Katedra Mikroelektroniki i Technik Informatycznych

Politechnika Łódzka

Instrukcja do laboratorium

Systemy mikroprocesorowe

Wersja dla II stopnia EiT

(program studiów 2019)

Wykorzystywany sprzęt - zestawy dydaktyczne:

- Arduino Uno Plus
- Arduino Mega2560

Autor: Zbigniew Kulesza

Wersja 0.9

Sieradz 2020

Regulamin laboratorium

1. Zaliczenia laboratorium odbywa się na podstawie pełnego sprawozdania zawierającego wszystkie ćwiczenia z własnymi uwagami i obliczeniami studenta oraz na podstawie ocen cząstkowych z subkolokwiów przeprowadzanych przez prowadzącego zajęcia.

2. Laboratorium jest obowiązkowe w związku z czym jedna nieusprawiedliwiona nieobecność powoduje niezaliczenie przedmiotu.

3. Sprawdzenie przygotowania do zajęć może być przeprowadzone za pomocą kolokwium. W przypadku stwierdzenia nieprzygotowania do zajęć student usuwany jest z laboratorium. Przewidziany jest jeden termin poprawkowy na odrobienie ćwiczeń.

4. W czasie laboratorium mogą przebywać wyłącznie osoby prowadzące lub uczestniczące w zajęciach, lub bezpośrednio związane z obsługą laboratorium.

5. Praca w katalogu innym niż własny jest zabroniona.

6. Otwieranie zbiorów innych niż własne powoduje natychmiastowe usunięcie z zajęć.

7. Praca w laboratorium odbywa się za pomocą komputerów połączonych w sieć. Usiłowanie samodzielnej zmiany przydzielonych zasobów, łamanie lub korzystanie z cudzych haseł powoduje skreślenie z laboratorium.

Spis treści

1. Informacje wstępne na temat AVR

1.1. Specyfikacja AVR ATmega328P i ATmega2560

Specyfikacja ATmega328 oraz ATmega2560		
Architektura:	AVR RISC, zmodyfikowany Harward	
ATmega - rodzina:	megaAVR - seria ATmega: • Pamięć programu 4-256 KB • Pakiet 28-100-pin • Rozszerzony zestaw instrukcji (wielokrotne instrukcje i instrukcje obsługi większych pamięci programów) • Rozbudowany zestaw urządzeń peryferyjnych	
Wersja układu	ATmega328P	ATmega2560
Cechy ogólne architektury	● 8-bitowy mikrokontroler AVR ® o wysokiej wydajności i niskiej mocy ● Zaawansowana architektura RISC ● 131 instrukcji - większość wykonywane w pojedynczym cyklu zegara ● 32 x 8 rejestrów roboczych ogólnego przeznaczenia ● W pełni statyczna praca ● Przepustowość do 16MIPS przy 16 MHz ● Mnożarka 2-cyklowa	• 8-bitowy mikrokontroler AVR Atmel o wysokiej wydajności i niskiej mocy • Zaawansowana architektura RISC - 135 instrukcji - większość wykonywana w jednym cyklu zegara - 32 × 8 rejestrów roboczych ogólnego zastosowania - W pełni statyczna praca - Przepustowość do 16 MIPS przy 16 MHz - 2-cyklowy multiplikator na chipie
Cechy pamięci	● Segmenty pamięci nieulotnej ● 32 KB bajtów wbudowanej samoprogramowalnej pamięci flash ● 1KB EEPROM ● Wewnętrzna pamięć SRAM 2 KB ● Cykle zapisu / kasowania: 10 000 pamięci flash / 100 000 pamięci EEPROM ● Opcjonalna sekcja kodu rozruchowego z niezależnymi bitami ustawień ● Programowanie w systemie za pomocą programu rozruchowego na chipie ● Prawdziwa operacja odczytu w trakcie	• Segmenty pamięci nieulotnej - 256 kB wbudowanej w system samoprogramowalnej pamięci flash - 4KB EEPROM - Wewnętrzna pamięć SRAM 8 KB - Cykle zapisu / kasowania: 10 000 Flash / 100 000 EEPROM - Podtrzymywanie danych: 20 lat w 85°C / 100 lat w 25°C - Opcjonalna sekcja kodu rozruchowego z niezależnymi bitami ustawień • Programowanie w systemie za pomocą programu rozruchowego na chipie • Prawdziwa operacja odczytu w trakcie

	zapisu ● Blokada programowania dla bezpieczeństwa oprogramowania	zapisu - Blokada programowania dla bezpieczeństwa oprogramowania • Do 64 KB opcjonalna pamięć zewnętrzna
Cechy szczególne		• Obsługa biblioteki Atmel QTouch - Pojemnościowe przyciski dotykowe, suwaki i koła - Przejęcie QTouch i QMatrix - Do 64 kanałów wykrywania
JTAG		• Interfejs JTAG (zgodny ze standardem IEEE® std. 1149.1) - Możliwości skanowania według standardu JTAG - Rozbudowane wsparcie debugowania na chipie - Programowanie pamięci Flash, EEPROM, bezpieczników i blokad za pomocą interfejsu JTAG
Funkcje peryferyjne	● Dwa 8-bitowe wyłączniki czasowe / liczniki z osobnym trybem wstępnego skalowania i porównaniem ● Jeden 16-bitowy Timer / Licznik z osobnym preskalerem, trybem porównawczym i trybem przechwytywania ● Licznik czasu rzeczywistego z oddzielnym oscylatorem ● Sześć kanałów PWM ● 8-kanałowy 10-bitowy ADC ● Pomiar temperatury ● Programowalny szeregowy USART ● Interfejs szeregowy SPI master / slave ● Zorientowany na bajty 2-przewodowy interfejs szeregowy (kompatybilny z Phillips I2C) ● Programowalny przekaźnik czasowy z oddzielnym oscylatorem na chipie ● Komparator analogowy na chipie ● Przerwanie i wybudzenie po zmianie stanu pinów	- Dwa 8-bitowe wyłączniki czasowe / liczniki z oddzielnym trybem wstępnym i trybem porównania - Cztery 16-bitowy timer / licznik z oddzielnym trybem wstępnego skalowania, trybem porównania i robienia zdjęć - Licznik czasu rzeczywistego z oddzielnym oscylatorem - Cztery 8-bitowe kanały PWM - Sześć / dwanaście kanałów PWM o programowalnej rozdzielczości od 2 do 16 bitów (ATmega1281 / 2561, ATmega640 / 1280/2560) - Modulator porównania wyników - 8/16 kanałów, 10-bit ADC (ATmega1281 / 2561, ATmega640 / 1280/2560) - Dwa / cztery programowalne szeregowe USART (ATmega1281 / 2561, ATmega640 / 1280/2560) - Interfejs szeregowy SPI Master / Slave - 2-żyłowy interfejs szeregowy zorientowany na bajty - Programowalny timer nadzoru z oddzielnym oscylatorem na chipie - Komparator analogowy na chipie - Przerwanie i wybudzenie po zmianie stanu wyprowadzenia

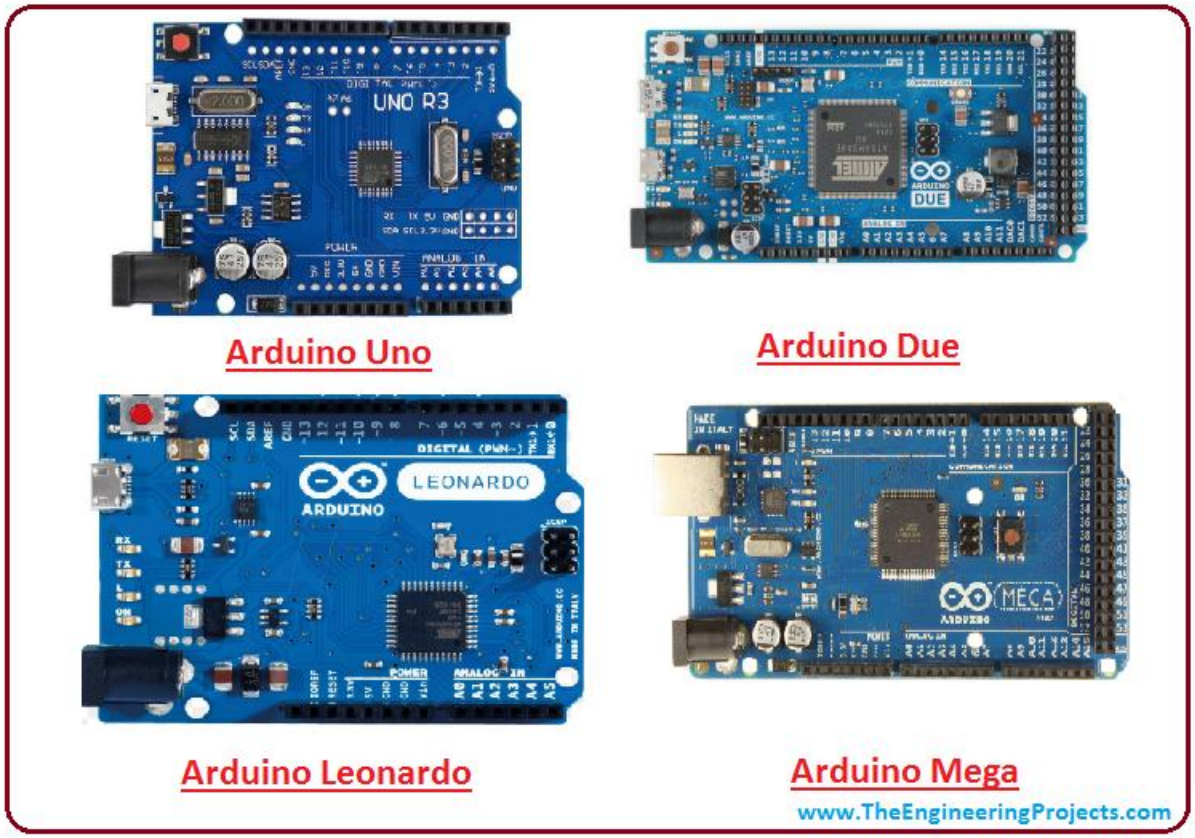
Funkcje specjalne	● Resetowanie po włączeniu i programowalne wykrywanie zaniku zasilania ● Wewnętrzny kalibrowany oscylator ● Zewnętrzne i wewnętrzne źródła przerwań ● Sześć trybów uśpienia: bezczynność, redukcja szumów ADC, oszczędzanie energii, wyłączanie, czuwanie i przedłużony tryb czuwania	- Reset po włączeniu zasilania i programowalne wykrywanie zaniku zasilania - Wewnętrzny kalibrowany oscylator - Zewnętrzne i wewnętrzne źródła przerwania - Sześć trybów uśpienia: bezczynność, redukcja szumów ADC, oszczędzanie energii, wyłączanie, tryb gotowości i dłuższy tryb gotowości
We / wy i różne obudowy	● 23 programowalne linie we / wy ● 32-odprowadzeniowy TQFP i 32-padowy QFN / MLF	- 54/86 programowalnych linii we / wy (ATmega1281 / 2561, ATmega640 / 1280/2560) - 64-pad QFN / MLF, 64-wyprowadzeniowy TQFP (ATmega1281 / 2561) - 100-wyprowadzeniowy TQFP, 100-kulowy CBGA (ATmega640 / 1280/2560) - RoHS / Fully Green
Zasilanie, pobór mocy, temperatury pracy	● Napięcie robocze: ● 2,7 V do 5,5 V dla ATmega328P ● Zakres temperatur: ● Samochodowy zakres temperatur: –40 ° C do + 125 ° C ● Klasa prędkości: ● 0 do 8 MHz przy 2,7 do 5,5 V (zakres temperatur w motoryzacji: –40 ° C do + 125 ° C) ● 0 do 16 MHz przy 4,5 do 5,5 V (zakres temperatur w motoryzacji: –40 ° C do + 125 ° C) ● Niskie zużycie energii ● Tryb aktywny: 1,5 mA przy 3 V - 4 MHz ● Tryb wyłączania: 1µA	• Zakres temperatury: - Od -40°C do 85°C Przemysł • Ultra-niskie zużycie energii - Tryb aktywny: 1 MHz, 1,8 V: 500µA - Tryb wyłączania: 0,1µA przy 1,8 V. • Klasa prędkości: - ATmega640V / ATmega1280V / ATmega1281V: • 0 - 4 MHz przy 1,8 V - 5,5 V, 0 - 8 MHz przy 2,7 V - 5,5 V. - ATmega2560V / ATmega2561V: • 0 - 2 MHz przy 1,8 V - 5,5 V, 0 - 8 MHz przy 2,7 V - 5,5 V. - ATmega640 / ATmega1280 / ATmega1281: • 0 - 8 MHz przy 2,7 V - 5,5 V, 0 - 16 MHz przy 4,5 V - 5,5 V. - ATmega2560 / ATmega2561: • 0 - 16 MHz przy 4,5 V - 5,5 V przy 3 V.
Cena:	Oficjalnie u producenta ok. 3-4 dolarów Aktualnie (2018) w sklepie ok. 17 zł	Oficjalnie u producenta ok. 3-4 dolarów Aktualnie (2018) w sklepie ok. 17 zł
WWW:	https://www.microchip.com/	https://www.microchip.com/

Porównanie mikrokontrolerów ATmega 64/128/256:

Device	Flash	EEPROM	RAM	General Purpose I/O pins	16 bits resolution PWM channels	Serial USARTs	ADC Channels
ATmega640	64KB	4KB	8KB	86	12	4	16
ATmega1280	128KB	4KB	8KB	86	12	4	16
ATmega1281	128KB	4KB	8KB	54	6	2	8
ATmega2560	256KB	4KB	8KB	86	12	4	16
ATmega2561	256KB	4KB	8KB	54	6	2	8

1.2. Opis i porównanie płytek Arduino

1.2.1. Porównanie różnych typów płytek Arduino:



1.2.2. Porównanie płytek Arduino Uno i Uno Plus

Cecha	Waveshare	Arduino Uno	Opis

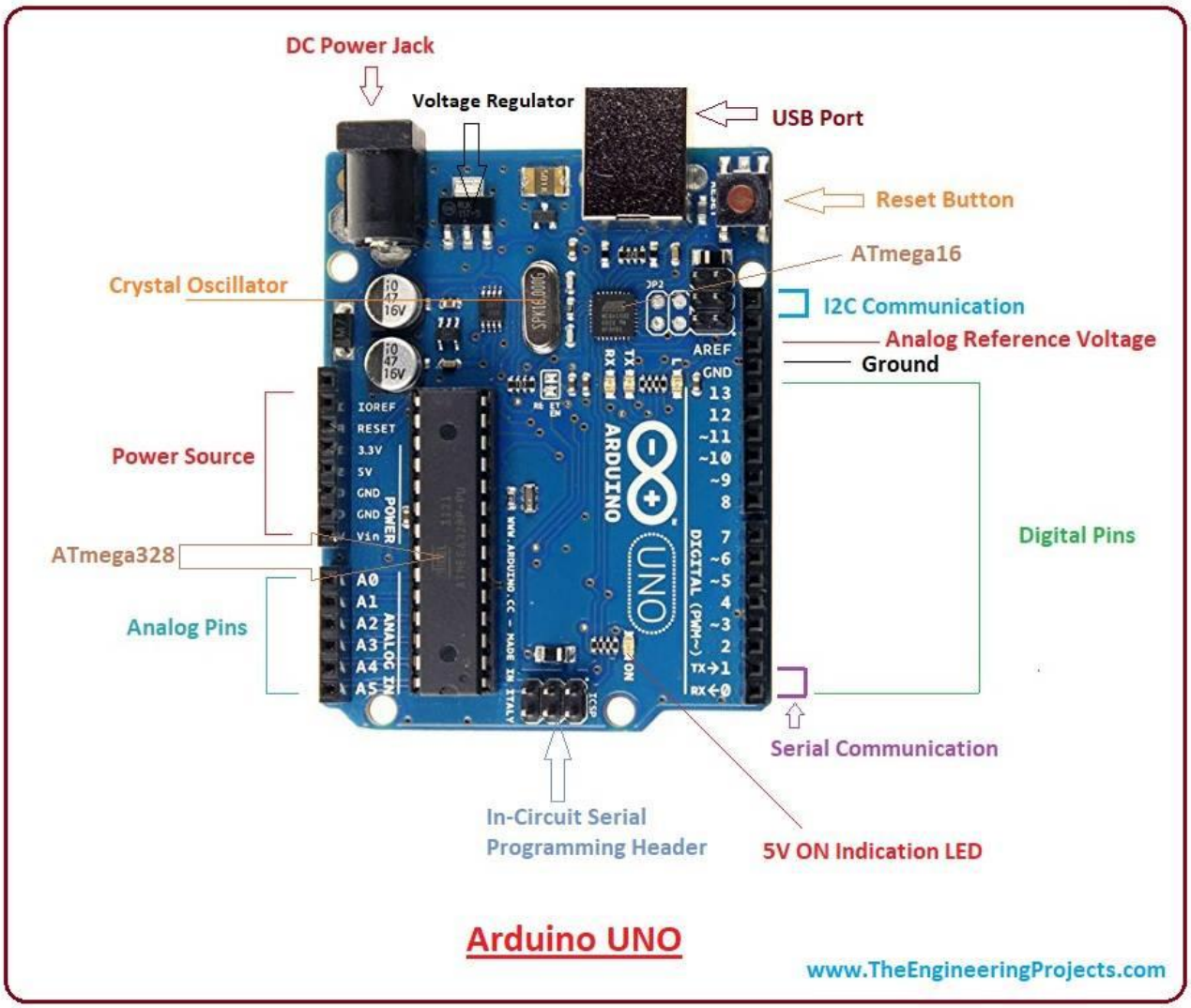
	Uno Plus		
Napięcie pracy wyprowadzeń	5 V / 3,3 V	5 V	Możliwość wyboru napięcia pracy, pozwała podłączyć urządzenia pracujące na 3,3 V. Zmiana odbywa się przy pomocy zworki.
Przycisk Reset	Naciskany z boku	Naciskany z góry	Boczny przycisk ułatwia naciskanie, gdy założony jest Shield.
Przycisk Bootloader	Tak	Nie	
Złącze USB	Micro USB	USB B	Możliwość zastosowania mniejszego, bardziej popularnego przewodu microUSB.
Wejście zasilające	DC niski profil	standardowe złącze	Niski profil gniazda nie przeszkadza w montażu nakładek.
Wyjście 3,3 V	Wydajność: 800mA	Wydajność: 150 mA	Uno plus zapewnia wyższą wydajność stabilizatora 3,3 V.
Oscylator	Kwarcowy	Ceramiczny	Rezonator kwarcowy zapewnia wyższą dokładność taktowania.
Wejścia analogowe	8	6	
Sterownik USB	FT232	ATmega	

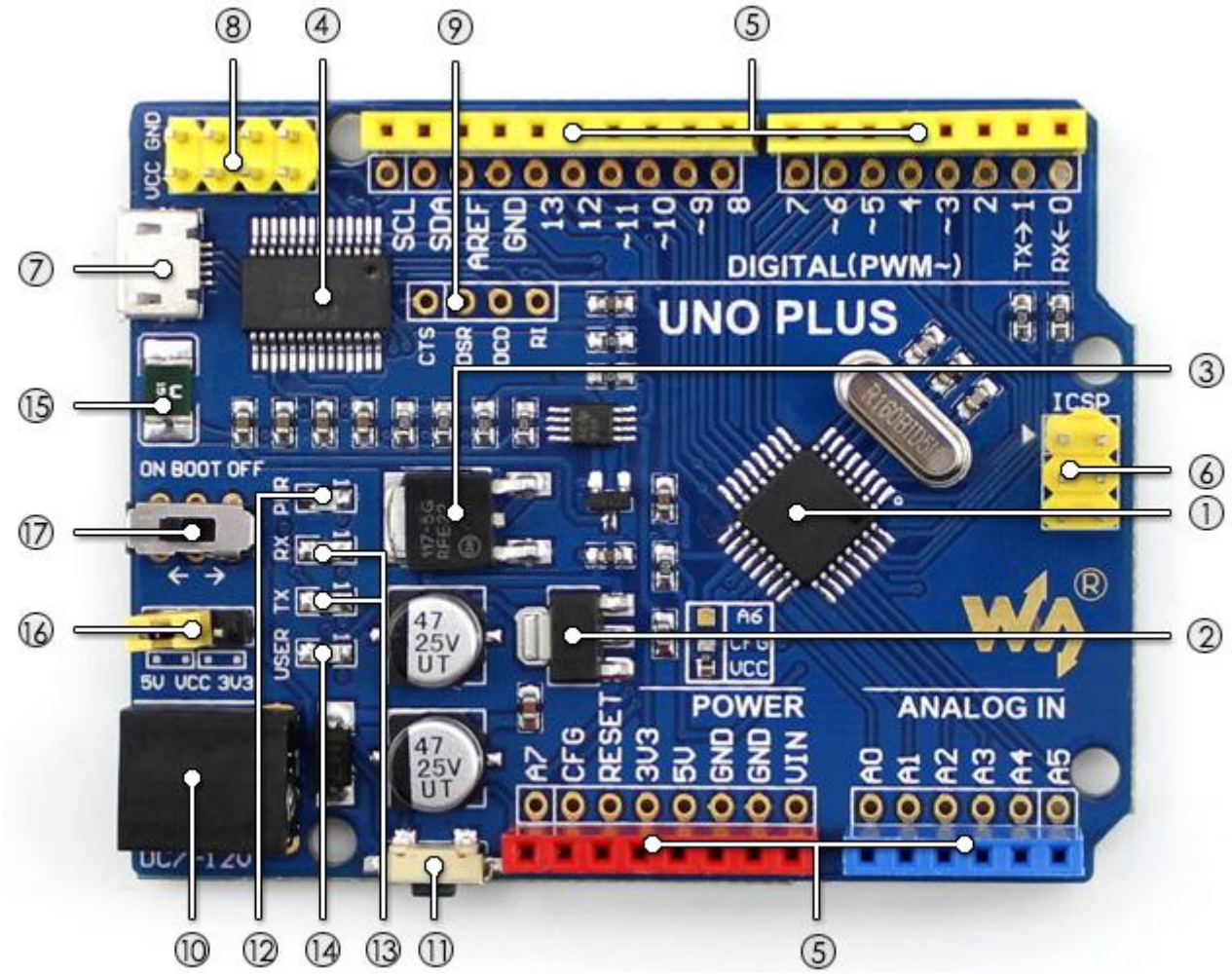
1.2.3. Porównanie płytek Arduino Uno Plus I Mega 2560

Brand	HHName (link)	HHProcessor	HHOperating Voltage	Input Voltage Range	CPU Speed	Analog In	Digital IO	PWM	EEPROM [kB]	SRAM [kB]	Flash [kB]	USB	SERIAL UART
Arduino	Mega 2560	ATmega2560	5 V	7 -12 V	16 MHz	16	54	15	4	8	256	Regular	4
Arduino	Uno	ATmega328P	5 V	7 12 V	16 MHz	6	14	6	1	2	32	Regular	1

1.3. Płytki Arduino Uno (Arduino Uno Plus są zgodne z Arduino Uno)

1.3.1. Arduino Uno





1. ATMEGA328P-AU
2. AMS1117-3.3: Regulator napięcia 3,3 V.
3. NCP1117ST50T3G: Regulator napięcia 5 V.
4. FT232RL: Konwerter USB na UART
5. Interfejs Arduino kompatybilny ze standardowym interfejsem Arduino z dwoma dodatkowymi wejściami analogowymi A6 (konfiguracja CFG), A7 dostarczone podkładki lutownicze, obsługują prototypową płytkę ścienną
6. Interfejs ICSP
7. Złącze MICRO USB: do przesyłania programu lub debugowania na porcie szeregowego
8. Listwa wyjściowa mocy: 3,3 V lub 5 V, poziom napięcia konfigurowany za pomocą wbudowanego przełącznika konfiguracji mocy, stosowany jako moc wyjściowa LUB wspólne uziemienie z innymi płytami
9. Piny FT232: do programowania Bootloadera w mikrokontrolerze
10. Wejście DC: 7 V ~ 12V
11. Przycisk reset
12. Wskaźnik mocy
13. Wskaźnik portu szeregowego Rx / Tx
14. Dioda LED użytkownika
15. Bezpiecznik szybkiego powrotu 500mA
16. Konfiguracja zasilania: do konfiguracji napięcia roboczego
17. Przełącznik wyboru bootloadera

17.1. WŁĄCZ: płyta zresetuje się, gdy zostanie wykryte włączenie zasilania LUB inne urządzenia USB łączące się z komputerem

17.2. WYŁĄCZ: program pokładowy uruchamia się natychmiast po włączeniu zasilania, a płyta nie resetuje się po wykryciu innych urządzeń USB podłączających się do komputera

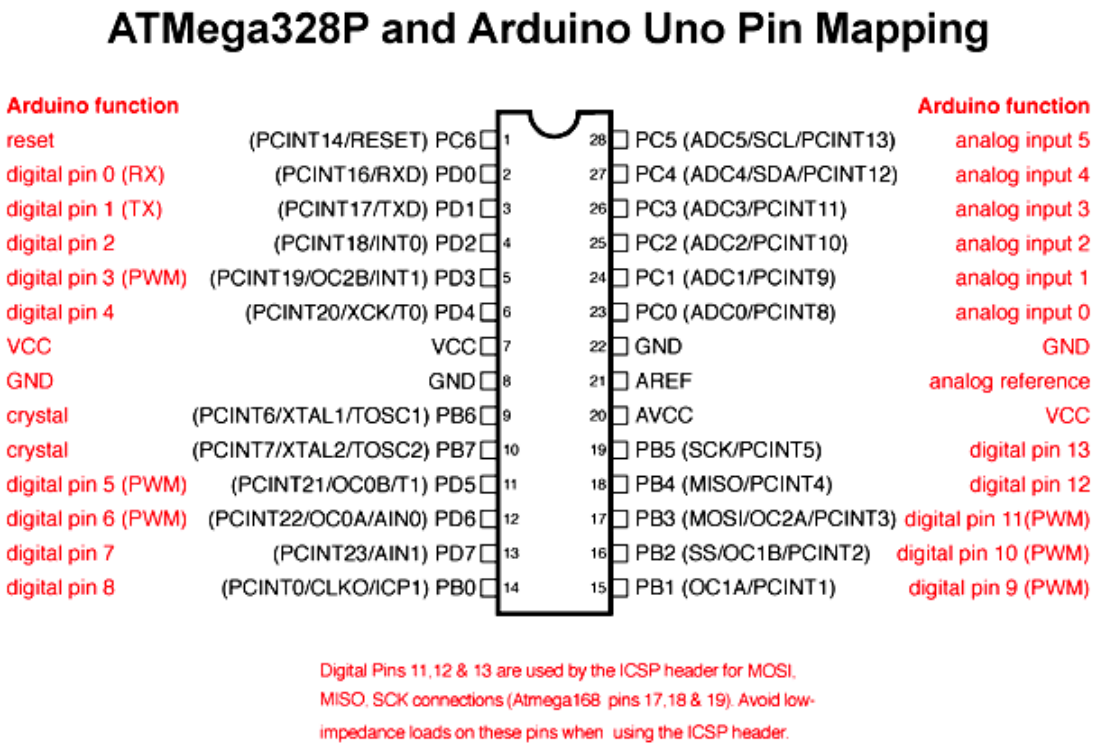
Strony producenta Waveshare:

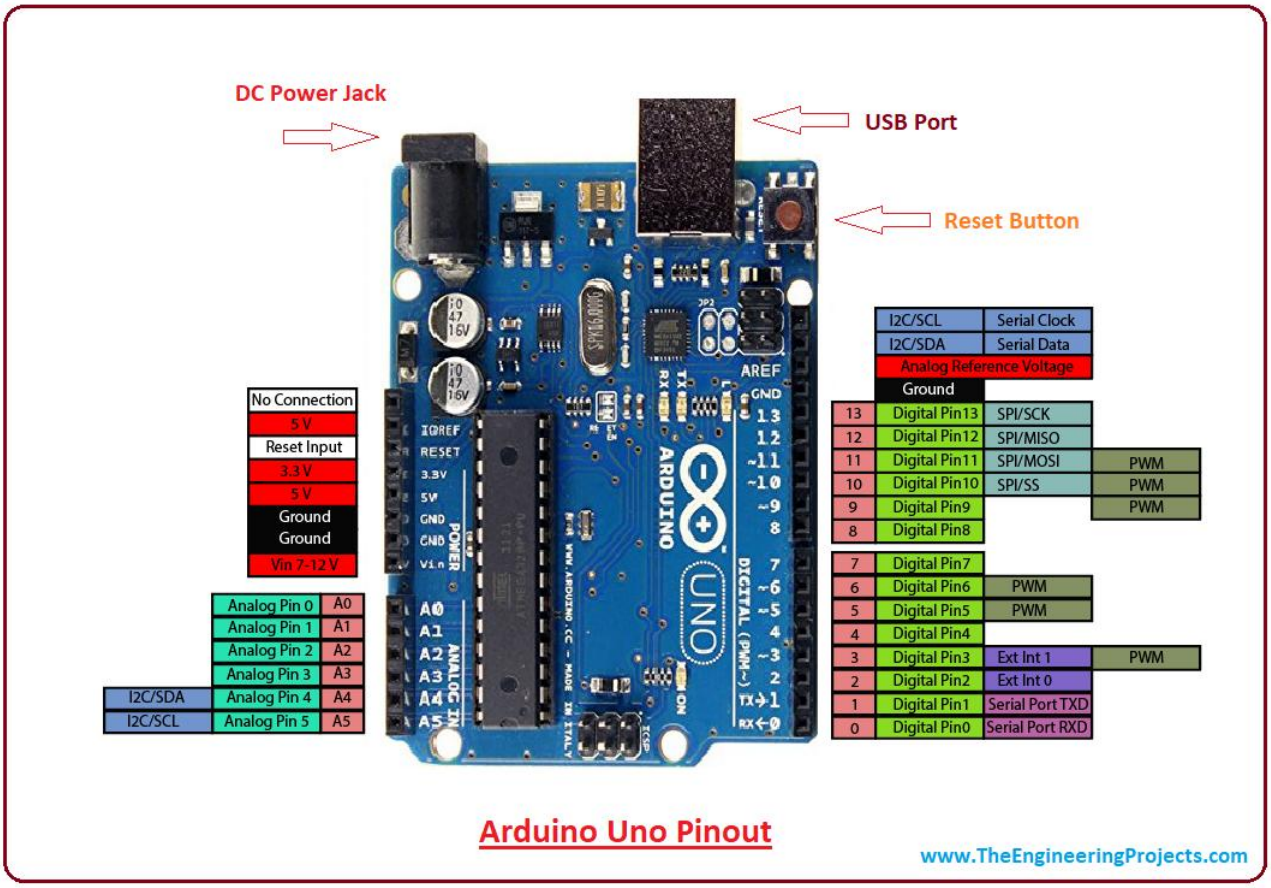
http://www.waveshare.com/UNO-PLUS.htm

http://www.waveshare.com/wiki/UNO_PLUS

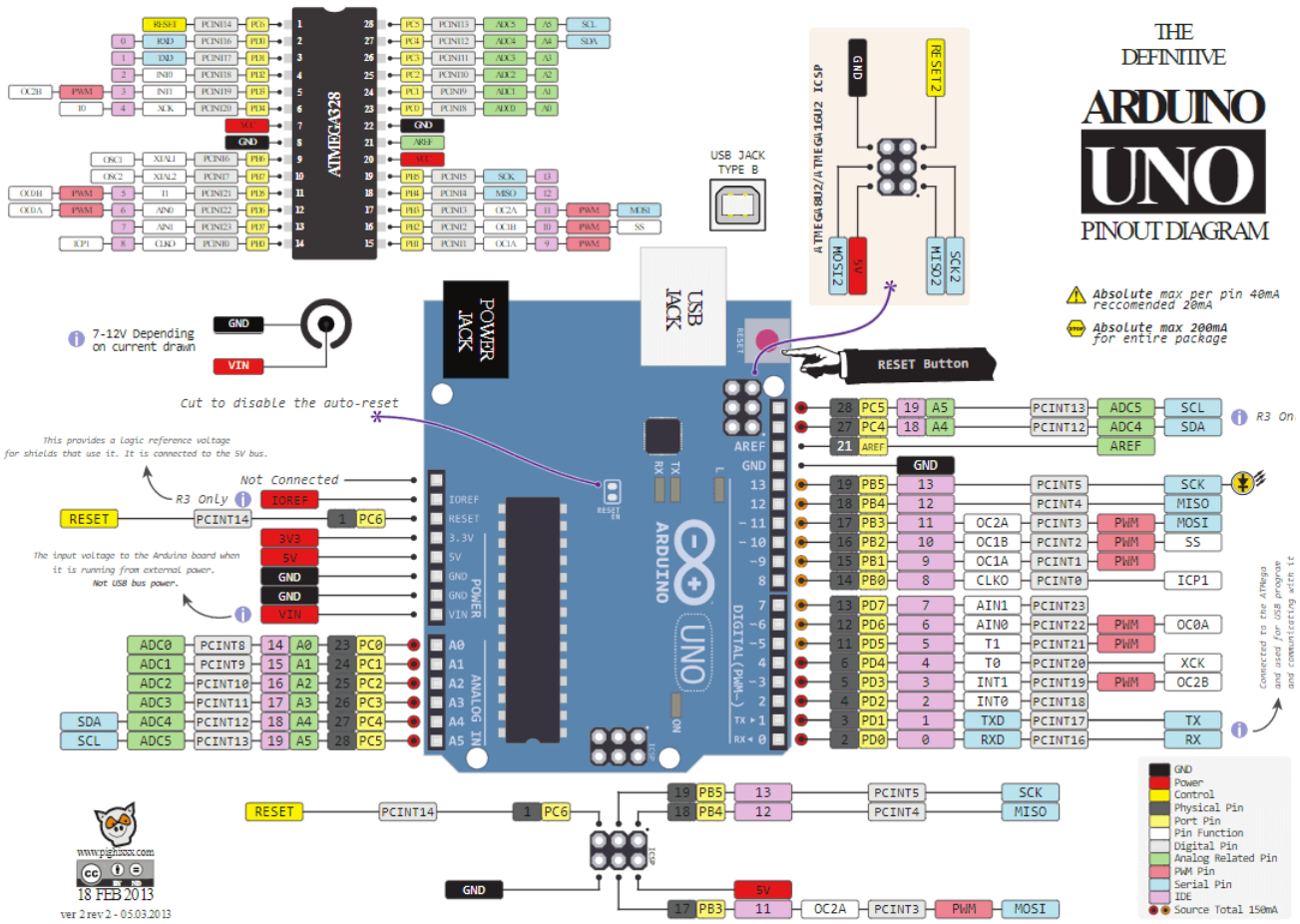
Sterowniki:

http://www.waveshare.com/wiki/UNO-PLUS_Software

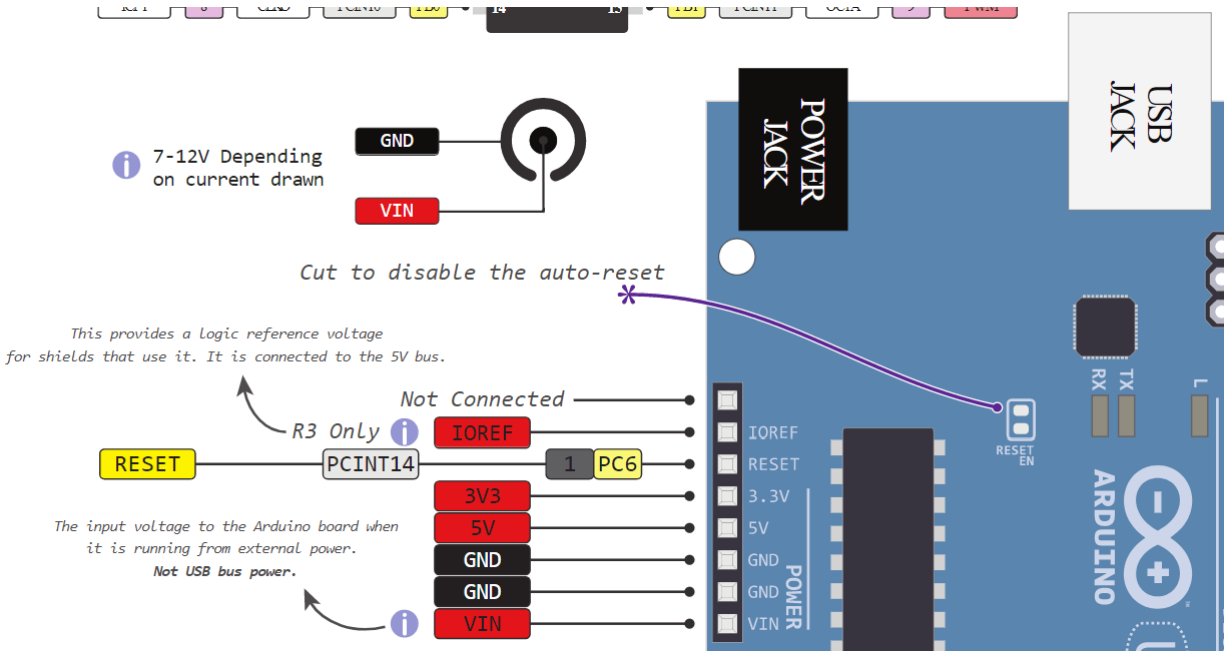




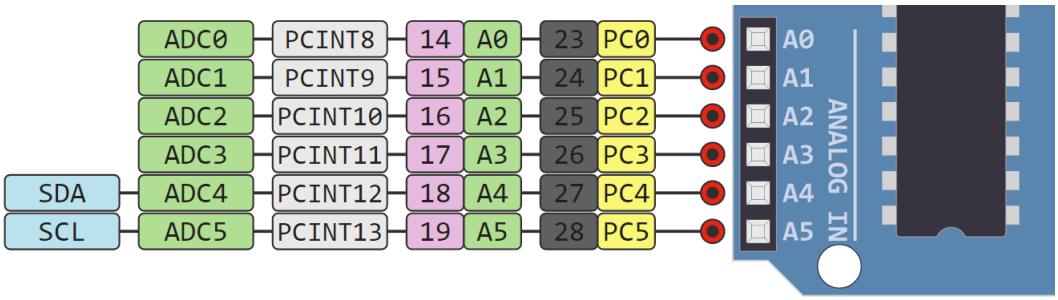
1.3.2. Wyprowadzenia Arduino



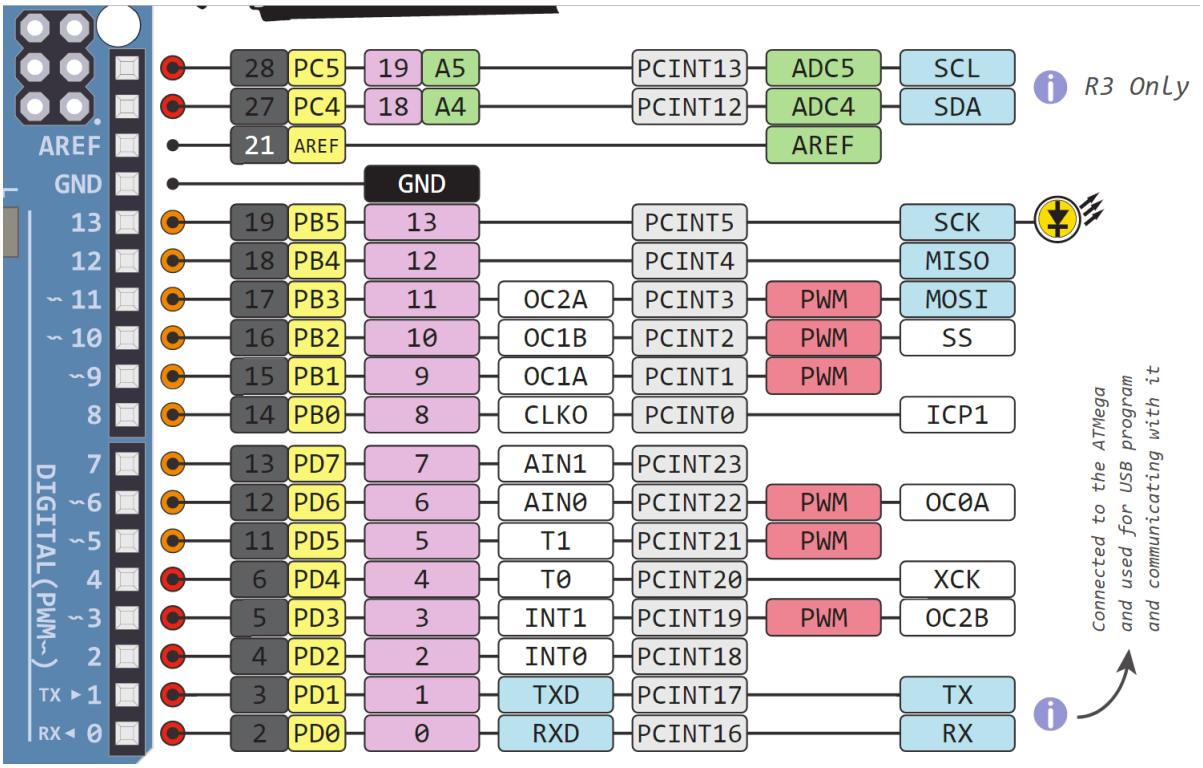
1.3.3. Wyprowadzenia zasilania



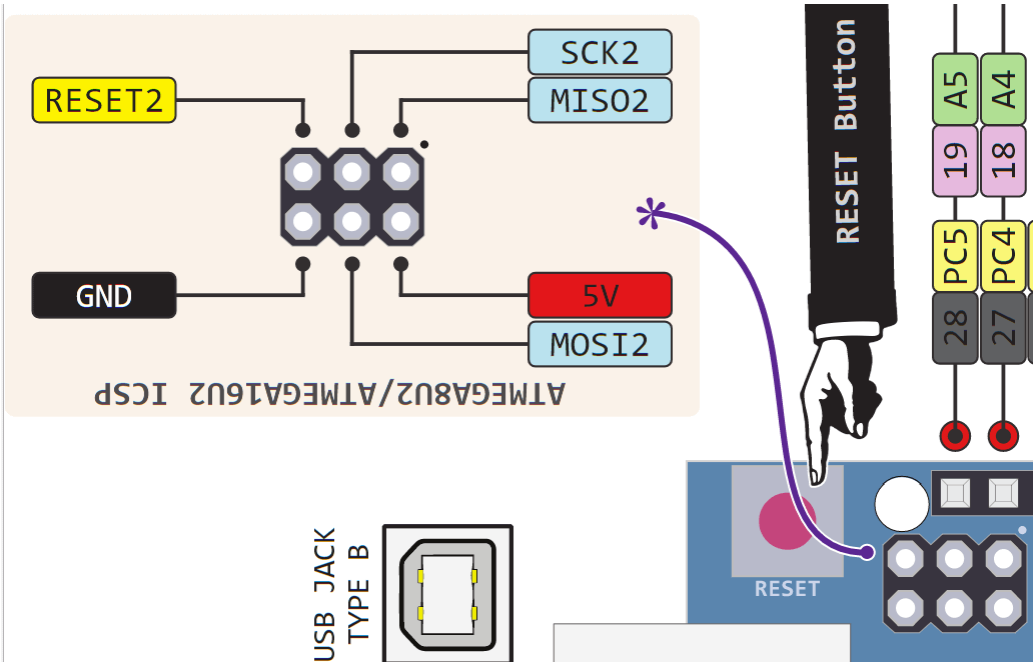
1.3.4. Wyprowadzenia analogowe



1.3.5. Wyprowadzenia cyfrowe



1.3.6. Wyprowadzenia programowania (nie JTAG)



1.3.7. Tabela mapowania wyprowadzeń ARD -> PROC

Port	Wyprowadzenie Arduino Mega	Wyprowadzenie procesora
Zasilanie	ADC6	ADC6
Zasilanie	5 V	5 V
Zasilanie	RESET	RESET
Zasilanie	3,3 V	3,3 V
Zasilanie	5 V	5 V
Zasilanie	GND	GND
Zasilanie	GND	GND
Zasilanie	VIN	VIN
Analog	A0	PC0 (ADC0/PCINT8)
Analog	A1	PC1 (ADC1/PCINT9)
Analog	A2	PC2 (ADC2/PCINT10)
Analog	A3	PC3 (ADC3/PCINT11)
Analog	A4 (SDA)	PC4 (ADC4/SDA/PCINT12)
Analog	A5 (SCL)	PC5 (ADC5/SCL/PCINT13)
Digital	SCL	PC5 (ADC5/SCL/PCINT13)
Digital	SDA	PC4 (ADC4/SDA/PCINT12)
Digital	AREF	AREF
Digital	GND	GND
Digital	D13	PB5 (SCK/PCINT5)
Digital	D12	PB4 (MISO/PCINT4)
Digital	D11	PB3 (MOSI/OC2A/PCINT3)
Digital	D10	PB2 (OC1B/nSS/PCINT2)

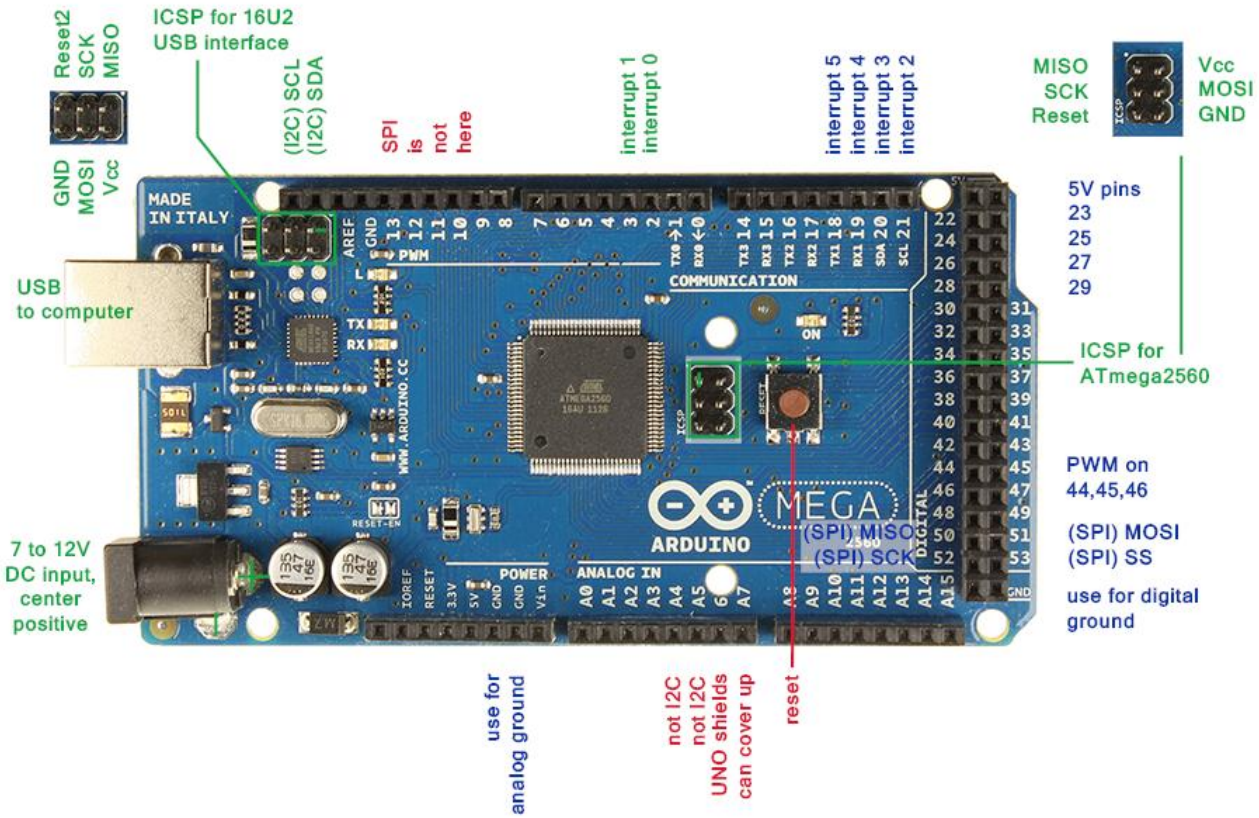
Digital	D9	PB1 (OC1A/PCINT1)
Digital	D8	PB0 (ICP1/CLKO/PCINT0)
Digital	D7	PD7 (AIN1/PCINT23)
Digital	D6	PD6 (AIN0/OC0A/PCINT22)
Digital	D5	PD5 (T1/OC0B/PCINT21)
Digital	D4	PD4 (T0/XCK/PCINT20)
Digital	D3	PD3 (INT1/OC2B/PCINT19)
Digital	D2	PD2 (INT0/PCINT18)
Digital	D1 (TX - prog)	PD1 (TXD/PCINT17 - prog)
Digital	D0 (RX - prog)	PD0 (RXD/PCINT16 - prog)

1.3.8. Tabela mapowania wyprowadzeń PROC -> ARD

Wyprowadzenie procesora	Wyprowadzenie Arduino Mega	Port
3,3 V	3,3 V	Zasilanie
5 V	5 V	Zasilanie
5 V	5 V	Zasilanie
ADC6	ADC6	Zasilanie
AREF	AREF	Digital
GND	GND	Zasilanie
GND	GND	Zasilanie
GND	GND	Digital
PB0 (ICP1/CLKO/PCINT0)	D8	Digital
PB1 (OC1A/PCINT1)	D9	Digital
PB2 (OC1B/nSS/PCINT2)	D10	Digital
PB3 (MOSI/OC2A/PCINT3)	D11	Digital
PB4 (MISO/PCINT4)	D12	Digital
PB5 (SCK/PCINT5)	D13	Digital
PC0 (ADC0/PCINT8)	A0	Analog
PC1 (ADC1/PCINT9)	A1	Analog
PC2 (ADC2/PCINT10)	A2	Analog
PC3 (ADC3/PCINT11)	A3	Analog
PC4 (ADC4/SDA/PCINT12)	A4 (SDA)	Analog
PC4 (ADC4/SDA/PCINT12)	SDA	Digital
PC5 (ADC5/SCL/PCINT13)	A5 (SCL)	Analog
PC5 (ADC5/SCL/PCINT13)	SCL	Digital
PD0 (RXD/PCINT16 - prog)	D0 (RX - prog)	Digital
PD1 (TXD/PCINT17 - prog)	D1 (TX - prog)	Digital
PD2 (INT0/PCINT18)	D2	Digital
PD3 (INT1/OC2B/PCINT19)	D3	Digital

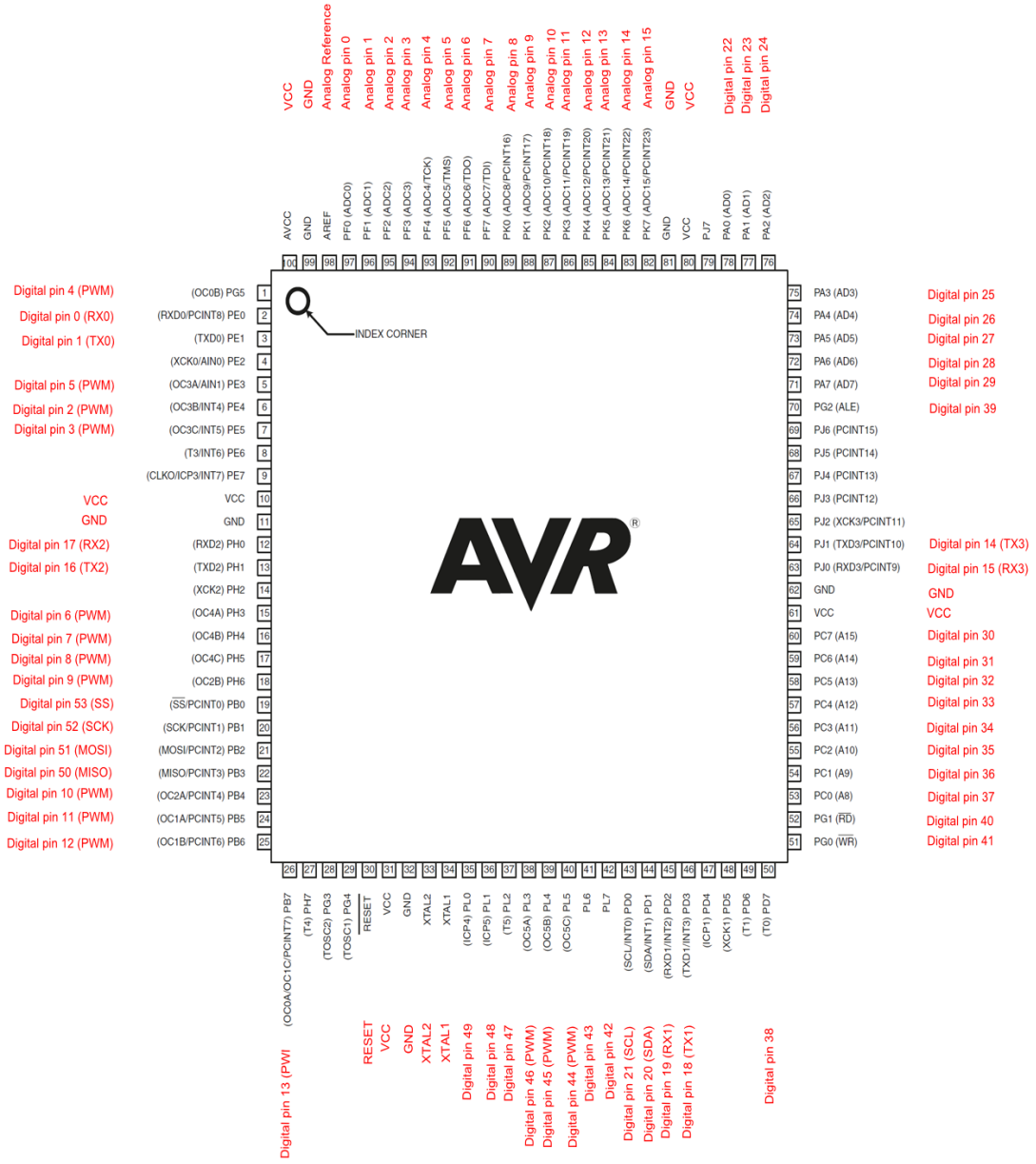
PD4 (T0/XCK/PCINT20)	D4	Digital
PD5 (T1/OC0B/PCINT21)	D5	Digital
PD6 (AIN0/OC0A/PCINT22)	D6	Digital
PD7 (AIN1/PCINT23)	D7	Digital
RESET	RESET	Zasilanie
VIN	VIN	Zasilanie

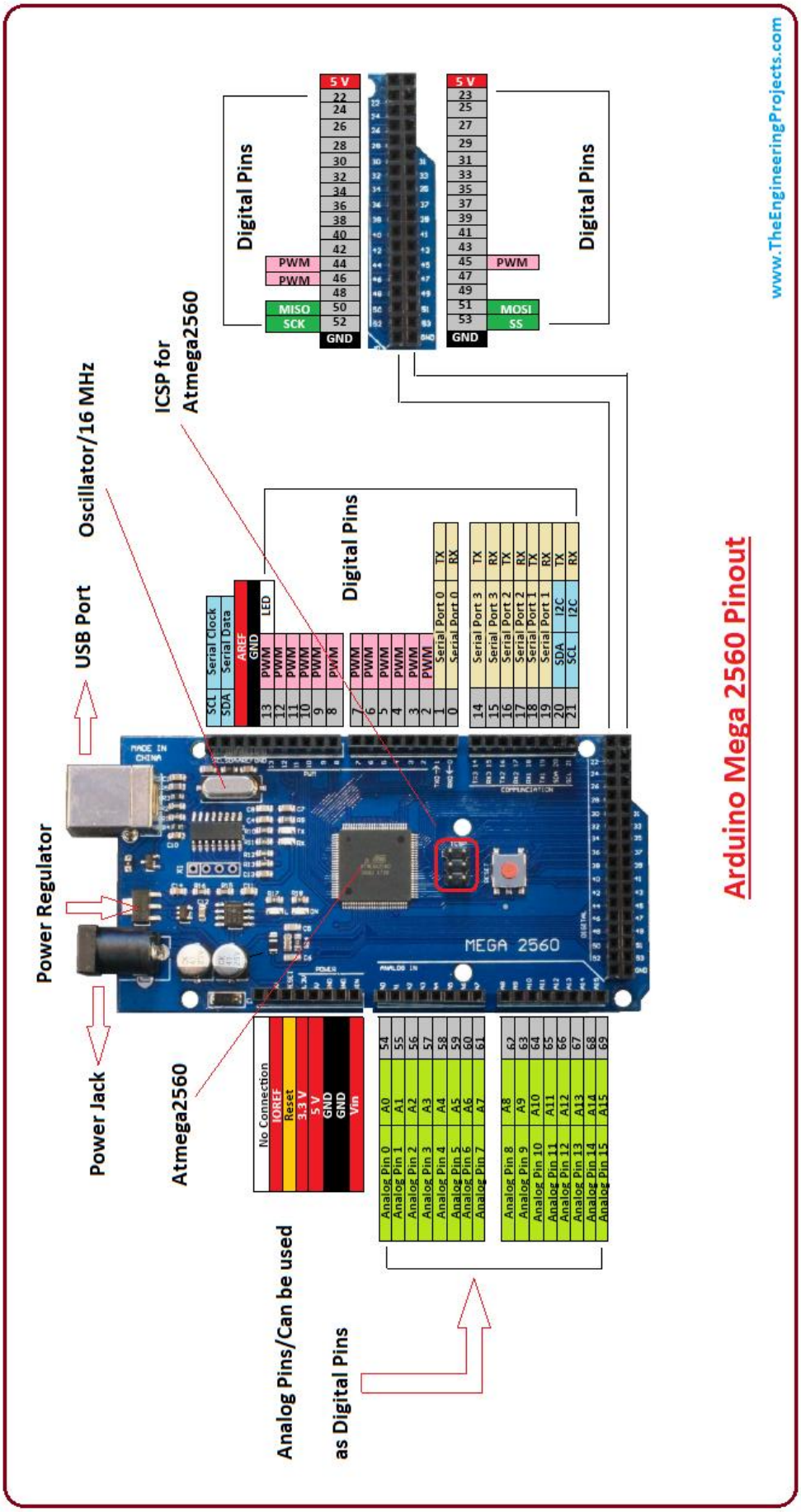
1.4. Płytki Arduino Mega 2560



Strona producenta:

http://arduino.cc/en/Main/ArduinoBoardMega




Arduino Mega 2560 Pinout
www.TheEngineeringProjects.com
Digital Pins
Digital Pins
Digital Pins
ICSP for Atmega2560
Oscillator/16 MHz
USB Port
Power Regulator
Power Jack
Atmega2560
Analog Pins/Can be used as Digital Pins
MEGA 2560
5 V
GND
PWM
MISO
SCK
MOSI
SS
Serial Clock
Serial Data
AREF
LED
Serial Port 0
Serial Port 1
Serial Port 2
Serial Port 3
I2C
No Connection
IOREF
Reset
3.3 V
Vin
Analog Pin 0
Analog Pin 1
Analog Pin 2
Analog Pin 3
Analog Pin 4
Analog Pin 5
Analog Pin 6
Analog Pin 7
Analog Pin 8
Analog Pin 9
Analog Pin 10
Analog Pin 11
Analog Pin 12
Analog Pin 13
Analog Pin 14
Analog Pin 15

1.4.1. Tabela mapowania wyprowadzeń PIN -> PROC

Pin Number	Pin Name	Mapped Pin Name
1	PG5 (OC0B)	Digital pin 4 (PWM)
2	PE0 (RXD0/PCINT8)	Digital pin 0 (RX0)
3	PE1 (TXD0)	Digital pin 1 (TX0)
4	PE2 (XCK0/AIN0)	
5	PE3 (OC3A/AIN1)	Digital pin 5 (PWM)
6	PE4 (OC3B/INT4)	Digital pin 2 (PWM)
7	PE5 (OC3C/INT5)	Digital pin 3 (PWM)
8	PE6 (T3/INT6)	
9	PE7 (CLKO/ICP3/INT7)	
10	VCC	VCC
11	GND	GND
12	PH0 (RXD2)	Digital pin 17 (RX2)
13	PH1 (TXD2)	Digital pin 16 (TX2)
14	PH2 (XCK2)	
15	PH3 (OC4A)	Digital pin 6 (PWM)
16	PH4 (OC4B)	Digital pin 7 (PWM)
17	PH5 (OC4C)	Digital pin 8 (PWM)
18	PH6 (OC2B)	Digital pin 9 (PWM)
19	PB0 (SS/PCINT0)	Digital pin 53 (SS)
20	PB1 (SCK/PCINT1)	Digital pin 52 (SCK)
21	PB2 (MOSI/PCINT2)	Digital pin 51 (MOSI)
22	PB3 (MISO/PCINT3)	Digital pin 50 (MISO)
23	PB4 (OC2A/PCINT4)	Digital pin 10 (PWM)
24	PB5 (OC1A/PCINT5)	Digital pin 11 (PWM)
25	PB6 (OC1B/PCINT6)	Digital pin 12 (PWM)
26	PB7 (OC0A/OC1C/PCINT7)	Digital pin 13 (PWM)
27	PH7 (T4)	
28	PG3 (TOSC2)	
29	PG4 (TOSC1)	
30	RESET	RESET
31	VCC	VCC
32	GND	GND
33	XTAL2	XTAL2
34	XTAL1	XTAL1
35	PL0 (ICP4)	Digital pin 49
36	PL1 (ICP5)	Digital pin 48
37	PL2 (T5)	Digital pin 47
38	PL3 (OC5A)	Digital pin 46 (PWM)
39	PL4 (OC5B)	Digital pin 45 (PWM)
40	PL5 (OC5C)	Digital pin 44 (PWM)
41	PL6	Digital pin 43
42	PL7	Digital pin 42

43	PD0 (SCL/INT0)	Digital pin 21 (SCL)
44	PD1 (SDA/INT1)	Digital pin 20 (SDA)
45	PD2 (RXDI/INT2)	Digital pin 19 (RX1)
46	PD3 (TXD1/INT3)	Digital pin 18 (TX1)
47	PD4 (ICP1)	
48	PD5 (XCK1)	
49	PD6 (T1)	
50	PD7 (T0)	Digital pin 38
51	PG0 (WR)	Digital pin 41
52	PG1 (RD)	Digital pin 40
53	PC0 (A8)	Digital pin 37
54	PC1 (A9)	Digital pin 36
55	PC2 (A10)	Digital pin 35
56	PC3 (A11)	Digital pin 34
57	PC4 (A12)	Digital pin 33
58	PC5 (A13)	Digital pin 32
59	PC6 (A14)	Digital pin 31
60	PC7 (A15)	Digital pin 30
61	VCC	VCC
62	GND	GND
63	PJ0 (RXD3/PCINT9)	Digital pin 15 (RX3)
64	PJ1 (TXD3/PCINT10)	Digital pin 14 (TX3)
65	PJ2 (XCK3/PCINT11)	
66	PJ3 (PCINT12)	
67	PJ4 (PCINT13)	
68	PJ5 (PCINT14)	
69	PJ6 (PCINT 15)	
70	PG2 (ALE)	Digital pin 39
71	PA7 (AD7)	Digital pin 29
72	PA6 (AD6)	Digital pin 28
73	PA5 (AD5)	Digital pin 27
74	PA4 (AD4)	Digital pin 26
75	PA3 (AD3)	Digital pin 25
76	PA2 (AD2)	Digital pin 24
77	PA1 (AD1)	Digital pin 23
78	PA0 (AD0)	Digital pin 22
79	PJ7	
80	VCC	VCC
81	GND	GND
82	PK7 (ADC15/PCINT23)	Analog pin 15
83	PK6 (ADC14/PCINT22)	Analog pin 14
84	PK5 (ADC13/PCINT21)	Analog pin 13
85	PK4 (ADC12/PCINT20)	Analog pin 12
86	PK3 (ADC11/PCINT19)	Analog pin 11
87	PK2 (ADC10/PCINT18)	Analog pin 10
88	PK1 (ADC9/PCINT17)	Analog pin 9
89	PK0 (ADC8/PCINT16)	Analog pin 8

90	PF7 (ADC7/TDI)	Analog pin 7
91	PF6 (ADC6/TDO)	Analog pin 6
92	PF5 (ADC5/TMS)	Analog pin 5
93	PF4 (ADC4/TCK)	Analog pin 4
94	PF3 (ADC3)	Analog pin 3
95	PF2 (ADC2)	Analog pin 2
96	PF1 (ADC1)	Analog pin 1
97	PF0 (ADC0)	Analog pin 0
98	AREF	Analog Reference
99	GND	GND
100	AVCC	VCC

1.4.2. Tabela mapowania wyprowadzeń ARD -> PROC

Port	Wyprowadzenie Arduino Mega	Wyprowadzenie procesora
Analog ADCL	ADC0	PF0 (ADC0)
Analog ADCL	ADC1	PF1 (ADC1)
Analog ADCL	ADC2	PF2 (ADC2)
Analog ADCL	ADC3	PF3 (ADC3)
Analog ADCL	ADC4	PF4 (ADC4/TCK - JTAG)
Analog ADCL	ADC5	PF5 (ADC5/TMS - JTAG)
Analog ADCL	ADC6	PF6 (ADC6/TDO – JTAG)
Analog ADCL	ADC7	PF7 (ADC7/TDI - JTAG)
Analog ADCH	ADC8	PK0 (ADC8/PCINT16)
Analog ADCH	ADC9	PK1 (ADC9/PCINT17)
Analog ADCH	ADC10	PK2 (ADC10/PCINT18)
Analog ADCH	ADC11	PK3 (ADC11/PCINT19)
Analog ADCH	ADC12	PK4 (ADC12/PCINT20)
Analog ADCH	ADC13	PK5 (ADC13/PCINT21)
Analog ADCH	ADC14	PK6 (ADC14/PCINT22)
Analog ADCH	ADC15	PK7 (ADC15/PCINT23)
PWML	PWM 0 - prog	PE0 (RX0 - prog)
PWML	PWM 1- prog	PE1 (TX0 - prog)
PWML	PWM 2	PE4 (OC3B/INT4)
PWML	PWM 3	PE5 (OC3C/INT5)
PWML	PWM 4	PG5 (OC0B)
PWML	PWM 5	PE3 (OC3A/AIN1)
PWML	PWM 6	PH3 (OC4A)
PWML	PWM 7	PH4 (OC4B)
PWMH	PWM 8	PH5 (OC4C)
PWMH	PWM 9	PH6 (OC2B)
PWMH	PWM 10	PB4 (OC2A/PCINT4)
PWMH	PWM 11	PB5 (OC1A/PCINT5)
PWMH	PWM 12	PB6 (OC1B/PCINT6)

PWMH	PWM 13	PB7 (OC0A/OC1C/PCINT7)
PWMH	GND	GND
PWMH	AREF	AREF
PWMH	SDA	PD1 (SDA/INT1)
PWMH	SCL	PD0 (SCL/INT0)
Communication	TXD3 14	PD7 (T0)
Communication	RXD3 15	PD6 (T1)
Communication	TXD2 16	PD3 (XCK1)
Communication	RXD2 17	PD4 (ICP1)
Communication	TXD1 18	PD3 (TXD1/INT3)
Communication	RXD1 19	PD2 (RXD1/INT2)
Communication	SDA 20	PD1 (SDA/INT1)
Communication	SCL 21	PD0 (SCL/INT0)
Digital	D22	PA0 (AD0)
Digital	D23	PA1 (AD1)
Digital	D24	PA2 (AD2)
Digital	D25	PA3 (AD3)
Digital	D26	PA4 (AD4)
Digital	D27	PA5 (AD5)
Digital	D28	PA6 (AD6)
Digital	D29	PA7 (AD7)
Digital	D30	PC7 (A15)
Digital	D31	PC6 (A14)
Digital	D32	PC5 (A13)
Digital	D33	PC4 (A12)
Digital	D34	PC3 (A11)
Digital	D35	PC2 (A10)
Digital	D36	PC1 (A9)
Digital	D37	PC0 (A8)
Digital	D38	PD7 (T0)
Digital	D39	PG2 (ALE)
Digital	D40	PG1 (nRD)
Digital	D41	PG0 (nWR)
Digital	D42	PL7
Digital	D43	PL6
Digital	D44	PL5 (OC5C)
Digital	D45	PL4 (OC5B)
Digital	D46	PL3 (OC5A)
Digital	D47	PL2 (T5)
Digital	D48	PL1 (ICP1)
Digital	D49	PL0 (ICP4)
Digital	D50 (MISO)	PB3 (MISO/PCINT3)
Digital	D51 (MOSI)	PB2 (MOSI/PCINT2)
Digital	D52 (SCK)	PB1 (SCK/PCINT1)
Digital	D53 (SS)	PB0 (SS/PCINT0)
Power	PWR 1	NC
Power	PWR 2	5 V

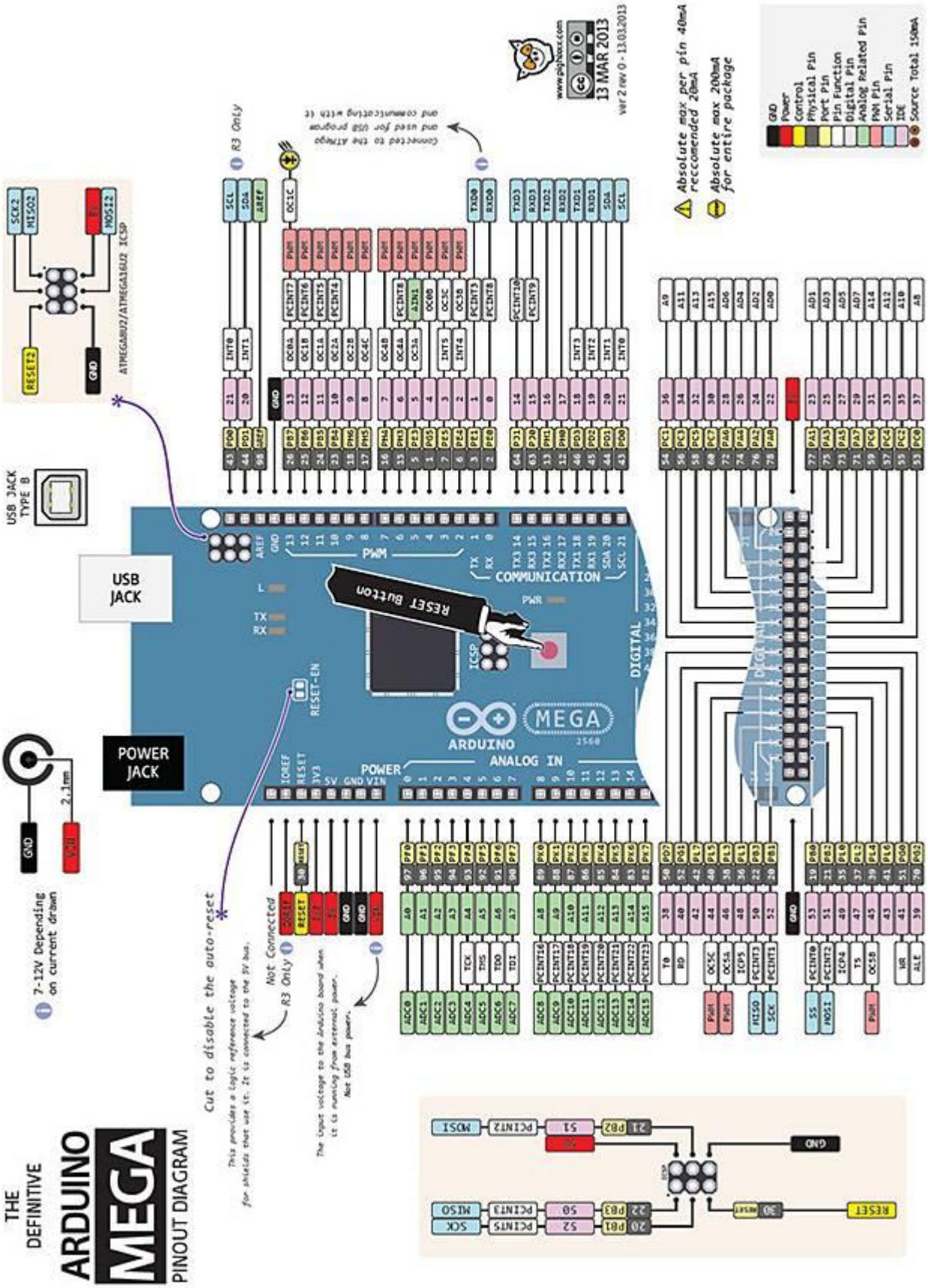
Power	PWR 3	RESET
Power	PWR 4	3,3 V
Power	PWR 5	5 V
Power	PWR 6	GND
Power	PWR 7	GND
Power	PWR 8	VIN

1.4.3. Tabela mapowania wyprowadzeń PROC -> ARD

Wyprowadzenie procesora	Wyprowadzenie Arduino Mega	Port
3,3 V	PWR 4	Power
5 V	PWR 2	Power
5 V	PWR 5	Power
AREF	AREF	PWMH
GND	GND	PWMH
GND	PWR 6	Power
GND	PWR 7	Power
NC	PWR 1	Power
PA0 (AD0)	D22	Digital
PA1 (AD1)	D23	Digital
PA2 (AD2)	D24	Digital
PA3 (AD3)	D25	Digital
PA4 (AD4)	D26	Digital
PA5 (AD5)	D27	Digital
PA6 (AD6)	D28	Digital
PA7 (AD7)	D29	Digital
PB0 (SS/PCINT0)	D53 (SS)	Digital
PB1 (SCK/PCINT1)	D52 (SCK)	Digital
PB2 (MOSI/PCINT2)	D51 (MOSI)	Digital
PB3 (MISO/PCINT3)	D50 (MISO)	Digital
PB4 (OC2A/PCINT4)	PWM 10	PWMH
PB5 (OC1A/PCINT5)	PWM 11	PWMH
PB6 (OC1B/PCINT6)	PWM 12	PWMH
PB7 (OC0A/OC1C/PCINT7)	PWM 13	PWMH
PC0 (A8)	D37	Digital
PC1 (A9)	D36	Digital
PC2 (A10)	D35	Digital
PC3 (A11)	D34	Digital
PC4 (A12)	D33	Digital
PC5 (A13)	D32	Digital

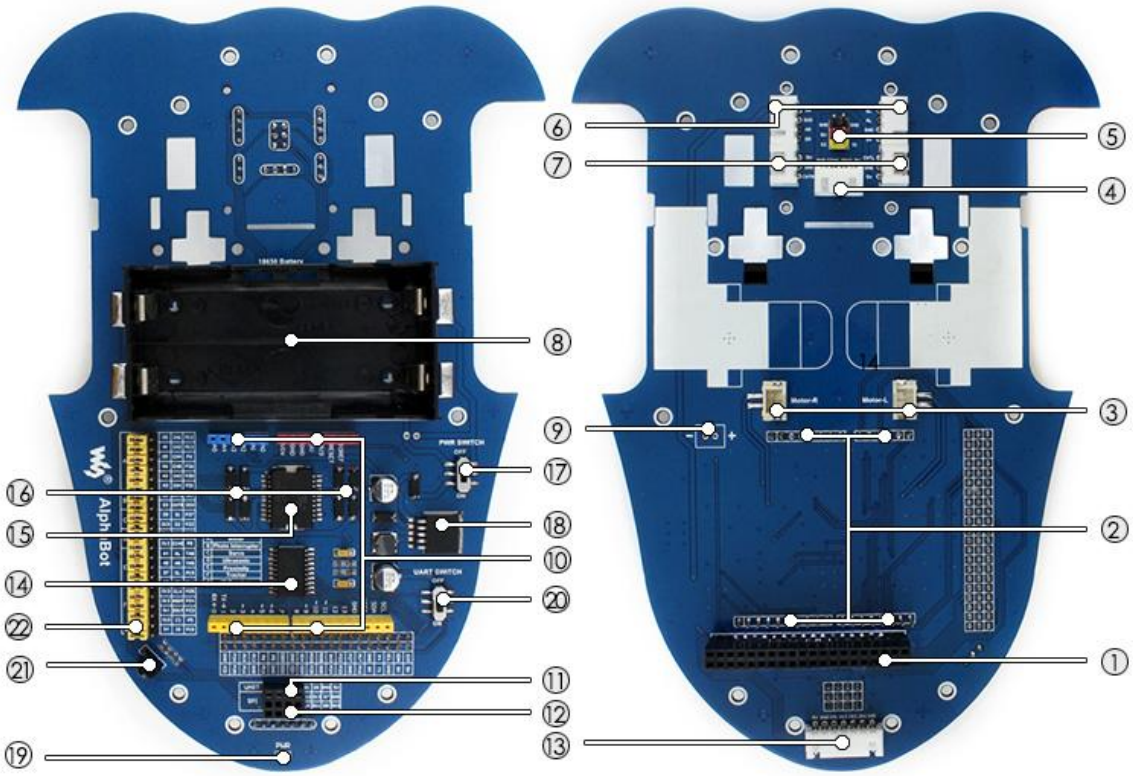
PC6 (A14)	D31	Digital
PC7 (A15)	D30	Digital
PD0 (SCL/INT0)	SCL	PWMH
PD0 (SCL/INT0)	SCL 21	Communication
PD1 (SDA/INT1)	SDA	PWMH
PD1 (SDA/INT1)	SDA 20	Communication
PD2 (RXD1/INT2)	RXD1 19	Communication
PD3 (TXD1/INT3)	TXD1 18	Communication
PD3 (XCK1)	TXD2 16	Communication
PD4 (ICP1)	RXD2 17	Communication
PD6 (T1)	RXD3 15	Communication
PD7 (T0)	TXD3 14	Communication
PD7 (T0)	D38	Digital
PE0 (RX0 - prog)	PWM 0 - prog	PWML
PE1 (TX0 - prog)	PWM 1- prog	PWML
PE3 (OC3A/AIN1)	PWM 5	PWML
PE4 (OC3B/INT4)	PWM 2	PWML
PE5 (OC3C/INT5)	PWM 3	PWML
PF0 (ADC0)	ADC0	Analog ADCL
PF1 (ADC1)	ADC1	Analog ADCL
PF2 (ADC2)	ADC2	Analog ADCL
PF3 (ADC3)	ADC3	Analog ADCL
PF4 (ADC4/TCK - JTAG)	ADC4	Analog ADCL
PF5 (ADC5/TMS - JTAG)	ADC5	Analog ADCL
PF6 (ADC6/TDO - JTAG)	ADC6	Analog ADCL
PF7 (ADC7/TDI - JTAG)	ADC7	Analog ADCL
PG0 (nWR)	D41	Digital
PG1 (nRD)	D40	Digital
PG2 (ALE)	D39	Digital
PG5 (OC0B)	PWM 4	PWML
PH3 (OC4A)	PWM 6	PWML
PH4 (OC4B)	PWM 7	PWML
PH5 (OC4C)	PWM 8	PWMH
PH6 (OC2B)	PWM 9	PWMH
PK0 (ADC8/PCINT16)	ADC8	Analog ADCH
PK1 (ADC9/PCINT17)	ADC9	Analog ADCH
PK2 (ADC10/PCINT18)	ADC10	Analog ADCH
PK3 (ADC11/PCINT19)	ADC11	Analog ADCH
PK4 (ADC12/PCINT20)	ADC12	Analog ADCH
PK5 (ADC13/PCINT21)	ADC13	Analog ADCH
PK6 (ADC14/PCINT22)	ADC14	Analog ADCH

PK7 (ADC15/PCINT23)	ADC15	Analog ADCH
PL0 (ICP4)	D49	Digital
PL1 (ICP1)	D48	Digital
PL2 (T5)	D47	Digital
PL3 (OC5A)	D46	Digital
PL4 (OC5B)	D45	Digital
PL5 (OC5C)	D44	Digital
PL6	D43	Digital
PL7	D42	Digital
RESET	PWR 3	Power
VIN	PWR 8	Power


THE DEFINITIVE
ARDUINO MEGA
PINOUT DIAGRAM
USB JACK
POWER JACK
USB JACK TYPE B
RESET Button
ARDUINO
MEGA
2560
PWM
COMMUNICATION
ANALOG IN
POWER
DIGITAL
ICSP
RESET-EN
7-12V Depending on current drawn
Cut to disable the auto-reset
Not Connected
R3 Only
13 MAR 2013
www.pighixxx.com
Absolute max per pin 40mA reccomended 20mA
Absolute max 200mA for entire package
GND
Power
Control
Physical Pin
Port Pin
Pin Function
Digital Pin
Analog Related Pin
PWM Pin
Serial Pin
IDE
Source Total 150mA

1.5. Płyta AlphaBot

1.5.1. AlphaBot – moduł główny



1. **Interfejs Raspberry Pi:** do połączenia z Raspberry
2. **Interfejs Arduino:** do połączenia z Arduino
3. Złącze po podłączenia **silników**
4. Złącze do podłączenia **ultradźwiękowego czujnika odległości**
5. Złącze do podłączenia **serwomechanizmu**
6. Złącze do podłączenia **modułu unikania przeszkód**
7. Złącze do podłączenia **miernika prędkości**
8. **Koszyk na baterie 18650**
9. **Miejsce do podłączenia zewnętrznego źródła zasilania**
10. **Złącza Arduino:** do podłączenia nakładek dla Arduino
11. **Interfejs UART:** do podłączenia modułu Bluetooth
12. **Interfejs SPI:** do podłączenia modułu NFR24L01
13. Złącze do podłączenia **czujnika linii**
14. **Przetwornik TLC1543:** 10-bitowy, pozwala na wykorzystanie analogowych modułów z Raspberry Pi
15. **Mostek LM298P:** do sterowania silnikami, do 2 A
16. **Dioda zapobiegająca prądowi wstecznemu**
17. **Włącznik zasilania**
18. **LM2596:** Regulator napięcia 5 V

19. **Dioda sygnalizująca o zasilaniu**
20. **Przełącznik UART:** włącza komunikację między Arduino i Raspberry
21. **Odbiornik IR:** pozwala na sterowanie robotem przez podczerwień
22. **Wybór Arduino / Raspberry:** wybiera, które urządzenie kontroluje peryferia robota

Strony AlphaBot:

http://www.waveshare.com/alphabot-ar-bluetooth.htm

https://www.waveshare.com/wiki/AlphaBot

1.5.2. Tabela mapy wyprowadzeń AlphaBot

	Wyprowadzenie na module	Wyprowadzenie na gnieździe Arduino	UWAGI
Moduł szybkości - impulsator	CNTL	D2	
	CNTR	D3	
Moduł ultradźwiękowy wykrywania przeszkód	Trig	D11	D11 wspólne dla modułu Bluetooth i śledzenia linii
	Echo	D12	D12 wspólne dla modułu Bluetooth i śledzenia linii
Moduł wykrywania przeszkód w podczerwieni	AL	A4	NIE PODŁACZAĆ DO A4 – konflikt z JTAG, wykorzystać inne wyprowadzenie Arduino
	AR	A5	NIE PODŁACZAĆ DO A5 – konflikt z JTAG, wykorzystać inne wyprowadzenie Arduino
	DL	D7	D7 wspólne dla modułu Bluetooth
	DR	D8	D8 wspólne dla modułu Bluetooth
Serwomechanizm	S1	D9	
	S2	D10	D10 wspólne dla modułu śledzenia linii
Moduł odbiornika podczerwieni	IRM	D4	D4 wspólne dla modułu Bluetooth
Moduł śledzenia linii	IOCLK	D13	D13 wspólne dla

			modułu Bluetooth
	ADDR	D12	D12 wspólne dla modułu Bluetooth i ultradźwiękowego
	DOUT	D11	D11 wspólne dla modułu Bluetooth i ultradźwiękowego
	CS	D10	D10 wspólne dla modułu serwomechanizmu
Moduł Bluetooth	CE	D8	D8 wspólne dla modułu wykrywania przeszkód w podczerwieni
	SCK	D13	D13 wspólne dla modułu śledzenia linii
	MISO	D12	D12 wspólne dla modułu śledzenia linii
	IRQ	D4	D4 wspólne dla modułu komunikacji w podczerwieni
	MOSI	D11	D11 wspólne dla modułu wykrywania przeszkód w podczerwieni
	CSN	D7	D7 wspólne dla modułu wykrywania przeszkód w podczerwieni
	D0/RX	Przełącznik RX	
	D1/TX	Przełącznik TX	
Sterownik silników	IN1	A0	
	IN2	A1	
	ENA	D5	
	ENB	D6	
	IN3	A2	
	IN4	A3	

1.5.3. Tabela konfliktów sprzętu AlphaBot

	1	2	3	4	5	6	7	8	9
1. Moduł szybkości - impulsator									
2. Moduł ultradźwiękowy						K	K		

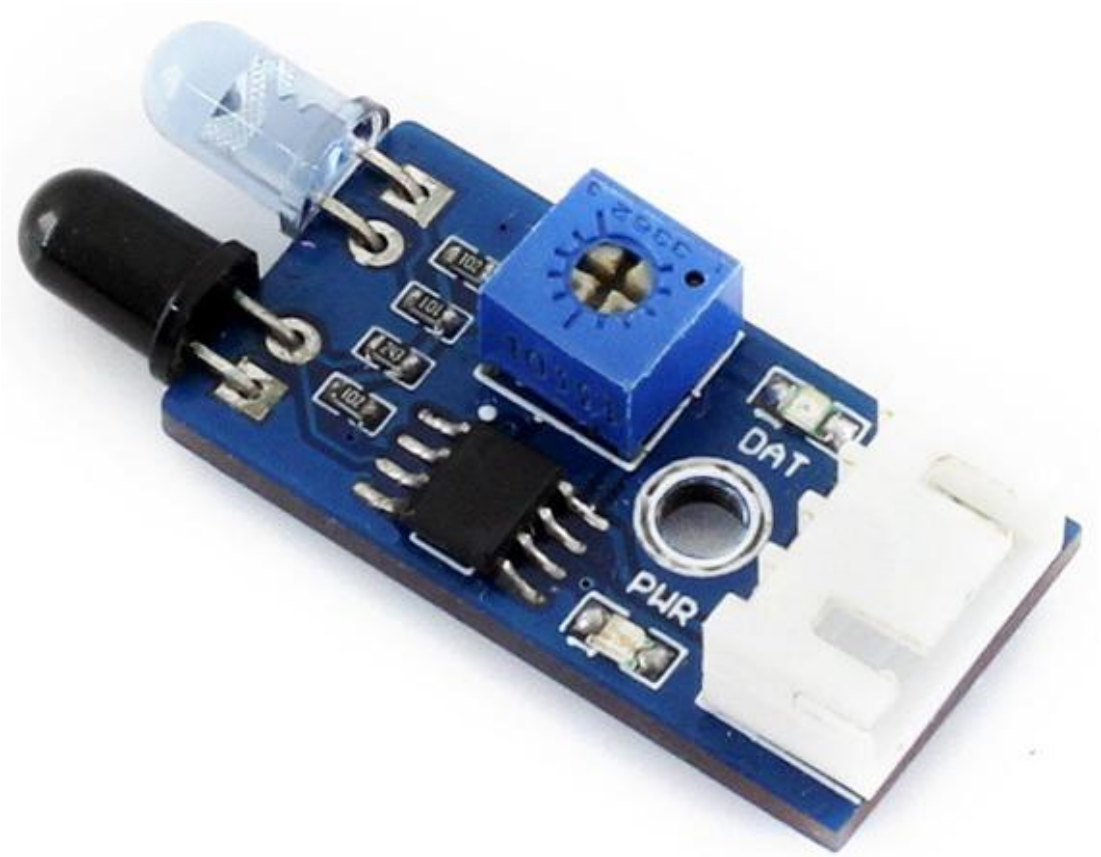
wykrywania przeszkód									
3. Moduł wykrywania przeszkód w podczerwieni							K		KKK
4. Serwomechanizm						K			
5. Moduł odbiornika podczerwieni							K		
6. Moduł śledzenia linii		K		K			K		
7. Moduł Bluetooth		K	K	K	K	K			
8. Sterownik silników									
9. JTAG			KKK						

1.5.4. AlphaBot - czujnik szczelinowy 2 szt.

Pin No.	Symbol	Descriptions

1	DOUT	Digital output
2	GND	Ground
3	VCC	Positive power supply (3.3V-5.0V)

1.5.5. AlphaBot - czujnik zbliżeniowy na podczerwień 2 szt.



Pin No.	Symbol	Descriptions
1	DOUT	Digital output
2	AOUT	Analog output
3	GND	Ground
4	VCC	Positive power supply (3.3V-5.0V)

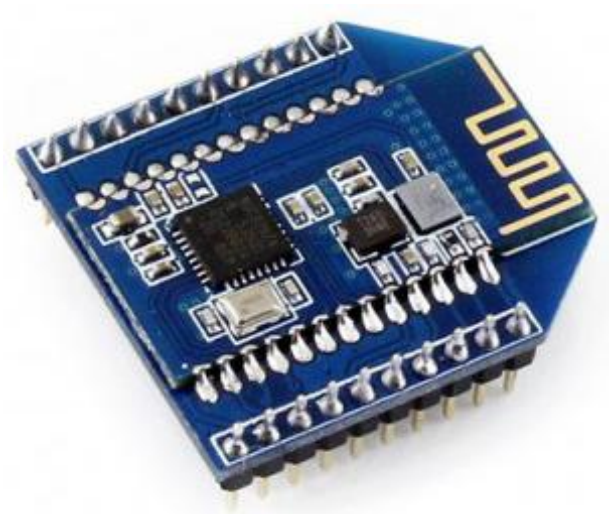
1.5.6. AlphaBot - czujnik linii 1 szt.

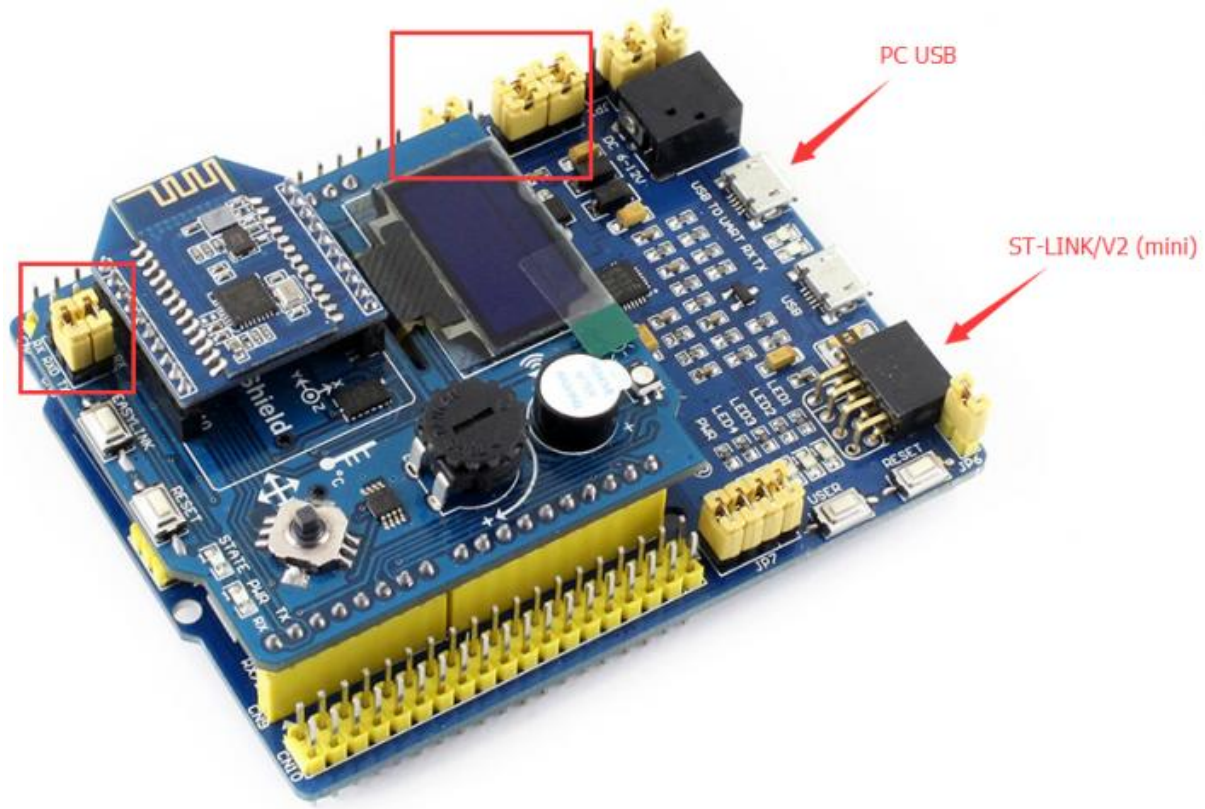
Port	Pins of Arduino
IIR1~IR5	A0~A4
IGND	GND
IVCC	5V

1.5.7. AlphaBot - czujnik ultradźwiękowy odległości 1 szt.

Port	Pins of Arduino
ECHO	D12
GND	GND
VCC	5V
TRIG	D11

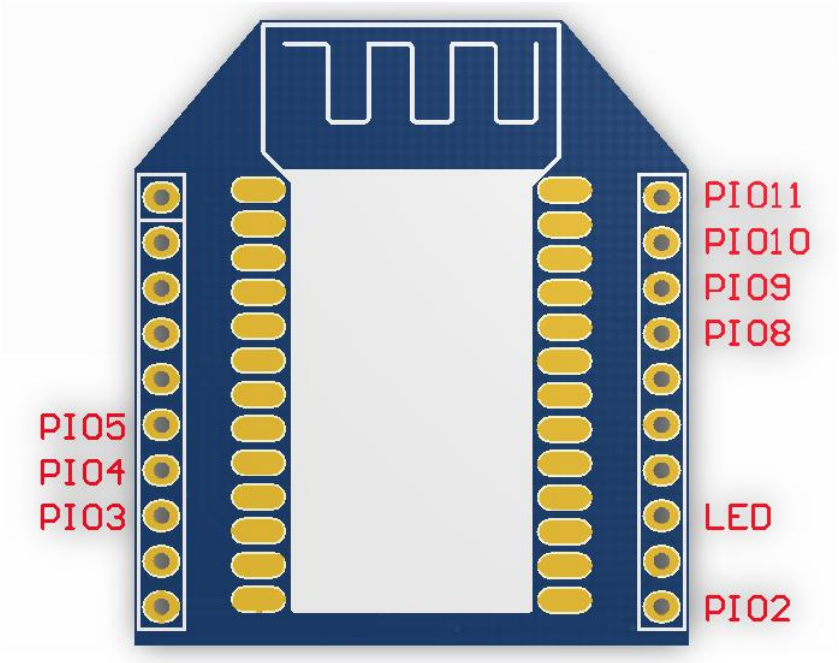
1.5.8. AlphaBot – moduł Bluetooth 1 szt.





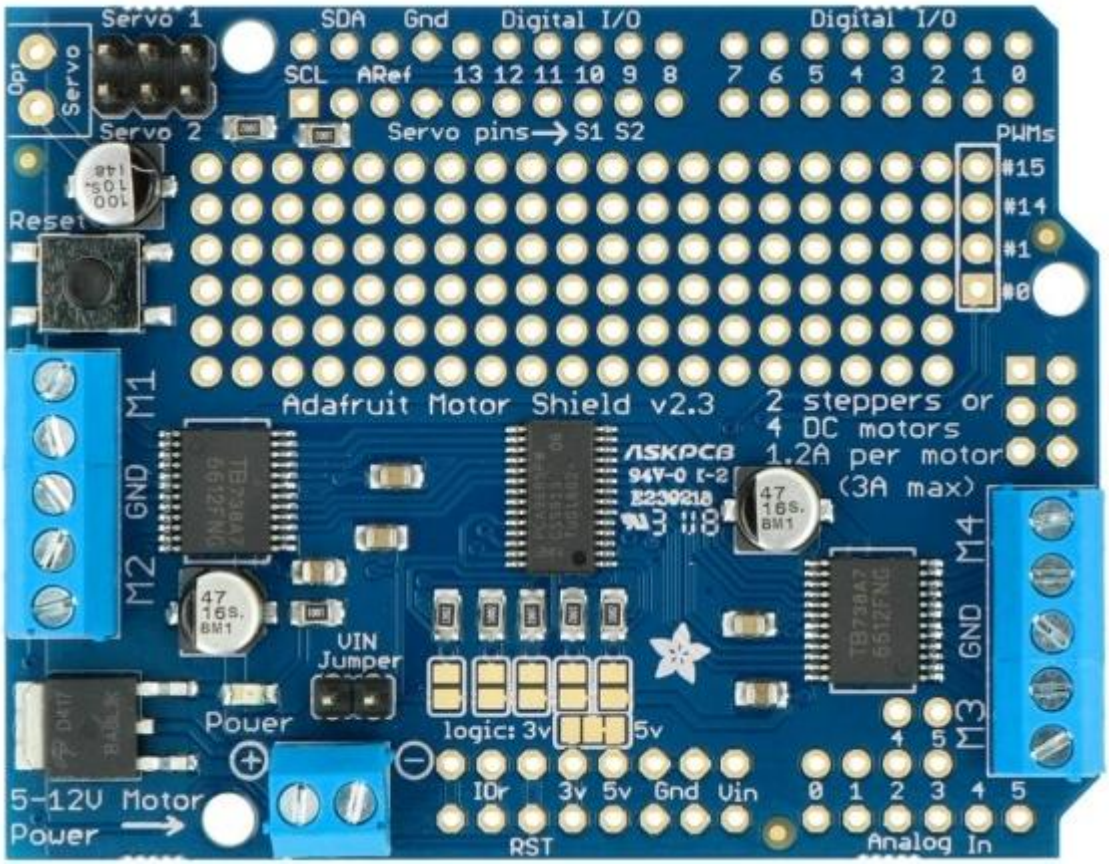
Pełna dokumentacja

https://www.waveshare.com/wiki/Dual-mode_Bluetooth



1.6. Płyty rozszerzeń

1.6.1. Płyta Adafruit Motor/Stepper/Servo Shield v. 2.3



Główne cechy modułu:

- Dwa złącza do podłączenia serwomechanizmów modelarskich - napięcie zasilania wynosi 5 V.
- Możliwość podłączenia czterech silników prądu stałego, dzięki zastosowaniu dwóch dwukanałowych sterowników TB6612:
  - napięcie zasilania: od 4,5 V do 13,5 V,
  - ciągły pobór prądu: do 1,2 A na kanał,
  - chwilowy pobór prądu: do 3 A na kanał.
- Możliwość podłączenia dwóch silników krokowych - unipolarnych, bipolarnych, z pojedynczą lub podwójną cewką.
- Możliwość podłączenia jednocześnie dwóch silników prądu stałego DC, jednego silnika krokowego oraz dwóch serwomechanizmów.
- Wlutowane złącza śrubowe, ułatwiające podłączenie silników i zasilania- kompatybilne z przewodami 18 - 26 AWG.
- Zabezpieczenie przed odwrotnym podłączeniem napięcia zasilania.
- Zworka umożliwiająca separację zasilania części logicznej od silników.
- Współpracuje z napięciami logicznymi 5 V i 3,3 V - wybierane za pomocą zworki.
- Produkt przetestowany z: Arduino Uno, Arduino Leonardo, Arduino Mega, Arduino Mega ADK, Arduino Due, Diecimila i Duemilanove.
- Wykorzystuje tylko dwa piny sygnałowe Arduino: SCL i SDA
- Udostępniona, darmowa biblioteka Arduino.
- Komunikacja poprzez magistralę I2C z 7-bitowym adresem, wybieranym za pomocą zworki z zakresu 0x60 - 0x80.
- Wymiary: 70 x 55 x 10 mm.

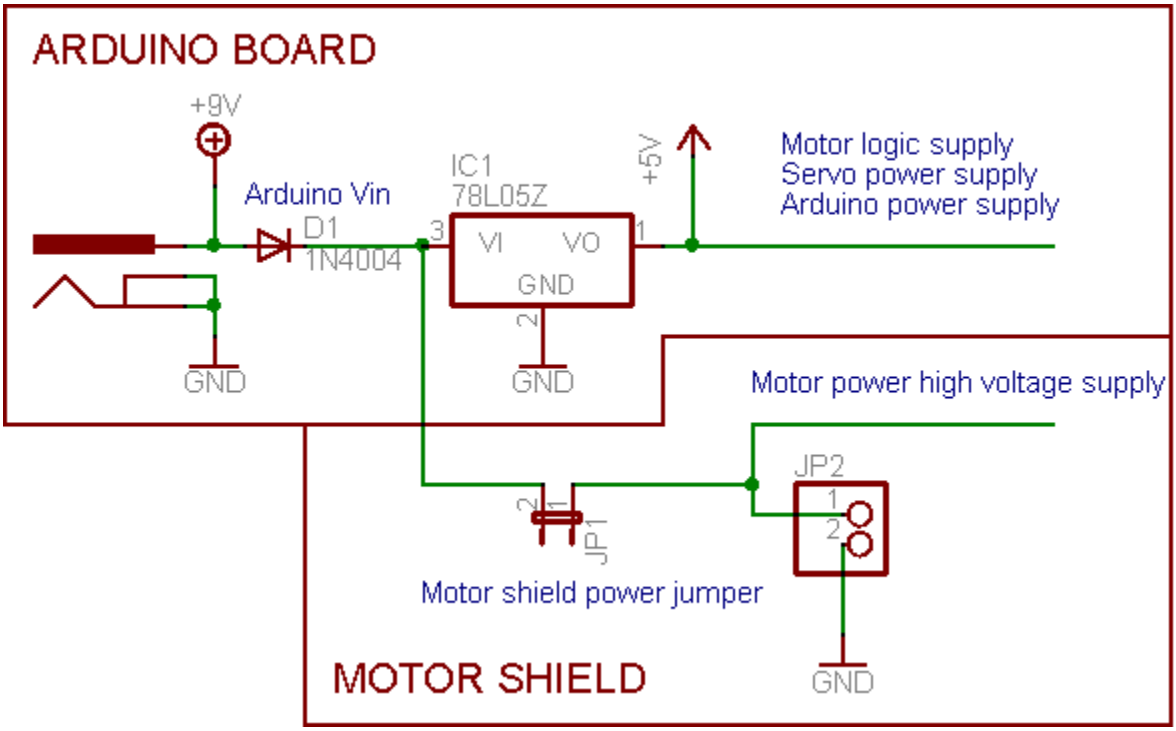
https://learn.adafruit.com/adafruit-motor-shield-v2-for-arduino

https://github.com/adafruit/Adafruit_Motor_Shield_V2_Library

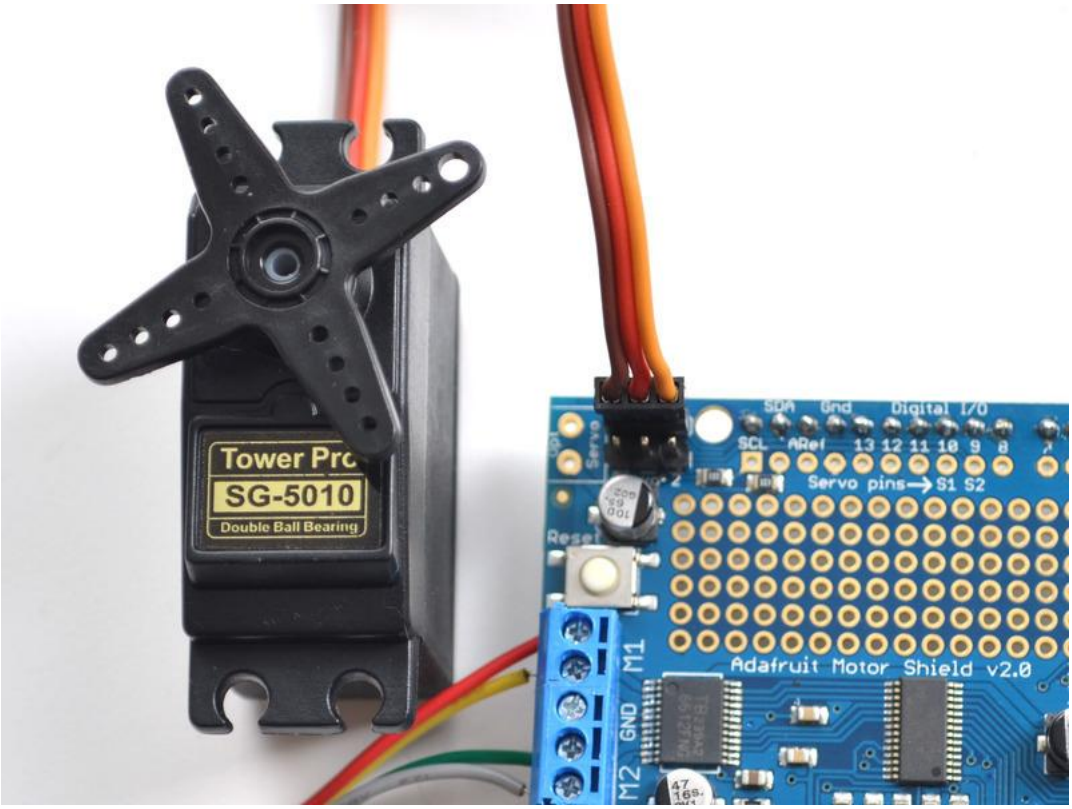
Tabela mapy wyprowadzeń:

	Wyprowadzenie na module	Wyprowadzenie na gnieździe Arduino	UWAGI
Komunikacja I2C	SDA	SDA	Korzystamy z pinów złącza PWM (NIE z pinów złącza analogowego A4/SDA)
	SCL	SCL	Korzystamy z punów złącza PWM (NIE z pinów złacza analogowego A5/SCL)

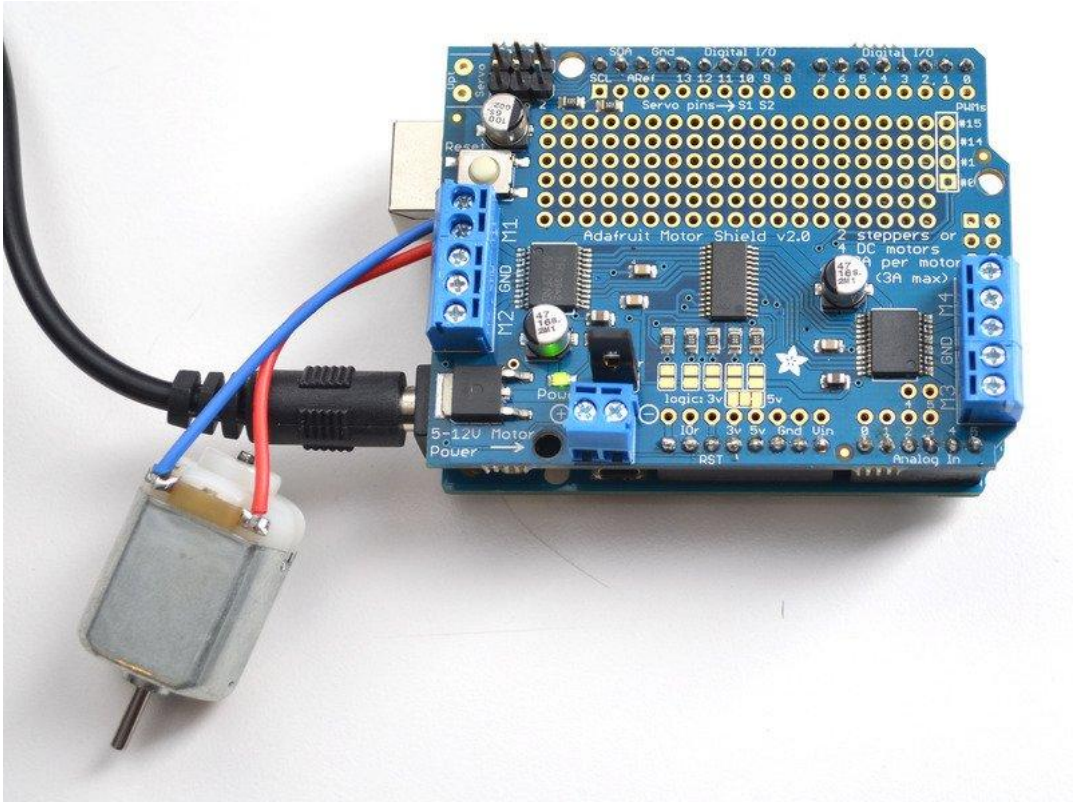
Zasilanie:



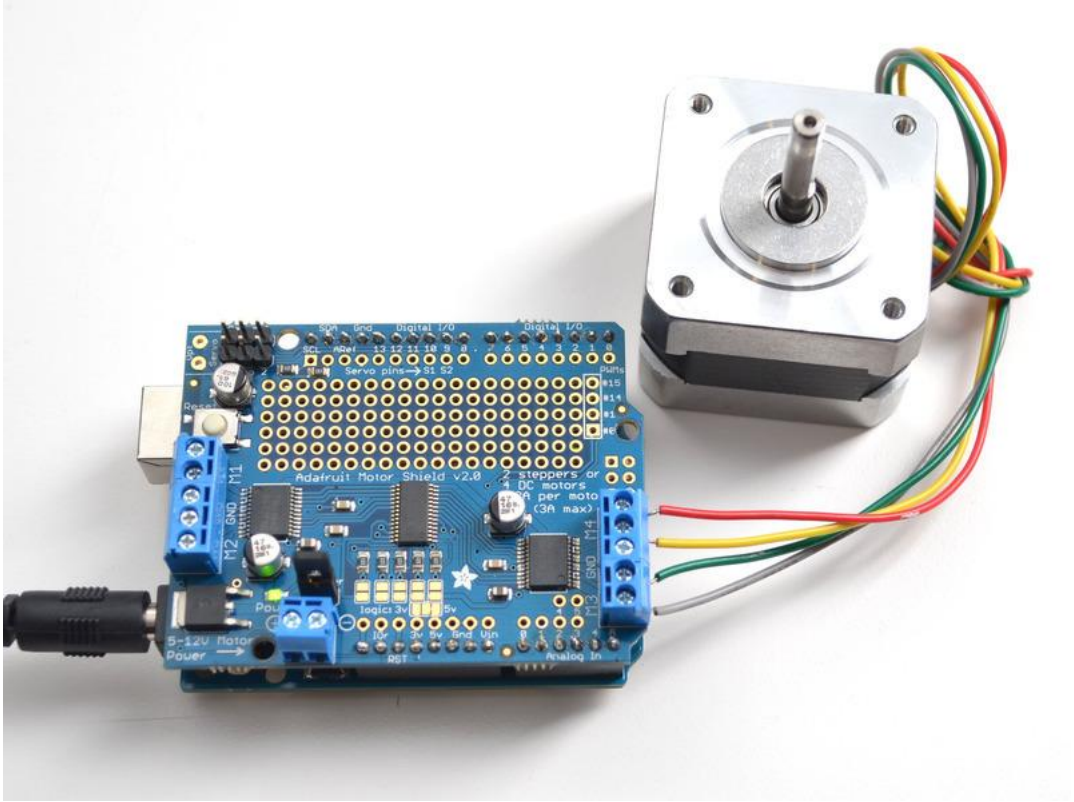
Praca z serwem (UWAGA – tylko 5 V):



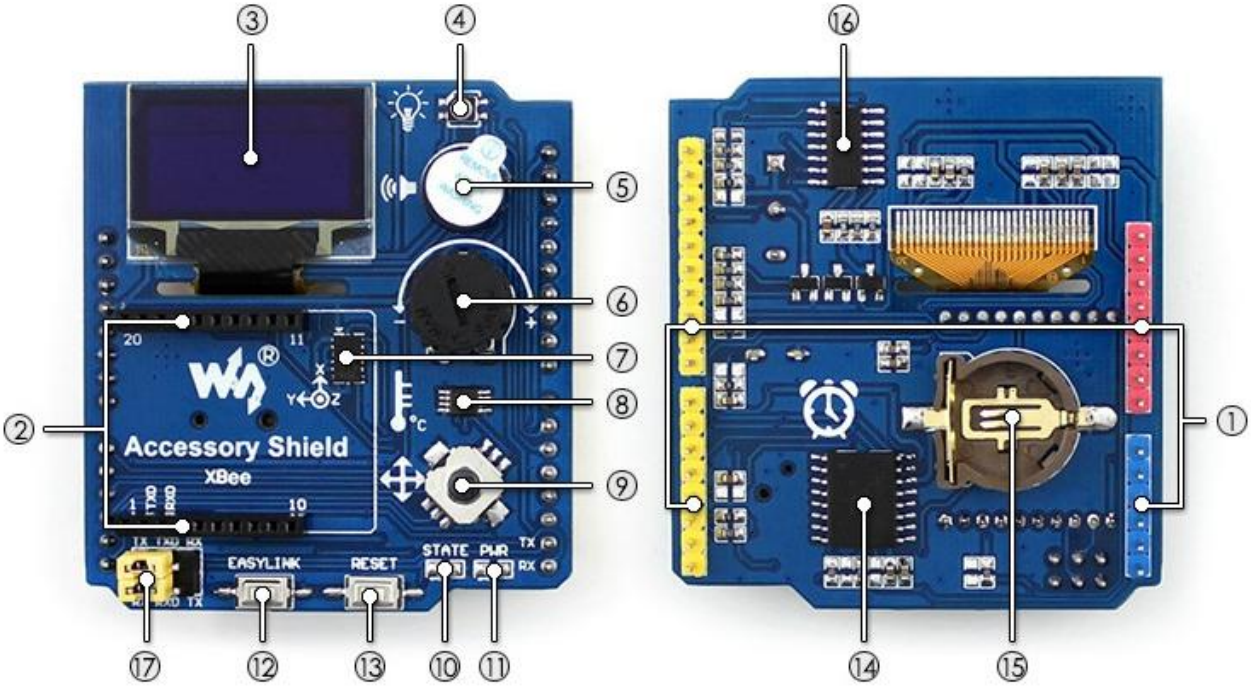
Praca z silnikiem DC (prąd ciągły nie więcej niż 1,2 A!) – do 4 silników



Praca z silnikiem krokowym (do 2 silników):



1.6.2. Płyta Waveshare Accesory Shield



Oznaczenia na rysunku:

1. Złącza kompatybilne z Arduino
2. Złącze dla modułu XBee
3. Wyświetlacz OLED o przekątnej 0.96 cala, rozdzielczość 128x64
4. Dioda LED RGB (8x8x8-bitowy kolor)
5. Głośnik sterowany linią PWM
6. Potencjometr 10k
7. 3-osiowy akcelerometr ADXL345 (maksymalny zakres pomiaru ±16g)
8. Czujnik temperatury LM75BDP o rozdzielczości 0.125℃
9. 5-pozycyjny joystick
10. Dioda statusu modułu XBee
11. Dioda zasilania
12. Przycisk easylink modułu XBee
13. Przycisk reset dla modułu XBee oraz Arduino
14. Zegar RTC DS3231
15. Gniazdo baterii CR1220 podtrzymującej RTC
16. Sterownik diody RGB P9813
17. Zworki wyboru funkcji linii interfejsu UART:
    17.1. Przy połączeniu TXD-TX i RXD-RX możliwe jest debugowanie i konfigurowanie modułu XBee przez port szeregowy Arduino z dołączonego do Arduino komputera
    17.2. Przy połączeniu TXD-RX i RXD-TX możliwa jest komunikacja Arduino przez moduł XBee

Właściwości

- Pięciopozycyjny joystick
- Potencjometr
- Głośnik
- Diodę LED RGB
- Czujnik temperatury
- 3-osiowy akcelerometr
- Zegar RTC
- 0.96-calowy wyświetlacz OLED
- Złącze modułu komunikacji bezprzewodowej XBee

Strona producenta:

http://www.waveshare.com/wiki/Accessory_Shield

http://www.waveshare.com/wiki/File:Accessory-Shield-Code.7z

http://www.waveshare.com/wiki/Accessory_Shield

Materiały na chmurze:

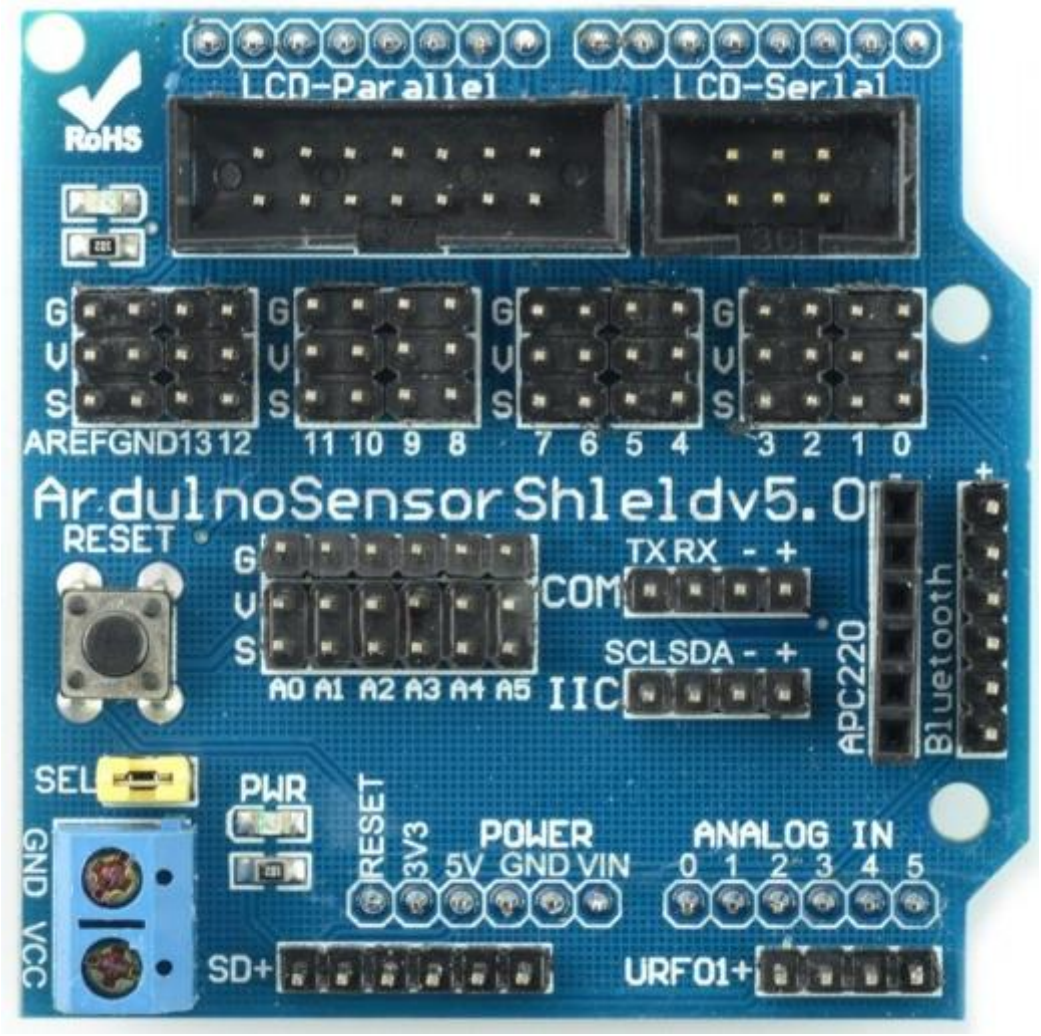
- Dokumentacja akcelerometru ADXL345
- Dokumentacja układu DS3231
- Dokumentacja układu SSD1306
- Dokumentacja układu P9813
- Dokumentacja układu LM75B

Tabela mapy wyprowadzeń

	Wyprowadzenie na module	Wyprowadzenie na gnieździe Arduino	UWAGI
Wyświetlacz OLED (adres 0x3C/0x3D)	14	AR_D7	
	15	AR_D8	
	18	SCL	Korzystamy z pinów złącza PWM (NIE z pinów złącza analogowego A4/SDA)
	19-20	SDA	Korzystamy z pinów złącza PWM (NIE z pinów złącza analogowego A5/SCL)
Głośniczek	in	AR_D11	
Sterownik diody RGB	ENABLE	AR_D12	
	DIN	AR_D5	
	CIN	AR_D6	
Akcelerometr ADXL (adres 0x53)	SCL	SCL	Korzystamy z pinów złącza PWM (NIE z pinów złącza analogowego A4/SDA)
	SDA	SDA	Korzystamy z pinów złącza PWM (NIE z pinów złącza analogowego A5/SCL)
	INT1	AR_D4	
Potencjometr	Odczep suwaka	AR_A0	
Joystick	A	AR_A3	
	B	AR_A1	
	C	AR_A5	KONFLIKT z JTAG – nie używać tego kierunku joysticka
	D	AR_A2	
	CTR	AR_A4	KONFLIKT z JTAG – nie używać tego kierunku joysticka
Złącze XBee	DOUT	XBee_TXD	Uaktywnić zworki XBee_TXD z AR_TX
	DIN	XBee_RXD	Uaktywnić zworki XBee_RXD z AR_RX
	RESET	AR_RST + AR_D2	
	RTS	AR_D10	
	ON	AR_D9	
	CTS	AR_D3	
Zegar RTC (adres 0x68)	SDA	SCL	Korzystamy z pinów

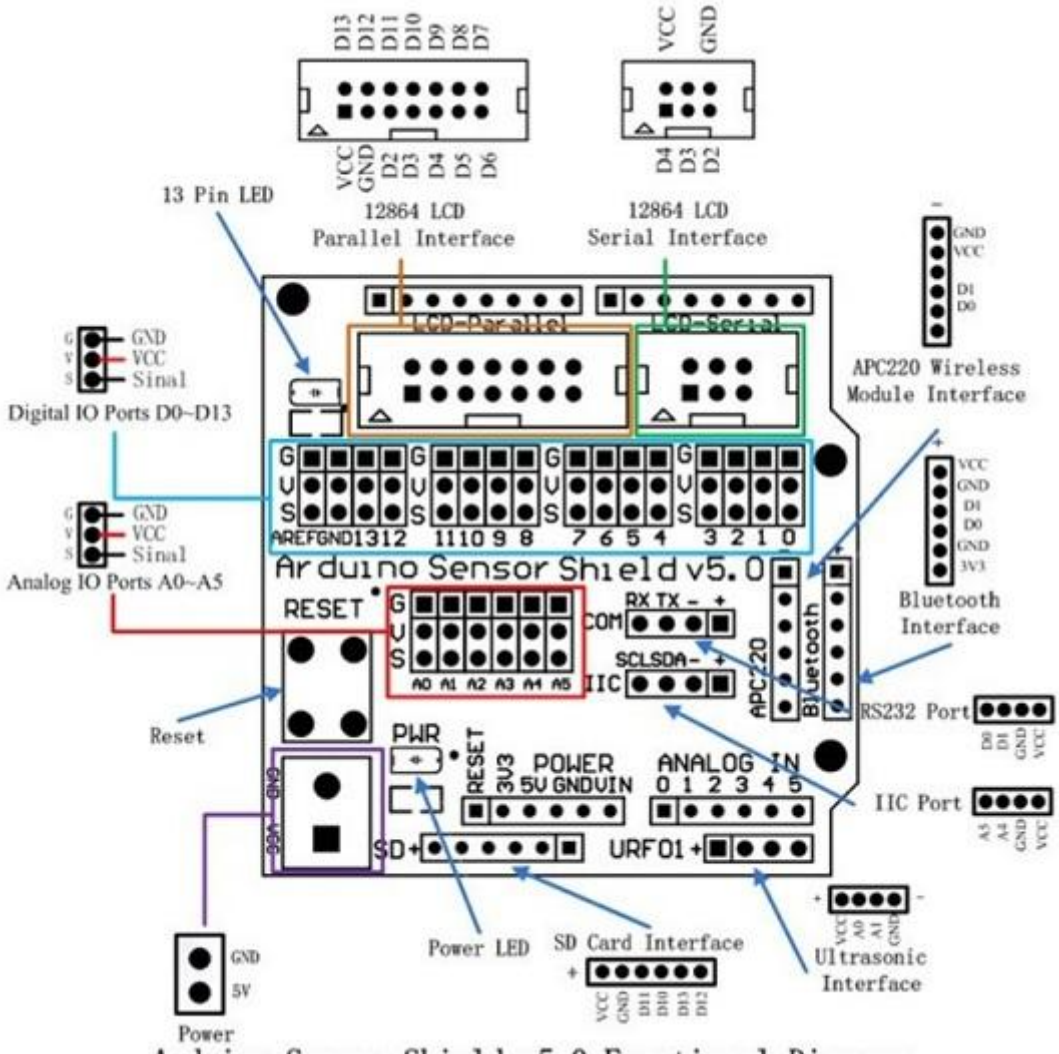
			złącza PWM (NIE z pinów złącza analogowego A4/SDA)
	SCL	SDA	Korzystamy z pinów złącza PWM (NIE z pinów złącza analogowego A5/SCL)
Czujnik LM75BDP (adres 0x48)	SDA	SCL	Korzystamy z pinów złącza PWM (NIE z pinów złącza analogowego A4/SDA)
	SCL	SDA	Korzystamy z pinów złącza PWM (NIE z pinów złącza analogowego A5/SCL)

1.6.3. Płyta Waveshare Sensor Shield v5.0



Specyfikacja:

- Napięcie zasilanie: 5 V z Arduino
- Złącze interfejsu UART
- Złącze interfejsu APC220
- Złącze do podłączenia czujnika ultradźwiękowego
- Złącze interfejsu LCD szeregowego i równoległego
- 14 cyfrowych portów IO
- 6 analogowych wejść
- Zewnętrzne wejście zasilania
- Przycisk reset
- Wymiary: 57 x 56 mm



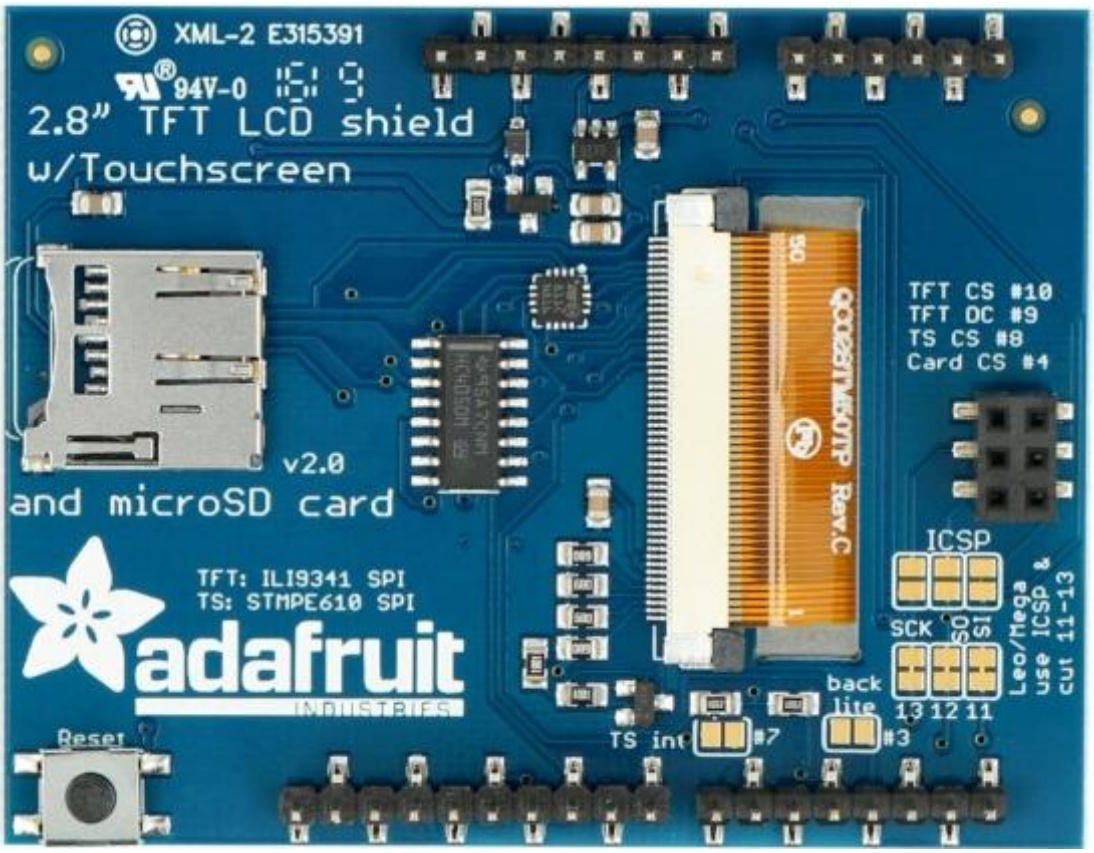
Arduino Sensor Shield v5.0 Functional Diagram

Tabela mapy wyprowadzeń

	Wyprowadzenie na module	Wyprowadzenie na gnieździe Arduino	UWAGI
Analogowe wejścia	A0	A0	
	A1	A1	
	A2	A2	
	A3	A3	
	A4	A4	KONFLIKT z JTAG – nie używać tego wyprowadzenia
	A5	A5	KONFLIKT z JTAG – nie używać tego wyprowadzenia
Cyfrowe wejścia	D0-D13	D0-D13	
UART / RS232	D0	D0	
	D1	D1	
Karta SD	PWM5/SS	D10	
	PWM6/MOSI	D11	

	D12/MISO	D12	
	D13/SCK	D13	
Port IIC	AD4		KONFLIKT z JTAG – nie używać tego wyprowadzenia
	AD5		KONFLIKT z JTAG – nie używać tego wyprowadzenia
Czujnik ultradźwiękowy URF01	A0	A0	
	A1	A1	
Bluetooth	D0	D0	
	D1	D1	
APC220	D0	D0	
	D1	D1	
12864 LCD równoległe	D2-D13	D2-D13	
12864 LCD szeregowe	D2-D4	D2-D4	
Dioda LED	D13	D13	

1.6.4. Płyta wyświetlacza dotykowego 2,8 cala



Wbudowany kontroler z pamięcią RAM sprawia, że mikrokontroler nie jest obciążony operacjami związanymi z wyświetlaniem danych.

Urządzenie wykorzystuje tylko 5 wyprowadzeń:

- 3 dla magistrali SPI,
- 1 dla kontrolera dotyku
- 1 dla czytnika kart microSD.

W celu uruchomienia wyświetlacza z funkcją dotyku należy pobrać i umieścić w folderze Arduinolibraries pliki bibliotek z chmury lub githuba.

Dokumentacja i biblioteki

https://www.adafruit.com/product/1651

https://github.com/adafruit/Adafruit_STMPE610

https://github.com/adafruit/Adafruit_ILI9341

https://learn.adafruit.com/adafruit-2-8-tft-touch-shield-v2

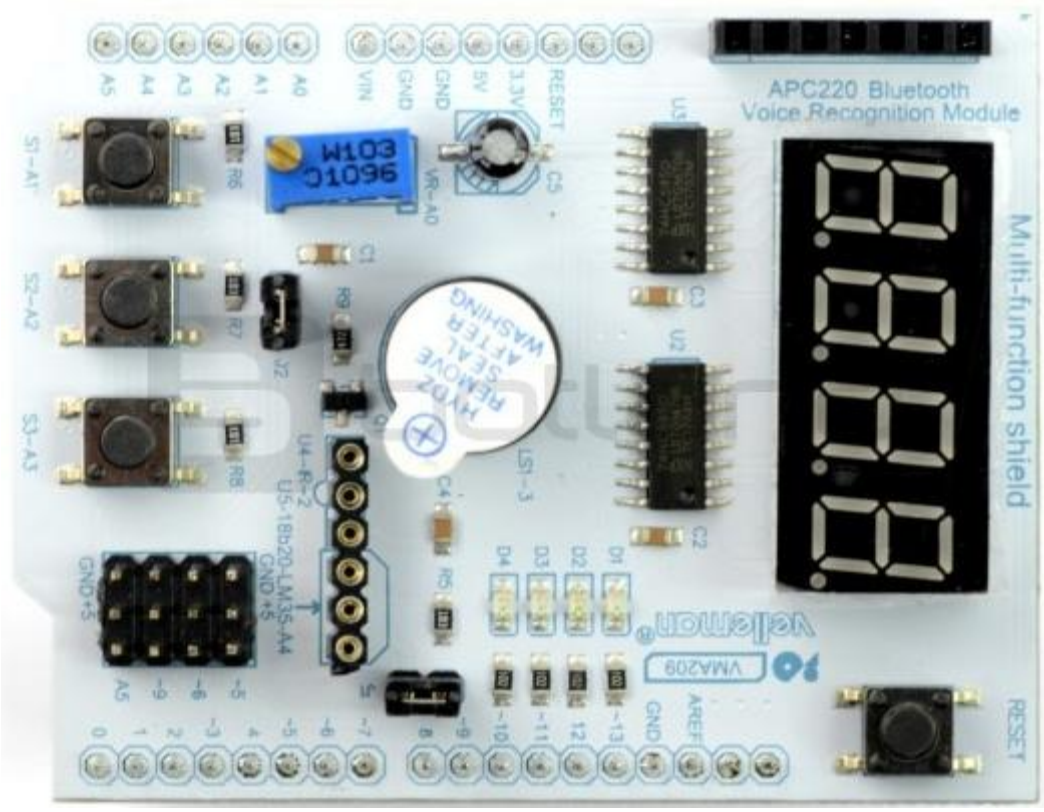
1.6.5. Power Driver Shield Kit

Płytka z 6 wyjściami z tranzystorami RFP30N06LE MOSFETS 30 A 60 V.

Tabela mapy wyprowadzeń

	Wyprowadzenie na module	Wyprowadzenie na gnieździe Arduino	UWAGI
Kanał 0	D11	D11	
Kanał 1	D10	D10	
Kanał 2	D9	D9	
Kanał 3	D6	D6	
Kanał 4	D5	D5	
Kanał 5	D3	D3	

1.6.6. Nakładka wielofunkcyjna dla Arduino: Velleman VMA209 czujniki LM35 + DS18B20

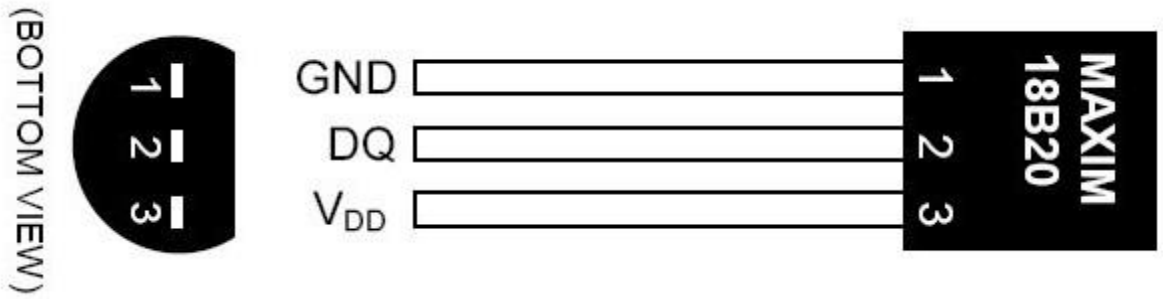


Płytka zawiera:

- czteroznakowy, 7 segmentowy wyświetlacz LED
- 4 diody LED SMD
- 3 przyciski
- buzzer
- interfejsy dla czujnika temperatury LM35 oraz DS18B20
- interfejs odbiornika podczerwieni
- potencjometr precyzyjny 10 kΩ
- złącze interfejsu szeregowego

Specyfikacja czujnika LM35DZ

- Napięcie zasilania: od 4 V do 30 V
- Skala skalibrowana dla stopni Celsjusza
- Liniowa skala: 10,0 mV/°C
- Dokładność 0,5 °C
- Zakres pomiarowy: od -55 °C do 150 °C
- Prąd drenu poniżej 60 mA



Specyfikacja DS18B20

- Napięcie zasilania: od 3,0 V do 5,5 V
- Zakres pomiarowy: od -55 °C do 125 °C
- Dokładność: +/- 0,5 °C w zakresie -10 °C do 85 °C
- Rozdzielczość: od 9 do 12 bitów
- Obudowa THT TO92

Tabela mapowania wyprowadzeń

	Wyprowadzenie na module	Wyprowadzenie na gnieździe Arduino	UWAGI
Dioda D1	13	D13	
Dioda D2	12	D12	
Dioda D3	11	D11	
Dioda D4	10	D10	
Złącze pinowe 5	5	D5	
Złącze pinowe 6	6	D6	
Złącze pinowe 9	9	D9	
Złącze pinowe A5	A5	A5	Konflikt z JTAG – nie używać
Złącze 18B20/LM35	5	D0	
	4	D1	
Wyświetlacz 7-segmentowy	SDI	D8	
	SFTCLK	D7	
	LCHCLK	D4	
Potencjometr	Odczep	A0	

Przycisk 1	S1	A1	
	S2	A2	
	S3	A3	
Głośnik	We	D3	
SFH506 – odbiornik podczerwieni	OUT	D2	
DS18B20	DQ	A4	Konflikt z JTAG – nie używać

1.6.7. Zestaw czujników

W skład zestawu wchodzą:

L.p.	Zdjęcie	Opis
1		Czujnik gazu LPG MQ-5 - wykrywa LPG, gaz ziemny oraz tlenek węgla.
2		Czujnik koloru z układem TCS3200 - wykrywa kolor statyczny.
3		Czujnik płomieni - zakres wykrywanej fali: od 760 do 1100 nm.
4		Liniowy czujnik efektu Halla - posiada zmienny stopień czułości.

5		Czujnik odbiciowy na podczerwień - posiada zmienny stopień czułości.
6		Czujnik odległości optyczny, cyfrowy - efektywny zakres pomiaru: 0,8 m.
7		Czujnik wilgotności gleby - głębokość detekcji: 38 mm.
8		Czujnik obrotu - enkoder - posiada 20 kroków na jeden obrót.
9		Czujnik dźwięku - działa w zakresie od 50 Hz do 20 kHz.
10		Czujnik temperatury i wilgotności - posiada zakres temperatury od 0 do 50 °C, a wilgotności od 20 do 90 % RH.
11		Czujnik wychylenia / wstrząsu - komunikuje się cyfrowo z mikrokontrolerem.
12		Czujnik ultrafioletu - posiada zakres wykrywanej fali: od 200 do 370 nm.

13		Czujnik poziomu cieczy - umożliwia detekcję na głębokość do 48 mm.
14		Przewody połączeniowe ze złączami żeńskimi - raster 2,54 mm.

1.7. Strony i inne dokumenty źródłowe ATmega

Strony i inne dokumenty źródłowe ATmega	
Aktualna strona główna:	https://www.microchip.com/
Strona główna mikrokontrolera:	https://www.microchip.com/wwwproducts/en/ATmega32
Oprogramowanie dla AVR:	https://www.microchip.com/avr-support/atmel-start
Atmel Studio 7:	https://www.microchip.com/avr-support/atmel-studio-7
Lista instrukcji::	http://ww1.microchip.com/downloads/en/devicedoc/atmel-0856-avr-instruction-set-manual.pdf https://en.wikipedia.org/wiki/Atmel_AVR_instruction_set
Asembler AVR i dyrektywy:	https://www.microchip.com/webdoc/avrassembler/ http://www.avr-tutorials.com/assembly/basics-assembly-language/

	dokumentacja w plikach pdf
Dokumentacje języka C:	https://www.gnu.org/software/gnu-c-manual/ https://www.gnu.org/software/gnu-c-manual/gnu-c-manual.html - można też korzystać z wersji PDF.
Biblioteki funkcji i prekompilowane biblioteki pełne języka C:	Libc.pdf https://www.nongnu.org/avr-libc/user-manual/index.html - można też korzystać z wersji PDF. http://nongnu.org/avr-libc/user-manual/pages.html http://nongnu.org/avr-libc/user-manual/modules.html http://www.microchip.com/webdoc/AVRLibcReferenceManual/
Inne WWW:	

1.8. Dokumentacje do pobrania, wydrukowania

Pobrać dostępne dokumentacje - podane przez Prowadzącego:

A. Pliki dostępne w Internecie.

B. Pliki do zgrania od Prowadzącego.

C. Pliki zgromadzone w chmurze (adres, hasła, konta – do pobrania u Prowadzącego).

1.9. Programowanie Arduino Uno Plus w środowisku Atmel Studio

Atmel Studio 7 jako IDE dla Arduino:

1. Pobierz i zainstaluj AtmelStudio 7 - http://www.atmel.com/microsite/atmel-studio/;

2. Pobierz i zainstaluj Arduino IDE w wersji 1.8.2 lub nowszej, najlepiej na c: / driver - https://www.arduino.cc/en/Main/Software;

3. Otwórz Arduino IDE i Przykład > Podstawowy > Szkic Blink;

4. Zmodyfikuj ten szkic, dodając „const int led = 13;” i zmień odpowiednią klauzulę w loop ();

5. Zapisz, jak chcesz, w folderze projektu; Przejdę do nazwania go jako „_26_arduSerie_atmelStudio_00.ino”;

6. Otwórz aplikację Atmel Studio 7 - podłącz Arduino na USB;

7. Plik > Nowy > Projekt i kliknij „Utwórz nowy projekt ze szkicu Arduino”;

8. Wybierz „Nazwa” = studio_code, dodaj lokalizację pliku swojego projektu; utworzy to folder „studio_code” w twoim projekcie;

9. Skonfiguruj „Sketch File” = lokalizacja pliku blink.ino, Arduino IDE Path = twoje Arduino IDE, „Board” = Arduino/Genuino Uno, „Device” = atmega328P i naciśnij „Ok”;

10. Spowoduje to zainstalowanie niezbędnych bibliotek arduino, głównie załadowanie pliku core.a, pins_arduino.h, all.h, który jest podstawą konfiguracji arduino wybranej platformy, w tym przypadku Arduino Uno;

11. Otrzymasz kod gotowy do dlaszej pracy;

12. Kompiluj > Kompiluj rozwiązanie, a dane wyjściowe pokażą:

„Kompilacja powiodła się.

== Kompilacja: 2 udane lub aktualne, 0 nieudane, 0 pominięte === ’;

13. Teraz wyślij kod na płytkę: w Atmel Studio 7 kliknij „Narzędzia” > „Narzędzia zewnętrzne”;

14. Skonfiguruj to w następujący sposób:

Tytuł: „Send to Arduino UNO”

Polecenia: kliknij „…” i przejdź do instalacji arduino: C:\arduino\hardware\tools\avr\bin\avrdude.exe i naciśnij „otwórz”;

Argumenty: w tym celu należy załadować szkic za pomocą Arduino IDE nominalnie przy użyciu ustawienia „Preferencje”, wybierając opcję „Pokaż pełne dane wyjściowe podczas”: „upload”;

Uwaga: pole Polecenia wymaga pliku avrdude.exe. Widzisz, możesz bezpośrednio pobrać program avrdude: http://mirror.rackdc.com/savannah//avrdude/. Pobierz najnowszą wersję http://mirror.rackdc.com/savannah//avrdude/.

15. Przejdź do Arduino IDE, a w oknie kompilacji zobaczysz:

Szkic wykorzystuje 928 bajtów (2%) miejsca do przechowywania programu. Maksymalnie wynosi 32256 bajtów.

Zmienne globalne używają 9 bajtów (0%) pamięci dynamicznej, pozostawiając 2039 bajtów dla zmiennych lokalnych. Maksymalnie jest 2048 bajtów.C: \ arduino \ hardware \ tools \ avr / bin / avrdude -CC: \ arduino \ hardware \ tools \ avr / etc / avrdude.conf -v -patmega328p -carduino -PCOM3 -b115200 -D -Uflash : w: C: \ Users \ giljr \ AppData \ Local \ Temp \ arduino_build_666279 / _26_arduSerie_atmelStudio_00.ino.ino.hex: i avrdude: Wersja 6.3, skompilowana 17 stycznia 2017 o 12:00:53 Copyright © 2000–2005 Brian Dean , http://www.bdmicro.com/

Copyright © 2007–2014 Joerg Wunsch Systemowy plik konfiguracyjny to C: \ arduino \ hardware \ tools \ avr / etc / avrdude.conf ”Korzystanie z portu: COM3

Za pomocą programatora: arduino

Nadrzędna prędkość transmisji: 115200

Część AVR: ATmega328P (…)

Ovrdude to program odpowiedzialny za zapisywanie kodu na chipie. To ten program jest potrzebny w Atmel Studio 7;

16 Skopiuj ten fragment kodu do „Argumentów”:

-CC:\arduino\hardware\tools\avr/etc/avrdude.conf -v -patmega328p -carduino -PCOM3 -b115200 -D -Uflash:w:

17. Uzupełnij go:

“$(ProjectDir)Debug\$(TargetName) .hex”:i

więc wynikiem będzie:

-CC:\arduino\hardware\tools\avr/etc/avrdude.conf -v -patmega328p -carduino -PCOM3 -b115200 -D -Uflash:w:"$(ProjectDir)Debug\$(TargetName) .hex":i

18. Wybierz „Użyj okna wyników” i naciśnij „Ok”;

19. To wszystko: teraz kliknij „Narzędzia > Wyślij do Arduino UNO”, a kod musi zostać przesłany do Arduino Uno, a na płytce dioda LED musi migać;

Zobaczysz na Output:

Wykonano avrdude.exe. Dziękuję Ci.

UWAGA:

Problem z folderem Arduino zagnieżdżonym w folderze Program Files (x86), po przeniesieniu pliku Arduino bezpośrednio do folderu C:/ działa bez problemów. Występujący błąd to: „avrdude.exe: błąd odczytu ogólnosystemowego pliku konfiguracyjnego „C:\Program""

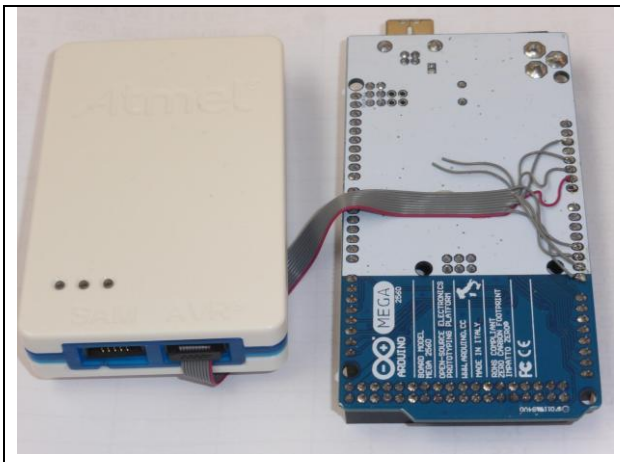
1.10. Programowanie i debugowanie ATMega2560 z wykorzystaniem złącza JTAG

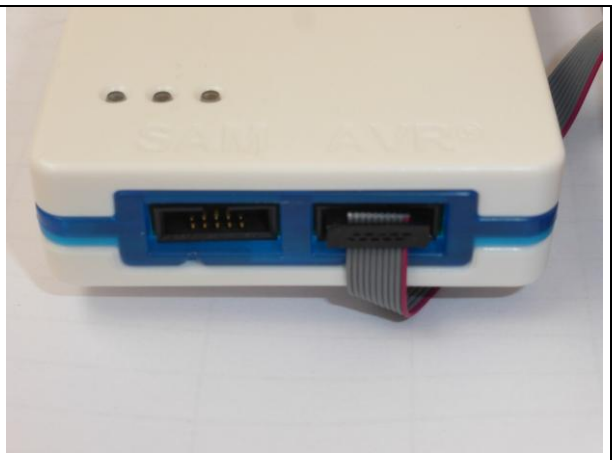
Wyprowadzenia na taśmie programatora – złącze AVR	Wyprowadzenie Arduino Mega
1	A4
2	GND
3	A6
4	5 V
5	A5
6	RESET
7	NC
8	NC
9	A7
10	GND

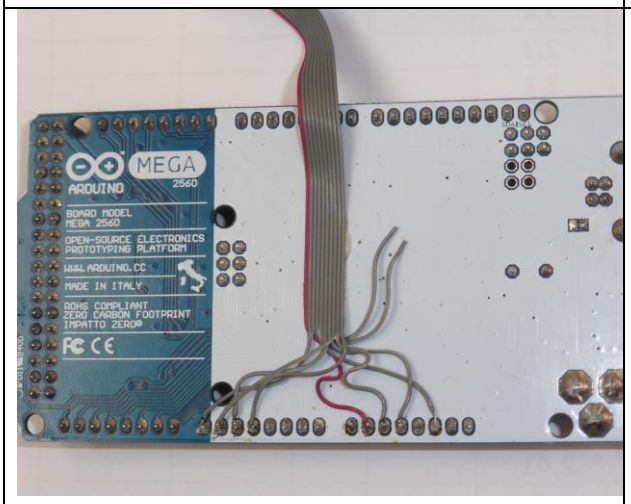
UWAGA: w ustawieniach projektu wybieramy programator z ustawieniem trybu pracy JTAG.

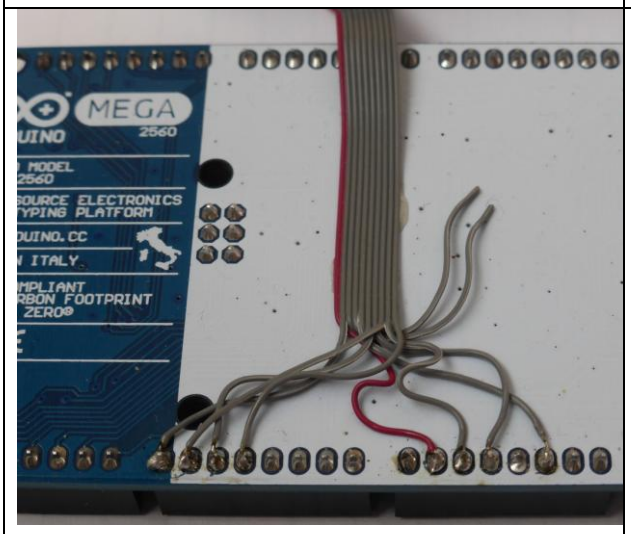
JTAG Pin	Arduino Pin	Kolor (jeśli występuje)
TCK	ADC4	brown
TMS	ADC5	green
TDO	ADC6	orange
TDI	ADC7	white
GND	GND	black, red
Vref	+5	yellow
SRST	Reset	blue

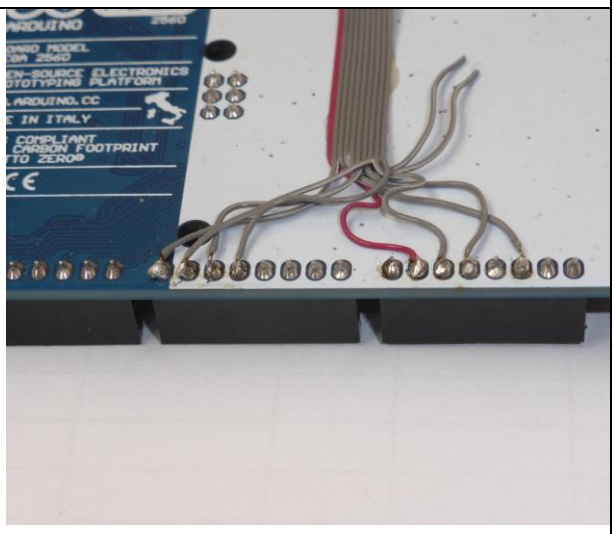
1.11. Samodzielne przygotowanie AtMega2560 do debugowania z wykorzystaniem interfejsu JTAG











Na zdjęciach wyprowadzenie numer 10 jest zaznaczone czerwonym kabelkiem (w oryginalnej specyfikacji kabel numer 1 powinien być czerwony).

UWAGA: przed podłączeniem i wykorzystywaniem złącza JTAG należy z wykorzystaniem złącza ICSP:

- Zgrać oryginalne oprogramowanie (ładujące), jeśli jeszcze chcemy używać kiedyś tej funkcji

- W ustawieniach bitów specjalnych:
  - Zezwolić bit JTAGEN
  - Zabronić bit BOOTRSZ
  - Zezwolić bit OCDBEN
- Po podłączeniu do komputera w ustawieniach projektu wybieramy debuger sprzętowy i ustawienie trybu pracy: JTAG

1.12. Biblioteki i pliki funkcji

Strony i inne dokumenty źródłowe ATmega		
Nazwa	Lokalizacja	Cechy
AVRlibc	Wbudowane w AVR Studio	• Rozbudowane – w formie skompilowanych bibliotek • Aktualne • Zalecane
Procyon	http://www.procyonengineering.com/embedded/avr/avrlib/	• Rozbudowane – w formie plików funkcji, sterowniki do peryferiów • Wymagają dodania kilku opcji do poprawnej kompilacji • Zalecane
Kamami	http://kamami.pl/dl/zl15avr_c_examples.zip	• Dostosowane do zestawu ZL15AVR • Aktualne • Nie wszystkie funkcje są godne polecenia
Arduino	Import do AVR Studio po wydaniu polecenia	• Dostosowane do zestawów Arduino • Aktualne • Nie stosujemy – jednak są godne polecenia

2. Przykłady zakładania projektów

2.1. Projekt asemblera

1. Uruchom AVR Studio - przetestuj podłączenie do programatora.
2. Wybierz nowy projekt typu asembler i utwórz go. Kod wpisujemy w pliku main.asm.
3. Zalecenia pisania programu:

   3.2. Asembler i dyrektywy AVR: https://www.microchip.com/webdoc/avrassembler/

   3.3. Przykłady: http://www.avr-tutorials.com/assembly/basics-assembly-language/

   3.4. W programie wykorzystuj dyrektywy asemblera np. „.org”.

   3.5. Ładowanie liczby do rejestru trzeba wykonywać specjalną instrukcją asemblera, dodatkowo z ograniczeniem dotyczącym liczby rejestrów, z którymi działa.

   3.6. Należy sprawdzić przypisanie i faktyczne podłączenie zasobów do płytki dydaktycznej. Dla przykładu diody świecące podłączone są do portu PORTC i PORTD (ale tylko do wybranych wyprowadzeń), dioda świeci dla stanu „1”.

   3.7. Aby dany działał jako wyjściowy, należy ustawić kierunek w rejestrze DDRx, wyjście dla stanu „1”.

   3.8. Przy operacjach na rejestrach specjalnych stosujemy predefiniowane nazwy.

   3.9. Etykiety skoków i podprogramów zaznaczamy dwukropkiem.

   3.10. Pamiętaj, że asemblerowe instrukcje inkrementacji nie ustawiają pewnych flag.

   3.11. Adresowanie rejestrów portu dotyczy specjalnego obszaru adresowania tzw. I/O – wymagane jest użycie specjalnej instrukcji.

4. W ustawieniach projektu – kompilator, optymalizacja – należy koniecznie ustawić / zmienić domyślna opcję optymalizacji -O1 na:

   4.2. Opcję -O0 (none).

5. Podczas uruchamiania programu wybierz programator/debugger JTAGICE3 z interfejsem JTAG.

2.2. Projekt języka C z bibliotekami wbudowanymi AVRlibc

1. Uruchom AVR Studio - przetestuj podłączenie do programatora.
2. Wybierz nowy projekt typu język C i utwórz go. Kod wpisujemy w pliku main.c.

3. Zalecenia pisania programu:

3.2. Język C – dokumentacja i przykłady:

3.2.1. https://www.gnu.org/software/gnu-c-manual/

3.2.2. https://www.gnu.org/software/gnu-c-manual/gnu-c-manual.html

3.3. Dokumentacja i przykłady dla bibliotek AVRlibc:

3.3.1. Libc.pdf

3.3.2. https://www.nongnu.org/avr-libc/user-manual/index.html - można też korzystać z wersji PDF.

3.3.3. http://nongnu.org/avr-libc/user-manual/pages.html

3.3.4. http://nongnu.org/avr-libc/user-manual/modules.html

3.4. Należy sprawdzić przypisanie i faktyczne podłączenie zasobów do płytki dydaktycznej. Dla przykładu diody świecące podłączone są do portu PORTC i PORTD (ale tylko do wybranych wyprowadzeń), dioda świeci dla stanu „1”. Aby dany działał jako wyjściowy, należy ustawić kierunek w rejestrze DDRx, wyjście dla stanu „1”.

3.5. Zapisy rejestrów wykonujemy stosując ich predefiniowane nazwy np. PORTA.

3.6. Biblioteki AVRlibc dołączamy do projektu przez dopisanie do głównego pliku nagłówkowego (jeśli używamy) lub do pliku „main.c”.

3.6.1. Przykład: dołączenie biblioteki matematycznej: #include <math.h>

3.6.2. Przykład: dołączenie biblioteki IO: #include <stdio.h>

3.6.3. Przykład: dołączenie biblioteki obsługującej przerwania: #include <avr/interrupts.h>

3.7. W przypadku bibliotek innych niż AVRlibc: biblioteki i pliki funkcji wypakowujemy do wybranego katalogu (może znajdować się poza katalogiem domowym AVR Studio).

3.8. W przypadku bibliotek innych niż AVRlibc: dołączamy wybrane biblioteki / pliki funkcji metodą referencji do projektu (wybierz okno projektu – prawy przycisk i „Add” – „Existing item” - **********).

3.9. UWAGA: dołączenie bibliotek do projektu nie oznacza, że zostanie ona „zauważona” przez kompilator i odwrotnie – dodanie „include” biblioteki nie dołącza go wprost do projektu, tylko do jego zależności. Dlatego w praktyce, aby zachować spójność projektu: zawsze bibliotekę jednocześnie dołączamy do projektu i includujemy w plikach programu.

4. W ustawieniach projektu – kompilator, optymalizacja – należy koniecznie ustawić / zmienić domyślna opcję optymalizacji -O1 na:

4.2. Opcję -O0 (none).

5. Podczas uruchamiania programu wybierz programator/debugger JTAGICE3 z interfejsem JTAG.

2.3. Projekt języka C z bibliotekami zewnętrznymi KAMAMI

1. Uruchom AVR Studio - przetestuj podłączenie do programatora.
2. Wybierz nowy projekt typu język C i utwórz go. Kod wpisujemy w pliku main.c.
3. Zalecenia pisania programu:

3.2. Język C – dokumentacja i przykłady:

3.2.1. https://www.gnu.org/software/gnu-c-manual/

3.2.2. https://www.gnu.org/software/gnu-c-manual/gnu-c-manual.html

3.3. Dokumentacja i przykłady dla bibliotek KAMAMI:

3.3.1. http://kamami.pl/dl/zl15avr_c_examples.zip

3.4. Należy sprawdzić przypisanie i faktyczne podłączenie zasobów do płytki dydaktycznej. Dla przykładu diody świecące podłączone są do portu PORTC i PORTD (ale tylko do wybranych wyprowadzeń), dioda świeci dla stanu „1”. Aby dany działał jako wyjściowy, należy ustawić kierunek w rejestrze DDRx, wyjście dla stanu „1”.

3.5. Zapisy rejestrów wykonujemy stosując ich predefiniowane nazwy np. PORTA.

3.6. Dołączanie do projektu bibliotek KAMAMI:

3.6.1. Biblioteki i pliki funkcji wypakowujemy do wybranego katalogu (może znajdować się poza katalogiem domowym AVR Studio).

3.6.2. Następnie dołączamy wybrane biblioteki / pliki funkcji metodą referencji do projektu (wybierz okno projektu – prawy przycisk i „Add” – „Existing item” - **********).

3.6.3. Wreszcie biblioteki dołączamy do plików projektu przez dopisanie do głównego pliku nagłówkowego (jeśli używamy) lub do pliku „main.c”.

3.6.3.1. Przykład: dołączenie biblioteki obsługi wyświetlacza LCD: #include <HD44780.h>

3.6.3.2. Przykład: dołączenie biblioteki spi: #include <spi.h>

3.6.3.3. Przykład: dołączenie biblioteki i2c (uwaga na inną nazwę interfejsu): #include <twi.h>

3.6.4. Jeśli to konieczne – sprawdzamy, czy biblioteki nie wymagają dostosowania do wykorzystywanej płytki dydaktycznej (elektrycznych połączeń na niej wykonanych).

Przykład: dla wyświetlacza LCD należy zmienić w bibliotece HD44780 wyprowadzenia domyślne, na faktycznie wykonane na zestawie dydaktycznym.

3.7. UWAGA: dołączenie bibliotek do projektu nie oznacza, że zostanie ona „zauważona” przez kompilator i odwrotnie – dodanie „include” biblioteki nie dołącza go wprost do projektu, tylko do jego zależności. Dlatego w praktyce, aby zachować spójność projektu: zawsze bibliotekę jednocześnie dołączamy do projektu i includujemy w plikach programu.

4. W ustawieniach projektu – kompilator, optymalizacja – należy koniecznie ustawić / zmienić domyślna opcję optymalizacji -O1 na:

4.2. Opcję -O0 (none).

5. Podczas uruchamiania programu wybierz programator/debugger JTAGICE3 z interfejsem JTAG.

2.4. Projekt języka C z bibliotekami zewnętrznymi Procyon

1. Uruchom AVR Studio - przetestuj podłączenie do programatora.
2. Wybierz nowy projekt typu język C i utwórz go. Kod wpisujemy w pliku main.c.
3. Zalecenia pisania programu:

3.2. Język C – dokumentacja i przykłady:

3.2.1. https://www.gnu.org/software/gnu-c-manual/

3.2.2. https://www.gnu.org/software/gnu-c-manual/gnu-c-manual.html

3.3. Dokumentacja i przykłady dla bibliotek Procyon:

3.3.1. http://www.procyonengineering.com/embedded/avr/avrlib/

3.3.2. Na stronie istnieje dokumentacja bibliotek online i do pobrania.

3.4. Należy sprawdzić przypisanie i faktyczne podłączenie zasobów do płytki dydaktycznej. Dla przykładu diody świecące podłączone są do portu PORTC i PORTD (ale tylko do wybranych wyprowadzeń), dioda świeci dla stanu „1”. Aby dany działał jako wyjściowy, należy ustawić kierunek w rejestrze DDRx, wyjście dla stanu „1”.

3.5. Zapisy rejestrów wykonujemy stosując ich predefiniowane nazwy np. PORTA.

3.6. Dołączanie do projektu bibliotek Procyon:

3.6.1. Biblioteki i pliki funkcji wypakowujemy do wybranego katalogu (może znajdować się poza katalogiem domowym AVR Studio).

3.6.1.1. UWAGA: liczba i stopień skomplikowania plików bibliotek jest duża – dlatego najlepiej poprosić Prowadzącego o gotowy zestaw wybranych bibliotek przystosowanych do zajęć.

3.6.2. Następnie dołączamy wybrane biblioteki / pliki funkcji metodą referencji do projektu (wybierz okno projektu – prawy przycisk i „Add" – „Existing item" - **********).

3.6.3. Wreszcie biblioteki dołączamy do plików projektu przez dopisanie do głównego pliku nagłówkowego (jeśli używamy) lub do pliku „main.c".

3.6.3.1. Przykład: dołączenie biblioteki obsługi wyświetlacza LCD: #include <HD44780.h>

3.6.3.2. Przykład: dołączenie biblioteki spi: #include <spi.h>

3.6.3.3. Przykład: dołączenie biblioteki i2c (uwaga na inną nazwę interfejsu): #include <twi.h>

3.6.4. Jeśli to konieczne – sprawdzamy, czy biblioteki nie wymagają dostosowania do wykorzystywanej płytki dydaktycznej (elektrycznych połączeń na niej wykonanych). Przykład: dla wyświetlacza LCD należy zmienić w bibliotece HD44780 wyprowadzenia domyślne, na faktycznie wykonane na zestawie dydaktycznym.

3.6.5. Edycję opcji konfigurowalnych dla bibliotek wykonujemy w plikach XXXXXconf.h.

3.7. UWAGA: jako pierwszy należy dołączyć do projektu plik nagłówkowy global.h. Jego treść modyfikujemy przez dodanie:

3.7.1. Dla bibliotek Procyon – konieczne jest dodanie definicji w pliku global.h:
3.7.2. #define SIG_INTERRUPT0 _VECTOR(1)
3.7.3. #define SIG_INTERRUPT1 _VECTOR(2)
3.7.4. #define SIG_INTERRUPT2 _VECTOR(3)
3.7.5. #define SIG_OUTPUT_COMPARE2 _VECTOR(4)
3.7.6. #define SIG_OVERFLOW2 _VECTOR(5)
3.7.7. #define SIG_INPUT_CAPTURE1 _VECTOR(6)
3.7.8. #define SIG_OUTPUT_COMPARE1A _VECTOR(7)
3.7.9. #define SIG_OUTPUT_COMPARE1B _VECTOR(8)
3.7.10. #define SIG_OVERFLOW1 _VECTOR(9)
3.7.11. #define SIG_OUTPUT_COMPARE0 _VECTOR(10)
3.7.12. #define SIG_OVERFLOW0 _VECTOR(11)
3.7.13. #define SIG_SPI _VECTOR(12)
3.7.14. #define SIG_UART_RECV _VECTOR(13)
3.7.15. #define SIG_UART_DATA _VECTOR(14)
3.7.16. #define SIG_UART_TRANS _VECTOR(15)
3.7.17. #define SIG_ADC _VECTOR(16)
3.7.18. /* EEPROM Ready */
3.7.19. //#define EE_RDY_vect _VECTOR(17)
3.7.20. /* Analog Comparator */
3.7.21. //#define ANA_COMP_vect _VECTOR(18)
3.7.22. #define SIG_2WIRE_SERIAL _VECTOR(19) //
3.7.23. /* Store Program Memory Ready */

3.7.26. //#define SPM_RDY_vect _VECTOR(20)

3.8. W pliku global.h komentujemy każde wystapienie definicji F_CPU (najlepiej pobrać gotowy plik global.h od Prowadzącego).

3.9. Natomiast we właściwościach projektu – kompilacja, miscallineus, opcje dodatkowe – dostawiamy w jednej linii następujące opcje:

a. -DF_CPU=16000000UL
b. -Wno-deprecated-declarations
c. -D__PROG_TYPES_COMPAT__

3.10. UWAGA: jeśli chcemy korzystać z drukowania / porównywania łańcuchów w formatach float (np. wykorzystanie w sprintf opcji %f), należy wykonać edycje ustawień projektu:

3.10.1. Opcje linkera AVR/GNU Linker – General – okienko “Use vprintf library (-Wl,-u,vfprintf)” ustawić jako aktywne

3.10.2. Opcje linkera AVR/GNU Linker – Miscallineus – dodać w linii Other link flags: “-lprintf_flt -lm”

3.10.3. UWAGA: dodanie tych opcji znacznie zwiększa rozmiar program wynikowego!

3.11. UWAGA: dołączenie bibliotek do projektu nie oznacza, że zostanie ona „zauważona” przez kompilator i odwrotnie – dodanie „include” biblioteki nie dołącza go wprost do projektu, tylko do jego zależności. Dlatego w praktyce, aby zachować spójność projektu: zawsze bibliotekę jednocześnie dołączamy do projektu i includujemy w plikach programu.

4. W ustawieniach projektu – kompilator, optymalizacja – należy koniecznie ustawić / zmienić domyślna opcję optymalizacji -O1 na:

4.2. Opcję -O0 (none).

5. Podczas uruchamiania programu wybierz programator/debugger JTAGICE3 z interfejsem JTAG.

3. Przygotowanie płyty dydaktycznej

3.1. Informacje ogólne

Do pobrania u Prowadzącego

Przygotowanie do pracy:

1. Komputer z Win32/64 z zainstalowanym AVR Studio.
2. Płyta dydaktyczna Arduino Uno Plus lub Mega 2560.
3. 2 kable USB: A-B do zasilania oraz A-Amini dla programatora.
4. Ew. zasilacz 9-12 V min. 0,2 A z wtykiem zasilania 5 mm bolec 2 mm (na bolcu „+”). Domyślnie zestaw zasilamy z portu USB – konieczne jest włączenie przełącznika kołyskowego SW8.

3.2. Załączenie płyty dydaktycznej

Włączenie zestawu:

1. Podłącz kabel sieciowy / zasilacz – lub kabel USB (zasilanie kablem USB jest domyślne).
2. Podłącz kabel do programatora / debuggera.
    a. Sprawdź poprawność instalacji sterowników.
3. Włącz zasilanie zestawu wyłącznikiem kołyskowym SW8.
    a. Powinna zaświecić się dioda zasilania oraz podświetlenie wyświetlacza LCD.

3.3. Opcje i ustawienia kompilacji

Ustawienia kompilatora I:

1. W ustawieniach projektu – kompilator, optymalizacja – należy koniecznie ustawić / zmienić domyślna opcję optymalizacji -O1. Zamiast niej ustawić:
    a. Ustawić opcję -O0 (none).
    b. Wybranie innych lub pozostawienie domyślnej opcji optymalizacji może powodować usuwanie niektórych fragmentów kodu, zmiennych.

Ustawienia kompilatora II:

1. Dla bibliotek lub plików pobranych z Procyona – kompilacja, miscallineus, opcje dodatkowe – dostawiamy w jednej linii następujące opcje:
    a. -DF_CPU=16000000UL
    b. -Wno-deprecated-declarations
    c. -D__PROG_TYPES_COMPAT__
    d. Wymagane dodanie definicji podanych w punkcie 3.3 do pliku global.h:

```
#define SIG_INTERRUPT0 _VECTOR(1)
#define SIG_INTERRUPT1 _VECTOR(2)
#define SIG_INTERRUPT2 _VECTOR(3)
#define SIG_OUTPUT_COMPARE2 _VECTOR(4)
#define SIG_OVERFLOW2 _VECTOR(5)
#define SIG_INPUT_CAPTURE1 _VECTOR(6)
#define SIG_OUTPUT_COMPARE1A _VECTOR(7)
#define SIG_OUTPUT_COMPARE1B _VECTOR(8)
#define SIG_OVERFLOW1 _VECTOR(9)
#define SIG_OUTPUT_COMPARE0 _VECTOR(10)
#define SIG_OVERFLOW0 _VECTOR(11)
#define SIG_SPI _VECTOR(12)
#define SIG_UART_RECV _VECTOR(13)
#define SIG_UART_DATA _VECTOR(14)
#define SIG_UART_TRANS _VECTOR(15)
#define SIG_ADC _VECTOR(16)
/* EEPROM Ready */
//#define EE_RDY_vect          _VECTOR(17)
/* Analog Comparator */
//#define ANA_COMP_vect          _VECTOR(18)
```

```
#define SIG_2WIRE_SERIAL _VECTOR(19) //
/* Store Program Memory Ready */
//#define SPM_RDY_vect          _VECTOR(20)
```

Ustawienia kompilatora II:

1. UWAGA: jeśli chcemy korzystać z drukowania / porównywania łańcuchów w formatach float (np. wykorzystanie w sprintf opcji %f), należy wykonać edycje ustawień projektu:
    a. Opcje linkera AVR/GNU Linker – General – okienko “Use vprintf library (-Wl,-u,vfprintf)” ustawić jako aktywne
    b. Opcje linkera AVR/GNU Linker – Miscallineus – dodać w linii Other link flags: “-lprintf_flt -lm”
    c. UWAGA: dodanie tych opcji znacznie zwiększa rozmiar program wynikowego!

Ustawienia kompilatora IV:

1. Należy natychmiast wprowadzić zalecenia dotyczące użytkowania płyt dydaktycznych opisane w następnym punkcie – OSTRZEŻENIA.

Ustawienia środowiska:

1. Podczas uruchamiania programu wybierz programator/debugger JTAGICE3 z interfejsem JTAG.

Ustawienia sprzętowe:

1. Sprawdzić i ew. skorygować podłączenie zasobów na płycie dydaktycznej we punktu instrukcji „Przypisanie zasobów sprzętowych płytki”.

4. Użytkowanie – OSTRZEŻENIA!

Odpowiedzialność studenta za korzystanie z zestawów dydaktycznych – brak bieżącego monitorowania temperatury urządzenia lub niestosowanie się do wymogów ochrony elektrostatycznej jest podstawą do usunięcia studenta z zajęć i w przypadku uszkodzenia sprzętu do domagania się jego naprawy (wymiany na nowy - sprawny).

Jednocześnie jest zabronione modyfikowanie bitów blokujących programowanie procesora – np. bitów ochrony poszczególnych obszarów pamięci, a także innych bitów konfiguracji wewnętrznej procesora (nie uszkodzi to procesora, ale może powodować np. niestabilną jego pracę).

Urządzenie jest wrażliwe na ładunki elektrostatyczne – należy zachować szczególną ostrożność podczas jeg dotykania. Najlepej użytkować je na znajdujących się na stołach matach odprowadzających ładunki elektrostatyczne.

Włączenie zestawu - obowiązkowy monitoring jej pracy: Po uruchomieniu zestawu – należy sprawdzić czy nie występuje przegrzewanie elementów.

Włączenie płyt dydaktycznych - obowiązkowy monitoring jej pracy: Jeśli występują jakiekolwiek problemy z pracą zestawu dydaktycznego, programatora / debugera, komputera – należy je natychmiast zgłaszać Prowadzącemu.

Aktualizacja firmware programatora:

Po podłączeniu programatora i połączeniu się ze środowiskiem programistycznym - może pojawić się komunikat o braku zgodności firmware programatora ze środowiskiem .

Należy bezwzględnie wezwać Prowadzącego – nie wydawać zgody na automatyczny upgrade firmware.

5. Ćwiczenia do wykonania

5.1. Ćwiczenie 1 (Tydzień 1). Obsługa stanowiska, oprogramowania i płytki dydaktycznej

A. Przygotuj zestaw do pracy. Sprawdź poprawność podłączenia do zasilania i do programatora.

B. Na podstawie dokumentacji sprawdź podłączenie peryferiów lub płytek z peryferiami do układu płytki głównej mikrokontrolera – w szczególności podłączenie programatora.

C. Uruchom AVR Studio - przetestuj podłączenie do programatora.

D. Wybierz nowy projekt typu asembler i utwórz go. Kod wpisujemy w pliku main.asm.

E. Ustawienia kompilatora co do optymalizacji zmieniamy na –O0 (none).

F. Podczas uruchamiania programu wybierz programator/debugger JTAGICE3 z interfejsem JTAG.

G. Miganie diodą w pętli opóźniającej.

   a. Wykorzystane zasoby: dowolnie wybrana dioda LED.

H. Zmodyfikuj ustawienia tak aby zmienić okres migania diody.

I. Ustaw miganie diodą na okres podany przez Prowadzącego. Dokonaj obliczeń na podstawie napisanego programu i wiedzy o czasie wykonywania poszczególnych poleceń asemblera – wylicz teoretycznie, a następnie zweryfikuj praktycznie poprawność obliczeń.

J. Podłącz więcej niż jedną diodę. Uzyskaj efekt świetlny typu „wąż” itp.

K. **Wskazówki**:

   a. Asembler i dyrektywy: https://www.microchip.com/webdoc/avrassembler/

   b. Przykłady: http://www.avr-tutorials.com/assembly/basics-assembly-language/

   c. W programie wykorzystuj dyrektywy asemblera np. „.org”.

   d. Ładowanie liczby do rejestru trzeba wykonywać specjalną instrukcją asemblera, dodatkowo z ograniczeniem dotyczącym liczby rejestrów, z którymi działa.

   e. Diody świecące podłączone zgodnie z wykorzystywaną płytka rozszerzeń.

   f. Aby port C działał jako wyjściowy, należy ustawić kierunek w rejestrze DDRA i DDRC, wyjście dla stanu „1”.

   g. Przy operacjach na rejestrach specjalnych stosujemy predefiniowane nazwy.

h. Aby miganie diody było widoczne, częstotliwość nie powinna być większa niż 25-50 Hz. W tym ćwiczeniu wyjątkowo czas odmierzamy pętlą opóźniającą (ogólnie niezalecane rozwiązanie).

i. Etykiety skoków i podprogramów zaznaczamy dwukropkiem.

j. Pamiętaj, że asemblerowe instrukcje inkrementacji nie ustawiają pewnych flag.

k. Adresowanie rejestrów portu dotyczy specjalnego obszaru adresowania tzw. I/O – wymagane jest użycie specjalnej instrukcji.

L. **Sprawozdanie**: załącz kluczową część programu – wyjaśnij jego działanie, uzupełnij i załącz komentarze w istotnych liniach kodu.

M. **Przygotować na następne zajęcia**: należy dokładnie zapoznać się z dyrektywami asemblera oraz poleceniami asemblera. Odpowiednie materiały zostały wymienione w instrukcji i są do pobrania w miejscu wskazanym przez Prowadzącego.

5.2. Ćwiczenie 2 (Tydzień 1). Asembler - program

A. Zmodyfikuj zadania wykonane w ćwiczeniu 1. Wybierz 4 przyciski, odczytuj ich stany. Do każdego przycisku przypisz okres migania diod (i/lub) zmiany sposobu świecenia / efektu świetlnego diod – przyciśnięcie przycisków niech wywołuje zmiany migania diod.

B. Wykonaj filtr – debouncer – układ do eliminacji drgań styków przycisków.

C. Wykorzystaj jeden przycisk jako „modyfikator” (shift) – razem z innymi przyciskami niech utworzy zwiększoną macierz klawiatury.

D. **Wskazówki**:

a. Przyciski są podłączone do odpowiedniego portu płytki rozszerzeń, wciśnięcie przycisku wymusza stan „0”, przycisk zwolniony to „1”.

b. Aby port działał jako wejściowy należy ustawić wymagane bity rejestru DDRB w stan „0”, dane czytać z rejestru PINB.

c. Filtr eliminacji drgań styków można wykonać na kilka sposobów – np. przez kilkukrotne odczytanie stanu wejścia i wybranie najczęściej występującej wartości.

E. **Sprawozdanie**: załącz kluczową część programu – wyjaśnij jego działanie, uzupełnij i załącz komentarze w istotnych liniach kodu.

F. **Przygotować na następne zajęcia**: należy dokładnie zapoznać się z przerwaniami procesora AVR i ich obsługą: wektory przerwań, stos, powrót z przerwania. Odpowiednie materiały zostały wymienione w instrukcji i są do pobrania w miejscu wskazanym przez Prowadzącego.

5.3. Ćwiczenie 3 (Tydzień 1). Asembler - przerwania

A. Zmodyfikuj zadania wykonane w ćwiczeniu 1 i 2. Tym razem do odmierzania czasu użyj przerwań.

B. **Wskazówki**:

a. Wypełnić tabelę wektorów przerwań. Pamiętaj o podaniu dyrektywy kodu programu „.cseg", dyrektywach wskazujących na poszczególne wektory w tablicy przerwań „.org", a także o bardzo małym rozmiarze pojedynczego wpisu w wektorze (zaledwie na jedną-dwie instrukcje, zależnie od ich rozmiaru w pamięci).

b. Program główny wraz z pętlą główną przesuwamy poza obszar wektorów przerwań (przynajmniej 50d), skok do tego obszaru wykonujemy z wektora resetu.

c. Przerwanie umieszczamy zazwyczaj za pętlą główną, musi być zakończone instrukcją „reti".

d. UWAGA: nie wolno „wskakiwać do" albo „wyskakiwać z" przerwania, choć skoki w obrębie przerwania są możliwe (z wyjątkiem pętli opóźniających – dlaczego?)

e. Ustawić wskaźnik stosu. Polecana wartość to najlepiej koniec obszaru pamięci SRAM. Rejestr stosu SP jest 16 bitowy – ładujemy oddzielnie jego część SPH i SPL (UWAGA – kolejność 8-bitowego ładowania i odczytywania rejestrów traktowanych jako 16-bitowe ma znaczenie):

```
***; Set TCNT1 to 0x01FF
ldi r17,0x01
ldi r16,0xFF
out TCNT1H,r17
out TCNT1L,r16
***; Read TCNT1 into r17:r16
in r16,TCNT1L
in r17,TCNT1H
```

f. Ustawić źródło przerwań (licznik):

i. Ustawić odpowiedni tryb pracy.

ii. Ustawić źródło zegara i prescaler.

iii. Zezwolić na przerwania timera (trzeba wybrać odpowiedni typ w lokalnej masce przerwań).

g. Zezwolić na pracę przerwań flagą globalną – i powinno to nastąpić po skonfigurowaniu wszystkich przerwań.

h. Debugowanie programu w obrębie przerwania wykonujemy wyłącznie ustawiając w nim pułapki (breakpoint).

C. **Sprawozdanie**: załącz kluczową część programu – wyjaśnij jego działanie, uzupełnij i załącz komentarze w istotnych liniach kodu.

D. **Przygotować na następne zajęcia**: należy dokładnie zapoznać się z materiałami dotyczącymi języka C: polecenia C, zmienne i stałe, typy danych. Odpowiednie materiały zostały wymienione w instrukcji i są do pobrania w miejscu wskazanym przez Prowadzącego.

5.4. Ćwiczenie 4 (Tydzień 2). Język C – tworzenie projektu

A. Wybierz nowy projekt typu język C (gcc) i utwórz go. Skonfiguruj narzędzie do debugowania – czyli JTAGICE3 w trybie JTAG.

B. W ustawieniach projektu – kompilatora wybieramy optymalizację na -O0 (none) zamiast innych wstępnie ustawionych (podobnie postępujemy w pozostałych ćwiczeniach)

C. Wykonaj to samo zadanie co w ćwiczeniu 1 – miganie diodą z użyciem pętli opóźniającej.

D. A następnie wprowadź takie same modyfikacje:

   a. Zmianę okresu migania diody.

   b. Ustawienie okresu migania diody na wartość podaną przez Prowadzącego.

   c. Miganie kilku diod.

   d. Efekt świetlny.

E. **Wskazówki**:

   a. Zapisy rejestrów wykonujemy stosując ich predefiniowane nazwy np. PORTA.

   b. Wstępny licznik pętli opóźniającej należy ustawić na wartość ok. 1 000 000, w związku z tym należy dobrać odpowiedni typ wykorzystywanej zmiennej!

F. **Sprawozdanie**: załącz kluczową część programu – wyjaśnij jego działanie, uzupełnij i załącz komentarze w istotnych liniach kodu.

G. **Przygotować na następne zajęcia**: należy dokładnie zapoznać się z materiałami dotyczącymi języka C: polecenia C, zmienne i stałe, typy danych. Odpowiednie materiały zostały wymienione w instrukcji i są do pobrania w miejscu wskazanym przez Prowadzącego.

5.5. Ćwiczenie 5 (Tydzień 2). Język C – prosty program

A. Uzupełnij program z poprzedniego ćwiczenia o odczyt danych z przycisków – zaprojektuj i uruchom funkcje powiązane z przyciskami.

B. Zaprojektuj debouncer w języku C.

C. Podobnie jak w asemblerze wykonaj przycisk „modyfikatora” (shift).

D. **Wskazówki**:

   a. Przyciski są podłączone do odpowiedniego portu płytki rozszerzeń, wciśnięcie przycisku wymusza stan „0”, przycisk zwolniony to „1”.

E. **Sprawozdanie**: załącz kluczową część programu – wyjaśnij jego działanie, uzupełnij i załącz komentarze w istotnych liniach kodu.

F. **Przygotować na następne zajęcia**: należy dokładnie zapoznać się z materiałami dotyczącymi języka C w zakresie obsługi przerwań. Odpowiednie materiały zostały wymienione w instrukcji i są do pobrania w miejscu wskazanym przez Prowadzącego.

5.6. Ćwiczenie 6 (Tydzień 2). Język C – przerwania, czasomierze, liczniki

A. Zmodyfikuj poprzednie ćwiczenie, tak aby do odmierzania czasu wykorzystać licznik oraz generowane przez niego przerwanie.

   a. Zrealizuj miganie diody z okresem 1 s z dużą dokładnością wyznaczoną przez przerwania wybranego licznika.

   b. Wykonaj stoper (działający na przerwaniach) do pomiaru czasu wykonywania obliczeń, wynik wyświetlaj na diodach świecących.

   c. Wykorzystując brzęczyk/głośnik oraz przyciski – zaprojektuj „pianino” z wybranymi dźwiękami odpowiadającymi fragmentowi skali muzycznej.

B. Wykorzystaj różne tryby pracy licznika oraz tym samym różne źródła przerwań (czyli przynajmniej jeszcze jedną wersję).

C. Wykorzystaj przerwanie do precyzyjnego obliczania czasu naciśnięcia przycisku, wynik wyświetlaj na diodach LED (rozdzielczość i sposób wyświetlania do ustalenia z Prowadzącym).

D. **Wskazówki**:

   a. Należy koniecznie dołączyć plik nagłówkowy #include<avr/interrupt.h>

   b. Konfiguracja liczników i przerwań jest identyczna jak dla ćwiczenia napisanego w asemblerze, chyba że wybierzesz inną wersję.

   c. Rejestry 16-bitowe można ładować danymi 16-bitowymi bezpośrednio.

   d. Zezwolenie na przerwania wykonujemy makrem sei(), zabronienie cli().

   e. Przerwanie projektujemy jak funkcję, tylko z symbolem ISR() (beztypowo) i parametrem odpowiadającym nazwie wybranego wektora przerwania np. TIMER1_OVR_vect, pomijamy polecenie „return”.

f. Testowanie przyjęcia przerwania wykonujemy wstawiając pułapkę (breakpoint) w treści przerwania.

E. **Sprawozdanie**: załącz kluczową część programu – wyjaśnij jego działanie, uzupełnij i załącz komentarze w istotnych liniach kodu.

F. **Przygotować na następne zajęcia**: należy dokładnie zapoznać się z materiałami dotyczącymi języka C: biblioteki i pliki dedykowanych zbiorów funkcji. Odpowiednie materiały zostały wymienione w instrukcji i są do pobrania w miejscu wskazanym przez Prowadzącego.

5.7. Ćwiczenie 7 (Tydzień 3). Język C – biblioteki i pliki dedykowanych zbiorów funkcji – PWM, ADC, potencjometr

A. Korzystanie z bibliotek w języku C – dokładniej plików .c i nagłówkowych .h. Dodaj podane przez Prowadzącego biblioteki do projektu. Zmodyfikuj zadania wykonane w ćwiczeniach poprzednich, tak aby korzystać z bibliotek – obsługi liczników.

   a. Z wykorzystaniem licznika pracującego w trybie PWM wykonaj regulację jasności jej świecenia.

   b. Uruchom potencjometr analogowy z płytki rozszerzeń – wartość przetworzona w przetworniku AC wykorzystaj do regulacji PWM – jasności świecenia diody.

B. Szczegóły dotyczące stosowania różnych bibliotek i plików funkcji (AVRlibc, KAMAMI, Procyon): zostały opisane w rozdziale „Przykłady zakładania projektów".

C. Biblioteki i pliki dedykowanych zbiorów funkcji:

   a. Przejrzyj biblioteki w katalogu instalacyjnym np. ProgramFiles\Atmel\Studio\7.0\toolchain\avr8\avr8-gnu-toolchain\avr\include, dokumentacja znajduje się w sąsiednim katalogu doc.

   b. Do dyspozycji są także: zbiory dedykowanych dla AVR funkcji publikowane przez KAMAMI (jako producent zestawu dydaktycznego).

   c. Najbardziej rozbudowane biblioteki sprzętowe: Procyon (uniwersalne, dla różnych podtypów rodziny AVR, m.in. sterowniki peryferiów).

D. **Wskazówki**:

   a. Biblioteki i pliki funkcji wypakowujemy do wybranego katalogu (może znajdować się poza katalogiem domowym AVR Studio).

   b. Następnie dołączamy wybrane biblioteki / pliki funkcji metodą referencji do projektu (wybierz okno projektu – prawy przycisk i „Add" – „Existing item" - **********).

   c. Pliki nagłówkowe dopisujemy do głównego pliku nagłówkowego (jeśli używamy) lub do pliku „main.c".

d. W przypadku liczników / timerów – korzystamy z biblioteki …..

e. Należy koniecznie sprawdzić, czy prawidłowo została zadeklarowana stała częstotliwości zegara mikrokontrolera F_CPU dla bibliotek Proycon. Pozostałe istotne modyfikacje dla bibliotek Procyon opisano w rozdziale „Przykłady zakładania projektów”.

E. **Sprawozdanie**: załącz kluczową część programu – wyjaśnij jego działanie, uzupełnij i załącz komentarze w istotnych liniach kodu.

F. **Przygotować na następne zajęcia**: należy dokładnie zapoznać się z materiałami dotyczącymi języka C: operacje na łańcuchach oraz dokumentacją wyświetlacza OLED. Odpowiednie materiały zostały wymienione w instrukcji i są do pobrania w miejscu wskazanym przez Prowadzącego.

5.8. Ćwiczenie 8 (Tydzień 3). Język C - operacje na łańcuchach. Wyświetlacz.

A. Zapoznaj się z dokumentacją wyświetlacza OLED (do wyboru jest magistrala komunikacyjna, sugerowana jest I2C).

B. Z wykorzystaniem gotowych bibliotek napisz program i uruchom wyświetlanie krótkiego komunikatu na ekranie wyświetlacza.

a. Wykorzystaj co najmniej dwa łańcuchy tekstowe i zmienne liczbowe różnych typów.

b. Wykonaj operację połączenia dwóch łańcuchów w trzecim, z jednoczesnym drukowaniem i formatowaniem danych liczbowych.

c. Wykonaj operacje przeszukiwania wybranego ciągu tekstowego w celu znalezienia i zliczania wystąpień określonej frazy.

C. **Wskazówki**:

a. Płytka dydaktyczna jest przygotowana do pracy z wyświetlaczem, który wykorzystuje kontroler SSD1306 lub inny zgodny z nim. Na zajęciach stosujemy wyświetlacz sterowany kontrolerem SSD1306.

b. Podłączenie zostało wykonane do odpowiedniego portu mikrokontrolera na płytce rozszerzeń. Stosujemy magistralę xxxxxxxxxxxxx.

D. **Sprawozdanie**: załącz kluczową część programu – wyjaśnij jego działanie, uzupełnij i załącz komentarze w istotnych liniach kodu.

E. **Przygotować na następne zajęcia**: należy dokładnie zapoznać się z materiałami dotyczącymi języka C: obliczenia z wykorzystaniem różnych typów danych, operacje na liczbach stałoprzecinkowych i zmiennoprzecinkowych, błędy obliczeń dla liczb stałoprzecinkowych i zmiennoprzecinkowych. Przygotuj stablicowane dane dla obliczania funkcji sinus oraz wzory do

rozwinięcia funkcji sinus w szereg Taylora. Odpowiednie materiały zostały wymienione w instrukcji i są do pobrania w miejscu wskazanym przez Prowadzącego.

5.9. Ćwiczenie 9 (Tydzień 3). Język C – liczby stało i zmiennoprzecinkowe. Efekty optymalizacji kompilatora.

A. Dołącz wymagane biblioteki do obliczeń matematycznych.

B. Przygotuj program do:

   a. Obliczania promienia okręgu opisanego na trójkącie nierównoramiennym, o długościach ramion podawanych przez użytkownika.

   b. Wykonaj obliczenia stosując:

      i. Liczby typu char.

      ii. Liczby typu int lub long int.

      iii. Liczby zmiennoprzecinkowe.

   c. Jako funkcję obliczania wartości sinus wykorzystaj:

      i. Dane stablicowane.

      ii. Rozwinięcie funkcji w szereg Taylora (ew. Maclaurina).

      iii. Gotowe funkcje zmiennoprzecinkowe.

   d. Wykorzystaj zrealizowany wcześniej stoper (działający na przerwaniach) do pomiaru czasu wykonywania obliczeń, wynik wyświetlaj na OLED.

   e. Konieczny jest komentarz do dokładności obliczeń funkcji sinus.

C. Przygotuj pomiary bazujące na poprzednim ćwiczeniu:

   a. Zastosuj w opcjach kompilatora optymalizację kolejno: O0, O1, O2, O3, Os.

      i. Porównaj rozmiar i szybkość wykonania programów poddanych optymalizacji. Koniecznie wyciągnąć wnioski!

   b. Wstaw do programu zmienna dowolnego typu i w pętli głównej programu wykonuj jej inkrementacje.

      i. Poddaj tak przygotowany program debugowaniu przy kolejnych opcjach kompilacji. Podać wnioski co się dzieje z tą zmienna i jak tego zjawiska uniknać!

D. **Wskazówki**:

a. Biblioteki zmiennoprzecinkowe dołączamy biblioteką wbudowaną „math”.

b. Jako biblioteki stałoprzecinkowe można wykorzystać bibliotekę Procyon „fixedpt”.

E. **Sprawozdanie**: załącz kluczową część programu – wyjaśnij jego działanie, uzupełnij i załącz komentarze w istotnych liniach kodu.

F. **Przygotować na następne zajęcia**: należy dokładnie zapoznać się z materiałami dotyczącymi magistral SPI, I2C, 1-wire, port szeregowy UART oraz czujnika temperatury i zegara RTC. Odpowiednie materiały zostały wymienione w instrukcji i są do pobrania w miejscu wskazanym przez Prowadzącego.

5.10. Ćwiczenie 10 (Tydzień 4). Magistrale mikrokontrolera. SPI, I2C, port szeregowy UART – czujnik temperatury, zegar RTC

A. Dołącz wymagane biblioteki dla obsługi magistrali SPI, I2C, portu szeregowego, 1-wire.

B. Przygotuj programy:

a. Odczytywanie temperatury z czujnika DS18B20 z płytki rozszerzeń.

b. Obsługa zegara czasu rzeczywistego:

i. Odczytywanie i wyświetlanie bieżącego czasu.

ii. Zapisywanie stałej, z góry ustalonej wartości do zegara RTC.

iii. Pełna obsługa zegara RTC z możliwością ręcznego (przyciski) ustawiania czasu.

c. Wykorzystaj zrealizowany wcześniej zegar do realizacji stopera.

C. **Wskazówki**:

a. Biblioteki zmiennoprzecinkowe dołączamy biblioteką wbudowaną „math”.

b. Jako biblioteki stałoprzecinkowe można wykorzystać bibliotekę Procyon „fixedpt”.

D. **Sprawozdanie**: załącz kluczową część programu – wyjaśnij jego działanie, uzupełnij i załącz komentarze w istotnych liniach kodu.

E. **Przygotować na następne zajęcia**: należy dokładnie zapoznać się z materiałami dotyczącymi elektronicznego termometru joysticka, impulsatora i diody RGB. Odpowiednie materiały zostały wymienione w instrukcji i są do pobrania w miejscu wskazanym przez Prowadzącego.

5.11. Ćwiczenie 11 (Tydzień 4). Interfejs użytkownika. Joystick, impulsator, dioda RGB.

A. Dołącz wymagane biblioteki.

B. Przygotuj program:

   a. Obsługi joysticka – wyświetlanie dwóch zmiennych inkrementowanych / dekrementowanych danym kierunkiem joysticka.

   b. Obsługi impulsatora – zliczanie impulsów przy obrotach w lewo i prawo.

   c. Obsługa diody RGB – uzyskiwanie palety barw z regulacją za pomocą joysticka lub impulsatora.

C. **Wskazówki**:

   a. Biblioteki zmiennoprzecinkowe dołączamy biblioteką wbudowaną „math".

   b. Jako biblioteki stałoprzecinkowe można wykorzystać bibliotekę Procyon „fixedpt".

D. **Sprawozdanie**: załącz kluczową część programu – wyjaśnij jego działanie, uzupełnij i załącz komentarze w istotnych liniach kodu.

E. **Przygotować na następne zajęcia**: należy dokładnie zapoznać się z materiałami dotyczącymi wyświetlacza 7 segmentowego i wyświetlacza OLED. Odpowiednie materiały zostały wymienione w instrukcji i są do pobrania w miejscu wskazanym przez Prowadzącego.

5.12. Ćwiczenie 12 (Tydzień 4). Interfejs użytkownika. Wyświetlacz 7 segmentowy, graficzna obsługa wyświetlacza OLED.

A. Dołącz wymagane biblioteki.

B. Przygotuj program:

   a. Obsługi wyświetlacza 7 segmentowego. Wykorzystać go do np. wyświetlania czasu, stoper.

   b. Obsługi wyświetlacza OLED w celu pokazania na nim:

      i. Wykresu (np. funkcji trygonometrycznej)

      ii. Obrazu (przekonwertowanego np. z bitmapy).

      iii. Fraktala.

C. **Wskazówki**:

   a. Biblioteki zmiennoprzecinkowe dołączamy biblioteką wbudowaną „math".

   b. Jako biblioteki stałoprzecinkowe można wykorzystać bibliotekę Procyon „fixedpt".

D. **Sprawozdanie**: załącz kluczową część programu – wyjaśnij jego działanie, uzupełnij i załącz komentarze w istotnych liniach kodu.

E. **Przygotować na następne zajęcia**: należy dokładnie zapoznać się z materiałami dotyczącymi silnika DC. Odpowiednie materiały zostały wymienione w instrukcji i są do pobrania w miejscu wskazanym przez Prowadzącego.

5.13. Ćwiczenie 13 (Tydzień 5). Układy wykonawcze. Silnik DC

A. Dołącz wymagane biblioteki. Wykorzystaj zestaw Adafruit Motor/Stepper/Servo z odpowiednim silnikiem.

B. Przygotuj program:

   a. Sterowania silnikiem pełne obroty ze zmianą kierunku.

   b. Dodaj PWM do regulacji obrotów silnika – szybkość obrotów regulować np. z joysticka.

   c. UWAGA: jakie jest znaczenie częstotliwości podstawowej PWM? Wyjaśnić teoretycznie i przetestować eksperymentalnie.

C. **Wskazówki**:

   a. Biblioteki zmiennoprzecinkowe dołączamy biblioteką wbudowaną „math”.

   b. Jako biblioteki stałoprzecinkowe można wykorzystać bibliotekę Procyon „fixedpt”.

D. **Sprawozdanie**: załącz kluczową część programu – wyjaśnij jego działanie, uzupełnij i załącz komentarze w istotnych liniach kodu.

E. **Przygotować na następne zajęcia**: należy dokładnie zapoznać się z materiałami dotyczącymi silnika krokowego. Odpowiednie materiały zostały wymienione w instrukcji i są do pobrania w miejscu wskazanym przez Prowadzącego.

5.14. Ćwiczenie 14 (Tydzień 5). Układy wykonawcze. Silnik krokowy

A. Dołącz wymagane biblioteki. Wykorzystaj zestaw Adafruit Motor/Stepper/Servo z odpowiednim silnikiem.

G. Przygotuj program:

   a. Sterowania silnikiem krokowym w trybie bipolarnym lub unipolarnym - pełne obroty ze zmianą kierunku.

   b. Dodaj tryb pracy półkrokowej.

c. Zrealizuj sterowanie silnika przez podawanie w funkcji liczby kroków, które silnik powinien wykonać.

d. UWAGA: jakie jest znaczenie częstotliwości podawania impulsów do silnika? Wyjaśnić teoretycznie i przetestować eksperymentalnie.

H. **Wskazówki**:

a. Biblioteki zmiennoprzecinkowe dołączamy biblioteką wbudowaną „math".

b. Jako biblioteki stałoprzecinkowe można wykorzystać bibliotekę Procyon „fixedpt".

I. **Sprawozdanie**: załącz kluczową część programu – wyjaśnij jego działanie, uzupełnij i załącz komentarze w istotnych liniach kodu.

J. **Przygotować na następne zajęcia**: należy dokładnie zapoznać się z materiałami dotyczącymi serwomechanizmów. Odpowiednie materiały zostały wymienione w instrukcji i są do pobrania w miejscu wskazanym przez Prowadzącego.

5.15. Ćwiczenie 15 (Tydzień 5). Układy wykonawcze. Serwomechanizm

A. Dołącz wymagane biblioteki. Wykorzystaj zestaw Adafruit Motor/Stepper/Servo z odpowiednim silnikiem.

B. Przygotuj program:

a. Sterowania serwem zgodnie z ogólnymi zasadami sterowania tego typu urządzeniami wykonawczymi.

b. Dodania regulacji położenia serwa z joysticka.

c. UWAGA: jakie jest znaczenie częstotliwości podawania impulsów do serwa? Wyjaśnić teoretycznie i przetestować eksperymentalnie.

K. **Wskazówki**:

a. Biblioteki zmiennoprzecinkowe dołączamy biblioteką wbudowaną „math".

b. Jako biblioteki stałoprzecinkowe można wykorzystać bibliotekę Procyon „fixedpt".

L. **Sprawozdanie**: załącz kluczową część programu – wyjaśnij jego działanie, uzupełnij i załącz komentarze w istotnych liniach kodu.

M. **Przygotować na następne zajęcia**: należy dokładnie zapoznać się z materiałami dotyczącymi czujnika wykrywania przeszkód oraz obwodu pomiaru przebytej drogi. Odpowiednie materiały zostały wymienione w instrukcji i są do pobrania w miejscu wskazanym przez Prowadzącego.

5.16. Ćwiczenie 16 (Tydzień 6). Układy pomiarowe. Wykrywanie przeszkód – czujnik podczerwieni. Pomiar przebytej drogi.

A. Dołącz wymagane biblioteki . Wykorzystaj zestaw AlphaBot podwozie sterowane silnikami oraz czujnik wykrywania przeszkód załączony do zestawu.

B. Przygotuj program:

   a. Wykrywania przeszkody statycznie.

   b. Połączenia działania silników AlphaBota – z zatrzymywaniem ich w momencie wykrycia przeszkody.

   c. Dla chętnych – skręcanie podwoziem w celu kontynuacji jazdy po wykryciu przeszkody.

   d. Używając wbudowanych w AlphaBota czujników obrotów kół zliczaj przebytą drogą i wyświetl ją w dowolny, ale zrozumiały sposób.

C. **Wskazówki**:

   a. Biblioteki zmiennoprzecinkowe dołączamy biblioteką wbudowaną „math”.

   b. Jako biblioteki stałoprzecinkowe można wykorzystać bibliotekę Procyon „fixedpt”.

D. **Sprawozdanie**: załącz kluczową część programu – wyjaśnij jego działanie, uzupełnij i załącz komentarze w istotnych liniach kodu.

E. **Przygotować na następne zajęcia**: należy dokładnie zapoznać się z materiałami dotyczącymi układu śledzenia linii. Odpowiednie materiały zostały wymienione w instrukcji i są do pobrania w miejscu wskazanym przez Prowadzącego.

5.17. Ćwiczenie 17 (Tydzień 6). Układy pomiarowe. Śledzenie linii.

A. Dołącz wymagane biblioteki . Wykorzystaj zestaw AlphaBot podwozie sterowane silnikami oraz czujnik śledzenia linii załączony do zestawu.

B. Przygotuj program:

   a. Wykrywania linii statycznie.

   b. Połączenia działania silników AlphaBota – z nadążaniem za wykrywaną linią.

   c. Dla chętnych – poszukiwanie podwoziem w celu kontynuacji jazdy po utracie wykrycia linii.

C. **Wskazówki**:

   a. Biblioteki zmiennoprzecinkowe dołączamy biblioteką wbudowaną „math”.

b. Jako biblioteki stałoprzecinkowe można wykorzystać bibliotekę Procyon „fixedpt”.

D. **Sprawozdanie**: załącz kluczową część programu – wyjaśnij jego działanie, uzupełnij i załącz komentarze w istotnych liniach kodu.

E. **Przygotować na następne zajęcia**: należy dokładnie zapoznać się z materiałami dotyczącymi ultradźwiękowego czujnika odległości. Odpowiednie materiały zostały wymienione w instrukcji i są do pobrania w miejscu wskazanym przez Prowadzącego.

5.18. Ćwiczenie 18 (Tydzień 6). Układy pomiarowe. Wykrywanie przeszkód – czujnik ultradźwiękowy odległości.

A. Dołącz wymagane biblioteki . Wykorzystaj zestaw AlphaBot podwozie sterowane silnikami oraz czujnik ultradźwiękowy załączony do zestawu.

B. Przygotuj program:

a. Wykrywania przeszkody statycznie.

b. Połączenia działania silników AlphaBota – z zatrzymywaniem ich w momencie wykrycia przeszkody.

c. Dla chętnych – skręcanie podwoziem w celu kontynuacji jazdy po wykryciu przeszkody.

C. **Wskazówki**:

a. Biblioteki zmiennoprzecinkowe dołączamy biblioteką wbudowaną „math”.

b. Jako biblioteki stałoprzecinkowe można wykorzystać bibliotekę Procyon „fixedpt”.

D. **Sprawozdanie**: załącz kluczową część programu – wyjaśnij jego działanie, uzupełnij i załącz komentarze w istotnych liniach kodu.

E. **Przygotować na następne zajęcia**: należy dokładnie zapoznać się z materiałami dotyczącymi czujników różnych typów – zgodnie ze wskazaniami Prowadzącego. Odpowiednie materiały zostały wymienione w instrukcji i są do pobrania w miejscu wskazanym przez Prowadzącego.

5.19. Ćwiczenie 19/20/21 (Tydzień 7). Czujniki różnych typów, czujniki specjalne.

A. Dołącz wymagane biblioteki . Wykorzystaj zestaw 13 czujników. Zgodnie z wyborem prowadzącego przygotuj obsługę przynajmniej 3 różnych typów czujników.

B. Czujniki do wyboru:

   a. Obrotowy enkoder (zestaw 13)
   b. Czujnik temperatury i wilgotności (zestaw 13)
   c. Czujniki oświetlenia: koloru + płomieni + UV(zestaw 13)
   d. Czujnik dźwięku (zestaw 13)
   e. Akcelerometr 3 osie (płytki rozszerzeń)
   f. Odbiciowy miernik podczerwieni (zestaw 13)
   g. Laserowy czujnik odległości (zestaw 13)
   h. Czujnik wychylenia (zestaw 13)
   i. Żyroskop (brak)
   j. Kompas elektroniczny (brak)
   k. GPS (brak)
   l. Czujnik Halla (zestaw 13)
   m. Czujnik indukcyjny (do samodzielnego wykonania)
   a. Z zestawu 13 nie korzystamy z: gazu, wilgotności gleby i poziomu cieczy

C. Obsługa czujnika powinna polegać na nie tylko nawiązaniu z nim prawidłowej komunikacji, ale także przygotowania / przeliczenia wyników i wyświetlania ich w poprawnej i atrakcyjnej formie.

D. **Wskazówki**:

   a. Biblioteki zmiennoprzecinkowe dołączamy biblioteką wbudowaną „math”.
   b. Jako biblioteki stałoprzecinkowe można wykorzystać bibliotekę Procyon „fixedpt”.

E. **Sprawozdanie**: załącz kluczową część programu – wyjaśnij jego działanie, uzupełnij i załącz komentarze w istotnych liniach kodu.

F. **Przygotować na następne zajęcia**: należy dokładnie zapoznać się z materiałami dotyczącymi komunikacji międzyprocesorowej – w szczególności interfejsu UART. Odpowiednie materiały zostały wymienione w instrukcji i są do pobrania w miejscu wskazanym przez Prowadzącego.

5.20. Ćwiczenie 22/23/24 (Tydzień 8). Komunikacja.

A. Dołącz wymagane biblioteki.

B. Przygotuj program:

a. Komunikacji międzyprocesorowej. Dwie grupy studenckie powinny przygotować program do wymiany danych miedzy swoimi płytkami. Przesyłane dane mogą pochodzić np. z wybranego czujnika. Zalecane wykorzystanie portu szeregowego.

b. Rozszerzenia komunikacji z poprzedniego punktu przez dodanie interfejsu bezprzewodowego np. podczerwieni (nadal z wykorzystaniem portu szeregowego).

c. Dla chętnych – przesyłanie danych interfejsem Bluetooth (nadal z wykorzystaniem portu szeregowego).

C. **Wskazówki**:

a. Biblioteki zmiennoprzecinkowe dołączamy biblioteką wbudowaną „math".

b. Jako biblioteki stałoprzecinkowe można wykorzystać bibliotekę Procyon „fixedpt".

D. **Sprawozdanie**: załącz kluczową część programu – wyjaśnij jego działanie, uzupełnij i załącz komentarze w istotnych liniach kodu.

E. **Przygotować na następne zajęcia**: należy dokładnie zapoznać się z materiałami dotyczącymi utrwalania danych w pamięci Flash, EEPROM i na karcie SD. Odpowiednie materiały zostały wymienione w instrukcji i są do pobrania w miejscu wskazanym przez Prowadzącego.

5.21. Ćwiczenie 25/26/27 (Tydzień 9). Zachowywanie i graficzna wizualizacja danych. Pamięć Flash / EEPROM, karta SD, ekran graficzny.

A. Dołącz wymagane biblioteki. Wykorzystaj w ćwiczeniu ekran graficzny 2,8 cala.

B. Przygotuj program:

a. Zbierania danych pomiarowych np. czujnika temperatury / wilgotności itp. I zapisywania ich w pamięci EEPROM / Flash mikrokontrolera.

b. Zapis zebranych danych na karcie SD.

c. Dla chętnych – wyświetlanie zebranych danych w przyjaznej formie na ekranie dotykowym. Po dotknięciu ekranu wykonanie powiększenia wykresu.

C. **Wskazówki**:

a. Biblioteki zmiennoprzecinkowe dołączamy biblioteką wbudowaną „math".

b. Jako biblioteki stałoprzecinkowe można wykorzystać bibliotekę Procyon „fixedpt".

D. **Sprawozdanie**: załącz kluczową część programu – wyjaśnij jego działanie, uzupełnij i załącz komentarze w istotnych liniach kodu.

E. **Przygotować na następne zajęcia**: należy dokładnie zapoznać się z materiałami dotyczącymi wymogów Prowadzącego dotyczących projektu z AlphaBotem. Odpowiednie materiały zostały wymienione w instrukcji i są do pobrania w miejscu wskazanym przez Prowadzącego.

5.22. Ćwiczenie 28/29 (Tydzień 10). Projekt

A. Wykonaj projekt – AlphaBot jako „światłolub” albo „dźwiękolub” (lub odwrotnie np. „światłowstręt”).

B. Założeniem projektu jest pełne sterowanie ruchem pojazdu zgodnie z wybranym sposobem „czułości na otoczenie”.

C. **Sprawozdanie**: załącz kluczową część programu – wyjaśnij jego działanie, uzupełnij i załącz komentarze w istotnych liniach kodu.

D. **Przygotować na następne zajęcia**: sprawdzić u Prowadzącego czy nie ma zaległych, niezaliczonych ćwiczeń lub istnieje konieczność odrobienia zajęć.

5.23. Ćwiczenie 30 (Tydzień 10). Zaległości, zaliczenie

A. Zaliczenia.

B. Termin odróbkowy na wykonanie zaległych ćwiczeń lub usprawiedliwionych braków obecności.

6. Zasady wykonania sprawozdania i oceny

6.1. Sprawozdanie

Ćwiczenia powinny być dokładnie opisane i kompletne. Do każdego wykonanego punktu wymagane jest zamieszczenie w sprawozdaniu:

- zamierzony cel ćwiczenia,
- kroki w realizacji, schematy, obliczenia,
- uzyskane wyniki,
- interpretacja wyniku,
- wnioski, podsumowanie.

6.2. Zasady oceny

Ujednolicenie oceny wystawianej studentom na podstawie przebiegu zajęć i sprawozdania - stosujemy następujące kryteria oceniania studentów:

1. Na ocenę 4 (proponuję przyjęcie oceny 4 za punkt odniesienia):

- brak zaległości i nieobecności (usprawiedliwione),
- sprawozdanie oddane w terminie (koniec pierwszego tygodnia sesji) kompletne i starannie wykonane (wykonanie wszystkich wymaganych ćwiczeń),
- każde ćwiczenie opatrzone krótkim opisem i sformułowaniem problemu,
- obecność rozsądnie dobranych wyników, wykresów (nie ich nadmiar), umiejętność wykazywania i rozstrzygania jednym wynikiem / wykresem wymaganych w danym punkcie ćwiczenia problemów,
- poprawne i przejrzyste opisy wyników / podpisy wykresów, poprawnie dobrane skale oraz krok analizy,
- krótkie i przejrzyste WNIOSKI (a nie tylko opis wyników/wykresów).

2. Na ocenę 2 (odrzucenie sprawozdania do poprawy):

- nieusprawiedliwione obecności (max. 2 dopuszczalne), brak aktywności na zajęciach,
- sprawozdanie opóźnione, niekompletne (braki w ćwiczeniach),
- brak informacji o sposobie wykonania ćwiczenia, brak sformułowania problemu,
- nadmiar lub braki wyników, wykresów, chaotyczny ich dobór,
- braki w opisach wyników, wykresów, brak staranności w doborze skali, nieprawidłowy dobór kroku analizy,
- brak faktycznych wniosków (tylko opis wyników, wykresów).

3. Na ocenę 3:

- "stan pośredni" między oceną 2 i 4, czyli ćwiczenia wykonane ze zróżnicowaną jakością, drobne braki.

4. Na ocenę 5:

- sprawozdania jednoznacznie wyróżniające się, czyli oprócz wszelkich wymogów na ocenę 4 - interesujące wnioski, staranne wykonanie,
- aktywność studenta na zajęciach,
- poprawnie wykonane projekty dodatkowe do każdego ćwiczenia.

7. Literatura

1.	Mikrokontrolery AVR - niezbędnik programisty Autor: Jarosław Doliński ISBN: 978-83-60233-47-4 Format: B6, 134 str. Wydawnictwo BTC Legionowo - wydanie papierowe 2009 r.
2.	Mikrokontrolery AVR ATmega w praktyce Autor: Rafał Baranowski ISBN: 83-60233-02-0 Format: B5, 390 str. Wydawnictwo BTC Warszawa 2005
3.	Mikrokontrolery AVR w praktyce Autor: Jarosław Doliński ISBN: 83-910067-6-X Format: B5, 452 str. Wydawnictwo BTC Warszawa 2004
4.	Mikrokontrolery dla początkujących Autor: Piotr Górecki ISBN: 83-60233-06-3 Format: B5, 408 str. Wydawnictwo BTC Warszawa 2006
5.	Projektowanie systemów mikroprocesorowych Autor: Paweł Hadam ISBN: 83-910067-9-4 Format: B5, 216 str. Wydawnictwo BTC

	Warszawa 2004
6.	RS232 w przykładach na PC i AVR Autor: Rafał Chromik ISBN: 978-83-60233-59-7 Format: B5, 168 str. Wydawnictwo BTC Legionowo 2010
7.	Sztuka programowania mikrokontrolerów AVR - przykłady Autor: Andrzej Pawluczuk ISBN: 978-83-60233-21-4 Format: B5, 293 str. Wydawnictwo BTC Warszawa 2007
8.	Wyświetlacze graficzne i alfanumeryczne w systemach mikroprocesorowych Autor: Rafał Baranowski ISBN: 978-83-60233-34-4 Format: B5, 176 str. Wydawnictwo BTC Legionowo 2008
9.	Mikrokontrolery AVR. Język C - podstawy programowania. Wydanie II poprawione i uzupełnione Autor: Mirosław Kardaś Wydawnictwo: Wydawnictwo ATNEL
10.	AVR. Praktyczne projekty Autor: Tomasz Francuz Wydawnictwo: Helion
11.	AVR. Układy peryferyjne Autor: Tomasz Francuz Wydawnictwo: Helion
12.	Programowanie układów AVR dla praktyków Autor: Elliot Williams Wydawnictwo: Helion
13.	Język C dla mikrokontrolerów AVR. Od podstaw do zaawansowanych aplikacji. Wydanie II Także wersja ebook Autor: Tomasz Francuz Wydawnictwo: Helion

8. Załączniki

8.1. ATmega328P – rejestry i instrukcje

30. Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0xFF)	Reserved	–	–	–	–	–	–	–	–	
(0xFE)	Reserved	–	–	–	–	–	–	–	–	
(0xFD)	Reserved	–	–	–	–	–	–	–	–	
(0xFC)	Reserved	–	–	–	–	–	–	–	–	
(0xFB)	Reserved	–	–	–	–	–	–	–	–	
(0xFA)	Reserved	–	–	–	–	–	–	–	–	
(0xF9)	Reserved	–	–	–	–	–	–	–	–	
(0xF8)	Reserved	–	–	–	–	–	–	–	–	
(0xF7)	Reserved	–	–	–	–	–	–	–	–	
(0xF6)	Reserved	–	–	–	–	–	–	–	–	
(0xF5)	Reserved	–	–	–	–	–	–	–	–	
(0xF4)	Reserved	–	–	–	–	–	–	–	–	
(0xF3)	Reserved	–	–	–	–	–	–	–	–	
(0xF2)	Reserved	–	–	–	–	–	–	–	–	
(0xF1)	Reserved	–	–	–	–	–	–	–	–	
(0xF0)	Reserved	–	–	–	–	–	–	–	–	
(0xEF)	Reserved	–	–	–	–	–	–	–	–	
(0xEE)	Reserved	–	–	–	–	–	–	–	–	
(0xED)	Reserved	–	–	–	–	–	–	–	–	
(0xEC)	Reserved	–	–	–	–	–	–	–	–	
(0xEB)	Reserved	–	–	–	–	–	–	–	–	
(0xEA)	Reserved	–	–	–	–	–	–	–	–	
(0xE9)	Reserved	–	–	–	–	–	–	–	–	
(0xE8)	Reserved	–	–	–	–	–	–	–	–	
(0xE7)	Reserved	–	–	–	–	–	–	–	–	
(0xE6)	Reserved	–	–	–	–	–	–	–	–	
(0xE5)	Reserved	–	–	–	–	–	–	–	–	
(0xE4)	Reserved	–	–	–	–	–	–	–	–	
(0xE3)	Reserved	–	–	–	–	–	–	–	–	
(0xE2)	Reserved	–	–	–	–	–	–	–	–	
(0xE1)	Reserved	–	–	–	–	–	–	–	–	
(0xE0)	Reserved	–	–	–	–	–	–	–	–	
(0xDF)	Reserved	–	–	–	–	–	–	–	–	
(0xDE)	Reserved	–	–	–	–	–	–	–	–	
(0xDD)	Reserved	–	–	–	–	–	–	–	–	
(0xDC)	Reserved	–	–	–	–	–	–	–	–	
(0xDB)	Reserved	–	–	–	–	–	–	–	–	
(0xDA)	Reserved	–	–	–	–	–	–	–	–	

Notes:
1. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
2. I/O registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions.
3. Some of the status flags are cleared by writing a logical one to them. Note that, unlike most other AVR®, the CBI and SBI instructions will only operate on the specified bit, and can therefore be used on registers containing such status flags. The CBI and SBI instructions work with registers 0x00 to 0x1F only.
4. When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O registers as data space using LD and ST instructions, 0x20 must be added to these addresses. The ATmega328P is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in opcode for the IN and OUT instructions. For the extended I/O space from 0x60 - 0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

30. Register Summary (Continued)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0xD9)	Reserved	–	–	–	–	–	–	–	–	
(0xD8)	Reserved	–	–	–	–	–	–	–	–	
(0xD7)	Reserved	–	–	–	–	–	–	–	–	
(0xD6)	Reserved	–	–	–	–	–	–	–	–	
(0xD5)	Reserved	–	–	–	–	–	–	–	–	
(0xD4)	Reserved	–	–	–	–	–	–	–	–	
(0xD3)	Reserved	–	–	–	–	–	–	–	–	
(0xD2)	Reserved	–	–	–	–	–	–	–	–	
(0xD1)	Reserved	–	–	–	–	–	–	–	–	
(0xD0)	Reserved	–	–	–	–	–	–	–	–	
(0xCF)	Reserved	–	–	–	–	–	–	–	–	
(0xCE)	Reserved	–	–	–	–	–	–	–	–	
(0xCD)	Reserved	–	–	–	–	–	–	–	–	
(0xCC)	Reserved	–	–	–	–	–	–	–	–	
(0xCB)	Reserved	–	–	–	–	–	–	–	–	
(0xCA)	Reserved	–	–	–	–	–	–	–	–	
(0xC9)	Reserved	–	–	–	–	–	–	–	–	
(0xC8)	Reserved	–	–	–	–	–	–	–	–	
(0xC7)	Reserved	–	–	–	–	–	–	–	–	
(0xC6)	UDR0	USART I/O data register								159
(0xC5)	UBRR0H					USART baud rate register high				162
(0xC4)	UBRR0L	USART baud rate register low								162
(0xC3)	Reserved	–	–	–	–	–	–	–	–	
(0xC2)	UCSR0C	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01 /UDORD0	UCSZ00 / UCPHA0	UCPOL0	161/172
(0xC1)	UCSR0B	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80	160
(0xC0)	UCSR0A	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0	159
(0xBF)	Reserved	–	–	–	–	–	–	–	–	
(0xBE)	Reserved	–	–	–	–	–	–	–	–	
(0xBD)	TWAMR	TWAM6	TWAM5	TWAM4	TWAM3	TWAM2	TWAM1	TWAM0	–	201
(0xBC)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	199
(0xBB)	TWDR	2-wire serial interface data register								200
(0xBA)	TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	201
(0xB9)	TWSR	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0	200
(0xB8)	TWBR	2-wire serial interface bit rate register								198
(0xB7)	Reserved	–		–	–	–	–	–	–	
(0xB6)	ASSR	–	EXCLK	AS2	TCN2UB	OCR2AUB	OCR2BUB	TCR2AUB	TCR2BUB	133
(0xB5)	Reserved	–	–	–	–	–	–	–	–	

Notes:
1. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
2. I/O registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions.
3. Some of the status flags are cleared by writing a logical one to them. Note that, unlike most other AVR®, the CBI and SBI instructions will only operate on the specified bit, and can therefore be used on registers containing such status flags. The CBI and SBI instructions work with registers 0x00 to 0x1F only.
4. When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O registers as data space using LD and ST instructions, 0x20 must be added to these addresses. The ATmega328P is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in opcode for the IN and OUT instructions. For the extended I/O space from 0x60 - 0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

30. Register Summary (Continued)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0xB4)	OCR2B	Timer/Counter2 output compare register B								131
(0xB3)	OCR2A	Timer/Counter2 output compare register A								131
(0xB2)	TCNT2	Timer/Counter2 (8-bit)								131
(0xB1)	TCCR2B	FOC2A	FOC2B	–	–	WGM22	CS22	CS21	CS20	130
(0xB0)	TCCR2A	COM2A1	COM2A0	COM2B1	COM2B0	–	–	WGM21	WGM20	127
(0xAF)	Reserved	–	–	–	–	–	–	–	–	
(0xAE)	Reserved	–	–	–	–	–	–	–	–	
(0xAD)	Reserved	–	–	–	–	–	–	–	–	
(0xAC)	Reserved	–	–	–	–	–	–	–	–	
(0xAB)	Reserved	–	–	–	–	–	–	–	–	
(0xAA)	Reserved	–	–	–	–	–	–	–	–	
(0xA9)	Reserved	–	–	–	–	–	–	–	–	
(0xA8)	Reserved	–	–	–	–	–	–	–	–	
(0xA7)	Reserved	–	–	–	–	–	–	–	–	
(0xA6)	Reserved	–	–	–	–	–	–	–	–	
(0xA5)	Reserved	–	–	–	–	–	–	–	–	
(0xA4)	Reserved	–	–	–	–	–	–	–	–	
(0xA3)	Reserved	–	–	–	–	–	–	–	–	
(0xA2)	Reserved	–	–	–	–	–	–	–	–	
(0xA1)	Reserved	–	–	–	–	–	–	–	–	
(0xA0)	Reserved	–	–	–	–	–	–	–	–	
(0x9F)	Reserved	–	–	–	–	–	–	–	–	
(0x9E)	Reserved	–	–	–	–	–	–	–	–	
(0x9D)	Reserved	–	–	–	–	–	–	–	–	
(0x9C)	Reserved	–	–	–	–	–	–	–	–	
(0x9B)	Reserved	–	–	–	–	–	–	–	–	
(0x9A)	Reserved	–	–	–	–	–	–	–	–	
(0x99)	Reserved	–	–	–	–	–	–	–	–	
(0x98)	Reserved	–	–	–	–	–	–	–	–	
(0x97)	Reserved	–	–	–	–	–	–	–	–	
(0x96)	Reserved	–	–	–	–	–	–	–	–	
(0x95)	Reserved	–	–	–	–	–	–	–	–	
(0x94)	Reserved	–	–	–	–	–	–	–	–	
(0x93)	Reserved	–	–	–	–	–	–	–	–	
(0x92)	Reserved	–	–	–	–	–	–	–	–	
(0x91)	Reserved	–	–	–	–	–	–	–	–	
(0x90)	Reserved	–	–	–	–	–	–	–	–	
(0x8F)	Reserved	–	–	–	–	–	–	–	–	

Notes:
1. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
2. I/O registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions.
3. Some of the status flags are cleared by writing a logical one to them. Note that, unlike most other AVR®, the CBI and SBI instructions will only operate on the specified bit, and can therefore be used on registers containing such status flags. The CBI and SBI instructions work with registers 0x00 to 0x1F only.
4. When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O registers as data space using LD and ST instructions, 0x20 must be added to these addresses. The ATmega328P is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in opcode for the IN and OUT instructions. For the extended I/O space from 0x60 - 0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

30. Register Summary (Continued)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0x8E)	Reserved	–	–	–	–	–	–	–	–	
(0x8D)	Reserved	–	–	–	–	–	–	–	–	
(0x8C)	Reserved	–	–	–	–	–	–	–	–	
(0x8B)	OCR1BH	Timer/Counter1 - Output compare register B high byte								111
(0x8A)	OCR1BL	Timer/Counter1 - Output compare register B low byte								111
(0x89)	OCR1AH	Timer/Counter1 - Output compare register A high byte								111
(0x88)	OCR1AL	Timer/Counter1 - Output compare register A low byte								111
(0x87)	ICR1H	Timer/Counter1 - Input capture register high byte								112
(0x86)	ICR1L	Timer/Counter1 - Input capture register low byte								112
(0x85)	TCNT1H	Timer/Counter1 - Counter register high byte								111
(0x84)	TCNT1L	Timer/Counter1 - Counter register low byte								111
(0x83)	Reserved	–	–	–	–	–	–	–	–	
(0x82)	TCCR1C	FOC1A	FOC1B	–	–	–	–	–	–	111
(0x81)	TCCR1B	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	110
(0x80)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	108
(0x7F)	DIDR1	–	–	–	–	–	–	AIN1D	AIN0D	204
(0x7E)	DIDR0	–	–	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D	220
(0x7D)	Reserved	–	–	–	–	–	–	–	–	
(0x7C)	ADMUX	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	217
(0x7B)	ADCSRB	–	ACME	–	–	–	ADTS2	ADTS1	ADTS0	220
(0x7A)	ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	218
(0x79)	ADCH	ADC data register high byte								219
(0x78)	ADCL	ADC data register low byte								219
(0x77)	Reserved	–	–	–	–	–	–	–	–	
(0x76)	Reserved	–	–	–	–	–	–	–	–	
(0x75)	Reserved	–	–	–	–	–	–	–	–	
(0x74)	Reserved	–	–	–	–	–	–	–	–	
(0x73)	Reserved	–	–	–	–	–	–	–	–	
(0x72)	Reserved	–	–	–	–	–	–	–	–	
(0x71)	Reserved	–	–	–	–	–	–	–	–	
(0x70)	TIMSK2	–	–	–	–	–	OCIE2B	OCIE2A	TOIE2	132
(0x6F)	TIMSK1	–	–	ICIE1	–	–	OCIE1B	OCIE1A	TOIE1	112
(0x6E)	TIMSK0	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	88
(0x6D)	PCMSK2	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16	57
(0x6C)	PCMSK1	–	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8	57
(0x6B)	PCMSK0	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	57
(0x6A)	Reserved	–	–	–	–	–	–	–	–	
(0x69)	EICRA	–	–	–	–	ISC11	ISC10	ISC01	ISC00	54

Notes:
1. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
2. I/O registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions.
3. Some of the status flags are cleared by writing a logical one to them. Note that, unlike most other AVR®, the CBI and SBI instructions will only operate on the specified bit, and can therefore be used on registers containing such status flags. The CBI and SBI instructions work with registers 0x00 to 0x1F only.
4. When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O registers as data space using LD and ST instructions, 0x20 must be added to these addresses. The ATmega328P is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in opcode for the IN and OUT instructions. For the extended I/O space from 0x60 - 0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

30. Register Summary (Continued)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0x68)	PCICR	–	–	–	–	–	PCIE2	PCIE1	PCIE0	
(0x67)	Reserved	–	–	–	–	–	–	–	–	
(0x66)	OSCCAL	Oscillator calibration register								32
(0x65)	Reserved	–	–	–	–	–	–	–	–	
(0x64)	PRR	PRTWI	PRTIM2	PRTIM0	–	PRTIM1	PRSPI	PRUSAR0	PRADC	36
(0x63)	Reserved	–	–	–	–	–	–	–	–	
(0x62)	Reserved	–	–	–	–	–	–	–	–	
(0x61)	CLKPR	CLKPCE	–	–	–	CLKPS3	CLKPS2	CLKPS1	CLKPS0	33
(0x60)	WDTCSR	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	47
0x3F (0x5F)	SREG	I	T	H	S	V	N	Z	C	10
0x3E (0x5E)	SPH	–	–	–	–	–	(SP10)	SP9	SP8	13
0x3D (0x5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	13
0x3C (0x5C)	Reserved	–	–	–	–	–	–	–	–	
0x3B (0x5B)	Reserved	–	–	–	–	–	–	–	–	
0x3A (0x5A)	Reserved	–	–	–	–	–	–	–	–	
0x39 (0x59)	Reserved	–	–	–	–	–	–	–	–	
0x38 (0x58)	Reserved	–	–	–	–	–	–	–	–	
0x37 (0x57)	SPMCSR	SPMIE	(RWWSB)	–	(RWWSRE)	BLBSET	PGWRT	PGERS	SELFPRGN	239
0x36 (0x56)	Reserved	–	–	–	–	–	–	–	–	
0x35 (0x55)	MCUCR	–	BODS	BODSE	PUD	–	–	IVSEL	IVCE	38/52/72
0x34 (0x54)	MCUSR	–	–	–	–	WDRF	BORF	EXTRF	PORF	46
0x33 (0x53)	SMCR	–	–	–	–	SM2	SM1	SM0	SE	35
0x32 (0x52)	Reserved	–	–	–	–	–	–	–	–	
0x31 (0x51)	Reserved	–	–	–	–	–	–	–	–	
0x30 (0x50)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	203
0x2F (0x4F)	Reserved	–	–	–	–	–	–	–	–	
0x2E (0x4E)	SPDR	SPI data register								142
0x2D (0x4D)	SPSR	SPIF	WCOL	–	–	–	–	–	SPI2X	141
0x2C (0x4C)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	140
0x2B (0x4B)	GPIOR2	General purpose I/O register 2								23
0x2A (0x4A)	GPIOR1	General purpose I/O register 1								23
0x29 (0x49)	Reserved	–	–	–	–	–	–	–	–	
0x28 (0x48)	OCR0B	Timer/Counter0 output compare register B								
0x27 (0x47)	OCR0A	Timer/Counter0 output compare register A								
0x26 (0x46)	TCNT0	Timer/Counter0 (8-bit)								
0x25 (0x45)	TCCR0B	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	
0x24 (0x44)	TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	
0x23 (0x43)	GTCCR	TSM	–	–	–	–	–	PSRASY	PSRSYNC	115/134

Notes:
1. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
2. I/O registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions.
3. Some of the status flags are cleared by writing a logical one to them. Note that, unlike most other AVR®, the CBI and SBI instructions will only operate on the specified bit, and can therefore be used on registers containing such status flags. The CBI and SBI instructions work with registers 0x00 to 0x1F only.
4. When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O registers as data space using LD and ST instructions, 0x20 must be added to these addresses. The ATmega328P is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in opcode for the IN and OUT instructions. For the extended I/O space from 0x60 - 0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

30. Register Summary (Continued)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x22 (0x42)	EEARH	(EEPROM address register high byte)								20
0x21 (0x41)	EEARL	EEPROM address register low byte								20
0x20 (0x40)	EEDR	EEPROM data register								20
0x1F (0x3F)	EECR	–	–	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE	20
0x1E (0x3E)	GPIOR0	General purpose I/O register 0								23
0x1D (0x3D)	EIMSK	–	–	–	–	–	–	INT1	INT0	55
0x1C (0x3C)	EIFR	–	–	–	–	–	–	INTF1	INTF0	55
0x1B (0x3B)	PCIFR	–	–	–	–	–	PCIF2	PCIF1	PCIF0	
0x1A (0x3A)	Reserved	–	–	–	–	–	–	–	–	
0x19 (0x39)	Reserved	–	–	–	–	–	–	–	–	
0x18 (0x38)	Reserved	–	–	–	–	–	–	–	–	
0x17 (0x37)	TIFR2	–	–	–	–	–	OCF2B	OCF2A	TOV2	132
0x16 (0x36)	TIFR1	–	–	ICF1	–	–	OCF1B	OCF1A	TOV1	113
0x15 (0x35)	TIFR0	–	–	–	–	–	OCF0B	OCF0A	TOV0	
0x14 (0x34)	Reserved	–	–	–	–	–	–	–	–	
0x13 (0x33)	Reserved	–	–	–	–	–	–	–	–	
0x12 (0x32)	Reserved	–	–	–	–	–	–	–	–	
0x11 (0x31)	Reserved	–	–	–	–	–	–	–	–	
0x10 (0x30)	Reserved	–	–	–	–	–	–	–	–	
0x0F (0x2F)	Reserved	–	–	–	–	–	–	–	–	
0x0E (0x2E)	Reserved	–	–	–	–	–	–	–	–	
0x0D (0x2D)	Reserved	–	–	–	–	–	–	–	–	
0x0C (0x2C)	Reserved	–	–	–	–	–	–	–	–	
0x0B (0x2B)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	73
0x0A (0x2A)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	73
0x09 (0x29)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	73
0x08 (0x28)	PORTC	–	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	73
0x07 (0x27)	DDRC	–	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	73
0x06 (0x26)	PINC	–	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	73
0x05 (0x25)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	72
0x04 (0x24)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	72
0x03 (0x23)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	72
0x02 (0x22)	Reserved	–	–	–	–	–	–	–	–	
0x01 (0x21)	Reserved	–	–	–	–	–	–	–	–	
0x0 (0x20)	Reserved	–	–	–	–	–	–	–	–	

Notes:
1. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
2. I/O registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions.
3. Some of the status flags are cleared by writing a logical one to them. Note that, unlike most other AVR®, the CBI and SBI instructions will only operate on the specified bit, and can therefore be used on registers containing such status flags. The CBI and SBI instructions work with registers 0x00 to 0x1F only.
4. When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O registers as data space using LD and ST instructions, 0x20 must be added to these addresses. The ATmega328P is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in opcode for the IN and OUT instructions. For the extended I/O space from 0x60 - 0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

31. Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks
Arithmetic and Logic Instructions					
ADD	Rd, Rr	Add two registers	Rd ← Rd + Rr	Z,C,N,VH	1
ADC	Rd, Rr	Add with carry two registers	Rd ← Rd + Rr + C	Z,C,N,V,H	1
ADIW	Rdl, K	Add immediate to word	Rdh: Rdl ← Rdh: Rdl + K	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two registers	Rd ← Rd – Rr	Z,C,N,V,H	1
SUBI	Rd, K	Subtract constant from register	Rd ← Rd – K	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with carry two registers	Rd ← Rd – Rr – C	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with carry constant from reg.	Rd ← Rd – K – C	Z,C,N,V,H	1
SBIW	Rdl, K	Subtract immediate from word	Rdh: Rdl ← Rdh: Rdl – K	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND registers	Rd ← Rd × Rr	Z,N,V	1
ANDI	Rd, K	Logical AND register and constant	Rd ← Rd × K	Z,N,V	1
OR	Rd, Rr	Logical OR registers	Rd ← Rd v Rr	Z,N,V	1
ORI	Rd, K	Logical OR register and constant	Rd ← Rd v K	Z,N,V	1
EOR	Rd, Rr	Exclusive OR registers	Rd ← Rd ⊕ Rr	Z,N,V	1
COM	Rd	One's complement	Rd ← 0xFF – Rd	Z,C,N,V	1
NEG	Rd	Two's complement	Rd ← 0x00 – Rd	Z,C,N,V,H	1
SBR	Rd, K	Set bit(s) in register	Rd ← Rd v K	Z,N,V	1
CBR	Rd, K	Clear bit(s) in register	Rd ← Rd × (0xFF – K)	Z,N,V	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V	1
DEC	Rd	Decrement	Rd ← Rd – 1	Z,N,V	1
TST	Rd	Test for zero or minus	Rd ← Rd × Rd	Z,N,V	1
CLR	Rd	Clear register	Rd ← Rd ⊕ Rd	Z,N,V	1
SER	Rd	Set register	Rd ← 0xFF	None	1
MUL	Rd, Rr	Multiply unsigned	R1:R0 ← Rd × Rr	Z,C	2
MULS	Rd, Rr	Multiply signed	R1:R0 ← Rd × Rr	Z,C	2
MULSU	Rd, Rr	Multiply signed with unsigned	R1:R0 ← Rd × Rr	Z,C	2
FMUL	Rd, Rr	Fractional multiply unsigned	R1:R0 ← (Rd × Rr) << 1	Z,C	2
FMULS	Rd, Rr	Fractional multiply signed	R1:R0 ← (Rd × Rr) << 1	Z,C	2
FMULSU	Rd, Rr	Fractional multiply signed with unsigned	R1:R0 ← (Rd × Rr) << 1	Z,C	2
Branch Instructions					
RJMP	k	Relative jump	PC ← PC + k + 1	None	2
IJMP		Indirect jump to (Z)	PC ← Z	None	2
JMP	k	Direct jump	PC ← k	None	3
RCALL	k	Relative subroutine call	PC ← PC + k + 1	None	3
ICALL		Indirect call to (Z)	PC ← Z	None	3
CALL	k	Direct subroutine call	PC ← k	None	4
RET		Subroutine return	PC ← STACK	None	4
RETI		Interrupt return	PC ← STACK	I	4
CPSE	Rd, Rr	Compare, skip if equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1/2/3
CP	Rd, Rr	Compare	Rd – Rr	Z,N,V,C,H	1

31. Instruction Set Summary (Continued)

Mnemonics	Operands	Description	Operation	Flags	#Clocks
CPC	Rd, Rr	Compare with carry	Rd – Rr – C	Z, N,V,C,H	1
CPI	Rd, K	Compare register with immediate	Rd – K	Z, N,V,C,H	1
SBRC	Rr, b	Skip if bit in register cleared	if (Rr (b) = 0) PC ← PC + 2 or 3	None	1/2/3
SBRS	Rr, b	Skip if bit in register is set	if (Rr(b)=1) PC ← PC + 2 or 3	None	1/2/3
SBIC	P, b	Skip if bit in I/O register cleared	if (P(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBIS	P, b	Skip if bit in I/O register is set	if (P(b)=1) PC ← PC + 2 or 3	None	1/2/3
BRBS	s, k	Branch if status flag set	if (SREG(s) = 1) then PC←PC + k + 1	None	1/2
BRBC	s, k	Branch if status flag cleared	if (SREG(s) = 0) then PC←PC + k + 1	None	1/2
BREQ	k	Branch if equal	if (Z = 1) then PC ← PC + k + 1	None	1/2
BRNE	k	Branch if not equal	if (Z = 0) then PC ← PC + k + 1	None	1/2
BRCS	k	Branch if carry set	if (C = 1) then PC ← PC + k + 1	None	1/2
BRCC	k	Branch if carry cleared	if (C = 0) then PC ← PC + k + 1	None	1/2
BRSH	k	Branch if same or higher	if (C = 0) then PC ← PC + k + 1	None	1/2
BRLO	k	Branch if lower	if (C = 1) then PC ← PC + k + 1	None	1/2
BRMI	k	Branch if minus	if (N = 1) then PC ← PC + k + 1	None	1/2
BRPL	k	Branch if plus	if (N = 0) then PC ← PC + k + 1	None	1/2
BRGE	k	Branch if greater or equal, signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1/2
BRLT	k	Branch if less than zero, signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1/2
BRHS	k	Branch if half carry flag set	if (H = 1) then PC ← PC + k + 1	None	1/2
BRHC	k	Branch if half carry flag cleared	if (H = 0) then PC ← PC + k + 1	None	1/2
BRTS	k	Branch if T flag set	if (T = 1) then PC ← PC + k + 1	None	1/2
BRTC	k	Branch if T flag cleared	if (T = 0) then PC ← PC + k + 1	None	1/2
BRVS	k	Branch if overflow flag is set	if (V = 1) then PC ← PC + k + 1	None	1/2
BRVC	k	Branch if overflow flag is cleared	if (V = 0) then PC ← PC + k + 1	None	1/2
BRIE	k	Branch if interrupt enabled	if (I = 1) then PC ← PC + k + 1	None	1/2
BRID	k	Branch if interrupt disabled	if (I = 0) then PC ← PC + k + 1	None	1/2
Bit and Bit-Test Instructions					
SBI	P, b	Set bit in I/O register	I/O (P, b) ← 1	None	2
CBI	P, b	Clear bit in I/O register	I/O (P, b) ← 0	None	2
LSL	Rd	Logical shift left	Rd(n+1) ← Rd (n), Rd(0) ← 0	Z,C,N,V	1
LSR	Rd	Logical shift right	Rd (n) ← Rd(n+1), Rd(7) ← 0	Z,C,N,V	1
ROL	Rd	Rotate left through carry	Rd(0) ← C,Rd(n+1) ← Rd (n), C ← Rd(7)	Z,C,N,V	1
ROR	Rd	Rotate right through carry	Rd(7) ← C, Rd (n) ← Rd(n+1),C ← Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic shift right	Rd (n) ← Rd(n+1), n=0..6	Z,C,N,V	1
SWAP	Rd	Swap nibbles	Rd(3..0) ← Rd(7..4),Rd(7..4) ← Rd(3..0)	None	1
BSET	s	Flag set	SREG (s) ← 1	SREG (s)	1
BCLR	s	Flag clear	SREG (s) ← 0	SREG (s)	1
BST	Rr, b	Bit store from register to T	T ← Rr (b)	T	1
BLD	Rd, b	Bit load from T to register	Rd (b) ← T	None	1
SEC		Set carry	C ← 1	C	1
CLC		Clear carry	C ← 0	C	1
SEN		Set negative flag	N ← 1	N	1

31. Instruction Set Summary (Continued)

Mnemonics	Operands	Description	Operation	Flags	#Clocks
CLN		Clear negative flag	N ← 0	N	1
SEZ		Set zero flag	Z ← 1	Z	1
CLZ		Clear zero flag	Z ← 0	Z	1
SEI		Global interrupt enable	I ← 1	I	1
CLI		Global interrupt disable	I ← 0	I	1
SES		Set signed test flag	S ← 1	S	1
CLS		Clear signed test flag	S ← 0	S	1
SEV		Set twos complement overflow.	V ← 1	V	1
CLV		Clear twos complement overflow	V ← 0	V	1
SET		Set T in SREG	T ← 1	T	1
CLT		Clear T in SREG	T ← 0	T	1
SEH		Set half carry flag in SREG	H ← 1	H	1
CLH		Clear half carry flag in SREG	H ← 0	H	1
Data Transfer Instructions					
MOV	Rd, Rr	Move between registers	Rd ← Rr	None	1
MOVW	Rd, Rr	Copy register word	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	Rd, K	Load immediate	Rd ← K	None	1
LD	Rd, X	Load indirect	Rd ← (X)	None	2
LD	Rd, X+	Load indirect and post-inc.	Rd ← (X), X ← X + 1	None	2
LD	Rd, – X	Load indirect and pre-dec.	X ← X – 1, Rd ← (X)	None	2
LD	Rd, Y	Load indirect	Rd ← (Y)	None	2
LD	Rd, Y+	Load indirect and post-inc.	Rd ← (Y), Y ← Y + 1	None	2
LD	Rd, – Y	Load indirect and pre-dec.	Y ← Y – 1, Rd ← (Y)	None	2
LDD	Rd, Y+ q	Load indirect with displacement	Rd ← (Y + q)	None	2
LD	Rd, Z	Load indirect	Rd ← (Z)	None	2
LD	Rd, Z+	Load indirect and post-inc.	Rd ← (Z), Z ← Z+1	None	2
LD	Rd, –Z	Load indirect and pre-dec.	Z ← Z – 1, Rd ← (Z)	None	2
LDD	Rd, Z+ q	Load indirect with displacement	Rd ← (Z + q)	None	2
LDS	Rd, k	Load direct from SRAM	Rd ← (k)	None	2
ST	X, Rr	Store indirect	(X) ← Rr	None	2
ST	X+, Rr	Store indirect and post-inc.	(X) ← Rr, X ← X + 1	None	2
ST	– X, Rr	Store indirect and pre-dec.	X ← X – 1, (X) ← Rr	None	2
ST	Y, Rr	Store indirect	(Y) ← Rr	None	2
ST	Y+, Rr	Store indirect and post-inc.	(Y) ← Rr, Y ← Y + 1	None	2
ST	– Y, Rr	Store indirect and pre-dec.	Y ← Y – 1, (Y) ← Rr	None	2
STD	Y+ q, Rr	Store indirect with displacement	(Y + q) ← Rr	None	2
ST	Z, Rr	Store indirect	(Z) ← Rr	None	2
ST	Z +, Rr	Store indirect and post-inc.	(Z) ← Rr, Z ← Z + 1	None	2
ST	–Z, Rr	Store indirect and pre-dec.	Z ← Z – 1, (Z) ← Rr	None	2
STD	Z + q, Rr	Store indirect with displacement	(Z + q) ← Rr	None	2
STS	k, Rr	Store direct to SRAM	(k) ← Rr	None	2
LPM		Load program memory	R0 ← (Z)	None	3

31. Instruction Set Summary (Continued)

Mnemonics	Operands	Description	Operation	Flags	#Clocks
LPM	Rd, Z	Load program memory	Rd ← (Z)	None	3
LPM	Rd, Z+	Load program memory and post-inc	Rd ← (Z), Z ← Z+1	None	3
SPM		Store program memory	(Z) ← R1:R0	None	–
IN	Rd, P	In port	Rd ← P	None	1
OUT	P, Rr	Out port	P ← Rr	None	1
PUSH	Rr	Push register on stack	STACK ← Rr	None	2
POP	Rd	Pop register from stack	Rd ← STACK	None	2
MCU Control Instructions					
NOP		No operation		None	1
SLEEP		Sleep	(see specific descr. for sleep function)	None	1
WDR		Watchdog reset	(see specific descr. for WDR/timer)	None	1
BREAK		Break	For on-chip debug only	None	N/A

284 ATmega328P [DATASHEET]
7810D–AVR–01/15

Atmel

8.2. ATmega2560 – rejestry i instrukcje

33. Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0x1FF)	Reserved	-	-	-	-	-	-	-	-	
...	Reserved	-	-	-	-	-	-	-	-	
(0x13F)	Reserved									
(0x13E)	Reserved									
(0x13D)	Reserved									
(0x13C)	Reserved									
(0x13B)	Reserved									
(0x13A)	Reserved									
(0x139)	Reserved									
(0x138)	Reserved									
(0x137)	Reserved									
(0x136)	UDR3	USART3 I/O Data Register								page 218
(0x135)	UBRR3H	-	-	-	-	USART3 Baud Rate Register High Byte				page 222
(0x134)	UBRR3L	USART3 Baud Rate Register Low Byte								page 222
(0x133)	Reserved	-	-	-	-	-	-	-	-	
(0x132)	UCSR3C	UMSEL31	UMSEL30	UPM31	UPM30	USBS3	UCSZ31	UCSZ30	UCPOL3	page 235
(0x131)	UCSR3B	RXCIE3	TXCIE3	UDRIE3	RXEN3	TXEN3	UCSZ32	RXB83	TXB83	page 234
(0x130)	UCSR3A	RXC3	TXC3	UDRE3	FE3	DOR3	UPE3	U2X3	MPCM3	page 233
(0x12F)	Reserved	-	-	-	-	-	-	-	-	
(0x12E)	Reserved	-	-	-	-	-	-	-	-	
(0x12D)	OCR5CH	Timer/Counter5 - Output Compare Register C High Byte								page 160
(0x12C)	OCR5CL	Timer/Counter5 - Output Compare Register C Low Byte								page 160
(0x12B)	OCR5BH	Timer/Counter5 - Output Compare Register B High Byte								page 160
(0x12A)	OCR5BL	Timer/Counter5 - Output Compare Register B Low Byte								page 160
(0x129)	OCR5AH	Timer/Counter5 - Output Compare Register A High Byte								page 160
(0x128)	OCR5AL	Timer/Counter5 - Output Compare Register A Low Byte								page 160
(0x127)	ICR5H	Timer/Counter5 - Input Capture Register High Byte								page 161
(0x126)	ICR5L	Timer/Counter5 - Input Capture Register Low Byte								page 161
(0x125)	TCNT5H	Timer/Counter5 - Counter Register High Byte								page 158
(0x124)	TCNT5L	Timer/Counter5 - Counter Register Low Byte								page 158
(0x123)	Reserved	-	-	-	-	-	-	-	-	
(0x122)	TCCR5C	FOC5A	FOC5B	FOC5C	-	-	-	-	-	page 157
(0x121)	TCCR5B	ICNC5	ICES5	-	WGM53	WGM52	CS52	CS51	CS50	page 156
(0x120)	TCCR5A	COM5A1	COM5A0	COM5B1	COM5B0	COM5C1	COM5C0	WGM51	WGM50	page 154
(0x11F)	Reserved	-	-	-	-	-	-	-	-	
(0x11E)	Reserved	-	-	-	-	-	-	-	-	
(0x11D)	Reserved	-	-	-	-	-	-	-	-	
(0x11C)	Reserved	-	-	-	-	-	-	-	-	
(0x11B)	Reserved	-	-	-	-	-	-	-	-	
(0x11A)	Reserved	-	-	-	-	-	-	-	-	
(0x119)	Reserved	-	-	-	-	-	-	-	-	
(0x118)	Reserved	-	-	-	-	-	-	-	-	
(0x117)	Reserved	-	-	-	-	-	-	-	-	
(0x116)	Reserved	-	-	-	-	-	-	-	-	
(0x115)	Reserved	-	-	-	-	-	-	-	-	
(0x114)	Reserved	-	-	-	-	-	-	-	-	
(0x113)	Reserved	-	-	-	-	-	-	-	-	
(0x112)	Reserved	-	-	-	-	-	-	-	-	
(0x111)	Reserved	-	-	-	-	-	-	-	-	
(0x110)	Reserved	-	-	-	-	-	-	-	-	
(0x10F)	Reserved	-	-	-	-	-	-	-	-	
(0x10E)	Reserved	-	-	-	-	-	-	-	-	
(0x10D)	Reserved	-	-	-	-	-	-	-	-	
(0x10C)	Reserved	-	-	-	-	-	-	-	-	
(0x10B)	PORTL	PORTL7	PORTL6	PORTL5	PORTL4	PORTL3	PORTL2	PORTL1	PORTL0	page 100
(0x10A)	DDRL	DDL7	DDL6	DDL5	DDL4	DDL3	DDL2	DDL1	DDL0	page 100
(0x109)	PINL	PINL7	PINL6	PINL5	PINL4	PINL3	PINL2	PINL1	PINL0	page 100
(0x108)	PORTK	PORTK7	PORTK6	PORTK5	PORTK4	PORTK3	PORTK2	PORTK1	PORTK0	page 99
(0x107)	DDRK	DDK7	DDK6	DDK5	DDK4	DDK3	DDK2	DDK1	DDK0	page 99
(0x106)	PINK	PINK7	PINK6	PINK5	PINK4	PINK3	PINK2	PINK1	PINK0	page 99
(0x105)	PORTJ	PORTJ7	PORTJ6	PORTJ5	PORTJ4	PORTJ3	PORTJ2	PORTJ1	PORTJ0	page 99
(0x104)	DDRJ	DDJ7	DDJ6	DDJ5	DDJ4	DDJ3	DDJ2	DDJ1	DDJ0	page 99
(0x103)	PINJ	PINJ7	PINJ6	PINJ5	PINJ4	PINJ3	PINJ2	PINJ1	PINJ0	page 99
(0x102)	PORTH	PORTH7	PORTH6	PORTH5	PORTH4	PORTH3	PORTH2	PORTH1	PORTH0	page 98
(0x101)	DDRH	DDH7	DDH6	DDH5	DDH4	DDH3	DDH2	DDH1	DDH0	page 99

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0x100)	PINH	PINH7	PINH6	PINH5	PINH4	PINH3	PINH2	PINH1	PINH0	page 99
(0xFF)	Reserved	-	-	-	-	-	-	-	-	
(0xFE)	Reserved	-	-	-	-	-	-	-	-	
(0xFD)	Reserved	-	-	-	-	-	-	-	-	
(0xFC)	Reserved	-	-	-	-	-	-	-	-	
(0xFB)	Reserved	-	-	-	-	-	-	-	-	
(0xFA)	Reserved	-	-	-	-	-	-	-	-	
(0xF9)	Reserved	-	-	-	-	-	-	-	-	
(0xF8)	Reserved	-	-	-	-	-	-	-	-	
(0xF7)	Reserved	-	-	-	-	-	-	-	-	
(0xF6)	Reserved	-	-	-	-	-	-	-	-	
(0xF5)	Reserved	-	-	-	-	-	-	-	-	
(0xF4)	Reserved	-	-	-	-	-	-	-	-	
(0xF3)	Reserved	-	-	-	-	-	-	-	-	
(0xF2)	Reserved	-	-	-	-	-	-	-	-	
(0xF1)	Reserved	-	-	-	-	-	-	-	-	
(0xF0)	Reserved	-	-	-	-	-	-	-	-	
(0xEF)	Reserved	-	-	-	-	-	-	-	-	
(0xEE)	Reserved	-	-	-	-	-	-	-	-	
(0xED)	Reserved	-	-	-	-	-	-	-	-	
(0xEC)	Reserved	-	-	-	-	-	-	-	-	
(0xEB)	Reserved	-	-	-	-		-	-	-	
(0xEA)	Reserved	-	-	-	-	-	-	-	-	
(0xE9)	Reserved	-	-	-	-	-	-	-	-	
(0xE8)	Reserved	-	-	-	-	-	-	-	-	
(0xE7)	Reserved	-	-	-	-		-	-	-	
(0xE6)	Reserved	-	-	-	-	-	-	-	-	
(0xE5)	Reserved	-	-	-	-	-	-	-	-	
(0xE4)	Reserved	-	-	-	-	-	-	-	-	
(0xE3)	Reserved	-	-	-	-		-	-	-	
(0xE2)	Reserved	-	-	-	-	-	-	-	-	
(0xE1)	Reserved	-	-	-	-		-	-	-	
(0xE0)	Reserved	-	-	-	-		-	-	-	
(0xDF)	Reserved	-	-	-	-	-	-	-	-	
(0xDE)	Reserved	-	-	-	-	-	-	-	-	
(0xDD)	Reserved	-	-	-	-		-	-	-	
(0xDC)	Reserved	-	-	-	-	-	-	-	-	
(0xDB)	Reserved	-	-	-	-	-	-	-	-	
(0xDA)	Reserved	-	-	-	-	-	-	-	-	
(0xD9)	Reserved	-	-	-	-		-	-	-	
(0xD8)	Reserved	-	-	-	-	-	-	-	-	
(0xD7)	Reserved	-	-	-	-	-	-	-	-	
(0xD6)	UDR2	USART2 I/O Data Register								page 218
(0xD5)	UBRR2H	-	-	-	-	USART2 Baud Rate Register High Byte				page 222
(0xD4)	UBRR2L	USART2 Baud Rate Register Low Byte								page 222
(0xD3)	Reserved	-	-	-	-	-	-	-	-	
(0xD2)	UCSR2C	UMSEL21	UMSEL20	UPM21	UPM20	USBS2	UCSZ21	UCSZ20	UCPOL2	page 235
(0xD1)	UCSR2B	RXCIE2	TXCIE2	UDRIE2	RXEN2	TXEN2	UCSZ22	RXB82	TXB82	page 234
(0xD0)	UCSR2A	RXC2	TXC2	UDRE2	FE2	DOR2	UPE2	U2X2	MPCM2	page 233
(0xCF)	Reserved	-	-	-	-	-	-	-	-	
(0xCE)	UDR1	USART1 I/O Data Register								page 218
(0xCD)	UBRR1H	-	-	-	-	USART1 Baud Rate Register High Byte				page 222
(0xCC)	UBRR1L	USART1 Baud Rate Register Low Byte								page 222
(0xCB)	Reserved	-	-	-	-	-	-	-	-	
(0xCA)	UCSR1C	UMSEL11	UMSEL10	UPM11	UPM10	USBS1	UCSZ11	UCSZ10	UCPOL1	page 235
(0xC9)	UCSR1B	RXCIE1	TXCIE1	UDRIE1	RXEN1	TXEN1	UCSZ12	RXB81	TXB81	page 234
(0xC8)	UCSR1A	RXC1	TXC1	UDRE1	FE1	DOR1	UPE1	U2X1	MPCM1	page 233
(0xC7)	Reserved	-	-	-	-	-	-	-	-	
(0xC6)	UDR0	USART0 I/O Data Register								page 218
(0xC5)	UBRR0H	-	-	-	-	USART0 Baud Rate Register High Byte				page 222
(0xC4)	UBRR0L	USART0 Baud Rate Register Low Byte								page 222
(0xC3)	Reserved	-	-	-	-	-	-	-	-	
(0xC2)	UCSR0C	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01	UCSZ00	UCPOL0	page 235
(0xC1)	UCSR0B	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80	page 234
(0xC0)	UCSR0A	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0	page 234
(0xBF)	Reserved	-	-	-	-	-	-	-	-	
(0xBE)	Reserved	-	-	-	-	-	-	-	-	
(0xBD)	TWAMR	TWAM6	TWAM5	TWAM4	TWAM3	TWAM2	TWAM1	TWAM0	-	page 264

Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V [DATASHEET] 400
2549Q–AVR–02/2014

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0xBC)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE	page 261
(0xBB)	TWDR	2-wire Serial Interface Data Register								page 263
(0xBA)	TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	page 263
(0xB9)	TWSR	TWS7	TWS6	TWS5	TWS4	TWS3	-	TWPS1	TWPS0	page 262
(0xB8)	TWBR	2-wire Serial Interface Bit Rate Register								page 261
(0xB7)	Reserved	-	-	-	-	-	-	-	-	
(0xB6)	ASSR	-	EXCLK	AS2	TCN2UB	OCR2AUB	OCR2BUB	TCR2AUB	TCR2BUB	page 179
(0xB5)	Reserved	-	-	-	-	-	-	-	-	
(0xB4)	OCR2B	Timer/Counter2 Output Compare Register B								page 186
(0xB3)	OCR2A	Timer/Counter2 Output Compare Register A								page 186
(0xB2)	TCNT2	Timer/Counter2 (8 Bit)								page 186
(0xB1)	TCCR2B	FOC2A	FOC2B	-	-	WGM22	CS22	CS21	CS20	page 185
(0xB0)	TCCR2A	COM2A1	COM2A0	COM2B1	COM2B0	-	-	WGM21	WGM20	page 186
(0xAF)	Reserved	-	-	-	-	-	-	-	-	
(0xAE)	Reserved	-	-	-	-	-	-	-	-	
(0xAD)	OCR4CH	Timer/Counter4 - Output Compare Register C High Byte								page 160
(0xAC)	OCR4CL	Timer/Counter4 - Output Compare Register C Low Byte								page 160
(0xAB)	OCR4BH	Timer/Counter4 - Output Compare Register B High Byte								page 160
(0xAA)	OCR4BL	Timer/Counter4 - Output Compare Register B Low Byte								page 160
(0xA9)	OCR4AH	Timer/Counter4 - Output Compare Register A High Byte								page 159
(0xA8)	OCR4AL	Timer/Counter4 - Output Compare Register A Low Byte								page 159
(0xA7)	ICR4H	Timer/Counter4 - Input Capture Register High Byte								page 161
(0xA6)	ICR4L	Timer/Counter4 - Input Capture Register Low Byte								page 161
(0xA5)	TCNT4H	Timer/Counter4 - Counter Register High Byte								page 158
(0xA4)	TCNT4L	Timer/Counter4 - Counter Register Low Byte								page 158
(0xA3)	Reserved	-	-	-	-	-	-	-	-	
(0xA2)	TCCR4C	FOC4A	FOC4B	FOC4C	-	-	-	-	-	page 157
(0xA1)	TCCR4B	ICNC4	ICES4	-	WGM43	WGM42	CS42	CS41	CS40	page 156
(0xA0)	TCCR4A	COM4A1	COM4A0	COM4B1	COM4B0	COM4C1	COM4C0	WGM41	WGM40	page 154
(0x9F)	Reserved	-	-	-	-	-	-	-	-	
(0x9E)	Reserved	-	-	-	-	-	-	-	-	
(0x9D)	OCR3CH	Timer/Counter3 - Output Compare Register C High Byte								page 159
(0x9C)	OCR3CL	Timer/Counter3 - Output Compare Register C Low Byte								page 159
(0x9B)	OCR3BH	Timer/Counter3 - Output Compare Register B High Byte								page 159
(0x9A)	OCR3BL	Timer/Counter3 - Output Compare Register B Low Byte								page 159
(0x99)	OCR3AH	Timer/Counter3 - Output Compare Register A High Byte								page 159
(0x98)	OCR3AL	Timer/Counter3 - Output Compare Register A Low Byte								page 159
(0x97)	ICR3H	Timer/Counter3 - Input Capture Register High Byte								page 161
(0x96)	ICR3L	Timer/Counter3 - Input Capture Register Low Byte								page 161
(0x95)	TCNT3H	Timer/Counter3 - Counter Register High Byte								page 158
(0x94)	TCNT3L	Timer/Counter3 - Counter Register Low Byte								page 158
(0x93)	Reserved	-	-	-	-	-	-	-	-	
(0x92)	TCCR3C	FOC3A	FOC3B	FOC3C	-	-	-	-	-	page 157
(0x91)	TCCR3B	ICNC3	ICES3	-	WGM33	WGM32	CS32	CS31	CS30	page 156
(0x90)	TCCR3A	COM3A1	COM3A0	COM3B1	COM3B0	COM3C1	COM3C0	WGM31	WGM30	page 154
(0x8F)	Reserved	-	-	-	-	-	-	-	-	
(0x8E)	Reserved	-	-	-	-	-	-	-	-	
(0x8D)	OCR1CH	Timer/Counter1 - Output Compare Register C High Byte								page 159
(0x8C)	OCR1CL	Timer/Counter1 - Output Compare Register C Low Byte								page 159
(0x8B)	OCR1BH	Timer/Counter1 - Output Compare Register B High Byte								page 159
(0x8A)	OCR1BL	Timer/Counter1 - Output Compare Register B Low Byte								page 159
(0x89)	OCR1AH	Timer/Counter1 - Output Compare Register A High Byte								page 159
(0x88)	OCR1AL	Timer/Counter1 - Output Compare Register A Low Byte								page 159
(0x87)	ICR1H	Timer/Counter1 - Input Capture Register High Byte								page 160
(0x86)	ICR1L	Timer/Counter1 - Input Capture Register Low Byte								page 160
(0x85)	TCNT1H	Timer/Counter1 - Counter Register High Byte								page 158
(0x84)	TCNT1L	Timer/Counter1 - Counter Register Low Byte								page 158
(0x83)	Reserved	-	-	-	-	-	-	-	-	
(0x82)	TCCR1C	FOC1A	FOC1B	FOC1C	-	-	-	-	-	page 157
(0x81)	TCCR1B	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10	page 156
(0x80)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	COM1C1	COM1C0	WGM11	WGM10	page 154
(0x7F)	DIDR1	-	-	-	-	-	-	AIN1D	AIN0D	page 267
(0x7E)	DIDR0	ADC7D	ADC6D	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D	page 287
(0x7D)	DIDR2	ADC15D	ADC14D	ADC13D	ADC12D	ADC11D	ADC10D	ADC9D	ADC8D	page 288
(0x7C)	ADMUX	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	page 281
(0x7B)	ADCSRB	-	ACME	-	-	MUX5	ADTS2	ADTS1	ADTS0	page 266, 282, 287
(0x7A)	ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	page 285
(0x79)	ADCH	ADC Data Register High byte								page 286

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0x78)	ADCL	ADC Data Register Low byte								page 286
(0x77)	Reserved	-	-	-	-	-	-	-	-	
(0x76)	Reserved	-	-	-	-	-	-	-	-	
(0x75)	XMCRB	XMBK	-	-	-	-	XMM2	XMM1	XMM0	page 38
(0x74)	XMCRA	SRE	SRL2	SRL1	SRL0	SRW11	SRW10	SRW01	SRW00	page 36
(0x73)	TIMSK5	-	-	ICIE5	-	OCIE5C	OCIE5B	OCIE5A	TOIE5	page 162
(0x72)	TIMSK4	-	-	ICIE4	-	OCIE4C	OCIE4B	OCIE4A	TOIE4	page 161
(0x71)	TIMSK3	-	-	ICIE3	-	OCIE3C	OCIE3B	OCIE3A	TOIE3	page 161
(0x70)	TIMSK2	-	-	-	-	-	OCIE2B	OCIE2A	TOIE2	page 188
(0x6F)	TIMSK1	-	-	ICIE1	-	OCIE1C	OCIE1B	OCIE1A	TOIE1	page 161
(0x6E)	TIMSK0	-	-	-	-	-	OCIE0B	OCIE0A	TOIE0	page 131
(0x6D)	PCMSK2	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16	page 113
(0x6C)	PCMSK1	PCINT15	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8	page 113
(0x6B)	PCMSK0	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	page 114
(0x6A)	EICRB	ISC71	ISC70	ISC61	ISC60	ISC51	ISC50	ISC41	ISC40	page 110
(0x69)	EICRA	ISC31	ISC30	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00	page 110
(0x68)	PCICR	-	-	-	-	-	PCIE2	PCIE1	PCIE0	page 112
(0x67)	Reserved	-	-	-	-	-	-	-	-	
(0x66)	OSCCAL	Oscillator Calibration Register								page 48
(0x65)	PRR1	-	-	PRTIM5	PRTIM4	PRTIM3	PRUSART3	PRUSART2	PRUSART1	page 56
(0x64)	PRR0	PRTWI	PRTIM2	PRTIM0	-	PRTIM1	PRSPI	PRUSART0	PRADC	page 55
(0x63)	Reserved	-	-	-	-	-	-	-	-	
(0x62)	Reserved	-	-	-	-	-	-	-	-	
(0x61)	CLKPR	CLKPCE	-	-	-	CLKPS3	CLKPS2	CLKPS1	CLKPS0	page 48
(0x60)	WDTCSR	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	page 65
0x3F (0x5F)	SREG	I	T	H	S	V	N	Z	C	page 13
0x3E (0x5E)	SPH	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	page 15
0x3D (0x5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	page 15
0x3C (0x5C)	EIND	-	-	-	-	-	-	-	EIND0	page 16
0x3B (0x5B)	RAMPZ	-	-	-	-	-	-	RAMPZ1	RAMPZ0	page 16
0x3A (0x5A)	Reserved	-	-	-	-	-	-	-	-	
0x39 (0x59)	Reserved	-	-	-	-	-	-	-	-	
0x38 (0x58)	Reserved	-	-	-	-	-	-	-	-	
0x37 (0x57)	SPMCSR	SPMIE	RWWSB	SIGRD	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	page 323
0x36 (0x56)	Reserved	-	-	-	-	-	-	-	-	
0x35 (0x55)	MCUCR	JTD	-	-	PUD	-	-	IVSEL	IVCE	page 64, 108, 96, 301
0x34 (0x54)	MCUSR	-	-	-	JTRF	WDRF	BORF	EXTRF	PORF	page 301
0x33 (0x53)	SMCR	-	-	-	-	SM2	SM1	SM0	SE	page 50
0x32 (0x52)	Reserved	-	-	-	-	-	-	-	-	
0x31 (0x51)	OCDR	OCDR7	OCDR6	OCDR5	OCDR4	OCDR3	OCDR2	OCDR1	OCDR0	page 294
0x30 (0x50)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	page 266
0x2F (0x4F)	Reserved	-	-	-	-	-	-	-	-	
0x2E (0x4E)	SPDR	SPI Data Register								page 199
0x2D (0x4D)	SPSR	SPIF	WCOL	-	-	-	-	-	SPI2X	page 198
0x2C (0x4C)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	page 197
0x2B (0x4B)	GPIOR2	General Purpose I/O Register 2								page 36
0x2A (0x4A)	GPIOR1	General Purpose I/O Register 1								page 36
0x29 (0x49)	Reserved	-	-	-	-	-	-	-	-	
0x28 (0x48)	OCR0B	Timer/Counter0 Output Compare Register B								page 130
0x27 (0x47)	OCR0A	Timer/Counter0 Output Compare Register A								page 130
0x26 (0x46)	TCNT0	Timer/Counter0 (8 Bit)								page 130
0x25 (0x45)	TCCR0B	FOC0A	FOC0B	-	-	WGM02	CS02	CS01	CS00	page 129
0x24 (0x44)	TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00	page 126
0x23 (0x43)	GTCCR	TSM	-	-	-	-	-	PSRASY	PSRSYNC	page 166, 189
0x22 (0x42)	EEARH	-	-	-	-	EEPROM Address Register High Byte				page 34
0x21 (0x41)	EEARL	EEPROM Address Register Low Byte								page 34
0x20 (0x40)	EEDR	EEPROM Data Register								page 34
0x1F (0x3F)	EECR	-	-	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE	page 34
0x1E (0x3E)	GPIOR0	General Purpose I/O Register 0								page 36
0x1D (0x3D)	EIMSK	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0	page 111
0x1C (0x3C)	EIFR	INTF7	INTF6	INTF5	INTF4	INTF3	INTF2	INTF1	INTF0	page 112
0x1B (0x3B)	PCIFR	-	-	-	-	-	PCIF2	PCIF1	PCIF0	page 113
0x1A (0x3A)	TIFR5	-	-	ICF5	-	OCF5C	OCF5B	OCF5A	TOV5	page 162
0x19 (0x39)	TIFR4	-	-	ICF4	-	OCF4C	OCF4B	OCF4A	TOV4	page 162
0x18 (0x38)	TIFR3	-	-	ICF3	-	OCF3C	OCF3B	OCF3A	TOV3	page 162
0x17 (0x37)	TIFR2	-	-	-	-	-	OCF2B	OCF2A	TOV2	page 188
0x16 (0x36)	TIFR1	-	-	ICF1	-	OCF1C	OCF1B	OCF1A	TOV1	page 162
0x15 (0x35)	TIFR0	-	-	-	-	-	OCF0B	OCF0A	TOV0	page 131

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x14 (0x34)	PORTG	-	-	PORTG5	PORTG4	PORTG3	PORTG2	PORTG1	PORTG0	page 98
0x13 (0x33)	DDRG	-	-	DDG5	DDG4	DDG3	DDG2	DDG1	DDG0	page 98
0x12 (0x32)	PING	-	-	PING5	PING4	PING3	PING2	PING1	PING0	page 98
0x11 (0x31)	PORTF	PORTF7	PORTF6	PORTF5	PORTF4	PORTF3	PORTF2	PORTF1	PORTF0	page 97
0x10 (0x30)	DDRF	DDF7	DDF6	DDF5	DDF4	DDF3	DDF2	DDF1	DDF0	page 98
0x0F (0x2F)	PINF	PINF7	PINF6	PINF5	PINF4	PINF3	PINF2	PINF1	PINF0	page 98
0x0E (0x2E)	PORTE	PORTE7	PORTE6	PORTE5	PORTE4	PORTE3	PORTE2	PORTE1	PORTE0	page 97
0x0D (0x2D)	DDRE	DDE7	DDE6	DDE5	DDE4	DDE3	DDE2	DDE1	DDE0	page 97
0x0C (0x2C)	PINE	PINE7	PINE6	PINE5	PINE4	PINE3	PINE2	PINE1	PINE0	page 98
0x0B (0x2B)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	page 97
0x0A (0x2A)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	page 97
0x09 (0x29)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	page 97
0x08 (0x28)	PORTC	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	page 97
0x07 (0x27)	DDRC	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	page 97
0x06 (0x26)	PINC	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	page 97
0x05 (0x25)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	page 96
0x04 (0x24)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	page 96
0x03 (0x23)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	page 96
0x02 (0x22)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	page 96
0x01 (0x21)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	page 96
0x00 (0x20)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	page 96

Notes:
1. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
2. I/O registers within the address range $00 - $1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions.
3. Some of the status flags are cleared by writing a logical one to them. Note that the CBI and SBI instructions will operate on all bits in the I/O register, writing a one back into any flag read as set, thus clearing the flag. The CBI and SBI instructions work with registers 0x00 to 0x1F only.
4. When using the I/O specific commands IN and OUT, the I/O addresses $00 - $3F must be used. When addressing I/O registers as data space using LD and ST instructions, $20 must be added to these addresses. The ATmega640/1280/1281/2560/2561 is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from $60 - $1FF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

34. Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	Add two Registers	Rd ← Rd + Rr	Z, C, N, V, H	1
ADC	Rd, Rr	Add with Carry two Registers	Rd ← Rd + Rr + C	Z, C, N, V, H	1
ADIW	Rdl,K	Add Immediate to Word	Rdh:Rdl ← Rdh:Rdl + K	Z, C, N, V, S	2
SUB	Rd, Rr	Subtract two Registers	Rd ← Rd - Rr	Z, C, N, V, H	1
SUBI	Rd, K	Subtract Constant from Register	Rd ← Rd - K	Z, C, N, V, H	1
SBC	Rd, Rr	Subtract with Carry two Registers	Rd ← Rd - Rr - C	Z, C, N, V, H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	Rd ← Rd - K - C	Z, C, N, V, H	1
SBIW	Rdl,K	Subtract Immediate from Word	Rdh:Rdl ← Rdh:Rdl - K	Z, C, N, V, S	2
AND	Rd, Rr	Logical AND Registers	Rd ← Rd • Rr	Z, N, V	1
ANDI	Rd, K	Logical AND Register and Constant	Rd ← Rd • K	Z, N, V	1
OR	Rd, Rr	Logical OR Registers	Rd ← Rd v Rr	Z, N, V	1
ORI	Rd, K	Logical OR Register and Constant	Rd ← Rd v K	Z, N, V	1
EOR	Rd, Rr	Exclusive OR Registers	Rd ← Rd ⊕ Rr	Z, N, V	1
COM	Rd	One's Complement	Rd ← 0xFF − Rd	Z, C, N, V	1
NEG	Rd	Two's Complement	Rd ← 0x00 − Rd	Z, C, N, V, H	1
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z, N, V	1
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd • (0xFF - K)	Z, N, V	1
INC	Rd	Increment	Rd ← Rd + 1	Z, N, V	1
DEC	Rd	Decrement	Rd ← Rd − 1	Z, N, V	1
TST	Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z, N, V	1
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z, N, V	1
SER	Rd	Set Register	Rd ← 0xFF	None	1
MUL	Rd, Rr	Multiply Unsigned	R1:R0 ← Rd x Rr	Z, C	2
MULS	Rd, Rr	Multiply Signed	R1:R0 ← Rd x Rr	Z, C	2
MULSU	Rd, Rr	Multiply Signed with Unsigned	R1:R0 ← Rd x Rr	Z, C	2
FMUL	Rd, Rr	Fractional Multiply Unsigned	R1:R0 ← (Rd x Rr) << 1	Z, C	2
FMULS	Rd, Rr	Fractional Multiply Signed	R1:R0 ← (Rd x Rr) << 1	Z, C	2
FMULSU	Rd, Rr	Fractional Multiply Signed with Unsigned	R1:R0 ← (Rd x Rr) << 1	Z, C	2
BRANCH INSTRUCTIONS					
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC ← Z	None	2
EIJMP		Extended Indirect Jump to (Z)	PC ←(EIND:Z)	None	2
JMP	k	Direct Jump	PC ← k	None	3
RCALL	k	Relative Subroutine Call	PC ← PC + k + 1	None	4
ICALL		Indirect Call to (Z)	PC ← Z	None	4
EICALL		Extended Indirect Call to (Z)	PC ←(EIND:Z)	None	4
CALL	k	Direct Subroutine Call	PC ← k	None	5
RET		Subroutine Return	PC ← STACK	None	5
RETI		Interrupt Return	PC ← STACK	I	5
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1/2/3
CP	Rd,Rr	Compare	Rd − Rr	Z, N, V, C, H	1
CPC	Rd,Rr	Compare with Carry	Rd − Rr − C	Z, N, V, C, H	1
CPI	Rd,K	Compare Register with Immediate	Rd − K	Z, N, V, C, H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBRS	Rr, b	Skip if Bit in Register is Set	if (Rr(b)=1) PC ← PC + 2 or 3	None	1/2/3
SBIC	P, b	Skip if Bit in I/O Register Cleared	if (P(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBIS	P, b	Skip if Bit in I/O Register is Set	if (P(b)=1) PC ← PC + 2 or 3	None	1/2/3
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC←PC+k + 1	None	1/2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC←PC+k + 1	None	1/2
BREQ	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1/2
BRNE	k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1/2
BRCS	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1/2
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1/2
BRSH	k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	1/2
BRLO	k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	1/2
BRMI	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1/2
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1/2
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1/2
BRLT	k	Branch if Less Than Zero, Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1/2
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1/2
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1/2
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1/2
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1/2
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1/2

Mnemonics	Operands	Description	Operation	Flags	#Clocks
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	1/2
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1/2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1/2
BIT AND BIT-TEST INSTRUCTIONS					
SBI	P,b	Set Bit in I/O Register	I/O(P,b) ← 1	None	2
CBI	P,b	Clear Bit in I/O Register	I/O(P,b) ← 0	None	2
LSL	Rd	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z, C, N, V	1
LSR	Rd	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z, C, N, V	1
ROL	Rd	Rotate Left Through Carry	Rd(0)←C,Rd(n+1)← Rd(n),C←Rd(7)	Z, C, N, V	1
ROR	Rd	Rotate Right Through Carry	Rd(7)←C,Rd(n)← Rd(n+1),C←Rd(0)	Z, C, N, V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6	Z, C, N, V	1
SWAP	Rd	Swap Nibbles	Rd(3..0)←Rd(7..4),Rd(7..4)←Rd(3..0)	None	1
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1
BST	Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1
BLD	Rd, b	Bit load from T to Register	Rd(b) ← T	None	1
SEC		Set Carry	C ← 1	C	1
CLC		Clear Carry	C ← 0	C	1
SEN		Set Negative Flag	N ← 1	N	1
CLN		Clear Negative Flag	N ← 0	N	1
SEZ		Set Zero Flag	Z ← 1	Z	1
CLZ		Clear Zero Flag	Z ← 0	Z	1
SEI		Global Interrupt Enable	I ← 1	I	1
CLI		Global Interrupt Disable	I ← 0	I	1
SES		Set Signed Test Flag	S ← 1	S	1
CLS		Clear Signed Test Flag	S ← 0	S	1
SEV		Set Twos Complement Overflow.	V ← 1	V	1
CLV		Clear Twos Complement Overflow	V ← 0	V	1
SET		Set T in SREG	T ← 1	T	1
CLT		Clear T in SREG	T ← 0	T	1
SEH		Set Half Carry Flag in SREG	H ← 1	H	1
CLH		Clear Half Carry Flag in SREG	H ← 0	H	1
DATA TRANSFER INSTRUCTIONS					
MOV	Rd, Rr	Move Between Registers	Rd ← Rr	None	1
MOVW	Rd, Rr	Copy Register Word	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LD	Rd, X	Load Indirect	Rd ← (X)	None	2
LD	Rd, X+	Load Indirect and Post-Inc.	Rd ← (X), X ← X + 1	None	2
LD	Rd, - X	Load Indirect and Pre-Dec.	X ← X - 1, Rd ← (X)	None	2
LD	Rd, Y	Load Indirect	Rd ← (Y)	None	2
LD	Rd, Y+	Load Indirect and Post-Inc.	Rd ← (Y), Y ← Y + 1	None	2
LD	Rd, - Y	Load Indirect and Pre-Dec.	Y ← Y - 1, Rd ← (Y)	None	2
LDD	Rd,Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2
LD	Rd, Z	Load Indirect	Rd ← (Z)	None	2
LD	Rd, Z+	Load Indirect and Post-Inc.	Rd ← (Z), Z ← Z+1	None	2
LD	Rd, -Z	Load Indirect and Pre-Dec.	Z ← Z - 1, Rd ← (Z)	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2
LDS	Rd, k	Load Direct from SRAM	Rd ← (k)	None	2
ST	X, Rr	Store Indirect	(X) ← Rr	None	2
ST	X+, Rr	Store Indirect and Post-Inc.	(X) ← Rr, X ← X + 1	None	2
ST	- X, Rr	Store Indirect and Pre-Dec.	X ← X - 1, (X) ← Rr	None	2
ST	Y, Rr	Store Indirect	(Y) ← Rr	None	2
ST	Y+, Rr	Store Indirect and Post-Inc.	(Y) ← Rr, Y ← Y + 1	None	2
ST	- Y, Rr	Store Indirect and Pre-Dec.	Y ← Y - 1, (Y) ← Rr	None	2
STD	Y+q,Rr	Store Indirect with Displacement	(Y + q) ← Rr	None	2
ST	Z, Rr	Store Indirect	(Z) ← Rr	None	2
ST	Z+, Rr	Store Indirect and Post-Inc.	(Z) ← Rr, Z ← Z + 1	None	2
ST	-Z, Rr	Store Indirect and Pre-Dec.	Z ← Z - 1, (Z) ← Rr	None	2
STD	Z+q,Rr	Store Indirect with Displacement	(Z + q) ← Rr	None	2
STS	k, Rr	Store Direct to SRAM	(k) ← Rr	None	2
LPM		Load Program Memory	R0 ← (Z)	None	3
LPM	Rd, Z	Load Program Memory	Rd ← (Z)	None	3
LPM	Rd, Z+	Load Program Memory and Post-Inc	Rd ← (Z), Z ← Z+1	None	3
ELPM		Extended Load Program Memory	R0 ← (RAMPZ:Z)	None	3
ELPM	Rd, Z	Extended Load Program Memory	Rd ← (RAMPZ:Z)	None	3
ELPM	Rd, Z+	Extended Load Program Memory	Rd ← (RAMPZ:Z), RAMPZ:Z ←RAMPZ:Z+1	None	3
SPM		Store Program Memory	(Z) ← R1:R0	None	-
IN	Rd, P	In Port	Rd ← P	None	1

Mnemonics	Operands	Description	Operation	Flags	#Clocks
OUT	P, Rr	Out Port	P ← Rr	None	1
PUSH	Rr	Push Register on Stack	STACK ← Rr	None	2
POP	Rd	Pop Register from Stack	Rd ← STACK	None	2
MCU CONTROL INSTRUCTIONS					
NOP		No Operation		None	1
SLEEP		Sleep	(see specific descr. for Sleep function)	None	1
WDR		Watchdog Reset	(see specific descr. for WDR/timer)	None	1
BREAK		Break	For On-chip Debug Only	None	N/A

Note: EICALL and EIJMP do not exist in ATmega640/1280/1281.
ELPM does not exist in ATmega640.