Projekt

Systemy komputerowe: architektura i programowanie
2022L

Jakub Romanek (311065)

2 czerwca 2022

Spis treści

1 Przygotowanie środowiska do pracy

Projekt był wykonywany na maszynie wirtualnej, do której dostęp był przyznawany przez System Rezerwacji zasobów. Wszystkie pliki z projektu są umieszczone w zdalnym repozytorium, z racji tego, że maszyna była dostępna przez 2h.

2 Specyfikacja modułu gpioemu.v z laboratorium nr 2

Smartwatch wyposażony jest w wiele czujników środowiskowych (np. czujnik tętna, żyroskop, czujnik oświetlenia itp.) i komunikacyjnych (np. WiFi, Bluetooth, wyświetlacz), których obsługa realizowana jest przez dedykowane moduły peryferyjne wbudowane w strukturę krzemową procesora. Układy te dołączone są do magistrali systemowej, która charakteryzuje się 16-bitową przestrzenią adresową, oraz odrębnymi 32-bitowymi magistralami danych - wejściową i wyjściową. Niezakłóconą współpracę wszystkich układów peryferyjnych zapewniają odpowiednio skonfigurowane dekodery adresów, które umieszczają poszczególne peryferia w przestrzeni adresowej w sposób wykluczający kolizje (tzn. pod jednym adresem może znajdować się tylko jedno peryferium). Wśród nich znajduje się moduł prostego 2-osiowego akcelerometru, który za pomocą kodu 4-bitowego sygnalizuje zmierzone wartości przeciążeń (osobno dla każdej osi). Ponadto, dla każdej osi można niezależnie ustawić czułość w zakresie 0...15. Twoim zadaniem jest zaprojektowanie Dekodera Adresów, który umieści dwa 4-bitowe porty czujnika w przestrzeni adresowej procesora zgodnie z podaną specyfikacją:

- Adresy: 0xYc90 (oś 1), 0xYe90 (oś 2) (Y oznacza dowolną cyfrę szesnastkową)
- Przesunięcie bitowe dla osi 1: 9
- Przesunięcie bitowe dla osi 2: 21
- Obsługa osi 1 dołączona jest do 4 najmłodszych bitów 8-bitowego portu GPIO, natomiast oś 2 dołączona jest do 4 najstarszych bitów.

Dodatkowym układem w projektowanym urządzeniu jest wbudowany licznik o parametrach:

- Licznik 8-bitowy, zliczający w dół od wartości początkowej 0x56,
- Po osiągnięciu wartości minimalnej generuje sygnał INT. Sygnał INT (jako bit statusowy) dostępny jest pod adresem 0xYea0 z przesunięciem 5 bitów,
- Globalny reset powoduje rozpoczęcie zliczenia od wartości początkowej oraz wyzerowanie sygnału INT,
- Zapis jedynki logicznej z przesunięciem 5 bitów pod adres 0xYea0 powoduje rozpoczęcie zliczania od wartości początkowej.

Odczyt i zapis dla czujników oraz zegara odbywa się przez ten sam adres.

Musisz zadbać o to, aby Twój moduł reagował w poprawny sposób przy próbie odczytu/zapisu z/pod adresów wskazanych w specyfikacji. Nie mniej ważne jest, aby Twój układ peryferyjny nie przeszkadzał w komunikacji z innymi modułami podłączonymi do magistrali. Oznacza to, że w przypadku, gdy na magistrali pojawiają się odczyty lub zapisy pod inne adresy niż wynikające ze specyfikacji, Twój moduł nie może na nie reagować, tzn. nie może zmieniać wyjść (przy próbach zapisu) oraz musi zapewnić stan neutralny dla działania magistrali, tj. dostęp innych peryferiów (przy próbach odczytu).

3 Specyfikacja zadania projektowego

Systemy komputerowe: architektura i oprogramowanie (SYKOM)

Projekt

Politechnika Warszawska, Instytut Telekomunikacji

Prowadzący: Aleksander Pruszkowski

Organizacja projektu:

- Osoba dla której przygotowano ten dokument: Romanek Jakub Piotr

Przed przystąpiniem do realizacji zadania projektowego należy: Podobnie jak w zajeciach laboratoryjnych, pobrać plik z rozszerzeniem OVPN z serwera WWW używając adresu `https://resrepo.tele.pw.edu.pl` i otrzymanych poprzez email danych do zalogowania się do tego serwera WWW. Zarezerwowac sobie tzw. wirtualny komputer za pomoca serwera WWW i adresu `http://zsutresv.tele.pw.edu.pl/ResourceReservation`. Dla potrzeb zajęć projektowych rezerwacja jest dokonywana wyłącznie w slotach 1h lub 2h. Rezerwować można jednak ponawiać wielokrotnie w semestrze.
Przeczytać ze zrozumieniem wszystkie dokumenty wpowadzające do laboratorów i projektu.

Oczekiwane wyniki pracy: Zgodnie z opisem wprowadzającym `SYKOM_proj.pdf` proszę utworzyć:

- verilogowy moduł GpioEmu, który ma działać zgodnie ze specyfikacją otrzymną w ramach zajęć lab.2. Proszę zwrócić uwagę, że dla modułu GpioEmu należy zgodnie z zdobytą wiedzą dostarczyć także testy potwierdzające jego działanie - i takowe pliki testowe umieścić na przydzielonym sobie repozytorium GIT podobnie jak inne produkty pracy nad tym projektem,
- moduł jądra systemu Linux komunikujący moduł GpioEmu z aplikacją użytkownika, tu także dla testów wymagne jest wytworzenie plików: Image, rootfs.ext2 - plików tych jednak nie należy wrzucać na zdalne repozytorium GIT,
- aplikację użytkownika testującą poprawne działanie całego systemu, aplikacja ta ma być podczas testów wbudowana w docelowy rootfs.ext2.

Nawiązujac do dokumnetu wprowadzającego (`SYKOM_proj.pdf`) adresy portów GpioEmu widoczne przez CPU powiny być następujące:
$SYKT\_GPIO\_ADDR\_SPACE$: ustalony na podstawie konfiguracji wewnętrznej emulatora QEMU (odkrywany przez odpowiednie urzycie narzędzia DTS, zgodnie z procedurami poznanymi w ramach lab.1), adres ten jest globalnym offsetem w 32 bitowej przestrzeni w której umieszczono podprzestrzeń adresowaną 16 bitowo, używaną w specyfikacji zadania dla lab.2 dla określonych tam elementów (np.: czujniki elementy sterujące modułem verilogowym). W zadaniu projektowym nadaj zdefionwanym w specyfikacji lab.2 kolejnym elementom następujące lokacje: 0x210, 0x214, 0x218, 0x21C, 0x220. Gdyby z Twojej analizy treści zadania otrzymanego w ramach lab.2, wynikało, że dostępnych jest więcej elementów, je także użyj, nadając im kolejne lokacje: 0x224, 0x228, 0x22C. Swoją motywację do takiej modyfikacji przedstaw jednak w formie uzasadnienia w raporcie.
Utworzony w ramach zadania projektowego moduł jądra systemu Linux, niech w wyniku swojej pracy umożliwia odwołania do powyższych elementów poprzez następujące pliki w tzw. PROC-FS:
$/proc/sykom/rj9del1$ - odowłania do elementu pod adresem 0x210,
$/proc/sykom/rj9del2$ - odowłania do elementu pod adresem 0x214,
$/proc/sykom/rj9del3$ - odowłania do elementu pod adresem 0x218,
$/proc/sykom/rj9del4$ - odowłania do elementu pod adresem 0x21C,
$/proc/sykom/rj9del5$ - odowłania do elementu pod adresem 0x220,
$/proc/sykom/rj9del6$ - odowłania do elementu pod adresem 0x224,
$/proc/sykom/rj9del7$ - odowłania do elementu pod adresem 0x228,
$/proc/sykom/rj9del8$ - odowłania do elementu pod adresem 0x22C.
Dane przekazywane między aplikacją użytkownika a plikami PROC-FS czyli jądrem systemu niech będą w reprezentacji: HEX.

Zawartość raportu: Raport powienien ukazywać na zamieszczonych w nim obrazkach (tzw. screen'y) działanie systemu w różnych a zarazem ważnych(!) i sensownie wybranych chwilach - sensowność doboru tych obrazków także będzie oceniana, jest ona dowodem, że autor jest pewien poprawności działania utworzonego przez siebie systemu. Dla uniknięcia nieporozumień w raporcie zacytuj przekazaną Tobie treść zadania z laboratorium 2. Raport proszę utworzyć w dowolnym edytorze tekstowym, ale po jego przygotowaniu należy raport taki skonwertować do formatu PDF. Żadne inne formaty dokumentów elektornicznych np.: DOC, DOCX, ... nie będą przyjmowane. Fianlnie raport oraz wszelkie pliki źródłowe będące wynikiem prac nad projektem proszę umieść w przydzielonym Tobie indywidualnym repozytorium GIT w jego katalogu projektowym - z tego (i tylko z tego) miejsca prowadzacy będzie pobierał te pliki do późniejszego ocenienia i wystawienia oceny.

Uwaga! Proszę nie umieszczać w przydzielonym repozytorium GIT plików generowanych automatycznie, czyli: qemu-system-riscv32-sykt, Image, rootfs.ext2, natomiast zadbać aby do tego repozytorium trafiały wyłącznie ważne pliki wytworzone przez Ciebie a nie elementy wygenerowane innymi narzędziami.

4 Wymagania dotyczące adresacji

We wszystkich komponentach odczyt i zapis odbywa się pod ten sam adres. W projekcie zostały wyróżnione dwie osie oraz licznik. Ich adresacja jest następująca:

- Oś 1: jest dostępna pod adresem 0x210, tworzy 4 najmłodsze bity portu GPIO, jej wartość przesunięta jest o 9 bitów w lewo
- Oś 2: jest dostępna pod adresem 0x214, tworzy 4 najstarsze bity portu GPIO, jej wartość przesunięta jest o 21 bitów w lewo
- Licznik: jest dostępny pod adresem 0x218, zlicza w dół od wartości początkowej 0x56, po osiągnięciu wartości minimalnej zmienia sygnał INT z 0 na 1, po odczytaniu zwraca wartość sygnału INT przesuniętą o 5 bitów w lewo, zapis jedynki logicznej z przesunięciem 5 bitów powoduje reset licznika i zmianę sygnału INT na 0.

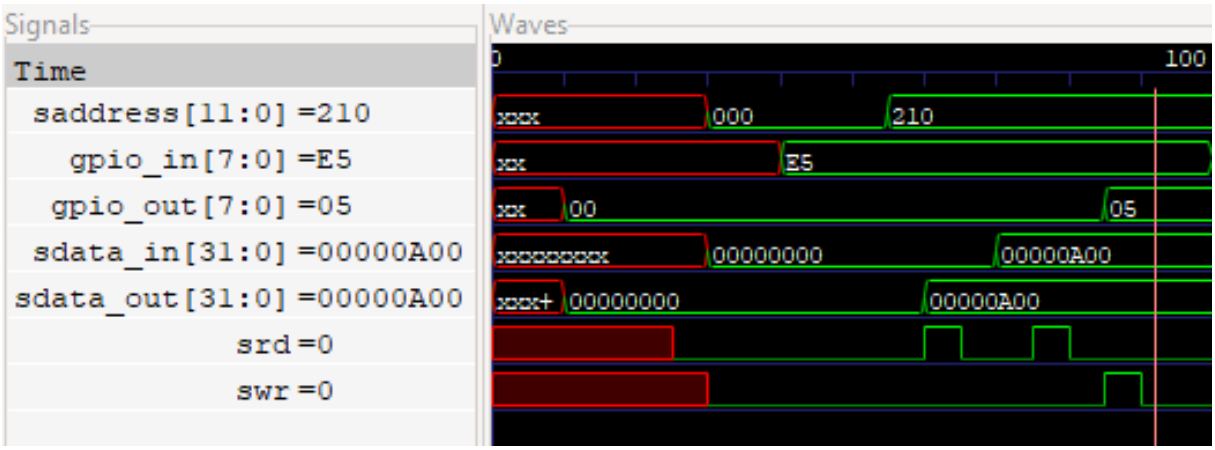
Wartości hexadecymalne po odpowiednich przesunięciach zostały umieszczone w tabeli, w celu potwierdzenia działania późniejszych testów.

Wartość	Oś 1 (przesunięcie o 9 w lewo)	Oś 2 (przesunięcie o 21 w lewo)
0x00	0x0	0x0
0x01	0x200	0x200000
0x02	0x400	0x400000
0x03	0x600	0x600000
0x04	0x800	0x800000
0x05	0xa00	0xa00000
0x06	0xc00	0xc00000
0x07	0xe00	0xe00000
0x08	0x1000	0x1000000
0x09	0x1200	0x1200000
0x0a	0x1400	0x1400000
0x0b	0x1600	0x1600000
0x0c	0x1800	0x1800000
0x0d	0x1a00	0x1a00000
0x0e	0x1c00	0x1c00000
0x0f	0x1e00	0x1e00000

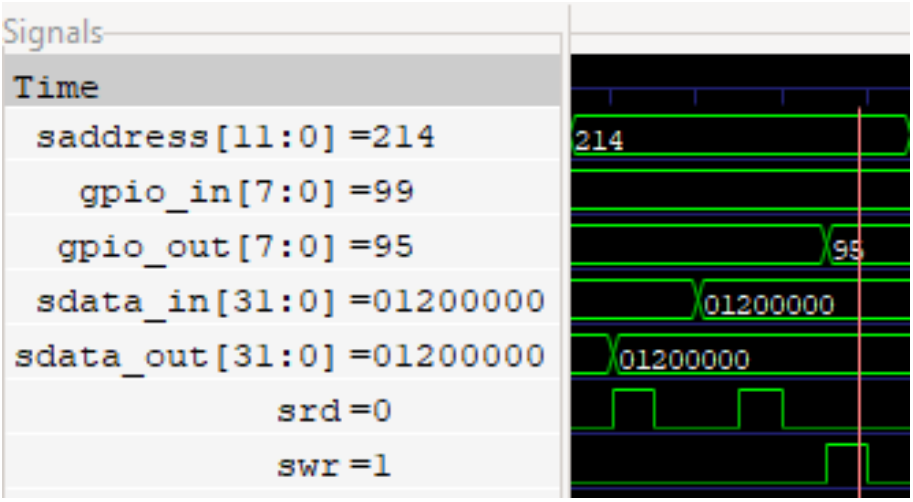
Wartość sygnału INT równego 1, przesuniętego o 5 bitów w lewo daje wartość 0x20.

5 Testbench

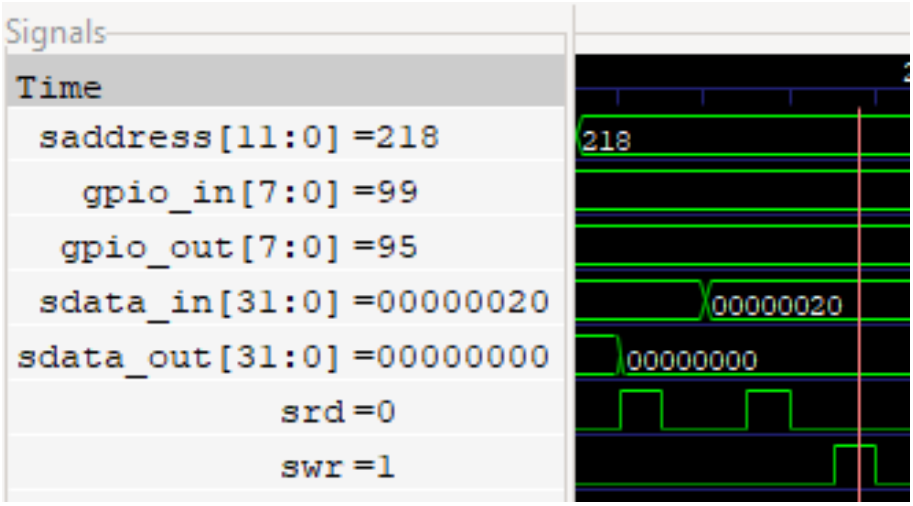
W ramach laboratorium nr 2 został przygotowany testbench. W tym projekcie rejestr adresów został skrócony do 12 bitów - tak, aby zgadzał się z podaną specyfikacją. Poniżej zaprezentowane są testy modułu gpioemu.v.



Rys. 1: Test osi 1, 0x05 « 9 daje 0xA00



Rys. 2: Test osi 2, 0x9 « 21 daje 0x1200000



Rys. 3: Test sygnału INT, 0x1 « 5 daje 0x20

6 Moduł jądra Linux

W tym projekcie został napisany moduł jądra Linux o nazwie `kernel_module.c`. Za pomocą odpowiedniej biblioteki moduł w katalogu *proc* tworzy folder *sykom*, w którym to znajdują się pliki PROC-FS, dzięki którym można odwoływać się do poszczególnych komponentów modułu gpioemu. Za pomocą pliku */proc/sykom/rj9del1* można odwołać się do osi 1, za pomocą pliku */proc/sykom/rj9del2* można odwołać się do osi 2, natomiast za pomocą pliku */proc/sykom/rj9del3* można odwołać się do modułu licznika.

Przed przystąpieniem do uruchamiania aplikacji demonstracyjnej, należało sprawdzić czy nowy system będzie poprawnie działał z nowym systemem operacyjnym. W celu uruchomienia emulacji, wykonane zostało następujące polecenie:

```
./qemu-system-riscv32-sykt -M sykt -nographic
-bios fw_jump.elf
-kernel Image
-append "root=/dev/vda ro"
-drive file=rootfs.ext2,format=raw,id=hd0
-device virtio-blk-device,drive=hd0
-netdev user,id=net0 -device virtio-net-device,netdev=net0
```

Po wykonaniu tego polecenia nowy system rozpoczął pracę, po zakończeniu procesu rozruchu, ukazał się komunikat o konieczności zalogowania się. Należało wpisać hasło: `root` i pojawił się znak zachęty `#`, po czym powłoka oczekiwała na polecenia. Aby sprawdzić, czy moduł jądra został poprawnie skompilowany, został on załadowany za pomocą polecenia `modprobe kernel_module`. Aby usunąć moduł jądra zostało wykonane polecenie `rmmod kernel_module`. Potwierdzenie kompilacji znajduje się na zrzucie ekranu poniżej.

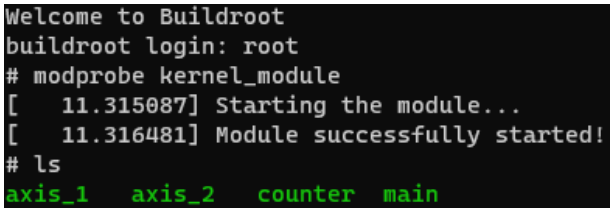
```
sykt@deb4sykom15:~/Romanek_Jakub_Piotr/projekt$ ./qemu-system-riscv32-sykt -M sykt -nographic -bios fw
_jump.elf -kernel Image -append "root=/dev/vda ro" -drive file=rootfs.ext2,format=raw,id=hd0 -device v
irtio-blk-device,drive=hd0
qemu-system-riscv32-sykt: info: Qemu for SYKT lecture made by A.Pruszkowski (Compiled at: Feb 28 2022
09:32:23)
Qemu internal configuration was found.
QEMU start report:
Compilation: Feb 28 2022 09:32:23
NIC:         enp0s3
NIC MAC:     08:00:2f:c5:d4:92 (17)
HASH:        0x025d127d97918635
NIC2:        0x000008002fc5d492
Qemu internal configuration was found.
QEMU start report:
Compilation: Feb 28 2022 09:32:23
NIC:         enp0s3
NIC MAC:     08:00:2f:c5:d4:92 (17)
HASH:        0x025d127d97918635
NIC2:        0x000008002fc5d492
GPIO Emulator for QEMU initializing .... (Compiled at: Jun  1 2022 23:30:08)
qemu-system-riscv32-sykt: info: Qemu for SYKT lecture begin (SYKT_IO)

OpenSBI v0.5 (Mar 17 2021 16:41:14)
   ____                    _____ ____ _____
  / __ \                  / ____|  _ \_   _|
 | |  | |_ __   ___ _ __ | (___ | |_) || |
 | |  | | '_ \ / _ \ '_ \ \___ \|  _ < | |
 | |__| | |_) |  __/ | | |____) | |_) || |_
  \____/| .__/ \___|_| |_|_____/|____/_____|
        | |
        |_|
```

Rys. 4: Uruchomienie utworzonego systemu Linux

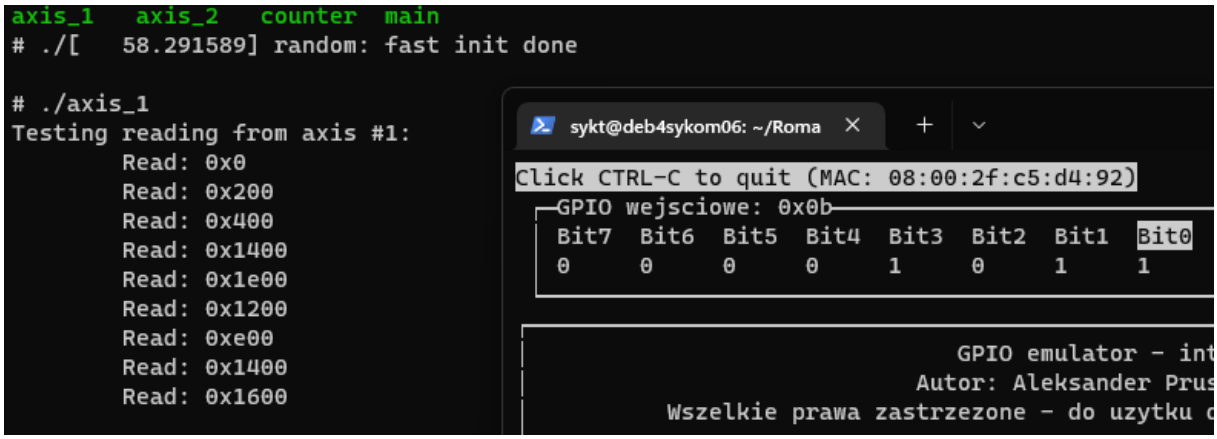
7 Testy demonstrujące działania całościowego nowo utworzonego systemu

7.1 Testowanie z wykorzystaniem własnej aplikacji

Aby przetestować działanie systemu, napisane zostały aplikacje w języku C. W celu przetestowania programu, w systemie macierzystym została wywołane komendy `make_busybox_compile axis_1.c, make_busybox_compile axis_2.c, make_busybox_compile counter.c`. Poniżej przedstawione są zrzuty ekranu z przeprowadzonych testów.

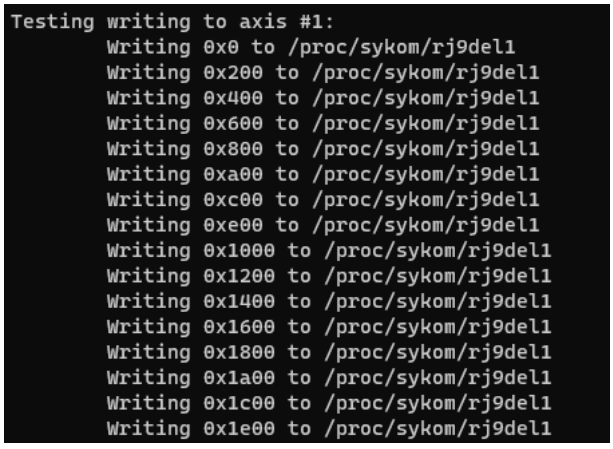


Rys. 5: Wyświetlenie aplikacji testowych



Rys. 6: Test odczytu z osi 1

Na cztery najmłodsze bity GPIO wejściowego podawana jest wartość `1011` co w notacji szesnastkowej daje `0xb`. Aplikacja testowa zwraca wartość odczytaną z pliku rj9del1 i zwraca wartość przesuniętą o 9 bitów w lewo, co daje nam `0x1600`.

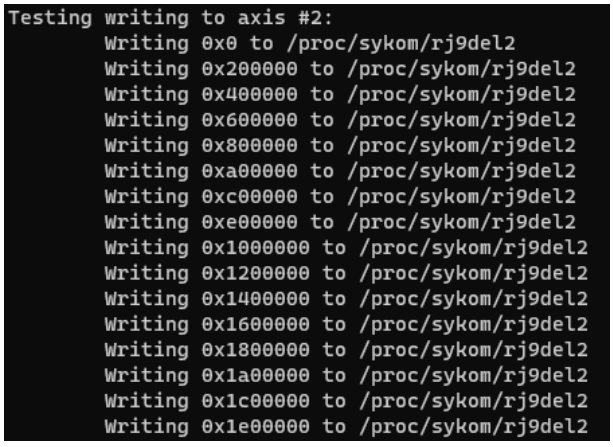


Rys. 7: Test zapisu do osi 1

Rys. 8: GPIO wyjściowe (podawana jest wartość 0x800 co po przesunięciu w prawo o 9 bitów daje 4

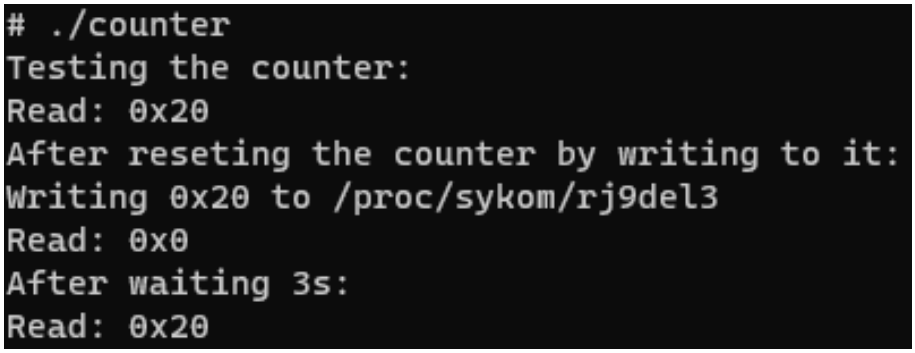
Rys. 9: Test odczytu z osi 2

Na cztery najstarsze bity GPIO wejściowego podawana jest wartość `0111` co w notacji szesnastkowej daje `0x7`. Aplikacja testowa zwraca wartość odczytaną z pliku rj9del2 i zwraca wartość przesuniętą o 21 bitów w lewo, co daje nam `0xe00000`.



Rys. 10: Test zapisu do osi 2

Rys. 11: GPIO wyjściowe (podawana jest wartość 0x800000 co po przesunięciu w prawo o 21 bitów daje 4



Rys. 12: Test licznika, najpierw jest odczytywana wartość sygnału INT, następnie jest on zerowany, a po 3 sekundach jego wartość jest ponownie odczytywana. Widzimy że wartość sygnału INT jest przesunięta o 5 miejsc w lewo co daje wartość 0x20