Laboratorium 2

Sterowanie urządzeniami z wykorzystaniem systemu plików Intel Galileo

Zakres: Laboratorium obrazuje podstawy sterowania urządzeń z wykorzystaniem wirtualnego systemu plików sysfs z poziomu systemu Linux.
Zasady zaliczenia:
zadanie 1 – 10%; Przygotowanie stanowiska
zadanie 2 – 10%; Porty GPIO Galileo
zadanie 3 – 10%; Zapis do portu
zadanie 4 – 10%; Odczyt z portu GPIO
zadanie 5 – 10%; Zapis i odczyt z portu
zadanie 6 – 10%; Odczyt wejścia analogowego (ADCs)
zadanie 7 – 20%; Oprogramowanie kanału PWM (Pulse-Width Modulation)
zadanie 8 – 20%; Zadanie do samodzielnej realizacji.

1. Przygotowanie stanowiska

Pobierz od prowadzącego, albo przygotuj według jego wskazań kartę SD z systemem linux. Na stanowisku laboratoryjnym wyposażanym w system Linux wykorzystaj polecenie *dd* do wgrania obrazu systemu na kartę. W systemie Windows wykorzystaj Win32DiskImager

Włóż kartę do platformy Galileo, podłącz się terminalem do platformy i uruchom system. Po zalogowaniu skonfiguruj połączenie sieciowe i przetestuj wgrywanie plików na Galileo przez sieć. Pod Linuxem za pomocą scp, pod Windowsem np. za pomocą WinSCP.

Korzystając z edytora tekstowego napisz skrypt wyświetlający zawartość podanego folderu, wgraj go na Galileo i uruchom w terminalu.

```
#!/bin/bash
echo `ls -l`
```

2. Porty GPIO Galileo

Należy pamiętać że ze względu a różnice konstrukcyjne platformy Galileo Gen1 (chip Cypress CY8C9540A) and Galileo Gen2 (chip PCAL9555A) programowanie tych dwóch platform trochę się różni. W tym laboratorium omówiony jest tylko przypadek Galileo Gen2.

Table 1. Configuration of Arduino-compatible ports on Galileo Gen2 platform

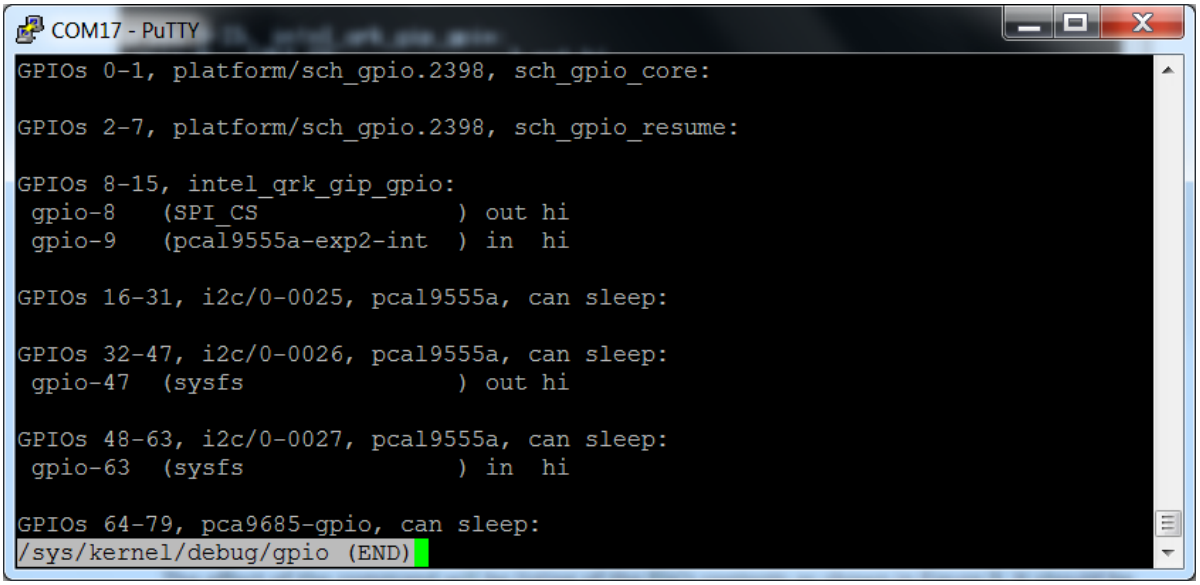
Shield pin	Function	Linux	Level Shifter GPIO L: dir_out H: dir_in I: *	22k Pull-Up GPIO L: pulldown H: pullup I: off	Pin Mux 1 GPIO	Pin Mux 2 GPIO	Interrupt modes L: low-level H:high-level R:rising-edge F:falling-edge B:both edges
IO0	UART0 RX	ttyS0	gpio32	gpio33	-	-	-
	GPIO	gpio11			-	-	L/H/R/F
IO1	UART0 TX	ttyS0	gpio28	gpio29	gpio45 (H)	-	-
	GPIO	gpio12			gpio45 (L)	-	L/H/R/F
IO2	UART1 RX	ttyS1	gpio34	gpio35	gpio77 (H)	-	
	GPIO	gpio13			gpio77 (L)	-	L/H/R/F
	GPIO	gpio61	-		gpio77 (L)	-	R/F/B
IO3	UART1 TX	ttyS1	gpio16	gpio17	gpio76(H)	-	-
	GPIO	gpio14			gpio76(L)	gpio64(L)	L/H/R/F
	PWM	pwm1			gpio76(L)	gpio64(H)	-
	GPIO	gpio62	-		gpio76(L)	gpio64(L)	R/F/B
IO4	GPIO	gpio6	gpio36	gpio37	-	-	R/F/B
IO5	GPIO	gpio0	gpio18	gpio19	gpio66(L)	-	R/F/B
	PWM	pwm3			gpio66(H)	-	-
IO6	GPIO	gpio1	gpio20	gpio21	gpio68(L)	-	R/F/B
	PWM	pwm5			gpio68(H)	-	-
IO7	GPIO	gpio38	-	gpio39	-	-	-
IO8	GPIO	gpio40	-	gpio41	-	-	-
IO9	GPIO	gpio4	gpio22	gpio23	gpio70(L)	-	R/F/B
	PWM	pwm7			gpio70(L)	-	-
IO10	GPIO	gpio10	gpio26	gpio27	gpio74(L)	-	L/H/R/F
	PWM	pwm11			gpio74(H)	-	-
IO11	GPIO	gpio5	gpio24	gpio25	gpio44(L)	gpio72(L)	R/F/B
	SPI MOSI	spidev1.0			gpio44(H)	gpio72(L)	-
	PWM	pwm9			-	gpio72(H)	-
IO12	GPIO	gpio15	gpio42	gpio43	-	-	L/H/R/F
	SPI MISO	spidev1.0					-
IO13	GPIO	gpio7	gpio30	gpio31	gpio46(L)	-	R/F/B
	SPI SCK	spidev1.0			gpio46(H)	-	-
IO14	GPIO	gpio48	-	gpio49	-	-	R/F/B
	ADC A0	in_voltage0_raw					-
IO15	GPIO	gpio50	-	gpio51	-	-	R/F/B
	ADC A1	in_voltage1_raw					-
IO16	GPIO	gpio52	-	gpio53	-	-	R/F/B
	ADC A2	in_voltage2_raw					-
IO17	GPIO	gpio54	-	gpio55	-	-	R/F/B
	ADC A3	in_voltage3_raw					-
IO18	GPIO	gpio56	-	gpio57	gpio60(H)	gpio78(H)	R/F/B
	ADC A4	in_voltage4_raw			gpio60(H)	gpio78(L)	-
	I2C SDA	i2c-0			gpio60(L)	-	-
IO19	GPIO	gpio58	-	gpio59	gpio60(H)	gpio79(H)	R/F/B
	ADC A5	in_voltage5_raw			gpio60(H)	gpio79(L)	-
	I2C SCL	i2c-0			gpio60(L)	-	-

There are the following designations in the table:

- "L" – GPIO port is configured as output in low state
- "H" – GPIO port is configured as output in high state
- "I" – GPIO port is configured as input in high impedance state

Status portów GPIO można sprawdzić poprzez odczyt pliku */sys/kernel/debug/gpio* np. za pomocą polecenia less:

```
less /sys/kernel/debug/gpio
```



Intel Galileo Gen2 posiada 79 portów, ale nie wszystkie sa wyprowadzone jako kompatybilne z Arduino. Zestaw kompatybilnych z Arduino przedstawiono w tabeli 1. Cześć portów jest kontrolowana bezpośrednio przez procesor Intel Quar, a część jest podłączona do pośredniczących układów PCAL9555 i PCA9685 (block GPIOexp, PWM, MUX,SHIFT)

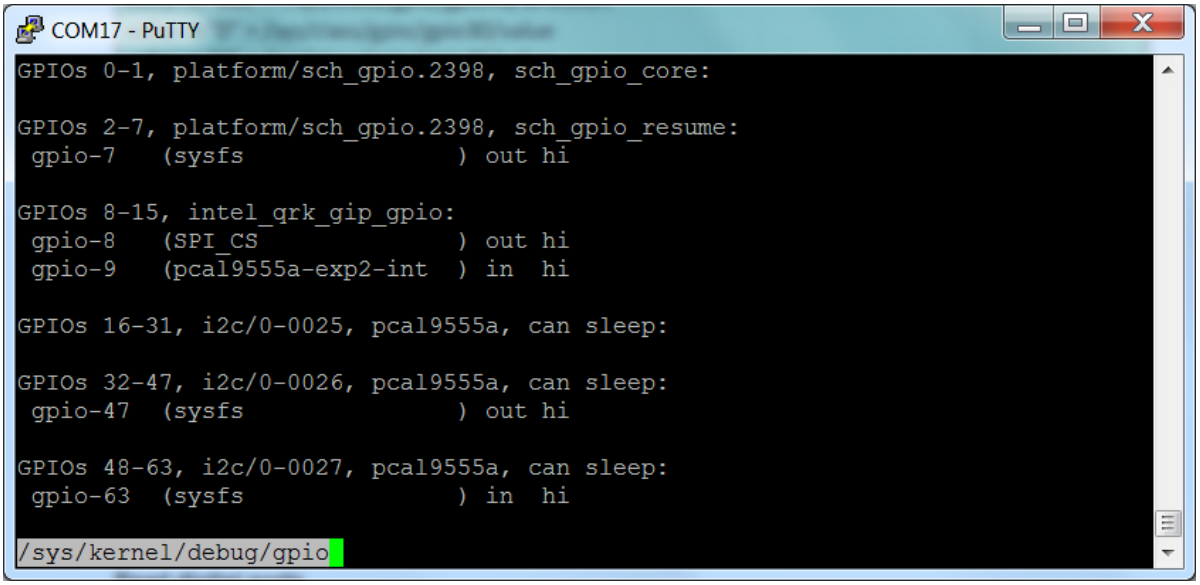
3. Zapis do portu

Przykładowo port IO13 jest podłączony do diody LED „L”. W celu sterowania tą diodą należy odpowiednio skonfigurować ten port poprzez zapis do plików określonych wartości. Można to zrobić poprzez wykonanie poszczególnych poleceń echo, ale warto napisać odpowiedni skrypt:

```
#!/bin/bash

#Eksport portu tak aby byl kontorlowany przez wirtualny system plikow
echo -n "7" > /sys/class/gpio/export
#Ustawienie portu jako wyjsciowego
echo -n "out" > /sys/class/gpio/gpio7/direction
#Zapalenie diody
echo -n "1" > /sys/class/gpio/gpio7/value
```

Niestety powyższe polecenia nie zadziałają (dioda się nie zapali) chociaż konfiguracja tego portu jest prawidłowa:



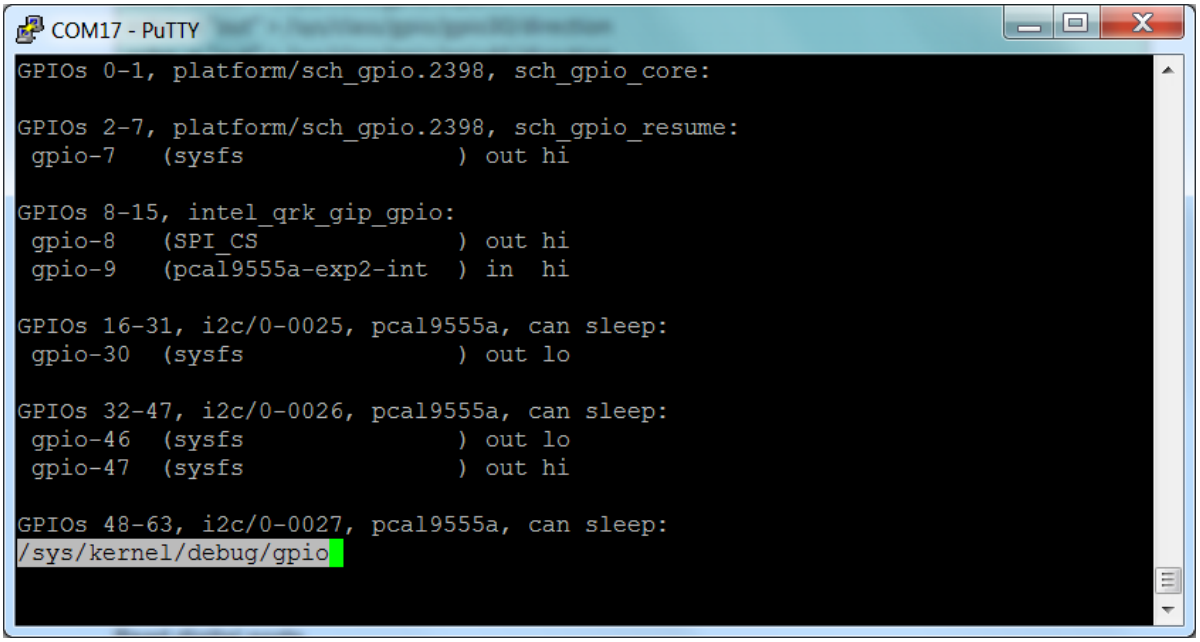
Wymagane jest jeszcze skonfigurowanie multiplekserów które występują pomiędzy portami mikroprocesora a konektorami kompatybilnymi z Arduino. W tum celu należy gpio30 i gpio46 zdefiniować jako wyjścia w stanie niskim:

```
echo -n "30" > /sys/class/gpio/export
echo -n "46" > /sys/class/gpio/export
echo -n "out" > /sys/class/gpio/gpio30/direction
echo -n "out" > /sys/class/gpio/gpio46/direction
echo -n "0" > /sys/class/gpio/gpio30/value
echo -n "0" > /sys/class/gpio/gpio46/value
```

W tym momencie można sterować diodą za pomocą zapisu stanu do plik:

```
echo -n "0" > /sys/class/gpio/gpio7/value
echo -n "1" > /sys/class/gpio/gpio7/value
```

Warto sprawdzić bieżącą konfigurację:



W ramach laboratorium umieść odpowiednie polecenia konfiguracyjne w skryptach *led_config.sh* oraz *multplexers_config.sh*.

4. Odczyt z portu GPIO

Każdy port kompatybilny z Arduino może być skonfigurowany jako wejście. Przykładowo konfiguracja dla portu IO2 wygląda następująco:

```
echo -n "13" > /sys/class/gpio/export
echo -n "34" > /sys/class/gpio/export
echo -n "35" > /sys/class/gpio/export
echo -n "77" > /sys/class/gpio/export
echo -n "in" > /sys/class/gpio/gpio13/direction
echo -n "out" > /sys/class/gpio/gpio34/direction
echo -n "out" > /sys/class/gpio/gpio35/direction
echo -n "out" > /sys/class/gpio/gpio77/direction
echo -n "1" > /sys/class/gpio/gpio34/value
echo -n "0" > /sys/class/gpio/gpio35/value
echo -n "0" > /sys/class/gpio/gpio77/value
```

Warto zwrócić uwagę że port gpio77 jest domyślnie skonfigurowany jako wyjście – dlatego konfiguracja kierunki zgłoś błąd braku pliku direction.
Stan portu IO2 można odczytać komendą:

```
cat /sys/class/gpio/gpio13/value
```

W ramach sprawozdania umieść polecenia konfiguracyjne w pliku *input_config.sh*.

5. Zapis i odczyt z portu

W ramach laboratorium napisz skrypt *led_button.sh* który będzie zapalał diodę w zależności od stanu portu IO2. W celu weryfikacji podłącz przycisk z zestawu Grove starter kit do konektora D2 w Base Shield.

6. Odczyt wejścia analogowego (ADCs)

Wejścia analogowe Interl Galile wykorzystują przetwornik analogowo cyfrowy AD7298. Jest on 8 kanałowy, ale tylko 6 kanałów jest podłączonych do portów A0-A5 kompatybilnych z Arduino. Wykorzystany ADC ma rozdzielczość 12 bitów, stąd odczytana wartość jest z przedziału 0-4095.
Z poziomu systemu Linux'a odczyt wejścia analogowego (np. A0) odbywa się podobnie jak dla wejść cyfrowych z wykorzystaniem systemu plików np.:

```
cat /sys/bus/iio/devices/iio\:device0/in_voltage0_raw
```

Przed uruchomieniem sprawdź konfigurację odpowiednich multiplekserów. Porty A0-A3 są domyślnie skonfigurowane jako wejścia analogowe, natomiast porty A4 i A5 wymagają ustawienia odpowiedniej konfiguracji.
W ramach laboratorium podłącz potencjometr z zestawu Grove Starter Kit do portu A0 i napisz skrypt *analog_in.sh* wyświetlający stan wejścia analogowego.
W sprawozdaniu zamieść wartości minimalne i maksymalne zwrócone przez skrypt przy podłączonym potencjometrze, oraz w przypadku nie podłączenia potencjometru. W przypadku braku potencjometru zaobserwuj wpływ podłączenia rezystorów podciągających do napięcia zasilania (pullup) i do masy (pulldown)

7. Oprogramowanie kanału PWM (Pulse-Width Modulation)

Sygnał PWM ma szereg zastosowań w systemach wbudowanych. W większości przypadków odpowiada za jasność świecenia diody LED, prędkość obrotową silników czy sterowanie serwomechanizmami. Galileo Gen2 posiada 16 kanałowy 12 bitowy kontroler PWM (PCA9685) podłączony do porcesora za pomocą interfejsu I^2C. Z 16 kanałów tylko 6 jest podpięta do konekotra kompatybilnego z Arduino. Pozostale są wykorzystywane jako porty GPIO.
Okres sygnału PWM **jest konfigurowany równocześnie dla wszystkich kanałów** i może się zawierać pomiędzy 666,666 a 41,666,666 nanoseund. W ramach sprawozdania oblicz jaka jest minimalna i maksymalna częstotliwość PWM.
Dostęp do konfiguracji kanałów PWM odbywa się poprzez folder */sys/class/PWM/pwmchip0*.
Podobnie jak dla zwykłych GPIO konfigurację kanałów PWM należy wyeksportować np. dla kanału 1:

```
echo -n "1" > /sys/class/pwm/pwmchip0/export
```

Włączenie kanału odbywa się za pomocą komendy

```
echo -n "1" > /sys/class/pwm/pwmchip0/pwm1/enable
```

Ustawienie okres np. dla 20ms odbywa się za pomocą komendy

```
echo -n "20000000" > /sys/class/pwm/pwmchip0/device/pwm_period
```

Wypełnienie ustawia się poprzez podanie czasu w nanosekunadach np dla okresu 20ms I wypełnienia 50% jest to 10000000ns.

```
echo -n "10000000" > /sys/class/pwm/pwmchip0/pwm1/duty_cycle
```

Podobnie jak dla GPIO należy odpowiednio skonfigurować multipleksery:

```
echo -n "16" > /sys/class/gpio/export
echo -n "17" > /sys/class/gpio/export
echo -n "76" > /sys/class/gpio/export
echo -n "64" > /sys/class/gpio/export
echo -n "out" > /sys/class/gpio/gpio16/direction
echo -n "in" > /sys/class/gpio/gpio17/direction
echo -n "out" > /sys/class/gpio/gpio76/direction
echo -n "out" > /sys/class/gpio/gpio64/direction
echo -n "0" > /sys/class/gpio/gpio16/value
echo -n "0" > /sys/class/gpio/gpio76/value
echo -n "1" > /sys/class/gpio/gpio64/value
```

W ramach laboratorium napisz skrypt *led_bright.sh* sterujący jasnością diody w przedziale 0-100% LED. Podłącz niebieską diodę LED do konektora D3 i sprawdź działanie.

8. Zadanie do samodzielnej realizacji.

Napisz skrypt *servo.sh* umożliwiający sterowanie wychyleniem serwomechanizmu podłączonego do IO5 za pomocą potencjometru podłączonego do konektora A1. Weź pod uwagę że servo akceptuje PWM o częstotliwości 50Hz i wypełnieniu w przedziale między 4%, a 12%.