AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Studia podyplomowe – Elektronika Stosowana
Wydział IEiT, rok akademicki 2015/2016

Praca Podyplomowa

Projekt sterownika prędkości silnika DC z wykorzystaniem funkcji PWM mikrokontrolera ATMEGA 32

Dariusz Łuczyński

Spis Treści

Rozdział 1. Wstęp

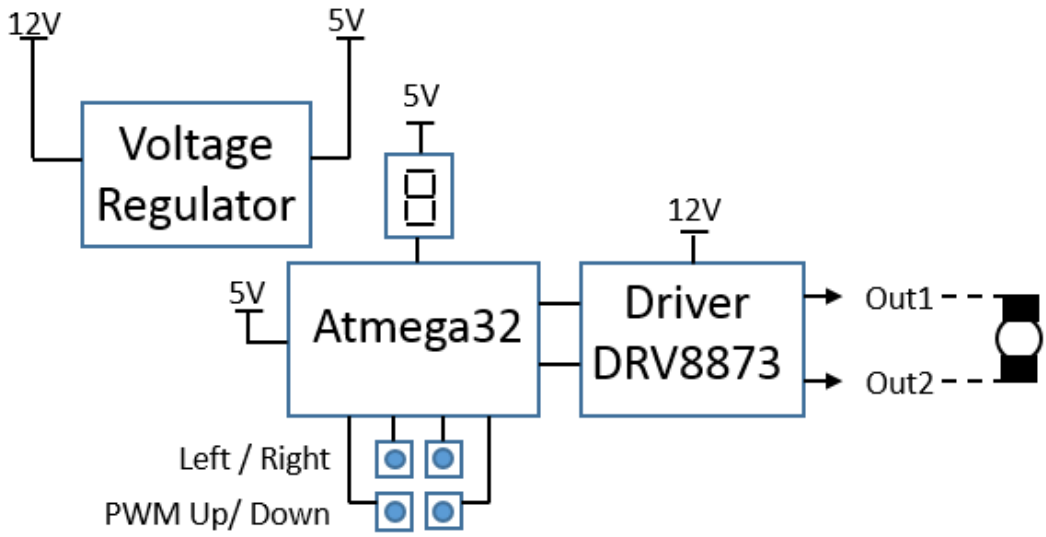
Zadaniem poniższej pracy podyplomowej jest projekt urządzenia, które pośredniczy pomiędzy silnikiem prądu stałego a człowiekiem oraz pozwala na jego sterowanie poprzez przyciski. Założeniem przy budowie poniższego sterownika jest spełnianie dwóch funkcji:

- zmiana kierunku obrotów silnika CW/CCW (Clockwise / Counter clockwise);

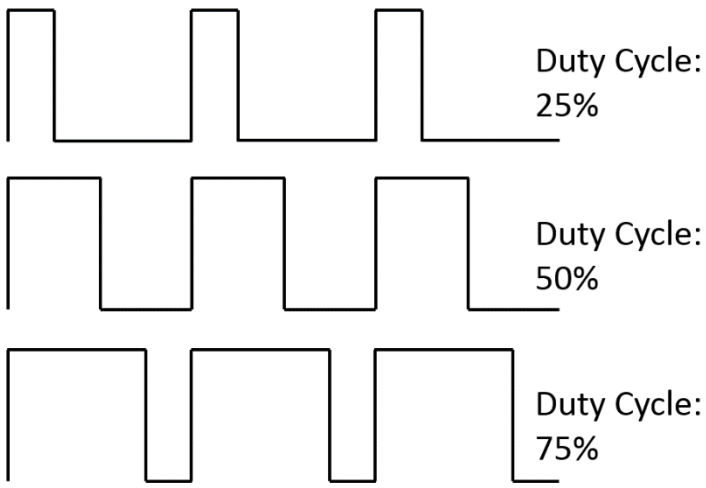
- zmiana wypełnienia sygnału PWM;

Zmiana kierunku obrotów silnika polega na naprzemiennym załączaniu sygnału sterującego na Out1 i Out2 oraz masy, tj. gdy na Out1 jest sygnał PWM to na Out2 jest masa i na zmianę.

Rysunek 1.1. Schemat ideowy układu.



Rysunek 1.2. Modulacja PWM (modulacja szerokości impulsu).



Rozdział 2. Oprogramowanie mikrokontrolera

Podstawą procesora AVR jest czasomierz/licznik (Timer/Counter), w przypadku Atmega32 są to trzy timery, które zliczają impulsy zegarowe. Po osiągnięciu maksymalnej wartości licznik przekręca się i zaczyna zliczanie od nowa. Częstotliwość sygnału zegarowego określa minimalną rozdzielczość pomiaru, a każde zwiększenie licznika będzie następowało z tą rozdzielczością. Licznik po osiągnięciu wartości maksymalnej zaczyna zliczanie od początku. Atmega32 udostępnia dwa liczniki 8-bit oraz jeden 16-bit. Częstotliwość sygnału zegarowego może być podzielona, zmniejszona poprzez użycie Preskalera, poprzez użycie jednej z predefiniowanych wartości: 1, 8, 64, 256, 1024. Odbywa się to poprzez wybranie odpowiednich bitów Clock Select.

Rysunek 2.1. Preskaler Timera0 oraz Timera1.

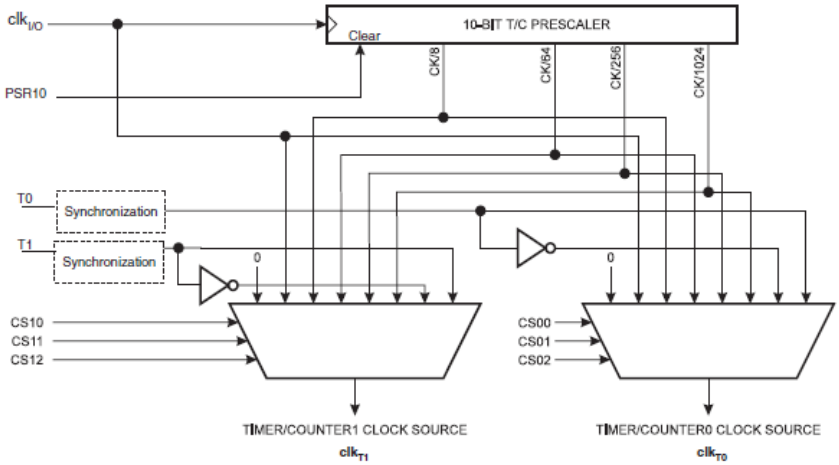
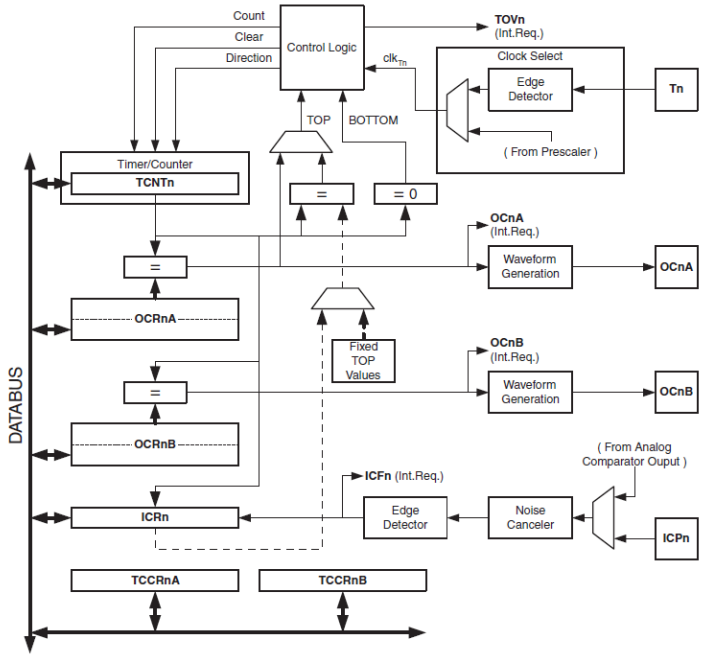


Tabela 2.1. Ustawienie bitów clock select.

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

Rysunek 2.2. Schemat blokowy Timera1.



Początek może być wartością maksymalną, w przypadku liczników 8-bit jest 255, lub wartością minimalną, czyli 0. Mikrokontroler daje możliwość zliczania sygnałów od wartości minimalnej (Bottom) do maksymalnej (Top), oraz w przeciwnym kierunku. Zakładając, że kierunek zliczania impulsów jest w kierunku od Bottom do Top, to po osiągnięciu wartości 255 następuje zmiana flagi przepełnienia licznika TOV1 (Timer Overflow), co może skutkować wykonaniem procedury obsługi przerwania, o ile taka jest zdefiniowana w programie.

Tabela 2.2. Ustawienie bitów Waveform Generation Mode (WGM) dla Timera1

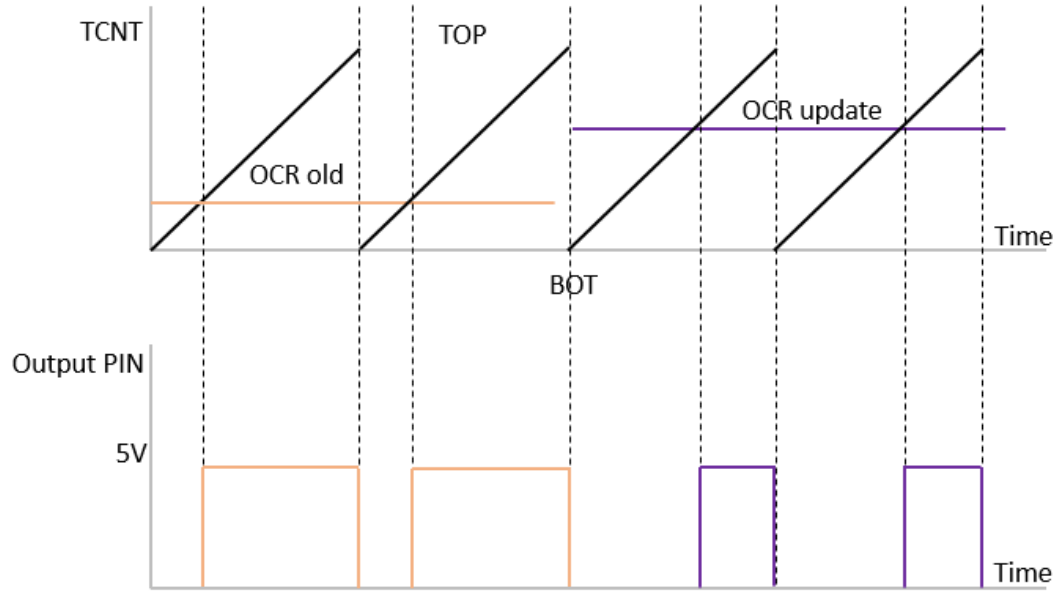
Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1x	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	Reserved	–	–	–
14	1	1	1	0	Fast PWM	ICR1	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCR1A	BOTTOM	TOP

Zliczana wartość przez licznik, rejestr TCNT1, morze zostać porównana z wartością predefiniowaną przez użytkownika, wartością, która znajduje się w rejestrze OCR1A/OCR1B (Output Compare Register). Pozytywny wynik porównania rejestru licznika powoduje ustawienie flagi OCF1A/ OCF1B i skutkuje wystąpieniem zdarzenia Compare Match oraz może powodować wygenerowanie żądania wykonania obsługi przerwania. Compare Match może również służyć do wygenerowania przebiegów na pinach IO procesora, a jego wpływ na PINy jest określany poprzez odpowiednie ustawienie bitów COM0 i COM1.

Tabela 2.3. Znaczenie bitów COM w przypadku pracy timera w trybie FAST PWM

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM13:0 = 15: Toggle OC1A on Compare Match, OC1B disconnected (normal port operation). For all other WGM13:0 settings, normal port operation, OC1A/OC1B disconnected.
1	0	Clear OC1A/OC1B on compare match, set OC1A/OC1B at BOTTOM, (non-inverting mode)
1	1	Set OC1A/OC1B on compare match, clear OC1A/OC1B at BOTTOM, (inverting mode)

Rysunek 2.3. Tryb Fast PWM.



Stąd oprócz podstawowych funkcjonalności timera, to jest działania, jako licznik oraz precyzyjnego odmierzania czasu, mierzenia czasu między impulsami, dochodzi jeszcze

możliwość generowania przebiegów PWM, generowania przebiegów o określonej częstotliwości oraz pomiarów wypełnienia przebiegów zewnętrznych.

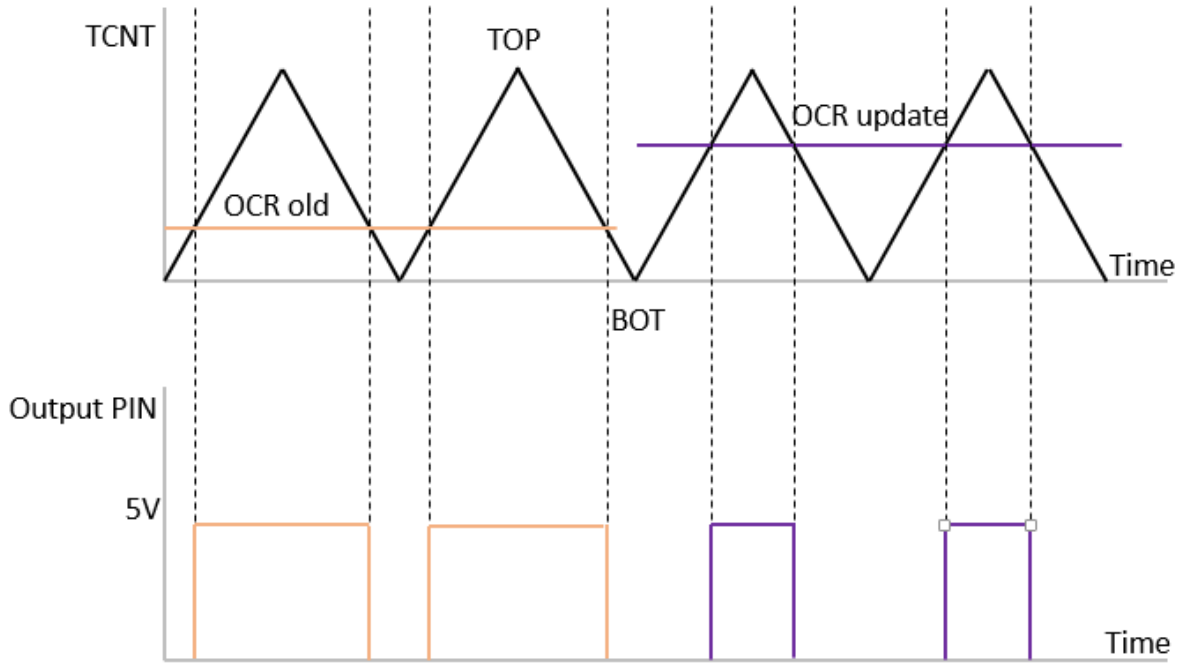
Timer może pracować w pięciu trybach:

- trybie normalnym;
- trybie CTC (Clear Timer on Compare Match);
- trybie Fast PWM;
- trybie z korekcją fazy;
- trybie z korekcją fazy i częstotliwości;

Tabela 2.4. Znaczenie bitów COM w przypadku pracy timera w trybie PWM z korekcją fazy

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM13:0 = 9 or 14: Toggle OC1A on Compare Match, OC1B disconnected (normal port operation). For all other WGM13:0 settings, normal port operation, OC1A/OC1B disconnected.
1	0	Clear OC1A/OC1B on compare match when up-counting. Set OC1A/OC1B on compare match when downcounting.
1	1	Set OC1A/OC1B on compare match when up-counting. Clear OC1A/OC1B on compare match when downcounting.

Rysunek 2.4. Tryb PWM z korekcją fazy (i/lub korekcją częstotliwości)



W trybie normalnym, wbudowany Waveform Generation Mode nie jest wykorzystywany, natomiast używa się interwałów czasowych ograniczanych przez preskaler, w których na port wyjściowy, OC0 lub OC1, jest podawany stan wysoki.

Tryb CTC wymaga ustawienia odpowiednich bitów WGM, zgodnie z Tabelą 2.2, w rejestrach TCCR. Umożliwia na precyzyjniejszą kontrolę nad częstotliwością przebiegu na pinach wyjściowych, która wynosi Fclk / (2*Preskaler*(1+OCR1A)). Licznik zlicza do OCR1A, i za każdym razem, gdy osiąga tą wartość, generowane jest zdażenie TIMER1 COMP lub TIMER1 CAPT (o ile ustawiamy ICR). Przerwanie przepełnienia licznika TIMER1 OVF nie występuje, licznik nie zlicza do maksymalnej wartości.

Podobnie jak CTC tryb Fast PWM wymaga odpowiedniego zdefiniowania bitów WGM. Licznik zlicza od wartości Bottom do wartości Top, a po jej osiągnięciu, zlicza ponownie od wartości Bottom. Częstotliwość wynosi Fclk / (*Preskaler*(1+Licznik max)), gdzie Licznik max, jest maksymalną wartością trybu fast PWM, np. dla 8-bit jest to 255.

W przypadku trybów PWM z korekcją fazy oraz z korekcja fazy częstotliwości licznik odlicza od wartości Bottom do Top, a następnie od Top do Bottom, jak na Rysunku 2.4. Częstotliwość wynosi Fclk/(2*Preskaler(1+Wartość Top)). Oba tryby powinny mieć ustawione odpowiednie flagi WGM.

Przed przygotowaniem oprogramowania zostały przyjęte następujące założenia:

- zostanie wykorzystany wewnętrzny zegar mikroprocesora, bez posiłkowania się zewnętrznym rezonatorem kwarcowym;

- sterowanie będzie obywało się za pomocą czterech przycisków, a będzie realizować inkrementację i dekrementację sygnału PWM, a także zmianę wyjścia, na którym jest on podawany;

- do sterowania nie zostanie wykorzystany mechanizm przerwań;

- zostanie wykorzystany tryb Fast PWM;

- oprogramowanie zostanie napisane w języku C;

Oprogramowanie składa się z dwóch plików z kodem źródłowym, „constants.h” zawiera stałe oraz częstotliwość taktu zegarowego, natomiast w „sterownikdc.c” znajduje się pełen kod źródłowy programu.

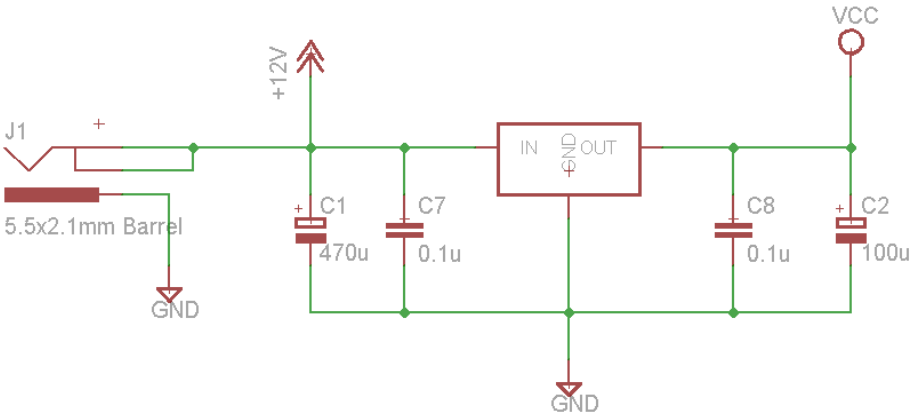
Główny kod źródłowy programu definiuje globalne flagi, które określają, na jakim PIN jest PWM (OC1 lub OC0), oraz wartość wypełnienia. Jest zdefiniowana również tablica, która zawiera dwie zmienne ośmiobitowe typu unsign integer, służące do wyświetlania na przemian „l” lub „r”, w zależności od kierunku obrotów silnika. Tablica jest umieszczona w pamięci flash. W funkcji Main programu w pierwszej kolejności następuje inicjalizacja Timer1, za pomocą metody timer1_initialize(), ustawienie bitów WGM dla Fast PWM, bitów Compare Match A/B, a także rejestrów Output Compare Register A/B. Kierunek portów A zostaje ustawionyna wyjście, będą one obsługiwać wyświetlacz siedmiosegmentowy. Zostaje zdefiniowana częstotliwość Fclk/ (Preskaler * 256), w tym przypadku 1 MHZ/ (8 * 256), 488 Hz. Na czterech mniej znaczących bitach portu D zostaje ustawiony stan wysoki, porty te są wykorzystywane

do obsługi przycisków. Nieskończona pętla programu wywołuje kolejno trzy metody. Funkcja scan_key() sprawdza stan przycisków (jest to tzw. pooling). Odczytywana jest wartość z rejestru PIND, natępenie zmieniana jest flaga lub wartość wypełnienia PWM. w zależności od tego, który przycisk został wciśnięty. Dekrementacja zmiennej counter, służy, jako mikroopóźnienie, ze względu na mechaniczne drgania przycisków występujące po ich naciśnięciu, „bouncing”. Funkcja show_direction() przełącza kierunek obrotów silnika oraz zmienia pokazywaną literę na wyświetlaczu siedmiosegmentowym. Ostatnia w kolejności metoda pwm_set() ma za zadanie ustawienie we właściwym OCR, wartości wypełnienia, która została uprzednio inkrementowana lub dekrementowana.

Rozdział 3. Schemat elektryczny i obwód drukowany

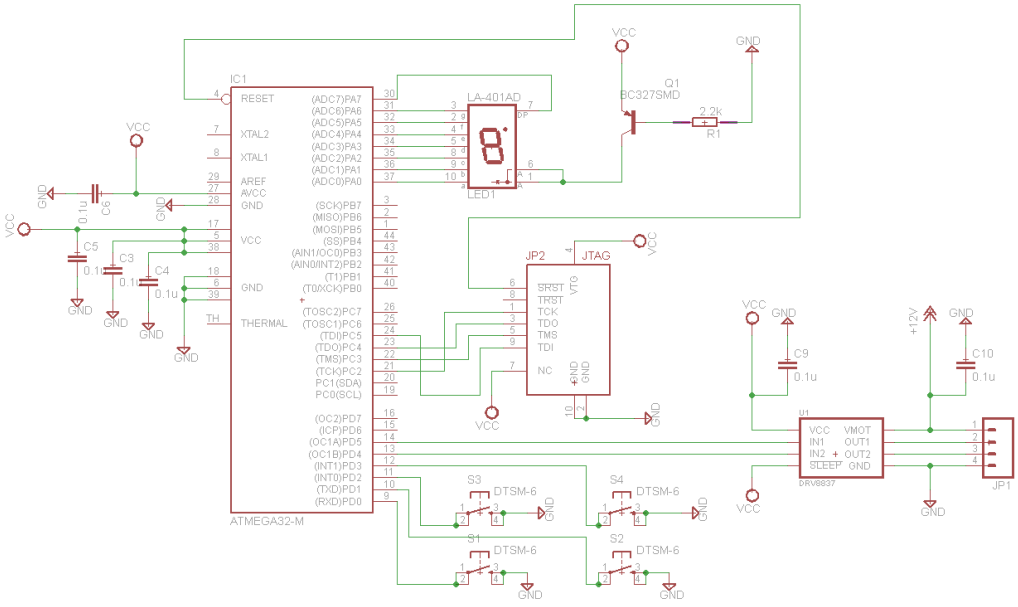
Do układu zostaje doprowadzone 12V DC z zewnętrznego zasilacza impulsowego do 2.1 mm gniazda, które zostaje przekształcone na napięcie 5V, przy użyciu popularnej przetwornicy 7805.

Rysunek 3.1. Schemat stabilizatora napięcia.



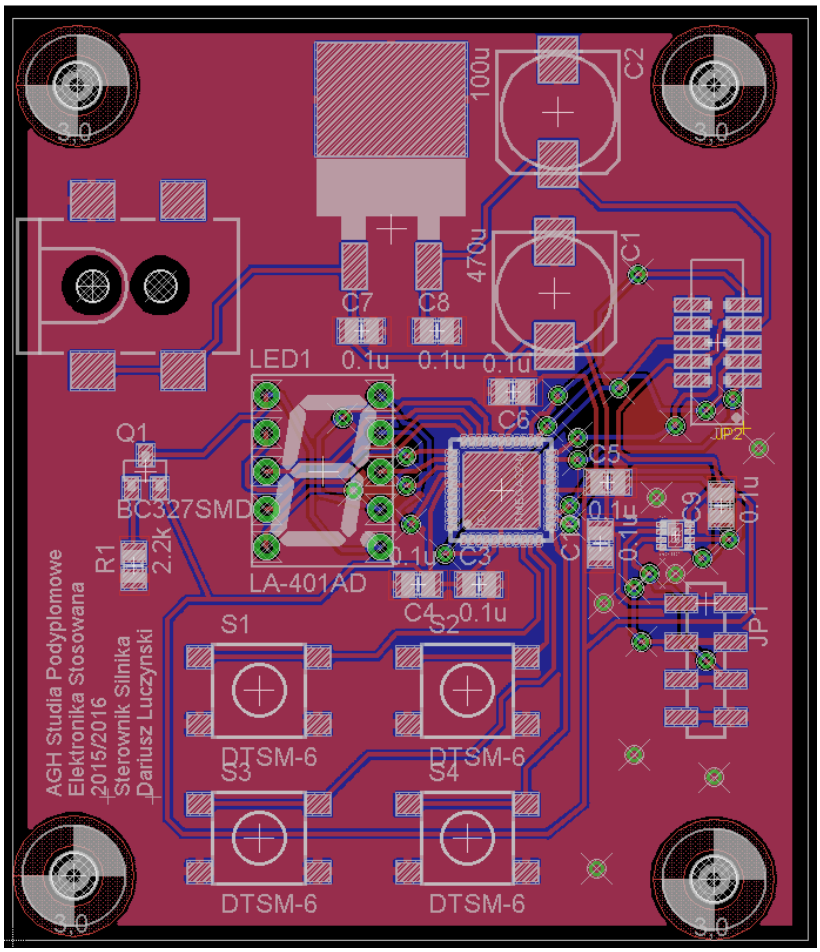
Z portu A zostało wyprowadzonych osiem bitów kontrolujących wyświetlacz siedmiosegmentowy, w tym DP. Zasilanie jest wpięte poprzez tranzystor BC327. Wyświetlacz jest typu: wspólna anoda. Musi zostać podany stan niski, aby świeciły poszczególne segmenty. Do portu C jest doprowadzony interfejs JTAG. Do portu D0~D3 zostały przyłączone linie odpowiadające za przyciski. Na liniach utrzymuje się stan wysoki, dopóki dany przycisk nie zostanie naciśnięty, zwarty do masy. Linie D4 (OC1B) i D5 (OC1A) doprowadzają sygnał PWM do drivera. W projekcie został użyty driver DRV8837 TI, niemniej nie jest to najlepszy wybór, ponieważ Vmot, 12V, w tym przypadku przekracza maksymalne napięcie z karty katalogowej.

Rysunek 3.2. Schemat elektryczny układu sterującego.



Obwód drukowany został przygotowany przede wszystkim do montażu powierzchniowego, niemniej na płycie znajduje się jeden komponent z wyprowadzeniami przewlekanymi (wyświetlacz LA-401AD). Wszystkie komponenty są montowane po stronie TOP, niemniej ścieżki zostały poprowadzone po stronie TOP jak i BOT. Ścieżki są 10 MILS, wynika to z 'footprintu' mikrokontrolera. Sygnały wyjściowe ze sterownika DRV8837 oraz masa i napięcie zasilające zostały wyprowadzone cztero-pinowym gniazdem wykonanym w technologii SMD.

Rysunek 3.3. Obwód drukowany.

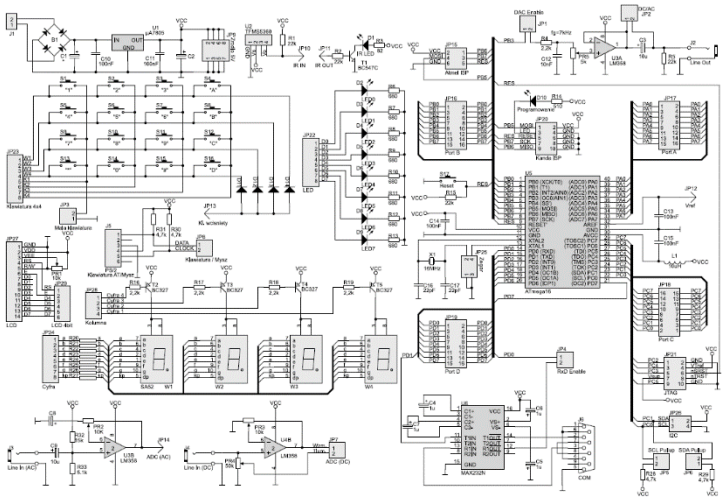


Pojedynczy obwód drukowany nie jest wystarczającym projektem, w przypadku produkcji większej niż prototypowa. Takie PCB należy „zpanelizować", tj. połączyć w minimum cztery obwody na jednym laminacie, pozwoli to zredukować koszty produkcji. Należy również dodać tzw. „fiduciale", czyli znaczniki, które pozwolą maszynom Pick'n'Place właściwie ustawić początek układów współrzędnych.

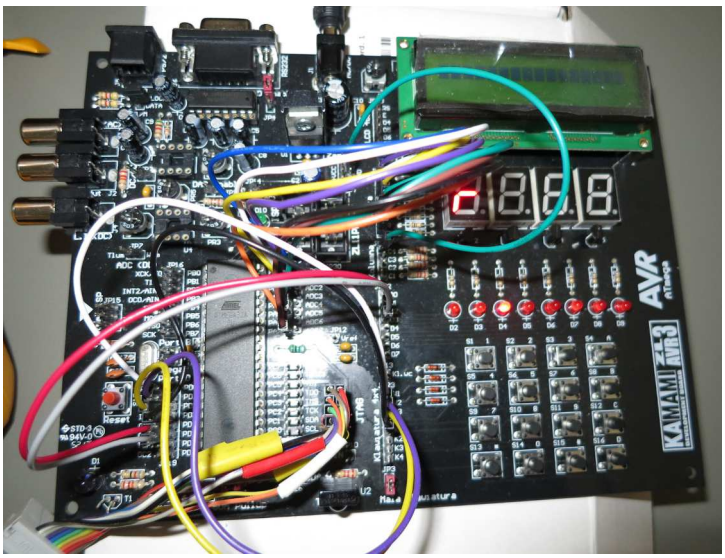
Rozdział 4. Testy

Do sprawdzenia poprawności działania programu został użyty zestaw uruchomieniowy Kamami ZL3. Do JP24, wyprowadzenia wyświetlacza W1, zostały wpięte linie portu A (JP17). Pin pierwszy JP28 został podłączony do uziemienia. Do portów D0~D3 (JP19) został przyłączona „mała klawiatura”: S1, S5, S9, S13. Na JP3 została założona zwora. OC1B (port D4) zostało podane na pin trzeci, natomiast OC1A na pin czwarty JP22.

Rysunek 4.1. Schemat elektryczny zestawu uruchomieniowego ZL3.

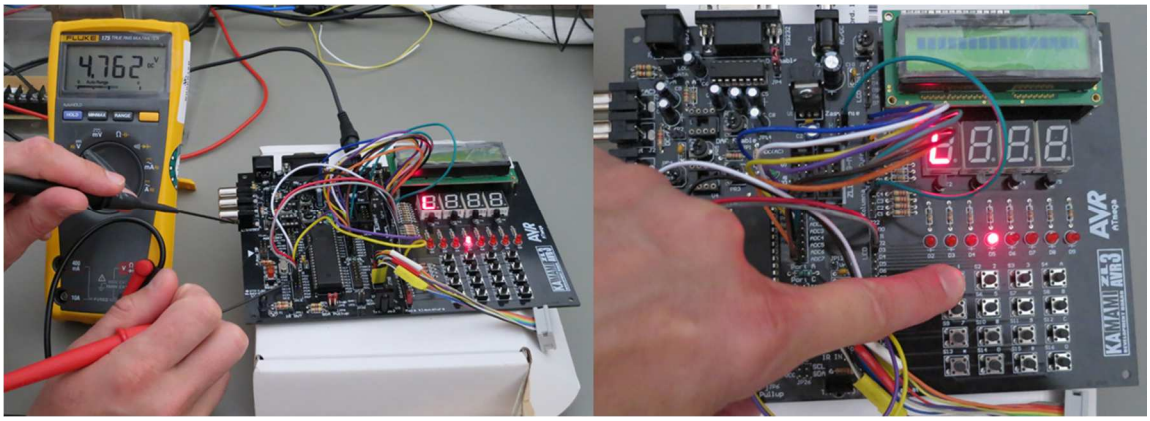


Rysunek 4.2. Podłączenie we/wy na ZL3.



Program działa zgodnie z założeniami. Przy włączonym OC1A, naciśnięcie S1 nie powoduje reakcji wyświetlacza, ani zmian napięcia. Reakcja następuje po naciśnięciu S5, wyświetlacz pokazuje literę „r”, oraz zamiast diody D5, świeci D4. Naciskając S9 lub S13, LED mniej lub bardziej intensywnie.

Rysunek 4.3. Naciśnięcie przycisku S1



Rysunek 4.4. Naciśnięcie przycisku S5

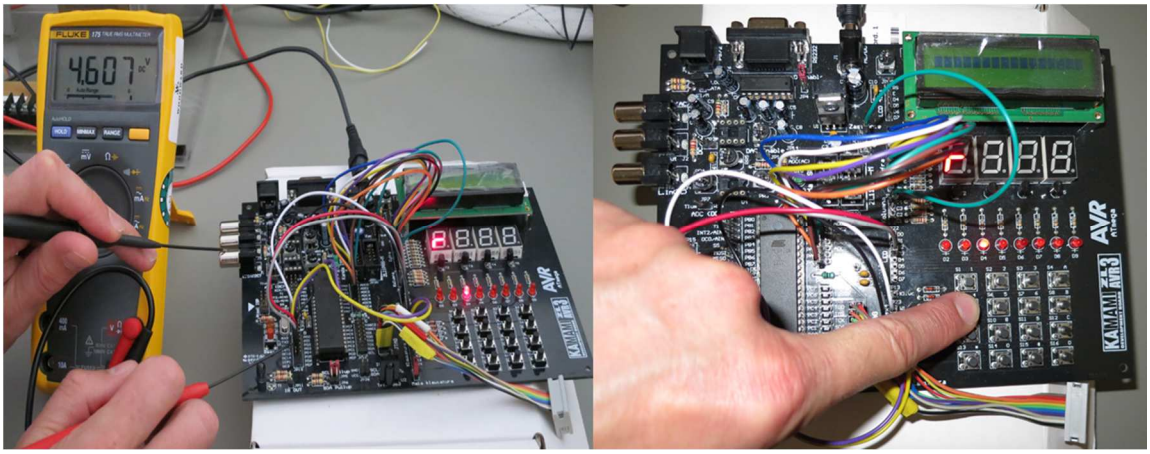


Tabela 4.1. Lista materiałowa (BOM)

Number	Description	Value Device Package Description	Designator	EA
1	Connector	M04SMD_STRAIGHT_COMBO 1X04_SMD_STRAIGHT_COMBO Header 4 Total	JP1	1
2	Capacitor	0.1u C-EUC2012 C2012 CAPACITOR, European symbol	C10 C3 C4 C5 C6 C7 C8 C9	8
3	Capacitor	100u CPOL-EU175TMP-0810 175TMP-0810 POLARIZED CAPACITOR, European symbol	C2	1
4	Resitor	2.2k RR2012 R2012 RESISTOR, European symbol	R1	1
5	Capacitor	470u CPOL-EU175TMP-0810 175TMP-0810 POLARIZED CAPACITOR, European symbol	C1	1
6	Connector	5.5x2.1mm Barrel POWER_JACKSMD POWER_JACK_SMD Power Jack	J1	1
7	IC	ATMEGA32-M ATMEGA32-M MLF44-TH MICROCONTROLLER	IC1	1
8	Transistor	BC327SMD BC327SMD SOT23-BEC PNP Transistror	Q1	1
9	IC	DRV8837 DSG	U1	1
10	Switch	DTSM-6	S1 S2 S3 S4	4
11	Connector	JTAGICE3-CONNECTOR50MIL-SMD JTAGICE3-CONNECTOR50MIL-SMD JTAGIC3-50MIL-SMD JTAG Connector for JTAGICE3	JP2	1
12	Display	LA-401AD LA-401AD LA-401 Single Digit LED Numeric Display (anode common)	LED1	1
13	IC	VREG-7805-DPAK2 VREG-7805-DPAK2 VREG-7805-DPAK2	U2	1

Dane projektowe, schemat, projekt obwodu drukowanego oraz BOM są dostępne na GitHub:

https://github.com/LuczynskiDar/SilnikDC/

Rozdział 5. Wykonanie projektu

Dalszym etapem jest wygenerowanie, bazując na zaprojektowanym obwodzie drukowanym, tzw. „gerberów", czyli plików CAM (Computer Aided Manufacturing), NC Drill wskazujących miejsce oraz średnicę wierteł, oraz plików tekstowych zawierających koordynaty dla maszyn do montażu powierzchniowego. Gerbery do produkcji obwodu drukowanego należy przesłać do jednego z lokalnych producentów PCB: Hatron, Techno-Service, Evatronix etc. Pliki opisujące apertury dla matryc do sitodruku należy udostępnić firmie produkującej tzw. „stencile", np. Scanditron lub Semicon. Do montażu powierzchniowego można użyć pasty lutowniczej ołowiowej (OM-5100) lub bezołowiowej (np. nisko srebrowa OL107F). Wyświetlacz siedmiosegmentowy, jedyny element przewlekany, można przylutować ręcznie, lub przy użyciu niewielkiego tygla lutowniczego.

Rysunek 5.1. HAKKO FX-300, Dip solder system /hakko.com/

Należy pamiętać, że skala produkcji powinna być uzasadniona zwrotem z inwestycji.

Bibliografia

1. ATmega32 (32L) - 8-bit AVR Microcontroller with 32KBytes In-System Programmable Flash.
2. Język C dla mikrokontrolerów AVR. Od podstaw do zaawansowanych aplikacji, Tomasz Francuz, Helion 2011.
3. AVR Atmega w praktyce. Rafał Baranowski, BTC 2015.

Załącznik A. Kod programu

Kod programu A.1. Constants.h

```
#ifndef CONSTANTS_H_
#define CONSTANTS_H_

//Clock
#define F_CPU 1000000UL

// Display LEDs
#define KEYS 2
// Definition of default PWM, which is 40% duty cycle
#define PWMDEF 102
#define PWMGND 0

#endif /* CONSTANTS_H_ */
```

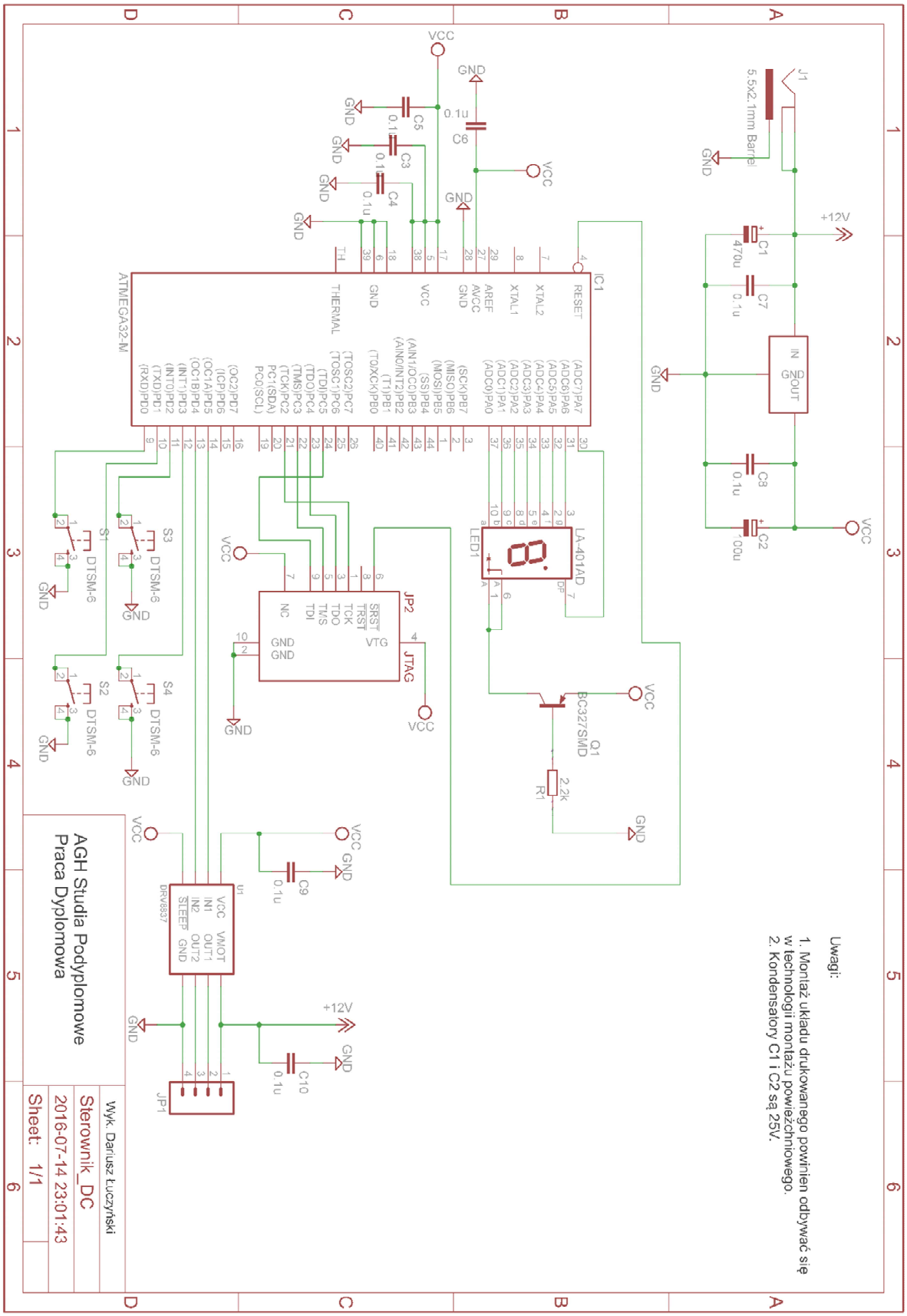
Kod programu A.2. Sterownikdc.c

```
/*
 * Sterownik_DC.c
 *
 * Author : Dariusz Luczynski
 */



#include "constants.h"
#include <avr/pgmspace.h>
#include <stdlib.h>
#include <util/delay.h>
#include <avr/io.h>
#include <avr/interrupt.h>

//To set the two element LED  display
static const uint8_t PROGMEM DIRECTION[2]={0xC7, 0xAF};

// The direction indicator flag
static uint8_t dir_flag = 0xFF;

static uint8_t handler = 0;
// Direction flag indicator
volatile uint8_t pwm_flag=0xFF;

volatile uint8_t pwm_handler=102;

// Read pins indicator
volatile uint8_t pin_reader;

// Initialization of Timer1 for the PWM outputs
void timer1_initialize(void)
{

// To set the prescaler to divide the frequency per 8, resulting 488 Hz
       TCCR1B |= (1<<CS11);
//  To set the Waveform Generator Mode to 8-bit Fast PWM
```

```
	TCCR1A |= (1<<WGM10);
	TCCR1B |= (1<<WGM12);
//  Set the PINs to zero, after the compare match event,
// when the minimum value is being reached
	TCCR1A |= (1<<COM1A1) | (1<<COM1B1) ;

// Set to the direction registers, PIN 4 and PIN5
	DDRD |= (1<<PD4) | (1<<PD5);

//Initialize output compare registers
	OCR1A =	PWMDEF;
	OCR1B = PWMGND;

}


//To chose from the tables a proper value for a proper LED display
//Inline do not recall the function just puts the code of the function there

void scan_key(void)
{
	uint8_t temp_dir = dir_flag;
	uint8_t temp_hand = handler;
	uint8_t temp_pwm = pwm_handler;

	uint8_t counter=0;
	if(counter==0)
	{
		pin_reader = ((~PIND) & 0x0F);
		// Set the column and calculate the key number
		if((pin_reader == 0x01) && ( (temp_dir & temp_hand) == 0))
		{
			temp_dir = 0xFF;
			temp_hand = 0x00;
		}
		else if((pin_reader ==0x02) && ( (dir_flag & temp_hand) == 0))
		{
			temp_dir = 0x00;
			temp_hand = 0xFF;
		}
		else if((pin_reader == 0x04) && ( temp_pwm > 0))
		{
			temp_pwm--;
		}
		else if((pin_reader == 0x08) && ( temp_pwm < 255))
		{
			temp_pwm++;

		}
		counter=100;
	} else counter--;

	dir_flag = temp_dir;
	handler = temp_hand;
	pwm_handler = temp_pwm;
}

void pwm_set(void)
{
	if (pwm_flag == 0xff) OCR1A = pwm_handler;
	if (pwm_flag == 0x00) OCR1B = pwm_handler;
}
```

```
void show_direction( void)

{
	uint8_t temp = 0xFF;
	uint8_t pwm_holder;
	PORTA = 0xFF;
	if(dir_flag == 0xFF)
	{
		temp = pgm_read_byte(&DIRECTION[0]);

		pwm_holder = OCR1A;
		DDRD &= ~(1<<PD4);
		DDRD = (1<<PD5);
		OCR1B = pwm_holder;
		pwm_flag = 0xFF;
	}
	if(dir_flag == 0x00)
	{
		temp = pgm_read_byte(&DIRECTION[1]);
		pwm_holder = OCR1B;
		DDRD &= ~(1<<PD5);
		DDRD = (1<<PD4);
		OCR1A = pwm_holder;
		pwm_flag = 0x00;
	}
	PORTA = temp;
}


int main(void)
{
    timer1_initialize();
	   DDRA = 0xFF;
	   PORTD |= 0x0F;
    while (1)

    {
		scan_key();
		show_direction();
		pwm_set();
		_delay_ms(10);
    }
}
```

Załącznik B. Schemat elektryczny układu.



Załącznik C. Obwód drukowany.