Zakład Teorii Maszyn i Robotów
ITLiMS, MEiL, PW

Praca Przejściowa Inżynierska

Rok akademicki 2022/2023, semestr letni

Implementacja środowiska ROS na urządzeniu RaspberryPi

Autor

Jakub Głowacki

Warszawa, 23.02.2023

Spis treści

1 Wstęp

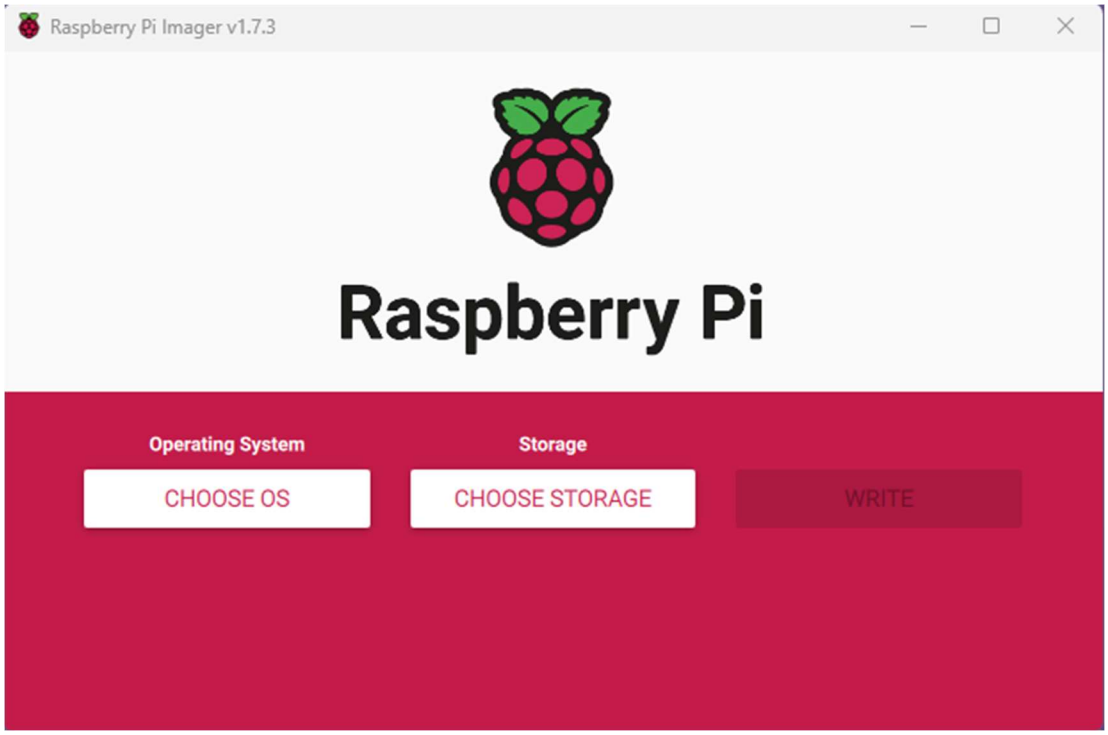
Celem pracy jest implementacja środowiska ROS2 na urządzeniu Raspberry Pi, a następnie skomunikowaniu się z komputerem personalnym.

2 Przygotowanie Raspberry Pi

Ze względu na pełne wsparcie jedynie systemów Ubuntu przez środowisko ROS2 należy odejść od standardowego oprogramowania Raspberry, zwanego RaspbianOS. Zamiast tego korzystać będziemy z systemu Ubuntu Server 22.04 w wersji 64-bitowej. Ze względu na charakter użyteczności Raspberry, które może służyć jako na przykład komputer pokładowy robota mobilnego, nie będziemy instalować GUI systemu, wystarczy nam dostęp do terminala.

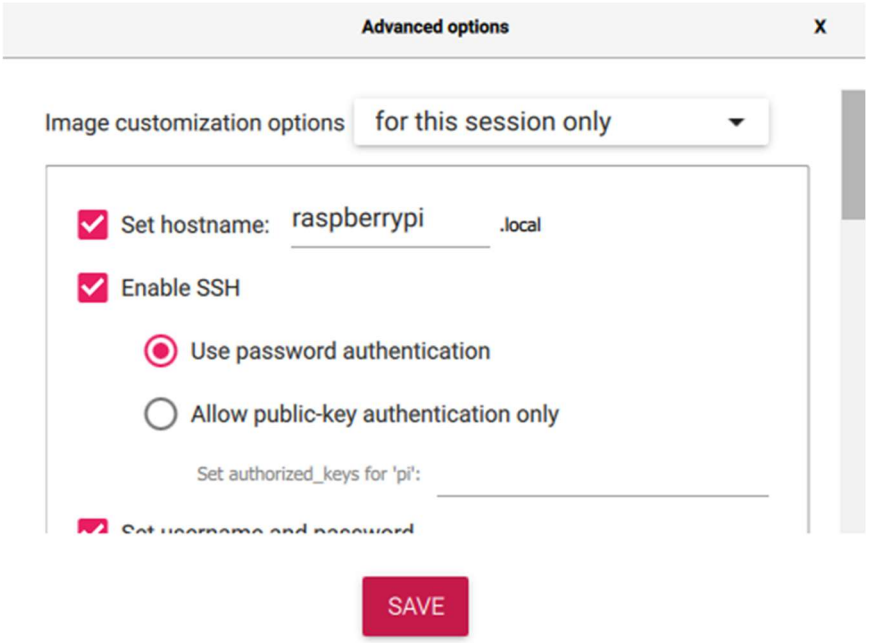
Aby zainstalować system musimy skorzystać z programu Imager dostępnego na stronie producenta za darmo do pobrania, aby nagrać obraz systemu na kartę microSD pełniącą funkcję dysku twardego.

Ekran główny programu wygląda następująco:

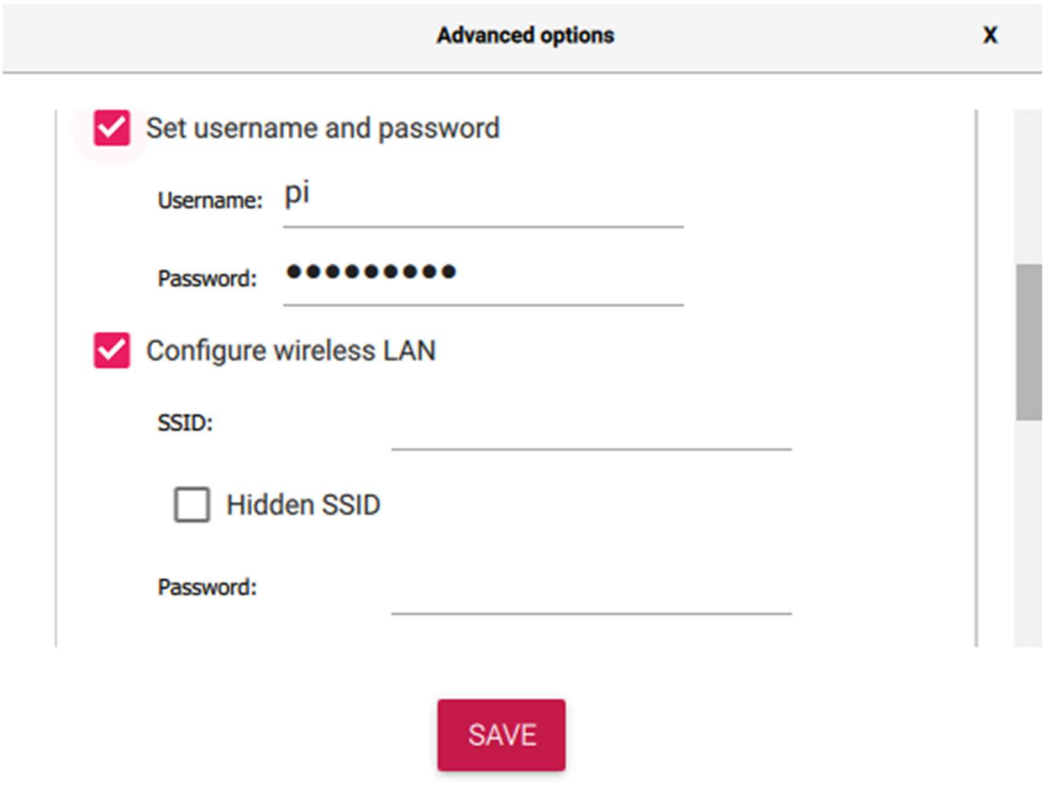


Rysunek 1 - imager

Następnie wybieramy interesujący nas system, czyli Ubuntu Server 22.04 64-bit i docelowe urządzenie (nasza karta microSD). Następnie powinien nam się wyświetlić dodatkowy przycisk opcji zaawansowanych, w które wchodzimy i konfigurujemy kolejno nazwę hosta, włączamy SSH, tworzymy profil podstawowy (domyślnie jest to zwykle nazwa użytkownika pi, hasło raspberry), oraz podajemy SSID oraz hasło do naszego Wi-Fi (inną możliwością jest podłączenie się do sieci przy pomocy kabla Ethernet). Ważnym jest, aby łączyć się do Wi-Fi o częstotliwość 2.4Ghz, ponieważ nie każde urządzenie obsługuje 5Ghz. Opcje zaawansowane powinny wyglądać jak na rysunkach poniżej:



Rysunek 2 – opcje zaawansowane cz. 1



Rysunek 3 – opcje zaawansowane cz. 2. Sieć WLAN należy uzupełnić w zależności od wykorzystywanej sieci.

Gdy już wszystko ustawimy należy wcisnąć przycisk WRITE i poczekać aż karta zostanie nagrana.

3 Przygotowanie komputera klasy PC

3.1 System

Nasz prywatny komputer, który będzie używany jako główny host programu również musi być wyposażony w oprogramowanie ROS2. W ostatnim czasie następuje migracja na nową wersję oprogramowania, a co za tym idzie pojawia się wsparcie dla systemów Windows, jednak ze względu na lepsza kompatybilność dalej wybieram Ubuntu 22.04 LTS jako system, w którym będę pracował.

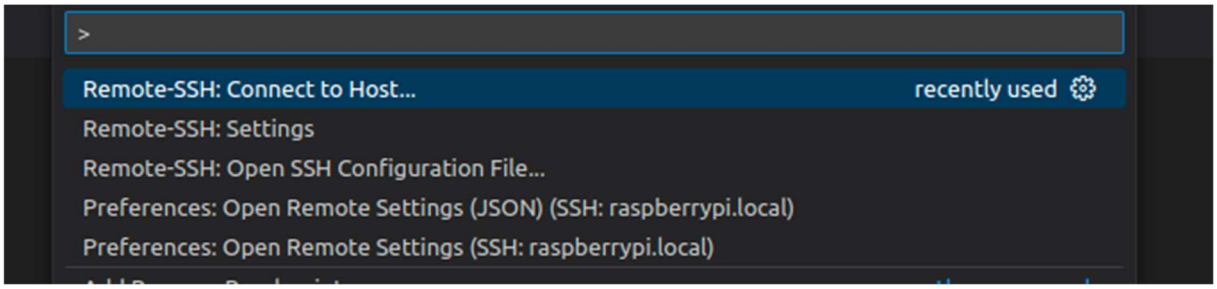
3.2 Oprogramowanie

W celu prostej pracy z ROS warto przygotować kilka programów. Oprogramowanie zalecane to Visual Studio Code. Jest to edytor tekstowy oparty na licencji Open-Source, który pozwala na dodawanie wielu rozszerzeń ułatwiających prace programisty. W naszym przypadku będziemy z niego korzystać, aby ustanowić polaczenie SSH z RaspberryPI, jak również pomoże nam w organizacji wierszy poleceń. W przypadku pracy z ros często pojawia się ich duża ilość, co często może sprawiać problemy w pracy.

3.3 Połączenie SSH

W tym momencie powinniśmy mieć dostęp do Internetu na RPi, poprzez Wifi, którego dane uzupełniliśmy podczas nagrywania karty SD. Jeżeli jednak występują problemy, dobrym rozwiązaniem jest skorzystanie z kabla Ethernet.

W celu ustanowienia polaczenia SSH z RPi należy pobrać w programie Code dodatek Remote-SSH, wcisnąć F1, aby otworzyć konsole programu i wpisać/znaleźć komendę “Connect to Host” podświetloną na rysunku poniżej:



Rysunek 4

Następnie wybieramy opcje stworzenia nowego hosta i wpisujemy komendę pozwalającą połączyć się zdalnie z urządzeniem - “ssh {username}@{adress}”, gdzie username to nazwa użytkownika, a adres to adres hosta urządzenia. Zakładając ze w zaawansowanych ustawieniach instalacji ustawiliśmy wszystko zgodnie z instrukcja możemy wpisać “ssh pi@raspberrypi.local”. W przypadku błędów możemy również zamienić drugi człon na adres IP urządzenia w sieci LAN.

W przypadku znalezienia urządzenia w sieci, należy zaakceptować wszystkie komunikaty które się pojawia na ekranie, odczekać na nawiązanie połączenia i zainstalowanie odpowiednich pakietów na urządzeniu docelowym i można przejść do rozdziału 4.

3.4 Skanowanie sieci LAN

W przypadku problemów z połączeniem warto przeskanować sieć LAN w poszukiwaniu urządzeń aktywnych, co pomoże nam zidentyfikować, czy nasze RPi ma dostęp do sieci lokalnej.

Polecanym programem do znalezienia urządzeń w sieci LAN jest program nutty, który można pobrać wpisując następujące komendy w terminalu:

sudo apt-add-repository ppa:bablu-boy/nutty

sudo apt-get update

sudo apt-get install com.github.babluboy.nutty

W przypadku problemów z instalacją należy spróbować dodać repozytorium elementaryOS i ponownie zainstalować komendami:

sudo add-apt-repository ppa:elementary-os/stable

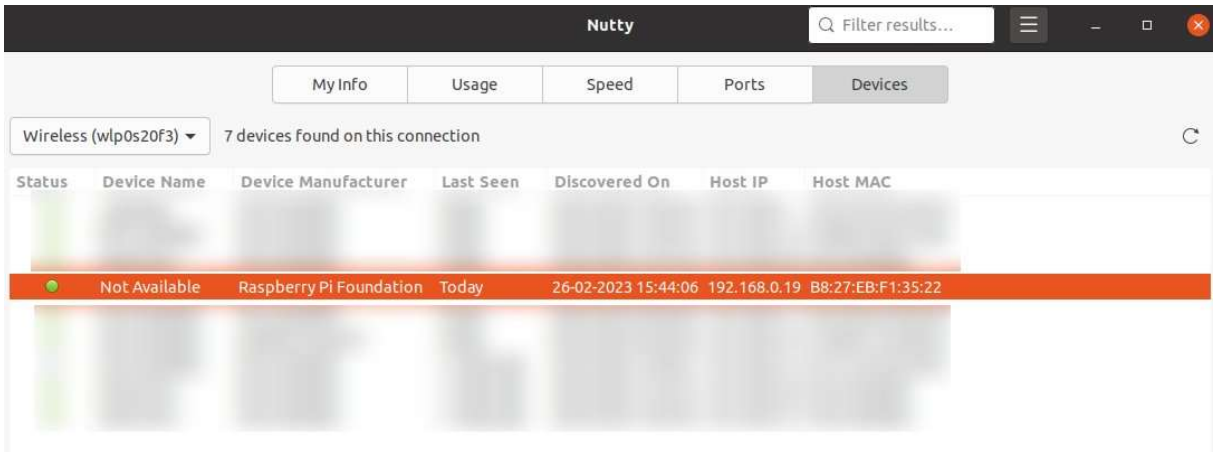
sudo apt-get update

sudo apt-get install com.github.babluboy.nutty

Po instalacji można usunąć dodatkowe repozytorium analogiczną komenda z dodatkowym argumentem --remove:

sudo add-apt-repository -–remove ppa:elementary-os/stable

Po instalacji otwieramy program, wybieramy zakładkę “Devices”, a następnie wybieramy sterownik sieci z listy rozwijanej (Rysunek 5), oraz odświeżamy listę. RPi powinno się wyświetlić bez nazwy, jednak rozpoznamy je po producencie Raspberry Pi Foundation. Jeżeli istnieje, to w komendzie “ssh {user}@{adress}” zamiast członu raspberrypi.local możemy wpisać IP urządzenia. Musimy jednak uważać, gdyż w przypadku dynamicznej alokacji adresów przez router, adres ten może ulec zmianie i będziemy musieli ponownie konfigurować polaczenie. Nazwa hosta jest polecanym rozwiązaniem, gdyż jest ona niezależna od sieci LAN do której jesteśmy podłączeni.



Rysunek 5 - okno programu nutty

4 Instalacja ROS2

Instalacja zarówno dla komputera sterującego robotem, jak i dla RPi jest z drobnymi różnicami taka sama, więc można ją przeprowadzać jednocześnie w dwóch oknach VSCode, dla komputera lokalnie, oraz tym, w którym połączyliśmy się przez SSH z RPi.

Aby uzyskać dostęp do terminala należy w oknie VSCode wcisnąć kombinacje klawiszy Ctrl+`, lub wybrać opcję nowy terminal w menu Terminal na górnym pasku programu.

Na początku chciałbym zaznaczyć, że jeżeli ROS instalowany jest na urządzeniu w niektóre alternatywne sposoby, przed przystąpieniem do instalacji warto sprawdzić przy pomocy komendy “locale” czy wspierany jest system kodowania UTF-8. W niektórych minimalnych warunkach, takich jak na przykład zawieranie ROSa w kontenerze Docker’a, może wystąpić jakiś prostszy system kodowania, przez który mogą później wystąpić problemy.

Aby przejść do instalacji należy upewnić się, że dodane jest do systemu repozytorium Ubuntu Universe, poprzez wpisanie do terminala następujących komend:

sudo add-apt-repository universe

Następnie dodajemy klucz GPG ROSa:

sudo apt update && sudo apt install curl

sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -o /usr/share/keyrings/ros-archive-keyring.gpg

I w końcu dodajemy repozytorium ROSa:

echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/ros-archive-keyring.gpg] http://packages.ros.org/ros2/ubuntu $(. /etc/os-release && echo $UBUNTU_CODENAME) main" | sudo tee /etc/apt/sources.list.d/ros2.list > /dev/null

Następnie przy świeżej instalacji systemu należy zaktualizować oprogramowanie przy pomocy komend:

sudo apt update

sudo apt upgrade

Pozwala to na uniknięcie nadpisywania związanych z systemem pakietów *systemd* oraz *udev*, we wczesnych aktualizacjach systemu w wersji 22.04. Możliwe ze podczas wykonywania funkcji pojawi się ekran mówiący o restartowaniu procesów systemowych, lub o konieczności restartu systemu. Bez zmiany żadnej opcji akceptujemy oba, po czym po wykonaniu się aktualizacji należy zrestartować system[1].

[1] Dla RPi gdy nie mamy GUI możemy to zrobić np. komendą *sudo reboot*. Przedrostek sudo w systemach UNIX uruchamia komendę z uprawnieniami administratora, przez co należy podać hasło użytkownika przed jej wykonaniem.

W końcu instalujemy ROSa w jednej z trzech wersji:

Desktop - pełny pakiet ROS ze wszystkimi programami współpracującymi takimi jak symulatory graficzne i tym podobne. Będzie to idealne dla naszego komputera sterującego robotem.

sudo apt install ros-humble-desktop

Ros base – bardziej okrojona wersja, posiadająca biblioteki do komunikacji, pakiety wiadomości, narzędzia korzystające z terminala. Brakuje głownie narzędzi GUI. Jest to zalecana wersja na RPi do naszego zastosowania, ze względu na to, że symulacje i tym podobne rzeczy uruchamiać możemy na komputerze personalnym nie obciążając jednocześnie RPi.

sudo apt install ros-humble-ros-base

Development tools – najbardziej okrojona wersja, posiadająca jedynie kompilator i inne narzędzia wymagane do budowania pakietów ROS. Nie będziemy używać jej w tej instrukcji.

sudo apt install ros-dev-tools

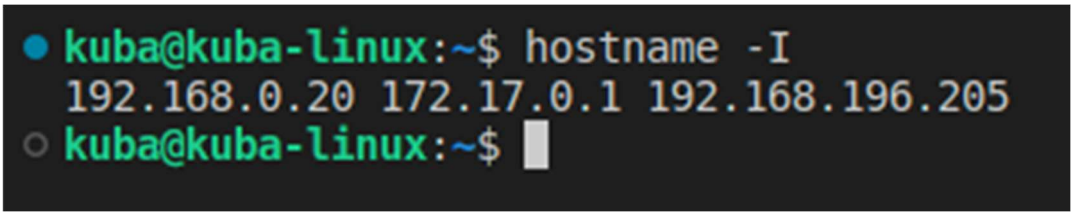
5 Nawiązanie komunikacji oraz przykładowy program w ROS

5.1 Przygotowanie środowiska

Aby uzyskać połączenie w ROSie potrzebujemy przede wszystkim adresów IP komputera, który będzie sercem programu, oraz maszyny, która będzie zbierała i wysyłała dane na komputer sterujący. Możemy do tego użyć wcześniej wspomnianego programu nutty, lub mając już dostęp do obu terminali przy pomocy komendy:

hostname –I

Powinna ona pokazać nam adresy IP maszyny (Rysunek 6). W najlepszym wypadku będzie to jeden adres, którego następnie użyjemy do konfiguracji. W przypadku większej liczby kart sieciowych adresów może być więcej, jednak są to zwykle te, które mają format 192.168.x.xxx.



Rysunek 6 - sprawdzanie adresu

Można sprawdzić komunikację między nimi używając komendy:

ping {adres IP}

Gdzie adresem będzie oczywiście IP maszyny przeciwnej do tej, z której uruchamiamy komendę.

Jeżeli udaje się odebrać pakiety, oznacza to, że połączenie jest ustanowione. Następnie, aby rozdzielić procesy między różne programy w sieci lokalnej, istnieje zmienna środowiskowa ROS_DOMAIN_ID. Można przypisać jej wartość od 1 do 232. Każda wartość będzie odpowiadać za oddzielny projekt w sieci. Jeżeli jesteśmy pewni, że będziemy korzystać na danej maszynie tylko z jednego projektu, możemy dodać w pliku w *~/.bashrc* linijkę “ROS_DOMAIN_ID={ID}”, gdzie ID będzie naszym wybranym numerem z zakresu. W innym wypadku, musimy za każdym razem otwierając nowy terminal napisać komendę:

export ROS_DOMAIN_ID={ID}

Aby terminal wiedział w której domenie się komunikować.

Rozwiązanie to jest na tyle wygodne, że możemy w sieci lokalnej i na oddzielnych komputerach w dowolny sposób rozdzielać procesy. Na jednym komputerze mogą działać programy działające w obrębie różnych programów, które będą komunikowały się z dowolną liczbą innych maszyn.

Jako, że ROS oparty jest na wielu zmiennych środowiskowych, należy podobnie jak w przypadku domeny w każdym terminalu załączyć skrypt, w którym zawarta jest większość domyślnych wartości zmiennych środowiskowych ROSa, komendą:

source opt/ros/humble/setup.bash

W przypadku gdy wiemy, że zawsze będziemy korzystać z tego samego skryptu, można dodać linijkę taką jak komenda powyżej znów do pliku *~/.bashrc*.

Na koniec warto zainstalować dodatkowy pogram o nazwie colcon, który będzie nam służył do budowania programów.

sudo apt install python3-colcon-common-extensions

5.2 Pierwszy program

W celu przetestowania działania ROSa, oraz komunikacji posłużymy się jednym z najprostszych przykładowych projektów. Będzie on wykorzystywał dwa programy, jeden wysyłający ciągu znaków z RPi, a drugi odbierający je na komputerze. W tym celu należy stworzyć folder o nazwie naszego projektu, przykładowo "ProjektROS", a następnie w nim stworzyć folder "src" w którym zawarte będą kody źródłowe projektu. Zrobimy do komendami:

mkdir ProjektROS/src

cd ProjektROS

Będąc w folderze src należy stworzyć pakiet, w którym zawarte będą programy, wejść do nowo utworzonego folderu, oraz z poziomu komputera pobrać listener node[2] natomiast z poziomu RPi talker node:

ros2 pkg create --build-type ament_python py_listentalk

cd py_listentalk/py_listentalk

wget https://raw.githubusercontent.com/ros2/examples/foxy/rclpy/topics/minimal_publisher/examples_rclpy_minimal_publisher/publisher_member_function.py

wget https://raw.githubusercontent.com/ros2/examples/foxy/rclpy/topics/minimal_subscriber/examples_rclpy_minimal_subscriber/subscriber_member_function.py

Następnie należy dokonać pewnych zmian w plikach o nazwach *package.xml*, oraz *setup.py*, które znajdują się w lokalizacji */ProjektROS/src/py_listentalk*. W pierwszym z nich należy uzupełnić informacje nt. autora pakietu uzupełniając następujące tagi:

```
6  <description>Krótki opis działania pakietu</description>
7  <maintainer email="email@email.pl">Imię i nazwisko</maintainer>
8  <license>Rodzaj licencji</license>
```

Rysunek 7 - zmiany w package.xml

Oraz dodając informacje o zależnościach z jakich będą nasze programy korzystały dodając po powyższych informacjach następujące dwie linijki:

```
<exec_depend>rclpy</exec_depend>
<exec_depend>std_msgs</exec_depend>
```

Rysunek 8 - dodatkowe linie w package.xml

[2] Node – podstawowy budulec sieci ROS. Jest on odpowiednikiem pojedynczego programu, który z podstawowych funkcji może słuchać tematów, w których pojawiają się wiadomości, lub samemu je nadawać.

Analogicznie należy w pliku setup.py uzupełnić informacje nt. autora oraz dodać kilka linijek kodu, tak aby plik wyglądał jak poniżej:

```
16      maintainer='imię',
17      maintainer_email='email@email.pl',
18      description='krótki opis działania',
19      license='rodzaj licencji',
20      tests_require=['pytest'],
21      entry_points={
22          'console_scripts': [
23              'listener = py_listentalk.subscriber_member_function:main',
24          ],
25      },
26  )
27
```

Rysunek 9 - zmiany w setup.py

W przypadku RPi należy wszystkie kroki wykonać analogicznie, jednak podczas uzupełniania setup.py zamienić w ostatniej linijce *listener* na *talker*, oraz *subscriber* na *publisher*.

Następnie możemy przejść do zbudowania projektu oraz dodaniu nowego pakietu do środowiska. W tym celu w folderze z projektem wpisujemy komendy:

colcon build --packages-select py_listentalk

. install/setup.bash

Teraz możemy już uruchomić nasze programy komendami:

ros2 run py_listentalk listener

ros2 run py_listentalk talker

Na terminalach powinny wyświetlać się cyklicznie następujące wiadomości:

```
pi@raspberrypi:~/przejsciowka$ ros2 run py_listentalk talker
[INFO] [1678825024.162663364] [minimal_publisher]: Publishing: "Hello World: 0"
[INFO] [1678825024.519711072] [minimal_publisher]: Publishing: "Hello World: 1"
[INFO] [1678825025.018260939] [minimal_publisher]: Publishing: "Hello World: 2"
[INFO] [1678825025.519712553] [minimal_publisher]: Publishing: "Hello World: 3"
[INFO] [1678825026.017175195] [minimal_publisher]: Publishing: "Hello World: 4"
[INFO] [1678825026.518522544] [minimal_publisher]: Publishing: "Hello World: 5"
```

```
kuba@kuba-linux:~/przejciowka$ ros2 run py_listentalk listener
[INFO] [1678825186.907803725] [minimal_subscriber]: I heard: "Hello World: 16"
[INFO] [1678825187.363739602] [minimal_subscriber]: I heard: "Hello World: 17"
[INFO] [1678825187.865044619] [minimal_subscriber]: I heard: "Hello World: 18"
[INFO] [1678825188.447189446] [minimal_subscriber]: I heard: "Hello World: 19"
```

Rysunek 10 - terminale obu urządzeń

Jeżeli wiadomości się zgadzają, oznacza to, że komunikacja przebiega pomyślnie, oraz że udało się stworzyć nasz pierwszy prosty pakiet programów ROS2.

6 Użycie kamery

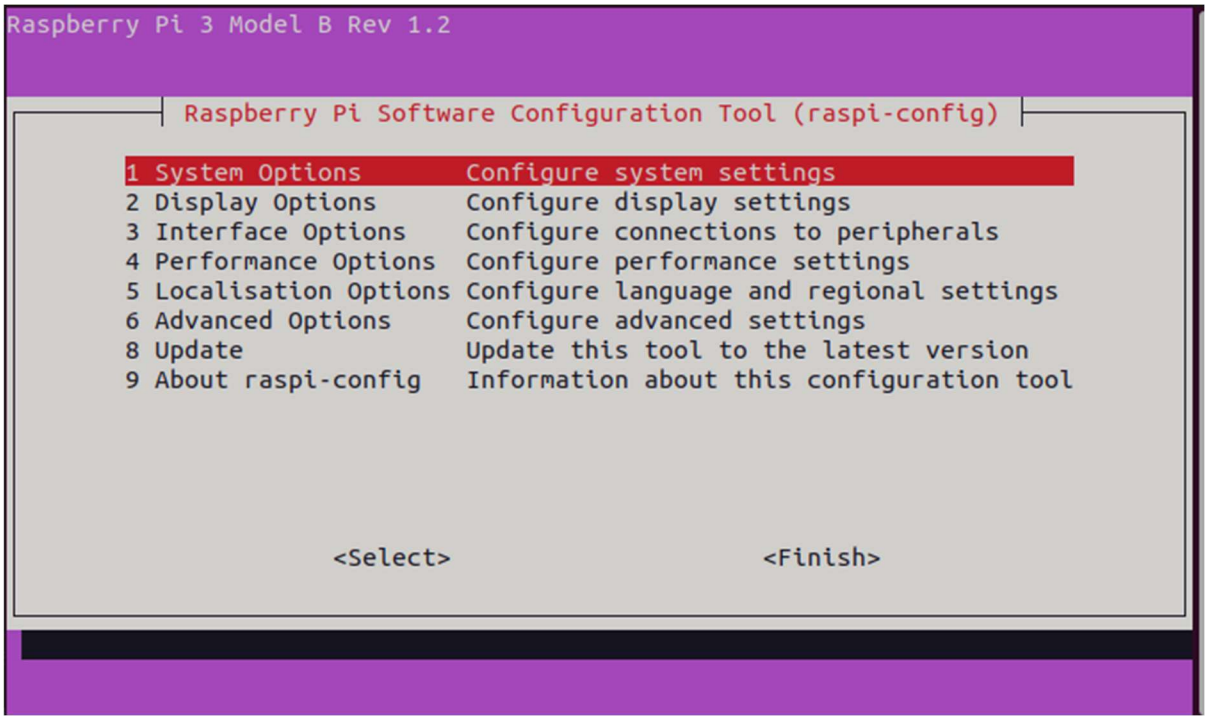
6.1 Konfiguracja kamery

W celu użycia kamery w ROS2 należy przede wszystkim aktywować kamerę w opcjach konfiguracyjnych. Najpierw pobieramy, a następnie uruchamiamy program do obsługi RPi. W przypadku problemów z obsługą w terminalu VSCode, można skorzystać z domyślnego terminala zainstalowanego na systemie, uprzednio łącząc się przez SSH poprzez tę samą komendę co w przypadkach powyżej: *ssh pi@raspberrypi.local*, oraz wpisanie hasła.

sudo apt install raspi-config

sudo raspi-config

Ukaże nam się następujący ekran:



Rysunek 11 - ekran głowny menedżera konfiguracji RPi

Przy pomocy strzałek wybieramy w menu opcję *Interface options* a następnie włączamy tam interfejs obsługujący kamerę oraz SPI. Następnie w *Permormance Options -> GPU Memory* upewniamy się, że jest ustawione 128MB. W tym momencie możemy opuścić menedżer wybierając opcję *Finish.* Aby wszystkie zmiany zaczęły działać, należy zrestartować RPi przy pomocy komendy *sudo reboot.*

6.2 Korzystanie z kamery w ROS

W tym momencie kamera powinna działać. Można więc przejść do integracji jej z systemem ROS. Użyjemy do tego dostępnego z oficjalnego repozytorium *v4l2-camera node*, który korzysta z powszechnego sterownika do obsługi video dla systemów opartych na Linuxie. Dodatkowo aby wspomóc przesył danych, będziemy odczytywali jedynie obraz po kompresji, do czego użyjemy *transport_image node*. Ten pierwszy wystarczy zainstalować jedynie na RPi, drugi natomiast będzie również potrzebny na komputerze personalnym do dekompresji obrazu. Instalujemy przy pomocy komend:

sudo apt update

sudo apt install ros-humble-v4l2-camera

sudo apt install ros-humble-image-transport-plugins

Do analizy systemu polecam również zainstalować na komputerze personalnym rqt, czyli program do wizualizacji przesyłu danych przy pomocy różnych dodatków. W naszym przypadku bardzo pomocny okaże się *Node_Graph* oraz *Image_View*.

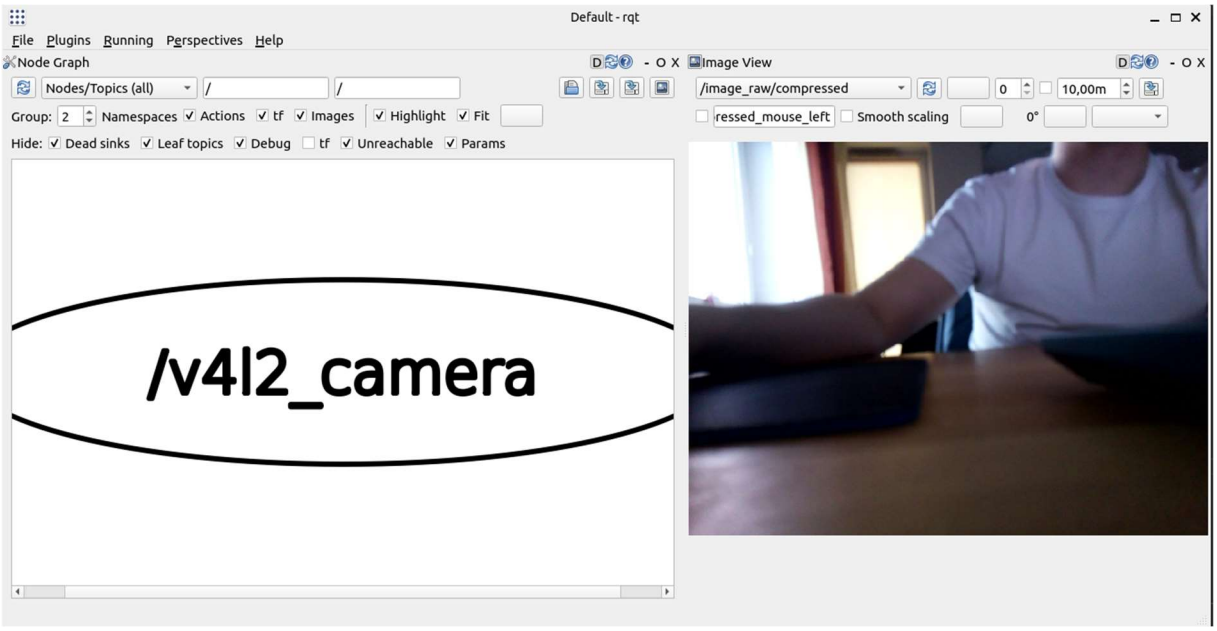
*sudo apt install ros-humble-rqt**

Teraz możemy już włączyć odczyt z kamery na RPi, a na komputerze rqt:

ros2 run v4l2-camera v4l2-camera-node

rqt

Powinno się wyświetlić okno rqt, w którym należy wejść w menu *plugins->instrospection->node_graph*, oraz *plugins->visualization->image_view*, w celu otworzenia odpowiednich narzędzi podglądowych. Jeżeli wszystko jest poprawnie ustawione, powinniśmy być w stanie po odświeżeniu *Node_graph,* oraz po odświeżeniu *Image_view* i wybraniu tematu */image_raw/compressed* zobaczyć następujący widok.

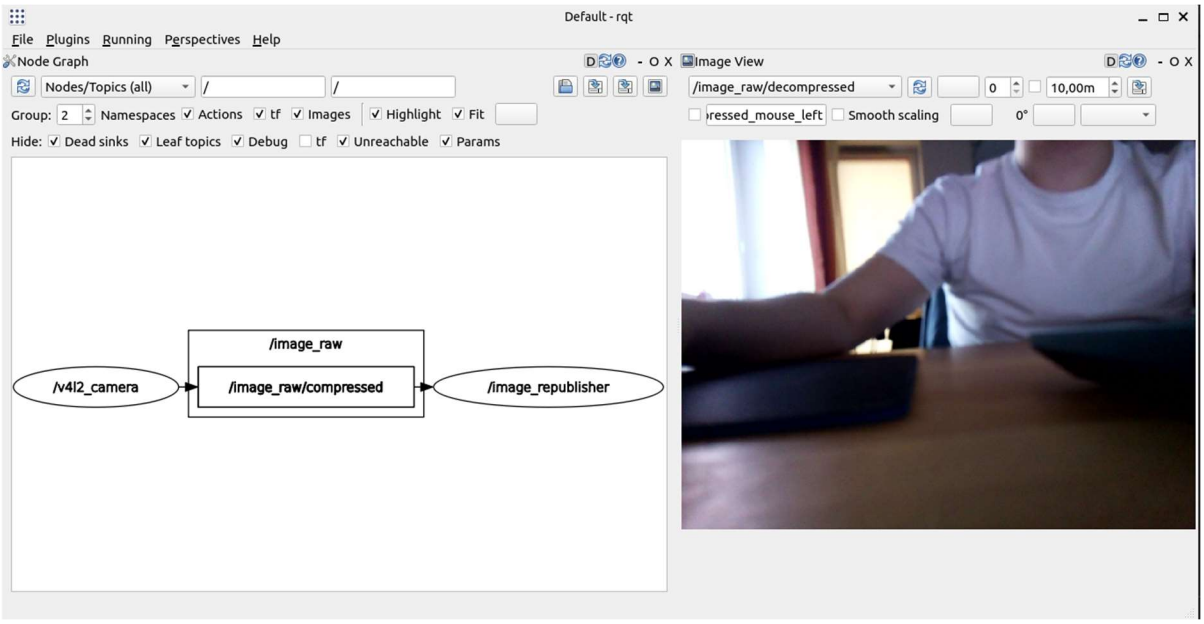


Rysunek 12 - rqt wraz z pluginami użytymi do podglądu przesyłu informacji

Jest to oczywiście obraz z kamery oraz wykres (póki co dosyć ubogi, bo składający się tylko z jednego programu) projektu. Następnie w celu obsługi obrazu przy pomocy kodu należy dokonać dekompresji obrazu za pomocą pakietu *image_transport*. Polegać to będzie na subskrypcji do tematu */image_raw/compressed,* a następnie publikacji nowego tematu */image_raw/decompressed.* Dziać się to będzie już na komputerze personalnym, a dzięki temu jesteśmy w stanie zachować płynność przy przesyle obrazu z RPi na komputer.

ros2 run image_transport republish compressed --ros-args --remap in/compressed:=image_raw/compressed --remap out:=image_raw/decompressed

Jeżeli wszystko działa poprawnie zobaczymy po odświeżeniu nowy wątek w image_view, oraz nasz wykres projektu się powiększy:



Rysunek 13 - rqt po dekompresji obrazu

6.3 Detekcja Aruco

Teraz skorzystamy z pakietu, który został napisany na potrzeby tej pracy. Projekt, który zawiera pakiety które uprzednio zostały w tej pracy opisane, oraz dodatkowe, z których będziemy korzystać można pobrać, klonując repozytorium gita komendą:

git clone https://github.com/qbaaa-0/rpi_ros2_project.git

cd rpi_ros2_project

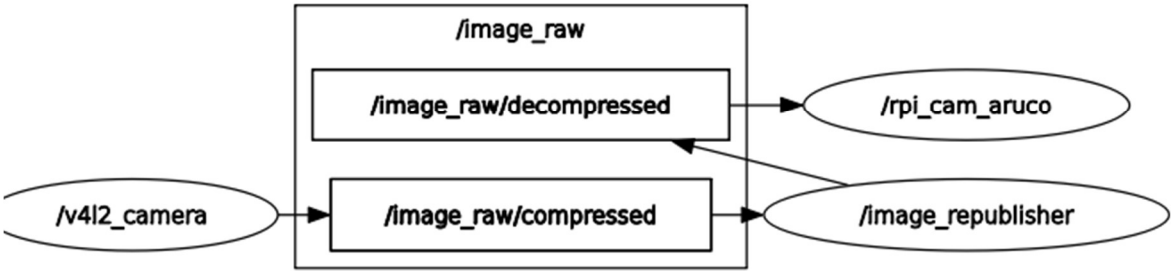
Następnie należy zbudować projekt, użyć skryptu instalacyjnego i włączyć node odpowiadający za detekcję markerów Aruco (użyte przykłady kodów znajdziemy do wydruku w /rpi_ros2_project/src/rpi_cam_handler/aruco_markers/, lub można wygenerować w Internecie korzystając z biblioteki DICT_4x4_50, numery 0 oraz 1).

colcon build

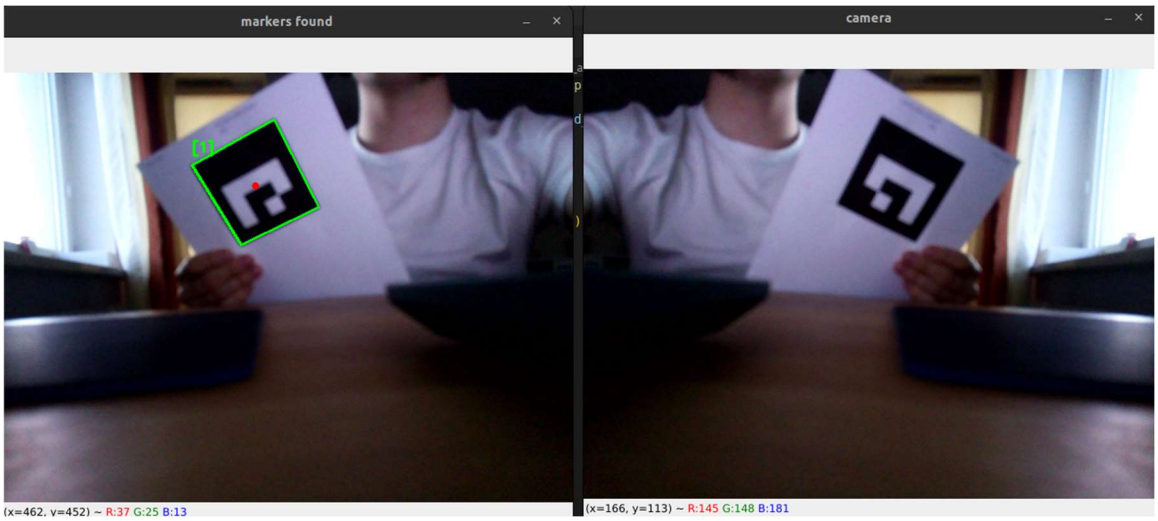
source install/setup.bash

ros2 run rpi_cam_handler rpi_cam_aruco

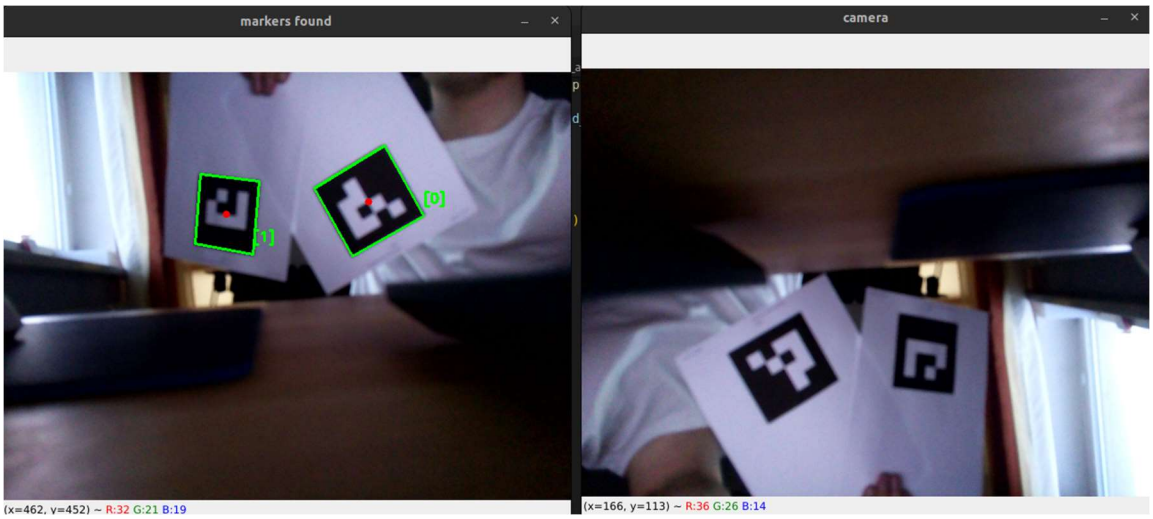
Jeżeli wszystko zadziała, po odświeżeniu node_graph znów się powiększy oraz pokażą nam się dwa dodatkowe okna (camera oraz markers_found). W oknie markers_found pokazane zostaną wykryte markery Aruco, a obraz w oknie camera będzie się obracał horyzontalnie i/lub wertykalnie w zależności od tego czy wykryty będzie marker o ID odpowiednio 0 i 1.



Rysunek 14 - node_graph pełnego systemu



Rysunek 15 - wykrywanie markera Aruco o ID=1, oraz kamera odwrócona wertykalnie



Rysunek 16 - wykrycie obu markerów, oraz obraz kamery podwójnie obrócony

7 Wnioski

Jak widać komunikacja między różnymi platformami jest dość prosta. Instalacja oprogramowania może przynieść nieco problemów, ze względu na kompatybilność różnych wersji systemów z różnymi wersjami programów, jednak efekt końcowy jest warty trudności.

Jako kontynuacje projektu zaproponowałbym budowę autonomicznego robota mobilnego, wykonywującego różne instrukcje w zależności od napotkanego znaku Aruco. Jest to wstęp dla praktycznego zastosowania ROSa w projektach robotów mobilnych i autonomicznych.

Najwięcej problemów przysporzyła mi wiadomość o pozostawieniu przez społeczność ROSa w wersji pierwszej na rzecz kolejnej iteracji w trakcie pisania pracy, jednak aktualizacja systemu nie była aż tak pracochłonna, oraz zmiana środowiska nie wymagała wielkiego nakładu dodatkowej nauki, przez co mogę stwierdzić, że przesiadka z ROS1 na ROS2 dla osoby zaznajomionej z systemem nie powinna sprawić większych problemów.