# Systemy Wbudowane

## Arduino – dołączanie urządzeń

Wersja 2022

dr inż. Marek Wilkus
Wydział Inżynierii Metali i Informatyki Przemysłowej
AGH Kraków
http://home.agh.edu.pl/~mwilkus



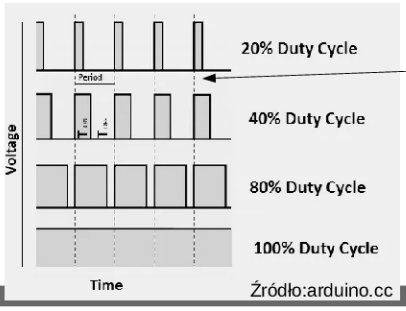
## PWM

- Wyjścia cyfrowe pozwalają nam na włączanie i wyłączanie urządzeń. Jak jednak sterować płynnie?
  - Kolorami LEDa?
  - Szybkością silnika?
  - Jasnością wyświetlacza?
- Nie możemy zmieniać poziomu stanów HIGH i LOW, którymi włączamy urządzenie.
- A jeżeli będziemy włączali urządzenie na krótszy lub dłuższy czas i robili to dostatecznie szybko, by bezwładność spowodowała, że sprawia to wrażenie pracy ciągłej?



## PWM

- PWM – Pulse width modulation – modulacja szerokością impulsu – to właśnie takie sterowanie.
- Możemy regulować **współczynnik wypełnienia** czyli to, czy w danym okresie impulsy będą "chudsze" czy "grubsze".




Źródło:arduino.cc



## PWM

- Piny, które sprzętowo realizują PWM oznaczane są na płytce znaczkiem tyldy: ~
- W programie pin ustawiamy jako **OUTPUT**.
- Następnie wykorzystujemy funkcję:

```
analogWrite( PIN, współczynnik );
```

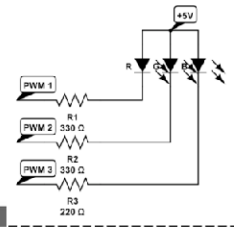
Numer lub definicja pinu Umożliwiającego PWM (na płytce oznaczony znakiem ~ )

To jest **byte:**
0 – 0% wypełnienia
127 – 50% wypełnienia
255 – 100% wypełnienia



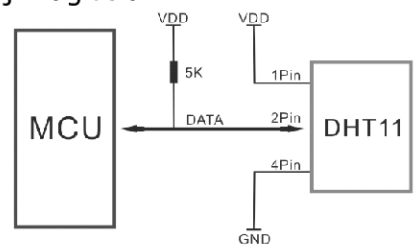
## I tak...

- Jednym pinem PWM i odpowiednim sterownikiem możemy kontrolować szybkość obrotów silnika DC.
- Trzema pinami PWM możemy kontrolować LED RGB – bowiem LED RGB to trzy diody świecące: czerwona, zielona i niebieska.
  - Uwaga: Diody RGB mają często wspólną **anodę**. Oznacza to, że sterujemy katodami (masą). Wówczas im mniejszy współczynnik wypełnienia, tym więcej stanu niskiego i tym jaśniej świeci LED podłączony za wspólną anodę do +5V.
  - Wówczas musimy stosować 3 rezystory – dla każdego koloru.
  - Jeżeli kolory świecą nierównomiernie, możemy zastosować różne wartości rezystorów ---->




## DHT11 jeszcze raz...

- Sensor temperatury i wilgotności działający na jednoprzewodowej magistrali.
- Zasilanie: 5V
- Podłączenie: --->
- Komunikacja dwustronna
- 0..50 *C, max 80RH
- Uwaga na pin 3/4!



## DHT-11 - protokół

- Zapytanie z MCU: Linia komunikacyjna w stan niski na 18ms.
- Później z powrotem w stan wysoki.
- DHT11 wysyła stan niski (80µs)->wysoki(80µs)
- DHT11 wysyła dane (po ok. 40µs):
  - 1: 8 bit: Wilgotność
  - 2: 8 bit: Wilgotność, część dziesiętna
  - 3: 8 bit: Temperatura
  - 4: 8 bit: Temperatura, część dziesiętna
  - 5: 8 bit: Suma kontrolna:
    - Ostatnie 8 bit z 1+2+3+4



## DHT-11: Transmisja

- Każdy bit zaczyna się 50µs stanem niskim, czas późniejszego stanu wysokiego identyfikuje czy przesyłane jest 0 czy 1:

  – 26-28µs – 0

  – 70µs – 1

- Następnie w ten sam sposób przesyłany jest kolejny bit.

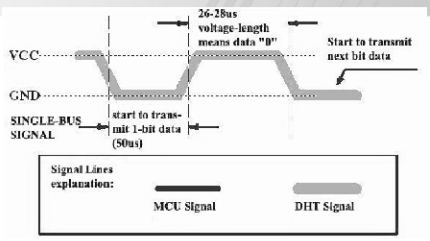



Figure 4 Data "0" Indication

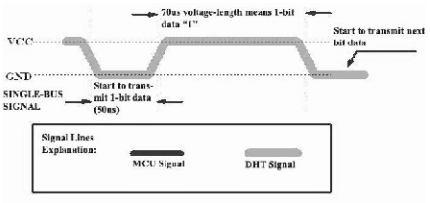
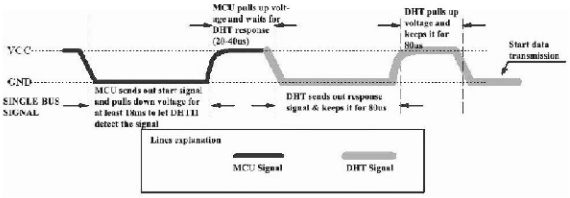



Figure 5 Data "1" Indication

## DHT-11: Implementacja

```
//Zapytanie:
pinMode(pin, OUTPUT);
digitalWrite(pin, LOW);
delay(18);
digitalWrite(pin, HIGH);
delayMicroseconds(40);
pinMode(pin, INPUT);

//Czekamy na dane
while(digitalRead(pin) == LOW)
    {...} //Obsługa time-out'u

while(digitalRead(pin) == HIGH)
    {...} //Obsługa time-out'u
```






## DHT-11: Odbiór danych

```
byte data[5];       //dla przejrzystości pominięto zerowanie tablicy
byte currentByte=0;
byte currentBit=7;
for (int i=0; i<40; i++)
{
    while(digitalRead(pin) == LOW)
        {. . .} //Obsługa time-out'u

    unsigned long t = micros(); //pomiar czasu - start

    while(digitalRead(pin) == HIGH)
        {. . .} //Obsługa time-out'u

    if ((micros() - t) > 30) data[currentByte] |= (1 << currentBit);

    if (currentBit == 0) // następny bajt
    {
        currentBit = 7;
        currentByte++;
    }
    else
        currentBit--;
}
```



## Arduino bez płytki Arduino

- Kompletne Arduino Uno jest 3-4x droższe od samego mikrokontrolera,
- Do danego układu niekoniecznie potrzebne są wszystkie oferowane przez moduł Arduino Uno urządzenia, np.
  - Port szeregowy przez USB
  - Słabe stabilizatory zasilania
  - Słaby stabilizator 3.3V

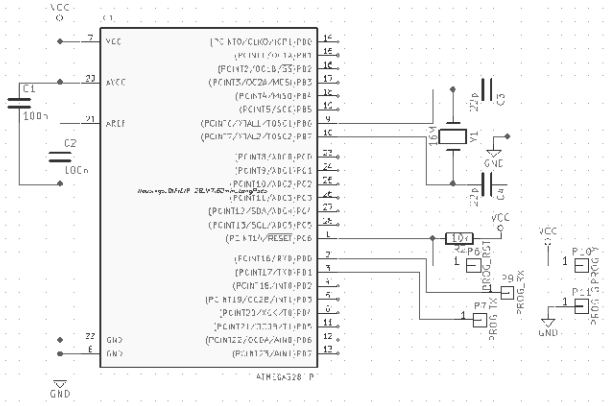
- Rozwiązanie: Użycie samego chipu ATMega328 (lub ATMega8 – 8kB na program, chip jeszcze tańszy) we własnym układzie



## Co musimy mieć, aby uruchomić uC?

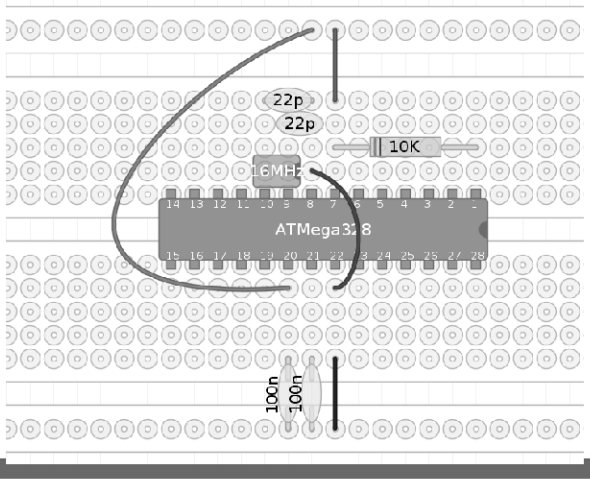
- Mikrokontroler
- Zasilanie +5V,
- Źródło częstotliwości taktowania (16MHz, względnie 8MHz),
- Podciągnięcie pinu RESET do stanu wysokiego,
- Kondensator odsprzęgający.

## Czyli...





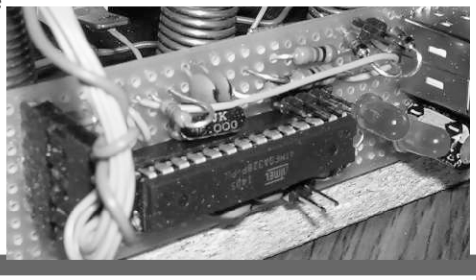
## Na płytce...





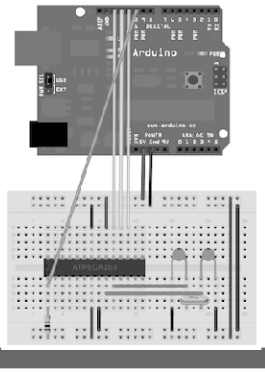

## Eliminowanie modułu z projektu

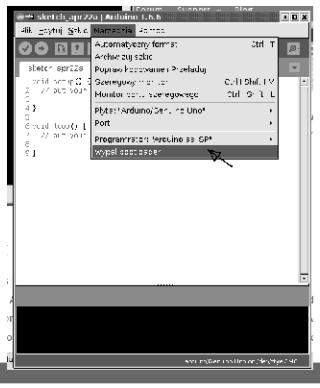
- Zapisujemy do Arduino końcową wersję programu,
- Po skończonym testowaniu z modułem wykonujemy i testujemy podstawowy układ bez modułu:
  - Podstawa: Zapewnienie zasilania, taktowania , rezystora i kondensatorów,
  - Tłumaczenie pinów Arduino na piny ATMega, odpowiednie podłączenia,
  - Próba na płytce stykowej z układem usuniętym z Arduino,
- Wykonanie końcowej wersji układu.



## A co z Arduino?

- Programowanie nowego, czystego układu AVR jako Arduino przy użyciu istniejącego modułu Arduino ze szkicem ArduinoISP:



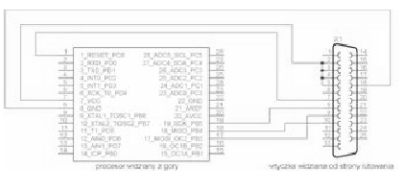




## Najprostszy programator: „bsd”

- Działa przy użyciu portu równoległego,
- Bardzo prosta budowa,
- Działa niezależnie od Arduino – przyjmuje pliki HEX bootloadera oraz fuse bity,
- Program avrdude działa dla każdej platformy (w Linuksie niezbędne uprawnienia administratora).

| ATMega | LPT |
|---|---|
| Vcc | 2+3+4+5 |
| /RESET | 7 |
| SCK | 8 |
| MOSI | 9 |
| MISO | 10 |
| GND | 18 |





## Programator „bsd”

- Użytkowanie przy pomocy programu avrdude:
- Czyścimy układ:
  - `avrdude -p m328p -c bsd -e`
- `Wgrywamy plik HEX:`
  - `avrdude -p m328p -c bsd -U flash:w:cpu.hex`
- `Ustawiamy fuse bity:`
  - `avrdude -p m328p -c bsd -u -U hfuse:w:0xC9:m -U lfuse:w:0xFF:m`

***Od tej pory układ nie będzie widoczny dla programatora dopóki nie dostarczymy źródła częstotliwości!***

## Plik .hex? Fusebity?

- Plik „hardware/arduino/avr/boards.txt” w instalacji Arduino:

```
uno.name=Arduino/Genuino Uno

uno.vid.0=0x2341
uno.pid.0=0x0043
uno.vid.1=0x2341
uno.pid.1=0x0001
uno.vid.2=0x2A03
uno.pid.2=0x0043
uno.vid.3=0x2341
uno.pid.3=0x0243

uno.upload.tool=avrdude
uno.upload.protocol=arduino
uno.upload.maximum_size=32256
uno.upload.maximum_data_size=2048
uno.upload.speed=115200
```

```
uno.bootloader.tool=avrdude
uno.bootloader.low_fuses=0xFF
uno.bootloader.high_fuses=0xDE
uno.bootloader.extended_fuses=0x05
uno.bootloader.unlock_bits=0x3F
uno.bootloader.lock_bits=0x0F
uno.bootloader.file=optiboot/
optiboot_atmega328.hex

uno.build.mcu=atmega328p
uno.build.f_cpu=16000000L
uno.build.board=AVR_UNO
uno.build.core=arduino
uno.build.variant=standard
```



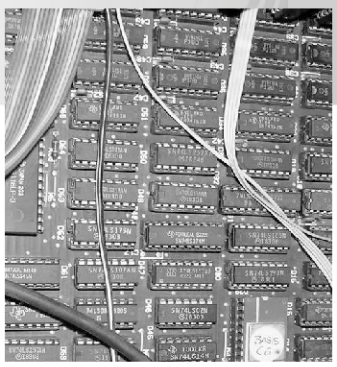
## Arduino – więcej portów I/O

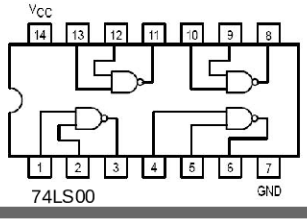
- Użycie pinów analogowych
- Liczniki
- Multipleksery
- Rejestr przesuwny
- Zatrzaski
- Drugi uC
- 8255
- Komercyjne ekspandery



## Układy serii 74

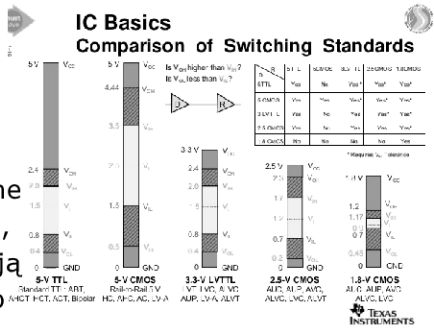
- Realizują proste funkcje logiczne: bramki, liczniki, rejestry.
- Poziomy TTL
- Są „cegiełkami” do budowy bardziej złożonych układów
- Bezpośrednio podłączane
- Podciągnięte wejścia
- Wyjścia normalne i z otwartym kolektorem
- Szeroko dostępne w różnych wykonaniach





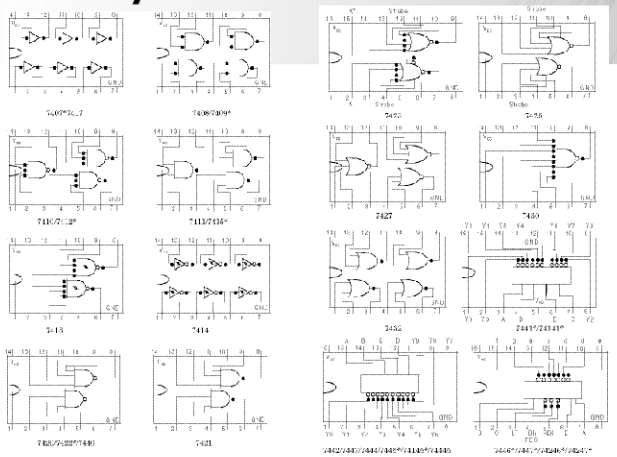

## Układy serii 74xx a seria 40xx

- Poziomy CMOS,
- Podobne funkcje, inne wyprowadzenia,
- Zasilanie do 15V,
- Mogą je uszkodzić wyładowania elektrostatyczne (szczególnie starsze układy),
- Do połączenia z 74 wymagają rezystora podciągającego do poziomu wysokiego **CMOS**.
- Kompatybilne z TTL i CMOS: 74H**CT**...






## Układy serii 74



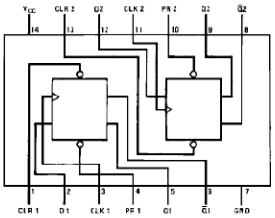
http://www.globalspec.com/reference/4290/348308/appendix-3-pin-configuration-of-74-series-integrated-circuits



## Seria 74 – do czego służy układ?

- **Nota katalogowa**
  - Tablica prawdy
  - Schemat logiczny
- Zastosowania
- Eksperymenty

**Connection Diagram**




**Function Table**

| Inputs | | | | Outputs | |
|---|---|---|---|---|---|
| PR | CLR | CLK | D | Q | $\overline{Q}$ |
| L | H | X | X | H | L |
| H | L | X | X | L | H |
| L | L | X | X | H (Note 1) | H (Note 1) |
| H | H | ↑ | H | H | L |
| H | H | ↑ | L | L | H |
| H | H | L | X | $Q_0$ | $\overline{Q}_0$ |

H = HIGH Logic Level
X = Either LOW or HIGH Logic Level
L = LOW Logic Level
↑ = Positive-going transition
$Q_0$ = The output logic level of Q before the indicated input conditions were established.
Note 1: This configuration is nonstable; that is, it will not persist when either the preset and/or clear inputs return to their inactive (HIGH) level.

74LS74

## Użycie wyprowadzeń analogowych

+ Nie są potrzebne dodatkowe biblioteki,
+ Nie jest potrzebny dodatkowy sprzęt,
+ Samo programowanie,

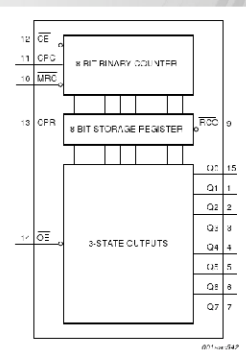
Pin 14 - 19

- Brak PWM,
- Kosztem ADC,
- Tylko 6 pinów



## Licznik

+ Rozszerzanie wyjść,
+ Tanie i dostępne układy
+ Dużo wyjść (np. 74HC590 – 8szt)
+ Linia OE.



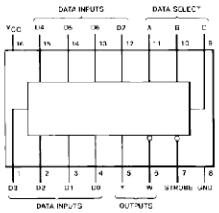

- Tylko wyjścia,
- Potrzebny czas na „wyklikanie" stanu,
- Dodatkowy układ.

| Inputs | | | | | Description |
|---|---|---|---|---|---|
| $\overline{OE}$ | CPR | $\overline{MRC}$ | $\overline{CE}$ | CPC | |
| H | X | X | X | X | Q outputs disable |
| L | X | X | X | X | Q outputs enable |
| X | ↑ | X | X | X | counter data stored into register |
| X | ↓ | X | X | X | register stage is not changed |
| X | X | L | X | X | counter clear |
| X | X | H | L | ↑ | advance one count |
| X | X | H | L | ↓ | no count |
| X | X | H | H | X | no count |

## Multiplekser

+ Zarówno wejścia jak i wyjścia
+ Możliwy przesył szeregowy danych z kilku pinów.
+ Możliwe przełączanie przełączanych sygnałów – łączenie kaskadowe.

- Szybkość działania,
- Wysoka cena układów o dużej szybkości.



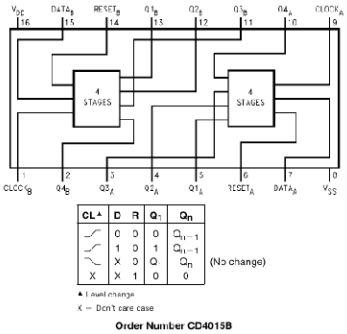

| Inputs | | | | Outputs | |
|---|---|---|---|---|---|
| Select | | | Strobe | | |
| C | B | A | S | Y | W |
| X | X | X | H | L | H |
| L | L | L | L | $D_0$ | $\overline{D_0}$ |
| L | L | H | L | $D_1$ | $\overline{D_1}$ |
| L | H | L | L | $D_2$ | $\overline{D_2}$ |
| L | H | H | L | $D_3$ | $\overline{D_3}$ |
| H | L | L | L | $D_4$ | $\overline{D_4}$ |
| H | L | H | L | $D_5$ | $\overline{D_5}$ |
| H | H | L | L | $D_6$ | $\overline{D_6}$ |
| H | H | H | L | $D_7$ | $\overline{D_7}$ |

H = HIGH Level
L = LOW Level
X = Don't Care

## Rejestr przesuwny

+ Możliwość znacznego rozszerzania wyjść,
+ „Serial input - parallel output",
+ Niska cena.

- Tylko wyjścia
- Konieczność załadowania stanu.
- Dodatkowy układ (można je łączyć w kaskadę).




| CL | D | R | $Q_1$ | $Q_n$ | |
|---|---|---|---|---|---|
| ↑ | 0 | 0 | 0 | $Q_{n-1}$ | |
| ↑ | 1 | 0 | 1 | $Q_{n-1}$ | |
| 0 | X | 0 | Q | $Q_n$ | (No change) |
| X | X | 1 | 0 | 0 | |

↑ = Level change
X = Don't care case

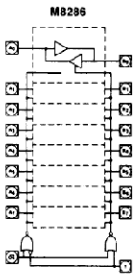
Order Number CD4015B



## Przełączniki

+ Proste przełączanie dużej ilości wyprowadzeń,
+ Możliwy wybór kierunku,

- Wyższa cena układów,
- Mniejsza popularność,
- Konsekwencje w przypadku uszkodzenia

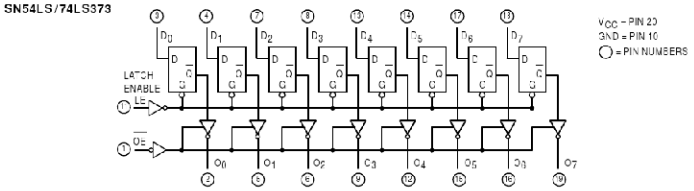




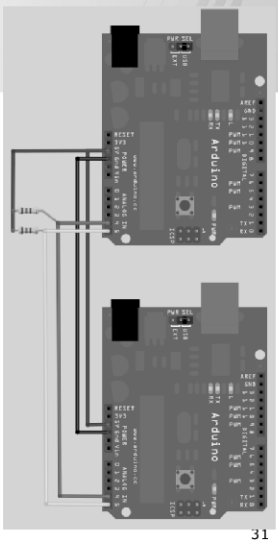

## Zatrzaski

+ Szybkie przełączanie wyprowadzeń
+ Multipleksowanie

- Skomplikowane sterowanie
- Możliwość uszkodzenia
- Tylko jeden kierunek



## Drugi uC



+ Łatwość użycia
+ Biblioteki
+ Wejście/wyjście, ADC, PWM.

- Wymaga oprogramowania
- Cena
- Niższa szybkość.



## 8255 i podobne

+ Wejście i wyjście,
+ Programowanie przez zapis wartości
+ 24 piny I/O

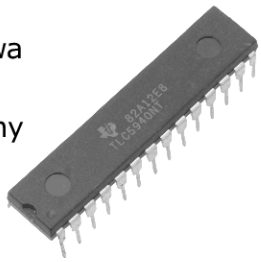
- Wymagania mikroprocesorowe, nie dla mikrokontrolera (konieczność emulacji sygnałów),
- Niska prędkość,
- Wyższa cena układów




## Programowalny sterownik

+ Dodatkowe funkcje (np. PWM),
+ Łatwiejsze programowanie,
+ Najczęściej tylko jeden układ.

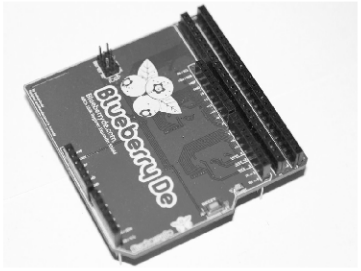
- Często niska wydajność prądowa
- Wysoka cena
- Specjalizowany układ - Problemy z przyszłą dostępnością.





## Komercyjne ekspandery

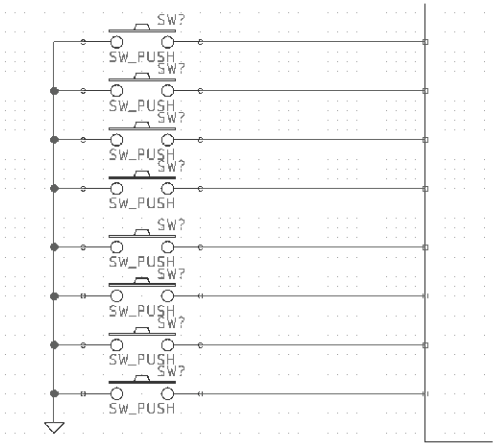
+ Łatwość programowania
+ Gotowe biblioteki
+ Łatwe podłączenie
+ Dodatkowe interfejsy




- Bardzo wysokie ceny
- Wewnątrz jest któreś z omawianych rozwiązań.

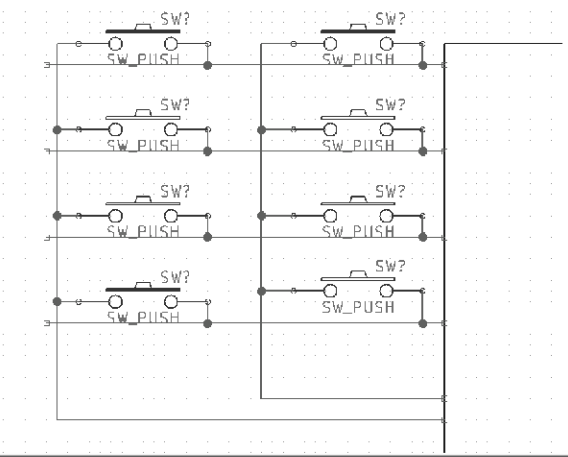


## Jak użyć mniej pinów? - Klawiatura

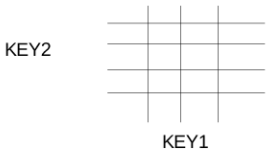


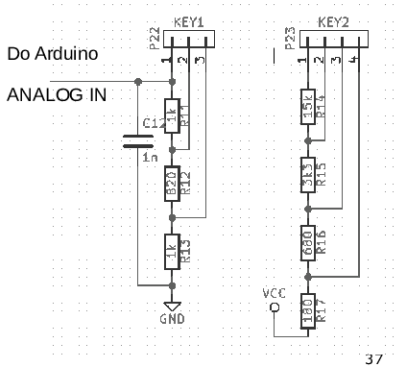



## Klawiatura: Lepsze rozwiązanie



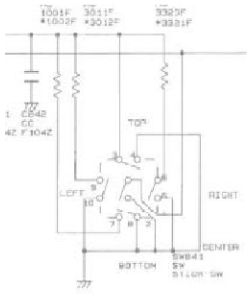
## Klawiatura: Pin analogowy






## Klawiatura: Ekstremalne rozwiązanie (RC)

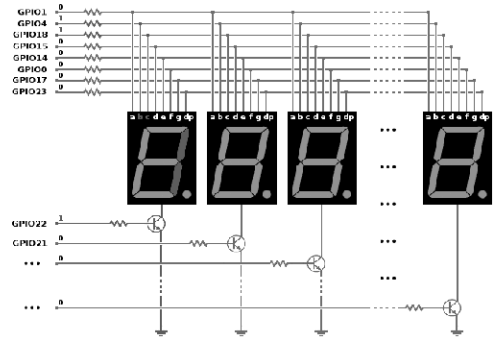



*(Eizo F980 schematic)*



## Wyjścia: Multipleksowanie wyjść

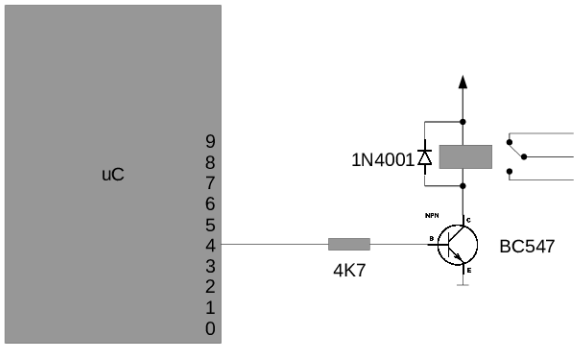
- Zamiast 4x8=32 wyjść użyte 8+4=12.
- Możliwość dalszego zmniejszania wyjść:
  - np. wejścia wyświetlaczy (4) – zapis na 2 bitach, użycie dekodera.




*http://hackyourmind.org*

## Urządzenia wyjścia

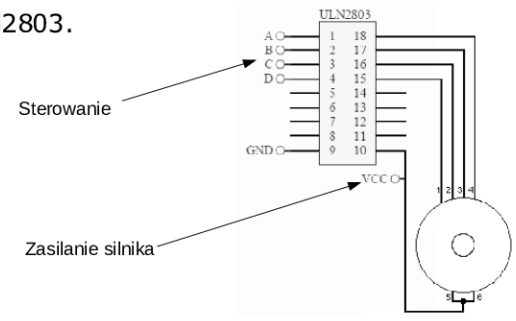
- Przekaźnik, odbiorniki do ok. mocy tranzystora:






## Urządzenia wyjścia

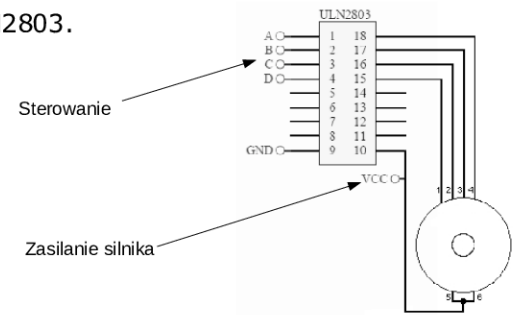
- Układy Darlingtona: Sterowanie silnikiem krokowym:
  - np. ULN2803.






## Urządzenia wyjścia

- Układy Darlingtona: Sterowanie silnikiem krokowym:
  - np. ULN2803.



## Silniki krokowe

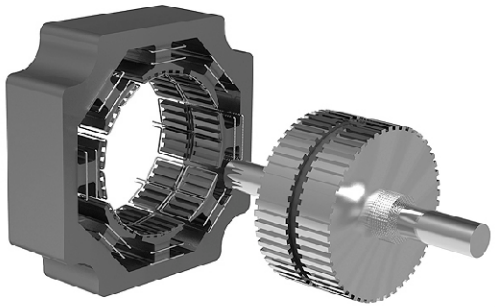
- W przeciwieństwie do liniowych, możliwe jest przestawienie o ustalony kąt,
- Moment jest (w przedziale roboczym) odwrotnie proporcjonalny do prędkości,
- Wymagają znacznych prądów (więc i sterowników),
- Łatwa dostępność z odzysku – drukarki, napędy CD/FDD, skanery,
- Biblioteki do ich obsługi są w Arduino, a sterowniki są proste w budowie.
- Poruszane są przez doprowadzenie prądu do odpowiednich uzwojeń w prawidłowej kolejności, silniki takie mają 2..6 uzwojeń.



## Jak to działa?

- Wał jest namagnesowany w odpowiedni sposób,
- Uaktywniane są kolejne elektromagnesy,
  Magnes na wale jest przyciągany przez jeden elektromagnes, a odpychany przez inny co powoduje obrót o jeden krok.

- Możliwa praca wyłącznie „na przyciąganie” - mniejszy moment, łatwiejsze sterowanie.

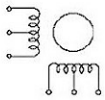


*Źródło: Wikimedia comons*

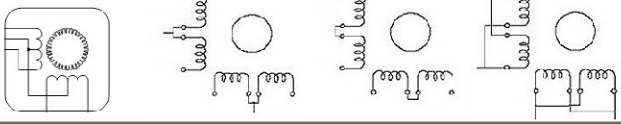
## Rodzaje silników krokowych

- Bipolarne
  - 4 wyprowadzenia

  - 6 wyprowadzeń

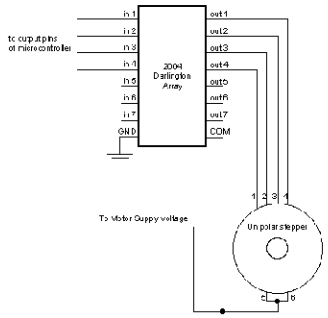


- Unipolarne
  - 4, 8, 5 wyprowadzeń



## Sterownik silnika

- Silniki unipolarne:

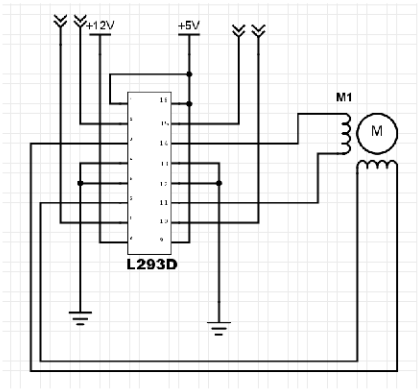



*Źródło: Dokumentacja Arduino https://www.arduino.cc/en/Reference/Stepper*

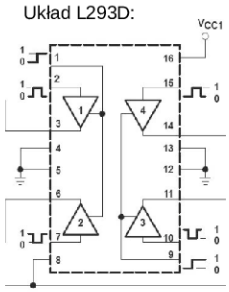


## Sterownik silnika

- Silniki bipolarne:




Układ L293D:






## Sekwencja sterowania

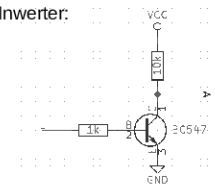
Dla 4 pinów:

```
* Step C0 C1 C2 C3
*   1   1  0  1  0
*   2   0  1  1  0
*   3   0  1  0  1
*   4   1  0  0  1
```

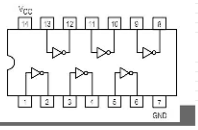
Zauważmy, że zawsze:
- $C_0 != C_1$
- $C_2 != C_3$

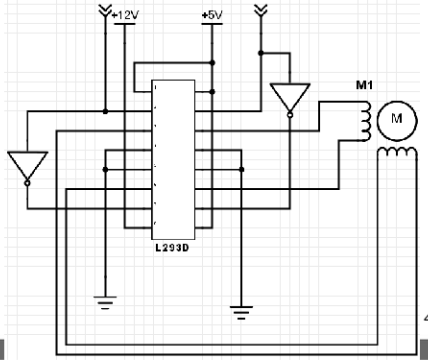
→ Możemy więc użyć tylko 2 pinów!

Inwerter:




...lub układ 74LS04 (6x inwerter)



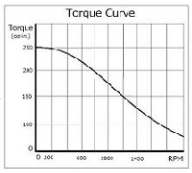
## Biblioteka stepper

```
#include <Stepper.h>

Stepper myStepper(200, 8, 9, 10, 11);

void setup() {
   myStepper.setSpeed(60);
}

void loop() {
  myStepper.step(1);
  delay(20);
}
```

Ilość kroków/obrót

Piny, do których podłączono sterownik (2, 4 lub 5 pinów)

Prędkość (~obr/min)



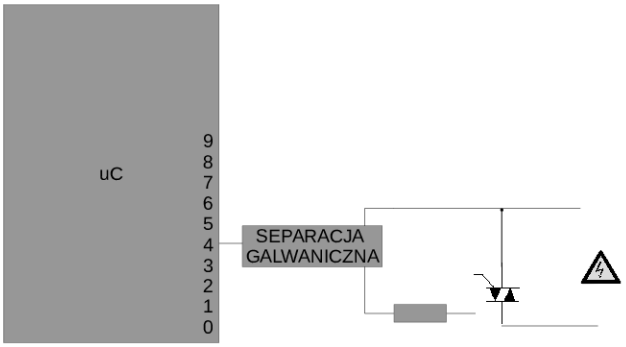

Źródło: CNC Router Source

Ilość kroków (może być ujemna)



## Urządzenia wyjścia

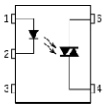
- Tyrystor/triak – sterowanie prądem zmiennym:



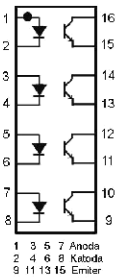



## Separacja galwaniczna:

- Transoptor (DC), Optotriak (AC)
- Podłączenie:
  - Jak LED (separacja wyjścia)
  - Jak łącznik (separacja wejścia)



- Używane do zabezpieczenia przed:
  - Wysokim napięciem
  - Uszkodzeniem portu
  - Pętlą masy




- Pamiętamy o ograniczeniu prądu!



## Zasilanie układów



## Zasilacz

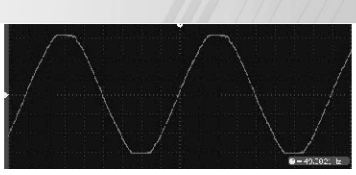
...to urządzenie przekształcające prąd o jednych parametrach na prąd o parametrach jakich potrzebuje nasze urządzenie.

- Układ zasilacza może czerpać energię z np:
  - Instalacji elektrycznej
  - Baterii ogniw lub akumulatorów
  - Innego zasilacza – np.. urządzenia na USB
  - Generatorów, ogniw, innych źródeł



## Zasilacze sieciowe

- Napięcie sieci: **230V/50Hz**
  - "Sinusoidalne"



- W zasilaczu sieciowym występują **niebezpieczne napięcia**!
  - Nie należy modyfikować konstrukcji zasilacza sieciowego bez wiedzy jak on działa
  - Przed jakimkolwiek serwisowaniem zasilacza sieciowego:
    - Odłączyć fizycznie zasilanie,
    - Zabezpieczyć przed przypadkowym włączeniem zasilania,
    - Rozładować kondensatory! - **niebezpieczeństwo porażenia przy odłączonym zasilaczu**!

## Zasilacze

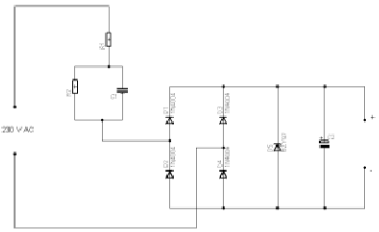
- Do projektowanego urządzenia musimy dostarczyć prąd:
  - O odpowiednich **napięciach**
    - np. 5V dla AVR, 12V dla LED mocy
  - Odpowiednim maksymalnym **natężeniu**
    - np. 200mA dla AVR, 1A dla LED mocy
  - Przy czym przynajmniej dla układów cyfrowych muszą być to napięcia **stabilizowane**.



## Zasilacz beztransformatorowy

- Historycznie pierwszy, prosty zasilacz do zasilania z sieci energetycznej
- **Potencjał „fazy" (>230V!) na jednym wyjściu!**
- Mała masa, niskie bezpieczeństwo

- Praktycznie brak stabilizacji
- Nieodporny na zakłócenia
- **NIEBEZPIECZNY.**

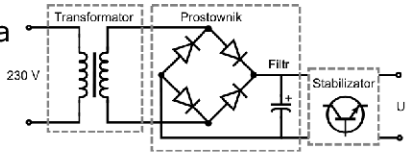




## Zasilacz liniowy

- Bezpieczniejszy – separacja galwaniczna przez transformator.
- Transformator zamienia 230V na np. 6V AC
- Prostownik+filtr – 6V*1.41=8,46V
- Stabilizator – 8.46V (zmienne w zależności od obciążenia) → 5V DC.

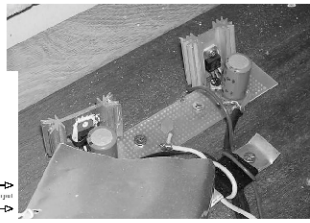
- Wydziela się dużo ciepła
  niech I=2A:
  8,46V-5V=3.46V
  3.46V*2A=6.92W w ciepło!

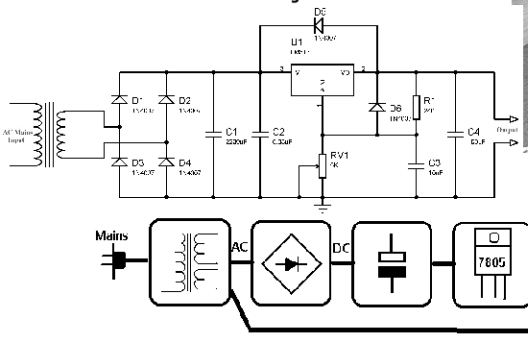





## Konstrukcje zasilaczy liniowych

- Przy użyciu gotowego układu serii 78xx
  - (max 1A)
  - Łatwa konstrukcja



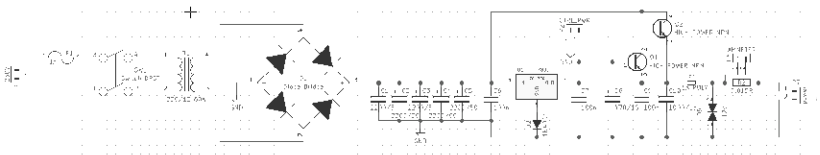






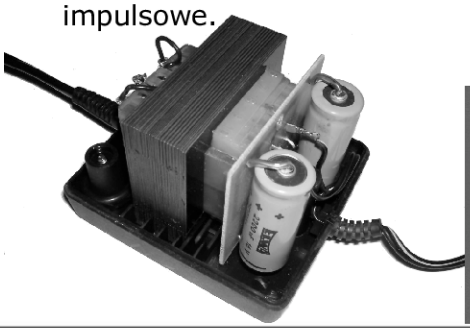

## Konstrukcje zasilaczy liniowych

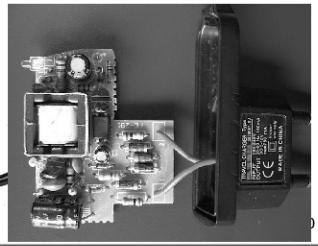
- Wersja z tranzystorem
  - Źródło napięcia odniesienia – tu układ 7809, może by„ stosowana np. dioda Zenera
  - Niespecjalna stabilizacja (spadek napięcia na złączach tranzystorów występuje i tak).
  - Tranzystory powinny mieć ( ;-) ) rezystory emiterowe.



## Zasilacze liniowe vs impulsowe

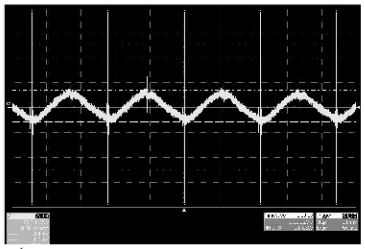
- Zasilacze liniowe są proste w konstrukcji i bezpieczne tak jak bezpieczny jest transformator. Jednak ze względu na lepszą sprawność i mniejsze wymiary coraz popularniejsze stają się przetwornice impulsowe.

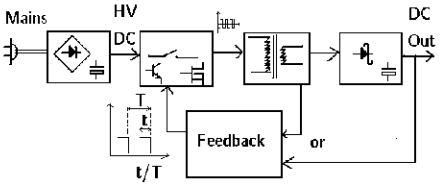




## Zasilacze impulsowe

- Mniejsze wymiary, większa sprawność
- ...ale bardziej skomplikowana budowa
- Niektóre nie zastartują bez obciążenia
- Tańsze zasilacze wprowadzają zakłócenia,
- Możliwe konstrukcje DC-DC



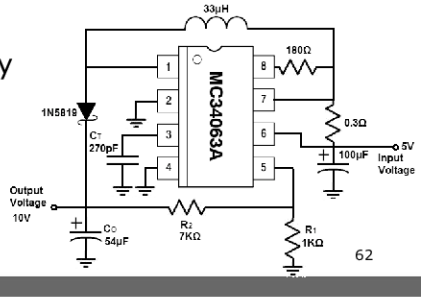



Źródło: E2E TI Community



## Na przykład MC34063

- Układy ciągle produkowane, choć nie najnowsza technologia
- Przetwornica 5V → 10V
- Możliwe zastosowanie tranzystora dla zwiększenia mocy
- Istotny dobór cewki!
- Gorsza dioda – gwałtowny spadek sprawności.




## Komunikacja i protokoły



## Komunikacja

- ...z innymi urządzeniami – by umożliwić wymianę informacji.
- ...z istniejącym systemem sieciowym – by usprawnić przetwarzanie danych
- ...z komputerem – by używać urządzenia.

- Musimy opracować:
  - Interfejs
  - Protokół komunikacyjny



## Protokoły komunikacyjne

- Niezbędne jest użycie istniejącego lub utworzenie nowego języka komunikacji urządzeń – protokołu
- Jeżeli urządzenie ma udawać inne, np.. klawiatura – dobrze udokumentowane protokoły już istnieją
- ...ale jeżeli tworzymy całkowicie nowe urządzenie, istniejące standardy, choć mocno naginane, mogą okazać się niewystarczające.



## Interfejs komunikacyjny

np.

- Port szeregowy oferowany przez Arduino
- USB i emulacja USB przez AVR
- Sieć – przy użyciu układu NIC
- Wi-fi (np.. układ ESP8266)
- Magistrale szeregowe oferowane przez AVR
- Własne interfejsy szeregowe i równoległe

## Opracowywanie protokołu

- Ściśle zależne od tego, jaką funkcję ma wykonywać urządzenie. Inny protokół będzie opracowany dla kontrolerów sterowanych przez komputer, a inny dla sterowników IoT.
- O ile to możliwe należy użyć istniejących interfejsów (np.. USB, Bluetooth, RS232), a protokoły powinny być udokumentowane!



## RS232 – ASCII czy binarny?

- ASCII – łatwy w debugowaniu, możliwe ręczne testy, ale małe możliwości transmisji.
- Binarny – możliwość przesłania dużej ilości danych, większa szybkość (brak złożonego dekodowania), ale trudniejszy w weryfikacji i przeprowadzaniu testów.

- Jeżeli wykonujemy proste zadania stosunkowo rzadko (np. sterowanie przekaźnikami, sprawdzanie czujnika) możemy użyć protokołu w ASCII.



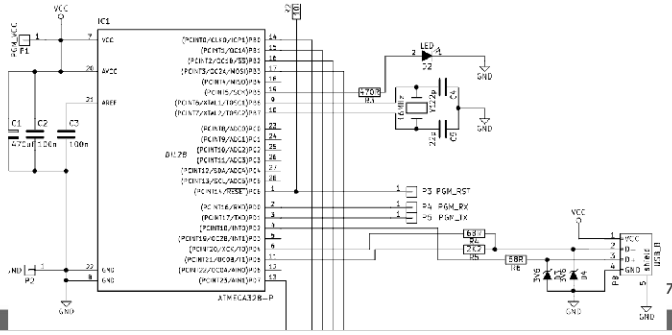
## Co powinno być w protokole? Minimum

- „Identyfikacja” - komunikat, który powoduje odesłanie przez urządzenia informacji o sobie. Umożliwia wybranie urządzenia i uzyskanie informacji o jego wersji, możliwościach, funkcji.
- „Reset” - komunikat, który powoduje natychmiastowe przywrócenie wejść/wyjść do stanu pierwotnego, wywołującego najmniej możliwych negatywnych konsekwencji. Wykonywane na starcie urządzenia oraz w przypadku problemów z programem klienckim.



## Przykład

- Sterownik klawiatury akordowej:
  - Interfejs: USB, emulacja klawiatury USB,
  - Protokół – klawiatury USB,
  - Użycie protokołu: Niestandardowe – naciśnięcie przycisku wywołuje emulację wciśnięcia jednej kombinacji klawiszy, zwolnienie – innego.






## Przykład – c.d.

- Użycie istniejącego protokołu całkowicie „załatwia” nam kwestię identyfikacji (urządzenie zgłosi się jako klawiatura USB) i resetu.
- Istnieje gotowa biblioteka do low-speed USB: V-USB.
- Wystarczy oprogramować naciskanie i zwalnianie klawiszy w kodzie dla AVR.

```
#include "UsbKeyboard.h"

void setup() {
  TIMSK0 &= !(1<TOIE0); //atmega328
  cli(); // Clear interrupts

  UsbDeviceDisconnect(); //make pc re-discover
  delayMs(250);
  usbDeviceConnect();

  sei(); // Set interrupts again
  pinMode(KEY_1, INPUT_PULLUP);
}

void loop() {
  UsbKeyboard.update();

(. . .)

if (digitalRead(KEY_1) == LOW)  {
 UsbKeyboard.sendKeyStroke(CODE1_DOWN, MODIFIER);
 delay(100);
}

(. . .)
```



## Komunikacja z własną aplikacją

- Przykład 2: Sterowanie płytką przekaźników przez komputer.
- Arduino będzie spełniać role interfejsu.

- Więc:
  - Interfejs: RS232 (via USB).
  - Protokół: ASCII.

- Polecenia:
  - Identyfikacja
  - Reset
  - Ustaw
  - Odczytaj stan

## Przykład 2: c.d.

- Np. tak:

| PC->AVR | Funkcja | Użycie / Odpowiedź |
|---|---|---|
| I | IDENTIFY | Łańcuch tekstu zawierający „INTERFACE” |
| R | RESET | Wszystkie przekaźniki wyłączone.<br>AVR odpowiada „O” jak „OK”. |
| Sxx.0, Sxx.1 | SET | Ustawia przekaźnik xx w stan 0 lub 1.<br>AVR odpowiada „O” jak „OK”. |
| Gxx | GET | Pobiera informację o stanie przekaźnika.<br>AVR odpowiada „0” lub „1”. |



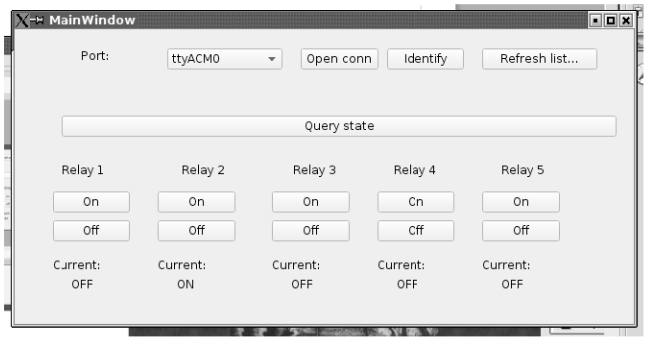
## Kod Arduino

```
#define ...

void setup() {
  Serial.begin(9600);
  pinMode(RELAY1,OUTPUT);
. . .
  pinMode(RELAY5,OUTPUT);
  reset();
}

void reset() {
 digitalWrite(RELAY1,LOW);
. . .
 digitalWrite(RELAY5,LOW);
}

byte getState(byte relay)
{
  if (relay==1)
    return digitalRead(RELAY1);
 . . .
  if (relay==5)
    return digitalRead(RELAY5);
return 255; //error condition
}

void setState(byte relay, byte state) {
    if (relay==1)
    digitalWrite(RELAY1,state);
 . . .
  if (relay==5)
    digitalWrite(RELAY5,state);
}
```

```
void loop() {
  if (Serial.available()) {
    char k=Serial.read();
    delay(10);
    switch (k) {
      case 'I': {
        Serial.print("RELAY INTERFACE v. 1.0");
        break;
      }
      case 'R': {
        reset();
        Serial.print("O");
        break;
      }
      case 'G': {
        byte which=Serial.read()-'0';
        Serial.print(getState(which));
        break;
       }
      case 'S': {
        byte which=Serial.read()-'0';
        byte what=Serial.read(); //dot
        what=Serial.read()-'0';
        setState(which,what);
        Serial.print("O");
        break;
      }
    }
  }
}
```



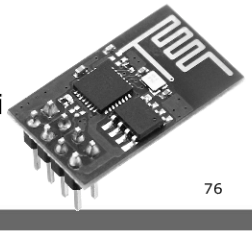
## Aplikacja






## Łączność – modemy szeregowe

- Dla Arduino istnieje wiele urządzeń dołączanych do portu szeregowego. Urządzenia te zachowują się jak modemy.
- Obsługa np.
  - Bluetooth
  - Wi-fi (ESP8266)
  - GSM (SIM800)
  - GPS (SIM900 i późne SIM800)
  - I wiele innych
- Sterowanie odbywa się poleceniami modemowymi (Hayesa) i ich rozszerzeniami.





## Dziękuję za uwagę