PROJEKT Z SYSTEMÓW WBUDOWANYCH

Michał Szczygieł

Politechnika Krakowska

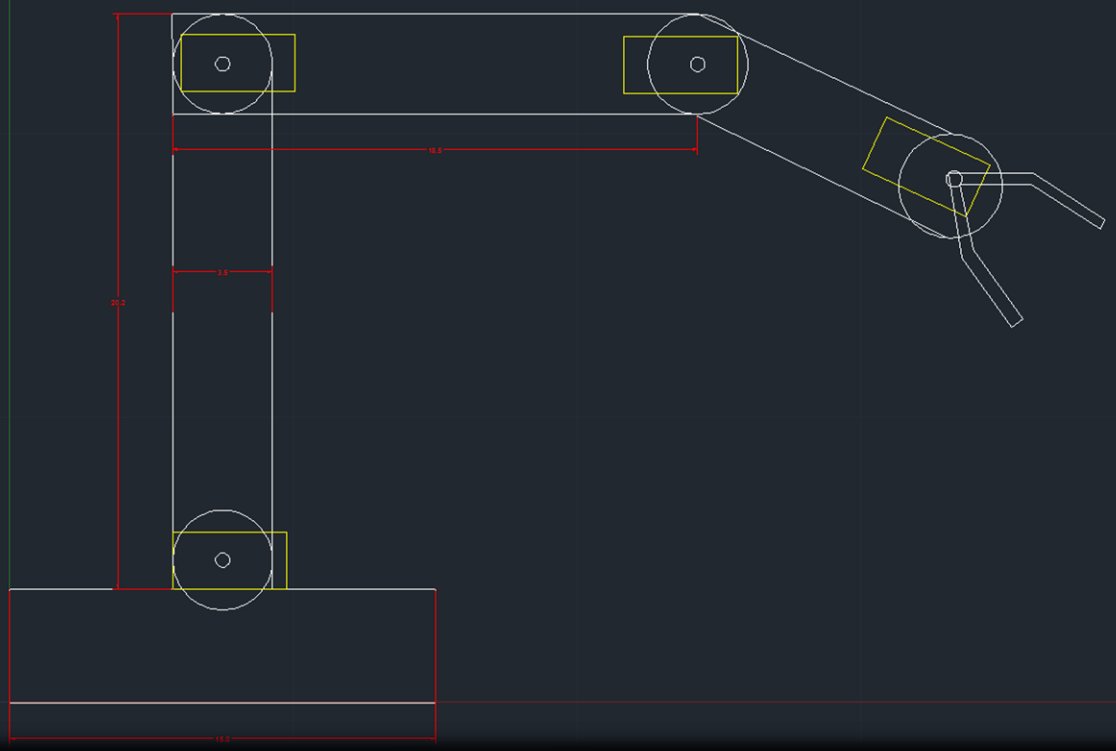
TEMAT : MANIPULATOR MECHANICZNY

Z wykorzystaniem ATMEGI 8

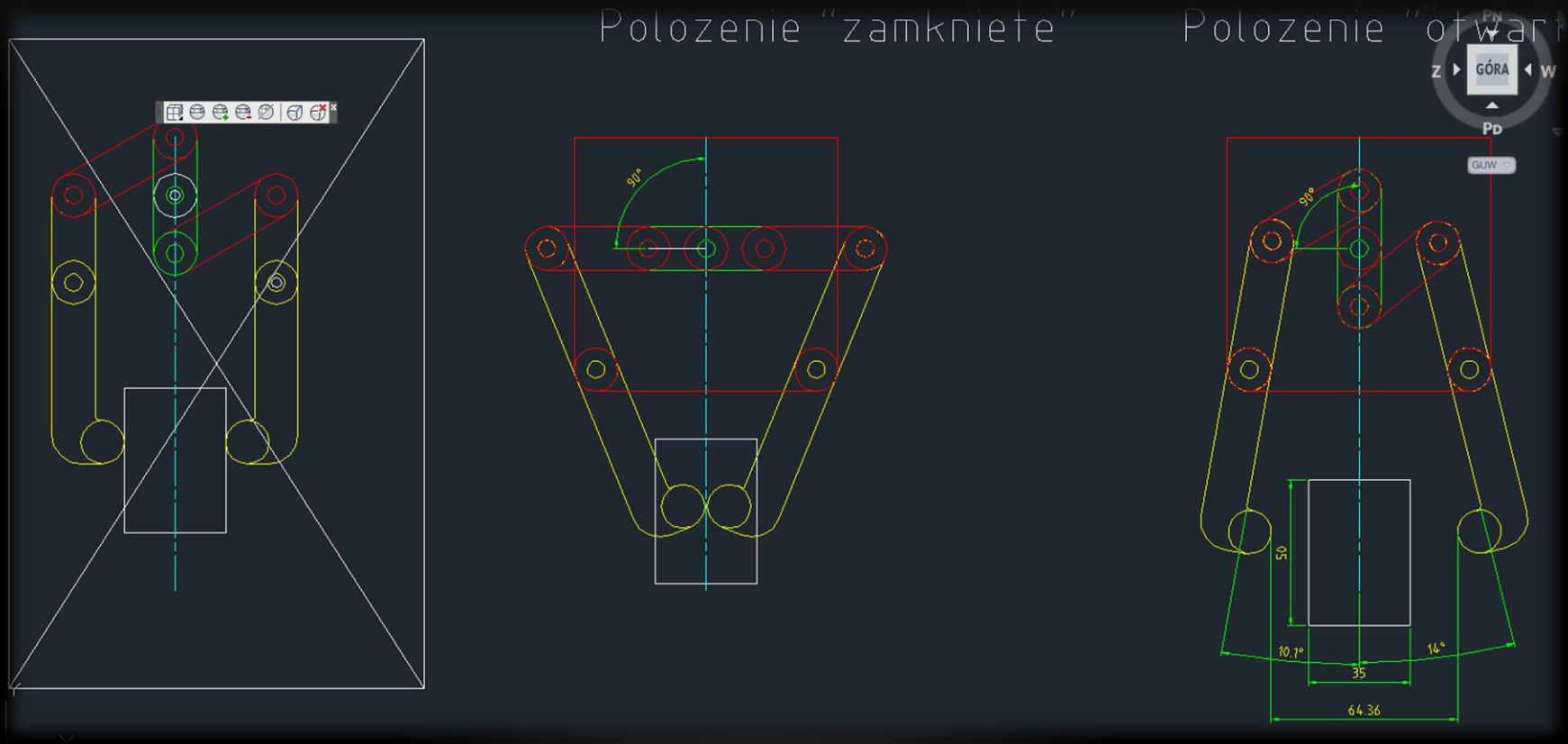
CELE :

ZBUDOWANIE UKŁADU, REALIZUJĄCEGO DANY PROBLEM Z WYKORZYSTANIEM MIKROKONTROLERA ATMEGA 8.
STEROWANIE MANIPULATORA BĘDZIE SIĘ ODBYWAŁO ZA POMOCĄ UARTA.
NAPISANIE OPROGRAMOWANIA UMOŻLIWIAJĄCEGO WYGODNE STEROWANIE MECHANICZNYM RAMIENIEM.

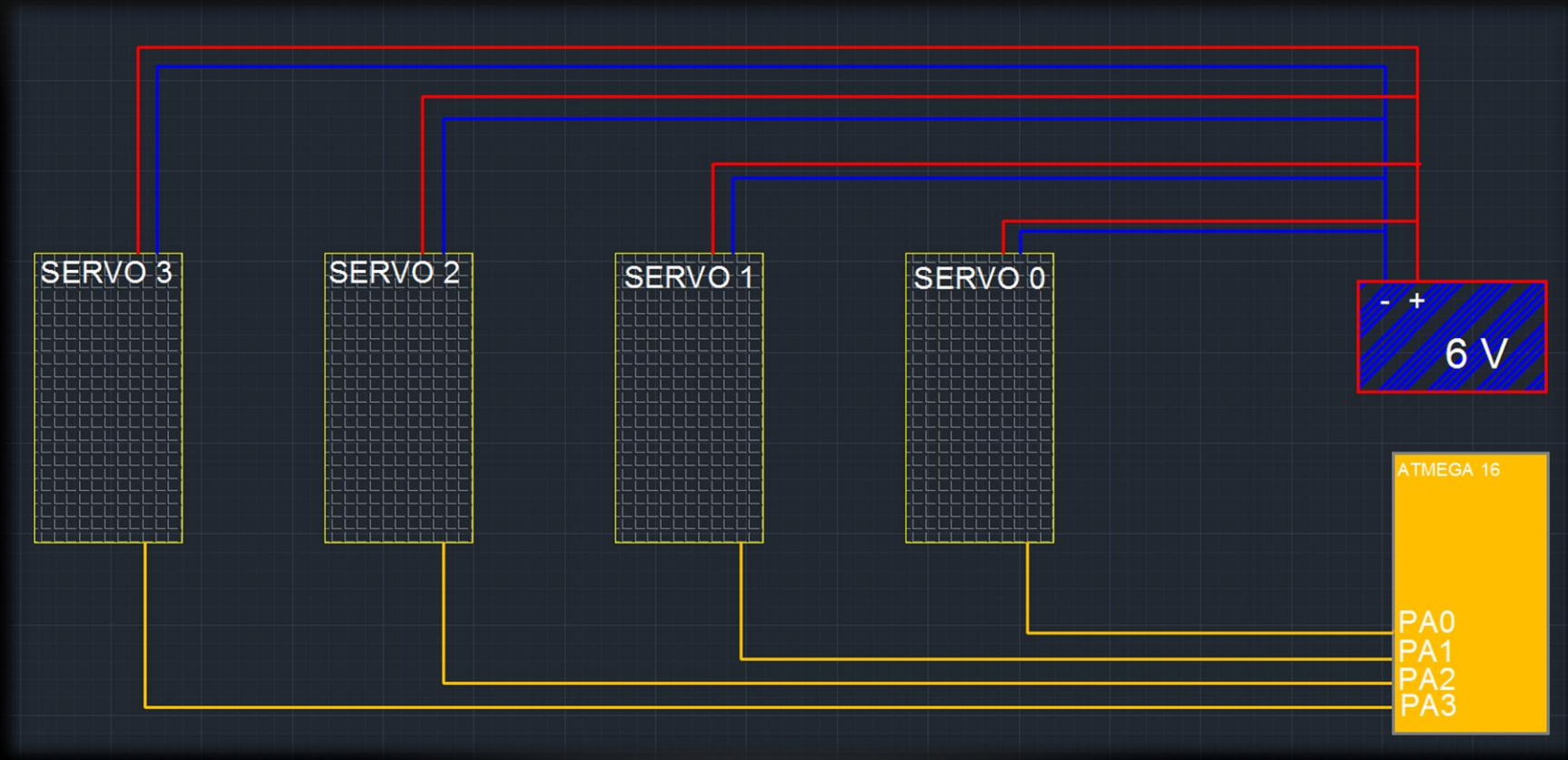
SCHEMAT MECHANICZNY MANIPULATORA



SCHEMAT MECHANICZNY CHWYTAKA

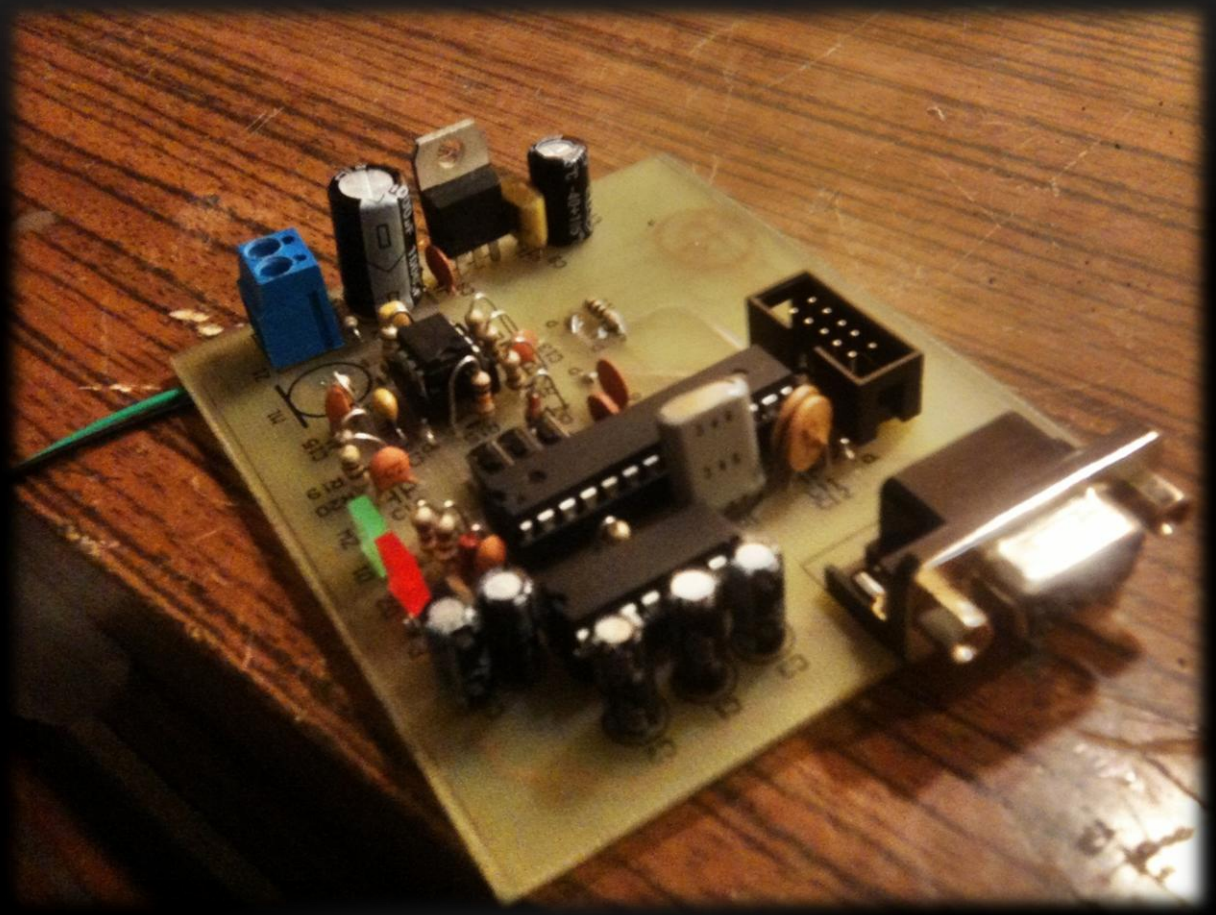


SCHEMAT ELEKTRONICZNY MANIPULATORA

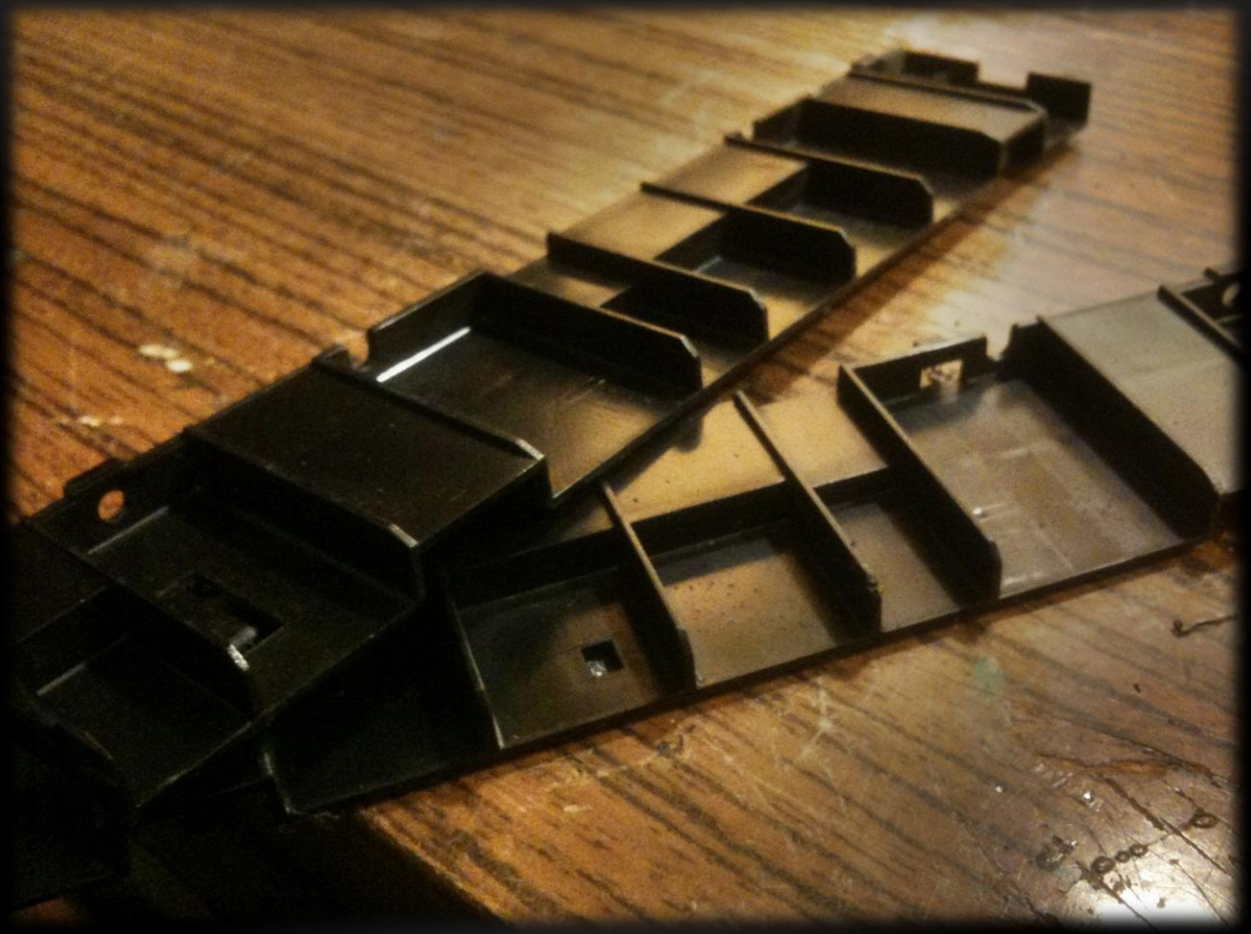


SERVO MECHANIZMY

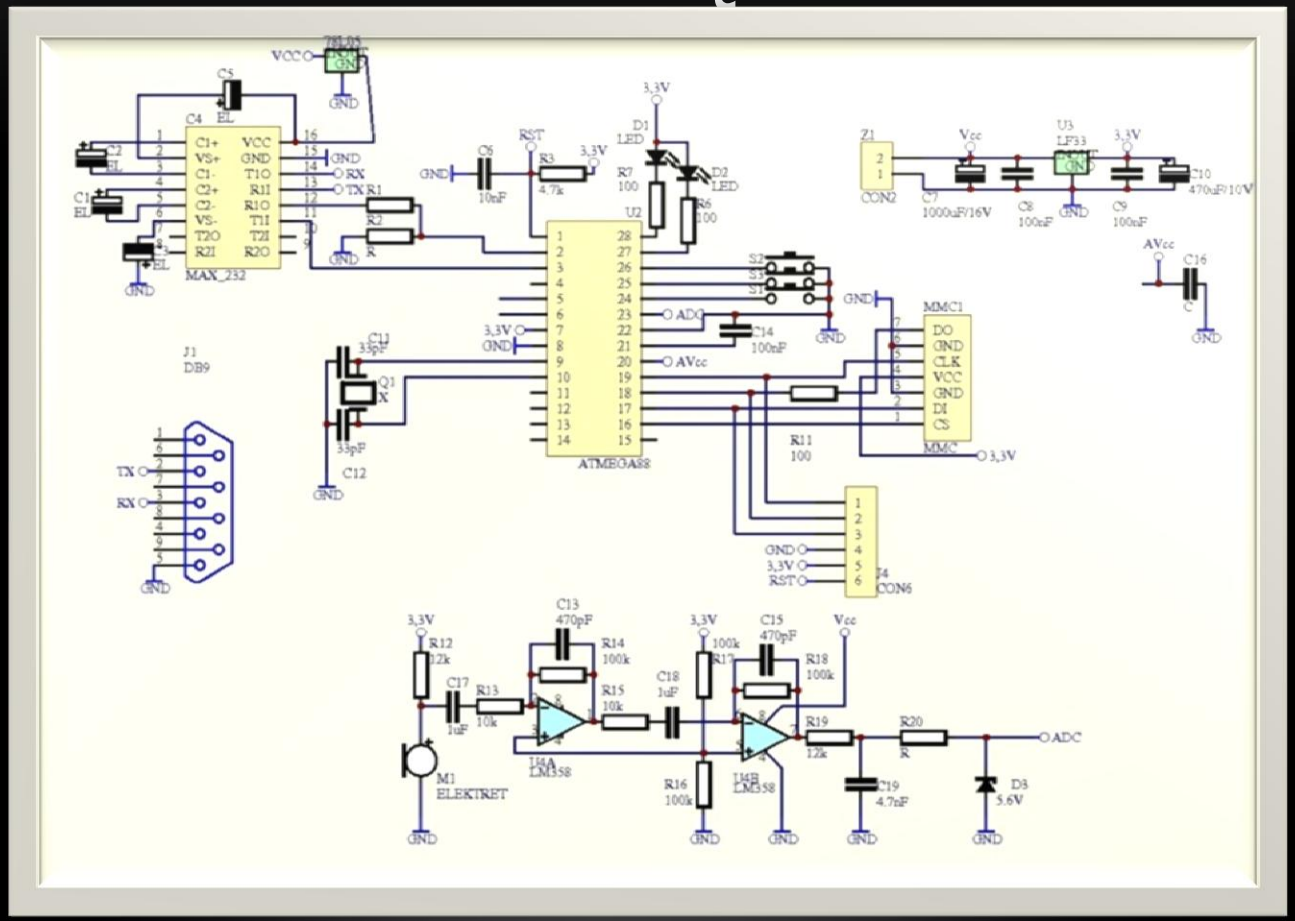
UKŁAD MIKROKONTROLERA



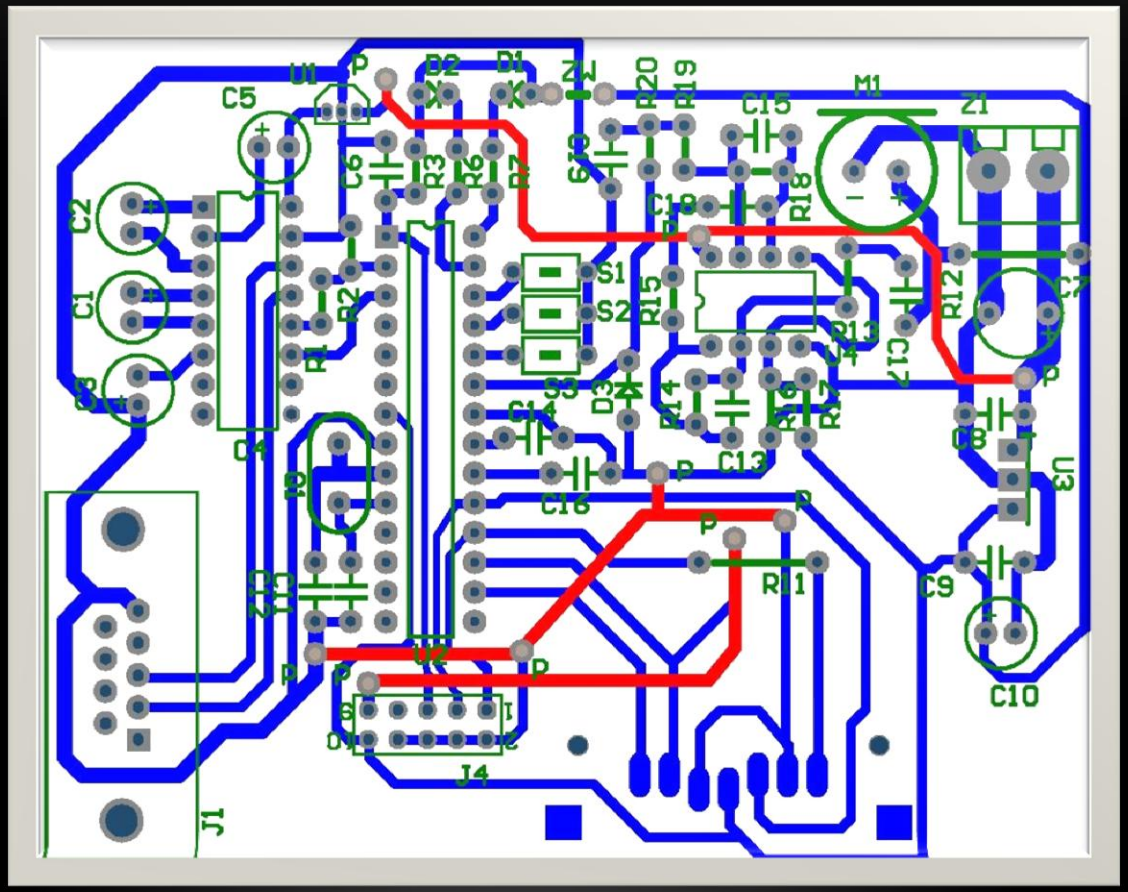
MATERJAŁY KONSTRUKCYJNE



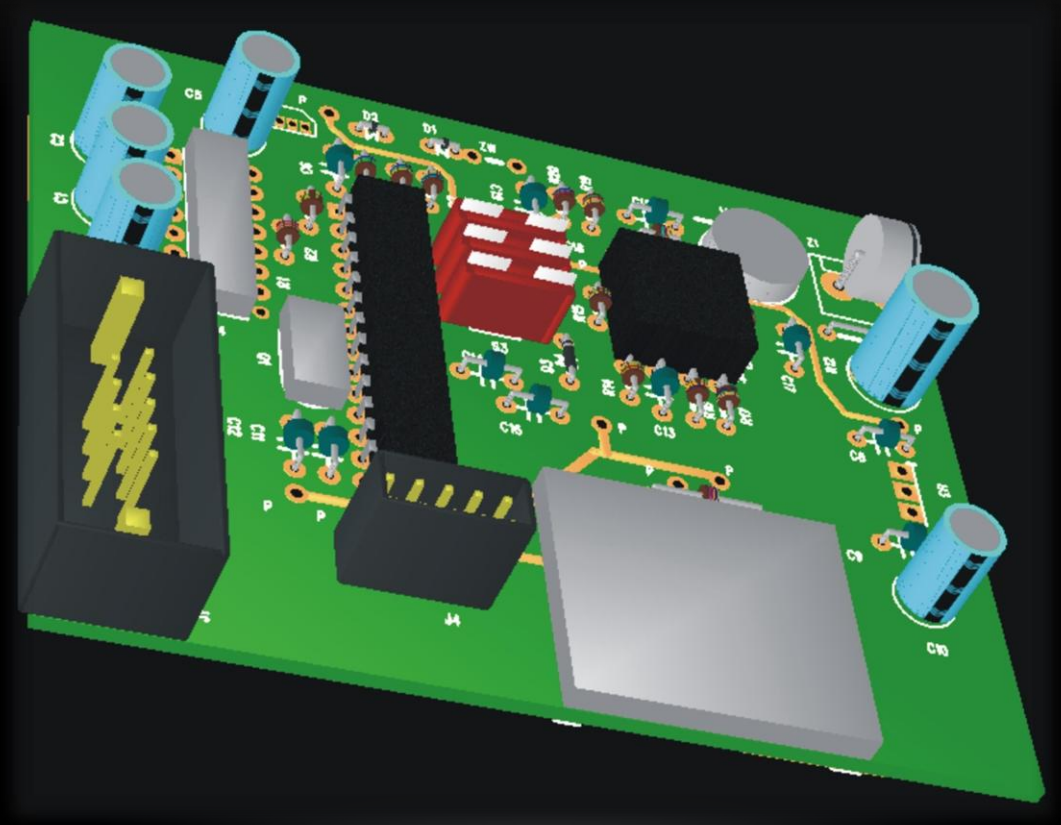
SCHEMAT TECHNICZNY UKŁADU STERUJĄCEGO



SCHEMAT TECHNICZNY UKŁADU STERUJĄCEGO



PROJEKT UKŁADU STERUJĄCEGO W 3D



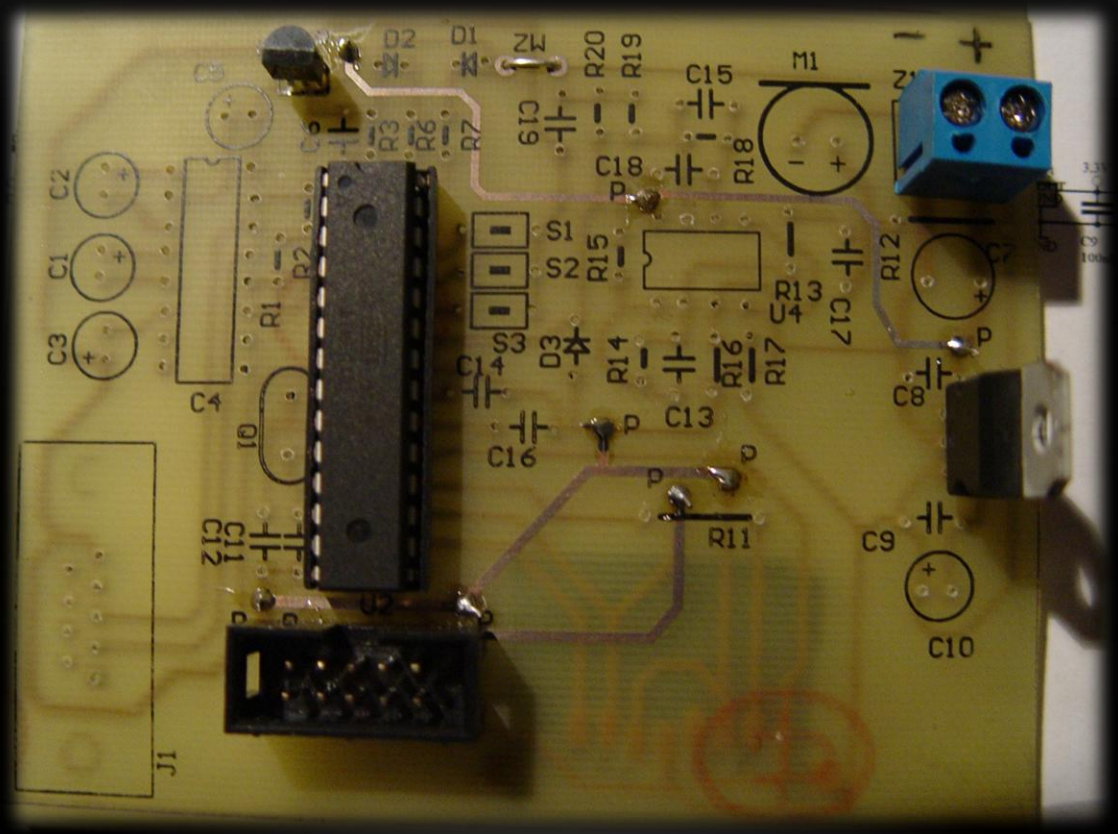
DOKUMENTACJA SPRZĘTOWA:

- Stabilizator 3.3 V 1 A
- przedwzmacniacz operacyjny LM358
- Podstawka 8-pinowa
- Złącze programatora Kanda
- Kwarc 8 MHz
- Interfejs TTL/RS232 MAX232
- Złącze DB9
- Kabel RS232->USB
- Atmega8l + podstawka 28-pinowa
- Mikrofon pojemnościowy
- Czytnik SD
- Dioda Zenera 5.6 V
- Programator AVR ISP II STK500
- Microswitch x 3
- Dioda LED x 2
- Serwo mechanizmy x 4

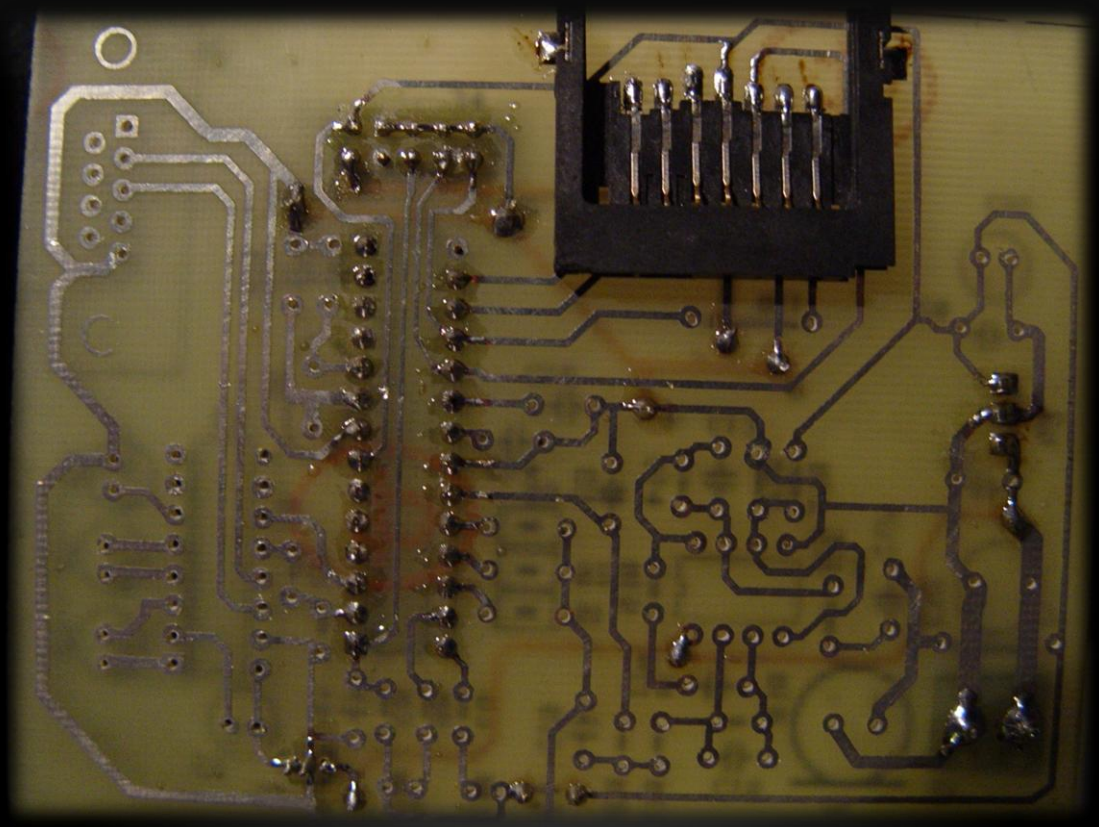
- Kondensator 1 µF elektrolityczny 16V x5
- Kondensator 470 pF elektrolityczny 16V x2
- Kondensator 4.7 nF elektrolityczny 16V x2
- Kondensator 1000 µF elektrolityczny 16V x2
- Kondensator 100 nF ceramiczny x3
- Kondensator 470 µF elektrolityczny 10V x2
- Kondensator 10 nF ceramiczny x2
- Kondensator 33 pF ceramiczny x4
- Kondensator 1 µF foliowy x3

- Rezystor 4.7 kW x10
- Rezystor 100 kW x6
- Rezystor 100 W x5
- Rezystor 12 kW x3

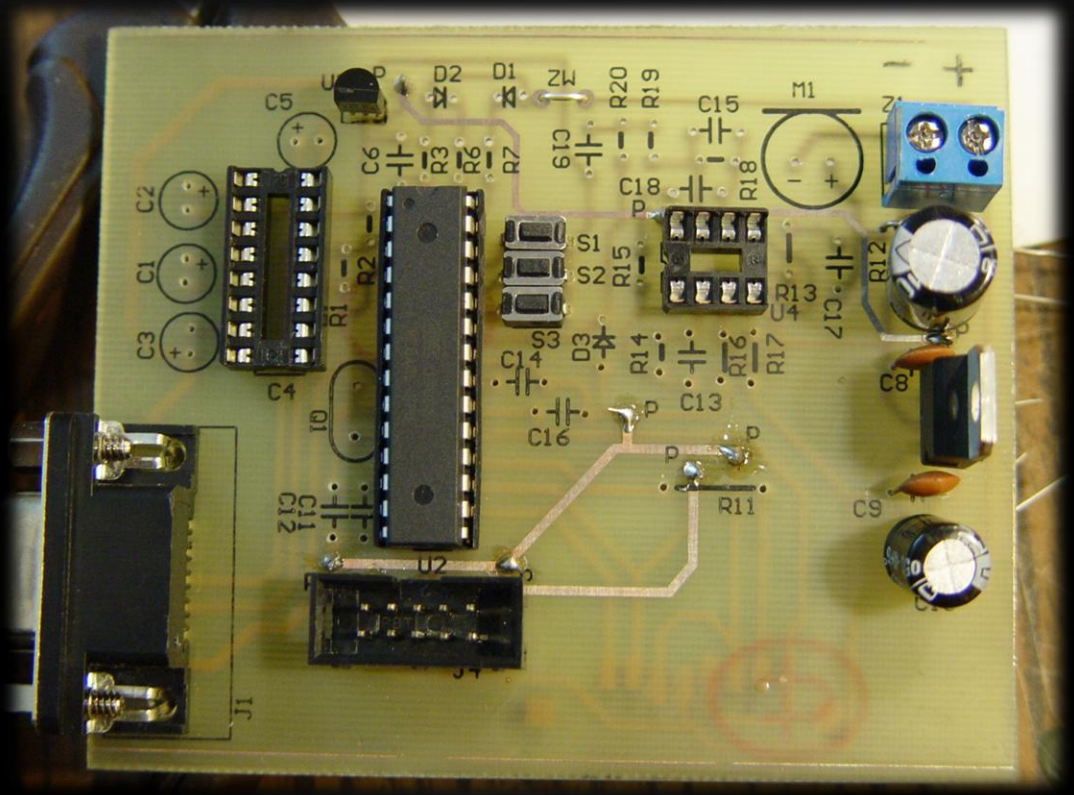
KOLEJNE ETAPY BUDOWANIA UKŁADU STERUJĄCEGO



KOLEJNE ETAPY BUDOWANIA UKŁADU STERUJĄCEGO

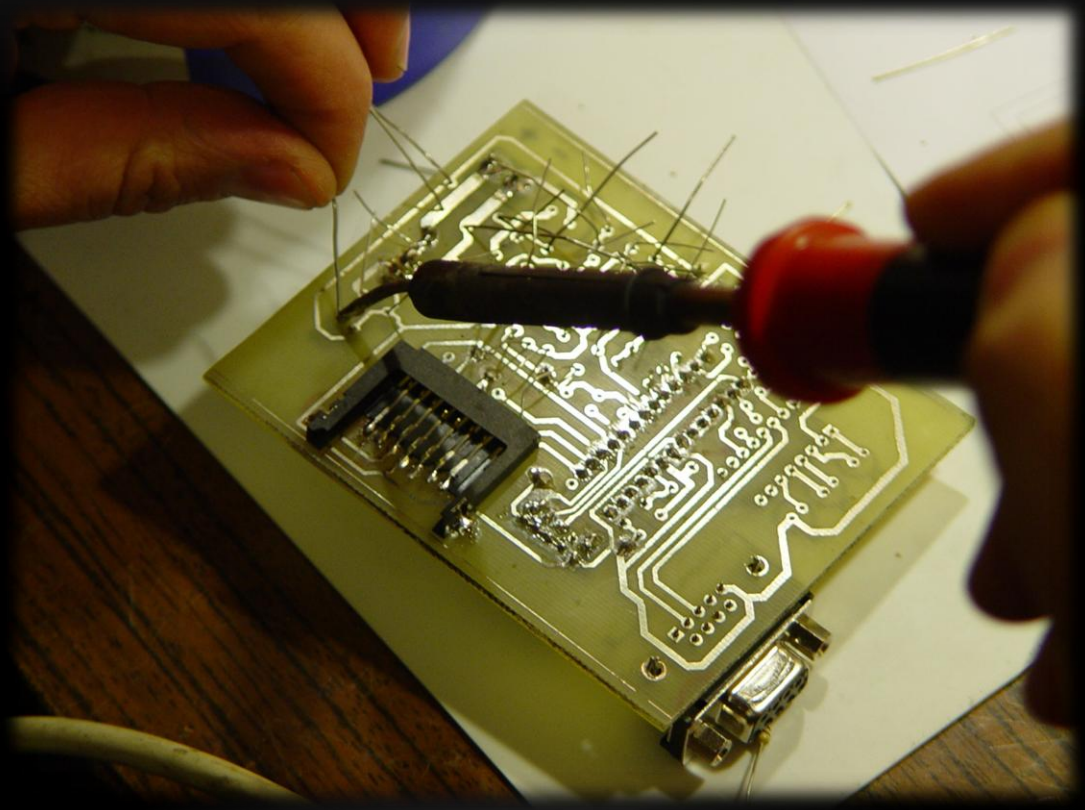


KOLEJNE ETAPY BUDOWANIA UKŁADU STERUJĄCEGO

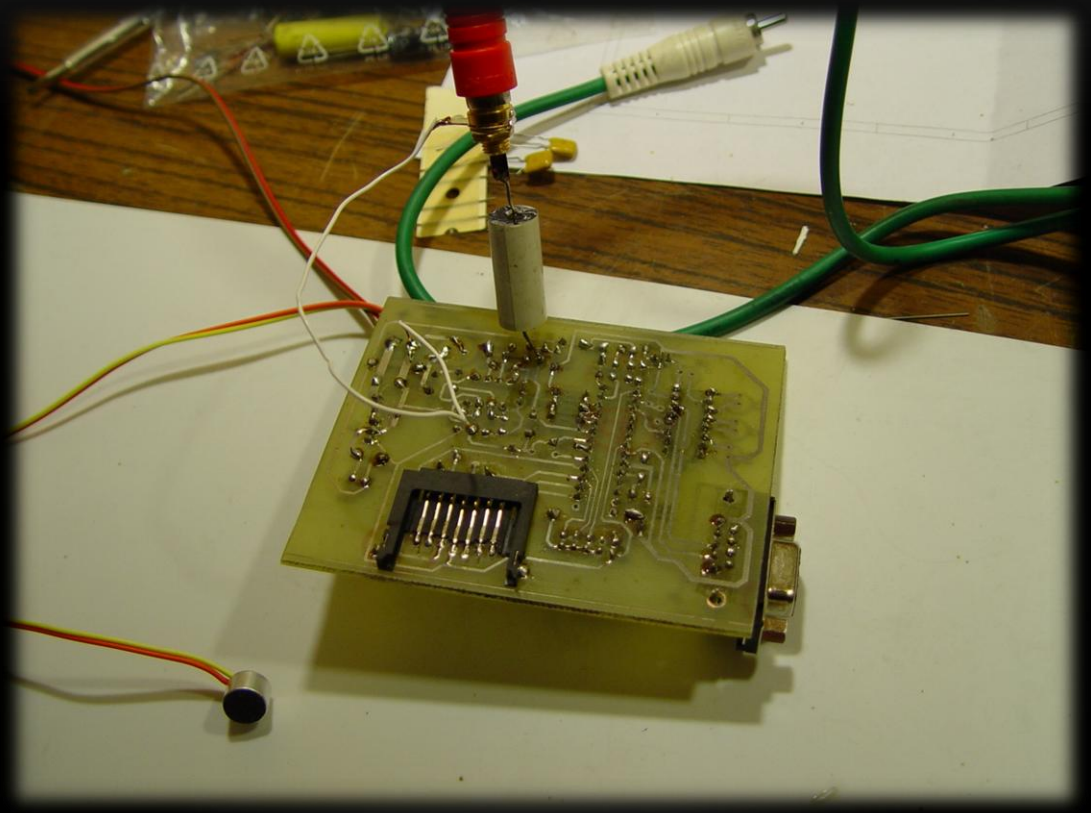


KOLEJNE ETAPY BUDOWANIA UKŁADU STERUJĄCEGO

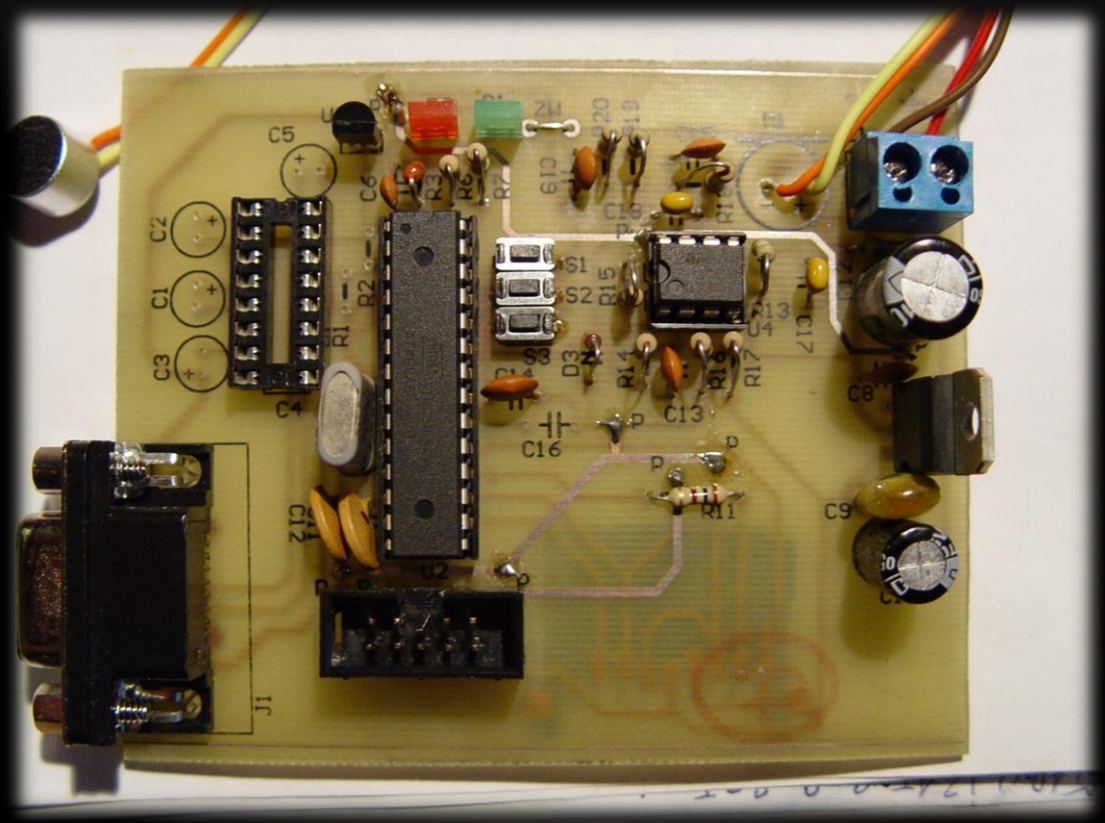
KOLEJNE ETAPY BUDOWANIA UKŁADU STERUJĄCEGO



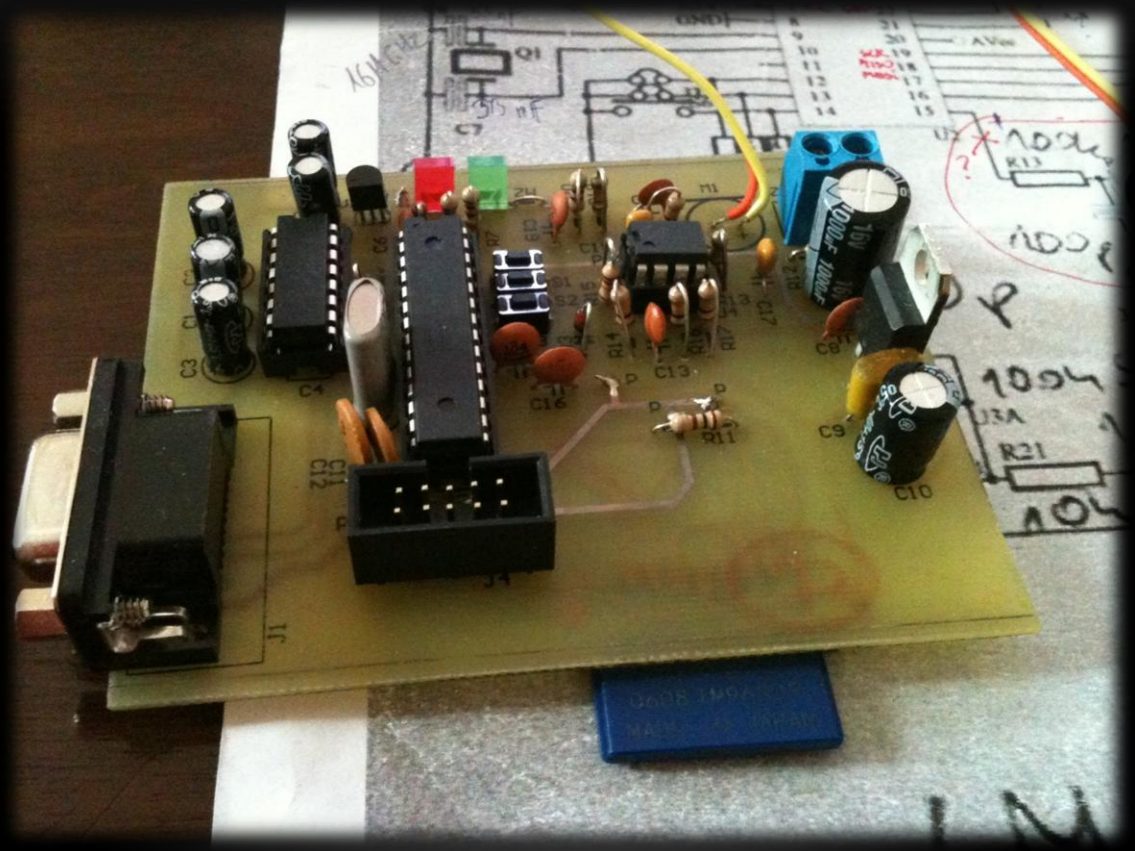
KOLEJNE ETAPY BUDOWANIA UKŁADU STERUJĄCEGO



KOLEJNE ETAPY BUDOWANIA UKŁADU STERUJĄCEGO



EFEKT KOŃCOWY BUDOWANIA UKŁADU STERUJĄCEGO



UŻYTY PROGRAMATOR:

Podstawowe właściwości:

- Kompatybilny z programatorem stk500v2;
- Umożliwia programowanie w systemie wszystkich mikrokontrolerów AVR obsługiwanych przez programator stk500v2;
- Współpraca z Atmel AVRStudio, WinAVR, Bascom-AVR, CodeVisoinAVR;
- Współpraca z układami zasilanymi napięciami 2.0 V – 5.5 V ;
- Standardowy interfejs KANDA 10 pin;
- Zabezpieczanie nadprądowe chroniące port USB;
- Diody LED sygnalizujące stan pracy programatora;
- Wyprowadzone linie Rx i Tx, co stanowi pełnofunkcyjny port szeregowy COM TTL;
- Wyprowadzony sygnał generatora 6 MHz, służący do odblokowywania mikrokontrolerów;
- Aktualizacja z poziomu AVR Studio dzięki rezydentnemu bootloaderowi

FRAGMENTY KODU ŹRÓDŁOWEGO:

```
//-------------------------------------
// Konfiguracja Timera
//-------------------------------------
ISR(TIMER1_COMPA_vect)
{
    licznik++;
    if(licznik >= 1999) licznik = 0;

    if(licznik  < servo_1) PORTD |= 1 << PD2;
    else PORTD &= ~(1 << PD2);

    if(licznik < servo_2) PORTD |= 1 << PD3;
    else PORTD &= ~(1 << PD3);

    if(licznik < servo_3) PORTD |= 1 << PD4;
    else PORTD &= ~(1 << PD4);

    if(licznik < servo_4) PORTD |= 1 << PD5;
    else PORTD &= ~(1 << PD5);

    LED_ON;
}
```

FRAGMENTY KODU ŹRÓDŁOWEGO:

```
//---------------------------------------
// Konfiguracja wyjść i UARTA
//---------------------------------------
void f_setting()
{

    TCCR1B = (1<<CS10) | (1<<WGM12); //preskaler 256
    TIMSK |= (1<<OCIE1A)  ;  //enable przerwań od przepełnienia
    OCR1A = 80;//31; //2 ?

    DDRD = 0xFF;


    sei();
    //LED_RED;
}
```

FRAGMENTY KODU ŹRÓDŁOWEGO:

```
//--------------------------------------
// Funkcja przkształcająca sygnał z UARTA na wartości przyjmowane przez servo mechanizm
//--------------------------------------
void f_transpute( int liczba )
{
    if(liczba <= 100)
        servo_1 = liczba + 28;
    else if(liczba >= 100 && liczba < 200)
        servo_2 = liczba - 100 + 28;
    else if(liczba >= 200 && liczba < 300)
        servo_3 = liczba - 200 + 28;
    else if(liczba >= 300 && liczba < 400)
        servo_4 = liczba - 300 + 28;
    else
        {
            LED_RED;
        }
}
```

FRAGMENTY KODU ŹRÓDŁOWEGO:

```
//-------------------------------------
// Main program
//-------------------------------------
int main(void)
{
    //-----------------------------
    //Ustawianie Portow i Timera
    //-----------------------------
    f_setting();
    USART_Init(BAUD(1200));
    char Message[18];
    int liczba = 0;
    //-------------------------

    do
    {

            USART_GetString(Message);

            liczba = atoi(Message);

            f_transpute(liczba);

    }while(1);
```

WYKONANE RAMIĘ

- **Podstawowe właściwości:**

Ramię jest obsługiwane przez specjalną aplikację napisaną przeze mnie, wykorzystującą port COM.

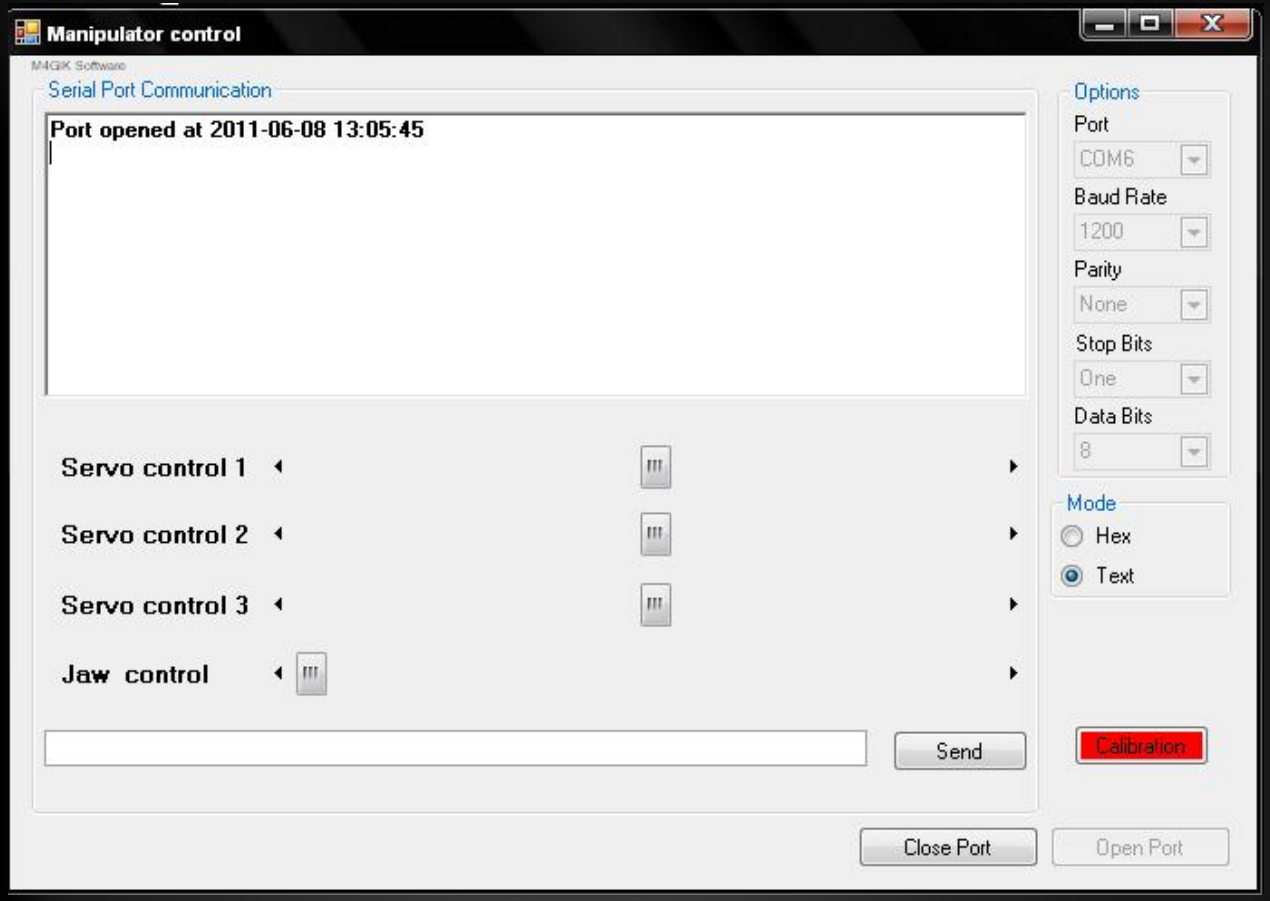
Program znajdujący się na płytce sterującej ramieniem interpretuje sygnały przychodzące z UARTA i w zależności od odpowiedniej wartości ustawia servo-mechanizmy pod odpowiednim kątem .

DOKUMENTACJA INTERPRETACJI SYGNAŁÓW PRZYCHODZĄCYCH DO RAMIENIA

Sygnał w postaci stringa o zadanej wartości ustawia servo pod odpowiednim kątem. Każdy string musi być opatrzony na końcu wartości w znak ‚k' w celu rozpoznania konca wartosci sygnału.

- Dla serva 1 wartosci : 0 – 100 , gdzie 50 – oznacza stan neutralny serva (90˚)
- Dla serva 2 wartosci : 100 – 200 , gdzie 150 – oznacza stan neutralny serva (90˚)
- Dla serva 3 wartosci : 200 – 300 , gdzie 250 – oznacza stan neutralny serva (90˚)
- Dla serva 4 wartosci : 300 – 400 , gdzie 350 – oznacza stan neutralny serva (90˚)

APLIKACJA STERUJĄCA MANIPULATOREM



RAMIĘ W AKCJI