Podstawy techniki mikroprocesorowej 2

Ćwiczenie 1

Łukasz Chwistek

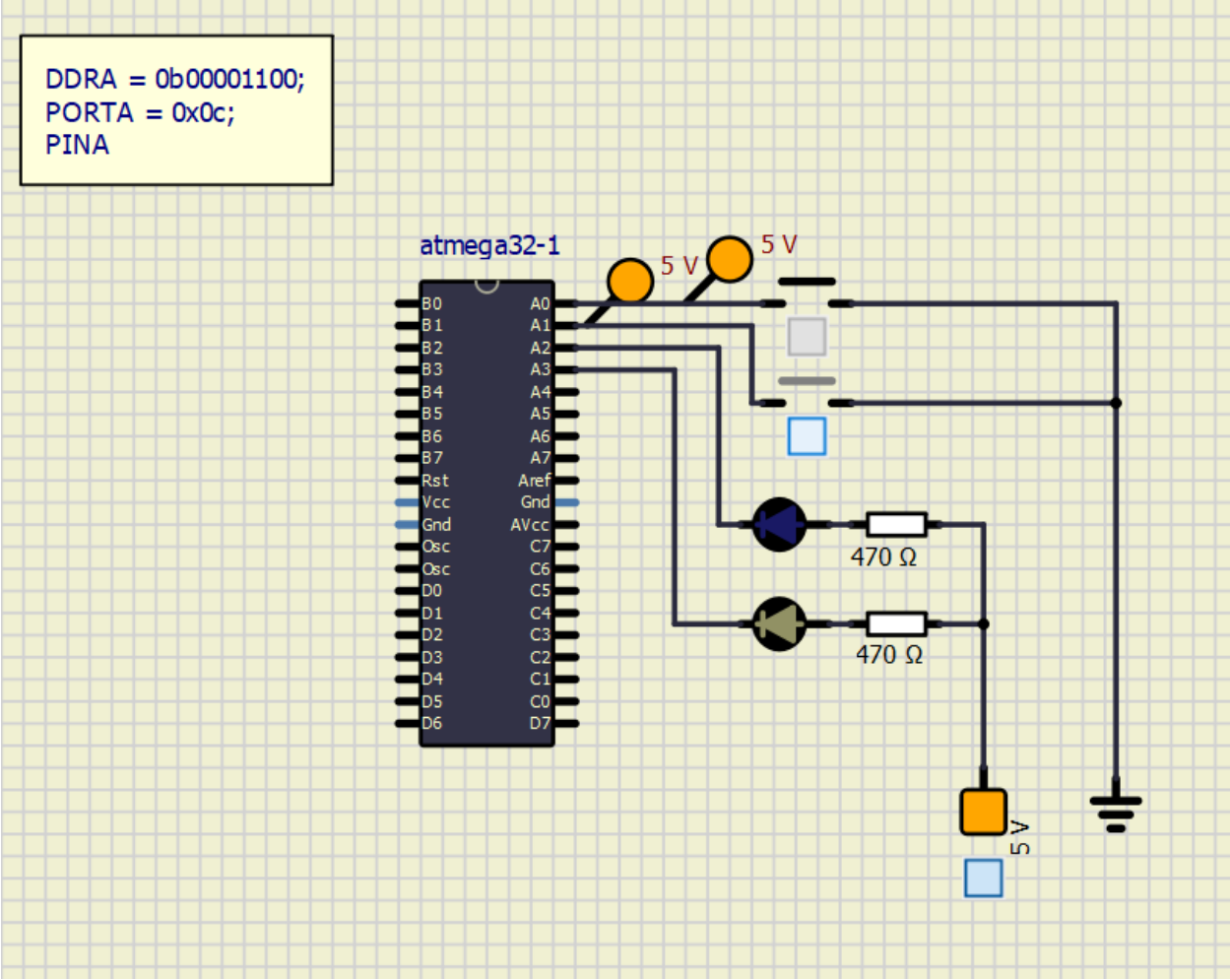
Nr. albumu: 243662

1. Wstęp

Ćwiczenie polegało na symulacji układu z mikrokontrolerem atmega32, dwóch przycisków oraz dwóch diod LED. Miały być one włączane lub wyłączane przy naciśnięciu przypisanego jej przycisku.

2. Schemat układu

Porty A0 i A1 domyślnie znajdują się w stanie wysokim, naciśnięcie przycisku powoduje zwarcie portu do masy i wystawienie 0 logicznego. Diody nie świeca się kiedy na portach A2 i A3 występuje stan wysoki. Przejście portów A2 i A3 na stan niski powoduje zwarcie do masy źródła zasilania, dzięki czemu diody LED zaczynają przewodzić i świecą.



3. Kod programu

Program został wykonany i skompilowany za pomocą pakietu WinAVR oraz VSCode z konfiguracją zawartą w pliku Makefile.

Porty są konfigurowane za pomocą odpowiedniego ustawienia rejestru DDRx, gdzie ustalany jest kierunek pracy portu. Poprzez zmiany odpowiednich bitów w rejestrze PORTx ustawiane są wartości na

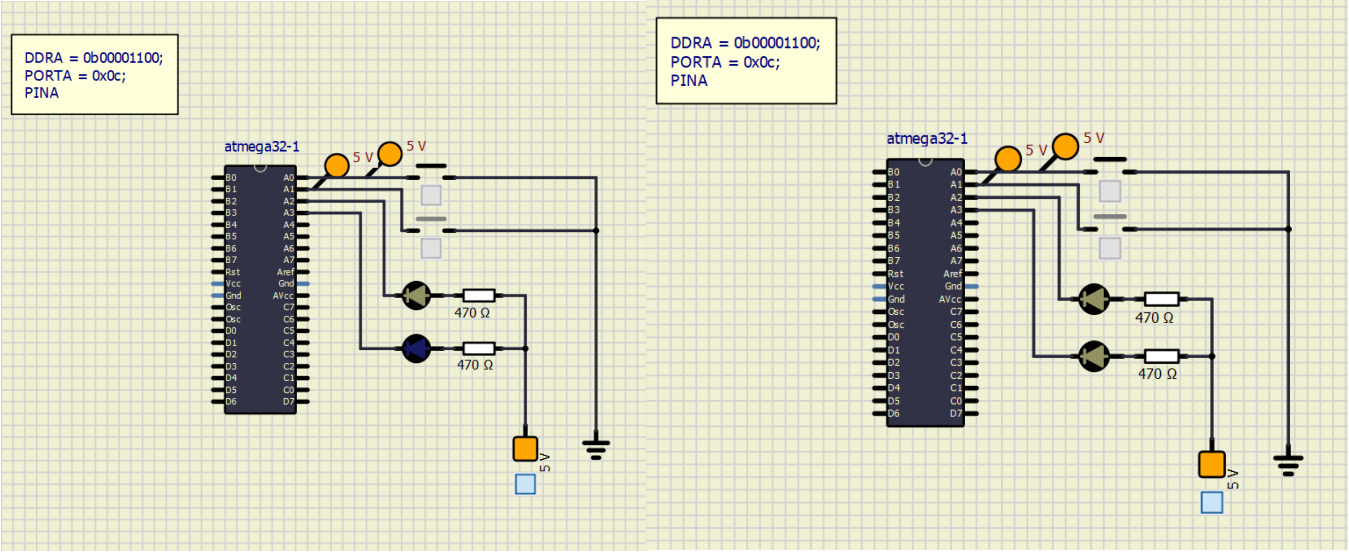
pinach I/O, a ich odczyt odbywa się poprzez sprawdzanie wartości w rejestrze PINx. Stan diody jest przełączany przez użycie odpowiednich warunków oraz maski NOT, która jest zaprzeczeniem stanu logicznego. Wprowadzone opóźnienia maja na celu redukcję drgań styków.

```c
/*
 * main.c
 *
 *  Created on: 2 gru 2020
 *      Author: Lukasz Chwistek
 */
#define __AVR_ATmega32__
#define F_CPU 8000000UL  //definiujemy F_CPU na 8MHz

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

/*
 * PINx     - rejestr zwracajacy stan pinow
 * PORTx    - rejestr do ustawiania stanów na pinach
 * DDRx     - rejestr ustalajacy kierunek pracy 0 - wejscie, 1 - wyjscie
 */

int main(void){
    DDRA |= (1 << PA2) | (1 << PA3);    //piny PA2 i PA3 ustawione jako wyjscia
                                        //pozostale sa domyslnie wejsciami
    PORTA = 0xFF;                       //wszystkie piny portu A ustawione jako stan wysoki

    uint8_t led1 = 0, led2 = 0;

    while(1)
    {


        if(!(PINA & (1 << PINA0)))      //jezeli na pinie 0 portu A wystopi 0 to
        {
            _delay_ms(200);         //opoznienie by zniwelowac drgania stykow
            if(0 == led1)
            {
                PORTA &= ~(1 << PA2);   //ustawienie wartosci 0 na pinie 2 portu A, wlaczenie diody
                led1=1;
            }
            else
            {
                PORTA |= (1 << PA2); //ustawienie wartosci 1 na pinie 2 portu A, czyli wylaczenie diody
                led1 = 0;
            }
            while(!(PINA & (1 << PINA0))){}
        }


        if(!(PINA & (1 << PINA1)))
        {
            _delay_ms(200);
            if(0 == led2)
            {
                PORTA &= ~(1 << PA3);
```

```
53                  PORTA &= ~(1 << PA3);
54                  led2 = 1;
55              }
56              else
57              {
58                  PORTA |= (1 << PA3);
59                  led2 = 0;
60              }
61              while(!(PINA & (1 << PINA1))){}
62          }
63
64      }
65
66      return 0;
67  }
68
```

4. Wyniki

Po kompilacji programów i wgraniu ich do mikrokontrolera oba programy pracują zgodnie z założeniami. Wciśnięcie przycisku zapala diodę LED, a ponowne wciśnięcie ją gasi.



5. Wnioski

Programy wykonują swoje algorytmy zgodnie z założeniami, co pokazuje, że schemat i programy zostały wykonane poprawnie. Skorzystanie z Visual Studio Code z programem WinAVR usprawniło przebieg pracy.