

# BUKU AJAR

MATA KULIAH MIKROKONTROLER

HAPPY NUGROHO, S.T., M.T.



LABORATORIUM KOMPUTASI & TEKNOLOGI INFORMASI

UNIVERSITAS MULAWARMAN

FAKULTAS TEKNIK

PROGRAM STUDI TEKNIK ELEKTRO

TAHUN 2022

# LAB 1. Manipulasi Display LCD

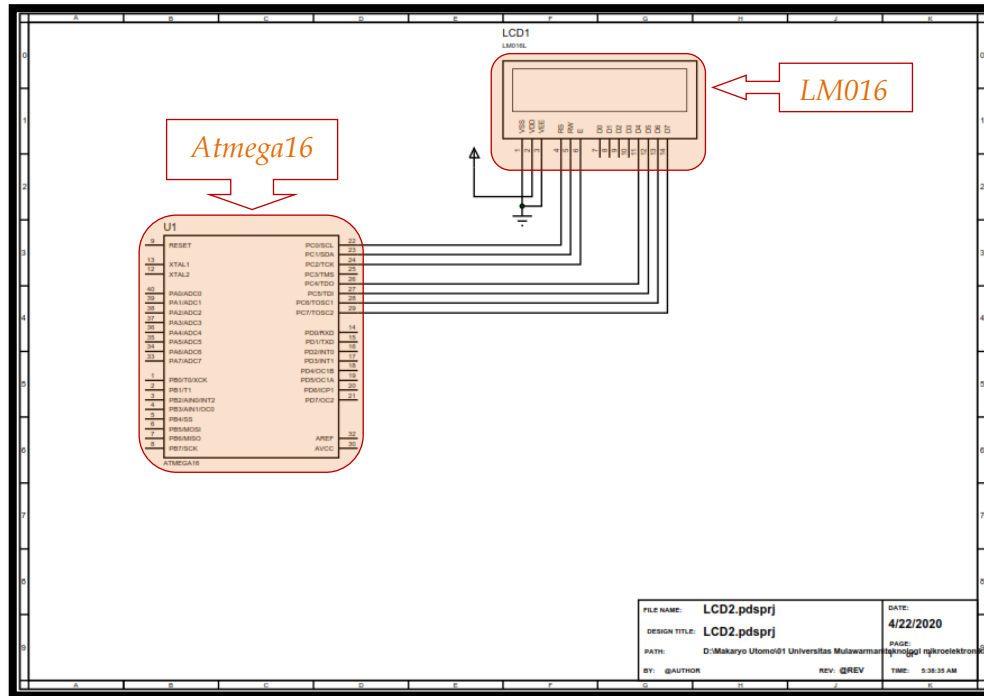
Asisten Percobaan :

NIM :

Tanggal Percobaan :

## Lab 1. Display LCD

### Latihan menampilkan kalimat ke dalam layar LCD



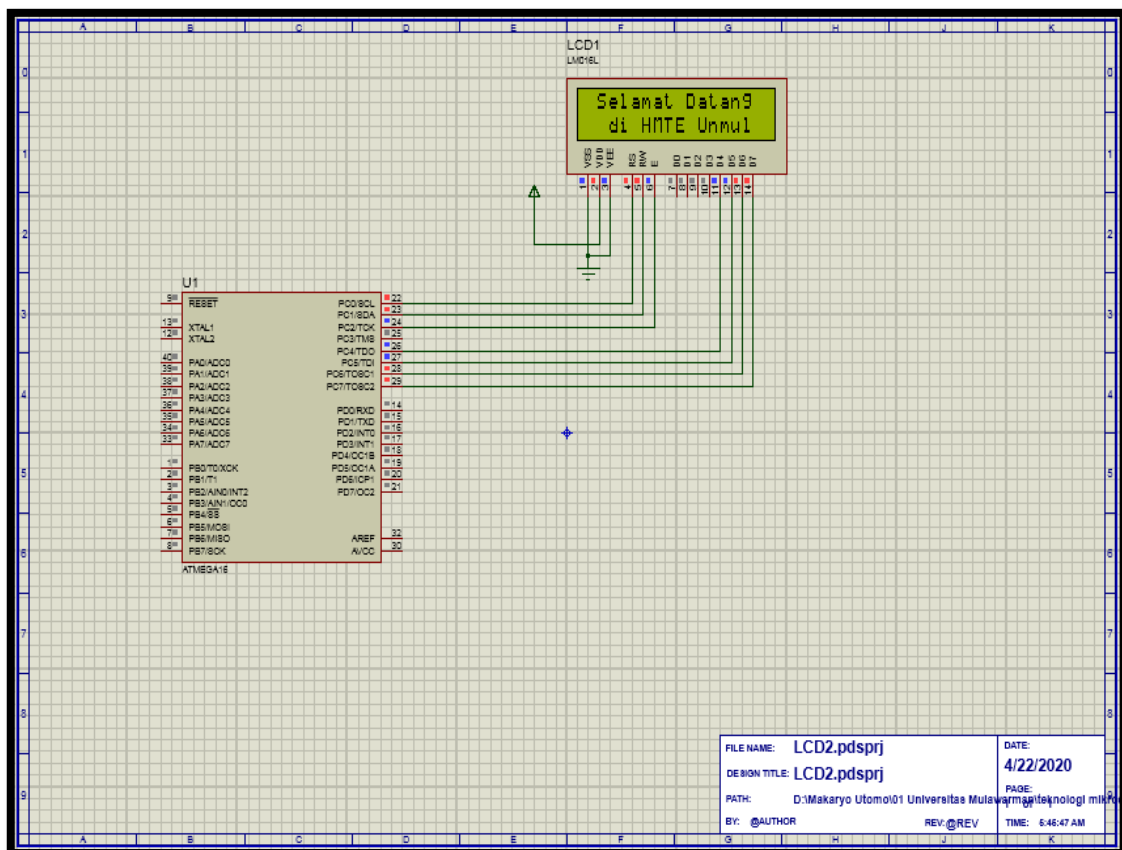
Gambar 1 Rangkaian skematik display LCD

```
CodeVisionAVR - D:\Makaryo Utomo\01 Universitas Mulawarman\teknologi mikroelektronika\AVR\LCD.pj
File Edit Search View Project Tools Settings Help
Code Navigator
Project: LCD
LCD_mega16.c
Headers
lcd.h
mega16.h
mega16_bits.h
mega32.h
mega32_bits.h
stdarg.h
stdio.h
List Files
LCD.asm
LCD.lst
LCD.map
Other Files
Notes LCD_mega16.c
1 #include <mega16.h>
2
3 #asm
4 .equ __lcd_port = 0x15; // lsd di portC
5 #endasm
6 #include <lcd.h>
7
8 //unsigned char Tulisan1[] = "Selamat datang!"; //adi 1 byte data dengan memori 8 bit; "[]" = tipe data array
9 //unsigned char Tulisan_2[] = "Happy";
10
11 void main (void)
12 {
13
14     lcd_init(16);           //karena menggunakan LM016
15     lcd_clear();
16     lcd_gotoxy(1,0);      //(col,row)
17     lcd_putsf("Selamat Datang"); //puts utk men-disp variabel; putsfuk men-disp huruf
18     lcd_gotoxy(2,1);
19     lcd_putsf("di HUTE Unmul");
20
21     while (1)
22     {
23     }
24 }
25
Messages
Errors Warnings
18:17 Insert
```

Gambar 2. Coding untuk menuliskan instruksi pada Code Vision AVR

Langkah mempersiapkan Alat dan Bahan, yakni:

1. Laptop yang telah ter-*install* aplikasi *Proteus 8 Professional* dan *Code Vision AVR*!
2. Merancang Rangkaian Skematik seperti terlihat pada Gambar 1, menggunakan aplikasi *Proteus 8 Professional* pada laptop masing-masing!
3. Menuliskan "*Coding*" seperti terlihat pada Gambar 2 ke dalam aplikasi *Code Vision AVR*!
4. Meng-*Upload* "*Coding*" yang telah dituliskan, ke dalam desain rancangan Rangkaian Skematik kalian masing-masing!
5. *Screenshot* atau *Capture* hasilnya yang tampak di *LCD LM016!* seperti tampak pada Gambar 3 di bawah.



**Gambar 3. Contoh hasil percobaan Lab. 1**

6. Meng-*Upload* file (*fullscreen!*) dalam bentuk format **\*.pdf** ke Google Classroom, antara lain:
  - a. Rangkaian Skematik
  - b. *Coding*
  - c. Hasil *Capture*

# **LAB 2. Merancang Jam Digital DS1302 dan LM016**

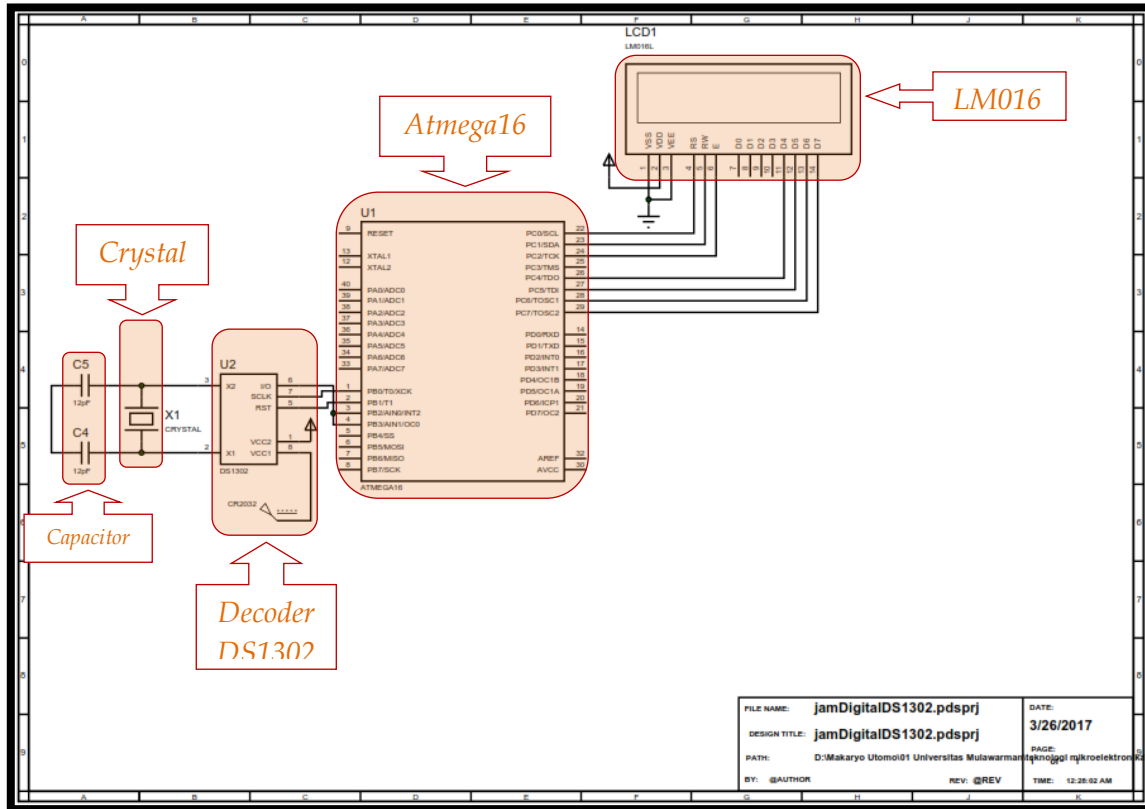
Asisten Percobaan :

NIM :

Tanggal Percobaan :

## Lab 2. Merancang Jam Digital

Latihan merancang sebuah jam digital dengan dimensi jam:menit:detik, menggunakan *chip Real Time Clock (RTC) DS1302* dan menampilkannya ke LCD LM016.

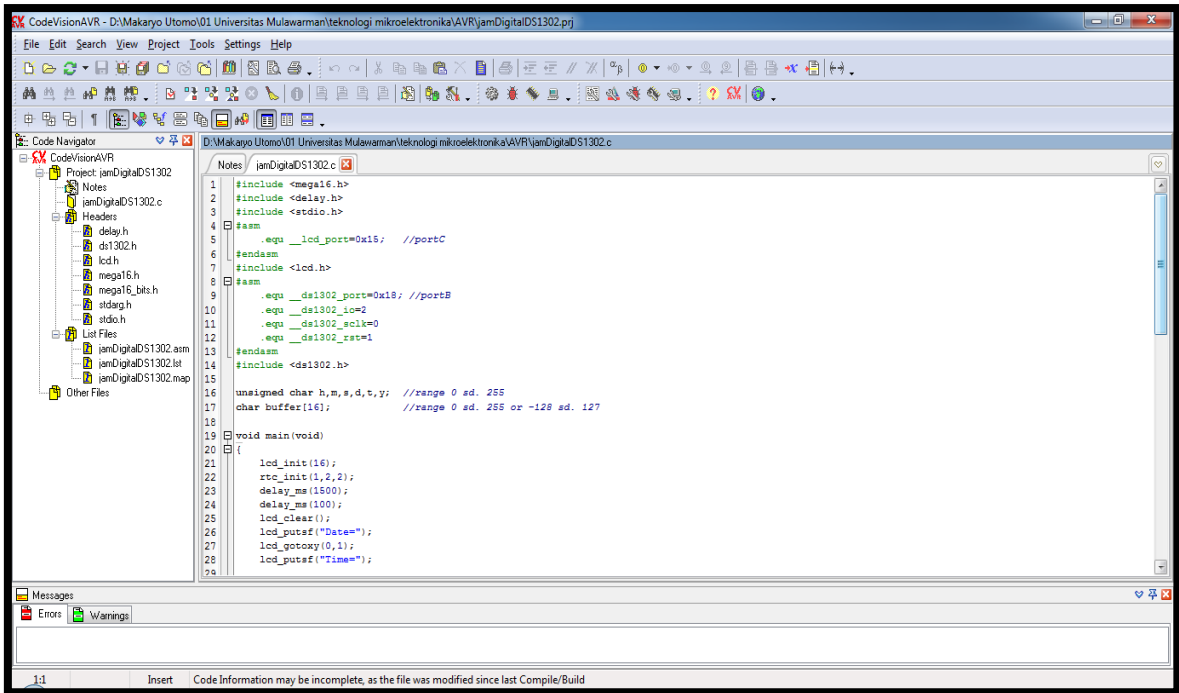


Gambar 4 Rangkaian skematik Jam Digital

Langkah mempersiapkan Alat dan Bahan, yakni:

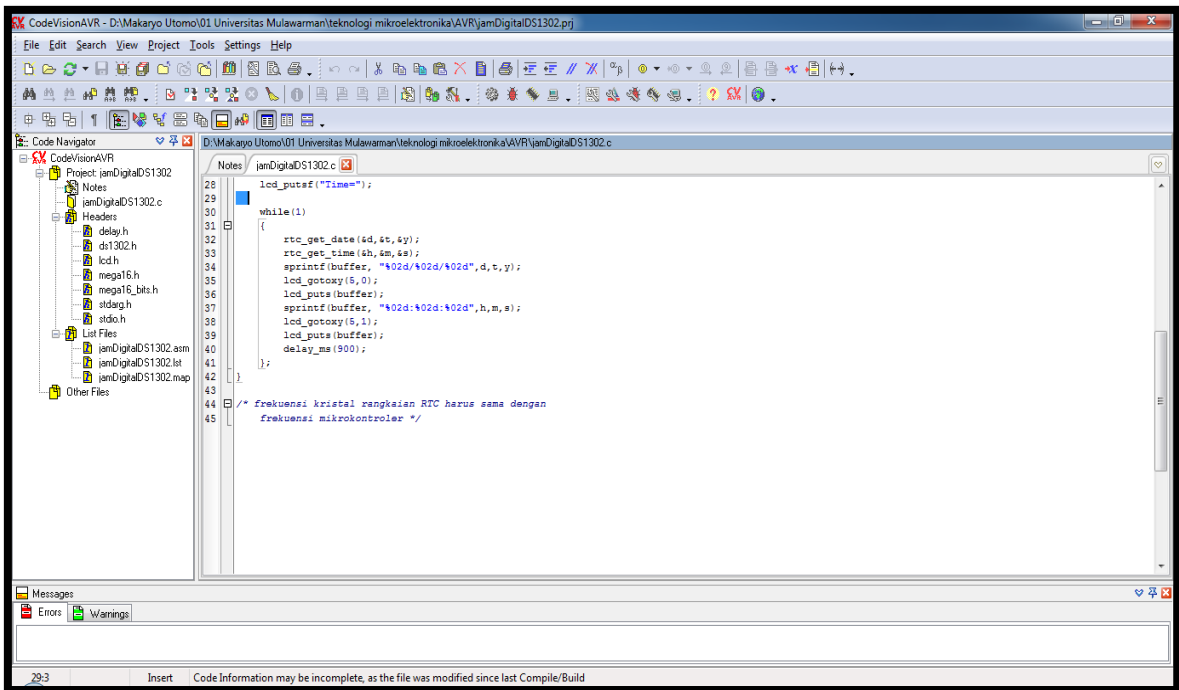
1. Laptop yang telah ter-install aplikasi *Proteus 8 Professional* dan *Code Vision AVR*!
2. Merancang Rangkaian Skematik seperti terlihat pada Gambar 4, menggunakan aplikasi *Proteus 8 Professional* pada laptop masing-masing!
3. Menuliskan "*Coding*" seperti terlihat pada Gambar 5 ke dalam aplikasi *Code Vision AVR*!
4. Meng-*Upload* "*Coding*" yang telah dituliskan, ke dalam desain rancangan Rangkaian Skematik kalian masing-masing!
5. *Screenshot* atau *Capture* hasilnya! seperti tampak pada Gambar 6.
6. Meng-*Upload* file (*fullscreen!*) dalam bentuk format **\*.pdf** ke Google Classroom, antara lain:
  - a. Rangkaian Skematik
  - b. *Coding*
  - c. Hasil *Capture*

Adapun *Coding* yang digunakan dalam Modul Lab 2 adalah:



```
1 #include <mega16.h>
2 #include <delay.h>
3 #include <stdio.h>
4 #asm
5 .equ __lcd_port=0x15; //portC
6 #endasm
7 #include <lcd.h>
8 #asm
9 .equ __ds1302_port=0x18; //portB
10 .equ __ds1302_io=2
11 .equ __ds1302_clk=0
12 .equ __ds1302_est=1
13 #endasm
14 #include <ds1302.h>
15
16 unsigned char h,m,s,d,t,y; //range 0 sd. 255
17 char buffer[16]; //range 0 sd. 255 or -128 sd. 127
18
19 void main(void)
20 {
21     lcd_init(16);
22     rtc_init(1,2,2);
23     delay_ms(1500);
24     delay_ms(100);
25     lcd_clear();
26     lcd_putsf("Date=");
27     lcd_gotoxy(0,1);
28     lcd_putsf("Time=");
29 }
```

(a)

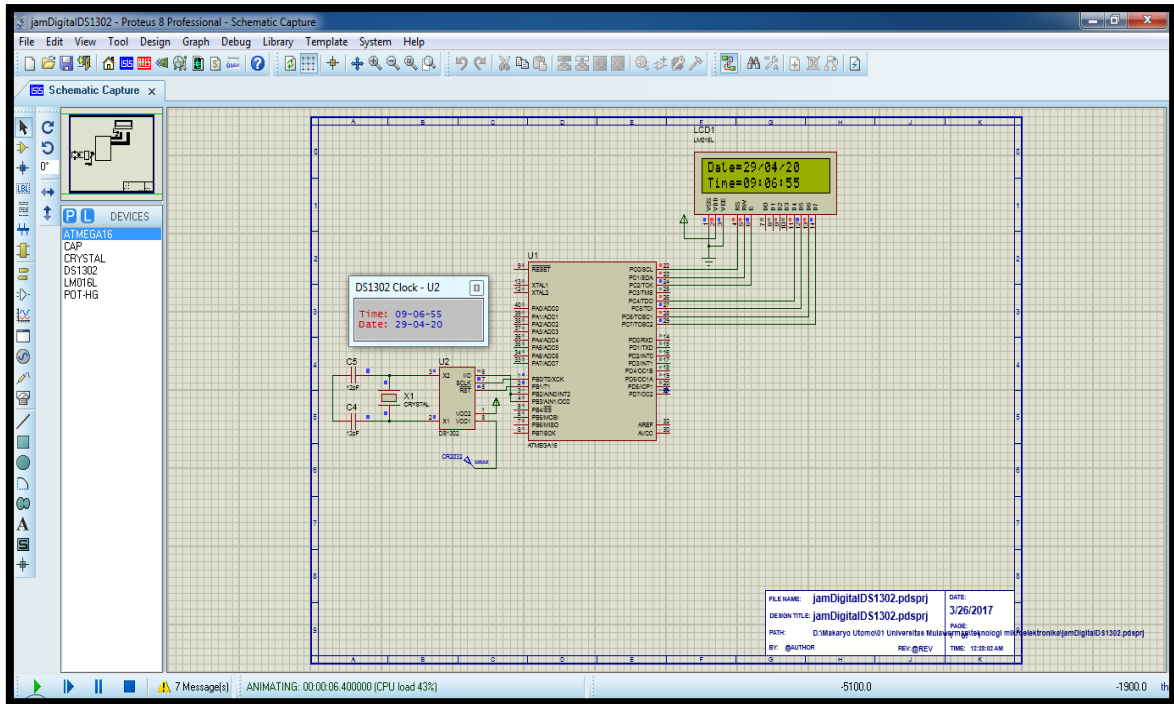


```
28     lcd_putsf("Time=");
29
30     while(1)
31     {
32         rtc_get_date(&d,&t,&y);
33         rtc_get_time(&h,&m,&s);
34         sprintf(buffer, "%02d/%02d/%02d", d,t,y);
35         lcd_gotoxy(6,0);
36         lcd_puts(buffer);
37         sprintf(buffer, "%02d:%02d:%02d",h,m,s);
38         lcd_gotoxy(6,1);
39         lcd_puts(buffer);
40         delay_ms(900);
41     }
42 }
43
44 /* frekuensi kristal rangkaian RTC harus sama dengan
45 frekuensi mikrokontroler */
```

(b)

**Gambar 5.(a) (b). Coding untuk menuliskan instruksi pada Code Vision AVR**

Hasil *Capture* Percobaan Lab 2 (merancang jam digital), yakni:



**Gambar 6. Contoh hasil percobaan Lab. 2**



**LAB 3.**

**Manipulasi Data I/O (Part I)**

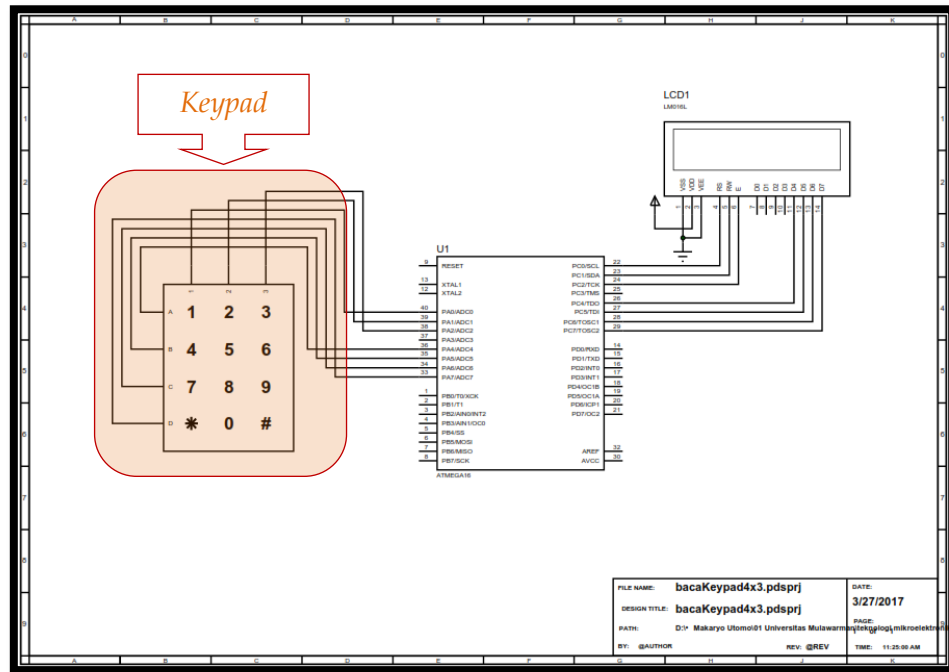
**Asisten Percobaan :**

**NIM :**

**Tanggal Percobaan :**

### Lab 3. Manipulasi Data I/O (Part I)

Latihan menggunakan *Matrix Push Buttons (keypad)* berdimensi 4x3 dan Display LM016.

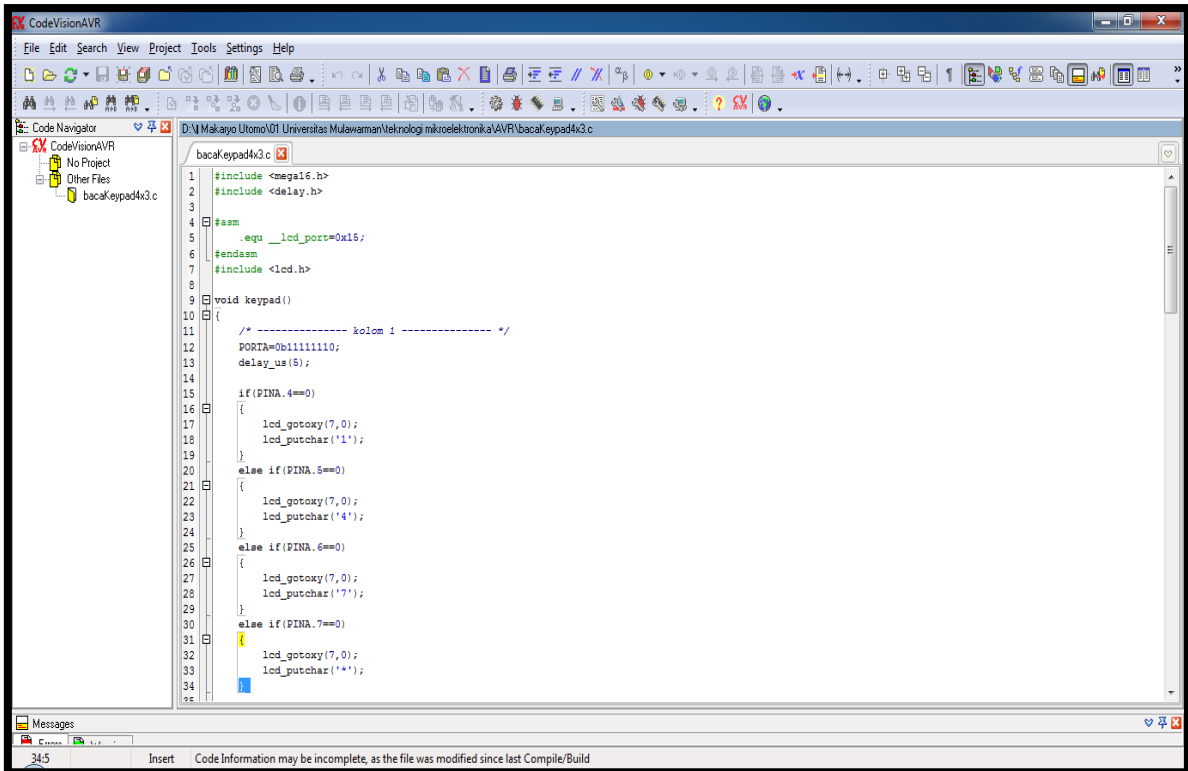


Gambar 7. Rangkaian skematik kontrol keypad

Langkah mempersiapkan Alat dan Bahan, yakni:

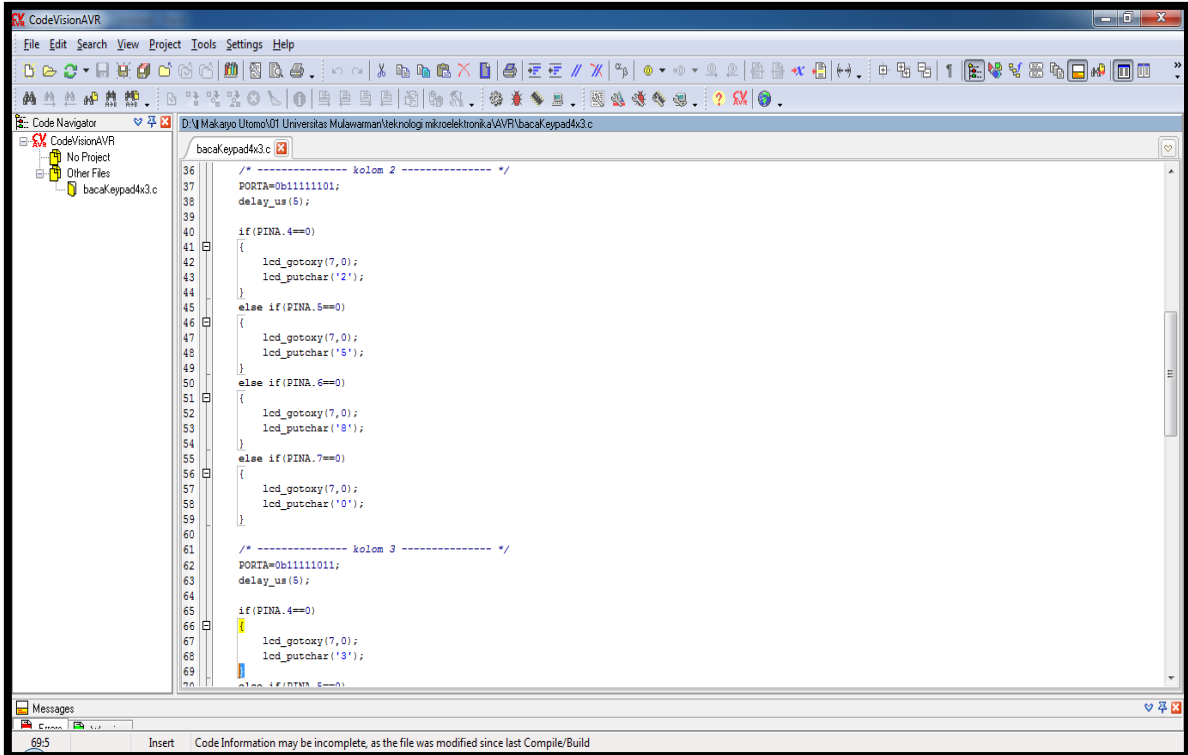
1. Laptop yang telah ter-install aplikasi *Proteus 8 Professional* dan *Code Vision AVR*!
2. Merancang Rangkaian Skematik seperti terlihat pada Gambar 7, menggunakan aplikasi *Proteus 8 Professional* pada laptop masing-masing!
3. Menuliskan "*Coding*" seperti terlihat pada Gambar 8 ke dalam aplikasi *Code Vision AVR*!
4. Meng-*Upload* "*Coding*" yang telah dituliskan, ke dalam desain rancangan Rangkaian Skematik kalian masing-masing!
5. *Screenshot* atau *Capture* hasilnya! seperti tampak pada Gambar 9.
6. Meng-*Upload* file (*fullscreen!*) dalam bentuk format **\*.pdf** ke Google Classroom, antara lain:
  - a. Rangkaian Skematik
  - b. *Coding*
  - c. Hasil *Capture*

Adapun Coding yang digunakan dalam Modul Lab 3 (kontrol keypad) adalah:



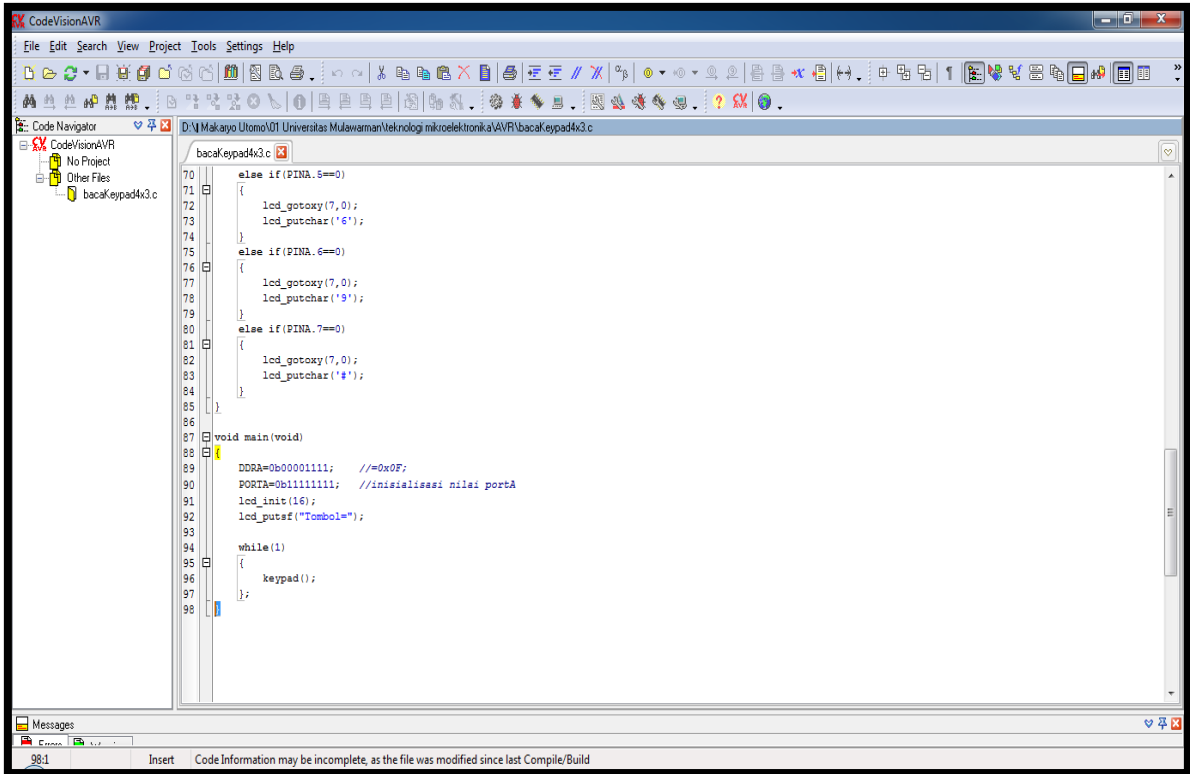
```
1 #include <mega16.h>
2 #include <delay.h>
3
4 #asm
5 .equ __lcd_port=0x15;
6 #endasm
7 #include <lcd.h>
8
9 void keypad()
10 {
11     /* ----- kolom 1 ----- */
12     PORTA=0b11111110;
13     delay_us(5);
14
15     if(PINA.4==0)
16     {
17         lcd_gotoxy(7,0);
18         lcd_putchar('1');
19     }
20     else if(PINA.5==0)
21     {
22         lcd_gotoxy(7,0);
23         lcd_putchar('4');
24     }
25     else if(PINA.6==0)
26     {
27         lcd_gotoxy(7,0);
28         lcd_putchar('7');
29     }
30     else if(PINA.7==0)
31     {
32         lcd_gotoxy(7,0);
33         lcd_putchar('*');
34     }
35 }
```

(a)



```
36     /* ----- kolom 2 ----- */
37     PORTA=0b11111011;
38     delay_us(5);
39
40     if(PINA.4==0)
41     {
42         lcd_gotoxy(7,0);
43         lcd_putchar('2');
44     }
45     else if(PINA.5==0)
46     {
47         lcd_gotoxy(7,0);
48         lcd_putchar('5');
49     }
50     else if(PINA.6==0)
51     {
52         lcd_gotoxy(7,0);
53         lcd_putchar('8');
54     }
55     else if(PINA.7==0)
56     {
57         lcd_gotoxy(7,0);
58         lcd_putchar('0');
59     }
60
61     /* ----- kolom 3 ----- */
62     PORTA=0b11111011;
63     delay_us(5);
64
65     if(PINA.4==0)
66     {
67         lcd_gotoxy(7,0);
68         lcd_putchar('3');
69     }
70     else if(PINA.5==0)
```

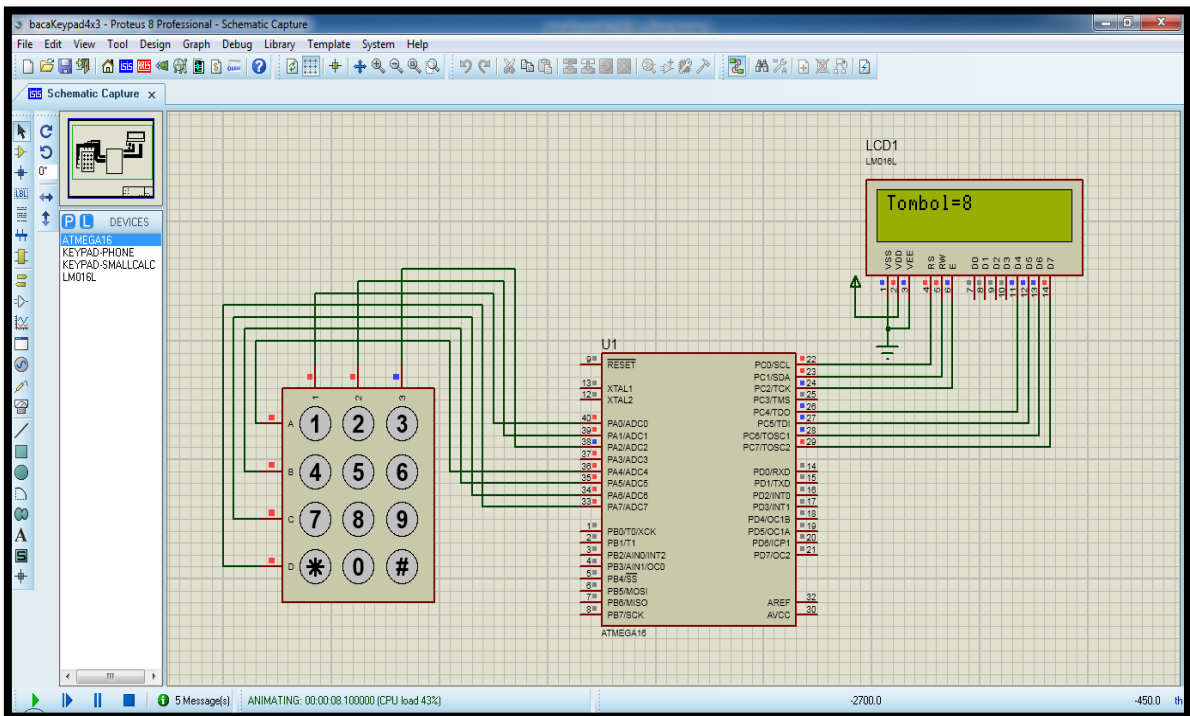
(b)



(c)

Gambar 8 (a) (b) (c). Coding untuk menuliskan instruksi pada Code Vision AVR

Hasil Capture Percobaan Lab 3 (kontrol keypad), yakni:



Gambar 9. Contoh hasil percobaan Lab 3

**LAB 3.**

**Manipulasi Data I/O (Part II)**

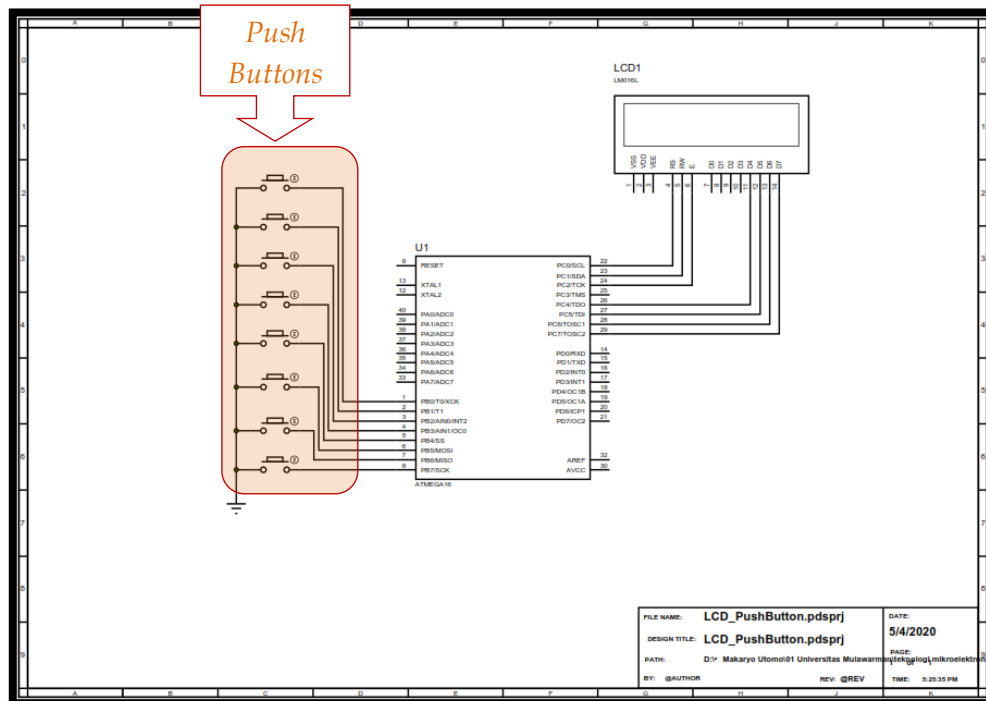
**Asisten Percobaan :**

**NIM :**

**Tanggal Percobaan :**

### Lab 3. Manipulasi Data I/O (Part II)

Latihan menggunakan *Push Buttons* 8 unit sebagai input, untuk menampilkan karakter bertipe data *char* pada Display LM016.

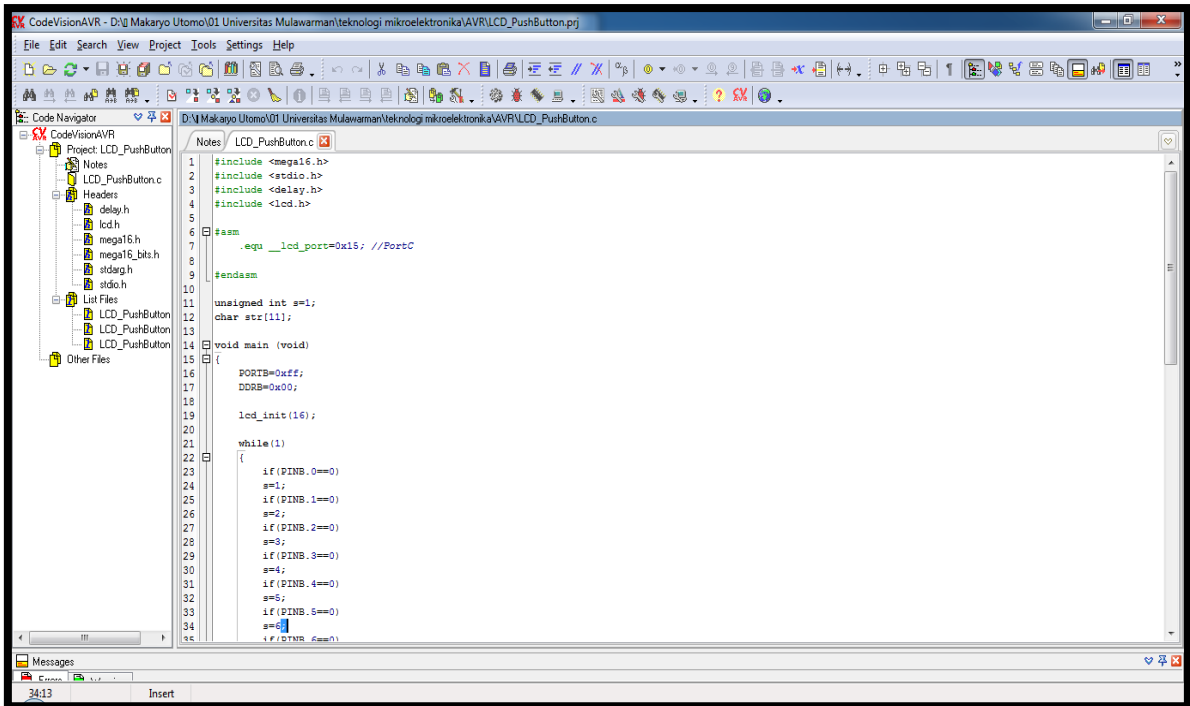


Gambar 10. Rangkaian skematik kontrol *push buttons*

Langkah mempersiapkan Alat dan Bahan, yakni:

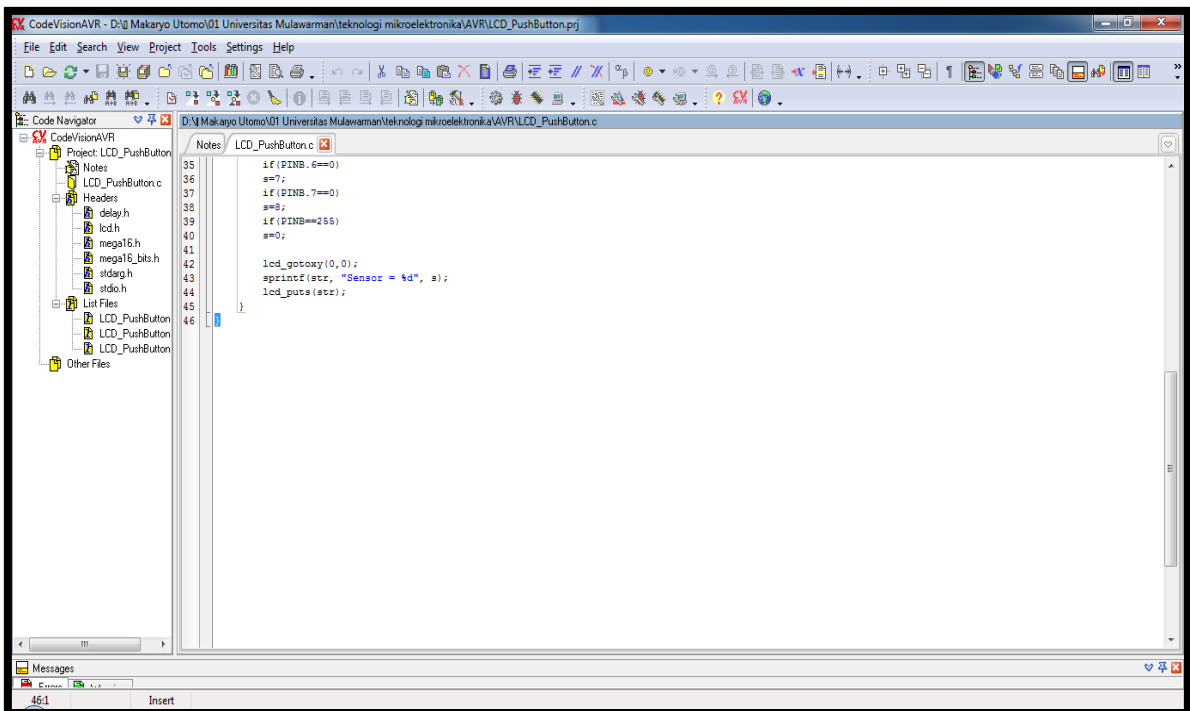
1. Laptop yang telah ter-*install* aplikasi *Proteus 8 Professional* dan *Code Vision AVR*!
2. Merancang Rangkaian Skematik seperti terlihat pada Gambar 10, menggunakan aplikasi *Proteus 8 Professional* pada laptop masing-masing!
3. Menuliskan "*Coding*" seperti terlihat pada Gambar 11 ke dalam aplikasi *Code Vision AVR*!
4. Meng-*Upload* "*Coding*" yang telah ditulis, ke dalam desain rancangan Rangkaian Skematik kalian masing-masing!
5. *Screenshot* atau *Capture* hasilnya! seperti tampak pada Gambar 12.
6. Meng-*Upload* file (*fullscreen!*) dalam bentuk format **\*.pdf** ke Google Classroom, antara lain:
  - a. Rangkaian Skematik
  - b. *Coding*
  - c. Hasil *Capture*

Adapun *Coding* yang digunakan dalam Modul Lab 3 (kontrol *push buttons*) adalah:



```
1 #include <mega16.h>
2 #include <stdio.h>
3 #include <delay.h>
4 #include <lcd.h>
5
6 #asm
7     .equ __lcd_port=0x15; //PortC
8
9 #endasm
10
11 unsigned int s=1;
12 char str[11];
13
14 void main (void)
15 {
16     PORTB=0xff;
17     DDRB=0x00;
18
19     lcd_init(16);
20
21     while(1)
22     {
23         if (PINSB.0==0)
24             s=1;
25         if (PINSB.1==0)
26             s=2;
27         if (PINSB.2==0)
28             s=3;
29         if (PINSB.3==0)
30             s=4;
31         if (PINSB.4==0)
32             s=5;
33         if (PINSB.5==0)
34             s=6;
35         if (PINSB.6==0)
```

(a)

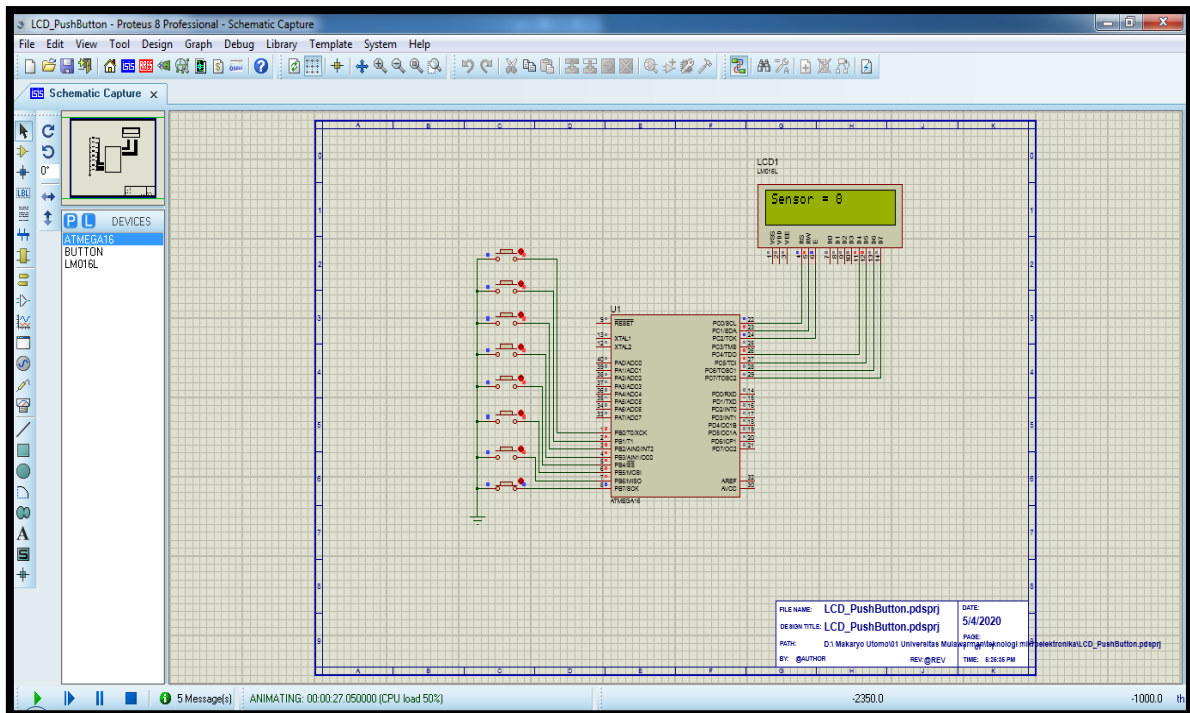


```
35         if (PINSB.6==0)
36             s=7;
37         if (PINSB.7==0)
38             s=8;
39         if (PINSB==255)
40             s=0;
41
42         lcd_gotoxy(0,0);
43         sprintf(str, "Sensor = %d", s);
44         lcd_puts(str);
45     }
46 }
```

(b)

**Gambar 11 (a) (b). Coding pada Code Vision AVR**

Hasil *Capture* Percobaan Lab 3 (kontrol *push buttons*), yakni:



**Gambar 12. Contoh hasil percobaan Lab 3**



**LAB 4.**

**Manipulasi Data I/O (Part III)**

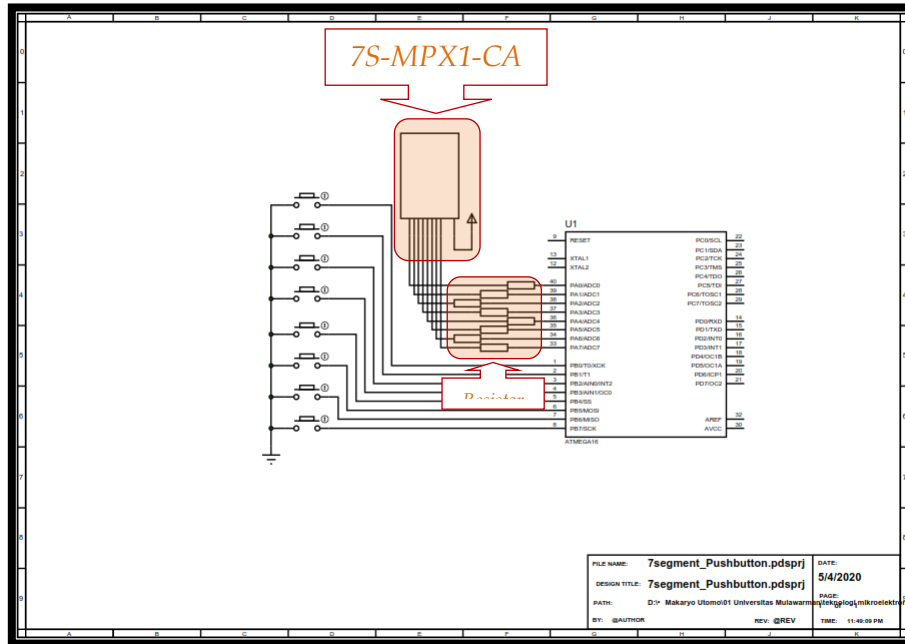
**Asisten Percobaan :**

**NIM :**

**Tanggal Percobaan :**

## Lab 4. Manipulasi Data I/O (Part I)

Latihan menggunakan *Push Buttons* 8 unit sebagai input, untuk menampilkan angka atau bilangan ke dalam Display *seven segment 7S-MPX1-CA*.

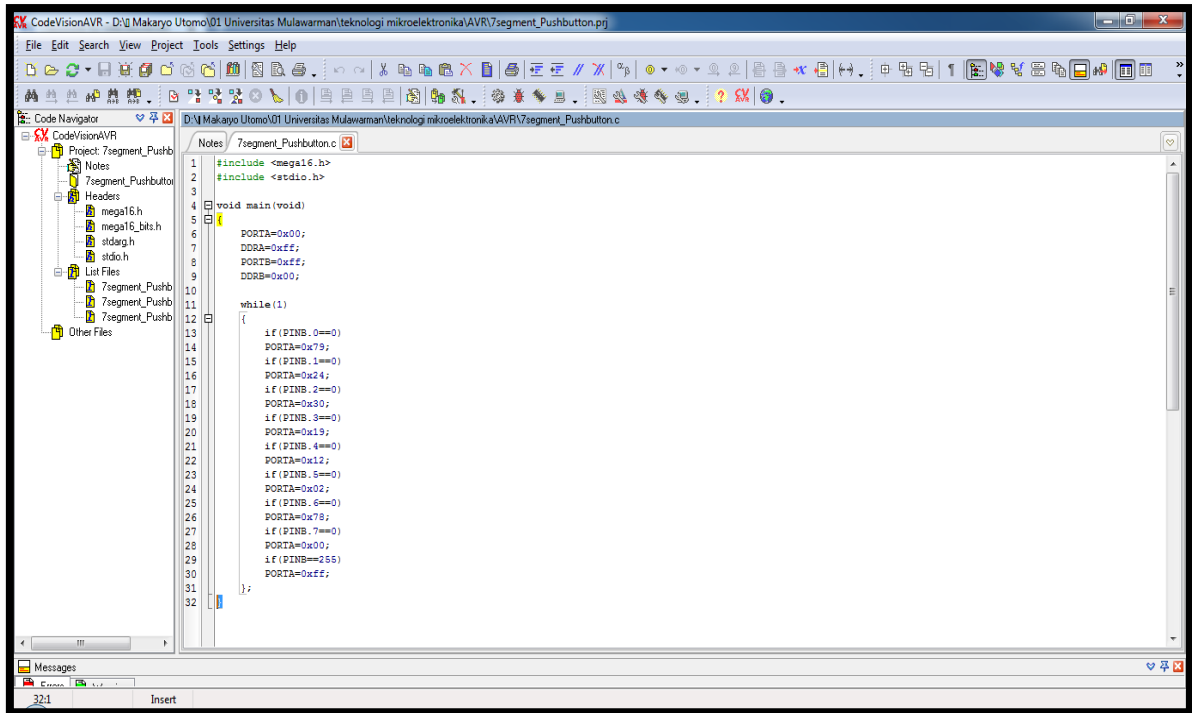


Gambar 13. Rangkaian skematik kontrol *push buttons*

Langkah mempersiapkan Alat dan Bahan, yakni:

1. Laptop yang telah ter-*install* aplikasi *Proteus 8 Professional* dan *Code Vision AVR*!
2. Merancang Rangkaian Skematik seperti terlihat pada Gambar 13, menggunakan aplikasi *Proteus 8 Professional* pada laptop masing-masing!
3. Menuliskan "*Coding*" seperti terlihat pada Gambar 14 ke dalam aplikasi *Code Vision AVR*!
4. Meng-*Upload* "*Coding*" yang telah ditulis, ke dalam desain rancangan Rangkaian Skematik kalian masing-masing!
5. *Screenshot* atau *Capture* hasilnya! seperti tampak pada Gambar 15.
6. Meng-*Upload* file (*fullscreen!*) dalam bentuk format **\*.pdf** ke Google Classroom, antara lain:
  - a. Rangkaian Skematik
  - b. *Coding*
  - c. Hasil *Capture*

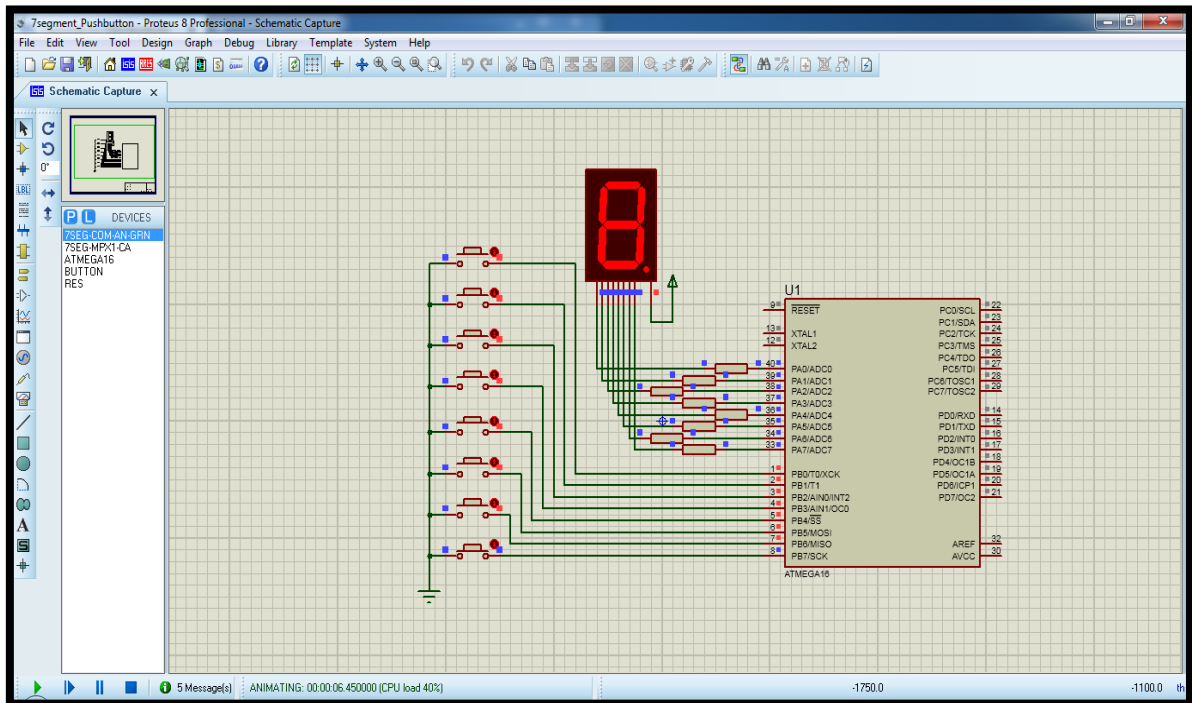
Adapun *Coding* yang digunakan dalam Modul Lab 4 (kontrol *push buttons*) adalah:



```
1 #include <mega16.h>
2 #include <stdio.h>
3
4 void main(void)
5 {
6     PORTA=0x00;
7     DDRB=0xff;
8     PORTB=0xff;
9     DDRB=0x00;
10
11     while(1)
12     {
13         if (PINB_0==0)
14             PORTA=0x79;
15         if (PINB_1==0)
16             PORTA=0x24;
17         if (PINB_2==0)
18             PORTA=0x30;
19         if (PINB_3==0)
20             PORTA=0x19;
21         if (PINB_4==0)
22             PORTA=0x12;
23         if (PINB_5==0)
24             PORTA=0x02;
25         if (PINB_6==0)
26             PORTA=0x78;
27         if (PINB_7==0)
28             PORTA=0x00;
29         if (PINB_255)
30             PORTA=0xff;
31     }
32 }
```

Gambar 14. *Coding* pada Code Vision AVR

Hasil *Capture* Percobaan Lab 4 (kontrol *push buttons*), yakni:



Gambar 15. Contoh hasil percobaan Lab 4

**LAB 4.**

**Manipulasi Data I/O (Part IV)**

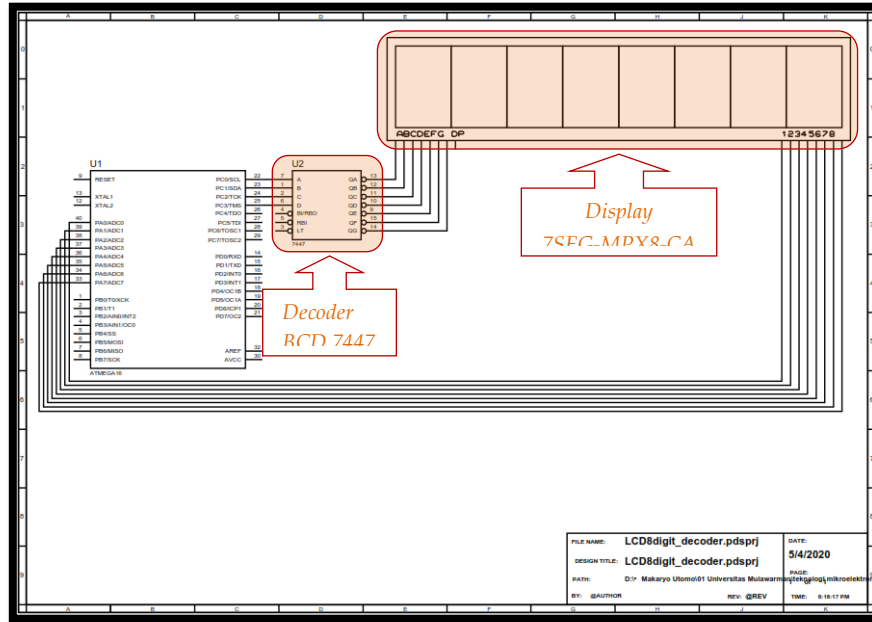
**Asisten Percobaan :**

**NIM :**

**Tanggal Percobaan :**

#### Lab 4. Manipulasi Data I/O (Part IV)

Latihan mengonversi bilangan digital ke bentuk bilangan *Binary Coded Decimal* (BCD) menggunakan 1 unit *decoder* BCD7447, dan menampilkannya ke dalam Display *seven segment* 7SEG-MPX8-CA.

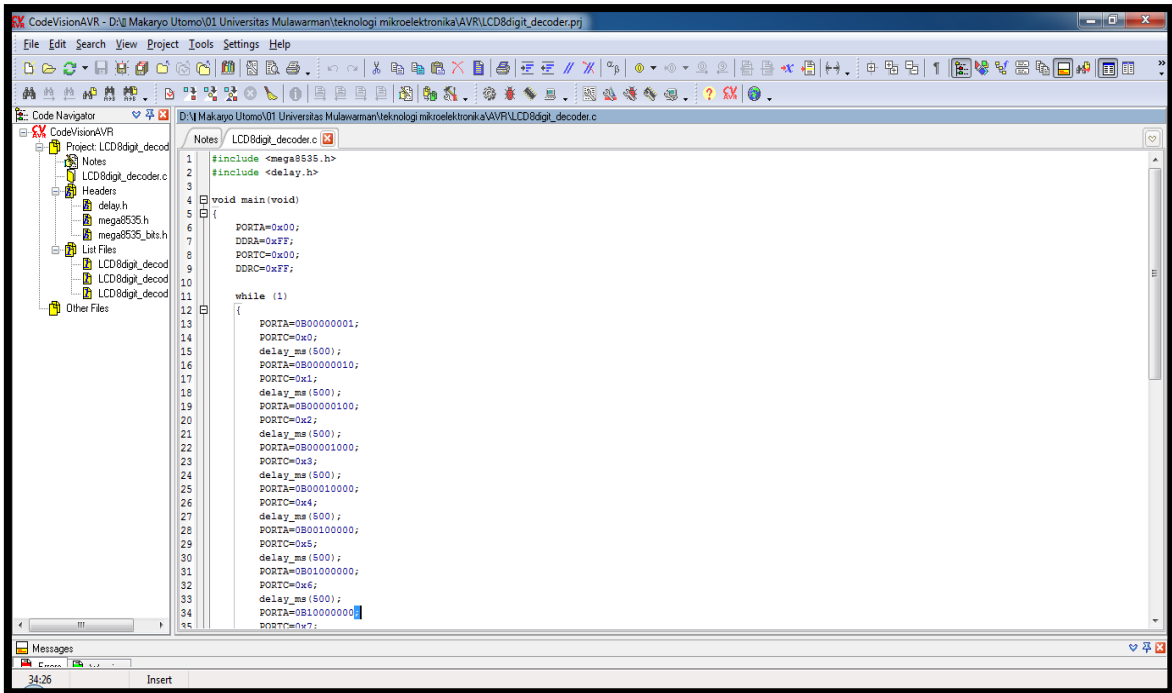


Gambar 16. Rangkaian skematik untuk menampilkan bilangan BCD

Langkah mempersiapkan Alat dan Bahan, yakni:

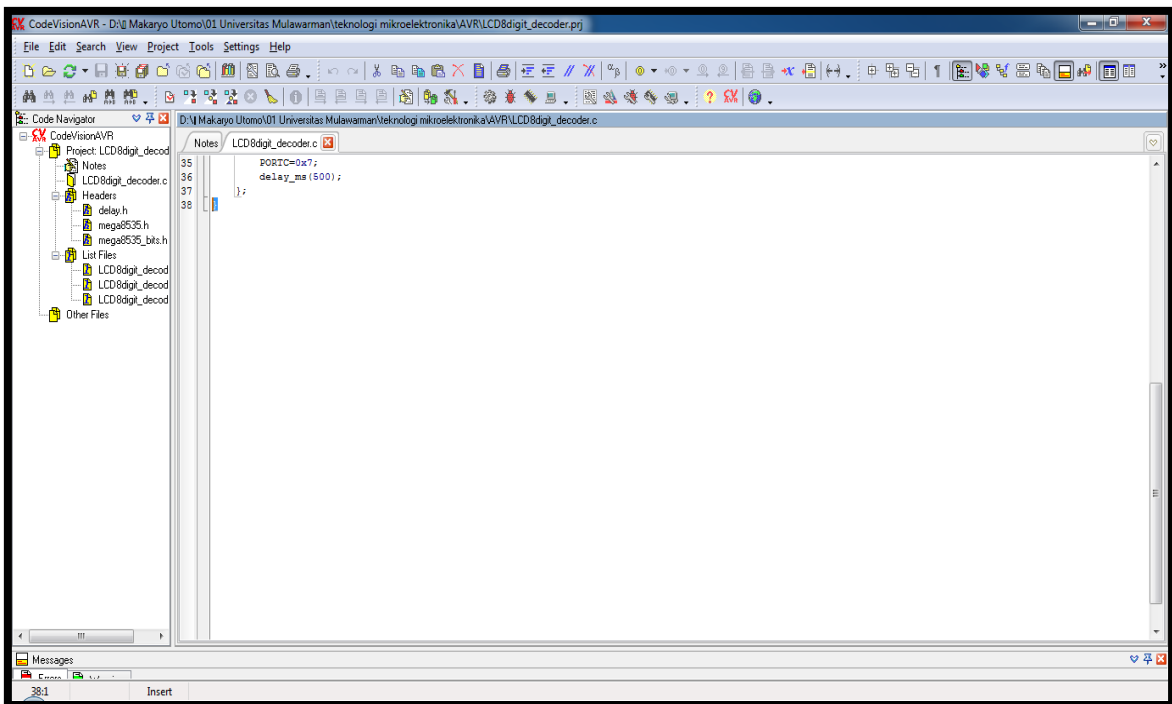
1. Laptop yang telah ter-*install* aplikasi *Proteus 8 Professional* dan *Code Vision AVR*!
2. Merancang Rangkaian Skematik seperti terlihat pada Gambar 16, menggunakan aplikasi *Proteus 8 Professional* pada laptop masing-masing!
3. Menuliskan "*Coding*" seperti terlihat pada Gambar 17 ke dalam aplikasi *Code Vision AVR*!
4. Meng-*Upload* "*Coding*" yang telah ditulis, ke dalam desain rancangan Rangkaian Skematik kalian masing-masing!
5. *Screenshot* atau *Capture* hasilnya! seperti tampak pada Gambar 18.
6. Meng-*Upload* file (*fullscreen!*) dalam bentuk format **\*.pdf** ke Google Classroom, antara lain:
  - a. Rangkaian Skematik
  - b. *Coding*
  - c. Hasil *Capture*

Adapun *Coding* yang digunakan dalam Modul Lab 4 (*display* bilangan/angka) adalah:



```
1 #include <mega8535.h>
2 #include <delay.h>
3
4 void main(void)
5 {
6     PORTA=0x00;
7     DDRA=0xFF;
8     PORTC=0x00;
9     DDRC=0xFF;
10
11     while (1)
12     {
13         PORTA=0B00000001;
14         PORTC=0x0;
15         delay_ms(500);
16         PORTA=0B00000010;
17         PORTC=0x1;
18         delay_ms(500);
19         PORTA=0B00000100;
20         PORTC=0x2;
21         delay_ms(500);
22         PORTA=0B00001000;
23         PORTC=0x3;
24         delay_ms(500);
25         PORTA=0B00010000;
26         PORTC=0x4;
27         delay_ms(500);
28         PORTA=0B00100000;
29         PORTC=0x5;
30         delay_ms(500);
31         PORTA=0B01000000;
32         PORTC=0x6;
33         delay_ms(500);
34         PORTA=0B10000000;
35         PORTC=0x7;
```

(a)

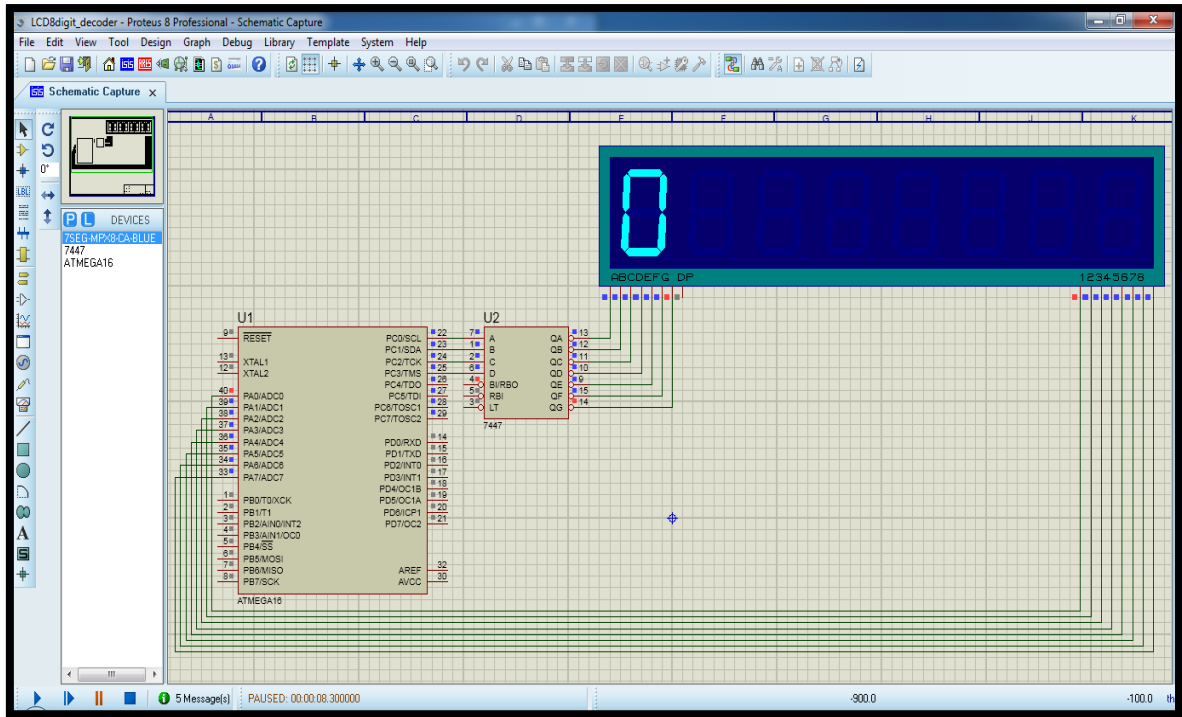


```
35         PORTC=0x7;
36         delay_ms(500);
37     };
38 }
```

(b)

**Gambar 17 (a) (b). Coding pada Code Vision AVR**

Hasil Capture Percobaan Lab 4 (*display bilangan/angka*), yakni:



Gambar 18. Contoh hasil percobaan Lab 4

LAB 5.

**Manipulasi Fitur *Pulse Width*  
*Modulation* (PWM)**

Asisten Percobaan :

NIM :

Tanggal Percobaan :

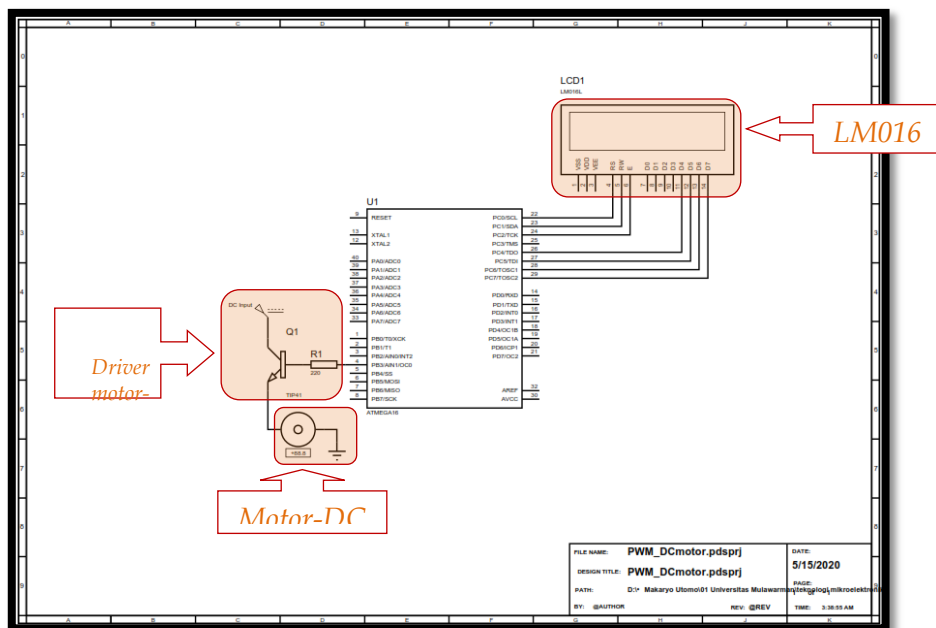


## Lab 5. Manipulasi Sinyal PWM

Latihan menggunakan fitur pada mikrokontroler, khususnya PWM, untuk mengendalikan kecepatan motor-DC. Fitur PWM ini umumnya digunakan untuk mengendalikan motor-DC, Servo, intensitas terang-redup cahaya pada lampu *Light Emitting Diode* (LED), maupun aplikasi lain yang memanfaatkan *Timer/Counter* sebagai Pembangkit Gelombang (*Wave Generator*), yang dalam kasus ini adalah PWM.

Nilai Input PWM akan menentukan seberapa cepat putaran motor-DC. Semakin besar nilai Input PWM, maka semakin cepat pula putaran motor-DC yang digunakan, begitupun sebaliknya. Nilai Input PWM memiliki rentang antara nilai 0 s.d. 255 berdasarkan jumlah data yang digunakan, yaitu 8 bit data. Untuk mengendalikan fitur PWM, ada dua *register* yang berperan signifikan dalam mengontrol kecepatan motor-DC pada modul Lab 5 ini, antara lain:

- a. TCCR0; dan
- b. OCR0

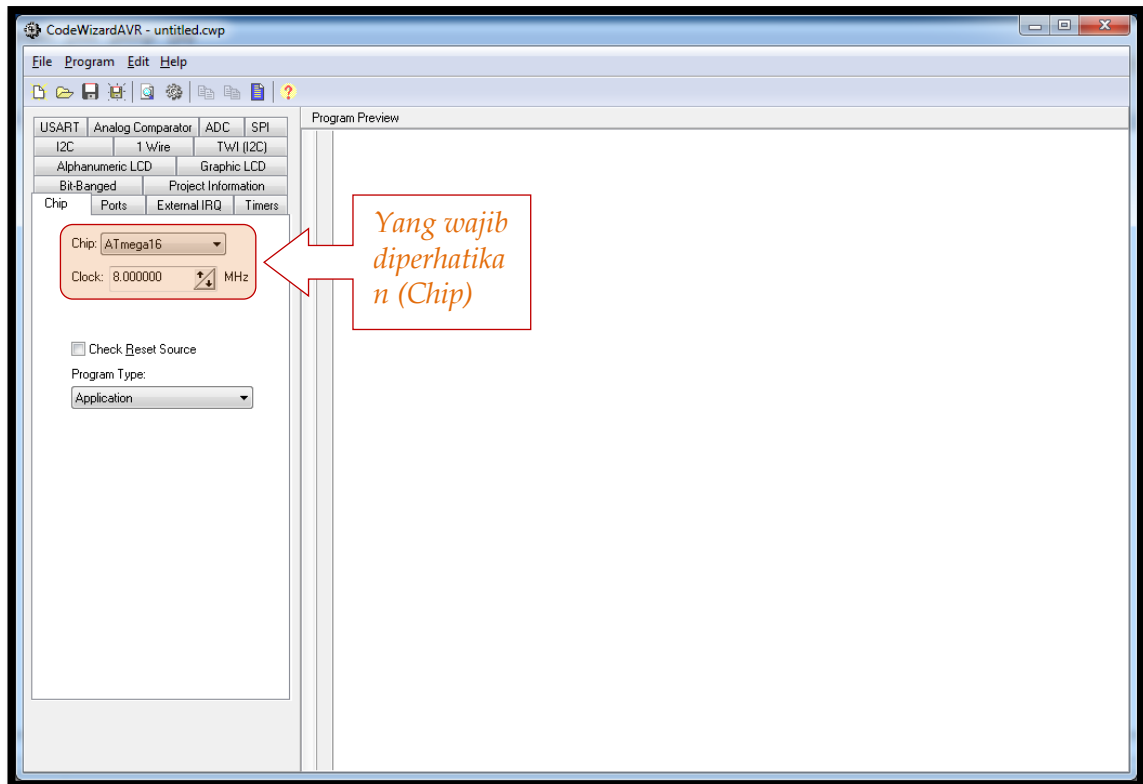


Gambar 19. Rangkaian skematik kontrol kecepatan pada motor-DC

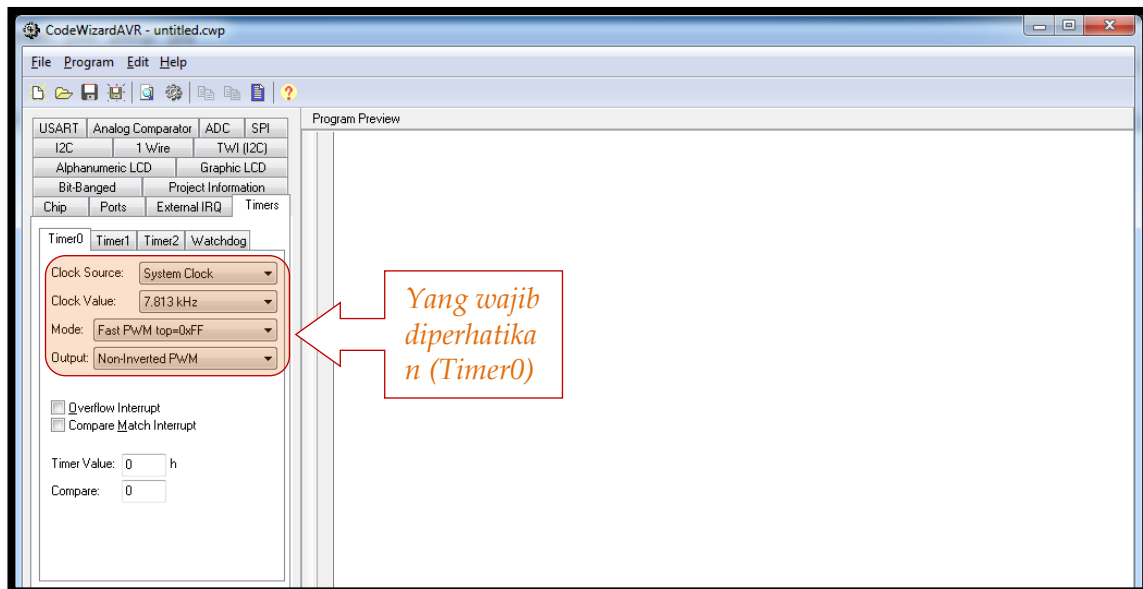
Langkah mempersiapkan Alat dan Bahan, yakni:

1. Laptop yang telah ter-*install* aplikasi *Proteus 8 Professional* dan *Code Vision AVR*!
2. Merancang Rangkaian Skematik seperti terlihat pada Gambar 19, menggunakan aplikasi *Proteus 8 Professional* pada laptop masing-masing!
3. Merancang *Project* awal menggunakan fasilitas *CodeWizardAVR*, sesuai dengan *setting* pada Gambar 20 (a), (b), dan (c)!

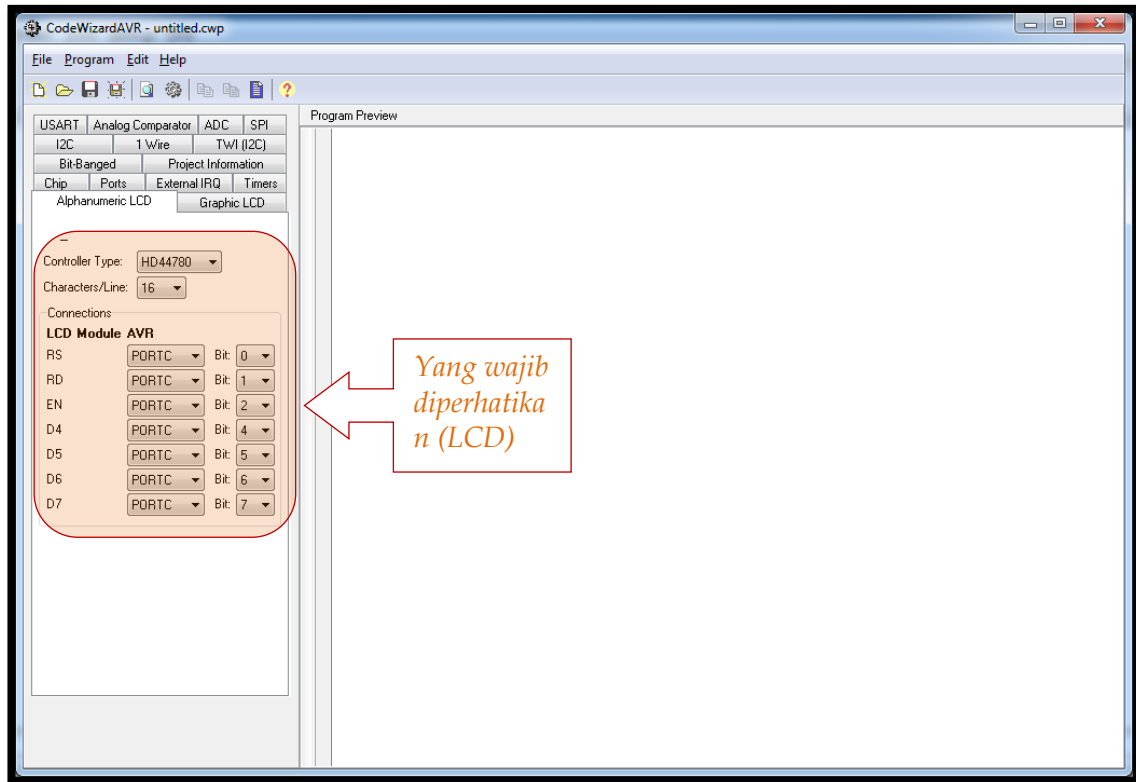
Setting konfigurasi CodeWizardAVR, yakni:



(a)



(b)

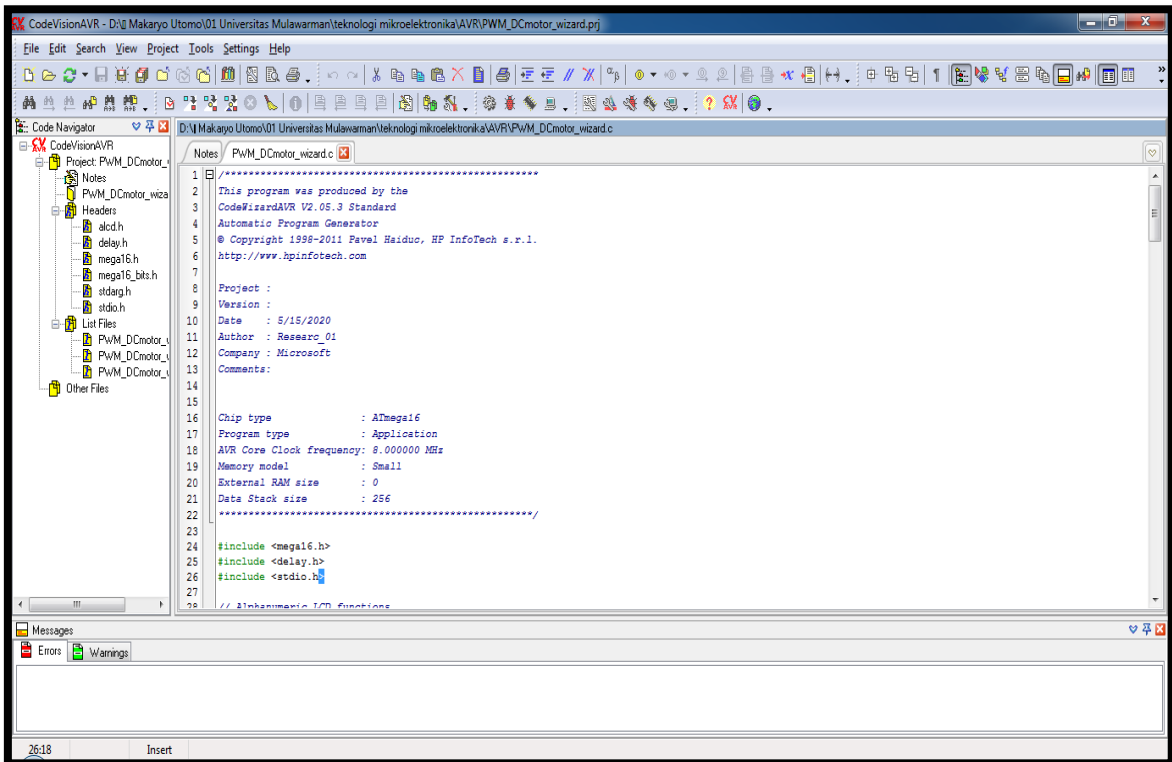


(b)

**Gambar 20 (a) (b) (c). Konfigurasi CodeWizardAVR**

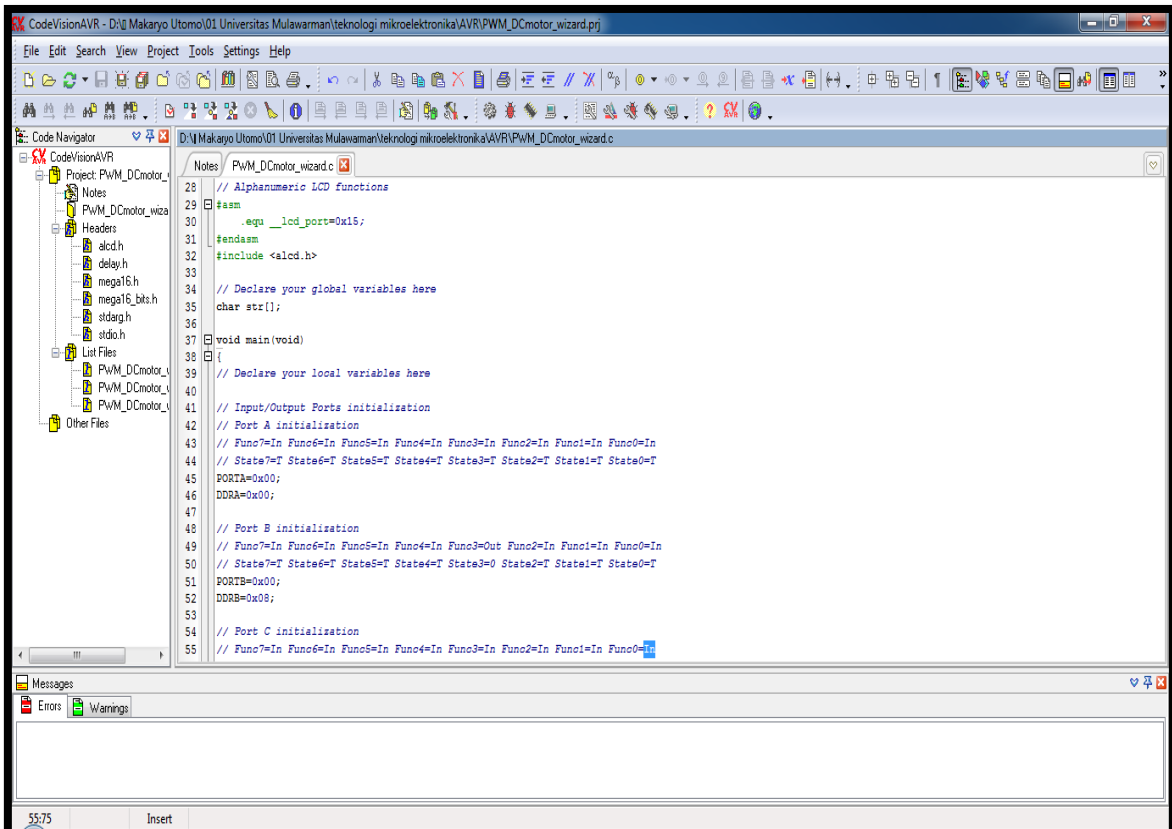
4. Menuliskan "Coding" seperti terlihat pada Gambar 21 ke dalam aplikasi *Code Vision AVR*!
5. Meng-*Upload* "Coding" yang telah ditulis, ke dalam desain rancangan Rangkaian Skematik kalian masing-masing!
6. *Screenshot* atau *Capture* hasilnya! seperti tampak pada Gambar 22.
7. Meng-*Upload* file (*fullscreen!*) dalam bentuk format **\*.pdf** ke Google Classroom, antara lain:
  - a. Rangkaian Skematik
  - b. *Coding*
  - c. Hasil *Capture*

Adapun *Coding* yang digunakan dalam Modul Lab 5 (kontrol kecepatan motor-DC) adalah:



```
1 //-----
2 This program was produced by the
3 CodeWizardAVR V2.05.3 Standard
4 Automatic Program Generator
5 © Copyright 1998-2011 Pavel Haiduc, HP InfoTech s.r.l.
6 http://www.hpinfotech.com
7
8 Project :
9 Version :
10 Date : 5/15/2020
11 Author : Researc_01
12 Company : Microsoft
13 Comments:
14
15
16 Chip type : ATmega16
17 Program type : Application
18 AVR Core Clock frequency: 8.000000 MHz
19 Memory model : Small
20 External RAM size : 0
21 Data Stack size : 256
22 //-----
23
24 #include <mega16.h>
25 #include <delay.h>
26 #include <stdio.h>
27
28 // Alphanumeric LCD functions
```

(a)



```
28 // Alphanumeric LCD functions
29 #asm
30 .equ __lcd_port=0x15;
31 #endasm
32 #include <lcd.h>
33
34 // Declare your global variables here
35 char str[];
36
37 void main(void)
38 {
39 // Declare your local variables here
40
41 // Input/Output Ports initialization
42 // Port A initialization
43 // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
44 // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
45 PORTA=0x00;
46 DDRA=0x00;
47
48 // Port B initialization
49 // Func7=In Func6=In Func5=In Func4=In Func3=Out Func2=In Func1=In Func0=In
50 // State7=T State6=T State5=T State4=T State3=0 State2=T State1=T State0=T
51 PORTB=0x00;
52 DDRB=0x08;
53
54 // Port C initialization
55 // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=
```

(b)

```

56 // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
57 PORTC=0x00;
58 DDRC=0x00;
59
60 // Port D initialization
61 // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
62 // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
63 PORTD=0x00;
64 DDRD=0x00;
65
66 // Timer/Counter 0 initialization
67 // Clock source: System Clock
68 // Clock value: 7.813 kHz
69 // Mode: Fast PWM top=0xFF
70 // OC0 output: Non-Inverted PWM
71 TCRC0=0x6D;
72 TCNT0=0x00;
73 OCR0=0x00;
74
75 // Timer/Counter 1 initialization
76 // Clock source: System Clock
77 // Clock value: Timer1 Stopped
78 // Mode: Normal top=0xFFFF
79 // OCA1A output: Discon.
80 // OCA1B output: Discon.
81 // Noise Canceler: Off
82 // Input Capture on Falling Edge
83 // Timer1 Overflow Interrupt: Off

```

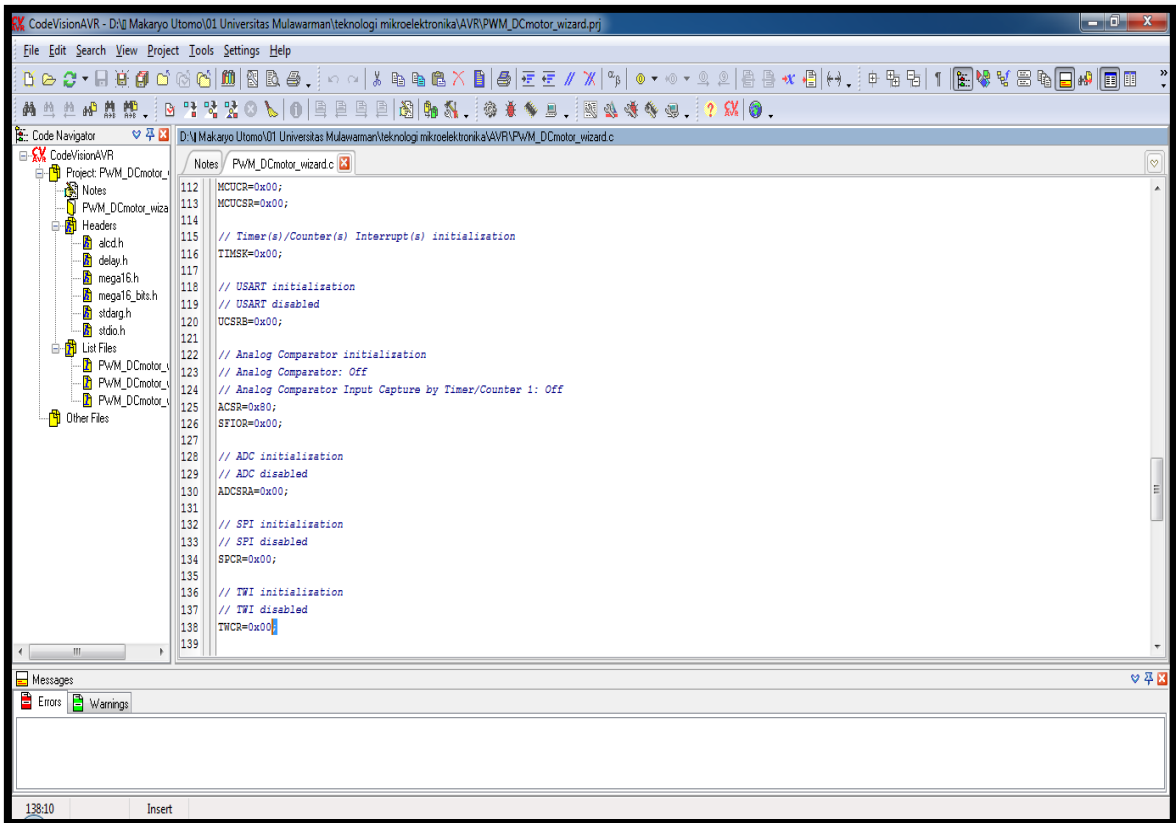
(c)

```

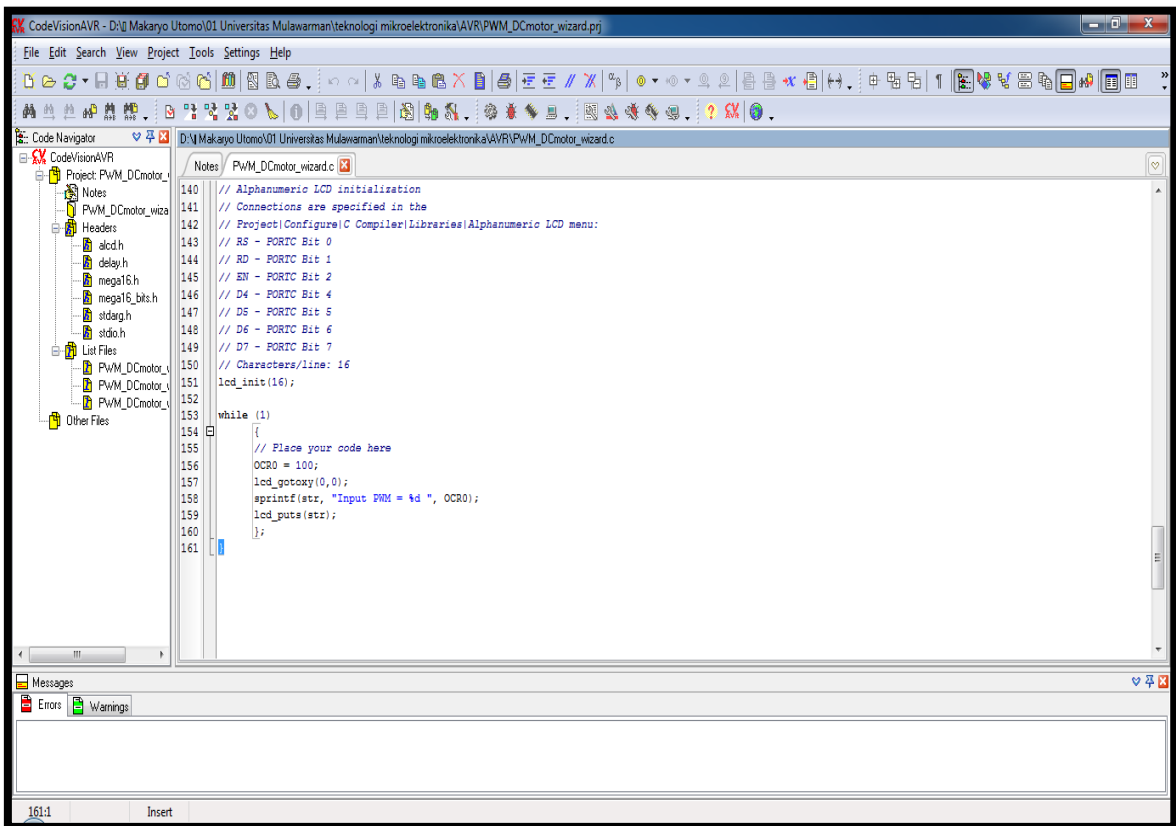
84 // Input Capture Interrupt: Off
85 // Compare A Match Interrupt: Off
86 // Compare B Match Interrupt: Off
87 TCRC1A=0x00;
88 TCRC1B=0x00;
89 TCNT1H=0x00;
90 TCNT1L=0x00;
91 ICR1H=0x00;
92 ICR1L=0x00;
93 OCR1A=0x00;
94 OCR1AL=0x00;
95 OCR1BH=0x00;
96 OCR1BL=0x00;
97
98 // Timer/Counter 2 initialization
99 // Clock source: System Clock
100 // Clock value: Timer2 Stopped
101 // Mode: Normal top=0xFF
102 // OC2 output: Disconnected
103 ASSR=0x00;
104 TCCR2=0x00;
105 TCNT2=0x00;
106 OCR2=0x00;
107
108 // External Interrupt(s) initialization
109 // INT0: Off
110 // INT1: Off
111 // INT2: Off

```

(d)



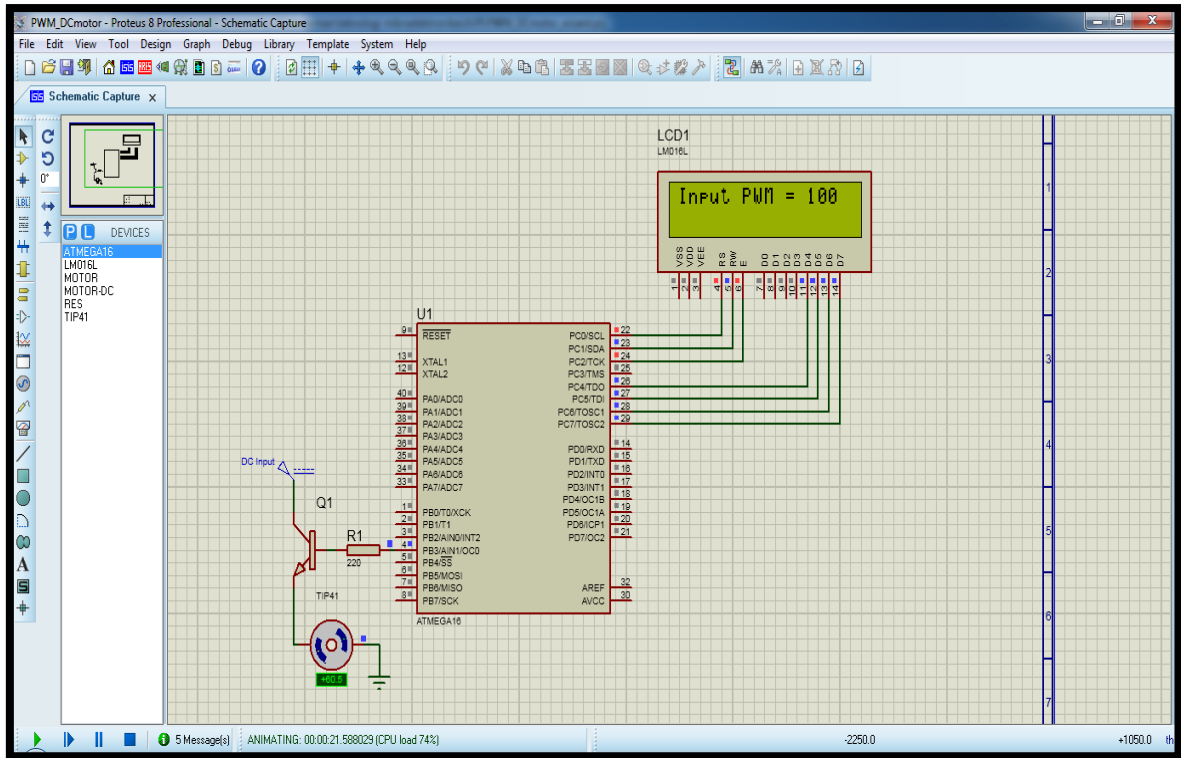
(e)



(f)

**Gambar 21. (a) (b) (c) (d) (e) (f). Coding pada Code Vision AVR**

Hasil *Capture* Percobaan Lab 5, yakni:



**Gambar 22.** Contoh hasil percobaan Lab 5

**LAB 6.**

**Interaksi Data I/O dan *Serial  
Communication* (Serial Comm)**

**Asisten Percobaan :**

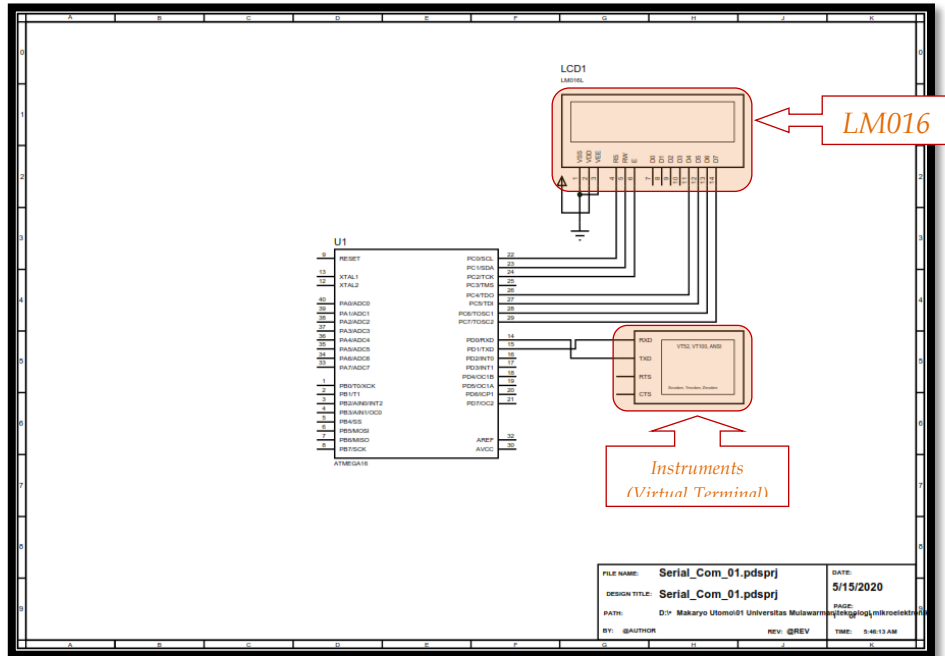
**NIM :**

**Tanggal Percobaan :**



## Lab 6. Komunikasi Serial Data I/O (Part IV)

Latihan berinteraksi antara operator dengan data I/O pada mikrokontroler melalui *Serial Communication (Serial Comm)*.

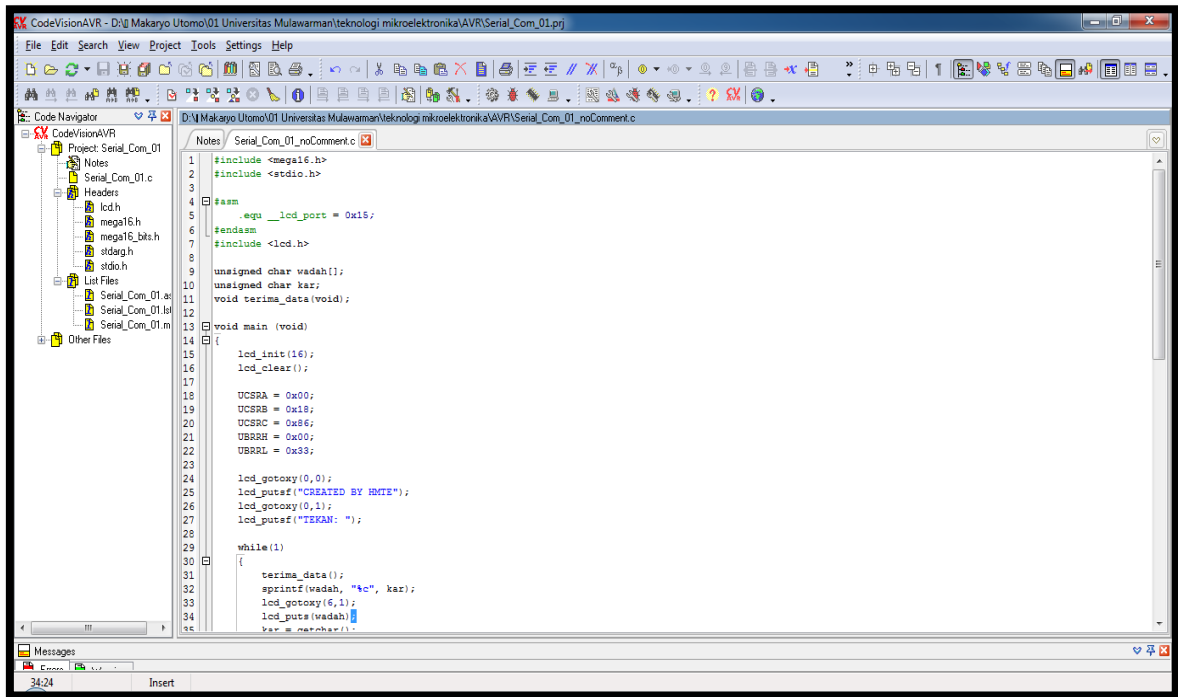


Gambar 23. Rangkaian skematik *Serial Comm*

Langkah mempersiapkan Alat dan Bahan, yakni:

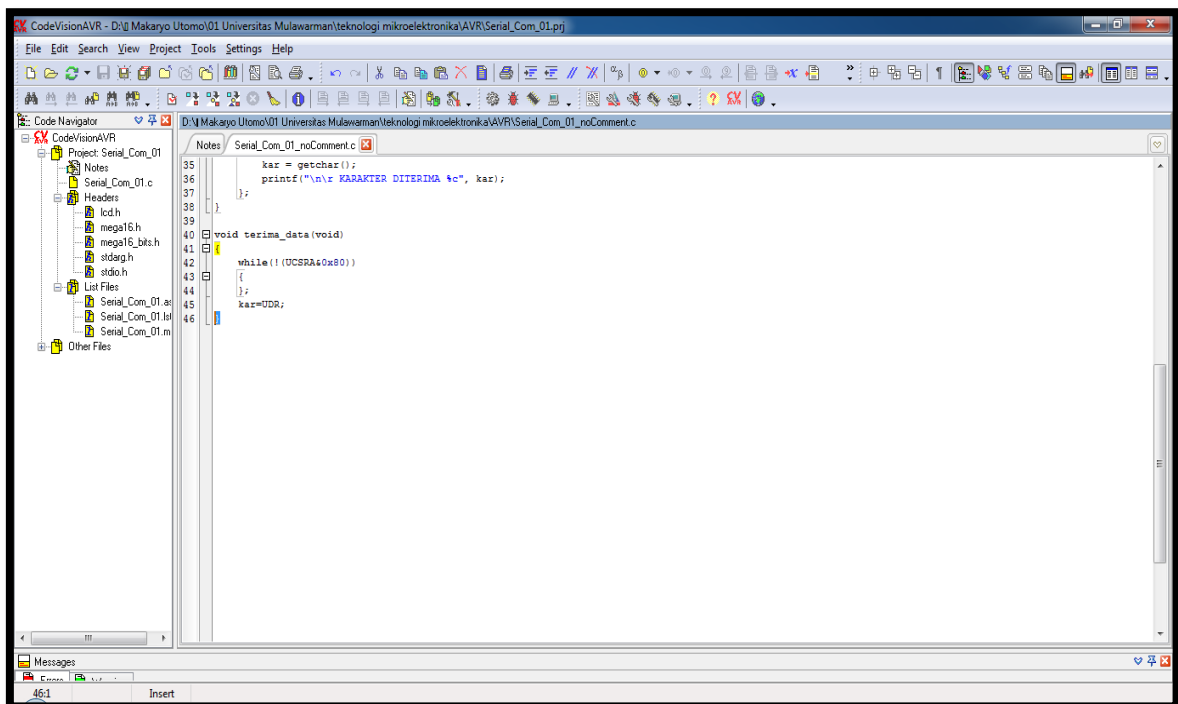
1. Laptop yang telah ter-*install* aplikasi *Proteus 8 Professional* dan *Code Vision AVR*!
2. Merancang Rangkaian Skematik seperti terlihat pada Gambar 23, menggunakan aplikasi *Proteus 8 Professional* pada laptop masing-masing!
3. Menuliskan "*Coding*" seperti terlihat pada Gambar 24 ke dalam aplikasi *Code Vision AVR*!
4. Meng-*Upload* "*Coding*" yang telah ditulis, ke dalam desain rancangan Rangkaian Skematik kalian masing-masing!
5. *Screenshot* atau *Capture* hasilnya! seperti tampak pada Gambar 25.
6. Meng-*Upload* file (*fullscreen!*) dalam bentuk format **\*.pdf** ke Google Classroom, antara lain:
  - a. Rangkaian Skematik
  - b. *Coding*
  - c. Hasil *Capture*

Adapun *Coding* yang digunakan dalam Modul Lab 6 (interaksi data I/O dengan *Serial Comm*) adalah:



```
1 #include <mega16.h>
2 #include <stdio.h>
3
4 #asm
5     .equ __lcd_port = 0x15;
6 #endasm
7 #include <lcd.h>
8
9 unsigned char wadah[];
10 unsigned char kar;
11 void terima_data(void);
12
13 void main(void)
14 {
15     lcd_init(16);
16     lcd_clear();
17
18     UCSRA = 0x00;
19     UCSRB = 0x18;
20     UCSRC = 0x86;
21     UBRRH = 0x00;
22     UBSRL = 0x33;
23
24     lcd_gotoxy(0,0);
25     lcd_putsf("CREATED BY RMTE");
26     lcd_gotoxy(0,1);
27     lcd_putsf("TEKSAH: ");
28
29     while(1)
30     {
31         terima_data();
32         sprintf(wadah, "%c", kar);
33         lcd_gotoxy(6,1);
34         lcd_puts(wadah);
35         kar = getch();
36     }
37 }
```

(a)

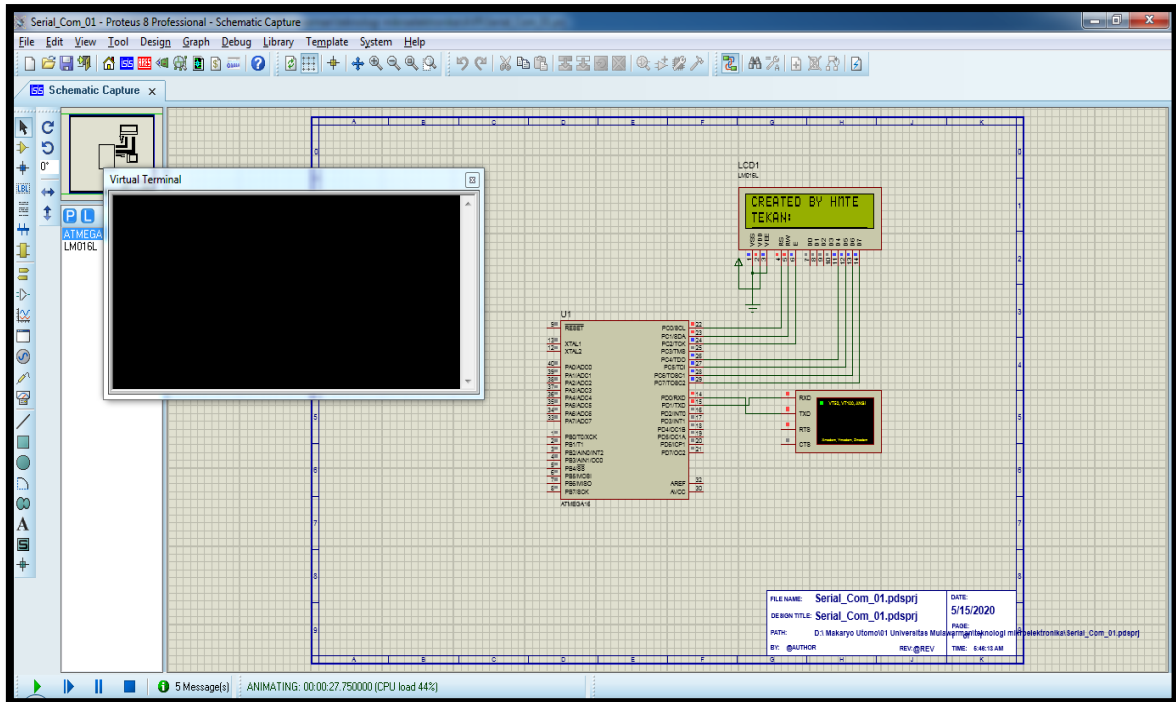


```
35     kar = getch();
36     printf("\n\t KARAKTER DITERIMA %c", kar);
37 }
38
39 void terima_data(void)
40 {
41     while(!(UCSRA&0x80))
42     {
43     };
44     kar=UDR;
45 }
```

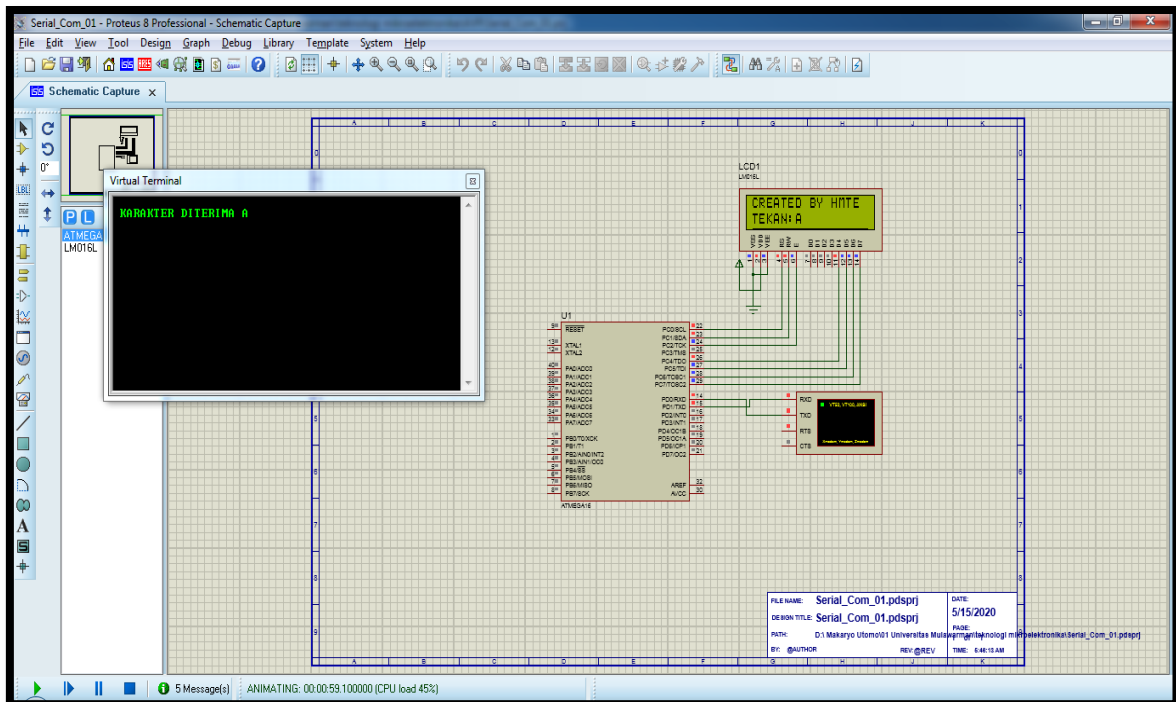
(b)

Gambar 24 (a) (b). *Coding* pada *Code Vision AVR*

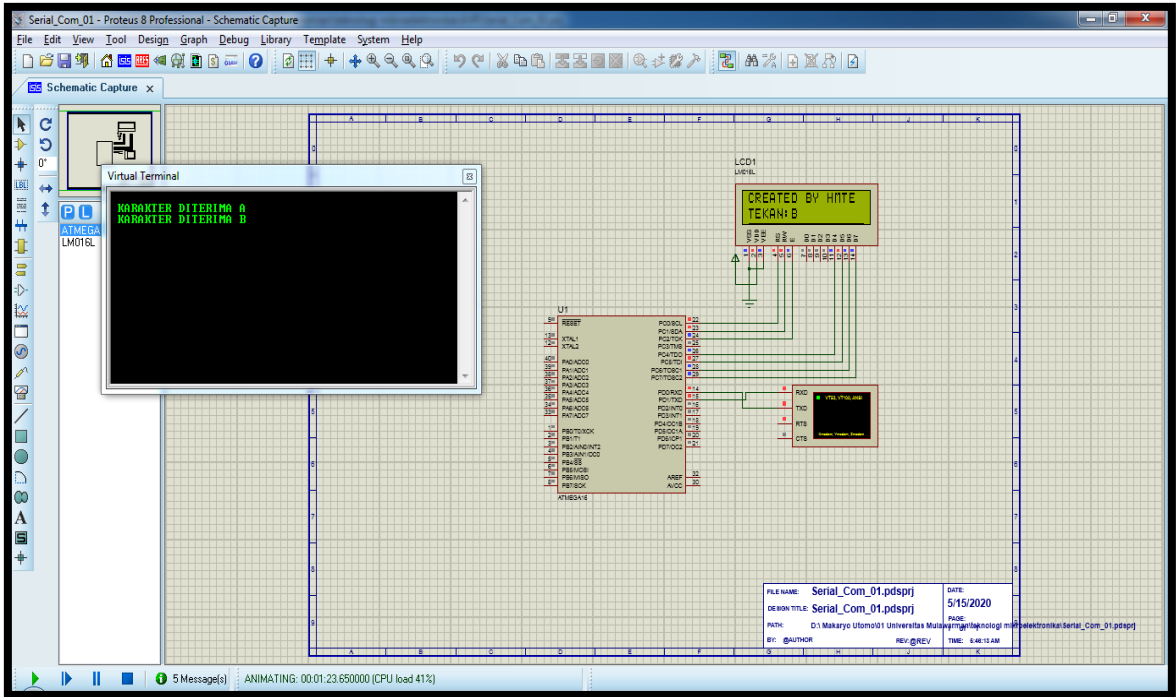
Hasil Capture Percobaan Lab 6, yakni:



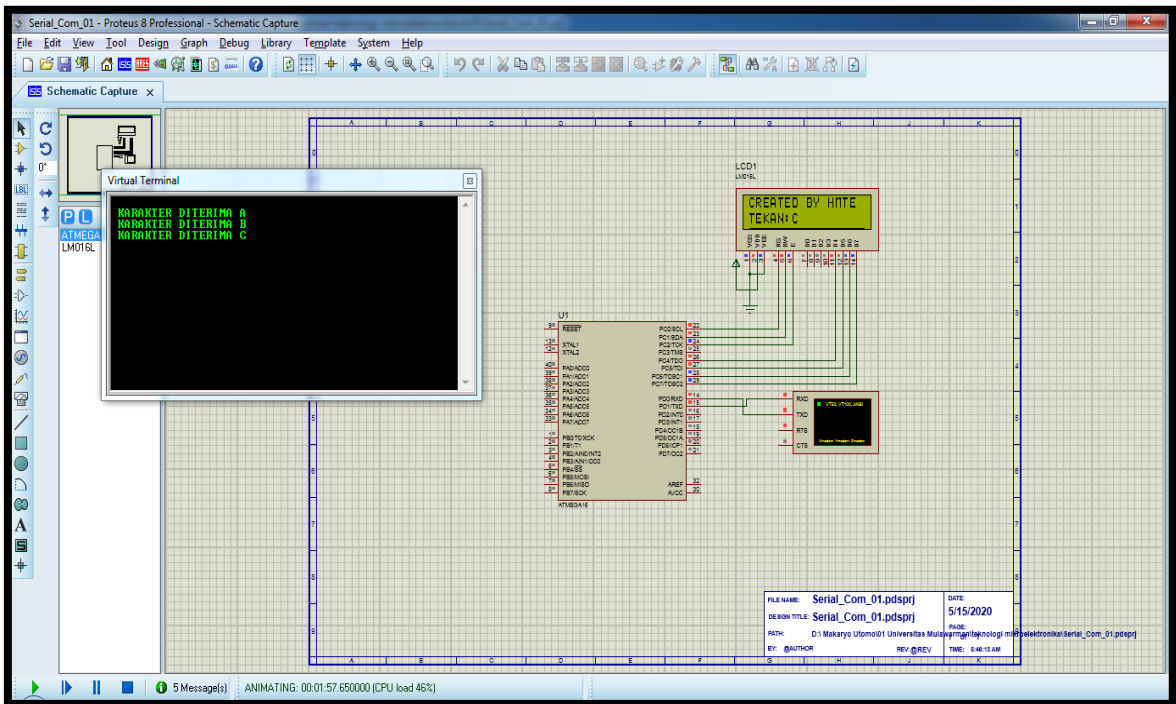
(a)



(b)



(c)



(d)

Gambar 25 (a) (b) (c) (d). Contoh hasil percobaan Lab 6

LAB 7.

**Akuisis Data Menggunakan Fitur**  
*Analog to Digital Converter (ADC)*

Asisten Percobaan :

NIM :

Tanggal Percobaan :

## Lab 7. Akuisi Data dengan *Analog to Digital Converter* (ADC)

Latihan mengakuisi data menggunakan fitur *Analog to Digital Converter* (ADC) dengan memanfaatkan perubahan intensitas cahaya pada sensor *Light Dependent Resistor* (LDR). Dalam proses untuk memperoleh data analog dengan ADC ini, ada dua *register* yang berperan sangat penting dalam proses akuisisi data, antara lain:

- a. ADCSRA; dan
- b. ADMUX

ADC merupakan suatu piranti yg berfungsi mengonversi sinyal analog menjadi sinyal digital. Mikrokontroler AVR jenis AtMega16 yang digunakan pada Modul ini sudah *include* ADC di dalam Chip mikrokontrolernya, sehingga tidak perlu lagi membuat rangkaian ADC tambahan. Adapun pin Analog ADC terdapat pada **PORTA.0** hingga **PORTA.7**.

Fitur ADC yang terdapat dalam mikrokontroler jenis AtMega16 yang digunakan dalam Modul Lab 7 ini menggunakan Resolusi 10 bit ( $2^{10} = 1024$ ), maka hasil konversi **Nilai ADC** dapat ditentukan berdasarkan Persamaan (1) di bawah.

$$ADC = \left( \frac{V_{in} \cdot 1024}{V_{ref}} \right) - 1 \text{ LSB}$$

**Persamaan (1)**

dimana, *ADC* adalah Nilai ADC hasil konversi,  $V_{in}$  (*Volt*) adalah Nilai tegangan luaran dari rangkaian sensor LDR yg masuk ke pin ADC mikrokontroler, dan  $V_{ref}$  (*Volt*) adalah tegangan referensi yang digunakan mikrokontroler.

Dari Persamaan (1), diperoleh,

apabila tegangan referensi  $V_{ref} = 5 \text{ Volt}$  dan tegangan input minimal yakni  $V_{in} = 0 \text{ Volt}$ , maka:

$$ADC = 0, ; \text{ dan,}$$

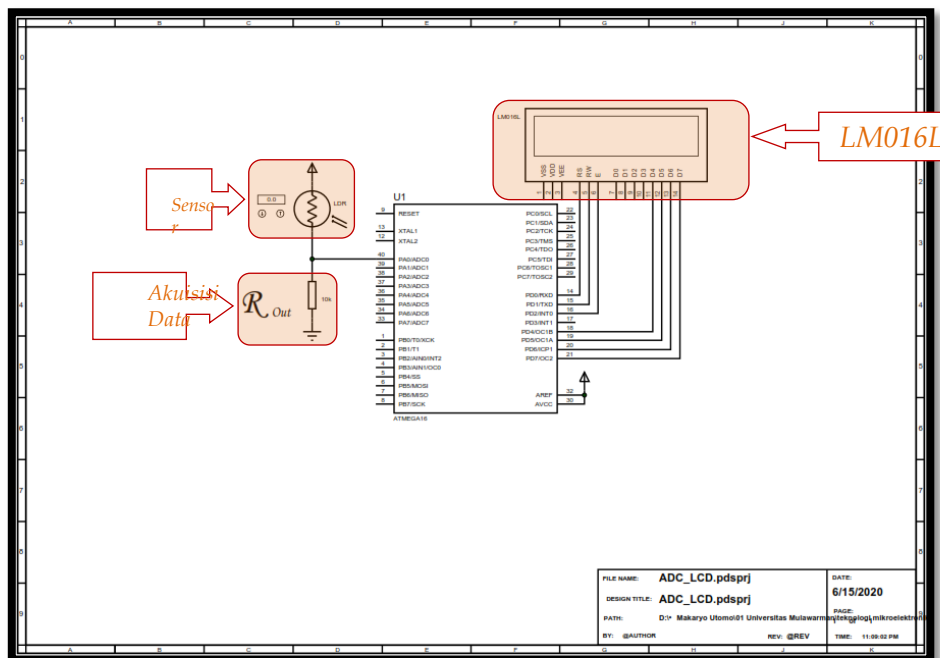
apabila tegangan referensi  $V_{ref} = 5 \text{ Volt}$  dan tegangan input maksimal yakni  $V_{in} = 5 \text{ Volt}$ , maka:

$$ADC = \left( \frac{V_{in} \cdot 1024}{V_{ref}} \right) - 1 \text{ LSB} = \left( \frac{5 \cdot 1024}{5} \right) - 1 \text{ LSB} = 1023.$$

Dari Persamaan (1), juga dapat dihitung besarnya nilai kenaikan yang diperoleh setiap perubahan 1 bit ADC, adalah sebesar  $V_{in} = 4,88 \text{ miliVolt}$ . Adapun *clock input* ADC yg digunakan dalam Modul Lab 7 ini adalah  $62.500 \text{ Hz}$  ( $62,5 \text{ kHz}$ ).

*Light Dependent Resistor* (LDR) / *Photo-resistor* / *Photo-conduction* / *Photo-cell* adalah komponen yang hambatannya berubah-ubah tergantung cahaya yang diterima dipermukaan sensor.

Umumnya LDR berkaki dua atau tiga, namun untuk percobaan Modul Lab 7 ini menggunakan jenis dua kaki, dan tidak memiliki polaritas sehingga dipasang bolak-balik sama saja, dimana sensor terhubung pada **Pin. 0** pada **PORTA** mikrokontroler AtMega16. Satuan untuk intensitas cahaya yg ditunjukkan pada LDR adalah *Illumination* (*Lux*).



**Gambar 26. Rangkaian skematik Akuisisi Data**

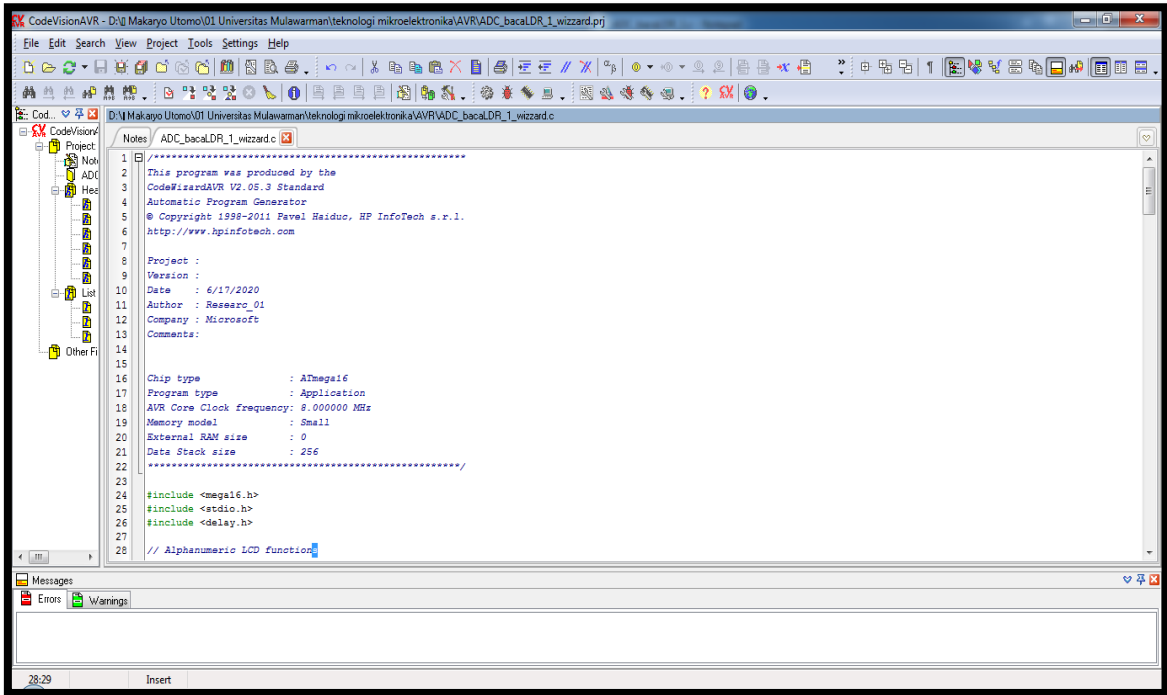
Langkah mempersiapkan Alat dan Bahan, yakni:

1. Laptop yang telah ter-*install* aplikasi *Proteus 8 Professional* dan *Code Vision AVR*!
2. Merancang Rangkaian Skematik seperti terlihat pada Gambar 26, menggunakan aplikasi *Proteus 8 Professional* pada laptop masing-masing!
3. Menuliskan "*Coding*" seperti terlihat pada Gambar 27 ke dalam aplikasi *Code Vision AVR*!
4. Meng-*Upload* "*Coding*" yang telah ditulis, ke dalam desain rancangan Rangkaian Skematik kalian masing-masing!
5. *Screenshot* atau *Capture* hasilnya! seperti tampak pada Gambar 28.

6. Meng-*Upload* file (*fullscreen!*) dalam bentuk format **\*.pdf** ke Google Classroom, antara lain:
  - a. Rangkaian Skematik
  - b. *Coding*
  - c. Hasil *Capture*

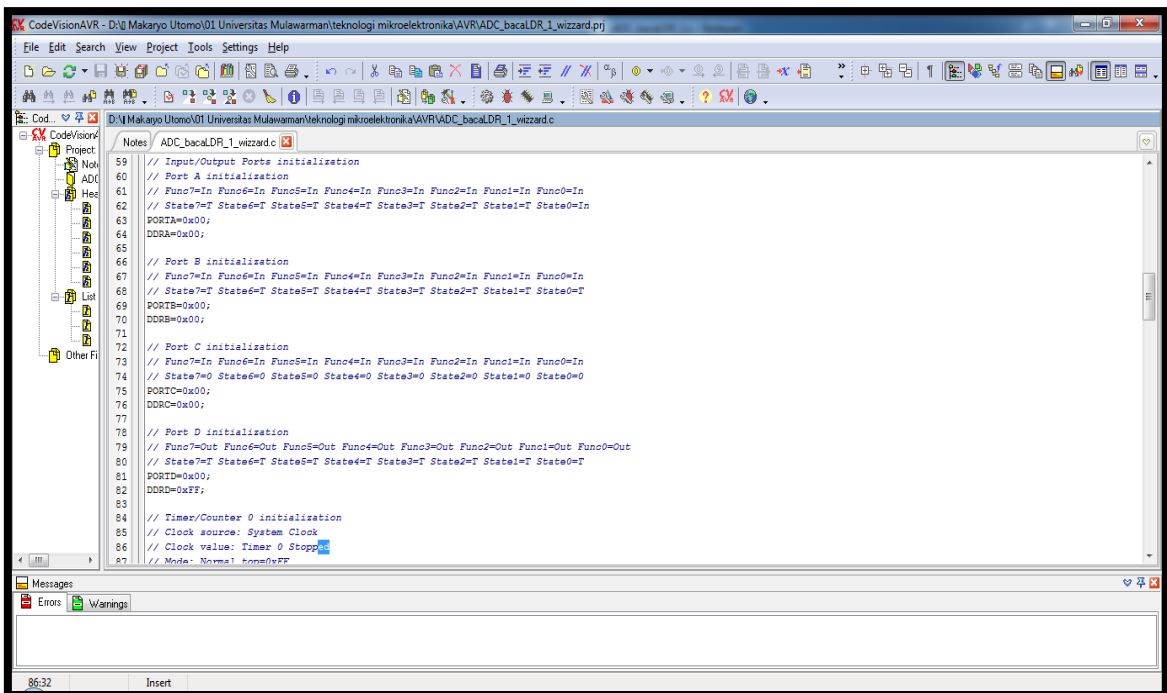


Adapun *Coding* yang digunakan dalam Modul Lab 7 (akuisi data menggunakan ADC) adalah:



```
CodeVisionAVR - D:\Makayo Utomo\01 Universitas Mulawarman\teknologi mikroelektronika\AVR\ADC_bacaLDR_1_wizard.prj
File Edit Search View Project Tools Settings Help
D:\Makayo Utomo\01 Universitas Mulawarman\teknologi mikroelektronika\AVR\ADC_bacaLDR_1_wizard.c
Notes ADC_bacaLDR_1_wizard.c
1 //*****
2 This program was produced by the
3 CodeWizardAVR V2.05.3 Standard
4 Automatic Program Generator
5 © Copyright 1998-2011 Faval Haiduo, HP InfoTech s.r.l.
6 http://www.hpinfo.tech.com
7
8 Project :
9 Version :
10 Date : 6/17/2020
11 Author : Research_01
12 Company : Microsoft
13 Comments :
14
15
16 Chip type : ATmega16
17 Program type : Application
18 AVR Core Clock Frequency: 8.000000 MHz
19 Memory model : Small
20 External RAM size : 0
21 Data Stack size : 256
22 //*****
23
24 #include <mega16.h>
25 #include <stdio.h>
26 #include <delay.h>
27
28 // Alphanumeric LCD function
```

(a)



```
CodeVisionAVR - D:\Makayo Utomo\01 Universitas Mulawarman\teknologi mikroelektronika\AVR\ADC_bacaLDR_1_wizard.prj
File Edit Search View Project Tools Settings Help
D:\Makayo Utomo\01 Universitas Mulawarman\teknologi mikroelektronika\AVR\ADC_bacaLDR_1_wizard.c
Notes ADC_bacaLDR_1_wizard.c
59 // Input/Output Ports initialization
60 // Port A initialization
61 // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
62 // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=In
63 PORTA=0x00;
64 DDRA=0x00;
65
66 // Port B initialization
67 // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
68 // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
69 PORTB=0x00;
70 DDRB=0x00;
71
72 // Port C initialization
73 // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
74 // State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
75 PORTC=0x00;
76 DDRC=0x00;
77
78 // Port D initialization
79 // Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out Func0=Out
80 // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
81 PORTD=0x00;
82 DDRD=0xFF;
83
84 // Timer/Counter 0 initialization
85 // Clock source: System Clock
86 // Clock value: Timer 0 Stopp
87 // Mode: Normal bps/bvFF
```

(b)

The screenshot shows the CodeVisionAVR IDE with the following code in the main editor window:

```
Notes ADC_bacaLDR_1_wizard.c
59 // Input/Output Ports initialization
60 // Port A initialization
61 // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
62 // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=In
63 PORTA=0x00;
64 DDRA=0x00;
65
66 // Port B initialization
67 // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
68 // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
69 PORTB=0x00;
70 DDRB=0x00;
71
72 // Port C initialization
73 // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
74 // State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
75 PORTC=0x00;
76 DDRC=0x00;
77
78 // Port D initialization
79 // Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out Func0=Out
80 // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
81 PORTD=0x00;
82 DDRD=0xFF;
83
84 // Timer/Counter 0 initialization
85 // Clock source: System Clock
86 // Clock value: Timer 0 Stopped
87 // Mode: Normal top=0xFF
```

(c)

The screenshot shows the CodeVisionAVR IDE with the following code in the main editor window:

```
Notes ADC_bacaLDR_1_wizard.c
87 // Mode: Normal top=0xFF
88 // OC0 output: Disconnected
89 TCCR0=0x00;
90 TCNT0=0x00;
91 OCR0=0x00;
92
93 // Timer/Counter 1 initialization
94 // Clock source: System Clock
95 // Clock value: Timer1 Stopped
96 // Mode: Normal top=0xFFFF
97 // OC1A output: Discon.
98 // OC1B output: Discon.
99 // Noise Canceler: Off
100 // Input Capture on Falling Edge
101 // Timer1 Overflow Interrupt: Off
102 // Input Capture Interrupt: Off
103 // Compare A Match Interrupt: Off
104 // Compare B Match Interrupt: Off
105 TCCR1A=0x00;
106 TCCR1B=0x00;
107 TCNT1H=0x00;
108 TCNT1L=0x00;
109 ICR1H=0x00;
110 ICR1L=0x00;
111 OCR1AH=0x00;
112 OCR1AL=0x00;
113 OCR1BH=0x00;
114 OCR1BL=0x00;
115
```

(d)

The screenshot shows the CodeVisionAVR IDE interface. The main window displays a C program with initialization code for Timer/Counter 2. The code includes comments and register assignments for various hardware modules. The status bar at the bottom shows the cursor is at line 143:11, with the text 'Insert'.

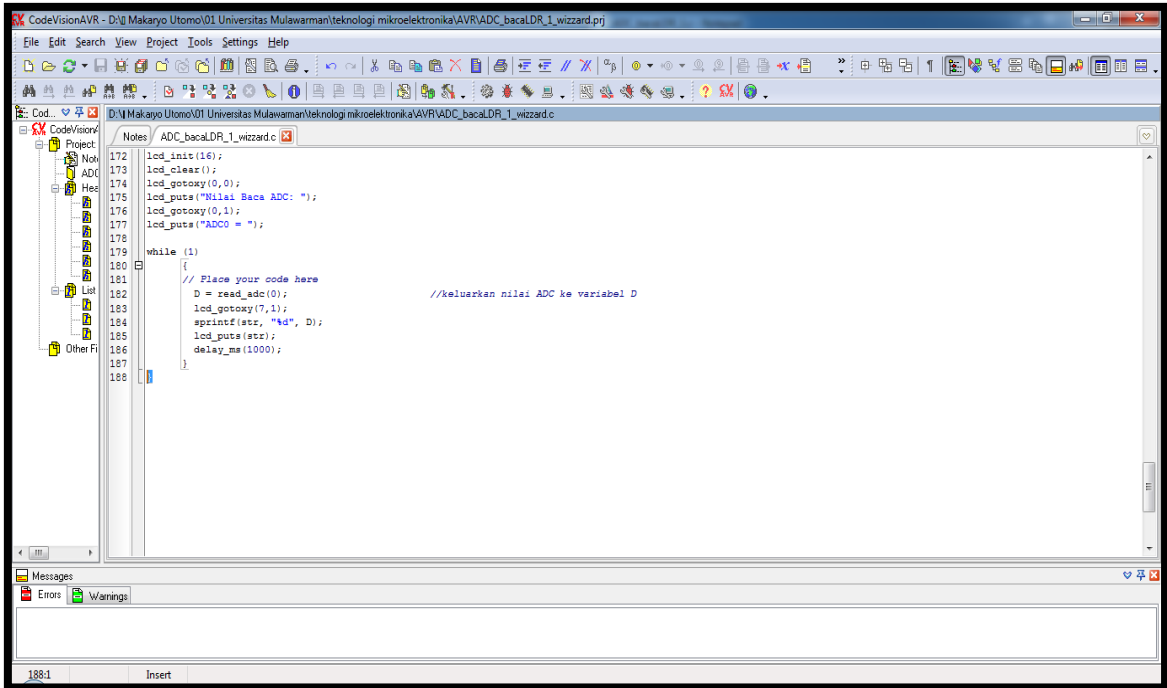
```
116 // Timer/Counter 2 initialization
117 // Clock source: System Clock
118 // Clock value: Timer2 Stopped
119 // Mode: Normal top=0xFF
120 // OC2 output: Disconnected
121 ASSR=0x00;
122 TCCR2=0x00;
123 TCNT2=0x00;
124 OCR2=0x00;
125
126 // External Interrupt(s) initialization
127 // INT0: Off
128 // INT1: Off
129 // INT2: Off
130 MCUCR=0x00;
131 MCUCSR=0x00;
132
133 // Timer(s)/Counter(s) Interrupt(s) initialization
134 TIMSK=0x00;
135
136 // USART initialization
137 // USART disabled
138 UCSRB=0x00;
139
140 // Analog Comparator initialization
141 // Analog Comparator: Off
142 // Analog Comparator Input Capture by Timer/Counter 1: Off
143 ACSR=0x00;
144 SFIOR=0x00;
```

(e)

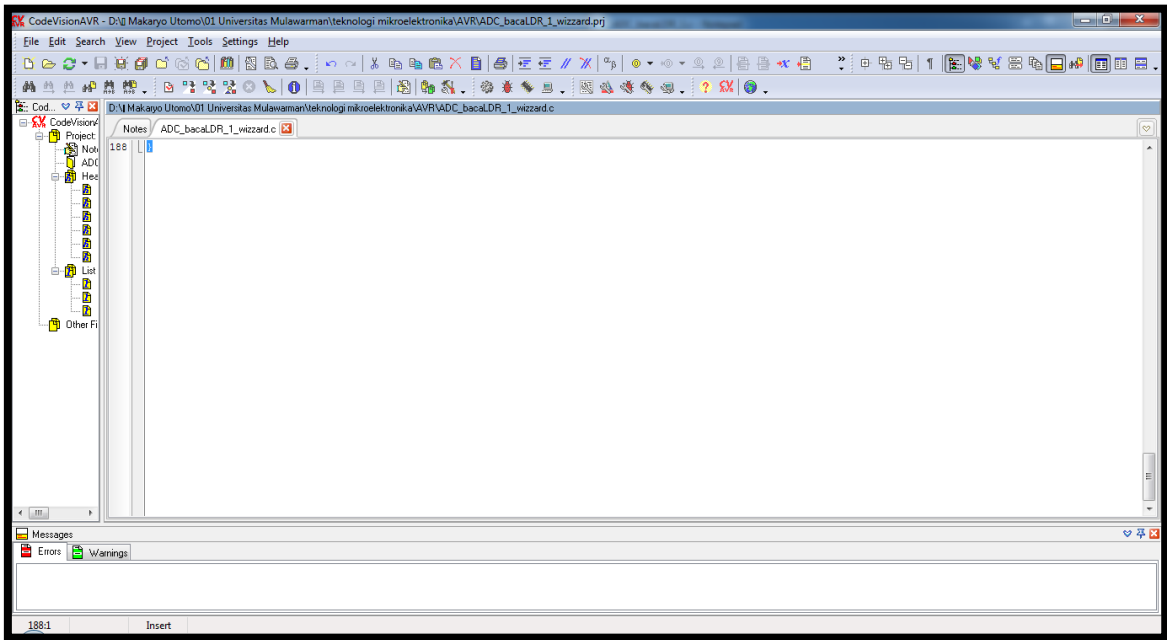
The screenshot shows the CodeVisionAVR IDE interface. The main window displays a C program with initialization code for ADC, SPI, TWI, and LCD. The code includes comments and register assignments for various hardware modules. The status bar at the bottom shows the cursor is at line 171:23, with the text 'Insert'.

```
144 SFIOR=0x00;
145
146 // ADC initialization
147 // ADC Clock frequency: 62.500 kHz
148 // ADC Voltage Reference: AVCC pin
149 // ADC Auto Trigger Source: ADC Stopped
150 ADMUX=ADC_VREF_TYPE & 0xFF; //ADMUX=0x40;
151 ADCSRA=0x97;
152
153 // SPI initialization
154 // SPI disabled
155 SPCR=0x00;
156
157 // TWI initialization
158 // TWI disabled
159 TWCR=0x00;
160
161 // Alphanumeric LCD initialization
162 // Connections are specified in the
163 // Project/Configure/Compiler/Libraries/Alphanumeric LCD menu:
164 // RS - PORTD Bit 0
165 // RD - PORTD Bit 1
166 // EN - PORTD Bit 2
167 // D4 - PORTD Bit 4
168 // D5 - PORTD Bit 5
169 // D6 - PORTD Bit 6
170 // D7 - PORTD Bit 7
171 // Characters/Line: 16
172
```

(f)



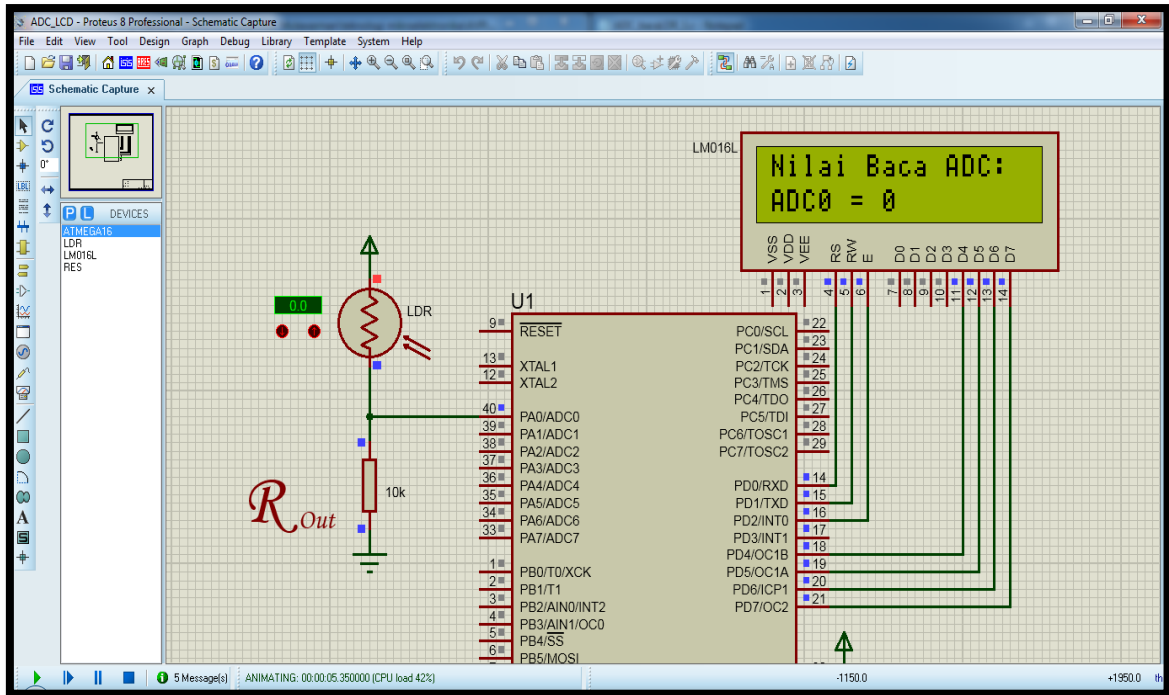
(g)



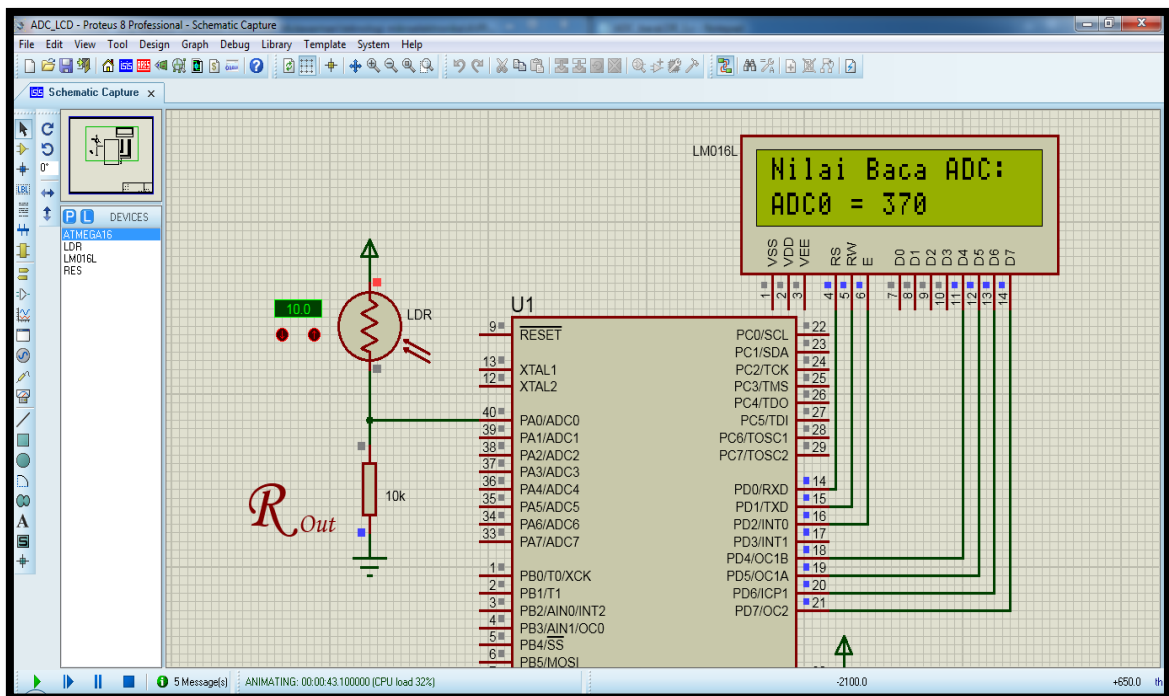
(h)

**Gambar 27 (a) (b) (c) (d) (e) (f) (g) (h). Coding pada Code Vision AVR**

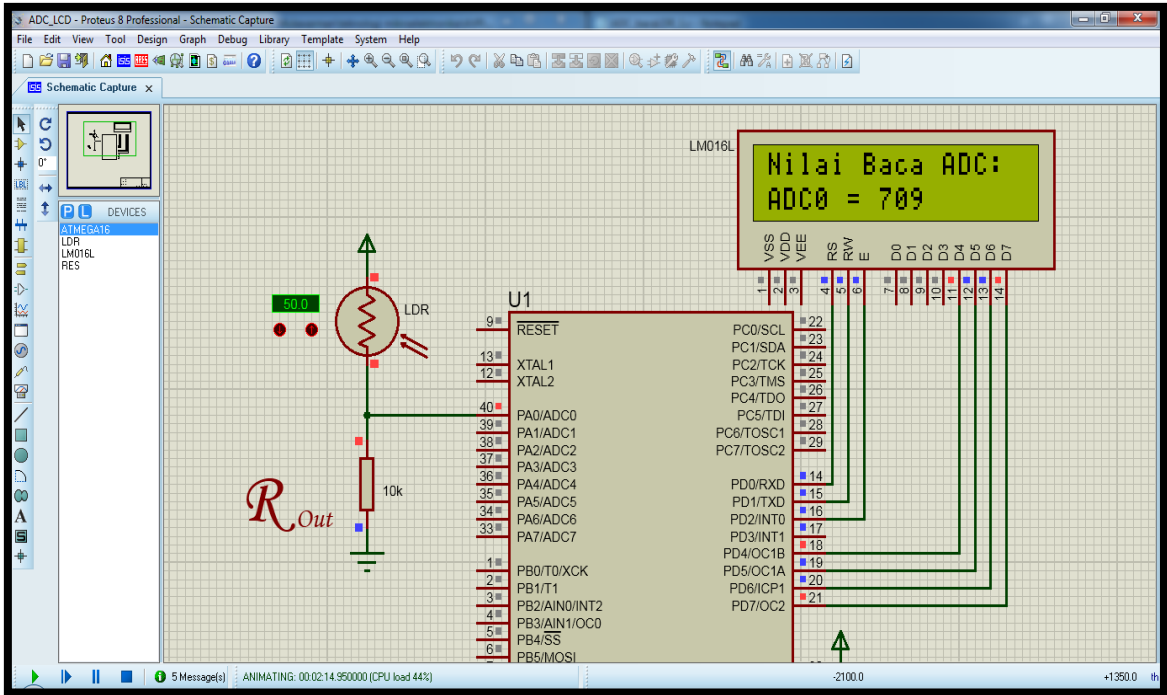
Hasil Capture Percobaan Lab 7, yakni:



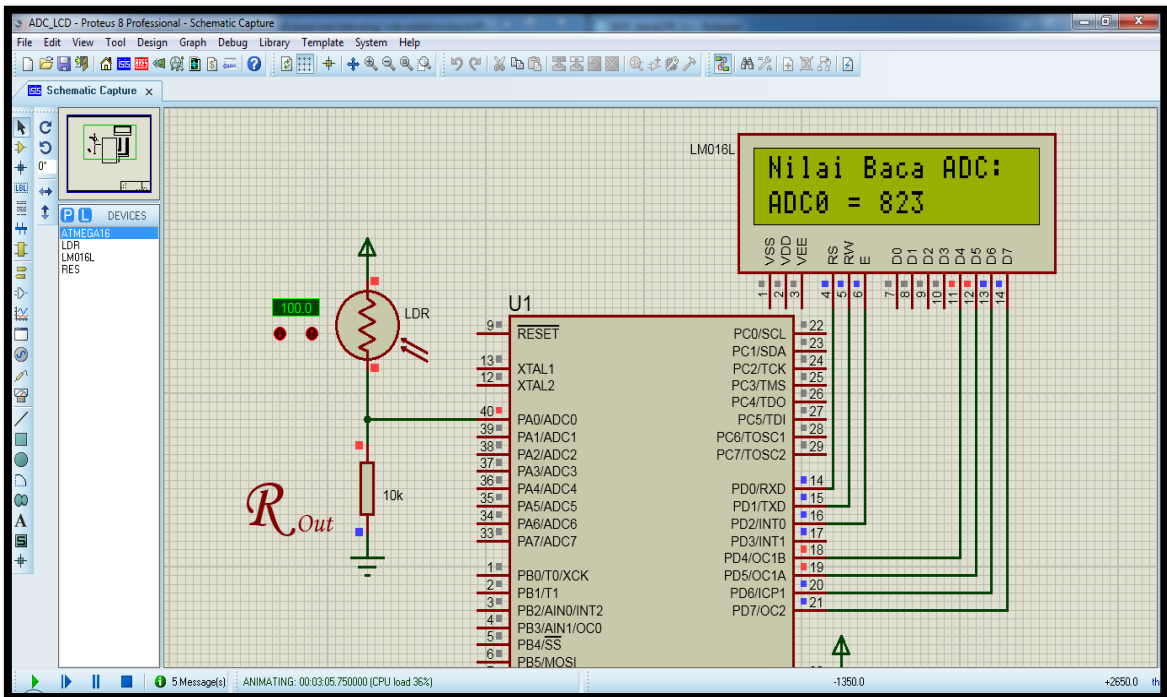
(a)



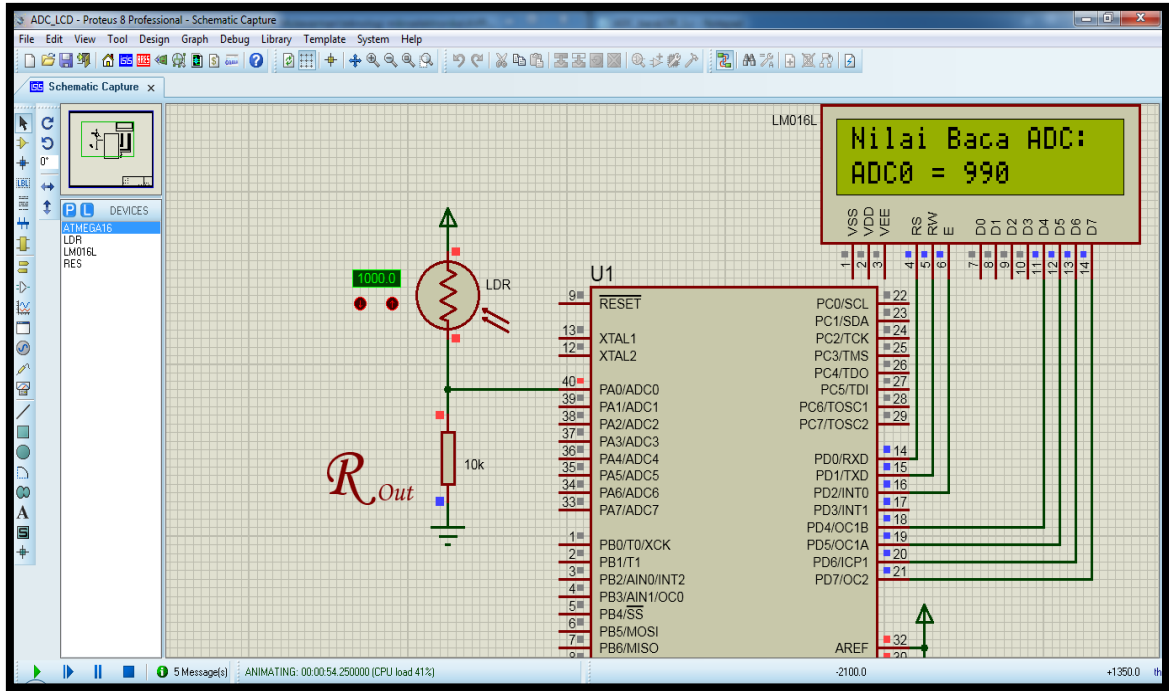
(b)



(c)



(d)



(e)

Gambar 28 (a) (b) (c) (d) (e). Hasil percobaan Lab 7

LAB 8.

**Interaksi I/O antara Respon  
Akuisisi Data ADC terhadap  
Kendali Sinyal PWM**

Asisten Percobaan :

NIM :

Tanggal Percobaan :



## Lab 8. Kontrol sinyal PWM menggunakan Data Input ADC

Latihan mengendalikan sinyal *Pulse Width Modulation* (PWM) yang berupa gelombang kotak, menggunakan kumpulan data yang telah diperoleh dari fitur *Analog to Digital Converter* (ADC). Prinsip kerja dari Modul Lab 8 ini adalah data analog yang diperoleh dan dikumpulkan dari perubahan nilai resistor (sebagai masukan), kemudian digunakan untuk mengendalikan bentuk gelombang pada sinyal PWM (sebagai luaran). Untuk memperoleh data dengan fitur ADC, ada dua *register* yang berperan sangat penting dalam proses Akuisisi Data pada Modul Lab 8, antara lain:

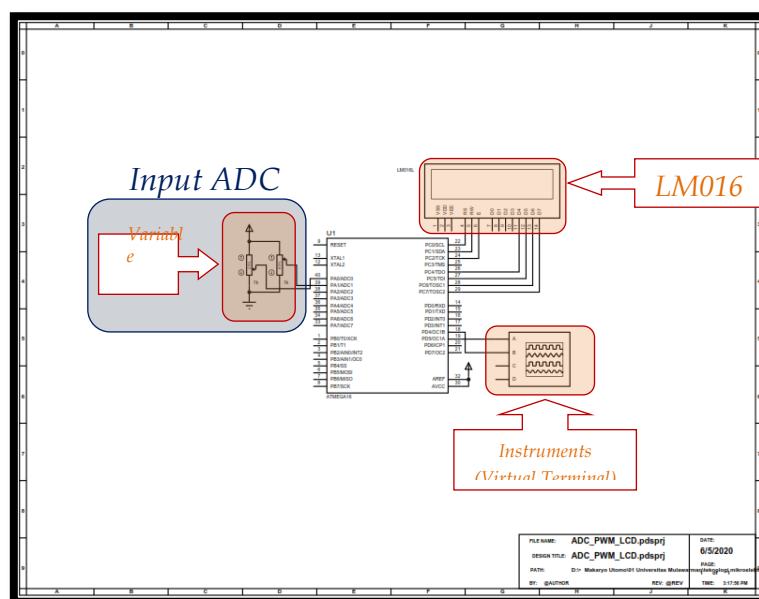
- a. ADCSRA; dan,
- b. ADMUX

Selain itu, untuk mengendalikan sinyal PWM, ada empat *register* yang berperan sangat penting dalam mengendalikan perubahan sinyal PWM pada percobaan Modul Lab 8 ini, antara lain:

- a. TCCR1A;
- b. TCCR1B;
- c. OCR1AL; dan,
- c. OCR1BL

Indikator perubahan sinyal PWM dapat dicari dengan menganalisa perubahan nilai dan siklus *duty cycle* pada gelombang kotak sinyal PWM, seperti tampak pada Gambar 30 (b).

Luaran (output) dari sinyal PWM ini dapat diterapkan untuk mengontrol motor-DC, motor *Servo*, terang-redupnya cahaya LED, dan aplikasi yang lainnya.

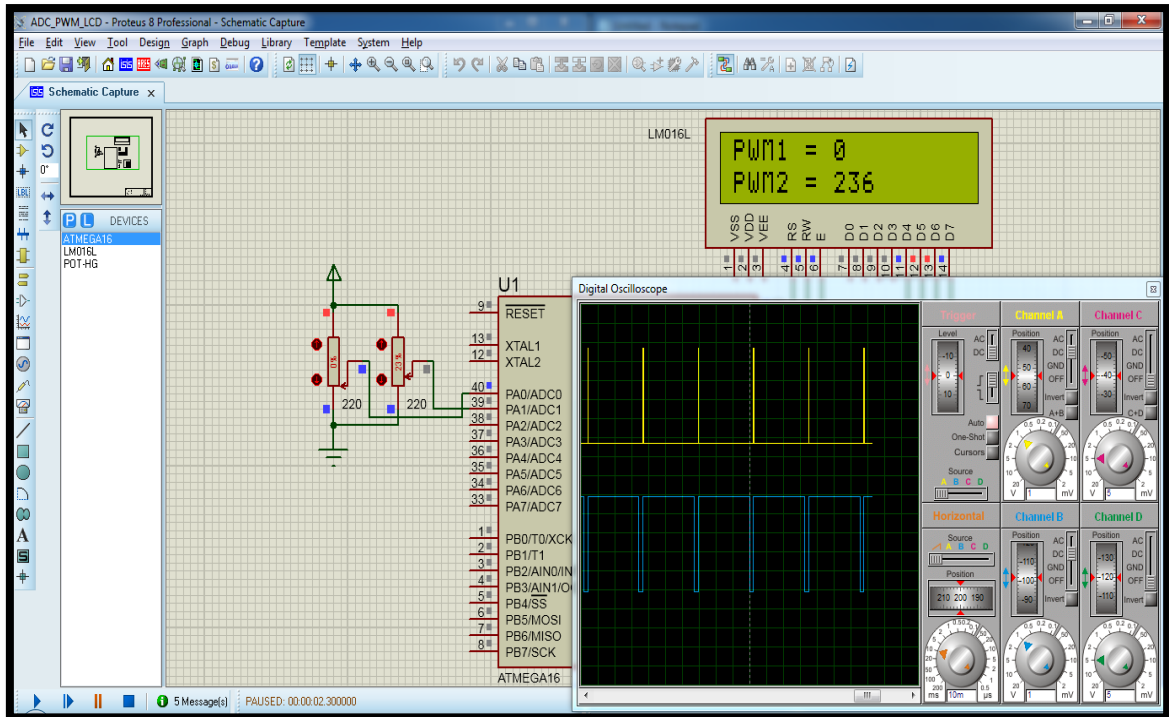


Gambar 29. Rangkaian skematik untuk menampilkan bilangan BCD

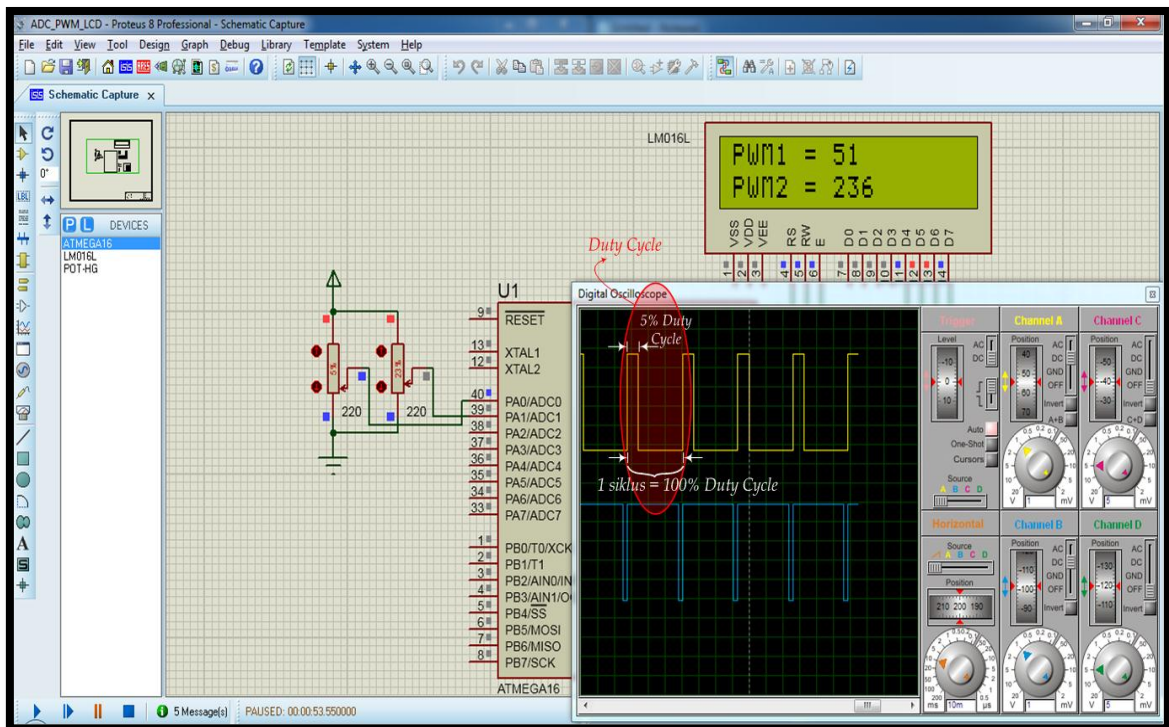
Langkah mempersiapkan Alat dan Bahan, yakni:

1. Laptop yang telah ter-*install* aplikasi *Proteus 8 Professional* dan *Code Vision AVR*!
2. Merancang Rangkaian Skematik seperti terlihat pada Gambar 29, menggunakan aplikasi *Proteus 8 Professional* pada laptop masing-masing!
3. Menuliskan "*Coding*" (**Tugas kalian**) ke dalam aplikasi *Code Vision AVR*!
4. Meng-*Upload* "*Coding*" yang telah ditulis, ke dalam desain rancangan Rangkaian Skematik kalian masing-masing!
5. *Screenshot* atau *Capture* hasilnya! seperti tampak pada Gambar 30.
6. *Upload* semua file (*fullscreen!*) menjadi satu dalam bentuk format **\*.pdf** via Google Classroom, antara lain:
  - a. Rangkaian Skematik
  - b. *Coding* CVAVR
  - c. Hasil *Capture*

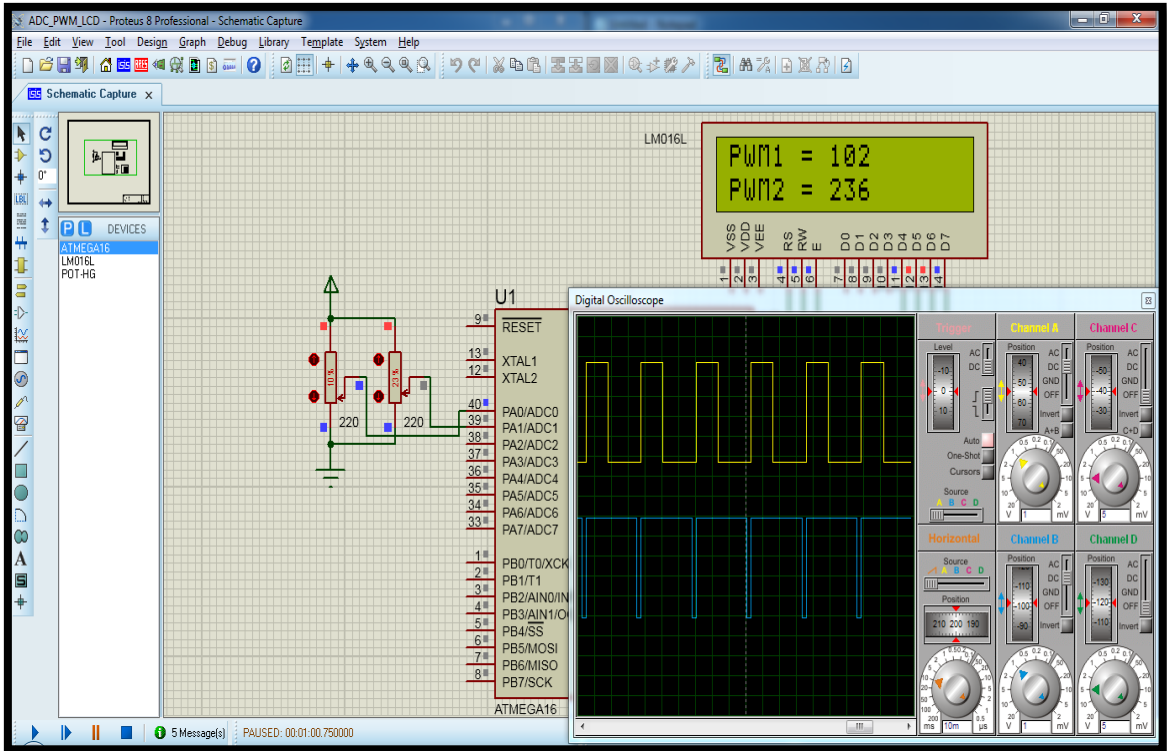
Hasil Capture Percobaan Lab 8, yakni:



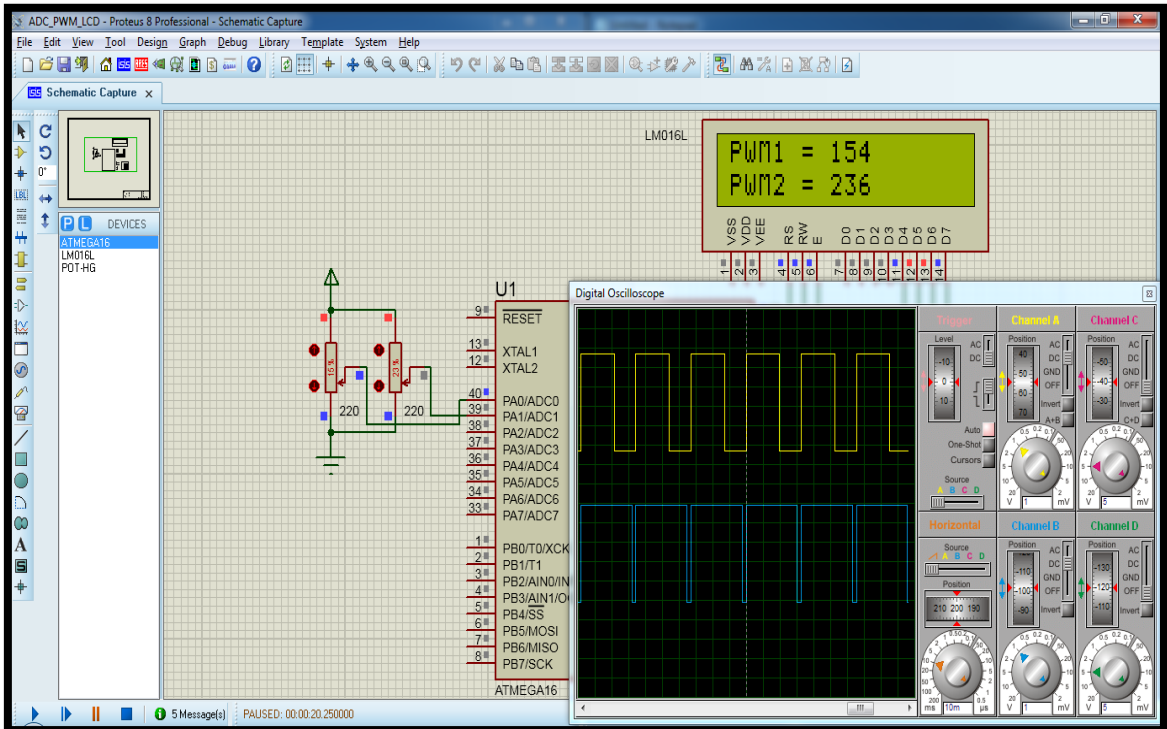
(a)



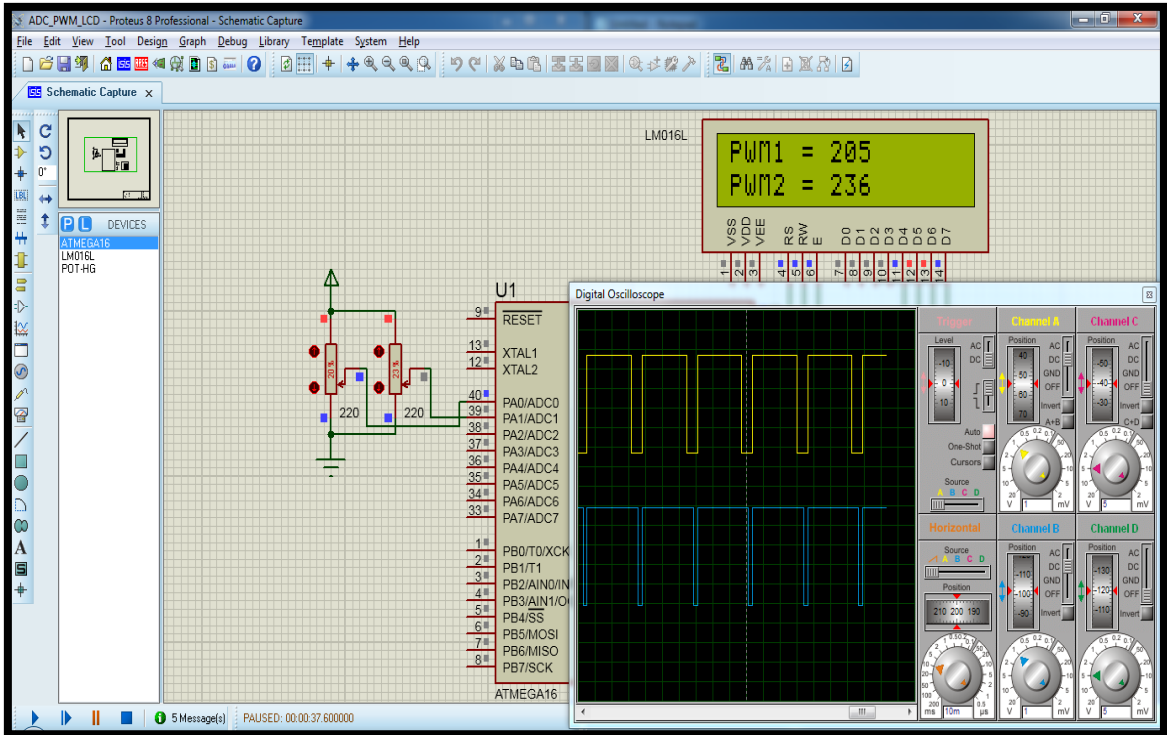
(b)



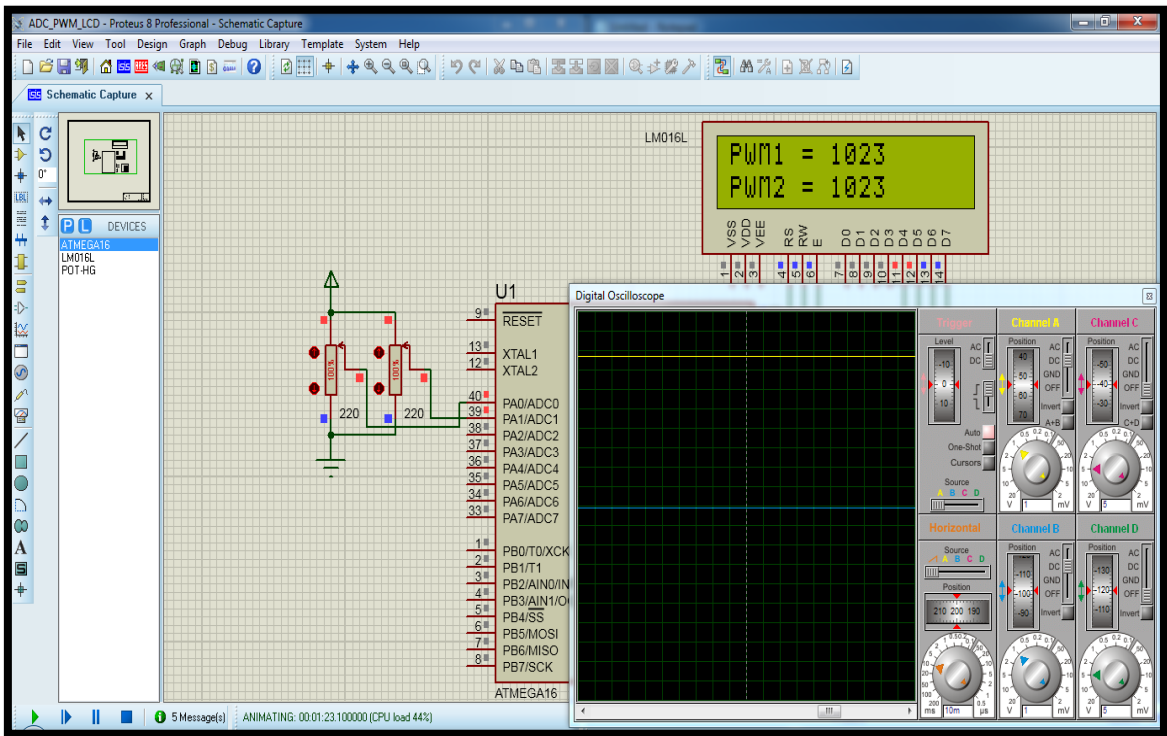
(c)



(d)



(e)



(f)

Gambar 30 (a) (b) (c) (d) (e) (f). Hasil percobaan Lab 8