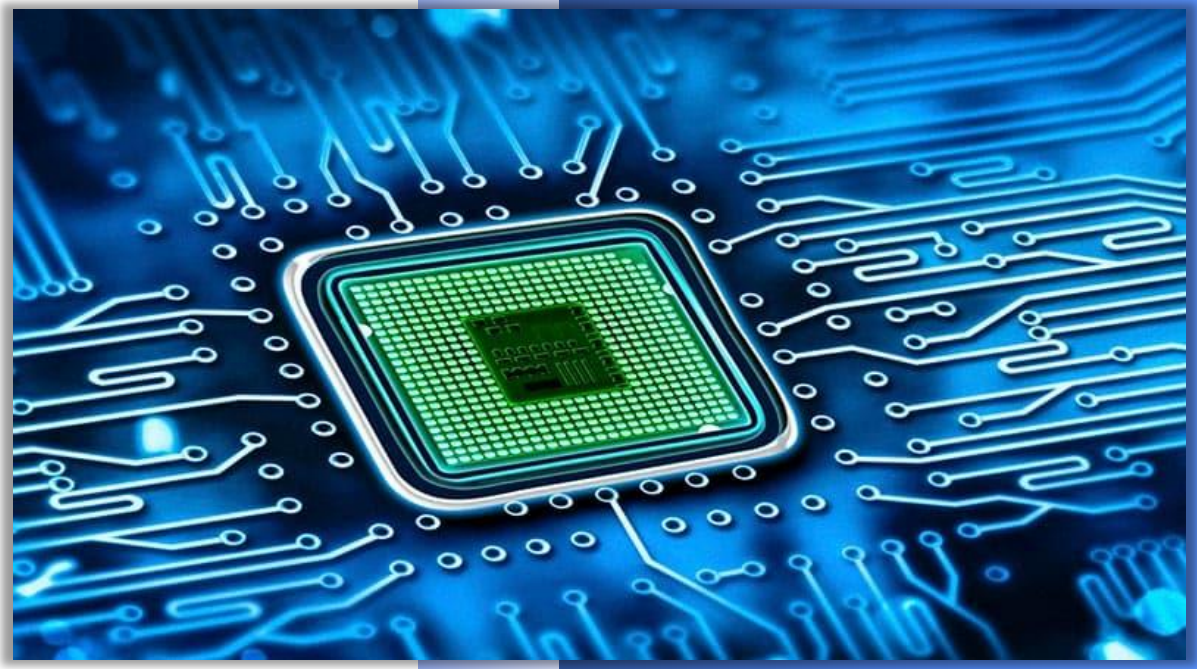
Projekt uruchomienia robota w trybie manualnym z działającym chwytakiem



Technika Mikroprocesorowa

Autorzy:
Zuzanna Nowak, 261051
Patryk Olearczyk, 261089

Biomechanika Inżynierska

Grupa:
wtorek TN 13:15-15:00

Spis treści

1. Przyjęte założenia 3
2. Wstęp teoretyczny 3
   2.1. Definicje 3
   2.2. Środowisko projektowe 3
3. Budowa układu 4
4. Napędy stosowane w manipulatorach/robotach/urządzeniach chwytających 5
5. Schemat układu 6
6. Przebieg stworzenia projektu 6
7. Projekt (kod) w programie Microchip Studio 7
8. Wnioski 15
9. Bibliografia 15

`

1. Przyjęte założenia

W zadaniach projektowych założono:

- Stworzenie urządzenia sterowanego mikroprocesorowo, które będzie miało w swojej budowie elementy napędowe umożliwiające uzyskanie określonej funkcjonalności, np.: ramię manipulatora/robota,
- Zasilanie urządzenia: bezpieczne elektryczne – 5V,
- Sterowanie urządzeniem: kodem assemblera z wykorzystaniem mikroswitchów,
- Tryby pracy urządzenia: manualny.

2. Wstęp teoretyczny

2.1. Definicje

Robot – maszyna, w szczególności system komputerowy, w którym program steruje peryferiami w celu wykonania określonego zadania. [1]

Chwytak – jest to oprzyrządowanie technologiczne manipulatorów robotów przeznaczone do manipulowania przedmiotami. Zadaniem tych narzędzi jest uchwycenie detalu, utrzymanie go podczas transportu oraz jego zwolnienie w miejscu docelowym. Zadaniem chwytaka jest: uchwycenie manipulowanego przedmiotu z zapewnieniem mu właściwej orientacji, utrzymanie przedmiotu pomimo działających sił zewnętrznych i przyspieszeń transportowych i pozostawienie przedmiotu we właściwej orientacji w miejscu przeznaczenia.[2]

W projekcie chwytak sterowany jest za pomocą sygnałów PWM.

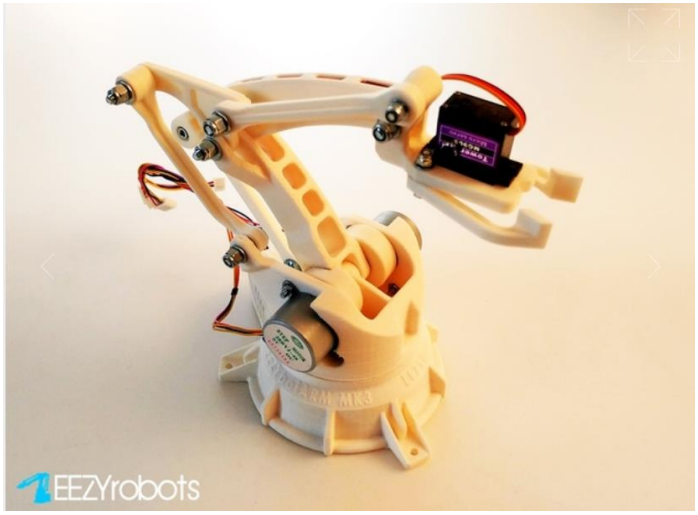
PWM (Pulse Width Modulation) – to inaczej modulacja szerokości impulsu, czyli metoda regulacji sygnału napięciowego lub prądowego o stałej częstotliwości i amplitudzie. Technologia ta pozwala na dostarczenie do urządzenia mniejszej ilości energii w określonym czasie. [3]

Silniki krokowe są doskonałym rozwiązaniem dla aplikacji, w których wymagana jest wysoka dokładność położenia geometrycznego układu napędzanego przez ten silnik. Z silnikami krokowymi mamy do czynienia m.in. w zegarach elektromechanicznych, drukarkach atramentowych i laserowych, drukarkach 3D, sprzęcie RTV, maszynach CNC, a także w motoryzacji - np. w samochodach i motocyklach napędzanych silnikami spalinowymi. Zadaniem silnika krokowego jest stabilizacji obrotów silnika napędzającego pojazd/ urządzenie na biegu jałowym. Dzięki prostej konstrukcji silniki krokowe są niezawodne i łatwe w obsłudze. [4]

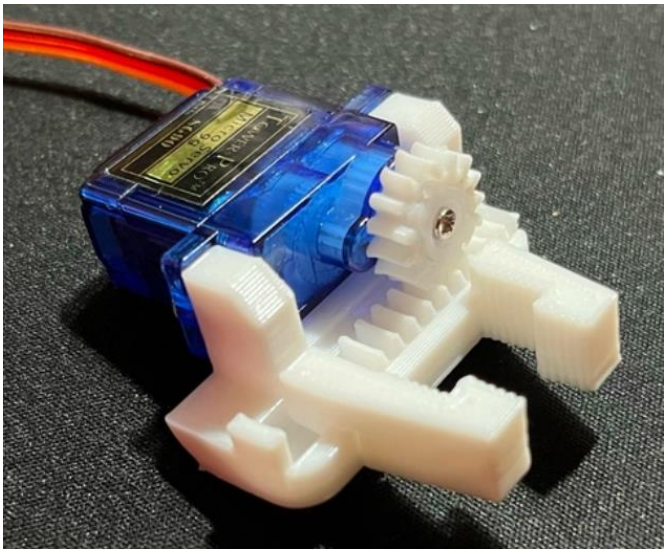
2.2. Środowisko projektowe

Atmel Studio (w wersji 6.1 lub wyższej, np. 7.0) to darmowe środowisko, które można pobrać ze strony producenta – firmę Microchip. Program Atmel Studio bazuje na Microsoft Visual Studio jako na środowisku programisty, w którym wykorzystywane są różne kompilatory (C, C++, Assembler). W środowisku projektowym pakiet programów GNU Compiler Collection zawiera min. program GCC, który jest kompilatorem języka wysokiego poziomu C. Moduł symulatora działania programu umożliwia programowanie i testowanie działania programu bez sprzętu (offline) – dzięki bibliotekom mikrosterowników AVR i ARM, jak i po zaimplementowaniu kodu maszynowego na kontroler (symulator online). Moduły: symulatora i programatora JTAG umożliwiają śledzenie w czasie rzeczywistym stanu pojedynczych bitów w całej, rozległej przestrzeni pamięci mikrokontrolera. Dla lepszego przybliżenia działania układów scalonych AVR, a także reguł algebry stosowanej w elektronice cyfrowej będzie wykorzystywany język asemblera (niższego poziomu). [5]

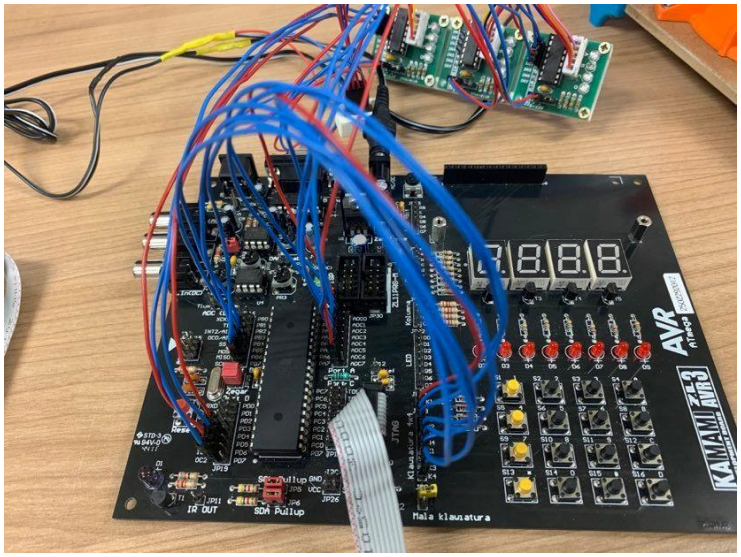
3. Budowa układu



Rysunek 3.1. Urządzenie wykorzystane w projekcie. [7]



Rysunek 3.2. Servo zastosowane w projekcie. [7]



Rysunek 3.3. Płyta z mikrokontrolerem ZL3AVR ATMEGA32 (zdjęcie własne).

W skład układu projektowego wchodzą cztery główne zespoły:

1) Ramię robota/ manipulatora, które składa się z:
    - Trzech silników krokowych (28BYJ-48),
    - Elementów ruchomych (połączenia śrubowe, dzięki którym realizowany jest ruch translacyjny i obrotowy),
    - Jednego silnika Servo (MG90).
2) Płyta z mikrokontrolerem ZL3AVR ATMEGA32 (tak zwane „centrum sterowania"), dodatkowo z zestawem przewodów łączących i zasilaczem
3) Sterownik silnika krokowego i układ diod LED, które wskazują zadaną fazę zasilania układu
4) Urządzenie PC, na którym przygotowano kod dzięki programowi Assembler. Środowisko projektowe zostało opisane w punkcie 2.2.

4. Napędy stosowane w manipulatorach/robotach/urządzeniach chwytających

Urządzenia chwytające mogą być napędzane za pomocą napędu:

- Pneumatycznego,
- Elektrycznego,
- Hydraulicznego.

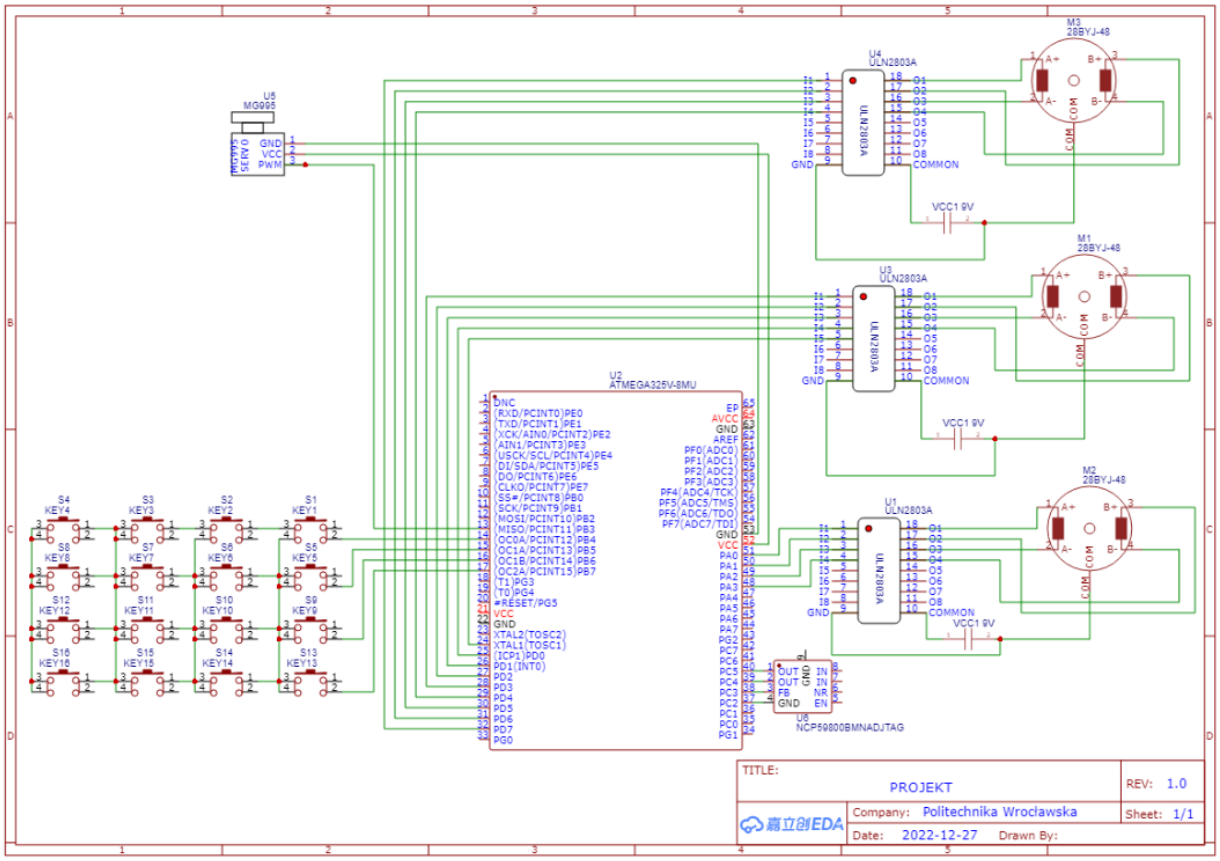
Napęd pneumatyczny działa na zasadzie energii sprężonego powietrza, wykorzystuje się ją do: przemieszczania tłoczyska siłownika, realizacji czynności pomocniczych, odkształcania elastycznych końcówek lub przepon kształtowych lub wytwarzania podciśnienia w przyssawkach. Żądane wartości przemieszczeń tłoczyska decydują o zastosowaniu jednego z poniższych siłowników: tłokowe, membranowe oraz mieszkowe. Najczęściej wykorzystywane są siłowniki tłokowe jednostronnego działania, gdzie końcówki chwytne rozwierane są pod wpływem sprężonego powietrza, natomiast zwieranie dokonuje się pod wpływem siły sprężyny zwrotnej.

Napędy elektryczne są szeroko stosowane w przemyśle, przeznaczone są do zastosowania w chwytakach robotów przemysłowych. W skład napędu elektrycznego wchodzą: przekładnie, przekazujące napęd z silnika, serwonapędy, zawierające: silniki, czujniki sprzężenia zwrotnego, regulatory (prądu, prędkości, położenia), mechanizm (układ kinematyczny) przeniesienia napędu na końcówki chwytne. Wymagania jakie muszą spełnić napędy elektryczne to między innymi: niewielka masa silników, stabilna praca silnika w stanie uchwycenia, możliwość sterowania pracą silnika w różnych warunkach, ekonomiczność rozwiązania oraz bezpieczeństwo pracy.

Napędy hydrauliczne, które pomału zostają wyparte przez napędy elektryczne nadal należą do podstawowej grupy napędów. Szczególnie dobrze odnajdują się w momencie potrzeby szybkiego przemieszczenia przy znaczącym obciążeniu robota. Dodatkowo posiadają bardzo dużą zaletę, jaką jest krótki czas rozruchu, trwający od kilkudziesięciu milisekund od sekundy. [6]

5. Schemat układu

Za pomocą programu **EasyEDA** narysowano schemat połączeń silników krokowych i Servo, znajdujących się w układzie manipulatora, z płytą ATMEGA32 i klawiaturą oraz JTAG.



Rysunek 5.1. Schemat przedstawiający cały układ sterujący.

6. Przebieg stworzenia projektu

1) **Założenie nowego projektu**- wybrano język programowania, w którym zostanie utworzony kod (Assembler). Następnie wybrano mikroprocesor ATMEGA32 oraz programator AVR Dragon wraz z interfejsem JTAG.
2) **Uruchomienie silników krokowych i ich sterowanie**- impulsy elektryczne zadawane na silniki krokowe są przekształcane w ruchy mechaniczne manipulatora. Dzięki ustawianiu odpowiednich wartości na bitach użytkownik kontroluje kierunek obrotu i częstotliwość kręcenia silników.
3) **Uruchomienie klawiatury**- klawiatura 4x4, która odpowiada za sterowanie urządzeniem zamontowana jest na płytce. Dodatkowo należy pamiętać o zdjęciu zworki znajdującej się przy klawiaturze, tak by umożliwić użytkownikowi korzystanie z całej klawiatury, a nie tylko z przycisków oznaczonych w kolorze (w przypadku płytki projektowej- żółtym).
4) **Uruchomienie Servo i sterowanie**- w chwytaku, na zakończeniu ramienia robota znajduje się zamontowane Servo, które spełnia funkcję otwierania i zamykania, tak by móc chwytać wybrane elementy przez użytkownika.
5) **Przetestowanie pełnego uruchomienia i poprawienie ewentualnych błędów**- podczas tworzenia projektu niektóre przyciski w klawiaturze, z powodu zużycia nie odpowiadały na zadane polecenia. Dlatego, też na potrzeby sprawnego korzystania z projektu zmieniano przyciski na klawiaturze, tak by projekt spełniał wszystkie zadane mu funkcje.

7. Projekt (kod) w programie Microchip Studio

Poniżej znajdują się dwie części kodu, pierwsza z nich jest aktywacją programu i wprowadzeniem procedury, która ma za zadanie sprawdzenie jaki przycisk na klawiaturze został wciśnięty.

Część I

```
.include "m32def.inc"             ; dołączenie pliku z definicjami rejestrów
.def licznik1 = R20
.def licznik2 = R21
.def licznik3 = R22

jmp Start

; ******** miejsce na ISRy przerwan ********
nop
nop
ldi R28, 0b11000011
out DDRC, R28
; ******** AKTYWACJA STOSU ********
Start:                      ; etykieta programu głównego
       ldi R17, high(RAMEND)      ; ładuj rejestr roboczy MSB adresu końca pamięci
SRAM
       out SPH, R17         ; inicjalizuj MSB wskaźnika stosu
       ldi R17, low(RAMEND)       ; ładuj rejestr roboczy LSB adresu końca pamięci
SRAM
       out SPL, R17         ; inicjalizuj LSB wskaźnika stosu
; ******** PETLA GLOWNA ********
       clr R16
       out DDRA, R16 ; załadowanie do rejestru kierunku portu A wartości z rejestru r16
celem ponownego udrożnienia komunikacji
       out PORTA, R16       ; załadowanie do rejestru danych portu A wartości z
rejestru r16 celem ponownego udrożnienia komunikacji
       out DDRB, R16 ; załadowanie do rejestru kierunku portu B wartości z rejestru r16
celem ponownego udrożnienia komunikacji
       out PORTB, R16       ; załadowanie do rejestru danych portu B wartości z
rejestru r16 celem ponownego udrożnienia komunikacji
       ser R19       ; zapisanie w rejestrze roboczym wartości FF
       out DDRD, R19 ; załadowanie do rejestru kierunku portu D wartości z rejestru
roboczego celem wyswietlania na diodach nr przyciskow
       out PORTD, R16       ; załadowanie do rejestru danych portu D wartości z
rejestru r16 celem ponownego udrożnienia komunikacji
; ******** TESTOWANIE W KTOREJ KOLUMNIE WCISNIETO SWITCH ********
Kolumna:
       ldi R16, 0xff        ; zapisanie w rejestrze 16 wartości FF
       out DDRA, R16 ; załadowanie do rejestru kierunku portu A wartości z rejestru r16
dla zmiany kierunku portu A
       ldi R16, 0x0f ; przygotowanie rejestru pod DDRB aby mlodsza czesc była nadająca
a starsza z pull-up
       out DDRb, R16 ; port B bedzie portem, w którym będzie prowadzona rejestracja
zmian stanow pinow
       ldi R16, 0xff ;
       out portb, R16       ; nadawanie z pull-up
       nop           ; czas na wcisniecie switcha
       nop           ; czas na wcisniecie switcha
       nop           ; czas na wcisniecie switcha
       in R18, PINB         ; kopiowanie stanu portu b do badania czy jakis switch w
wierszu nie zostal wcisniety
       nop           ; sprawdzenie w oknie Procesor czy skopiowal
       nop           ; sprawdzenie w oknie Procesor czy skopiowal
```

```
; ******** TESTOWANIE BITOW W REJESTRZE ZE SKOKIEM NAD 1 INSTRUKCJA ********
	sbrs r18, PB4 ; jesli nie ma wcisnietego to przeskocz nad rcall do kolejnego
testu kolejnego bitu (PB4 to maska bitu nr 4)
	rcall ktory_wiersz1  ; jesli wcisnieto, to przejdz do kolejnego etapu testowania
z wykrywaniem ktory to switch
	sbrs r18, PB5
	rcall ktory_wiersz2
	sbrs r18, PB6
	rcall ktory_wiersz3
	sbrs r18, PB7
	rcall ktory_wiersz4
	rjmp Start    ; powrót do programu glownego z przywroceniem pierwotnej
konfiguracji portów


; ******** PONIZEJ PROCEDURY KTORY SWITCH ZOSTAL WCISNIETY ********
ktory_wiersz1:
	ldi R17, 0xff       ;
	out DDRB, r17 ; teraz port B będzie służył do odbierania sygnalow z portu A
	ldi R17, 0x00 ;
	out PORTB, r17      ;
	ldi r17, 0xf0 ;
	out DDRA, r17 ; port A bedzie portem, w którym będzie prowadzona rejestracja
zmian stanow pinow
	ldi r17, 0xff;
	out PORTA, r17 ;pull-up
	nop           ; czas na wcisniecie switcha
	nop           ; czas na wcisniecie switcha
	nop           ; czas na wcisniecie switcha
	in r18, PINA        ; kopiowanie stanu portu A do badania czy jakis przycisk
nie zostal wcisniety
	nop           ; sprawdzenie w oknie Procesor czy skopiowal
	nop           ; sprawdzenie w oknie Procesor czy skopiowal
; ******** TESTOWANIE BITOW W REJESTRZE ZE SKOKIEM NAD 1 INSTRUKCJA ********
	sbrs r18, PA0
	rcall przycisk_s1
	sbrs r18, PA1
	rcall przycisk_s5
	sbrs r18, PA2
	rcall przycisk_s9
	sbrs r18, PA3
	rcall przycisk_s13
	ret           ; powrót z wywołania podprogramu instukcja rcall
; *******
ktory_wiersz2:
	ldi R17, 0xff ; zapisanie w rejestrze 17 wartości FF
	out DDRB, r17; załadowanie do rejestru kierunku portu A wartości z rejestru r17
celem udrożnienia komunikacji
	ldi R17, 0x00 ; zapisanie w rejestrze 17 wartości FF
	out PORTB, r17; załadowanie do rejestru kierunku portu A wartości z rejestru r17
celem udrożnienia komunikacji
	ldi r17, 0xf0 ;przygotowanie rejestru pod ddrb
	out DDRA, r17; załadowanie do rejestru kierunku portu b wartości z rejestru
celem ustalenia kierunku na nadajnik z pull-up
	ldi r17, 0xff;
	out PORTA, r17 ;pull-up
	nop           ; czas na wcisniecie switcha
	nop           ; czas na wcisniecie switcha
	nop           ; czas na wcisniecie switcha
	in r18, PINA  ;kopiowanie stanu portu b do badania czy jakis przycisk w wierszu
nie zostal wcisniety
	nop
```

```
        sbrs r18, PA0
        rcall przycisk_s2
        sbrs r18, PA1
        rcall przycisk_s6
        sbrs r18, PA2
        rcall przycisk_s10
        sbrs r18, PA3
        rcall przycisk_s14
        ret
; *******
ktory_wiersz3:
        ldi R17, 0xff ; zapisanie w rejestrze 17 wartości FF
        out DDRB, r17; załadowanie do rejestru kierunku portu A wartości z rejestru r17
celem udrożnienia komunikacji
        ldi R17, 0x00 ; zapisanie w rejestrze 17 wartości FF
        out PORTB, r17; załadowanie do rejestru kierunku portu A wartości z rejestru r17
celem udrożnienia komunikacji
        ldi r17, 0xf0 ;przygotowanie rejestru pod ddrb
        out DDRA, r17; załadowanie do rejestru kierunku portu b wartości z rejestru
celem ustalenia kierunku na nadajnik z pull-up
        ldi r17, 0xff;
        out PORTA, r17 ;pull-up
        nop           ; czas na wcisniecie switcha
        nop           ; czas na wcisniecie switcha
        nop           ; czas na wcisniecie switcha
        in r18, PINA  ;kopiowanie stanu portu b do badania czy jakis przycisk w wierszu
nie zostal wcisniety
        nop
        sbrs r18, PA0
        rcall przycisk_s3
        sbrs r18, PA1
        rcall przycisk_s7
        sbrs r18, PA2
        rcall przycisk_s11
        sbrs r18, PA3
        rcall przycisk_s15
        ret
; *******
ktory_wiersz4:
        ldi R17, 0xff ; zapisanie w rejestrze 17 wartości FF
        out DDRB, r17; załadowanie do rejestru kierunku portu A wartości z rejestru r17
celem udrożnienia komunikacji
        ldi R17, 0x00 ; zapisanie w rejestrze 17 wartości FF
        out PORTB, r17; załadowanie do rejestru kierunku portu A wartości z rejestru r17
celem udrożnienia komunikacji
        ldi r17, 0xf0 ;przygotowanie rejestru pod ddrb
        out DDRA, r17; załadowanie do rejestru kierunku portu b wartości z rejestru
celem ustalenia kierunku na nadajnik z pull-up
        ldi r17, 0xff;
        out PORTA, r17 ;pull-up
        nop           ; czas na wcisniecie switcha
        nop           ; czas na wcisniecie switcha
        nop           ; czas na wcisniecie switcha
        in r18, PINA  ;kopiowanie stanu portu b do badania czy jakis przycisk w wierszu
nie zostal wcisniety
        nop
        sbrs r18, PA0
        rcall przycisk_s4
        sbrs r18, PA1
        rcall przycisk_s8
        sbrs r18, PA2
        rcall przycisk_s12
```

```
        sbrs r18, PA3
        rcall przycisk_s16
        ret
```

Druga część kodu, to projekt autorów, który przypisuje odpowiednim przyciskom wartości zadane i powoduje obrót w prawo lub w lewo (okręcenie lub odkręcenie) silnika krokowego. Następnie procedura sterowania PWM, która odpowiada za rozwieranie i zwieranie chwytaka. Dodatkowo została wprowadzona procedura opóźnienia, odpowiadająca za odpowiedni czas przeskoku pomiędzy wartościami i prawidłowe działanie układu.

Część II

```
; ******* PRZYPISANIE FUNKCJI WYKONAWCZEJ POD KAZDY ZE SWITCHOW *******
; ************************************************
; ******* ZESTAW SWITCHOW W PIERWSZYM WIERSZU ********
; ************************************************
przycisk_s1:

Procedura_krecenia_s1:
ldi R16, 0b00010000
out PORTA, R16
rcall Opoznienie
nop
ldi R16, 0b00110000
out PORTA, R16
rcall Opoznienie
nop
ldi R16, 0b00100000
out PORTA, R16
rcall Opoznienie
nop
ldi R16, 0b011000000
out PORTA, R16
rcall Opoznienie
nop
ldi R16, 0b01000000
out PORTA, R16
rcall Opoznienie
nop
ldi R16, 0b11000000
out PORTA, R16
rcall Opoznienie
nop
ldi R16, 0b10000000
out PORTA, R16
rcall Opoznienie
nop
ldi R16, 0b10010000
out PORTA, R16
rcall Opoznienie
nop
ret



; *******
przycisk_s5:
Procedura_krecenia_s5:
ldi R16, 0b10000000
out PORTA, R16
```

```
rcall Opoznienie
nop
ldi R16, 0b01000000
out PORTA, R16
rcall Opoznienie
nop
ldi R16, 0b00100000
out PORTA, R16
rcall Opoznienie
nop
ldi R16, 0b00010000
out PORTA, R16
rcall Opoznienie
nop
ret    ; i tak w nieskonczonosc…

przycisk_s9:
sbi PORTD, 0
sbi PORTD, 3
nop
nop
nop
ret
; *******
przycisk_s13:
sbi PORTD, 0
sbi PORTD, 2
sbi PORTD, 3
nop
nop
nop
ret
; ******* ZESTAW SWITCHOW W DRUGIM WIERSZU ********
; ************************************************
przycisk_s2:

Procedura_krecenia_s2:
ldi R16, 0b00010000
out PORTD, R16
rcall Opoznienie
nop
ldi R16, 0b00100000
out PORTD, R16
rcall Opoznienie
nop
ldi R16, 0b01000000
out PORTD, R16
rcall Opoznienie
nop
ldi R16, 0b10000000
out PORTD, R16
rcall Opoznienie
nop
ret    ; i tak w nieskonczonosc…

; *******
przycisk_s10:
Procedura_krecenia_s10:
ldi R16, 0b11000000
out PORTD, R16
rcall Opoznienie
nop
```

```
ldi R16, 0b01100000
out PORTD, R16
rcall Opoznienie
nop
ldi R16, 0b00110000
out PORTD, R16
rcall Opoznienie
nop
ldi R16, 0b10010000
out PORTD, R16
rcall Opoznienie
nop
ret    ; i tak w nieskonczonosc…

przycisk_s6:
sbi PORTD, 1
sbi PORTD, 3
nop
nop
nop
ret
; *******
przycisk_s14:
sbi PORTD, 1
sbi PORTD, 2
sbi PORTD, 3
nop
nop
nop
ret

; ******* ZESTAW SWITCHOW W TRZECIM WIERSZU ********
; ************************************************
przycisk_s3:
Procedura_krecenia_s3:
ldi R16, 0b00001000
out PORTD, R16
rcall Opoznienie
nop
ldi R16, 0b00000100
out PORTD, R16
rcall Opoznienie
nop
ldi R16, 0b00000010
out PORTD, R16
rcall Opoznienie
nop
ldi R16, 0b00000001
out PORTD, R16
rcall Opoznienie
nop
ret    ; i tak w nieskonczonosc…

przycisk_s7:
Procedura_krecenia_s7:
ldi R16, 0b00000011
out PORTD, R16
rcall Opoznienie
nop
ldi R16, 0b00000110
out PORTD, R16
rcall Opoznienie
```

```
nop
ldi R16, 0b00001100
out PORTD, R16
rcall Opoznienie
nop
ldi R16, 0b00001001
out PORTD, R16
rcall Opoznienie
nop
ret    ; i tak w nieskonczonosc…

przycisk_s11:
sbi PORTD, 0
sbi PORTD, 1
sbi PORTD, 3
nop
nop
nop
ret
; *******
przycisk_s15:
sbi PORTD, 0
sbi PORTD, 1
sbi PORTD, 2
sbi PORTD, 3
nop
nop
nop
ret

; ******* ZESTAW SWITCHOW W CZWARTYM WIERSZU ********
; ************************************************
przycisk_s4:
rcall Procedura_krecenia_w_lewo_pwm1
nop
nop
nop
ret; *******

przycisk_s8:
rcall Procedura_krecenia_prawo_pwm2
nop
nop
nop
ret; *******

przycisk_s12:
sbi PORTD, 2
sbi PORTD, 3
nop
nop
nop
ret
; *******
przycisk_s16:
sbi PORTD, 4
nop
nop
nop
ret
```

```
Procedura_krecenia_w_lewo_pwm1:
	ldi R26, 10 ; wartosc 10 bedzie odpowiadala zadanemu przez uĹźytkownika
polozeniu orczyka na serwie
	sbi DDRB, 3 		; na PB3 będzie generowane PWM, praca w trybie
wyjś?ciowym
	ldi R17, (1<<WGM00) | (1<<WGM01) | (1<<COM01) | (0<<COM00) | (1<<CS00) |
(1<<CS01) | (0<<CS02)	; uzupeĹ?nij # aby praca PWM byla w  trybie nieodwracajÄ
	out TCCR0, R17	; ustawienie rejestru TCCR0 w celu skonfigurowania pracy TC0
	out OCR0, R26	; wyświetlanie z określoną częstotliwością zmian długości
impulsu PWM
	rcall Opoznienie
	nop
	ret

Procedura_krecenia_prawo_pwm2:
	ldi R26, 35 ; drugie skrajne położenie
	sbi DDRB, 3 ; na PB3 będzie generowane PWM, praca w trybie wyjściowym
ldi R17, (1<<WGM00) | (1<<WGM01) | (1<<COM01) | (0<<COM00) | (1<<CS00) | (1<<CS01) |
(0<<CS02)
out TCCR0, R17 ; ustawienie rejestru TCCR0 w celu skonfigurowania pracy TC0
	out OCR0, R26 ; wyświetlanie z określoną częstotliwością zmian długości impulsu
PWM
rcall Opoznienie
	nop
ret



; procedura Opoznienie sluzyc będzie do sterowania czestotliwoscia generowanych
impulsów elektrycznych
;funkcja wywołująca opóźnienie

Opoznienie:
ldi licznik3, 20
LoopA:
ldi licznik2, 10
LoopA0:
ldi licznik1, 10
LoopA1:
dec licznik1
brne LoopA1
dec licznik2
brne LoopA0
dec licznik3
brne LoopA

ret
```

8. Wnioski

Wykonując powyższy projekt stwierdzono, że sam układ ma kilka mocnych i słabych stron. Mianowicie, płytka do podłączeń najprawdopodobniej była już używana wielokrotnie, przez co można było zauważyć zużycie niektórych przycisków powodując, że autorzy projektu podczas tworzenia kodu, musieli uwzględnić ten fakt. Dodatkowo, każda płytka mogła mieć nie działający przycisk klawiatury, przez co należy aktualizować kod do danego stanu układu.

Problemy, jakie towarzyszyły etapowi projektowania często miały powiązanie z kondycją urządzeń na jakich pracowali autorzy raportu. Dodatkowo, najcięższym zadaniem było logiczne rozmieszczenie odpowiednich silników w odpowiednich portach. Podczas tej czynności należało uwzględnić wszelkie przeciążenia układu, jakie mogły wystąpić. W trakcie podłączeń urządzenia pojawiały się błędy „natury ludzkiej", które polegały na złym przyłączeniu przewodów, co powodowało brak przewidywanych rezultatów.

Jednakże dzięki realizacji projektu, autorzy nauczyli się wielu rzeczy z dziedzin takich jak: techniki mikroprocesorowe i automatyka. Podszkolone zostały umiejętności pisania kodów w języku Assembler, które na pewno przydadzą się w dalszej ścieżce edukacyjnej. Dodatkowo, tworząc dokumentację techniczną otrzymali informację na temat różnych rodzajów manipulatorów i robotów.

9. Bibliografia

[1] https://pl.wikipedia.org/wiki/Robot [data dostępu: 20.12.2022]

[2] https://pl.wikipedia.org/wiki/Chwytak_(robotyka) [data dostępu: 20.12.2022]

[3] https://allweld.pl/PWM-modulacja-blog [data dostępu: 20.12.2022]

[4] https://botland.com.pl/32-silniki-krokowe [data dostępu: 20.12.2022]

[5] https://kotwa.pwr.edu.pl/IB_INSTRUKCJA_NR1_2022.pdf [data dostępu: 20.12.2022]

[6] https://wobit.com.pl/oferta/351/robotyka-przemyslowa/chwytaki-przemyslowe/ [data dostępu: 20.12.2022]

[7] https://www.thingiverse.com/thing:2838859 [data dostępu: 20.12.2022]