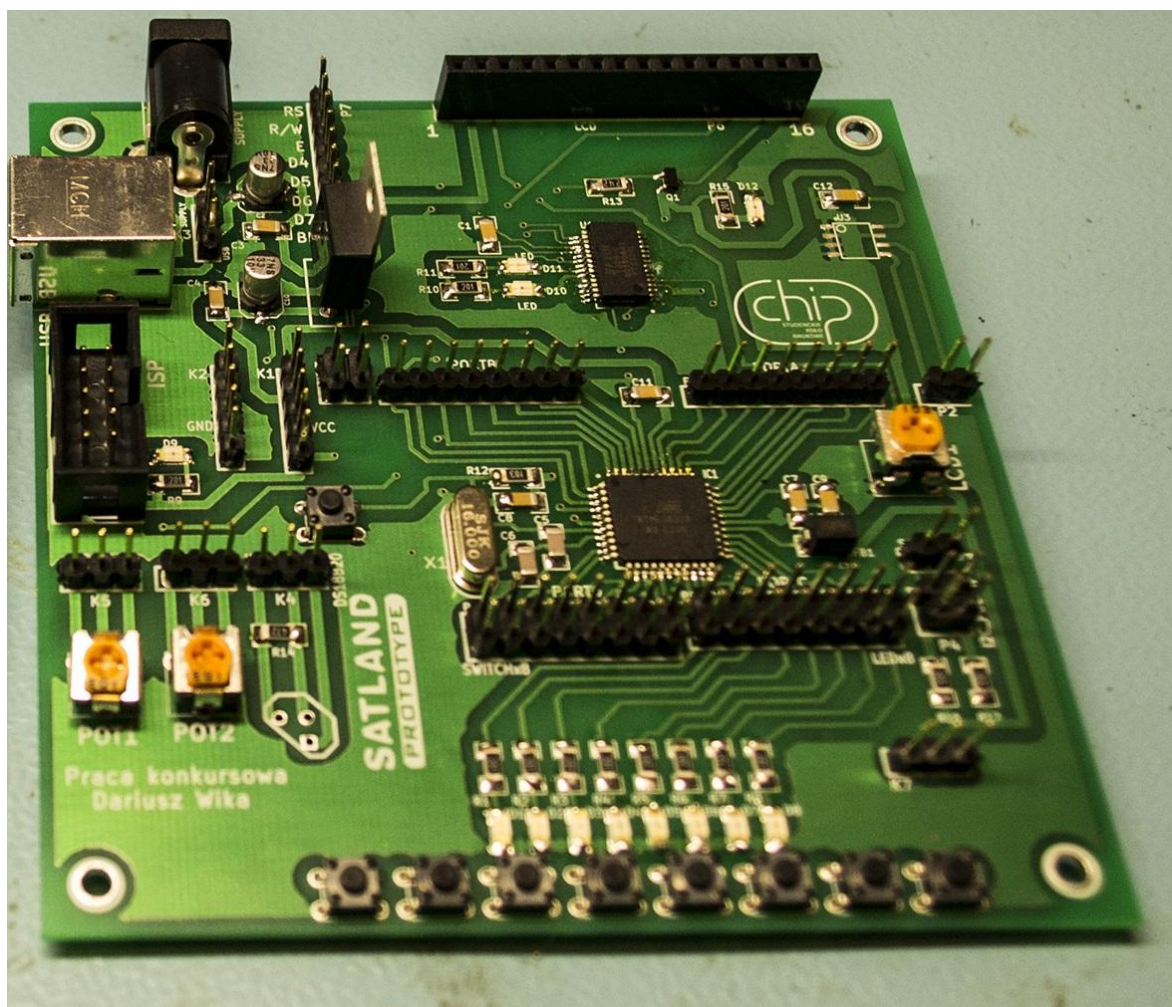


## Płyta ewaluacyjna z mikrokontrolerem Atmega32



Autor

Dariusz Wika

## Płyta ewaluacyjna z mikrokontrolerem Atmega32

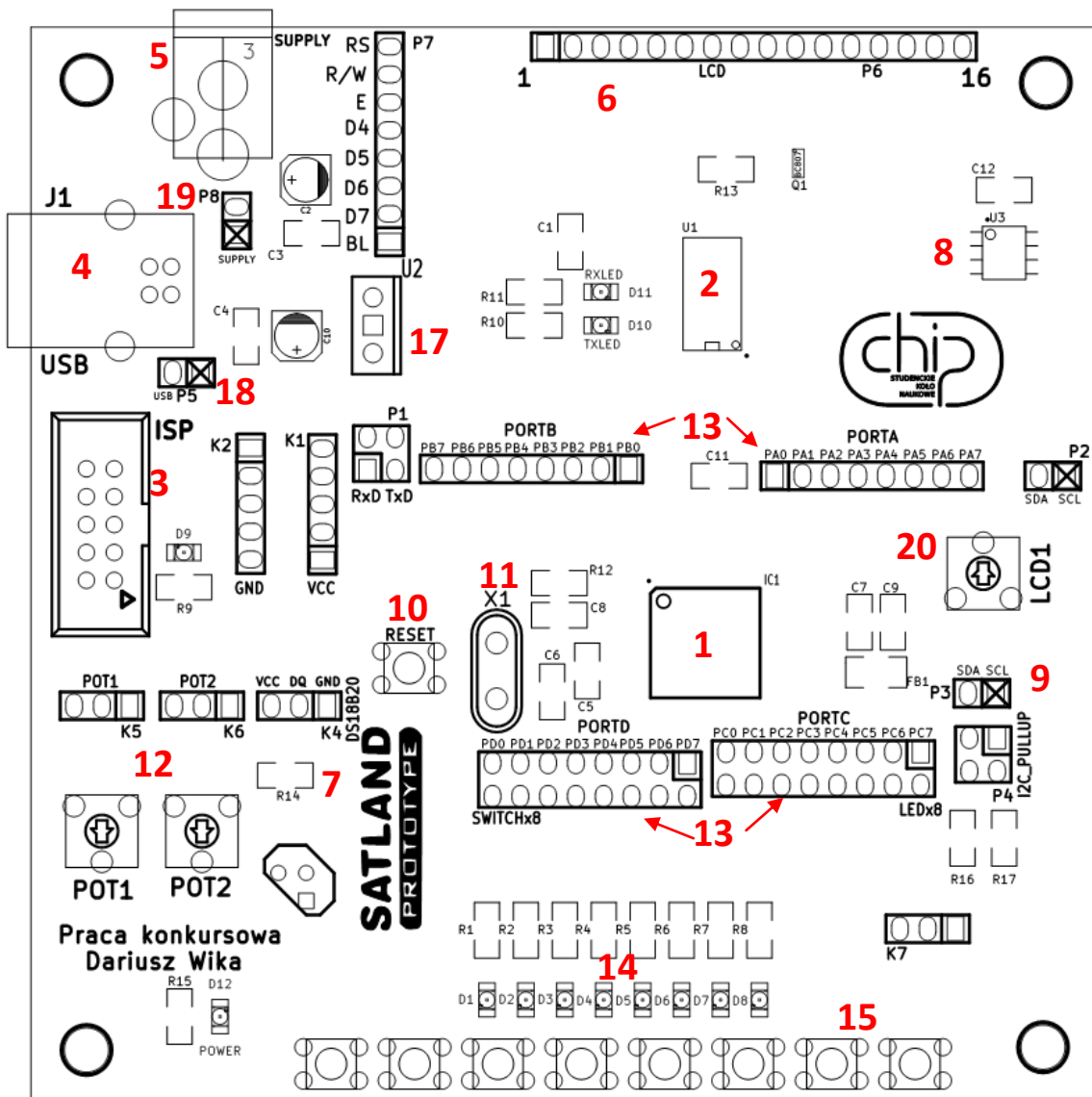
Płyta ewaluacyjna jest idealnym wyborem dla osób które chcą poznać architekturę 8-bitowych mikrokontrolerów z rodziny ATmega na przykładzie ATmega32 w który wyposażono prezentowany układ. Peryferia znajdujące się w zestawie tworzą dobrą bazę do nauki programowania na przykład w języku C oraz obsługi podstawowych interfejsów komunikacyjnych takich jak I2C oraz 1-wire.

Dzięki bootloaderowi wgranemu do pamięci mikrokontrolera do pracy nie są potrzebne żadne dodatkowe programatory oraz zewnętrzne urządzenia wystarczy tylko komputer z zainstalowanymi sterownikami oraz kabel USB.

Zestaw został wyposażony w:

- mikrokontroler Atmega32
- cyfrowy czujnik temperatury
- zewnętrzną pamięć EEPROM
- układ FT232RL
- potencjometry
- wyświetlacz LCD 2x16 zgodny z HD44780
- 8 diod LED
- 8 przycisków
- rezonator kwarcowy 16MHz
- stabilizator 7805

## Płyta ewaluacyjna z mikrokontrolerem Atmega32

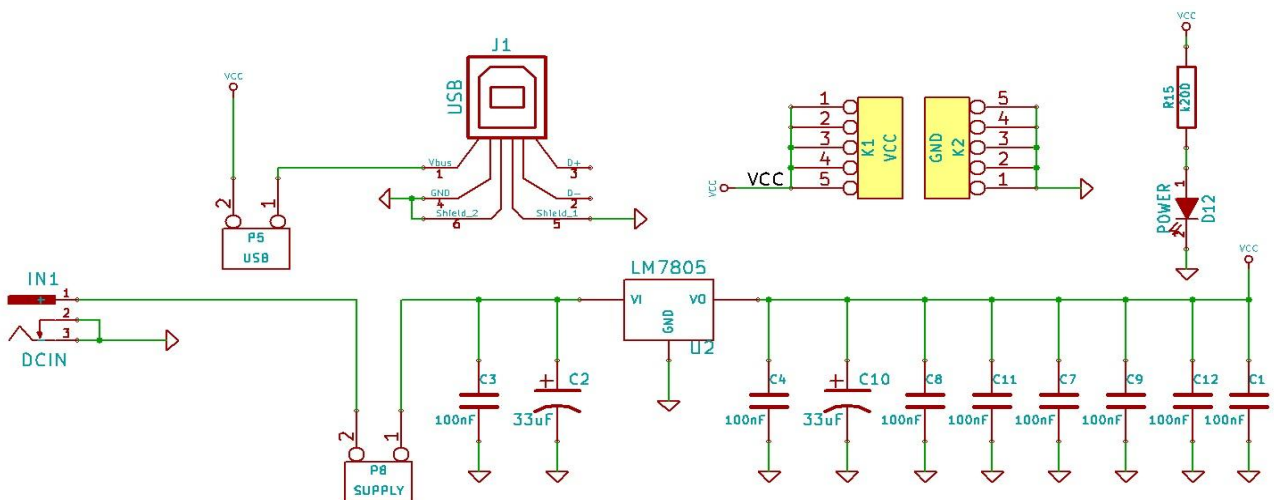


Rysunek 1 Płyta uruchomieniowa

Na płycie uruchomieniowej znajdują się:

- |   |   |
|---|---|
| <ol style="list-style-type: none"> <li>1. Mikrokontroler Atmega32</li> <li>2. Układ FT232RL</li> <li>3. Gniazdo zewnętrznego programatora w standardzie KANDA</li> <li>4. Gniazdo USB</li> <li>5. Gniazdo zewnętrznego zasilacza</li> <li>6. Wyprowadzenia wyświetlacza zgodnego z HD44780</li> <li>7. Cyfrowy czujnik temperatury DS18B20 działający w oparciu o interfejs 1Wire wraz z wyprowadzeniami</li> <li>8. Zewnętrzna pamięć EEPROM 24C32</li> <li>9. Wyprowadzeni interfejsu I2C</li> <li>10. Przycisk Reset</li> <li>11. Zewnętrzny rezonator kwarcowy 16MHz</li> </ol> | <ol style="list-style-type: none"> <li>12. Dwa potencjometry wraz z wyprowadzeniami</li> <li>13. Wyprowadzenia portów mikrokontrolera</li> <li>14. 8 diod LED</li> <li>15. 8 Przycisków</li> <li>16. Wyprowadzenia wejść wyświetlacza LCD</li> <li>17. Stabilizator napięci</li> <li>18. Zworka umożliwiająca zasilanie układu z gniazda USB</li> <li>19. Zworka umożliwiające zasilanie układu z zewnętrznego zasilacza</li> <li>20. Potencjometr umożliwiający zmianę kontrastu wyświetlacza</li> </ol> |
|---|---|

## Zasilanie



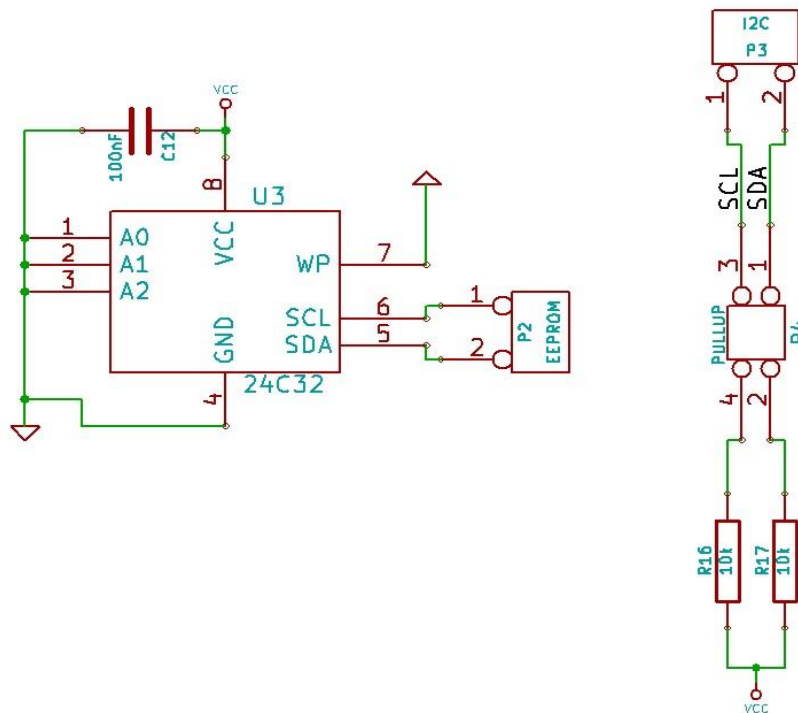
Rysunek 2 Zasilanie układu

Płyta ewaluacyjna może być zasilana z 3 źródeł:

1. Z zewnętrznego programatora
2. Z zasilacza o napięciu do 12V
3. Z USB

Wybór źródła zasilania odbywa się za pomocą założenie odpowiedniej zworki. Założenie zworki na złącze P8 oznacza wybór zewnętrznego zasilacza jako źródła zasilania układu, założenie zworki na złącze P5 oznacza zasilanie układu z USB. Wskazane jest aby układ nie pracował przy jednoczesnym zasilaniu z dwóch źródeł dlatego użytkownik musi uważać aby przy podłączeniu do układu zasilacza i kabla USB nie były założone obie zworki jednocześnie. Obecność napięcia sygnalizuje dioda POWER. NA złącza K1 i K2 wyprowadzone zastały napięcie zasilania oraz masa które mogą posłużyć do zasilania zewnętrznych układów oraz peryferii znajdujących się w zestawie uruchomieniowym.

## I2C i zewnętrzna pamięć EEPROM

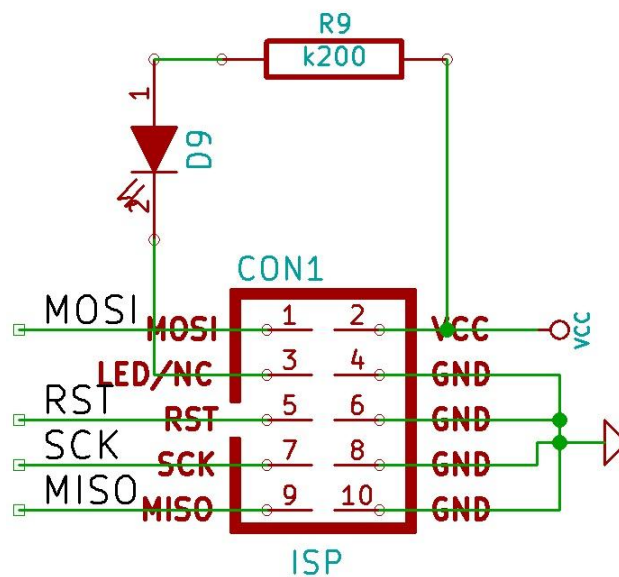


Rysunek 3 Pamięć EEPROM i I2C

W zestawie uruchomieniowym zaimplementowano interfejs I2C którego linie SCL i SDA zostały wyprowadzone na złączu P3. Założenie zwerek na złączu P4 włącza podciąganie linii do napięcia zasilania przez rezystory R16 i R17 o rezystancji 10kΩ.

W zestawie umieszczona została zewnętrzna pamięć EEPROM 24C32 której linie SCL i SDA zostały wyprowadzone na złączu P2. Wyprowadzenia adresowe pamięci zostały podciągnięte do masy układu co w konsekwencji dają adres zapisu równy A0 i adres odczytu równy A1.

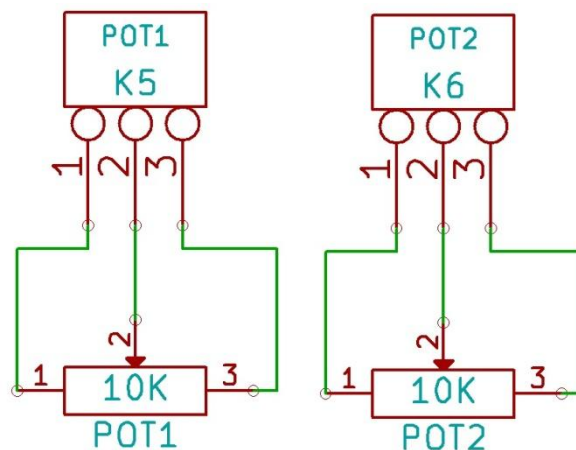
## ISP KANDA



Rysunek 4 ISP KANDA

W zestawie uruchomieniowym wyprowadzone zostało złącze interfejsu ISP w standardzie KANDA które może służyć do programowania mikrokontrolera zewnętrznym programatorem oraz zasilania układu.

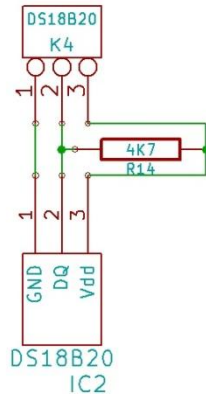
## Potencjometry



Rysunek 5 Potencjometry

W układzie umieszczone zostały dwa potencjometry których wyprowadzenia znajdują się na złączach K5 i K6. Potencjometry mogą być wykorzystane do nauki obsługi przetwornika analogowo cyfrowego lub komparatora wbudowanego w mikrokontroler.

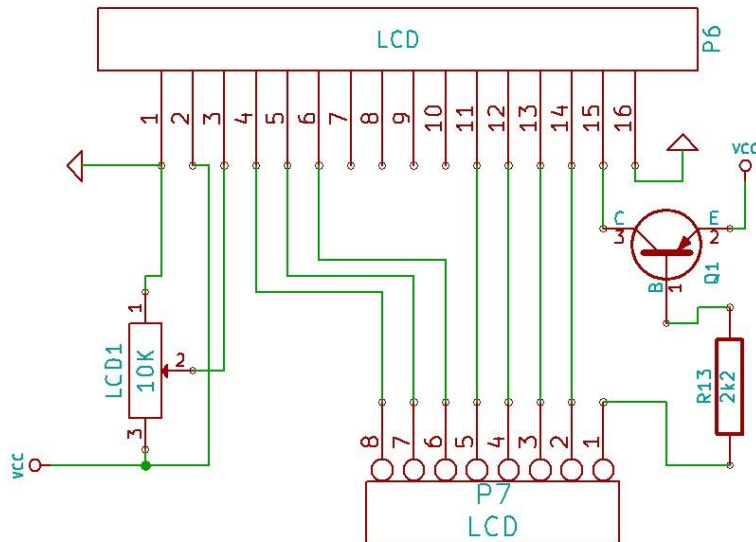
## Cyfrowy termometr, interfejs 1-wire



Rysunek 6 Cyfrowy Termometr

W zestawie znajduje się również cyfrowy czujnik temperatury. Komunikacja z czujnikiem odbywa się za pomocą interfejsu 1-WIRE którego wyprowadzeni znajdują się na złączu K4

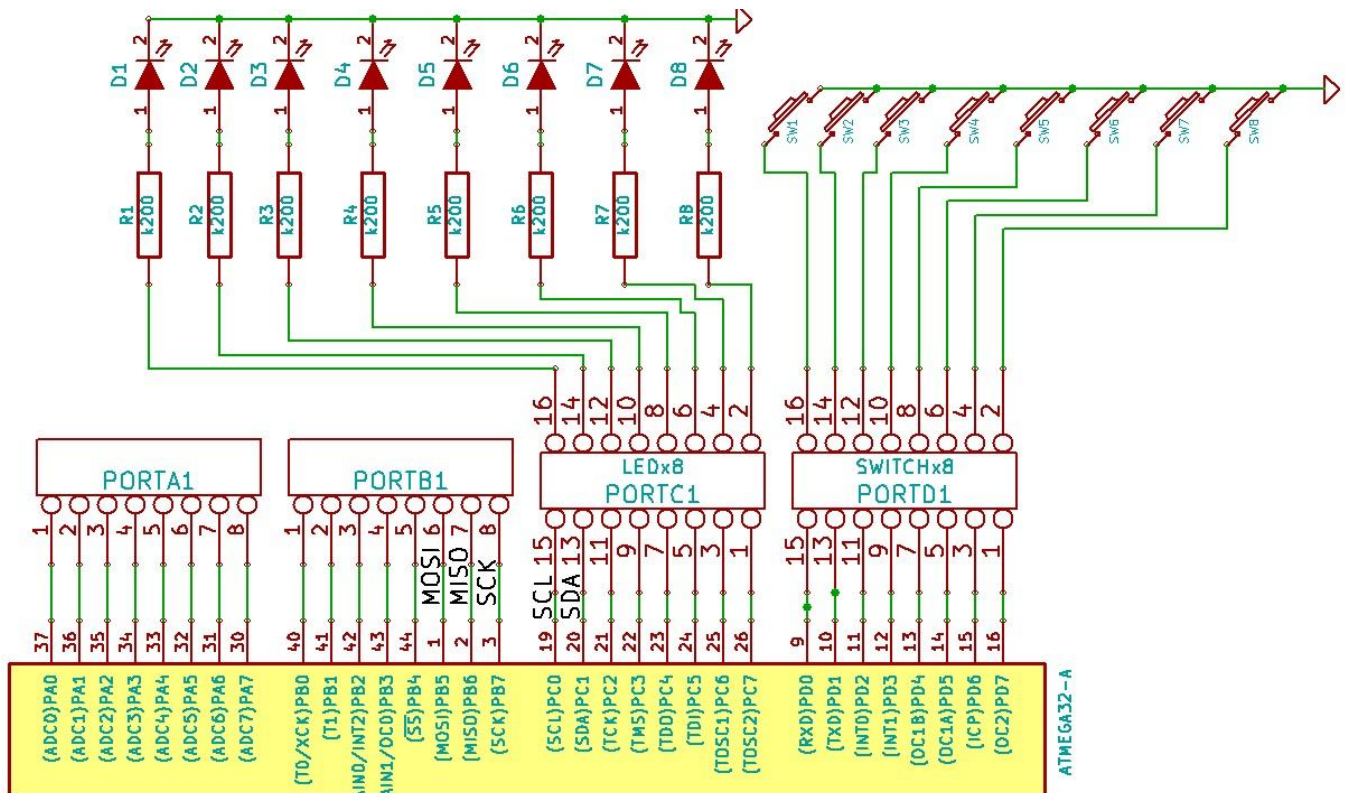
## Wyświetlacz LCD



Rysunek 7 Wyświetlacz LCD

Na płycie znajdują się złącze oznaczone jako P6 do którego można podłączyć wyświetlacz LCD zgodny z HD44780. Linie wyświetlacza zostały wyprowadzone na złączy P7. Potencjometr LCD1 służy do regulacji kontrastu wyświetlacza. Podświetlenie wyświetlacza jest aktywne po podaniu stanu niskiego na PIN 1 złącza P7 który na płycie oznaczony jest jako BL.

## Diody, przyciski i wyprowadzenia portów mikrokontrolera



Rysunek 8 Diody przyciski i GPIO

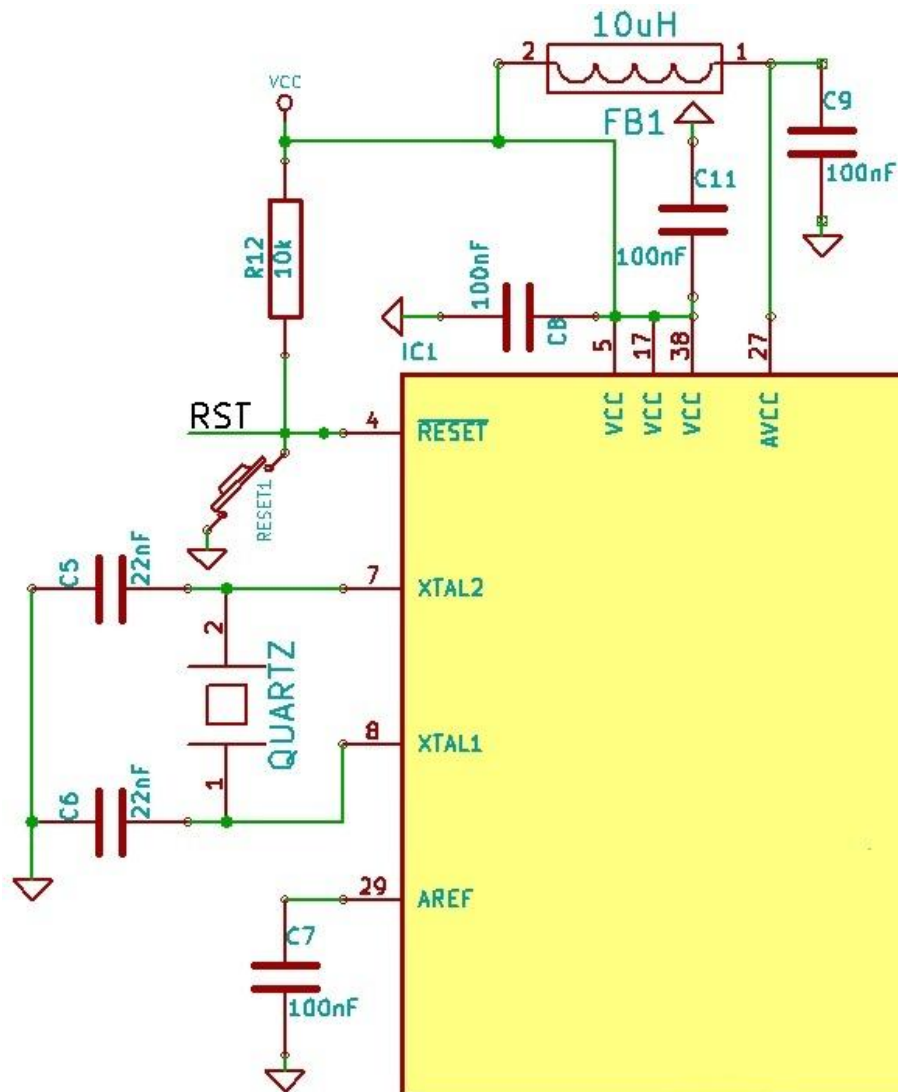
Wszystkie porty mikrokontrolera zostały w całości wyprowadzone na złącza typu GOLDPIN. W zestawie dostępne jest 8 różnokolorowych diod LED oraz 8 przycisków typu TACT SWITCH.

Aby skorzystać z diod i przycisków należy na wyprowadzenie wybranego pinu mikrokontrolera oraz wyprowadzenie wybranego elementu nałożyć zworkę. Na przykład jeśli zamierzamy w jednym projekcie skorzystać z 8 diod LED należy założyć 8 zwrotek na wyprowadzenia Portu D oraz wyprowadzenia diod. Diody i przyciski za pomocą kabli połączeniowych mogą być podłączone do wyprowadzeń innego portu.

Aby skorzystać z diody LED należy wystawić na odpowiedni pin wybranego portu stan wysoki. Przyciski zwierają do masy dlatego aby wykryć wciśnięcie przycisku należy wykrywać stan niski na wybranym pinie przy włączonym wewnętrznym podciągnięciu pinu do napięcia zasilania.



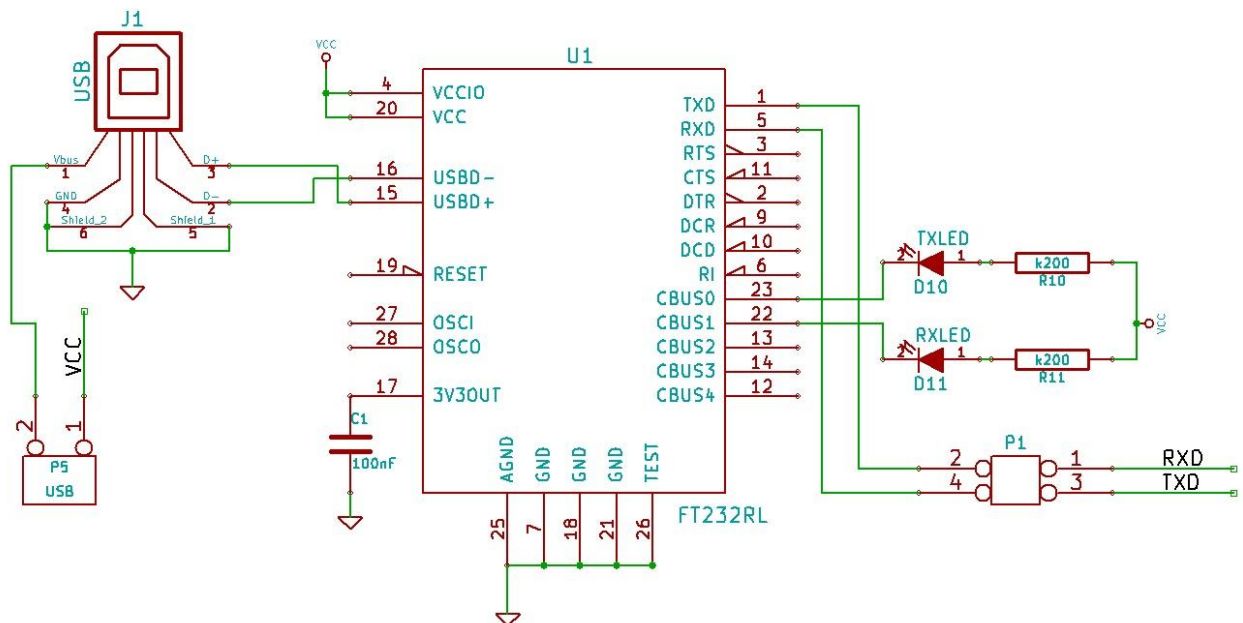
## Układ resetu i zewnętrzny rezonator kwarcowy



Rysunek 9 Układ resetu oraz rezonator kwarcowy

Do układu został dołączony zewnętrzny rezonator kwarcowy o częstotliwości 16MHz. Do wykonania resetu służy przycisk oznaczony RESET.

## Komunikacja z układem



Rysunek 10 Układ FT232RL

Programowanie układu odbywa się za pomocą zapisanego w pamięci mikrokontrolera bootloadera. Układ widziany jest przez komputer jako programator stk500v2. Aby zaprogramować układ wymagane jest połączenie z komputerem za pomocą kabla USB oraz założenie zworek na złącze P1 na którym wyprowadzone są linie RXD i TXD zarówno mikrokontrolera jak i układu FT232RL który spełnia rolę konwertera USB – UART. Obecność sygnałów na liniach RXD i TXD sygnalizuje świecenie diod RXLED i TXLED. Stan aktywny bootloadera sygnalizuje stan wysoki na wyjściu PD7 mikrokontrolera, który można sprawdzić za pomocą dowolnie wybranej diody. Komunikacja odbywa się za pomocą wirtualnego portu COM którego sterowniki znajdują się na płycie oraz na stronie producenta układu FTDI pod adresem <http://www.ftdichip.com/Drivers/D2XX.htm>.

## Konfiguracja oraz programowanie układu przez USB

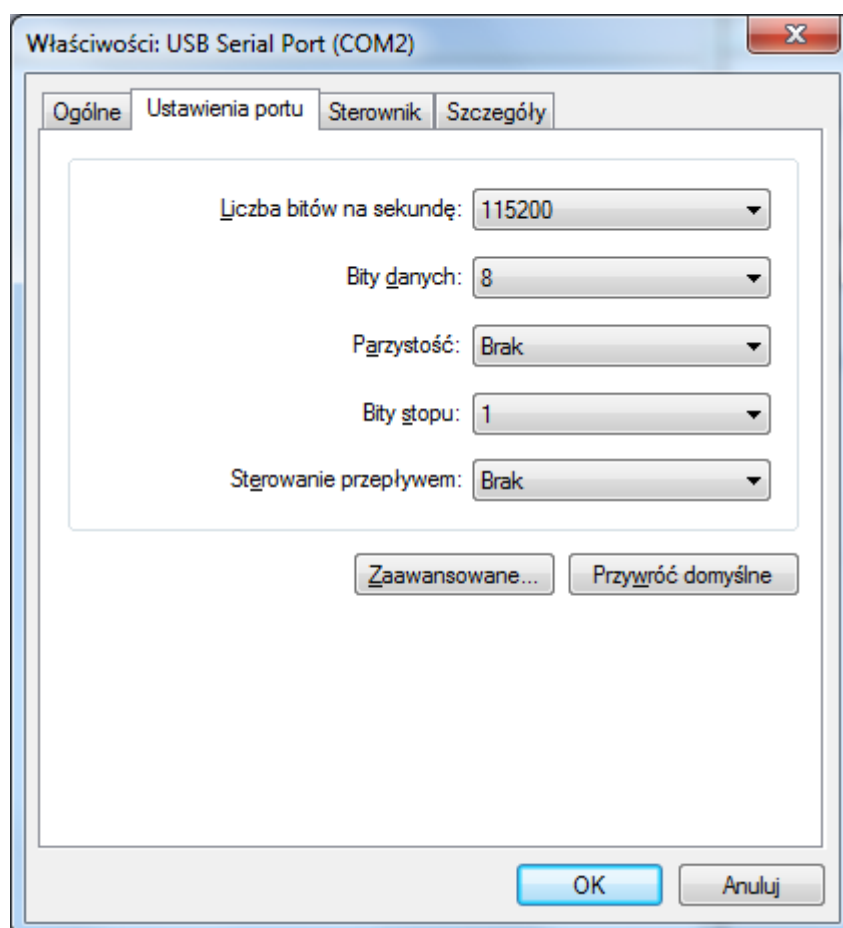
Po zainstalowaniu sterowników należy odpowiednio skonfigurować parametry transmisji

Bitrate – 115200

Bity danych – 8

Bit stopu -1

Brak kontroli parzystości



Rysunek 11 Konfiguracja portu COM

Po resecie lub włączeniu zasilanie bootloader jest aktywny przez 8 sekund co jest sygnalizowane stanem wysokim na pinie PD7, w tym czasie należy zgrać wsad do pamięci mikrokontrolera i ponownie wcisnąć przycisk reset. Po 8 sekundach zacznie działać wgrany program. Aby ponownie wgrać wsad do mikrokontrolera należy kolejny raz wcisnąć przycisk reset.

## Środowisko Eclipse

Tworzenie projektu, pisanie programu oraz uruchamianie go w docelowym układzie przedstawione na przykładzie. Do tworzenia projektu oraz programowania układu zostanie wykorzystane środowisko Eclipse, które można znaleźć na dołączonej do zestawu płycie. Zastosowanym językiem programowania będzie język C.

Do poprawnego działania środowiska Eclipse, wymagana jest zainstalowana na komputerze Java zgodna z wersją Eclipse, AVR GCC Toolchain, który pozwala tworzyć projekty na mikrokontrolery AVR oraz WinAVR.

Gotowe do pracy środowisko znajdują się na płycie dołączonej do zestawu, jednak nie jest możliwe zamieszczenie tutaj oprogramowania zgodnego z wszystkimi ważnymi systemami operacyjnymi dlatego też zamieszczam linki do potrzebnego oprogramowania.

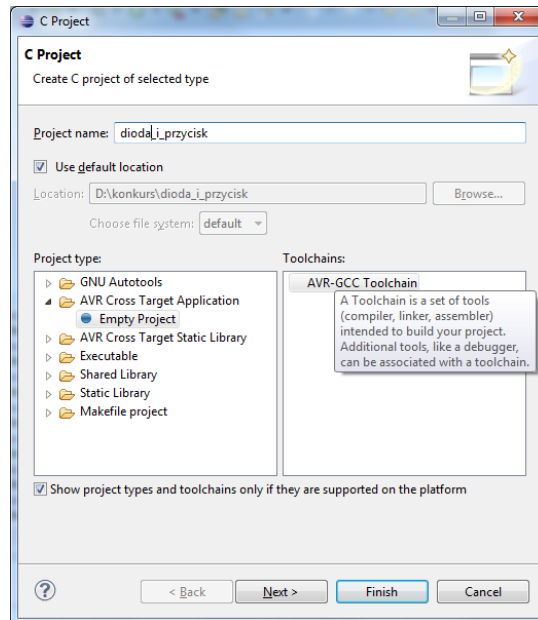
Potrzebne linki:

1. <http://www.eclipse.org/downloads/packages/eclipse-ide-cc-developers/keplersr1>
2. <http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html>
3. [http://avr-eclipse.sourceforge.net/wiki/index.php/Plugin\\_Download](http://avr-eclipse.sourceforge.net/wiki/index.php/Plugin_Download)
4. <http://sourceforge.net/projects/winavr/files/WinAVR/>

Proces instalacji środowiska Eclipse należy zacząć od instalacji Javy, następnie należy pobrać oraz rozpakować środowisko Eclipse, kolejnym etapem instalacji jest instalacja WinAVR. Następnie uruchamiamy Eclipse i w wyświetlonym oknie dialogowym podajemy ścieżkę dostępu do folderu **Workspace**, folder ten służy do przechowywania projektów nad którymi obecnie pracujemy i może być dowolnym stworzonym przez nas folderem w dowolnej lokalizacji. Po uruchomieniu środowiska należy, wcisnąć przycisk **Help** i z wyświetlonego okna wybrać opcje **Install New Software**. Następnie należy wcisnąć przycisk **Add** a potem przycisk **Local** i wskazujemy ścieżkę pod którą znajdują się **AVR GCC Toolchain** i potwierdzamy przyciskiem **OK**. Kolejnym etapem instalacji jest zaznaczenie opcji **AVR Eclipse Plugin** która przy poprawnym wykonaniu powyższych kroków pojawiła się w centralnej części okna **Install** i wciskamy przycisk **Next** i postępujemy zgodnie z instrukcjami wyświetlającymi się na ekranie.

## Utworzenie i konfiguracja projektu

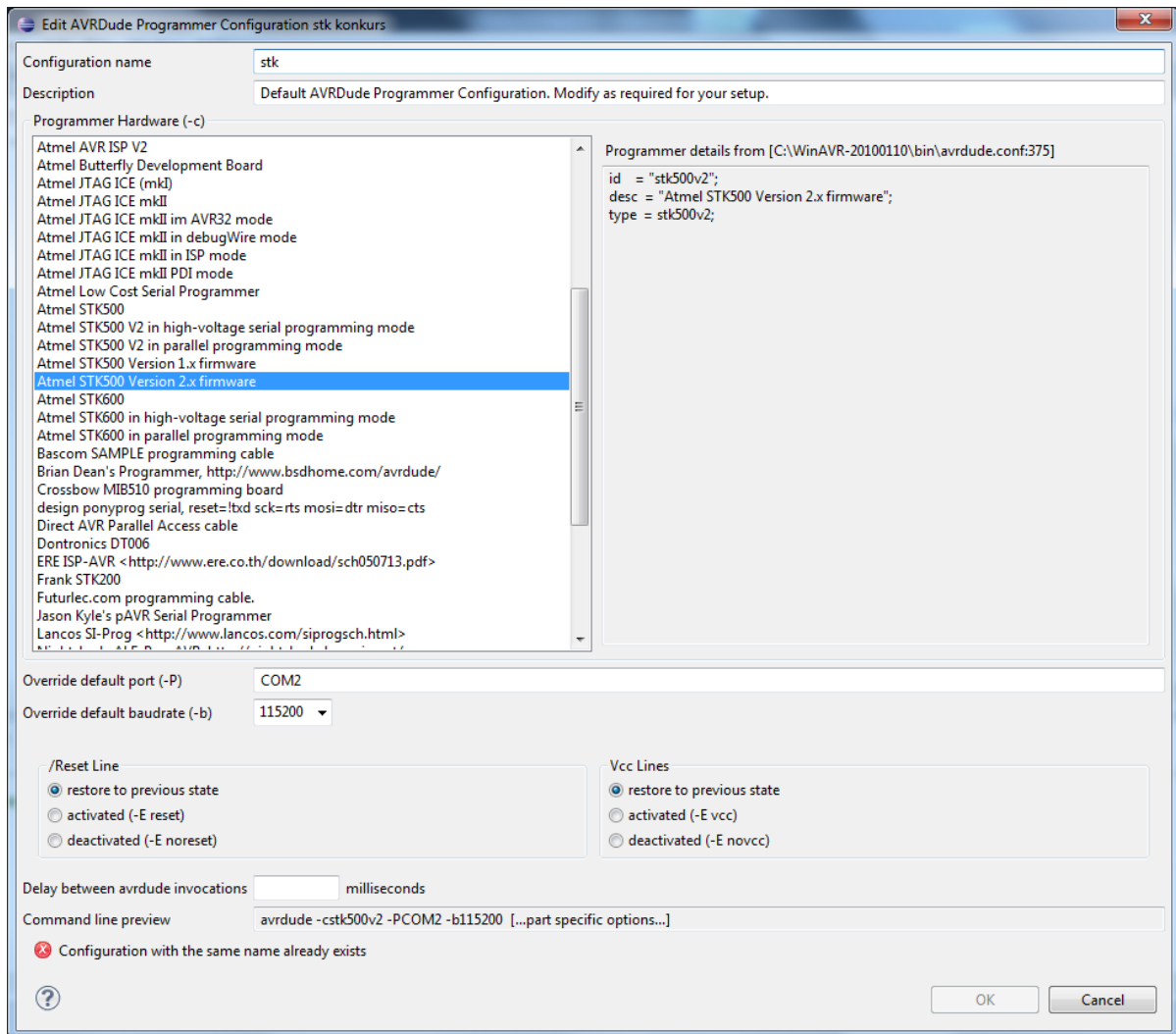
Po odpowiednim przygotowaniu środowiska należy przygotować nowy projekt. W tym celu należy w z menu Eclipse wybrać opcję **File** → **New** → **C Projekt**. Kolejnym krokiem jest nadanie nazwy projektu oraz wybór jego typu tak jak na załączonym rysunku.



Rysunek 12 Tworzenie projektu

Następnie należy nacisnąć przycisk **NEXT** i pozostawić zaznaczoną opcję **Release**, ponieważ opcja **Debug** nie jest wykorzystywana w tworzonym projekcie, należy ponownie nacisnąć przycisk **NEXT**. W celu ukończenia tworzenia projektu z rozwijanej listy wybrano typ mikrokontrolera ATmega32 oraz podano częstotliwość jaką taktowany jest układ czyli **16000000MHz** i wcisnięto przycisk **Finish**. Po poprawnym utworzeniu projektu w oknie po lewej stronie ekranu powinien pojawić się projekt. Kolejnym krokiem jest dodanie pliku który będzie przechowywał nasz kod programu. W tym celu należy kliknąć prawym przyciskiem myszy na folder reprezentujący projekt i wybierać kolejno **New** → **Source File**, w polu **Source File** należy podać wpisujemy nazwę pliku „**main.c**”. Do rozpoczęcia programowania pozostał już tylko jeden krok czyli wybór programatora. Prawym przyciskiem myszy wybrano folder reprezentujący projekt, kliknięto opcję **Properties** po lewej stronie okna **Properties** lewym przyciskiem myszy rozwinięto opcję **AVR** i wybrano opcje **AVRDude**. Dokonując wyboru programatora należy wcisnąć przycisk **New** i ustawić wszystkie opcje jak na poniższym rysunku.

## Płyta ewaluacyjna z mikrokontrolerem Atmega32



Rysunek 13 Wybór programatora

Powyższe zmiany należy zatwierdzić klikając przycisk **OK**. Teraz pozostają już tylko upewnić się że w polu **Programmer configuration** została wybrana opcja **stk**. Kolejnym etapem pracy jest pisanie pierwszego programu który pozwoli skorzystać z diod i przycisków znajdujących się na płycie uruchomieniowej.



## Pierwszy Program

W celu obsługi przycisków i diod należy nałożyć wszystkie zworki na złącza przy **PORTC D** i **PORTC C**, należy założyć również zworkę odpowiadającą za wybór zasilania oraz zworki na złącze **P1** które pozwolą zaprogramować układ.

```
#include <avr/io.h>
#include <util/delay.h>
int main(void) {
    DDRC = 0xFF;
    PORTC = 0x00;
    DDRD = 0x00;
    PORTD = 0xFF;
    while (1) {
        if (!(PIND & (1 << PD0))) {
            PORTC ^= (1 << PC0);
        }
        if (!(PIND & (1 << PD1))) {
            PORTC ^= (1 << PC1);
        }
        if (!(PIND & (1 << PD2))) {
            PORTC ^= (1 << PC2);
        }
        if (!(PIND & (1 << PD3))) {
            PORTC ^= (1 << PC3);
        }
        if (!(PIND & (1 << PD4))) {
            PORTC ^= (1 << PC4);
        }
        if (!(PIND & (1 << PD5))) {
            PORTC ^= (1 << PC5);
        }
        if (!(PIND & (1 << PD6))) {
            PORTC ^= (1 << PC6);
        }
        if (!(PIND & (1 << PD7))) {
            PORTC ^= (1 << PC7);
        }
    }
}
```

Załączony powyżej kod znajduje się na płycie dołączonej do zestawu wraz z komentarzami wyjaśniającymi poszczególne linie kodu.

## Płyta ewaluacyjna z mikrokontrolerem Atmega32

Założono że poprawnie dokonano wyboru programatora, mikrokontroler oraz częstotliwość taktowania podczas tworzenie projektu a kod programu nie ma błędów. Kolejnym etapem pracy jest kompilacja programu oraz umieszczeniu go w pamięci flash mikrokontrolera. Aby dokonać kompilacji należy wcisnąć ikonkę w kształcie młotka  na pasku menu w Eclipse. Po poprawnej kompilacji w konsoli u dołu ekranu ukaże się napis **Build Finished**. Aby umieścić program w docelowym układzie skorzystamy z opcji **Upload Projekt to Target Device** dostępnej jako  po poprawnym wgraniu pliku do układu w konsoli pojawi się komunikat **avrdude done**.

Należy jednak pamiętać o uaktywnieniu bootloadera na czas wgrywania danych do układu zgodnie z procedurą opisaną na stronie **11**.

Zaprezentowany programu pozwala na zapalenie diody odpowiadającej naciśniętemu przyciskowi. Można zauważyć że nie każde wciśnięcie przycisku zapala lub gasi diodę albo dioda tylko delikatnie miga, zjawisko to jest efektem drgania styków po wciśnięciu przycisku. Aby w najprostszy sposób wyeliminować to niechciane zjawisko wystarczy dodać odpowiedni warunek do kodu oraz skorzystać z funkcji **\_delay\_ms(xxx)** gdzie xxx to ilość milisekund na jaką program ma się zatrzymać w danym miejscu.

```
if(!(PIND & (1 << PD0))){
    _delay_ms(100);
    if (!(PIND & (1 << PD0))) {
        PORTC ^= (1 << PC0);
        _delay_ms(200);
    }
}
```

Powyżej znajdują się przykład kodu który eliminuje drgania styków. Eliminacja polega na sprawdzeniu stanu przycisku dwukrotnie, przy czym drugie sprawdzenie odbywa się 100ms po pierwszym co daje odpowiednia ilość czasu aby stan przycisku ustabilizował się. Kolejne opóźnienie trwające 200ms daje czas na zwolnienie przycisku co zabezpiecza przed miganiem diody przy zbyt długim wciśnięciu przycisku. Jednak w tym przypadku rozwiązanie najprostsze nie znaczy najlepsze, ponieważ w czasie trwania funkcji **\_delay\_ms(xxx)** nasz mikrokontroler nic nie robi co powodują marnotrawienie jego mocy obliczeniowej dlatego należy unikać korzystania z tej funkcji w programach w których czas dopowiedzi na wystąpienie zdarzenia gra dużą rolę.



## Prosty pomiar Napięcia

W tym projekcie wykorzystany zostanie wbudowany w mikrokontroler przetwornik ADC, znajdujący się w zestawie potencjometr oraz diody. Podłączenie zworek jest identyczne jak w poprzednim projekcie, zasady tworzenia, wyboru programatora, kompilacji oraz uruchamiania programu w układzie docelowym pozostają nie zmienione. Podłączenie potencjometru sprawdza się do podłączenia jednego z jego wyprowadzeń do masy drugiego do napięcia zasilania, które znajdują się na wyprowadzeniach **K1** i **K2** oraz trzeciego wyprowadzenia do wejścia przetwornika. Wybrano potencjometr wyprowadzony na złączu **K5**, lewy pin wyprowadzania podłączono za pomocą kabla połączeniowego do napięcia zasilania **VCC** jego prawy pin podłączono do masy układu **GND** zaś środkowe wyprowadzenie do pinu portu A oznaczonego jako **PA1**.

Ważnym elementem programowania na mikrokontrolery jest korzystanie z ich dokumentacji. Dokumentację ważniejszych elementów znajduję się na płycie dołączonej do układu. W dokumentacji układu ATmega32 w rozdziale zatytułowanym **Analog to Digital Converter** znajdują się szczegółowe informacje dotyczące parametrów przetwornika oraz jego rejestrów konfiguracyjnych od których zależy między innymi szybkość pracy i dokładność obliczeń wykonanych z jego pomocą.

Podstawowe informacje zawarte w dokumentacji mówią że przetwornik w naszym układzie ma 10-bitową rozdzielczość co sprawia że wynik jego pomiaru zawiera się w zakresie od 0 do 1023. W obecnym przypadku 0 oznacza 0V na wejściu przetwornika a 1023 5V. Zauważono że do pracy przetwornika z pełną rozdzielczością, jego częstotliwość taktowania nie może być większa niż 200kHz. Układ taktowany jest rezonatorem o częstotliwości pracy 16MHz, aby zapewnić taktowanie mniejsze od 200kHz skorzystano z wbudowanego w układ preskalera, czyli dzielnika częstotliwości. Pomiar w przetworniku odbywa się w oparciu o porównanie wartości mierzonej z wartości odniesienia, którą została wybrana jako napięcie zasilania.

Poniżej znajduję się kod, który wraz z objaśnieniami znajduję się na płycie dołączonej do zestawu. Działanie programu sprawdza się do zmiany ilości zapalonych diod wraz ze zmianą napięcia wejściowego przetwornika na skutek obrotu pokrętki potencjometru. Wartości progowe w których zapalają się kolejne diody są dowolne.

Sposób ustawiania bitów w rejestrach konfiguracyjnych przedstawia się następująco, linia **ADMUX |= (1 << REFS0);** oznacza ustawienie bitu REFS0 w rejestrze ADMUX co decyduje o wyborze napięcia z pinu AVCC mikrokontrolera jako źródło napięcia odniesienia

```

#include <util/delay.h>
#include <avr/io.h>

uint16_t pomiar(uint8_t kanal) {
    ADMUX |= (ADMUX & 0xF8) | kanal;
    ADCSRA |= (1 << ADSC);
    while ( ADCSRA & (1 << ADSC));
    return ADCW;
}

int main(void) {

    DDRC = 0xFF;
    PORTC = 0x00;
    ADMUX |= (1 << REFS0);
    ADCSRA |= (1 << ADEN) | (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);

    while (1) {

        if (pomiar(1) < 100) {
            PORTC = 0x01;
        }
        if (pomiar(1) > 100 && pomiar(1) < 200) {
            PORTC = 0x03;
        }

        if (pomiar(1) > 200 && pomiar(1) < 300) {
            PORTC = 0x07;
        }
        if (pomiar(1) > 400 && pomiar(1) < 500) {
            PORTC = 0x0F;
        }
        if (pomiar(1) > 500 && pomiar(1) < 700) {
            PORTC = 0x1F;
        }

        if (pomiar(1) > 700 && pomiar(1) < 900) {
            PORTC = 0x3F;
        }
        if (pomiar(1) > 900 && pomiar(1) < 1000 ) {
            PORTC = 0x7F;
        }
        if (pomiar(1) > 1000) {
            PORTC = 0xFF;
        }

    }

}

```