Sprawozdanie z Laboratorium Wstępu do Robotyki

Jakub Arnold Postępski

22 marca 2018

1 Wstęp

Celem projektu przebadanie możliwości instalacji klienta ROS Kinetic na kostce zestawu Lego Mindstorms EV3 [1]. Badania na prośbę prowadzących, pragnących zmienić zakres i cel laboratorium.

Studenci w trakcie laboratoriów budują określonego robota. Używają wgrywanego na kartę SD Debiana Wheezy. Wykorzysują bibliotekę Pythona do pisania skryptów sterujących robotem. W pesymistycznym przypadku używane są wszystkie porty zestawu. Połączenie z robotem przez Wi-Fi oraz SSH. Wymagana częstotliwość obsługi zdarzeń to 10 Hz.

Wolny procesor zestawu ARM926EJ-S (architektura ARM, taktowanie 300 MHz, 64 MB RAM) oraz system z karty SD (mała szybkość IO) utrudniają pracę oraz uniemożliwiają swobodną kompilację.

2 Konfiguracja kostki do pracy z ROS

Niemożliwe uruchomienie tzw. mastera ROS ze względu na zasoby sprzętowe. Zastosowano podejście z wygenerowaniem nowego obrazu OS z już zainstalowanymi pakietami [2]. Do kompilacji zainstalowano programem *apt*:

```
unzip
bzip2
apt-utils
build-essential
cmake
initramfs-tools
libboost-all-dev
libboost-dev
libbz2-dev
libc6-dev
libconsole-bridge-dev
libgtest-dev
liblog4cxx10
```

```
liblog4cxx10-dev
liblz4-dev
libtinyxml-dev
libpython2.7-stdlib
libyaml-cpp-dev
libyaml-dev
python-coverage
python-empy
python-imaging
python-mock
python-netifaces
python-nose
python-numpy
python-paramiko
python-pip
python-yaml
```

i skompilowano ręcznie pakiet *sbcl*. Zainstalowano pakiety przy pomocy programu *pip*

```
rosdep
rosinstall_generator
wstool
rosinstall
catkin_pkg
rospkg
```

Do pliku */etc/ros/rosdep/sources.list.d/20-default.list* dodano nowe repozytorium:

```
yaml https://raw.githubusercontent.com/moriarty/ros-ev3/master/ev3dev.yaml
```

I skompilowano pakiet *ros_comm* przy wykorzystaniu *catkin_make_isolated*

```
'rosinstall_generator ros_comm common_msgs --rosdistro kinetic --deps
--wet-only --tar > kinetic-ros_comm-wet.rosinstall'
wstool init src kinetic-ros_comm-wet.rosinstall
'./src/catkin/bin/catkin_make_isolated --install
--install-space /opt/ros/kinetic -DCMAKE_BUILD_TYPE=Release'
```

3 Testy

Po zalogowaniu przez SSH obciążenie procesora na poziomie 5% przez programy oraz 5% przez OS. ROS Master uruchomiony na odpowiednio mocnej maszynie. Po uruchomieniu prostego skryptu Pythona wysyłającego wiadomość na określony temat z częstotliwością 10 Hz obciążenie procesora przez programy na poziomie 20% oraz przez OS na poziomie 60%. Wiadomości nadawane z właściwą

częstotliwością (sprawdzenie po stronie odbiorcy, *rostopic hz*). Po uruchomieniu drugiego procesu nadającego (z tą samą częstotliwością, inny temat) pełne obciążenie procesora (OS na poziomie 80%) oraz obniżenie częstotliwości nadawania. Prosty skrypt z pętlą częstotliwości 10 Hz obsługujący dwa silniki oraz czujnik koloru i podczerwieni (bez obsługi ROSa) obciążał procesor na poziomie 50 %.

4 Wnioski

Negatywny wynik badań. Brak odpowiedniej wydajności sprzętu przy wymaganej częstotliwości, nawet dla prostych programów, przez wysycenie procesora. Brak możliwości optymalizacji OS (niskie obciążenie procesora w stanie bezczynności). Brak możliwości optymalizacji przez przepisanie programów do C++ (biblioteka ROS obciąża procesor głównie przez odwołania systemowe).

5 Zalecenia

- Kompilacja skrośna
- Napisanie serwera który będzie pośrednikiem między ROSem a urządzeniami zestawu i odpowiedniego klienta na mocniejszej maszynie

Literatura

[1] Lego mindstorms ev3. `https://www.lego.com/pl-pl/mindstorms`. Online; accessed 22-March-2018.

[2] Postępski, J. Praca z kontenerami Dockera dla Lego Mindstorms EV3.