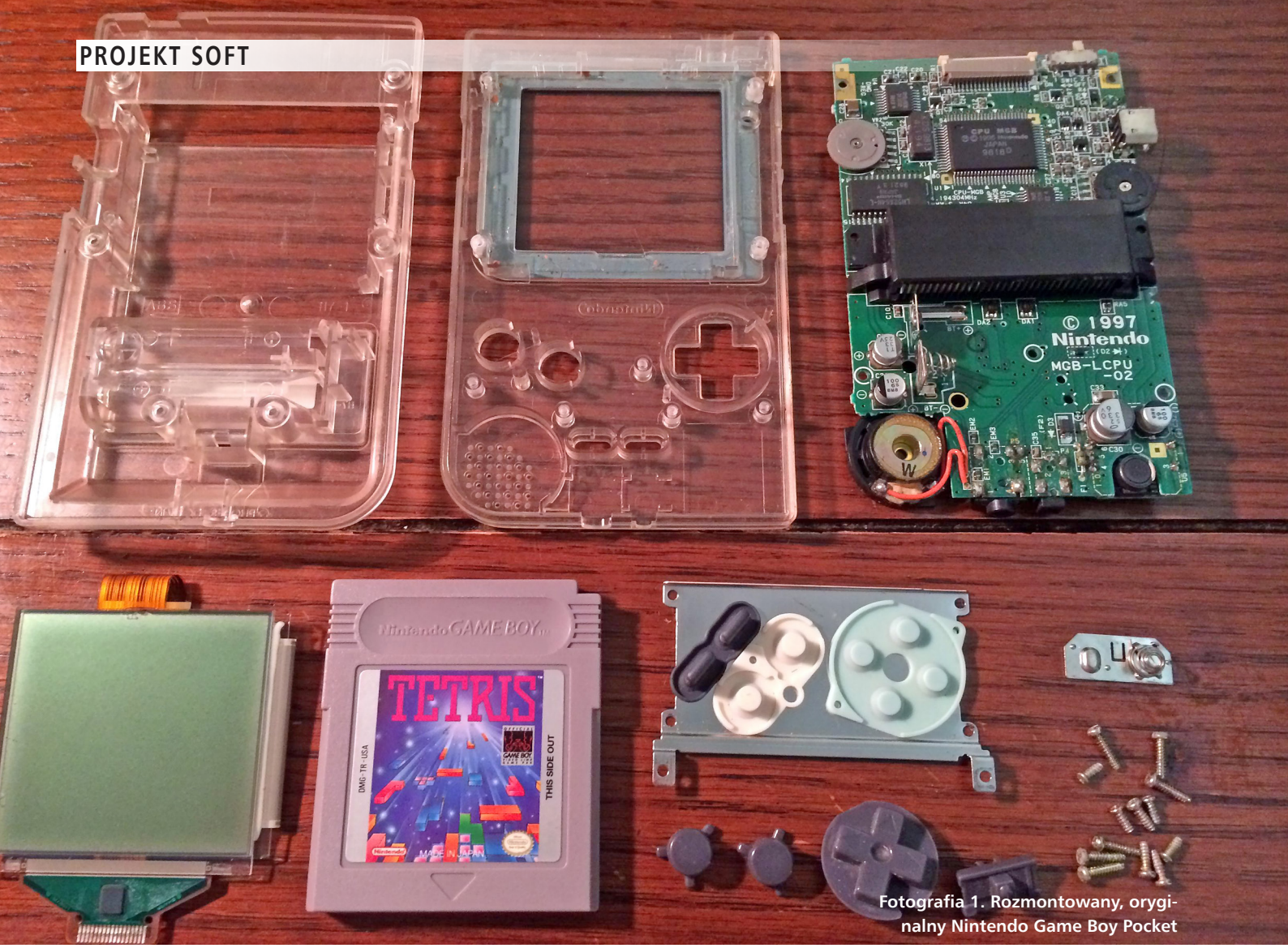


Fotografia 1. Rozmontowany, oryginalny Nintendo Game Boy Pocket

Pi-Pocket – przenośna konsola do gier z Raspberry Pi

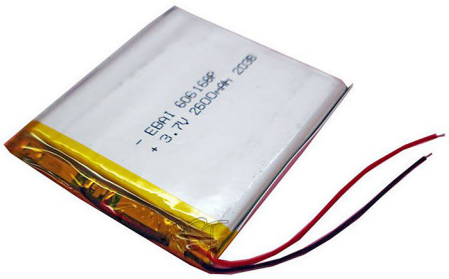
Zapewne większość czytelników EP pamięta emocje i pierwsze konsole do gier. Ówczesne gry, choć nie zachwycały grafiką, potrafiły wciągnąć na długie godziny. Co prawda, tamtego sprzętu na próżno szukać w sprzedaży, ale współczesna technika i technologia dają możliwość „powrotu" do tamtych lat. Tak właśnie postąpił autor opisywanego projektu, który przygotował nowoczesną wersję konsoli Nintendo Game Boy, która na dodatek pozwala na uruchamianie nowszych gier i ma kolorowy wyświetlacz.

Potrzebne podzespoły:
- Raspberry Pi Model B z 512 MB pamięci RAM.
- 2,5-calowy, kolorowy wyświetlacz LCD z wejściem analogowym.
- Karta pamięci SD (w projekcie użyto modelu SanDisk Ultra Micro SDHC Class 10 30 Mb/s o pojemności 32 GB.
- Płytka Teensy 2.0 USB Development Board.
- Game Boy Pocket – może być niesprawny – ważne, by miał działające przyciski.
- Akumulator Li-Ion lub Li-Po o napięciu 3,7 V i pojemności około 2600 mAh (w tym wypadku chiński Flat Cell LP606168).
- Przetwornica podwyższająca napięcie z 3,7 V do 5 V.
- Przetwornica obniżająca 3,3 V / 1 A.
- Miniaturowy wzmacniacz audio.
- Moduł ładowania akumulatorów litowych 3,7 V.
- Miniaturowy, okrągły głośnik o średnicy 22 mm.
- Płaskie złącze USB.
- Szybka chroniąca ekran konsoli.

Autorem projektu jest Travis Brown z USA. Choć prezentowane urządzenie opiera się o Raspberry Pi i różne moduły, a także bazuje na zaawansowanym, gotowym oprogramowaniu, wymaga bardzo dużo „kombinowania" oraz spędzenia czasu nie tylko z lutownicą, ale i z miniaturową wiertarką/szlifierką kątową. Opis projektu został opublikowany pod adresem http://goo.gl/x8Lsbr.

Wstęp

Autor projektu miał niesprawnego Game Boy'a Pocket – niewielką, przenośną konsolę z wbudowanym kontrolerem i wyświetlaczem monochromatycznym. Chcąc ją przywrócić do „życia" zauważył, że obudowa konsoli jest na tyle duża, że zmieści Raspberry Pi, a ten komputer z odpowiednim systemem operacyjnym jest w stanie uruchomić emulator, który pozwoliłby na włączanie różnego rodzaju 8- i 16-bitowych gier, takich jak wydawane na konsole Nintendo Entertainment System (popularny



Fotografia 2. Zastosowany akumulator litowo-jonowy

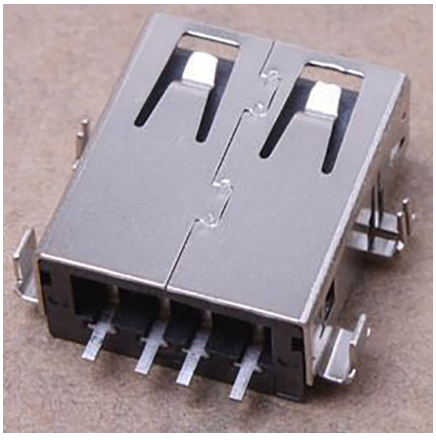
NES, znany w Polsce w wersji Pegasus), Game Boy, Sega Master System i Game Gear. Co więcej, Linux sprawia, że na zmontowanej konsoli da się też grać w niektóre gry PeCetowe – i to wszystko na przenośnym mini-komputerku, bez potrzeby noszenia ze sobą stosu kartridży.

Autor wykorzystał posiadany przez siebie model Nintendo Game Boy Pocket, przy czym kluczowe elementy tej konsoli, które posłużyły do stworzenia Pi-Pocket to obudowa wraz z płytką z przyciskami. Większość pozostałych komponentów została wymieniona.

Obudowa

Jako obudowa posłużył Game Boy Pocket w wersji przezroczystej, co nie tylko sprawiło, że wnętrze urządzenia jest widoczne z zewnątrz, ale w razie potrzeby łatwo jest dorobić kawałek obudowy z również przezroczystego pleksi, bez potrzeby dobierania koloru czy malowania całości.

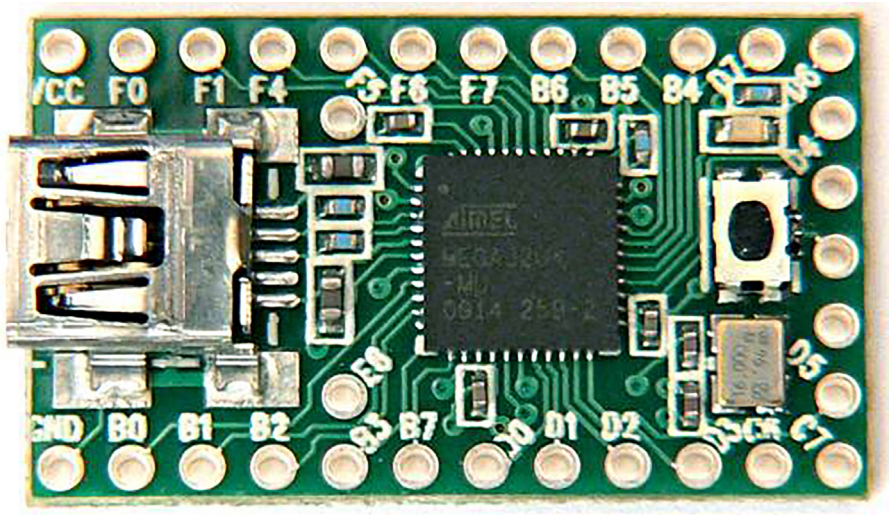
Game Boy został rozłożony na kawałki (**fotografia 1**). Ponieważ Pi-Pocket miał być zasilany wbudowanym akumulatorem, zamiast bateriami AAA, autor zakleił na stałe klapkę baterii, po czym wypolerował wnętrze obudowy i wyciął elementy, które służyły tylko i wyłącznie do pozycjonowania kartridży z grami. W miejsce ich i obudowy, wmontowany został akumulator litowo-jonowy (**fotografia 2**). Gniazdo kartridży zostało zaklejone przezroczystym plastikiem, a w nim wycięto mały otwór na kartę SD, na której będą przechowywane gry i system.



Fotografia 3. Użyte płaskie gniazdo USB

Tabela 1. Parametry dostępnych płytek Teensy

Specyfikacja	Teensy 2.0	Teensy++ 2.0	Teensy 3.0	Teensy 3.1
Procesor	ATMEGA32U4 8 bit AVR 16 MHz	AT90USB1286 8 bit AVR 16 MHz	MK20DX128 32 bit ARM Cortex-M4 48 MHz	MK20DX256 32 bit ARM Cortex-M4 72 MHz
Pamięć Flash	32256	130048	131072	262144
Pamięć RAM	2560	8192	16384	65536
EEPROM	1024	4096	2048	2048
Wejścia i wyjścia	25, 5 V	46, 5 V	34, 3,3 V	34, 3,3V, 5V
Wejścia analogowe	12	8	14	21
PWM	7	9	10	12
UART / I²C / SPI	1 / 1 / 1	1 / 1 / 1	3 / 1 / 1	3 / 2 / 1
Cena [USD]	16,00	24,00	19,00	19,80



Fotografia 4. Moduł Teensy 2.0

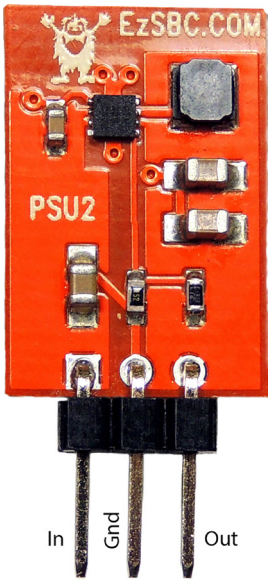
W obudowie (w miejscu dotychczasowego portu kabla EXT) przygotowano też miejsce na złącze USB (**fotografia 3**), za którego pomocą Pi-Pocket można będzie w przyszłości rozbudować o dodatkowe funkcje, takie jak choćby dostęp do sieci Wi-Fi za pomocą karty sieciowej na USB.

Pad

Nieodłącznym elementem każdej prawdziwej konsoli do gier jest przynajmniej jeden pad, czyli kontroler, za pomocą którego sterujemy postaciami w grach. W przypadku Game Boy'a, pad jest zintegrowany w obudowie. Zastąpienie go innym kontrolerem byłoby bardzo niewygodne, dlatego autor zdecydował się wykorzystać oryginalną płytkę Pada.

Kontroler Game Boy'a ma cztery przyciski kursorów, dwa przyciski akcji i dwa dodatkowe przyciski „select" i „start". Płytka drukowana, na której zostały wykonane zawiera trochę niepotrzebnych elementów, które zajmują cenną przestrzeń w obudowie, dlatego autor je wyciął – usunął m.in. złącza i kondensatory, przycinając całość do potrzeb, ale tak by na płytce pozostały oryginalne otwory montażowe. Konieczne było też przecięcie niektórych ścieżek, które łączyły ze sobą wyprowadzenia przycisków. Wynika to z faktu, że pad był podłączony do oryginalnej jednostki centralnej w inny sposób, niż będzie do Raspberry Pi. W tworzonym projekcie, każdy z przycisków łączony jest za pomocą dwóch przewodów, gdzie jeden przewód prowadzi do masy, a drugi do interfejsu Teensy (**fotografia 4**), który symuluje klawiaturę. W praktyce, ze względu na istniejące wspólne połączenia,

Fotografia 5. 2,5-calowy wyświetlacz, oklejony paskami czarnej taśmy izolacyjnej



Fotografia 6. Miniaturowa przetwornica obniżająca

wystarczyło zastosowanie 8 przewodów do Teensy i 2 przewodów do masy.

Sam moduł Teensy, choć jest bardzo mały, wymagał przycięcia, by zmieścić się w obudowie. Autor obciął brzegi płytki, usunął jej gniazdo USB i przymocował ją do PCB pada, podłączając wyprowadzenia.

Raspberry Pi

Kolejnym krokiem było przygotowanie Raspberry Pi do montażu w obudowie. Niewielkie rozmiary Game Boy'a sprawiły, że konieczne okazało się usunięcie niepotrzebnych złączy i wyprowadzeń z komputerka. Autor pozostawił tylko gniazdo kart SD, analogowe wyjścia audio i wideo oraz port USB. W efekcie z płytki usunięto interfejs wyświetlacza cyfrowego, kamery, wyprowadzenia GPIO, gniazda ethernetowe i HDMI, a nawet porty USB, audio i wideo oraz regulator zasilania wraz z kondensatorem, które zastąpiono mniejszą przetwornicą. Jeśli operację ich demontażu wykona się poprawnie, Raspberry Pi będzie wciąż dobrze działać.

Wyświetlacz

Z punktu widzenia użytkownika, podstawową różnicą pomiędzy Pi-Pocket a oryginalnym Game Boyem jest wyświetlacz. Zamiast starego, monochromatycznego, o rozdzielczości 160×144 piksele, autor zastosował większy, kolorowy, 2,5-calowy ekran o rozdzielczości 480×234 piksele (**fotografia 5**). Co więcej, w obudowie mógłby się nawet zmieścić wyświetlacz 3-calowy, ale autor projektu nie był w stanie znaleźć odpowiedniego modelu z analogowym interfejsem. Użycie niekompatybilnego ekranu wymagałoby zamontowania dodatkowego konwertera sygnałów, który nie tylko zajmowałby cenne miejsce w obudowie, ale też niepotrzebnie zwiększałby komplikację układu i wymagałby włożenia dodatkowej pracy w projekt.

W Pi-Pocket wykorzystano istniejącą płytkę interfejsu wyświetlacza, przy czym po przyjrzeniu się oznaczeniom jej wyprowadzeń, da się zauważyć, że jest tam miejsce na napięcie 3,3 V. Okazuje się, że jest ono wystarczające do zasilenia płytki, mimo że standardowo jest ona zasilana napięciem z zakresu od 5 V do 12 V, co skutkuje niewielką zmianą jasności wyświetlacza i wynikało z konieczności współpracy Game Boy'a z bateriami, które mogły się wyczerpywać. Co więcej, użycie napięcia 3,3 V zamiast wyższego spowodowało zmniejszenie poboru prądu przez wyświetlacz o 30 mA do 100 mA.

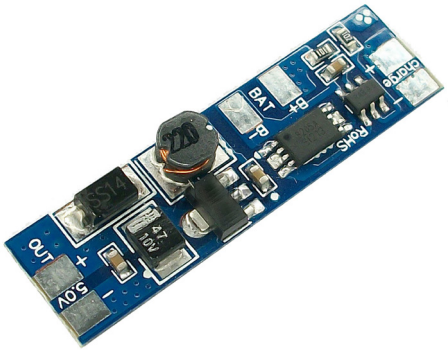
Nowy wyświetlacz mieści się w obudowie niemal perfekcyjnie. Autor musiał tylko delikatnie przyciąć niektóre wystające elementy plastikowe obudowy, w efekcie czego idealnie pasuje na szerokość. Jest on natomiast krótszy na długość, co wynika z faktu, że oryginalny wyświetlacz był nietypowy pod względem wymiarów: niemal kwadratowy, podczas gdy standardowo produkowane ekrany mają proporcje 4:3. Pozostałą w pionie przestrzeń warto czymś zakryć, dla uzyskania atrakcyjnego widoku. Można użyć np. czarnego kartonu, albo choćby taśmy izolacyjnej, którą większość czytelników ma pod ręką. Taśmę można nakleić na brzegi wyświetlacza, optycznie zwiększając jego rozmiary, przy czym jego krawędzie i tak są zakryte przez oryginalną czarną obwódkę na obudowie.

Wyświetlacz został na stałe przyklejony do frontu obudowy za pomocą kleju na gorąco. Z płytki interfejsu wyświetlacza wymontowano niepotrzebne komponenty, a kilka przeniesiono. Autor usunął trzy kondensatory i jedną cewkę. Pozostały jeden kondensator i jedna cewka były konieczne do uzyskiwania napięcia 2 V, by zasilić LED-owe podświetlenie ekranu, przy czym zostały one umieszczone na dole ekranu, dzięki czemu całość poprawnie mieści się w obudowie.

W górnej części obudowy, tuż przy wyświetlaczu, w oryginalnym Game Boy'u Pocket znajdował się włącznik zasilania. Autor go nieco zmodyfikował i podłączył do niego przewody biegnące do dołu obudowy, a także zamontował tam dwukolorową diodę LED, która informuje o stanie pracy i zasilania urządzenia.

Zasilanie

Cały Pi-Pocket potrzebuje trzech głównych regulatorów zasilania, przystosowujących poziomy napięć do poszczególnych podzespołów. Pierwszy z nich to przetwornica 3,3 V (**fotografia 6**), która służy do zasilania wyświetlacza, by – jak wspomniano wcześniej, obniżyć jego pobór prądu. W oryginalnym Game Boy'u znajduje się regulator liniowy, pozwalający uzyskać takie właśnie napięcie (potrzebne jednostce centralnej), ale zastąpienie go nowoczesną przetwornicą pozwoliło zmniejszyć pobór prądu o około



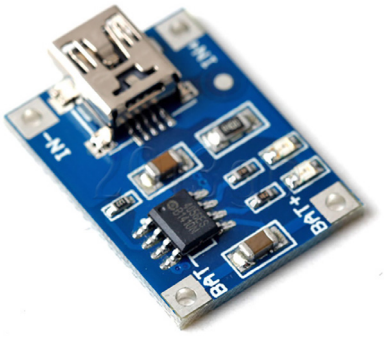
Fotografia 7. Przetwornica podwyższająca napięcie do 5 V, podłączona bezpośrednio do akumulatora

25%. Rozmiar nowej, gotowej przetwornicy jest tak mały, ze mieści się w obudowie, nawet jeśli zostanie zamontowana pod kątem prostym do płytki Raspberry Pi. PCB regulatora ma wymiary odpowiadające rozstawowi wyprowadzeń obudowy TO-220. Przetwornica ta została podłączona do wyjść GPIO Raspberry Pi, by pobierać napięcie 5 V i dostarczać 3,3 V. Sam wyświetlacz, dla uproszczenia połączeń, również został podłączony do GPIO komputerka.

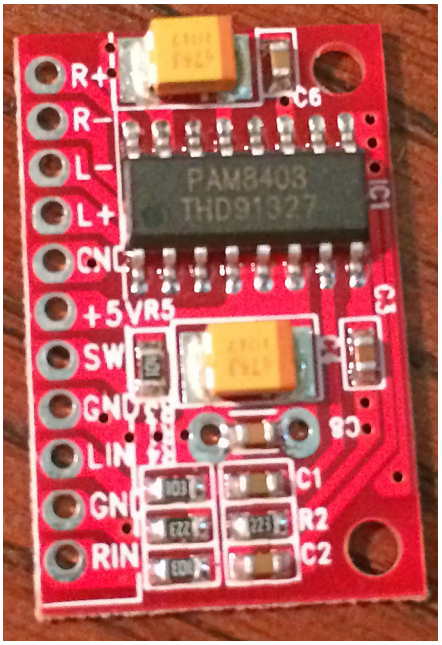
Drugi regulator służy do kontroli pracy akumulatora i podnoszenia jego napięcia do poziomu 5 V (**fotografia 7**). Zabezpiecza on akumulator i resztę urządzenia przed zwarciami. Został zamontowany w wolnej przestrzeni w bocznej części obudowy. Trzecia przetwornica (**fotografia 8**) służy ładowaniu akumulatora prądem z 5-voltowego wejścia umieszczonego w dolnej części obudowy. Umieszczono ją mniej więcej tam, gdzie znajdował się oryginalny regulator konsoli.

Dźwięk

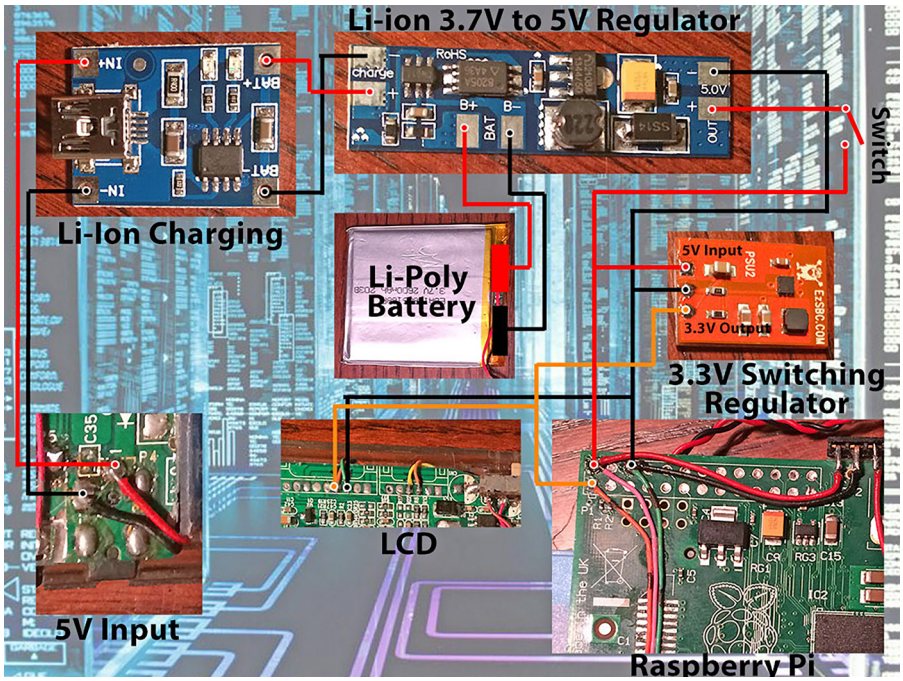
Pewnym problemem okazało się uzyskanie dźwięku na Pi-Pocket. Podstawowym problemem był fakt, że Raspberry Pi, w przeciwieństwie do PCB oryginalnego Game Boy'a, nie ma wyjścia głośnikowego. Zamiast tego ma wyjście słuchawkowe, którego poziomy napięcia zdecydowanie nie wystarczają do zasilania wbudowanego głośniczka, więc autor musiał znaleźć odpowiednią płytkę wzmacniacza (**fotografia 9**), aby móc dostarczyć wystarczający poziom sygnału



Fotografia 8. Układ ładowania akumulatora



Fotografia 9. Płytka wzmacniacza



Rysunek 11. Sposób podłączenia elementów zasilających

audio – w praktyce wystarczyło ją delikatnie przyciąć i przykleić w pozostałej wolnej przestrzeni.

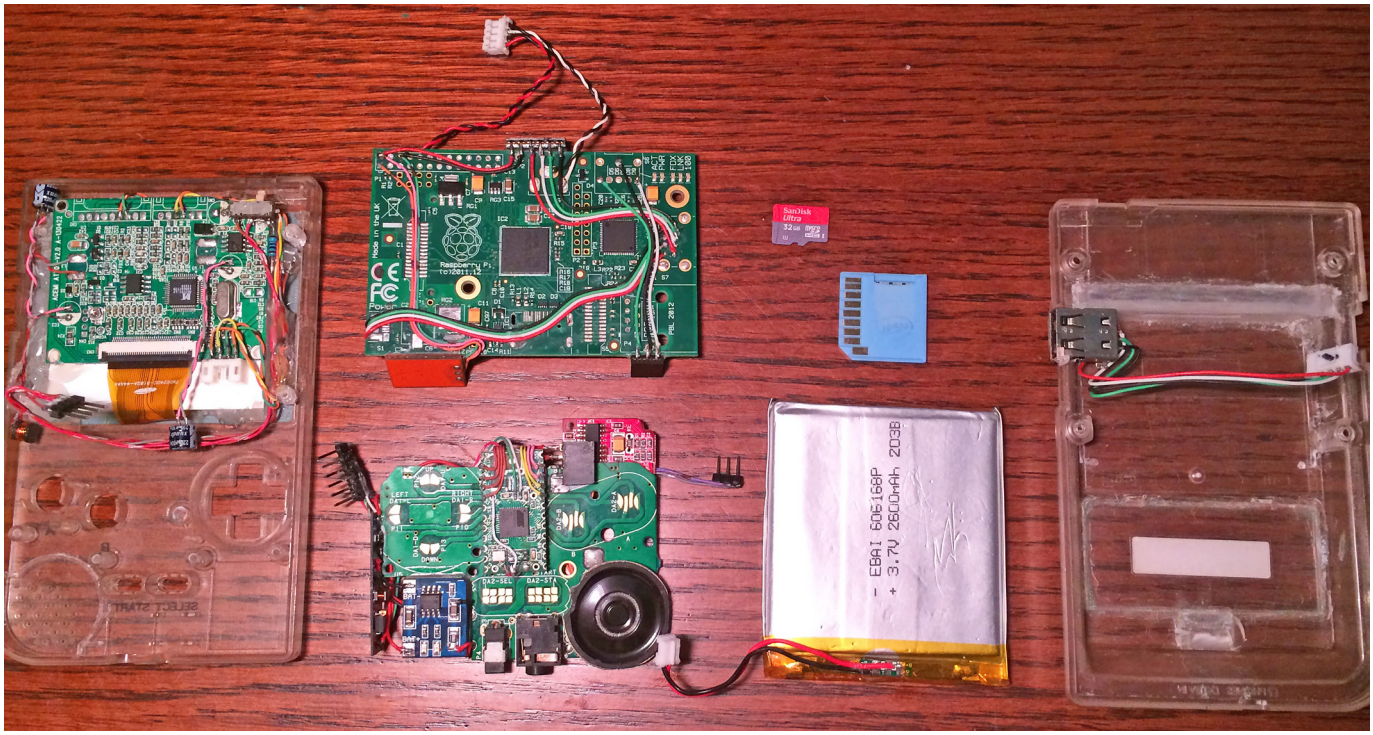
Konieczna okazała się też wymiana głośnika, gdyż oryginalny był zbyt duży, by zmieścić się wraz z pozostałymi komponentami. Na szczęście postęp technologiczny na przestrzeni niemal 20 lat, od kiedy wyprodukowano pierwsze Game Boy'e tego typu, sprawił, że obecnie na rynku znaleźć można głośniki o znacznie lepszych parametrach, a jednocześnie o mniejszych wymiarach. Kluczowe było takie dobranie elementu, aby jego średnica pasowała do otworów w obudowie, w miejscu gdzie znajdował się oryginalny głośnik. Autorowi nie udało się to idealnie, ale znalazł podzespół tylko o nieznacznie większej średnicy. Komponent miał ponadto połowę głębokości oryginalnego głośnika i kilkakrotnie większą moc. Wystarczyły tylko ponownie delikatnie przyciąć obudowę.

Trzeba było też zastosować regulator głośności – autor wykorzystał w tym celu oryginalne pokrętło, przy czym ponieważ standardowo było ono przymocowane do głównego PCB Game Boy'a, konieczne było znalezienie innego sposobu i miejsca montażu. Padło na otwór przy złączu USB, do którego to gniazda zresztą przymocowano potencjometr.

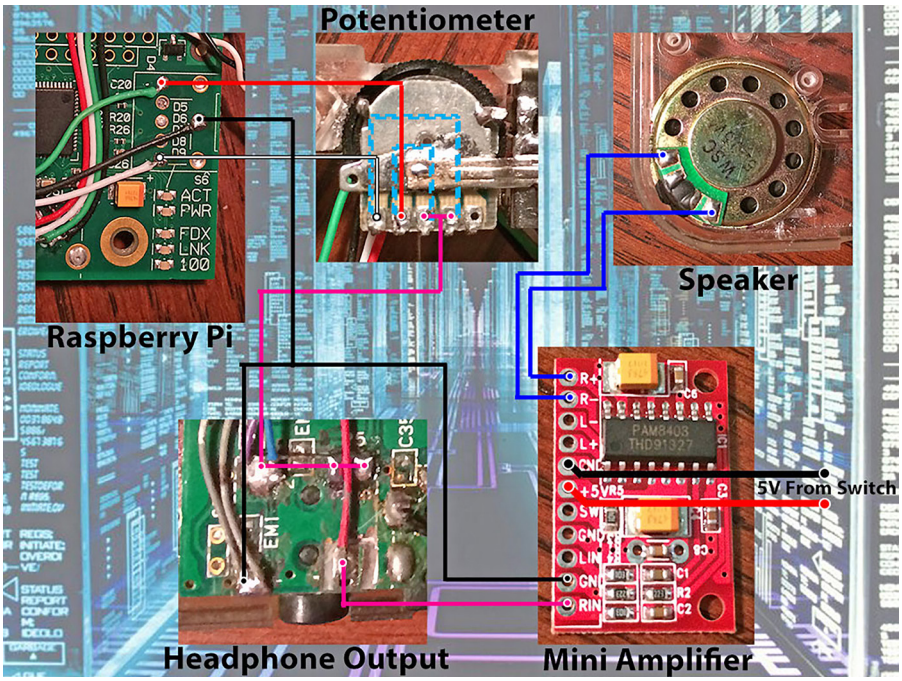
Montaż całości

Połączenie ze sobą wszystkich komponentów (**fotografia 10**) nie było łatwe, ale autorowi udało zrobić się to w taki sposób, by niektóre elementy można było swobodnie odczepić razie potrzeby, podczas gdy inne były przylutowane na stałe. Do frontowej części obudowy przymocowano wyświetlacz i płytkę jego kontrolera. Do tylnej – złącze USB za pomocą kleju szybkoschnącego. Do Raspberry Pi podłączono przewody, które pozwalają rozprowadzić napięcie 5 V w odpowiednie miejsca (**rysunek 11**).

Całość mieści się niemal idealnie, pozostawiając tylko przestrzenie o szerokości



Fotografia 10. Zebrane komponenty przed montażem



Rysunek 12. Sposób podłączenia elementów audio

Fotografia 13. Działający oryginalny Game Boy Pocket oraz Pi-Pocket z kolorowym wyświetlaczem

około 1 milimetra, dzięki czemu ciężar wewnątrz obudowy jest dosyć równomiernie rozmieszczony i trzymając urządzenie w ręce ma się wrażenie, że ma się do czynienia z oryginalnym produktem.

Oprogramowanie

Na kartę SD autor nagrał obraz systemu RetroPie. Jest on dostępny bezpłatnie, do pobrania, w wersjach na różne odmiany Raspberry Pi, ze strony *http://goo.gl/OSn8HU*. Standardowa wersja systemu automatycznie uruchamia się i emuluje około 30 różnych konsol. System RetroPie można też samodzielnie dostosować do swoich potrzeb.

Ponadto konieczne było oprogramowanie modułu Teensy. Kod źródłowy programu został pokazany na **listingu 1**.

Podsumowanie

Omówiony projekt jest dosyć żmudny w wykonaniu, ale uzyskany efekt może wynagrodzić starania. Zabierając się do wykonania takiej przeróbki warto dokładnie przeanalizować i wcześniej zaplanować rozkład komponentów w obudowie urządzenia. Autor niejednokrotnie odkrywał w trakcie pracy, że jego wcześniejsze szacunki były nazbyt optymistyczne i tak naprawdę ani większy akumulator, ani większy wyświetlacz, prawdopodobnie by się nie zmieściły. Projekt pokazuje natomiast, jak łatwo można korzystać z modułowych podzespołów, które tanim kosztem można nabyć np. na aukcjach internetowych. Niestety, jeśli policzyć łączne koszty, to wynoszą one około 140 dolarów, nie licząc kosztów przesyłki komponentów, które kupowane były u różnych sprzedawców. Miłośnikom starych gier, którzy nie mają ochoty wydawać 600 zł na komponenty do konsoli i samodzielnie jej montować można polecić instalację RetroPie na Raspberry Pi podłączonym do telewizora lub po prostu zainstalowanie emulatorów gier z RetroPie na zwykłym komputerze.

Marcin Karbowniczek, EP

Listing 1. Kod źródłowy programu modułu Teensy

```
#include <Bounce.h>

Bounce bUp = Bounce(1, 20);
Bounce bDown = Bounce(2, 20);
Bounce bLeft = Bounce(3, 20);
Bounce bRight = Bounce(0, 20);
Bounce bA = Bounce(21, 20);
Bounce bB = Bounce(20, 20);
Bounce bStart = Bounce(19, 20);
Bounce bSelect = Bounce(18, 20);

int hidMode = 0;

boolean mLeft = false;
boolean mRight = false;

void setup() {
  pinMode(0, INPUT_PULLUP);
  pinMode(1, INPUT_PULLUP);
  pinMode(2, INPUT_PULLUP);
  pinMode(3, INPUT_PULLUP);
  pinMode(18, INPUT_PULLUP);
  pinMode(19, INPUT_PULLUP);
  pinMode(20, INPUT_PULLUP);
  pinMode(21, INPUT_PULLUP);
  delay(500);
}

void loop() {
  updateButtons();
  processFallingEdges();
  processRisingEdges();
}

void updateButtons() {
  bUp.update();
  bDown.update();
  bLeft.update();
  bRight.update();
  bA.update();
  bB.update();
  bStart.update();
  bSelect.update();
}

void processFallingEdges() {
  if ((bSelect.read() == LOW)
        && (bUp.read() == LOW)) {
    hidMode = ((hidMode) ? 0 : 1);
    Keyboard.set_key1(0);
    Keyboard.set_key2(0);
    Keyboard.set_key3(0);
    Keyboard.set_key4(0);
    Keyboard.set_key5(0);
    Keyboard.set_key6(0);
    Keyboard.send_now();
    delay(500);
  }

  if ((bSelect.read() == LOW)
        && (bDown.read() == LOW)) {
    Keyboard.set_key1(KEY_ESC);
    Keyboard.set_key2(0);
    Keyboard.set_key3(0);
    Keyboard.set_key4(0);
    Keyboard.set_key5(0);
    Keyboard.set_key6(0);
    Keyboard.send_now();
    delay(100);
    Keyboard.set_key1(0);
    Keyboard.send_now();
    delay(500);
    return;
  }

  if (hidMode == 0) {
```

Listing 1. c.d.

```
    if (bUp.fallingEdge()) {
      Keyboard.set_key1(KEY_U);
      Keyboard.send_now();
    }

    if (bDown.fallingEdge()) {
      Keyboard.set_key1(KEY_D);
      Keyboard.send_now();
    }

    if (bLeft.fallingEdge()) {
      Keyboard.set_key2(KEY_L);
      Keyboard.send_now();
    }

    if (bRight.fallingEdge()) {
      Keyboard.set_key2(KEY_R);
      Keyboard.send_now();
    }

    if (bA.fallingEdge()) {
      Keyboard.set_key3(KEY_A);
      Keyboard.send_now();
    }

    if (bB.fallingEdge()) {
      Keyboard.set_key4(KEY_B);
      Keyboard.send_now();
    }

    if (bStart.fallingEdge()) {
      Keyboard.set_key5(KEY_S);
      Keyboard.send_now();
    }

    if (bSelect.fallingEdge()) {
      Keyboard.set_key6(KEY_E);
      Keyboard.send_now();
    }
  }
    if (hidMode == 1) {
      int moveSpeed = max(((bUp.
read() == LOW) ? bUp.duration() :
0), ((bDown.read() == LOW) ? bDown.
duration() : 0));
      moveSpeed = max(moveSpeed,
max(((bLeft.read() == LOW) ? bLeft.
duration() : 0), ((bRight.read() ==
LOW) ? bRight.duration() : 0)));
      moveSpeed = min(1 + (moveSpeed /
250), 10);
      if (bUp.read() == LOW) {
        Mouse.move(0, -moveSpeed);
      }
      if (bDown.read() == LOW) {
        Mouse.move(0, moveSpeed);
      }
      if (bLeft.read() == LOW) {
        Mouse.move(-moveSpeed, 0);
      }
      if (bRight.read() == LOW) {
        Mouse.move(moveSpeed, 0);
      }
      if (bA.fallingEdge()) {
        mLeft = true;
        Mouse.set_buttons(mLeft, false,
mRight);
      }
      if (bB.fallingEdge()) {
        mRight = true;
        Mouse.set_buttons(mLeft, false,
mRight);
      }
      delay(10);
    }
}

void processRisingEdges() {
  if (hidMode == 0) {
    if ((bUp.risingEdge()) || (bDown.
risingEdge())) {
      Keyboard.set_key1(0);
      Keyboard.send_now();
    }
    if ((bLeft.risingEdge()) ||
(bRight.risingEdge())) {
      Keyboard.set_key2(0);
      Keyboard.send_now();
    }
    if (bA.risingEdge()) {
      Keyboard.set_key3(0);
      Keyboard.send_now();
    }
    if (bB.risingEdge()) {
      Keyboard.set_key4(0);
      Keyboard.send_now();
    }
    if (bStart.risingEdge()) {
      Keyboard.set_key5(0);
      Keyboard.send_now();
    }
    if (bSelect.risingEdge()) {
      Keyboard.set_key6(0);
      Keyboard.send_now();
    }
  }

  if (hidMode == 1) {
    if (bA.risingEdge()) {
      mLeft = false;
      Mouse.set_buttons(mLeft, false,
mRight);
    }
    if (bB.risingEdge()) {
      mRight = false;
      Mouse.set_buttons(mLeft, false,
mRight);
    }
  }
}
```