

BUKU AJAR
PRAKTIKUM TEKNOLOGI SENSOR



Oleh :

Dr. Setiawardhana, S.T., M.T.

Ir. Sigit Wasista, M.Kom.

PROGRAM STUDI TEKNIK KOMPUTER
POLITEKNIK ELEKTRONIKA NEGERI SURABAYA

KATA PENGANTAR

Alhamdulillah, puji syukur ke hadirat Allah SWT, atas limpahan karunia-NYA kami dapat menyelesaikan diktat Praktikum Teknologi Sensor. Diharapkan dengan adanya diktat ini, mahasiswa dapat belajar secara mandiri dan lebih memahami mata kuliah Teori dan Praktikum Teknologi Sensor.

Terima kasih kami sampaikan kepada berbagai pihak yang telah memberikan kontribusi dalam penyelesaian buku ini, yaitu:

- Dosen dan staf di Program Studi Teknik Komputer
- Mahasiswa kami : Ahmad Reza Muztofa, D4 Teknik Komputer Angkatan 2009

Tak ada gading yang tak retak, begitu kata pepatah. Demikian juga dengan kami, sekalipun buku ini telah selesai melalui proses dan review yang cukup lama, namun masih terbuka kemungkinan adanya beberapa kekurangan di dalamnya. Oleh karena itu, masukan, kritik dan saran sangat kami harapkan untuk lebih menyempurnakan diktat ini pada kesempatan mendatang.

Mudah-mudahan sedikit yang kami bisa sumbangkan ini, akan dicatat oleh Allah SWT sebagai bagian dari amal sholeh kami dan akan menjadi ilmu yang bermanfaat, yang senantiasa akan mengalirkan pahala bagi orang-orang yang mengajarkannya. Aamiin.

Surabaya, Nopember 2012

Penulis

DAFTAR ISI

Halaman Depan	i
Kata Pengantar	ii
Daftar Isi	iii
Daftar Gambar	iv
Pengantar Teknik Instalasi dan Penggunaan Perangkat	1
Percobaan 1 Sensor Cahaya	10
Percobaan 2 Sensor Suhu : LM 35	28
Percobaan 3 Sensor Suara	46
Percobaan 4 Aktuator : Motor DC	64
Percobaan 5 Counter Rotary Encoder	83
Percobaan 6 Sensor Jarak : Ultrasonik SRF04	102
Percobaan 7 Sensor Jarak : Ultrasonik Ping Parallax	119
Percobaan 8 Sensor Posisi : Kompas CMPS04	137
Percobaan 9 Sensor Panas Api : UVTRON	153
Percobaan 10 Sensor Tekanan MPX5100GP	170
Percobaan 11 Sensor Gerakan : PIR Motion Parallax	187
Percobaan 12 Sensor Kelembaban : SHT 10	204
Percobaan 13 Sensor Suhu : SHT 10	219
Percobaan 14 Sensor Gas LPG : MQ5 LPG Gas	234
Daftar Pustaka	252
Lampiran Modul Praktikum	253

DAFTAR GAMBAR

Gambar A.1. eXtremeBurnerAVRSetupV1.2.exe pada windows explorer	2
Gambar A.2 Dialog Informasi	3
Gambar A.3 Dialog Welcome	3
Gambar A.4 Setting directory	4
Gambar A.5 Ready to Install	4
Gambar A.6 Progres instalasi	5
Gambar A.7 Instalasi selesai	5
Gambar B.1 Shortcut eXtreme Burner – AVR	6
Gambar B.2 Splash screen eXtreme Burner AVR	6
Gambar B.3 eXtreme Burner AVR	7
Gambar B.4 Tipe Mikrokontroler	7
Gambar B.5 Open Flash	8
Gambar B.6 Message	8
Gambar B.7 Proses Berhasil	9
Gambar B.8 Proses Gagal	9
Gambar B.9 Proses Gagal	9
Gambar 1.1. Timer Diagram untuk Mode Single Conversion	11
Gambar 1.2 Development board keseluruhan	14
Gambar 1.3. Komponen Training board	14
Gambar 1.4 Komponen Training board	15
Gambar 1.5 Shortcut CodeVisionAVR	15
Gambar 1.6 Membuat File baru	16
Gambar 1.7 Membuat project baru	16
Gambar 1.8 Memilih untuk membuat project baru pada CodeVisionAVR	17
Gambar 1.9 Setting LCD pada PortB dan Setting ADC 8 bit	17
Gambar 1.10 Menyimpan setting	18
Gambar 1.11 Memberi nama pada file C (*.c)	18
Gambar 1.12 Memberi nama project (*.prj)	19
Gambar 1.13 Memberi nama CodeWizardProject (*.cwp)	19
Gambar 1.14 Editor Program	20

Gambar 1.15 Build Source	24
Gambar 1.16 Dialog informasi status kompilasi	24
Gambar 1.17 Shortcut eXtreme Burner – AVR	25
Gambar 1.18 Tipe Mikrokontroler	25
Gambar 1.19 Open Flash	26
Gambar 1.20 Message	26
Gambar 1.21 Proses Berhasil	27
Gambar 1.22 Running Program	27
Gambar 2.1. Timer Diagram untuk Mode Single Conversion	29
Gambar 2.2 Development board dan rangkaian sensor suhu	31
Gambar 2.3 Komponen Training board	32
Gambar 2.4 Komponen Training board	32
Gambar 2.5 Shortcut CodeVisionAVR	33
Gambar 2.6 Membuat File baru	33
Gambar 2.7 Membuat project baru	34
Gambar 2.8 Memilih untuk membuat project baru pada CodeVision AVR	34
Gambar 2.9 Setting LCD pada PortB dan Setting ADC 8 bit	35
Gambar 2.10 Menyimpan setting	35
Gambar 2.11 Memberi nama pada file C (*.c)	36
Gambar 2.12 Memberi nama project (*.prj)	36
Gambar 2.13 Memberi nama CodeWizardProject (*.cwp)	37
Gambar 2.14 Editor Program	37
Gambar 2.15 Build Source	41
Gambar 2.16 Dialog informasi status kompilasi	42
Gambar 2.17 Shortcut eXtreme Burner – AVR	42
Gambar 2.18 Tipe Mikrokontroler	43
Gambar 2.19 Open Flash	43
Gambar 2.20 Message	44
Gambar 2.21 Proses Berhasil	44
Gambar 2.22 Running Program	45
Gambar 3.1 Timer Diagram untuk Mode Single Conversion	47
Gambar 3.2 Rangkaian sensor suara	50
Gambar 3.3 Komponen Training board	50

Gambar 3.4 Komponen Training board	51
Gambar 3.5 Shortcut CodeVisionAVR	51
Gambar 3.6 Membuat File baru	52
Gambar 3.7 Membuat project baru	52
Gambar 3.8 Memilih untuk membuat project baru pada CodeVisionAVR	53
Gambar 3.9 Setting LCD pada PortB dan Setting ADC 8 bit	53
Gambar 3.10 Menyimpan setting	54
Gambar 3.11 Memberi nama pada file C (*.c)	54
Gambar 3.12 Memberi nama project (*.prj)	55
Gambar 3.13 Memberi nama CodeWizardProject (*.cwp)	55
Gambar 3.14 Editor Program	56
Gambar 3.15 Build Source	60
Gambar 3.16 Dialog informasi status kompilasi	60
Gambar 3.17 Shortcut eXtreme Burner – AVR	61
Gambar 3.18 Tipe Mikrokontroler	61
Gambar 3.19 Open Flash	62
Gambar 3.20 Message	62
Gambar 3.21 Proses Berhasil	63
Gambar 4.1 Mekanisme motor DC	65
Gambar 4.2 Diagram timer/counter	66
Gambar 4.3 Timing diagram timer/counter, tanpa prescaling	66
Gambar 4.4 Timing diagram timer/counter, dengan prescaling	67
Gambar 4.5 Rangkaian driver motor pada modul	67
Gambar 4.6 Pengaturan koneksi pada Training board	68
Gambar 4.7 Shortcut CodeVisionAVR	68
Gambar 4.8 Membuat File baru	69
Gambar 4.9 Membuat project baru	69
Gambar 4.10 Memilih untuk membuat project baru pada CodeVisionAVR	70
Gambar 4.11 Setting LCD pada PortB	71
Gambar 4.12 Dialog Warning	71
Gambar 4.13 Menyimpan setting	72
Gambar 4.14 Memberi nama pada file C (*.c)	72
Gambar 4.15 Memberi nama project (*.prj)	73

Gambar 4.16 Memberi nama CodeWizardProject (*.cwp)	73
Gambar 4.17 Editor Program	74
Gambar 4.18 Build Source	76
Gambar 4.19 Dialog informasi status kompilasi	77
Gambar 4.20 Shortcut eXtreme Burner – AVR	77
Gambar 4.21 Tipe Mikrokontroler	78
Gambar 4.22 Open Flash	78
Gambar 4.23 Message	79
Gambar 4.24 Proses Berhasil	79
Gambar 5.1 Rotary Encoder Absolute	85
Gambar 5.2 Rotary Encoder Relative	85
Gambar 5.3 Development board keseluruhan	89
Gambar 5.4 Komponen Training board	89
Gambar 5.5 Komponen Training board	90
Gambar 5.6 Shortcut CodeVisionAVR	90
Gambar 5.7 Membuat File baru	91
Gambar 5.8 Membuat project baru	91
Gambar 5.9 Memilih untuk membuat project baru pada CodeVisionAVR	92
Gambar 5.10 Setting Timer 0 dan Setting Port D sebagai pin output	92
Gambar 5.11 Menyimpan setting	93
Gambar 5.12 Memberi nama pada file C (*.c)	93
Gambar 5.13 Memberi nama project (*.prj)	94
Gambar 5.14 Memberi nama CodeWizardProject (*.cwp)	94
Gambar 5.15 Editor Program	95
Gambar 5.16 Build Source	98
Gambar 5.17 Dialog informasi status kompilasi	98
Gambar 5.18 Shortcut eXtreme Burner – AVR	99
Gambar 5.19 Tipe Mikrokontroler	99
Gambar 5.20 Open Flash	100
Gambar 5.21 Message	100
Gambar 5.22 Proses Berhasil	101
Gambar 6.1 Sensor ultrasonik SRF04	103
Gambar 6.2 Timming SRF04	104

Gambar 6.3 Development board keseluruhan	104
Gambar 6.4 Komponen Training board	105
Gambar 6.5 Komponen Training board	105
Gambar 6.6 Shortcut CodeVisionAVR	106
Gambar 6.7 Membuat File baru	107
Gambar 6.8 Membuat project baru	107
Gambar 6.9 Memilih untuk membuat project baru pada CodeVisionAVR	108
Gambar 6.10 Setting Port A sebagai pin input - output dan LCD pada PortB	108
Gambar 6.11 Menyimpan setting	109
Gambar 6.12 Memberi nama pada file C (*.c)	109
Gambar 6.13 Memberi nama project (*.prj)	110
Gambar 6.14 Memberi nama CodeWizardProject (*.cwp)	110
Gambar 6.15 Editor Program	111
Gambar 6.16 Build Source	115
Gambar 6.17 Dialog informasi status kompilasi	115
Gambar 6.18 Shortcut eXtreme Burner – AVR	116
Gambar 6.19 Tipe Mikrokontroler	116
Gambar 6.20 Open Flash	117
Gambar 6.21 Message	117
Gambar 6.22 Proses Berhasil	118
Gambar 6.23 Running Program	118
Gambar 7.1 Sensor PING Parallax	120
Gambar 7.2 Konfigurasi Sensor PING Parallax	121
Gambar 7.3 Timming Sensor PING Parallax	121
Gambar 7.4 Development board keseluruhan	122
Gambar 7.5 Komponen Training board	123
Gambar 7.6 Komponen Training board	123
Gambar 7.7 Shortcut CodeVisionAVR	124
Gambar 7.8 Membuat File baru	125
Gambar 7.9 Membuat project baru	125
Gambar 7.10 Memilih untuk membuat project baru pada CodeVisionAVR	126
Gambar 7.11 Setting LCD pada PortB	126
Gambar 7.12 Menyimpan setting	127

Gambar 7.13 Memberi nama pada file C (*.c)	127
Gambar 7.14 Memberi nama project (*.prj)	128
Gambar 7.15 Memberi nama CodeWizardProject (*.cwp)	128
Gambar 7.16 Editor Program	129
Gambar 7.17 Build Source	133
Gambar 7.18 Dialog informasi status kompilasi	133
Gambar 7.19 Shortcut eXtreme Burner – AVR	134
Gambar 7.20 Tipe Mikrokontroler	134
Gambar 7.21 Open Flash	135
Gambar 7.22 Message	135
Gambar 7.23 Proses Berhasil	136
Gambar 8.1 bentuk fisik dan koneksi pin modul kompas CMPS03	138
Gambar 8.2 Development board keseluruhan	139
Gambar 8.3 Komponen Training board	140
Gambar 8.4 Komponen Training board	140
Gambar 8.5 Shortcut CodeVisionAVR	141
Gambar 8.6 Membuat File baru	142
Gambar 8.7 Membuat project baru	142
Gambar 8.8 Memilih untuk membuat project baru pada CodeVisionAVR	143
Gambar 8.9 Setting I2C Port pada Port A dan Setting LCD pada PortB	143
Gambar 8.10 Menyimpan setting	144
Gambar 8.11 Memberi nama pada file C (*.c)	144
Gambar 8.12 Memberi nama project (*.prj)	145
Gambar 8.13 Memberi nama CodeWizardProject (*.cwp)	145
Gambar 8.14 Editor Program	146
Gambar 8.15 Build Source	149
Gambar 8.16 Dialog informasi status kompilasi	150
Gambar 8.17 Shortcut eXtreme Burner – AVR	150
Gambar 8.18 Tipe Mikrokontroler	151
Gambar 8.19 Open Flash	151
Gambar 8.20 Message	152
Gambar 8.21 Proses Berhasil	152
Gambar 9.1 UVTron Flame Detector	154

Gambar 9.2 Jangkauan sensor pada posisi tidur	155
Gambar 9.3 Jangkauan sensor pada posisi berdiri	156
Gambar 9.4 Development board	156
Gambar 9.5 Komponen Training board	157
Gambar 9.6 Konfigurasi Training board	157
Gambar 9.7 Shortcut CodeVisionAVR	158
Gambar 9.8 Membuat File baru	158
Gambar 9.9 Membuat project baru	159
Gambar 9.10 Memilih untuk membuat project baru pada CodeVisionAVR	159
Gambar 9.11 CodeWizardAVR pada tab Chip dan Setting LCD pada PortB	160
Gambar 9.12 Menyimpan setting	160
Gambar 9.13 Memberi nama pada file C (*.c)	161
Gambar 9.14 Memberi nama project (*.prj)	161
Gambar 9.15 Memberi nama CodeWizardProject (*.cwp)	161
Gambar 9.16 Build Source	165
Gambar 9.17 Dialog informasi status kompilasi	165
Gambar 9.18 Shortcut eXtreme Burner – AVR	166
Gambar 9.19 Tipe Mikrokontroler	166
Gambar 9.20 Open Flash	167
Gambar 9.21 Message	167
Gambar 9.22 Proses Berhasil	168
Gambar 9.23 Tidak ada api	168
Gambar 9.24 Ada api	169
Gambar 10.1 Sensor Tekanan MPX5100GP	171
Gambar 10.2 Grafik hubungan tekanan dengan output (V)	172
Gambar 10.3 Development board	173
Gambar 10.4 Komponen Training board	173
Gambar 10.5 Komponen Training board	174
Gambar 10.6 Shortcut CodeVisionAVR	175
Gambar 10.7 Membuat File baru	175
Gambar 10.8 Membuat project baru	176
Gambar 10.9 Memilih untuk membuat project baru pada CodeVisionAVR	176
Gambar 10.10 Setting LCD pada PortB dan Setting ADC 8 bit	177

Gambar 10.11 Menyimpan setting	178
Gambar 10.12 Memberi nama pada file C (*.c)	178
Gambar 10.13 Memberi nama project (*.prj)	179
Gambar 10.14 Memberi nama CodeWizardProject (*.cwp)	179
Gambar 10.15 Build Source	183
Gambar 10.16 Dialog informasi status kompilasi	183
Gambar 10.17 Shortcut eXtreme Burner – AVR	184
Gambar 10.18 Tipe Mikrokontroler	184
Gambar 10.19 Open Flash	185
Gambar 10.20 Message	185
Gambar 10.21 Proses Berhasil	186
Gambar 10.22 Running Program	186
Gambar 11.1 PIR	188
Gambar 11.2 Fresnel Lens	189
Gambar 11.3 Rangkaian Sensor Pyroelectric	190
Gambar 11.4 Cara Kerja Sensor Pyroelectric	190
Gambar 11.5 Development board	191
Gambar 11.6 Komponen Training board	191
Gambar 11.7 Komponen Training board	192
Gambar 11.8 Shortcut CodeVisionAVR	192
Gambar 11.9 Membuat File baru	193
Gambar 11.10 Membuat project baru	193
Gambar 11.11 Memilih untuk membuat project baru pada CodeVisionAVR	194
Gambar 11.12 CodeWizardAVR pada tab Chip dan Setting LCD pada PortB	194
Gambar 11.13 Menyimpan setting	195
Gambar 11.14 Memberi nama pada file C (*.c)	195
Gambar 11.15 Memberi nama project (*.prj)	196
Gambar 11.16 Memberi nama CodeWizardProject (*.cwp)	196
Gambar 11.17 Build Source	199
Gambar 11.18 Dialog informasi status kompilasi	199
Gambar 11.19 Shortcut eXtreme Burner – AVR	200
Gambar 11.20 Tipe Mikrokontroler	200
Gambar 11.21 Open Flash	201

Gambar 11.22 Message	201
Gambar 11.23 Proses Berhasil	202
Gambar 11.24 Dekatkan suatu obyek diatas PIR	202
Gambar 11.25 Respon Program	203
Gambar 12.1 Akurasi maksimal RH SHT10 dan Akurasi maksimal suhu SHT10	205
Gambar 12.2 Development board	206
Gambar 12.3 Komponen Training board	206
Gambar 12.4 Komponen Training board	207
Gambar 12.5 Shortcut CodeVisionAVR	207
Gambar 12.6 Membuat File baru	208
Gambar 12.7 Membuat project baru	208
Gambar 12.8 Memilih untuk membuat project baru pada CodeVisionAVR	209
Gambar 12.9 Setting LCD pada PortB dan Setting USART 8 bit	209
Gambar 12.10 Menyimpan setting	210
Gambar 12.11 Memberi nama pada file C (*.c)	210
Gambar 12.12 Memberi nama project (*.prj)	211
Gambar 12.13 Memberi nama CodeWizardProject (*.cwp)	211
Gambar 12.14 Build Source	215
Gambar 12.15 Dialog informasi status kompilasi	215
Gambar 12.16 Shortcut eXtreme Burner – AVR	216
Gambar 12.17 Tipe Mikrokontroler	216
Gambar 12.18 Open Flash	217
Gambar 12.19 Message	217
Gambar 12.20 Proses Berhasil	218
Gambar 12.21 Running Program	218
Gambar 13.1 Akurasi maksimal RH SHT10 dan Akurasi maksimal suhu SHT10	220
Gambar 13.2 Development board	221
Gambar 13.3 Komponen Training board	221
Gambar 13.4 Komponen Training board	222
Gambar 13.5 Shortcut CodeVisionAVR	223
Gambar 13.6 Membuat File baru	223
Gambar 13.7 Membuat project baru	224
Gambar 13.8 Memilih untuk membuat project baru pada CodeVisionAVR	224

Gambar 13.9 Setting LCD pada PortB dan Setting USART 8 bit	225
Gambar 13.10 Menyimpan setting	225
Gambar 13.11 Memberi nama pada file C (*.c)	226
Gambar 13.12 Memberi nama project (*.prj)	226
Gambar 13.13 Memberi nama CodeWizardProject (*.cwp)	226
Gambar 13.14 Build Source	230
Gambar 13.15 Dialog informasi status kompilasi	230
Gambar 13.16 Shortcut eXtreme Burner – AVR	231
Gambar 13.17 Tipe Mikrokontroler	231
Gambar 13.18 Open Flash	232
Gambar 13.19 Message	232
Gambar 13.20 Proses Berhasil	233
Gambar 13.21 Running Program	233
Gambar 14.1 Sensor Gas MQ5	235
Gambar 14.2 Rangkaian Sensor gas	236
Gambar 14.3 Development board	237
Gambar 14.4 Komponen Training board	237
Gambar 14.5 Komponen Training board	238
Gambar 14.6 Shortcut CodeVisionAVR	239
Gambar 14.7 Membuat File baru	239
Gambar 14.8 Membuat project baru	240
Gambar 14.9 Memilih untuk membuat project baru pada CodeVisionAVR	240
Gambar 14.10 Setting LCD pada PortB dan Setting ADC 8 bit	241
Gambar 14.11 Menyimpan setting	242
Gambar 14.12 Memberi nama pada file C (*.c)	242
Gambar 14.13 Memberi nama project (*.prj)	243
Gambar 14.14 Memberi nama CodeWizardProject (*.cwp)	243
Gambar 14.15 Build Source	247
Gambar 14.16 Dialog informasi status kompilasi	247
Gambar 14.17 Shortcut eXtreme Burner – AVR	248
Gambar 14.18 Tipe Mikrokontroler	248
Gambar 14.19 Open Flash	249
Gambar 14.20 Message	249

Gambar 14.21 Proses Berhasil	250
Gambar 14.22 Tes menggunakan gas korek api	250
Gambar 14.23 Respon Program	251
Gambar L.1 Minimum System AVR ATMega16	254
Gambar L.2 Training Board – Input/Output Module	255
Gambar L.3 Training Board – Input/Output Module	256

PENGANTAR

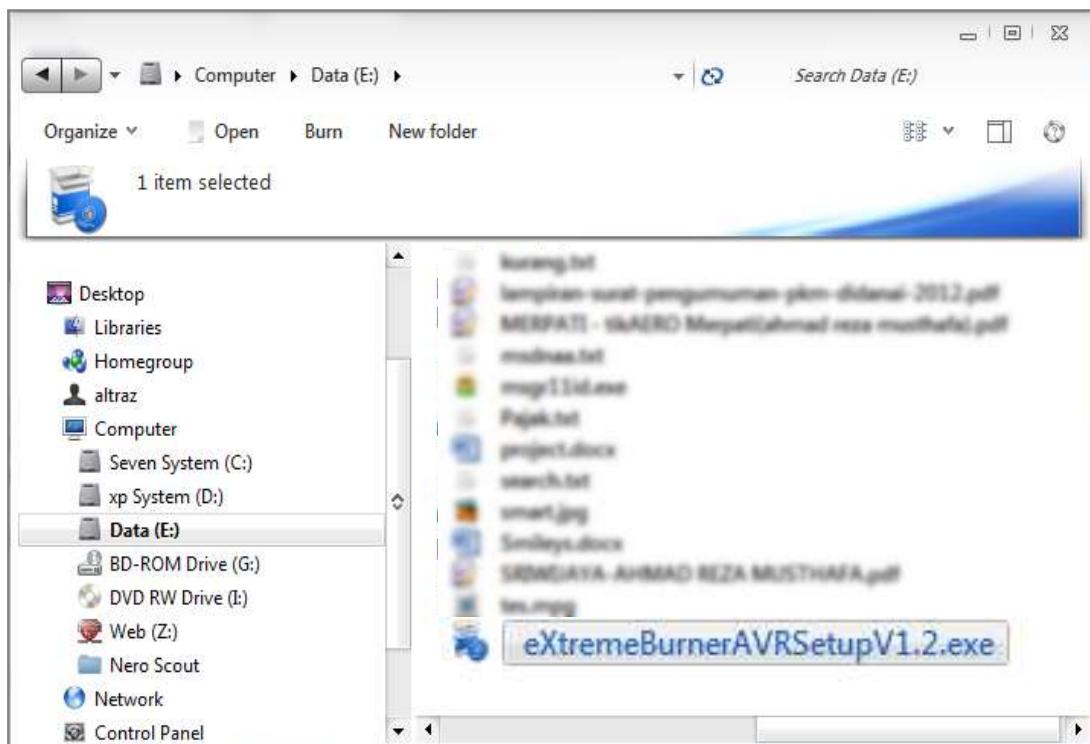
TEKNIK INSTALASI DAN PENGGUNAAN PERANGKAT

A. Instalasi eXtreme Burner AVR

Sebelum Anda memulai praktikum ini, sebaiknya terlebih dahulu melakukan instalasi terhadap aplikasi yang bernama *eXtreme Burner AVR*. Aplikasi ini berfungsi untuk melakukan download file hexadecimal (*.hex) ke dalam mikrokontroler AVR.

Berikut ini merupakan langkah-langkah untuk melakukan instalasi aplikasi eXtreme Burner AVR :

1. Pastikan Anda sudah memiliki file installer yang bernama **eXtremeBurnerAVRSetupV1.2.exe**. Apabila Anda sudah memiliki file installer, terlebih dahulu lakukan eksekusi dengan cara klik ganda file installer **eXtremeBurnerAVRSetupV1.2.exe** untuk memulai proses instalasi pada sistem operasi windows.



Gambar A.1. eXtremeBurnerAVRSetupV1.2.exe pada windows explorer

2. Setelah itu akan muncul dialog yang berisi pemberitahuan yang mengharapkan USBASP dalam keadaan terpasang pada port USB computer, seperti ditunjukkan pada gambar berikut :



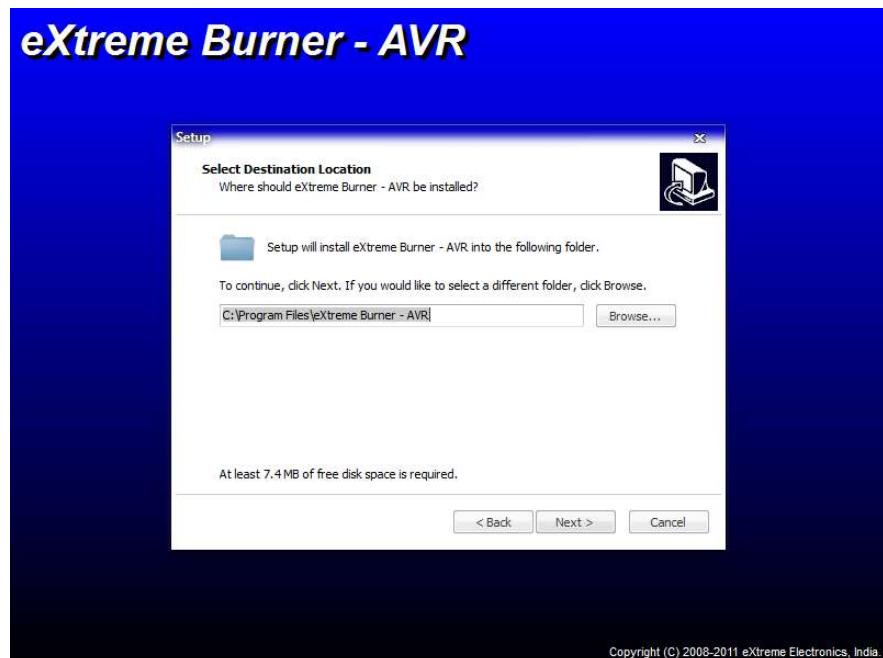
Gambar A.2 Dialog Informasi

Pastikan USBASP terpasang pada port USB komputer Anda. Jika sudah terpasang, tekan tombol “OK” untuk melanjutkan ke proses selanjutnya.



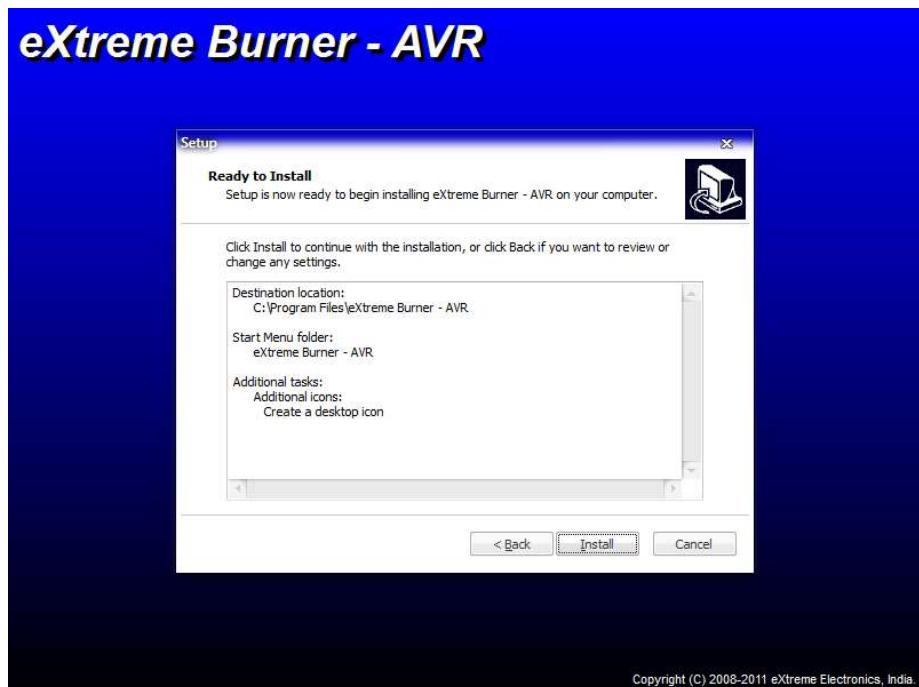
Gambar A.3 Dialog Welcome

3. Setelah itu akan muncul dialog selamat datang, tekan tombol “next” untuk melanjutkan ke proses selanjutnya. Setelah penekanan tombol “next”, Anda akan dihadapkan pada opsi untuk menentukan tujuan lokasi instalasi. Tekan tombol “next” apabila Anda sudah menentukan tujuan lokasi instalasi dan untuk melanjutkan ke proses selanjutnya.

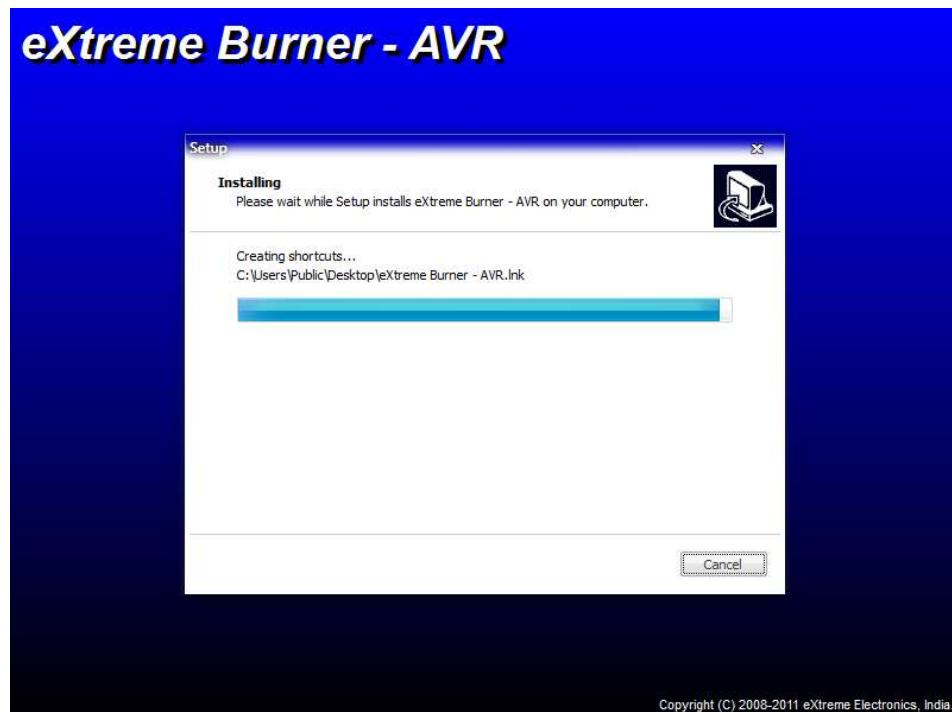


Gambar A.4 Setting directory

4. Setelah itu akan muncul beberapa pengaturan yang lain, tekan tombol “next” untuk melanjutkan ke tahap selanjutnya. Berikutnya akan muncul dialog seperti Gambar 1.5, tekan tombol “install” untuk memulai proses instalasi, dan progress instalasi ditunjukkan pada Gambar 1.6



Gambar A.5 Ready to Install



Gambar A.6 Progres instalasi

5. Ketika program berhasil di-install, maka akan muncul dialog seperti ditunjukkan pada gambar berikut :



Gambar A.7 Instalasi selesai

B. Download file*.hex dengan eXtreme Burner AVR

1. Jalankan program eXtreme Burner - AVR dengan cara melakukan klik ganda pada shortcut icon eXtreme Burner—AVR pada desktop



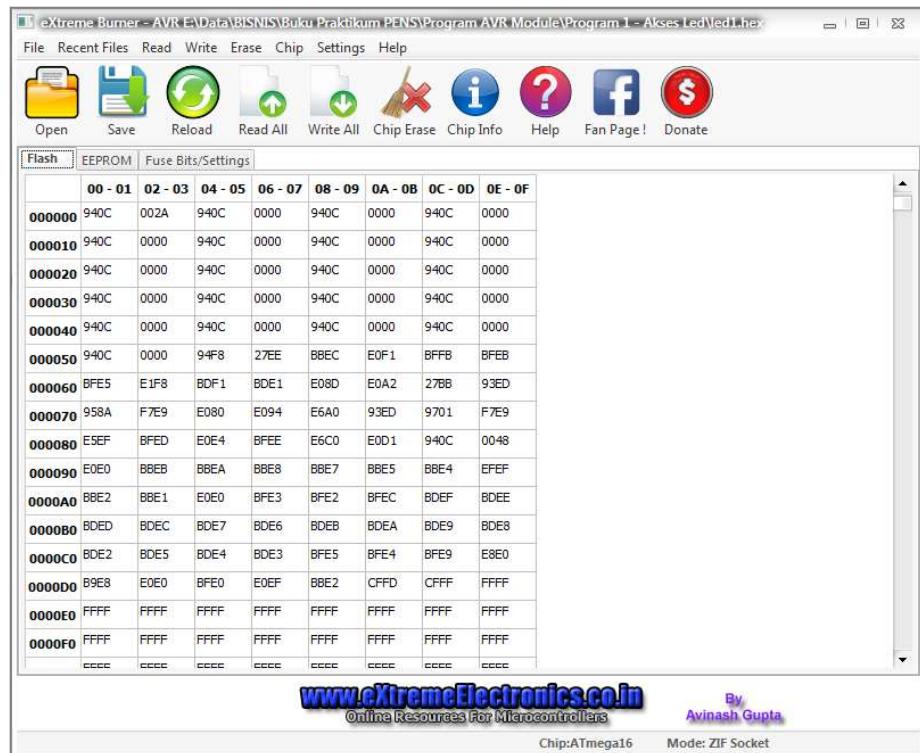
Gambar B.1 Shortcut eXtreme Burner – AVR

Setelah itu akan muncul splash screen seperti ditunjukkan pada gambar berikut:



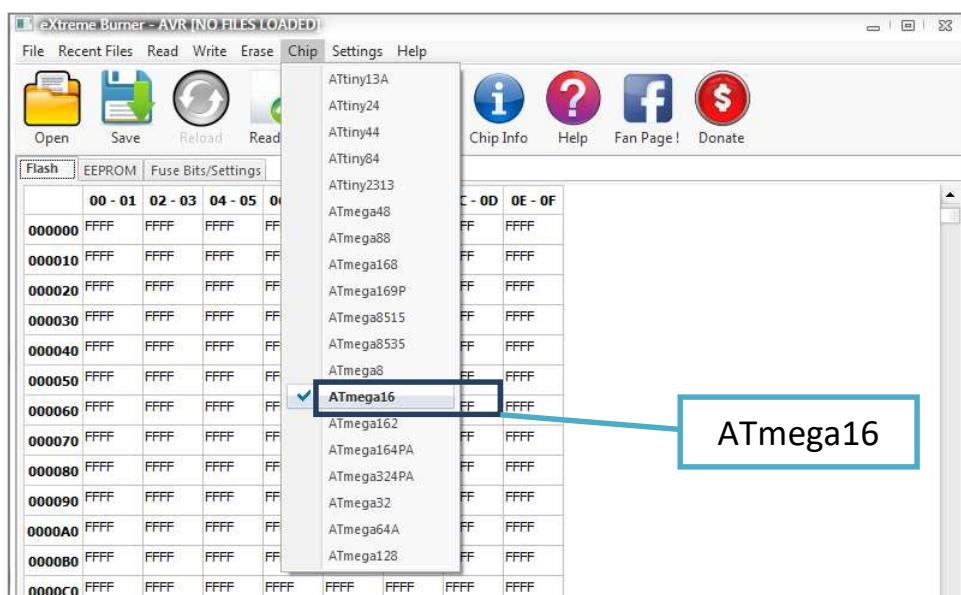
Gambar B.2 Splash screen eXtreme Burner AVR

2. Selanjutnya akan muncul tampilan utama dari program eXtreme Burner – AVR, seperti ditunjukkan pada gambar berikut:



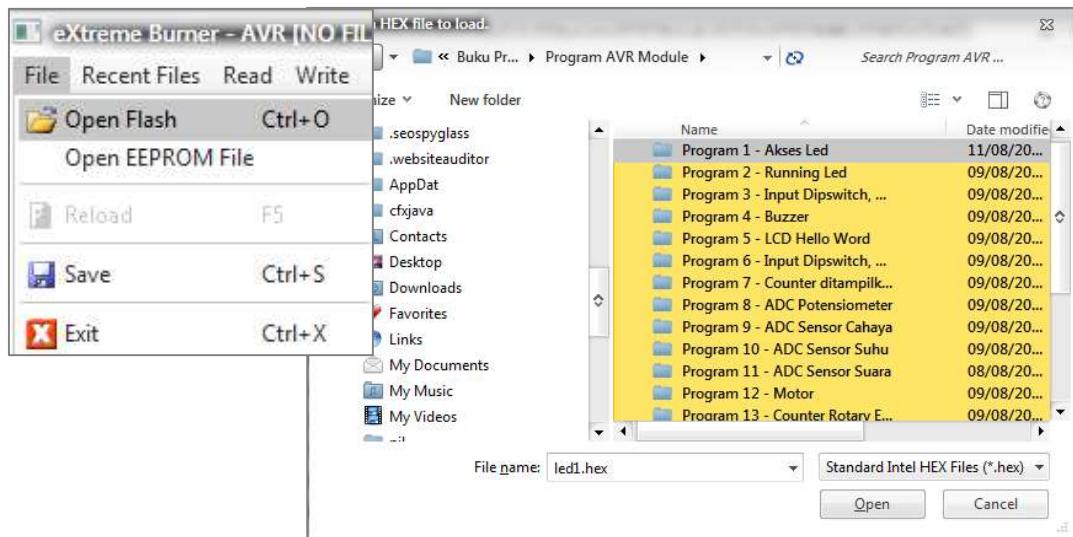
Gambar B.3 eXtreme Burner AVR

3. Pada praktikum ini menggunakan mikrokontroler keluarga AVR. Oleh karena itu, lakukan pengaturan terhadap tipe mikrokontroler terlebih dahulu dengan memilih menu Chip → <tipe_mikrokontroler> seperti dicontohkan pada gambar berikut:



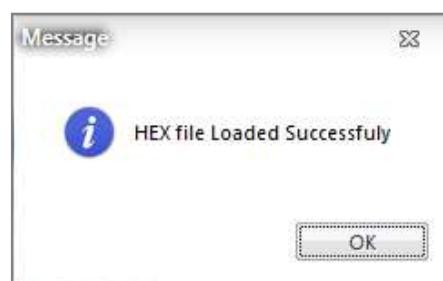
Gambar B.4 Tipe Mikrokontroler

4. Setelah itu, lakukan open file *.hex dengan memilih menu File → Open Flash (Ctrl+O) seperti ditunjukkan pada gambar berikut:



Gambar B.5 Open Flash

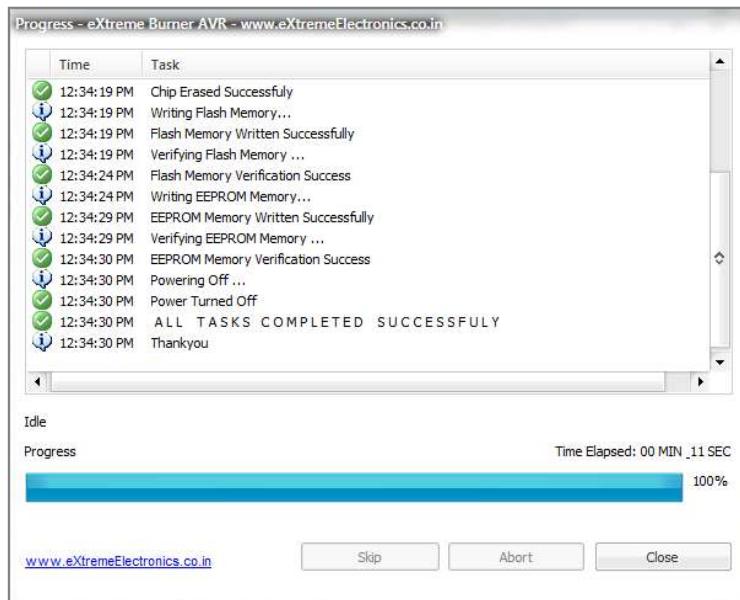
Lakukan pencarian terhadap lokasi file *.hex Anda, untuk selanjutnya tekan tombol “Open” apabila sudah ditemukan file yang dimaksud. Setelah itu akan muncul pesan yang memberitahukan bahwa file *.hex berhasil di-load program eXtreme Burner – AVR.



Gambar B.6 Message



5. Selanjutnya tekan tombol untuk memulai download ke dalam mikrokontroler Anda. Berikut ini merupakan tampilan apabila proses download ke dalam mikrokontroler berhasil dilakukan:

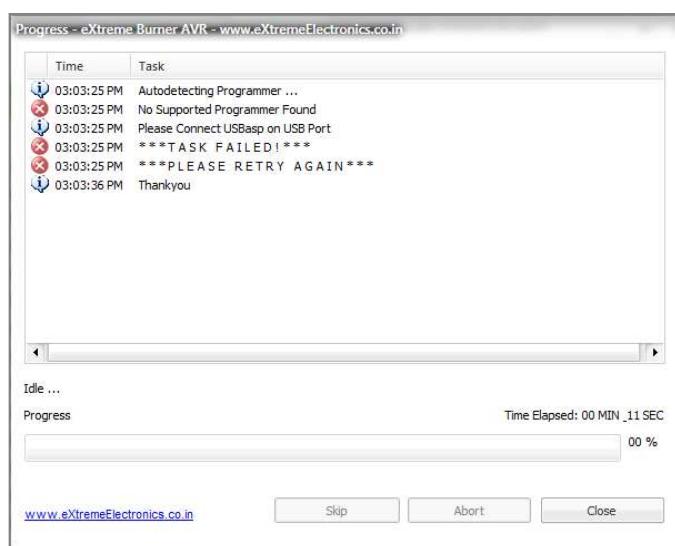


Gambar B.7 Proses Berhasil

Apabila proses gagal, maka akan muncul dialog seperti ditunjukkan pada gambar berikut:



Gambar B.8 Proses Gagal



Gambar B.9 Proses Gagal

PERCOBAAN 1

SENSOR CAHAYA

PERCOBAAN 1

SENSOR CAHAYA

1.1. Tujuan Percobaan

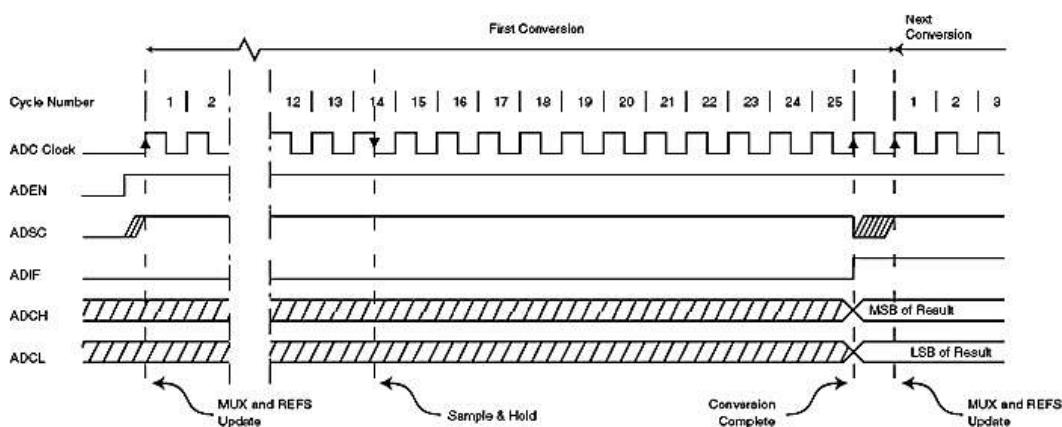
1. Mahasiswa dapat lebih memahami konsep dan cara kerja sensor cahaya
2. Mahasiswa dapat mengambil data analog sensor cahaya melalui ADC pada ATMega16
3. Mahasiswa dapat membuat aplikasi sederhana untuk membaca data ADC sensor cahaya

1.2. Dasar Teori

Mikrokontroller AVR Atmega16 mempunyai perangkat Analog To Digital Converter atau ADC 8 bit yang berguna untuk mengubah data masukan berupa tegangan analog menjadi bentuk data digital.

ATMega16 mempunyai ADC (Analog to Digital Converter) internal dengan fitur sebagai berikut (untuk lebih detil dapat mengacu ada datasheet)

- 10-bit Resolution
- 65 - 260 μ s Conversion Time
- Up to 15 kSPS at Maximum Resolution
- 8 Multiplexed Single Ended Input Channels
- Optional Left Adjustment for ADC Result Readout
- 0 - VCC ADC Input Voltage Range
- Selectable 2.56V ADC Reference Voltage
- Free Running or Single Conversion Mode
- ADC Start Conversion by Auto Triggering on
- Interrupt Sources
- Interrupt on ADC Conversion Complete
- Sleep Mode Noise Canceler



Gambar 1.1. Timer Diagram untuk Mode Single Conversion

Register-register yang dipakai untuk mengakses ADC adalah:

- **ADMUX – ADC Multiplexer Selection Register**

Bit	7	6	5	4	3	2	1	0	ADMUX
Read/Write	R/W								
Initial Value	0	0	0	0	0	0	0	0	

- **ADCSRA – ADC Control and Status Register**

Bit	7	6	5	4	3	2	1	0	ADCSRA
Read/Write	R/W								
Initial Value	0	0	0	0	0	0	0	0	

- **ADCL, ADCH – ADC data register**

Bila ADCLR = 0

Bit	15	14	13	12	11	10	9	8	ADCH
	-	-	-	-	-	-	ADC9	ADC8	ADCL
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	
	7	6	5	4	3	2	1	0	

Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

Bila ADCLR = 1

Bit	15	14	13	12	11	10	9	8	ADCH
	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCL
	ADC1	ADC0	-	-	-	-	-	-	
	7	6	5	4	3	2	1	0	

Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

Setelah ADC selesai melakukan konversi kedua register ini berisi hasil konversi. Bila channel differensial dipilih maka hasilnya dalam format two's complement. Saat ADCL dibaca, data register tidak akan meng-update data sampai ADCH dibaca. Jika hasilnya dirata kiri (left adjust) dan hanya butuh 8-bit maka cukuplah dengan membaca ADCH. Jika butuh 10-bit, baca ADCL dahulu kemudian ADCH.

- **SFIOR – Special Function I/O Register untuk sumber auto trigger**

Bit	7	6	5	4	3	2	1	0	SFIOR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Berikut adalah langkah-langkah yang harus dilakukan untuk melakukan konversi mode single-conversion pada ADC internal dari Atmega16:

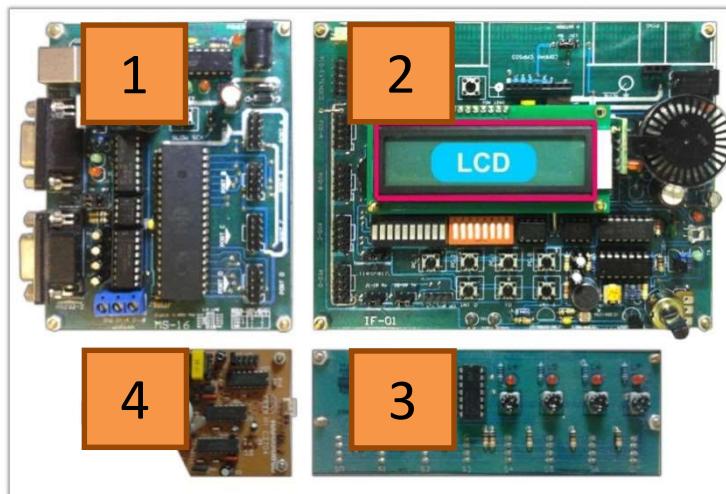
1. Seting register ADMUX.
2. Seting register ADCSRA.
3. Seting register SFIOR (untuk nomor 1, 2, dan 3 dilakukan oleh CodeWizardAVR).
4. Pilih channel ADC pada register ADMUX.
5. Start konversi ADC dengan cara mengaktifkan bit ADSC pada register ADCSRA.
6. Bila ADC selesai melakukan konversi, program akan meloncat ke layanan interupsi ADC.
7. Ambil data pada register ADCH.
8. Tampilkan nilai pada LCD.
9. Delay sekitar 500 milidetik, agar tampilan pada LCD dapat dilihat.
10. Jika diinginkan, ulangi langkah 4.

1.3. Peralatan

- 1 set PC yang dilengkapi dengan software CodeVision AVR.
- 1 set development board AVR ATmega16
- 1 power-supply +9VDC

1.4. Modul I/O

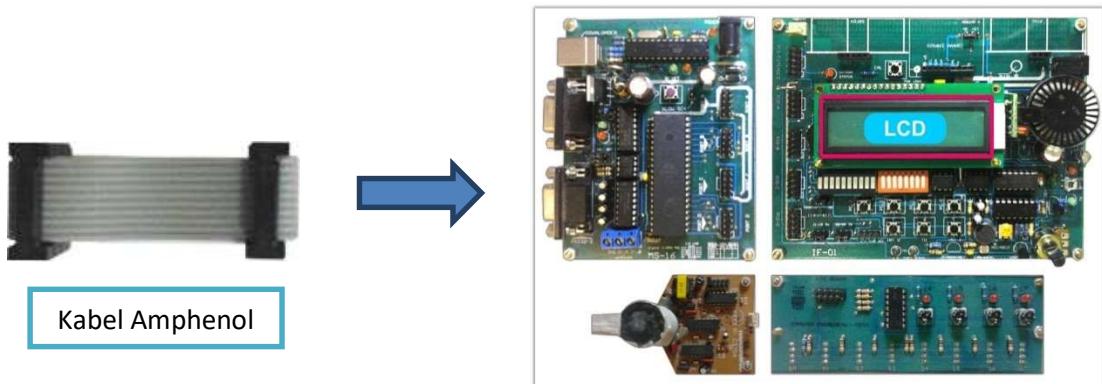
Berikut ini merupakan development board AVR ATmega16 secara keseluruhan:



Gambar 1.2 Development board keseluruhan

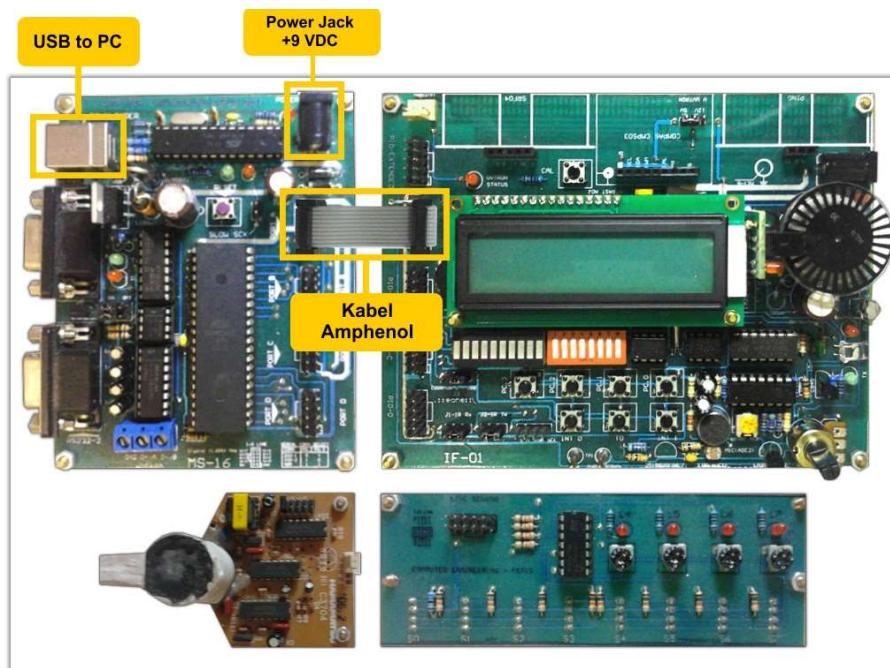
- **Konfigurasi modul pada percobaan ini :**

Pasangkan kabel amphenol pada modul development untuk menghubungkan sistem minimum ATMega16 dengan Training Board.



Gambar 1.3. Komponen Training board

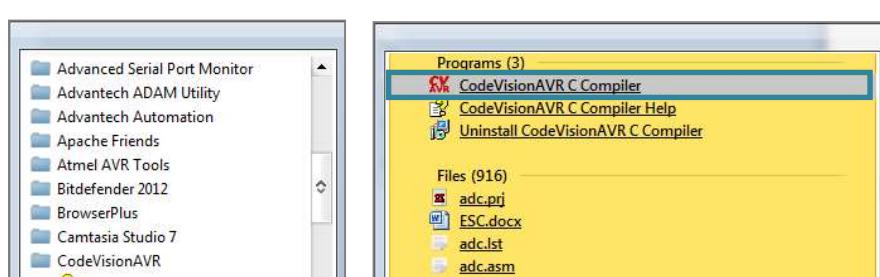
Sehingga menjadi seperti ditunjukkan pada gambar berikut:



Gambar 1.4 Komponen Training board

1.5. Prosedur Praktikum

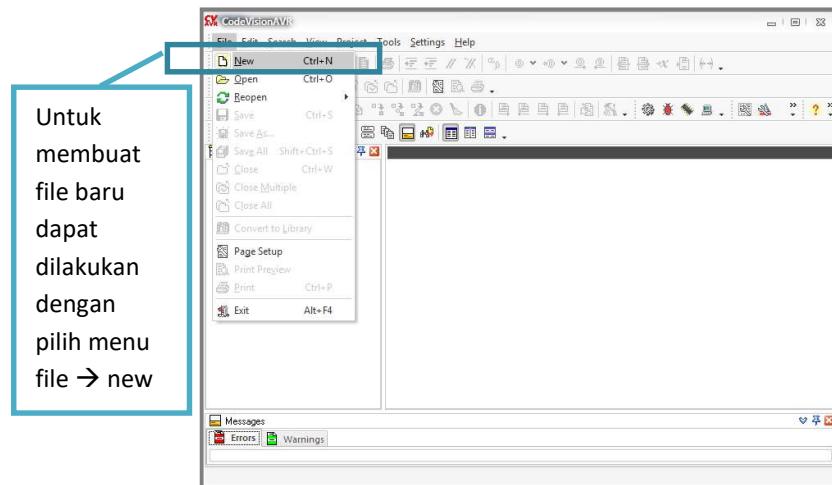
1. Jalankan aplikasi CodeVisionAVR dengan cara melakukan klik ganda pada shortcut icon CodeVisionAVR pada desktop atau dengan cara mencari shortcut program pada start menu system operasi windows.





Gambar 1.5 Shortcut CodeVisionAVR

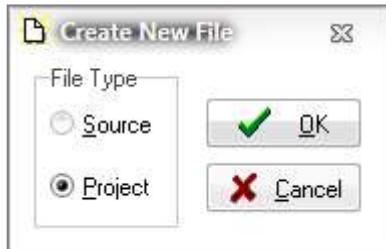
Untuk memulai membuat project baru, pada menubar, pilih File → New, seperti yang ditunjukkan oleh gambar berikut:



Gambar 1.6 Membuat File baru

Setelah itu akan muncul suatu dialog yang mengharuskan *user* untuk memilih antara pembuatan source file atau project. Dalam setiap percobaan pada praktikum ini, dibuat project baru untuk setiap percobaan, oleh karena itu *user*

sebaiknya melakukan *check* pada checkbox project dan dilanjutkan dengan menekan tombol “OK”



Gambar 1.7 Membuat project baru

Berikutnya akan keluar dialog yang menanyakan Anda apakah akan membuat project baru. Klik tombol “Yes” untuk menyetujui.

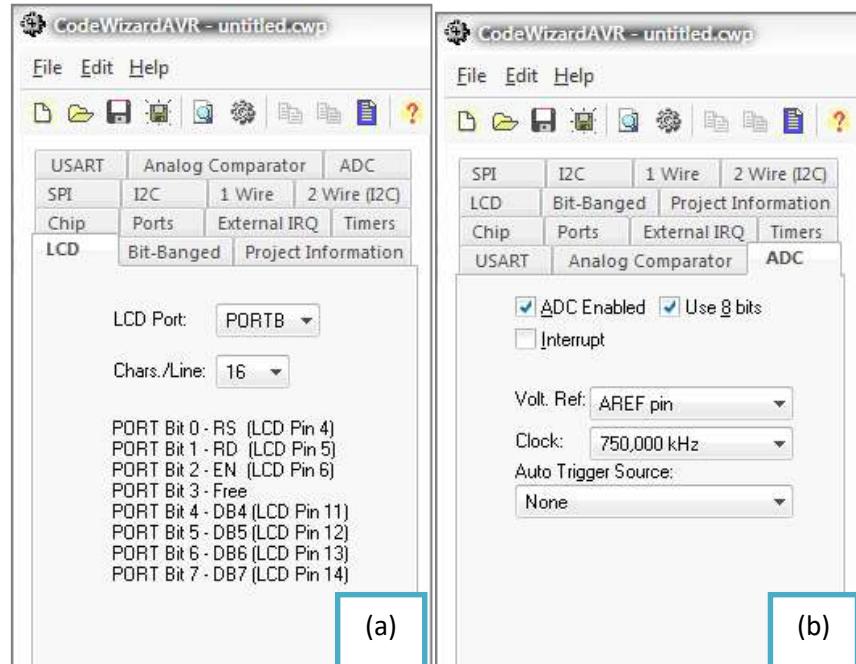


Gambar 1.8 Memilih untuk membuat project baru pada CodeVisionAVR

Setelah itu lakukan set pada CodeWizardAVR. Untuk chip diset dengan nama chip ATmega16 dengan clock 12 MHz.

Berikutnya Anda akan menginisialisasi LCD pada PortB, yaitu dengan cara masuk pada tab LCD dan pilih item **PORTB** pada selectbox “LCD Port”. Set banyaknya karakter dari LCD yang diinginkan. Dalam hal ini menggunakan 16 karakter, oleh karena itu pilih item **16** pada selectbox “Chars/Line”.

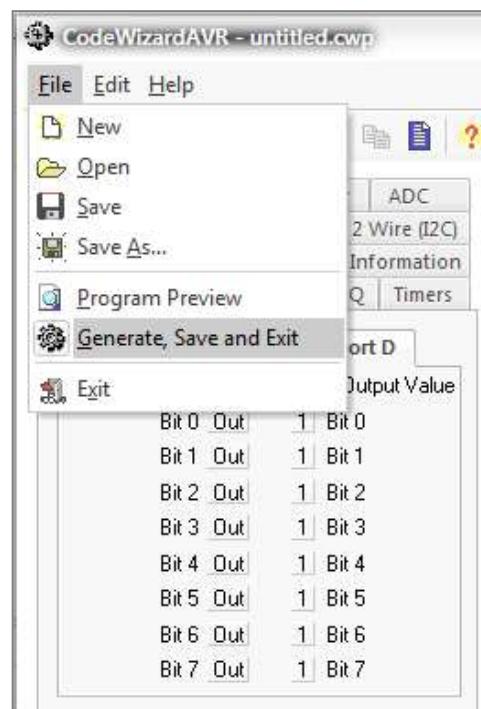
Setelah itu lakukan set ADC dengan masuk pada tab ADC. Berikan tanda check terhadap checkbox “ADC Enabled” untuk meng-enable ADC. Berikan tanda check terhadap checkbox “Use 8 bits” untuk menggunakan ADC 8 bit.



Gambar 1.9(a) Setting LCD pada PortB

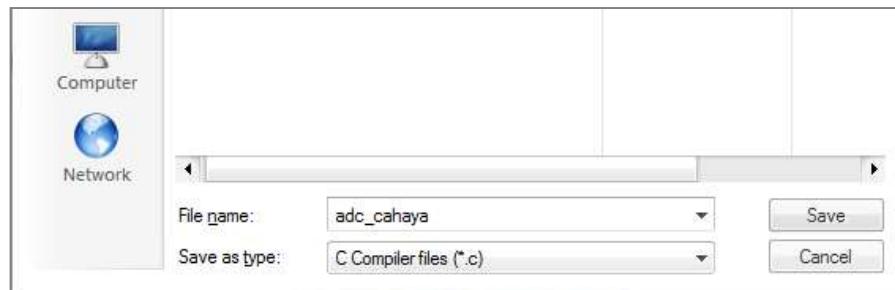
(b) Setting ADC 8 bit

Karena pada contoh ini tidak digunakan fasilitas lain maka setting CodeWizardAVR siap disimpan dalam file. Pada menu CodeWizardAVR, pilih File → Generate, Save and Exit, seperti yang ditunjukkan pada gambar berikut:



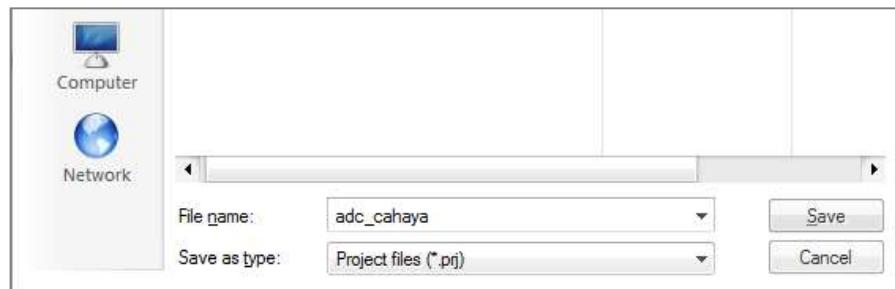
Gambar 1.10 Menyimpan setting

Setelah penekanan menu “Generate, Save and Exit”, akan muncul suatu dialog untuk melakukan set terhadap nama dan lokasi penyimpanan dari source file (*.c) seperti ditunjukkan pada gambar berikut:



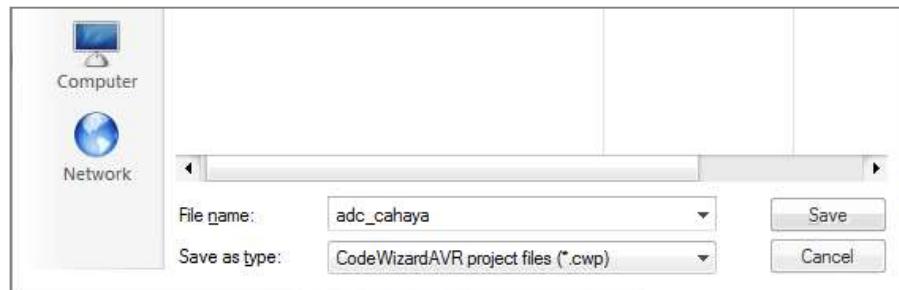
Gambar 1.11 Memberi nama pada file C (*.c)

Tekan tombol “Save” untuk menyetujui dan melanjutkan pada proses berikutnya. Setelah penekanan tombol “Save” akan muncul kembali dialog yang kedua, yaitu untuk melakukan set terhadap nama dan lokasi penyimpanan project file (*.prj) seperti ditunjukkan pada gambar berikut:



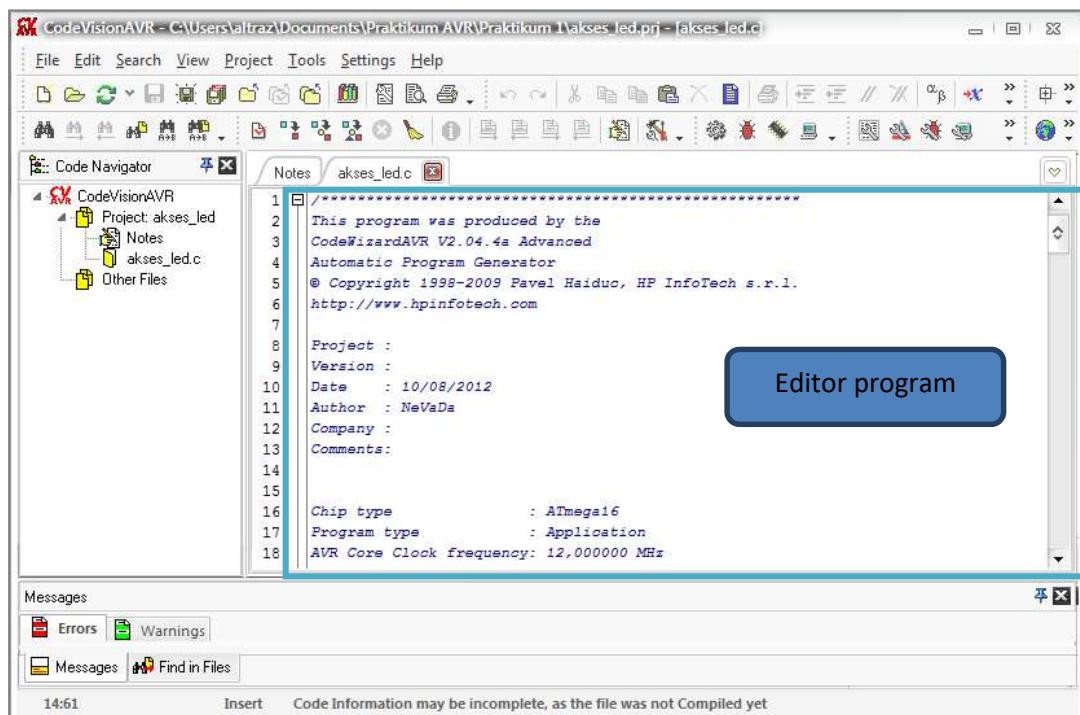
Gambar 1.12 Memberi nama project (*.prj)

Tekan tombol “Save” untuk menyetujui dan melanjutkan pada proses berikutnya. Setelah penekanan tombol “Save” akan muncul kembali dialog yang ketiga, yaitu untuk melakukan set terhadap nama dan lokasi penyimpanan CodeWizardProject file (*.cwp) seperti ditunjukkan pada gambar berikut:



Gambar 1.13 Memberi nama CodeWizardProject (*.cwp)

Tekan tombol “Save” untuk menyetujui dan melanjutkan. Selanjutnya akan muncul IDE editor program dari CodeVisionAVR. Anda siap untuk melakukan pembuatan program menggunakan CodeVisionAVR.



Gambar 1.14 Editor Program

2. Pada program-C yang dihasilkan, tambahkan header `stdio.h` dan `delay.h`
3. Lakukan pendeklarasian variable global, letakkan di atas fungsi main(void)

```
// Declare your global variables here
unsignedchar xstring[3],data_cahaya;
void main(void) { ... }
```

4. Pada bagian while(1) {...} tambahkan perintah yang telah diblok dengan warna kuning berikut :

```

while (1)
{
    // Place your code here
    lcd_clear();
        lcd_putstr("Data cahaya = ");
        data_cahaya = read_adc(1);
        sprintf(xstring, "%d", data_cahaya);
        lcd_puts(xstring);
        delay_ms(100);
}

```

5. Untuk lebih jelasnya, berikut adalah kode program secara keseluruhan

```

#include<mega16.h>
#include<stdio.h>

// Alphanumeric LCD Module functions
#asm
    .equ __lcd_port=0x18 ;PORTB
#endif
#include<lcd.h>

#include<delay.h>

#define ADC_VREF_TYPE 0x20

// Read the 8 most significant bits
// of the AD conversion result
unsignedchar read_adc(unsignedchar adc_input)
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Delay needed for the stabilization of the ADC input voltage
delay_us(10);
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCH;
}

// Declare your global variables here
unsignedchar xstring[3],data_cahaya;
void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
}

```

```

State0=T
PORTA=0x00;
DDRA=0x00;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTB=0x00;
DDRB=0x00;

// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTD=0x00;
DDRD=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

```

```

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

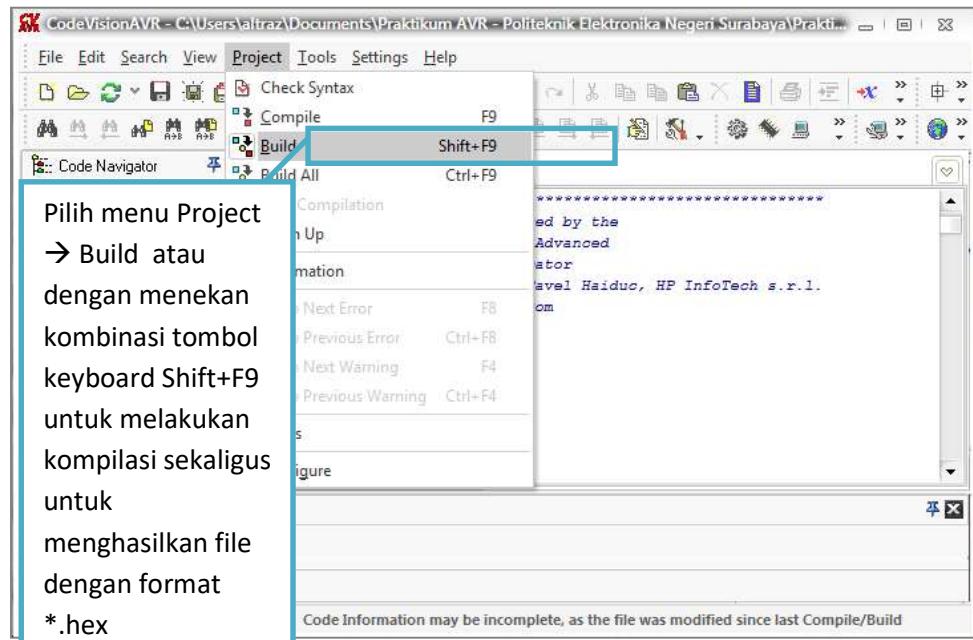
// ADC initialization
// ADC Clock frequency: 691,200 kHz
// ADC Voltage Reference: AREF pin
// ADC Auto Trigger Source: None
// Only the 8 most significant bits of
// the AD conversion result are used
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x84;

// LCD module initialization
lcd_init(16);

while (1)
{
    // Place your code here
    lcd_clear();
    lcd_putsf("Data cahaya = ");
    data_cahaya = read_adc(1);           //AVR Module V1---
>read_adc(2) , AVR Module V2--->read_adc(1)
    sprintf(xstring,"%d",data_cahaya);
    lcd_puts(xstring);
    delay_ms(100);
}

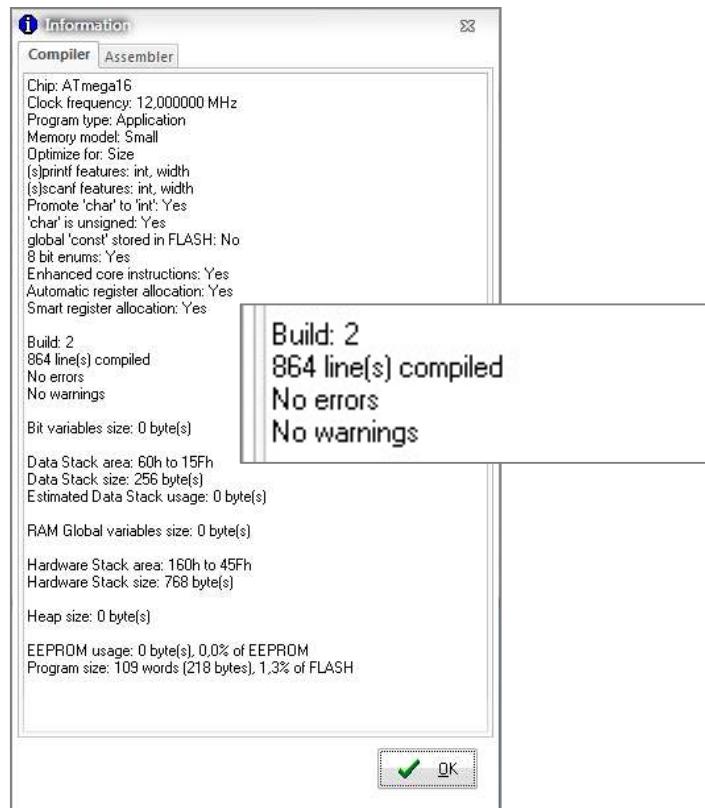
```

6. Setelah pembuatan program selesai, maka proses selanjutnya yaitu lakukan kompilasi terhadap kode program tersebut sekaligus untuk menghasilkan file dengan format hexadecimal (*.hex). Hal ini dapat dilakukan dengan memilih menu Project → Build (Shift+F9) seperti ditunjukkan pada gambar berikut :



Gambar 1.15 Build Source

Apabila kode program sudah benar (tanpa error) dan tidak menyalahi struktur pemrograman bahasa C, maka akan muncul dialog informasi seperti ditunjukkan pada gambar berikut:



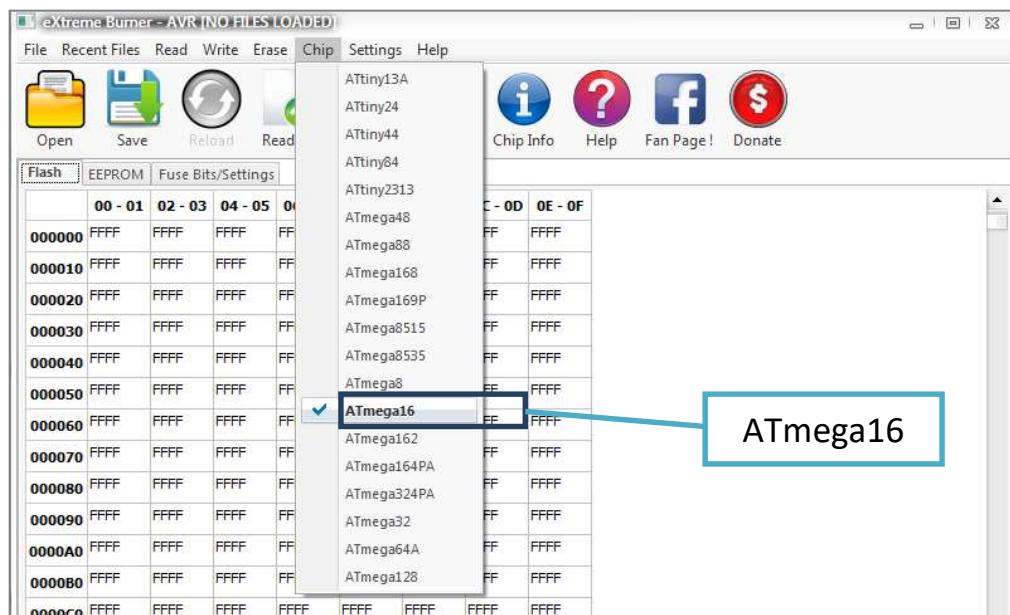
Gambar 1.16 Dialog informasi status kompilasi

7. Setelah diperoleh file dengan format *.hex, maka lakukan download file *.hex ke dalam mikrokontroler. Pertama, jalankan program eXtreme Burner - AVR dengan cara melakukan klik ganda pada shortcut icon eXtreme Burner– AVR pada desktop



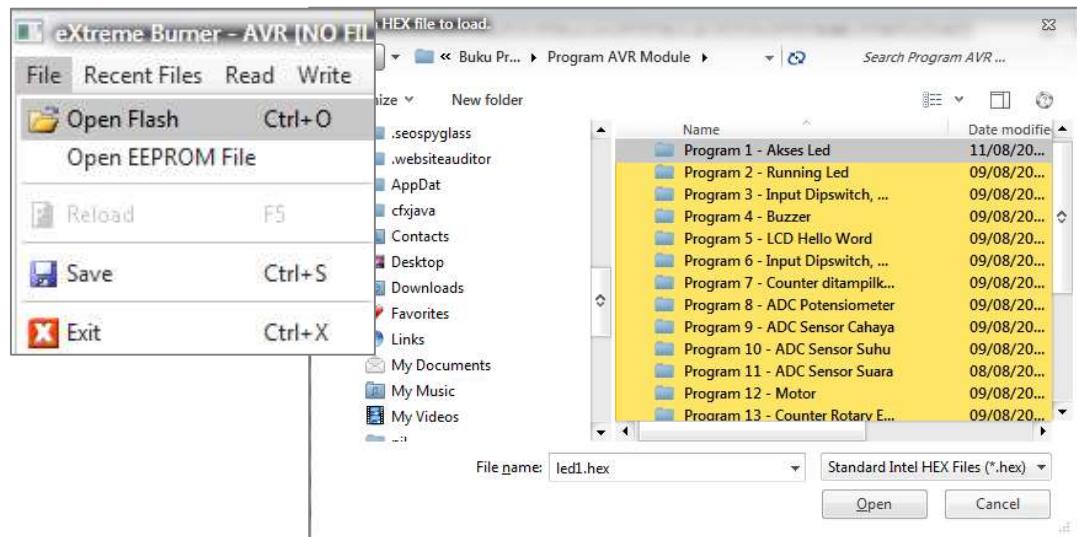
Gambar 1.17 Shortcut eXtreme Burner - AVR

Selanjutnya akan muncul tampilan utama dari program eXtreme Burner – AVR. Pada praktikum ini menggunakan mikrokontroler keluarga AVR dengan tipe IC ATMega16. Oleh karena itu, lakukan pengaturan terhadap tipe IC terlebih dahulu dengan memilih menu Chip → ATmega16 seperti ditunjukkan pada gambar berikut:



Gambar 1.18 Tipe Mikrokontroler

Setelah itu, lakukan open file *.hex dengan memilih menu File → Open Flash (Ctrl+O) seperti ditunjukkan pada gambar berikut:



Gambar 1.19 Open Flash

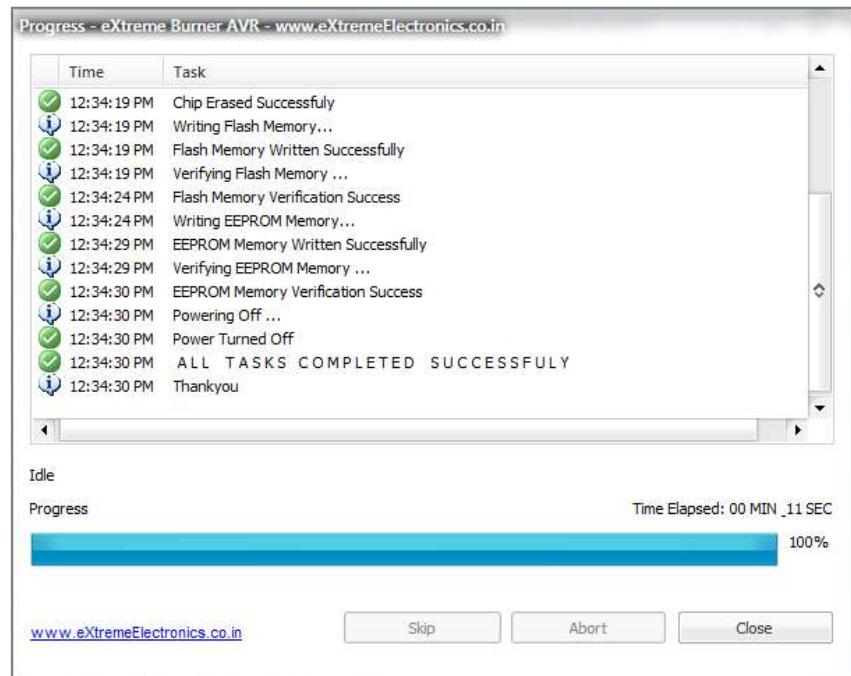
Lakukan pencarian terhadap lokasi file *.hex Anda, untuk selanjutnya tekan tombol “Open” apabila sudah ditemukan file yang dimaksud. Setelah itu akan muncul pesan yang memberitahukan bahwa file *hex berhasil di-load program eXtreme Burner – AVR.



Gambar 1.20 Message



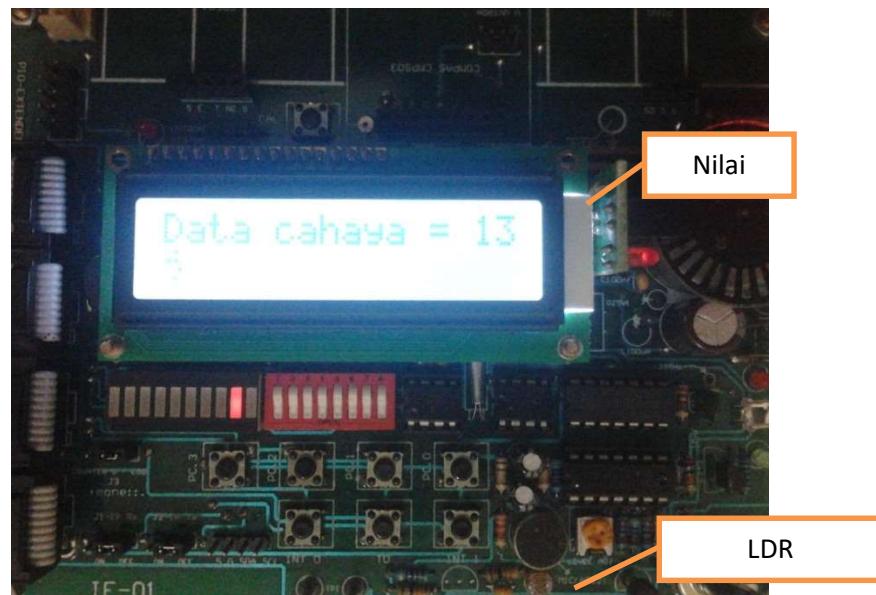
Selanjutnya tekan tombol untuk memulai download ke dalam mikrokontroler Anda. Berikut ini merupakan tampilan apabila proses download ke dalam mikrokontroler berhasil dilakukan:



Gambar 1.21 Proses Berhasil

8. Selesai

1.6. Hasil Percobaan



Gambar 1.22 Running Program

Tugas Praktikum : buat percobaan proyek lampu taman otomatis berdasarkan tingkat kecerahan cahaya !

PERCOBAAN 2

SENSOR SUHU : LM35

PERCOBAAN 2

SENSOR SUHU : LM35

2.1. Tujuan

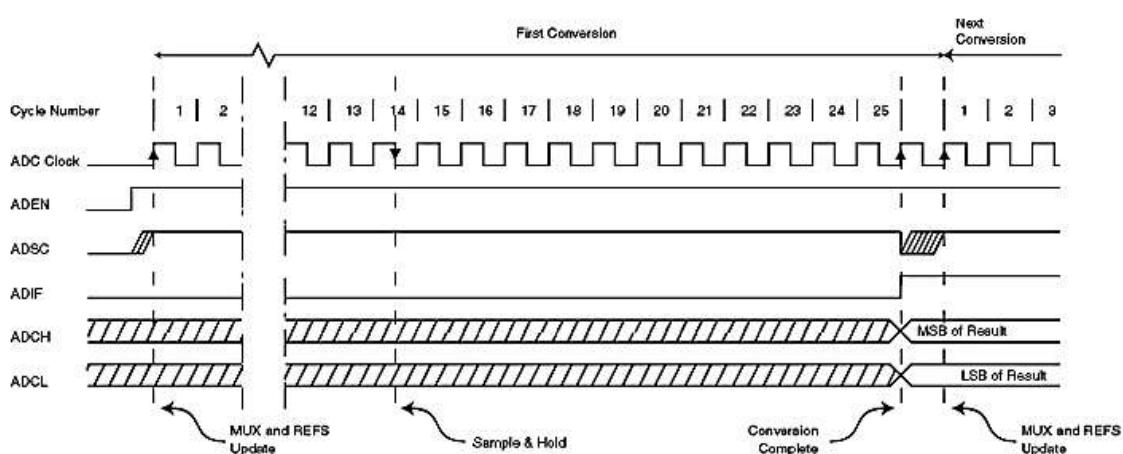
1. Mahasiswa dapat lebih memahami cara kerja sensor suhu LM35.
2. Mahasiswa dapat mengakses ADC dari sensor suhu pada ATMega16
3. Mahasiswa dapat membuat aplikasi ADC sederhana yang memanfaatkan sensor suhu

2.2. Dasar Teori

Mikrokontroller AVR Atmega16 mempunyai perangkat Analog To Digital Converter atau ADC 8 bit yang berguna untuk mengubah data masukan berupa tegangan analog menjadi bentuk data digital.

ATMega16 mempunyai ADC (Analog to Digital Converter) internal dengan fitur sebagai berikut (untuk lebih detil dapat mengacu ada datasheet)

- 10-bit Resolution
- 65 - 260 μ s Conversion Time
- Up to 15 kSPS at Maximum Resolution
- 8 Multiplexed Single Ended Input Channels
- Optional Left Adjustment for ADC Result Readout
- 0 - VCC ADC Input Voltage Range
- Selectable 2.56V ADC Reference Voltage
- Free Running or Single Conversion Mode
- ADC Start Conversion by Auto Triggering on
- Interrupt Sources
- Interrupt on ADC Conversion Complete
- Sleep Mode Noise Canceler



Gambar 2.1. Timer Diagram untuk Mode Single Conversion

Register-register yang dipakai untuk mengakses ADC adalah:

- **ADMUX – ADC Multiplexer Selection Register**

Bit	7	6	5	4	3	2	1	0	ADMUX
Read/Write	R/W								
Initial Value	0	0	0	0	0	0	0	0	

- **ADCSRA – ADC Control and Status Register**

Bit	7	6	5	4	3	2	1	0	ADCSRA
Read/Write	R/W								
Initial Value	0	0	0	0	0	0	0	0	

- **ADCL, ADCH – ADC data register**

Bila ADCLR = 0

Bit	15	14	13	12	11	10	9	8	ADCH
	-	-	-	-	-	-	ADC9	ADC8	ADCL
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	
	7	6	5	4	3	2	1	0	

Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

Bila ADCLR = 1

Bit	15	14	13	12	11	10	9	8	ADCH
	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCL
	ADC1	ADC0	-	-	-	-	-	-	
	7	6	5	4	3	2	1	0	

Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

Setelah ADC selesai melakukan konversi kedua register ini berisi hasil konversi. Bila channel differensial dipilih maka hasilnya dalam format two's complement. Saat ADCL dibaca, data register tidak akan meng-update data sampai ADCH dibaca. Jika hasilnya dirata kiri (left adjust) dan hanya butuh 8-bit maka cukuplah dengan membaca ADCH. Jika butuh 10-bit, baca ADCL dahulu kemudian ADCH.

- **SFIOR – Special Function I/O Register untuk sumber auto trigger**

Bit	7	6	5	4	3	2	1	0	SFIOR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Berikut adalah langkah-langkah yang harus dilakukan untuk melakukan konversi mode single-conversion pada ADC internal dari Atmega16:

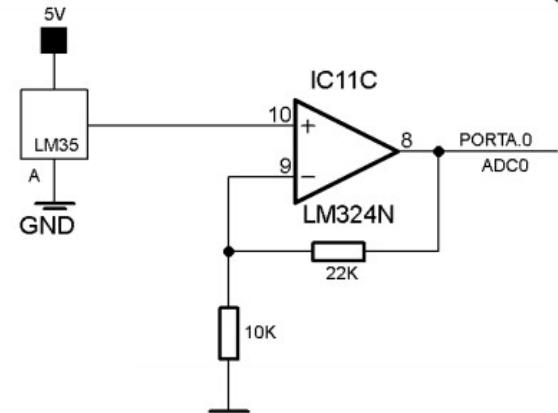
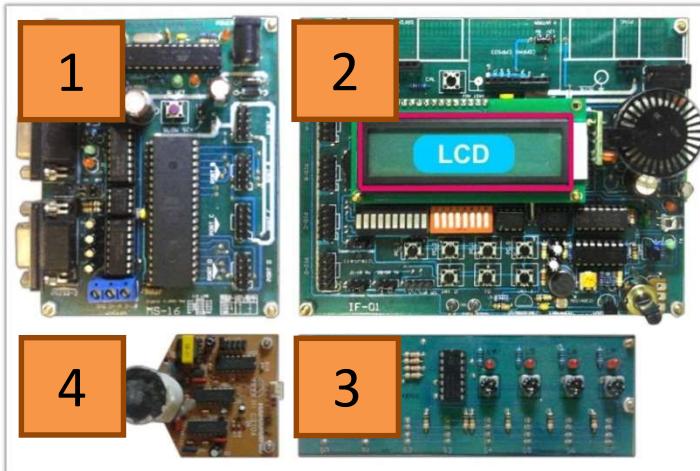
1. Seting register ADMUX.
2. Seting register ADCSRA.
3. Seting register SFIOR (untuk nomor 1, 2, dan 3 dilakukan oleh CodeWizardAVR).
4. Pilih channel ADC pada register ADMUX.
5. Start konversi ADC dengan cara mengaktifkan bit ADSC pada register ADCSRA.
6. Bila ADC selesai melakukan konversi, program akan meloncat ke layanan interupsi ADC.
7. Ambil data pada register ADCH.
8. Tampilkan nilai pada LCD.
9. Delay sekitar 500 milidetik, agar tampilan pada LCD dapat dilihat.
10. Jika diinginkan, ulangi langkah 4.

2.3. Peralatan

- 1 set PC yang dilengkapi dengan software CodeVision AVR.
- 1 set development board AVR ATmega16
- 1 power-supply +9VDC

2.4. Modul I/O

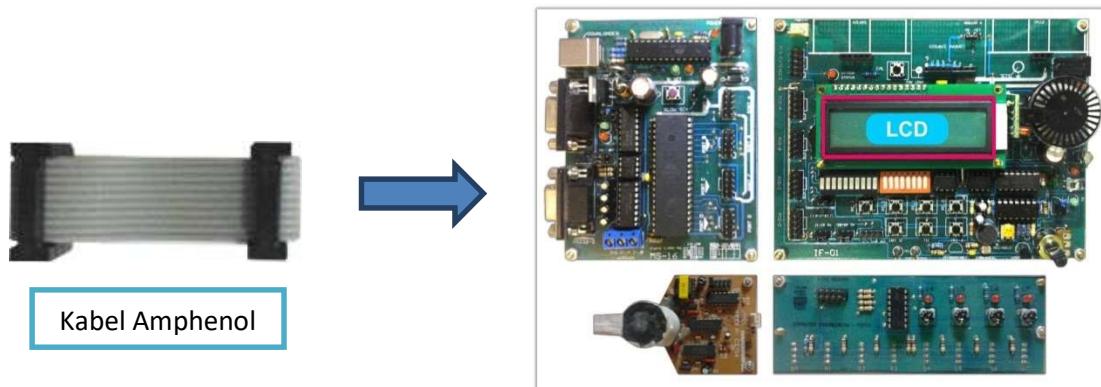
Berikut ini merupakan rangkaian sensor suhu dan development board AVR ATmega16:



Gambar 2.2 Development board dan rangkaian sensor suhu

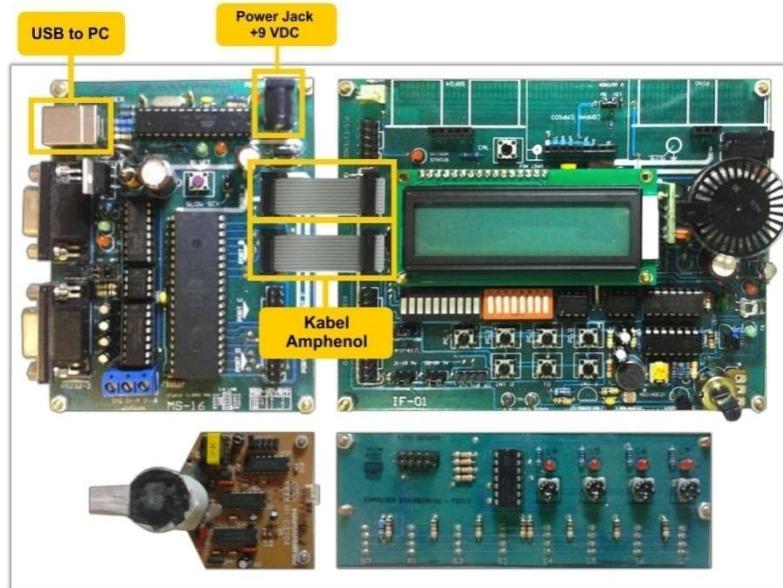
- **Konfigurasi modul pada percobaan ini :**

Pasangkan kabel amphenol pada modul development untuk menghubungkan system minimum ATMega16 dengan Training Board.



Gambar 2.3 Komponen Training board

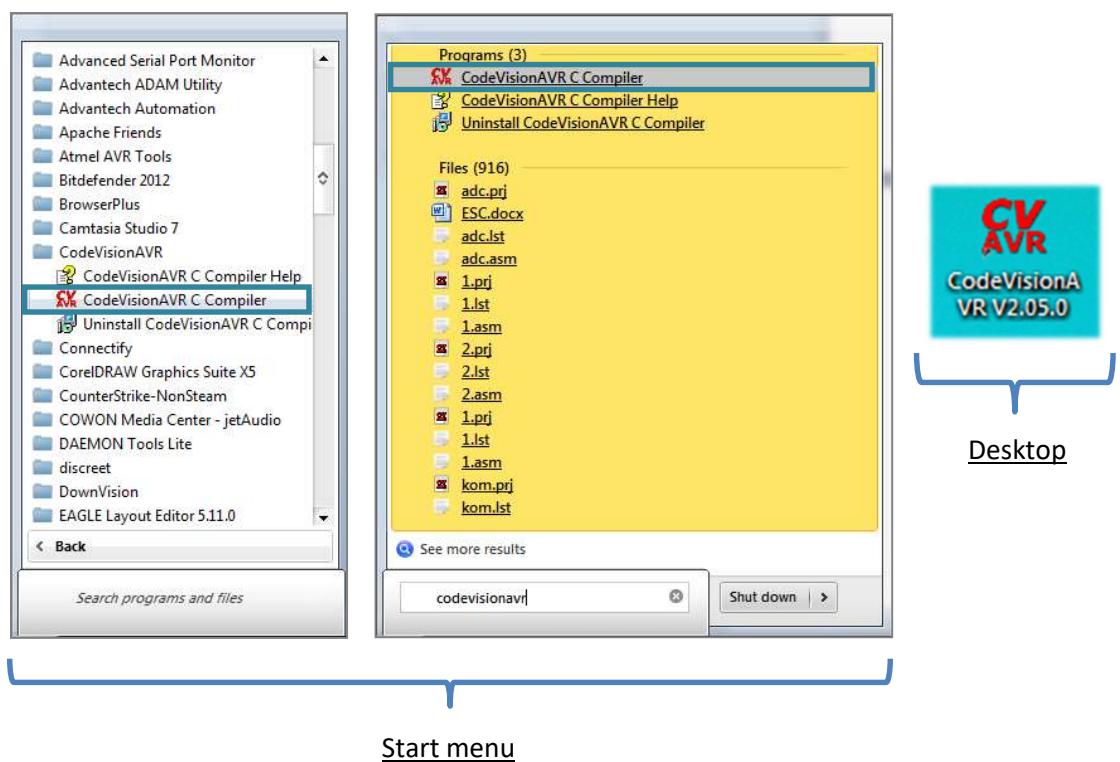
Sehingga menjadi seperti ditunjukkan pada gambar berikut:



Gambar 2.4 Komponen Training board

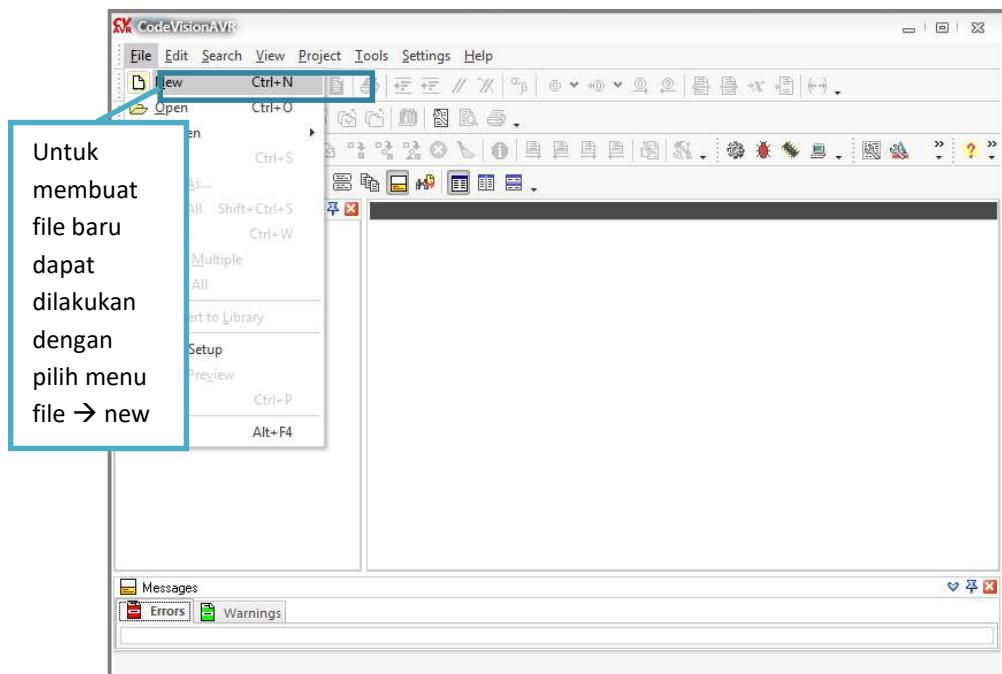
2.5. Prosedur Praktikum

1. Jalankan aplikasi CodeVisionAVR dengan cara melakukan klik ganda pada shortcut icon CodeVisionAVR pada desktop atau dengan cara mencari shortcut program pada start menu system operasi windows.



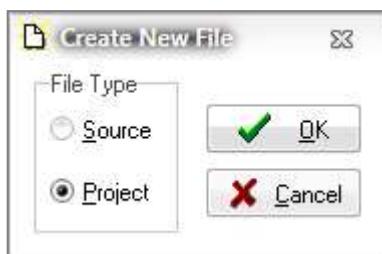
Gambar 2.5 Shortcut CodeVisionAVR

Untuk memulai membuat project baru, pada menubar, pilih File → New, seperti yang ditunjukkan oleh gambar berikut:



Gambar 2.6 Membuat File baru

Setelah itu akan muncul suatu dialog yang mengharuskan *user* untuk memilih antara pembuatan source file atau project. Dalam setiap percobaan pada praktikum ini, dibuat project baru untuk setiap percobaan, oleh karena itu *user* sebaiknya melakukan *check* pada checkbox project dan dilanjutkan dengan menekan tombol “OK”



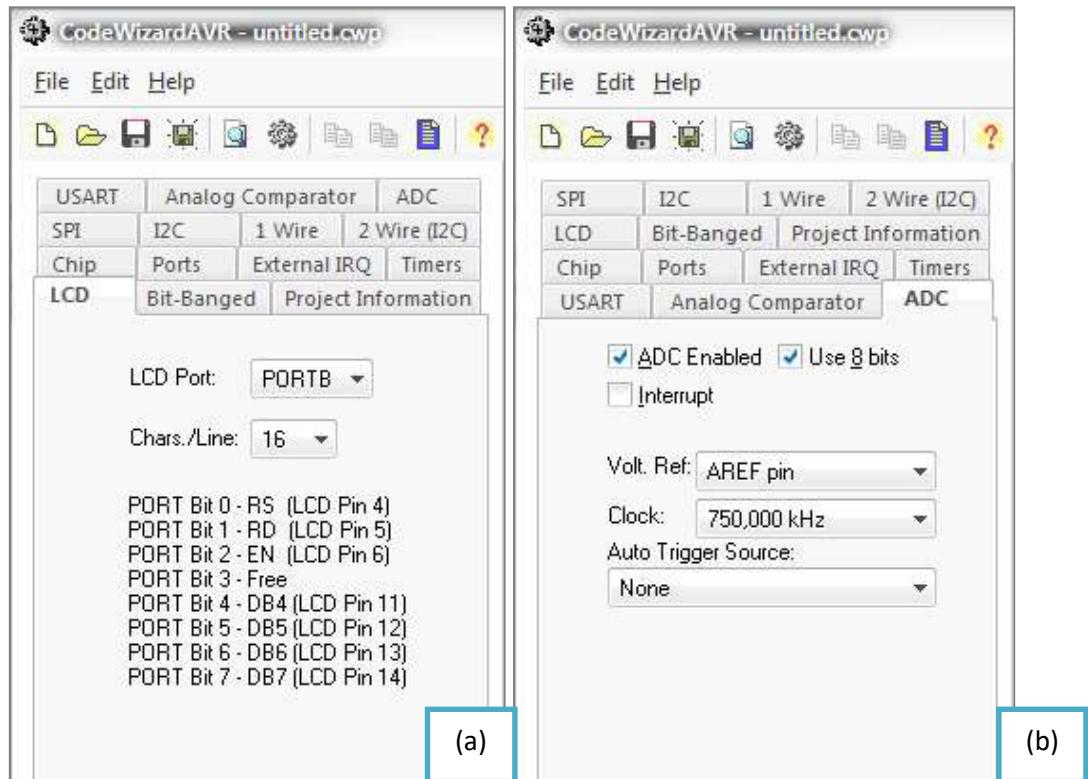
Gambar 2.7 Membuat project baru

Berikutnya akan keluar dialog yang menanyakan Anda apakah akan membuat project baru. Klik tombol “Yes” untuk menyetujui.



Gambar 2.8 Memilih untuk membuat project baru pada CodeVisionAVR

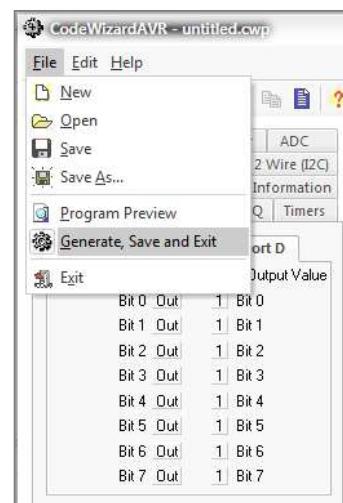
Setelah itu lakukan set pada CodeWizardAVR. Untuk chip diset dengan nama chip ATmega16 dengan clock 12 MHz. Berikutnya Anda akan menginisialisasi LCD pada PortB, yaitu dengan cara masuk pada tab LCD dan pilih item **PORTB** pada selectbox “LCD Port”. Set banyaknya karakter dari LCD yang diinginkan. Dalam hal ini menggunakan 16 karakter, oleh karena itu pilih item **16** pada selectbox “Chars/Line”. Setelah itu lakukan set ADC dengan masuk pada tab ADC. Berikan tanda check terhadap checkbox “ADC Enabled” untuk meng-enable ADC. Berikan tanda check terhadap checkbox “Use 8 bits” untuk menggunakan ADC 8 bit.



Gambar 2.9(a) Setting LCD pada PortB

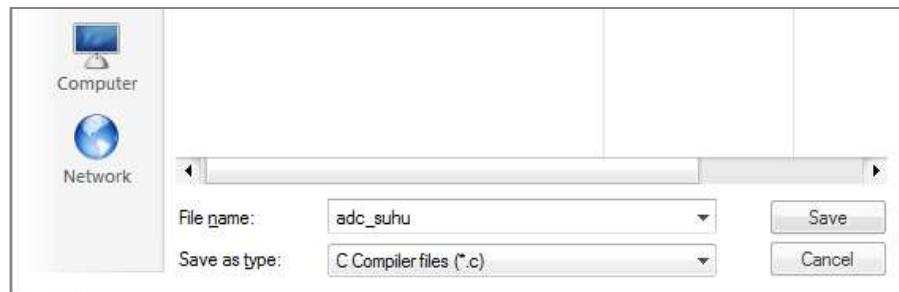
(b) Setting ADC 8 bit

Karena pada contoh ini tidak digunakan fasilitas lain maka setting CodeWizardAVR siap disimpan dalam file. Pada menu CodeWizardAVR, pilih File → Generate, Save and Exit, seperti yang ditunjukkan pada gambar berikut:



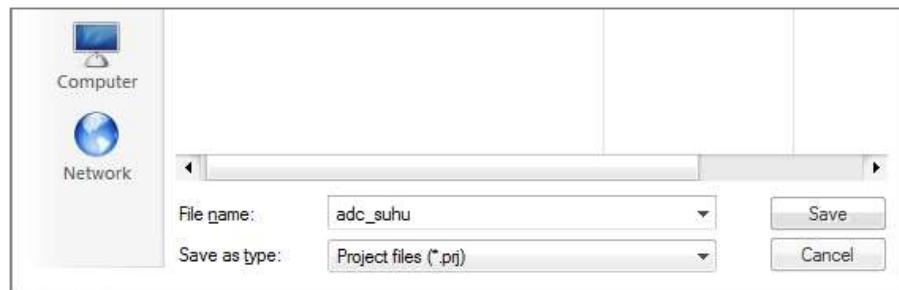
Gambar 2.10 Menyimpan setting

Setelah penekanan menu “Generate, Save and Exit”, akan muncul suatu dialog untuk melakukan set terhadap nama dan lokasi penyimpanan dari source file (*.c) seperti ditunjukkan pada gambar berikut:



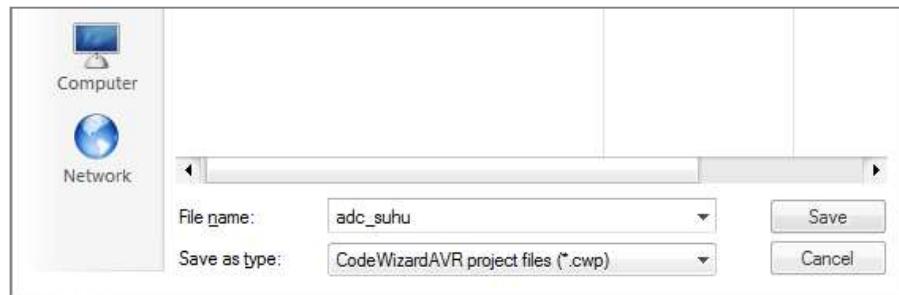
Gambar 2.11 Memberi nama pada file C (*.c)

Tekan tombol “Save” untuk menyetujui dan melanjutkan pada proses berikutnya. Setelah penekanan tombol “Save” akan muncul kembali dialog yang kedua, yaitu untuk melakukan set terhadap nama dan lokasi penyimpanan project file (*.prj) seperti ditunjukkan pada gambar berikut:



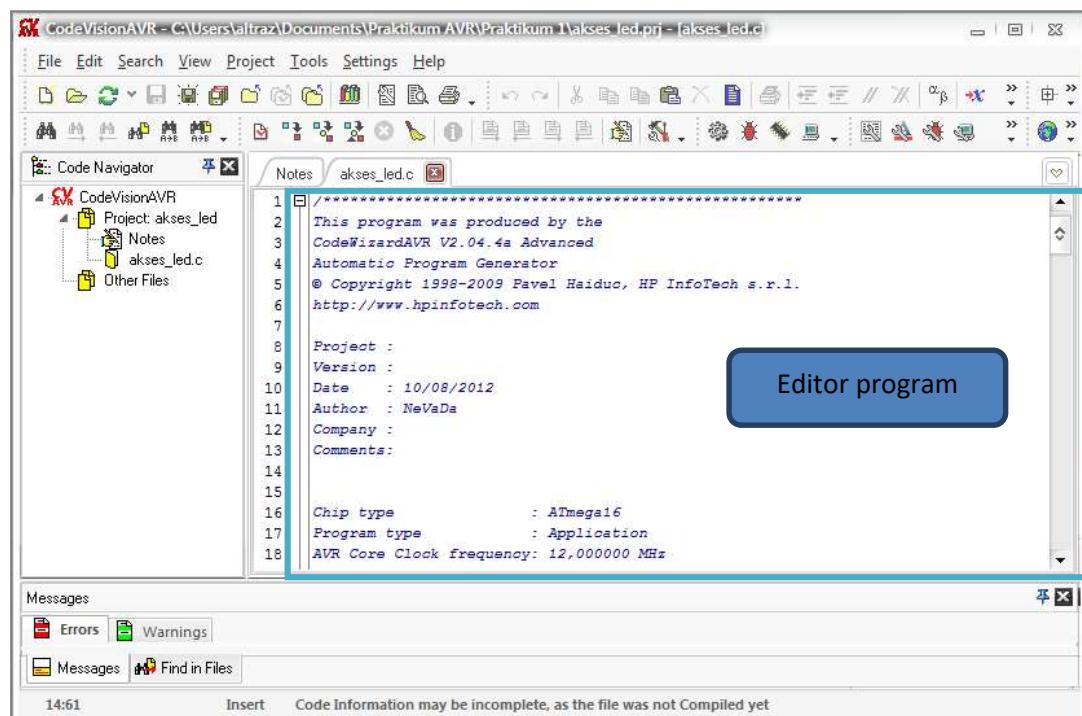
Gambar 2.12 Memberi nama project (*.prj)

Tekan tombol “Save” untuk menyetujui dan melanjutkan pada proses berikutnya. Setelah penekanan tombol “Save” akan muncul kembali dialog yang ketiga, yaitu untuk melakukan set terhadap nama dan lokasi penyimpanan CodeWizardProject file (*.cwp) seperti ditunjukkan pada gambar berikut:



Gambar 2.13 Memberi nama CodeWizardProject (*.cwp)

Tekan tombol “Save” untuk menyetujui dan melanjutkan. Selanjutnya akan muncul IDE editor program dari CodeVisionAVR. Anda siap untuk melakukan pembuatan program menggunakan CodeVisionAVR.



Gambar 2.14 Editor Program

2. Pada program-C yang dihasilkan, tambahkan header `stdio.h`
3. Lakukan pendeklarasian variable global, letakkan di atas fungsi `main(void)`

```
// Declare your global variables here
unsignedchar data_suhu,xstring[5];
float derajad_suhu;
void main(void) { ...
```

4. Pada bagian while(1) {...} tambahkan perintah yang telah diblok dengan warna kuning berikut :

```

while (1)
{
// Place your code here
lcd_clear();
    lcd_putsf("Data suhu = ");
    data_suhu = read_adc(0);
    sprintf(xstring, "%d", data_suhu);
    lcd_puts(xstring);

    lcd_gotoxy(0,1);
    lcd_putsf("Drjd suhu = ");
    derajad_suhu= data_suhu * 5.0 * 100.0 / 3.2 / 255.0;
    sprintf(xstring, "%.1f", derajad_suhu);
    lcd_puts(xstring);
    delay_ms(100);
}

```

5. Untuk lebih jelasnya, berikut adalah kode program secara keseluruhan:

```

#include<mega16.h>
#include<stdio.h>
// Alphanumeric LCD Module functions
#asm
    .equ __lcd_port=0x18 ;PORTB
#endif
#include<lcd.h>

#include<delay.h>

#define ADC_VREF_TYPE 0x20

// Read the 8 most significant bits
// of the AD conversion result
unsignedchar read_adc(unsignedchar adc_input)
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Delay needed for the stabilization of the ADC input voltage
delay_us(10);
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCH;
}

// Declare your global variables here
unsignedchar data_suhu,xstring[5];
float derajad_suhu;
void main(void)
{
// Declare your local variables here
}

```

```

// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTA=0x00;
DDRA=0x00;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTB=0x00;
DDRB=0x00;

// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTD=0x00;
DDRD=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;

```

```

ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// ADC initialization
// ADC Clock frequency: 691,200 kHz
// ADC Voltage Reference: AREF pin
// ADC Auto Trigger Source: None
// Only the 8 most significant bits of
// the AD conversion result are used
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x84;

// LCD module initialization
lcd_init(16);

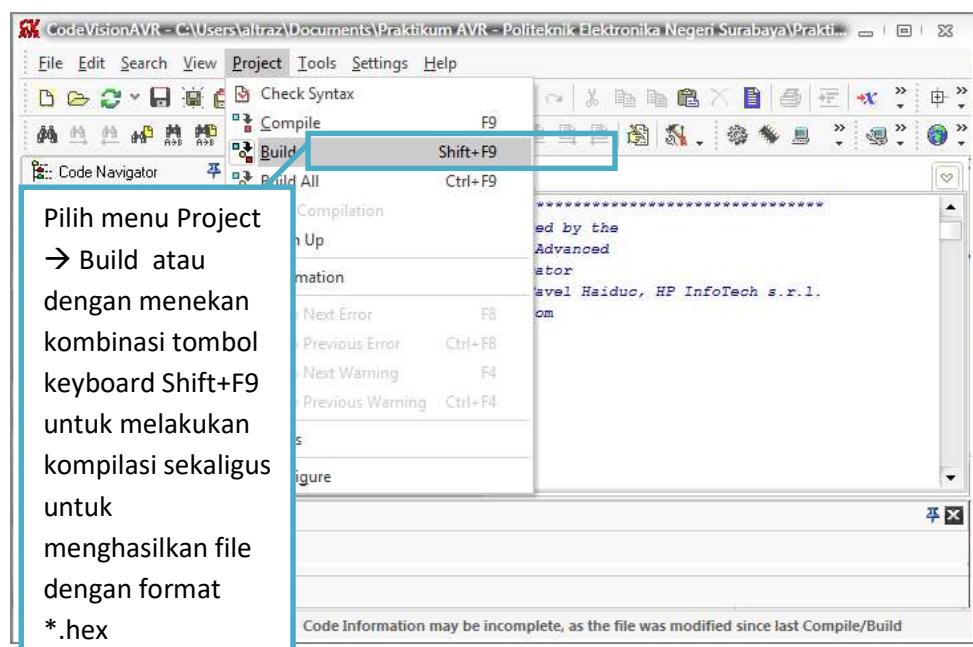
while (1)
{
// Place your code here
    lcd_clear();
    lcd_putsf("Data suhu = ");
    data_suhu = read_adc(0);
    sprintf(xstring,"%d",data_suhu);
    lcd_puts(xstring);

    lcd_gotoxy(0,1);
    lcd_putsf("Drjd suhu = ");
    derajad_suhu= data_suhu * 5.0 * 100.0 / 3.2 / 255.0; // gain op amp (rf/ri)+1 = (22K/10K)+1 = 3.2
    sprintf(xstring,"% .1f",derajad_suhu);
    lcd_puts(xstring);
    delay_ms(100);
}

```

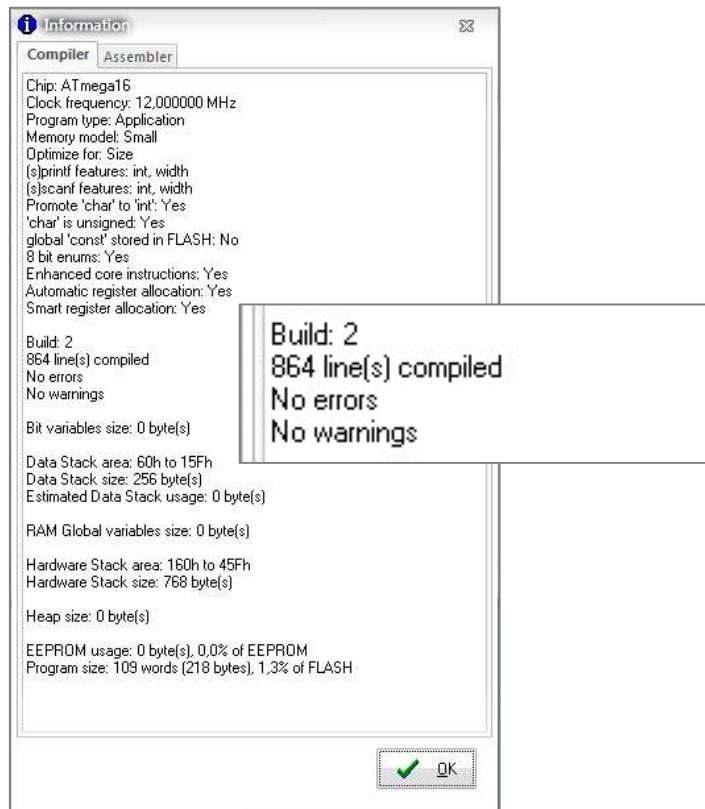
```
    };
```

6. Setelah pembuatan program selesai, maka proses selanjutnya yaitu lakukan kompilasi terhadap kode program tersebut sekaligus untuk menghasilkan file dengan format hexadecimal (*.hex). Hal ini dapat dilakukan dengan memilih menu Project → Build (Shift+F9) seperti ditunjukkan pada gambar berikut :



Gambar 2.15 Build Source

Apabila kode program sudah benar (tanpa error) dan tidak menyalahi struktur pemrograman bahasa C, maka akan muncul dialog informasi seperti ditunjukkan pada gambar berikut:



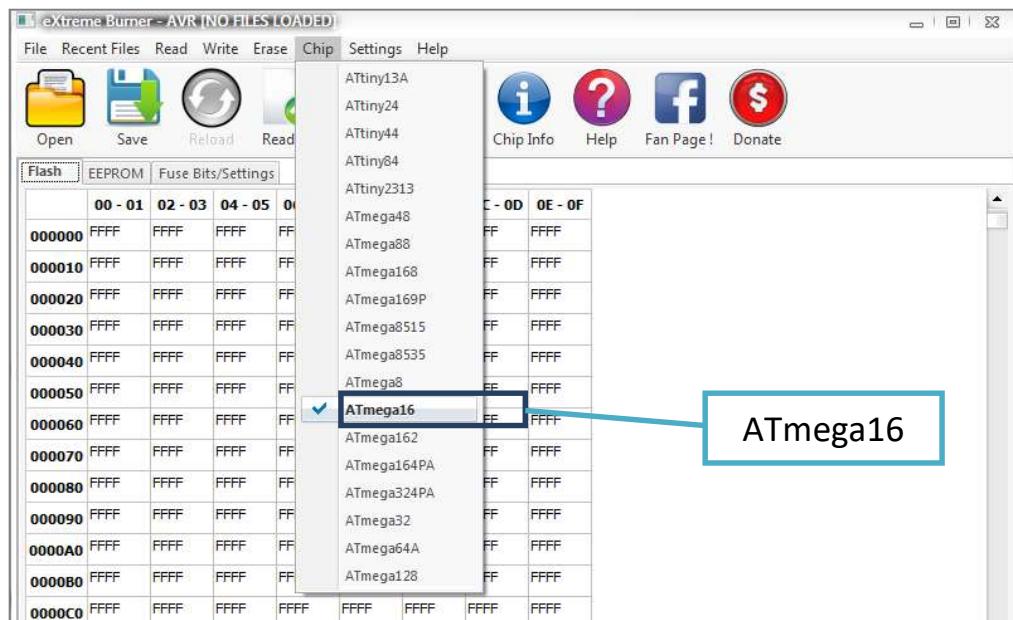
Gambar 2.16 Dialog informasi status kompilasi

7. Setelah diperoleh file dengan format *.hex, maka lakukan download file *.hex ke dalam mikrokontroler. Pertama, jalankan program eXtreme Burner - AVR dengan cara melakukan klik ganda pada shortcut icon eXtreme Burner– AVR pada desktop



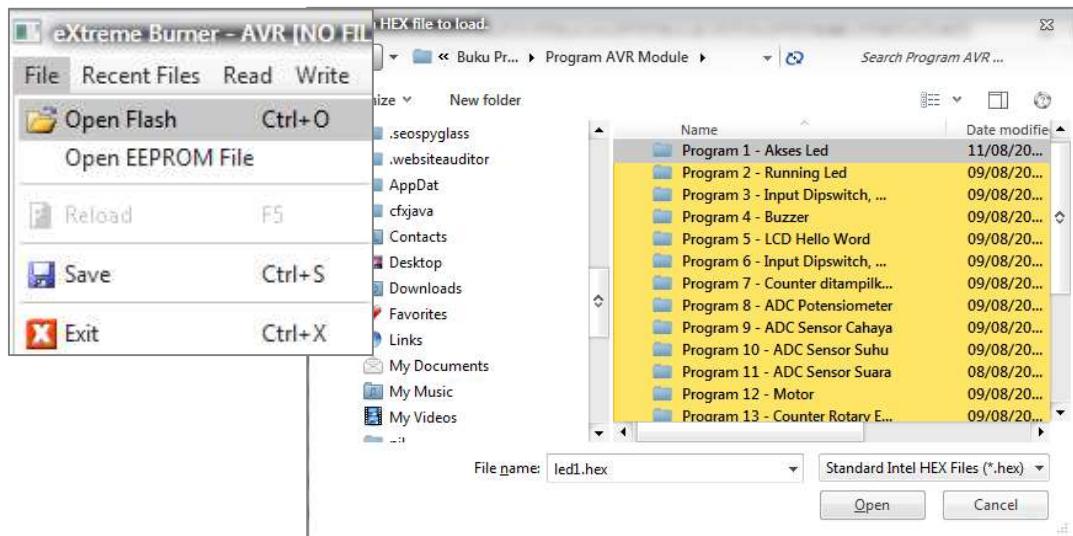
Gambar 2.17 Shortcut eXtreme Burner - AVR

Selanjutnya akan muncul tampilan utama dari program eXtreme Burner – AVR. Pada praktikum ini menggunakan mikrokontroler keluarga AVR dengan tipe IC ATMega16. Oleh karena itu, lakukan pengaturan terhadap tipe IC terlebih dahulu dengan memilih menu Chip → ATmega16 seperti ditunjukkan pada gambar berikut:



Gambar 2.18 Tipe Mikrokontroler

Setelah itu, lakukan open file *.hex dengan memilih menu File → Open Flash (Ctrl+O) seperti ditunjukkan pada gambar berikut:



Gambar 2.19 Open Flash

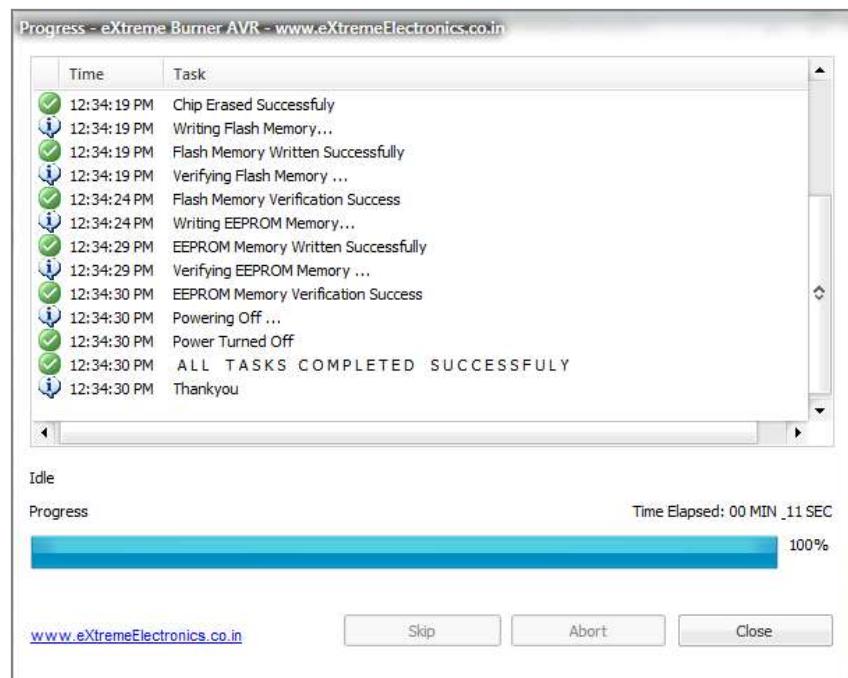
Lakukan pencarian terhadap lokasi file *.hex Anda, untuk selanjutnya tekan tombol “Open” apabila sudah ditemukan file yang dimaksud. Setelah itu akan muncul pesan yang memberitahukan bahwa file *.hex berhasil di-load program eXtreme Burner – AVR.



Gambar 2.20 Message



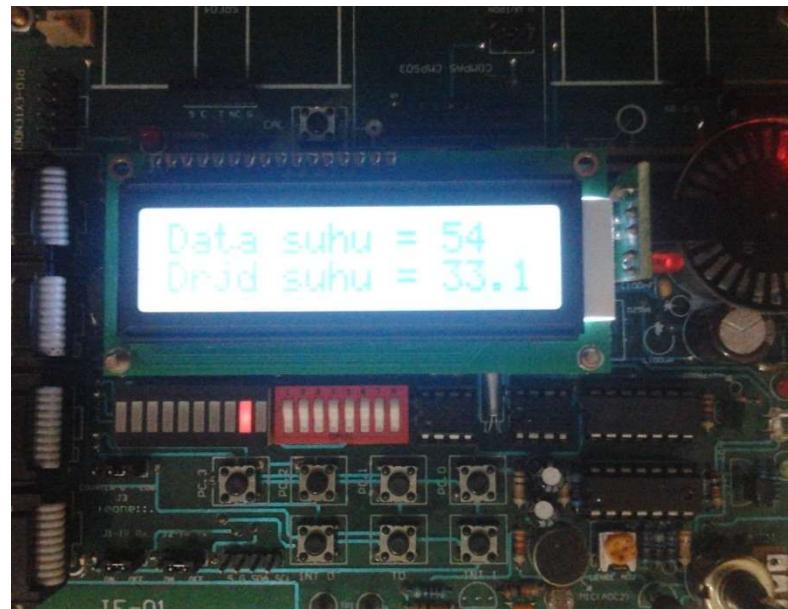
Selanjutnya tekan tombol **Write All** untuk memulai download ke dalam mikrokontroler Anda. Berikut ini merupakan tampilan apabila proses download ke dalam mikrokontroler berhasil dilakukan:



Gambar 2.21 Proses Berhasil

8. Selesai

2.6. Hasil Percobaan



Gambar 2.22 Running Program

2.7. Tugas

Buat simulasi : *Early Warning System* (EWS) untuk bencana alam kebakaran rumah dengan memperhatikan tingkat suhu ruangan, ada 3 kondisi suhu, normal ($0 \leq T \leq 40$ celcius), bahaya sedang ($40 < T \leq 50$ celcius), bahaya kritis/tinggi ($T > 50$ celcius)

PERCOBAAN 3

SENSOR SUARA

PERCOBAAN 3

SENSOR SUARA

3.1. Tujuan

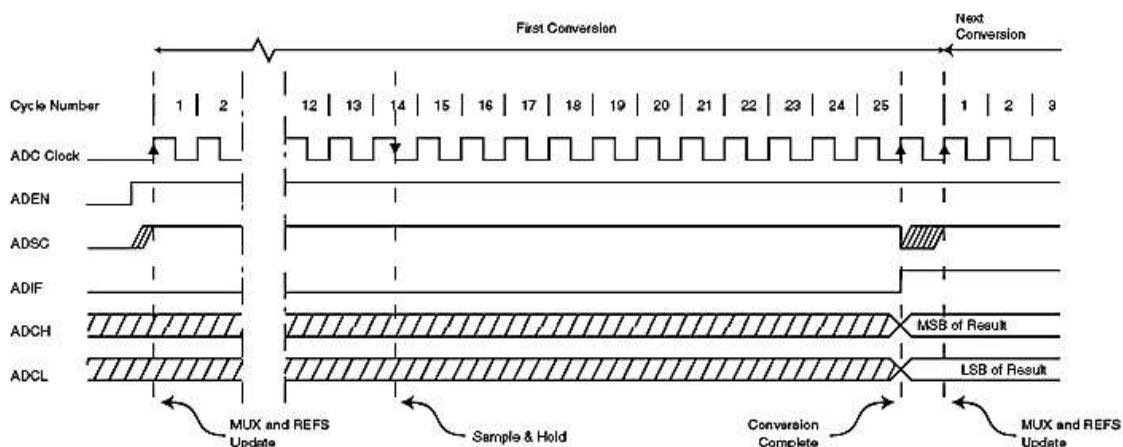
1. Mahasiswa dapat lebih memahami cara kerja microphone sebagai sensor suara
2. Mahasiswa dapat mengakses ADC sensor suara pada ATMega16
3. Mahasiswa dapat membuat aplikasi ADC sederhana yang memanfaatkan sensor suara

3.2. Dasar Teori

Mikrokontroller AVR Atmega16 mempunyai perangkat Analog To Digital Converter atau ADC 8 bit yang berguna untuk mengubah data masukan berupa tegangan analog menjadi bentuk data digital.

ATMega16 mempunyai ADC (Analog to Digital Converter) internal dengan fitur sebagai berikut (untuk lebih detil dapat mengacu ada datasheet)

- 10-bit Resolution
- 65 - 260 μ s Conversion Time
- Up to 15 kSPS at Maximum Resolution
- 8 Multiplexed Single Ended Input Channels
- Optional Left Adjustment for ADC Result Readout
- 0 - VCC ADC Input Voltage Range
- Selectable 2.56V ADC Reference Voltage
- Free Running or Single Conversion Mode
- ADC Start Conversion by Auto Triggering on
- Interrupt Sources
- Interrupt on ADC Conversion Complete
- Sleep Mode Noise Canceler



Gambar 3.1 Timer Diagram untuk Mode Single Conversion

Register-register yang dipakai untuk mengakses ADC adalah:

- **ADMUX – ADC Multiplexer Selection Register**

Bit	7	6	5	4	3	2	1	0	ADMUX
Read/Write	R/W								
Initial Value	0	0	0	0	0	0	0	0	

- **ADCSRA – ADC Control and Status Register**

Bit	7	6	5	4	3	2	1	0	ADCSRA
Read/Write	R/W								
Initial Value	0	0	0	0	0	0	0	0	

- **ADCL, ADCH – ADC data register**

Bila ADCLR = 0

Bit	15	14	13	12	11	10	9	8	ADCH
	-	-	-	-	-	-	ADC9	ADC8	ADCL
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	
	7	6	5	4	3	2	1	0	

Read/Write

R	R	R	R	R	R	R	R	R
R	R	R	R	R	R	R	R	R

Initial Value

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

Bila ADCLR = 1

Bit	15	14	13	12	11	10	9	8	ADCH
	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCL
	ADC1	ADC0	-	-	-	-	-	-	
	7	6	5	4	3	2	1	0	

Read/Write

R	R	R	R	R	R	R	R	R
R	R	R	R	R	R	R	R	R

Initial Value

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

Setelah ADC selesai melakukan konversi kedua register ini berisi hasil konversi. Bila channel differensial dipilih maka hasilnya dalam format two's complement. Saat ADCL dibaca, data register tidak akan meng-update data sampai ADCH dibaca. Jika hasilnya dirata kiri (left adjust) dan hanya butuh 8-bit maka cukuplah dengan membaca ADCH. Jika butuh 10-bit, baca ADCL dahulu kemudian ADCH.

- **SFIOR – Special Function I/O Register untuk sumber auto trigger**

Bit	7	6	5	4	3	2	1	0	SFIOR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Berikut adalah langkah-langkah yang harus dilakukan untuk melakukan konversi mode single-conversion pada ADC internal dari Atmega16:

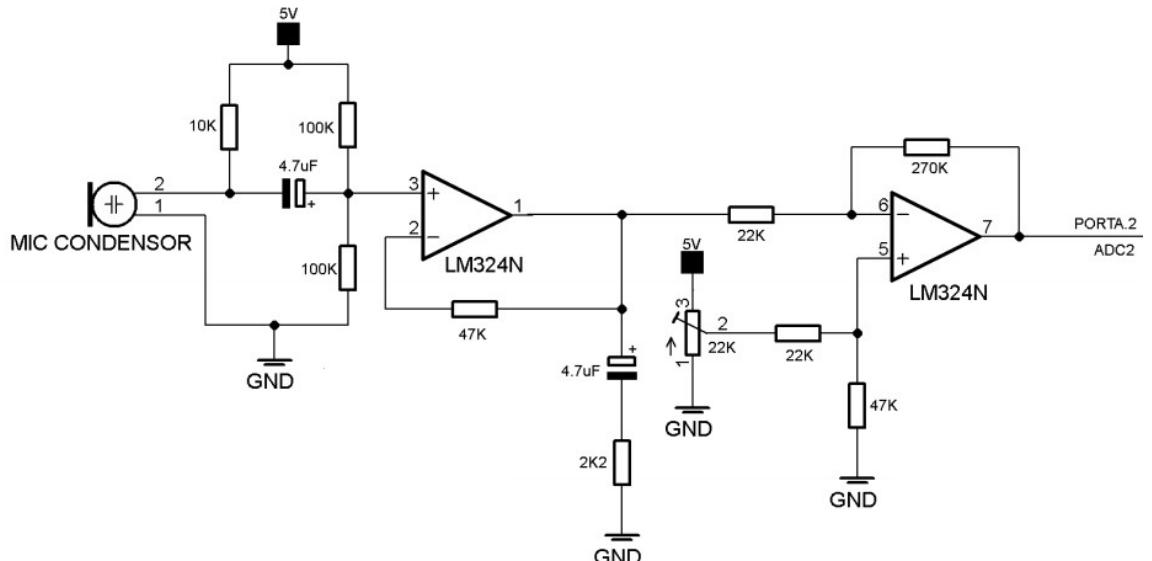
1. Seting register ADMUX.
2. Seting register ADCSRA.
3. Seting register SFIOR (untuk nomor 1, 2, dan 3 dilakukan oleh CodeWizardAVR).
4. Pilih channel ADC pada register ADMUX.
5. Start konversi ADC dengan cara mengaktifkan bit ADSC pada register ADCSRA.
6. Bila ADC selesai melakukan konversi, program akan meloncat ke layanan interupsi ADC.
7. Ambil data pada register ADCH.
8. Tampilkan nilai pada LCD.
9. Delay sekitar 500 milidetik, agar tampilan pada LCD dapat dilihat.
10. Jika diinginkan, ulangi langkah 4.

3.3. Peralatan

- 1 set PC yang dilengkapi dengan software CodeVision AVR.
- 1 set development board AVR ATmega16
- 1 power-supply +9VDC

3.4. Modul I/O

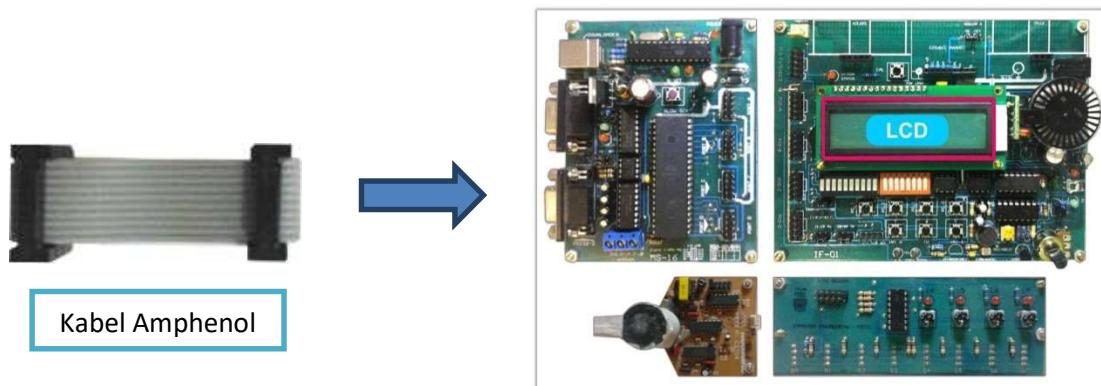
Berikut ini merupakan rangkaian sensor suara pada development board AVR ATmega16:



Gambar 3.2 Rangkaian sensor suara

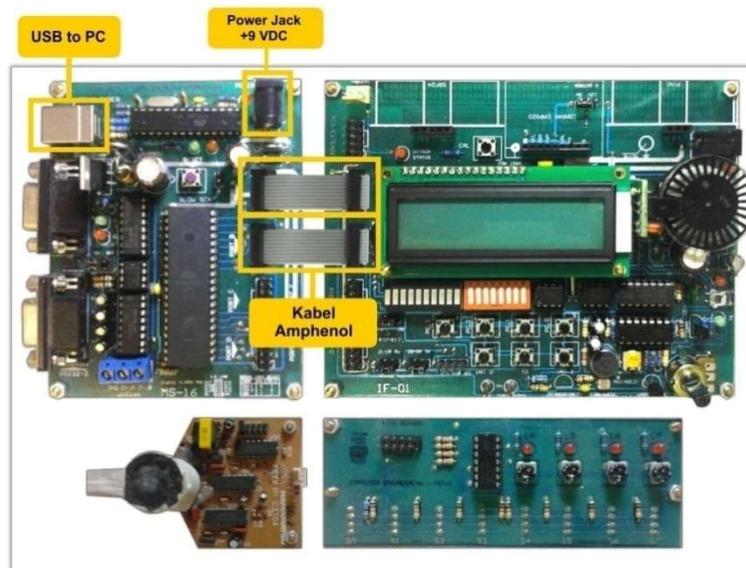
- **Konfigurasi modul pada percobaan ini :**

Pasangkan kabel amphenol pada modul development untuk menghubungkan sistem minimum ATMega16 dengan Training Board.



Gambar 3.3 Komponen Training board

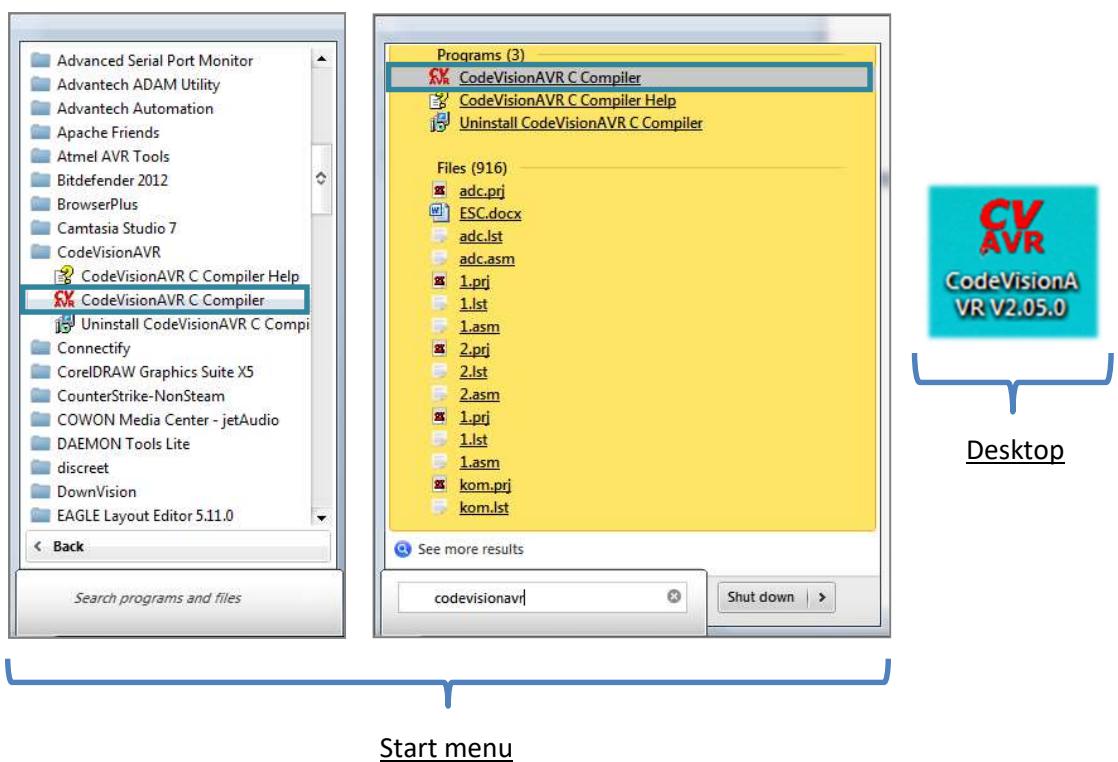
Sehingga menjadi seperti ditunjukkan pada gambar berikut:



Gambar 3.4 Komponen Training board

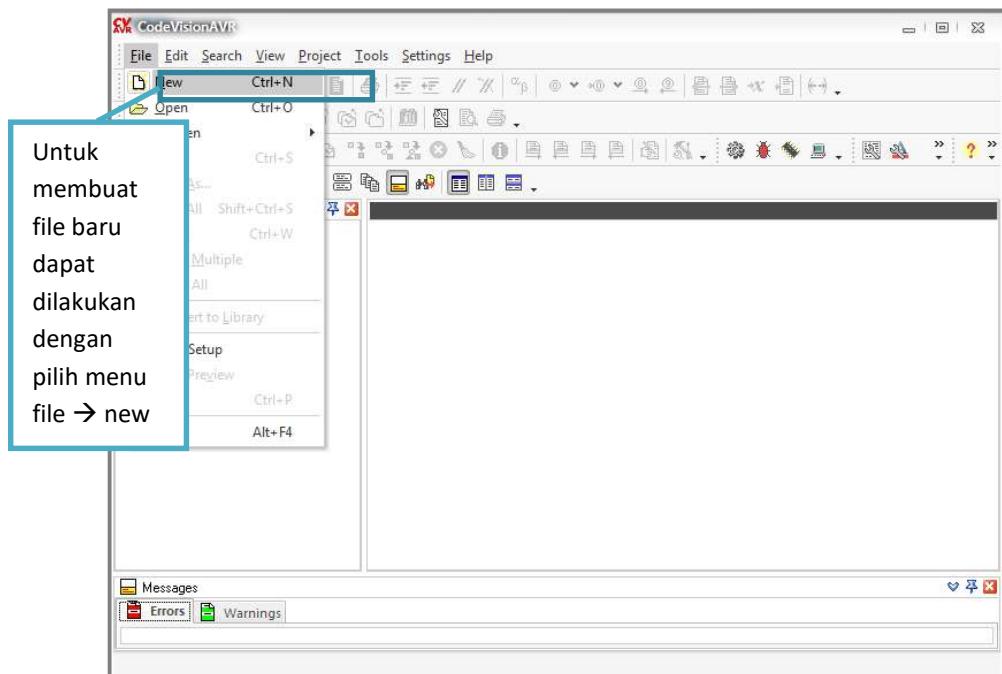
3.5. Prosedur Praktikum

1. Jalankan aplikasi CodeVisionAVR dengan cara melakukan klik ganda pada shortcut icon CodeVisionAVR pada desktop atau dengan cara mencari shortcut program pada start menu system operasi windows.



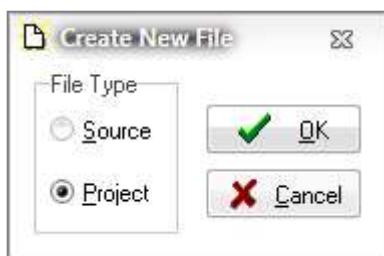
Gambar 3.5 Shortcut CodeVisionAVR

Untuk memulai membuat project baru, pada menubar, pilih File → New, seperti yang ditunjukkan oleh gambar berikut:



Gambar 3.6 Membuat File baru

Setelah itu akan muncul suatu dialog yang mengharuskan *user* untuk memilih antara pembuatan source file atau project. Dalam setiap percobaan pada praktikum ini, dibuat project baru untuk setiap percobaan, oleh karena itu *user* sebaiknya melakukan *check* pada checkbox project dan dilanjutkan dengan menekan tombol “OK”



Gambar 3.7 Membuat project baru

Berikutnya akan keluar dialog yang menanyakan Anda apakah akan membuat project baru. Klik tombol “Yes” untuk menyetujui.

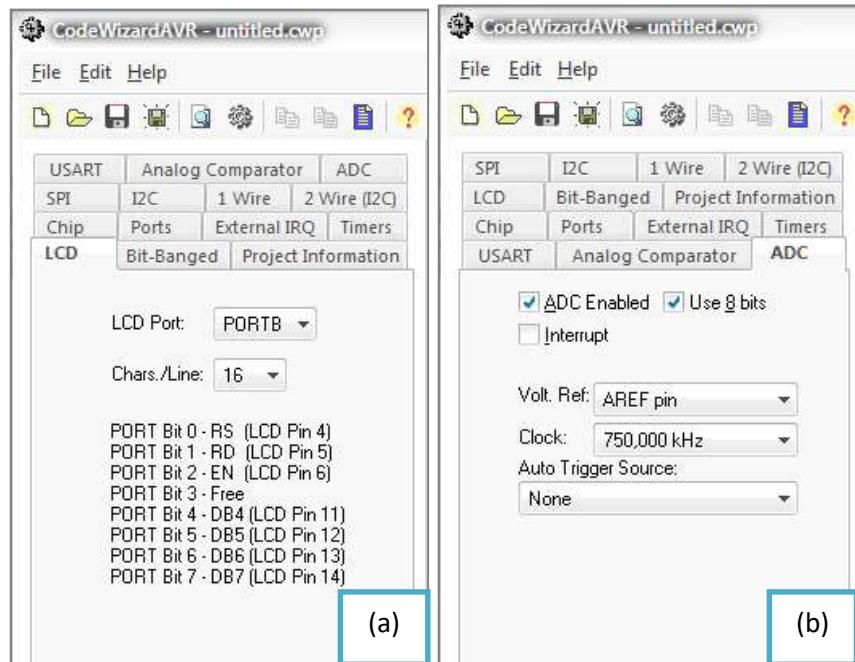


Gambar 3.8 Memilih untuk membuat project baru pada CodeVisionAVR

Setelah itu lakukan set pada CodeWizardAVR. Untuk chip diset dengan nama chip ATmega16 dengan clock 12 MHz.

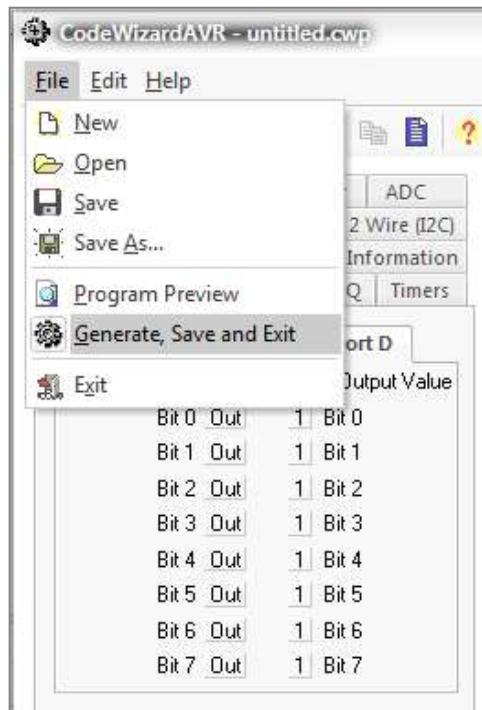
Berikutnya Anda akan menginisialisasi LCD pada PortB, yaitu dengan cara masuk pada tab LCD dan pilih item **PORTE** pada selectbox “LCD Port”. Set banyaknya karakter dari LCD yang diinginkan. Dalam hal ini menggunakan 16 karakter, oleh karena itu pilih item **16** pada selectbox “Chars/Line”.

Setelah itu lakukan set ADC dengan masuk pada tab ADC. Berikan tanda check terhadap checkbox “ADC Enabled” untuk meng-enable ADC. Berikan tanda check terhadap checkbox “Use 8 bits” untuk menggunakan ADC 8 bit.



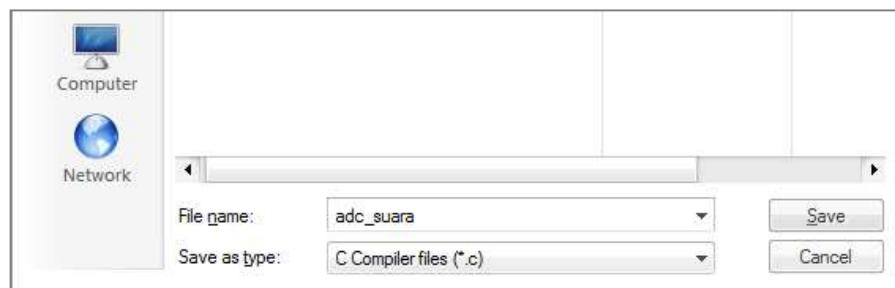
Gambar 3.9 (a) Setting LCD pada PortB (b) Setting ADC 8 bit

Karena pada contoh ini tidak digunakan fasilitas lain maka setting CodeWizardAVR siap disimpan dalam file. Pada menu CodeWizardAVR, pilih File → Generate, Save and Exit, seperti yang ditunjukkan pada gambar berikut:



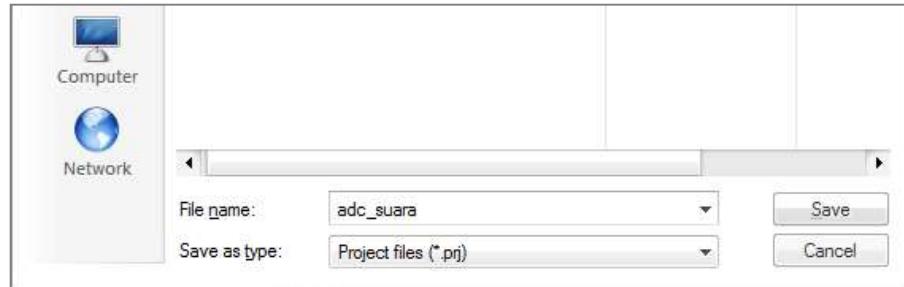
Gambar 3.10 Menyimpan setting

Setelah penekanan menu “Generate, Save and Exit”, akan muncul suatu dialog untuk melakukan set terhadap nama dan lokasi penyimpanan dari source file (*.c) seperti ditunjukkan pada gambar berikut:



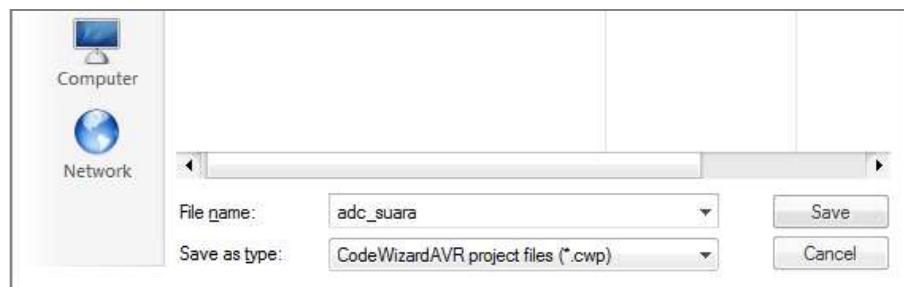
Gambar 3.11 Memberi nama pada file C (*.c)

Tekan tombol “Save” untuk menyetujui dan melanjutkan pada proses berikutnya. Setelah penekanan tombol “Save” akan muncul kembali dialog yang kedua, yaitu untuk melakukan set terhadap nama dan lokasi penyimpanan project file (*.prj) seperti ditunjukkan pada gambar berikut:



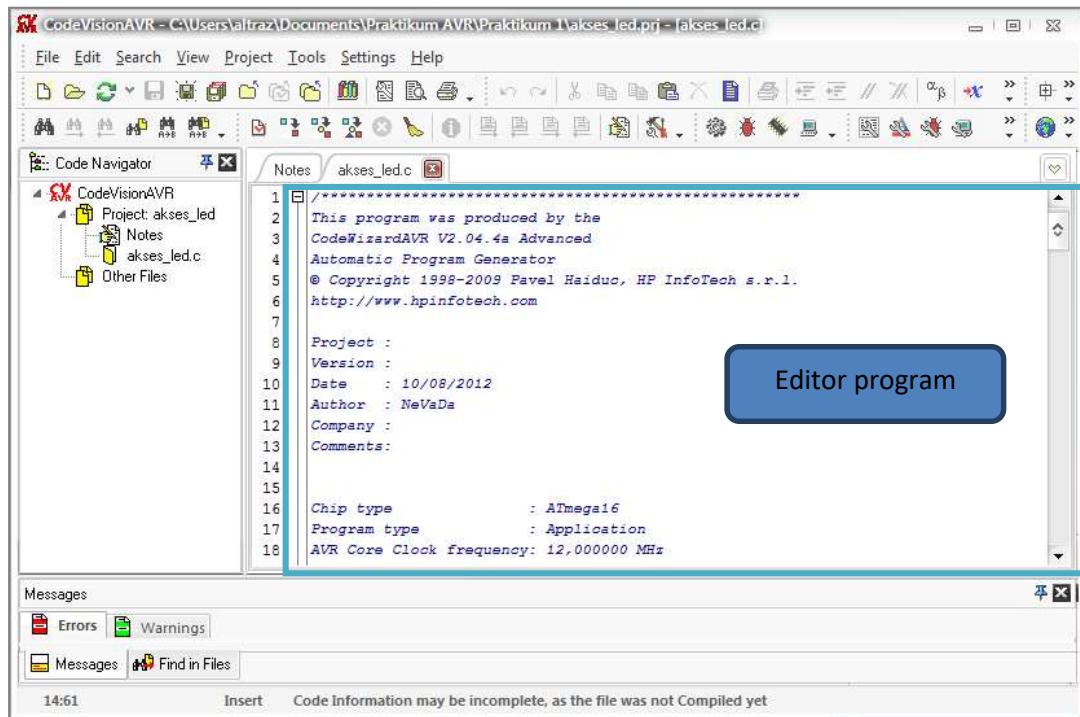
Gambar 3.12 Memberi nama project (*.prj)

Tekan tombol “Save” untuk menyetujui dan melanjutkan pada proses berikutnya. Setelah penekanan tombol “Save” akan muncul kembali dialog yang ketiga, yaitu untuk melakukan set terhadap nama dan lokasi penyimpanan CodeWizardProject file (*.cwp) seperti ditunjukkan pada gambar berikut:



Gambar 3.13 Memberi nama CodeWizardProject (*.cwp)

Tekan tombol “Save” untuk menyetujui dan melanjutkan. Selanjutnya akan muncul IDE editor program dari CodeVisionAVR. Anda siap untuk melakukan pembuatan program menggunakan CodeVisionAVR.



Gambar 3.14 Editor Program

2. Pada program-C yang dihasilkan, tambahkan header `stdio.h` dan `delay.h`
3. Lakukan pendeklarasian variable global, letakkan di atas fungsi `main(void)`

```
// Declare your global variables here
unsignedchar xstring[3],data_suara;
void main(void) { ... }
```

4. Pada bagian `while(1) {...}` tambahkan perintah yang telah diblok dengan warna kuning berikut :

```
while (1)
{
// Place your code here
lcd_clear();
    lcd_putsf("Data Suara = ");
    data_suara = read_adc(1);
    sprintf(xstring,"%d",data_suara);
    lcd_puts(xstring);
    delay_ms(100);
};
```

5. Untuk lebih jelasnya, berikut kode program secara keseluruhan:

```

#include<mega16.h>
#include<stdio.h>

// Alphanumeric LCD Module functions
#asm
    .equ __lcd_port=0x18 ;PORTB
#endifasm
#include<lcd.h>

#include<delay.h>

#define ADC_VREF_TYPE 0x20

// Read the 8 most significant bits
// of the AD conversion result
unsignedchar read_adc(unsignedchar adc_input)
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Delay needed for the stabilization of the ADC input voltage
delay_us(10);
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCH;
}

// Declare your global variables here
unsignedchar xstring[3],data_suara;
void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTA=0x00;
DDRA=0x00;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTB=0x00;
DDRB=0x00;

// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTC=0x00;
DDRC=0x00;

```

```

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTD=0x00;
DDRD=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

```

```

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

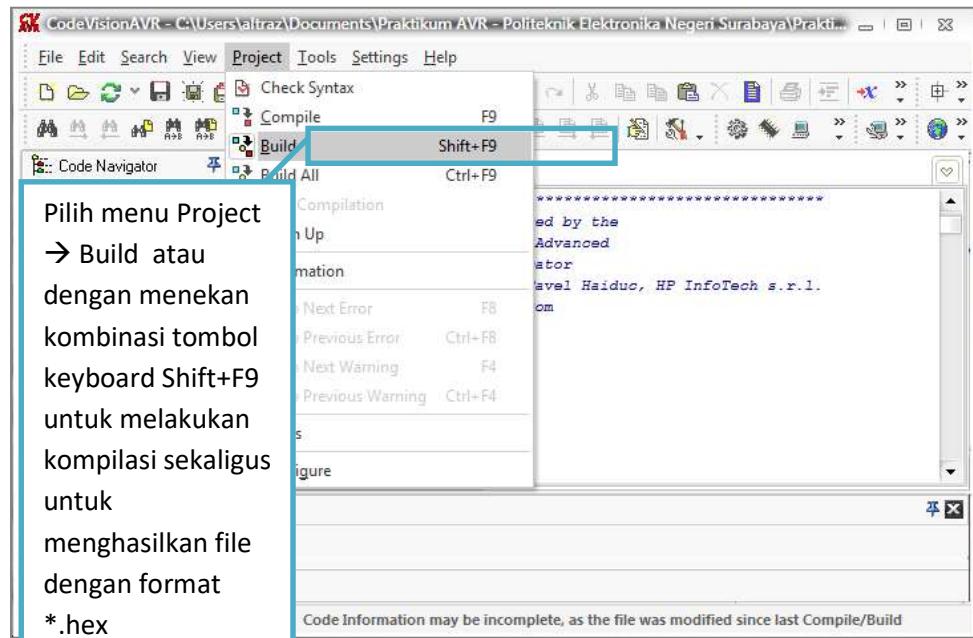
// ADC initialization
// ADC Clock frequency: 691,200 kHz
// ADC Voltage Reference: AREF pin
// ADC Auto Trigger Source: None
// Only the 8 most significant bits of
// the AD conversion result are used
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x84;

// LCD module initialization
lcd_init(16);

while (1)
{
// Place your code here
    lcd_clear();
    lcd_putsf("Data Suara = ");
    data_suara = read_adc(1);
    sprintf(xstring, "%d", data_suara);
    lcd_puts(xstring);
    delay_ms(100);
}
}

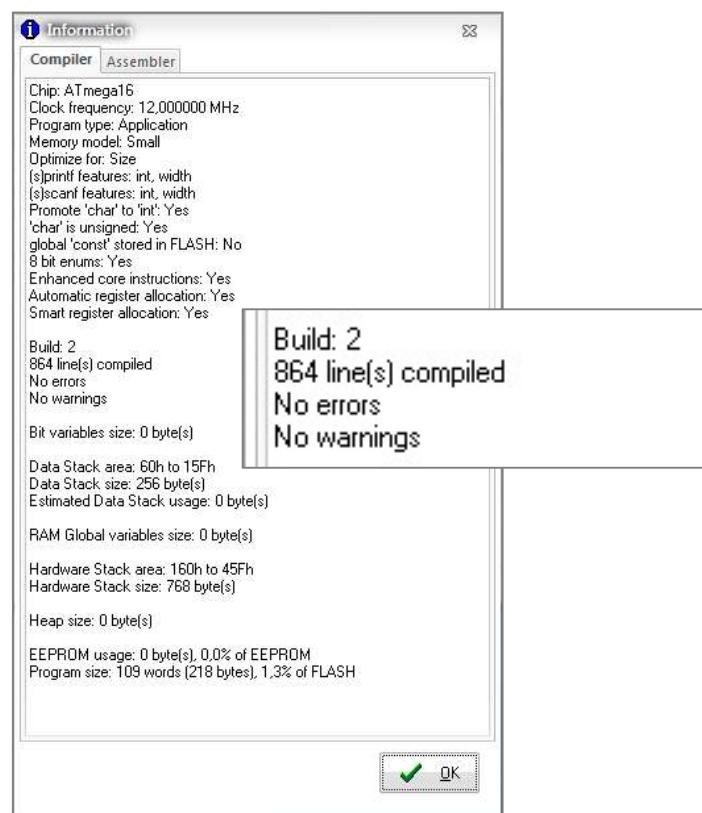
```

6. Setelah pembuatan program selesai, maka proses selanjutnya yaitu lakukan kompilasi terhadap kode program tersebut sekaligus untuk menghasilkan file dengan format hexadecimal (*.hex). Hal ini dapat dilakukan dengan memilih menu Project → Build (Shift+F9) seperti ditunjukkan pada gambar berikut :



Gambar 3.15 Build Source

Apabila kode program sudah benar (tanpa error) dan tidak menyalahi struktur pemrograman bahasa C, maka akan muncul dialog informasi seperti ditunjukkan pada gambar berikut:



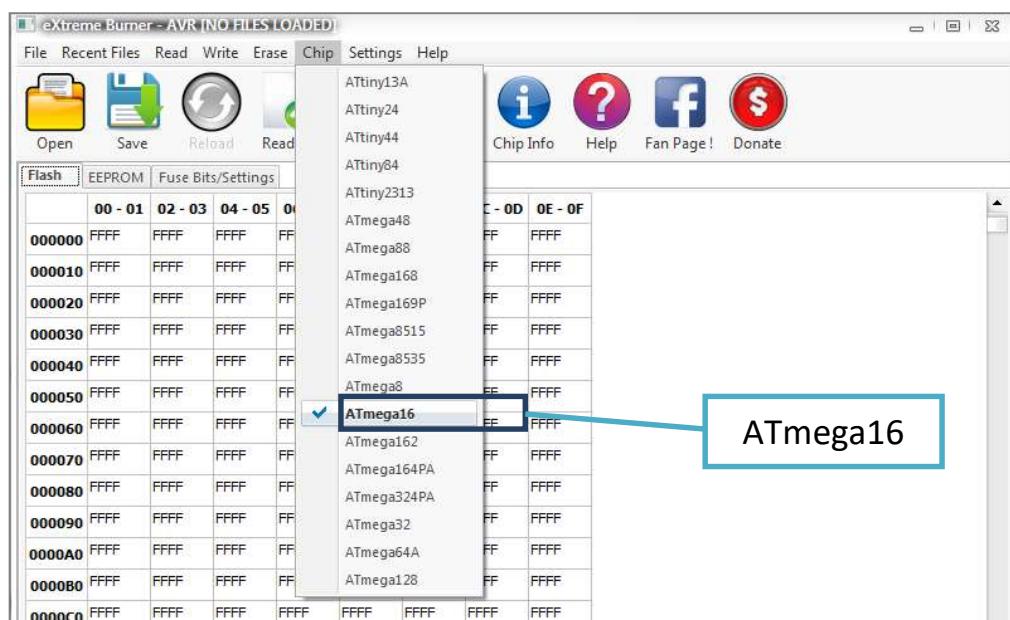
Gambar 3.16 Dialog informasi status kompilasi

7. Setelah diperoleh file dengan format *.hex, maka lakukan download file *.hex ke dalam mikrokontroler. Pertama, jalankan program eXtreme Burner - AVR dengan cara melakukan klik ganda pada shortcut icon eXtreme Burner – AVR pada desktop



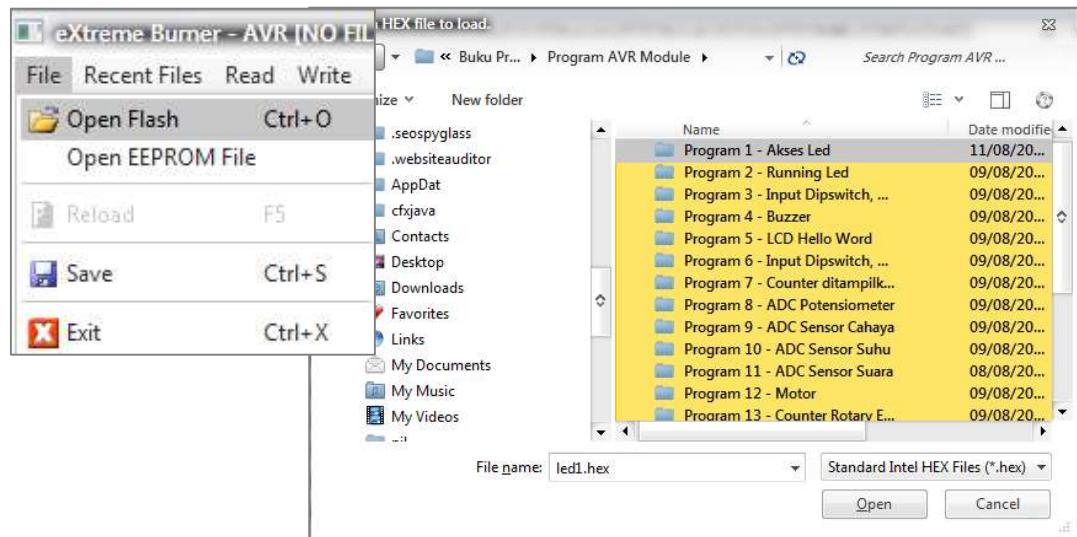
Gambar 3.17 Shortcut eXtreme Burner - AVR

Selanjutnya akan muncul tampilan utama dari program eXtreme Burner – AVR. Pada praktikum ini menggunakan mikrokontroler keluarga AVR dengan tipe IC ATMega16. Oleh karena itu, lakukan pengaturan terhadap tipe IC terlebih dahulu dengan memilih menu Chip → ATmega16 seperti ditunjukkan pada gambar berikut:



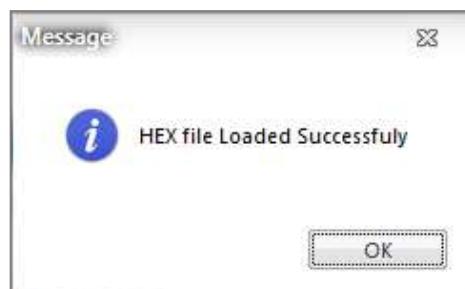
Gambar 3.18 Tipe Mikrokontroler

Setelah itu, lakukan open file *.hex dengan memilih menu File → Open Flash (Ctrl+O) seperti ditunjukkan pada gambar berikut:



Gambar 3.19 Open Flash

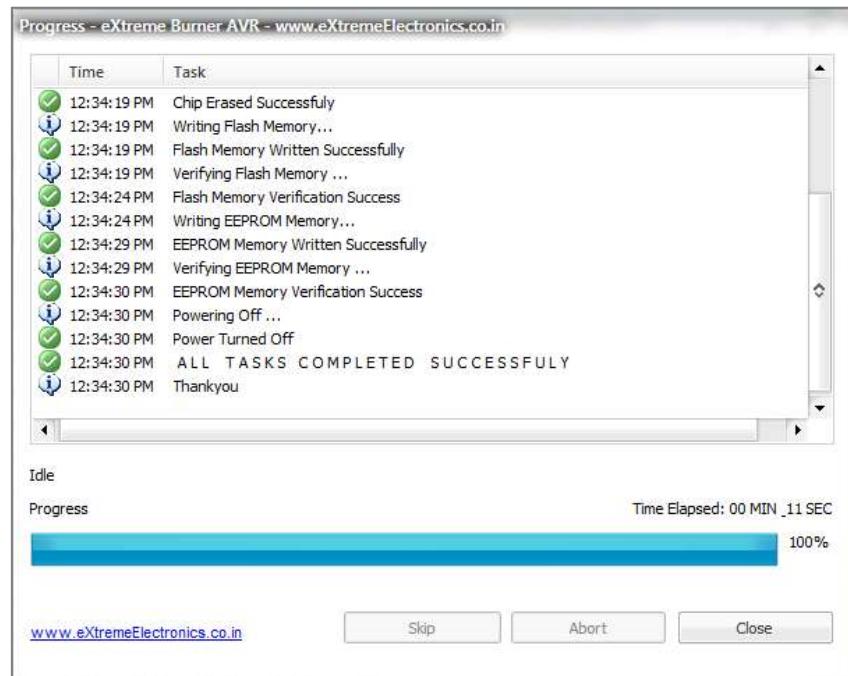
Lakukan pencarian terhadap lokasi file *.hex Anda, untuk selanjutnya tekan tombol “Open” apabila sudah ditemukan file yang dimaksud. Setelah itu akan muncul pesan yang memberitahukan bahwa file *hex berhasil di-load program eXtreme Burner – AVR.



Gambar 3.20 Message



Selanjutnya tekan tombol untuk memulai download ke dalam mikrokontroler Anda. Berikut ini merupakan tampilan apabila proses download ke dalam mikrokontroler berhasil dilakukan:



Gambar 3.21 Proses Berhasil

8. Selesai

3.6. Hasil Percobaan

Laporkan hasil percobaan anda.

3.7. Tugas

Buat *voice detektor* untuk mendeteksi 3 tingkat suara, kategorikan level suara rendah, sedang dan keras/bising.

PERCOBAAN 4

**AKTUATOR :
MOTOR DC**

PERCOBAAN 4

AKTUATOR : MOTOR DC

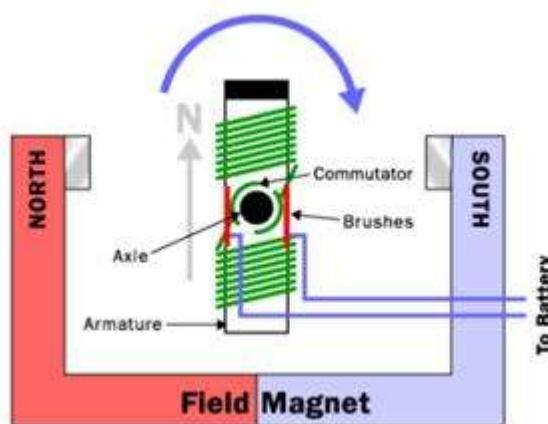
4.1. Tujuan

1. Mahasiswa dapat mengakses motor DC menggunakan ATMega16
2. Mahasiswa dapat membuat aplikasi sederhana yang mengimplementasikan penggunaan motor pada mikrokontroler
3. Mahasiswa dapat mengetahui penerapan timer pada ATMega16

4.2. Dasar Teori

1. Motor DC

Motor DC adalah piranti elektronik yang mengubah energi listrik menjadi energi mekanik berupa gerak rotasi. Pada motor DC terdapat jangkar dengan satu atau lebih kumparan terpisah. Tiap kumparan berujung pada cincin belah (komutator). Dengan adanya insulator antara komutator, cincin belah dapat berperan sebagai saklar kutub ganda (*double pole, double throw switch*). Motor DC bekerja berdasarkan prinsip gaya Lorentz, yang menyatakan ketika sebuah konduktor beraliran arus diletakkan dalam medan magnet, maka sebuah gaya (yang dikenal dengan gaya Lorentz) akan tercipta secara ortogonal diantara arah medan magnet dan arah aliran arus. Mekanisme ini diperlihatkan pada Gambar berikut ini.



Gambar 4.1 Mekanisme motor DC

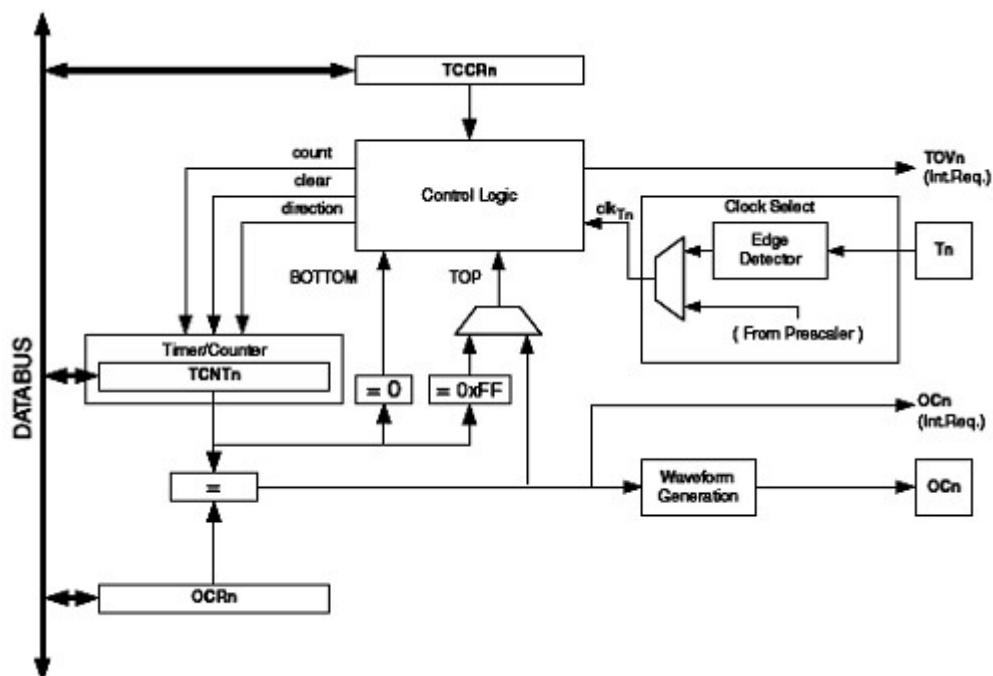
2. Timer/Counter

Timer/counter adalah fasilitas dari ATMega16 yang digunakan untuk perhitungan pewaktuan. Beberapa fasilitas channel dari timer counter antara lain: counter channel tunggal, pengosongan data timer sesuai dengan data pembanding, bebas

-glitch, tahap yang tepat Pulse Width Modulation (PWM), pembangkit frekuensi, event counter external.

Gambaran Umum

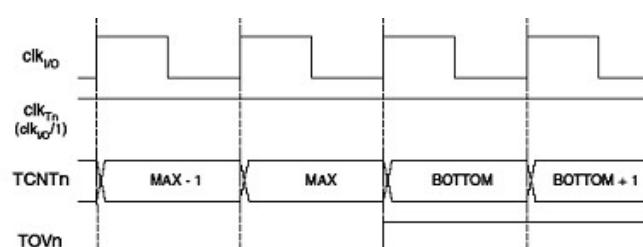
Gambar diagram block timer/counter 8 bit ditunjukkan pada gambar 2. Untuk penempatan pin I/O telah di jelaskan pada bagian I/O di atas. CPU dapat diakses register I/O, termasuk dalam pin-pin I/O dan bit I/O. Device khusus register I/O dan lokasi bit terdaftar pada deskripsi timer/counter 8 bit.



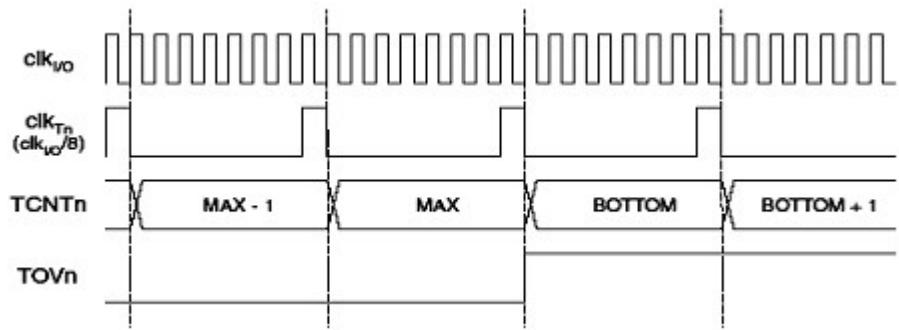
Gambar 4.2 Diagram timer/counter

Timing Diagram Timer/Counter

Timer/counter didesain sinkron clock timer (clkT0) oleh karena itu ditunjukkan sebagai sinyal enable clock pada gambar 3. Gambar ini termasuk informasi ketika flag interrupt dalam kondisi set. Data timing digunakan sebagai dasar dari operasi timer/counter.



Gambar 4.3 Timing diagram timer/counter, tanpa prescaling

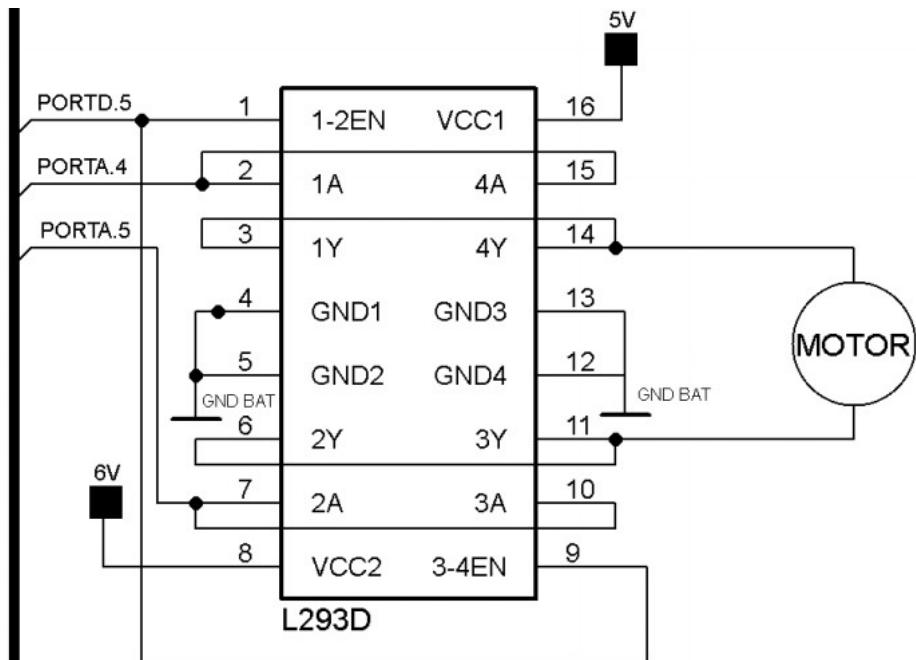


Gambar 4.4 Timing diagram timer/counter, dengan prescaling

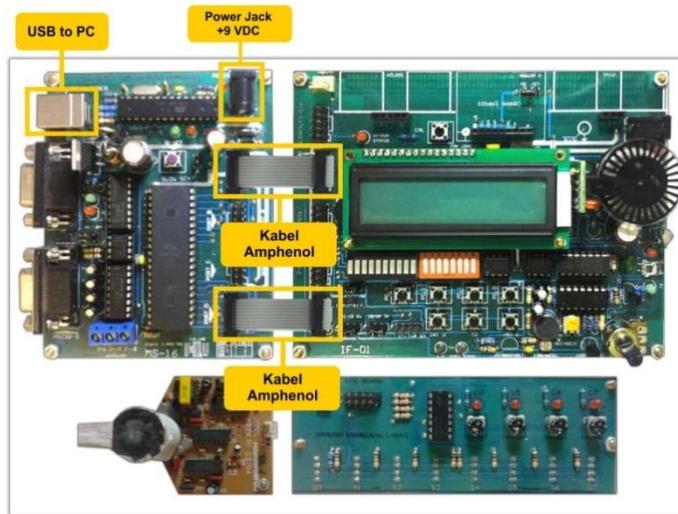
4.3. Peralatan

- 1 set PC yang dilengkapi dengan software CodeVision AVR.
- 1 set development board AVR ATmega16
- 1 power-supply +9VDC

4.4. Modul I/O



Gambar 4.5 Rangkaian driver motor pada modul

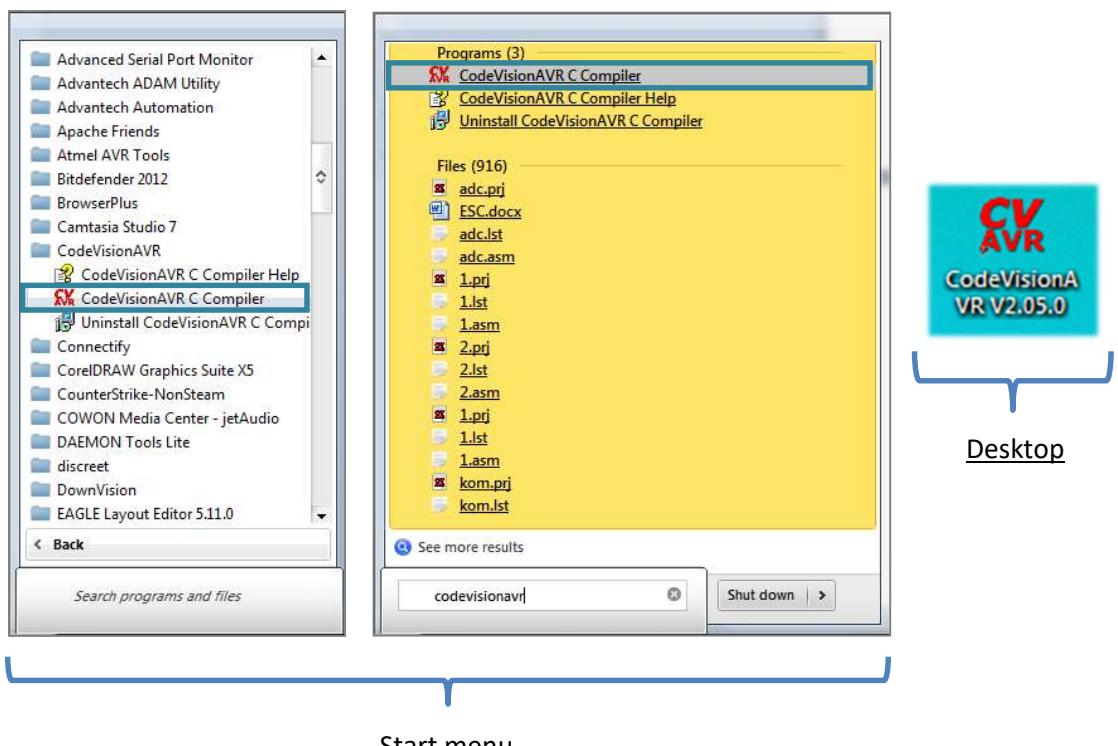


Gambar 4.6 Pengaturan koneksi pada Training board

4.5. Prosedur Praktikum

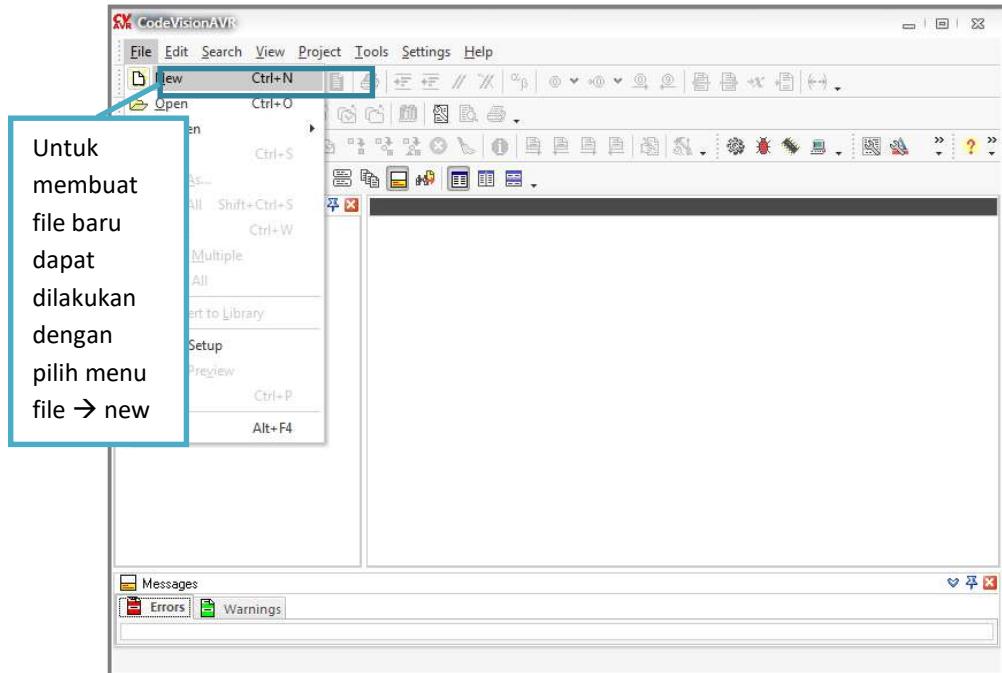
A. MOTOR 1

1. Jalankan aplikasi CodeVisionAVR dengan cara melakukan klik ganda pada shortcut icon CodeVisionAVR pada desktop atau dengan cara mencari shortcut program pada start menu system operasi windows.



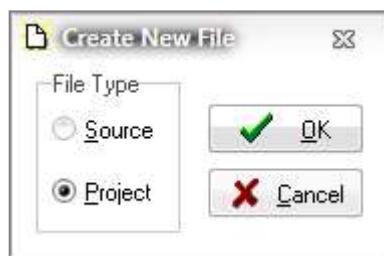
Gambar 4.7 Shortcut CodeVisionAVR

Untuk memulai membuat project baru, pada menubar, pilih File → New, seperti yang ditunjukkan oleh gambar berikut:



Gambar 4.8 Membuat File baru

Setelah itu akan muncul suatu dialog yang mengharuskan *user* untuk memilih antara pembuatan source file atau project. Dalam setiap percobaan pada praktikum ini, dibuat project baru untuk setiap percobaan, oleh karena itu *user* sebaiknya melakukan *check* pada checkbox project dan dilanjutkan dengan menekan tombol “OK”



Gambar 4.9 Membuat project baru

Berikutnya akan keluar dialog yang menanyakan Anda apakah akan membuat project baru. Klik tombol “Yes” untuk menyetujui.



Gambar 4.10 Memilih untuk membuat project baru pada CodeVisionAVR

Setelah itu lakukan set pada CodeWizardAVR. Untuk chip diset dengan nama chip ATmega16 dengan clock 12 MHz.

Berikutnya Anda akan memanfaatkan penggunaan Timer1. Berikut yang harus Anda set pada Timer1:

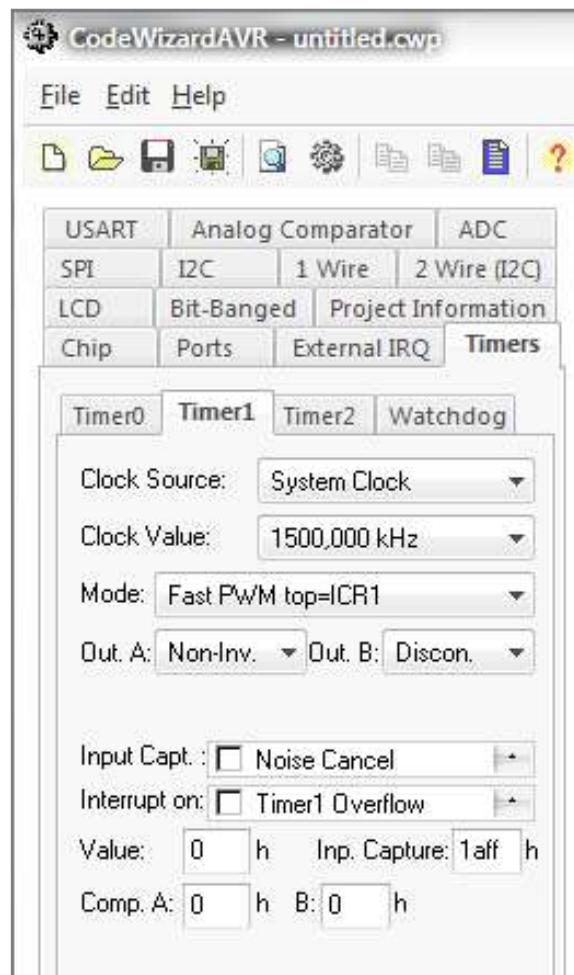
Clock Source = System Clock

Clock Value = 1500.000KHz

Mode = Fast PWM top=ICR1

Out A = Non Inv

Inp Capture = 1aff

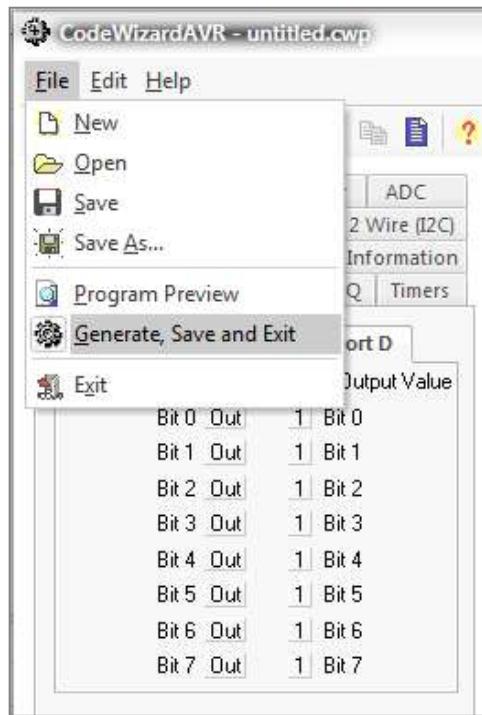


Gambar 4.11 Setting LCD pada PortB



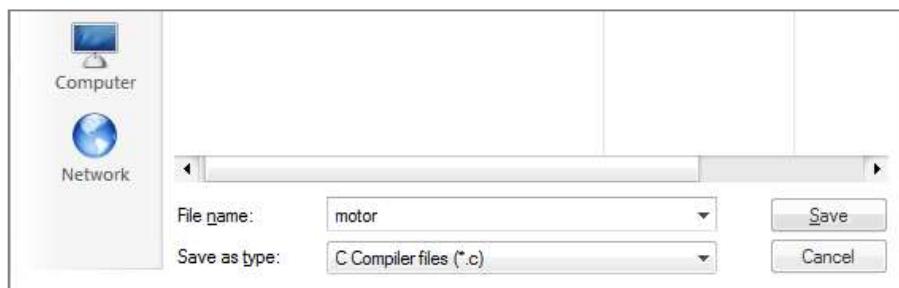
Gambar 4.12 Dialog Warning

Tekan tombol “No” pada dialog warning. Karena pada contoh ini tidak digunakan fasilitas lain maka setting CodeWizardAVR siap disimpan dalam file. Pada menu CodeWizardAVR, pilih File → Generate, Save and Exit, seperti yang ditunjukkan pada gambar berikut:



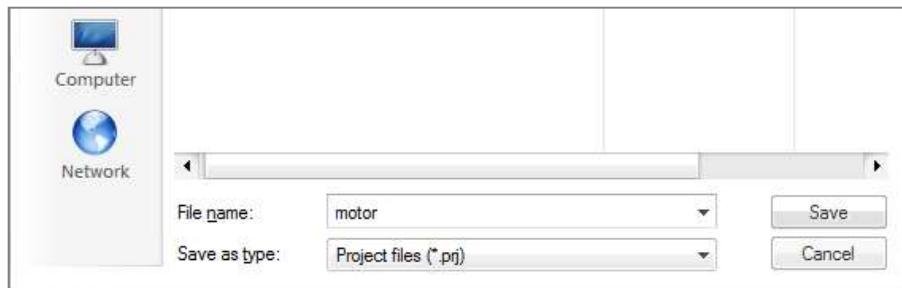
Gambar 4.13 Menyimpan setting

Setelah penekanan menu “Generate, Save and Exit”, akan muncul suatu dialog untuk melakukan set terhadap nama dan lokasi penyimpanan dari source file (*.c) seperti ditunjukkan pada gambar berikut:



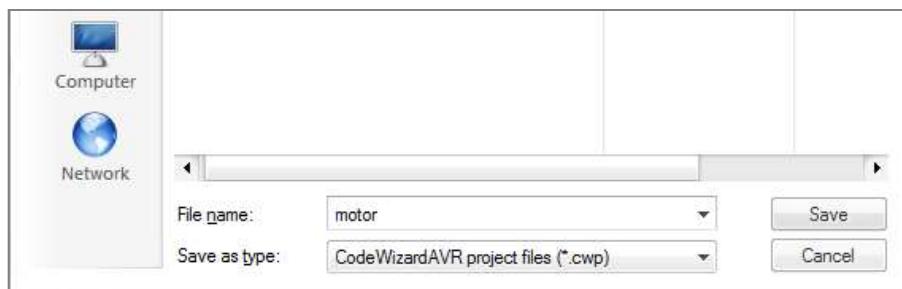
Gambar 4.14 Memberi nama pada file C (*.c)

Tekan tombol “Save” untuk menyetujui dan melanjutkan pada proses berikutnya. Setelah penekanan tombol “Save” akan muncul kembali dialog yang kedua, yaitu untuk melakukan set terhadap nama dan lokasi penyimpanan project file (*.prj) seperti ditunjukkan pada gambar berikut:



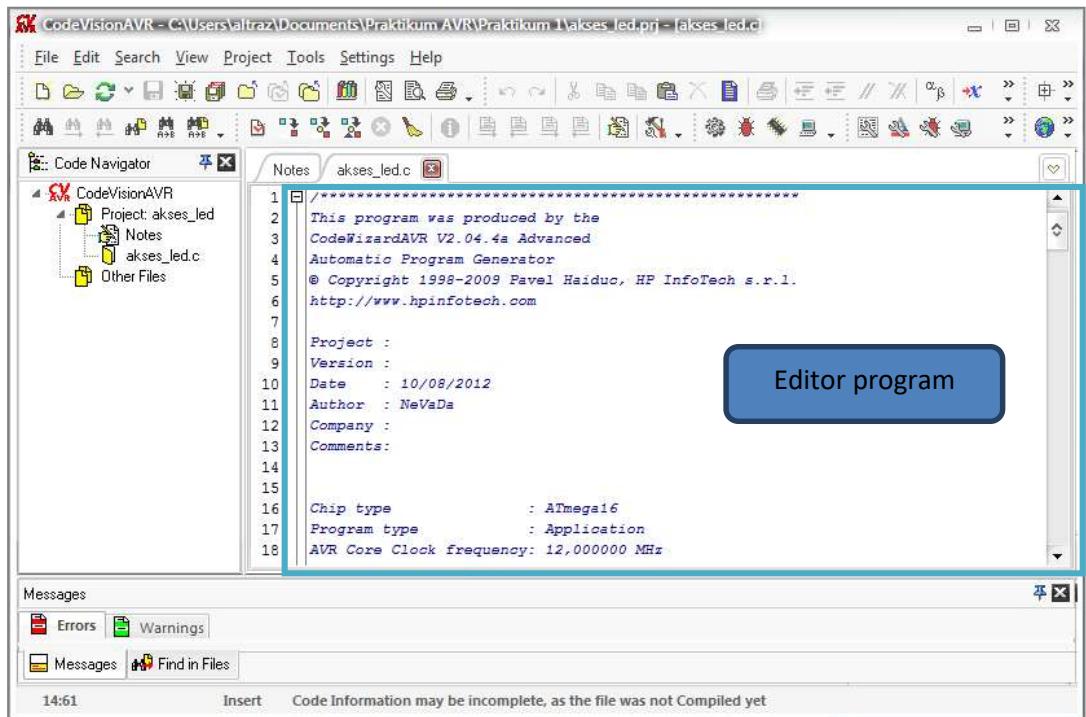
Gambar 4.15 Memberi nama project (*.prj)

Tekan tombol “Save” untuk menyetujui dan melanjutkan pada proses berikutnya. Setelah penekanan tombol “Save” akan muncul kembali dialog yang ketiga, yaitu untuk melakukan set terhadap nama dan lokasi penyimpanan CodeWizardProject file (*.cwp) seperti ditunjukkan pada gambar berikut:



Gambar 4.16 Memberi nama CodeWizardProject (*.cwp)

Tekan tombol “Save” untuk menyetujui dan melanjutkan. Selanjutnya akan muncul IDE editor program dari CodeVisionAVR. Anda siap untuk melakukan pembuatan program menggunakan CodeVisionAVR.



Gambar 4.17 Editor Program

2. Source Code

```

#include<mega16.h>
#include<delay.h>

#define A PORTA.4
#define B PORTA.5
#define EN PORTD.5
// Declare your global variables here

void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=Out Func4=Out Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=0 State4=0 State3=T State2=T State1=T
State0=T
PORTA=0x00;
DDRA=0x30;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTB=0x00;
DDRB=0x00;

```

```

// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func7=In Func6=In Func5=Out Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=0 State4=T State3=T State2=T State1=T
State0=T
PORTD=0x00;
DDRD=0x20;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization

```

```

// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

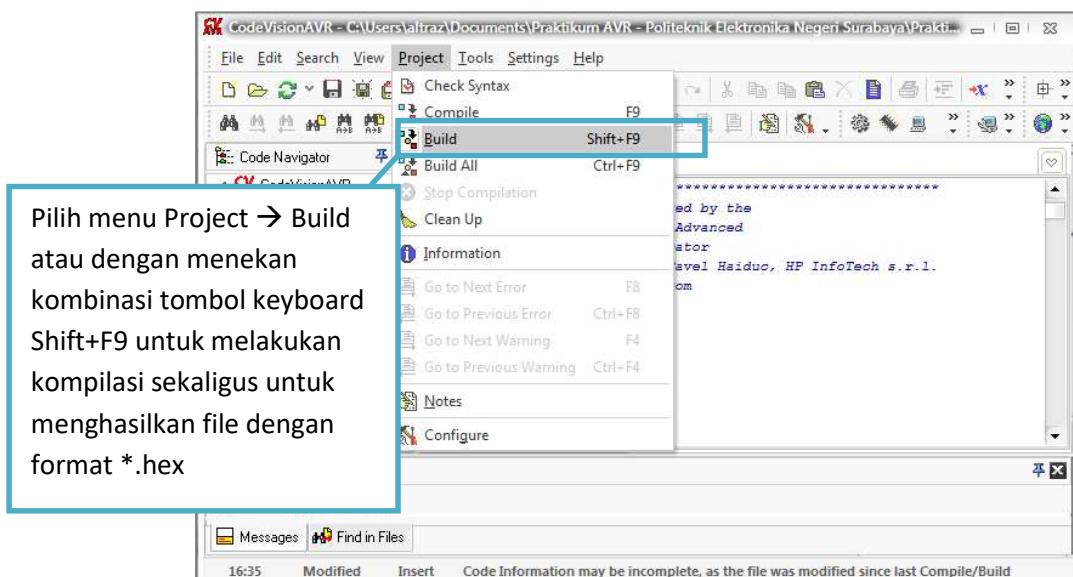
// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

while (1)
{
    // Place your code here
    // frekuensi = 200 Hz, t total = 1/frekuensi = 5 ms, t total = t on + t off = 5 ms
        A=0; B=1; EN=1; delay_us(2000);      //motor on ccw selama 2000us -->t on
        A=0; B=1; EN=0; delay_us(3000);      //motor off selama 3000us -->t off

    };
}

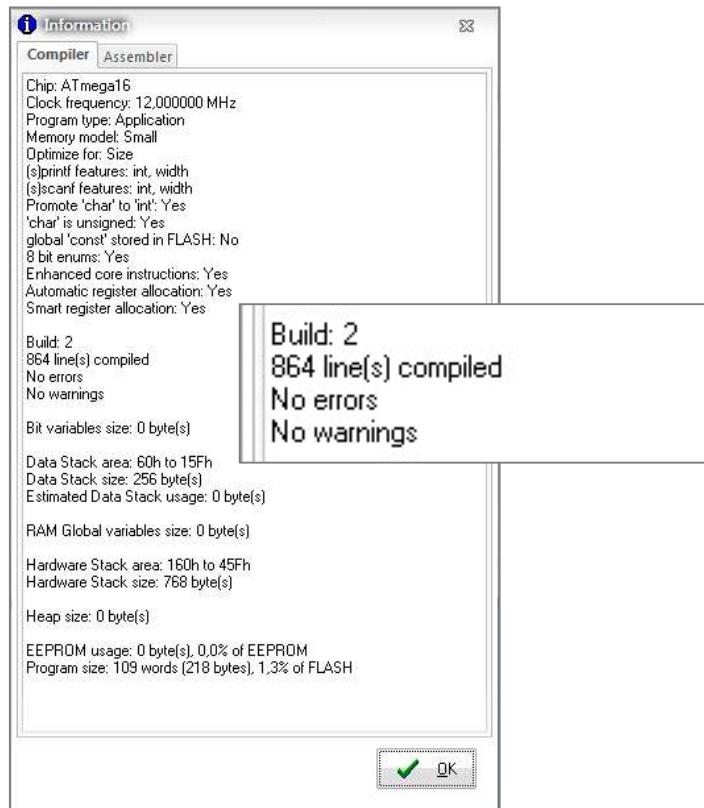
```

3. Setelah pembuatan program selesai, maka proses selanjutnya yaitu lakukan kompilasi terhadap kode program tersebut sekaligus untuk menghasilkan file dengan format hexadecimal (*.hex). Hal ini dapat dilakukan dengan memilih menu Project → Build (Shift+F9) seperti ditunjukkan pada gambar berikut :



Gambar 4.18 Build Source

Apabila kode program sudah benar (tanpa error) dan tidak menyalahi struktur pemrograman bahasa C, maka akan muncul dialog informasi seperti ditunjukkan pada gambar berikut:



Gambar 4.19 Dialog informasi status kompilasi

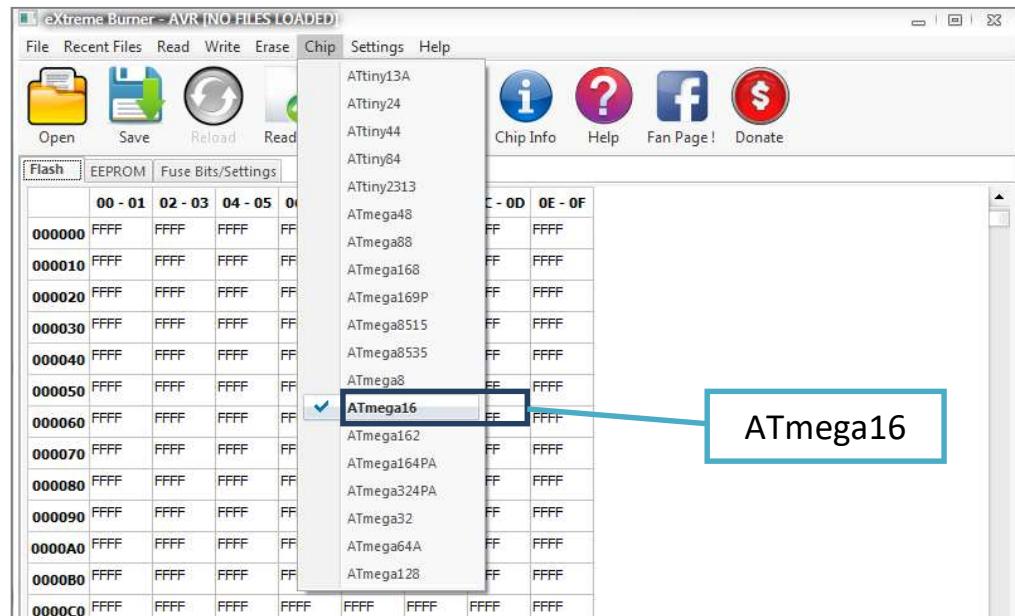
4. Setelah diperoleh file dengan format *.hex, maka lakukan download file *.hex ke dalam mikrokontroler. Pertama, jalankan program eXtreme Burner - AVR dengan cara melakukan klik ganda pada shortcut icon eXtreme Burner– AVR pada desktop



Gambar 4.20 Shortcut eXtreme Burner - AVR

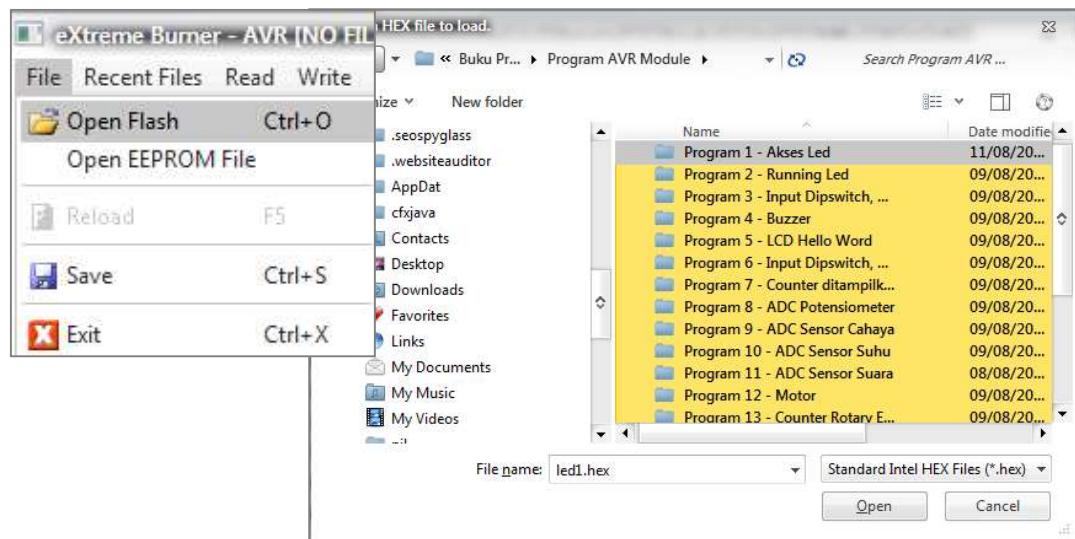
Selanjutnya akan muncul tampilan utama dari program eXtreme Burner – AVR. Pada praktikum ini menggunakan mikrokontroler keluarga AVR dengan tipe IC ATMega16. Oleh karena itu, lakukan pengaturan terhadap tipe IC terlebih dahulu

dengan memilih menu Chip → ATmega16 seperti ditunjukkan pada gambar berikut:



Gambar 4.21 Tipe Mikrokontroler

Setelah itu, lakukan open file *.hex dengan memilih menu File → Open Flash (Ctrl+O) seperti ditunjukkan pada gambar berikut:



Gambar 4.22 Open Flash

Lakukan pencarian terhadap lokasi file *.hex Anda, untuk selanjutnya tekan tombol "Open" apabila sudah ditemukan file yang dimaksud. Setelah itu akan

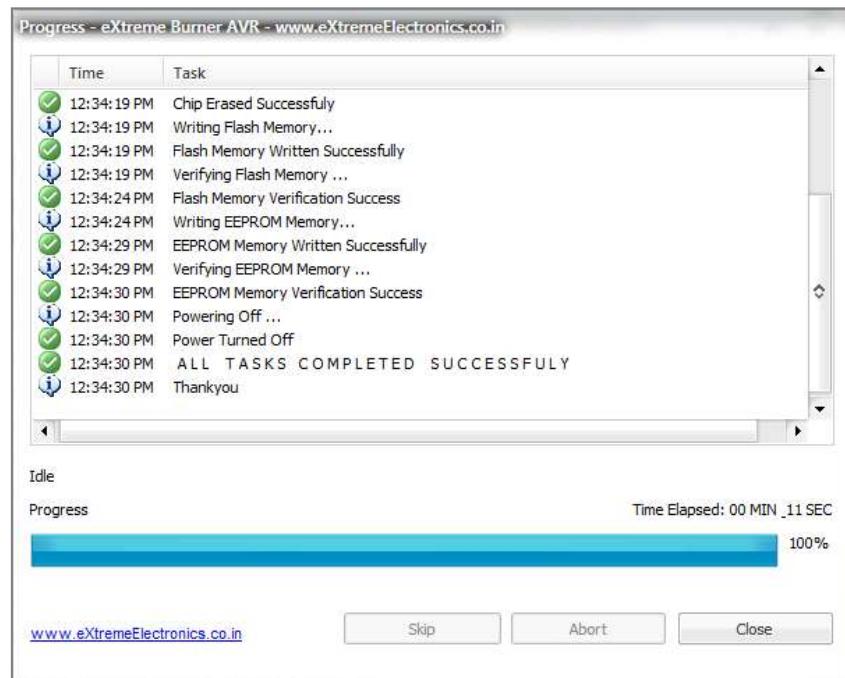
muncul pesan yang memberitahukan bahwa file *hex berhasil di-load program eXtreme Burner – AVR.



Gambar 4.23 Message



Selanjutnya tekan tombol **Write All** untuk memulai download ke dalam mikrokontroler Anda. Berikut ini merupakan tampilan apabila proses download ke dalam mikrokontroler berhasil dilakukan:



Gambar 4.24 Proses Berhasil

5. Selesai

B. MOTOR 2

Lakukan pembuatan project baru dengan pengaturan sama dengan project motor sebelumnya. Dan berikut merupakan source code untuk praktikum motor ke-2:

Source Code :

```
#include<mega16.h>

#define A      PORTA.4
#define B      PORTA.5
#define EN     PORTD.5

// Declare your global variables here

void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=Out Func4=Out Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=0 State4=0 State3=T State2=T State1=T
State0=T
PORTA=0x00;
DDRA=0x30;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTB=0x00;
DDRB=0x00;

// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func7=In Func6=In Func5=Out Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=0 State4=T State3=T State2=T State1=T
State0=T
PORTD=0x00;
DDRD=0x20;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 1382,400 kHz
```

```

// Mode: Fast PWM top=ICR1
// OC1A output: Non-Inv.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x82;
TCCR1B=0x1A;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x1A;
ICR1L=0xFF;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

while (1)
{
// Place your code here
// frekuensi motor = 200 Hz, ICR = (Clock value / frek motor) - 1 =
// (138400 / 200) - 1 = 6911 = 0x1AFF
A = 0; B=1; OCR1A = 4000;//kecepatan 0 - 6911;

}

```

Lakukan kompilasi seperti project sebelumnya. Kemudian lakukan download pada mikrokontroler ATMega16.

4.6. Hasil Percobaan

Amati hasil praktikum Anda

4.7. Tugas

Buat Program untuk pengaturan kecepatan motor DC dengan kecepatan berjenjang tiga : rendah, sedang, dan tinggi. Nilai RPM dapat ditentukan berdasarkan setting yang anda tentukan.

PERCOBAAN 5

COUNTER ROTARY ENCODER

PERCOBAAN 5

COUNTER ROTARY ENCODER

5.1. Tujuan

1. Mahasiswa dapat memahami penerapan sensor posisi dengan rotary encoder
2. Mahasiswa dapat memahami penggunaan timer pada ATMega16
3. Mahasiswa dapat mengakses data rotary encoder
4. Mahasiswa dapat membuat aplikasi sederhana yang mengimplementasikan penggunaan rotary encoder

5.2. Dasar Teori

1. Rotary Encoder

Rotary encoder, atau disebut juga Shaft encoder, merupakan perangkat elektro-mekanikal yang digunakan untuk mengkonversi posisi anguler (sudut) dari shaft (lubang) atau roda ke dalam kode digital, menjadikannya semacam tranduser. Perangkat ini biasanya digunakan dalam bidang robotika, perangkat masukan komputer (seperti optomekanikal mouse dan trackball), serta digunakan dalam kendali putaran radar, dll.

- **Rotary Encoder Absolute**

Tipe ini (rotay encoder absolut) menghasilkan kode digital yang unik/khas untuk masing-masing beda sudut poros. Plat baja dipotong dengan bentuk tertentu kemudian ditempelkan ke piringan/cakram dengan penyekat dimana terpasang kuat dengan poros (shaft). Saat piringan berputar, beberapa kontaknya menyentuh plat baja, dan kontak yang lain tak menyentuh plat (yang berlubang). Plat baja tersebut terhubung dengan sumber arus listrik, dan masing-masing kontak terhubung ke sensor elektrik. Bentuk potongan plat baja tersebut dirancang sedemikian rupa sehingga memungkinkan masing-masing posisi poros membentuk kode biner yang unik dimana beberapa kontak terhubung ke sumber arus (switch ON) dan yang lain tak terhubung (switch OFF).



Gambar 5.1 Rotary Encoder Absolute

- **Rotary Encoder Incremental / Relative**

Rotary Encoder Relatif (sering disebut juga Incremental Encoder) digunakan ketika metode pengkodean absolut tidak bisa digunakan (disebabkan ukuran dan bentuk piringan/cakram). Metode ini juga menggunakan piringan yang dipasang pada poros, tetapi ukuran piringan/cakram lebih kecil, dengan jumlah garis radial yang banyak, seperti jeruji roda. Sebuah saklar optik, seperti photodiode, menghasilkan pulsa listrik. Kemudian rangkaian kontrol elektronika menghitung pulsa untuk menerjemahkan sudut putar dari poros.



Gambar 5.2 Rotary Encoder Relative

2. Timer/Counter Mikrokontroler ATMEGA16

Mikrokontroler AVR ATMEGA16 memiliki tiga buah Timer/Counter, yaitu: Timer 0 (8 bit), Timer 1 (16 bit) dan Timer 2 (8 bit). Timer/Counter 1 mempunyai keunggulan dibanding Timer/Counter 0 atau 2, namun cara mengatur Timer 0, 1, 2 sama saja, yaitu pada masing-masing registernya. Timer/Counter 1 dapat menghitung sampai dengan 65536 Timer/Counter 0 atau 2 hanya sampai dengan 256. Selain itu, Timer 1 ini memiliki mode operasi sebanyak 16 mode(Tabel 2.8). Register pada Timer ini dibagi menjadi beberapa register dengan fungsi khusus, yaitu: control register A, control register B dan interrupt mask. Register – register pada

Timer/Counter 1 yang berfungsi untuk mengatur timer dan mode operasinya. Register tersebut mempunyai fungsi masing-masing sebagai berikut.

a. Timer/Counter 1 Control Register A (TCCR1A)

Tabel 1 Register TCCR1A

<i>Bit</i>	7	6	5	4	3	2	1	0
TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10

Keterangan:

- **Bit 7 dan 6** : Compare Output untuk kanal A
- **Bit 5 dan 4** : Compare Output untuk kanal B

Bit COM1 ini mempunyai Compare Output Mode pada setiap mode operasinya. Mode tersebut mempengaruhi pin I/O OC1 A dan B.

Tabel 2 Compare Output Mode, Non-PWM

COM1A1/COM1B1	COM1A1/COM1B1	Deskripsi
0	0	<i>Normal Port Operation, OC1A/OC1B disconnected</i>
0	1	<i>Toggle OC1A/OC1B on compare match</i>
1	0	<i>Clear OC1A/OC1B on compare match (low level)</i>
1	1	<i>Set OC1A/OC1B on compare match (high level)</i>

Tabel 3 Compare Output Mode
Phase Correct dan Phase Correct & Frequency PWM

COM1A1/COM1B1	COM1A1/COM1B1	Deskripsi
0	0	<i>Normal Port Operation, OC1A/OC1B disconnected</i>
0	1	<i>Toggle OC1A on compare match, OC1B disconnected</i>
1	0	<i>Clear OC1A/OC1B on compare match when up-counting, set OC1A/OC1B on compare match when down-counting</i>
		<i>OC1A/OC1B on compare match when down-counting</i>
1	1	<i>Clear OC1A/OC1B on compare match when up-counting, set OC1A/OC1B on compare match when down-counting</i>

Bit 3 : Force Output untuk kanal A

Bit 2 : Force Output untuk kanal B

Bit 1 dan 0 : Waveform Generation Mode

Mode operasi sebanyak 16 mode, diatur dalam bit WGM ini. Mode operasi tersebut ditunjukkan oleh Tabel 2.8 di bawah ini.

Tabel 4 Deskripsi Bit WGM

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Mode Operasi	TOP	Update of OCRn	TOVn 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

b. Timer/Counter Control Register 1 B (TCCR1B)

Tabel 5 TCCR1B

Bit	7	6	5	4	3	2	1	0
TCCR1B	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10

Keterangan:

- Bit 7** : Input Capture Noise Canceler, ketika bit ini diset 1(high) maka Noise Canceler aktif dan masukkan dari Input Capture Pin (ICP1) terfilter.

- **Bit 6** : Input Capture Edge Select, bit ini digunakan untuk trigger yang disebabkan oleh edge ICP1. Jika bit ini diset 1 maka sebuah rising edge (positif) akan men-trigger capture, Jika bit ini diset 0 maka sebuah falling edge (negatif) akan men-trigger capture.
 - **Bit 5** : Reserved, bit ini akan digunakan pada tahap pengembangan selanjutnya.
 - **Bit 4 dan 3** : lihat deskripsi register TCCR1A.
 - **Bit 2, 1 dan 0** : Clock Select, bit ini digunakan untuk memilih jenis sumber clock untuk digunakan pada suatu timer/counter.
- c. **TCNT1**, digunakan untuk menyimpan nilai timer yang diinginkan. TCNT1 dibagi menjadi 2 register 8 bit, yaitu TCNT1H dan TCNT1L.

Tabel 6 Register TCNT1

Bit	7	6	5	4	3	2	1	0
TCNT1H								TCNT1[15:8]
TCNT1L								TCNT1[7:0]

- d. **TIMSK dan TIFR**, Timer Interrupt Mask Register (TIMSK) dan Timer Interrupt Flag (TIFR) digunakan untuk mengendalikan interrupt mana yang diaktifkan, dengan cara melakukan setting pada TIMSK dan untuk mengetahui interrupt mana yang sedang terjadi.

Tabel 7 Register TIMSK

Bit	7	6	5	4	3	2	1	0
TIMSK	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0

- e. **OCR1n**, Output Compare Register Timer 1 n ($n = A, B$) merupakan register yang digunakan untuk membangkitkan interupsi eksternal dengan melakukan perbandingan (Output Compare) atau juga dapat digunakan untuk membangkitkan bentuk gelombang (PWM). Fungsi tersebut di atas dikeluarkan oleh pin OC1n ($n = A, B$).

Tabel 8 Register OCR1n

Bit	7	6	5	4	3	2	1	0
OCR1nH								OCR1n[15:8]
OCR1nL								OCR1n[7:0]

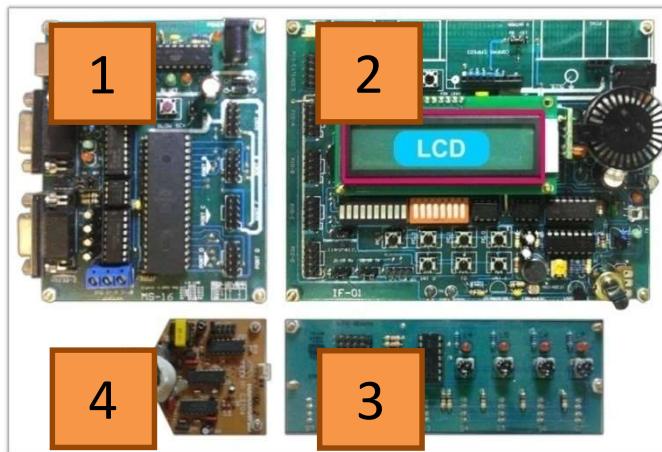
5.3. Peralatan

- 1 set PC yang dilengkapi dengan software CodeVision AVR.

- 1 set development board AVR ATmega16
- 1 power-supply +9VDC

5.4. Modul I/O

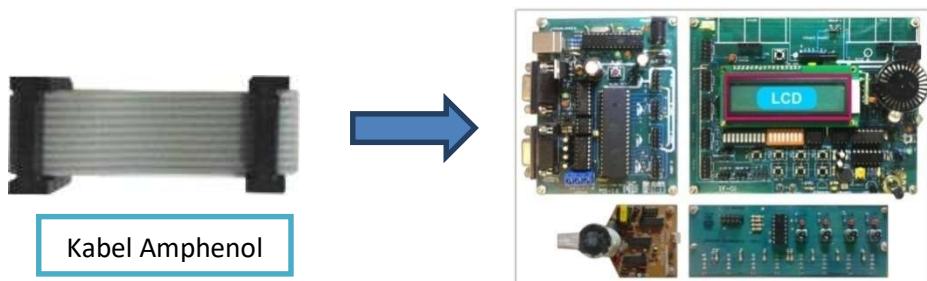
Berikut ini merupakan development board AVR ATmega16 secara keseluruhan:



Gambar 5.3 Development board keseluruhan

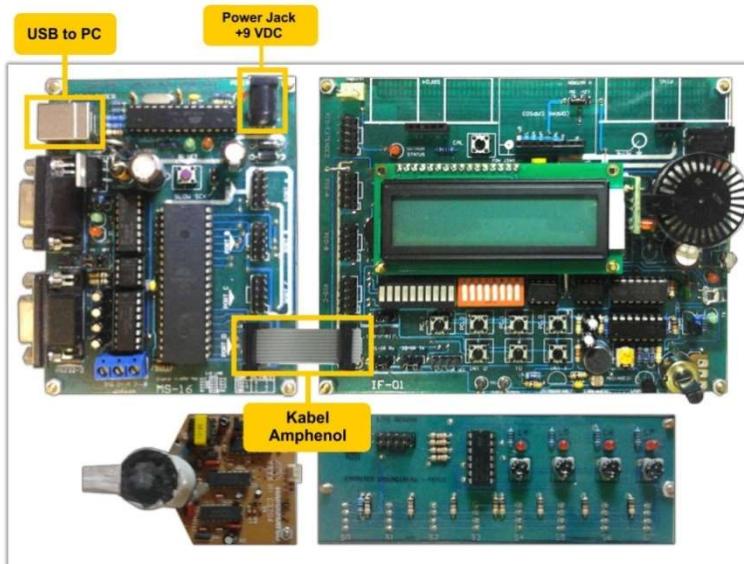
- **Konfigurasi modul pada percobaan ini :**

Pasangkan kabel amphenol pada modul development untuk menghubungkan system minimum ATMega16 dengan Training Board.



Gambar 5.4 Komponen Training board

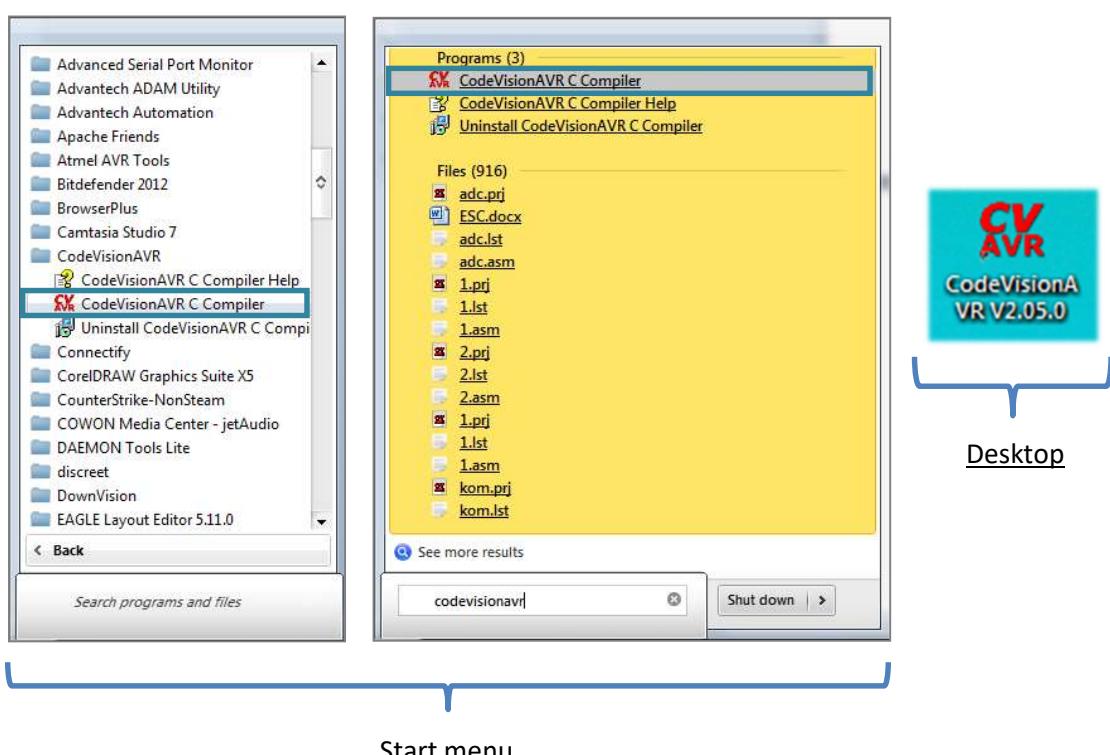
Sehingga menjadi seperti ditunjukkan pada gambar berikut:



Gambar 5.5 Komponen Training board

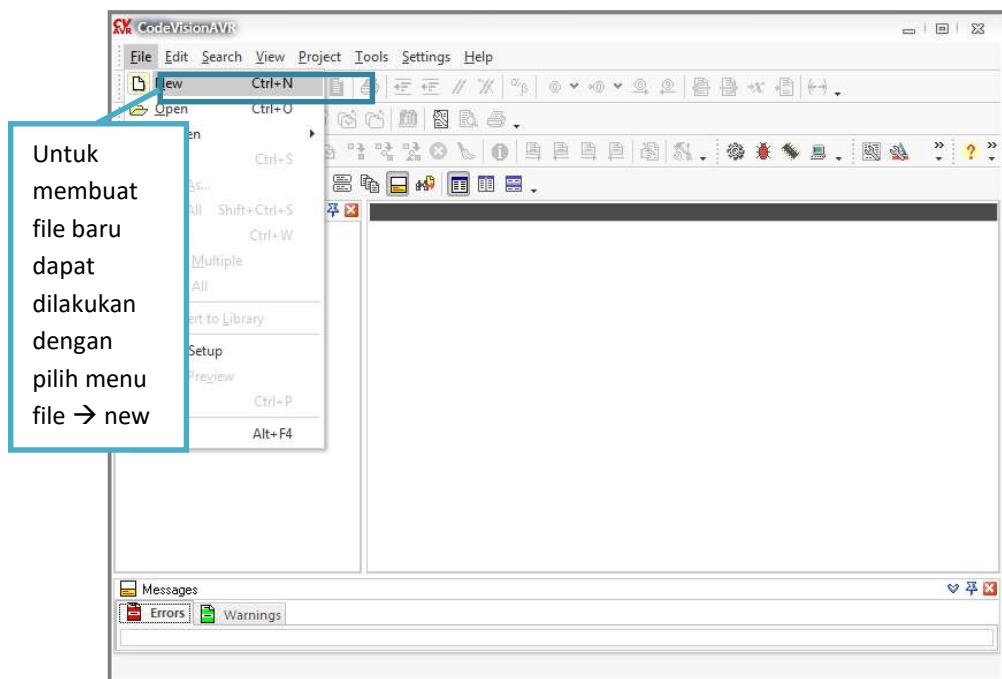
5.5. Prosedur Praktikum

1. Jalankan aplikasi CodeVisionAVR dengan cara melakukan klik ganda pada shortcut icon CodeVisionAVR pada desktop atau dengan cara mencari shortcut program pada start menu system operasi windows.



Gambar 5.6 Shortcut CodeVisionAVR

Untuk memulai membuat project baru, pada menubar, pilih File → New, seperti yang ditunjukkan oleh gambar berikut:



Gambar 5.7 Membuat File baru

Setelah itu akan muncul suatu dialog yang mengharuskan *user* untuk memilih antara pembuatan source file atau project. Dalam setiap percobaan pada praktikum ini, dibuat project baru untuk setiap percobaan, oleh karena itu *user* sebaiknya melakukan *check* pada checkbox project dan dilanjutkan dengan menekan tombol “OK”



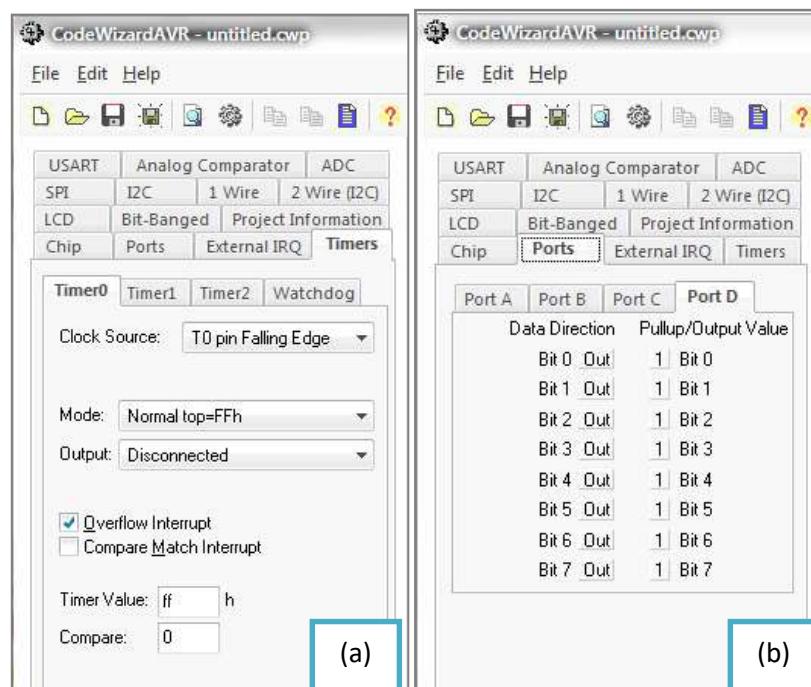
Gambar 5.8 Membuat project baru

Berikutnya akan keluar dialog yang menanyakan Anda apakah akan membuat project baru. Klik tombol “Yes” untuk menyetujui.



Gambar 5.9 Memilih untuk membuat project baru pada CodeVisionAVR

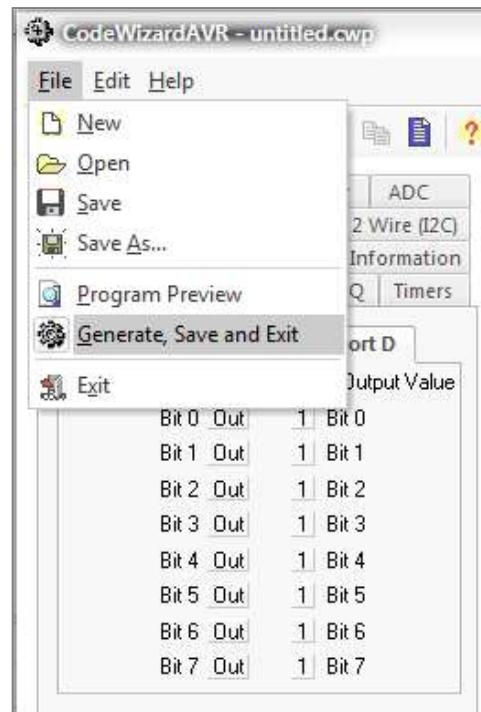
Setelah itu lakukan set pada CodeWizardAVR. Untuk chip diset dengan nama chip ATmega16 dengan clock 12 MHz, seperti ditunjukkan pada gambar 1.11(a). Lakukan Set Timer0 dengan Clock Source = “T0 pin Falling Edge”, kemudian berikan tanda check pada “Overflow Interrupt”. Berikutnya Anda akan menginisialisasi Port D yang terhubung dengan LED. LED merupakan modul output. Pada tab Port bagian Port D, ubah bagian Data Direction menjadi OUT dengan nilai output sama dengan 1 seperti pada Gambar 1.11(b). Artinya Port D digunakan sebagai port output dengan nilai awal satu setelah kondisi reset.



Gambar 5.10 (a) Setting Timer 0

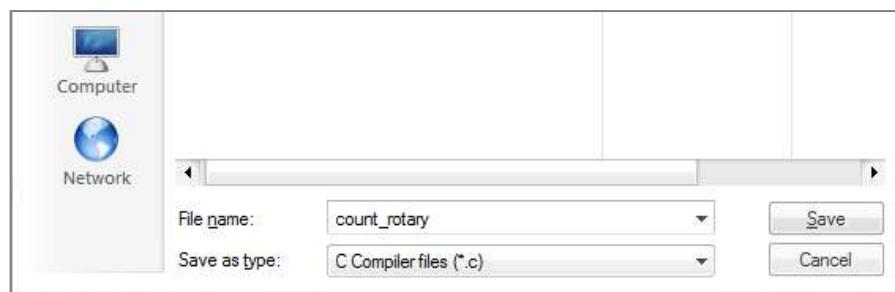
(b) Setting Port D sebagai pin output

Karena pada contoh ini tidak digunakan fasilitas lain maka setting CodeWizardAVR siap disimpan dalam file. Pada menu CodeWizardAVR, pilih File → Generate, Save and Exit, seperti yang ditunjukkan pada gambar berikut:



Gambar 5.11 Menyimpan setting

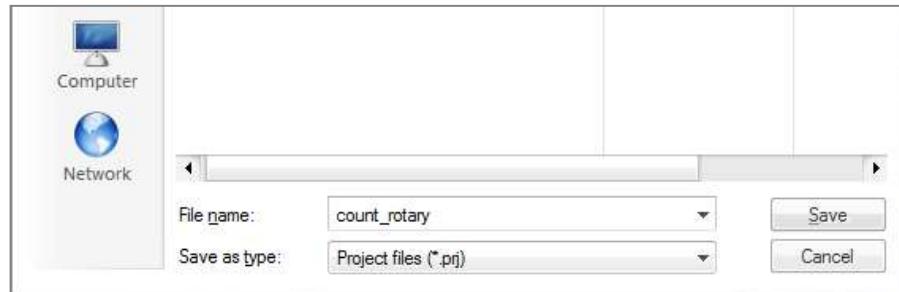
Setelah penekanan menu “Generate, Save and Exit”, akan muncul suatu dialog untuk melakukan set terhadap nama dan lokasi penyimpanan dari source file (*.c) seperti ditunjukkan pada gambar berikut:



Gambar 5.12 Memberi nama pada file C (*.c)

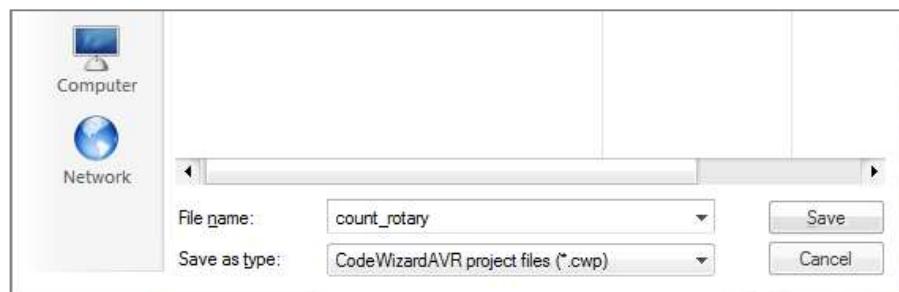
Tekan tombol “Save” untuk menyetujui dan melanjutkan pada proses berikutnya. Setelah penekanan tombol “Save” akan muncul kembali dialog yang kedua, yaitu

untuk melakukan set terhadap nama dan lokasi penyimpanan project file (*.prj) seperti ditunjukkan pada gambar berikut:



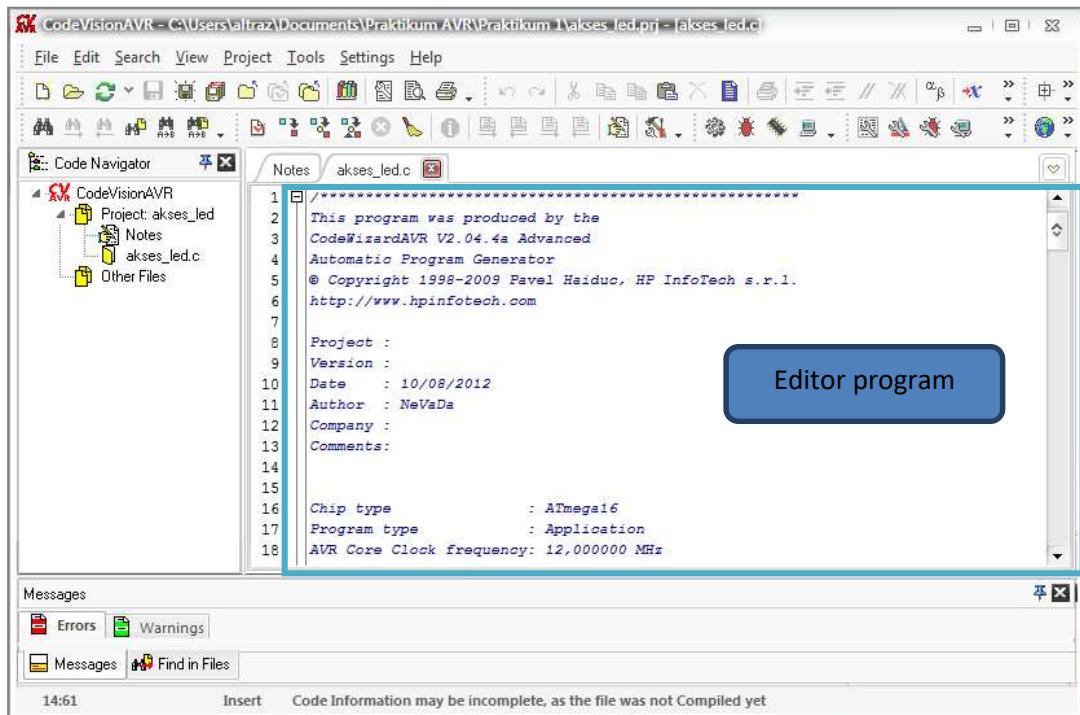
Gambar 5.13 Memberi nama project (*.prj)

Tekan tombol “Save” untuk menyetujui dan melanjutkan pada proses berikutnya. Setelah penekanan tombol “Save” akan muncul kembali dialog yang ketiga, yaitu untuk melakukan set terhadap nama dan lokasi penyimpanan CodeWizardProject file (*.cwp) seperti ditunjukkan pada gambar berikut:



Gambar 5.14 Memberi nama CodeWizardProject (*.cwp)

Tekan tombol “Save” untuk menyetujui dan melanjutkan. Selanjutnya akan muncul IDE editor program dari CodeVisionAVR. Anda siap untuk melakukan pembuatan program menggunakan CodeVisionAVR.



Gambar 5.15 Editor Program

2. Kemudian tambahkan definisi untuk mendefinisikan port. Letakkan definisi tersebut di bawah include dari header seperti berikut :

```
#include<mega16.h>
//tambahkan disini
#define led PORTD
```

3. Lakukan pendeklarasian variable global, letakkan di atas interrupt timer

```
unsignedchar count;
// Timer 0 overflow interrupt service routine
interrupt [TIM0_OVF] void timer0_ovf_isr(void) { ... }
```

4. Pada bagian interrupt [TIM0_OVF] tambahkan perintah yang telah diblok dengan warna kuning berikut :

```
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
// Place your code here
count++;
led = ~ count;
TCNT0 = 255;
}
```

5. Untuk lebih jelasnya, berikut adalah kode program secara keseluruhan:

```
#include<mega16.h>
#define led PORTD

unsignedchar count;
// Timer 0 overflow interrupt service routine
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
// Place your code here
    count++;
    led = ~ count;
    TCNT0 = 255;
}

// Declare your global variables here

void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTA=0x00;
DDRA=0x00;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTB=0x00;
DDRB=0x00;

// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out
Func1=Out Func0=Out
// State7=1 State6=1 State5=1 State4=1 State3=1 State2=1 State1=1
State0=1
PORTD=0xFF;
DDRD=0xFF;

// Timer/Counter 0 initialization
// Clock source: T0 pin Falling Edge
// Mode: Normal top=FFh
// OC0 output: Disconnected
```

```

TCCR0=0x06;
TCNT0=0xFF;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;
// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;
// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x01;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// Global enable interrupts
#asm("sei")

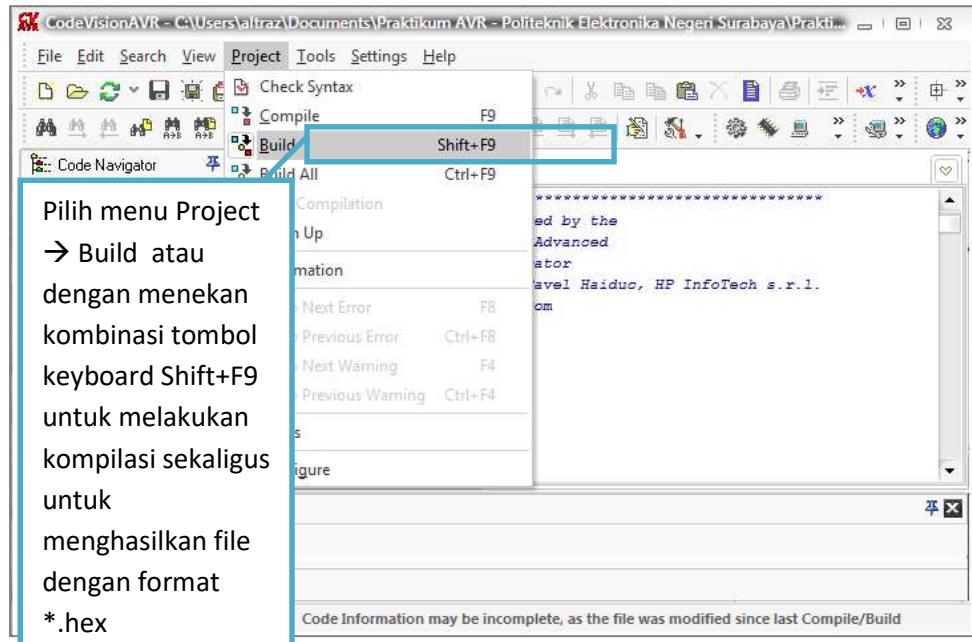
while (1)
{
// Place your code here

};

}

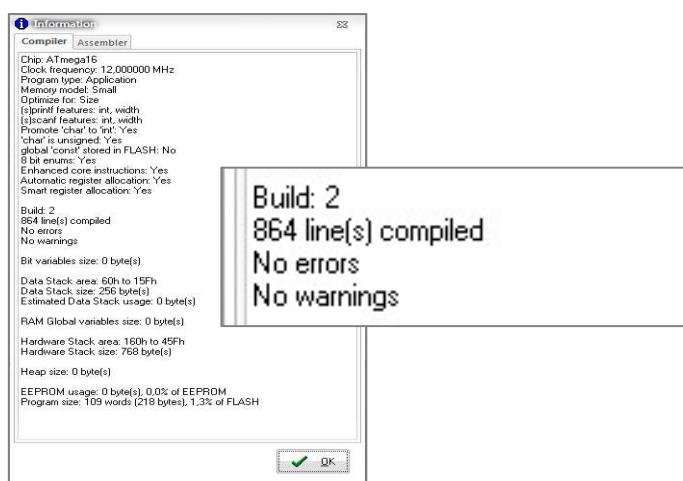
```

6. Setelah pembuatan program selesai, maka proses selanjutnya yaitu lakukan kompilasi terhadap kode program tersebut sekaligus untuk menghasilkan file dengan format hexadecimal (*.hex). Hal ini dapat dilakukan dengan memilih menu Project → Build (Shift+F9) seperti ditunjukkan pada gambar berikut :



Gambar 5.16 Build Source

Apabila kode program sudah benar (tanpa error) dan tidak menyalahi struktur pemrograman bahasa C, maka akan muncul dialog informasi seperti ditunjukkan pada gambar berikut:



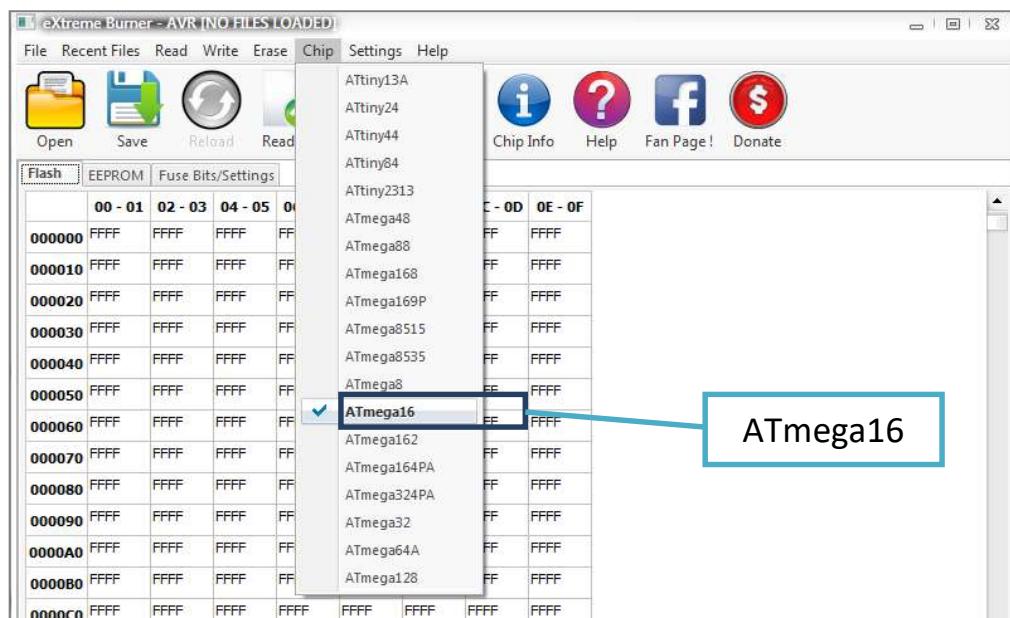
Gambar 5.17 Dialog informasi status kompilasi

7. Setelah diperoleh file dengan format *.hex, maka lakukan download file *.hex ke dalam mikrokontroler. Pertama, jalankan program eXtreme Burner - AVR dengan cara melakukan klik ganda pada shortcut icon eXtreme Burner – AVR pada desktop



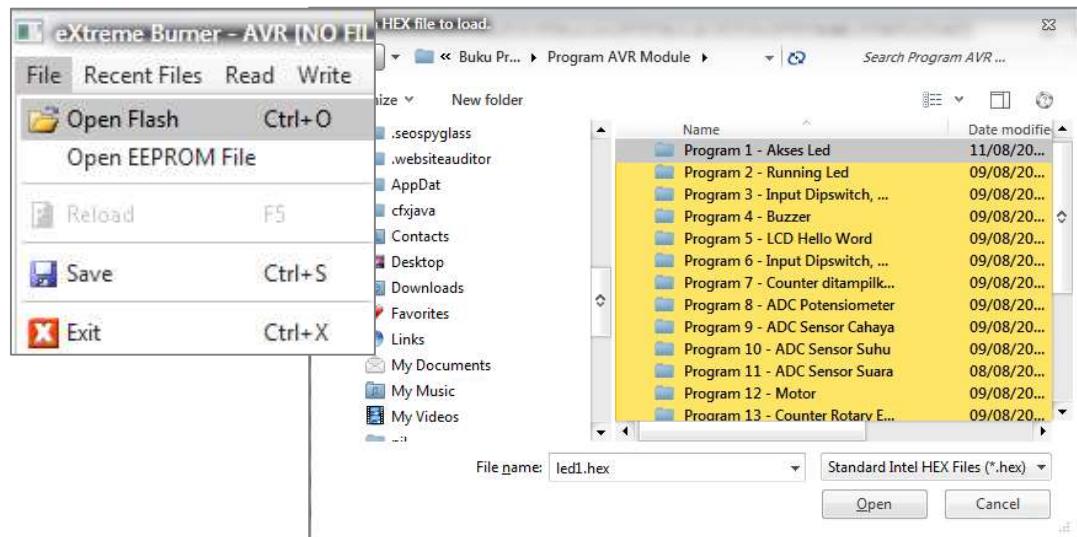
Gambar 5.18 Shortcut eXtreme Burner - AVR

Selanjutnya akan muncul tampilan utama dari program eXtreme Burner – AVR. Pada praktikum ini menggunakan mikrokontroler keluarga AVR dengan tipe IC ATMega16. Oleh karena itu, lakukan pengaturan terhadap tipe IC terlebih dahulu dengan memilih menu Chip → ATmega16 seperti ditunjukkan pada gambar berikut:



Gambar 5.19 Tipe Mikrokontroler

Setelah itu, lakukan open file *.hex dengan memilih menu File → Open Flash (Ctrl+O) seperti ditunjukkan pada gambar berikut:



Gambar 5.20 Open Flash

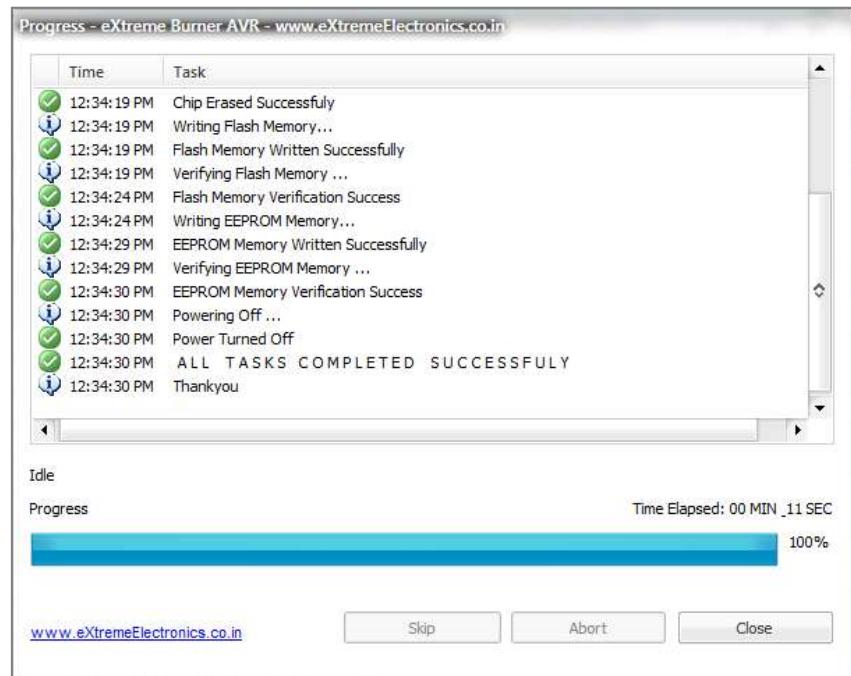
Lakukan pencarian terhadap lokasi file *.hex Anda, untuk selanjutnya tekan tombol “Open” apabila sudah ditemukan file yang dimaksud. Setelah itu akan muncul pesan yang memberitahukan bahwa file *hex berhasil di-load program eXtreme Burner – AVR.



Gambar 5.21 Message



Selanjutnya tekan tombol **Write All** untuk memulai download ke dalam mikrokontroler Anda. Berikut ini merupakan tampilan apabila proses download ke dalam mikrokontroler berhasil dilakukan:



Gambar 5.22 Proses Berhasil

8. Selesai

5.6. Hasil Percobaan

Amati hasil praktikum Anda

5.7. Tugas

Buat program untuk mengetahui posisi dengan 4 derajat: 0-90-180-270, tampilkan di LCD.

PERCOBAAN 6

**SENSOR JARAK :
ULTRASONIK SRF04**

PERCOBAAN 6

SENSOR JARAK : ULTRASONIK SRF04

6.1. Tujuan

1. Mahasiswa dapat mengetahui jarak obyek dengan sensor jarak ultrasonik
2. Mahasiswa dapat mengakses sensor ultrasonic SRF04 pada ATMega16
3. Mahasiswa dapat membuat aplikasi sederhana yang mengimplementasikan penggunaan sensor ultrasonic

6.2. Dasar Teori

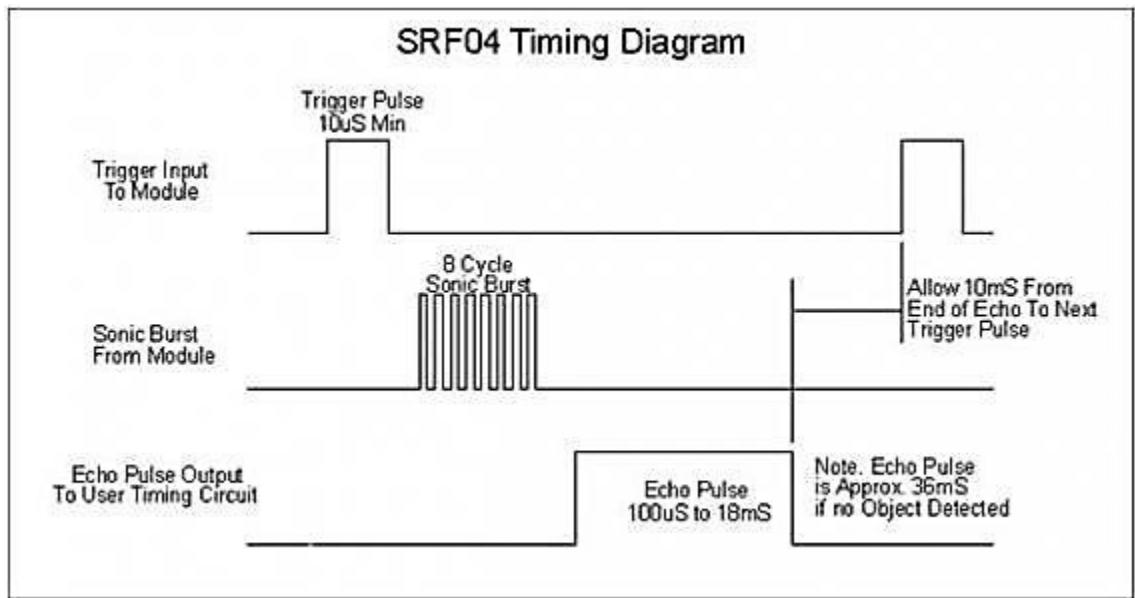
Sensor SFR04 adalah sensor ultrasonik yang diproduksi oleh Devantech. Sensor ini merupakan sensor jarak yang presisi. Dapat melakukan pengukuran jarak 3 cm sampai 3 meter dan sangat mudah untuk dihubungkan ke mikrokontroler menggunakan sebuah pin Input dan pin Output.



Gambar 6.1 Sensor ultrasonik SRF04

Sensor Devantech SRF-04 bekerja dengan cara memancarkan sinyal ultrasonik sesaat dan menghasilkan pulsa output yang sesuai dengan waktu pantul sinyal ultrasonik sesaat kembali menuju sensor. Dengan mengukur lebar pulsa pantulan tersebut jarak target didepan sensor dapat diketahui.

Untuk dapat memhami cara kerja dari sensor SRF04 ini perhatikan timming dari pulsa masukan dan keluaran sensor berikut ini:



Gambar 6.2 Timming SRF04

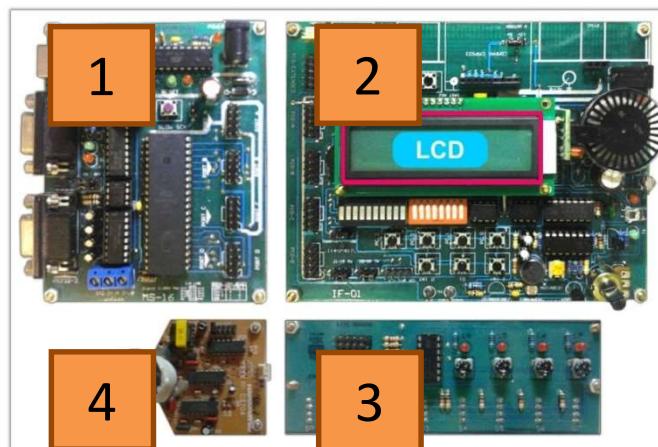
Berdasarkan data timming diagram sensor akan memberikan informasi jarak pembacaan dengan informasi berupa pulsa PWM dengan lebar $100\mu\text{s}$ sampai dengan 18mS .

6.3. Peralatan

- 1 set PC yang dilengkapi dengan software CodeVision AVR.
- 1 set development board AVR ATmega16
- 1 power-supply +9VDC

6.4. Modul I/O

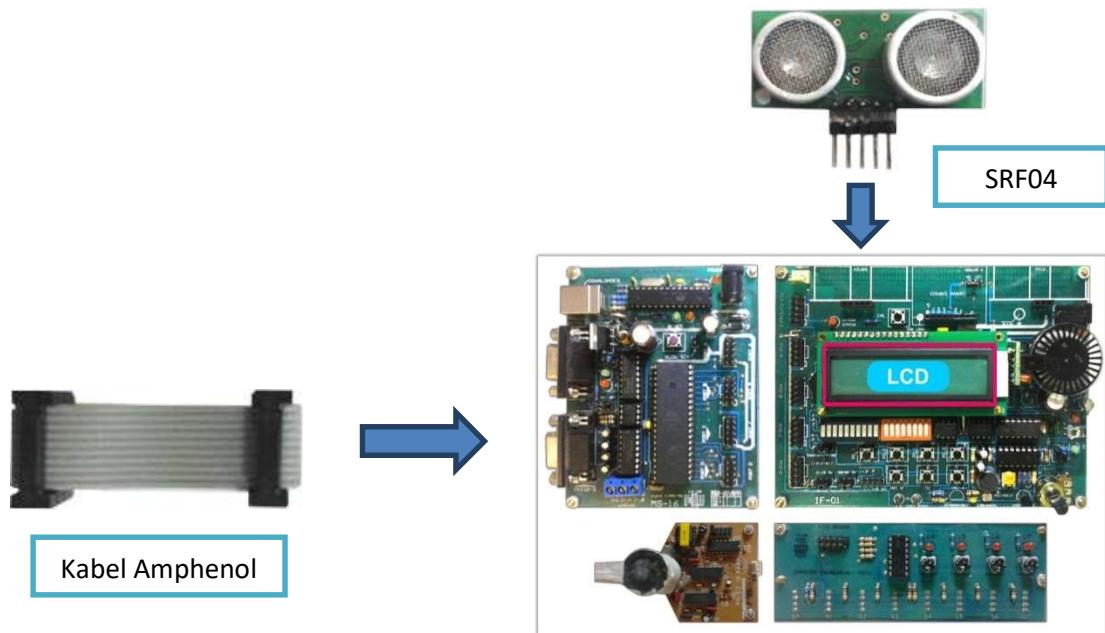
Berikut ini merupakan development board AVR ATmega16 secara keseluruhan:



Gambar 6.3 Development board keseluruhan

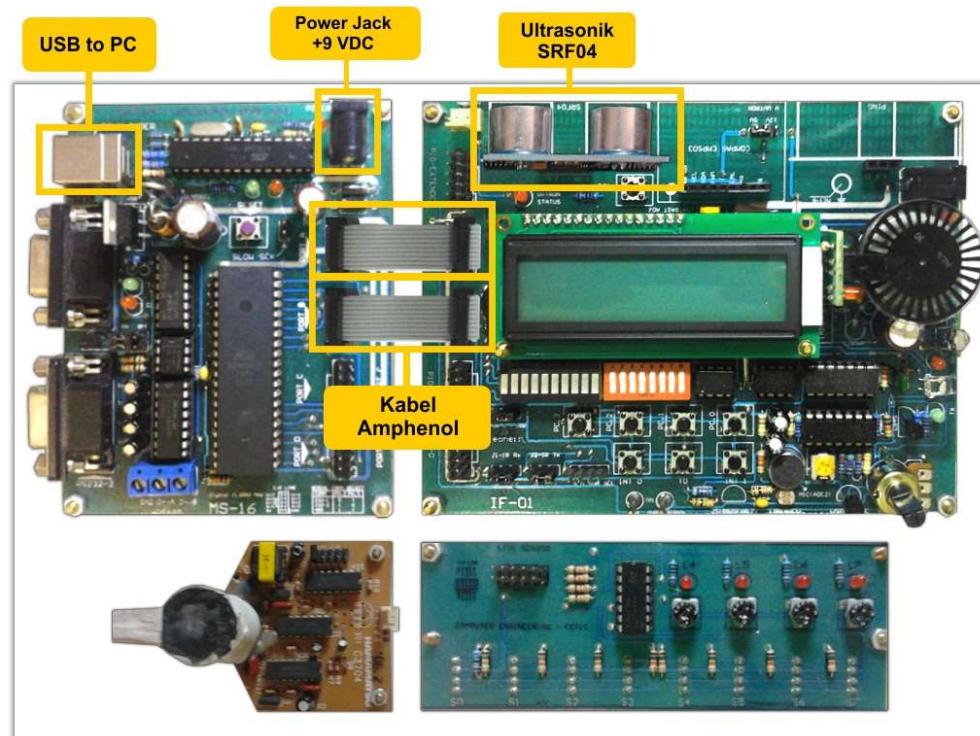
- **Konfigurasi modul pada percobaan ini :**

Pasangkan SRF04 dan kabel amphenol pada modul development untuk menghubungkan sistem minimum ATMega16 dengan Training Board.



Gambar 6.4 Komponen Training board

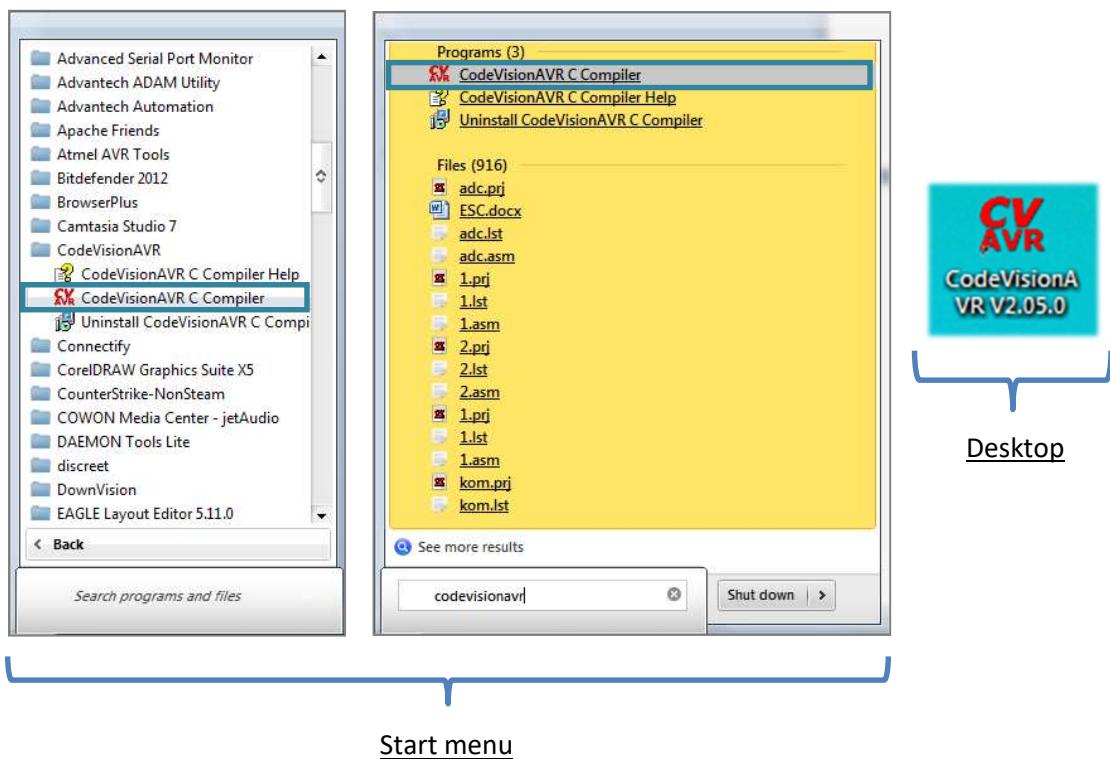
Sehingga menjadi seperti ditunjukkan pada gambar berikut:



Gambar 6.5 Komponen Training board

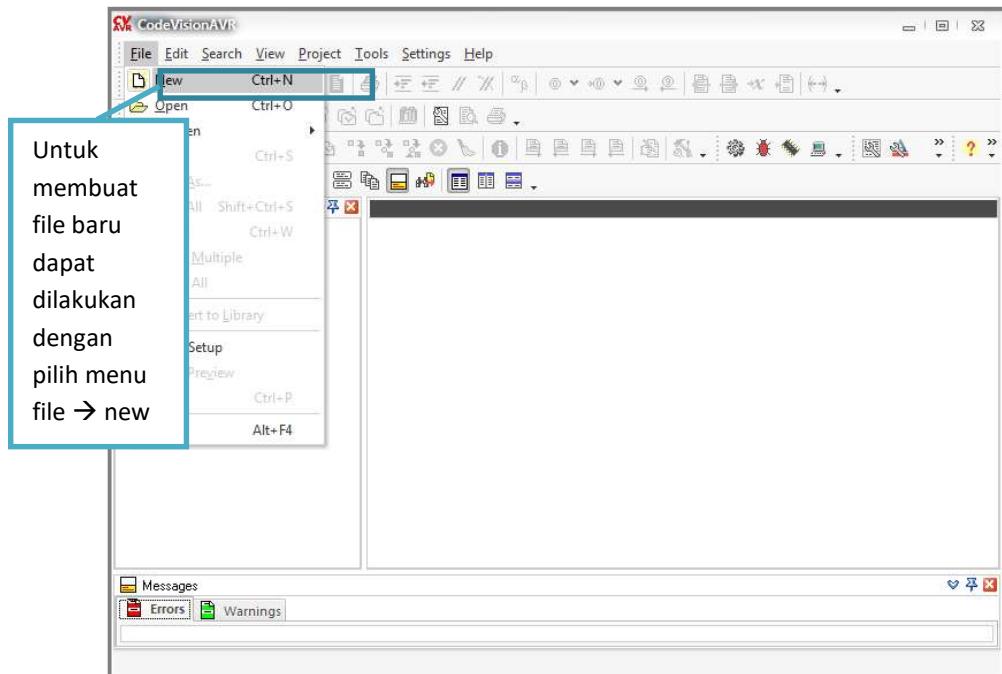
6.5. Prosedur Praktikum

1. Jalankan aplikasi CodeVisionAVR dengan cara melakukan klik ganda pada shortcut icon CodeVisionAVR pada desktop atau dengan cara mencari shortcut program pada start menu system operasi windows.



Gambar 6.6 Shortcut CodeVisionAVR

Untuk memulai membuat project baru, pada menubar, pilih File → New, seperti yang ditunjukkan oleh gambar berikut:



Gambar 6.7 Membuat File baru

Setelah itu akan muncul suatu dialog yang mengharuskan *user* untuk memilih antara pembuatan source file atau project. Dalam setiap percobaan pada praktikum ini, dibuat project baru untuk setiap percobaan, oleh karena itu *user* sebaiknya melakukan *check* pada checkbox project dan dilanjutkan dengan menekan tombol “OK”



Gambar 6.8 Membuat project baru

Berikutnya akan keluar dialog yang menanyakan Anda apakah akan membuat project baru. Klik tombol “Yes” untuk menyetujui.

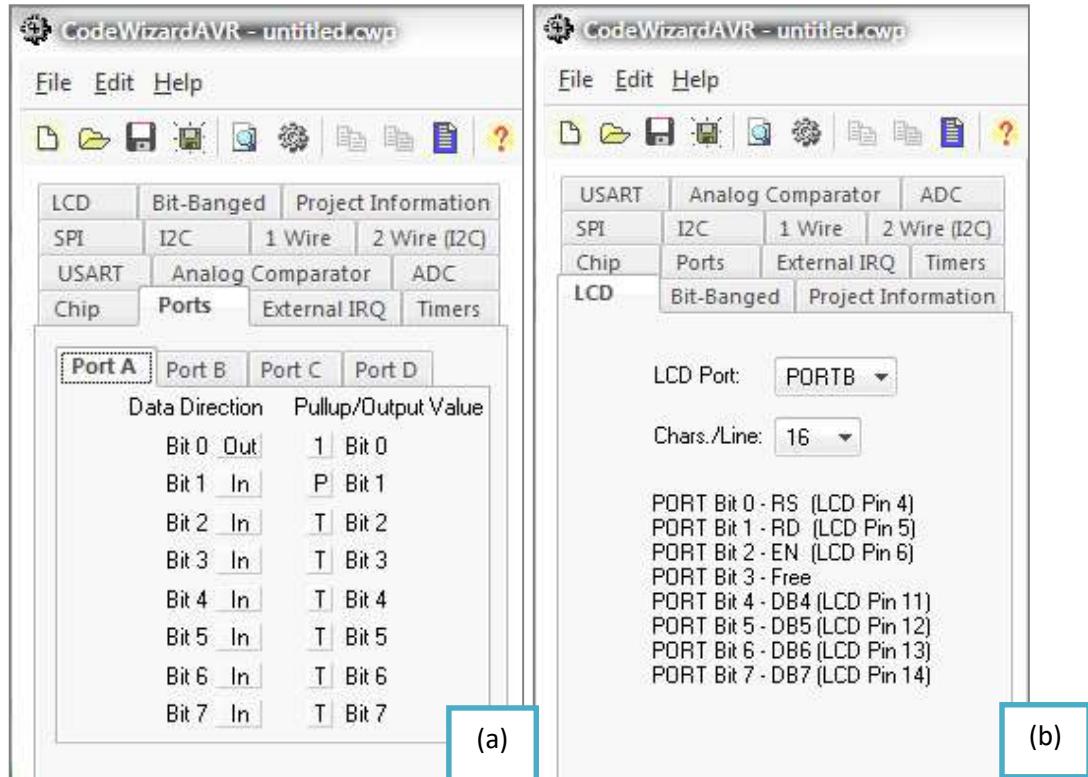


Gambar 6.9 Memilih untuk membuat project baru pada CodeVisionAVR

Setelah itu lakukan set pada CodeWizardAVR. Untuk chip diset dengan nama chip ATmega16 dengan clock 12 MHz.

Setelah itu Anda diharuskan menginisialisasi Port A. Port A bit ke-0 merupakan output dengan nilai awal 1. Port A bit ke-1 merupakan input (pullup).

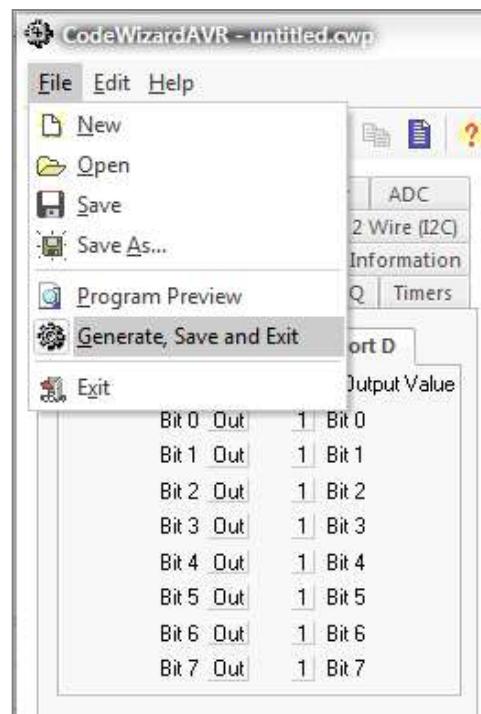
Berikutnya Anda akan menginisialisasi LCD pada PortB, yaitu dengan cara masuk pada tab LCD dan pilih item **PORTE** pada selectbox “LCD Port”. Set banyaknya karakter dari LCD yang diinginkan. Dalam hal ini menggunakan 16 karakter, oleh karena itu pilih item **16** pada selectbox “Chars/Line”



Gambar 6.10 (a) Setting Port A sebagai pin input dan output

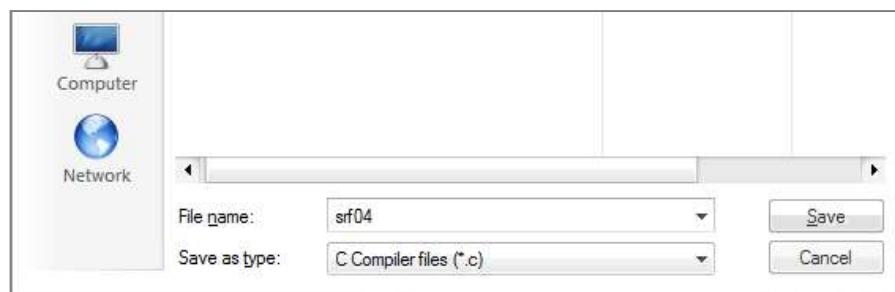
(b) Setting LCD pada PortB

Karena pada contoh ini tidak digunakan fasilitas lain maka setting CodeWizardAVR siap disimpan dalam file. Pada menu CodeWizardAVR, pilih File → Generate, Save and Exit, seperti yang ditunjukkan pada gambar berikut:



Gambar 6.11 Menyimpan setting

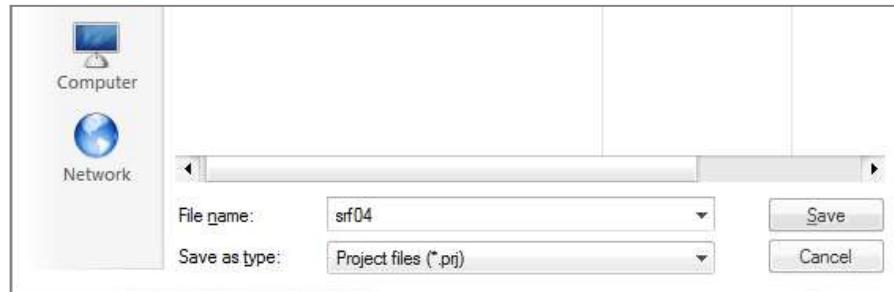
Setelah penekanan menu “Generate, Save and Exit”, akan muncul suatu dialog untuk melakukan set terhadap nama dan lokasi penyimpanan dari source file (*.c) seperti ditunjukkan pada gambar berikut:



Gambar 6.12 Memberi nama pada file C (*.c)

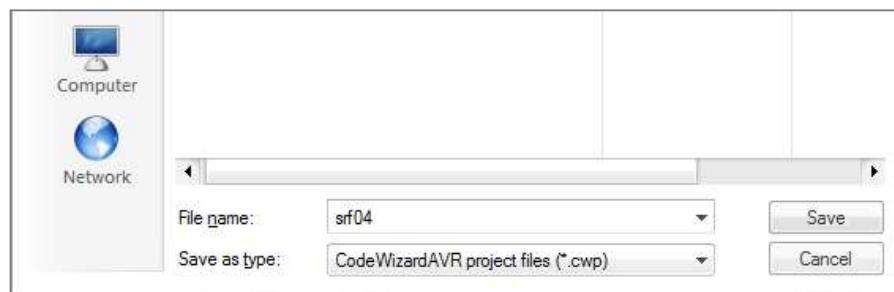
Tekan tombol “Save” untuk menyetujui dan melanjutkan pada proses berikutnya. Setelah penekanan tombol “Save” akan muncul kembali dialog yang kedua, yaitu

untuk melakukan set terhadap nama dan lokasi penyimpanan project file (*.prj) seperti ditunjukkan pada gambar berikut:



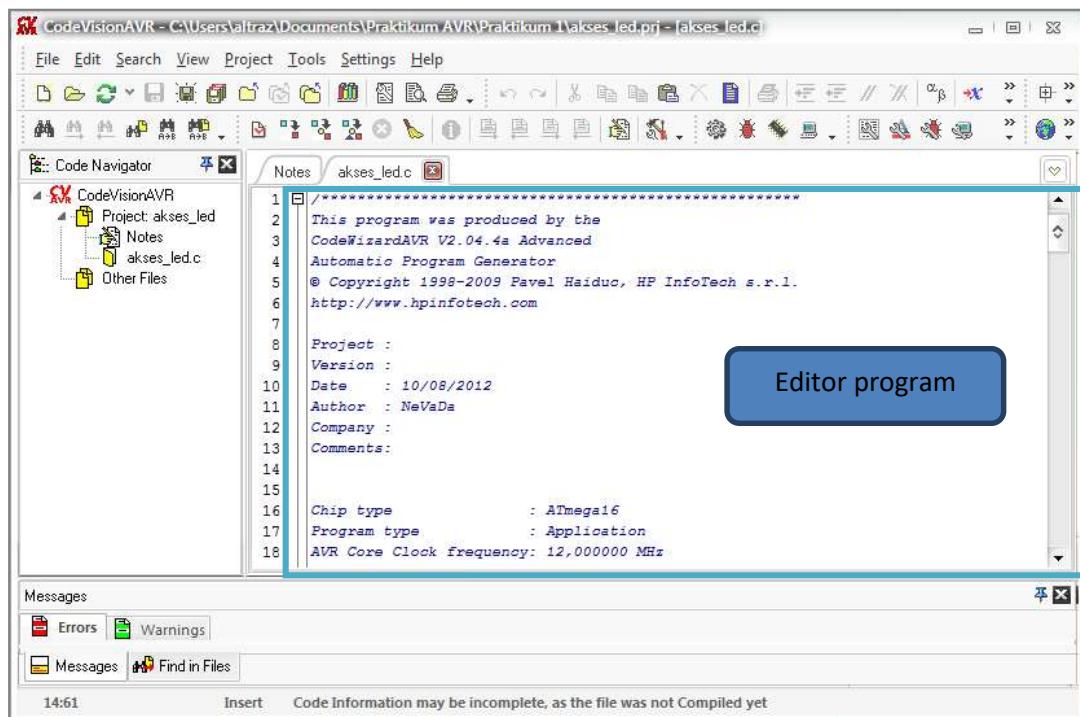
Gambar 6.13 Memberi nama project (*.prj)

Tekan tombol “Save” untuk menyetujui dan melanjutkan pada proses berikutnya. Setelah penekanan tombol “Save” akan muncul kembali dialog yang ketiga, yaitu untuk melakukan set terhadap nama dan lokasi penyimpanan CodeWizardProject file (*.cwp) seperti ditunjukkan pada gambar berikut:



Gambar 6.14 Memberi nama CodeWizardProject (*.cwp)

Tekan tombol “Save” untuk menyetujui dan melanjutkan. Selanjutnya akan muncul IDE editor program dari CodeVisionAVR. Anda siap untuk melakukan pembuatan program menggunakan CodeVisionAVR.



Gambar 6.15 Editor Program

2. Pada program-C yang dihasilkan, tambahkan header `delay.h` dan `stdio.h`
3. Kemudian tambahkan definisi untuk mendefinisikan port. Letakkan definisi tersebut di bawah include dari header seperti berikut :

```
#include<mega16.h>
#include<delay.h>
#include<stdio.h>
//tambahkan disini
#define ECHO  PINA.1
#define TRIG  PORTA.0
```

4. Lakukan pendeklarasian variable global, letakkan di atas fungsi `main(void)`

```
// Declare your global variables here
unsignedint count,timeout,US;
unsignedchar xstring[16];
void main(void) { ... }
```

5. Pada bagian `while(1) {...}` tambahkan perintah yang telah diblok dengan warna kuning berikut :

```
while (1)
{
// Place your code here
```

```

TRIG=0; delay_us(10); TRIG=1; delay_us(10); TRIG=0;
timeout=0;
while(!ECHO && timeout<=2000){timeout++;};
count=0;
while(ECHO && count<=8850 {
count++;
};

US = count / 59;
delay_ms(10);
lcd_gotoxy(0,0);
sprintf(xstring, "DATA US: %3d",US);
lcd_puts(xstring);
delay_ms(100);
};

```

6. Untuk lebih jelasnya, berikut adalah kode program secara keseluruhan:

```

#include<mega16.h>
#include<delay.h>
#include<stdio.h>
#define ECHO  PINA.1
#define TRIG  PORTA.0

// Alphanumeric LCD Module functions
#asm
    .equ __lcd_port=0x18 ;PORTB
#endasm
#include<lcd.h>

// Declare your global variables here
unsignedint count,timeout,US;
unsignedchar xstring[16];
void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=Out
// State7=T State6=T State5=T State4=T State3=T State2=T State1=P
State0=1
PORTA=0x03;
DDRA=0x01;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTB=0x00;
DDRB=0x00;

// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In

```

```

// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTD=0x00;
DDRD=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;

```

```

MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// LCD module initialization
lcd_init(16);

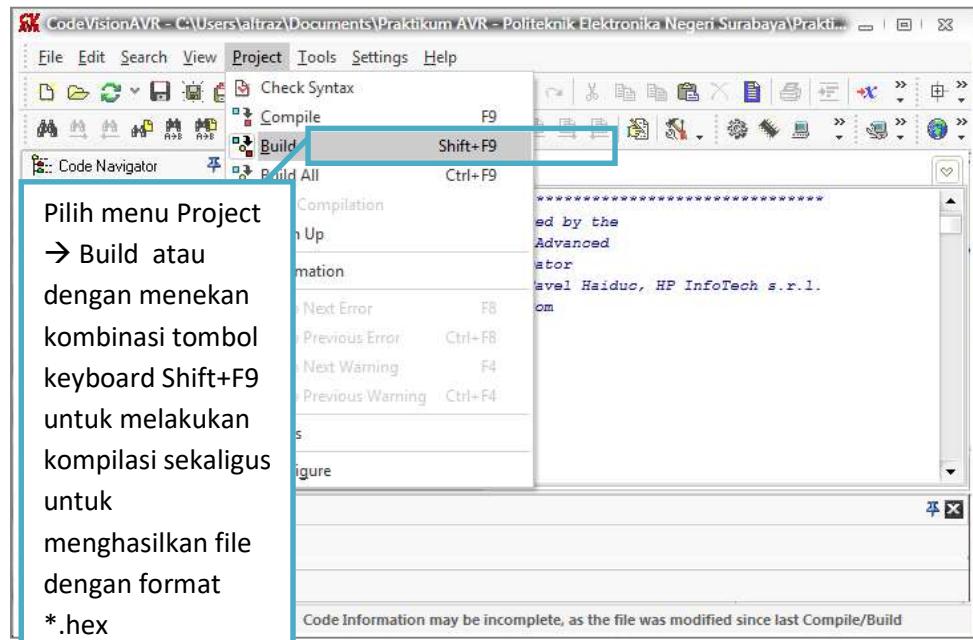
while (1)
{
// Place your code here
    TRIG=0; delay_us(10); TRIG=1; delay_us(10); TRIG=0; //burst
gelombang ultrasonik
    timeout=0;
while(!ECHO && timeout<=2000){timeout++;};
    count=0;
while(ECHO && count<=8850)//count<=8850 --> dibatasi hingga 150cm
    {
        count++;
    };
    US = count / 59; //konversi dari waktu ke jarak cm
    delay_ms(10); //delay untuk memastikan pantulan ultrasonik
telah hilang

    lcd_gotoxy(0,0);
    sprintf(xstring,"DATA US: %3d",US);
    lcd_puts(xstring);
    delay_ms(100);
};

}

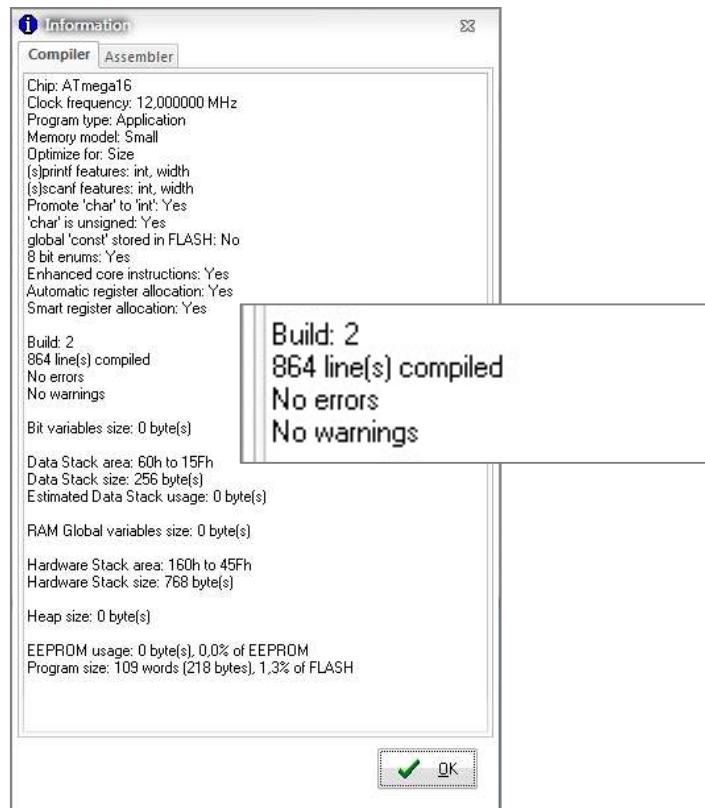
```

7. Setelah pembuatan program selesai, maka proses selanjutnya yaitu lakukan kompilasi terhadap kode program tersebut sekaligus untuk menghasilkan file dengan format hexadecimal (*.hex). Hal ini dapat dilakukan dengan memilih menu Project → Build (Shift+F9) seperti ditunjukkan pada gambar berikut :



Gambar 6.16 Build Source

Apabila kode program sudah benar (tanpa error) dan tidak menyalahi struktur pemrograman bahasa C, maka akan muncul dialog informasi seperti ditunjukkan pada gambar berikut:



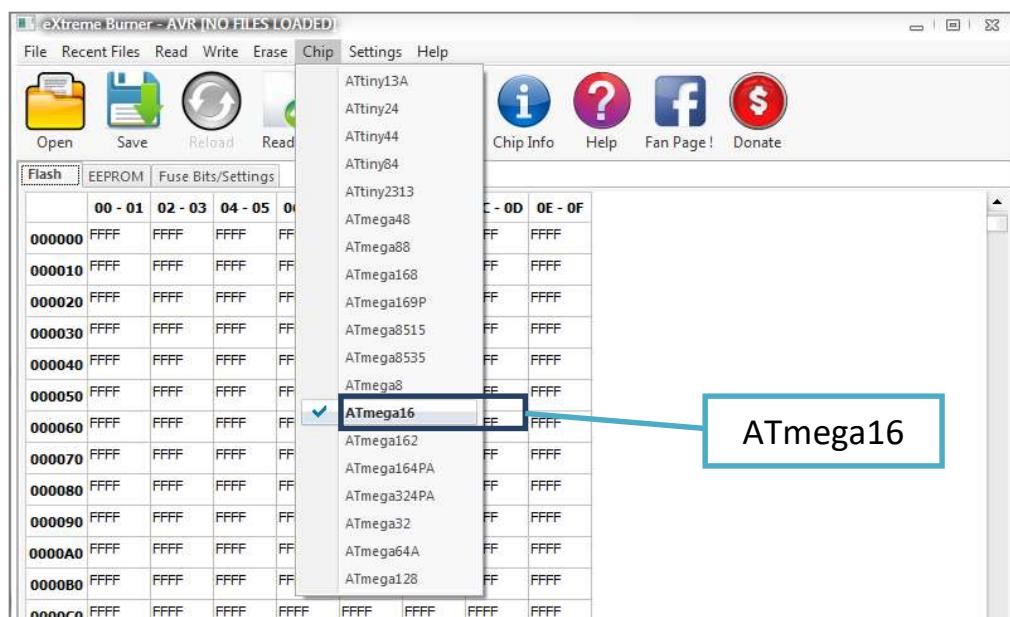
Gambar 6.17 Dialog informasi status kompilasi

8. Setelah diperoleh file dengan format *.hex, maka lakukan download file *.hex ke dalam mikrokontroler. Pertama, jalankan program eXtreme Burner - AVR dengan cara melakukan klik ganda pada shortcut icon eXtreme Burner– AVR pada desktop



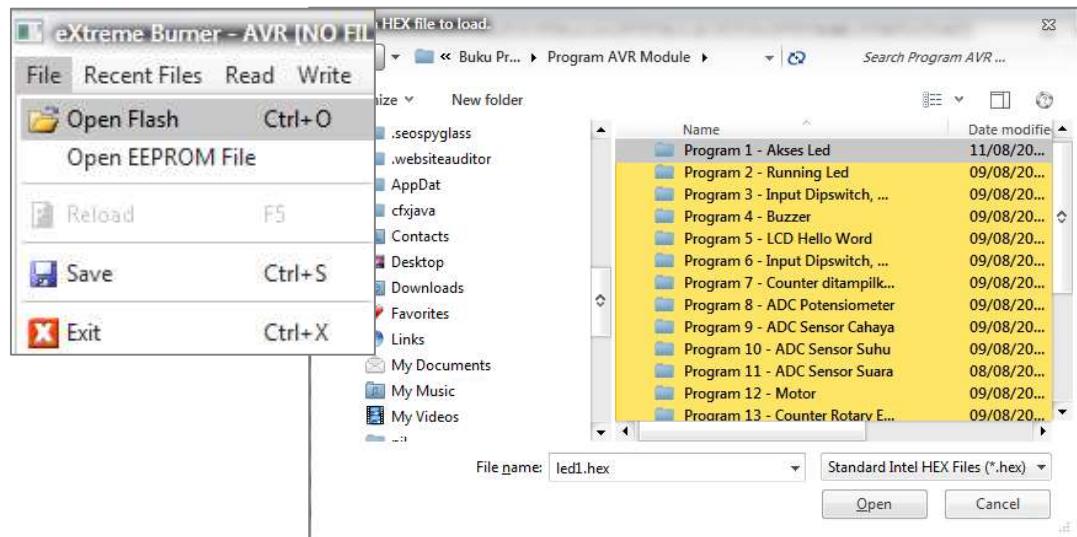
Gambar 6.18 Shortcut eXtreme Burner - AVR

Selanjutnya akan muncul tampilan utama dari program eXtreme Burner – AVR. Pada praktikum ini menggunakan mikrokontroler keluarga AVR dengan tipe IC ATMega16. Oleh karena itu, lakukan pengaturan terhadap tipe IC terlebih dahulu dengan memilih menu Chip → ATmega16 seperti ditunjukkan pada gambar berikut:



Gambar 6.19 Tipe Mikrokontroler

Setelah itu, lakukan open file *.hex dengan memilih menu File → Open Flash (Ctrl+O) seperti ditunjukkan pada gambar berikut:



Gambar 6.20 Open Flash

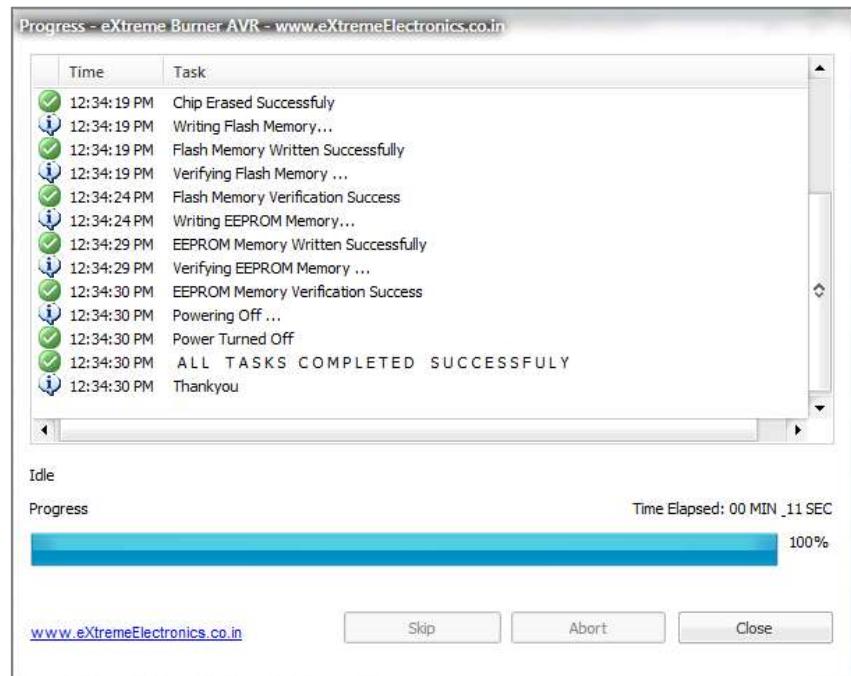
Lakukan pencarian terhadap lokasi file *.hex Anda, untuk selanjutnya tekan tombol “Open” apabila sudah ditemukan file yang dimaksud. Setelah itu akan muncul pesan yang memberitahukan bahwa file *hex berhasil di-load program eXtreme Burner – AVR.



Gambar 6.21 Message



Selanjutnya tekan tombol **Write All** untuk memulai download ke dalam mikrokontroler Anda. Berikut ini merupakan tampilan apabila proses download ke dalam mikrokontroler berhasil dilakukan:



Gambar 6.22 Proses Berhasil

9. Selesai

6.6. Hasil Percobaan



Gambar 6.23 Running Program

6.7. Tugas

Buat program simulai jarak robot terhadap dinding dengan 3 kondisi : dekat (0-3cm), sedang (3cm<D<6cm), jauh(6cm<=D<=8cm)

PERCOBAAN 7

SENSOR JARAK : ULTRASONIK PING PARALAX

PERCOBAAN 7

SENSOR JARAK ULTRASONIK PING PARALAX

7.1. Tujuan

1. Mahasiswa dapat memahami cara kerja sensor jarak ultrasonik ping parallax
2. Mahasiswa dapat mengakses sensor ultrasonic ping parallax pada ATMega16
3. Mahasiswa dapat membuat aplikasi sederhana untuk mengimplementasikan penggunaan sensor ultrasoik pada ATMega16
4. Mahasiswa dapat mengembangkan penggunaan sensor ultrasonic untuk mengetahui jarak suatu benda

7.2. Dasar Teori

Sensor PING merupakan sensor ultrasonik yang dapat mendeteksi jarak obyek dengan cara memancarkan gelombang ultrasonik dengan frekuensi 40 KHz dan kemudian mendeteksi pantulannya. Tampilan sensor jarak PING ditunjukkan pada Gambar berikut:



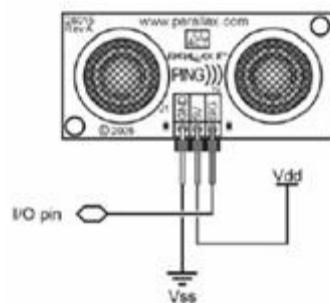
Gambar 7.1 Sensor PING Paralax

Sensor ini dapat mengukur jarak antara 3 cm sampai 300 cm. keluaran dari sensor ini berupa pulsa yang lebarnya merepresentasikan jarak. Lebar pulsanya bervariasi dari 115 uS sampai 18,5 mS. Pada dasanya, Ping))) terdiri dari sebuah chip pembangkit sinyal 40KHz, sebuah speaker ultrasonik dan sebuah mikropon ultrasonik. Speaker ultrasonik mengubah sinyal 40 KHz menjadi suara sementara mikropon ultrasonik berfungsi untuk mendeteksi pantulan suaranya.

Pin signal dapat langsung dihubungkan dengan mikrokontroler tanpa tambahan komponen apapun. Ping hanya akan mengirimkan suara ultrasonik ketika ada pulsa trigger dari mikrokontroler (Pulsa high selama 5uS). Suara ultrasonik dengan

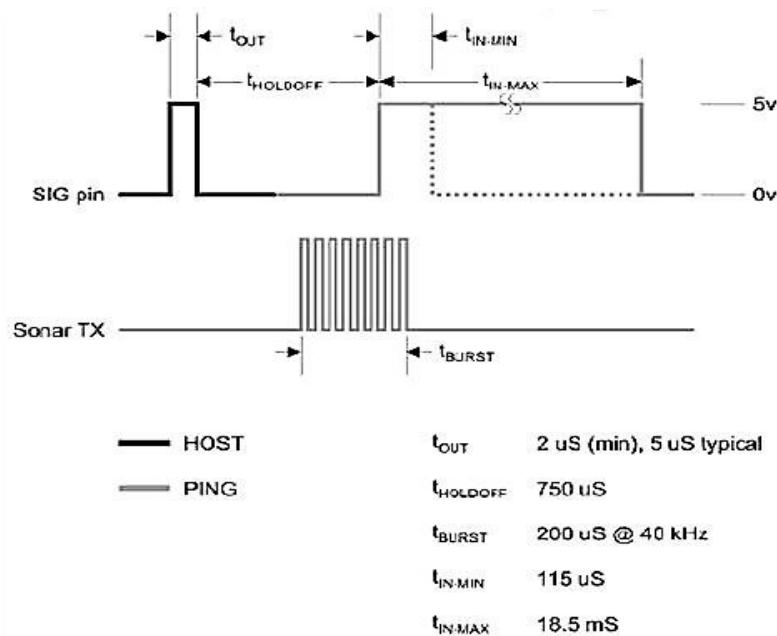
frekuensi sebesar 40KHz akan dipancarkan selama 200uS. Suara ini akan merambat di udara dengan kecepatan 344.424m/detik (atau 1cm setiap 29.034uS), mengenai objek untuk kemudian terpantul kembali ke Ping. Selama menunggu pantulan, Ping akan menghasilkan sebuah pulsa. Pulsa ini akan berhenti (low) ketika suara pantulan terdeteksi oleh Ping. Oleh karena itulah lebar pulsa tersebut dapat merepresentasikan jarak antara Ping dengan objek.

Untuk penjelasan atau prinsip aksesnya sama kok ma srf04, hanya saja untuk sensor PING hanya memakai 3 pin, pin trigger sama echo digunakan dalam 1 pin, sehingga dengan menggunakan sensor PING kita dapat menghemat penggunaan I/O mikrokontroler. Konfigurasi pin sensor PING sebagai berikut:



Gambar 7.2 Konfigurasi Sensor PING Paralax

Timming akses sensor PING



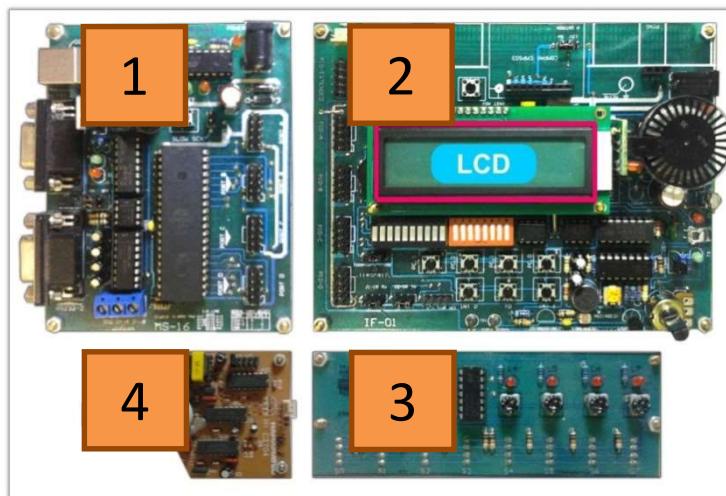
Gambar 7.3 Timming Sensor PING Paralax

7.3. Peralatan

- 1 set PC yang dilengkapi dengan software CodeVision AVR.
- 1 set development board AVR ATmega16
- 1 power-supply +9VDC

7.4. Modul I/O

Berikut ini merupakan development board AVR ATmega16 secara keseluruhan:

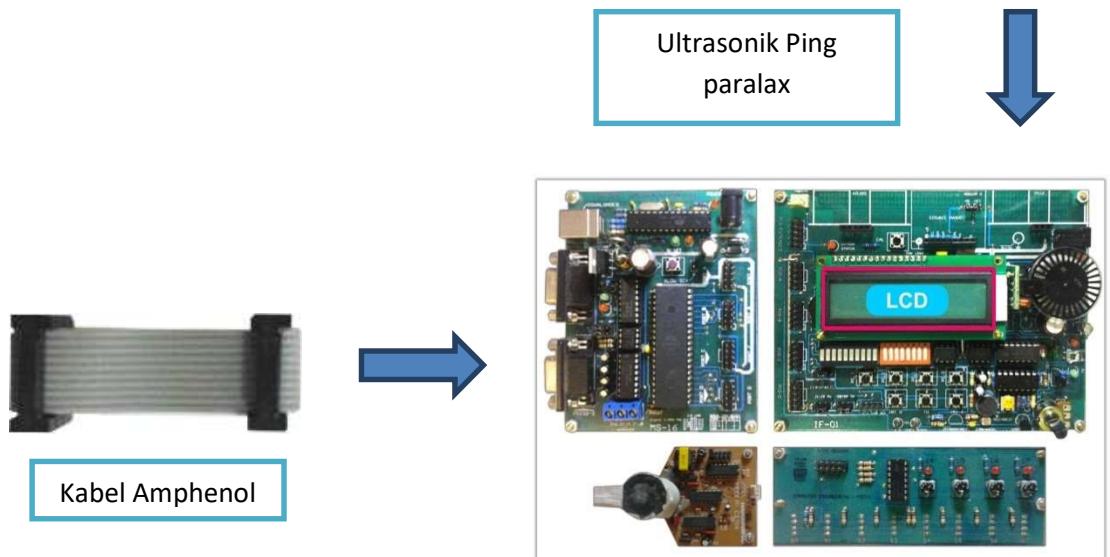


Gambar 7.4 Development board keseluruhan

- **Konfigurasi modul pada percobaan ini :**

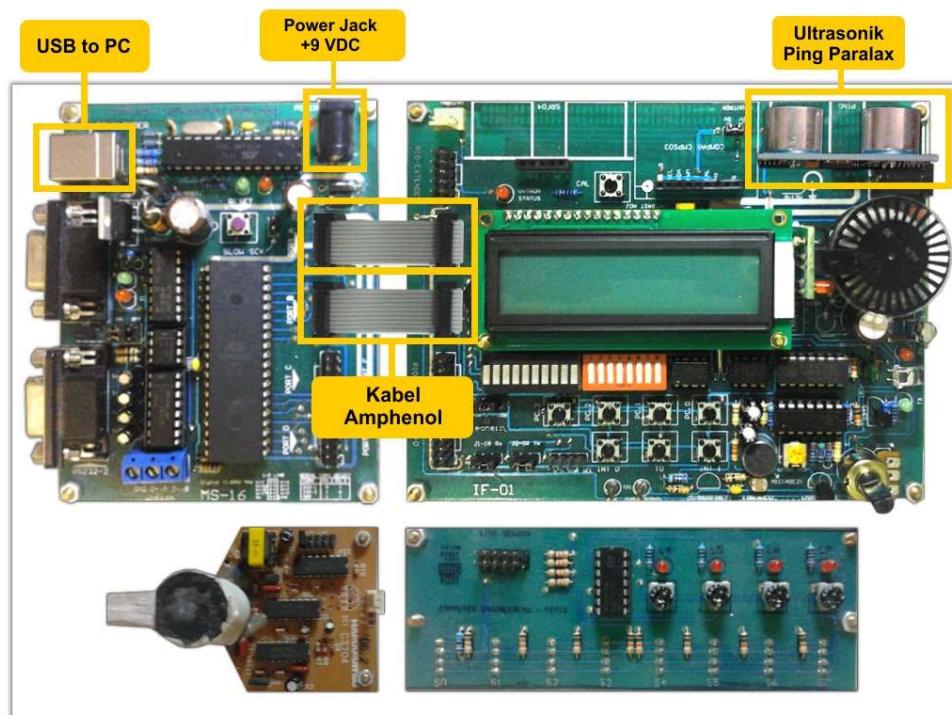
Pasangkan kabel amphenol pada modul development untuk menghubungkan system minimum ATMega16 dengan Training Board. Pasangkan sensor ultrasonic ping parallax pada training board.





Gambar 7.5 Komponen Training board

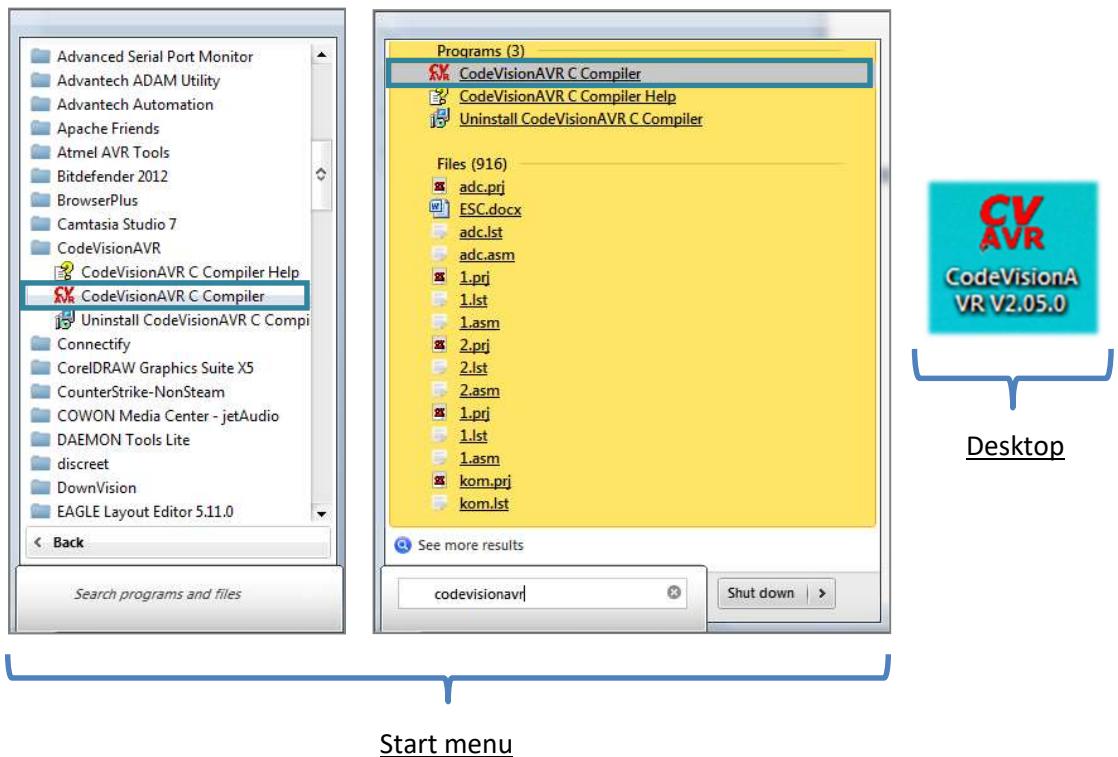
Sehingga menjadi seperti ditunjukkan pada gambar berikut:



Gambar 7.6 Komponen Training board

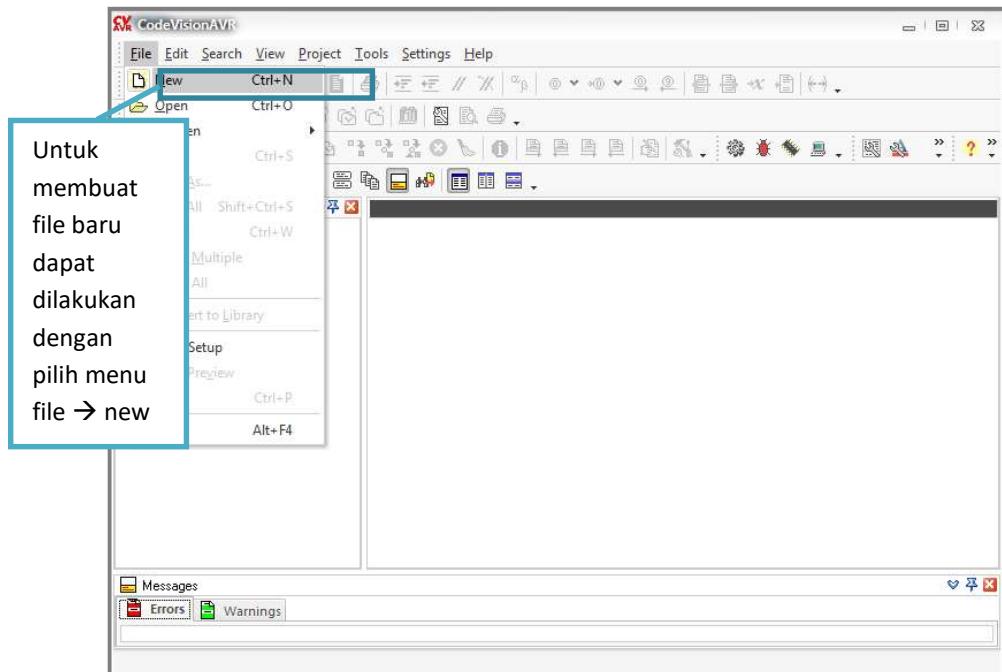
7.5. Prosedur Praktikum

1. Jalankan aplikasi CodeVisionAVR dengan cara melakukan klik ganda pada shortcut icon CodeVisionAVR pada desktop atau dengan cara mencari shortcut program pada start menu system operasi windows.



Gambar 7.7 Shortcut CodeVisionAVR

Untuk memulai membuat project baru, pada menubar, pilih File → New, seperti yang ditunjukkan oleh gambar berikut:



Gambar 7.8 Membuat File baru

Setelah itu akan muncul suatu dialog yang mengharuskan *user* untuk memilih antara pembuatan source file atau project. Dalam setiap percobaan pada praktikum ini, dibuat project baru untuk setiap percobaan, oleh karena itu *user* sebaiknya melakukan *check* pada checkbox project dan dilanjutkan dengan menekan tombol “OK”



Gambar 7.9 Membuat project baru

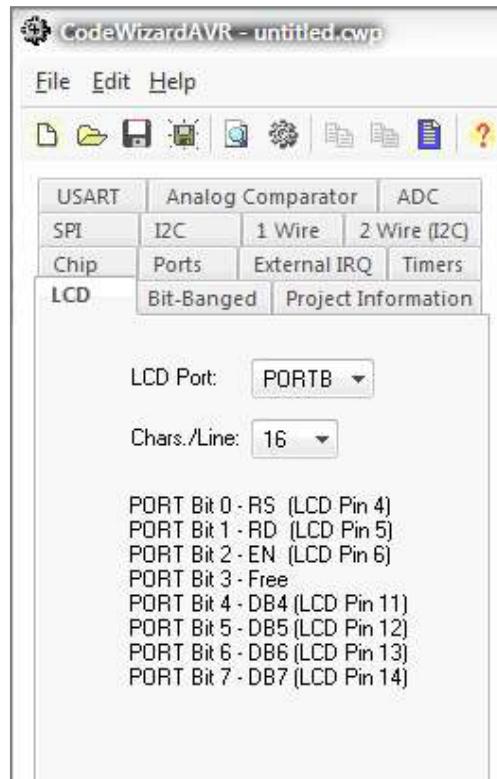
Berikutnya akan keluar dialog yang menanyakan Anda apakah akan membuat project baru. Klik tombol “Yes” untuk menyetujui.



Gambar 7.10 Memilih untuk membuat project baru pada CodeVisionAVR

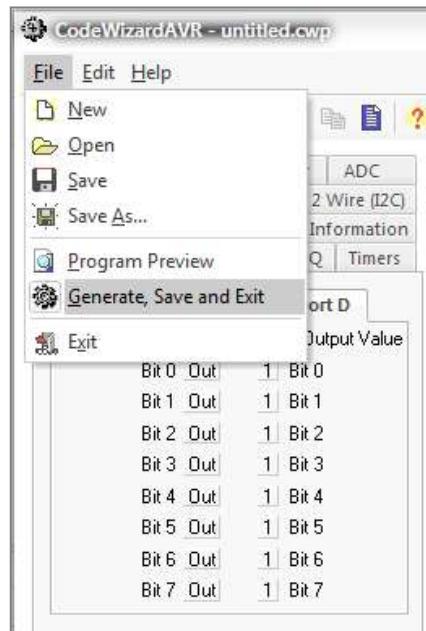
Setelah itu lakukan set pada CodeWizardAVR. Untuk chip diset dengan nama chip ATmega16 dengan clock 12 MHz.

Berikutnya Anda akan menginisialisasi LCD pada PortB, yaitu dengan cara masuk pada tab LCD dan pilih item **PORTE** pada selectbox "LCD Port". Set banyaknya karakter dari LCD yang diinginkan. Dalam hal ini menggunakan 16 karakter, oleh karena itu pilih item **16** pada selectbox "Chars/Line"



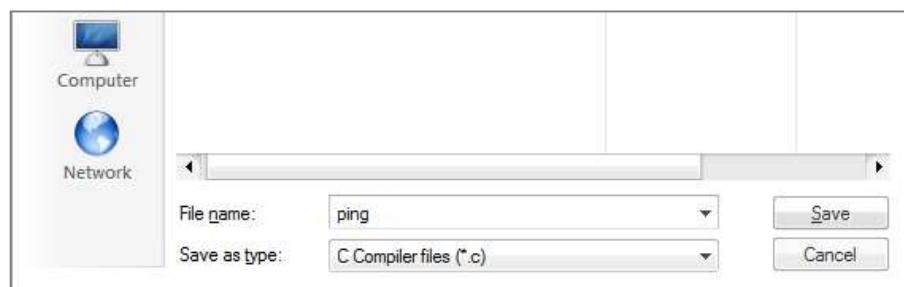
Gambar 7.11 Setting LCD pada PortB

Karena pada contoh ini tidak digunakan fasilitas lain maka setting CodeWizardAVR siap disimpan dalam file. Pada menu CodeWizardAVR, pilih File → Generate, Save and Exit, seperti yang ditunjukkan pada gambar berikut:



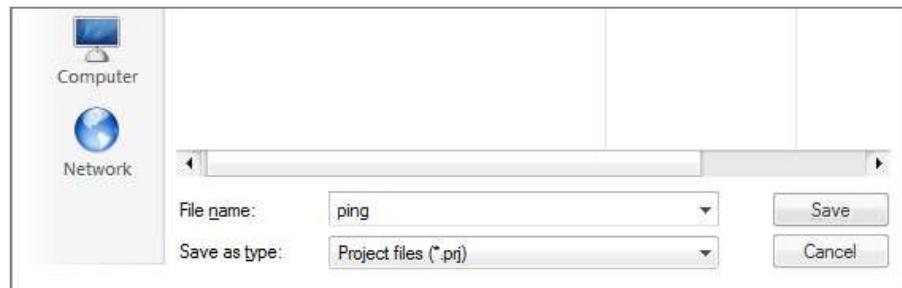
Gambar 7.12 Menyimpan setting

Setelah penekanan menu “Generate, Save and Exit”, akan muncul suatu dialog untuk melakukan set terhadap nama dan lokasi penyimpanan dari source file (*.c) seperti ditunjukkan pada gambar berikut:



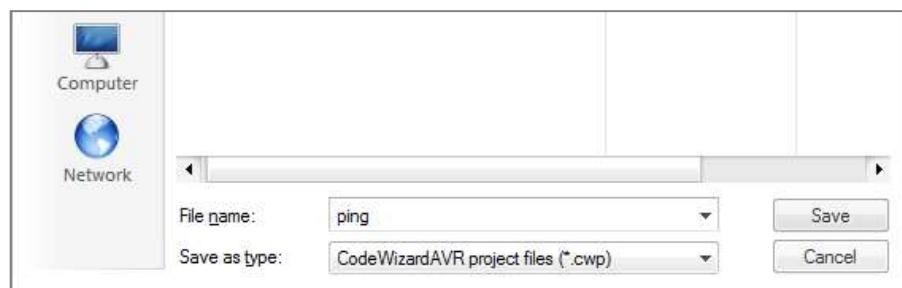
Gambar 7.13 Memberi nama pada file C (*.c)

Tekan tombol “Save” untuk menyetujui dan melanjutkan pada proses berikutnya. Setelah penekanan tombol “Save” akan muncul kembali dialog yang kedua, yaitu untuk melakukan set terhadap nama dan lokasi penyimpanan project file (*.prj) seperti ditunjukkan pada gambar berikut:



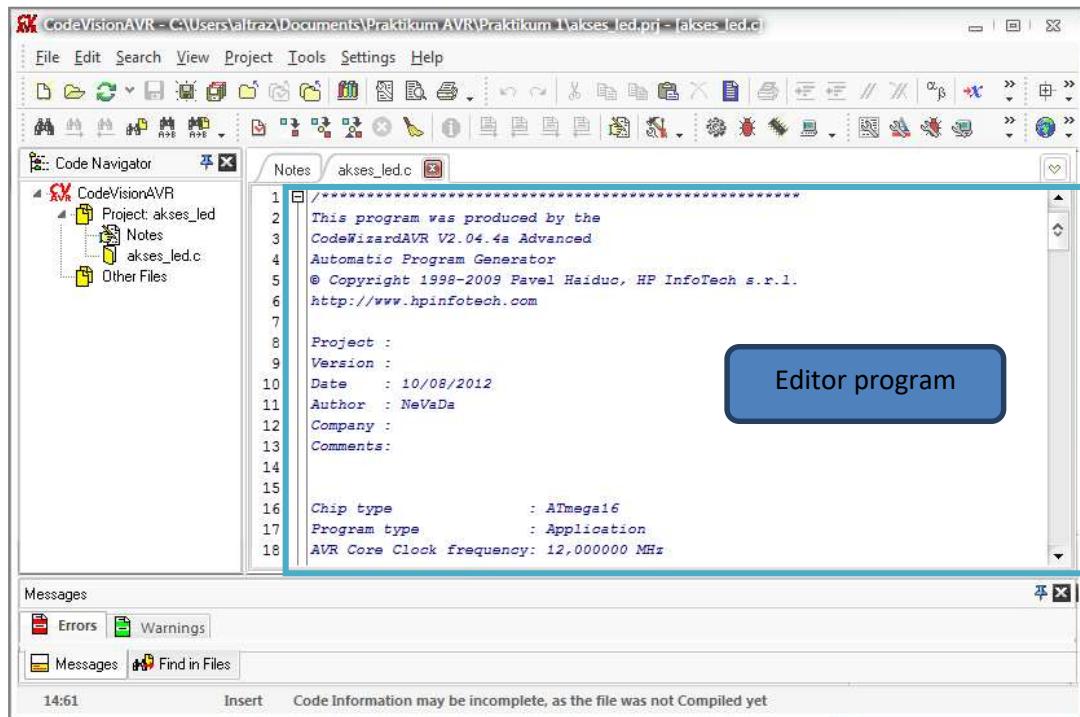
Gambar 7.14 Memberi nama project (*.prj)

Tekan tombol “Save” untuk menyetujui dan melanjutkan pada proses berikutnya. Setelah penekanan tombol “Save” akan muncul kembali dialog yang ketiga, yaitu untuk melakukan set terhadap nama dan lokasi penyimpanan CodeWizardProject file (*.cwp) seperti ditunjukkan pada gambar berikut:



Gambar 7.15 Memberi nama CodeWizardProject (*.cwp)

Tekan tombol “Save” untuk menyetujui dan melanjutkan. Selanjutnya akan muncul IDE editor program dari CodeVisionAVR. Anda siap untuk melakukan pembuatan program menggunakan CodeVisionAVR.



Gambar 7.16 Editor Program

2. Pada program-C yang dihasilkan, tambahkan header `delay.h` dan `stdio.h`
3. Kemudian tambahkan definisi untuk mendefinisikan port. Letakkan definisi tersebut di bawah include dari header seperti berikut :

```
#include<mega16.h>
#include<delay.h>
#include<stdio.h>
//tambahkan disini
#define PULSE PORTA.5 //kanan
#define ECHO PINA.5
#define ARAH DDRA.5
#define OUT 1
#define INP 0
```

4. Lakukan pendeklarasian variable global, letakkan di atas fungsi main(void)

```
// Declare your global variables here
unsignedint count,timeout,US;
unsignedchar xstring[16];
void main(void) { ... }
```

5. Pada bagian `while(1) {...}` tambahkan perintah yang telah diblok dengan warna kuning berikut :

```

while (1)
{
// Place your code here
ARAH=OUT;
    PULSE=1; delay_us(5); PULSE=0;
    ARAH=INP;
    PULSE=1;
    timeout=0;
while (!ECHO && timeout<=2000) {timeout++;};
    count=0;
while (ECHO && count<=8850)
{
    count++;
}
US = count / 59;
delay_ms(10);
lcd_gotoxy(0,0);
sprintf(xstring,"DATA US: %3d",US);
lcd_puts(xstring);
delay_ms(100);

};

```

6. Untuk lebih jelasnya, berikut adalah kode program secara keseluruhan:

```

#include<mega16.h>
#include<delay.h>
#include<stdio.h>

#define PULSE PORTA.5 //kanan
#define ECHO PINA.5
#define ARAH DDRA.5
#define OUT 1
#define INP 0

// Alphanumeric LCD Module functions
#asm
    .equ __lcd_port=0x18 ;PORTB
#endasm
#include<lcd.h>

// Declare your global variables here
unsignedint count,timeout,US;
unsignedchar xstring[16];
void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTA=0x00;
DDRA=0x00;

```

```

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTB=0x00;
DDRB=0x00;

// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTD=0x00;
DDRD=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped

```

```

// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// LCD module initialization
lcd_init(16);

while (1)
{
// Place your code here
    ARAH=OUT;
    PULSE=1; delay_us(5); PULSE=0; //burst gelombang ultrasonik
    ARAH=INP;
    PULSE=1;
    timeout=0;
    while (!ECHO && timeout<=2000) {timeout++;};
    count=0;
    while (ECHO && count<=8850) //count<=8850 --> dibatasi hingga 150cm
    {
        count++;
    }
    US = count / 59; //konversi dari waktu ke jarak
    delay_ms(10); //delay untuk memastikan pantulan ultrasonik
    telah hilang

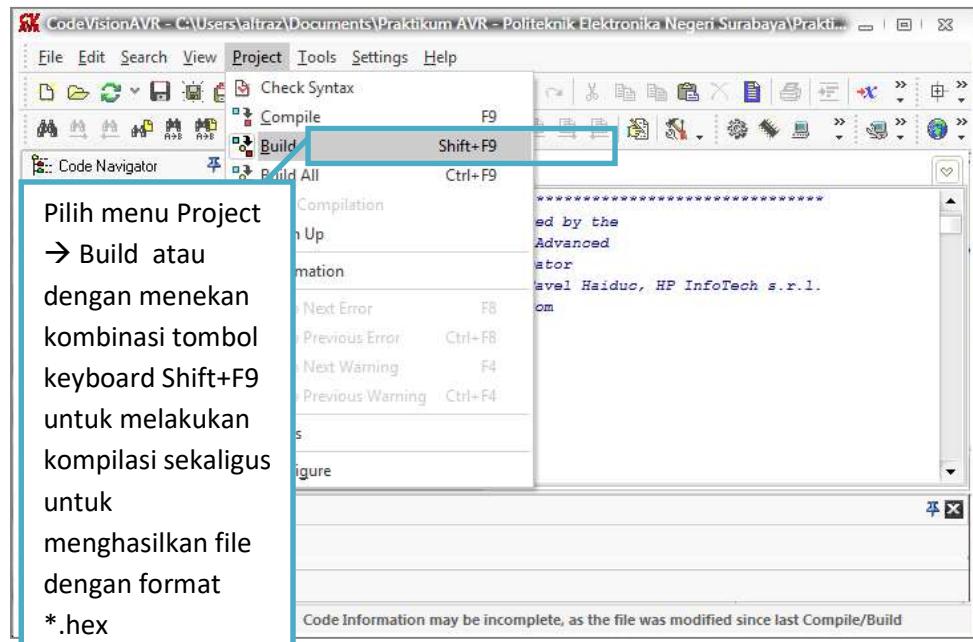
    lcd_gotoxy(0,0);
    sprintf(xstring, "DATA US: %3d",US);
    lcd_puts(xstring);
    delay_ms(100);

}
}

```

7. Setelah pembuatan program selesai, maka proses selanjutnya yaitu lakukan kompilasi terhadap kode program tersebut sekaligus untuk menghasilkan file

dengan format hexadecimal (*.hex). Hal ini dapat dilakukan dengan memilih menu Project → Build (Shift+F9) seperti ditunjukkan pada gambar berikut :



Gambar 7.17 Build Source

Apabila kode program sudah benar (tanpa error) dan tidak menyalahi struktur pemrograman bahasa C, maka akan muncul dialog informasi seperti ditunjukkan pada gambar berikut:



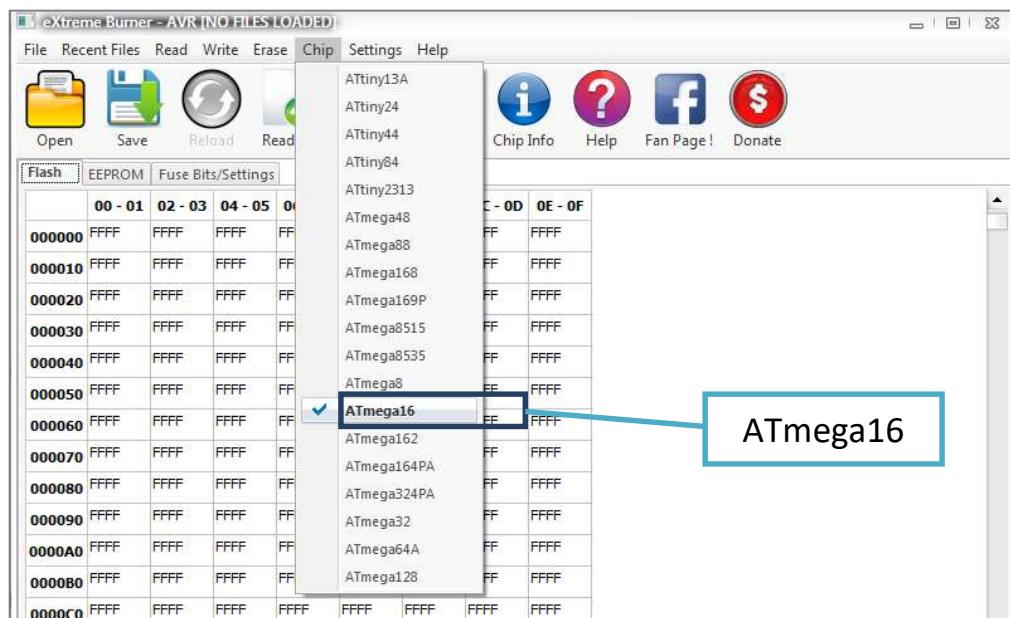
Gambar 7.18 Dialog informasi status kompilasi

8. Setelah diperoleh file dengan format *.hex, maka lakukan download file *.hex ke dalam mikrokontroler. Pertama, jalankan program eXtreme Burner - AVR dengan cara melakukan klik ganda pada shortcut icon eXtreme Burner– AVR pada desktop



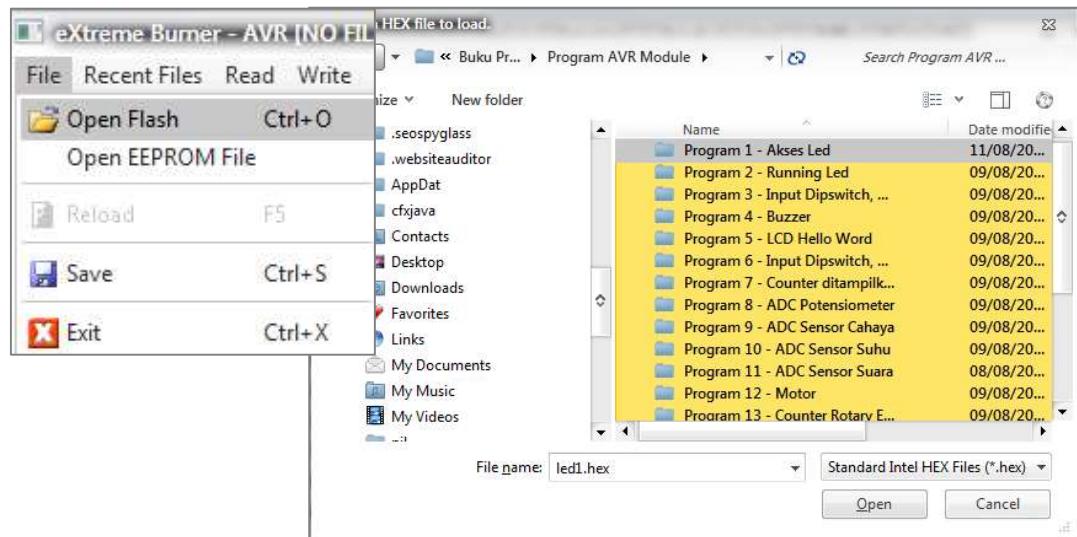
Gambar 7.19 Shortcut eXtreme Burner - AVR

Selanjutnya akan muncul tampilan utama dari program eXtreme Burner – AVR. Pada praktikum ini menggunakan mikrokontroler keluarga AVR dengan tipe IC ATMega16. Oleh karena itu, lakukan pengaturan terhadap tipe IC terlebih dahulu dengan memilih menu Chip → ATmega16 seperti ditunjukkan pada gambar berikut:



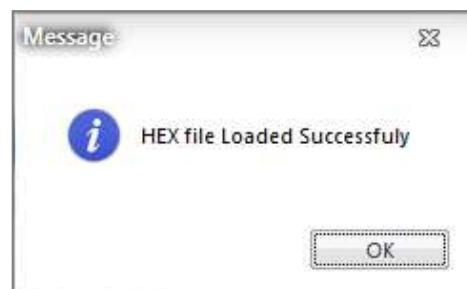
Gambar 7.20 Tipe Mikrokontroler

Setelah itu, lakukan open file *.hex dengan memilih menu File → Open Flash (Ctrl+O) seperti ditunjukkan pada gambar berikut:



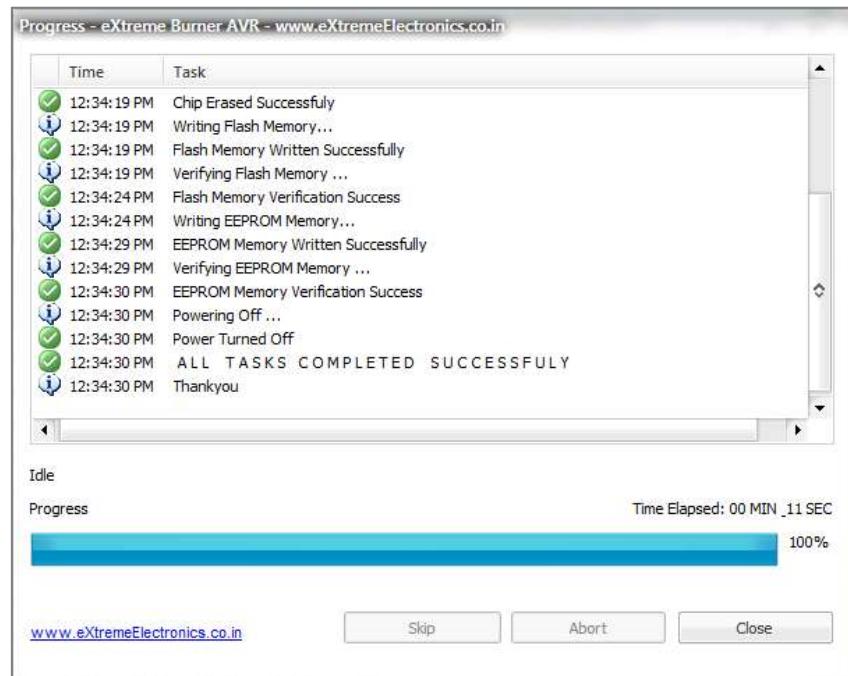
Gambar 7.21 Open Flash

Lakukan pencarian terhadap lokasi file *.hex Anda, untuk selanjutnya tekan tombol “Open” apabila sudah ditemukan file yang dimaksud. Setelah itu akan muncul pesan yang memberitahukan bahwa file *hex berhasil di-load program eXtreme Burner – AVR.



Gambar 7.22 Message

Selanjutnya tekan tombol  untuk memulai download ke dalam mikrokontroler Anda. Berikut ini merupakan tampilan apabila proses download ke dalam mikrokontroler berhasil dilakukan:



Gambar 7.23 Proses Berhasil

9. Selesai

7.6. Hasil Percobaan

Amati hasil praktikum Anda

7.7. Tugas

Buat program simulai jarak robot terhadap dinding dengan 3 kondisi : dekat (0-3cm), sedang (3cm<D<6cm), jauh(6cm<=D<=8cm)

PERCOBAAN 8

SENSOR POSISI KOMPAS CMPS03

PERCOBAAN 8

SENSOR POSISI KOMPAS CMPS03

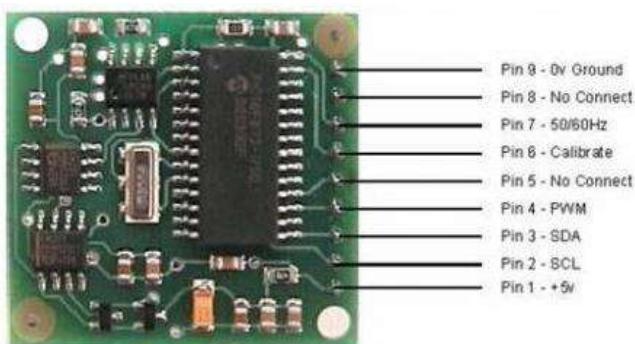
8.1. Tujuan

1. Mahasiswa dapat mengetahui cara kerja sensor posisi kompas ditigal CMPS03
2. Mahasiswa dapat mengakses CMPS03 pada ATMega16
3. Mahasiswa dapat membuat aplikasi sederhana untuk mendapat data magnitude dari CMPS03

8.2. Dasar Teori

Kompas Elektronik CMPS-03 buatan Devantech Ltd ini menggunakan sensor medan magnet Philips KMZ51 yang cukup sensitif untuk mendeteksi medan magnet bumi. Modul ini bekerja dengan mendeteksi magnetik bumi. Data yang dihasilkan dari kompas elektronik ini berupa data biner. Sebagai contoh jika modul menghadap utara maka data yang dihasilkan adalah data 00H, dan arah selatan data keluarannya adalah 7FH.

Koneksi dari modul ke mikrokontroller dapat dilakukan dengan 2 cara yaitu dengan menggunakan data PWM (*Pulse Width Modulation*), atau dengan I2C (*Inter Intergrated Circuit*). Jika menggunakan *interface* PWM, pulsa keluaran memiliki rentang 1mS untuk 0° atau arah utara sampai dengan 36.99 mS untuk 359.90°. Cara yang kedua menggunakan I2C, metode ini dapat digunakan langsung sehingga data yang dibaca tepat 0° – 360° sama dengan 0 – 255.



Gambar 8.1 bentuk fisik dan koneksi pin modul kompas CMPS03

Modul kompas CMPS03 membutuhkan kalibrasi untuk menentukan nilai data keluaran dari tiap-tiap arah kompas. Metode kalibrasi dapat dilakukan secara manual, adapun cara kalibrasi sebagai berikut:

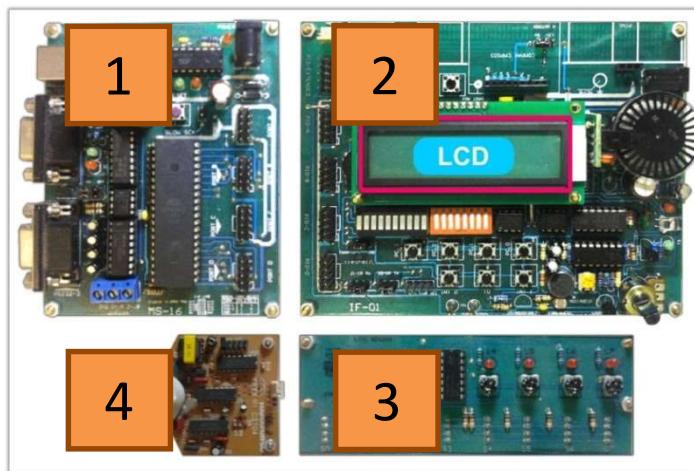
- a. Modul kompas dihadapkan ke utara, kemudian memberikan pulsa rendah pada pin kalibrasi
- b. Kompas diputar perlahan ke arah timur secara perlahan, kemudian pin kalibrasi diberi pulsa rendah
- c. Langkah berikutnya memutar modul kompas ke arah selatan secara perlahan, kemudian memberikan pulsa rendah ke pin kalibrasi.
- d. Langkah terakhir adalah memutar kompas ke arah barat dan kemudian memberikan pulsa rendah ke pin kalibrasi.

8.3. Peralatan

- 1 set PC yang dilengkapi dengan software CodeVision AVR.
- 1 set development board AVR ATmega16
- 1 power-supply +9VDC

8.4. Modul I/O

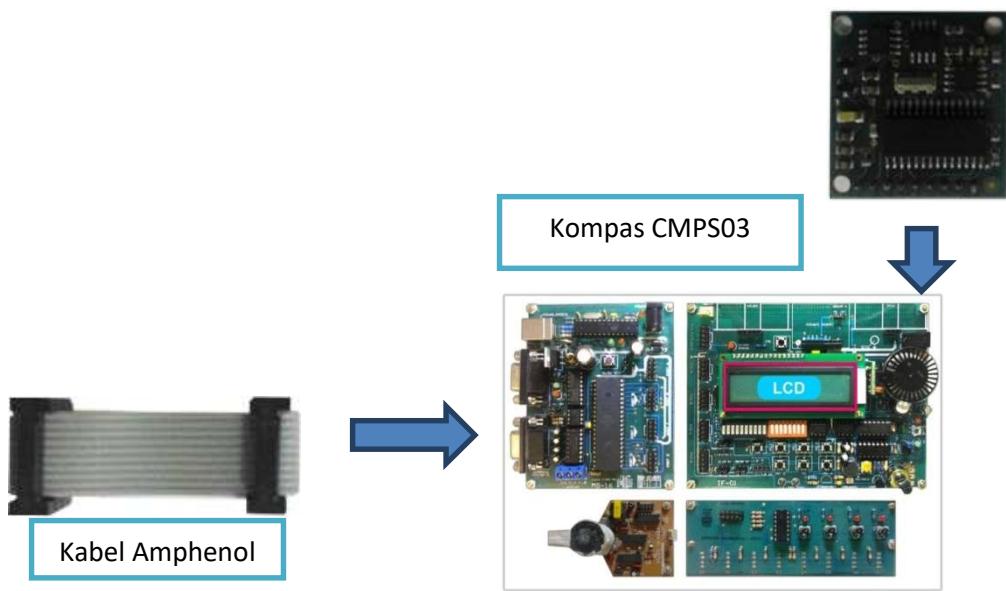
Berikut ini merupakan development board AVR ATmega16 secara keseluruhan:



Gambar 8.2 Development board keseluruhan

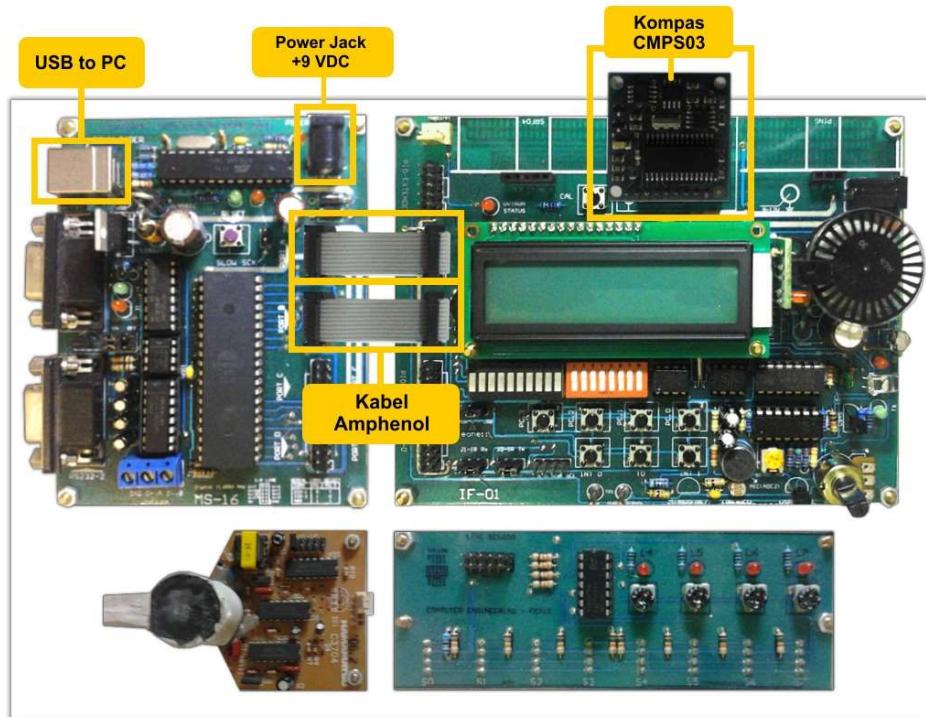
- **Konfigurasi modul pada percobaan ini :**

Pasangkan CMPS03 dan kabel amphenol pada modul development untuk menghubungkan sistem minimum ATmega16 dengan Training Board.



Gambar 8.3 Komponen Training board

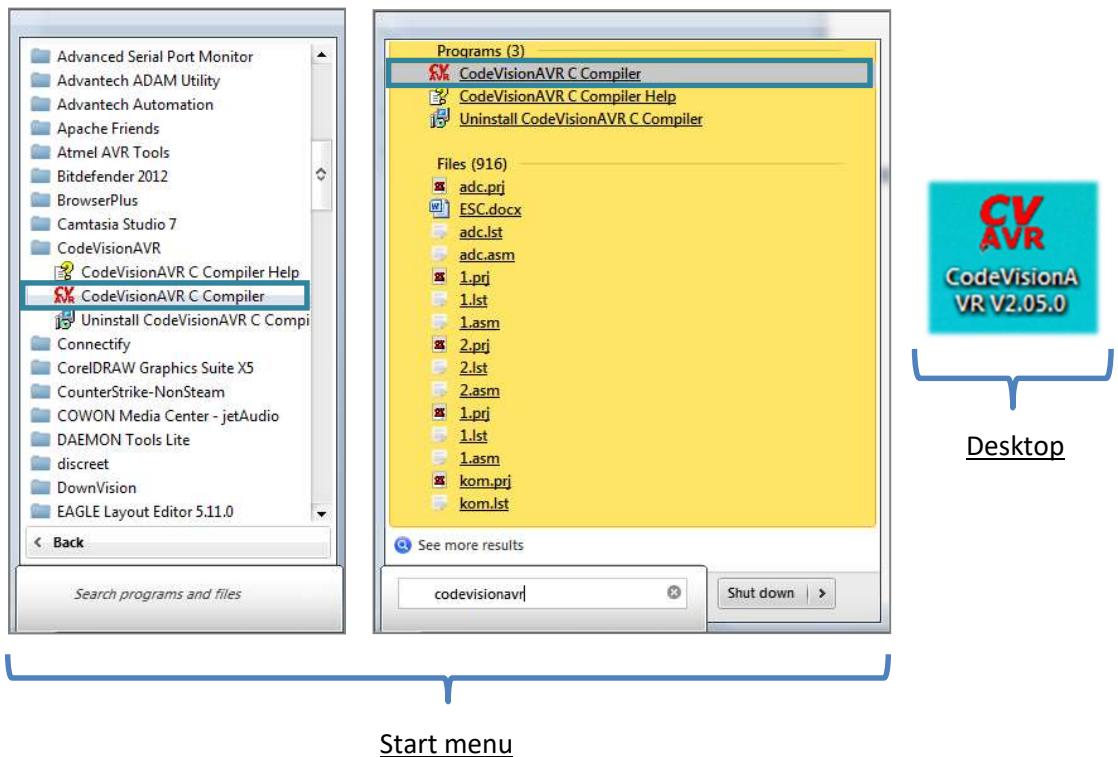
Sehingga menjadi seperti ditunjukkan pada gambar berikut:



Gambar 8.4 Komponen Training board

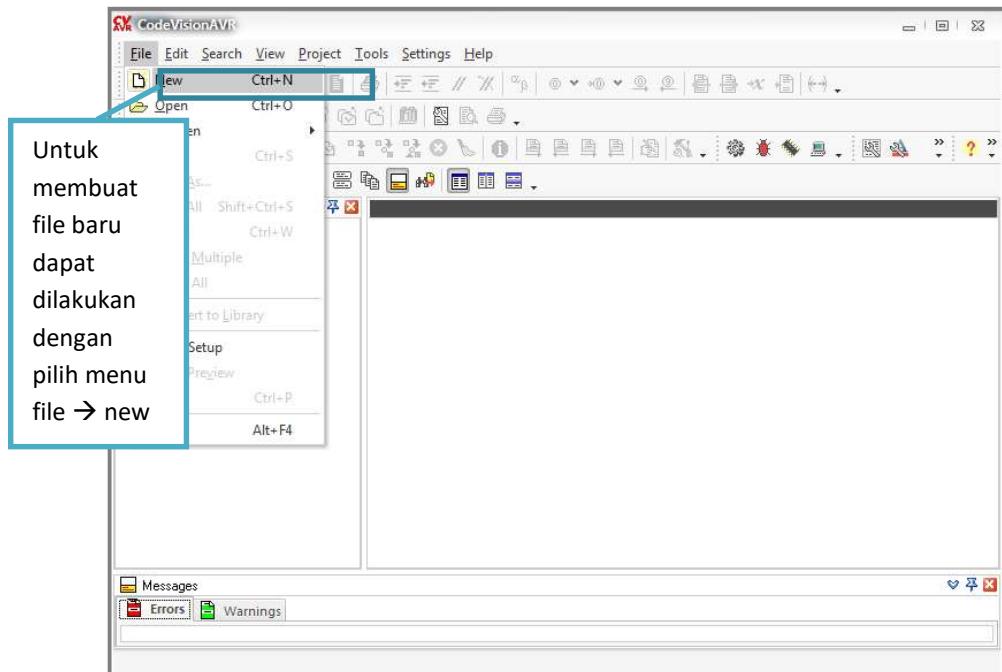
8.5. Prosedur Praktikum

1. Jalankan aplikasi CodeVisionAVR dengan cara melakukan klik ganda pada shortcut icon CodeVisionAVR pada desktop atau dengan cara mencari shortcut program pada start menu system operasi windows.



Gambar 8.5 Shortcut CodeVisionAVR

Untuk memulai membuat project baru, pada menubar, pilih File → New, seperti yang ditunjukkan oleh gambar berikut:



Gambar 8.6 Membuat File baru

Setelah itu akan muncul suatu dialog yang mengharuskan *user* untuk memilih antara pembuatan source file atau project. Dalam setiap percobaan pada praktikum ini, dibuat project baru untuk setiap percobaan, oleh karena itu *user* sebaiknya melakukan *check* pada checkbox project dan dilanjutkan dengan menekan tombol “OK”



Gambar 8.7 Membuat project baru

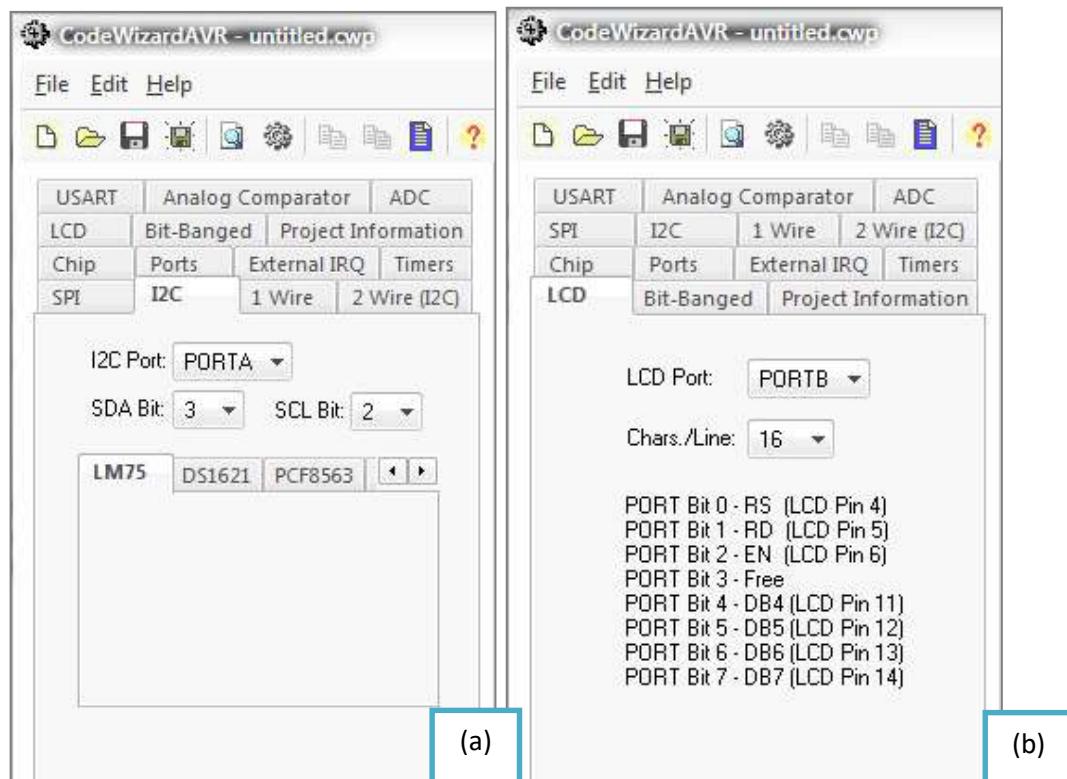
Berikutnya akan keluar dialog yang menanyakan Anda apakah akan membuat project baru. Klik tombol “Yes” untuk menyetujui.



Gambar 8.8 Memilih untuk membuat project baru pada CodeVisionAVR

Setelah itu lakukan set pada CodeWizardAVR. Untuk chip diset dengan nama chip ATmega16 dengan clock 12 MHz. Selanjtnya set I2C port pada Port A, dengan SDA Bit = 3, dan SCL Bit = 2.

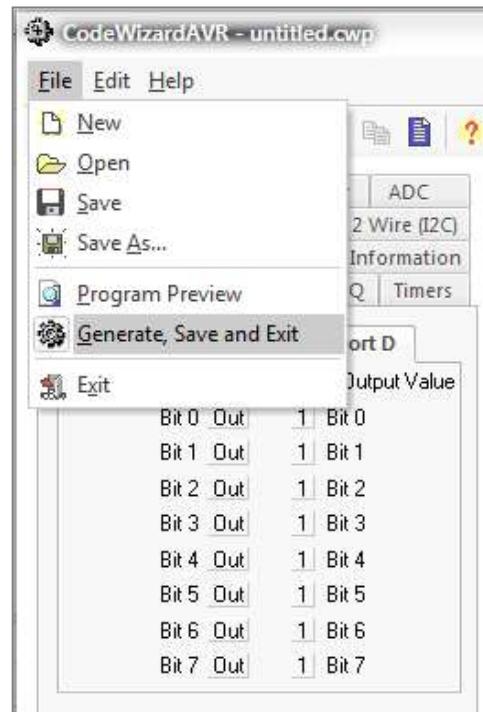
Berikutnya Anda akan menginisialisasi LCD pada PortB, yaitu dengan cara masuk pada tab LCD dan pilih item **PORTE** pada selectbox “LCD Port”. Set banyaknya karakter dari LCD yang diinginkan. Dalam hal ini menggunakan 16 karakter, oleh karena itu pilih item **16** pada selectbox “Chars/Line”



Gambar 8.9 (a) Setting I2C Port pada Port A

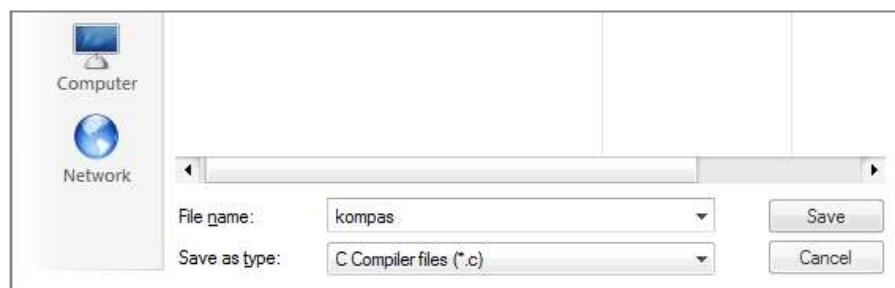
(b) Setting LCD pada PortB

Karena pada contoh ini tidak digunakan fasilitas lain maka setting CodeWizardAVR siap disimpan dalam file. Pada menu CodeWizardAVR, pilih File → Generate, Save and Exit, seperti yang ditunjukkan pada gambar berikut:



Gambar 8.10 Menyimpan setting

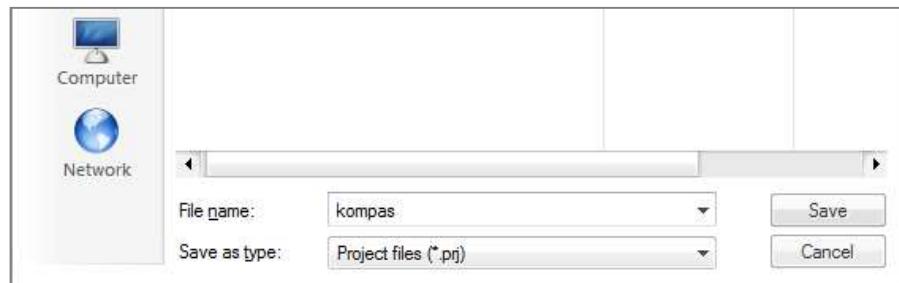
Setelah penekanan menu “Generate, Save and Exit”, akan muncul suatu dialog untuk melakukan set terhadap nama dan lokasi penyimpanan dari source file (*.c) seperti ditunjukkan pada gambar berikut:



Gambar 8.11 Memberi nama pada file C (*.c)

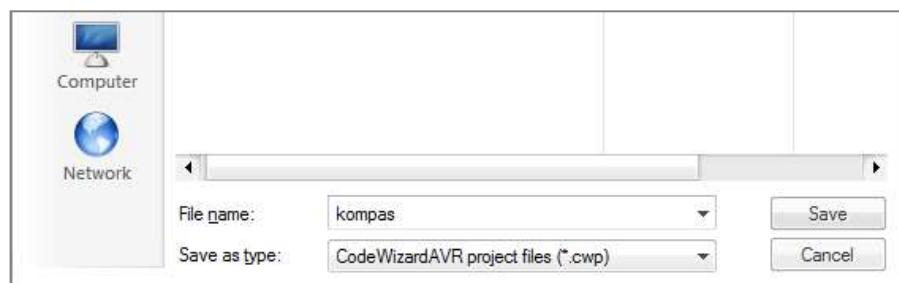
Tekan tombol “Save” untuk menyetujui dan melanjutkan pada proses berikutnya. Setelah penekanan tombol “Save” akan muncul kembali dialog yang kedua, yaitu

untuk melakukan set terhadap nama dan lokasi penyimpanan project file (*.prj) seperti ditunjukkan pada gambar berikut:



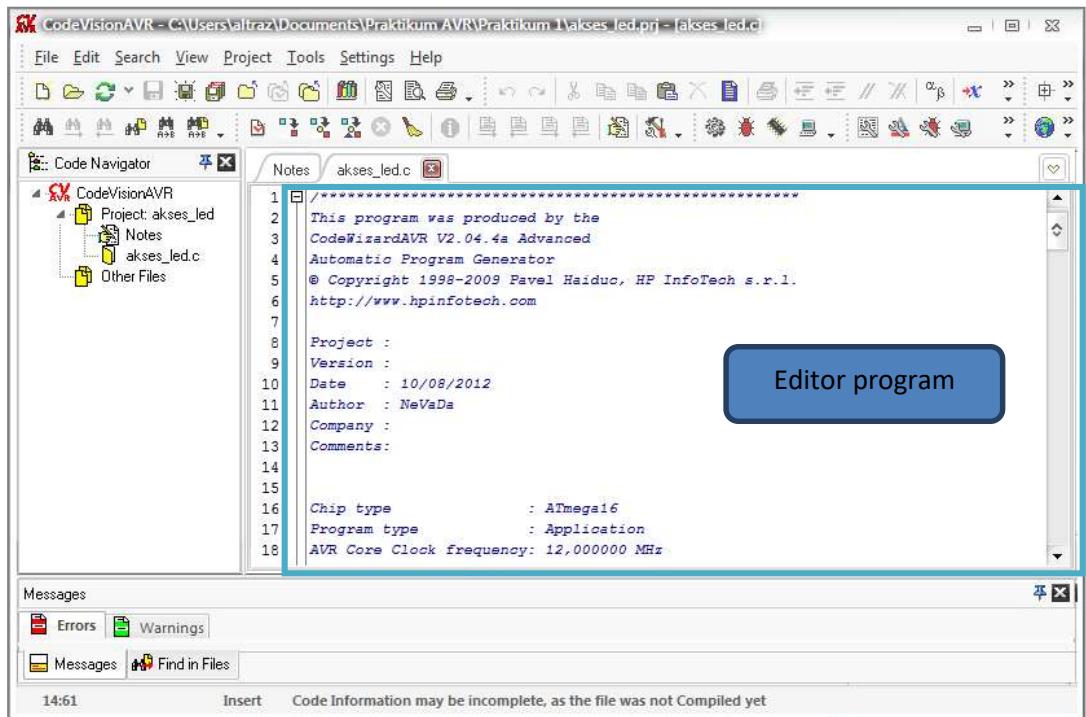
Gambar 8.12 Memberi nama project (*.prj)

Tekan tombol “Save” untuk menyetujui dan melanjutkan pada proses berikutnya. Setelah penekanan tombol “Save” akan muncul kembali dialog yang ketiga, yaitu untuk melakukan set terhadap nama dan lokasi penyimpanan CodeWizardProject file (*.cwp) seperti ditunjukkan pada gambar berikut:



Gambar 8.13 Memberi nama CodeWizardProject (*.cwp)

Tekan tombol “Save” untuk menyetujui dan melanjutkan. Selanjutnya akan muncul IDE editor program dari CodeVisionAVR. Anda siap untuk melakukan pembuatan program menggunakan CodeVisionAVR.



Gambar 8.14 Editor Program

2. Pada program-C yang dihasilkan, tambahkan header `delay.h` dan `stdio.h`
3. Lakukan pendeklarasian variable global, letakkan di atas fungsi main(void)

```
// Declare your global variables here
unsignedchar data,xstring[16];
void main(void) {...
```

4. Pada bagian `while(1) {...}` tambahkan perintah yang telah diblok dengan warna kuning berikut :

```
while (1)
{
// Place your code here
i2c_start();
    i2c_write(0xC0);
    i2c_write(0x01);
    i2c_start();
    i2c_write(0xC1);
    data=i2c_read(0);
    i2c_stop();
    lcd_gotoxy(0,0);
    sprintf(xstring,"Data = %3d",data);
    lcd_puts(xstring);
    delay_ms(150);
};
```

5. Untuk lebih jelasnya, berikut adalah kode program secara keseluruhan:

```
#include<mega16.h>
#include<delay.h>
#include<stdio.h>

// I2C Bus functions
#asm
    .equ __i2c_port=0x1B ;PORTA
    .equ __sda_bit=3
    .equ __scl_bit=2
#endasm
#include<i2c.h>

// Alphanumeric LCD Module functions
#asm
    .equ __lcd_port=0x18 ;PORTB
#endasm
#include<lcd.h>

// Declare your global variables here
unsignedchar data,xstring[16];
void main(void)
{
    // Declare your local variables here

    // Input/Output Ports initialization
    // Port A initialization
    // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
    Func0=In
    // State7=T State6=T State5=T State4=T State3=T State2=T State1=T
    State0=T
    PORTA=0x00;
    DDRA=0x00;

    // Port B initialization
    // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
    Func0=In
    // State7=T State6=T State5=T State4=T State3=T State2=T State1=T
    State0=T
    PORTB=0x00;
    DDRB=0x00;

    // Port C initialization
    // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
    Func0=In
    // State7=T State6=T State5=T State4=T State3=T State2=T State1=T
    State0=T
    PORTC=0x00;
    DDRC=0x00;

    // Port D initialization
    // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
    Func0=In
    // State7=T State6=T State5=T State4=T State3=T State2=T State1=T
    State0=T
    PORTD=0x00;
```

```

DDRD=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

```

```

// I2C Bus initialization
i2c_init();

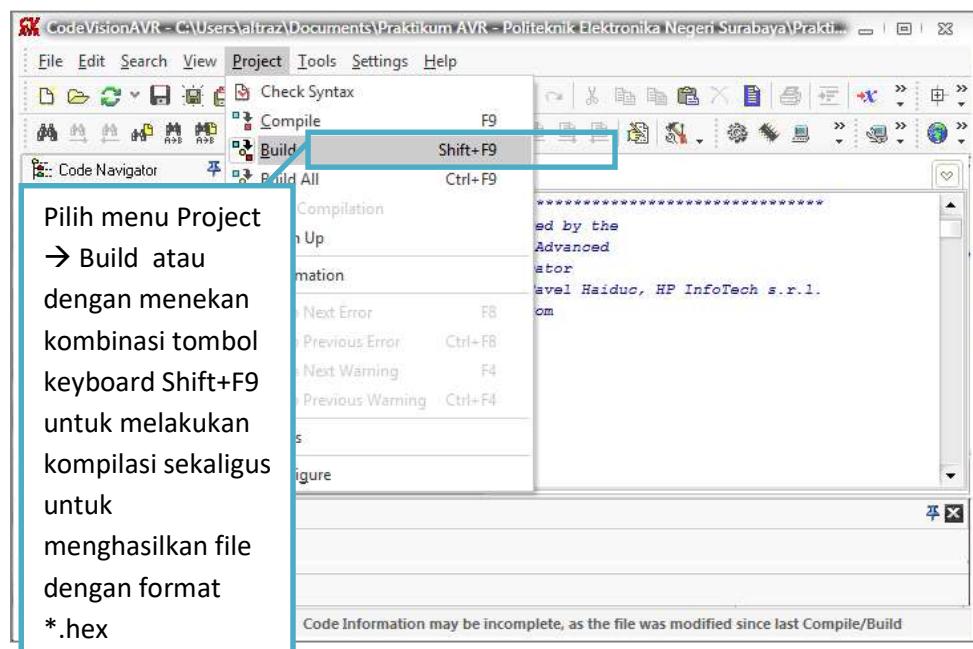
// LCD module initialization
lcd_init(16);

while (1)
{
    // Place your code here
    i2c_start();
    i2c_write(0xC0);
    i2c_write(0x01);
    i2c_start();
    i2c_write(0xC1);
    data=i2c_read(0);
    i2c_stop();
    lcd_gotoxy(0,0);
    sprintf(xstring,"Data = %3d",data);
    lcd_puts(xstring);
    delay_ms(150);

}

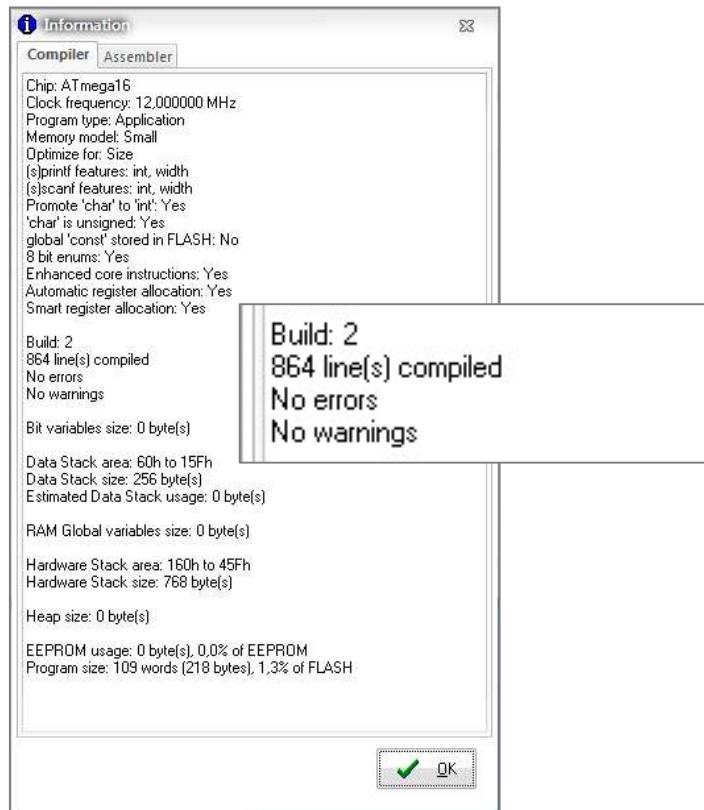
```

6. Setelah pembuatan program selesai, maka proses selanjutnya yaitu lakukan kompilasi terhadap kode program tersebut sekaligus untuk menghasilkan file dengan format hexadecimal (*.hex). Hal ini dapat dilakukan dengan memilih menu Project → Build (Shift+F9) seperti ditunjukkan pada gambar berikut :



Gambar 8.15 Build Source

Apabila kode program sudah benar (tanpa error) dan tidak menyalahi struktur pemrograman bahasa C, maka akan muncul dialog informasi seperti ditunjukkan pada gambar berikut:



Gambar 8.16 Dialog informasi status kompilasi

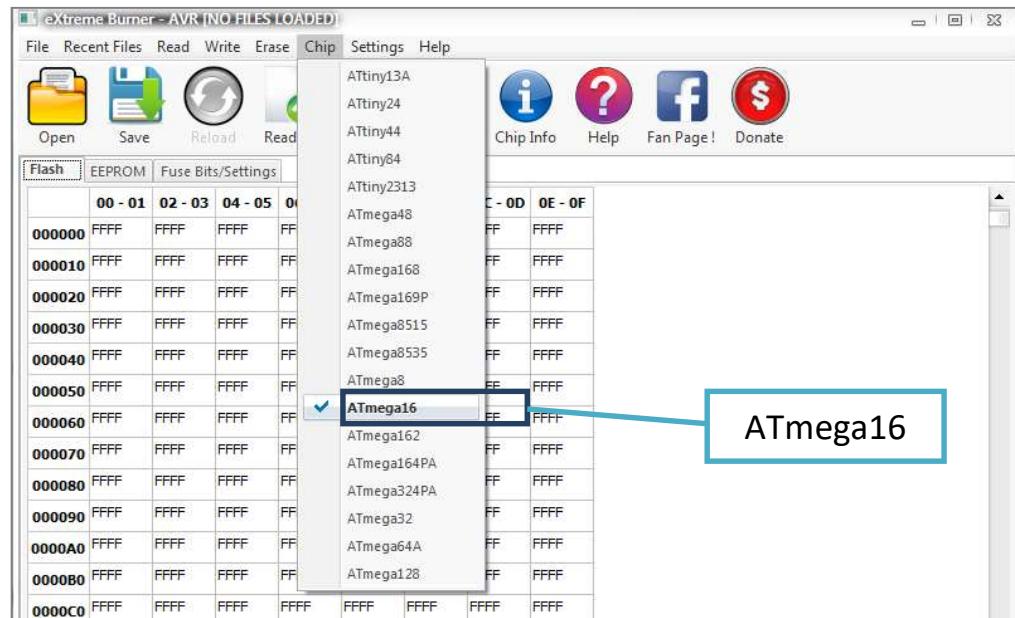
7. Setelah diperoleh file dengan format *.hex, maka lakukan download file *.hex ke dalam mikrokontroler. Pertama, jalankan program eXtreme Burner - AVR dengan cara melakukan klik ganda pada shortcut icon eXtreme Burner– AVR pada desktop



Gambar 8.17 Shortcut eXtreme Burner - AVR

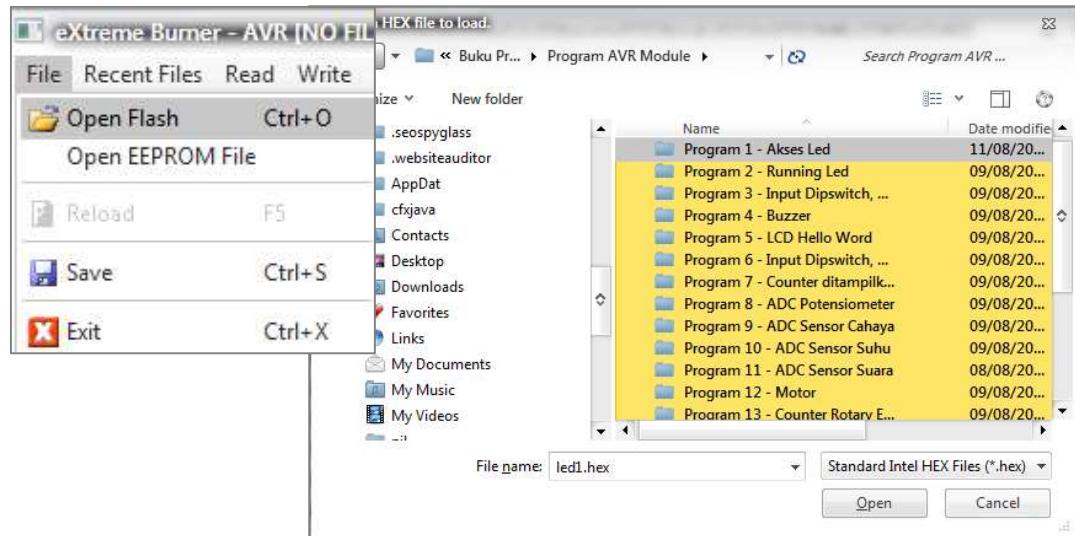
Selanjutnya akan muncul tampilan utama dari program eXtreme Burner – AVR. Pada praktikum ini menggunakan mikrokontroler keluarga AVR dengan tipe IC ATMega16. Oleh karena itu, lakukan pengaturan terhadap tipe IC terlebih dahulu

dengan memilih menu Chip → ATmega16 seperti ditunjukkan pada gambar berikut:



Gambar 8.18 Tipe Mikrokontroler

Setelah itu, lakukan open file *.hex dengan memilih menu File → Open Flash (Ctrl+O) seperti ditunjukkan pada gambar berikut:



Gambar 8.19 Open Flash

Lakukan pencarian terhadap lokasi file *.hex Anda, untuk selanjutnya tekan tombol "Open" apabila sudah ditemukan file yang dimaksud. Setelah itu akan

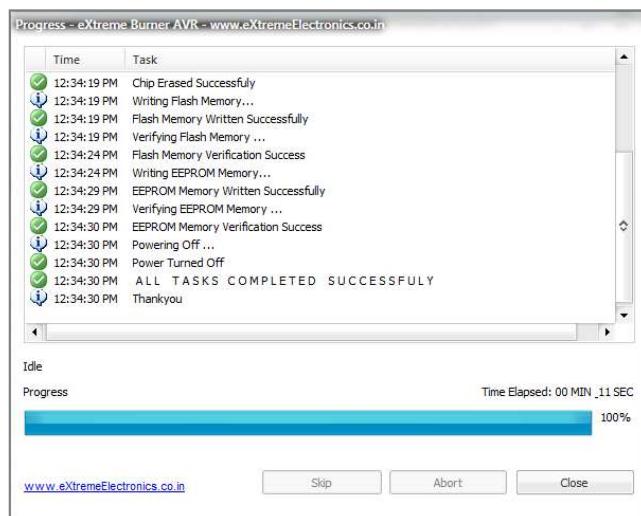
muncul pesan yang memberitahukan bahwa file *hex berhasil di-load program eXtreme Burner – AVR.



Gambar 8.20 Message



Selanjutnya tekan tombol **Write All** untuk memulai download ke dalam mikrokontroler Anda. Berikut ini merupakan tampilan apabila proses download ke dalam mikrokontroler berhasil dilakukan:



Gambar 8.21 Proses Berhasil

8. Selesai

8.6. Hasil Percobaan

Amati hasil praktikum Anda

8.7. Tugas

Buat simulasi posisi dengan 4 kondisi kuadran, dimana masing-masing menunjukkan kondisi aman, bahaya 1, bahaya 2, dan bahaya 4.

PERCOBAAN 9

SENSOR PANAS API : UVTRON

PERCOBAAN 9

SENSOR PANAS API : UVTRON

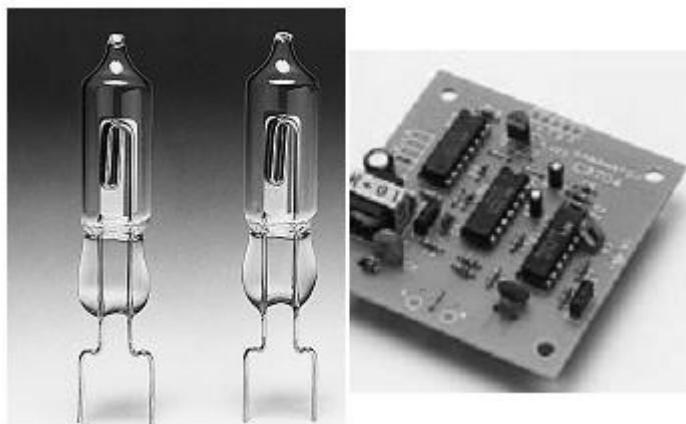
9.1. Tujuan

1. Mahasiswa dapat mengetahui cara kerja sensor api UVTron
2. Mahasiswa dapat mengakses UVTron pada ATMega16
3. Mahasiswa dapat membuat aplikasi sederhana yang mengimplementasikan UVTron pada ATMega16

9.2. Dasar Teori

Sensor UVTron Flame Detector memberikan sinyal aktif apabila mendeteksi adannya sinyal ultraviolet. UVTron dapat menemukan nyala api dalam jarak 5 meter dari sumber dan alat ini beroprasi dalam jangkauan spektrum 185 sampai dengan 160 nm. Alat ini terdiri dari 2 paket yaitu:

1. Hamamatsu R2868 Flame (UV) Sensor
2. UVTron C3704 Rangkaian driver



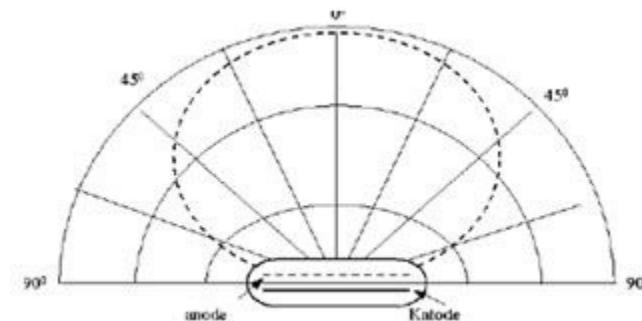
Gambar 9.1 UVTron Flame Detector

UVTron adalah suatu device yang sangat sederhana. Ketika katoda diarahkan pada sinar ultraviolet, photoelektron dipancarkan dari katode secara efek photoelectric dan kemudian dipercepat ke arah anoda dengan medan elektrik. Ketika tegangan yang diterapkan menjadi lebih tinggi dan medan elektrik bertambah kuat, energi kinetik dari elektron menjadi cukup besar untuk mengionisasikan molekul-molekul gas yang terdapat pada tabung dengan cara dibenturkan.

Elektron-elektron yang dihasilkan dari ionisasi dipercepat, sehingga memungkinkannya untuk mengionisasi molekul-molekul lain sebelum mencapai

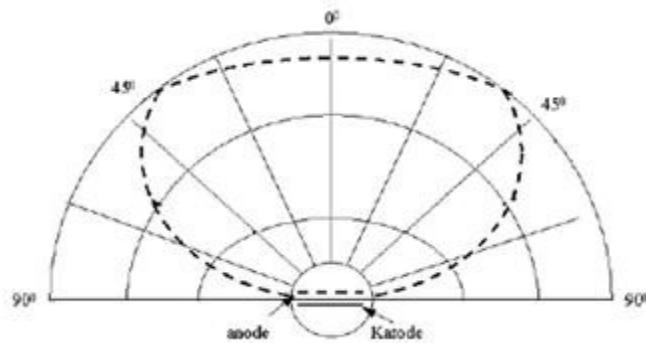
anoda. Pada sisi lain, ion positive dipercepat ke arah katode dan menabrak sehingga membangkitkan elektron-elektron kedua. Proses ini menyebabkan arus yang besar antara elektroda-elektroda dan saat proses pelepasan berlangsung. Pelepasan yang pertama terjadi, tabung terisi dengan electron-elektron dan ion-ion. Tegangan turun atau jatuh antara katoda dan anoda dengan cepat. Status ini akan terjadi tanpa menurunkan tegangan anode sampai di bawah titik jenuh.

Rangkaian pengarah menciptakan perbedaan tegangan yang diperlukan pada tabung untuk mengijinkan proses peluruhan ketika terkena sinar ultraviolet. Kemudian rangkaian mengamati arus keluaran dari tabung dan ketika proses peluruhan terjadi, tegangan pada anode dikurangi oleh rangkaian untuk mengijinkan bola lampu mengulang lagi atau mereset. Tiap waktu proses peluruhan dan pelepasan terjadi, sinyal dibangkitkan dengan sirkuit atau rangkaian dengan beberapa pengaruh untuk latar belakang. Gambar berikut menunjukkan jangkauan sensor pada posisi tidur



Gambar 9.2 Jangkauan sensor pada posisi tidur

Posisi dari tabung Uvtron mempengaruhi jarak dari jangkauan pendeksiian sinar. Dengan posisi berdiri jangkauan lebih jauh tetapi jangkauan luasan daerah lebih sempit hal ini berkebalikan dengan posisi tidur sehingga posisi dari tabung harus disesuaikan dengan kebutuhan. Gambar berikut menunjukkan jangkauan sensor posisi berdiri.

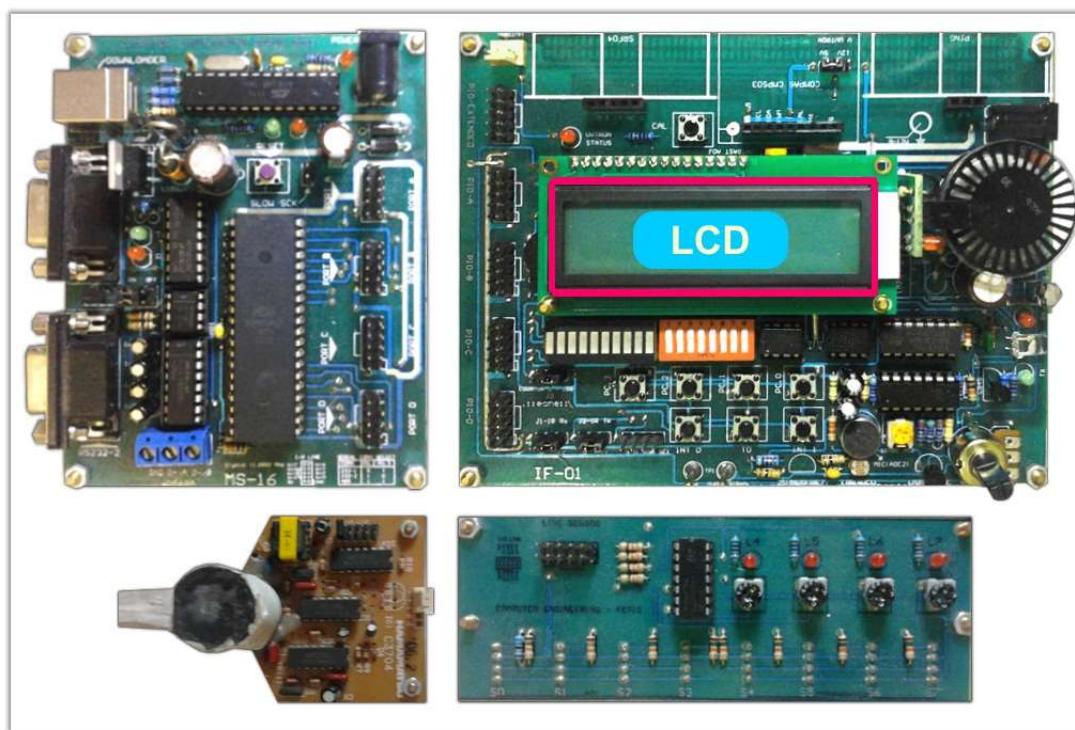


Gambar 9.3 Jangkauan sensor pada posisi berdiri

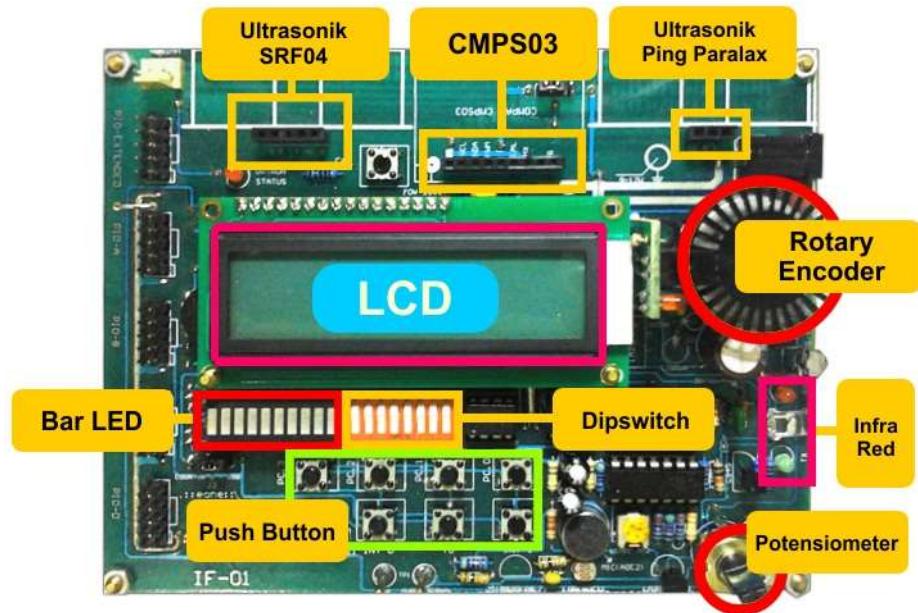
9.3. Peralatan

- 1 set PC yang dilengkapi dengan software CodeVision AVR.
- 1 set development board AVR ATmega16
- 1 power-supply +9VDC
- Korek api

9.4. Modul I/O

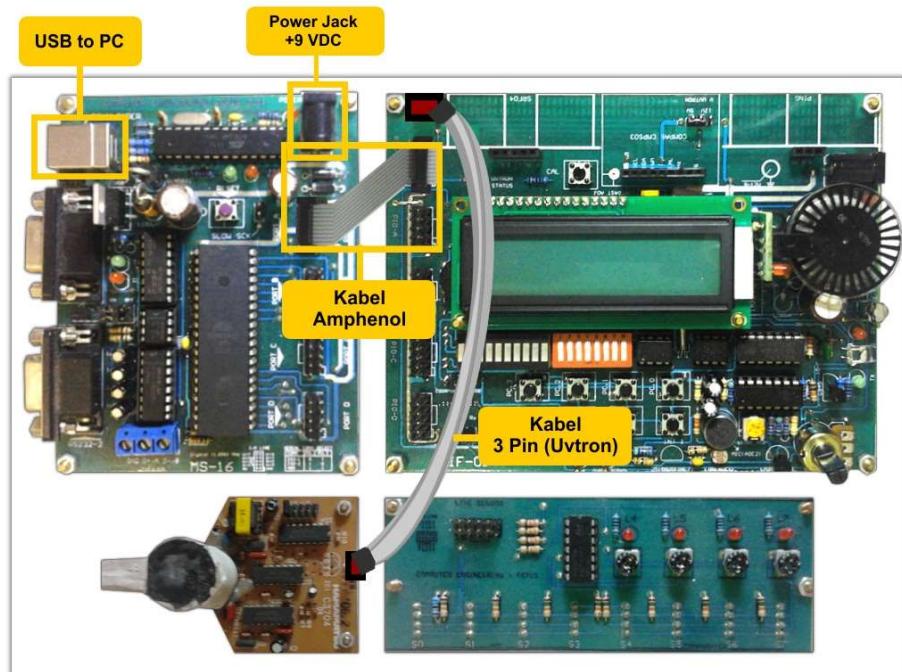


Gambar 9.4 Development board



Gambar 9.5 Komponen Training board

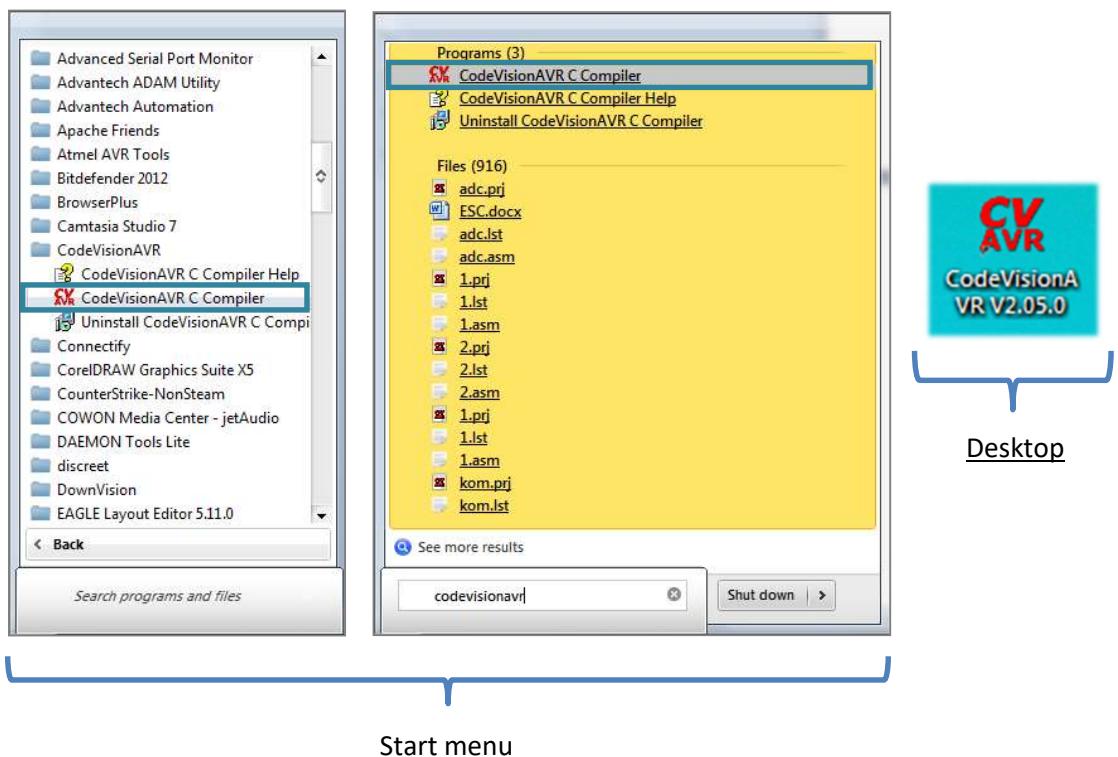
Hubungkan sensor UVtron pada training board menggunakan kabel 3 pin. Koneksi training board ke minimum system menggunakan kabel amphenol seperti ditunjukkan pada gambar berikut :



Gambar 9.6 Konfigurasi Training board

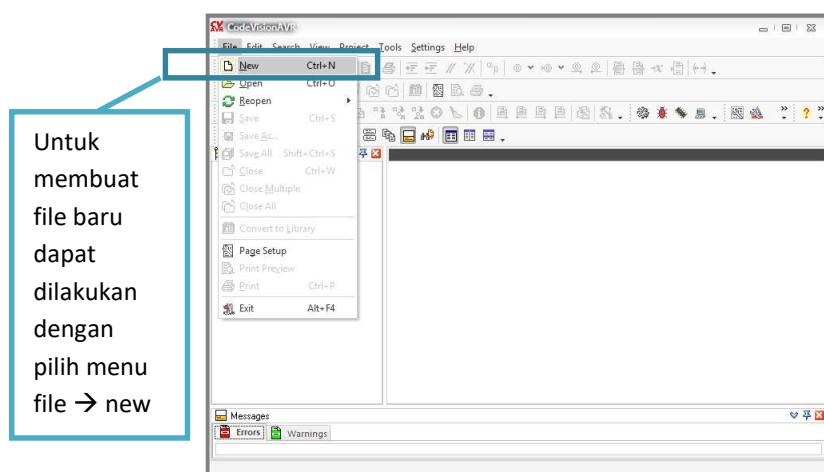
9.5. Prosedur Praktikum

1. Jalankan aplikasi CodeVisionAVR dengan cara melakukan klik ganda pada shortcut icon CodeVisionAVR pada desktop atau dengan cara mencari shortcut program pada start menu system operasi windows.



Gambar 9.7 Shortcut CodeVisionAVR

Untuk memulai membuat project baru, pada menubar, pilih File → New, seperti yang ditunjukkan oleh gambar berikut:



Gambar 9.8 Membuat File baru

Setelah itu akan muncul suatu dialog yang mengharuskan *user* untuk memilih antara pembuatan source file atau project. Dalam setiap percobaan pada praktikum ini, dibuat project baru untuk setiap percobaan, oleh karena itu *user* sebaiknya melakukan *check* pada checkbox project dan dilanjutkan dengan menekan tombol “OK”



Gambar 9.9 Membuat project baru

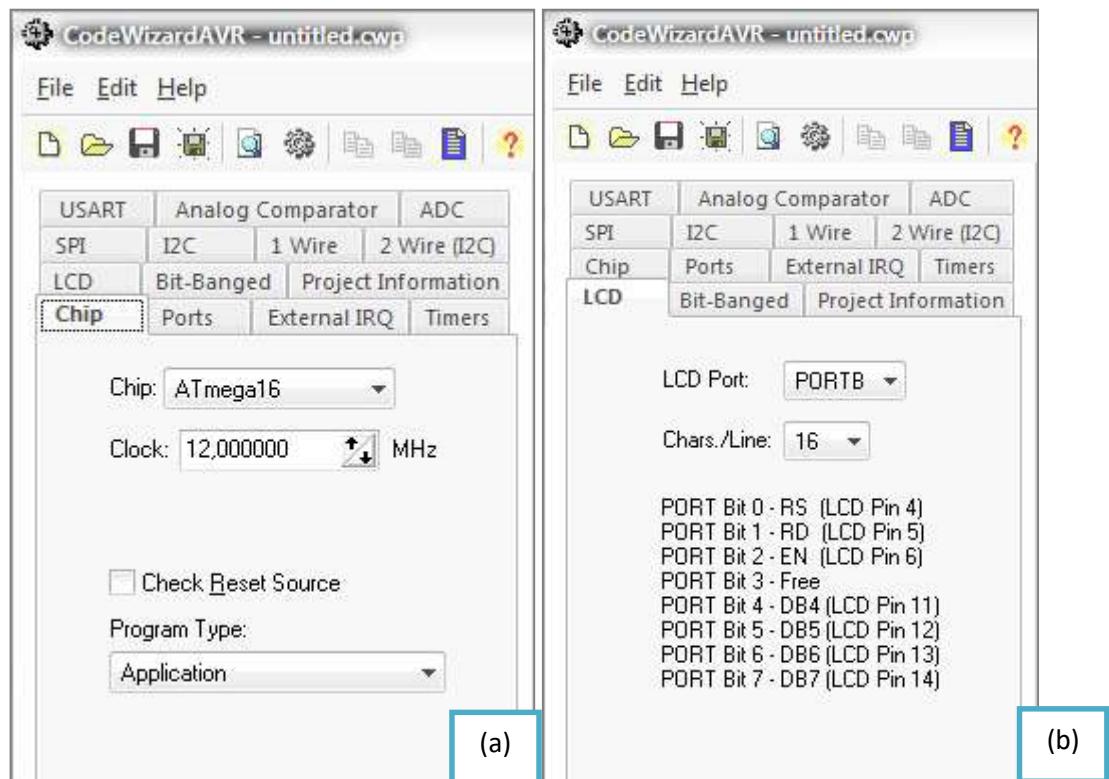
Berikutnya akan keluar dialog yang menanyakan Anda apakah akan membuat project baru. Klik tombol “Yes” untuk menyetujui.



Gambar 9.10 Memilih untuk membuat project baru pada CodeVisionAVR

Setelah itu lakukan set pada CodeWizardAVR. Untuk chip diset dengan nama chip ATmega16 dengan clock 12 MHz.

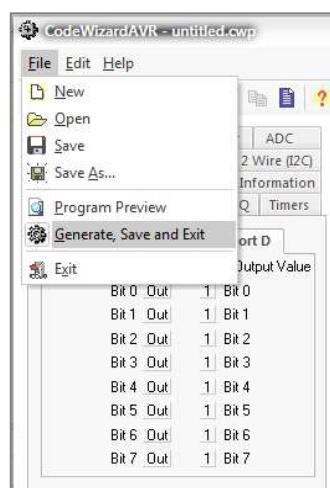
Berikutnya Anda akan menginisialisasi LCD pada PortB, yaitu dengan cara masuk pada tab LCD dan pilih item **PORTB** pada selectbox “LCD Port”. Set banyaknya karakter dari LCD yang diinginkan. Dalam hal ini menggunakan 16 karakter, oleh karena itu pilih item **16** pada selectbox “Chars/Line”



Gambar 9. 11(a) CodeWizardAVR pada tab Chip

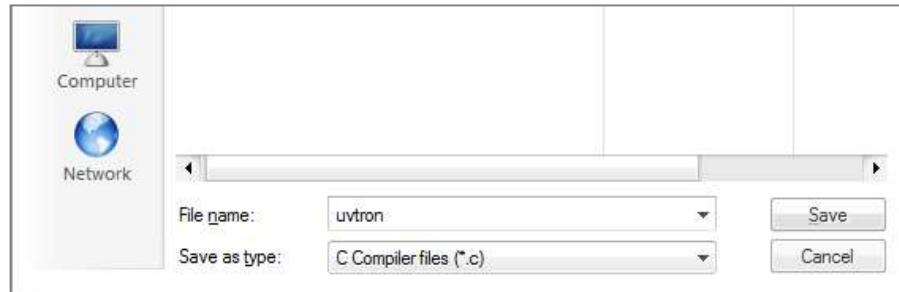
(b) Setting LCD pada PortB

Karena pada contoh ini tidak digunakan fasilitas lain maka setting CodeWizardAVR siap disimpan dalam file. Pada menu CodeWizardAVR, pilih File → Generate, Save and Exit, seperti yang ditunjukkan pada gambar berikut:



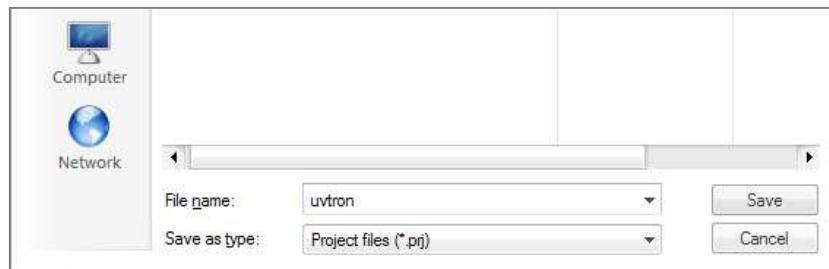
Gambar 9.12 Menyimpan setting

Setelah penekanan menu “Generate, Save and Exit”, akan muncul suatu dialog untuk melakukan set terhadap nama dan lokasi penyimpanan dari source file (*.c) seperti ditunjukkan pada gambar berikut:



Gambar 9.13 Memberi nama pada file C (*.c)

Tekan tombol “Save” untuk menyetujui dan melanjutkan pada proses berikutnya. Setelah penekanan tombol “Save” akan muncul kembali dialog yang kedua, yaitu untuk melakukan set terhadap nama dan lokasi penyimpanan project file (*.prj) seperti ditunjukkan pada gambar berikut:



Gambar 9.14 Memberi nama project (*.prj)

Tekan tombol “Save” untuk menyetujui dan melanjutkan pada proses berikutnya. Setelah penekanan tombol “Save” akan muncul kembali dialog yang ketiga, yaitu untuk melakukan set terhadap nama dan lokasi penyimpanan CodeWizardProject file (*.cwp) seperti ditunjukkan pada gambar berikut:



Gambar 9.15 Memberi nama CodeWizardProject (*.cwp)

Tekan tombol “Save” untuk menyetujui dan melanjutkan. Selanjutnya akan muncul IDE editor program dari CodeVisionAVR. Anda siap untuk melakukan pembuatan program menggunakan CodeVisionAVR.

2. Pada program-C yang dihasilkan, tambahkan header `delay.h`
3. Pada bagian `while(1) {...}` tambahkan perintah yang telah diblok dengan warna kuning berikut :

```
while (1)
{
// Place your code here
lcd_clear();
if(PINA.6==1){
    lcd_putsf("Ada Api!!");
}
else
    lcd_putsf("Tidak ada api");

    delay_ms(100);
};
```

4. Untuk lebih jelasnya, berikut kode program secara keseluruhan:

```
#include<mega16.h>
#include<delay.h>

// Alphanumeric LCD Module functions
#asm
    .equ __lcd_port=0x18 ;PORTB
#endasm
#include<lcd.h>

// Declare your global variables here

void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTA=0x00;
DDRA=0x00;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
```

```

// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTB=0x00;
DDRB=0x00;

// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTD=0x00;
DDRD=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;

```

```

TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

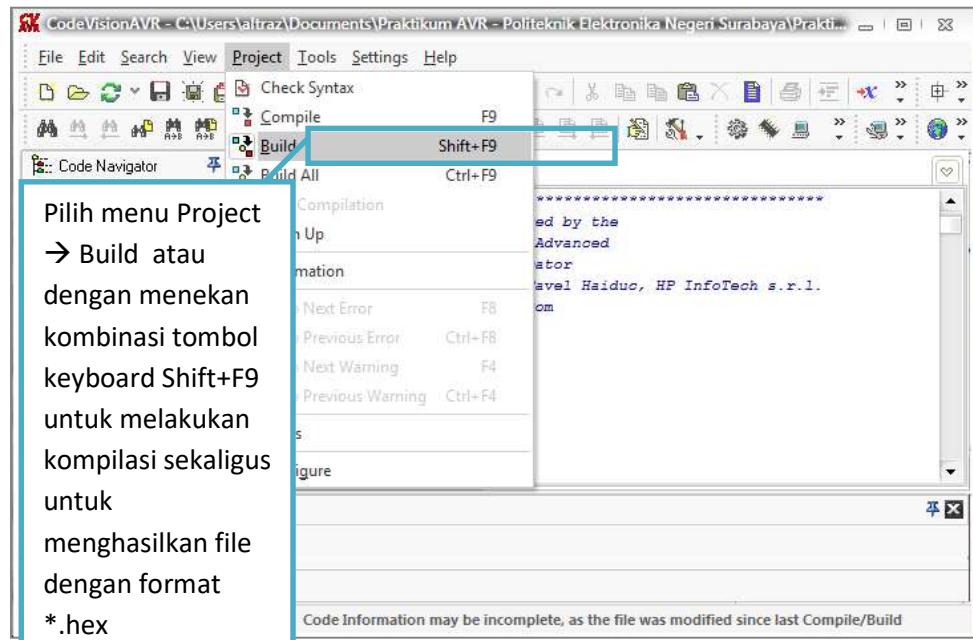
// LCD module initialization
lcd_init(16);

while (1)
{
    // Place your code here
    lcd_clear();
    if(PINA.6==1) {
        lcd_putsf("Ada Api!!");
    }
    else
        lcd_putsf("Tidak ada api");

    delay_ms(100);
}
}

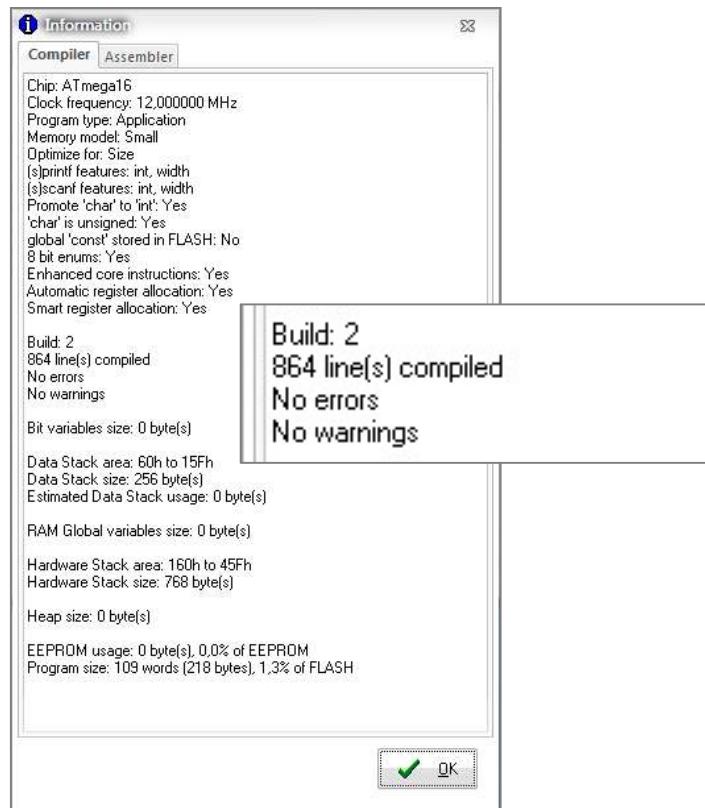
```

5. Setelah pembuatan program selesai, maka proses selanjutnya yaitu lakukan kompilasi terhadap kode program tersebut sekaligus untuk menghasilkan file dengan format hexadecimal (*.hex). Hal ini dapat dilakukan dengan memilih menu Project → Build (Shift+F9) seperti ditunjukkan pada gambar berikut :



Gambar 9.16 Build Source

Apabila kode program sudah benar (tanpa error) dan tidak menyalahi struktur pemrograman bahasa C, maka akan muncul dialog informasi seperti ditunjukkan pada gambar berikut:



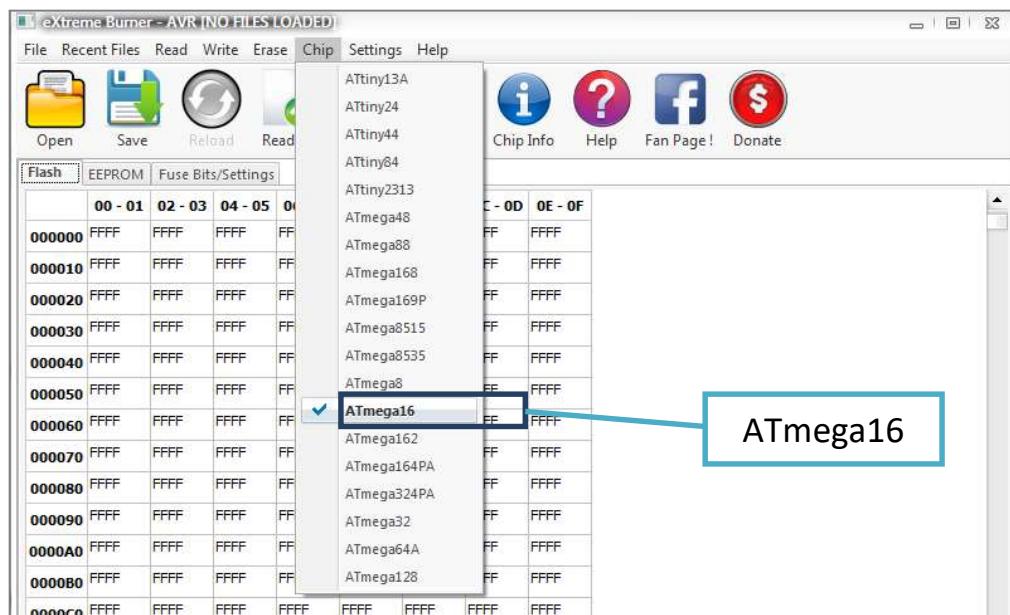
Gambar 9.17 Dialog informasi status kompilasi

6. Setelah diperoleh file dengan format *.hex, maka lakukan download file *.hex ke dalam mikrokontroler. Pertama, jalankan program eXtreme Burner - AVR dengan cara melakukan klik ganda pada shortcut icon eXtreme Burner– AVR pada desktop



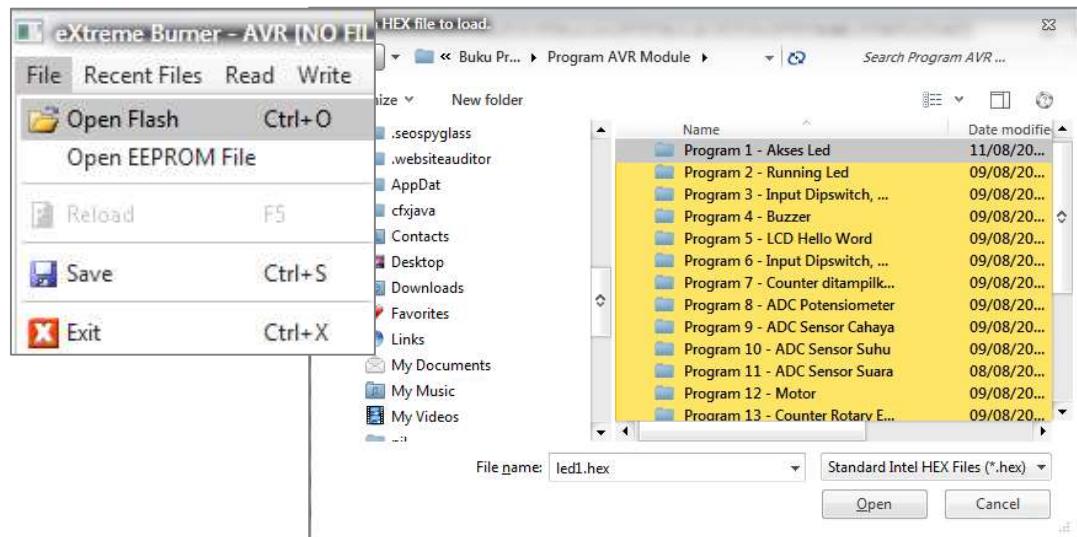
Gambar 9.18 Shortcut eXtreme Burner - AVR

Selanjutnya akan muncul tampilan utama dari program eXtreme Burner – AVR. Pada praktikum ini menggunakan mikrokontroler keluarga AVR dengan tipe IC ATMega16. Oleh karena itu, lakukan pengaturan terhadap tipe IC terlebih dahulu dengan memilih menu Chip → ATmega16 seperti ditunjukkan pada gambar berikut:



Gambar 9.19 Tipe Mikrokontroler

Setelah itu, lakukan open file *.hex dengan memilih menu File → Open Flash (Ctrl+O) seperti ditunjukkan pada gambar berikut:



Gambar 9.20 Open Flash

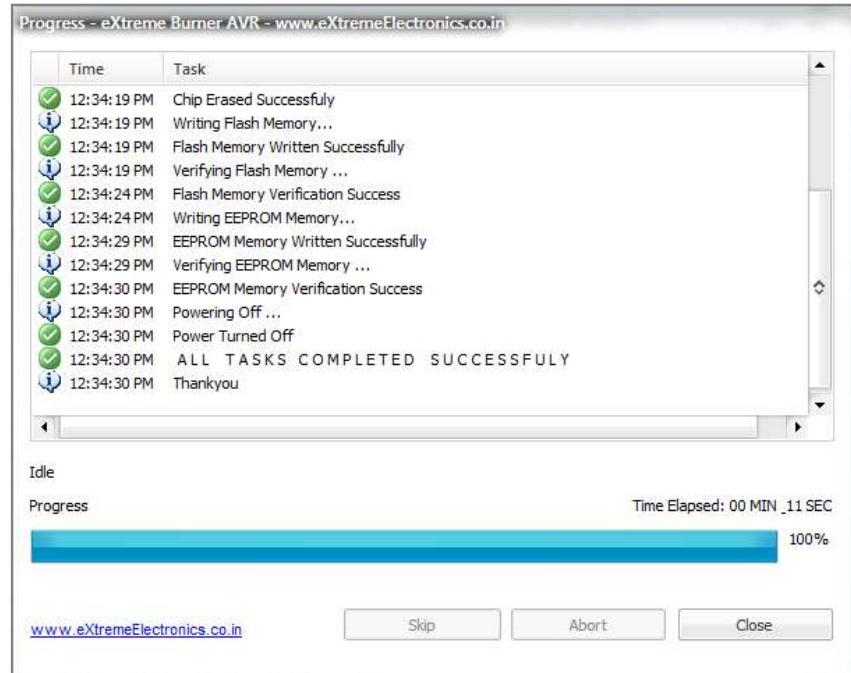
Lakukan pencarian terhadap lokasi file *.hex Anda, untuk selanjutnya tekan tombol “Open” apabila sudah ditemukan file yang dimaksud. Setelah itu akan muncul pesan yang memberitahukan bahwa file *hex berhasil di-load program eXtreme Burner – AVR.



Gambar 9.21 Message



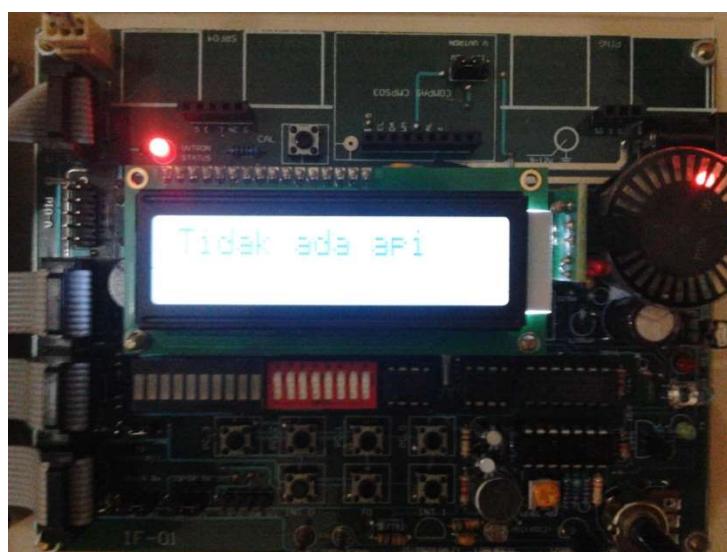
Selanjutnya tekan tombol **Write All** untuk memulai download ke dalam mikrokontroler Anda. Berikut ini merupakan tampilan apabila proses download ke dalam mikrokontroler berhasil dilakukan:



Gambar 9.22 Proses Berhasil

7. Selesai

9.6. Hasil Percobaan



Gambar 9.23 Tidak ada api



Gambar 9.24 Ada api

PERCOBAAN 10

SENSOR TEKANAN MPX5100GP

PERCOBAAN 10

SENSOR TEKANAN MPX5100GP

10.1. Tujuan

1. Mahasiswa dapat mengetahui cara kerja sensor tekanan MPX5100GP
2. Mahasiswa dapat mengakses sensor tekanan MPX5100GP dengan ATMega 16
3. Mahasiswa dapat membuat program sederhana untuk mengimplementasikan penggunaan sensor MPX5100GP

10.2. Dasar Teori

1. Sensor Tekanan

Sensor Tekanan adalah sensor untuk mengukur tekanan suatu zat. Tekanan (p) adalah satuan fisika untuk menyatakan gaya (F) per satuan luas (A). Satuan tekanan sering digunakan untuk mengukur kekuatan dari suatu cairan atau gas.

Satuan tekanan dapat dihubungkan dengan satuan volume (isi) dan suhu. Semakin tinggi tekanan di dalam suatu tempat dengan isi yang sama, maka suhu akan semakin tinggi. Hal ini dapat digunakan untuk menjelaskan mengapa suhu di pegunungan lebih rendah dari pada di dataran rendah, karena di dataran rendah tekanan lebih tinggi.

2. Sensor Tekanan MPX5100GP

Sensor tekanan MPX5100gp merupakan sensor tekanan dengan kompensasi suhu, pengkondisi sinyal, dan telah terkalibrasi.

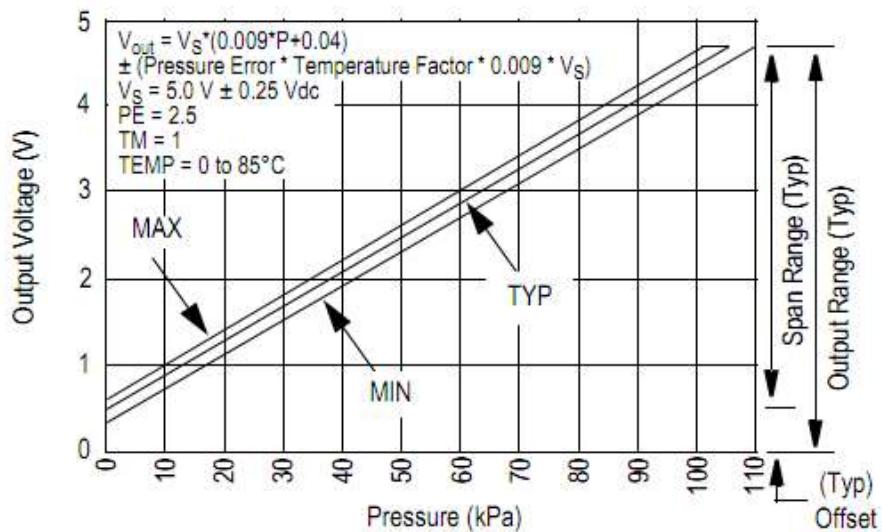


Gambar 10.1 Sensor Tekanan MPX5100GP

Spesifikasi:

- Type : gauge, single port.
- Pressure range : 0-100 kPa.

- Supply : 5 VDC, 7mA typ.
- Sensitivity : 45 mV/kPa.
- Response time : 1 ms.
- Warm-up time : 20 ms.
- Error : 2,5% max.



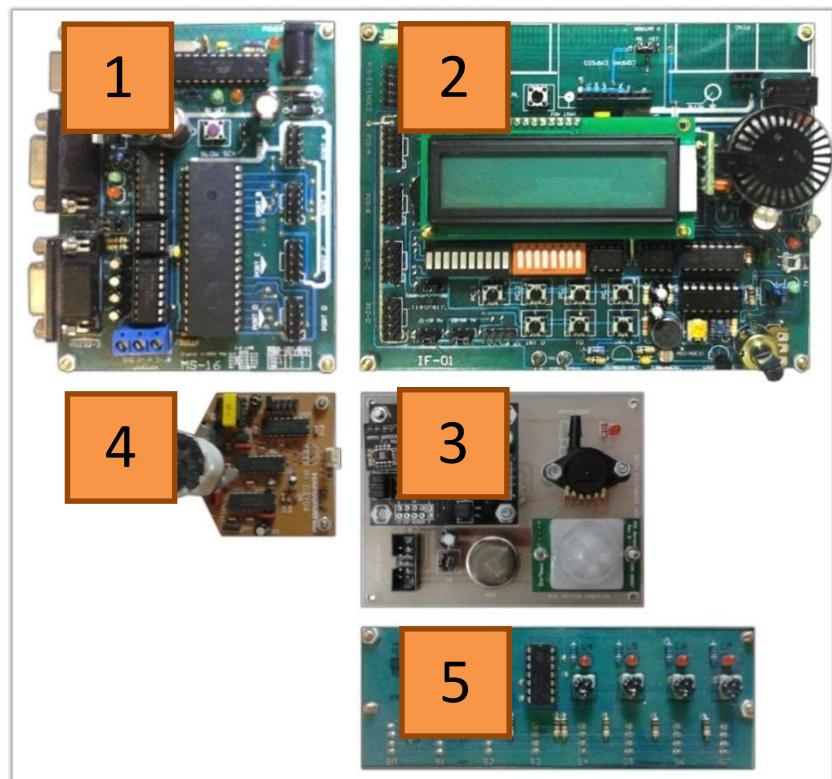
Gambar 10.2 Grafik hubungan tekanan dengan output (V)

10.3. Peralatan

- 1 set PC yang dilengkapi dengan software CodeVision AVR.
- 1 set development board AVR ATmega16
- 1 power-supply +9VDC
- Pompa Karet

10.4. Modul I/O

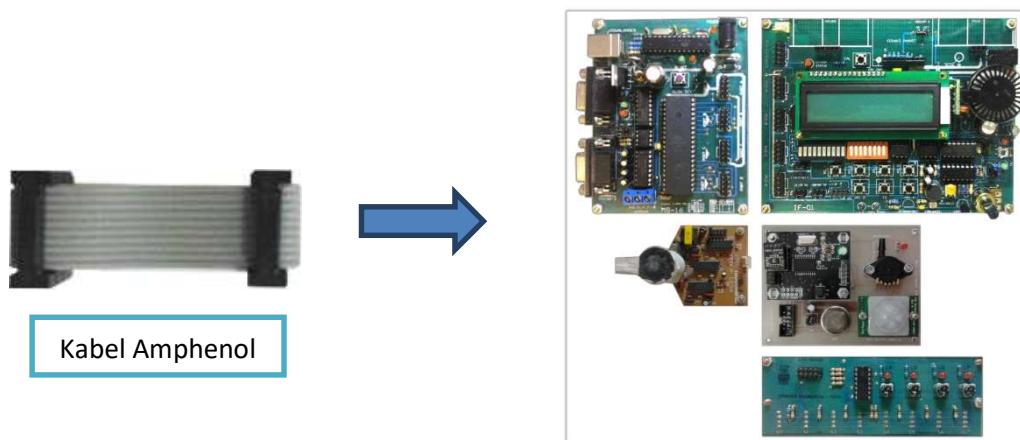
Berikut ini merupakan development board AVR ATmega16:



Gambar 10.3 Development board

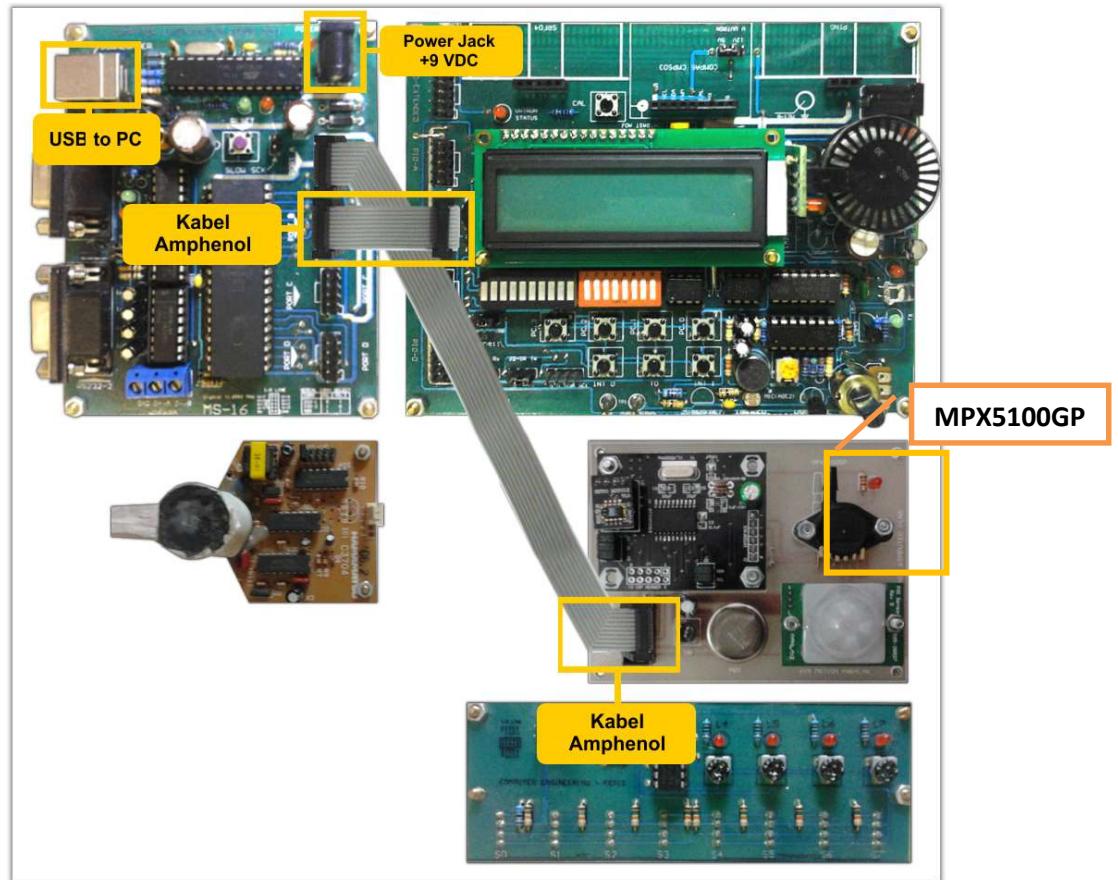
- **Konfigurasi modul pada percobaan ini :**

Pasangkan kabel amphenol pada modul development untuk menghubungkan sistem minimum ATMega16 dengan Training Board. Pasangkan sensor ultrasonic ping parallax pada training board.



Gambar 10.4 Komponen Training board

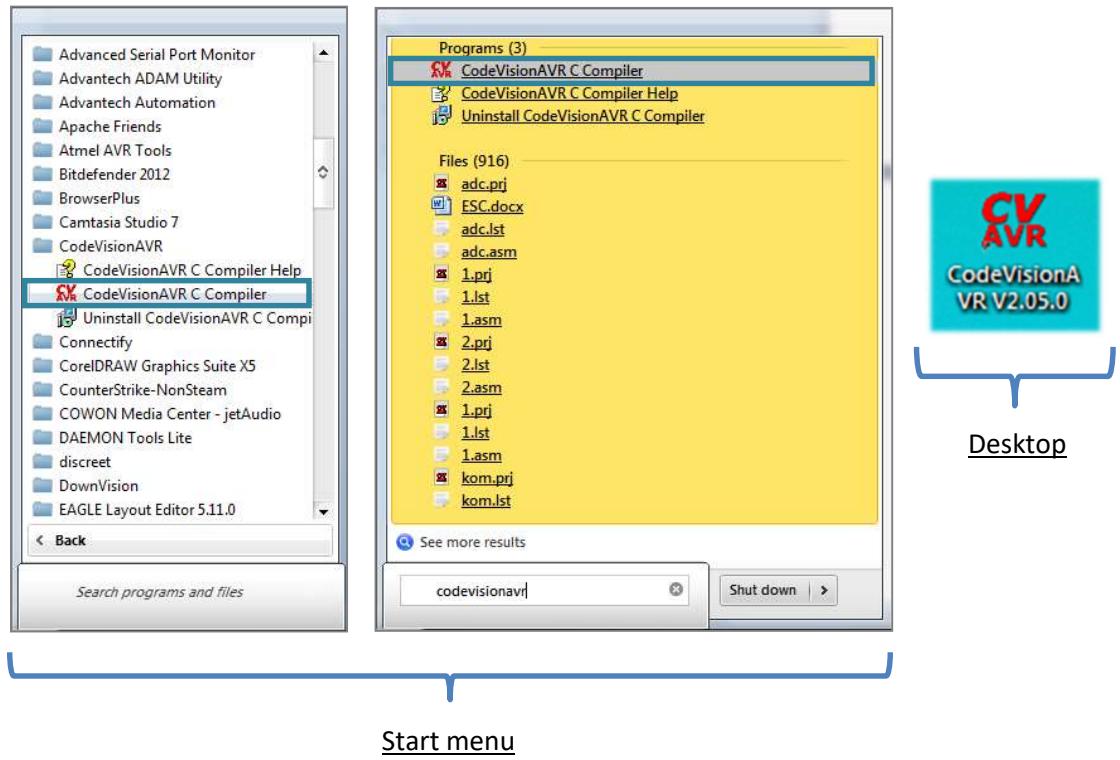
Sehingga menjadi seperti ditunjukkan pada gambar berikut:



Gambar 10.5 Komponen Training board

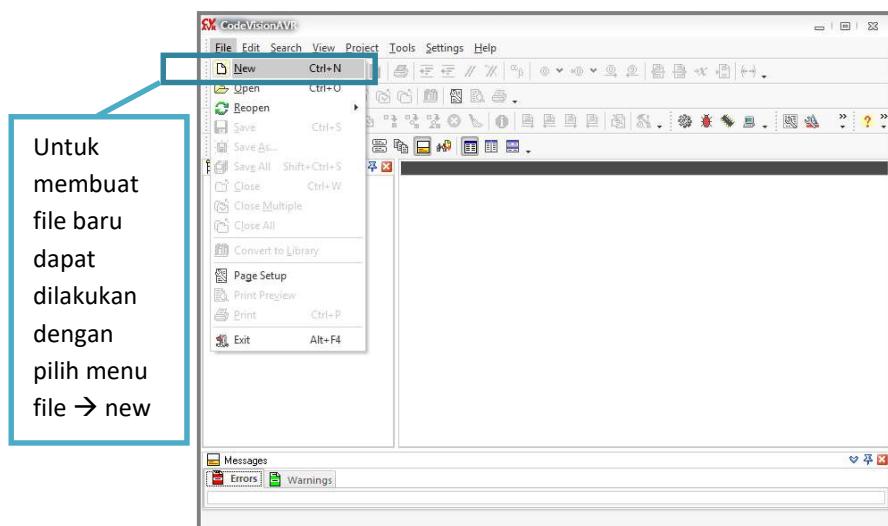
10.5. Prosedur Praktikum

1. Jalankan aplikasi CodeVisionAVR dengan cara melakukan klik ganda pada shortcut icon CodeVisionAVR pada desktop atau dengan cara mencari shortcut program pada start menu system operasi windows.



Gambar 10.6 Shortcut CodeVisionAVR

Untuk memulai membuat project baru, pada menubar, pilih File → New, seperti yang ditunjukkan oleh gambar berikut:



Gambar 10.7 Membuat File baru

Setelah itu akan muncul suatu dialog yang mengharuskan *user* untuk memilih antara pembuatan source file atau project. Dalam setiap percobaan pada praktikum ini, dibuat project baru untuk setiap percobaan, oleh karena itu *user* sebaiknya melakukan *check* pada checkbox project dan dilanjutkan dengan menekan tombol “OK”



Gambar 10.8 Membuat project baru

Berikutnya akan keluar dialog yang menanyakan Anda apakah akan membuat project baru. Klik tombol “Yes” untuk menyetujui.

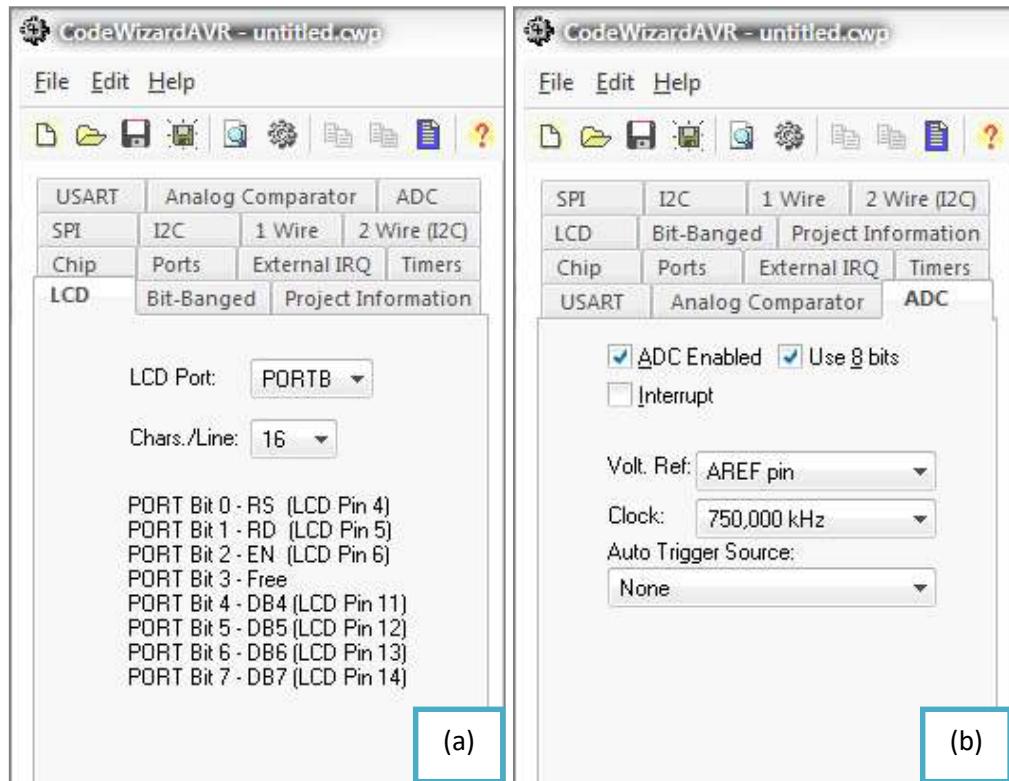


Gambar 10.9 Memilih untuk membuat project baru pada CodeVisionAVR

Setelah itu lakukan set pada CodeWizardAVR. Untuk chip diset dengan nama chip ATmega16 dengan clock 12 MHz.

Berikutnya Anda akan menginisialisasi LCD pada PortB, yaitu dengan cara masuk pada tab LCD dan pilih item **PORTB** pada selectbox “LCD Port”. Set banyaknya karakter dari LCD yang diinginkan. Dalam hal ini menggunakan 16 karakter, oleh karena itu pilih item **16** pada selectbox “Chars/Line”.

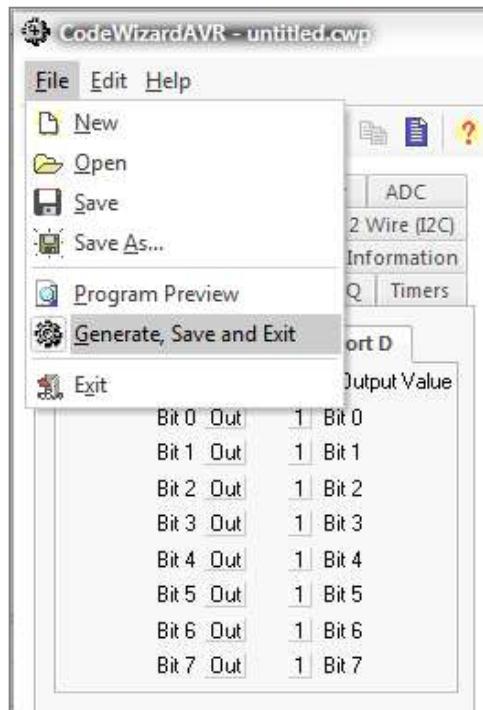
Setelah itu lakukan set ADC dengan masuk pada tab ADC. Berikan tanda check terhadap checkbox “ADC Enabled” untuk meng-enable ADC. Berikan tanda check terhadap checkbox “Use 8 bits” untuk menggunakan ADC 8 bit.



Gambar 10.10(a) Setting LCD pada PortB

(b) Setting ADC 8 bit

Karena pada contoh ini tidak digunakan fasilitas lain maka setting CodeWizardAVR siap disimpan dalam file. Pada menu CodeWizardAVR, pilih File → Generate, Save and Exit, seperti yang ditunjukkan pada gambar berikut:



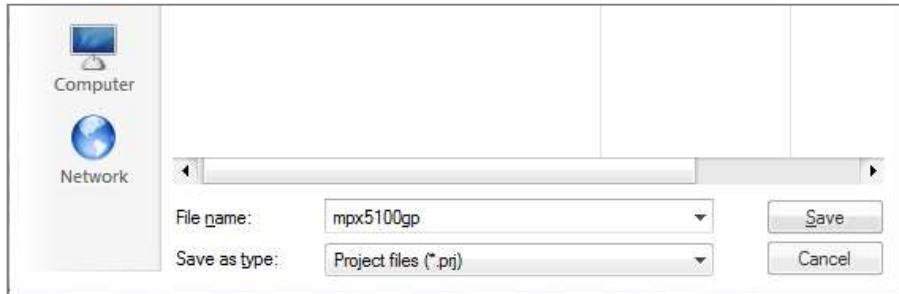
Gambar 10.11 Menyimpan setting

Setelah penekanan menu “Generate, Save and Exit”, akan muncul suatu dialog untuk melakukan set terhadap nama dan lokasi penyimpanan dari source file (*.c) seperti ditunjukkan pada gambar berikut:



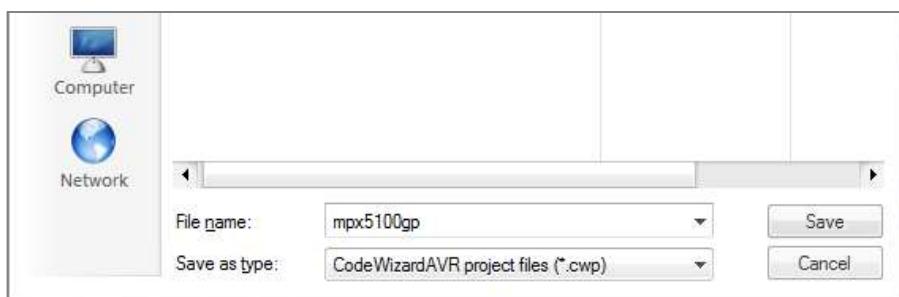
Gambar 10.12 Memberi nama pada file C (*.c)

Tekan tombol “Save” untuk menyetujui dan melanjutkan pada proses berikutnya. Setelah penekanan tombol “Save” akan muncul kembali dialog yang kedua, yaitu untuk melakukan set terhadap nama dan lokasi penyimpanan project file (*.prj) seperti ditunjukkan pada gambar berikut:



Gambar 10.13 Memberi nama project (*.prj)

Tekan tombol “Save” untuk menyetujui dan melanjutkan pada proses berikutnya. Setelah penekanan tombol “Save” akan muncul kembali dialog yang ketiga, yaitu untuk melakukan set terhadap nama dan lokasi penyimpanan CodeWizardProject file (*.cwp) seperti ditunjukkan pada gambar berikut:



Gambar 10.14 Memberi nama CodeWizardProject (*.cwp)

Tekan tombol “Save” untuk menyetujui dan melanjutkan. Selanjutnya akan muncul IDE editor program dari CodeVisionAVR. Anda siap untuk melakukan pembuatan program menggunakan CodeVisionAVR.

2. Pada program-C yang dihasilkan, tambahkan header `delay.h` dan `stdio.h`
3. Pada bagian `while(1) {...}` tambahkan perintah yang telah diblok dengan warna kuning berikut :

```
while (1)
{
// Place your code here
lcd_clear();
    lcd_putsf("Data tek = ");
    data_digital = read_adc(5);
    sprintf(xstring, "%3.0f", data_digital);
    lcd_puts(xstring);

    data_analog = data_digital*5/255;
    nilai_sbnrnya = data_analog / 0.045;
    lcd_gotoxy(0,1);
```

```

    sprintf(xstring,"Press:%3.2f Kpa",nilai_sbnrnya);
    lcd_puts(xstring);
    delay_ms(100);
};

```

4. Untuk lebih jelasnya, berikut kode program secara keseluruhan:

```

#include<mega16.h>

// Alphanumeric LCD Module functions
#asm
    .equ __lcd_port=0x18 ;PORTB
#endasm
#include<lcd.h>

#include<delay.h>
#include<stdio.h>

#define ADC_VREF_TYPE 0x20

// Read the 8 most significant bits
// of the AD conversion result
unsignedchar read_adc(unsignedchar adc_input)
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Delay needed for the stabilization of the ADC input voltage
delay_us(10);
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCH;
}

// Declare your global variables here
unsignedchar xstring[16];
float data_analog,data_digital,nilai_sbnrnya ;

void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTA=0x00;
DDRA=0x00;

// Port B initialization
PORTB=0x00;
DDRB=0x00;

```

```

// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
PORTD=0x00;
DDRD=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;

```

```

MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// Analog Comparator initialization
ACSR=0x80;
SFIOR=0x00;

// ADC initialization
// ADC Clock frequency: 691.200 kHz
// ADC Voltage Reference: AREF pin
// ADC Auto Trigger Source: None
// Only the 8 most significant bits of
// the AD conversion result are used
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x84;

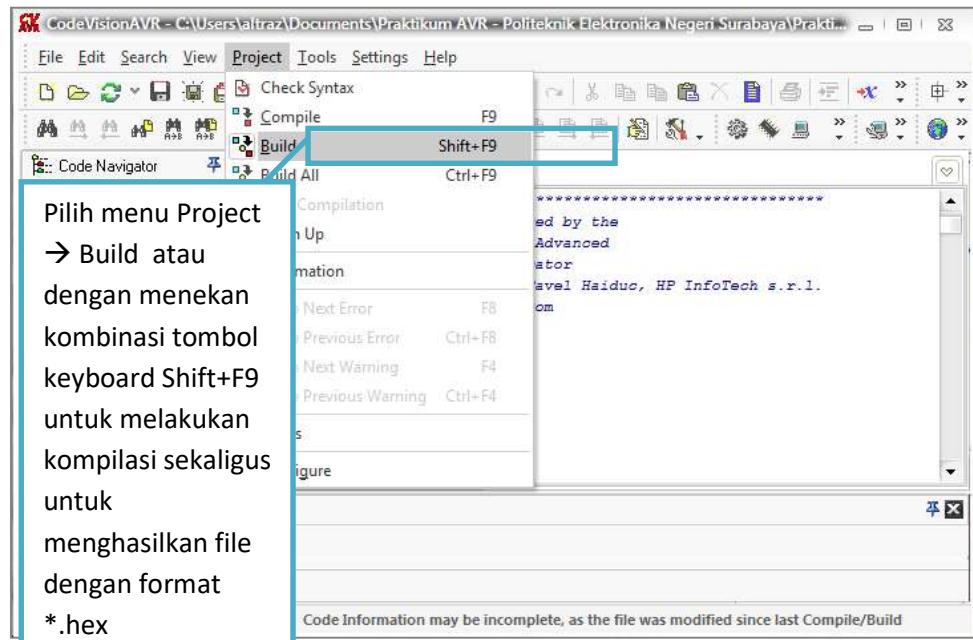
// LCD module initialization
lcd_init(16);

while (1)
{
// Place your code here
    lcd_clear();
    lcd_putsf("Data tek = ");
    data_digital = read_adc(5);
    sprintf(xstring,"%3.0f",data_digital);
    lcd_puts(xstring);

    data_analog = data_digital*5/255;//5:Vref ADC
    nilai_sbnrnya = data_analog / 0.045; //lihat datasheet 45mV/Kpa
    lcd_gotoxy(0,1);
    sprintf(xstring,"Press:%3.2f Kpa",nilai_sbnrnya);
    lcd_puts(xstring);
    delay_ms(100);
}
}

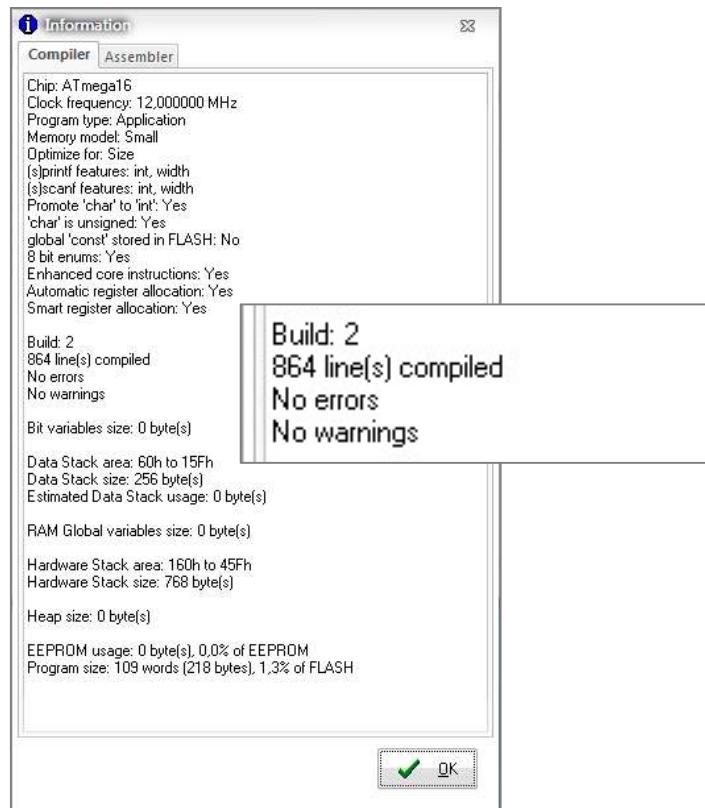
```

5. Setelah pembuatan program selesai, maka proses selanjutnya yaitu lakukan kompilasi terhadap kode program tersebut sekaligus untuk menghasilkan file dengan format hexadecimal (*.hex). Hal ini dapat dilakukan dengan memilih menu Project → Build (Shift+F9) seperti ditunjukkan pada gambar berikut :



Gambar 10.15 Build Source

Apabila kode program sudah benar (tanpa error) dan tidak menyalahi struktur pemrograman bahasa C, maka akan muncul dialog informasi seperti ditunjukkan pada gambar berikut:



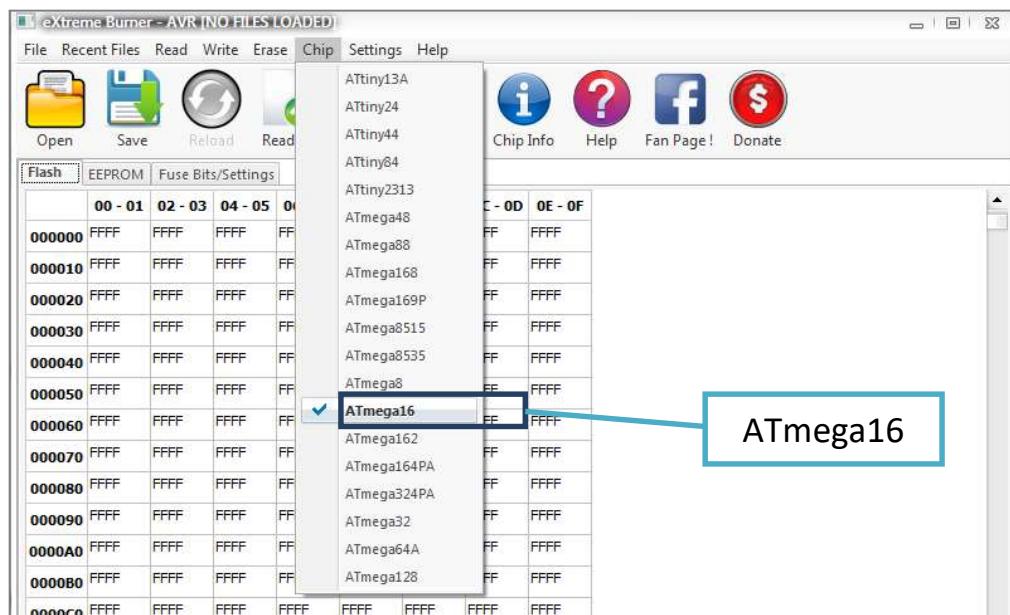
Gambar 10.16 Dialog informasi status kompilasi

6. Setelah diperoleh file dengan format *.hex, maka lakukan download file *.hex ke dalam mikrokontroler. Pertama, jalankan program eXtreme Burner - AVR dengan cara melakukan klik ganda pada shortcut icon eXtreme Burner– AVR pada desktop



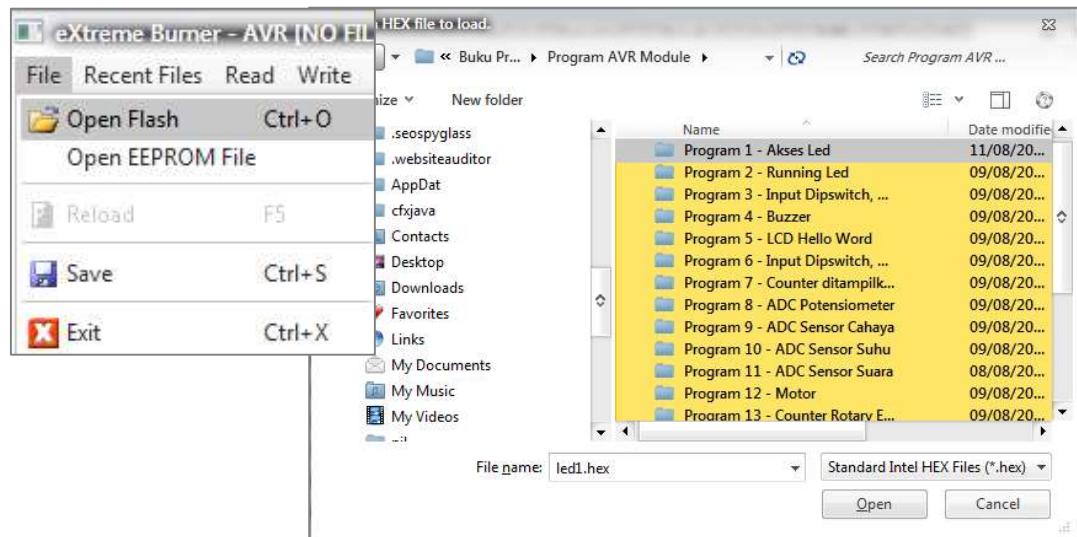
Gambar 10.17 Shortcut eXtreme Burner - AVR

Selanjutnya akan muncul tampilan utama dari program eXtreme Burner – AVR. Pada praktikum ini menggunakan mikrokontroler keluarga AVR dengan tipe IC ATMega16. Oleh karena itu, lakukan pengaturan terhadap tipe IC terlebih dahulu dengan memilih menu Chip → ATmega16 seperti ditunjukkan pada gambar berikut:



Gambar 10.18 Tipe Mikrokontroler

Setelah itu, lakukan open file *.hex dengan memilih menu File → Open Flash (Ctrl+O) seperti ditunjukkan pada gambar berikut:



Gambar 10.19 Open Flash

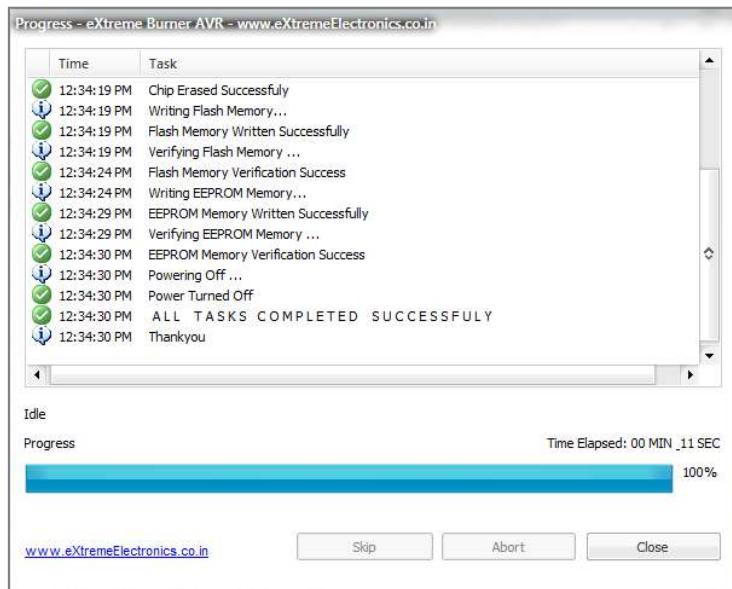
Lakukan pencarian terhadap lokasi file *.hex Anda, untuk selanjutnya tekan tombol “Open” apabila sudah ditemukan file yang dimaksud. Setelah itu akan muncul pesan yang memberitahukan bahwa file *hex berhasil di-load program eXtreme Burner – AVR.



Gambar 10.20 Message



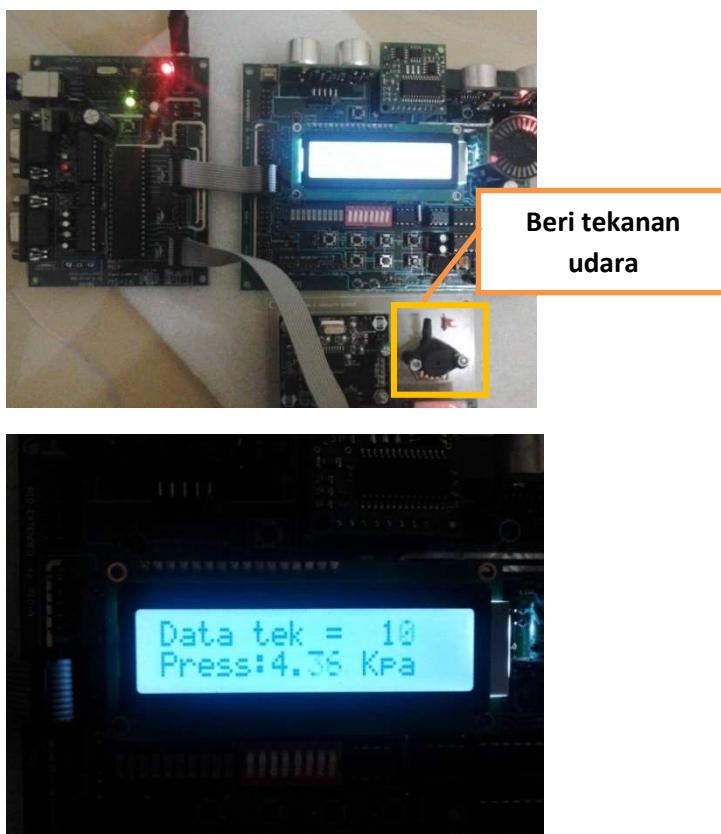
Selanjutnya tekan tombol  untuk memulai download ke dalam mikrokontroler Anda. Berikut ini merupakan tampilan apabila proses download ke dalam mikrokontroler berhasil dilakukan:



Gambar 10.21 Proses Berhasil

7. Selesai

10.6. Hasil Percobaan



Gambar 10.22 Running Program

PERCOBAAN 11

SENSOR GERAKAN :

PIR MOTION PARALAX

PERCOBAAN 11

SENSOR GERAKAN : PIR MOTION PARALAX

11.1. Tujuan

1. Mahasiswa dapat mengetahui cara kerja sensor gerakan PIR MOTION PARALAX
2. Mahasiswa dapat mengakses PIR pada mikrokontroler ATMega 16
3. Mahasiswa dapat membuat program sederhana untuk mengimplementasikan penggunaan PIR

11.2. Dasar Teori

1. Sensor

Sensor adalah suatu alat yang dapat mengukur atau mendeteksi kondisi sebenarnya di dunia nyata, seperti pergerakan, panas atau cahaya dan mengubah kondisi nyata tersebut ke dalam bentuk analog atau digital.

Sensor adalah alat yang merespon keadaan fisik, seperti energi panas, energi elektromagnetik, tekanan, magnetik atau pergerakan dengan menghasilkan sinyal elektrik.

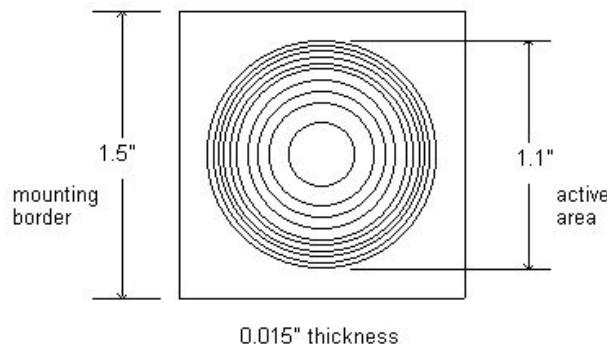
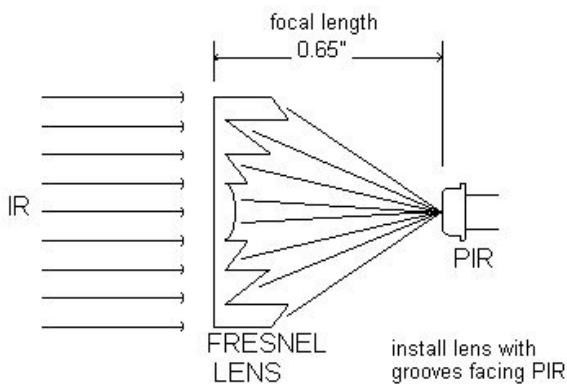
2. Pyroelectric Infrared (PIR) Motion Detector

Sensor ini terbuat dari bahan Crystalline yang dapat membangkitkan sinyal elektrik ketika terdapat energi panas pada radiasi inframerah, energi panas tersebut dapat berasal dari panas tubuh manusia dan hewan dengan sinyal gelombang yang panjangnya dari 9.4 mm (<http://www.glolab.com/-pirparts/infrared.html>).



Gambar 11.1 PIR

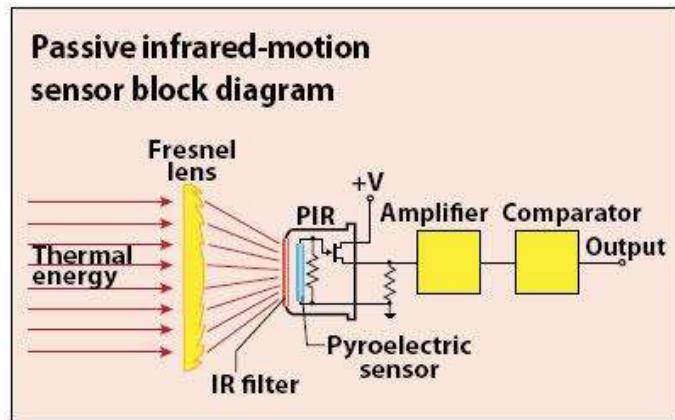
Untuk membantu kinerja dari sensor ini diperlukan Fresnel Lens yang dimana fungsi dari lensa tersebut adalah untuk mempertajam jarak fokus dari sensor. Jika tanpa lensa, jarak maksimum dari deteksi sensor hanya dapat mencapai beberapa centimeter saja, akan tetapi jika dipasang dengan lensa maka jarak maksimum dari deteksinya adalah 5 meter pada sudut 0 derajat (www.digi-ware.com/).



Gambar 11.2 Fresnel Lens

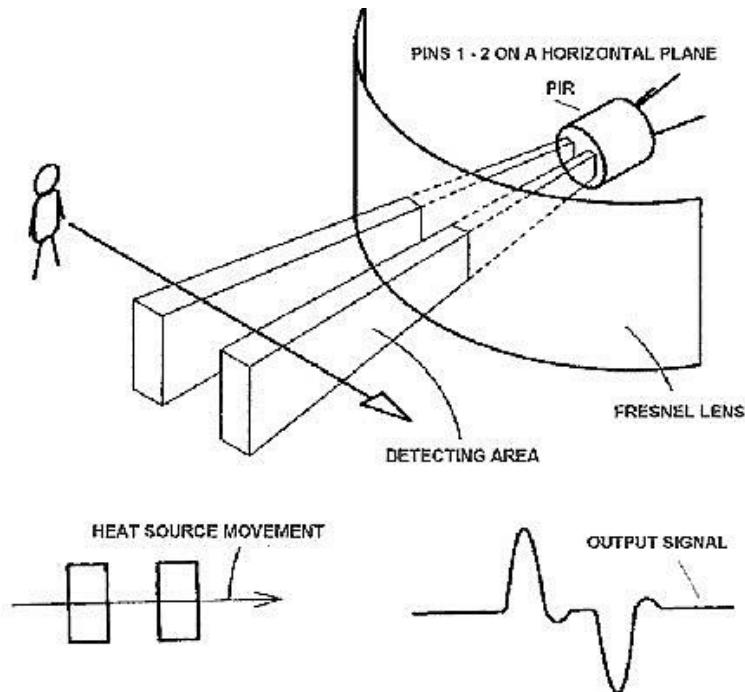
Didalam Sensor Pyroelectric memiliki 2 buah elemen yang dapat mendeteksi pergerakan dari arah kiri atau kanan. Jika sumber panas berasal dari kanan ke kiri maka elemen yang kanan mendeteksi terlebih dahulu dan sinyal keluaran yang dihasilkan adalah sinyal plus terlebih dahulu dan dilanjutkan dengan sinyal minus namun ketika elemen kiri mendeteksi adanya pergerakan terlebih dahulu maka sinyal yang keluar adalah minus terlebih dahulu dan dilanjutkan dengan sinyal plus. Pendekripsi pergerakan ini dapat digunakan sebagai alat yang mendeteksi orang yang masuk atau keluar dari suatu

gedung ataupun pada beberapa aplikasi robotic. Gambaran umum dari rangkaian sensor pyroelectric dapat dilihat pada dibawah ini:



Gambar 11.3 Rangkaian Sensor Pyroelectric

Gambaran umum dari cara kerja sensor pyroelectric:



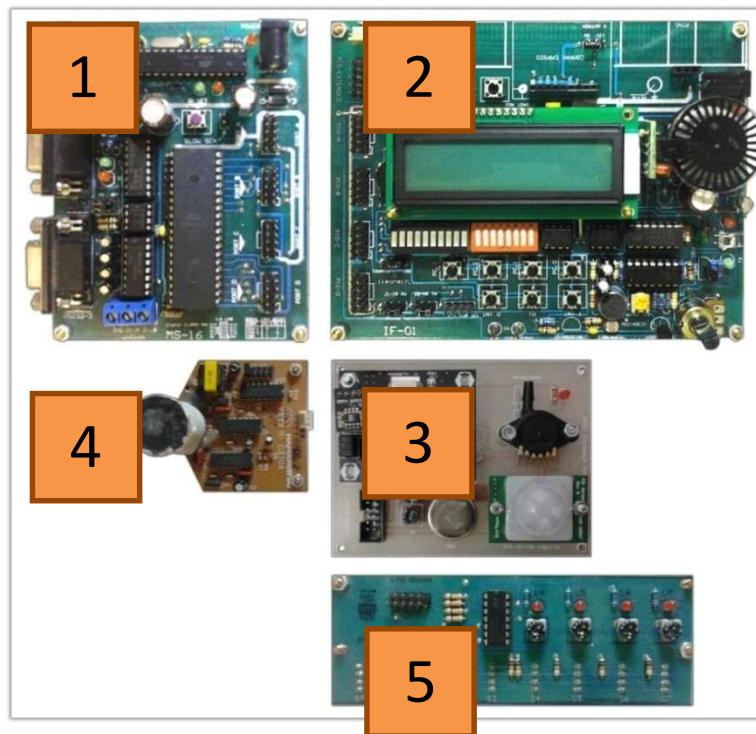
Gambar 11.4 Cara Kerja Sensor Pyroelectric

11.3. Peralatan

- 1 set PC yang dilengkapi dengan software CodeVision AVR.
- 1 set development board AVR ATmega16
- 1 power-supply +9VDC

11.4. Modul I/O

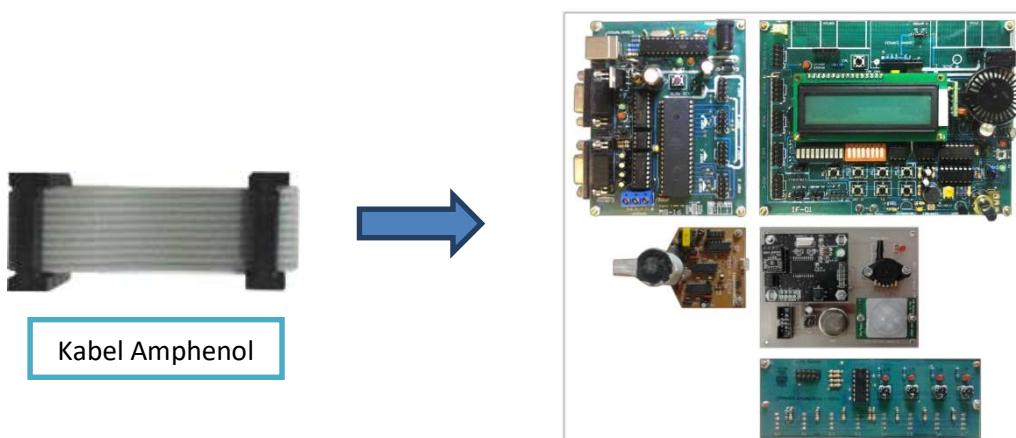
Berikut ini merupakan development board AVR ATmega16:



Gambar 11.5 Development board

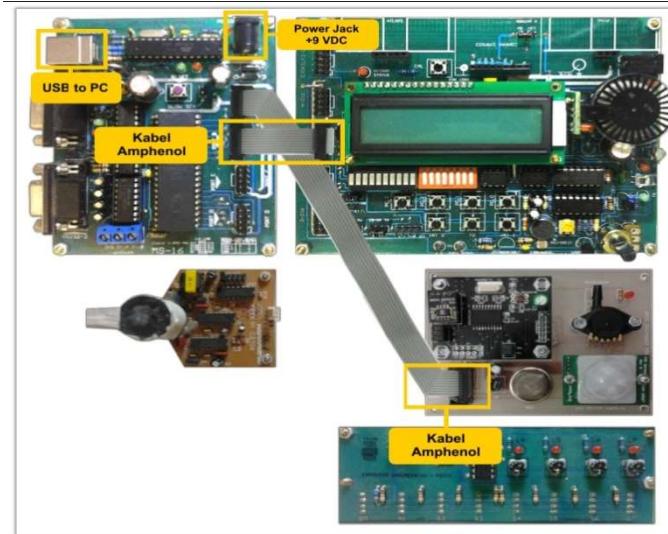
- **Konfigurasi modul pada percobaan ini :**

Pasangkan kabel amphenol pada modul development untuk menghubungkan sistem minimum ATMega16 dengan Training Board. Pasangkan sensor ultrasonic ping parallax pada training board.



Gambar 11.6 Komponen Training board

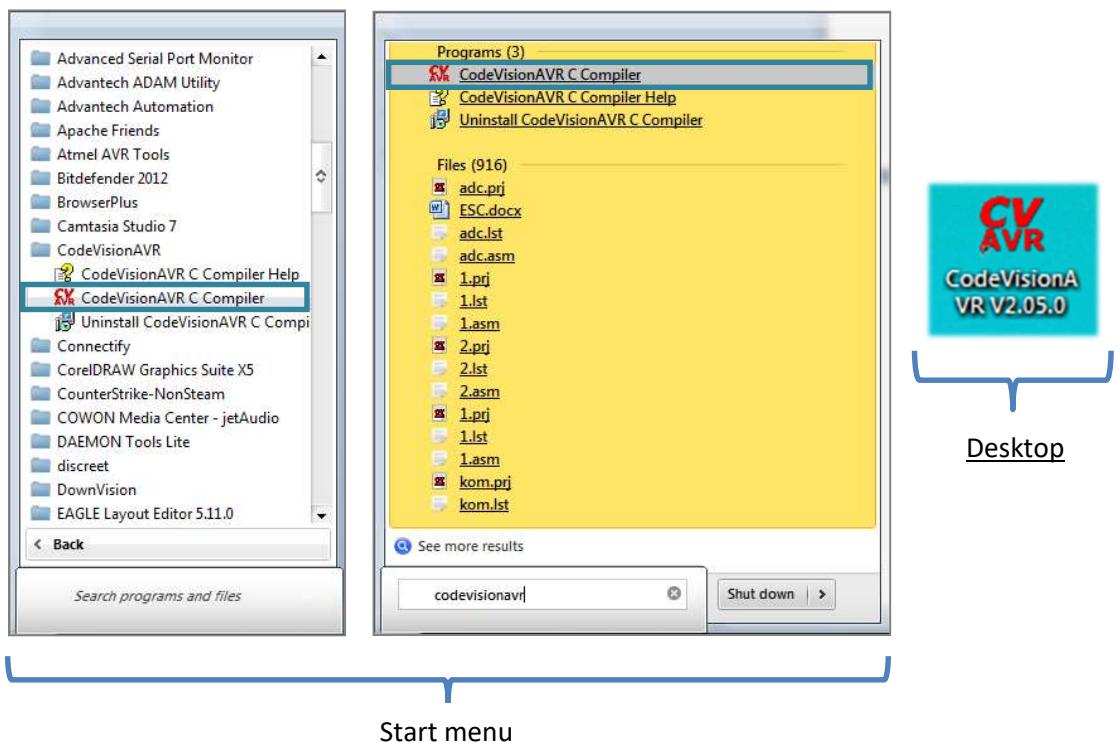
Sehingga menjadi seperti ditunjukkan pada gambar berikut:



Gambar 11.7 Komponen Training board

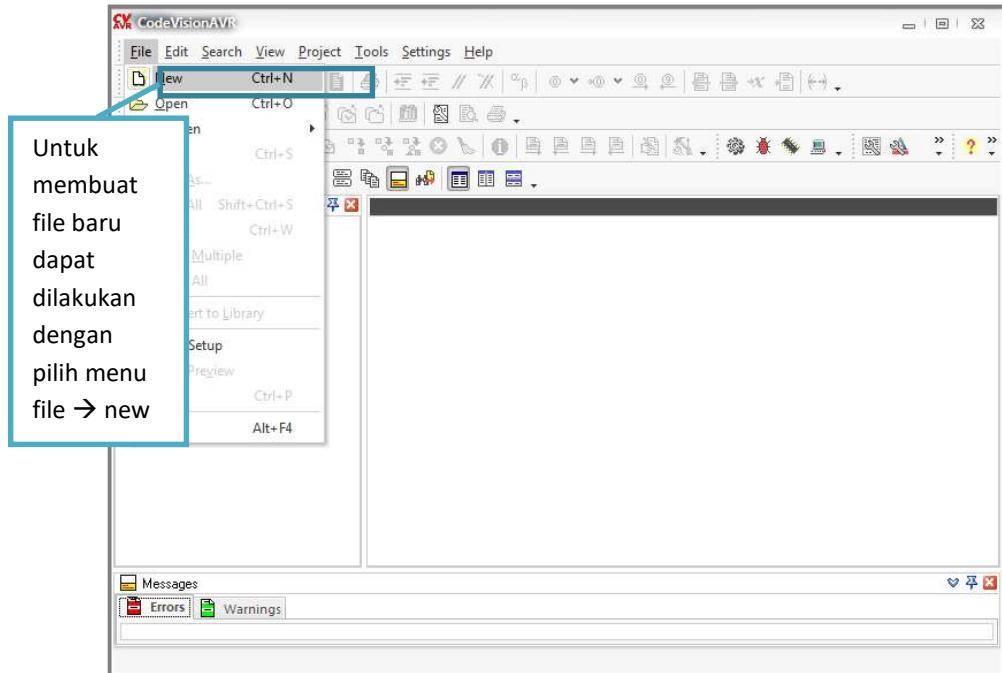
11.5. Prosedur Praktikum

1. Jalankan aplikasi CodeVisionAVR dengan cara melakukan klik ganda pada shortcut icon CodeVisionAVR pada desktop atau dengan cara mencari shortcut program pada start menu system operasi windows.



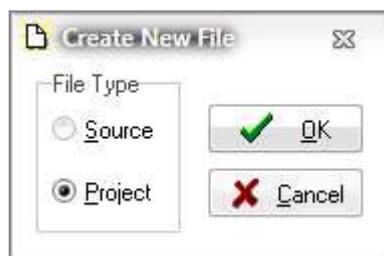
Gambar 11.8 Shortcut CodeVisionAVR

Untuk memulai membuat project baru, pada menubar, pilih File → New, seperti yang ditunjukkan oleh gambar berikut:



Gambar 11.9 Membuat File baru

Setelah itu akan muncul suatu dialog yang mengharuskan *user* untuk memilih antara pembuatan source file atau project. Dalam setiap percobaan pada praktikum ini, dibuat project baru untuk setiap percobaan, oleh karena itu *user* sebaiknya melakukan *check* pada checkbox project dan dilanjutkan dengan menekan tombol “OK”



Gambar 11.10 Membuat project baru

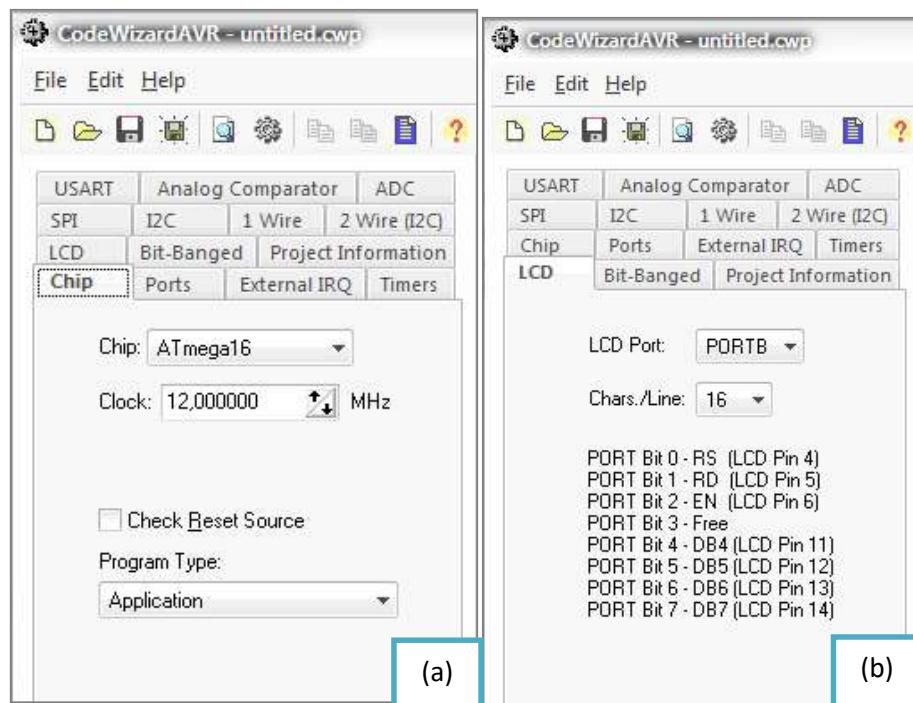
Berikutnya akan keluar dialog yang menanyakan Anda apakah akan membuat project baru. Klik tombol “Yes” untuk menyetujui.



Gambar 11.11 Memilih untuk membuat project baru pada CodeVisionAVR

Setelah itu lakukan set pada CodeWizardAVR. Untuk chip diset dengan nama chip ATmega16 dengan clock 12 MHz.

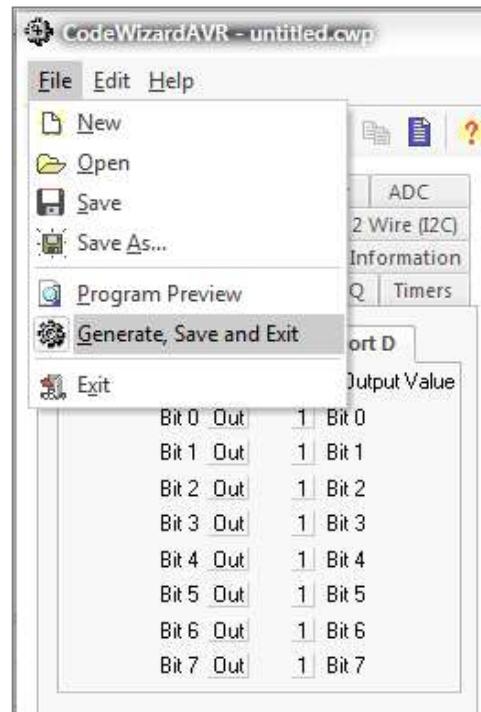
Berikutnya Anda akan menginisialisasi LCD pada PortB, yaitu dengan cara masuk pada tab LCD dan pilih item **PORTB** pada selectbox “LCD Port”. Set banyaknya karakter dari LCD yang diinginkan. Dalam hal ini menggunakan 16 karakter, oleh karena itu pilih item **16** pada selectbox “Chars/Line”



Gambar 11. 12(a) CodeWizardAVR pada tab Chip

(b) Setting LCD pada PortB

Karena pada contoh ini tidak digunakan fasilitas lain maka setting CodeWizardAVR siap disimpan dalam file. Pada menu CodeWizardAVR, pilih File → Generate, Save and Exit, seperti yang ditunjukkan pada gambar berikut:



Gambar 11.13 Menyimpan setting

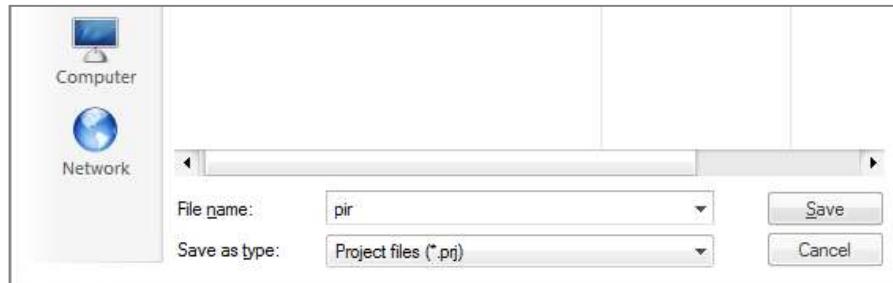
Setelah penekanan menu “Generate, Save and Exit”, akan muncul suatu dialog untuk melakukan set terhadap nama dan lokasi penyimpanan dari source file (*.c) seperti ditunjukkan pada gambar berikut:



Gambar 11.14 Memberi nama pada file C (*.c)

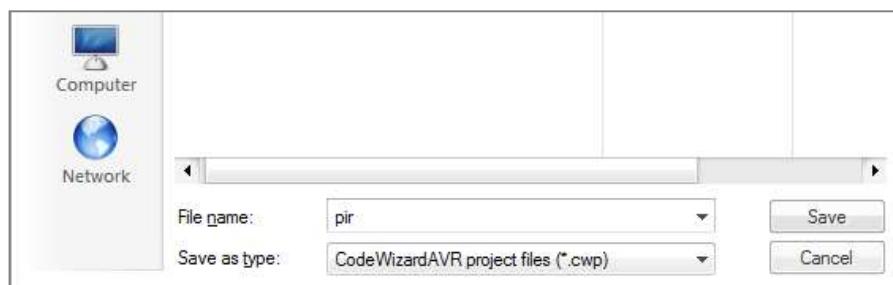
Tekan tombol “Save” untuk menyetujui dan melanjutkan pada proses berikutnya. Setelah penekanan tombol “Save” akan muncul kembali dialog yang kedua, yaitu

untuk melakukan set terhadap nama dan lokasi penyimpanan project file (*.prj) seperti ditunjukkan pada gambar berikut:



Gambar 11.15 Memberi nama project (*.prj)

Tekan tombol “Save” untuk menyetujui dan melanjutkan pada proses berikutnya. Setelah penekanan tombol “Save” akan muncul kembali dialog yang ketiga, yaitu untuk melakukan set terhadap nama dan lokasi penyimpanan CodeWizardProject file (*.cwp) seperti ditunjukkan pada gambar berikut:



Gambar 11.16 Memberi nama CodeWizardProject (*.cwp)

Tekan tombol “Save” untuk menyetujui dan melanjutkan. Selanjutnya akan muncul IDE editor program dari CodeVisionAVR. Anda siap untuk melakukan pembuatan program menggunakan CodeVisionAVR.

2. Pada bagian while(1) {...} tambahkan perintah yang telah diblok dengan warna kuning berikut :

```
while (1)
{
// Place your code here
lcd_gotoxy(0,0);
if(PINA.6)
{
    lcd_putsf("Obyek Terdeteksi");
}
```

```

else
{
    lcd_putsf("Obyek Tdk Ada    ");
}
};

```

3. Untuk lebih jelasnya, berikut kode program secara keseluruhan:

```

#include<mega16.h>

// Alphanumeric LCD Module functions
#asm
    .equ __lcd_port=0x18 ;PORTB
#endasm
#include<lcd.h>

// Declare your global variables here

void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTA=0x00;
DDRA=0x00;
// Port B initialization
PORTB=0x00;
DDRB=0x00;
// Port C initialization
PORTC=0x00;
DDRC=0x00;
// Port D initialization
PORTD=0x00;
DDRD=0x00;
// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge

```

```

// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

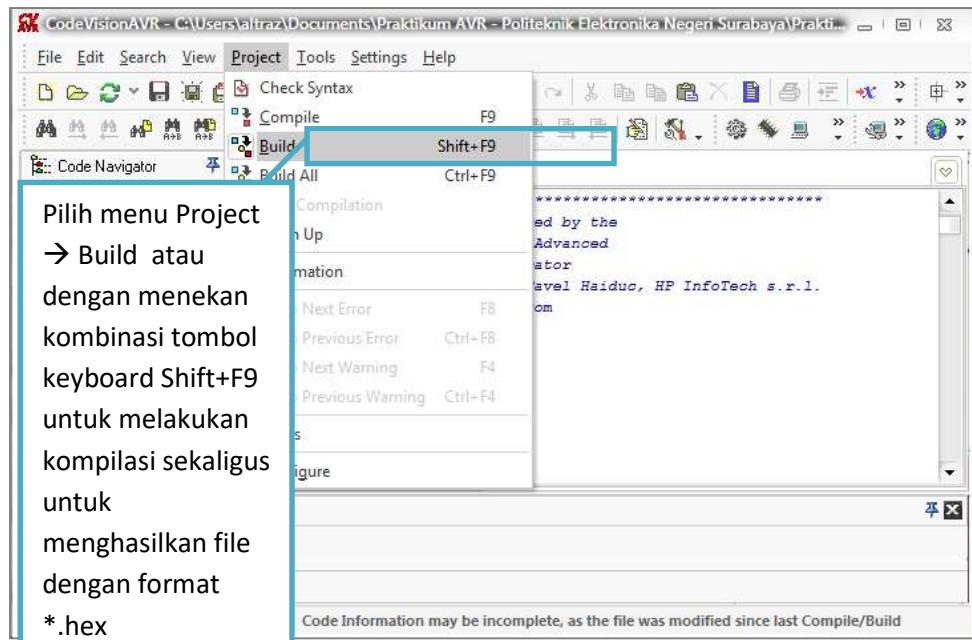
// LCD module initialization
lcd_init(16);

while (1)
{
// Place your code here
    lcd_gotoxy(0,0);
if(PINA.6)
{
    lcd_putsf("Obyek Terdeteksi");
}
else
{
    lcd_putsf("Obyek Tdk Ada    ");
}
};

}

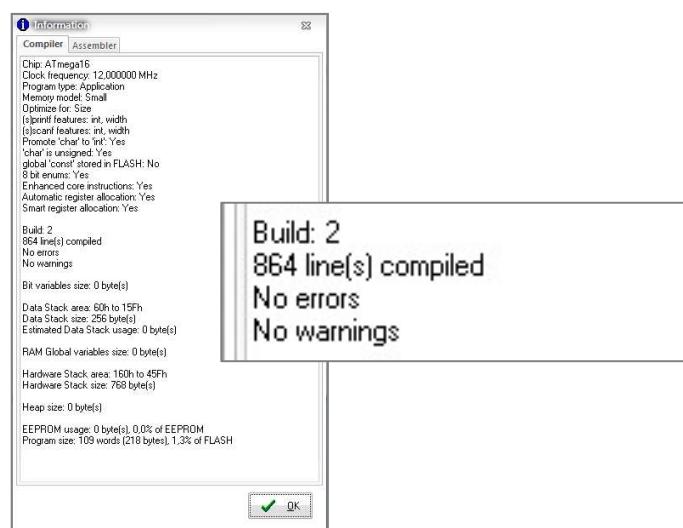
```

4. Setelah pembuatan program selesai, maka proses selanjutnya yaitu lakukan kompilasi terhadap kode program tersebut sekaligus untuk menghasilkan file dengan format hexadecimal (*.hex). Hal ini dapat dilakukan dengan memilih menu Project → Build (Shift+F9) seperti ditunjukkan pada gambar berikut :



Gambar 11.17 Build Source

Apabila kode program sudah benar (tanpa error) dan tidak menyalahi struktur pemrograman bahasa C, maka akan muncul dialog informasi seperti ditunjukkan pada gambar berikut:



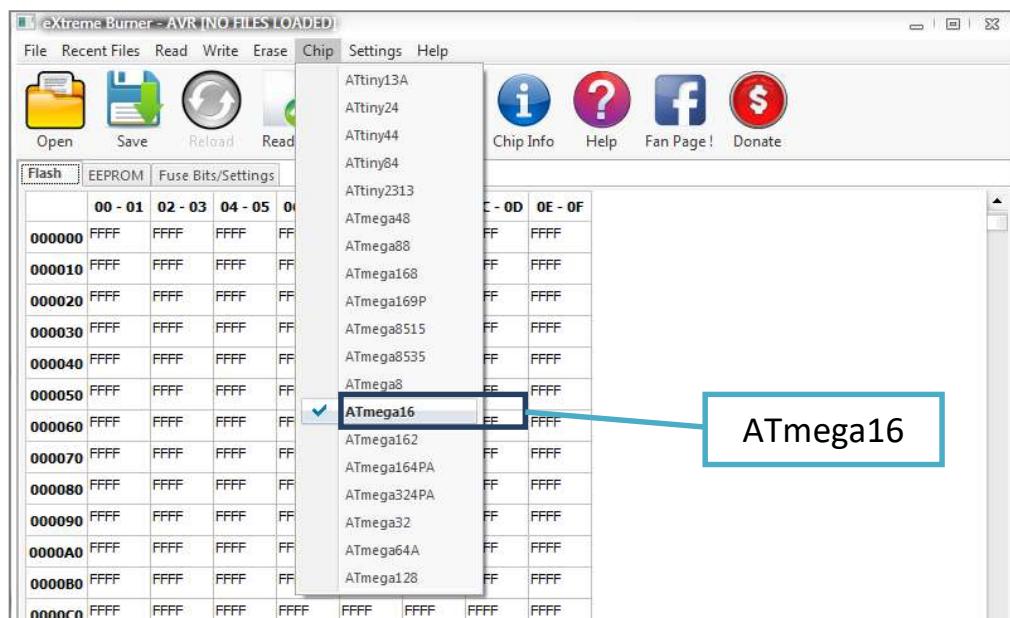
Gambar 11.18 Dialog informasi status kompilasi

5. Setelah diperoleh file dengan format *.hex, maka lakukan download file *.hex ke dalam mikrokontroler. Pertama, jalankan program eXtreme Burner - AVR dengan cara melakukan klik ganda pada shortcut icon eXtreme Burner – AVR pada desktop



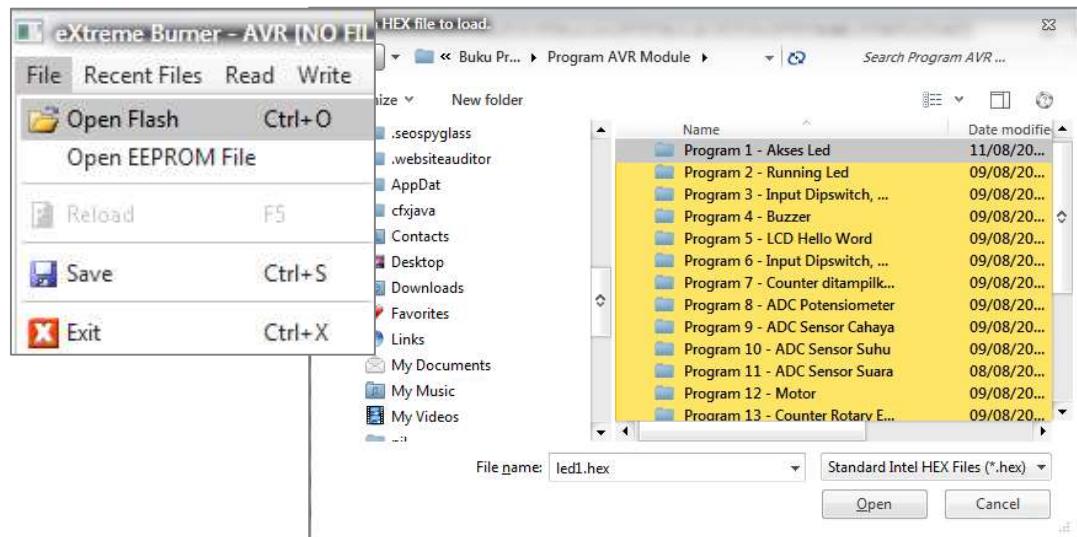
Gambar 11.19 Shortcut eXtreme Burner - AVR

Selanjutnya akan muncul tampilan utama dari program eXtreme Burner – AVR. Pada praktikum ini menggunakan mikrokontroler keluarga AVR dengan tipe IC ATMega16. Oleh karena itu, lakukan pengaturan terhadap tipe IC terlebih dahulu dengan memilih menu Chip → ATmega16 seperti ditunjukkan pada gambar berikut:



Gambar 11.20 Tipe Mikrokontroler

Setelah itu, lakukan open file *.hex dengan memilih menu File → Open Flash (Ctrl+O) seperti ditunjukkan pada gambar berikut:



Gambar 11.21 Open Flash

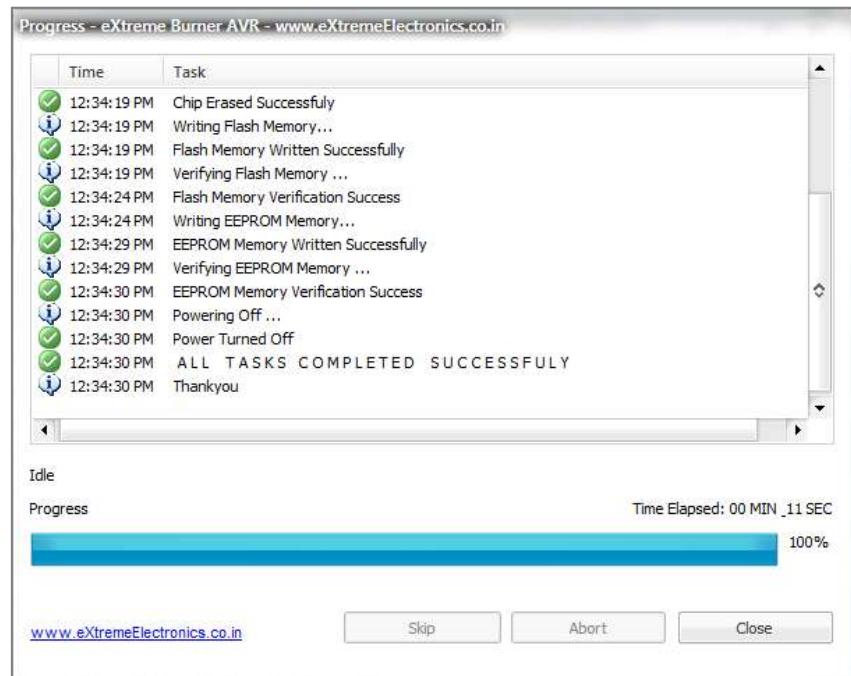
Lakukan pencarian terhadap lokasi file *.hex Anda, untuk selanjutnya tekan tombol “Open” apabila sudah ditemukan file yang dimaksud. Setelah itu akan muncul pesan yang memberitahukan bahwa file *hex berhasil di-load program eXtreme Burner – AVR.



Gambar 11.22 Message



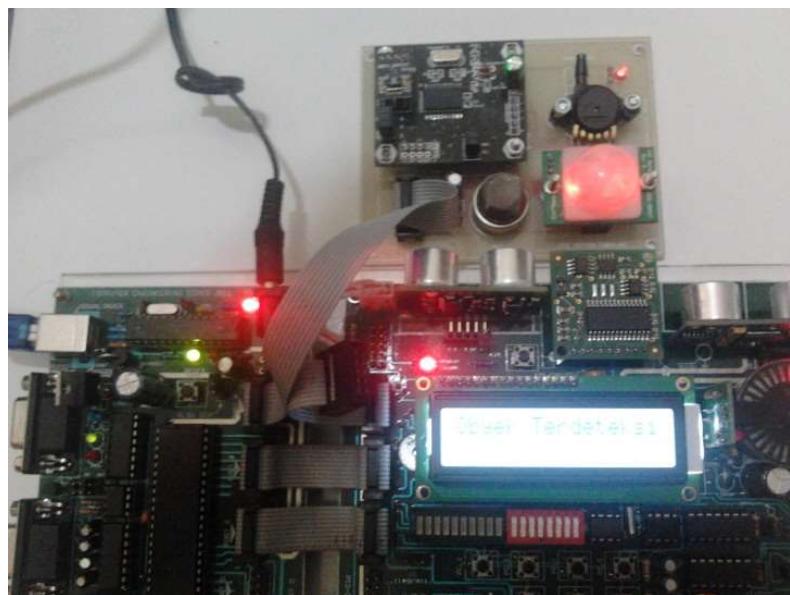
Selanjutnya tekan tombol  untuk memulai download ke dalam mikrokontroler Anda. Berikut ini merupakan tampilan apabila proses download ke dalam mikrokontroler berhasil dilakukan:



Gambar 11.23 Proses Berhasil

6. Selesai

11.6. Hasil Percobaan



Gambar 11.24 Dekatkan suatu obyek diatas PIR



Gambar 11.25 Respon Program

PERCOBAAN 12

SENSOR KELEMBABAN : SHT10

PERCOBAAN 12

SENSOR KELEMBABAN : SHT10

12.1. Tujuan

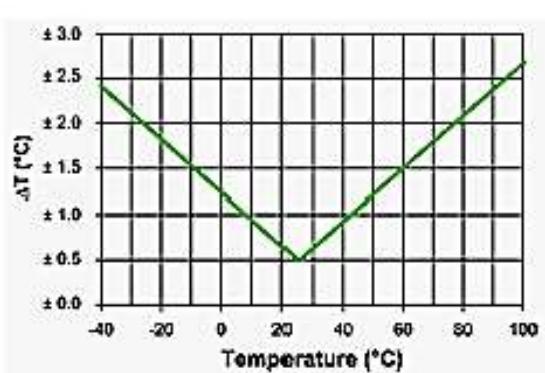
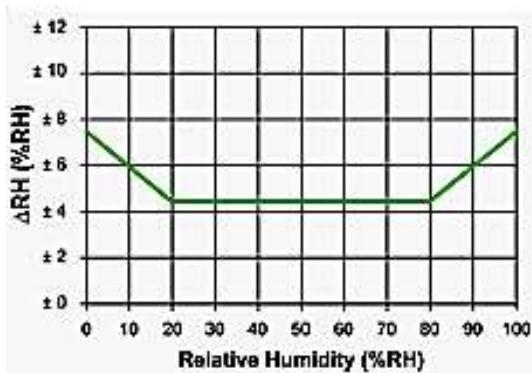
1. Mahasiswa dapat mengetahui cara kerja sensor kelembaban SHT10
2. Mahasiswa dapat mengakses sensor SHT10 pada mikrokontroler ATMega 16
3. Mahasiswa dapat membuat program sederhana untuk mengimplementasikan sensor kelembaman SHT10

12.2. Dasar Teori

- **Sensor SHT10**

SHT10 merupakan sebuah chip/sensor digital yang dapat mengukur suhu dan kelembaban relatif dengan biaya yang terjangkau (lebih rendah dari harga sensor SHT yang lainnya). Adapun spesifikasi SHT10 adalah sebagai berikut:

- Konsumsi energi : 80uW
- RH Jarak operasi : 0 - 100% RH
- T Jarak operasi : -40 - 125 ° C (-40 - 257 ° F)
- Output : digital
- Akurasi maksimal batas RH dan suhu:



Gambar 12.1(a) Akurasi maksimal RH SHT10

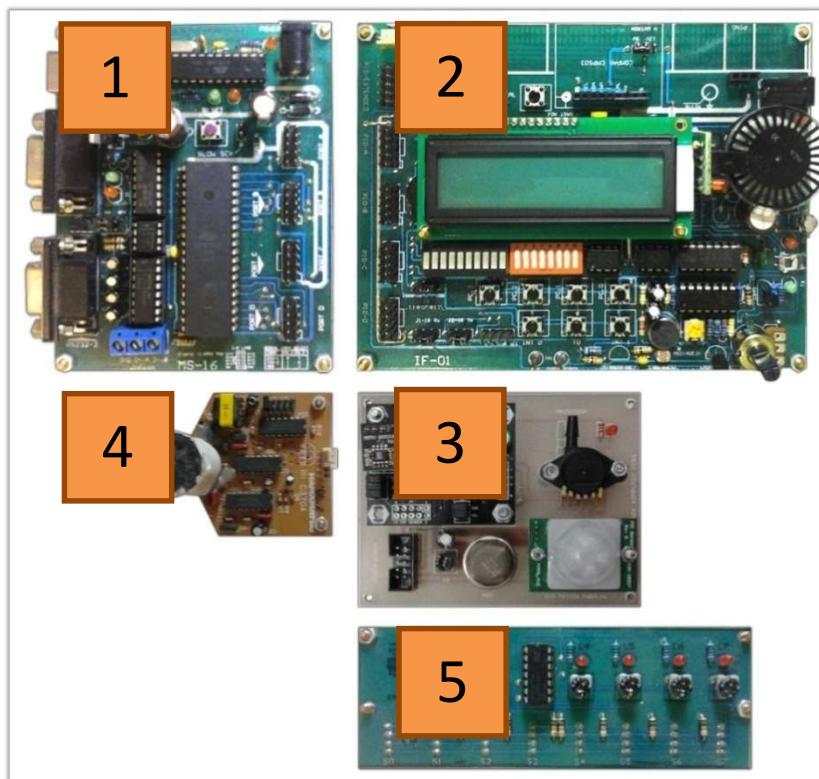
(b) Akurasi maksimal suhu SHT10

12.3. Peralatan

- 1 set PC yang dilengkapi dengan software CodeVision AVR.
- 1 set development board AVR ATmega16
- 1 power-supply +9VDC

12.4. Modul I/O

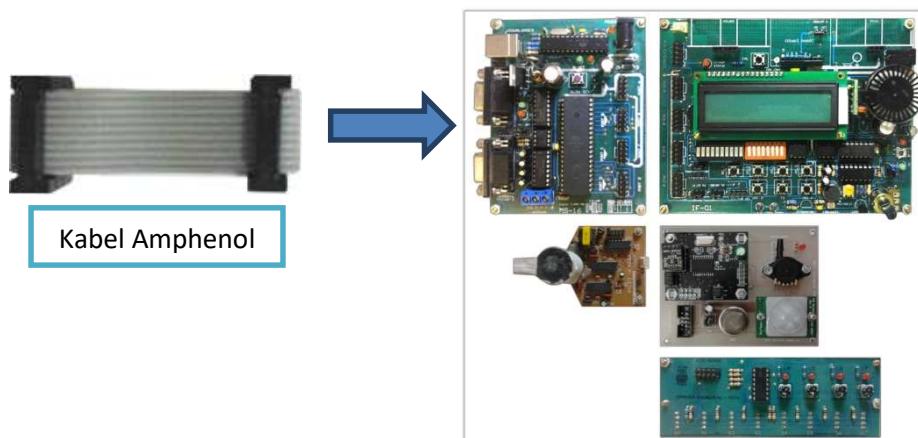
Berikut ini merupakan development board AVR ATmega16:



Gambar 12.2 Development board

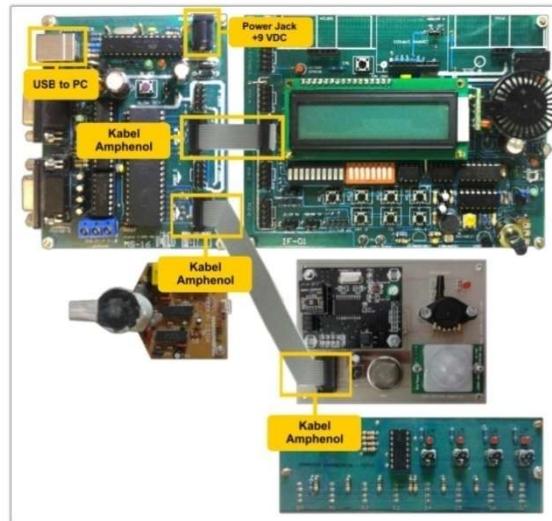
- **Konfigurasi modul pada percobaan ini :**

Pasangkan kabel amphenol pada modul development untuk menghubungkan sistem minimum ATMega16 dengan Training Board. Pasangkan sensor ultrasonic ping parallax pada training board.



Gambar 12.3 Komponen Training board

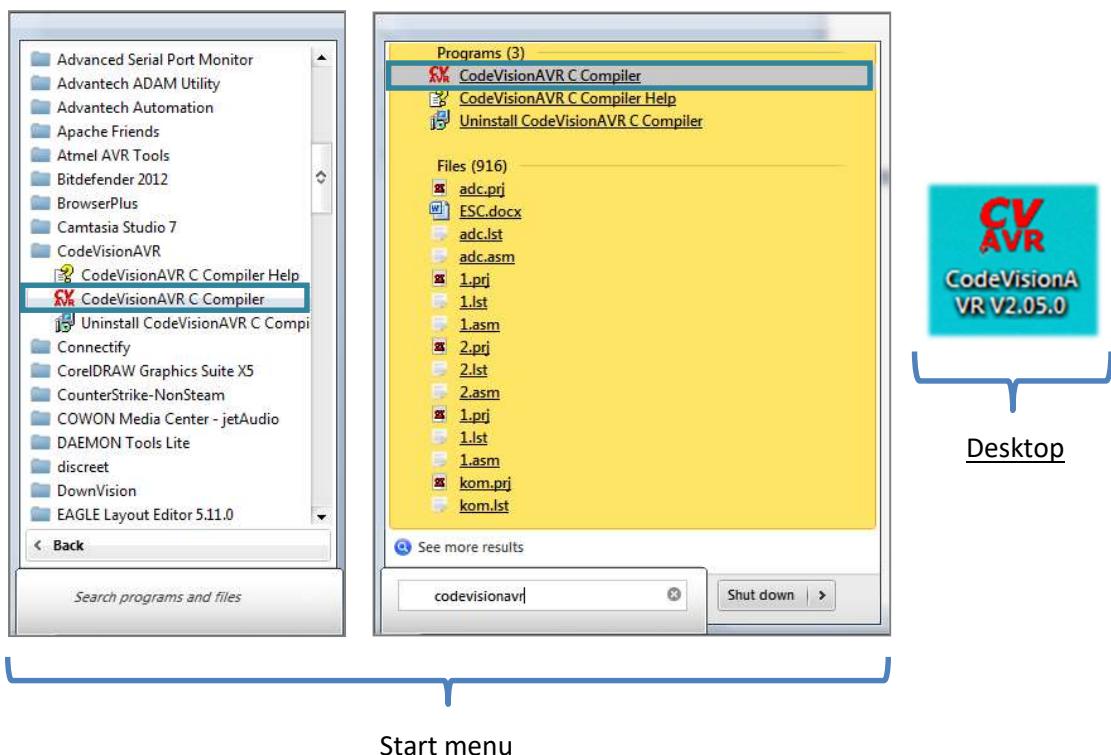
Sehingga menjadi seperti ditunjukkan pada gambar berikut:



Gambar 12.4 Komponen Training board

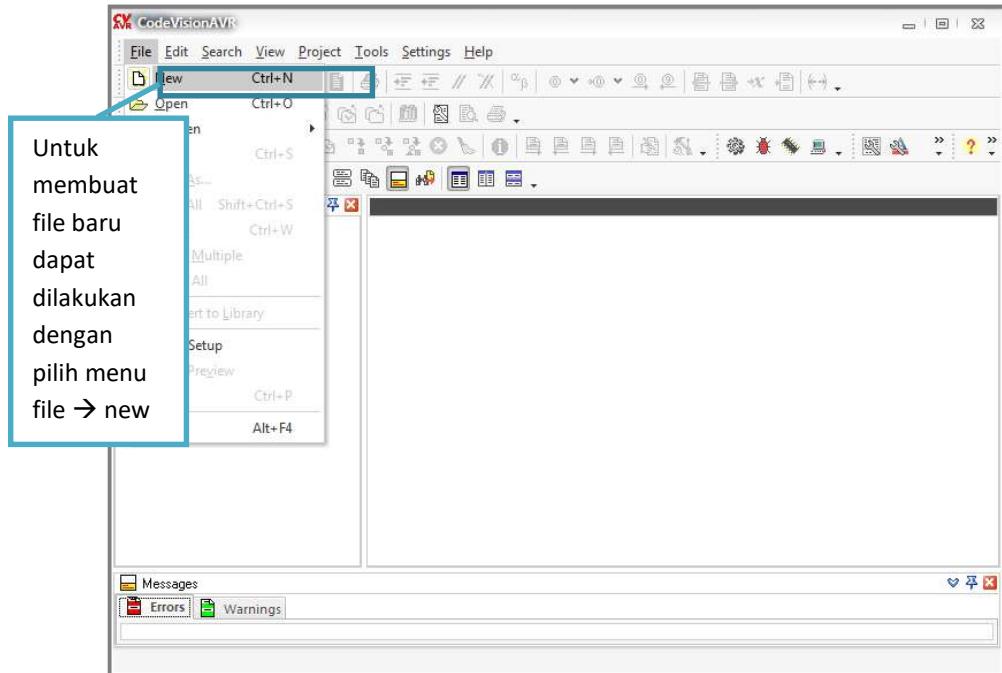
12.5. Prosedur Praktikum

1. Jalankan aplikasi CodeVisionAVR dengan cara melakukan klik ganda pada shortcut icon CodeVisionAVR pada desktop atau dengan cara mencari shortcut program pada start menu system operasi windows.



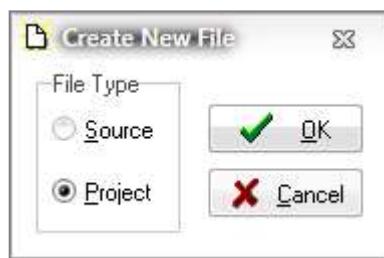
Gambar 12.5 Shortcut CodeVisionAVR

Untuk memulai membuat project baru, pada menubar, pilih File → New, seperti yang ditunjukkan oleh gambar berikut:



Gambar 12.6 Membuat File baru

Setelah itu akan muncul suatu dialog yang mengharuskan *user* untuk memilih antara pembuatan source file atau project. Dalam setiap percobaan pada praktikum ini, dibuat project baru untuk setiap percobaan, oleh karena itu *user* sebaiknya melakukan *check* pada checkbox project dan dilanjutkan dengan menekan tombol “OK”



Gambar 12.7 Membuat project baru

Berikutnya akan keluar dialog yang menanyakan Anda apakah akan membuat project baru. Klik tombol “Yes” untuk menyetujui.

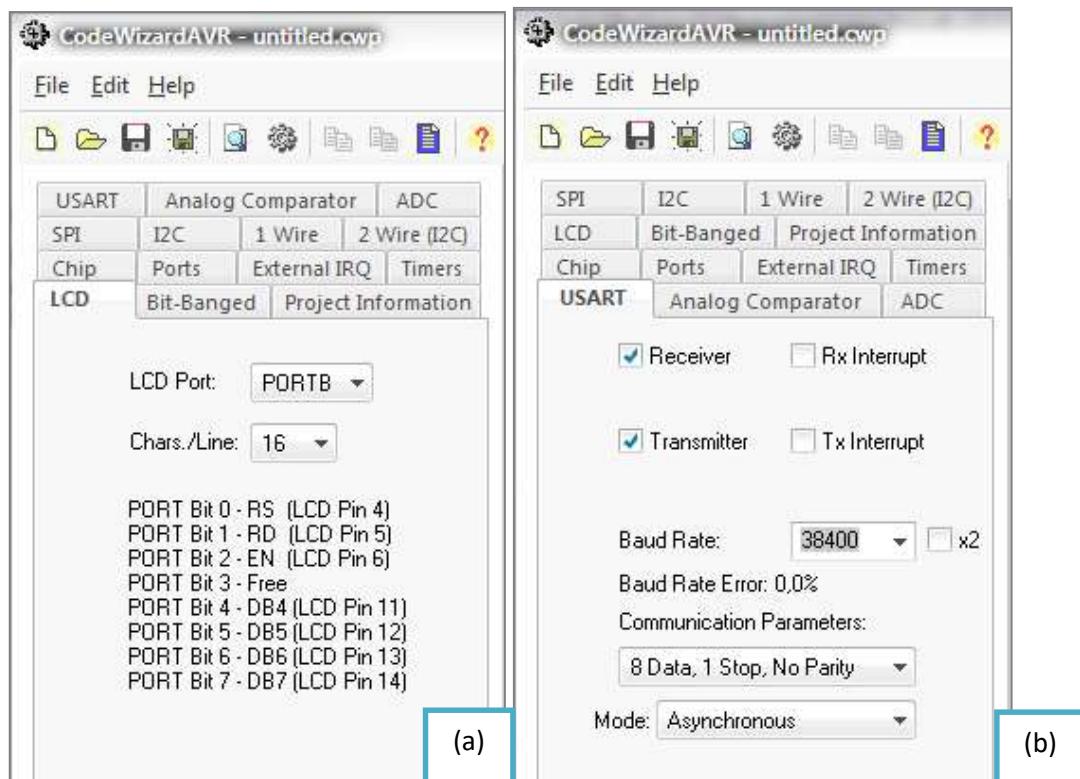


Gambar 12.8 Memilih untuk membuat project baru pada CodeVisionAVR

Setelah itu lakukan set pada CodeWizardAVR. Untuk chip diset dengan nama chip ATmega16 dengan clock 12 MHz.

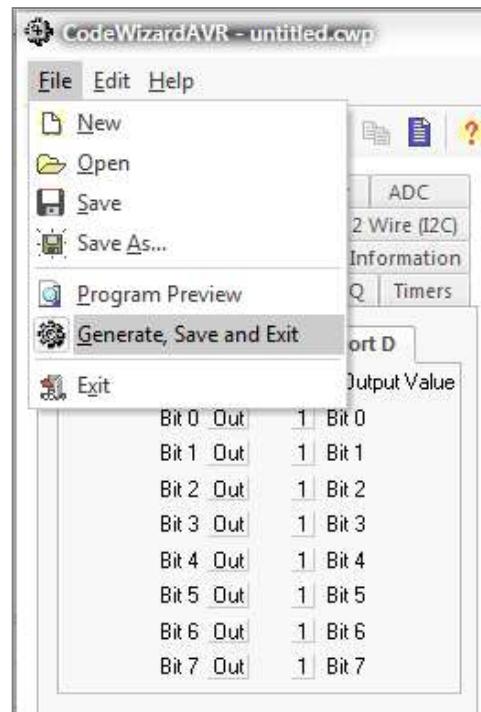
Berikutnya Anda akan menginisialisasi LCD pada PortB, yaitu dengan cara masuk pada tab LCD dan pilih item **PORTE** pada selectbox “LCD Port”. Set banyaknya karakter dari LCD yang diinginkan. Dalam hal ini menggunakan 16 karakter, oleh karena itu pilih item **16** pada selectbox “Chars/Line”.

Setelah itu anda diharuskan untuk melakukan setting USART dengan baudrate = 38400



Gambar 12.9 (a) Setting LCD pada PortB **(b)** Setting USART 8 bit

Karena pada contoh ini tidak digunakan fasilitas lain maka setting CodeWizardAVR siap disimpan dalam file. Pada menu CodeWizardAVR, pilih File → Generate, Save and Exit, seperti yang ditunjukkan pada gambar berikut:



Gambar 12.10 Menyimpan setting

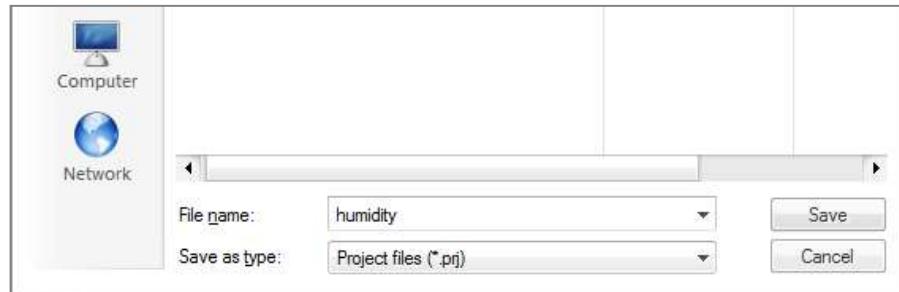
Setelah penekanan menu “Generate, Save and Exit”, akan muncul suatu dialog untuk melakukan set terhadap nama dan lokasi penyimpanan dari source file (*.c) seperti ditunjukkan pada gambar berikut:



Gambar 12.11 Memberi nama pada file C (*.c)

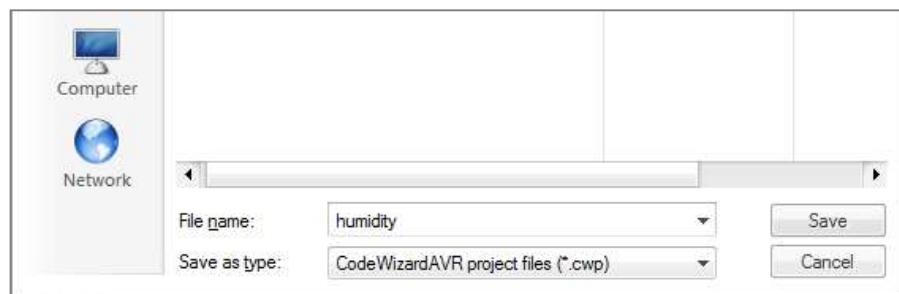
Tekan tombol “Save” untuk menyetujui dan melanjutkan pada proses berikutnya. Setelah penekanan tombol “Save” akan muncul kembali dialog yang kedua, yaitu

untuk melakukan set terhadap nama dan lokasi penyimpanan project file (*.prj) seperti ditunjukkan pada gambar berikut:



Gambar 12.12 Memberi nama project (*.prj)

Tekan tombol “Save” untuk menyetujui dan melanjutkan pada proses berikutnya. Setelah penekanan tombol “Save” akan muncul kembali dialog yang ketiga, yaitu untuk melakukan set terhadap nama dan lokasi penyimpanan CodeWizardProject file (*.cwp) seperti ditunjukkan pada gambar berikut:



Gambar 12.13 Memberi nama CodeWizardProject (*.cwp)

Tekan tombol “Save” untuk menyetujui dan melanjutkan. Selanjutnya akan muncul IDE editor program dari CodeVisionAVR. Anda siap untuk melakukan pembuatan program menggunakan CodeVisionAVR.

2. Pada program-C yang dihasilkan, tambahkan header `delay.h` dan `stdio.h`
3. Pada bagian `while(1) {...}` tambahkan perintah yang telah diblok dengan warna kuning berikut :

```
while (1)
{
// Place your code here
putchar(0x03);
    hum=getchar();

    lcd_clear();
    lcd_gotoxy(0,0);
```

```

        sprintf(xstring,"Hum:%d %RH",hum);
        lcd_puts(xstring);
        delay_ms(100);

    };

```

4. Untuk lebih jelasnya, berikut kode program secara keseluruhan:

```

#include<mega16.h>
#include<delay.h>

// Alphanumeric LCD Module functions
#asm
    .equ __lcd_port=0x18 ;PORTB
#endifasm
#include<lcd.h>

// Standard Input/Output functions
#include<stdio.h>

// Declare your global variables here
unsignedchar hum,temp,xstring[16];
void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTA=0x00;
DDRA=0x00;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTB=0x00;
DDRB=0x00;

// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T

```

```

PORTD=0x00;
DDRD=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous

```

```

// USART Baud Rate: 38400
UCSRA=0x00;
UCSRB=0x18;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x11;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// LCD module initialization
lcd_init(16);

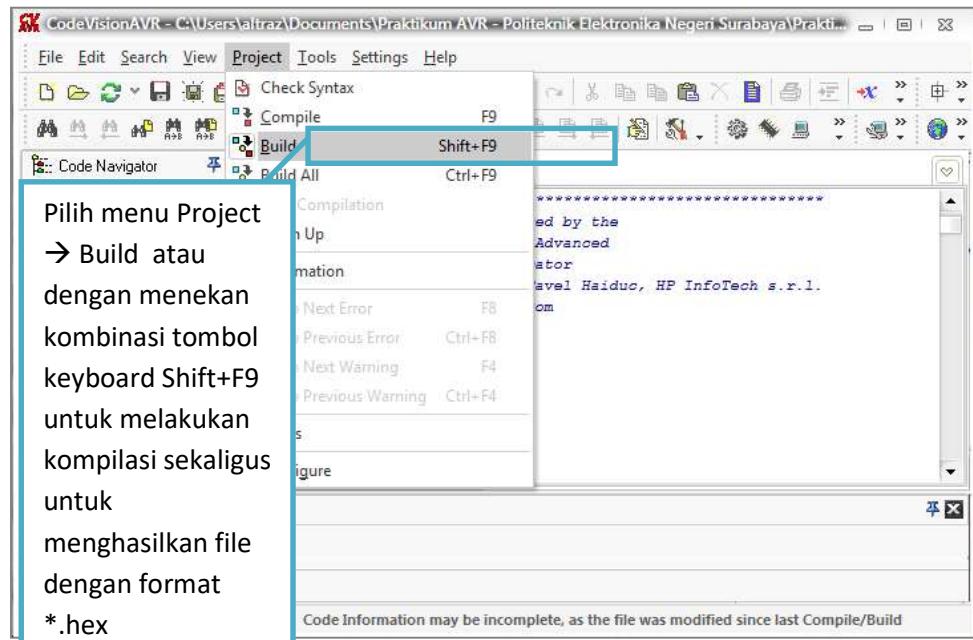
while (1)
{
    // Place your code here
    putchar(0x03);
    hum=getchar();

    lcd_clear();
    lcd_gotoxy(0,0);
    sprintf(xstring,"Hum:%d %RH",hum);
    lcd_puts(xstring);
    delay_ms(100);

}

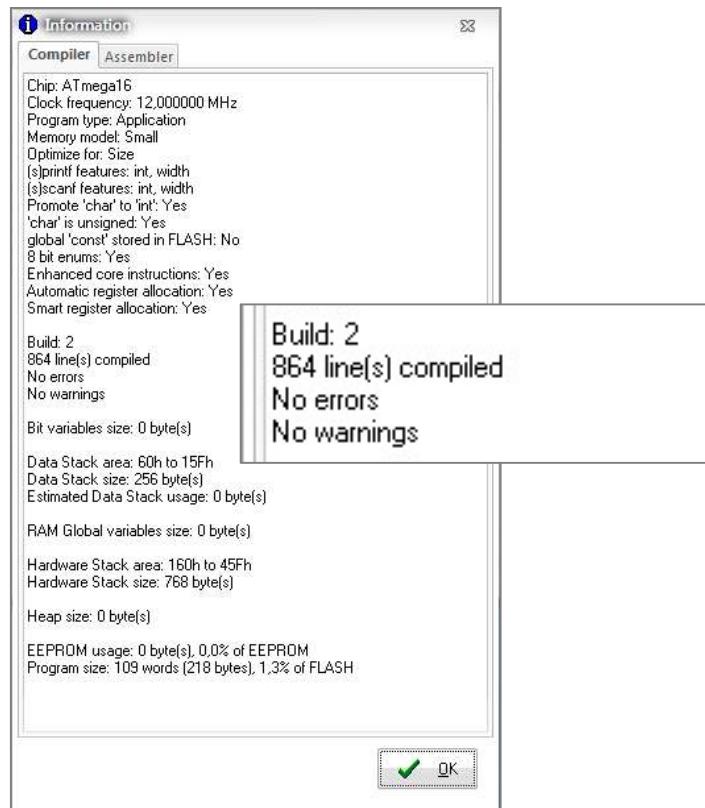
```

5. Setelah pembuatan program selesai, maka proses selanjutnya yaitu lakukan kompilasi terhadap kode program tersebut sekaligus untuk menghasilkan file dengan format hexadecimal (*.hex). Hal ini dapat dilakukan dengan memilih menu Project → Build (Shift+F9) seperti ditunjukkan pada gambar berikut :



Gambar 12.14 Build Source

Apabila kode program sudah benar (tanpa error) dan tidak menyalahi struktur pemrograman bahasa C, maka akan muncul dialog informasi seperti ditunjukkan pada gambar berikut:



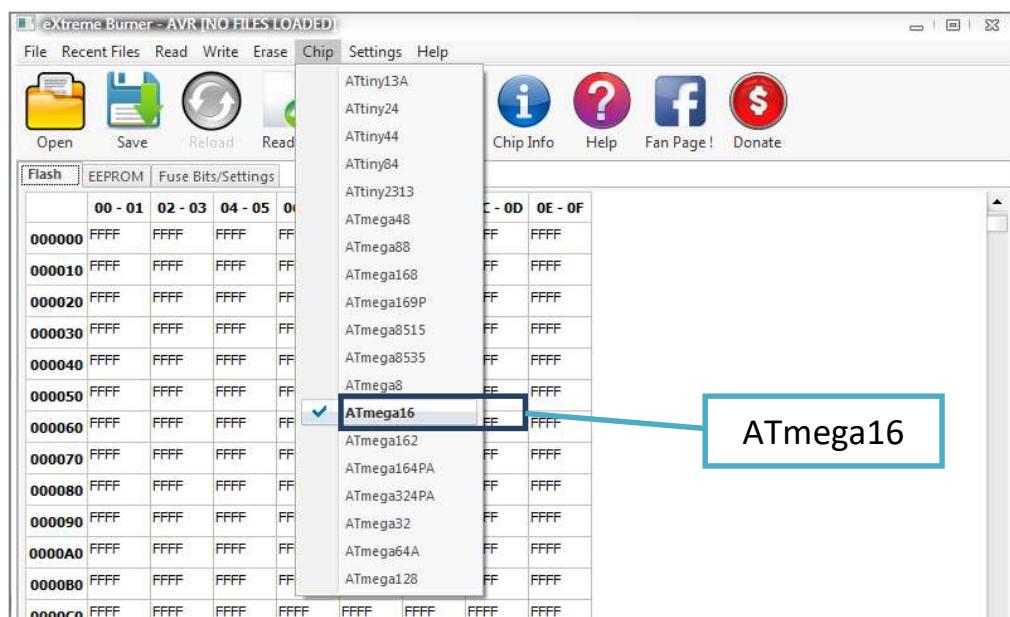
Gambar 12.15 Dialog informasi status kompilasi

6. Setelah diperoleh file dengan format *.hex, maka lakukan download file *.hex ke dalam mikrokontroler. Pertama, jalankan program eXtreme Burner - AVR dengan cara melakukan klik ganda pada shortcut icon eXtreme Burner– AVR pada desktop



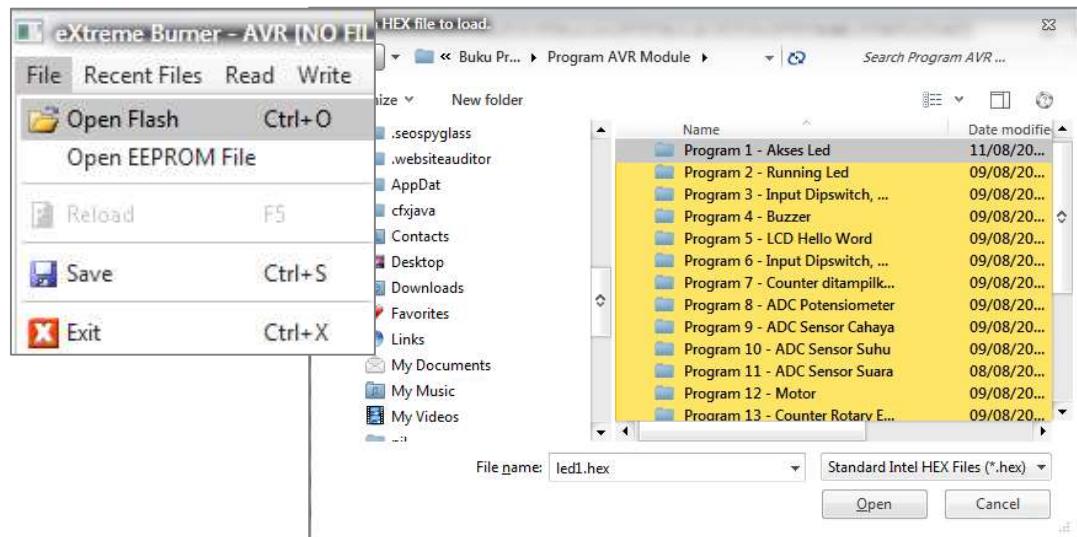
Gambar 12.16 Shortcut eXtreme Burner - AVR

Selanjutnya akan muncul tampilan utama dari program eXtreme Burner – AVR. Pada praktikum ini menggunakan mikrokontroler keluarga AVR dengan tipe IC ATMega16. Oleh karena itu, lakukan pengaturan terhadap tipe IC terlebih dahulu dengan memilih menu Chip → ATmega16 seperti ditunjukkan pada gambar berikut:



Gambar 12.17 Tipe Mikrokontroler

Setelah itu, lakukan open file *.hex dengan memilih menu File → Open Flash (Ctrl+O) seperti ditunjukkan pada gambar berikut:



Gambar 12.18 Open Flash

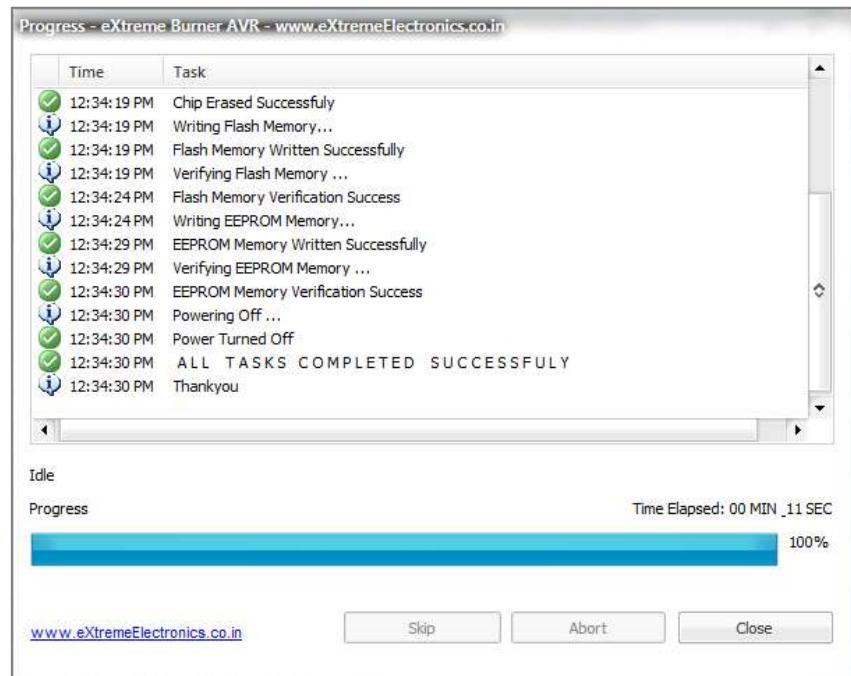
Lakukan pencarian terhadap lokasi file *.hex Anda, untuk selanjutnya tekan tombol “Open” apabila sudah ditemukan file yang dimaksud. Setelah itu akan muncul pesan yang memberitahukan bahwa file *hex berhasil di-load program eXtreme Burner – AVR.



Gambar 12.19 Message



Selanjutnya tekan tombol untuk memulai download ke dalam mikrokontroler Anda. Berikut ini merupakan tampilan apabila proses download ke dalam mikrokontroler berhasil dilakukan:



Gambar 12.20 Proses Berhasil

7. Selesai

12.6. Hasil Percobaan



Gambar 12.21 Running Program

PERCOBAAN 13

SENSOR SUHU : SHT10

PERCOBAAN 13

SENSOR SUHU : SHT10

13.1. Tujuan

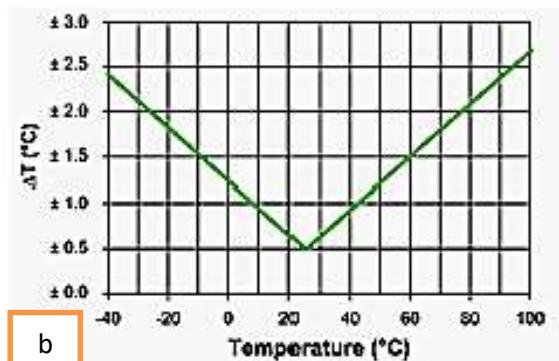
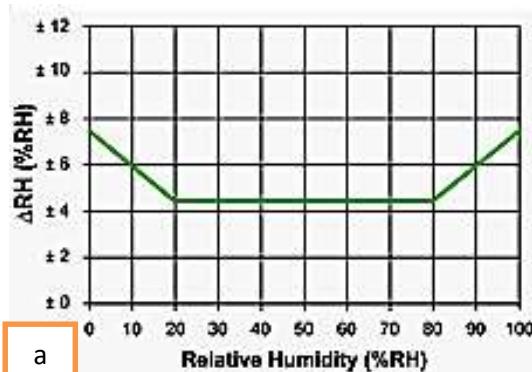
1. Mahasiswa dapat mengakses sensor SHT10 pada mikrokontroler ATMega 16
2. Mahasiswa dapat membuat program sederhana untuk mengimplementasikan sensor suhu SHT10

13.2. Dasar Teori

• Sensor SHT10

SHT10 merupakan sebuah chip/sensor digital yang dapat mengukur suhu dan kelembaban relatif dengan biaya yang terjangkau (lebih rendah dari harga sensor SHT yang lainnya). Adapun spesifikasi SHT10 adalah sebagai berikut:

- Konsumsi energi : 80uW
- RH Jarak operasi : 0 - 100% RH
- T Jarak operasi : -40 - 125 ° C (-40 - 257 ° F)
- Output : digital
- Akurasi maksimal batas RH dan suhu:



Gambar 13.1(a) Akurasi maksimal RH SHT10

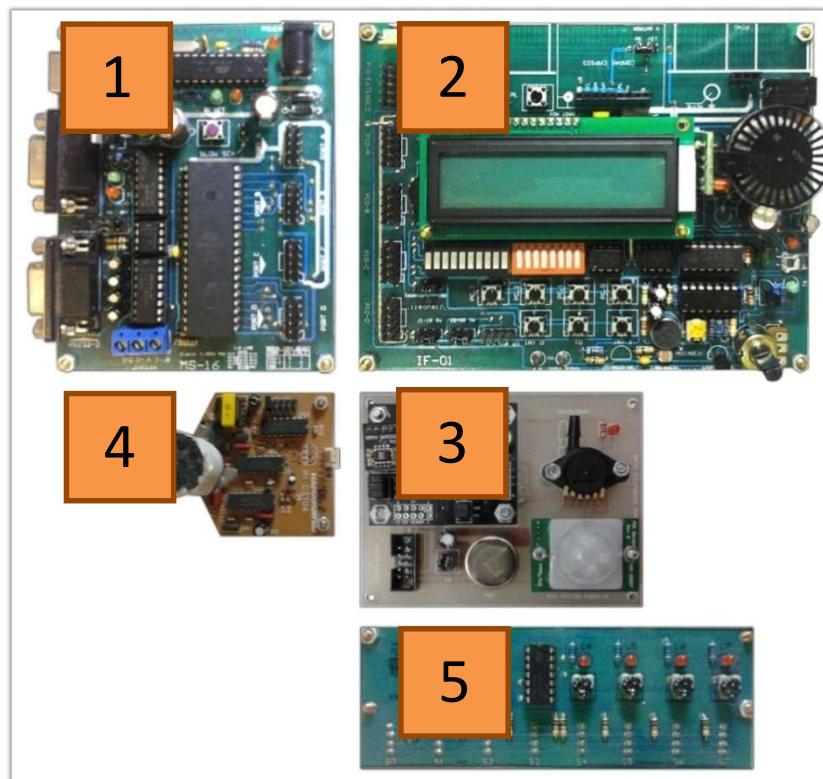
(b) Akurasi maksimal suhu SHT10

13.3. Peralatan

- 1 set PC yang dilengkapi dengan software CodeVision AVR.
- 1 set development board AVR ATmega16
- 1 power-supply +9VDC

13.4. Modul I/O

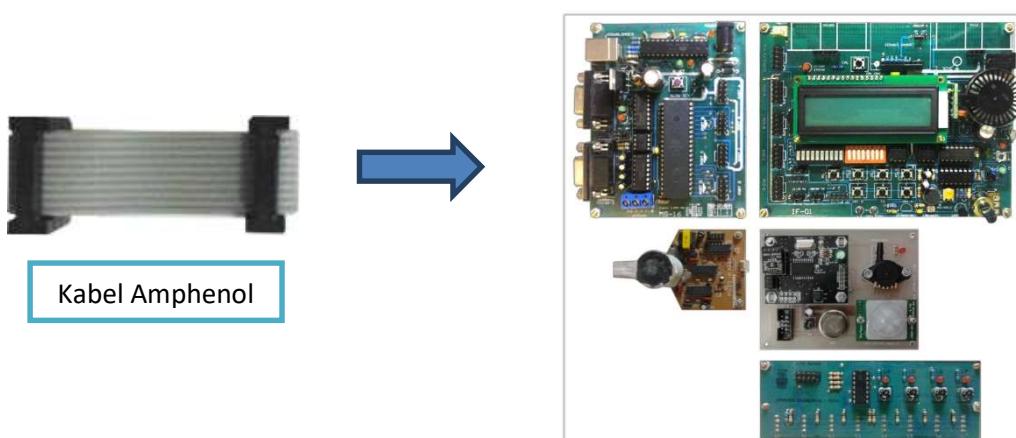
Berikut ini merupakan development board AVR ATmega16:



Gambar 13.2 Development board

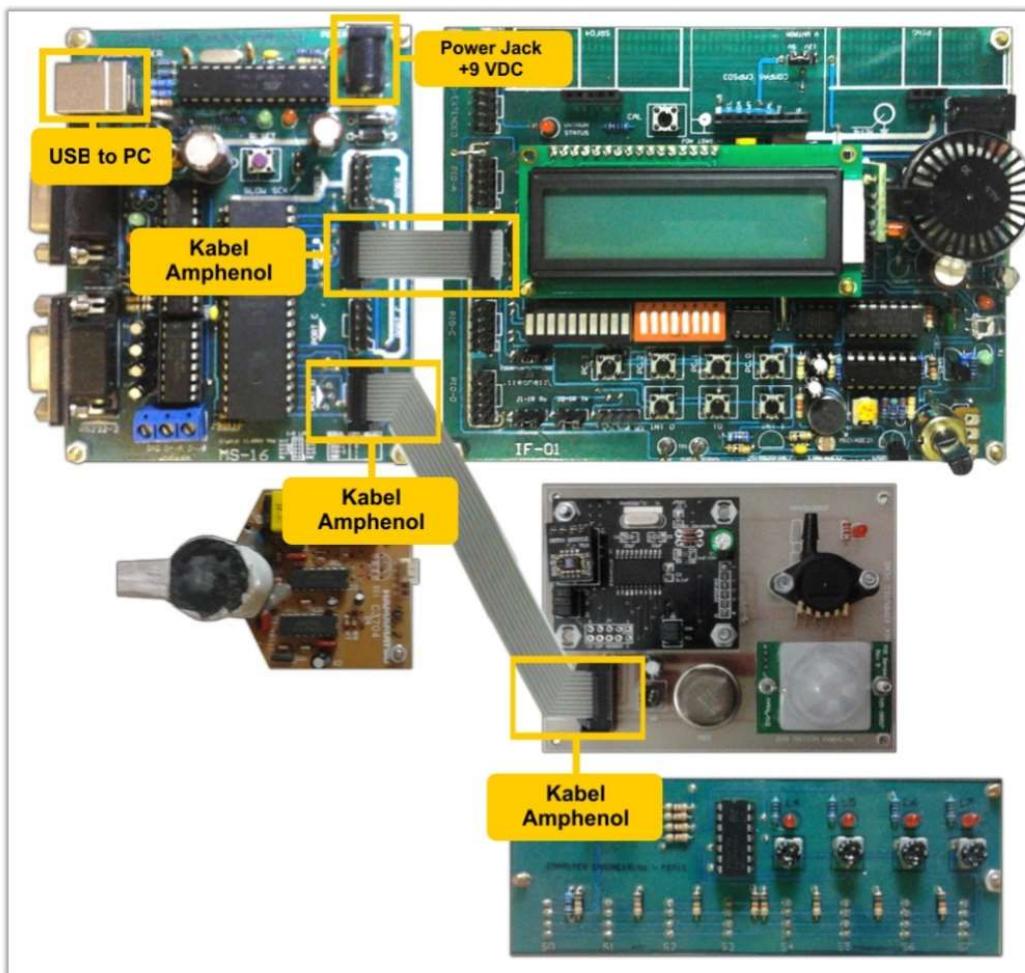
- **Konfigurasi modul pada percobaan ini :**

Pasangkan kabel amphenol pada modul development untuk menghubungkan system minimum ATMega16 dengan Training Board. Pasangkan sensor ultrasonic ping parallax pada training board.



Gambar 13.3 Komponen Training board

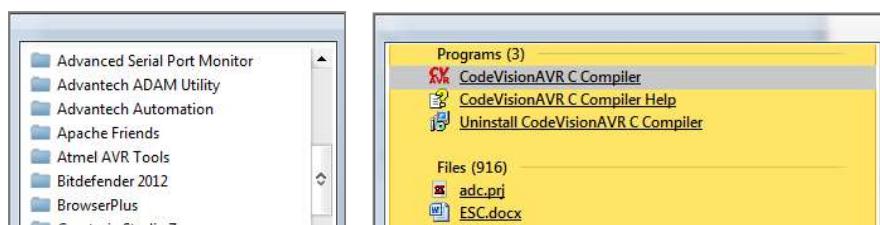
Sehingga menjadi seperti ditunjukkan pada gambar berikut:

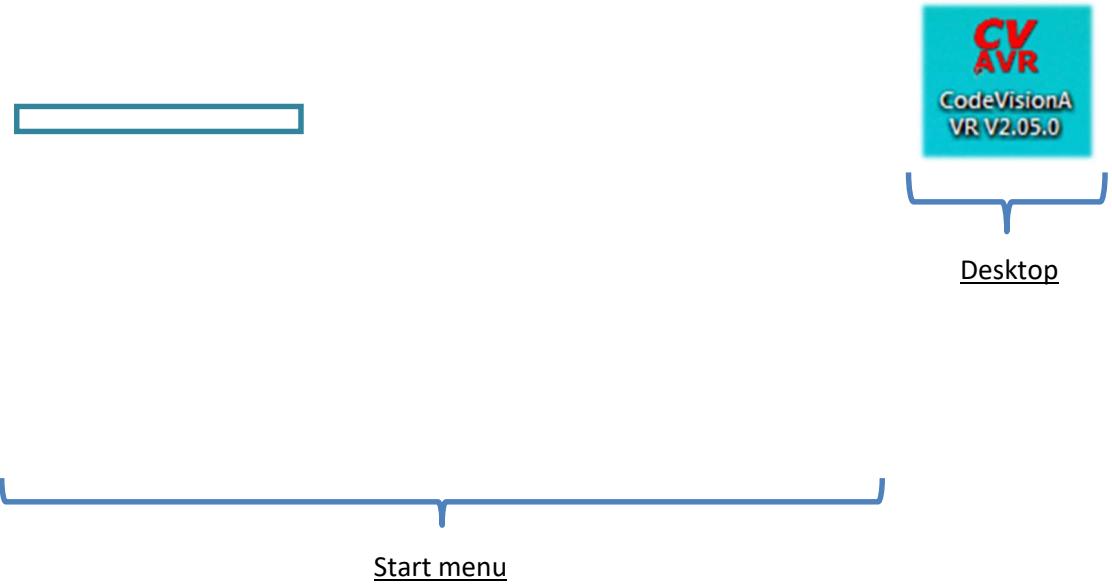


Gambar 13.4 Komponen Training board

13.5. Prosedur Praktikum

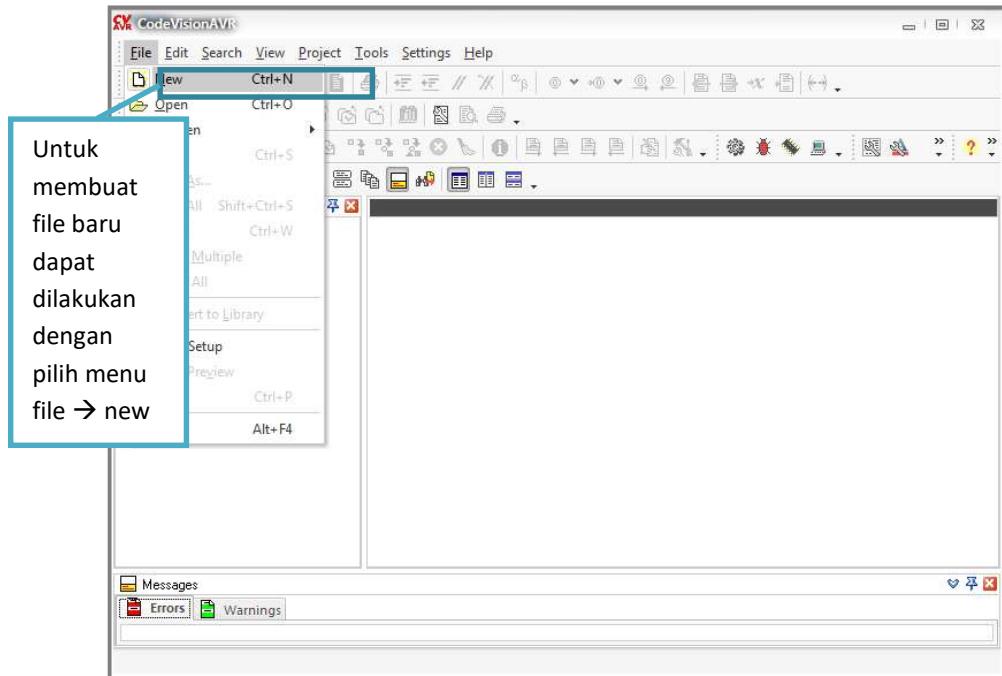
1. Jalankan aplikasi CodeVisionAVR dengan cara melakukan klik ganda pada shortcut icon CodeVisionAVR pada desktop atau dengan cara mencari shortcut program pada start menu system operasi windows.





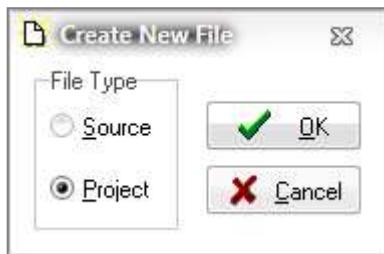
Gambar 13.5 Shortcut CodeVisionAVR

Untuk memulai membuat project baru, pada menubar, pilih File → New, seperti yang ditunjukkan oleh gambar berikut:



Gambar 13.6 Membuat File baru

Setelah itu akan muncul suatu dialog yang mengharuskan *user* untuk memilih antara pembuatan source file atau project. Dalam setiap percobaan pada praktikum ini, dibuat project baru untuk setiap percobaan, oleh karena itu *user* sebaiknya melakukan *check* pada checkbox project dan dilanjutkan dengan menekan tombol “OK”



Gambar 13.7 Membuat project baru

Berikutnya akan keluar dialog yang menanyakan Anda apakah akan membuat project baru. Klik tombol “Yes” untuk menyetujui.

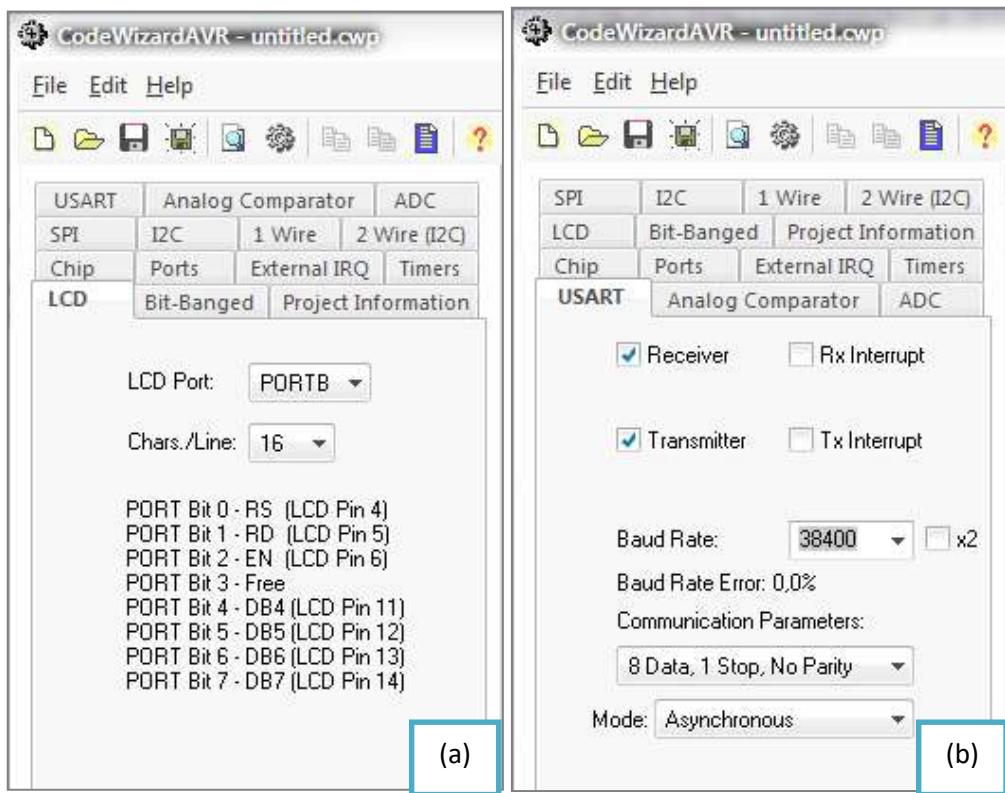


Gambar 13.8 Memilih untuk membuat project baru pada CodeVisionAVR

Setelah itu lakukan set pada CodeWizardAVR. Untuk chip diset dengan nama chip ATmega16 dengan clock 12 MHz.

Berikutnya Anda akan menginisialisasi LCD pada PortB, yaitu dengan cara masuk pada tab LCD dan pilih item **PORTB** pada selectbox “LCD Port”. Set banyaknya karakter dari LCD yang diinginkan. Dalam hal ini menggunakan 16 karakter, oleh karena itu pilih item **16** pada selectbox “Chars/Line”.

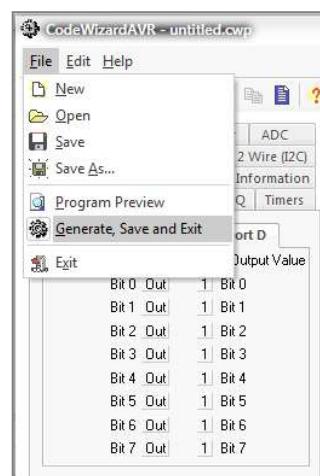
Setelah itu anda diharuskan untuk melakukan setting USART dengan baudrate = 38400



Gambar 13.9(a) Setting LCD pada PortB

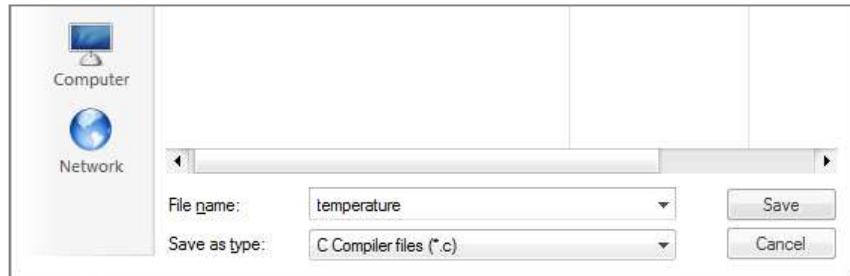
(b) Setting USART 8 bit

Karena pada contoh ini tidak digunakan fasilitas lain maka setting CodeWizardAVR siap disimpan dalam file. Pada menu CodeWizardAVR, pilih File → Generate, Save and Exit, seperti yang ditunjukkan pada gambar berikut:



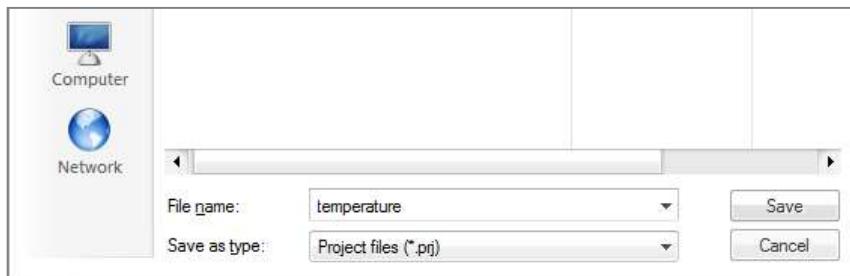
Gambar 13.10 Menyimpan setting

Setelah penekanan menu “Generate, Save and Exit”, akan muncul suatu dialog untuk melakukan set terhadap nama dan lokasi penyimpanan dari source file (*.c) seperti ditunjukkan pada gambar berikut:



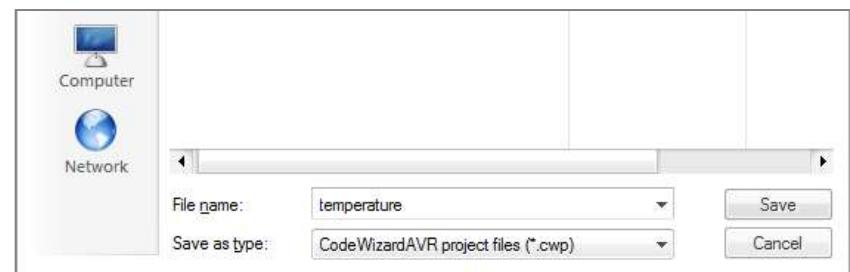
Gambar 13.11 Memberi nama pada file C (*.c)

Tekan tombol “Save” untuk menyetujui dan melanjutkan pada proses berikutnya. Setelah penekanan tombol “Save” akan muncul kembali dialog yang kedua, yaitu untuk melakukan set terhadap nama dan lokasi penyimpanan project file (*.prj) seperti ditunjukkan pada gambar berikut:



Gambar 13.12 Memberi nama project (*.prj)

Tekan tombol “Save” untuk menyetujui dan melanjutkan pada proses berikutnya. Setelah penekanan tombol “Save” akan muncul kembali dialog yang ketiga, yaitu untuk melakukan set terhadap nama dan lokasi penyimpanan CodeWizardProject file (*.cwp) seperti ditunjukkan pada gambar berikut:



Gambar 13.13 Memberi nama CodeWizardProject (*.cwp)

Tekan tombol “Save” untuk menyetujui dan melanjutkan. Selanjutnya akan muncul IDE editor program dari CodeVisionAVR. Anda siap untuk melakukan pembuatan program menggunakan CodeVisionAVR.

2. Pada program-C yang dihasilkan, tambahkan header `delay.h` dan `stdio.h`
3. Pada bagian `while(1) {...}` tambahkan perintah yang telah diblok dengan warna kuning berikut :

```
while (1)
{
    // Place your code here
    putchar(0x01);
    temp=getchar()-40;

    lcd_clear();
    lcd_gotoxy(0,0);
    lcd_putsf("Temp= ");
    sprintf(xstring,"%d C",temp);
    lcd_puts(xstring);

    delay_ms(100);

};
```

4. Untuk lebih jelasnya, berikut kode program secara keseluruhan:

```
#include<mega16.h>
#include<delay.h>

// Alphanumeric LCD Module functions
#asm
    .equ __lcd_port=0x18 ;PORTB
#endif
#include<lcd.h>

// Standard Input/Output functions
#include<stdio.h>

// Declare your global variables here
unsignedchar hum,temp,xstring[16];
void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTA=0x00;
```

```

DDRA=0x00;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTB=0x00;
DDRB=0x00;

// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTD=0x00;
DDRD=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock

```

```

// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud Rate: 38400
UCSRA=0x00;
UCSRB=0x18;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x11;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// LCD module initialization
lcd_init(16);

while (1)
{
// Place your code here
    putchar(0x01);
    temp=getchar()-40;

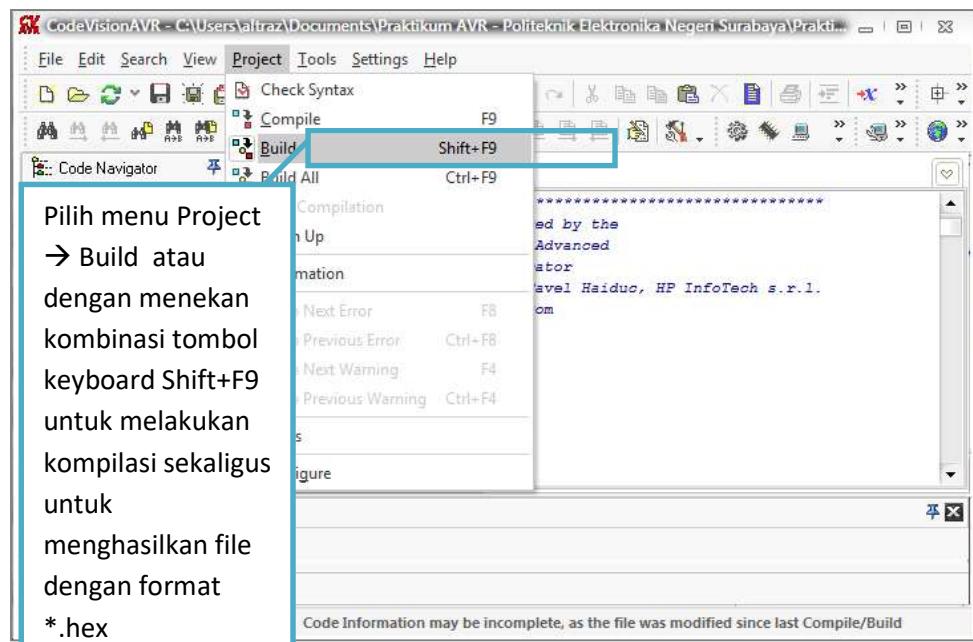
    lcd_clear();
    lcd_gotoxy(0,0);
    lcd_putsf("Temp= ");
    sprintf(xstring,"%d C",temp);
    lcd_puts(xstring);

    delay_ms(100);
}

}

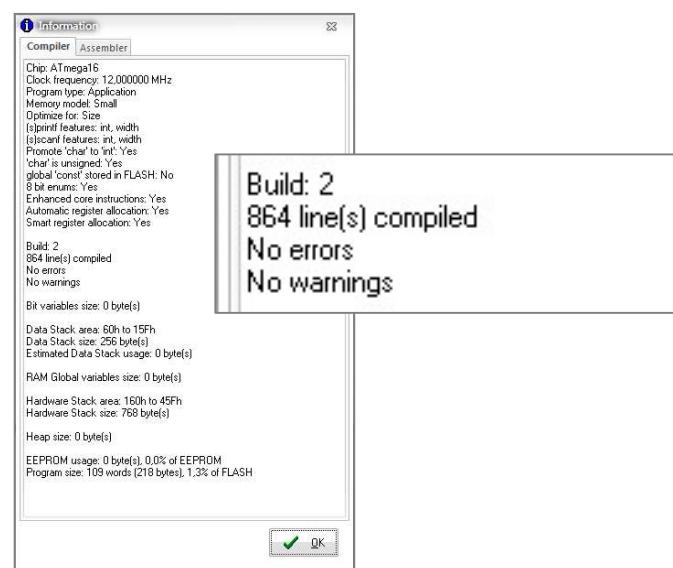
```

5. Setelah pembuatan program selesai, maka proses selanjutnya yaitu lakukan kompilasi terhadap kode program tersebut sekaligus untuk menghasilkan file dengan format hexadecimal (*.hex). Hal ini dapat dilakukan dengan memilih menu Project → Build (Shift+F9) seperti ditunjukkan pada gambar berikut :



Gambar 13.14 Build Source

Apabila kode program sudah benar (tanpa error) dan tidak menyalahi struktur pemrograman bahasa C, maka akan muncul dialog informasi seperti ditunjukkan pada gambar berikut:



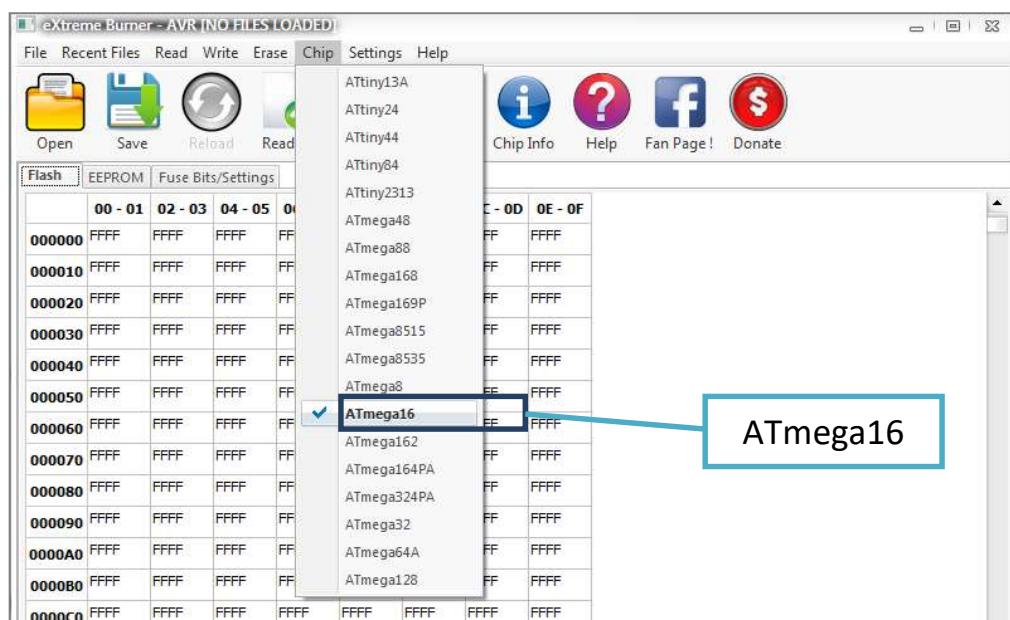
Gambar 13.15 Dialog informasi status kompilasi

6. Setelah diperoleh file dengan format *.hex, maka lakukan download file *.hex ke dalam mikrokontroler. Pertama, jalankan program eXtreme Burner - AVR dengan cara melakukan klik ganda pada shortcut icon eXtreme Burner – AVR pada desktop



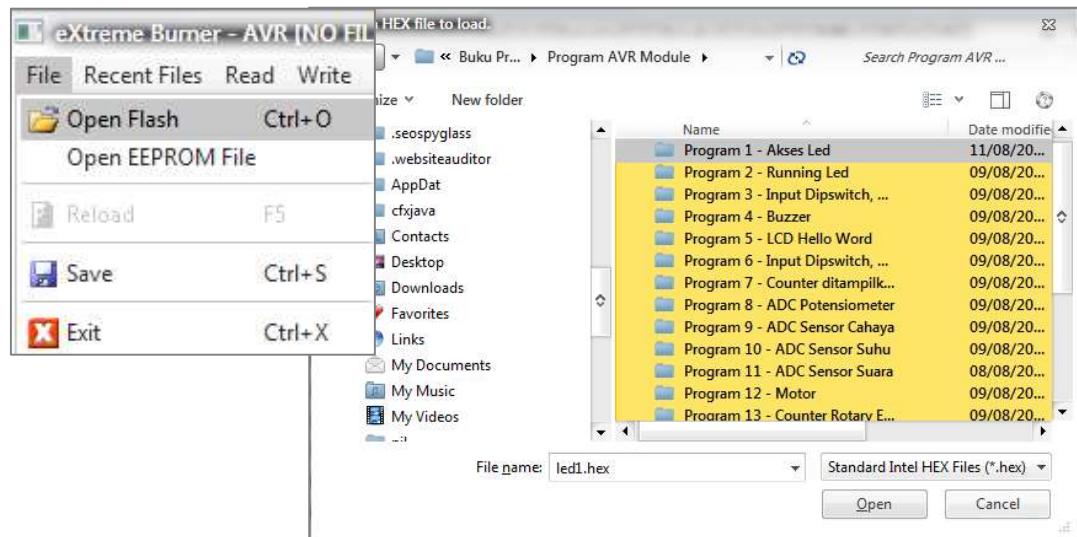
Gambar 13.16 Shortcut eXtreme Burner - AVR

Selanjutnya akan muncul tampilan utama dari program eXtreme Burner – AVR. Pada praktikum ini menggunakan mikrokontroler keluarga AVR dengan tipe IC ATMega16. Oleh karena itu, lakukan pengaturan terhadap tipe IC terlebih dahulu dengan memilih menu Chip → ATmega16 seperti ditunjukkan pada gambar berikut:



Gambar 13.17 Tipe Mikrokontroler

Setelah itu, lakukan open file *.hex dengan memilih menu File → Open Flash (Ctrl+O) seperti ditunjukkan pada gambar berikut:



Gambar 13.18 Open Flash

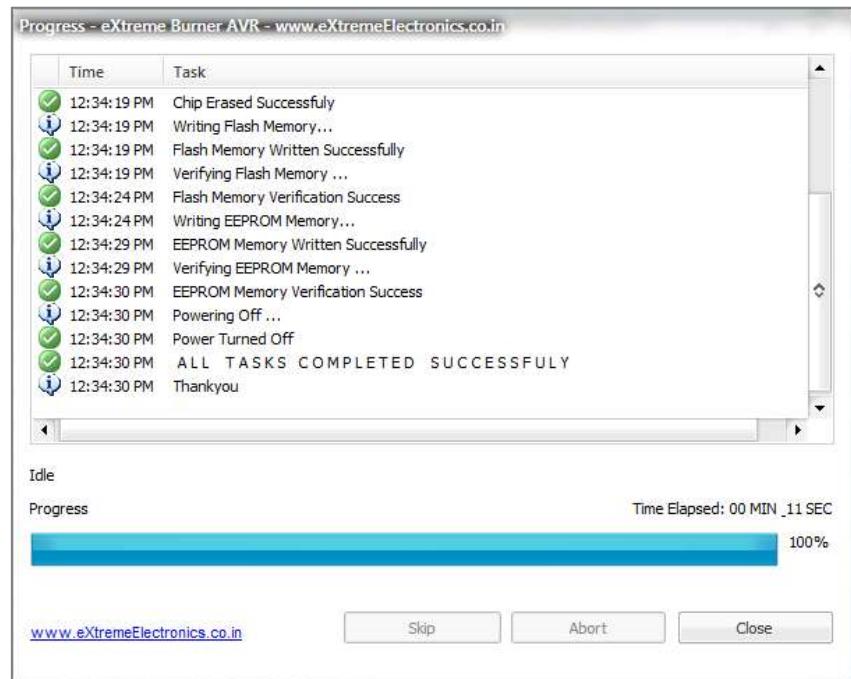
Lakukan pencarian terhadap lokasi file *.hex Anda, untuk selanjutnya tekan tombol “Open” apabila sudah ditemukan file yang dimaksud. Setelah itu akan muncul pesan yang memberitahukan bahwa file *hex berhasil di-load program eXtreme Burner – AVR.



Gambar 13.19 Message



Selanjutnya tekan tombol  untuk memulai download ke dalam mikrokontroler Anda. Berikut ini merupakan tampilan apabila proses download ke dalam mikrokontroler berhasil dilakukan:



Gambar 13.20 Proses Berhasil

7. Selesai

13.6. Hasil Percobaan



Gambar 13.21 Running Program

PERCOBAAN 14

**SENSOR GAS LPG :
MQ5 LPG GAS**

PERCOBAAN 14

SENSOR GAS LPG : MQ5 LPG GAS

14.1. Tujuan

1. Mahasiswa dapat mengetahui cara kerja sensor GAS LPG MQ5
2. Mahasiswa dapat mengakses sensor gas MQ5 LPG GAS pada mikrokontroler ATMega 16
3. Mahasiswa dapat membuat program sederhana untuk mengimplementasikan sensor gas MQ5 LPG GAS

14.2. Dasar Teori

1. Sensor Gas MQ5

Sangat baik digunakan di industri maupun rumah untuk mendeteksi LPG, natural gas dengan mengabaikan noise gas seperti alkohol, dan asap rokok.



Gambar 14.1 Sensor Gas MQ5

2. Spesifikasi

- Kondisi standart

Symbol	Parameter name	Technical condition	Remarks
V _c	Circuit voltage	5V±0.1	AC OR DC
V _H	Heating voltage	5V±0.1	ACOR DC
P _L	Load resistance	20KΩ	
R _H	Heater resistance	31±10%	Room Tem
P _H	Heating consumption	less than 800mw	

- Kondisi lingkungan

Symbol	Parameter name	Technical condition	Remarks
T _{a0}	Using Tem	-10°C-50°C	
T _{as}	Storage Tem	-20°C-70°C	
R _H	Related humidity	less than 95%Rh	
O ₂	Oxygen concentration	21%(standard condition)Oxygen concentration can affect sensitivty	minimum value is over 2%

- Karakteristik sensitivitas respon

Symbol	Parameter name	Technical parameter	Remarks
Rs	Sensing Resistance	10K Ω - 60K Ω (5000ppm methane)	Detecting concentration scope: 200-10000ppm LPG,LNG Natural gas, iso-butane, propane Town gas
a (5000ppm/1000 ppm CH ₄)	Concentration slope rate	≤ 0.6	
Standard detecting condition	Temp: 20°C $\pm 2^\circ\text{C}$ Humidity: 65% $\pm 5\%$	Vc:5V ± 0.1 Vh: 5V ± 0.1	
Preheat time	Over 24 hour		

- Struktur dan konfigurasi, rangkaian dasar pengukuran

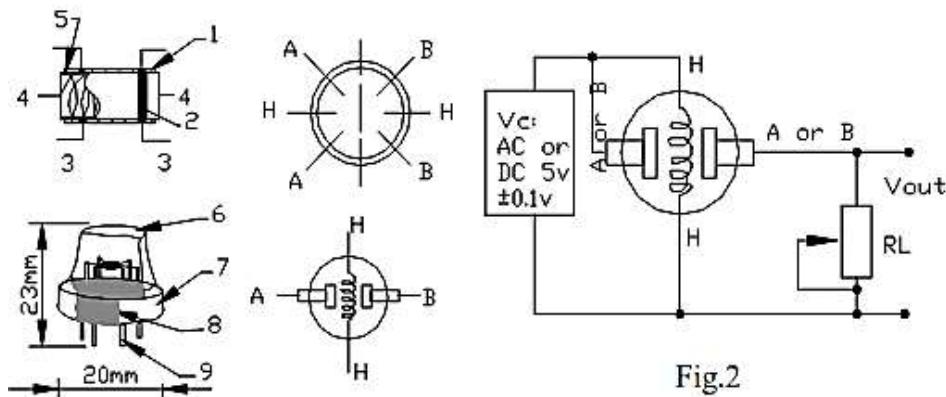


Fig.2

Gambar 14.2 Rangkaian Sensor gas

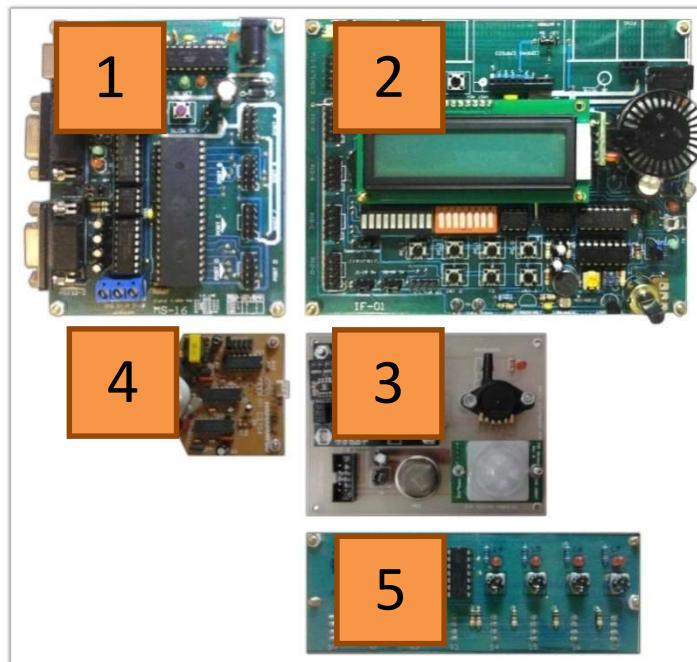
	Parts	Materials
1	Gas sensing Layer	SnO ₂
2	Electrode	Au
3	Electrode line	Pt
4	Heater coil	Ni-Cr alloy
5	Tubular ceramic	Al ₂ O ₃
6	Anti-explosion network	Stainless steel gauze (SUS316 100-mesh)
7	Clamp ring	Copper plating Ni
8	Resin base	Bakelite
9	Tube Pin	Copper plating Ni

14.3. Peralatan

- 1 set PC yang dilengkapi dengan software CodeVision AVR.
- 1 set development board AVR ATmega16
- 1 power-supply +9VDC
- Gas korek api

14.4. Modul I/O

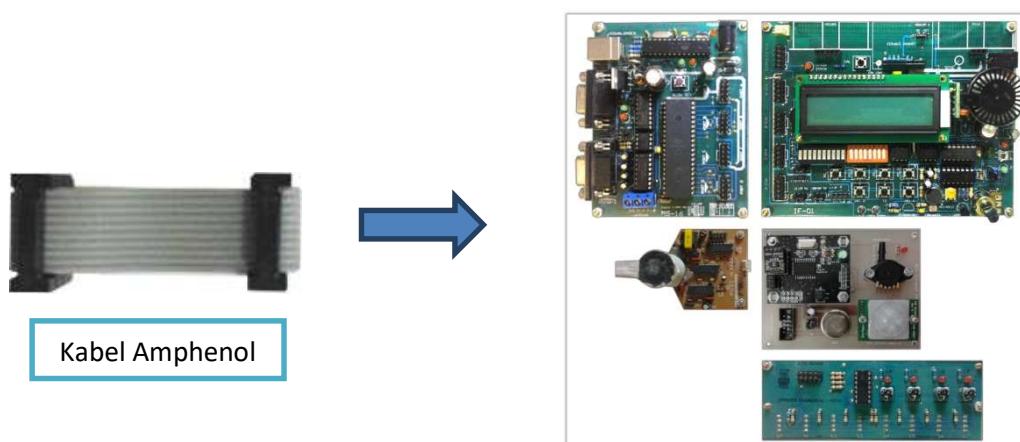
Berikut ini merupakan development board AVR ATmega16:



Gambar 14.3 Development board

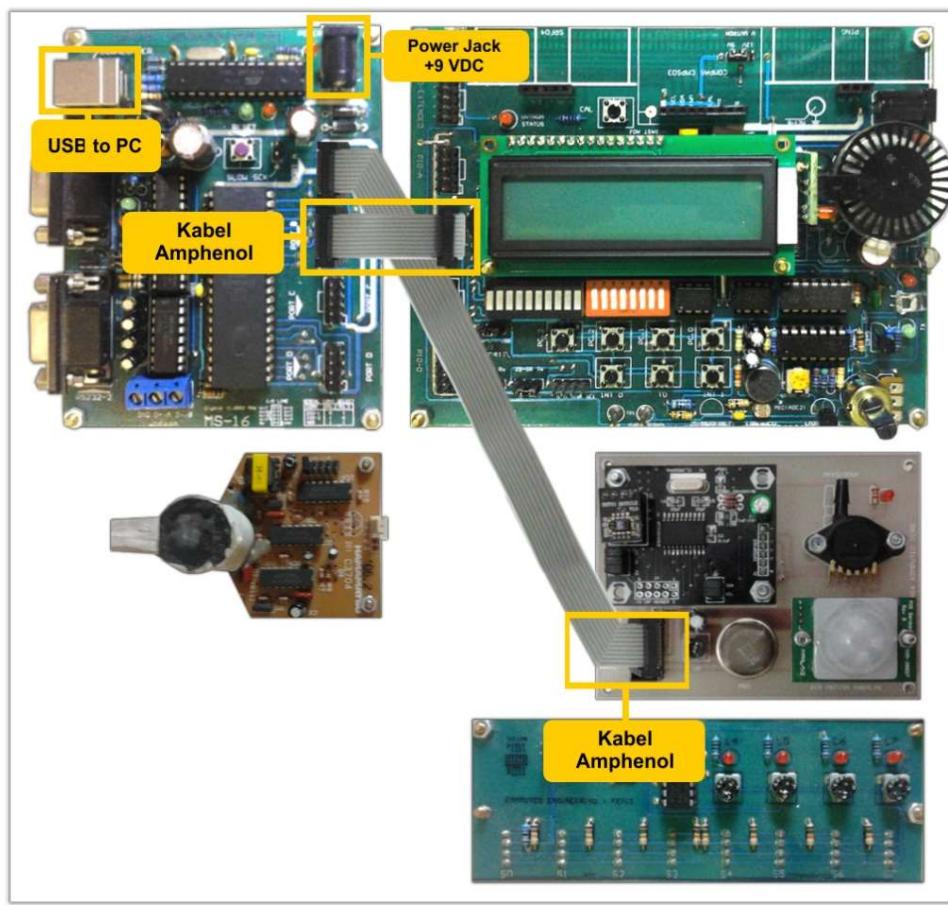
- **Konfigurasi modul pada percobaan ini :**

Pasangkan kabel amphenol pada modul development untuk menghubungkan sistem minimum ATMega16 dengan Training Board. Pasangkan sensor ultrasonic ping parallax pada training board.



Gambar 14.4 Komponen Training board

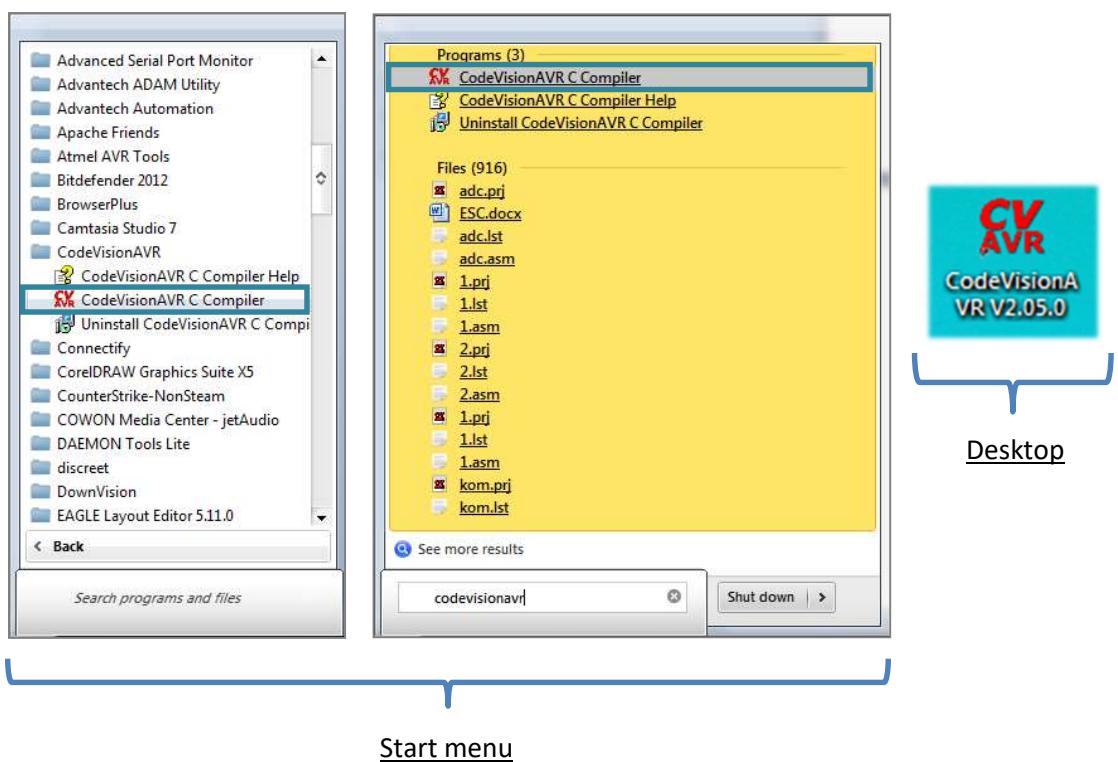
Sehingga menjadi seperti ditunjukkan pada gambar berikut:



Gambar 14.5 Komponen Training board

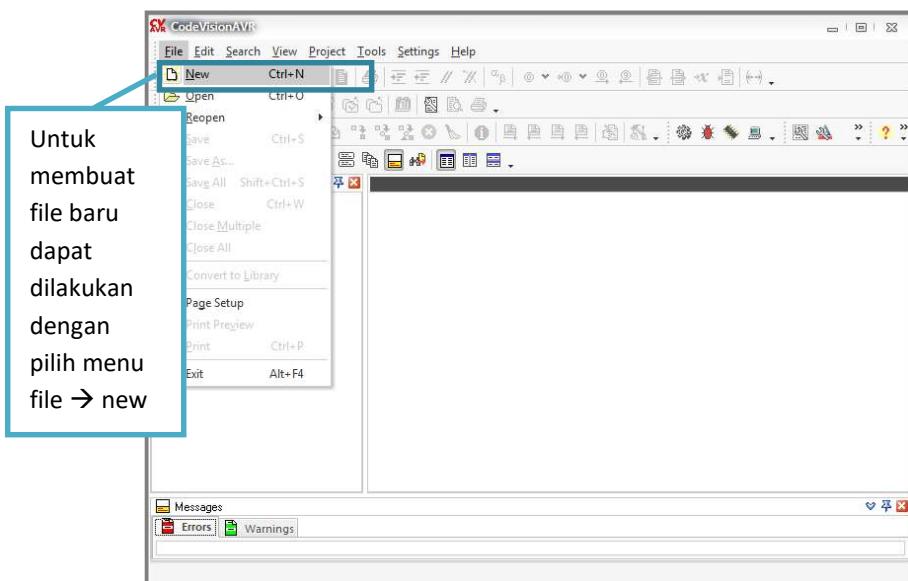
14.5. Prosedur Praktikum

1. Jalankan aplikasi CodeVisionAVR dengan cara melakukan klik ganda pada shortcut icon CodeVisionAVR pada desktop atau dengan cara mencari shortcut program pada start menu system operasi windows.



Gambar 14.6 Shortcut CodeVisionAVR

Untuk memulai membuat project baru, pada menubar, pilih File → New, seperti yang ditunjukkan oleh gambar berikut:



Gambar 14.7 Membuat File baru

Setelah itu akan muncul suatu dialog yang mengharuskan *user* untuk memilih antara pembuatan source file atau project. Dalam setiap percobaan pada praktikum ini, dibuat project baru untuk setiap percobaan, oleh karena itu *user* sebaiknya melakukan *check* pada checkbox project dan dilanjutkan dengan menekan tombol “OK”



Gambar 14.8 Membuat project baru

Berikutnya akan keluar dialog yang menanyakan Anda apakah akan membuat project baru. Klik tombol “Yes” untuk menyetujui.

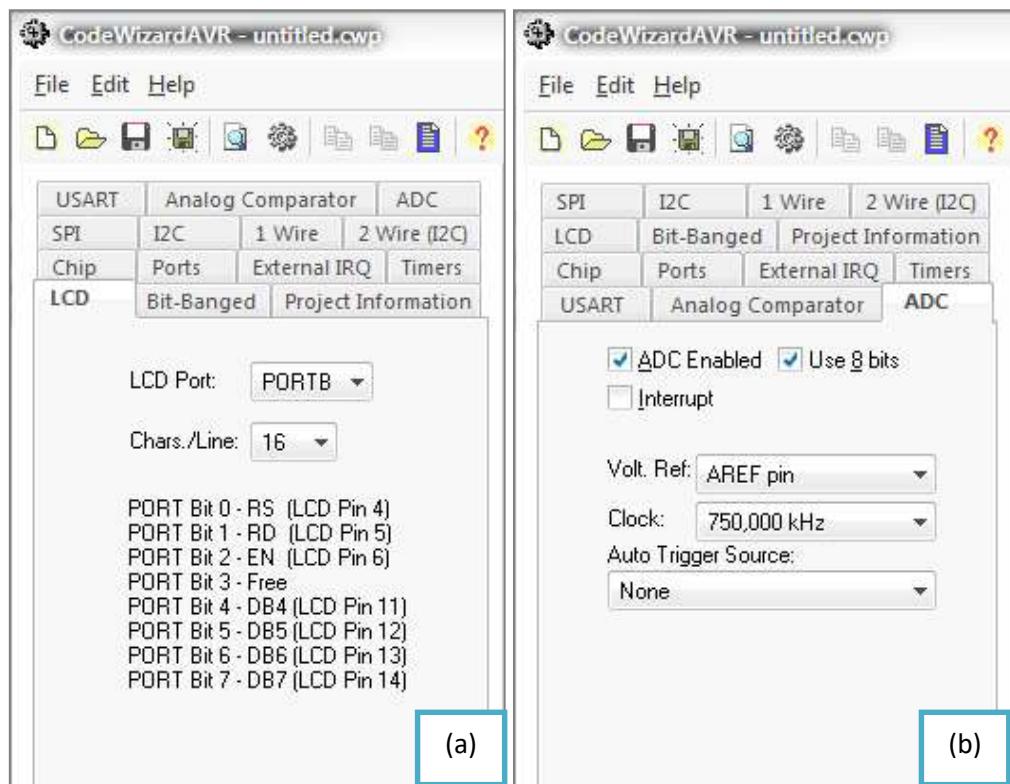


Gambar 14.9 Memilih untuk membuat project baru pada CodeVisionAVR

Setelah itu lakukan set pada CodeWizardAVR. Untuk chip diset dengan nama chip ATmega16 dengan clock 12 MHz.

Berikutnya Anda akan menginisialisasi LCD pada PortB, yaitu dengan cara masuk pada tab LCD dan pilih item **PORTB** pada selectbox “LCD Port”. Set banyaknya karakter dari LCD yang diinginkan. Dalam hal ini menggunakan 16 karakter, oleh karena itu pilih item **16** pada selectbox “Chars/Line”.

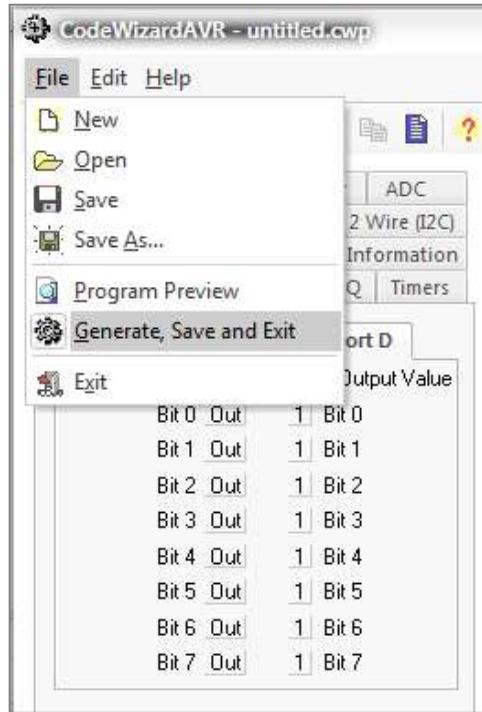
Setelah itu lakukan set ADC dengan masuk pada tab ADC. Berikan tanda check terhadap checkbox “ADC Enabled” untuk meng-enable ADC. Berikan tanda check terhadap checkbox “Use 8 bits” untuk menggunakan ADC 8 bit.



Gambar 14.10 (a) Setting LCD pada PortB

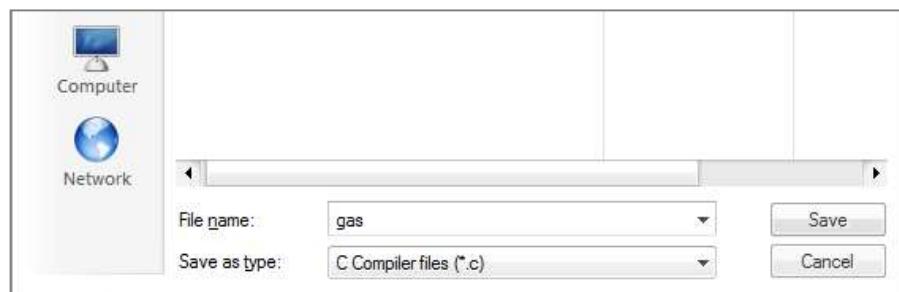
(b) Setting ADC 8 bit

Karena pada contoh ini tidak digunakan fasilitas lain maka setting CodeWizardAVR siap disimpan dalam file. Pada menu CodeWizardAVR, pilih File → Generate, Save and Exit, seperti yang ditunjukkan pada gambar berikut:



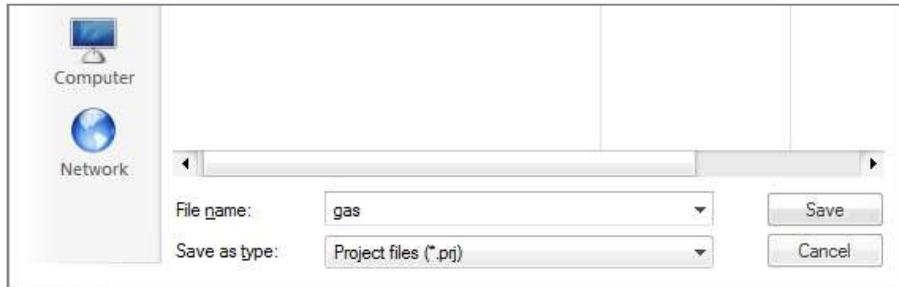
Gambar 14.11 Menyimpan setting

Setelah penekanan menu “Generate, Save and Exit”, akan muncul suatu dialog untuk melakukan set terhadap nama dan lokasi penyimpanan dari source file (*.c) seperti ditunjukkan pada gambar berikut:



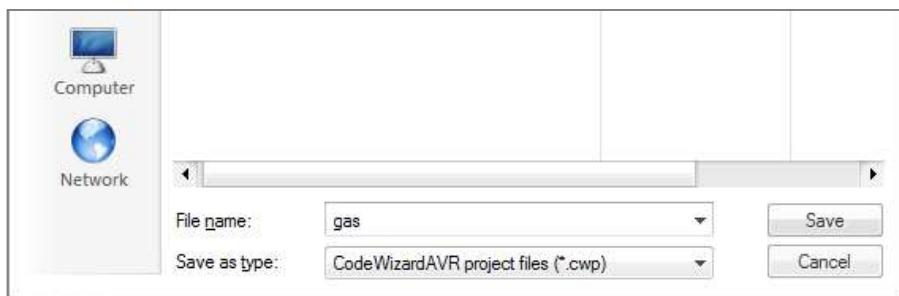
Gambar 14.12 Memberi nama pada file C (*.c)

Tekan tombol “Save” untuk menyetujui dan melanjutkan pada proses berikutnya. Setelah penekanan tombol “Save” akan muncul kembali dialog yang kedua, yaitu untuk melakukan set terhadap nama dan lokasi penyimpanan project file (*.prj) seperti ditunjukkan pada gambar berikut:



Gambar 14.13 Memberi nama project (*.pj)

Tekan tombol “Save” untuk menyetujui dan melanjutkan pada proses berikutnya. Setelah penekanan tombol “Save” akan muncul kembali dialog yang ketiga, yaitu untuk melakukan set terhadap nama dan lokasi penyimpanan CodeWizardProject file (*.cwp) seperti ditunjukkan pada gambar berikut:



Gambar 14.14 Memberi nama CodeWizardProject (*.cwp)

Tekan tombol “Save” untuk menyetujui dan melanjutkan. Selanjutnya akan muncul IDE editor program dari CodeVisionAVR. Anda siap untuk melakukan pembuatan program menggunakan CodeVisionAVR.

2. Pada program-C yang dihasilkan, tambahkan header `delay.h` dan `stdio.h`
3. Pada bagian `while(1) {...}` tambahkan perintah yang telah diblok dengan warna kuning berikut :

```

while (1)
{
// Place your code here
lcd_clear();
    lcd_putsf("Data MQ5 = ");
    data_digital = read_adc(7);
    sprintf(xstring, "%3.0f", data_digital);
    lcd_puts(xstring);

    data_analog = data_digital*5/255;//5:Vref ADC
    lcd_gotoxy(0,1);
    sprintf(xstring, "Konsen:%3.2f Volt", data_analog);
}

```

```

    lcd_puts(xstring);
    delay_ms(100);

};

```

4. Untuk lebih jelasnya, berikut kode program secara keseluruhan:

```

#include<mega16.h>
#include<stdio.h>

// Alphanumeric LCD Module functions
#asm
    .equ __lcd_port=0x18 ;PORTB
#endifasm
#include<lcd.h>

#include<delay.h>

#define ADC_VREF_TYPE 0x20

// Read the 8 most significant bits
// of the AD conversion result
unsignedchar read_adc(unsignedchar adc_input)
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Delay needed for the stabilization of the ADC input voltage
delay_us(10);
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCH;
}

// Declare your global variables here
unsignedchar xstring[16];
float data_analog,data_digital,nilai_sbnnrnya ;

void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTA=0x00;
DDRA=0x00;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T

```

```

State0=T
PORTB=0x00;
DDRB=0x00;

// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTD=0x00;
DDRD=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;

```

```

OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// ADC initialization
// ADC Clock frequency: 691.200 kHz
// ADC Voltage Reference: AREF pin
// ADC Auto Trigger Source: None
// Only the 8 most significant bits of
// the AD conversion result are used
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x84;

// LCD module initialization
lcd_init(16);

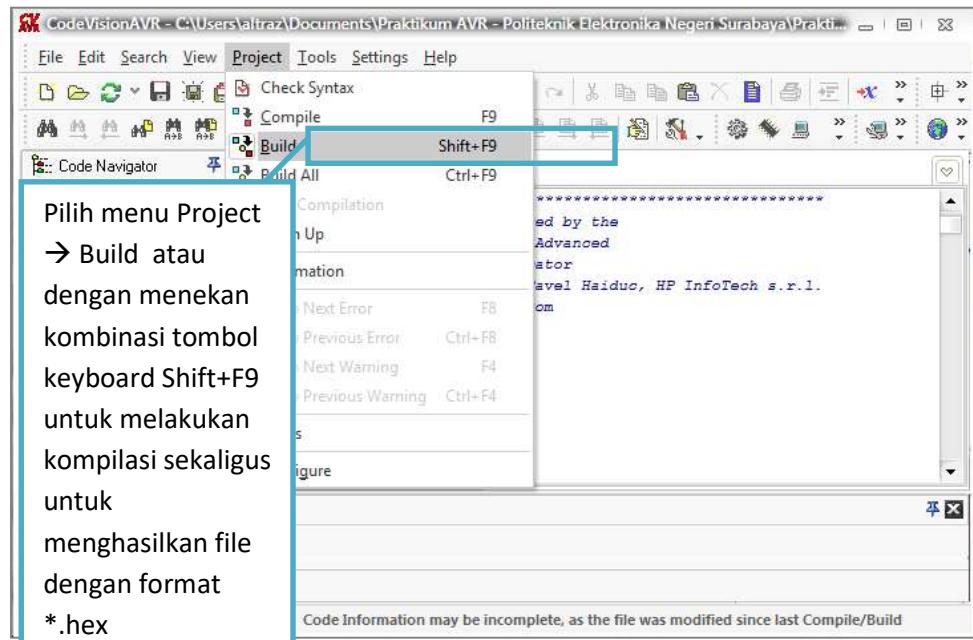
while (1)
{
// Place your code here
    lcd_clear();
    lcd_putsf("Data MQ5 = ");
    data_digital = read_adc(7);
    sprintf(xstring, "%3.0f", data_digital);
    lcd_puts(xstring);

    data_analog = data_digital*5/255;//5:Vref ADC
    lcd_gotoxy(0,1);
    sprintf(xstring, "Konsen:%3.2f Volt", data_analog);
    lcd_puts(xstring);
    delay_ms(100);

}
}

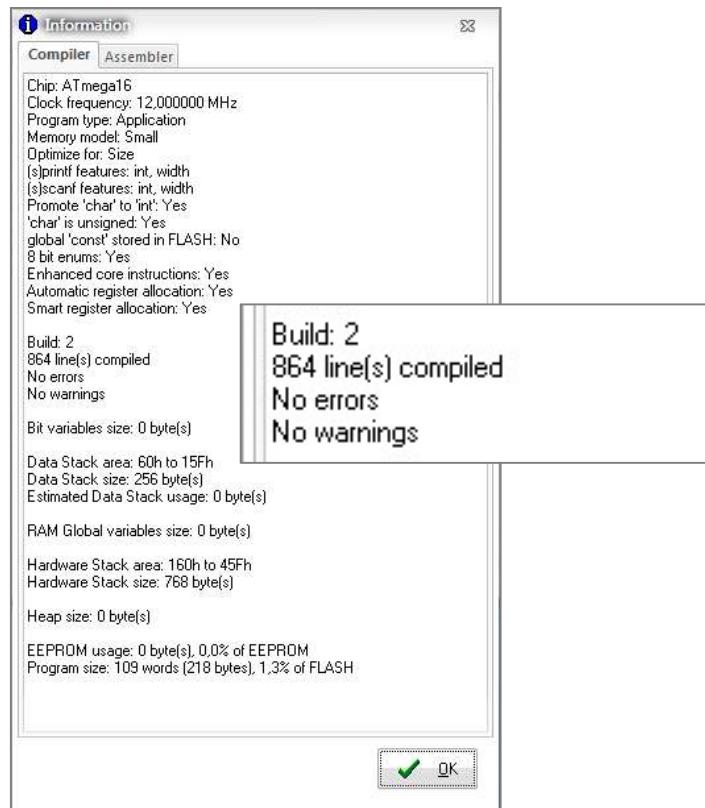
```

5. Setelah pembuatan program selesai, maka proses selanjutnya yaitu lakukan kompilasi terhadap kode program tersebut sekaligus untuk menghasilkan file dengan format hexadecimal (*.hex). Hal ini dapat dilakukan dengan memilih menu Project → Build (Shift+F9) seperti ditunjukkan pada gambar berikut :



Gambar 14.15 Build Source

Apabila kode program sudah benar (tanpa error) dan tidak menyalahi struktur pemrograman bahasa C, maka akan muncul dialog informasi seperti ditunjukkan pada gambar berikut:



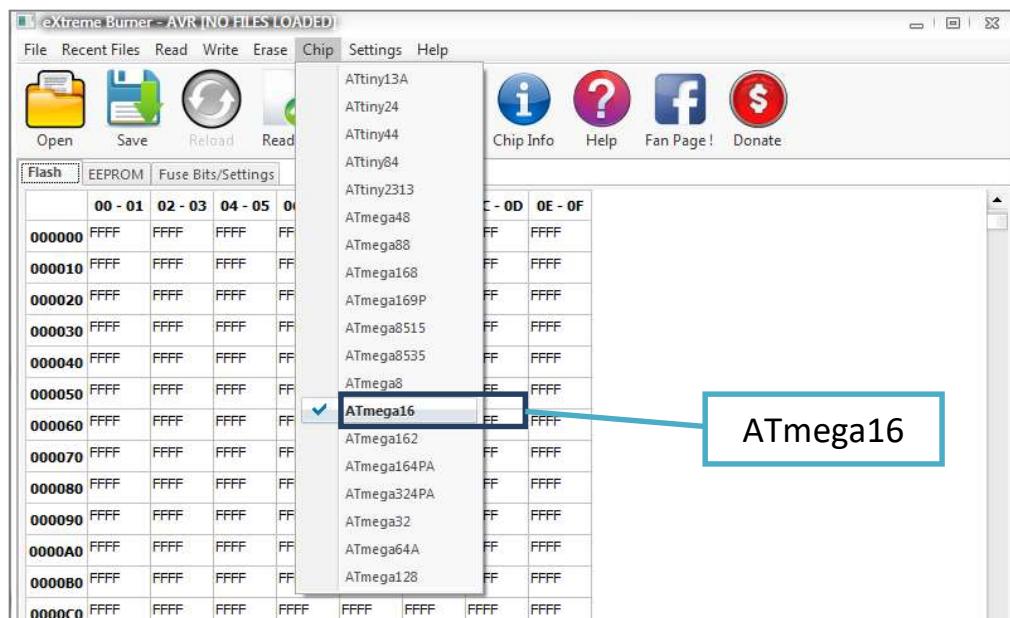
Gambar 14.16 Dialog informasi status kompilasi

6. Setelah diperoleh file dengan format *.hex, maka lakukan download file *.hex ke dalam mikrokontroler. Pertama, jalankan program eXtreme Burner - AVR dengan cara melakukan klik ganda pada shortcut icon eXtreme Burner– AVR pada desktop



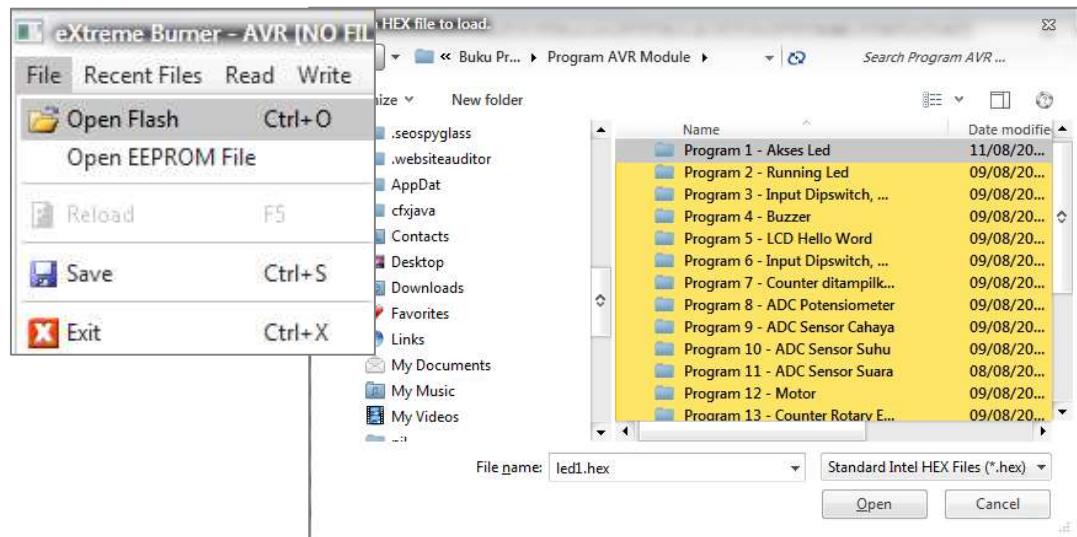
Gambar 14.17 Shortcut eXtreme Burner - AVR

Selanjutnya akan muncul tampilan utama dari program eXtreme Burner – AVR. Pada praktikum ini menggunakan mikrokontroler keluarga AVR dengan tipe IC ATMega16. Oleh karena itu, lakukan pengaturan terhadap tipe IC terlebih dahulu dengan memilih menu Chip → ATmega16 seperti ditunjukkan pada gambar berikut:



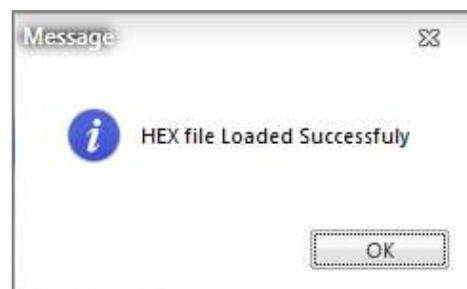
Gambar 14.18 Tipe Mikrokontroler

Setelah itu, lakukan open file *.hex dengan memilih menu File → Open Flash (Ctrl+O) seperti ditunjukkan pada gambar berikut:



Gambar 14.19 Open Flash

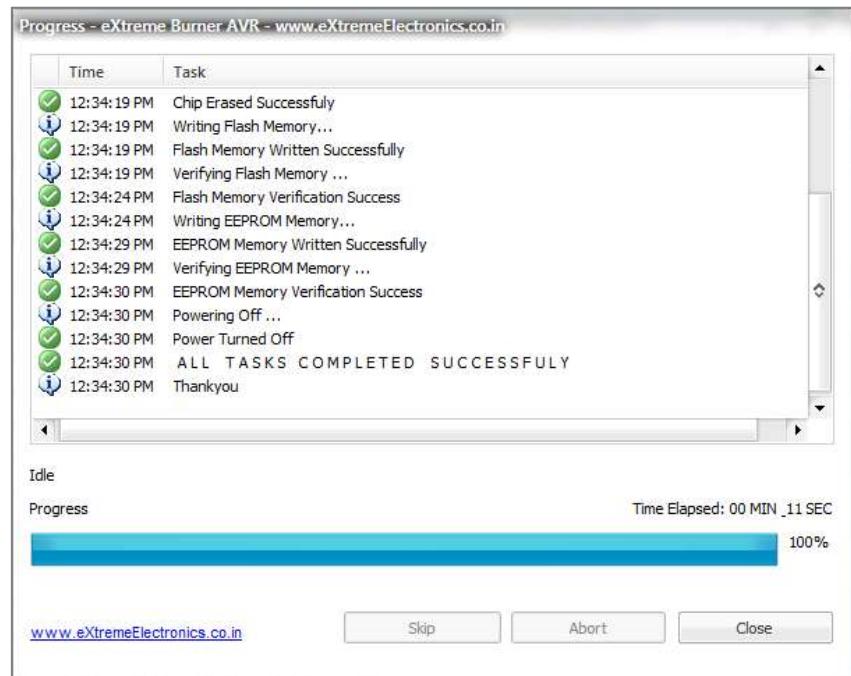
Lakukan pencarian terhadap lokasi file *.hex Anda, untuk selanjutnya tekan tombol “Open” apabila sudah ditemukan file yang dimaksud. Setelah itu akan muncul pesan yang memberitahukan bahwa file *hex berhasil di-load program eXtreme Burner – AVR.



Gambar 14.20 Message



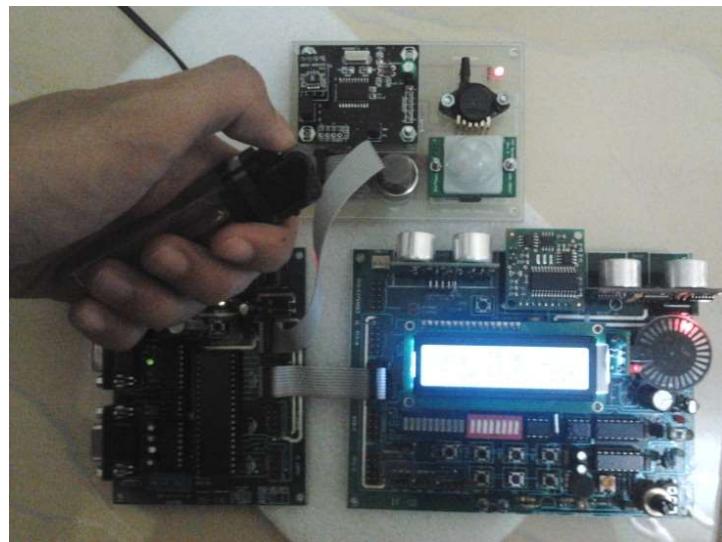
Selanjutnya tekan tombol  untuk memulai download ke dalam mikrokontroler Anda. Berikut ini merupakan tampilan apabila proses download ke dalam mikrokontroler berhasil dilakukan:



Gambar 14.21 Proses Berhasil

7. Selesai

14.6. Hasil Percobaan



Gambar 14.22 Tes menggunakan gas korek api



Gambar 14.23 Respon Program

DAFTAR PUSTAKA

1. Jon S. Wilson, Sensor Technology HandBook, Newnes, Copyright © 2005, Elsevier Inc. All rights reserved.
2. Song Jin Kim, Chong Hui Kim, Jin Baek Kim, and Byung Kook Kim, Department of Electrical Engineering & Computer Science Korea Advanced Institute of Science Technology, “*Ultrasonic Distance Measuring System for Docking of Robotic Wheelchair to Battery Charging Station*” Proceedings of the 8th International Workshop on Human-friendly Welfare Robotic Systems, 21-23 October 2007.
3. Mircea Badescu, Xiaoqi Bao, Yoseph Bar-Cohen, Zensheu Chang, Borna E. Dabiri, Brett Kennedy, Stewart Sherrit PL/Caltech, 4800 Oak Grove Drive, Pasadena, CA 91109-8099, “*Adapting the Ultrasonic/Sonic Driller/Corer for Walking/Climbing Robotic Applications*”, Proceedings of the SPIE Smart Structures Conference San Diego, CA., SPIE Vol. 5764-37, March 7-10, 2005.
4. William A. Lewinger, Michael S. Watson, Roger D. Quinn, “*Obstacle Avoidance Behavior for a Biologically-inspired Mobile Robot Using Binaural Ultrasonic Sensors*”.
5. Jian-Shuen Fang, Qi Hao, David J. Brady, Mohan Shankar, Bob D. Guenther, Nikos P. Pitsianis, Ken Y. Hsu, “Path-dependent human identification using a pyroelectric infrared sensor and Fresnel lens arrays” *Duke University, Durham, NC 27707 USA* (2006).
6. David Van Ess, *Pyroelectric Infrared Motion Detector, PSoC Style(2003)*
7. Fresnel Technologies Inc., <http://www.fresneltech.com/arrays.html>.
8. Glolab Corporation, “Infrared parts manual,” <http://www.glolab.com/pirparts/infrared.html>.
9. Hamamatsu, Flame Sensor, R2868 UVTron, C3704 Series.

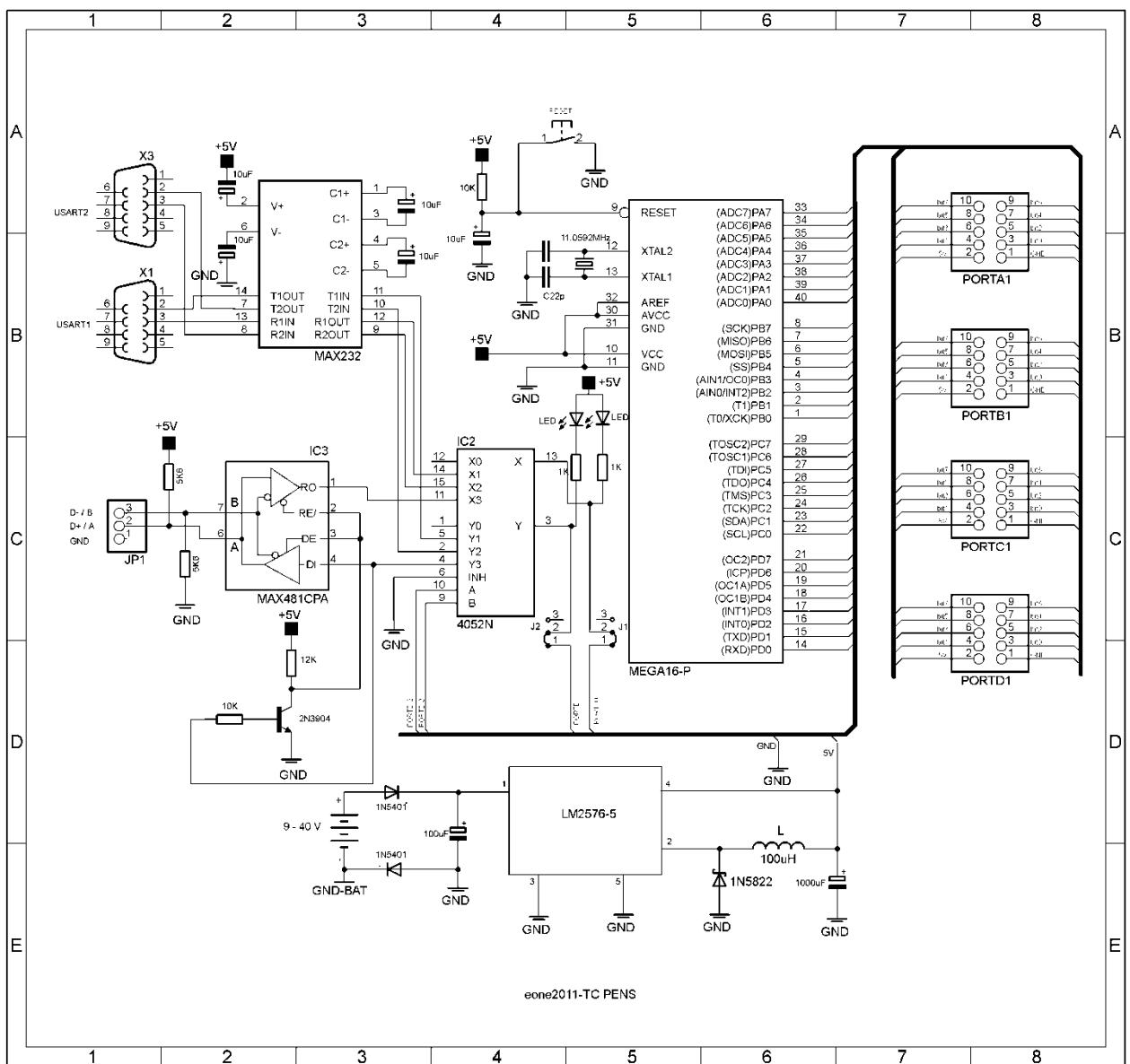
LAMPIRAN

MODUL PRAKTIKUM

LAMPIRAN

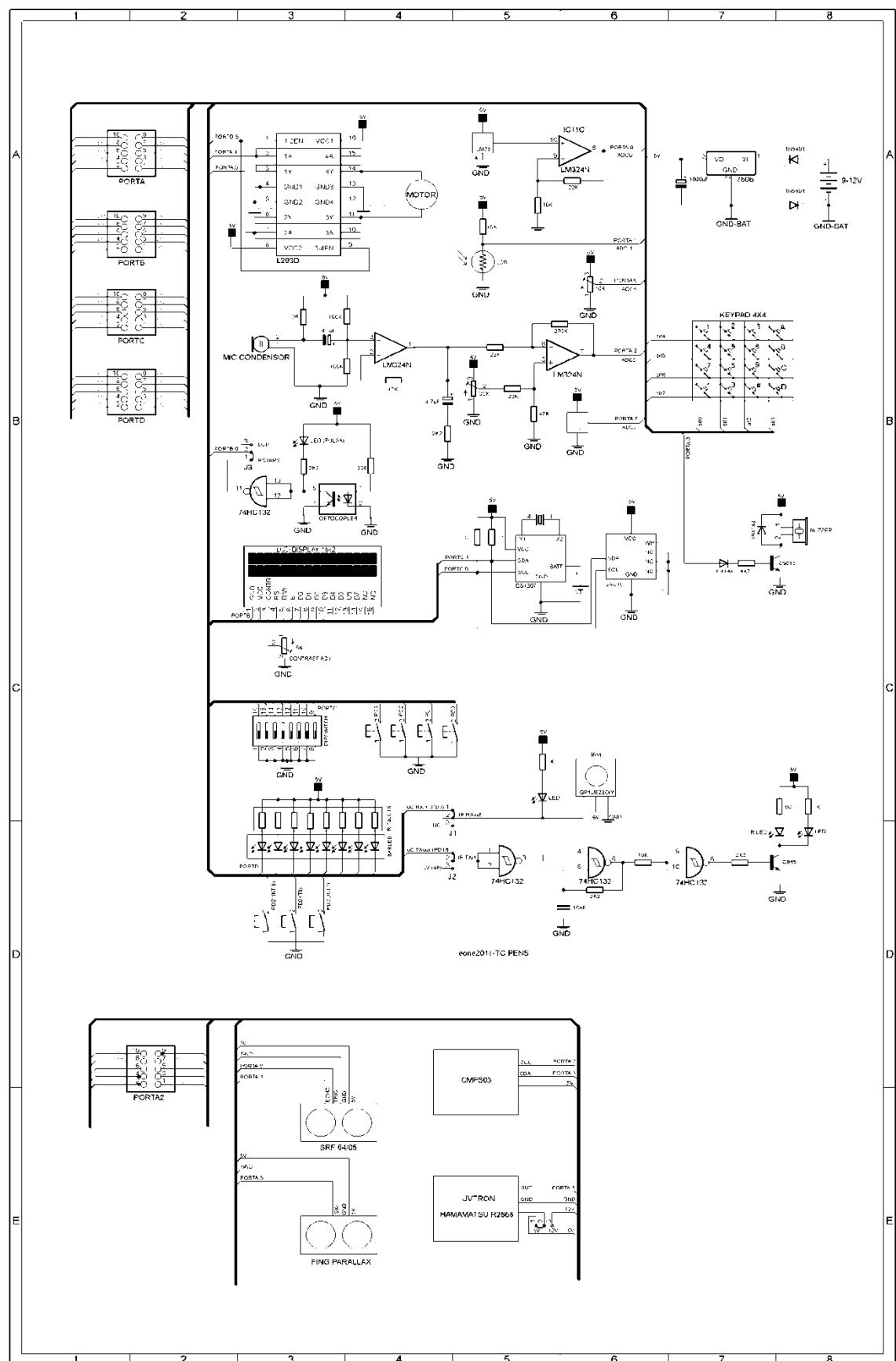
MODUL PRAKTIKUM

1. Minimum System ATMega16



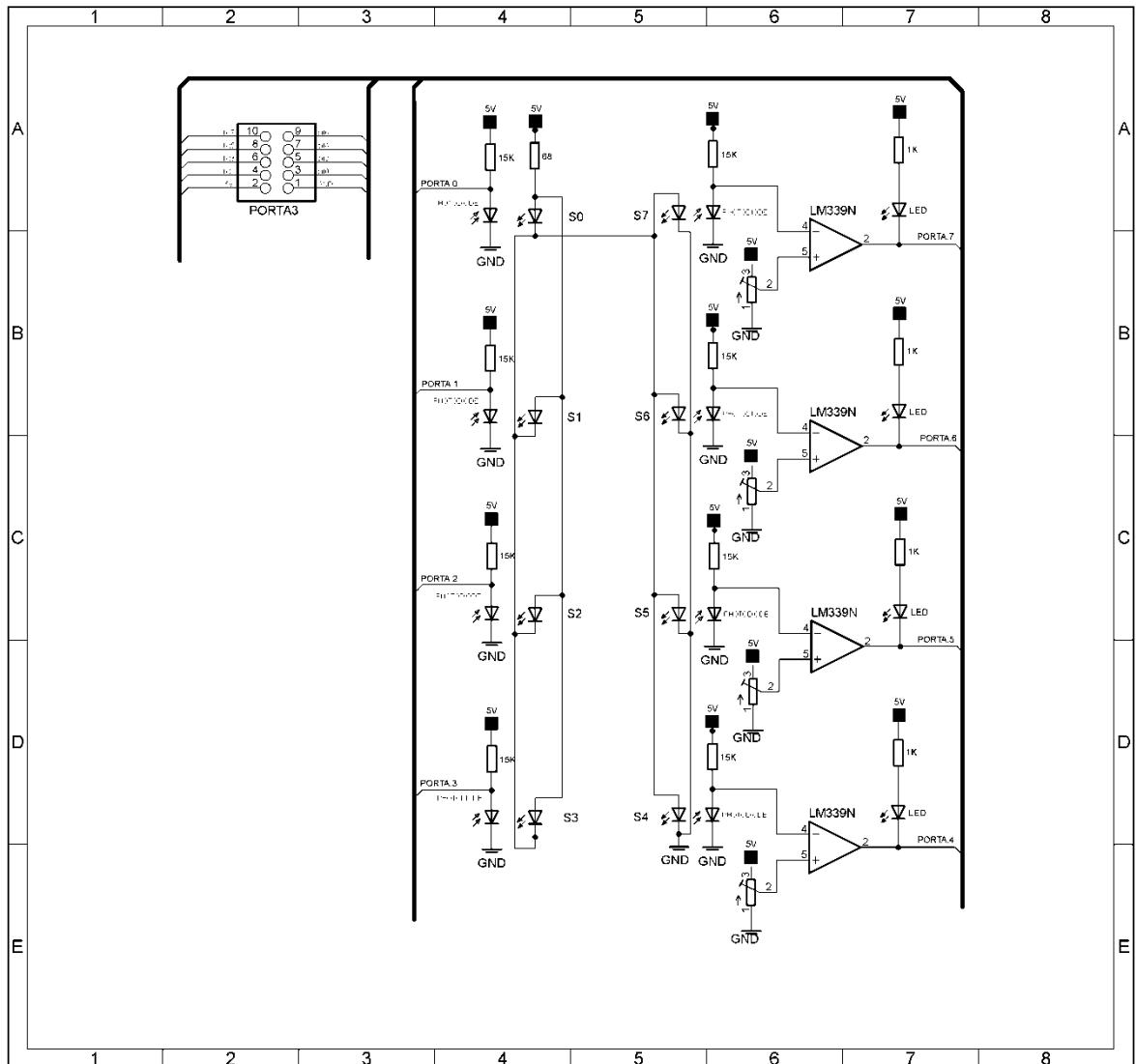
Gambar L.1 Minimum System AVR ATMega16

2. Training Board – Input/Output



Gambar L.2 Training Board – Input/Output Module

3. Training Board – Input/Output



Gambar L.3 Training Board – Input/Output Module