

AVR di Linux

Selain menggunakan Windows Operating System, ternyata pemrograman mikrokontroler AVR juga dapat dilakukan pada Linux Operating System. Pada Linux juga sudah tersedia software yang dapat digunakan untuk melakukan pemrograman, salah satunya adalah software kontrollerlab.

Kontrollerlab merupakan software pemrograman Linux berbasis GUI yang dapat digunakan dengan mudah sebagaimana layaknya kita menggunakan software pemrograman mikrokontroler AVR di Windows. Kontrollerlab sudah menyediakan feature-feature yang sangat lengkap. Jika berminat untuk memiliki software kontrollerlab, kita dapat men-download pada www.rpm.pbone.net.

Selain menggunakan software berbasis GUI, pemrograman mikrokontroler AVR di Linux juga dapat dilakukan dengan menggunakan console. Salah satu kelebihan pemrograman dengan menggunakan console adalah kita dapat lebih memahami perintah-perintah yang digunakan dalam proses pemrograman mikrokontroler AVR tersebut. Untuk dapat melakukan pemrograman mikrokontroler AVR di Linux dengan menggunakan console, berikut tutorial singkat untuk melakukan pemrograman mikrokontroler AVR di Linux:

1. Instalasi Software

Untuk melakukan instalasi software-software tersebut di atas, yang pertama harus kita lakukan adalah mencari *source* software. Anda bisa juga mendapatkan software tersebut dengan cara men-download-nya melalui internet. Sebagai referensi, berikut daftar website untuk men-download software-software yang dibutuhkan:

- binutils <ftp://ftp.gnu.org/gnu/binutils/>
<ftp://gatekeeper.dec.com/pub/GNU/binutils/>
- gcc-core <ftp://ftp.gnu.org/gnu/gcc/>
<ftp://gatekeeper.dec.com/pub/GNU/gcc/>
- avr-libc <http://savannah.nongnu.org/projects/avr-libc/>
- uisp <http://savannah.nongnu.org/projects/uisp>

Anda juga dapat men-download file rpm pada <http://rpm.pbone.net>

Setelah anda mendapatkan software-software tersebut, langkah selanjutnya adalah menginstal software-software tersebut ke dalam harddisk anda. Jika file rpm, anda dapat melakukan instalasi dengan menuliskan perintah

```
$ rpm -ivh [nama file]
```

Jika file anda merupakan file tar, berikut merupakan contoh untuk melakukan instalasi binutils-2.15.tar.bz2, gcc-core-3.4.2.tar.bz2, avr-libc-1.0.4.tar.bz2, dan uisp-20040311.tar.bz2:

```
mkdir /usr/local/avr
```

Tambahkan perintah di bawah ini pada PATH anda:

```
mkdir /usr/local/avr/bin  
export PATH=/usr/local/avr/bin:${PATH}
```

Instalasi Binutils

Pada paket binutils tersedia seluruh kebutuhan *low-level* yang dibutuhkan untuk membangun file-file objek. Pada paket binutils juga telah tersedia *AVR assembler (avr-as)*, *linker (avr-ld)*, *library handling tools (avr-ranlib, avr-ar)*, program untuk menghasilkan file-file objek yang dapat dimasukkan ke dalam EEPROM mikrokontroler (*avr-objcopy*), *disassembler (avr-objdump)* dan kebutuhan-kebutuhan lainnya seperti *avr-strip* dan *avr-size*. Lakukan eksekusi *command* di bawah ini untuk melakukan instalasi binutils :

```
tar jxvf binutils-2.15.tar.bz2
cd binutils-2.15/
mkdir obj-avr
cd obj-avr
../configure --target=avr --prefix=/usr/local/avr --disable-nls
make
# make install
```

Masukkan instruksi `/usr/local/avr/lib` ke dalam file `/etc/ld.so.conf` dan lakukan eksekusi perintah `/sbin/ldconfig` untuk membangun *linker cache*.

Instalasi AVR gcc

avr-gcc akan berfungsi sebagai *compiler* file C yang anda buat. Eksekusi perintah di bawah ini untuk melakukan instalasi avr gcc :

```
tar jxvf gcc-core-3.4.2.tar.bz2
cd gcc-3.4.2
mkdir obj-avr
cd obj-avr
../configure --target=avr --prefix=/usr/local/avr --disable-nls --
enable-language=c
make
# make install
```

Instalasi AVR C-Library

Untuk melakukan instalasi C-library, lakukan eksekusi perintah di bawah ini:

```
tar jxvf avr-libc-1.0.4.tar.bz2
cd avr-libc-1.0.4
PREFIX=/usr/local/avr
export PREFIX
sh -x ./doconf
./domake

cd build
#as root:
make install
```

Instalasi UISP

Software uisp berfungsi untuk menuliskan *object code* ke dalam EEPROM pada mikrokontroler anda.

Software uisp sangat bagus bagi pengguna Linux. Sangat mudah penggunaannya. Untuk melakukan instalasi uisp, anda cukup mengeksekusi perintah sebagai berikut :

```
tar jxvf uisp-20040311.tar.bz2.tar
cd uisp-20040311
./configure --prefix=/usr/local/avr
make
# make install
```

Setelah seluruh tahap instalasi di atas anda lakukan, maka selesailah proses instalasi software yang akan anda butuhkan dalam melakukan pemrograman mikrokontroler AVR di Linux.

Proses instalasi software di atas terkadang mengalami kegagalan. Hal tersebut bisa jadi disebabkan oleh *file source* yang kurang baik. Sebaiknya disarankan untuk mencarinya pada paket rpm. Hal ini akan lebih memudahkan anda dalam melakukan instalasi software-software tersebut.

2. Hardware Programmer

Sebelum melakukan pemrograman *chip* mikrokontroler, harus dipastikan bahwa hardware pemrograman yang akan kita gunakan sudah terinstalasi dengan baik. Atmega8535 merupakan *chip* yang tipe pemrogramannya *In System Programming (ISP)*. Karena itu tidak dibutuhkan *downloader hardware* untuk melakukan pemrogramannya. Untuk pemrograman ISP pada AVR hanya dibutuhkan konfigurasi sebagai berikut:

<i>Pin AVR</i>	<i>Resistor Proteksi</i>	<i>Pin Paralel Port</i>
Reset (10)	-	Init (16)
MOSI (6)	470 Ohm	D0 (2)
MISO (7)	220 Ohm	Busy (11)
SCK (8)	470 Ohm	Strobe (1)
GND (11)	-	GND (18)

Ada satu hal yang perlu diingat, sebaiknya panjang kabel koneksi tidak lebih dari 70 cm.

3. Compiling dan Loading

Setelah file program C telah selesai kita buat, maka proses selanjutnya adalah *compiling* dan *loading* program ke dalam *chip* AVR. Untuk melakukan proses ini, terlebih dahulu kita harus membuat *Make File* pada folder di mana file C tersebut disimpan. Namun sebelum kita melakukan *compiling* dan *loading* program, kita harus memastikan bahwa kita memasukkan */usr/local/avr/bin* ke dalam PATH. Untuk memasukkan */usr/local/avr/bin* ke dalam ke dalam PATH cukup dengan mengedit *.bash_profile* atau *.tcshrc* dengan menambahkan:

```
export PATH=/usr/local/avr/bin:${PATH} (untuk bash)
setenv PATH /usr/local/atmel/bin:${PATH} (untuk tcsh)
```

Untuk melakukan pemrograman AVR kita menggunakan port paralel dan *uisp programmer*. Uisp menggunakan *ppdev interface* pada kernel. Karena itu, kita harus menyertakan perintah di bawah ini pada modul kernel:

```
# /sbin/lsmmod
parport_pc
ppdev
parport
```

Untuk mengetahui apakah modul-modul tersebut sudah masuk ke dalam kernel, lakukan pengecekan dengan menjalankan perintah `/sbin/lsmmod`. Jika pada kernel belum terdapat modul-modul tersebut, jalankan perintah berikut:

```
modprobe parport_pc
modprobe ppdev
modprobe parport
```

Agar *command* tersebut secara otomatis dijalankan pada saat startup, tambahkanlah pada *rc script*. Sebagai contoh, `/etc/rc.d/rc.local` (untuk Redhat). Untuk menggunakan *ppdev* sebagai user normal, root membutuhkan akses *write* dengan cara mengubah status sebagai berikut:

```
chmod 666 /dev/parport0
```

Pastikan bahwa tidak ada divais apapun yang sedang menggunakan port paralel. Kemudian pasanglah *hardware programmer* ke port paralel. Setelah proses di atas selesai dijalankan, sekarang chip mikrokontroler siap diprogram.

Membuat Make File

Fungsi dari sebuah *Make File* adalah sebagai eksekutor perintah-perintah untuk melakukan *compiling* dan *loading* atau beberapa perintah lainnya yang kita inginkan. Di sinilah salah satu kelebihan perograman AVR di Linux atau di *open source*. Jika kita memakai OS *close source*, kita hanya dapat melakukan eksekusi sesuai dengan *default* yang telah dibuat oleh pembuat program tersebut, sedangkan pada OS *open source* kita dapat membuat *Make File* dengan perintah-perintah yang kita inginkan. Contoh dari sebuah *Make File* adalah sebagai berikut:

```
*****
# makefile, written by khoirul umam
MCU=atmega8535
CC=avr-gcc
OBJCOPY=avr-objcopy
CFLAGS=-g -mmcu=$(MCU) -Wall -Wstrict-prototypes -Os -mcall-prologues
all: avrttest.hex
avrttest.hex : avrttest.out
    $(OBJCOPY) -R .eeprom -O ihex avrttest.out avrttest.hex
avrttest.out : avrttest.o
    $(CC) $(CFLAGS) -o avrttest.out -Wl,-Map,avrttest.map avrttest.o
avrttest.o : avrttest.c
    $(CC) $(CFLAGS) -Os -c avrttest.c
load: avrttest.hex
    uisp -dlpt=0x378 --erase -dprog=stk200
    uisp -dlpt=0x378 --upload if=avrttest.hex -dprog=stk200 -v=3 --hash=32
    uisp -dlpt=0x378 --verify if=avrttest.hex -dprog=stk200 -v=3 --hash=32
erase: avrttest.hex
    uisp -dlpt=0x378 --erase -dprog=stk200
upload: avrttest.hex
    uisp -dlpt=0x378 --upload if=avrttest.hex -dprog=stk200 -v=3 --hash=32
verify: avrttest.hex
    uisp -dlpt=0x378 --verify if=avrttest.hex -dprog=stk200 -v=3 --hash=32
clean:
    rm -f *.o *.map *.out
*****
```

Pada contoh *Make File* di atas terdapat beberapa label eksekusi seperti *all:*, *load:*, *erase:*, *upload:*, *verify:*, dan *clean:*. Label-label tersebut merupakan perintah eksekusi yang akan dijalankan pada saat perintah tersebut dipanggil. Nama label-label tersebut dapat diganti

sesuai dengan keinginan kita. Cara untuk memanggil perintah tersebut adalah dengan menyertakan label tersebut setelah perintah *make* pada konsol. Sebagai contoh,

```
make load
```

ketika perintah *make load* dijalankan, maka *script* yang berlabel *load:* akan dieksekusi sehingga perintah-perintah yang berada di dalam label *load:* akan dieksekusi juga. Dengan demikian kita dapat memilih perintah mana yang akan kita jalankan.

Pada contoh *Make File* di atas, statement yang didahului dengan tanda “#” merupakan *comment*. Baris pertama berisi informasi pembuat *Make File* tersebut. *Comment* tidak akan dieksekusi ketika kita melakukan eksekusi.

Baris ke-2 hingga baris ke-5 merupakan perintah-perintah inisialisasi yang akan menunjukkan kepada *programmer* seputar *chip* yang akan diprogram, *C compiler* dan *copy object* yang digunakan, dan juga inisialisasi *flags C*.

Label *all:* merupakan label yang berisi perintah-perintah yang berfungsi membuat file-file object, output, map, dan hex. Apabila kita menjalankan perintah *make all*, maka secara otomatis akan terbentuk file-file tersebut. Pada *make file* di atas, apabila perintah *make all* dijalankan, maka akan terbentuk beberapa file, yaitu: *avrtest.o*, *avrtest.out*, *avrtest.map*, dan *avrtest.hex*. Perintah *make all* merupakan perintah yang harus dieksekusi pertama kali sebelum perintah-perintah lain dieksekusi.

Label *erase:* merupakan label yang berisi perintah-perintah untuk mengosongkan *chip*. Apabila pada *chip* tersebut terdapat program yang sudah tertanam, maka program tersebut akan terhapus dengan perintah *erase* tersebut.

Label *upload:* merupakan label yang berisi perintah-perintah untuk melakukan pengisian program ke dalam *chip*. File yang akan dimasukkan ke dalam *chip* hanyalah file *.hex*.

Label *verify:* merupakan label yang berisi perintah-perintah untuk melakukan verifikasi terhadap file *.hex* yang sudah di*upload* apakah file tersebut sudah sukses disimpan ke dalam *chip* atau belum. Perintah inilah yang akan memastikan proses pemrograman AVR yang telah dilakukan berhasil atau harus dilakukan pengulangan. Jika gagal, maka lakukan kembali proses *uploading*.

Label *clean:* merupakan label yang berisi perintah-perintah untuk menghapus beberapa file. Di dalam label *clean* terdapat nama file **.o*, **.map*, dan **.out*, yang merupakan file-file yang akan dihapus apabila perintah *make clean* dijalankan.

Label *Load:* merupakan label yang di dalamnya merupakan gabungan perintah *erase*, *upload*, dan *verify*. Ketika perintah *make load* dijalankan, maka ketiga instruksi tersebut akan dieksekusi sekaligus.

Berikut beberapa keterangan *script* yang terdapat pada *Make File* di atas:

- *uisp* menunjukkan *programmer* yang kita gunakan untuk melakukan pengisian program.
- *-dplt=0x378* menunjukkan alamat port paralel yang kita gunakan sebagai penghubung antara PC dengan *chip*.
- *-dprog=stk200* menunjukkan tipe *chip* yang akan kita program. Untuk mengetahui macam-macam tipe *chip* beserta kode program yang digunakan dalam *uisp*, silahkan lihat pada *uisp.spec* pada direktori *uisp* anda.

- `-v=3` menunjukkan line yang digunakan dalam pemrograman.
- `--hash=32` merupakan kecepatan pengisian program ke dalam *chip*.

Make File yang dicontohkan di atas berlaku untuk pemrograman *chip* ATmega8535. *Make File* tersebut bisa jadi tidak berlaku untuk *chip* tipe lain karena masing-masing *chip* memiliki spesifikasi yang berbeda. Apabila menemukan kesulitan penulisan kode-kode program dalam pembuatan *Make File*, lihatlah manual uisp dengan menjalankan perintah:

```
man uisp
```

4. Troubleshooting Pemrograman AVR

Dalam pemrograman AVR kendala juga menjadi bagian dari prosesnya, khususnya bagi pemula. Yang jelas beda antara pemrograman AVR dengan menggunakan *open source* dengan *close source* adalah pada pembuatan perangkat pemrogramannya. Jika dalam OS *close source* kita hanya dapat mengikuti *default* pemrograman yang telah tersedia, kita dapat sepenuhnya mengeluarkan kreasi kita dalam membuat perangkat pemrograman sesuai yang kita inginkan. Pada perbedaan ini ternyata juga terdapat beberapa kendala. Beberapa masalah yang sering ditemui pada saat melakukan pemrograman di antaranya adalah sebagai berikut:

Ditemukan pesan sebagai berikut:

```
*****
```

An error has occurred during the AVR initialization.

** Target status:*

Vendor Code = 0xff, Part Family = 0xff, Part Number = 0xff

Check if the programmer is properly connected.

The wiring may be incorrect or target might be 'damaged'.

*make: *** [load] Error 2*

```
*****
```

Pesan di atas memberikan informasi kepada kita bahwa ada beberapa kemungkinan yang menyebabkan pemrograman tidak sukses dilakukan, yaitu:

- Kabel konektor yang menghubungkan PC dengan *chip* tidak terpasang dengan baik.
 - *Chip* mengalami kerusakan sehingga program tidak dapat dimasukkan ke dalam *chip*.
- Apabila kedua hal tersebut telah dicoba untuk diatasi namun ternyata pesan tersebut masih juga muncul, maka ada kemungkinan ada beberapa script di dalam *Make File* yang belum benar. Pengalaman ini juga dialami oleh penulis pada awal-awal melakukan pemrograman AVR di Linux.

```
*****
```

Atmel AVR ATmega8535 is found.

Erasing device ...

Reinitializing device

Atmel AVR ATmega8535 is found.

```
*****
```

Pesan di atas terkadang kita dapatkan ketika menjalankan perintah *make erase*. Jika kita melihat pesan di atas, seolah-olah program telah mendapatkan *chip* sebagai target dan kemudian program melakukan proses pengosongan *chip*. Tapi mengapa aplikasi output program pada *chip* belum menandakan bahwa proses pemrograman telah berhasil dilakukan?.

Ternyata pesan di atas hanya menunjukkan bahwa data pada *-dpart* sudah ditemukan pada uisp. Jika pesan di atas ditemukan, ada kemungkinan *-dlpt* belum didefinisikan sehingga programmer tidak menemukan jalur untuk menuju target.

A ATmega8535 is found.

Up Loading : Flash

Pesan di atas juga serupa dengan pesan error sebelumnya. Bagi pemula, mungkin pesan ini sudah menunjukkan bahwa program sudah berhasil di-upload ke dalam *chip*. Tapi mengapa belum juga ada aplikasi output pada *chip* yang menandakan program tersebut berhasil di-upload?.

Kesalahan ini serupa dengan kesalahan sebelumnya. Jalur untuk menuju target belum didefinisikan sehingga *programmer* tidak dapat *uploading* file ke dalam *chip*.

Berbagai macam pesan mungkin ditemukan dalam pembuatan perangkat pemrograman. Namun pada prinsipnya ketika kita ingin melakukan pemrograman, ada beberapa hal yang harus tercantum dalam *Make File*, yaitu:

- Programmer yang digunakan (*uisp*)
- Alamat port yang digunakan
- Fungsi dari sebuah instruksi (contoh: *--erase*)
- File yang dikirim (ketika melakukan loading)

Error pada saat *make all*:

Pesan error yang tampil pada saat menjalankan perintah *make all* biasanya terjadi karena beberapa hal, yaitu:

- Terdapat kesalahan pada file *C*
- Terdapat kesalahan pada script *all*:

Kiat-kiat pembuatan *Make File*:

Dalam pembuatan *Make File*, ada beberapa kiat yang mungkin dapat memudahkan, yaitu:

- Gunakan manual *uisp* sebagai penuntun pembuatan *Make File*
- Terlebih dahulu buatlah perintah *erase*, *load*, *verify*, dan perintah-perintah lain secara terpisah untuk menguji kebenaran masing-masing instruksi tersebut.

Daftar Pustaka

- Socher Guido, *Programming the AVR microcontroller with GCC, libc 1.0.4*, linuxfocus.org, 2004.
- Doxygen, *avr-libc Reference Manual 1.0.4*, _____, 2004.

Biografi dan Profil



Khoirul Umam. Lahir di Jakarta, 10 Desember 1980. Menamatkan Madrasah di MAN 2 Jakarta Timur pada tahun 1998. Menyelesaikan program S1 pada jurusan Teknik Elektro Program Studi Elektronika di Universitas Negeri Jakarta pada tahun 2005. Saat ini bekerja di PT Nurul Fikri Cipta Inovasi sebagai Staff Teknik Komputer. Kompetensi inti adalah pada bidang Microcontroller dan Robotic.

Berpengalaman sebagai Konsultan Tugas Akhir Mahasiswa Elektronika Dan Teknik Komputer dari berbagai kampus dan juga perusahaan dalam pembuatan alat-alat otomatis berbasis microcontroller. Berpengalaman sebagai Trainer Indoor, Outdoor, dan Outbound Training, pembicara materi-materi organisasi dan life skill di berbagai lembaga, sekolah, dan kampus.

Aktif dalam berbagai organisasi pelajar, kemahasiswaan, sosial, dan politik. Ketua Forum Alumni Rohis MAN 2 dan MTsN 7 (FARM2) Jakarta, Ketua Divisi PSDM Partai Keadilan Sejahtera Dewan Pimpinan Ranting Kelurahan Pasir Gunung Selatan Cimanggis-Depok.

Khoirul Umam termasuk pendiri Kelompok Studi Dan Implementasi Ilmu Pengetahuan Dan Teknologi (KSI-IPTEK), Sentra Kajian Informasi Dan Literatur (SKIL) Learning Center.

Informasi lebih lanjut tentang penulis ini bisa didapat melalui:

Email: umam@nurulfikri.com