W pierwszej kolejności koniecznym jest zainstalowanie GitHuba, serwisu umożliwiającego udostępnianie kodu, a następnie sklonowanie odpowiedniego repozytorium [22],

- cd ~
- sudo apt install git
- git clone https://github.com/ArduPilot/ardupilot.git
- cd ardupilot

Następnie należy zainstalować zależności,

- cd ardupilot
- Tools/environment_install/install-prereqs-ubuntu.sh -y
- . ~/.profile

Kolejnym krokiem jest aktualizacja najnowszych plików ArduPilota,

- git checkout Copter-4.2.0
- git submodule update --init --recursive

Sprawdzamy prawidłowe funkcjonowanie zainstalowanych programów,

- cd ~/ardupilot/ArduCopter
- sim_vehicle.py -w

Dalej przechodzimy do instalacji środowiska symulacyjnego Gazebo. Przygotowujemy komputer na instalacje oprogramowania,

- sudo sh -c 'echo "deb http://packages.osrfoundation.org/gazebo/ubuntu-stable `lsb_release -cs` main" > /etc/apt/sources.list.d/gazebo-stable.list'

Następnie ustawiamy klucze,

- wget http://packages.osrfoundation.org/gazebo.key -O - | sudo apt-key add -

Aktualizujemy pakiety,

- sudo apt update

Instalujemy Gazebo,

- sudo apt-get install gazebo11 libgazebo11-dev

Dalej instalujemy wtyczki Gazebo dla APM (ArduPilot Master)

- cd ~
- git clone https://github.com/khancyr/ardupilot_gazebo.git
- cd ardupilot_gazebo
- mkdir build
- cd build
- cmake ..
- make -j4
- sudo make install
- echo 'source /usr/share/gazebo/setup.sh' >> ~/.bashrc

Ustawiamy ścieżkę dla modeli,

- echo 'export GAZEBO_MODEL_PATH=~/ardupilot_gazebo/models' >> ~/.bashrc
- . ~/.bashrc

Następnie instalujemy oprogramowanie ROS z głównej strony internetowej oprogramowania. Przygotowujemy komputer do zaakceptowania pakietów,

- sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'

Ustawiamy klucze,

- sudo apt install curl # if you haven't already installed curl
- curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add –

Następnie przeprowadzamy instalację,

- sudo apt update
- sudo apt install ros-noetic-desktop-full
- source /opt/ros/noetic/setup.bash
- echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
- source ~/.bashrc

Dalej przygotowujemy obszar roboczy Catkin,

- sudo apt-get install python3-wstool python3-rosinstall-generator python3-catkin-lint python3-pip python3-catkin-tools
- pip3 install osrf-pycommon

Inicjalizujemy Obszar roboczy

- mkdir -p ~/catkin_ws/src
- cd ~/catkin_ws
- catkin init

Instalujemy Mavros oraz Mavlink,

- cd ~/catkin_ws
- wstool init ~/catkin_ws/src
- rosinstall_generator --upstream mavros | tee /tmp/mavros.rosinstall
- rosinstall_generator mavlink | tee -a /tmp/mavros.rosinstall
- wstool merge -t src /tmp/mavros.rosinstall
- wstool update -t src
- rosdep install --from-paths src --ignore-src --rosdistro `echo $ROS_DISTRO` -y
- catkin build
- echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
- source ~/.bashrc
- sudo ~/catkin_ws/src/mavros/mavros/scripts/install_geographiclib_datasets.sh

Klonujemy pakiet IQ Simulation ROS oraz tworzymy obszar roboczy Catkin,

- cd ~/catkin_ws/src
- git clone https://github.com/Intelligent-Quads/iq_sim.git
- echo "GAZEBO_MODEL_PATH=${GAZEBO_MODEL_PATH}:$HOME/catkin_ws/src/iq_sim/models" >> ~/.bashrc
- cd ~/catkin_ws
- catkin build
- source ~/.bashrc

Następnie wpisujemy komendę,

- cp ~/catkin_ws/src/iq_sim/scripts/startsitl.sh ~

W celu uruchomienia programów, należy w oddzielne konsole wpisać poniższe komendy,

- roslaunch iq_sim runway.launch
- ~/startsitl.sh
- roslaunch iq_sim apm.launch

Następnie aby przeprowadzić symulacje należy pobrać i rozpakować folder Pliki_Działowski_Szymon_Praca_dyplomowa, a następnie uruchomić oddzielnie (np. poprzez konsolę) pliki get_camera_image.py oraz test.py. Po uruchomieniu okna ukazującego widok z kamery i widocznej zmiany położenia obiektu w programie symulacyjnym Gazebo należy, po ustabilizowaniu pozycji wielowirnikowca, uruchomić program autonomus_landing.py, który wykona automatyczne lądowanie.