Bare Metal C

Arduino merupakan sebuah platform yang memberikan pemula kemudahan untuk belajar mengenai Mikrokontroler, sebelum adanya Arduino kita diharuskan merakit sebuah alat yang nanti akan digunakan untuk berinteraksi dengan Mikrokontroler, dan terkadang kita juga harus memiliki keahlian menggunakan solder, dengan adanya Arduino kita bisa langsung berinteraksi dengan mikrokontroler dan melakukan komputasi, Arduino memiliki sebuah alat yang bernama Arduino IDE, dengan adanya alat ini kita dapat dengan mudah membuat instruksi untuk keperluan kita, Namun dengan adanya alat ini saya merasakan banyaknya abstraksi yang diperlukan untuk membuat Mikrokontroler melakukan apa yang saya ingin lakukan dan saya merasa saya tidak mempelajari hal baru.

Setelah itu mulailah saya mencari cara melakukan pemrograman untuk pemula seperti membuat lampu yang ada di papan Arduino berkedip, Saya menemukan dalam sebuah Blog yang berjudul [0]Programming Arduino Uno in pure C, sebuah blog yang bagus saya sarankan kamu membacanya juga, di dalam blog tersebut penulis menggunakan alat yang bernama avr-gcc, avr-objcopy dan avrdude, karena saat ini saya menggunakan [1]GNU GUIX, saya hanya perlu menginstall "gcc-cross-avr-toolchain" dan "avrdude", avrdude digunakan untuk mengupload atau menginstall program ke Mikrokontroler dan avr-gcc digunakan untuk melakukan kompilasi.

Tetapi terdapat suatu masalah, dalam blog yang saya sebutkan sebelumnya, penulis menggunakan Arduino UNO yang menggunakan Mikrokontroler Atmega328p, sedangkan saya menggunakan Arduino Leonardo yang menggunakan Mikrokontroler Atmega32u4, kedua Mikrokontroler tersebut sangatlah berbeda, karna itu saya memerlukan sebuah petunjuk, Saya mencari sebuah skema dan datasheet untuk [2]Sekema Arduino Leonardo dan [3]Datasheet Atmega32u4 dan setelah melihat kedua Dokumen itu saya membuat program lalu saya upload ke Mikrokontroler, seperti inilah program yang saya buat:

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

#define BLINK_DELAY_MS 500

int main (void) {
  DDRC |= _BV(DDC7);

  while(1) {
    PORTC |= _BV(PORTC7);
    _delay_ms(BLINK_DELAY_MS);

    PORTC &= ~_BV(PORTC7);
    _delay_ms(BLINK_DELAY_MS);
  }
}
```

Dari kode diatas terdapat beberapa C Makro yang asing, janganlah takut karna kita memiliki Datasheet, kita bisa langsung mencari kata-kata asing tersebut di Datasheet, untuk mengetahui apa itu DDRC saya melihat dari Datasheet halaman 213 tepatnya ada di bagian 18.11.5 "Data Direction Register bit must be correctly set to enable the pin as an output.", dari kutipan tersebut kita bisa mengetahui jika kita ingin menggunakan sebuah pin sebagai output kita perlu mengkonfigurasi DDR terlebih dahulu, setelah itu saya melihat Sekema Arduino Leonardo, untuk LED ada di IO13 yang berada di PortC7 jadi kita harus mengisi DDRC ke DDC7 menggunakan C Makro _BV, Makro _BV adalah sebuah makro yang akan melakukan operasi bit, dikonteks ini kita sama saja seperti memanggil kode

```
(1<<DDC7)
```

Artinya adalah kita akan menggeser angka satu ke Makro DDC7, Jika kita mencari definisi dari DDC7 didalam Header file avr/iom32u4.h, DDC7 memiliki arti 7 jika jika kita menggeser 1 ke 7 hasillnya adalah 128 atau dalam binary adalah 10000000, lalu operator |= memiliki arti yang sama dengan += yaitu melakukan operasi yang sama dengan

```
DDRC = DDRC | (1<<DDC7)
```

karna pin yang kita tuju berada di PortC jadi kita akan melakukan operasi di PortC, untuk Makro _delay_ms adalah sebuah makro yang digunakan untuk melakukan sleep pada Mikrokontroler, jika kamu tertarik kamu bisa mencari definisinya di Header file util/delay.h

Setelah saya membuat program saya ingin menjalankanya, karena dalam Programming C terdapat pengulangan perintah jadi saya memubuat sebuah Makefile sederhana, seperti inilah isinya

```
CC=avr-gcc
OBJ_COPY=avr-objcopy
PROGRAMMER=avr109
MCU=atmega32u4
MDIR=~/.guix-home/profile/avr/lib/avr5
BOARD=m32u4
PORT=/dev/ttyACM0
BAUD=57600
F_CPU=16000000UL
OUT = blink fade switch

all: $(addsuffix .elf, $(OUT))

%.o: %.c
        $(CC) -Os -DF_CPU=$(F_CPU) -mmcu=$(MCU) -c -o $@ $<

%.elf: %.o
        $(CC) -mmcu=$(MCU) $< -o $@ -B $(MDIR)

%.hex: %.elf
        $(OBJ_COPY) -O ihex -R .eeprom $< $@


upload_%: %.hex
        avrdude -F -V -v -c $(PROGRAMMER) -p $(BOARD) \
        -P $(PORT) -b $(BAUD) -D -U flash:w:$<

clean:
        rm -f *.hex *.o *.elf
```

Kamu harus menyesuaikan parameter seperti parameter PORT dan MDIR, setelah selesai kamu hanya perlu menjalankan perintah

```
make upload
```

dan akan otomatis terupload ke Mikrokontroler, namun terdapat beberapa yang harus diperhatikan untuk Arduino Leonardo, ketika 8 detik pertama saat terkoneksi melalui USB ke Laptop atau PC akan memasuki bootloader, saat ingin mengupload kita diharuskan berada pada bootloader, jika tidak maka proses upload akan gagal, kamu bisa menekan tombol reset untuk mereset Arduino dan akan memasuki bootloader.

Semua kode di postingan ini dapat diakses di Codeberg dengan judul [4]leonardo-pure-c sebagai hasil karna berhasil menggunakan bahasa C di Arduino saya bisa mengontrol lampu LED TxLED dan RxLED yang berada di PD5 dan PB0 berturut-turut

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

#define BLINK_DELAY_MS 500

int main (void) {

  DDRD |= _BV(DDD5);
  DDRB |= _BV(DDB0);
  DDRC |= _BV(DDC7);

  while(1) {
    PORTD |= _BV(PORTD5);
    _delay_ms(BLINK_DELAY_MS);

    PORTB |= _BV(PORTB0);
    _delay_ms(BLINK_DELAY_MS);

    PORTC |= _BV(PORTC7);
    _delay_ms(BLINK_DELAY_MS);

    PORTD &= ~_BV(PORTD5);
    _delay_ms(BLINK_DELAY_MS);

    PORTB &= ~_BV(PORTB0);
    _delay_ms(BLINK_DELAY_MS);

    PORTC &= ~_BV(PORTC7);
    _delay_ms(BLINK_DELAY_MS);
  }
}
```

Dengan menggunakan avr-gcc toolchain, saya dapat membuat program sederhana yang dapat membuat lampu menyala dan mati tanpa Arduono IDE, dan mempelajari lebih dalam hardware yang saya gunakan.

Referensi

[0] *Programming Arduino Uno in pure C* https://balau82.wordpress.com/2011/03/29/programming-arduino-uno-in-pure-c

[1] *GNU GUIX* https://guix.gnu.org

[2] *Skema Arduino Leonardo* https://www.arduino.cc/en/uploads/Main/arduino-leonardo-schematic.pdf

[3] *Datasheet Atmega32u4* https://ww1.microchip.com/downloads/en/devicedoc/atmel-7766-8-bit-avr-atmega16u4-32u4_datasheet.pdf