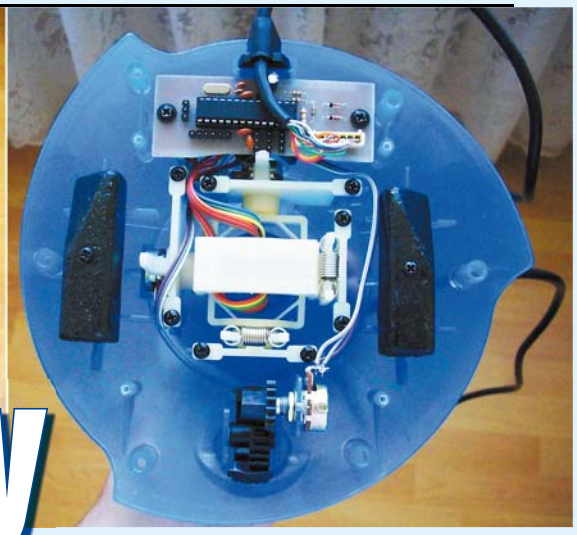
kit 2991 AVT



USB HID Joy

Do czego służy?

Projekt przedstawia przykład nietypowej przeróbki i podłączenia do gniazda USB joysticka przeznaczonego dla portu GAMEPORT. Gniazdo GAMEPORT obecnie nie występuje w komputerach – wyparł je port USB. Oczywiście można kupić gotowy joystick USB, ale ambitnym i interesującym zadaniem jest przeróbka starego, darzonego sentymentem joysticka.

Łącze USB niesie bardzo wiele możliwości, a rodzaj współpracującego urządzenia nie ma znaczenia. Jednak wymaga pisania sterowników i programów do ich obsługi, co do prostych rzeczy nie należy. A co, jeśli nie trzeba nic pisać, ponieważ system ma już wszystkie sterowniki? Tak właśnie jest z urządzeniami HID. Są to m.in. klawiatury, myszki, joysticki podłączane do portu USB. Prezentowana przeróbka wykorzystuje właściwości joysticka HID, nie jest zanadto skomplikowana i w większości przypadków będzie wyglądać podobnie. Niski koszt i nakład pracy oraz satysfakcja z modyfikacji odstawionego w kąt joysticka powinna Cię do tego zachęcić. Całość pisałem w darmowym środowisku AVR Studio i kompilatorze WinAVR.

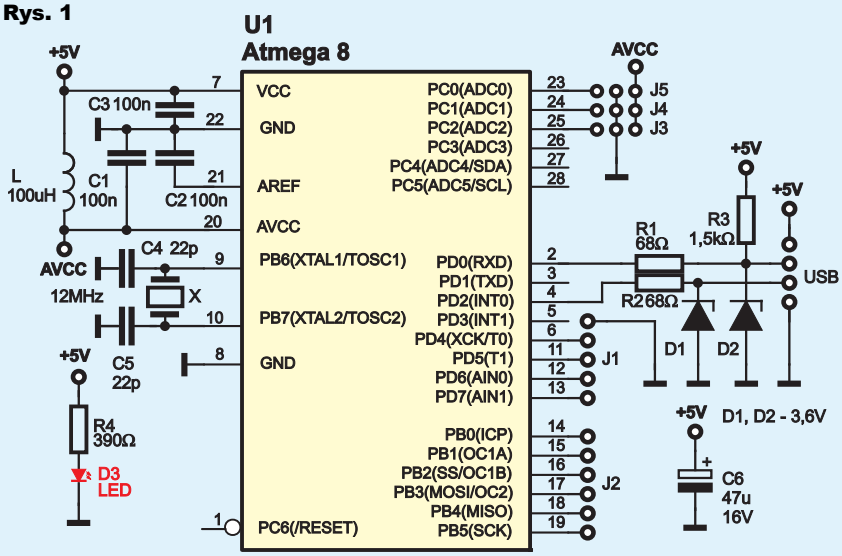
Jak to działa?

Na **rysunku 1** widzimy schemat ideowy. Nie jest on skomplikowany, ponieważ całą pracę wykonuje mikrokontroler U1 ATmega8. Elementy R1, R2, D1 i D2 tworzą prosty układ dopasowujący, ponieważ mikrokontroler pracuje przy napięciach +5V, a linie portu USB przy +3,3V. R3 podciąga linię D– do plusa, informując o podłączeniu urządzenia USB LOW SPEED. C1, C2 i L służą do filtrowania napięcia odniesienia, służącego do zasilania potencjometrów analogowych osi. C3 i C6 filtrują napięcie zasilania. Rezonator X 12MHz wraz z kondensatorami C4, C5 określają zegar mikrokontrolera. R4 wraz z diodą LED sygnalizują podłączenie joysticka. Do złącza J1 podłączony jest przełącznik widoków – HAT SWITCH, do J2 sześć przycisków, natomiast do J3...J5 – potencjometry analogowych osi X, Y i przepustnicy.

Część sprzętowa jest typowa. Mimo że EdW to czasopismo dla elektroników, muszę trochę miejsca poświęcić opisowi deskryptora HID oraz oprogramowaniu, z czego istotna dla nas część znajduje się w plikach *main.c* oraz *usbconfig.h*, dostępnych w Elportalu. Pozostałe umieszczone tam pliki stanowią programową bibliotekę do obsługi USB przez mikrokontrolery AVR, którą nie trzeba sobie zaprzątać głowy. Dalsze szczegóły, wraz z przykładami, są pod adresem *www.obdev.at/products/vusb/index.html*.

Deskryptor USB i HID. Podczas podłączania jakiegokolwiek urządzenia do portu USB przedstawia się ono, dlatego komputer może je zidentyfikować i załadować odpowiedni sterownik. Tak się dzieje w przypadku wszystkich urządzeń, ale urządzenia HID oprócz podstawowych informacji przesyłają też dane o budowie urządzenia (liczba przycisków, osi itd.). Zobaczmy, jak to wygląda

Rys. 1




```
PROGMEM char usbHidReportDescriptor[78] =
{
/*USB report descriptor, size must match usbconfig.h */
/*USAGE_PAGE (Generic Desktop)*/     0x05, 0x01,
/*LOGICAL_MINIMUM (0)*/              0x15, 0x00,
/*USAGE (Joystick)*/                 0x09, 0x04,
/*COLLECTION (Application)*/         0xA1, 0x01,
/* USAGE_PAGE (Simulation Controls)*/0x05, 0x02,

/*   USAGE (Throttle)*/              0x09, 0xBB,
/*   LOGICAL_MINIMUM (0)*/           0x15, 0x00,
/*   LOGICAL_MAXIMUM (255)*/    0x26, 0xFF, 0x00,
/*   REPORT_SIZE (8)*/               0x75, 0x08,
/*   REPORT_COUNT (1)*/              0x95, 0x01,
/*   INPUT (Data,Var,Abs)*/          0x81, 0x02,

/*   USAGE_PAGE (Generic Desktop)*/  0x05, 0x01,
/*   USAGE (Pointer)*/               0x09, 0x01,
/*   COLLECTION (Physical)*/         0xA1, 0x00,
/*     USAGE (X)*/                   0x09, 0x30,
/*     USAGE (Y)*/                   0x09, 0x31,
/*     REPORT_COUNT (2)*/            0x95, 0x02,
/*     INPUT (Data,Var,Abs)*/        0x81, 0x02,
/*   END_COLLECTION*/                0xC0,

/*   USAGE (Hat switch)*/            0x09, 0x39,
/*   LOGICAL_MINIMUM (0)*/           0x15, 0x00,
/*   LOGICAL_MAXIMUM (7)*/           0x25, 0x07,
/*   PHYSICAL_MINIMUM (0)*/          0x35, 0x00,
/*   PHYSICAL_MAXIMUM (315)*/   0x46, 0x3B, 0x01,
/*   UNIT (Eng Rot:Angular Pos)*/    0x65, 0x14,
/*   REPORT_SIZE (4)*/               0x75, 0x04,
/*   REPORT_COUNT (1)*/              0x95, 0x01,
/*   INPUT (Data,Var,Abs)*/          0x81, 0x02,

/*   USAGE_PAGE (Button)*/           0x05, 0x09,
/*   USAGE_MINIMUM (Button 1)*/      0x19, 0x01,
/*   USAGE_MAXIMUM (Button 6)*/      0x29, 0x06,
/*   LOGICAL_MINIMUM (0)*/           0x15, 0x00,
/*   LOGICAL_MAXIMUM (1)*/           0x25, 0x01,
/*   REPORT_SIZE (1)*/               0x75, 0x01,
/*   REPORT_COUNT (12)*/             0x95, 0x0c,
/*   UNIT_EXPONENT (0)*/             0x55, 0x00,
/*   UNIT (None)*/                   0x65, 0x00,
/*   INPUT (Data,Var,Abs)*/          0x81, 0x02,
/*END_COLLECTION*/                   0xC0,
};
```

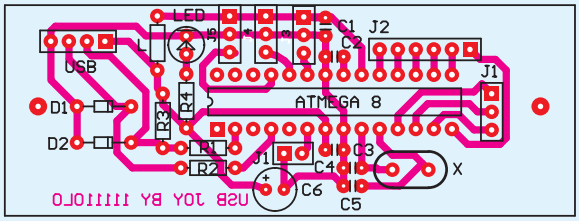
Listing 1

7	6	5	4	3	2	1	0
Przepustnica							
Os X							
Oś Y							
Przycisk 4	Przycisk 3	Przycisk 2	Przycisk 1	Hat 3	Hat 2	Hat 1	Hat 0
Pusty bit	Pusty bit	Pusty bit	Pusty bit	Pusty bit	Pusty bit	Przycisk 6	Przycisk 5

Tab. 1

Listing 2

```
typedef struct{
    uint8_t    dz;              // oś z
    uint8_t    dx;              // oś x
    uint8_t    dy;              // oś y
    uint8_t    hat_button;      // 4 hat i 4 button
    uint8_t    button;          // 2 button
}report_t;
```



Rys. 2

da na przykładzie joysticka z 3 osiami (z czego jedna jest przypisana do przepustnicy), ośmiokierunkowym przełącznikiem widoków (hat switch) oraz sześcioma przyciskami. W **listingu 1** widać taki deskryptor, który ma długość 78 bajtów. Kolejno są definiowane bity, które będą przesyłane między joystickiem a komputerem (tu akurat istotne dane są przesyłane w jednym kierunku) – w sumie 40 bitów wyszczególnionych w **tabeli 1.**

Nie będę szczegółowo opisywał deskryptora, gdyż w kilku zdaniach jest to niemożliwe. Zrozumienie tego zajęło mi jakieś 3 tygodnie, co jednak nie oznacza, że to takie trudne. Po prostu dużo czasu zeszło mi na wyłapanie, że trzeba wysyłać po całym bajcie, mimo że cały nie jest wykorzystywany, stąd 6 pustych bitów na końcu. Cała dokumentacja, w formacie PDF, licząca 168 stron, znajduje się pod adresem *www.usb.org/developers/hidpage*. Oprócz dokumentacji jest tam bardzo przydatny program *HID Descriptor Tool*, który pomaga przy składaniu deskryptora.

Jeśli mamy już deskryptor, trzeba jeszcze wszystkie obsługiwane osie i przyciski przypisać do zmiennych. Widać to na **listingu 2**. Nazwy nie mają znaczenia dla kompilatora, a jedynie dla nas. Dalsza część pliku *main.c* nie jest skomplikowana i po kolei sprawdza stan potencjometrów, hat switcha i przycisków i je odpowiednio przypisuje do zmiennych, które widać w listingu 2.

Co bardziej uważny Czytelnik zauważy, że przyciski nie występują po kolei, co zmusza do bitowego przesuwania. Z ważniejszych rzeczy jest jeszcze ograniczenie pomiaru ADC do ośmiu istotnych bitów. Udało się to osiągnąć w prosty sposób, gdyż ATmega zapisuje wynik pomiaru w dwóch komórkach na dwa sposoby: dwa bardziej istotne bity + osiem mniej istotnych lub osiem istotnych + dwa mniej istotne. Za te zmiany odpowiada rejestr ADLAR w ATmedze – szczegóły w karcie katalogowej. W ten sposób bez nadmiarowych sampli uzyskałem dokładniejszy pomiar kosztem rozdzielczości, jednak ta okazała się wystarczająca.

Montaż i uruchomienie

Montaż można wykonać na płytce przedstawionej na **rysunku 2**. Montaż jest typowy. Zaczynamy od najmniejszych elementów a kończymy na podstawce pod ATmegę. Dodatkowo trzeba wgrać wsad, który jest dostępny w Elportalu i skonfigurować fusebity według **rysunku 3**. Do łączenia joysticka z komputerem wykorzystałem oryginalny kabel, który pierwotnie był zakończony wtykiem GAMEPORT. Po jego odcięciu wykorzystałem cztery żyły, które przylutowałem do wtyczki USB. Wewnątrz joysticka trzeba podłączyć przyciski, hat switch oraz potencjometry. Nie jest to trudne, gdyż wszystkie takie joysticki reagują na zwarcie do masy. Na złączu J1 jest tylko jedno wyprowadzenie masy, gdyż w rękojeści prawie wszystkie przyciski są na wspólnej płytce. Tu musiałem rozdzielić zdublowane przyciski przez przecinanie ścieżek i wylutowanie zworek – normalnie były cztery przyciski (dwa zdublowane). Kolejność przycisków nie ma znaczenia, jednak warto je poukładać w sensownej kolejności. Jeśli chodzi o hat switch, trzeba przewody podłączyć w odpowiedniej kolejności, aby kierunki ruchu manetką były zgodne z rzeczywistością.

Najwięcej problemów sprawiły mi potencjometry. Wiem, że to powinno pójść gładko, jednak standardowe sprawdzanie w gameporcie polega na pomiarze stałej czasowej potencjometru z kondensatorem. Niby to nie powinno przeszkodzić, jednak z oszczędności jedno wyprowadzenie potencjometru nie miało zaciśniętej końcówki do lutowania. Musiałem ją przykręcić za pomocą śrubki, dodatkowo nieco rozwiercając standardowy otwór, aby śruba dobrze się wkręciła. Podłączenie potencjometru jest typowe, z tym że suwak trzeba podpiąć do wejść ADC ATmegi, a dwa pozostałe przewody do skrajnych wyprowadzeń potencjometru. Przy błędnym podłączeniu, wychylenie będzie przeciwne z założeniem. Ważne jest, aby nie wymieniać potencjometrów, gdyż może być problem z ich montażem lub zmniejszy to ich zakres pomiaru.

Dodanie kolejnej osi lub przycisku jest proste – HID przewiduje maksymalnie 8 analogowych osi, ośmiokierunkowy hat switch i 64 przyciski. Trzeba mieć na uwadze, że ATmega8 nie ma za wiele wolnych pinów, co zmusza do zamiany na np. ATmegę16 lub sprawdzanie przycisków w układzie matrycy. Wiąże się to ze zmianą deskryptora oraz oprogramowania. Oprócz tego można zmienić nazwę, pod jaką przedstawia się joystick – w pliku usbconfig.h. Jest tam również możliwa zmiana podłączenia linii USB D– (D+ musi być podłączona do INT0!). Dodatkowo długość deskryptora USB znajduje się w tym pliku i musi być identyczna jak w pliku *main.c*.

Aleksander Bernaczek
olo11111@gmail.com

Wykaz elementów

R1, R2	68Ω
R3	1,5kΩ
R4	390Ω
C1, C2, C3	100nF
C4, C5	22pF
C6	47uF/16V
L	100uH
D1, D2	3V6
D3	LED
X	12MHz
U1	ATmega 8
J1	5 pin
J2	6 pin
J3 - J5	3 pin
USB	4 pin
Podstawka 28 pin wąska	

Komplet podzespołów z płytką jest dostępny w sieci handlowej AVT jako kit szkolny AVT-2991.

Rys. 3