Podstawy Systemów Wbudowanych

Wykład 2: Język programowania Arduino

Angelika Tefelska Dariusz Tefelski

Zakład Fizyki Jądrowej, Wydział Fizyki PW

15 marca 2021

O czym będzie wykład...

1. Przegląd wbudowanych funkcji
2. Wyświetlacz 7seg
3. Wyświetlacz LCD
4. Magistrala komunikacyjna
5. I2C
6. 1-Wire
7. SPI

Przegląd wbudowanych funkcji

- Funkcja do mierzenia czasu między dwoma impulsami tego samego typu (HIGH/LOW):

  pulseIn();

- Funkcja, która zwraca ile milisekund upłynęło od uruchomienia szkicu:

  millis();

- Funkcja, która zwraca ile mikrosekund upłynęło od uruchomienia szkicu:

  micros();

- Funkcja, która wstrzymuje wykonywanie programu przez zadany czas w mikrosekundach:

  delayMicroseconds();

- Funkcja, która wstrzymuje wykonywanie programu przez zadany czas w milisekundach:

  delay();

Przegląd wbudowanych funkcji matematycznych

- Funkcja, która zwraca wartość mniejszą z dwóch podanych:

  min(wartość 1, wartość 2);

- Funkcja, która zwraca wartość większą z dwóch podanych:

  max(wartość 1, wartość 2);

- Funkcja, która zwraca wartość bezwzględną:

  abs(wartość);

- Funkcja, która przeskalowuje nam daną wartość w nowym zakresie:

  map(wartość, dolny limit, górny limit, nowy dolny limit, nowy górny limit);

- Funkcja, która zwraca nam liczbę podniesioną do żądanej potęgi:

  pow(liczba typu float, potęga typu float);

- Funkcja, która zwraca pierwiastek danej liczby:

  sqrt(liczba);

- Funkcje, które zwracają wartość wybranej funkcji trygonometrycznej:

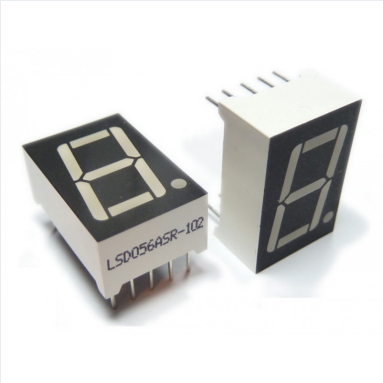
  sin(kąt w radianach);
  cos(kąt w radianach);
  tan(kąt w radianach);

- Funkcja, która zwraca pseudo-losową wartość z wybranego przedziału:

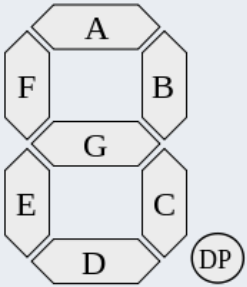
  random(min,max);

Wyświetlacz siedmiosegmentowy

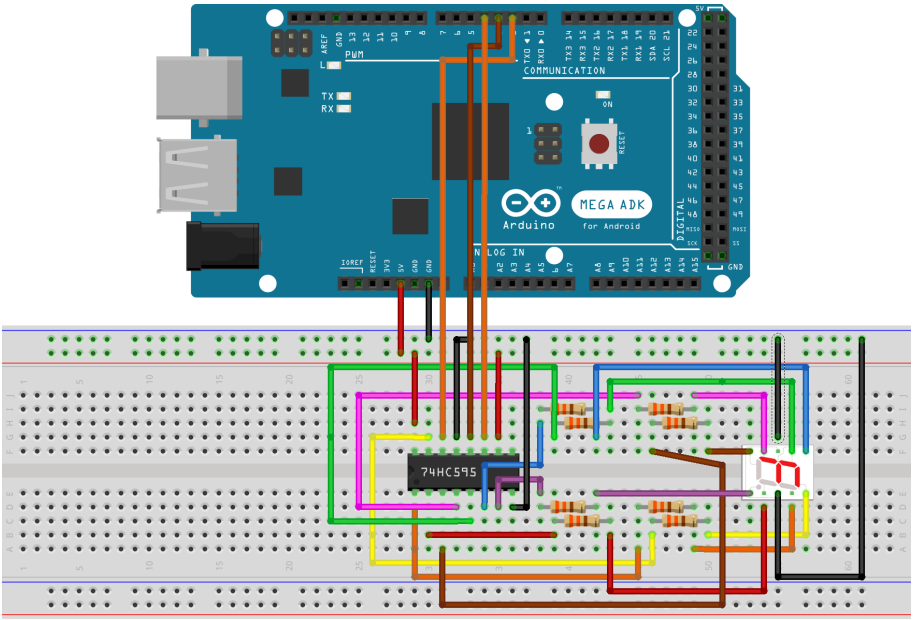
- Wyświetlacze siedmiosegmentowe są powszechnie stosowane w urządzeniach, w których wyświetlane są tylko liczby np. cyfrowe budziki, szybkościomierze. Wyświetlacze mogą być ze wspólną anodą lub katodą.



Rysunek: Wyświetlacz 7segmentowy. Źródło: http://extronic.pl



Rysunek: Schemat wyświetlacza 7segmentowego. Źródło: https://pl.wikipedia.org


PWM
TX
RX
COMMUNICATION
ON
RESET
MEGA ADK
Arduino
for Android
DIGITAL
ANALOG IN
74HC595
fritzing

Wyświetlacz 7seg - podłączenie

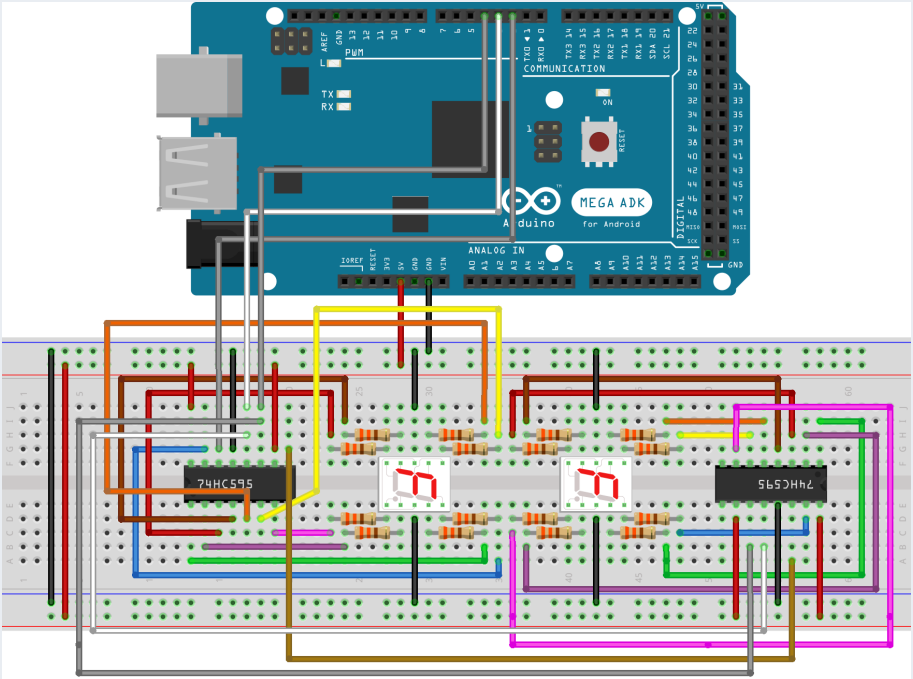
Rejestr przesuwający	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7	
Segment	A	B	C	D	E	F	G	DP	Liczba dziesiętna
0	1	1	1	1	1	1	0	0	252
1	0	1	1	0	0	0	0	0	96
2	1	1	0	1	1	0	1	0	218
3	1	1	1	1	0	0	1	0	242
4	0	1	1	0	0	1	1	0	102
5	1	0	1	1	0	1	1	0	182
6	1	0	1	1	1	1	1	0	190
7	1	1	1	0	0	0	0	0	224
8	1	1	1	1	1	1	1	0	254
9	1	1	1	1	0	1	1	0	246
A	1	1	1	0	1	1	1	0	238
B	0	0	1	1	1	1	1	0	62
C	1	0	0	1	1	1	0	0	156
D	0	1	1	1	1	0	1	0	122
E	1	0	0	1	1	1	1	0	158
F	1	0	0	0	1	1	1	0	142

Wyświetlacz 7seg - przykład 1

Przykład: Wyświetlenie liczby 3 na wyświetlaczu 7seg z wspólną katodą.

```
#define DATA 2
#define LATCH 3
#define CLOCK 4

int cyfra3=242;

void setup()
{
    pinMode(DATA, OUTPUT);
    pinMode(LATCH, OUTPUT);
    pinMode(CLOCK, OUTPUT);
}
void loop()
{
    digitalWrite(LATCH,LOW);
    shiftOut(DATA, CLOCK, LSBFIRST, cyfra3); //Dla wspolnej
       ↪ anody: ~cyfra3
    delay(100);
    digitalWrite(LATCH,HIGH);
}
```

Wyświetlacz 7seg - rozbudowa o dodatkowy wyświetlacz



Wyświetlacz 7seg - przykład 2

Przykład: Wyświetlenie cyfry 23 na wyświetlaczu 7seg z wspólną katodą.

```
#define DATA 2
#define LATCH 3
#define CLOCK 4

int digits[]={252, 96, 218, 242, 102, 182, 190, 224, 254,
    ↪ 246, 238, 62, 156, 122, 158, 142};

void setup()
{
    pinMode(LATCH, OUTPUT);
    pinMode(CLOCK, OUTPUT);
    pinMode(DATA, OUTPUT);
}
```

Wyświetlacz 7seg - przykład 2 ciąg dalszy

Przykład: Wyświetlenie cyfry 23 na wyświetlaczu 7seg z wspólną katodą.

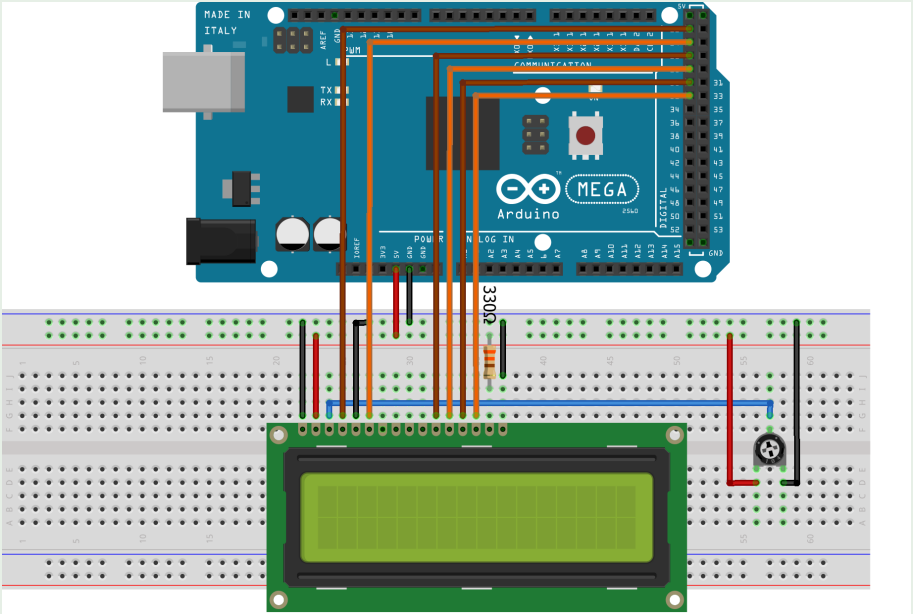
```
void display(int number)
{
    int left=0, right=0;
    if(number<10)
    {
         right=number;
         left=0;
    }
    if(number>=10)
    {
        right=n %10; //reszta z dzielenia
        left=n/10;
    }
    digitalWrite(LATCH,LOW);
    shiftOut(DATA,CLOCK, LSBFIRST, digits[right]);
    shiftOut(DATA,CLOCK,LSBFIRST,digits[left]);
    digitalWrite(LATCH,HIGH);
}
void loop()
{
    display(23);
}
```

Wyświetlacz LCD

- Moduły LCD wyświetlające znaki są jednym z najpopularniejszych i najtańszych produktów wykonanych w technologii ciekłokrystalicznej. Do wyboru jest wiele wielkości wyświetlaczy a ich wielkość opisywana jest przez liczbę wierszy i kolumn.
- Z arduino będą współpracować wszystkie wyświetlacze LCD z interfejsem zgodnym ze standardem: HD44780 i KS0066.

- Rodzaje trybów sterowania wyświetlaczem LCD:
  - 8-bitowy bez odczytu flagi zajętości (czyli do transmisji potrzeba 8 linii magistrali danych i dwie linie sterujące (RS,E)).
  - 8-bitowy z odczytem flagi zajętości (czyli do transmisji potrzeba 8 linii magistrali danych i trzy linie sterujące (RS,E,RW))
  - 4-bitowy z odczytem flagi zajętości (czyli do transmisji potrzeba 4 linie magistrali danych i trzy linie sterujące (RS,E,RW))
  - **4-bitowy bez odczytu flagi zajętości (czyli do transmisji potrzeba 4 linie magistrali danych i dwie linie sterujące (RS,E))**

Wyświetlacz LCD - podłączenie w trybie 4-bitowym bez odczytu flagi zajętości



Wyświetlacz LCD - biblioteka LiquidCrystal

- Do obsługi wyświetlacza gotowa jest biblioteka LiquidCrystal, której obiekt tworzy się w następujący sposób:
  - 8-bitowy bez odczytu flagi zajętości, jako argumenty konstruktora podaje się numery pinów podłączone odpowiednio do RS,E,D0-D7 na wyświetlaczu LCD:

    LiquidCrystal lcd(RS, E, d0, d1, d2, d3, d4, d5, d6, d7);
  - 8-bitowy z odczytem flagi zajętości:

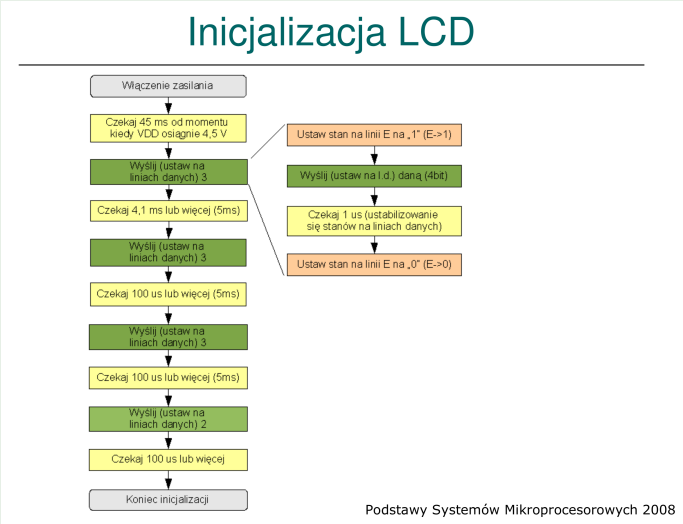
    LiquidCrystal lcd(RS, RW, E, d0, d1, d2, d3, d4, d5, d6, d7);
  - 4-bitowy z odczytem flagi zajętości:

    LiquidCrystal lcd(RS, RW, E, d4, d5, d6, d7);
  - **4-bitowy bez odczytu flagi zajętości:**

    LiquidCrystal lcd(RS, E, d4, d5, d6, d7);

Wyświetlacz LCD - biblioteka LiquidCrystal

- Schemat inicjalizacji wyświetlacza LCD:

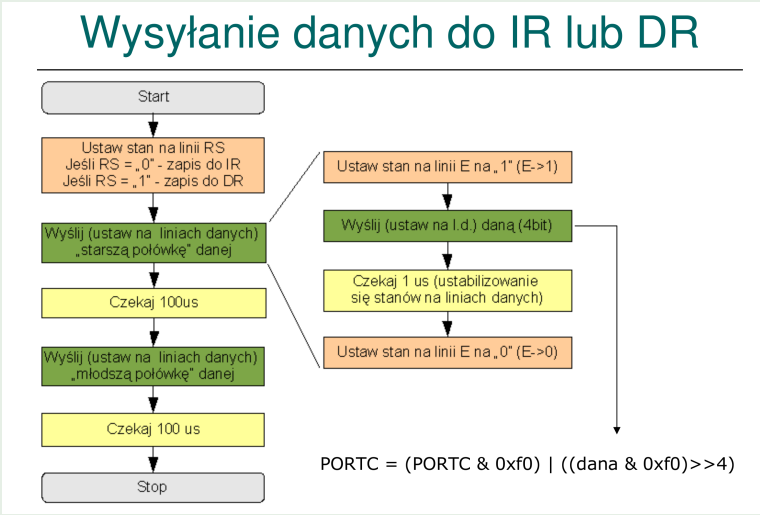


- a w praktyce:

lcd.begin(liczba kolumn,liczba wierszy);

Wyświetlacz LCD - biblioteka LiquidCrystal

- Wysyłanie instrukcji lub danych do wyświetlacza:



- a w praktyce, patrz następny slajd...

Wyświetlacz LCD - biblioteka LiquidCrystal

- Funkcja, która czyści bufor danych i ustawia kursor w lewym górnych rogu wyświetlacza LCD:

  lcd.clear();

- Funkcja, która ustawia kursor na wskazanej pozycji:

  lcd.setCursor(numer kolumny,numer wiersza);

- Funkcja, która wyświetla dane typu: char, byte, int, long, lub string:

  lcd.print(dana);

- Wyświetlenie liczby w postaci binarnej lub szesnastkowej:

  lcd.print(dana, BIN/HEX);

- Funkcja, która wyświetla dane na wyświetlaczu LCD (równi się tym od print, że przyjmuje więcej typów danych np. double):

  lcd.write(dana);

- Więcej funkcji jest opisanych na stronie: www.arduino.cc

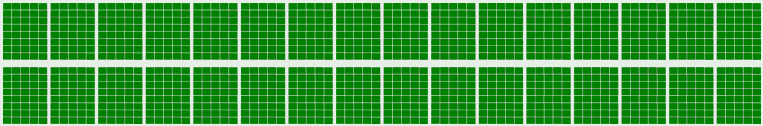
Wyświetlacz LCD - przykład

Przykład: Wyświetlenie tekstu na wyświetlaczu LCD

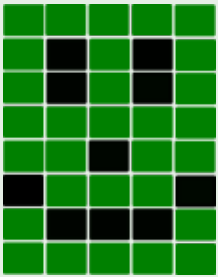
```
#include <LiquidCrystal.h>
#define RS 22
#define E 24
#define D4 26
#define D5 28
#define D6 30
#define D7 32

LiquidCrystal lcd(RS, E, D4, D5, D6, D7);

void setup()
{
    lcd.begin(16,2);
    lcd.clear();
}

void loop()
{
    lcd.setCursor(0,0);
    lcd.print("OK");
    delay(200);
}
```

Wyświetlacz LCD - definiowanie znaków niestandardowych

- Każdy znak wyświetlany na module LCD składa się z 8 wierszy po 5 kolumn pikseli.



- W celu wyświetlenia własnych znaków trzeba określić, które piksele mają być podświetlone, np:



```
byte znak[]={B00000,
             B01010,
             B01010,
             B00000,
             B00100,
             B10001,
             B01110,
             B00000};
```

- Uwaga: Wyświetlacz pozwala maksymalnie zdefiniować 8 niestandardowych znaków.

Wyświetlacz LCD - definiowanie niestandardowych znaków - przykład

Przykład: Wyświetlenie znaczka buźki na wyświetlaczu LCD.

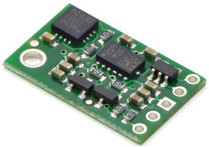
```
#include <LiquidCrystal.h>
#define RS 22
#define E 24
#define D4 26
#define D5 28
#define D6 30
#define D7 32

LiquidCrystal lcd(RS, E, D4, D5, D6, D7);
byte znak[]={B00000, B01010, B01010, B00000,  B00100, B10001,
    ↪ B01110, B00000};

void setup()
{
    lcd.createChar(0,znak); //Ustawienie wygladu 0-wego znaku
    lcd.begin(16,2);
}
void loop()
{
    lcd.write(byte(0)); //Wyswietlenie znaku niestandardowego
                        //zadeklarowanego na pozycji nr 0
}
```

Magistrala komunikacyjna

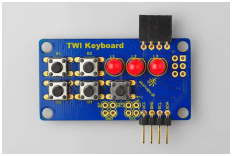
- Magistrala komunikacyjna - zespół linii przenoszących sygnały oraz układów wejścia-wyjścia służących do przesyłania sygnałów między połączonymi urządzeniami w systemach mikroprocesorowych.
- Większość magistrali składa się z 3 szyn:
  1. sterującej - która określa rodzaj operacji jaki ma być wykonany np. zapis, odczyt.
  2. adresowej - określa z jakie/do jakiej komórki sygnał ma zostać odczytany/zapisany.
  3. danych - szyna służąca do przesyłu właściwych danych.
- Ze względu na typ transmisji magistrale można podzielić na:
  1. równoległe - np. PCI, AGP, FSB.
  2. Szeregowe - np. USB, RS-232, I2C, SPI.
- Ze względu na sposób transmisji można wyróżnić magistrale:
  1. jednokierunkowe - dane są przesyłane tylko w jednym kierunku
  2. dwukierunkowe - dane mogą być przesyłane w obu kierunkach:
     - full duplex - dane mogą być przesyłane w obu kierunkach jednocześnie
     - half duplex - dane w określonym momencie mogą być przesyłane tylko w jednym kierunku [a]

[a] Źródło: https://pl.wikipedia.org

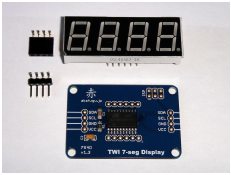
Dlaczego magistrala I2C?



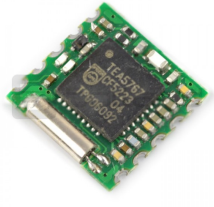
Akcelerometr, żyroskop i magnetometr



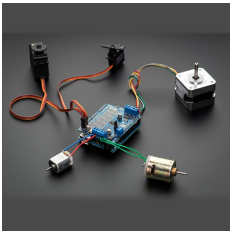
Klawiatura



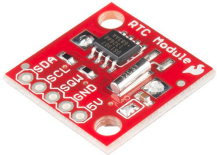
Wyświetlacz 7seg



Moduł tunera radiowego stereo FM



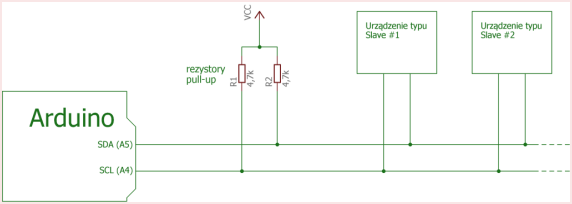
Sterownik silników DC, krokowych i serw



Moduł RTC

Magistrala I2C

- I2C może obsługiwać do 127 urządzeń przy 7-bitowym adresowaniu typu slave (urządzenia podrzędne - wysyłające dane np. jakiś czujnik). Do magistrali może być podłączone więcej niż jedno urządzeń typu master (nadrzędne - odbierające dane np. arduino).

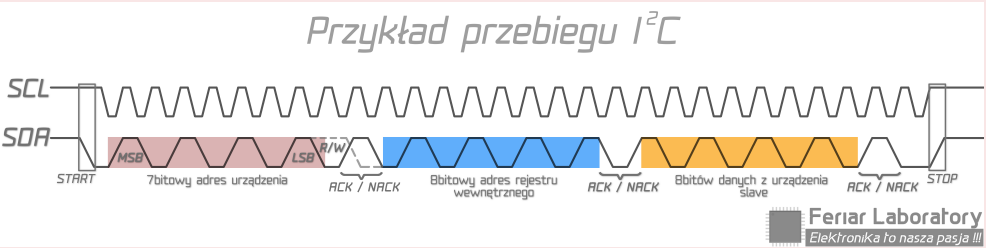


Rysunek: Schemat podłączenia urządzeń typu slave do arduino. [a]

- I2C do transmisji wykorzystuje dwie dwukierunkowe linie: SDA (Serial Data Line - linia danych) i SCL (Serial Clock Line - linia zegara). Obydwie linie są na stałe podciągnięte do źródła zasilania poprzez rezystory podciągające.
- Oporność rezystorów podciągających wpływa na maksymalną osiągalną prędkość transmisji danych. Zakres, w obrębie którego powinna być wybrana wartość rezystora wynosi: (1.8, 47) $k\Omega$. Jednak im mniejsza wartość rezystora tym możliwa szybsza transmisja danych (bardziej strome zbocza sygnału) (np. dla $2k\Omega$ wynosi ok 400kb/s) ale również większy pobór energii. Z tego powodu dobrym kompromisem jest 4.7 $k\Omega$.

Magistrala I2C

Zasada transmisji danych:



1. Wysłanie bitu startu czyli linia SDA przechodzi ze stanu wysokiego na niski gdy na linii SCL jest stan wysoki.
2. Wysłanie 7 bitów z adresem urządzenia oraz jednego bitu R/W informującego czy urządzenie typu master chce dokonać odczytu czy zapisu (odczyt=1, zapis=0).
3. Urządzenie typu slave zwraca bit ACK (Acknowledge). Jeśli adres został poprawnie wysłany i rozpoznany przez urządzenie typu slave, to poda ono stan niski na linię SDA. W przeciwnym wypadku procedura zostanie przerwania (stan wysoki - NACK).
4. Wysłanie adresu wewnętrznego adresu rejestru, który ma zwracać daną wartość z układu np. gdy chcemy odczytać godzinę to dla zegara RTC będzie to adres rejestru odpowiedzialny za godzinę.
5. Wysłanie bitu ACK jeśli adres został rozpoznany.
6. Wysłanie 8bitów danych.
7. Kończenie procesu wysyłania danych poprzez ustawienie bitu ACK i wysłanie bitu STOP.

Magistrala I2C - biblioteka Wire

- W celu korzystania z magistrali I2C niezbędne jest dołączenie biblioteki:

  #include <Wire.h>

- Aktywowanie magistrali I2C odbywa się za pomocą funkcji:

  Wire.begin();

- Inicjowanie komunikacji odbywa się za pomocą poniższej funkcji, do której należy przekazać adres urządzenia typu slave w formie szesnastkowej:

  Wire.beginTransmission(adress);

- Wysyłanie 1 bajta danych z Arduino do urządzenia typu slave zaadresowanego wcześniej:

  Wire.write(dane);

- Do zakończenia transmisji służy funkcja:

  Wire.endTransmission();

  UWAGA: Funkcja endTransmission() zwraca:

  - 0 - gdy transmisja zakończyła się sukcesem
  - 1 - odebrane dane są za duże aby zmieścić je w buforze
  - 2 - urządzenie zwróciło NACK przy wysyłanym adresie
  - 3 - urządzenie zwróciło NACK przy wysyłaniu danych
  - 4 - nieznany błąd

Magistrala I2C - biblioteka Wire

- W celu zażądania danych wysłanych przez urządzenie typu slave należy skorzystać z funkcji:

Wire.requestFrom(adres, liczba żądanych bajtów);

- Funkcja, która zwraca liczbę bajtów możliwych już do odczytu:

Wire.available();

- Odczytanie bajta danych jest możliwe przy użyciu funkcji:

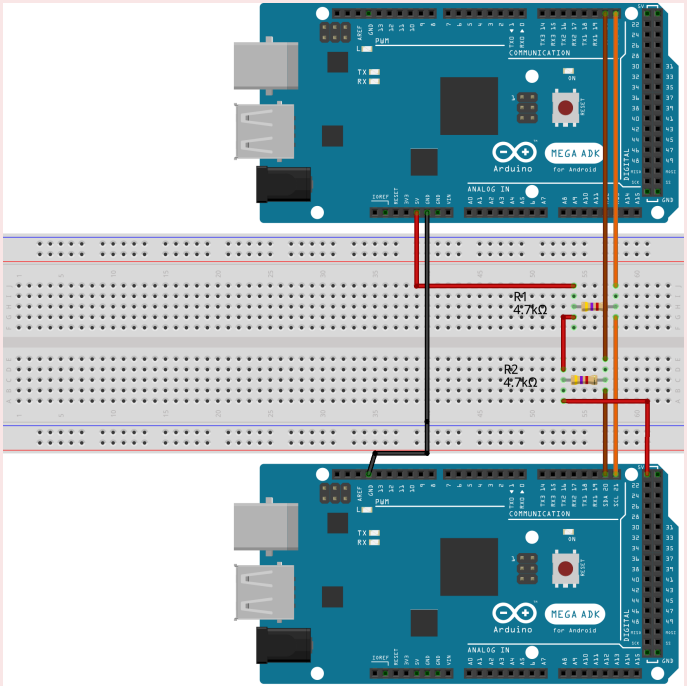
char dana=Wire.read();

Magistrala I2C - przykład

Przykład: Odbiór danych przez arduino z magistrali I2C i wyświetlenie w monitorze portu szeregowego.

```
#include <Wire.h>
void setup()
{
    Wire.begin();
    Serial.begin(9600);
}
void loop()
{
    Wire.beginTransmission(0x3f); //Adres urzadzenia slave
    Wire.write(0x2f); //Adres rejestru urzadzenia slave
    if(Wire.endTransmission()==0)
    {
        Wire.requestFrom(0x3f,2); //Wymuszamy zwrocenie 2 bajtow
        while(Wire.available()) //Petla po bajtach danych
        {
            char a=Wire.read(); //Odbior 1 bajta danych
            Serial.println(a); //Wyswietlenie 1bajta danych w
                ↪ monitorze portu szeregowego
        }
    }
}
```

Przesyłanie danych korzystając z magistrali I2C pomiędzy dwoma płytkami arduino



Przesyłanie danych korzystając z magistrali I2C pomiędzy dwoma płytkami arduino

Przykład: Szkic dla arduino, który ma być nadrzędny (master). Płytka nadrzędna ma odczytywać dane z płytki podrzędnej i wyświetlać otrzymaną informację na monitorze poru szeregowego.

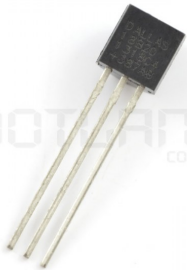
```
#include <Wire.h>

void setup()
{
    Wire.begin();
    Serial.begin(9600);
}

void loop()
{
    Wire.requestFrom(2,6);

    while(Wire.available())
    {
        char c=Wire.read();
        Serial.print(c);
    }
}
```

Przesyłanie danych korzystając z magistrali I2C pomiędzy dwoma płytkami arduino

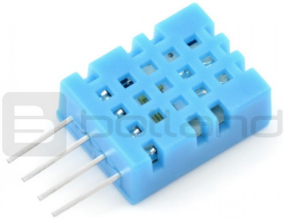
Przykład: Szkic dla arduino, który ma być podrzędny (slave). Płytka podrzędna ma wysyłać informacje do płytki nadrzędnej. Po wgraniu dwóch szkiców, płytka nadrzędna ma zostać podłączona do komputera i będzie zasilała płytkę podrzędną.

```
#include <Wire.h>

void setup()
{
    Wire.begin(2); //Podlacz sie do I2C z adresem 2
    Wire.onRequest(wyslijTekst); //Jak master wysle zapytanie to
       ↪ zostanie wykonana funkcja wyslijTekst
}

void wyslijTekst()
{
    Wire.write("hello");
}

void loop()
{
    delay(100);
}
```

Dlaczego 1-Wire?



Cyfrowy termometr DS18B20



Czujnik temperatury i wilgotności DHT11

iButton

Pamięć flash EEPROM DS24B33S+

1-Wire

- Interfejs 1-Wire został zaprojektowany przez Dallas Semiconductor. Jego zaletą jest minimalizacja użytych linii do minimum (czyli jednej) oraz pasożytnicze zasilanie. Natomiast jest to magistrala wolniejsza od I2C (16kb/s).
- Do magistrali 1-Wire można podłączyć do 255 urządzeń.
- Koncepcja działania magistrali jest zbliżona do I2C. Główna różnica polega do podejściu w definiowaniu logicznych wartości i bitu startu oraz ACK. Te bity są definiowane przez czas trwania sygnału niskiego (zwarcia linii do masy).
- Sygnał reset następuje gdy urządzenie master wyśle stan niski przez 480us(zwarcie linii do masy) a zgłoszenie swojej obecności przez urządzenie slave gdy wyśle stan niski przez 240us.
- Gdy urządzenie typu master chce wysłać wartość 1 to ustawia stan niski o czasie trwania 1-15us a następnie rozwiera linię na 60us. Logiczne zero odpowiada zwarciu linii na czas 60us.
- Urządzenie slave jak chce wysłać wartość 1 to nic nie robi a jak 0 to zwiera linię na 60us.
- Przed odbiorem każdego bitu master wysyła stan niski o długości 1-15us (zwiera linię) a następnie powraca do stanu wysokiego.

1-Wire - biblioteka OneWire

- Biblioteka stworzona przez Dallas Semiconductor do obsługi 1-Wire należy pobrać ze strony: playground.arduino.cc oraz rozpakować do katalogu libraries.
- W celu skorzystania z biblioteki należy ją dołączyć do szkicu:

#include <OneWire.h>

- Następnie należy utworzyć obiekt OneWire:

OneWire bus(numer pinu, który ma służyć nam jako linia);

- Funkcja zwracająca adresy modułów podłączonych do magistrali 1-wire:

bus.search(tablica typu byte, do której zostaną zapisane adresy);

- Funkcja, do resetu:

bus.reset();

- Funkcja, do wyboru urządzenia, z którym ma się komunikować:

bus.select(adres urządzenia, z którym chcemy się skomunikować);

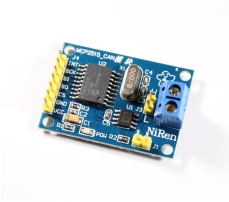
- Funkcja, do wysyłania danych:

bus.write(dana,czy zasilanie pasożytniczej 1/0);

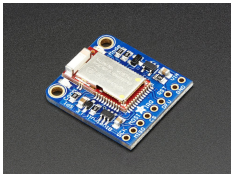
- Funkcja do odczytu:

bus.read();

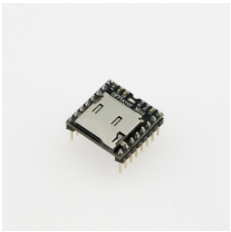
Dlaczego SPI?



Moduł CAN-BUS na MCP2515



Bluetooth Low Energy



Moduł do karty microSD

SPI LCD module with SD

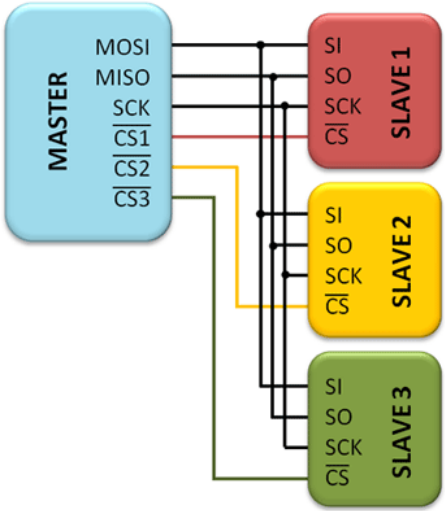
RFID Cards, and RFID Fobs

SPI WiFi Module

Serial Peripherial Interface

- SPI jest interfejsem typu full-duplex, co oznacza, że dane mogą być wysyłane i odbierane w tym samym czasie.
- SPI jest interfejsem synchronicznym, co oznacza, że jednym z przewodów przesyła się sygnał zegarowy, synchronizujący wszystkie układy. Generalnie SPI składa się z 4 linii:
  1. MOSI (master out, slave in) – linia łącząca wyjście danych z mastera i wejścia slave'ów.
  2. MISO (master in, slave out) – linia łącząca wyjście danych slave i wejście mastera.
  3. SCK (serial clock) – sygnał zegarowy, synchronizujący układy na magistrali.
  4. CS (chip select) – sygnał informujący slave o rozpoczęciu transmisji. Oznaczany często jako SS.
- Możliwe jest połączenie wielu układów natomiast może być tylko jeden układ typu master.
- Zaletą SPI jest efektywna komunikacja z wybranym urządzeniem peryferyjnym, przy uśpieniu pozostałych dzięki linii SS.
- Wadą SPI jest wykorzystanie aż 4 linii do komunikacji.

Połączenie urządzeń typu slave przez SPI



Rysunek: Schemat podłączenia urządzeń slave przez SPI. [1]

[1] Źródło: mikrokontroler.pl

Biblioteka SPI

- W ceku skorzystania z SPI należy dołączyć bibliotekę:

```
#include "SPI.h"
```

- Następnie należy skonfigurować pin podłączony do SS jako wyjście oraz wysłać stan HIGH dlatego, że większość urządzeń SPI jest aktywowane niskim stanem:

```
pinMode(SS, OUTPUT);
digitalWrite(SS,HIGH);
```

- W celu aktywowania magistrali SPI należy skorzystać z funkcji:

```
SPI.begin();
```

- Następnie należy określić sposób wysyłania i odbierania danych, który zależy od rodzaju czujnika podłączonego do magistrali.

```
SPI.setBitOrder(MSBFIRST/MSBLAST);
```

Biblioteka SPI

- Wysyłanie danych do urządzenia SPI:
  - Wysłanie stanu LOW na pin SS (poinformowanie, że urządzenie nadrzędne nawiązuje komunikacje):

    digitalWrite(SS, LOW);
  - Wysłanie bajtu danych:

    SPI.transfer(byte);
  - Wysłanie stanu HIGH na pinie SS (poinformowanie urządzenia slave o końcu komunikacji):

    digitalWrite(SS,HIGH);
- Odczyt danych z urządzenia slave:

  char dana=SPI.transfer(0);

SPI - przykład

Przykład: Wysłanie danych na magistrale SPI

```
#include "SPI.h"
#define SS 2
byte value=B01011111;

void setup()
{
    SPI.begin();
    pinMode(SS, OUTPUT);
    digitalWrite(SS,HIGH);
    SPI.setBitOrder(MSBFIRST);
}

void loop()
{
    digitalWrite(SS,LOW);
    SPI.transfer(value);
    digitalWrite(SS,HIGH);
}
```

THAT'S ALL FOR TODAY....
ANY QUESTION??