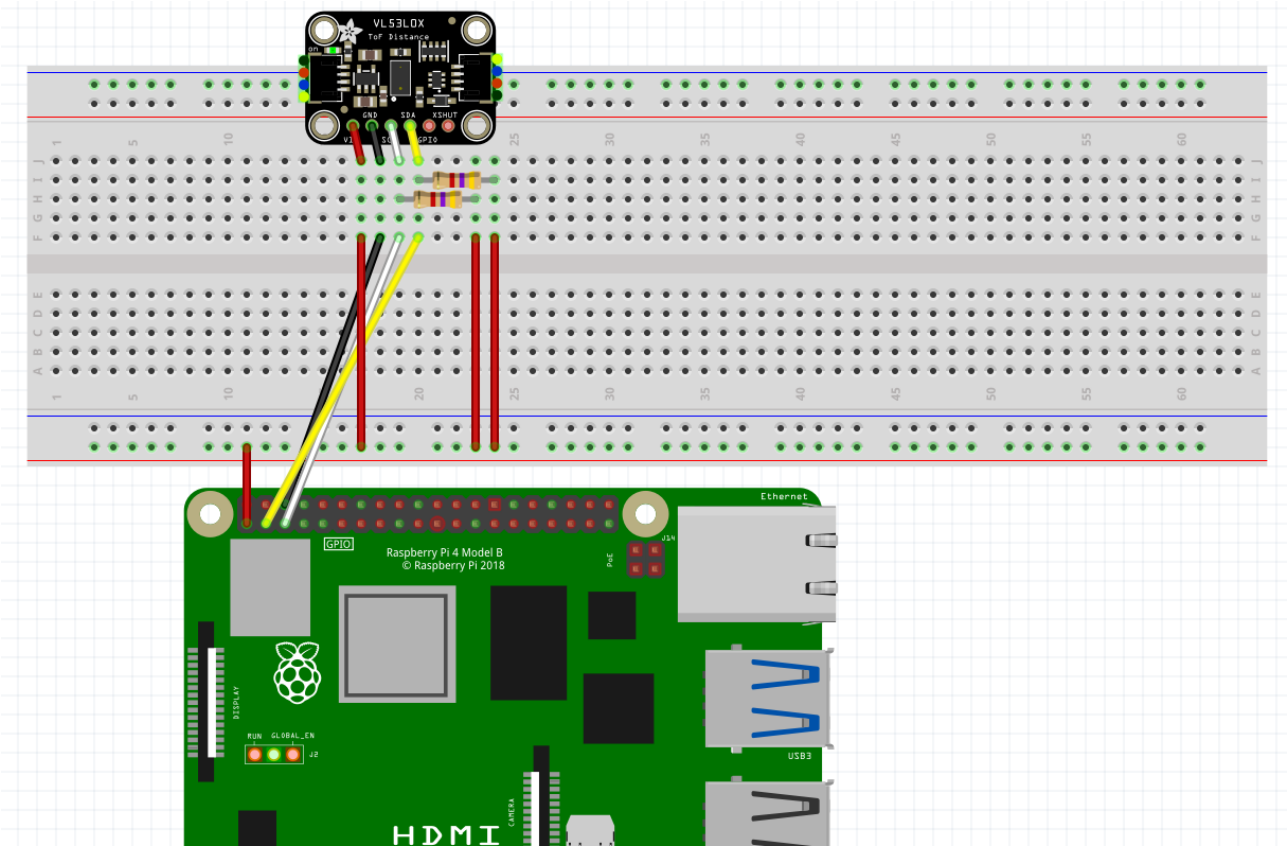
SKPS – projekt: lidar – sprawozdanie

Jakub Proboszcz, Paweł Kochański

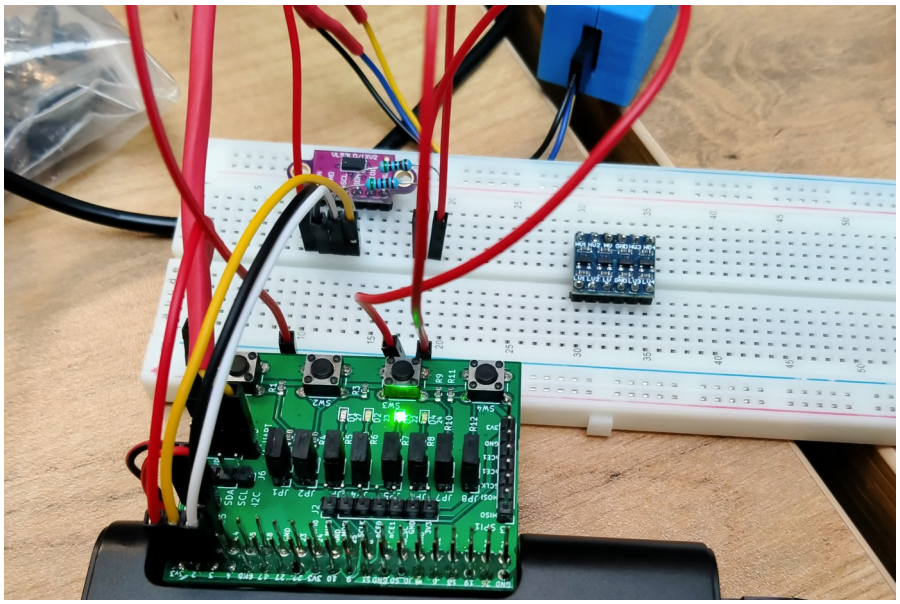
Tematem naszego projektu jest lidar. Na laboratorium 5 zajęliśmy się przygotowaniem do działania urządzeń zewnętrznych potrzebnych do naszego projektu: czujnika odległości Time-of-Flight oraz serwomechanizmu.

Przygotowanie czujnika odległości ToF

W programie Fritzing przygotowaliśmy schemat podłączenia czujnika odległości ToF do Raspberry Pi:



Po zatwierdzeniu przez prowadzącego podłączyliśmy czujnik do Raspberry Pi.



Po potwierdzeniu przez prowadzącego uruchomiliśmy system ratunkowy Buildroot na Raspberry Pi, z podłączonym do niego czujnikiem odległości.

```
OK
Starting dropbear sshd: [   12.065065] NET: Registered protocol family 10
[   12.071558] Segment Routing with IPv6
OK
[   14.661048] random: crng init done
ssh-keygen: generating new host keys: RSA
DSA ECDSA ED25519
Starting sshd: OK

Welcome to Buildroot rescue OS
rescue login: root
```

Zamontowaliśmy partycję boot oraz zedytowaliśmy plik config.txt:

```
# mkdir /mnt/boot
# mount /dev/mmcblk0p1 /mnt/boot
[  110.053989] FAT-fs (mmcblk0p1): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.
# cd /mnt/boot/
# ls
config.txt  fixup4.dat  overlays    rescue      start4.elf  user
# vi config.txt
```

```
[all]
dtoverlay=i2c1
```

Po dopisaniu powyższej linijki zresetowaliśmy Raspberry Pi trzymając przycisk SW4 – przywitał nas system OpenWRT:

```
[    9.251039] bcmgenet fd580000.ethernet eth0: Link is Up - 100Mbps/Full - flow
 control rx/tx
[    9.260271] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready



BusyBox v1.33.1 (2021-10-24 09:01:35 UTC) built-in shell (ash)

  _______                     ________        __
 |       |.-----.-----.-----.|  |  |  |.----.|  |_
 |   -   ||  _  |  -__|     ||  |  |  ||   _||   _|
 |_______||   __|_____|__|__||________||__|  |____|
          |__| W I R E L E S S   F R E E D O M
 -----------------------------------------------------
 OpenWrt 21.02.1, r16325-88151b8303
 -----------------------------------------------------
=== WARNING! =====================================
There is no root password defined on this device!
Use the "passwd" command to set up a new password
in order to prevent unauthorized SSH logins.
```

Żeby umożliwić obsługę urządzeń przez I2C, w tym czujnika odległości ToF, dodatkowo zainstalowaliśmy pakiet kmod-i2c-bcm2835:

```
root@OpenWrt:~# opkg install kmod-i2c-bcm2835
Installing kmod-i2c-bcm2835 (5.4.154-1) to root...
Downloading https://downloads.openwrt.org/releases/21.02.1/targets/bcm27xx/bcm27
11/packages/kmod-i2c-bcm2835_5.4.154-1_aarch64_cortex-a72.ipk
Installing kmod-i2c-core (5.4.154-1) to root...
Downloading https://downloads.openwrt.org/releases/21.02.1/targets/bcm27xx/bcm27
11/packages/kmod-i2c-core_5.4.154-1_aarch64_cortex-a72.ipk
Configuring kmod-i2c-core.
[  242.888179] kmodloader: loading kernel modules from /etc/modules.d/*
[  242.897052] i2c /dev entries driver
[  242.904168] kmodloader: done loading kernel modules from /etc/modules.d/*
Configuring kmod-i2c-bcm2835.
[  242.940192] kmodloader: loading kernel modules from /etc/modules.d/*
[  242.948346] kmodloader: done loading kernel modules from /etc/modules.d/*
root@OpenWrt:~#
```

W celu uzyskania programu i2c-detect pobraliśmy pakiet i2c-tools:

```
root@OpenWrt:/# opkg install i2c-tools
Installing i2c-tools (4.3-1) to root...
Downloading https://downloads.openwrt.org/releases/21.02.1/packages/aarch64_cort
ex-a72/packages/i2c-tools_4.3-1_aarch64_cortex-a72.ipk
Installing libi2c (4.3-1) to root...
Downloading https://downloads.openwrt.org/releases/21.02.1/packages/aarch64_cort
ex-a72/packages/libi2c_4.3-1_aarch64_cortex-a72.ipk
Configuring libi2c.
Configuring i2c-tools.
```

Uruchomienie programu i2c-detect potwierdza, że czujnik odległości jest podłączony i wykrywany poprawnie:

```
root@OpenWrt:/# i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                         -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- 29 -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

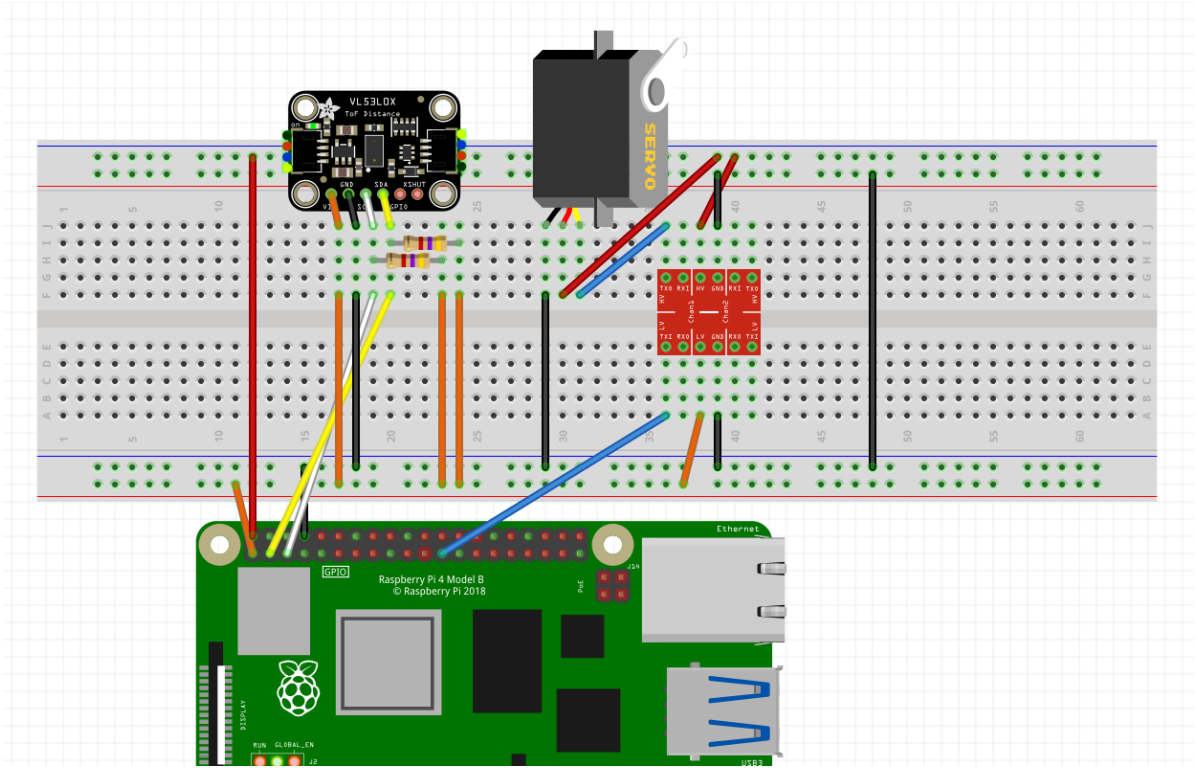
Następnie pobraliśmy biblioteki VL53L0X_1.0.4 oraz VL53L0X_rasp-master służące do obsługi czujnika odległości. Zamieściliśmy je w pakiecie OpenWRT o nazwie lidar; poza nimi zawiera on też napisany przez nas Makefile.
Z powodu błędów redefinicji typu, zakomentowaliśmy 2 linijki o treści
„typedef unsigned long long uint64_t;”
w dwóch plikach vl53l0x_types.h zarówno w oryginalnym API VL53L0X_1.0.4, jak i w bibliotece dla Raspberry Pi VL53L0X_rasp-master. Ta zmiana wystarczyła, żeby pakiet się skompilował.

Uruchomienie zainstalowanego pakietu na OpenWRT potwierdza, że czujnik odległości ToF działa.

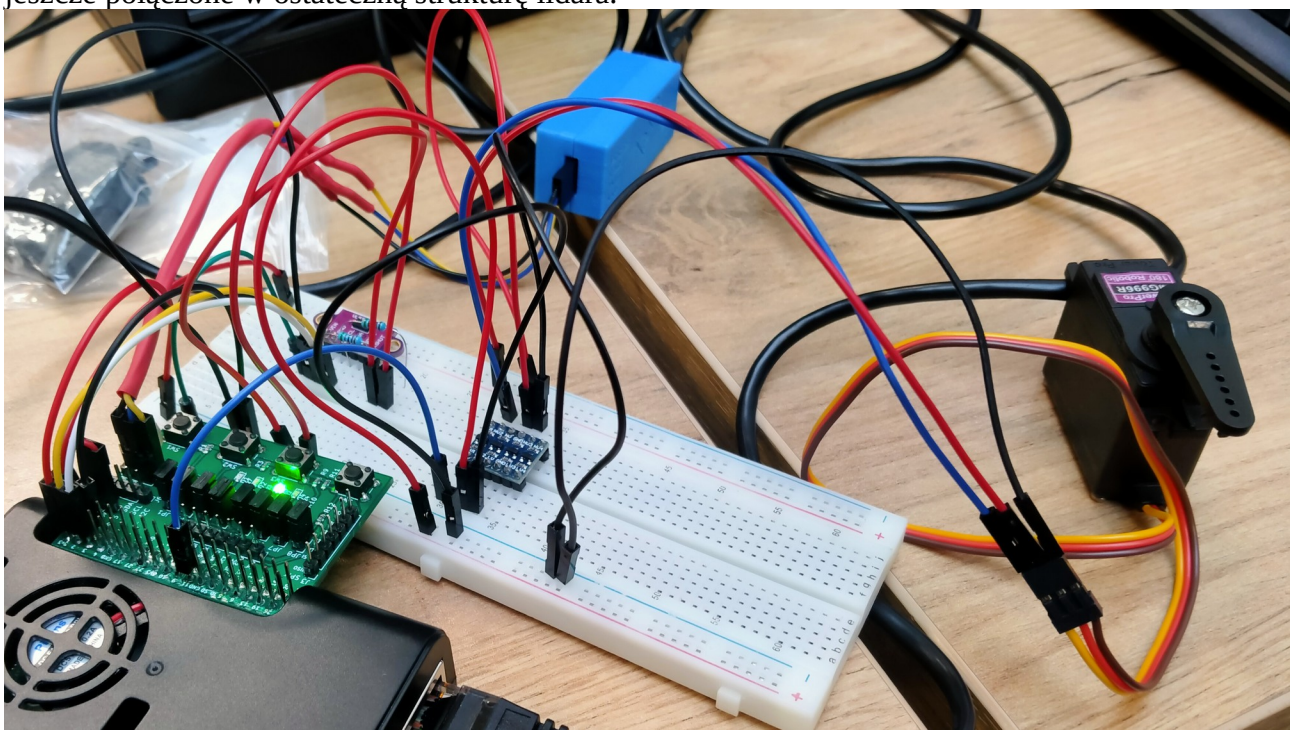
```
In loop measurement 582: 104
In loop measurement 583: 103
In loop measurement 584: 109
In loop measurement 585: 114
In loop measurement 586: 113
In loop measurement 587: 112
In loop measurement 588: 118
In loop measurement 589: 121
In loop measurement 590: 119
In loop measurement 591: 118
In loop measurement 592: 123
In loop measurement 593: 121
In loop measurement 594: 125
```

Przygotowanie serwomechanizmu

W programie Fritzing przygotowaliśmy schemat podłączenia serwomechanizmu razem z czujnikiem odległości ToF do Raspberry Pi. To będzie ostateczny schemat podłączeń w naszym projekcie.



Po weryfikacji przez prowadzącego podłączyliśmy urządzenia zgodnie ze schematem. Nie były jeszcze połączone w ostateczną strukturę lidara.



Po ponownej weryfikacji połączeń uruchomiliśmy system OpenWRT.

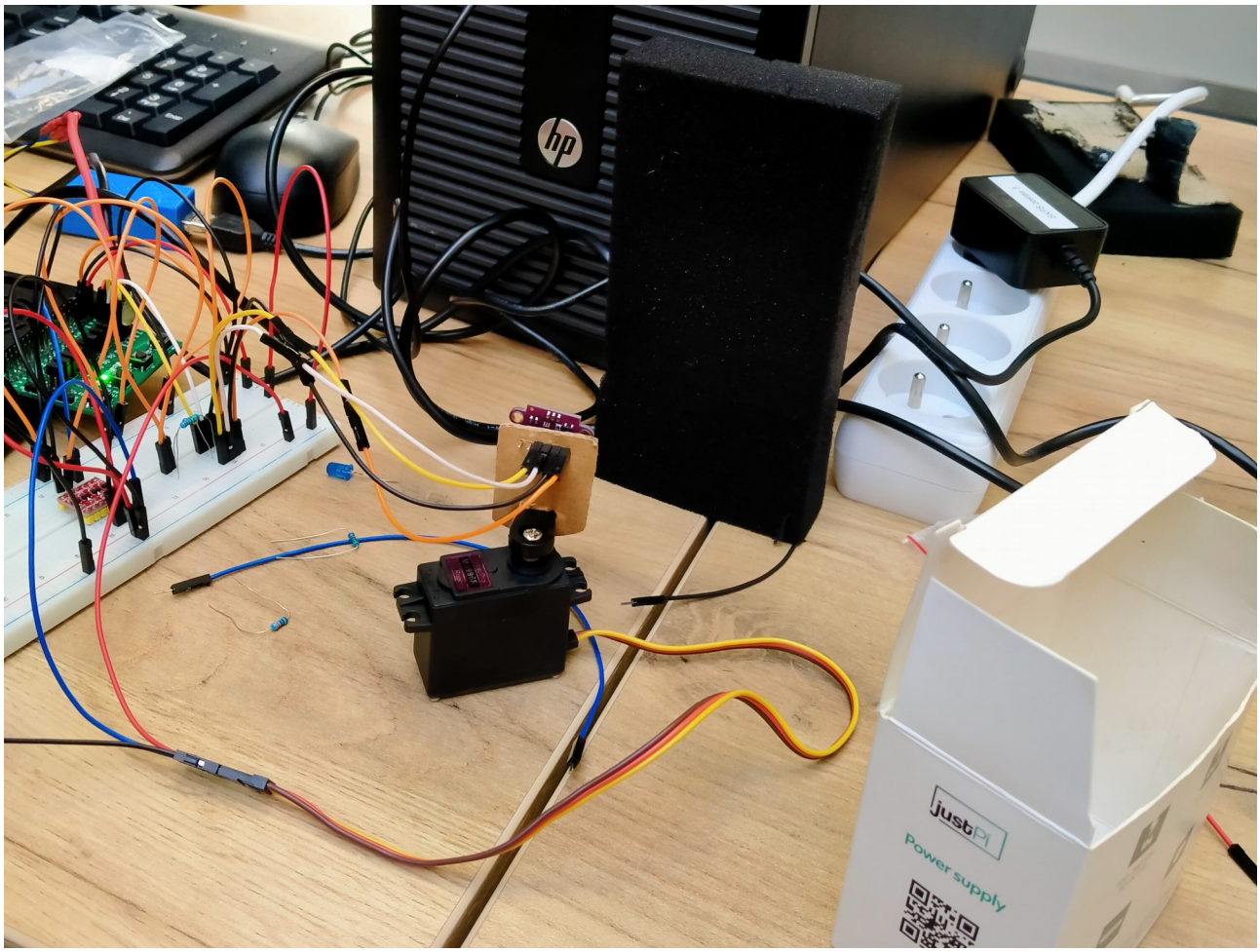
Przygotowaliśmy prosty program, servo, którego wywołanie z argumentem będącym liczbą zmiennoprzecinkową generuje sygnał PWM o okresie 20 ms przez około 5 sekund. Sygnał ten ma wartość 1 przez tyle milisekund w okresie, ile podano jako argument. Wysłanie sygnału o

odpowiednim wypełnieniu powoduje ustawienie się serwomechanizmu na odpowiednią pozycję, gdzie około 0,5 lub 2,5 ms sygnału o wartości 1 (w okresie) to wartości skrajne.
Program ten został zmodyfikowany do postaci trzech funkcji w pliku servo.c w kodzie właściwego programu lidar.

Program ten dodaliśmy do uprzednio przygotowanego pakietu lidar, skompilowaliśmy i uruchomiliśmy. Serwomechanizm obracał się poprawnie.

Laboratorium 6 – finalizacja projektu

Przed laboratorium 6 przygotowaliśmy program do wizualizacji pomiarów lidara – jego opis jest w oddzielnym pliku .pdf.
Na laboratorium 6 podłączyliśmy czujnik odległości ToF oraz serwomechanizm tak, jak na poprzednim schemacie (problemy z działaniem serwomechanizmu wymagały niewielkich modyfikacji okablowania). Zamontowaliśmy czujnik odległości na serwomechanizmie.



Przygotowaliśmy program lidar w ramach środowiska biblioteki VL53L0X_rasp-master.
Zmodyfikowaliśmy Makefile tej biblioteki, żeby kompilował nasz program, analogicznie do przykładów.
Powstały pakiet skompilowaliśmy i przenieśliśmy na OpenWRT:
dodaliśmy lokalizację pakietu do feeds.conf.default:

```
 GNU nano 2.9.3                feeds.conf.default                Zmodyfikowany

src-git base https://git.openwrt.org/openwrt/openwrt.git;v22.03.3
src-git-full packages https://git.openwrt.org/feed/packages.git^2048c5bbf6c482e$
src-git-full luci https://git.openwrt.org/project/luci.git^396f4048bd1f4cae7cb6$
src-git-full routing https://git.openwrt.org/feed/routing.git^1a87333f268bcf0a1$
src-git-full telephony https://git.openwrt.org/feed/telephony.git^49abbb97e113c$
src-link lidar /home/user/Pulpit/skps23l_jproboszcz_pkochanski/projekt/
```

przygotowaliśmy zmienną środowiskową:

```
user@lab-6:~/Pulpit/skps/sdk$ export LANG=C
```

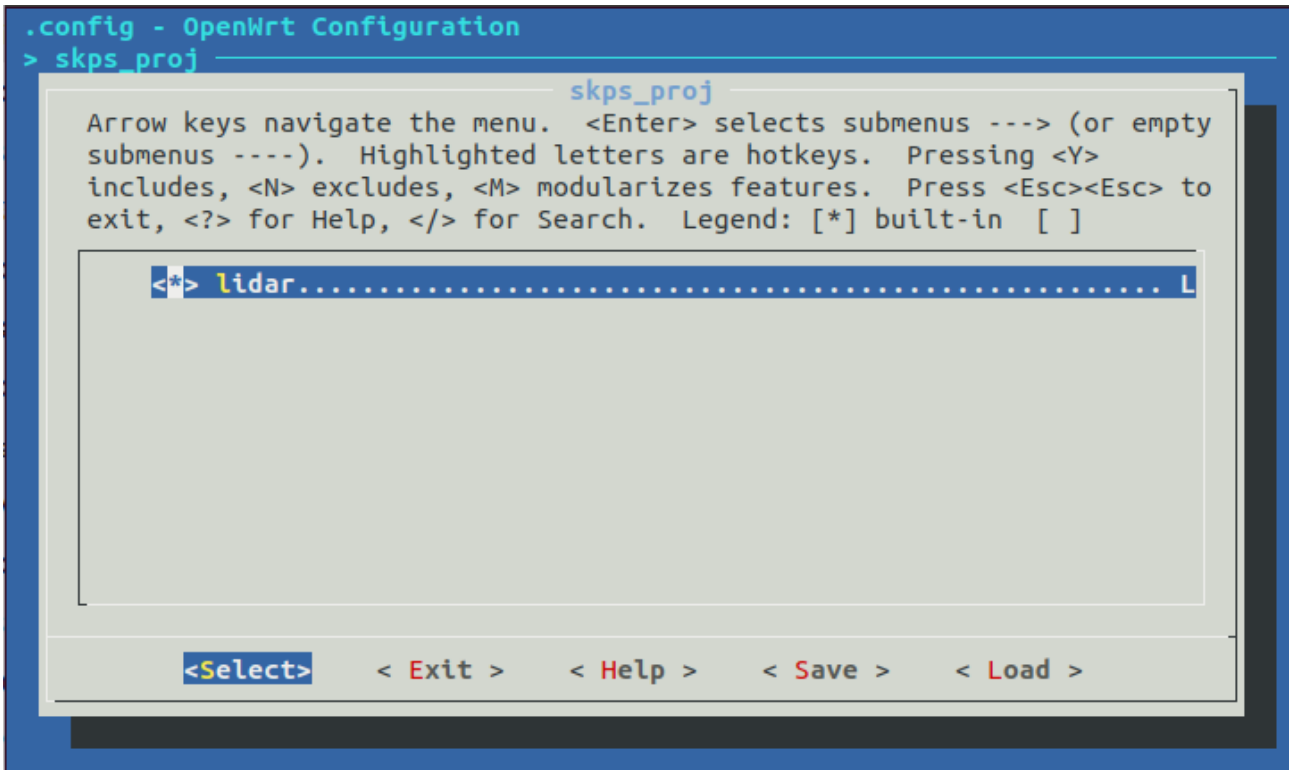
zainstalowaliśmy pakiet w SDK:

```
user@lab-6:~/Pulpit/skps/sdk$ scripts/feeds update -a
Updating feed 'base' from 'https://git.openwrt.org/openwrt/openwrt.git;v22.03.3'
 ...
Cloning into './feeds/base'...
remote: Enumerating objects: 9232, done.
remote: Counting objects: 100% (9232/9232), done.
remote: Compressing objects: 100% (7733/7733), done.
remote: Total 9232 (delta 1430), reused 4814 (delta 773), pack-reused 0
Receiving objects: 100% (9232/9232), 9.58 MiB | 8.86 MiB/s, done.
Resolving deltas: 100% (1430/1430), done.
Note: checking out '221fbfa2d854ccb6cd003c065ec308fbc0651b11'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.
```

```
user@lab-6:~/Pulpit/skps/sdk$ scripts/feeds install -p lidar -a
Collecting package info: done
Collecting target info: done
WARNING: Makefile 'package/kernel/linux/Makefile' has a dependency on 'eip197-mi
ni-firmware', which does not exist
WARNING: Makefile 'package/kernel/linux/Makefile' has a dependency on 'r8169-fir
mware', which does not exist
```

zaznaczyliśmy go do kompilacji



i skompilowaliśmy:

```
user@lab-6:~/Pulpit/skps/sdk$ make package/lidar/compile -j1 V=s
WARNING: Makefile 'package/kernel/linux/Makefile' has a dependency on 'eip197-mini-firmware', which does not exist
WARNING: Makefile 'package/kernel/linux/Makefile' has a dependency on 'r8169-firmware', which does not exist
WARNING: Makefile 'package/kernel/linux/Makefile' has a dependency on 'e100-firmware', which does not exist
WARNING: Makefile 'package/kernel/linux/Makefile' has a dependency on 'bnx2-firmware', which does not exist
WARNING: Makefile 'package/kernel/linux/Makefile' has a dependency on 'bnx2x-firmware', which does not exist
```

uruchomiliśmy serwer HTTP w katalogu, w którym powstał plik .ipk:

```
user@lab-6:~/Pulpit/skps/sdk/bin/packages/aarch64_cortex-a72/lidar$ python3 -m h
ttp.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

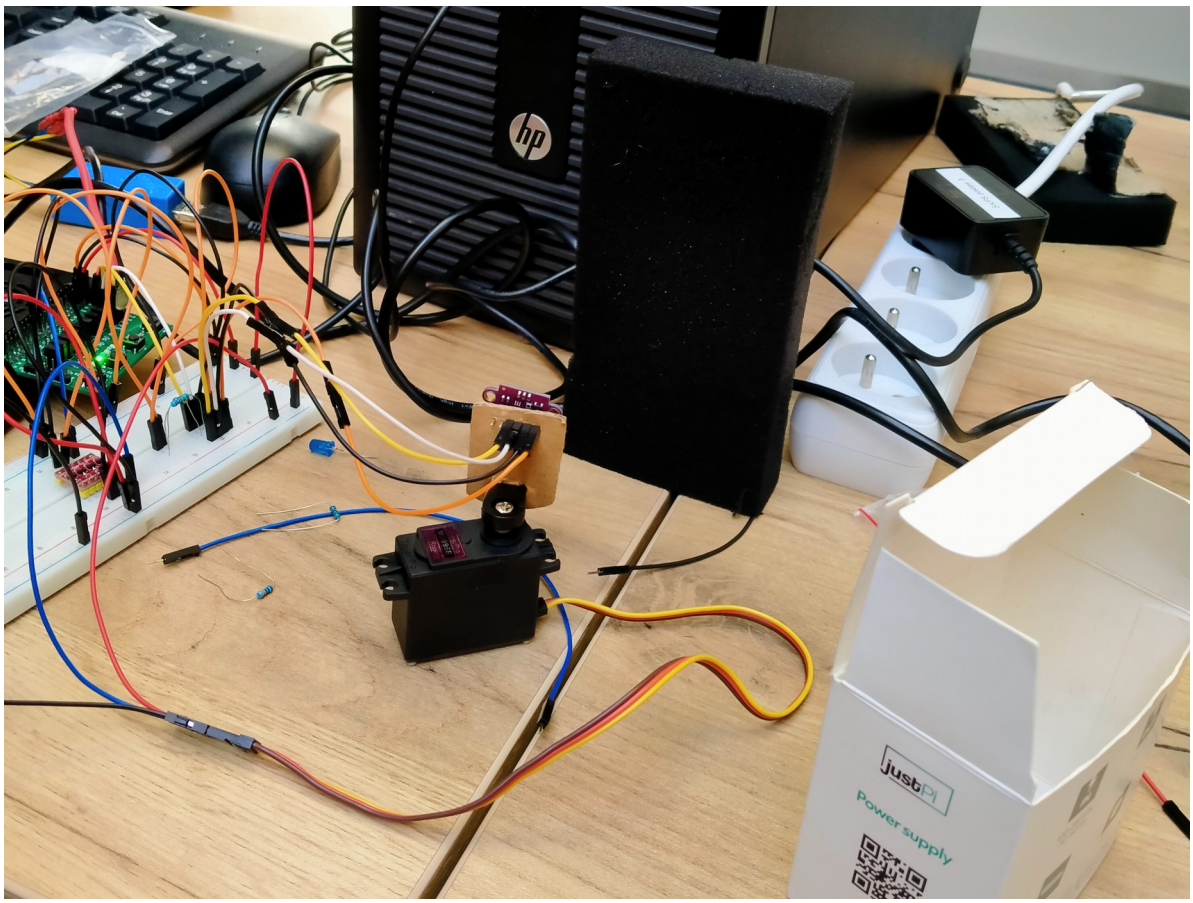
Następującymi komendami zainstalowaliśmy pakiet na OpenWRT:

```
wget http://10.42.0.1:8000/lidar_1.0-1_aarch64_cortex-a72.ipk
opkg install lidar_1.0-1_aarch64_cortex-a72.ipk --force-reinstall
```

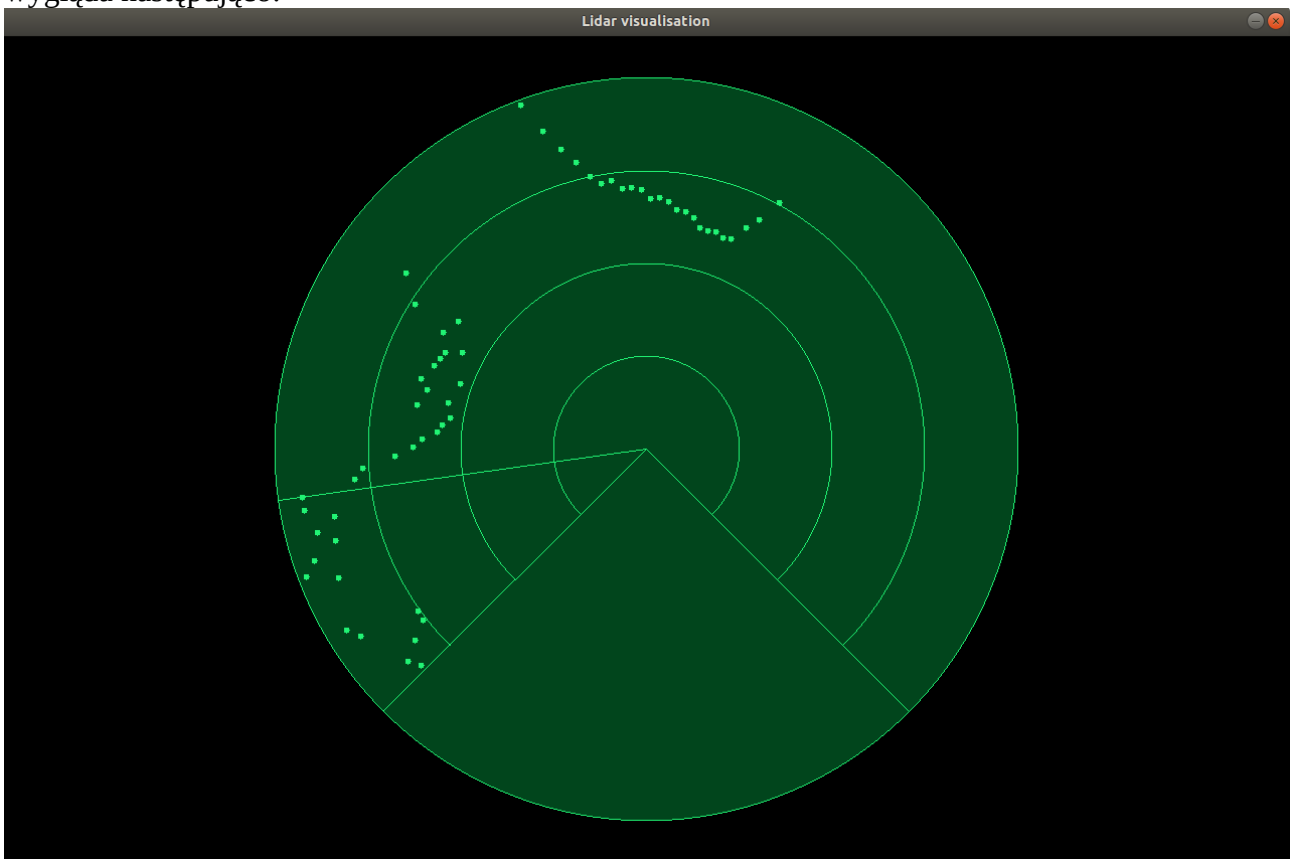
Uruchomienie programu lidar na OpenWRT wykonuje następujące akcje:
- inicjalizuje pomiar odległości i obsługę serwomechanizmu;
- przesuwa serwomechanizm na pozycję początkową (czasem trzeba mu pomóc, jeżeli serwomechanizm szwankuje);
- wykonuje pomiary, dopóki udaje mu się wysyłać dane do programu visual uruchomionego na host.

Każdy pomiar składa się z przesunięcia serwomechanizmu na następną pozycję, pomiaru odległości i przesłania informacji na host.
Pomiary są wykonywane dopóty, dopóki pomiar odległości działa i udaje się przesłać wyniki.
Program powinien być zatrzymywany przez zamknięcie okna wizualizacji (program visual powinien być uruchomiony na host przed uruchomieniem programu lidar na OpenWRT).

Pomiar otoczenia wokół lidara przedstawionego na tym zdjęciu:



wygląda następująco:



Na wizualizacji widać (od prawej): opakowanie zasilacza, gąbkę i niedokładne pomiary kabli.

Kod programu jest w katalogu biblioteki VL53L0X_rasp-master, dokładnie w katalogu lidar/src/VL53L0X_rasp-master/lidar/. Plik main.c to zmodyfikowany przykład pomiaru odległości; poza wypisywaniem na ekran pomiarów dodaliśmy ich wysyłanie (plik network.c, zaadaptowany z generatora syntetycznych danych z pracy domowej) i przesuwanie serwomechanizmu (plik servo.c).