Emulator konsoli NES w systemie Linux na komputerze VisionSOM

W portfolio większości elektroników/programistów bez trudu odnajdziemy hobbystyczne projekty, których finalna wartość użytkowa – zwłaszcza w stosunku do przeznaczonego na projekt nakładu pracy i środków – jest co najmniej dyskusyjna. Dlaczego zatem kosztem wolnego czasu decydujemy się budowanie takich urządzeń? Niech najprostszym wyjaśnieniem będzie to, że wraz z upływem lat wciąż nie wyrastamy z hasła „nauka poprzez zabawę", a hobbystycznie realizowane projekty mogą być dobrym ładunkiem praktycznej wiedzy, którą z powodzeniem wykorzystamy w życiu zawodowym.

W niniejszym artykule, na przykładzie komputera VisionSOM (z procesorem NXP i.MX6 ULL Cortex-A7) firmy SoMLabs [1], wyposażonego w dedykowany moduł wyświetlacza LCD-TFT, uruchomimy emulator konsoli Famicon/NES [2] (w Polsce znanej głównie dzięki klonowi oryginalnego projektu – konsoli Pegasus). Aby zwiększyć walor dydaktyczny projektu oraz by pod kątem „grywalności" jak najbardziej zbliżyć się do wspomnień z minionych lat, sterowanie za pomocą tradycyjnej klawiatury zastąpimy przyłączeniem i konfiguracją kontrolera konsoli Playstation 3/4 oraz prototypem własnego 8-przyciskowego gamepada, co pozwoli również na zapoznanie się z konfiguracją magistrali SPI oraz wyprowadzeń GPIO w systemie Linux. Zaczynajmy więc!

Przygotowanie karty z systemem Debian

Do budowy emulatora konsoli Famicon/NES wykorzystamy moduł komputera VisionSOM wyposażony w gniazdo karty pamięci SD, więc projekt rozpoczynamy od przygotowania karty z systemem operacyjnym Linux. Producent komputera – firma SoMLabs na stronach Wiki swojego produktu [3] dostarcza użytkownikowi wsparcie w postaci gotowych obrazów z dystrybucją Debian oraz opisem budowy obrazów z wykorzystaniem

takich narzędzi jak *Buildroot* oraz *Yocto*. Przy wstępnym wyborze odpowiedniej dystrybucji lub narzędzi do budowy obrazu pomocne jest jasne określenie celu projektu. Jeżeli finalnym celem projektu jest zbudowanie jak najwierniejszej kopii samej konsoli (z minimalnym obrazem systemu, wyposażonym wyłącznie w niezbędne oprogramowanie umożliwiające uruchomienie emulatora bezpośrednio po starcie urządzenia), wówczas zalecaną opcją jest wykorzystanie narzędzia *Buildroot*. Jeśli natomiast chcemy osiągnąć docelowy efekt jak najmniejszym nakładem pracy, a sam komputer jednopłytkowy ma pełnić również inne funkcje (np. korzystamy z pełnego środowiska graficznego, a emulacja konsoli NES jest wyłącznie dodatkową opcją), warto skorzystać z gotowych pakietów środowiska Debian – pozwoli to uniknąć m.in. samodzielnej kompilacji oprogramowania emulatora. W niniejszym artykule wykorzystano drogę na skróty i użyto gotowych obrazów z dystrybucją Debian (autor na własnym przykładzie przekonał się, że wybór drugiej ścieżki jest bardziej optymalny – zaoszczędzony czas można wówczas poświęcić na testowanie projektu i radosny powrót do czasów dzieciństwa).

Przygotowanie karty SD rozpoczynamy od pobrania obrazu systemu. W środowisku Linux operację tę możemy zrealizować poleceniem:

```
wget http://somlabs.com/os_images/debian-stretch-visionsosm-6ull.img.xz
```

Po pobraniu pliku rozpakujmy jego zawartość za pomocą narzędzia *unxz*:

```
unxz debian-stretch-visionsom-6ull.img.xz
```

Przed przystąpieniem do dalszych prac warto również sprawdzić integralność pobranych danych poprzez obliczenie sumy kontrolnej SHA256 i porównanie jej z informacjami udostępnianymi na stronie producenta:

```
sha256sum debian-stretch-visionsom-6ull.img
```

Po pobraniu oraz rozpakowaniu pliku możemy przystąpić do wgrania obrazu systemu na kartę micro-SD. Jedną najprostszych metod zapisania obrazu z pliku jest wykorzystanie linuksowego narzędzia *dd*. Ogólna składnia polecenia *dd* może może zostać zapisana następująco:

```
dd if=<pliku_wejściowy> of=<plik_wyjściowy> <dodatkowe opcje>
```

Plik wejściowy polecenia (*if – Input File*) będzie stanowił pobrany w uprzednim kroku obraz systemu *debian-stretch-visionsom-6ull.img*. Plik wyjściowy (*of – Output File*) to umieszczona w czytniku i podłączona do komputera karta SD, która jest reprezentowana w systemie jako plik specjalny w katalogu */dev*. W celu zweryfikowania, który wpis jest odpowiedzialny za podłączone urządzenie, korzystając z terminalu systemowego, możemy wydać polecenie *dmesg*, które wyświetli komunikaty wypisywane w buforze komunikatów jądra, w tym informacje o ostatnio przyłączonych urządzeniach, np:

```
[21870.506727] sdb: sdb1 sdb2
[21870.509486] sd 1:0:0:0: [sdb] Attached SCSI removable disk
```

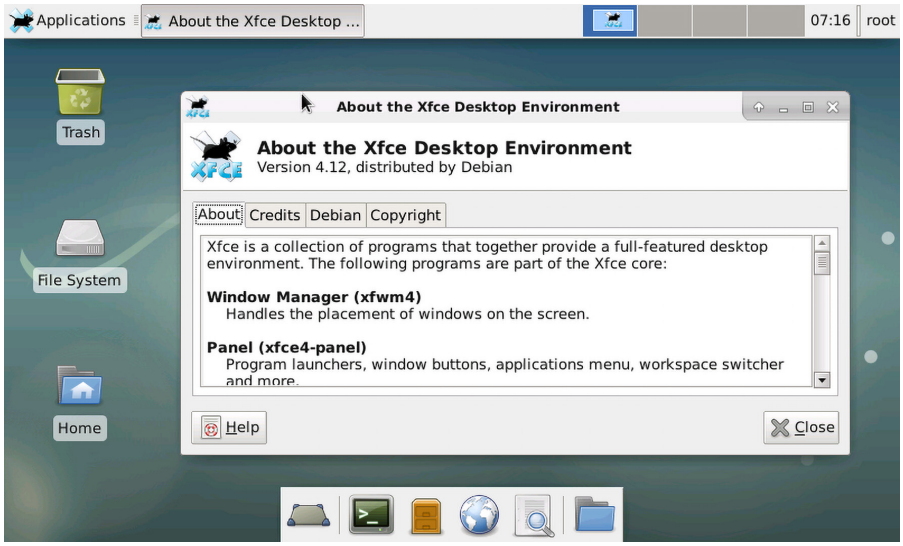
W powyższym przypadku wpis */dev/sdb* odnosi się do całej karty SD, natomiast wpisy */dev/sdb1*, */dev/sdb2*, *dev/sdbX*, do kolejnych partycji urządzenia – o ile zostały one uprzednio utworzone. W przypadku wykorzystania czytnika kart wbudowanego w komputer PC sterownik urządzenia może zarejestrować kartę SD w postaci pliku o nazwie */dev/mmcblk0*.

Polecenia *dd* używamy, korzystając z praw administratora, tak więc należy się bezwzględnie upewnić, że wprowadzono poprawną wartość dla pliku wyjściowego. Błędnie określenie wyjścia może spowodować nadpisanie danych na głównym dysku użytkownika – z powodu częstych pomyłek akronim narzędzia *dd* jest często rozwijany do „*destroy disk*" lub „*delete data*". Poprawne określenie pliku wyjściowego reprezentującego podłączoną do czytnika kartę SD pozwala na ostateczne określenie formy polecenia *dd*:

```
sudo dd if=./debian-stretch-visionsom-6ull.img of=/dev/sdX bs=4M oflag=dsync
```

Powyższe polecenie skopiuje obraz *debian-stretch-visionsom-6ull.img* na wskazaną kartę SD. Dodatkowe parametry polecenia *dd* określają zapis w blokach po 4 MB (*bs – Block Size*) w sposób synchroniczny (bez buforowania). Ponieważ polecenie *dd* nie umożliwia nam monitorowania postępu zapisu danych na kartę SD, co w zależności od wielkości obrazu systemu może być dość czasochłonne, wygodnym rozwiązaniem jest wykorzystanie narzędzia *pv*, które wyświetla informacje o postępie odczytu danych z pliku. Przykładowa składnia polecenia z wykorzystaniem narzędzia *pv* wygląda następująco:

```
pv debian-stretch-visionsom-6ull.img | dd of=/dev/sdX bs=4M ofag=dsync
```



Rysunek 1. Środowisko graficzne Xfce uruchomione na module VisionSOM

Po wgraniu obrazu na kartę, do konsoli systemu możemy zalogować się, wykorzystując wbudowany w płytę bazową *VisionCB-STD* konwerter *UART-USB* oraz dowolny program emulatora terminalu:

```
picocom -b 115200 /dev/ttyUSBX
```

Po otwarciu połączenia należy zalogować się na konto użytkownika *root* (bez hasła):

```
Debian GNU/Linux 9 localhost.localdomain ttymxc0
localhost login: root
root@localhost:~#
```

Środowisko kompilacji jądra Linux oraz Device Tree

Wraz z obrazem systemu Debian, firma SoMLabs dostarcza użytkownikowi wygodne środowisko konfiguracji i kompilacji jądra systemu oraz plików *Device Tree*. Środowisko to zostało zbudowane w oparciu o narzędzia *Qemu* oraz *chroot* [4], co pozwala na uruchomienie na komputerze PC (np. x86), nienatywnych plików wykonywalnych (np. dla architektury ARM) w standardowy dla plików natywnych sposób, tj. *./program*. Do poprawnego działania środowiska w dystrybucji *Ubuntu* niezbędna jest uprzednia instalacja pakietów *qemu*, *binfmt-support* oraz *qemu-user-static*:

```
apt-get install qemu binfmt-support qemu-user-static
```

Pobranie, rozpakowanie i uruchomienie (w tym wykonanie operacji *chroot* zmieniającej katalog główny) kompletnego środowiska:

```
wget http://somlabs.com/os_images/somlabs-visionsom-6ull-debian-rootfs-qemu.tar.xz
sudo tar xf somlabs-visionsom-6ull-debian-rootfs-qemu.tar.xz
sudo ./somlabs-debian/chtoolchain
```

Po uruchomieniu skryptu *chtoolchain*, źródła systemu Linux są dostępne w katalogu */home/developer/source/kernel/linux-rel_imx_4.1.15_2.1.0_ga*. Dla uproszczenia dalszego opisu przejdźmy do katalogu z kodem źródłowym:

```
cd /home/developer/source/kernel/linux-rel_imx_4.1.15_2.1.0_ga
```

Rysunek 2. Zainstalowany emulator konsoli Famicon/NES – pakiet Fceux

W kolejnym kroku, do katalogu *arch/arm/configs* należy skopiować domyślną konfigurację jądra (plik *visionsom-6ull-linux_defconfig*), udostępnioną przez producenta płytki *VisionSOM*:

```
cp ../../somlabs-dts-1.0/visionsom-6ull-linux_defconfig arch/arm/configs/
```

W chwili tworzenia niniejszego artykułu opis *Device Tree* (w postaci pliku *arch/arm/boot/dts/somlabs-visionsom-6ull.dts*) nie zawierał pełnego wsparcia dla ekranu LCD-TFT. Aby poprawnie uruchomić wyświetlacz, niezbędne jest pobranie i zaaplikowanie dodatkowej łatki:

```
wget http://wiki.somlabs.com/images/c/cd/Enable-tft-lcd.zip
unzip Enable-tft-lcd.zip
patch arch/arm/boot/dts/somlabs-visionsom-6ull.dts ./enable-tft-lcd.patch
```

W tak przygotowanym środowisku konfiguracja i kompilacja jądra oraz opisu *Device Tree* może zostać zrealizowana za pomocą następujących poleceń:

```
make ARCH=arm visionsom-6ull-linux_defconfig
make ARCH=arm menuconfig
make -j4 ARCH=arm zImage
make ARCH=arm somlabs-visionsom-6ull.dtb
```

Pliki wynikowe powyższych kompilacji, tj.:

arch/arm/boot/zImage,
arch/arm/boot/dts/somlabs-visionsom-6ull.dtb,

powinny zostać skopiowane do katalogu */boot* na karcie SD z obrazem systemu.

Instalowanie środowiska Xfce oraz emulatora Fceux

W konfiguracji domyślnej dostarczony przez producenta modułu obraz z dystrybucją Debian, nie ma zainstalowanego środowiska graficznego. Dzięki wykorzystaniu wbudowanego w system menedżera pakietów instalacja wybranego środowiska ogranicza się do dwóch prostych poleceń w konsoli systemu (w odróżnieniu od np. *Buildroota*, gdzie wymagana byłaby rekonfiguracja i ponowne przebudowanie obrazu systemu). Instalowanie „lekkiego" środowiska graficznego *Xfce* [5]:

```
root@somlabs:~# apt-get update
root@somlabs:~# apt-get install xfce4
```

Po podłączeniu klawiatury i myszki do portów USB (urządzenia te przydadzą się do pierwszych testów emulatora – zanim w kolejnych podpunktach przystąpimy do konfiguracji odpowiednich kontrolerów) oraz ponownym uruchomieniu komputera – urządzenie jest gotowe do pracy. Uruchomione środowisko *Xfce* (przy rozdzielczości ekranu 800×480) przedstawiono na **rysunku 1**.

Również z wykorzystaniem narzędzia *apt-get* zainstalujmy w systemie jeden z dostępnych emulatorów konsoli NES – oprogramowanie *Fceux* [6] (nie jest to oczywiście jedyny dostępny w repozytoriach dystrybucji *Ubuntu*, emulator konsoli *Famicon/NES*):

```
root@somlabs:~# apt-get install fceux
```

Po zakończonym etapie instalacji pakietu w sekcji *Games* środowiska *Xfce* zostanie dodany nowy wpis, pozwalający na uruchomienie emulatora bezpośrednio z interfejsu graficznego – **rysunek 2**.

Po uruchomieniu emulatora należy skonfigurować dołączoną klawiaturę, wybierając z górnej belki programu opcję *Fceux* → *Options* → *Gamepad Config*. Wczytanie gry jest realizowane poprzez wybranie opcji *Fceux* → *File* → *Open ROM* i wskazanie pliku **.rom*. Ze względu na aspekty prawne (emulacja sprzętu jest w pełni legalna – wątpliwości prawne może budzić wykorzystywanie plików ROM, nawet jeśli jesteśmy w posiadaniu oryginalnego kartridża z grą) autor nie umieścił w tekście bezpośrednich odnośników do źródeł.

Konfigurowanie kontrolera Sony DualShock 3/4 w jądrze Linux

Czytelnicy którzy dobrze pamiętają czasy świetności konsoli NES, dość szybko zauważą, że wykorzystanie standardowej klawiatury komputerowej nie pozwala na pełne oddanie magii i grywalności tego systemu. Pośrednim rozwiązaniem problemu (w stosunku do zakupu i przystosowania oryginalnego kontrolera konsoli Famicon/NES) może być wykorzystanie kontrolera od innych, współczesnych konsol, np. Sony Playstation lub Xbox. Jądro systemu Linux zapewnia pełne wsparcie dla urządzeń Sony DualShock 3 (od jądra w wersji 3.15) oraz DualShock 4 (od jądra w wersji 3.15 dla pierwszej generacji kontrolera oraz od wersji 4.10 dla drugiej generacji). Wsparcie po stronie jądra systemu wyklucza zatem konieczność samodzielnej programowej implementacji obsługi kontrolera, a całość konfiguracji ogranicza się do włączenia odpowiednich opcji jądra z wykorzystaniem polecenia: make arch=*ARM menuconfig*:

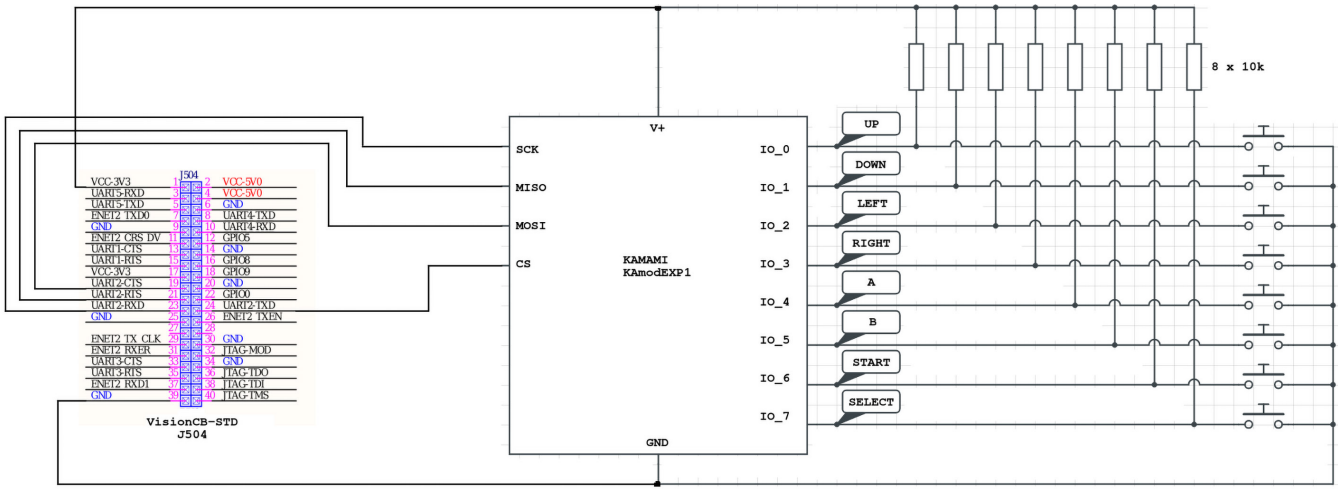
```
Device Drivers --->
    Input device support --->
        <*> Joystick interface
        <*> Event interface
    HID support --->
        [*] Battery level reporting for HID devices
        [*] /dev/hidraw raw HID device support
            <*> Generic HID driver
        Special HID drivers --->
            <*> Sony PS2/3/4 accessories
            [*] Sony PS2/3/4 accessories force feedback support
        USB HID support --->
            <*> USB HID transport layer
    [*] LED Support --->
        <*> LED Class Support
```

W domyślnej konfiguracji jądra systemu dostarczanej przez producenta płytki warto również wyłączyć opcję debugowania interfejsu zdarzeń (*Event debugging*) – w przeciwnym razie wszystkie zdarzenia z podsystemu wejścia będą rejestrowane, co niepotrzebnie obciąży urządzenie i zmniejszy bezpieczeństwo całego systemu (hasła wpisywane przez użytkownika również będą rejestrowane):

```
Device Drivers --->
    Input device support --->
        <*> Event debugging
```

Po ponownej kompilacji jądra systemu poleceniem *make zImage*, aktualizacji plików na karcie SD (obrazu jądra w katalogu */boot*) i podłączeniu kontrolera DualShock do portu USB, w buforze komunikatów jądra zostaną wyświetlone informacje o poprawnym wykryciu urządzenia:

```
root@somlabs:~# dmesg
```



Rysunek 3. Schemat połączeń dla modułu *KAmodEXP1* i płyty bazowej VisionCB-STD

```
<...>
usb 1-1: new full-speed USB device number 2
using ci_hdrc
input: Sony PLAYSTATION(R)3 Controller
as /devices/platform/soc/2100000.aips-
bus/2184000.usb/ci_hdrc.0/usb1/1-1/1-
1:1.0/0003:054C:0268.0004/input/input5
sony 0003:054C:0268.0004: input,hidraw3: USB
HID v1.11 Joystick [Sony PLAYSTATION(R)3
Controller] on usb-ci_hdrc.0-1/input0
```

W lokalizacji */dev/input* zostanie również utworzony wpis *jsX*, reprezentujący podłączony do systemu kontroler:

```
root@somlabs:~# ls -l /dev/input/
total 0
drwxr-xr-x 2 root root     160 Feb 23 10:24
by-id
drwxr-xr-x 2 root root     200 Feb 23 10:24
by-path
crw-rw---- 1 root input 13, 64 Feb 23 10:24
event0
crw-rw---- 1 root input 13, 65 Feb 23 10:24
event1
crw-rw-r-- 1 root input 13,  0 Feb 23 10:24
js0
crw-rw---- 1 root input 13, 63 Feb 23 10:24
mice
crw-rw---- 1 root input 13, 32 Feb 23 10:24
mouse0
```

W wypadku połączenia USB należy upewnić się, że układ zasilania komputera jednopłytkowego jest w stanie dostarczyć do podłączonego urządzenia odpowiedni zapas mocy (w przypadku modułów *VisionSOM* podłączonych do płyty bazowej *VisionCB-STD* zaleca się wykorzystanie odpowiedniego zasilacza – wykorzystanie złącza z oznaczeniem *„Linux Console + PWR-C”* może okazać się niewystarczające). Alternatywną możliwością zestawienia połączenia pomiędzy kontrolerem a komputerem jednopłytkowym jest wykorzystanie łączności Bluetooth oraz stosu BlueZ.

Po poprawnym wykryciu podłączonego kontrolera DualShock, można przystąpić do konfiguracji klawiszy w sposób analogiczny do przypadku standardowej klawiatury: *Fceux → Options → Gamepad Config*. Warto również wspomnieć, że sterownik kontrolera udostępnia również w przestrzeni użytkownika dodatkowe informacje o stanie naładowania baterii. Prostszą z opcji sprawdzenia stanu naładowanie kontrolera jest wykorzystanie interfejsu *sysfs*:

```
root@somlabs:~# cd /sys/class/power_supply/
root@somlabs:/sys/class/power_supply# ls -l
total 0
lrwxrwxrwx 1 root root 0 Feb 23 12:04 sony_
controller_battery_04:98:f3:95:a9:5a
root@somlabs:/sys/class/power_
supply# cat sony_controller_
battery_04\:98\:f3\:95\:a9\:5a/capacity
100
```

Dla kontrolera *DualShock 3* poziom naładowania jest raportowany w kroku 25% (100%, 75%, 50%, 25% → co odpowiada czterem poziomom naładowania wyświetlanych w interfejsie konsoli *Playstation3*), natomiast dla DualShock 4 wartość kroku pomiaru wynosi 10%. Alternatywną formą sprawdzenia stanu baterii jest wykorzystanie narzędzia *upower*, wówczas w pierwszej kolejności należy wyświetlić listę urządzeń udostępniających statystki:

```
root@somlabs:~# upower -e
/org/freedesktop/UPower/devices/keyboard_
sony_controller_battery_04o98of3o95oa9o5a
/org/freedesktop/UPower/devices/
DisplayDevice
```

A następnie, określając ścieżkę urządzenia, wyświetlić jego aktualne statystyki:

```
root@somlabs:~# upower -i /org/freedesktop/
UPower/devices/keyboard_sony_controller_
battery_04o98of3o95oa9o5a
  native-path:          sony_controller_
battery_04:98:f3:95:a9:5a
  power supply:         no
  updated:              Fri Feb 23 10:41:24
2018 (29 seconds ago)
  has history:          yes
  has statistics:       yes
  keyboard
    present:             yes
    rechargeable:        yes
    state:               fully-charged
    warning-level:       none
    percentage:          100%
    icon-name:          ,battery-full-
charged-symbolic'
```

Listing 1. Opis Device Tree dla kontrolera SPI3 oraz układu MCP23S08

```
&ecspi3 {
    fsl,spi-num-chipselects = <1>;
    cs-gpios = <&gpio1 20 0>;
    pinctrl-names = „default”;
    pinctrl-0 = <&pinctrl_ecspi3>;
    status = „okay”;
    gpiom1: gpio@0 {
        compatible = „microchip,mcp23s08”;
        gpio-controller;
        #gpio-cells = <2>;
        microchip,spi-present-mask = <0x01>;
        reg = <0>;
        spi-max-frequency = <10000000>;
    };
};
```

Listing 2. Konfiguracja wyprowadzeń I/O dla kontrolera SPI3

```
&iomuxc {
    pinctrl-names = „default”;
    pinctrl-0 = <&pinctrl_hog_1>;
    imx6ul-evk {
    /* … */
        pinctrl_ecspi3: ecspi3grp {
            fsl,pins = <
                MX6UL_PAD_UART2_RTS_B__ECSPI3_MISO      0x1b0b1
                MX6UL_PAD_UART2_CTS_B__ECSPI3_MOSI      0x1b0b1
                MX6UL_PAD_UART2_RX_DATA__ECSPI3_SCLK    0x1b0b1
                MX6UL_PAD_UART2_TX_DATA__GPIO1_IO20     0x1b0b0
            >;
        };
    };
};
```

Kontroler zbudowany z użyciem układu MCP23S08

Jeżeli nie dysponujemy żadnym kontrolerem, wówczas idealnym rozwiązaniem (choćby ze względu na walor edukacyjny) jest konstrukcja własnego prostego urządzenia. Oryginalny kontroler konsoli *NES* ma 8 przycisków: cztery do kontroli kierunku, dwa przyciski akcji (oznaczone jako A i B) oraz przyciski *START* i *SELECT*. Aby uniknąć problemów z wyborem i konfiguracją multipleksacji wyprowadzeń GPIO

Fotografia 4. Moduł VisionSOM z uruchomionym emulatorem konsoli Famicon/NES

(co nawet przy dużej liczbie wyprowadzeń I/O dostępnych na złączach komputerów jednopłytkowych nie zawsze jest proste), do sprzętowej budowy kontrolera wykorzystamy moduł *KAmodEXP1* [7], będący adresowalnym ekspanderem GPIO z układem *MCP23S08*. Układ ten może pracować zarówno na magistrali SPI, jak i I²C, oraz ma dokładnie osiem konfigurowalnych linii GPIO, a więc idealnie wpisuje się w realizowany projekt. Schemat połączeń przedstawiono na **rysunku 3** (układ *MCP23S08* podłączony w konfiguracji SPI).

Dla uniknięcia implementacji programowej obsługi układu *MCP23S08* w przestrzeni użytkownika, użyjemy gotowego sterownika dostępnego w jądrze systemu. Konfigurację jądra rozpoczynamy od włączenia sprzętowego kontrolera magistrali SPI dla układów *i.MX*:

```
Device Drivers  --->
    [*] SPI support  --->
        <*> Freescale i.MX SPI controllers
```

Następnie włączamy docelowy sterownik układu *MCP23S08*:

```
Device Drivers  --->
    -*- GPIO Support  --->
        SPI GPIO expanders  --->
            <*> Microchip MCP23xxx
I/O expander
```

Włączenie sterownika układu *MCP23S08* oraz przygotowanie dla niego opisu *Device Tree* (co przedstawiono poniżej) spowoduje pojawienie się w systemie nowego kontrolera GPIO, dostępnego w przestrzeni użytkownika poprzez interfejs */sys/class/gpio*. Aby maksymalnie uprościć obsługę projektowanego urządzenia, linie kontrolera GPIO wykorzystamy w sterowniku klawiatury *Polled GPIO buttons* [więcej informacji – patrz ramka]:

```
Device Drivers  --->
    Input device support  --->
        <*> Event interface
        [*] Keyboards  --->
            < > GPIO Buttons
            <*> Polled GPIO buttons
```

Konfigurację jądra uzupełniamy opisem *Device Tree* wymaganym przez włączone powyżej sterowniki. Edycję pliku *somlabs-visionsom-6ull.dts* rozpoczynamy od dodania opisów dla kontrolera SPI oraz układu *MCP23S08* (**listing 1**).

Listing 3. Wyłączenie kontrolera UART2 w opisie Device Tree

```
&uart2 {
    pinctrl-names = „default”;
    pinctrl-0 = <&pinctrl_uart2>;
    fsl,uart-has-rtscts;
    //status = „okay”;
};
```

Listing 4. Opis Device Tree dla sterownika gpio-keys-polled

```
gpio-keys {
    compatible = „gpio-keys-polled”;
    poll-interval = <100>;
    btn0 {
        label = „btn0”;
        gpios = <&gpiom1 0 GPIO_ACTIVE_HIGH>;
        linux,code = <103>; /* <KEY_UP> */
    };
    btn1 {
        label = „btn1”;
        gpios = <&gpiom1 1 GPIO_ACTIVE_HIGH>;
        linux,code = <108>; /* <KEY_DOWN> */
    };
    btn2 {
        label = „btn2”;
        gpios = <&gpiom1 2 GPIO_ACTIVE_HIGH>;
        linux,code = <105>; /* <KEY_LEFT> */
    };
    /* … */
};
```

Do komunikacji z układem *MCP23S08* wykorzystamy kontroler SPI3 (*ecspi3*) w konfiguracji z jedną linią CS (wartość określona poprzez pole *fsl,spi-num-chipselects*), sterowaną poprzez wyprowadzenie *GPIO1_20* (definicja linii w polu *cs-gpios*). W opisie kontrolera wskazujemy również odwołanie do multipleksacji wyprowadzeń I/O (*pinctrl-0=<&pinctrl_ecspi3>*). Następnie umieszczamy opis nowego kontrolera *gpiom1*, którego funkcje pełni układ *MCP23S08*. Opis rozpoczynamy od wskazania typu układu (układ ekspandera jest produkowany również w wersji z 16 liniami GPIO) i sposobu jego dołączenia (SPI lub I²C):

- *compatible* = *„microchip,mcp23s08”* – dla wersji SPI z 8 liniami GPIO,
- *compatible* = *„microchip,mcp23s17”* – dla wersji SPI z 16 liniami GPIO,
- *compatible* = *„microchip,mcp23008”* – dla wersji I²C z 8 liniami GPIO,
- *compatible* = *„microchip,mcp23017”* – dla wersji I²C z 16 liniami GPIO.

Wpisem *gpio-controller* oznaczamy urządzenie jako pełniące funkcję kontrolera GPIO. Pole *microchip,spi-present-mask* jest

Zaimplementowany w jądrze systemu Linux sterownik układu *MCP23S08* nie wspiera aktualnie funkcji przerwań dla konfiguracji SPI (opcja ta jest wspierana wyłącznie przez układy pracujące na magistrali I²C). Z tego powodu, na kolejnym etapie konfiguracji jądra wykorzystano sterownik *Polled GPIO buttons* z aktywnym odpytywaniem. Dla układów pracujących na magistrali I²C bardziej optymalnym rozwiązaniem będzie wybór sterownika *GPIO Buttons* oraz skonfigurowanie kontrolera GPIO jako *„interrupt-controller”*. Przykładowy opis *Device Tree* dla układu *MCP23S08* pracującego na magistrali I²C może wyglądać następująco:

```
gpiom1: gpio@20 {
        compatible =
„microchip,mcp23017”;
        gpio-controller;
        #gpio-cells = <2>;
        reg = <0x20>;
        interrupt-parent = <&gpio1>;
        interrupts = <17 IRQ_TYPE_LEVEL_
LOW>;
        interrupt-controller;
        #interrupt-cells=<2>;
        microchip,irq-mirror;
};
```

W powyżej przedstawionym opisie pole *reg* definiuje sprzętowy adres układu *MCP23S08* na magistrali I²C.

istotne wyłącznie wtedy, gdy jedna linia CS jest dzielona przez większą liczbę adresowalnych układów. Pole *reg* – do konfiguracji *MCP23S08* jako układu SPI – określa numer porządkowy wyprowadzenia linii CS. Wpisem *spi-max-frequency* określamy maksymalną dopuszczalną częstotliwość zegarową interfejsu SPI (tutaj 10 MHz).

Tak przygotowany opis kontrolera SPI oraz układu *MCP23S08* należy uzupełnić o opis konfiguracji multipleksowania wyprowadzeń I/O (**listing 2**). Wyprowadzenia, którym na listingu 2 przypisano role linii *SCLK* oraz *CS*, w domyślnym opisie *Device Tree* dostarczanym przez producenta płytki *VisionSOM* pełnią funkcję linii *RX* oraz *TX* kontrolera *UART2*. Aby uniknąć konfliktu, należy wyłączyć kontroler *UART2* poprzez wykomentowanie linii statusu, jak przedstawiono to na **listingu 3**.

Ostatnim etapem przygotowania opisu *Device Tree* jest zdefiniowanie 8 przycisków w ramach sterownika *gpio-keys-polled*. Ponieważ sterownik pracuje w trybie odpytywania, niezbędne jest określenie interwału czasowego poprzez pole *poll-interval* (wartość wyrażona w milisekundach). Następnie dokonano opisu ośmiu przycisków, definiując ich etykiety (pole *label*), odwołania do kontrolera GPIO (do kolejnych linii I/O kontrolera *gpiom1*/układu *MCP23S08*) oraz przypisując im wybrane kody zdefiniowane przez podsystem wejścia. Fragment opisu *Device Tree* przedstawiono na **listingu 4**. Po skompilowaniu nowego obrazu jądra oraz opisu *Device Tree* z wykorzystaniem poleceń:

```
make ARCH=arm zImage
make ARCH=arm somlabs-visionsom-6ull.dtb
```

pliki wynikowe (tj. *zImage* oraz *somlabs-visionsom-6ull.dtb*) należy skopiować do katalog */boot* na karcie SD. Po ponownym uruchomieniu systemu w buforze komunikatów jądra zostaną wyświetlone komunikaty informujące o poprawnej konfiguracji sterownika:

```
root@somlabs:~# dmesg | grep gpio-keys
input: gpio-keys as /devices/platform/gpio-
keys/input/input0
```

Poprawność działania kontrolera oraz podłączonych przycisków może zostać przetestowana za pomocą narzędzia *evtest*:

```
root@somlabs:~# apt-get install evtest
root@somlabs:~# evtest --grab /dev/input/
event0
Input driver version is 1.0.1
Input device ID: bus 0x19 vendor 0x1 product
0x1 version 0x100
Input device name: „gpio-keys”
Supported events:
  Event type 0 (EV_SYN)
  Event type 1 (EV_KEY)
    Event code 103 (KEY_UP)
    Event code 105 (KEY_LEFT)
    ….
Properties:
Testing ... (interrupt to exit)
Event: time 1519407395.872567, type 1 (EV_
KEY), code 108 (KEY_DOWN), value 0
Event: time 1519407395.872567, ----------
-- SYN_REPORT ------------
```

Ostatecznej konfiguracji przycisków należy dokonać z wykorzystaniem opcji *Gamepad Config* w emulatorze *Fceux* (**fotografia 4**).

Łukasz Skalski
contact@lukasz-skalski.com

Bibliografia:

- ***https://goo.gl/7JoUPg***
- ***https://goo.gl/DgRwoR***
- ***https://goo.gl/Y5WFjB***
- ***https://goo.gl/56DLir***
- ***https://goo.gl/ANvQL2***
- ***https://goo.gl/uuAKVF***
- ***https://goo.gl/hMFcCe***