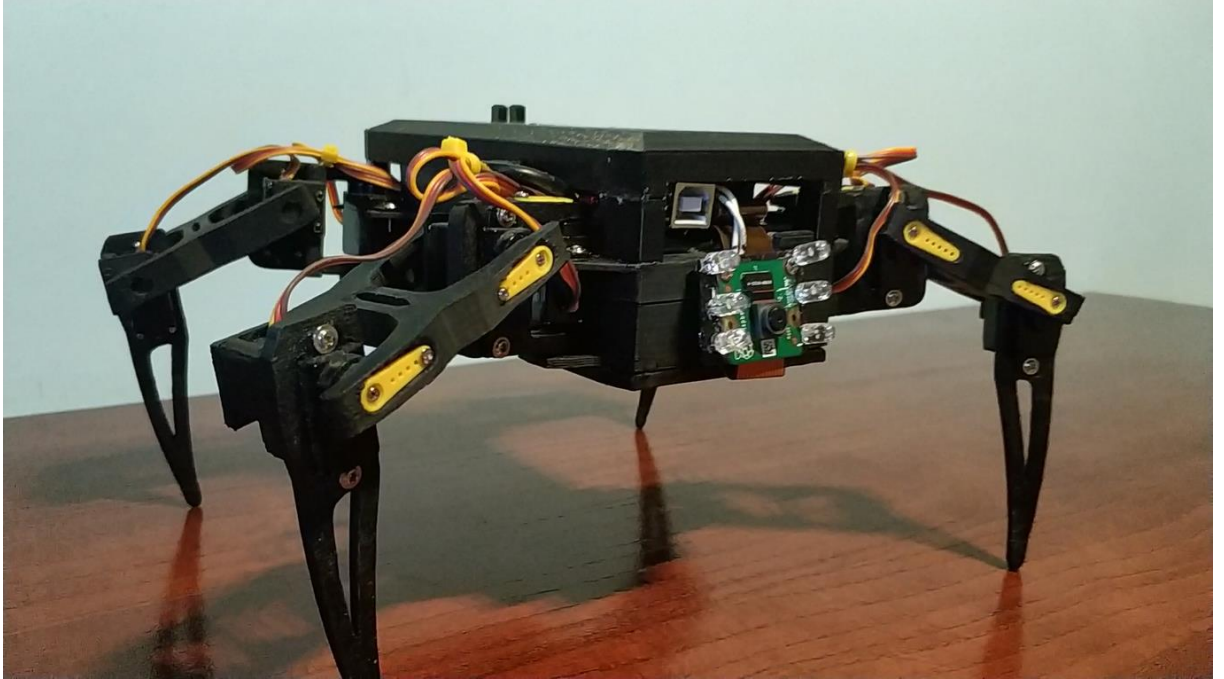




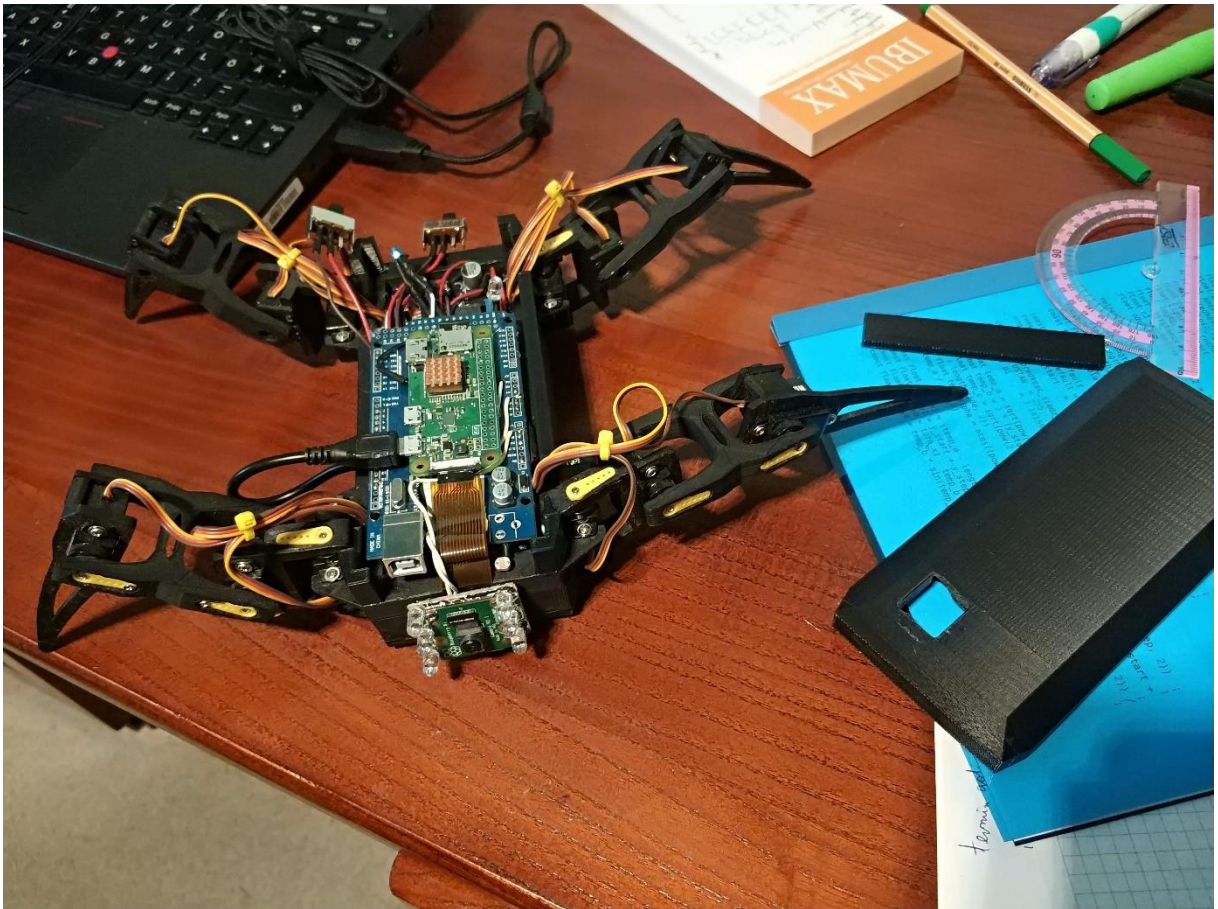
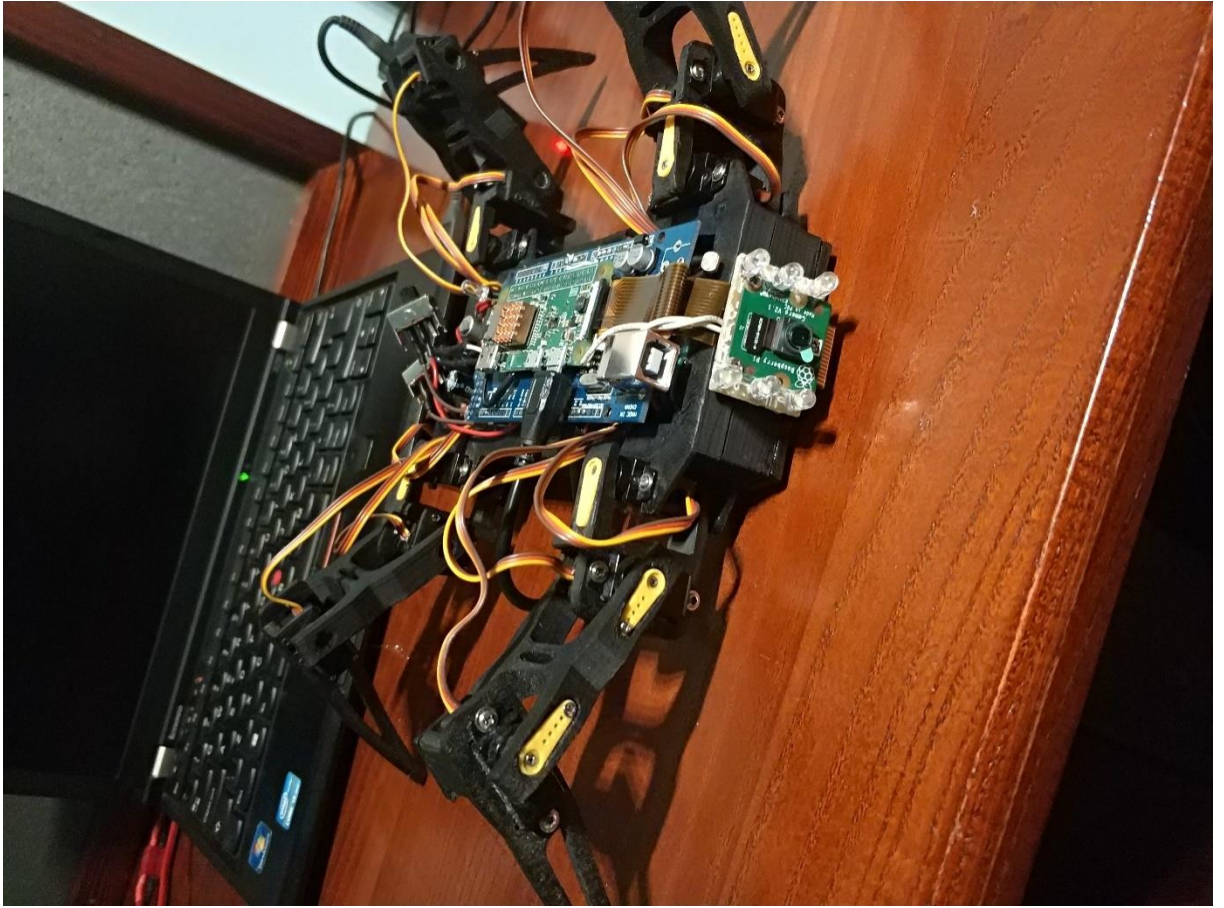
- Robot

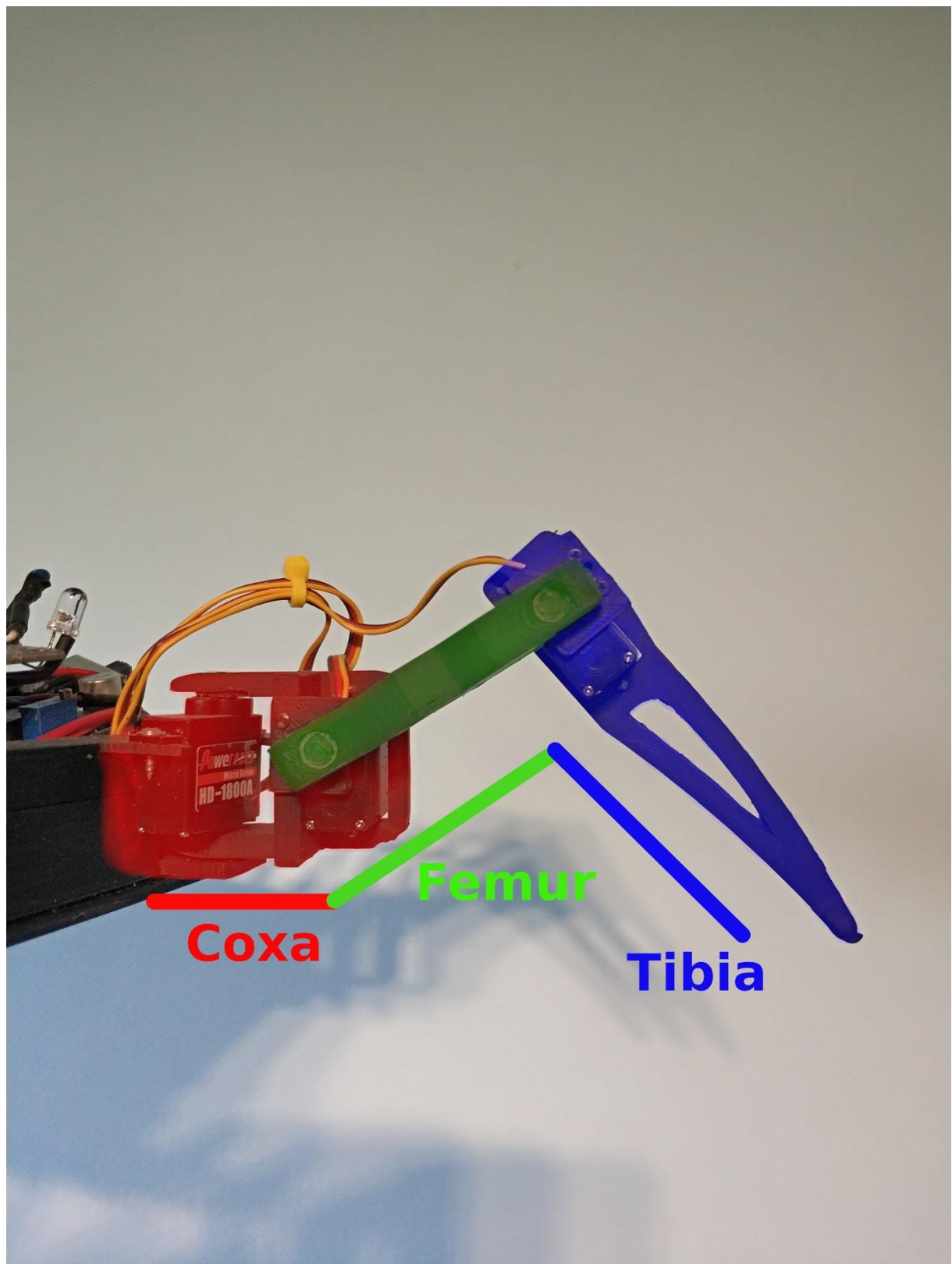












**Coxa**

**Femur**

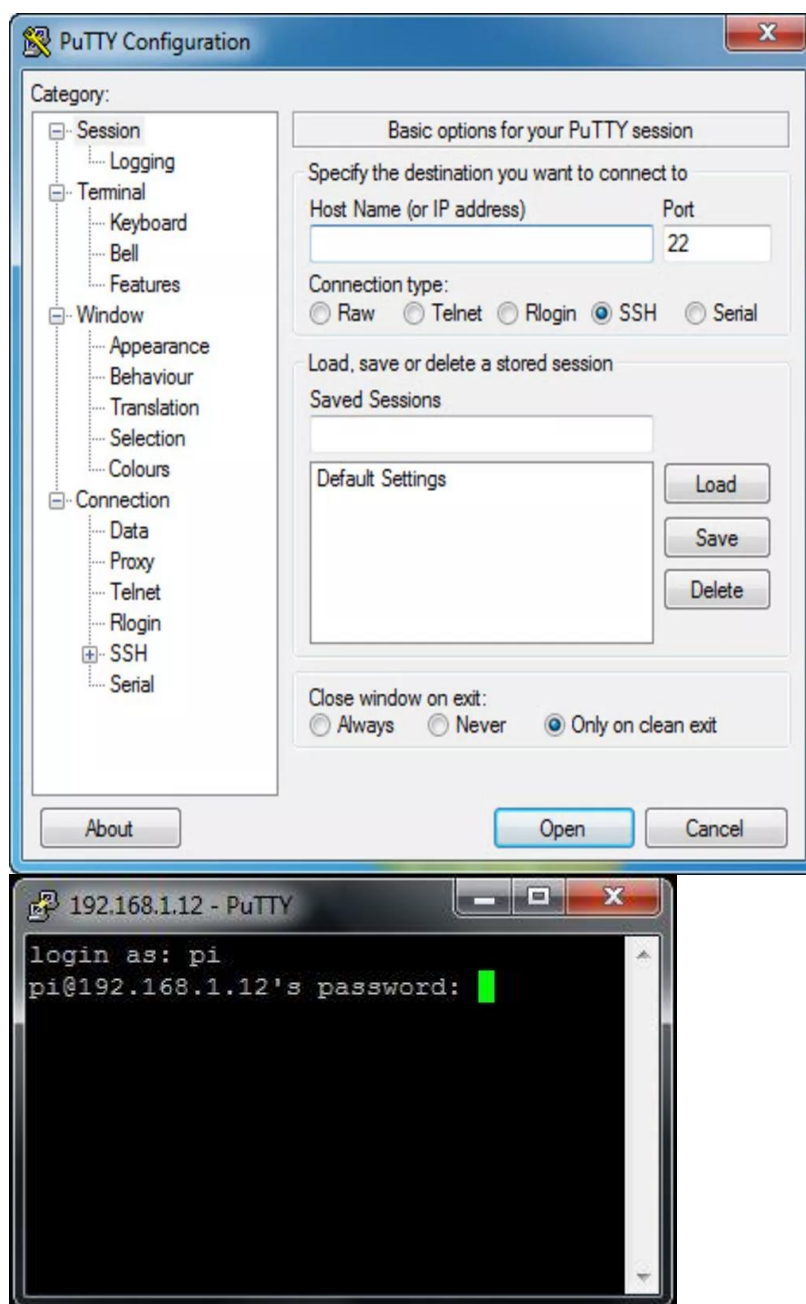
**Tibia**



## 6. Komunikacja w sieci komputerowej, protokół SSH

Często wykorzystywany protokół warstwy aplikacji, to protokół zdalnego zarządzania hostami, zwany SSH. Jest to standardowy protokół komunikacyjny używany w sieciach komputerowych TCP/IP. SSH jest następcą protokołu Telnet i służy on do terminalowego łączenia się ze zdalnymi komputerami. SSH różni się od Telnetu tym, że transfer wszelkich danych jest zaszyfrowany oraz możliwe jest rozpoznawanie użytkownika na wiele sposobów.

W projekcie program, w którym nawiązuje komunikację z Raspberry nazywa się PuTTY. PuTTY emuluje terminal tekstowy, co pozwala w łatwy sposób łączyć się z serwerem za pomocą wybranego protokołu jakim jest SSH.

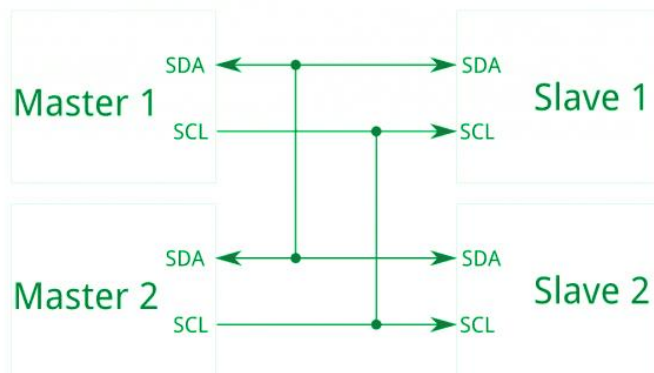




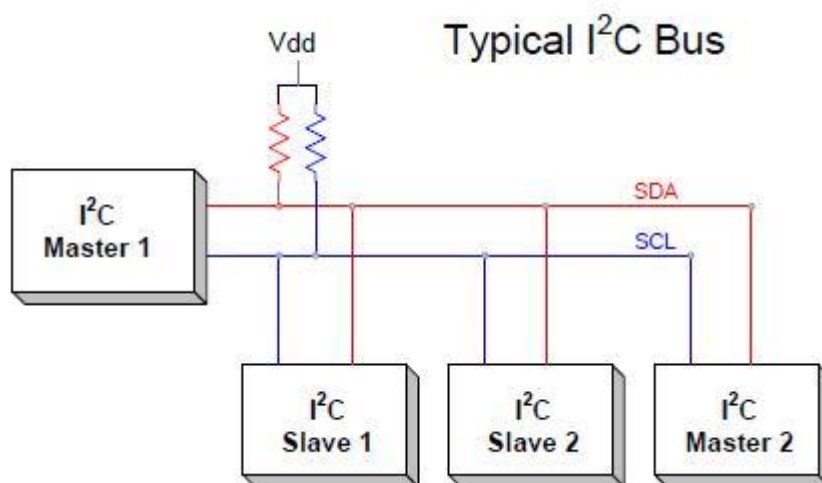
## 7. Magistrala I<sup>2</sup>C

Połączenie Arduino z Raspberry wydaje się być najbardziej przestronne jeśli chodzi o możliwości jakie przez integracje tych dwóch platform możemy uzyskać, ale również takie zestawienie jest najbardziej budżetowym rozwiązaniem.

Bardzo popularny interfejs komunikacyjny jakim możemy obsłużyć za pomocą tylko 2 linii aż 127 urządzeń jest I<sup>2</sup>C. Magistrala I<sup>2</sup>C to szeregową dwukierunkową magistralę służącą do przesyłania danych. Korzysta ona w tym celu tylko z dwóch pinów co jest oczywistą zaletą tego sposobu komunikacji. SCL - pin przez który przesyłany jest sygnał zegarowy, SDA - pin przez który przesyłane są dane.



Magistrala I<sup>2</sup>C przesyła dane w sposób synchroniczny, synchronizowany za pomocą sygnału zegarowego podawanego na pinie SCL. Komunikacja I<sup>2</sup>C działa w układzie Master, Slave, to znaczy że jedno urządzenie jest nadrzędne i steruje urządzeniem podrzędnym, które ma określony adres do którego Master czyli nasze urządzenie nadrzędne musi się odnieść. Możliwe jest przez to sterowanie wieloma urządzeniami podrzędnymi. W programie dla Arduino funkcja która odbiera dane spod zdefiniowanego wcześniej adresu nazywa się „receiveEvent” i odbiera dane w postaci kodu ASCII czyli siedmiobitowego systemu kodowania znaków, używanego aktualnie we współczesnych komputerach.

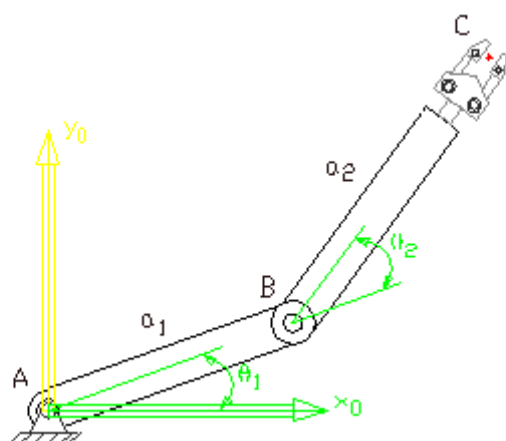


## 8. Obliczenia oraz opis kinematyki robota

Tak jak na wstępie zostało wspomniane, robot do realizacji ruchu korzysta z operacji matematycznych, przez które za pomocą macierzy przedstawia algorytm przemieszczenia, umożliwiając wyznaczenie położenia punktu końcowego w przestrzeni trójwymiarowej przechodząc przez poszczególne punkty pośrednie.

W robotyce bardzo często wykorzystuje się notację Denavita-Hartenberga umożliwiającą nam wyznaczenie równań kinematyki robota znając dane o zmiennych przegubowych możemy określić orientację końcówki roboczej.

Wyznaczenie pozycji i orientacji chwytaka na podstawie współrzędnych przegubowych to zadanie proste kinematyki. Najłatwiej będzie zacząć od przedstawienia tak prostego manipulatora dwuczłonowego.



$$\begin{aligned}x_0 &= a_1 \cos \theta_1 + a_2 \cos(\theta_1 + \theta_2) \\y_0 &= a_1 \sin \theta_1 + a_2 \sin(\theta_1 + \theta_2)\end{aligned}$$

Przedstawione działania pozwalają określić położenie punktu C dla poszczególnych osi X0 i Y0 znając kąty oraz długości poszczególnych członów manipulatora.

Gdybyśmy chcieli rozważyć ten sam problem stosując notację Denavita-Hartenberga musielibyśmy wyobrazić sobie dodatkową oś Z0 w naszym układzie współrzędnych oraz dodatkowe układy współrzędnych znajdujące przy każdym z członów robota.

Jedną z zasad jakie musimy przyjąć to ta, że obrót poszczególnych członów ramienia odbywa się względem osi Z, a przemieszczenie względem osi Z, a X.

Zgodnie z notacją należy przygotować tabele wraz z wypisanymi z parametrami kinematycznymi, takimi jak kąty obrotu, przemieszczenia członów oraz długości poszczególnych ramion. Przy parametrach które ulegają zmianie w czasie znajduje się indeks var.

Połączenie	$\Theta_i$	$D_i$	$A_i$	$\alpha_i$
1	$\Theta_1, \text{var}$	0	A1	0

2	$\theta_2, \text{var}$	0	A2	0
---	------------------------	---	----	---

Rozpatrywany przez nas manipulator płaski o dwóch stopniach swobody musimy przesunąć pomiędzy poszczególnymi układami współrzędnych, tak aby pomiędzy tym przesunięciem był tylko jeden parametr zmienny, a kolejno musimy rozpisac macierze przekształcenia jednorodnego dla poszczególnych członów, a wyglądają one tak.

Dla pierwszego połączenia

$$A_1 = Rot_{z, \theta_1} \cdot Trans_{x, a_1}$$

$$A_1 = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & 0 \\ \sin(\theta_1) & \cos(\theta_1) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & a_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & a_1 \cos(\theta_1) \\ \sin(\theta_1) & \cos(\theta_1) & 0 & a_1 \sin(\theta_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Dla drugiego połączenia

$$A_2 = Rot_{z, \theta_2} \cdot Trans_{x, a_2}$$

$$A_2 = \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 & 0 \\ \sin(\theta_2) & \cos(\theta_2) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & a_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 & a_2 \cos(\theta_2) \\ \sin(\theta_2) & \cos(\theta_2) & 0 & a_2 \sin(\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Mając zdefiniowane dwie macierze dla poszczególnych połączeń możemy ostatecznie wyciągnąć macieź transformacji będącą główną macieżą układu Y0 X0

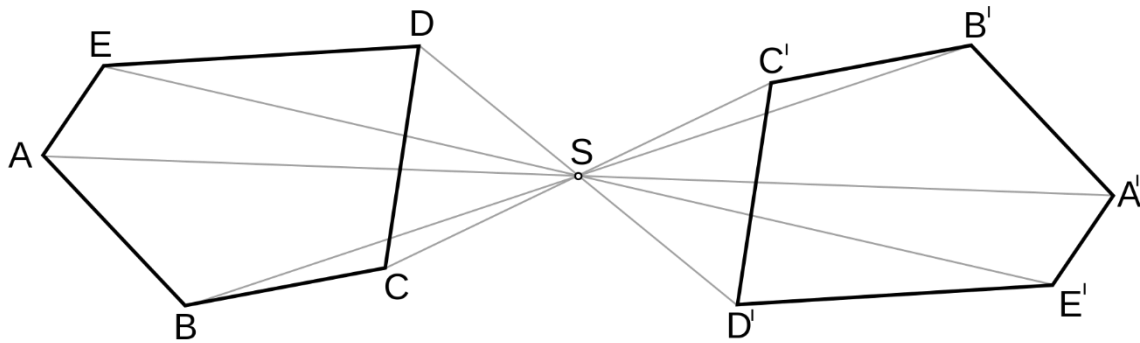
$$T_{2,0} = A_1 \cdot A_2$$

$$T_{2,0} = \begin{bmatrix} \cos(\theta_1 + \theta_2) & -\sin(\theta_1 + \theta_2) & 0 & a_1 \cos \theta_1 + a_2 \cos(\theta_1 + \theta_2) \\ \sin(\theta_1 + \theta_2) & \cos(\theta_1 + \theta_2) & 0 & a_1 \sin \theta_1 + a_2 \sin(\theta_1 + \theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Porównując równania przytoczone na samym początku możemy wywnioskować, że położenia chwytaka z obliczeń wychodzą identyczne jak w przypadku naszej wyjściowej macierzy.

Na tym relatywnie prostym przykładzie na chwilę przystaniemy abym mógł przytoczyć całościową macierz ruchu naszego robota, która jest bogatsza o macierz ruchu wektora w

przestrzeni euklidesowej to znaczy, że uwzględnia odwrócenia punktu docelowego, a odwrócenie punktu może zmienić chiralność, to znaczy zmienić prawy układ współrzędnych na lewy układ współrzędnych lub też na odwrót.



Cała rotacja plus inwersja tworzy zbiór macierzy ortogonalnych.

Tak całościowo już po wszelkich przejściach wygląda macierz robota

$$\begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} = [R_x * R_y * R_z] \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} \cos \beta & 0 & \sin \beta \\ \sin \alpha \sin \beta & \cos \alpha & -\sin \alpha \cos \beta \\ -\cos \alpha \sin \beta & \sin \alpha & \cos \alpha \cos \beta \end{bmatrix} \begin{bmatrix} \cos \gamma & -\sin \gamma & 0 \\ \sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$= \begin{bmatrix} \cos \beta \cos \gamma & -\cos \beta \sin \gamma & \sin \beta \\ \sin \alpha \sin \beta \cos \gamma + \cos \alpha \sin \gamma & -\sin \alpha \sin \beta \sin \gamma + \cos \alpha \cos \gamma & -\sin \alpha \cos \beta \\ -\cos \alpha \sin \beta \cos \gamma + \sin \alpha \sin \gamma & \cos \alpha \sin \beta \sin \gamma + \sin \alpha \cos \gamma & \cos \alpha \cos \beta \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Obrót elementarny, czyli taki wokół dowolnej osi układu współrzędnych wygląda tak:

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{bmatrix}$$

$$R_y(\beta) = \begin{bmatrix} \cos \beta & 0 & 0 \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{bmatrix}$$

$$R_z(\gamma) = \begin{bmatrix} \cos \gamma & -\sin \gamma & 0 \\ \sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Każda z tych macierzy obraca wektory kolumnowe przeciwnie do ruchu wskazówek zegara, jeśli spełniony jest któryś z warunków: oś obrotu kieruje się w stronę obserwatora, układ współrzędnych jest prawoskrętny, kąt obrotu  $\Theta$  jest dodatni.

Program jaki obsługuje Arduino znajduje się w „Załączniku 1”. Z racji tego, że kod jest dość długi i mimo starań nie jest na tyle klarowny by zrozumieć go od ręki, jego tok postępowania pozwolę sobie rozwinąć po krótku w tym miejscu.

Myślę że najwygodniej będzie zacząć patrząc już od elementu wykonawczego, serwomechanizmy swoje położenie wyznaczają poprzez odbiór co 20 ms sygnału długości 900-1500ms i ustawiają się stosownie do długości impulsu. Tak więc mamy zdefiniowaną tablicę, w której są wpisane nazwy pinów do których są podłączone serwomechanizmy „servo\_pin[4][3]”. Właśnie na te piny przypisywane są określone wartości przez funkcję „biegun\_dla\_serw”. Funkcja ta pobiera dane takie jak alpha, beta, gamma, które są danymi przeliczonymi ze współrzędnych kartezjańskich na biegun, to znaczy na odpowiednią wartość, którą możemy serwo ustawić. Funkcją odpowiadającą za wyliczenia danych takie jak, alpha, beta, gamma nazywa się „kartezjanski\_na\_biegunowy” i wykonuje potrzebne do tego obliczenia. Obie te funkcje są wywoływane przez funkcję „Serwa”, w której wywołana jest funkcja przerwań ustalająca w jakim czasie ma wykonać się przemieszczenie oraz ustawione zmienne statyczne, czyli widoczne tylko dla tej funkcji alpha, beta, gamma. Przez porównanie położenia funkcja przepisuje właśnie wartości do powyższych funkcji.

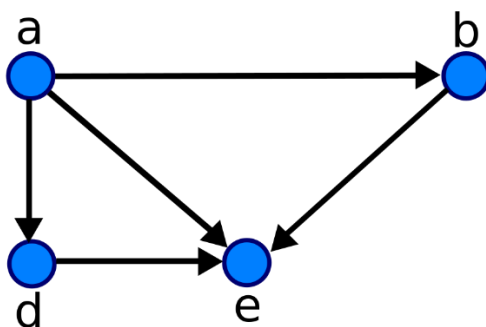
Główna funkcja programu czyli „setup” wpisuje wartości dla położenia początkowego odnosząc się do funkcji „ustawienie ” o której będzie zaraz, czyli po uruchomieniu arduino ustawiane są nogi robota na określone położenie później mamy inkrementację położenia czyli dla każdej nogi ustawiane jest kolejne położenie mamy tam zapis położenie teraz = położenie właściwe i są to tablice z wartościami kolejnych liczb wywołanych przez inkrementację w pętli „for” i jest to niejako powielenie funkcji „czekaj” która też będzie omówiona, a na końcu wywołujemy przerwanie w milisekundach dla funkcji „Serwa”. Idąc logiczną kolejnością w dalszej części mamy funkcję, która już dla danego ruchu ustawia poszczególne wartości i nazywa się ona „ustawienie” i właśnie w niej obliczane jest po przyjęciu danych zewnętrznych, które dla określonego ruchu są w nią wpisywane przez sekwencje ruchu jaką chcemy wykonać. Jej nieodłączną funkcją wykonywaną w danej sekwencji ruchu jest funkcja o nazwie „czekaj” i porównuje ona wartości będące wewnątrz tablicowych położeniach w danej chwili do położenia pożądanego. W pętli głównej naszego programu najistotniejsze jest postawienie nazwanej przeze mnie „flagowa\_przeszkoda” przypisanie do niej jakiejś wartości pozwala na określenie wykonania funkcji.

Pragnę zaznaczyć że wszelkie obliczenia w szerszym aspekcie są zapisane w programie, tutaj jedynie pobieżnie przedstawiam logikę działania.

## 9. Wykrywanie obiektów

W tym podpunkcie na samym początku opiszę jak całościowo zostały zaimplementowane i użyte biblioteki do wykrywania obiektów kolejno wgłębię się w zasadę działania czysto teoretyczną, ponieważ temat jest bardzo rozległy będzie to omówione w znacznym skrócie.

Tak więc jak na samym wstępie było powiedziane, że głównym silnikiem procesu jest biblioteka TensorFlow, potrafi ona nie tylko rozpoznawać obraz lecz także analizować mowę. Żeby zrozumieć zasadę działania TensorFlow trzeba zapoznać się z teorią grafu skierowanego ilustracja przedstawia właśnie przykładowy graf.

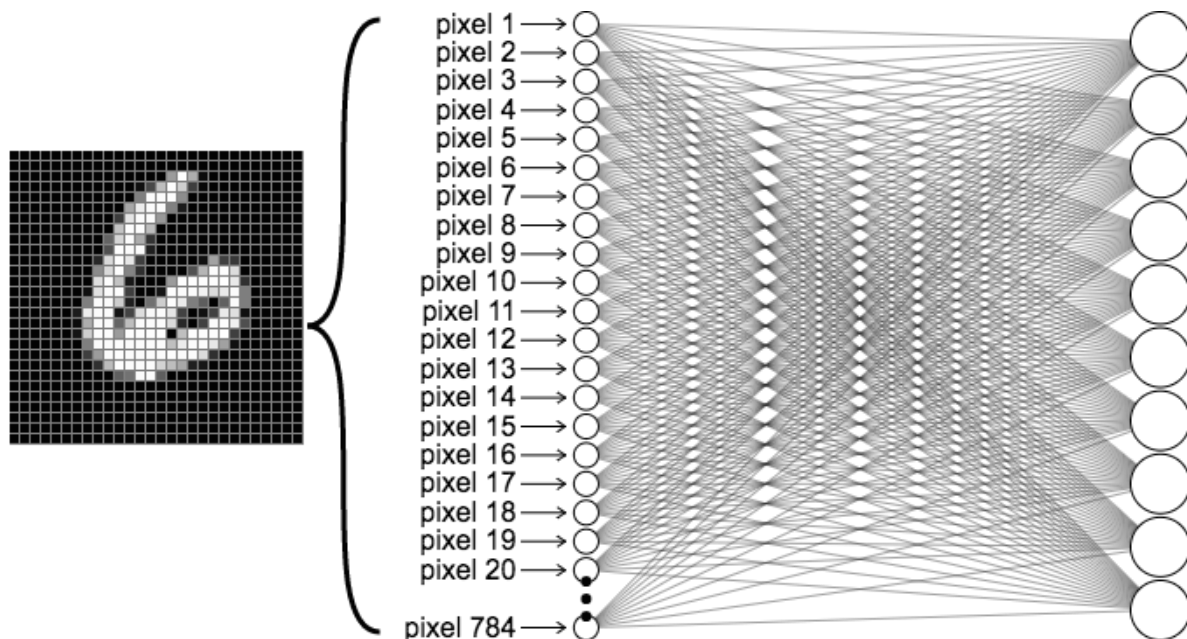


Jego wierzchołki to matematyczne operacje lub punkty, w których dochodzi do wymiany danych, zaś krawędzie opisują relacje wejścia/wyjścia pomiędzy węzłami, przedstawione w formie dynamicznych wielowymiarowych macierzy danych. Udostępniona przez Google biblioteka programistyczna pozwala na obliczenia numeryczne z wykorzystaniem takich grafów. Można tu korzystać zarówno z ogromnych klastrów superkomputerowych, stacji roboczych z wieloma GPU, jak i wielordzeniowych mobilnych procesorów w smartfonach. Jej twórcy podkreślają, że pozwala na elastyczne przetwarzanie danych bez wnikania w niskopoziomowe szczegóły, Google udostępniło nie tylko samą bibliotekę, ale też modele sieci neuronowych, wyszkolonych do rozpoznawania zdjęć i pisma odręcznego, analizowania tekstu oraz mowy.

Silnikiem odpowiedzialnym za przetwarzanie danych jest TensorFlow jednak on sam nie wystarczy. Niezbędna jest obróbka obrazu posłużyła mi do tego biblioteka również o otwartym kodzie źródłowym OpenCV, została zaprojektowana z myślą o wydajności obliczeniowej i silnym skupieniu się na aplikacjach czasu rzeczywistego. Zakres zastosowań obejmuje zestawy narzędzi 2D i 3D przez przeglądanie map w Internecie czy też właśnie w systemach rozpoznawania twarzy.

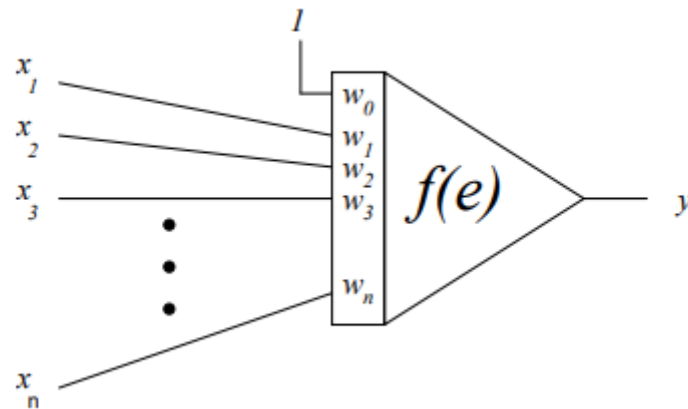
Do ważniejszych bibliotek które zostały zaimplementowane muszą zaliczyć jeszcze Protobuf - pełna nazwa to bufor protokołów, służy on do przechowywania oraz wymieniaania danych w strukturze danych .

Technika głębokiego uczenia się, która jest zdolna do kompleksowego wykrywania obiektów bez specyficznego definiowania cech opierają się na splotowych sieciach neuronowych. Sieci neuronowe są stworzone na wzór metaforycznie struktur nerwowych, rzeczywistych, biologicznych. Dobrym określeniem całości jest podejście techniczne do czegoś co jest bardzo biologiczne i inspiracja biologiczna do czegoś co możemy potem wykorzystywać w technice a mamy na myśli tutaj ludzki mózg. Operacje jakie są wykonywane w uczeniu maszynowym są bardzo bliskie operacjom wykonywanym przez ludzki układ nerwowy.

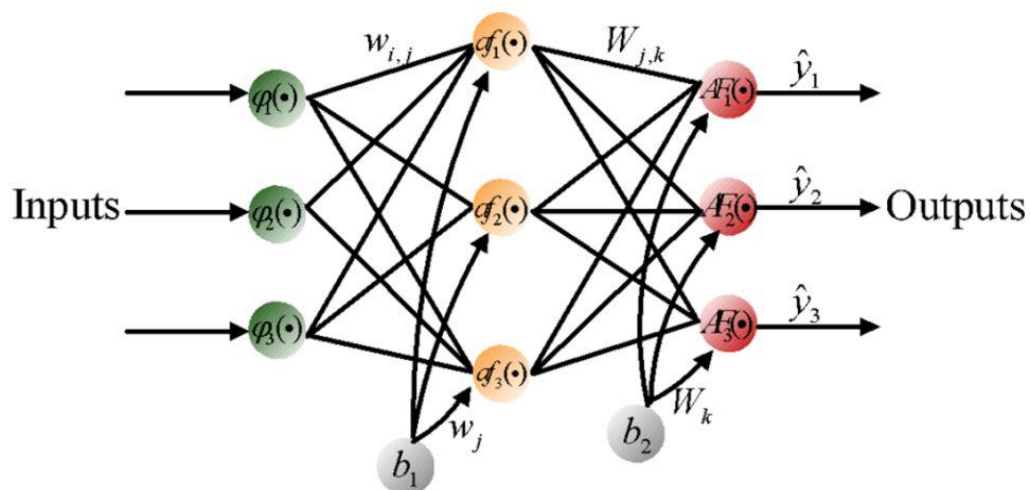


Sieć neuronowa to połączenie elementów zwanych sztucznymi neuronami, które tworzą co najmniej trzy warstwy: wejściową, ukrytą i wyjściową, przy czym warstw ukrytych może być wiele. Neurony sieci przetwarzają informacje dzięki temu, że podaje się na wejście sieci różne wzorce, a sieć na podstawie różnicy wyniku wyjściowego w stosunku do wzorca modyfikuje wagi połączeń, które modyfikują się podczas działania sieci. Modyfikowanie wag nazywane jest „uczeniem się” sieci. To właśnie w procesie uczenia się sieci tkwi istota sieci neuronowych. Ponieważ to właśnie modyfikacja wag pozwala metodą prób i błędów uwzględnić końcowy wynik z jak największą trafnością. Sama definicja sztucznej sieci neuronowej brzmi następująco – zbiór prostych jednostek obliczeniowych przetwarzających dane, komunikujące się ze sobą i pracujące ze sobą równolegle. Natomiast pojedynczy neuron jest jednostką obliczeniową zdolną do samodzielnego dokonania szacunku na dany temat przez co jest w pełni funkcjonalną jednostką. Własności neuronu determinuje przyjęta segregacja danych wejściowych oraz założona funkcja wyjściowa.

Jedną z identycznych własności komórki rzeczywistej, a sztucznego neuronu jest posiadanie jednego wyjścia i wielu wejść. Tak też posiadając dane z wielu źródeł ostateczny wynik jest kompensacją tych danych. Wynik wyjściowy podawany jest na wejście kolejnego neuronu tworząc poszczególne warstwy całej sieci.



Widzimy właśnie na zdjęciu graficzną interpretację pojedynczego neuronu w postaci specyficznego przetwornika sygnału, jak widzimy mamy jedno wyjście,  $n$  wejść sygnału wraz z wagami  $w_i$ , pobudzenie  $e$  neuronu czyli suma ważona sygnałów wejściowych oraz funkcję aktywności  $f(e)$ . Proces uczenia polega na modyfikowaniu wag sieci. Ma to odwzorowanie w schemacie działania naszego umysłu gdzie właśnie modyfikuje on swoje nawyki poprzez różnodoświadczenia. Zasada działania każdego neuronu jest identyczna, a dopiero ze złożoności struktur tworzonych przez miliony komórek wynika specjalizacja mózgu, zdolność nauki, zapamiętywania, rozwiązywania problemów. Całością zajmie się właśnie nasz silnik całego procesu jakim jest Tensorflow.





## 10. Zastosowanie w przemyśle

Obecnie branża technologiczna przechodzi światowy krok naprzód w stronę automatyzacji i robotyzacji przemysłu. Stanowi to znaczącą zmianę dla każdego zakładu niosąc za sobą poprawę i utrzymanie stałej jakości wytwarzanych produktów, znaczący wzrost efektywności linii produkcyjnej oraz zauważalną poprawę bezpieczeństwa stanowisk pracy. Mowa jest o tym tutaj dlatego że to właśnie popularyzacja robotów przemysłowych przyczyniła się do znacznych innowacji w przemyśle.

Omawiane roboty najczęściej występują w postaci mechanicznego ramienia o pewnej liczbie stopni swobody które możemy programowo ustawiać pod wybranym kątem. Środowiska w których są one programowane to w znacznej części uproszczone na korzyść operatora tło złożonego algorytmu do którego operator może wpisać dowolne parametry aby określić punkt w przestrzeni do którego robot końcówką robaczą ma dotrzeć. Tak też programowanie robotów w dużej mierze jest uproszczone i po części intuicyjne. Wgłębiając się w strukturę programu możemy zapoznać się z zasadą według, której robot znając punkt docelowy potrafi wyliczyć po kolei punkty pośrednie po których dotrze do swojego celu. Jest to w dużej mierze rozwijające i pozwala zrozumieć zasadę działania tych maszyn.

Możliwości jakie daje nam współczesna matematyka jak i oprogramowanie komputerowe są nieodłączną częścią tej dziedziny przemysłu, a ich wykorzystanie leży od strony innowacyjności oraz inwencji twórczej człowieka. Robot przedstawiony w projekcie wykorzystuje do realizacji swojego ruchu właśnie jedną z metod opartych na geometrii euklidesowej, przyjmując w tym przemieszczenie się w ortogonalny układzie współrzędnych.

Światowy boom w zakresie innowacji obchodzi w 2012 r. rozwój sztucznej inteligencji, w kolejnych latach przez ulepszanie algorytmów odsetek błędów przy rozpoznawaniu obrazu znacznie spadł. W odniesieniu do naszego robota mamy do czynienia z rozpoznawaniem i opisywaniem obiektów wizualnych. Minikomputer jakim jest Raspberry odpowiednio skonfigurowany potrafi po otrzymaniu próbki obrazu rozpoznać a także podpisać co znajduje się na zdjęciu z dokładnością wyliczonego przez algorytm prawdopodobieństwa procent. Działania jakie wykonuje algorytm oparte są nawielowarstwowej sieci neuronowej. Struktury matematyczne wraz z ich oprogramowaniem pozwalają na wykorzystanie najnowszych technologii w wielu branżach takich jak medycyna, kosmologia czy obsługa klienta. Architektura głęboka osiąga wyniki porównywalne, a w niektórych przypadkach lepsze od ludzkich ekspertów w takich dziedzinach jak rozpoznawanie mowy, obrazu, a po twarzy nawet orientacji seksualnej przetwarzanie naturalnego języka, filtrowanie dźwięku czy filtrowanie w sieciach społecznościowych. Słowem głębokie uczenie pozwala maszynie nie tyle co widzieć ale wiedzieć co widzi. W przemyśle, wizja komputerowa wykorzystywana jest przy procesach produkcyjnych i pozwala przykładowo określić położenie detali.

## 11.Program, Arduino

- Załącznik 1

## 12.Program, symulacji w języku Python

- Załącznik 2

## 13.Program, komunikacja Arduino z Raspberry

- Załącznik 3