Podstawy Systemów Wbudowanych

Wykład 1

Angelika Tefelska Dariusz Tefelski

Zakład Fizyki Jądrowej, Wydział Fizyki PW

Outlook

Informacje organizacyjne

- 12 wykładów po 45 min:
  - 5 zajęcia poświęcone arduino
  - 7 zajęcia poświęcone raspberry pi
- 11 zajęć laboratoryjnych po 3h:
  - 6 zajęcia poświęcone arduino
  - 4 zajęcia poświęcone raspberry pi
  - projekt
- Konsultacje: termin ustalany indywidualnie z osobą zainteresowaną drogą mailową: angelika.tefelska@pw*

Zasady zaliczenia:

- wykonanie zadań laboratoryjnych na poziomie podstawowym (ocena 3.0)
- wykonanie projektu na zajęciach (ocena $>$ 3.0)

Ocenę końcową może prowadzący zwiększyć o 0.5 na podstawie pracy studenta na zajęciach laboratoryjnych.

Literatura

- John Boxall „Arduino 65 praktycznych projektów". Wydawnictwo Helion 2014
- Simon Monk „Arduino dla początkujących. Kolejny krok". Wydawnictwo Helion
- Simon Monk „Raspberry Pi. Receptury". Wydawnictwo Helion 2014
- Donald Norris „Raspberry Pi. „Niesamowite projekty. Szalony Geniusz". Wydawnictwo Helion 2014

Materiały do zajęć

Wszystkie materiały do zajęć (wykłady, instrukcje do laboratoriów) będą umieszczane na stronie `http://labe.engined.eu` w zakładce PSW.

Co to są systemy wbudowane?

Ogólna def. systemu wbudowanego

System wbudowany to urządzenie używane do kontroli, monitoringu lub wspomagania pracy urządzeń i maszyn. Pojęcie "wbudowane" oznacza, że stanowią one integralną część większego systemu.

Def. Systemy Wbudowane

System wbudowany (ang. embedded system) – system komputerowy specjalnego przeznaczenia, który staje się integralną częścią obsługiwanego przez niego sprzętu komputerowego (hardware).[a]

[a] Źródło: wikipedia

Zastosowania

- automatyka przemysłowa, robotyka
- motoryzacja
- **w instytucjach badawczych np. CERN (eksperyment NA61/SHINE i inne)**
- **diagnostyka medyczna**
- **nawigacja satelitarna, lotnictwo**
- telekomunikacja
- sprzęt biurowy (drukarki, kalkulatory), elektroniczny, AGD
- **sprzęt pomiarowy (oscyloskop, analizator widma)**
- bankomaty
- sterowniki PLC stosowane w przemyśle do sterowania i kontroli procesów oraz maszyn
- systemy alarmowe
- sprzęt komputerowy (dyski twarde, napędy optyczne, routery)
- rozrywka (konsole do gier)
- i wiele, wiele innych...

Cechy systemu wbudowanego:

- dedykowany charakter (poziom rozbudowania dobrany do pełnionej funkcji)
- silnie zintegrowany
- ograniczony interfejs (często brak GUI)
- niski pobór energii
- stabilność układu (duża odporność na awarie i zakłócenia)
- poddany licznym testom zarówno od strony sprzętowej jak i oprogramowania
- bezobsługowy

Struktura systemu wbudowanego:

- Obwody wejściowe (rejestracja sygnałów, akwizycja danych) $\rightarrow$ czujniki, liczniki, komparatory, przetworniki A/C itd.
- Jednostka centralna (analiza sygnałów wejściowych, sterowanie) $\rightarrow$ zazwyczaj mikrokontroler
- Obwody wyjściowe $\rightarrow$ przetworniki C/A, układy PWM, wyjścia portów komunikacyjnych itd.

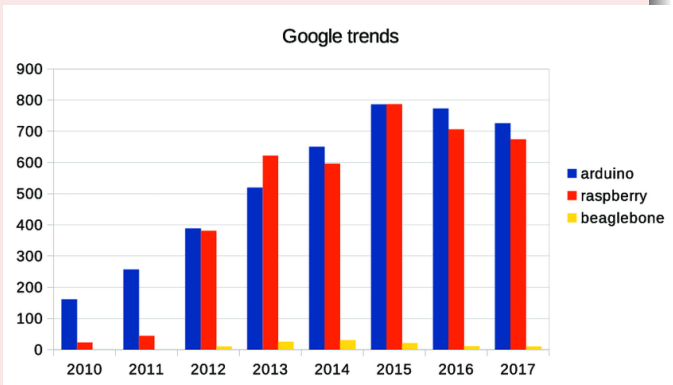
Platformy sprzętowe

1. Rozwiązania dedykowane:
   - Gotowe moduły mikrokontrolerów
   - Komputery przemysłowe
   - Sterowniki PLC
2. Rozwiązania uniwersalne:
   - **platformy uniwersalne (np. arduino)**
   - **układy *System On Chip* (np. Raspberry Pi)**

Obecny rynek systemów wbudowanych

Najpopularniejsze systemy wbudowane i ich sprzedaż:

- **Arduino** - projekt rozpoczął się w 2005 roku i już sprzedano ok 20M płytek. Prawdopodobnie rzeczywista sprzedaż płytek jest kilka razy większa ze względu na liczne podróbki.
- **Raspberry Pi** - premiera w 2012 roku, do tej pory sprzedano również ok 20M płytek.



Źródło: https://www.researchgate.net/publication/324099714_Real-Time_Processing_Library_for_Open-Source_Hardware_Biomedical_Sensorsg

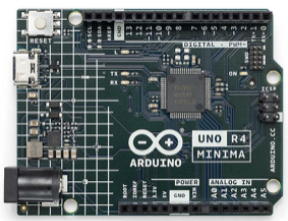
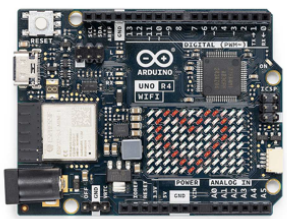
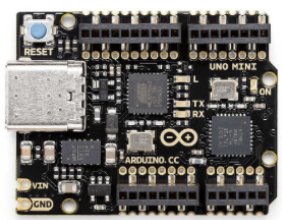
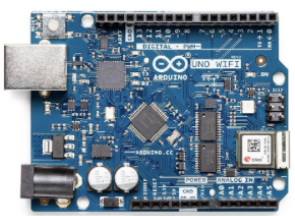
- Profesor Massimo Banzi (Interaction Design Institute Ivrea, Włochy) poszukiwał ekonomicznego rozwiązania mającego ułatwić studentom projektowania tworzenie interaktywnych prac.
- Swoim problemem podzielił się z Davidem Cuartiellesem z Uniwersytetu w Malmo (Szwecja), z którym opracował koncepcje arduino.
- Podobne produkty dostępne na rynku były kosztowne i trudne w obsłudze dla studentów sztuk pięknych.
- Z tego powodu zdecydowano stworzyć własną płytkę, która miała być prosta w obsłudze i tania (cena porównywalna do ceny pizzy we Włoszech).


CZY

?

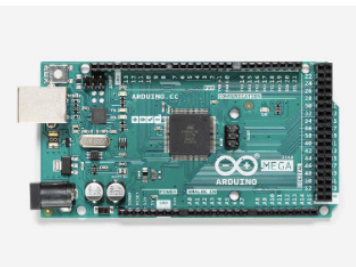
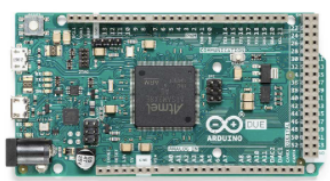
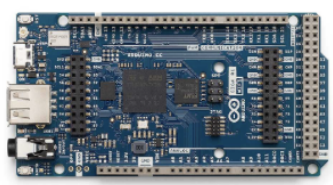
- Płytkę sterownika zaprojektował David Cuartielles a oprogramowanie zostało napisane prze studenta profesora M.Banzi - Davida Mellis.
- Płytkę nazwano na pamiątkę lokalnego baru (*Arduin of Ivrea*), do którego chodzili studenci i kadra instytutu.

- Podstawowa seria płytek Arduino, w której najprostsza i najpopularniejsza płytka to Arduino UNO.
- Druga najpopularniejsza płytka z tej serii to Arduino Leonardo, które różni się tym od UNO, że ma wbudowaną obsługę USB co umożliwia emulację klawiatury lub myszy.

			
Arduino UNO R4 Minima	Arduino UNO R4 WiFi	Arduino UNO R3	Arduino Leonardo
			
Arduino UNO Mini Limited Edition	Arduino Micro	Arduino Zero	Arduino UNO WiFi Rev2

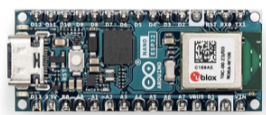
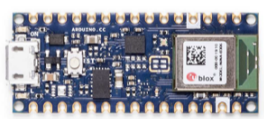
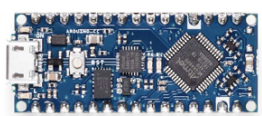
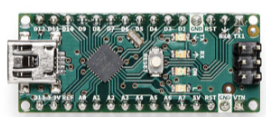
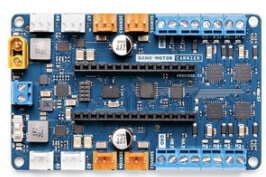
Źródło: `https://www.arduino.cc/en/hardware`

- **Arduino Mega 2560** - większa wersje Arduino UNO oparta na 8-bitowym ATmega2560, która zawiera 256kB pamięci flash i 8kB pamięci RAM. Ponadto zwiększono liczbę portów cyfrowych wejścia/wyjścia do 54 (z czego 14 może zapewnić sygnał PWM) oraz liczbę portów analogowych do 16.
- Arduino Due - płytka oparta na 32-bitowym ARM Cortex-M3, która zawiera 512 kB pamięci flash i 96 kB RAM ale działa na napięciu 3,3V zamiast 5V.
- Arduino Giga R1 jest oparty na 32-bitowym ARM Cortex-M7/M4 i posiada 2MB pamięci flash oraz 1MB RAM. Ponadto płytka jest wyposażona w Wi-Fi (ESP32) oraz Bluetooth 5.1 ale również działa na napięciu 3.3V.

			
Arduino Mega 2560 Rev3	Arduino Due	Arduino GIGA R1 WiFi	Arduino GIGA Display Shield

Źródło: `https://www.arduino.cc/en/hardware`

- Płytki Nano charakteryzują się niewielkimi rozmiarami.
- W zależności od wariantu mogą być wyposażone w moduł Bluetooth/Wi-Fi.
- Posiadają wbudowane czujniki, które umożliwiają pomiar: temperatury/wilgotności, ciśnienia, itd..
- Można je również programować za pomocą MicroPython i obsługują uczenie maszynowe.

			
Arduino Nano 33 IoT	Arduino Nano RP2040 Connect	Arduino Nano ESP32	Arduino Nano 33 BLE Sense
			
Arduino Nano 33 BLE	Arduino Nano Every	Arduino Nano	Arduino Nano Motor Carrier

Źródło: `https://www.arduino.cc/en/hardware`

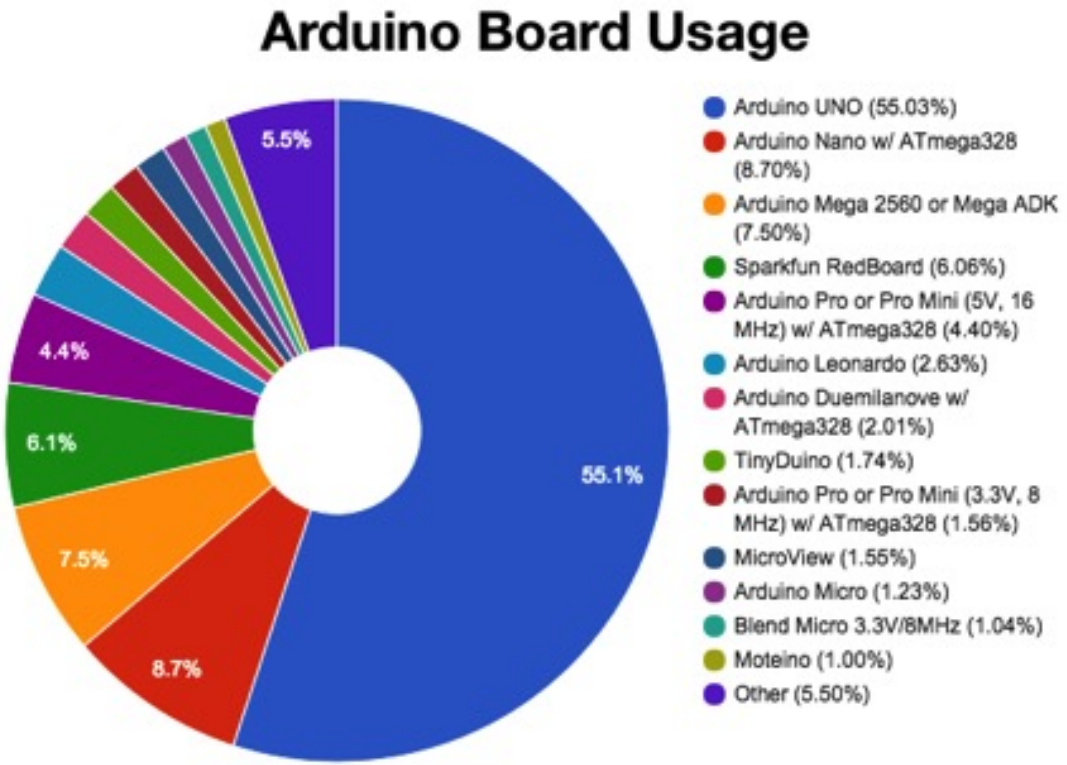
	Arduino Nano	Nano ESP32-S3	Nano 33 IoT	Nano 33 BLE	Nano 33 BLE Sense Rev2	Nano RP2040 Connect	Nano Every	Nano Matter
Microcontroller	ATmega328	ESP32-S3	SAMD21 Arm® Cortex®-M0+ / u-blox® NINA-W102	nRF52840 / u-blox® NINA-B306	nRF5284 / u-blox® NINA-B306	Raspberry Pi RP2040 / u-blox® NINA-W102	ATMega4809	Silicon Labs MGM24
USB connector	Mini-B USB	USB-C®	Micro USB	Micro USB	Micro USB	Micro USB	Micro USB	USB-C®
I/O voltage	5 V	3.3 V	3.3 V	3.3 V	3.3 V	3.3 V	5 V	3.3 V
Input range	7-12 V	5-21 V	7-12 V	7-12 V	7-12 V	5-21 V	7-12 V	5-21 V
Clock speed	16 MHz	up to 240 MHz	48 MHz	64 MHz	64 MHz	133 MHz	16 MHz	78 MHz
SRAM	2 kB	512 kB	256 kB	256 kB	256 kB	264 kB	6 kB	256 kB
Flash	32 kB	16 MB	1 MB	1 MB	1 MB	16 MB	48 kB	1536 kB
Wi-Fi®	✗	✓	✓	✗	✗	✓	✗	-
Bluetooth® LE	✗	✓	✓	✓	✓	✓	✗	✓
Digital inputs	20	22	22	14	14	22	22	22
Analog inputs	8	8	8	8	8	8	8	8

Źródło: `https://store.arduino.cc/pages/nano-family`

- Rodzina MRK to seria płytek, które można ze sobą łączyć w celu tworzenia projektów bez żadnych dodatkowych obwodów.
- Każda płytka jest wyposażona w moduł radiowy (oprócz MKR Zero), który umożliwia komunikację Wi-Fi, Bluetooth, LoRa, Sigfox i NB-IoT.
- Wszystkie płytki oparte są na 32-bitowym procesorze ARM Cortex-M0 o niskim poborze mocy i są wyposażone w układ kryptograficzny zapewniający bezpieczną komunikację.

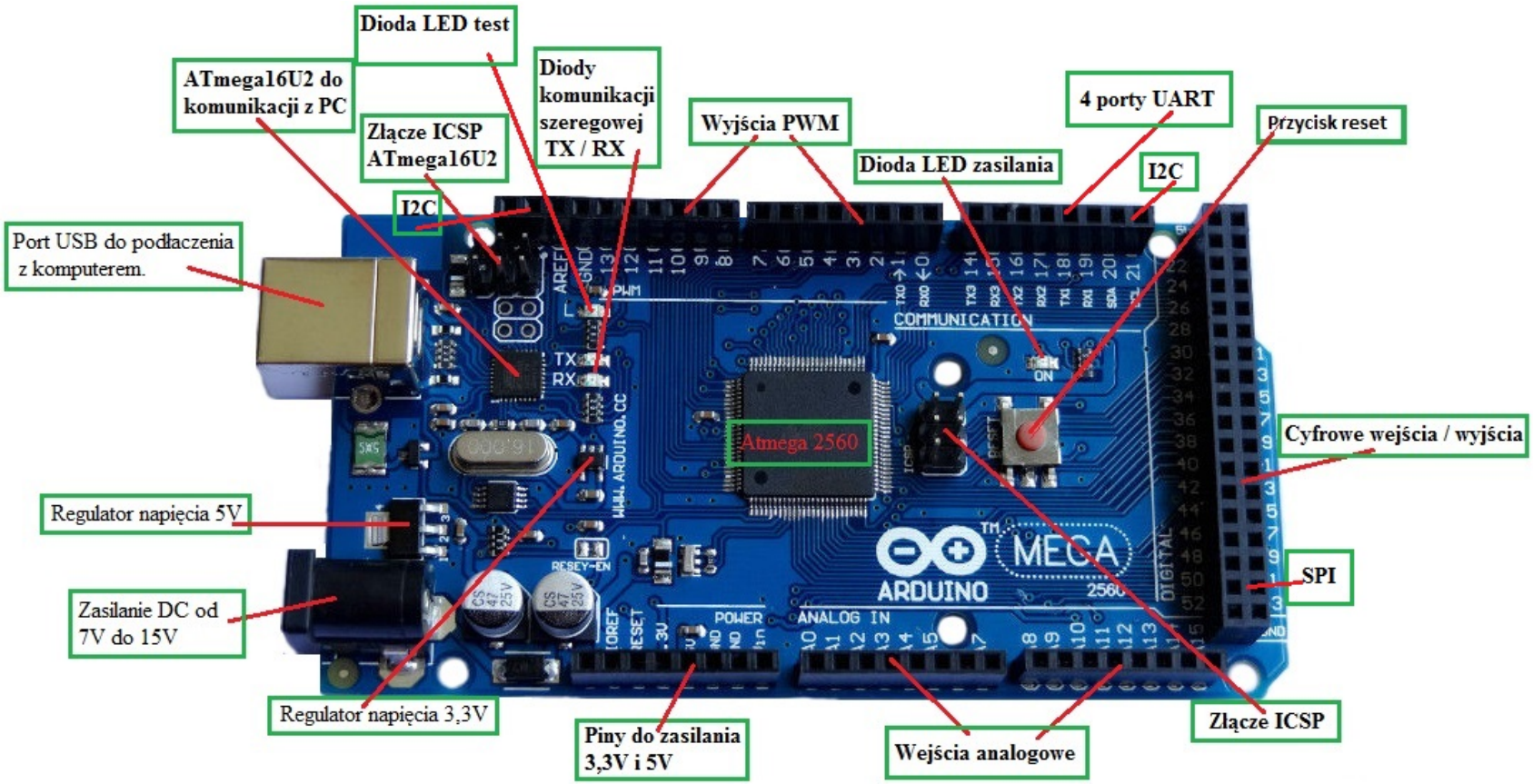
Źródło: `https://www.arduino.cc/en/hardware`

Popularność poszczególnych wersji Arduino



Źródło: `https://cdn.sparkfun.com/assets/home_page_posts/1/9/8/2/image1.jpg`

Moduł Arduino MEGA2560 R3



Rys.: Arduino Mega2560 R3. [1]

[1] Źródło: http://www.fvr.pl

Środowisko programistyczne

- Środowisko programistyczne IDE (*Integrated Development Enviroment*) jest oparte na uproszczonym języku C++.
- Środowisko IDE jest do pobrania ze strony: `http://arduino.cc/en/Main/Software`.
- Dla systemu Windows jest gotowy instalator.
- Dla dystrybucji Linux-a jest do pobrania paczka, którą należy rozpakować. Uwaga: po rozpakowaniu paczki upewnij się, że użytkownik ma prawo dostępu do odczytu/zapisu z portu szeregowego. Jeśli nie to dodaj użytkownika do grupy dialout.
- Po zainstalowaniu IDE należy wybrać rodzaj płytki (Narzędzia $\rightarrow$ Płytka $\rightarrow$ Arduino Mega 2560) oraz port (Narzędzia $\rightarrow$ Port Szeregowy $\rightarrow$ Nazwa portu).

Rys.: Środowisko programistyczne Arduino

Struktura programu

Każdy program musi zawierać:

- **void setup()** - w której zamieszcza się instrukcje, które mają się wykonać tylko raz po uruchomieniu programu (np. konfiguracja portów).
- **void loop()** - instrukcje, które mają się wykonywać ciągle (pętla analogiczna do while(1)).

Inicjalizacja portów

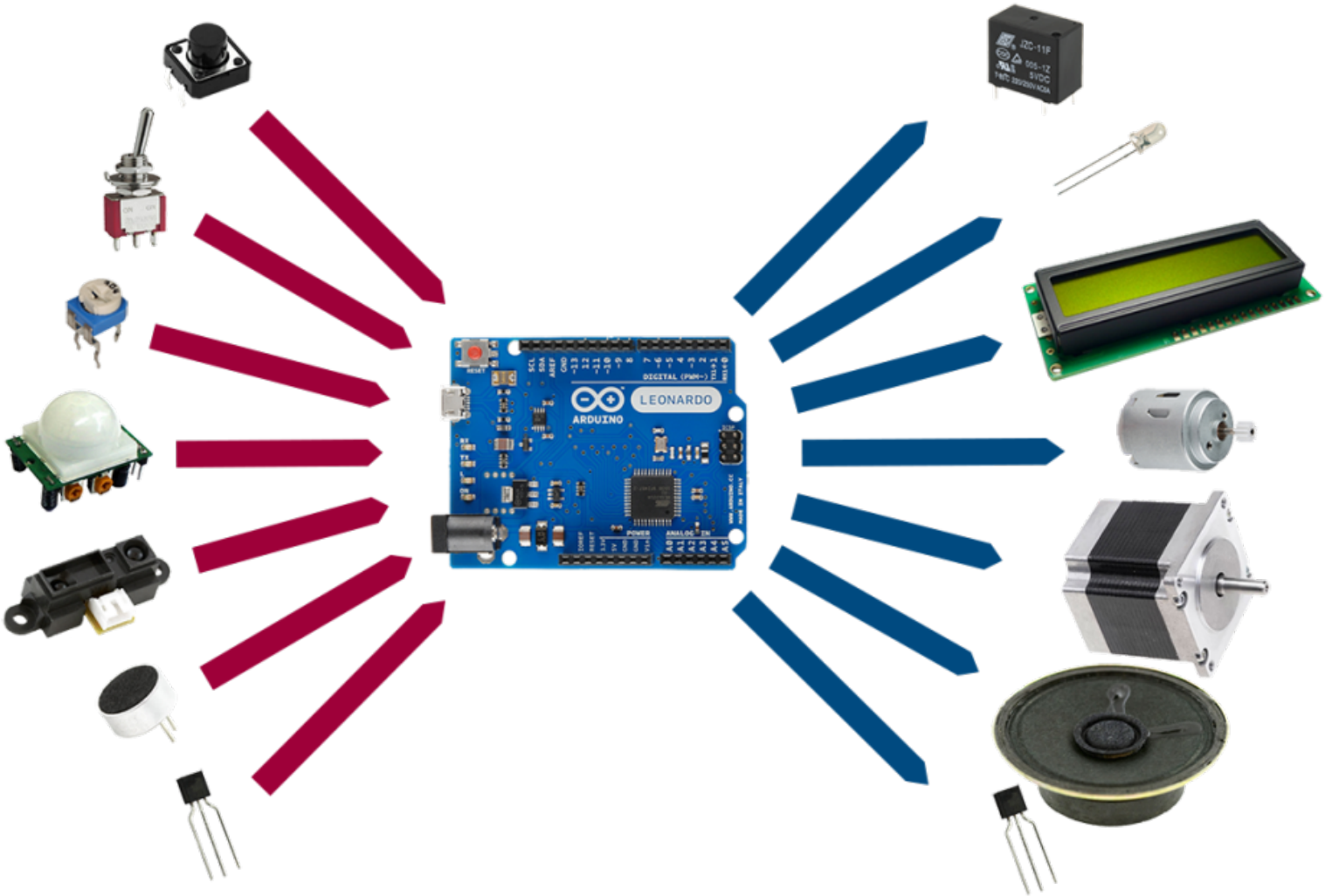
W celu zainicjowania danego portu należy skorzystać z funkcji:

pinMode(numer portu, INPUT/OUTPUT).

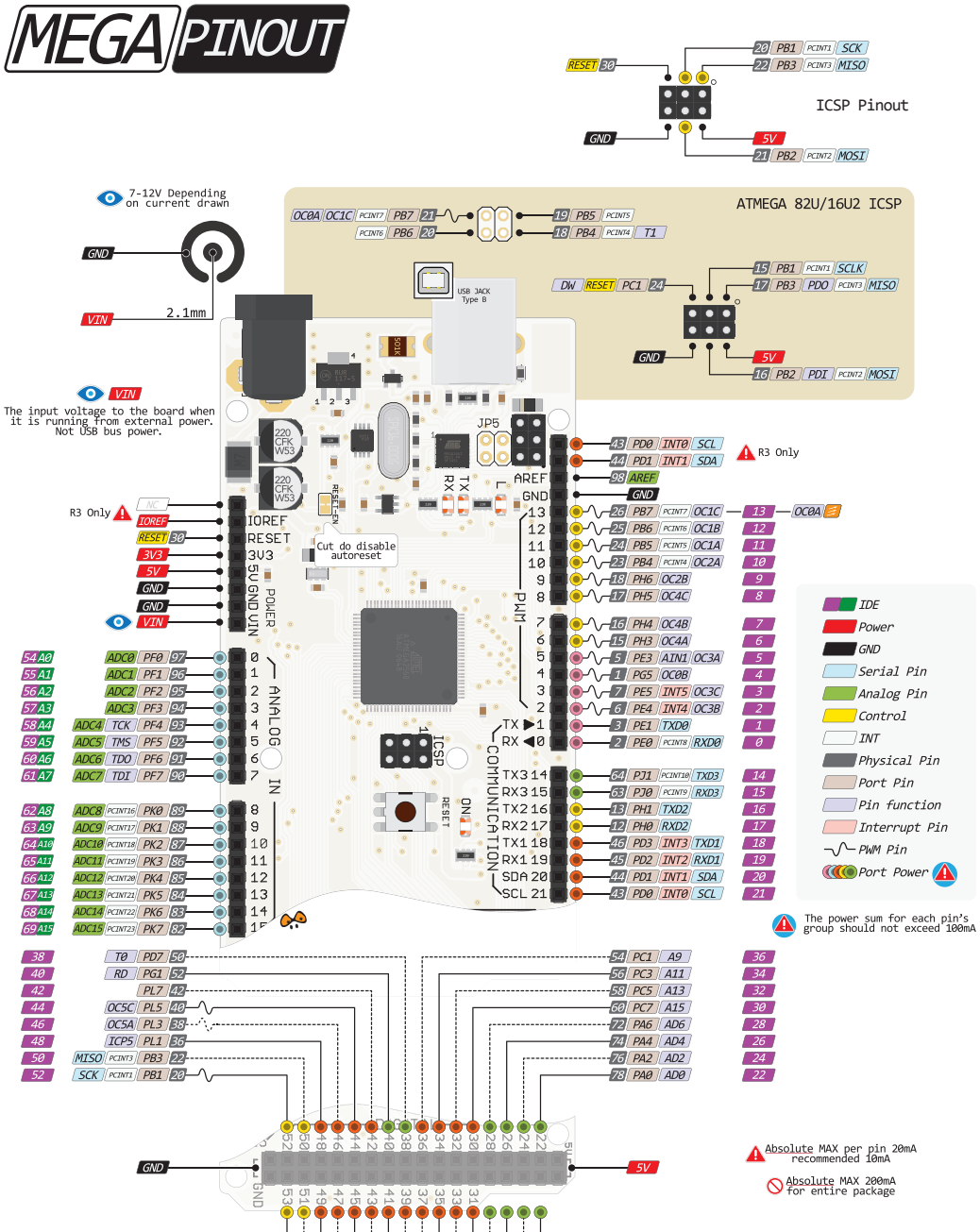
Przykład: ustawienie pinu nr 2 jako wyjściowy a pinu nr 3 jako wejściowy:

```
void setup()
{
    pinMode(2,OUTPUT);
    pinMode(3,INPUT);
}
```

Elementy wejściowe i wyjściowe



Źródło obrazka: `https://blog.codebender.cc/2014/03/07/lesson-1-inputs-and-outputs/`


MEGA PINOUT
ICSP Pinout
ATMEGA 82U/16U2 ICSP
7-12V Depending on current drawn
2.1mm
The input voltage to the board when it is running from external power. Not USB bus power.
Cut do disable autoreset
R3 Only
POWER
ANALOG IN
PWM
COMMUNICATION
IDE
Power
GND
Serial Pin
Analog Pin
Control
INT
Physical Pin
Port Pin
Pin function
Interrupt Pin
PWM Pin
Port Power
The power sum for each pin's group should not exceed 100mA
Absolute MAX per pin 20mA recommended 10mA
Absolute MAX 200mA for entire package

1. Wysłanie stanu HIGH/LOW na pin cyfrowy:

digitalWrite(numer pinu, HIGH/LOW);

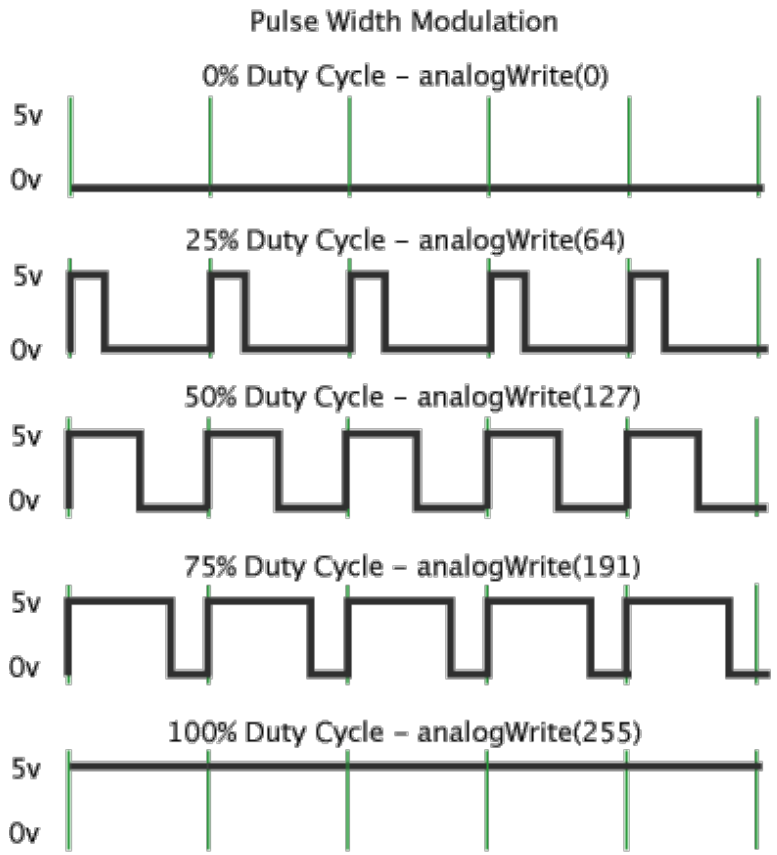
Przykład: zapalanie i gaszenie diody LED podłączonej do pinu nr 2:

```
unsigned int LED=2; // Lub #define LED 2

void setup()
{
    pinMode(LED,OUTPUT);
}

void loop()
{
    digitalWrite(LED, HIGH);
    delay(100); //Poczekaj 100ms
    digitalWrite(LED, LOW);
    delay(100);
}
```

Modulacja szerokości impulsu PWM

2. Funkcja do ustawienia szerokości impulsu (wypełnienie w zakresie (0,255)):

analogWrite(numer pinu,wypełnienie);



Źródło: www.arduino.cc

3. Odczyt stanu z pinu cyfrowego:

digitalRead(numer pinu);

Przykład: zapalenie diody LED w trakcie naciśnięcia przycisku:

```
#define button 5
#define LED 4

void setup()
{
    pinMode(button, INPUT);
    pinMode(LED, OUTPUT);
}

void loop()
{
    if(digitalRead(button)==HIGH) digitalWrite(
        LED, HIGH);
    else digitalWrite(LED, LOW);
}
```

4. Odbiór danych z wejścia analogowego:

analogRead(numer pinu);

- Funkcja ta zwraca wartości z zakresu (0, 1023) gdyż Arduino jest wyposażone w przetwornik 10-bitowy.
- Wartość 1023 odpowiada napięciu 5 V.

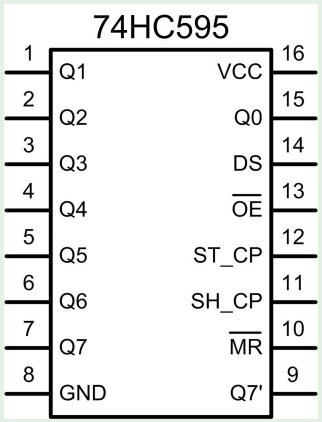
Przykład: odczyt napięcia z pinu analogowego o numerze 0:

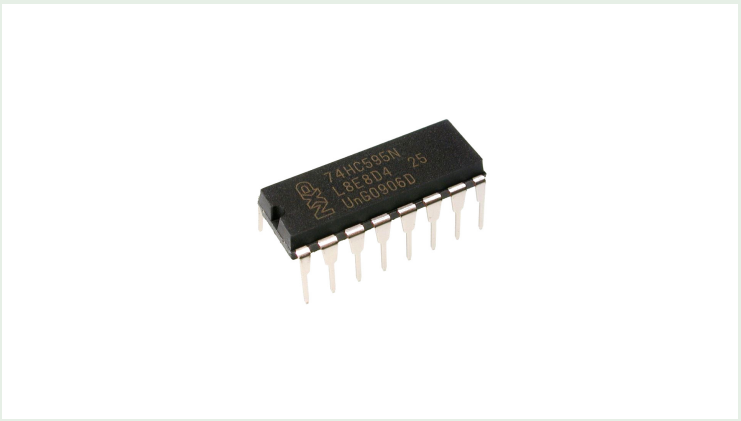
```
unsigned int analogValue;
double voltage;

void setup()
{

}

void loop()
{
    analogValue=analogRead(0);
    voltage=analogValue*5.0/1024.0;
}
```

5. **UWAGA: nie ma wyjść analogowych!**

Rozszerzenie liczby portów

- Czasami projekty są tak rozbudowane, że niezbędne jest skorzystanie z większej liczby pinów. Z pomocą może przyjść rejestr przesuwny 74HC595.





- W rejestrze przesuwnym dane wysłane na nóżkę 14 z płytki Arduino zostają przekazane na wyjścia Q7-Q0. Przy czym na Q7 trafia pierwszy bit otrzymany z Arduino przez rejestr a na Q0 ostatni.
- Pin 9 służy do wysyłania danych do ewentualnego kolejnego rejestru przesuwającego.
- Piny 11 i 12 to odpowiednio zegar i zatrzask.
- Przykład połączenia rejestru przesuwnego 74HC595 z płytką Arduino został przedstawiony w instrukcji do laboratorium nr 1.

Rozszerzenie liczby portów

W celu przekazania bajtu danych do rejestru przesuwnego należy:

- Wysłać stan *LOW* do pinu podłączonego do zatrzasku
- Skorzystać z funkcji do wysyłania bajtu danych do rejestru przesuwnego:

  shiftOut(pin połączony z nóżką 14 rejestru, pin połączony z zegarem, LSBFIRST/MSBFIRST, bajt danych);
- Wysłać stan *HIGH* do pinu podłączonego do zatrzasku.

Rozszerzenie liczby portów

W celu przekazania bajtu danych do rejestru przesuwnego należy:

- Przykład: wysłanie bajtu danych na rejestr przesuwny

```
#define DATA 2 //pin 2 z pinem 14 rejestru 74HC595
#define LATCH 3 //pin 3 z pinem 12 rejestru 74HC595
#define CLOCK 4 //pin 4 z pinem 11 rejestru 74HC595

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

void loop()
{
    digitalWrite(LATCH,LOW);
    shiftOut(DATA,CLOCK,LSBFIRST,3);
    digitalWrite(LATCH,HIGH);
}
```

Przerwania

- Przerwanie - to sygnał umożliwiający wywołanie określonej funkcji w dowolnym czasie w trakcie wykonywania programu. Gdy taki sygnał zostanie wykryty to instrukcje wykonywane w ramach pętli loop zostaną wstrzymane. Następnie zostanie wykonana funkcja obsługująca dane przerwanie. Po jej zakończeniu zostanie wznowione wykonywanie instrukcji z pętli loop.
- **UWAGA: pamiętaj aby instrukcje wykonywane w ramach funkcji przerwania były możliwie jak najkrótsze!**

Tryby przerwań

- LOW - na pinie przerwania nie ma napięcia
- CHANGE - napięcie na pinie przerwania zmieniło swoją wartość (5V $\rightarrow$ 0V lub 0V $\rightarrow$ 5V)
- RISING - napięcie na pinie przerwania zmieniło się z 0V na 5V
- FALLING - napięcie na pinie przerwania zmieniło się z 5V na 0V

Rodzaje przerwań dla Arduino Mega

- Int0 (pin 2)
- Int1 (pin 3)
- Int2 (pin 21)
- Int3 (pin 20)
- Int4 (pin 19)
- Int5 (pin 18)

Konfiguracja przerwań

- Funkcja służąca do konfiguracji przerwania:

  attachInterrupt(rodzaj przerwania, funkcja która ma się wykonać, tryb przerwania);

- Funkcja służąca do dezaktywacji przerwań:

  noInterrupts();

- Funkcja służąca do ponownego włączenia przerwań:

  interrupts();

Przykład zastosowania przerwań

Zapalenie diody po włączeniu przycisku korzystając z przerwania int0:

```
#define led 3

void setup()
{
    pinMode(led, OUTPUT);
    attachInterrupt(0,ledON, RISING);
}

void ledON()
{
    digitalWrite(led, HIGH);
}

void loop()
{

}
```

UWAGA: jeśli wartość jakieś zmiennej ma być zmieniona w funkcji przerwania to musi być ona typu volatile !

Komunikacja przez port szeregowy

- W celu otworzenia monitora portu szeregowego należy wybrać w IDE: ***Narzędzia*** $\rightarrow$ ***Monitor Portu Szeregowego***
- Inicjalizacja portu szeregowego:

  Serial.begin(przepustowość);
- Wysłanie tekstu do monitora portu szeregowego:

  Serial.print("Arduino jest super!");
- Wysłanie tekstu do monitora portu szeregowego ze znakiem nowej linii:

  Serial.println("Raspberry Pi też jest fajne!");
- Wysłanie liczby do monitora portu szeregowego:

  Serial.println(wartość, liczba miejsc po przecinku);

Komunikacja przez port szeregowy - przykład

Przykład: wyświetlenie w monitorze portu szeregowego wartości z wejścia analogowego.

```
unsigned int analogValue=0;

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

void loop()
{
    analogValue=analogRead(0);
    Serial.print("Wartosc=");
    Serial.println(analogValue);
}
```

Komunikacja przez port szeregowy

- Funkcja do usuwania danych z bufora:

  Serial.flush();

- Funkcja do sprawdzania nowych danych w buforze portu szeregowego:

  Serial.available()

- Odczyt danych z portu szeregowego:

  Serial.read()

Komunikacja przez port szeregowy - przykład

Przykład: odczyt danych z portu szeregowego.

```
int number=0;
int readValue=0;
void setup()
{
    Serial.begin(9600);
}
void loop()
{
    Serial.flush();
    while(Serial.available()>0)
    {
        number=number*10;
        readValue=Serial.read()-'0';
        number=number+readValue;
        delay(5);
    }
    Serial.print("Wpisano: ");
    Serial.println(number);
    number=0;
}
```

Serial Plotter (pl. Kreślarka)

- Czasem zauważenie zmian wartości wyświetlanych w monitorze portu szeregowego jest trudne. Zwłaszcza gdy wartości są wyświetlane bardzo szybko.
- Z pomocą przychodzi kreślarka. Aby wyświetlić dane w kreślarce należy wysłać je na port szeregowy korzystając z funkcji **Serial.println(wartość)** bez dodatkowych komentarzy. Przykład: