UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Enginyeria de Sistemes TIC

# Entorn de treball

## Dispositius Programables — Enginyeria de Sistemes TIC

Jordi Bonet Francisco del Águila

6 de novembre de 2022

## Índex



## 1 Objectiu

L'objectiu d'aquesta primera pràctica és la presentació de les eines de treball, així com l'entorn, per desenvolupar aplicacions en llenguatge Assemblador per dispositius AVR d'Atmel. En particular, la placa de desenvolupament que es farà servir és l'Arduino UNO.

## 2 Eines de treball

Les eines de treball es poden desglossar a nivell de maquinari i de programari.

A nivell de maquinari cal disposar d'un ordinador personal amb sistema operatiu Linux, d'un cable USB amb connector estàndard tipus A en un extrem i B a l'altre i de la placa de desenvolupament amb microcontrolador Arduino UNO. En pràctiques posteriors hi haurà la necessitat de fer servir la placa *protoboard* que ja s'ha fet servir en altres assignatures, així com alguns components electrònics bàsics com leds, resistències, ...

A nivell de programari només caldrà el *Toolchain* de GNU pels dispositius AVR. Aquest Toolchain consta de diferents eines de consola que permet l'assemblat del codi desenvolupat, la transformació i manipulació d'aquest codi en altres formats, i per últim la injecció del codi cap al microcontrolador.

## 2.1 Arduino UNO (Hardware)

Les plaques *Arduino* són una família de plaques de desenvolupament dissenyades al voltant dels microcontroladors de la família AVR d'Atmel. L'Arduino UNO, [Ard], és una placa amb les següents característiques:

1. Basada en el microcontrolador d'Atmel ATmega328p, amb arquitectura AVR, [ATmega328p].
2. Voltatge de funcionament: 5 V.
3. Voltatge recomanat d'entrada: 7 V–12 V.
4. Límits del voltatge d'entrada: 6 V–20 V.
5. 14 pins d'entrada / sortida digital (6 dels quals ofereixen sortida PWM).
6. 6 entrades analògiques
7. Màxima corrent contínua per pin d'entrada / sortida de 40 mA a 5 V i de 50 mA a 3.3 V.
8. Memòria flash de 32 kB, dels quals 0.5 kB els usa el carregador (*Bootloader*).
9. SRAM de 2 kB.
10. EEPROM de 1 kB
11. Velocitat de rellotge de 16 MHz.

Podem trobar tant una imatge com l'esquema de la placa a [Ard].

## 2.2 Toolchain de GNU (Software)

Per desenvolupar aplicacions sobre Arduino hi ha diferents entorns. Aquests entorns majoritàriament estan basats en el llenguatge C. Una de les característiques importants dels AVR és que van estar dissenyats tenint en consideració aquest llenguatge i la feina que fan els compiladors de C per generar codi Assemblador. Per aquest motiu, el compilador de C pels AVR genera un codi Assemblador molt eficaç i optimitzat. Per contra, per d'altres microcontroladors, el codi generat és més llarg i per tant menys eficaç.

Dos dels entorns més importants per desenvolupar aplicacions per Arduino són:

1. L'entorn de desenvolupament específic d'Arduino (Arduino IDE), que permet treballar amb C++. Aquest entorn és programari lliure.
2. Usar el conjunt d'eines estàndards de GNU per a l'arquitectura AVR. Aquestes eines inclouen el compilador (C, C++, Ada), el muntador, l'assemblador i les eines per gestionar binaris habituals (gestor de llibreries, convertidors de formats, etc.).

En aquest curs optarem per la segona opció ja que en primer lloc ofereix millor control del que s'està fent, també és la mateixa eina de desenvolupament d'aplicacions per d'altres arquitectures i permet una millor comprensió del elements que intervenen en el procés.

La descripció detallada d'aquest conjunt d'eines, les fases que intervenen en el procés de compilació a partir del codi font fins generar el codi binari, així com el llenguatge C pròpiament dit es veuran més detalladament en assignatures posteriors. En aquest curs es descriurà el procés de

compilació a partir del codi Assemblador per generar el codi binari en una màquina diferent al mateix AVR i el procés de transferència d'aquest codi binari cap a l'Arduino.

L'assignatura de dispositius programables està basada exclusivament en el llenguatge Assemblador. El Toolchain de GNU permet treballar directament en aquest nivell. Per altra banda, Microchip (actual propietària d'Atmel) ofereix unes eines per treballar en Assemblador específiques [AtmAss], però la seva especificitat i falta de suport fa que sigui descartada.

Anomenem Assemblador a l'eina que converteix llenguatge Assemblador a codi binari. Tots els Assembladors utilitzen les mateixes instruccions mnemòniques, però algunes funcionalitats poden ser diferents. Per tant s'ha de preveure una possible adaptació entre projectes realitzats amb diferents Assembladors.

L'eina que permet controlar les transferències entre la màquina que conté l'entorn de desenvolupament (el nostre ordinador personal) i els dispositius AVR és *avrdude*. Aquesta eina també és de codi lliure.

### 2.2.1 Instal · lació

Per instal · lar aquestes eines en un sistema Linux, distribució GNU/Debian o Ubuntu cal executar l'ordre:

```
apt install gcc-avr binutils-avr avr-libc avrdude
```

### 2.2.2 Eines importants

Les comandes més rellevants que s'utilitzaran són:

**avr-gcc** És el *front-end* principal de les eines de compilació. S'invoca tant per compilar codi C com per compilar codi Assemblador. Pot realitzar totes les tasques relacionades amb la compilació de forma automàtica fins obtenir directament el codi binari. Si el codi font és Assemblador en realitat la comanda executada és **avr-as** ja que és l'assemblador pròpiament dit.

**avr-objcopy** És una aplicació que permet canviar de format els executables. Habitualment l'usarem per canviar el format *elf* (format estàndard de UNIX), a *hex,* que és el format necessari per carregar el programa en el microcontrolador.

**avr-objdump** Aquesta aplicació, entre altres coses, permet a partir del format executable *elf* obtenir el codi des-assemblat.

**avrdude** És el que ens permet transferir un programa, generalment en format *hex*, [ihex], al microcontrolador.

## 3 Llenguatge Assemblador

La programació basada en Assemblador consta essencialment de la crida seqüencial de les instruccions màquina (en forma de mnemònic) definides per al microprocessador amb els paràmetres adequats i d'un conjunt de directives que determinen alguna característica concreta del codi. Aquesta estructura seqüencial respon al fet de que el programa serà emmagatzemat en una memòria amb una disposició igualment seqüencial.

El joc d'instruccions màquina corresponent als AVR [Instr] s'anirà introduint a les pràctiques en funció de les necessitats. De la mateixa manera, s'aniran introduint les directives de l'Assemblador del GNU [AS].

## 4 Programació

A continuació teniu l'estructura bàsica d'un programa dins el fitxer a.s.

```
.global main

main:
    nop
    rjmp main
```

Amb aquest programa farem diverses proves amb el Toolchain de GNU per tal de passar aquest codi al microcontrolador.

```
avr-gcc -mmcu=atmega328p -o a.elf a.s

avr-objcopy --output-target=ihex a.elf a.hex

avrdude -c arduino -P /dev/ttyACM0 -p m328p -U flash:w:a.hex:i
```

### 4.1 El primer programa: actuar sobre una sortida

El següent codi posa a valor lògic '0' i '1' alternativament els 8 bits del PORTB. En un d'aquests pins (pin 5 del PORTB de l'AVR, mapejat al pin 13 de l'Arduino UNO) tenim el LED de color taronja de la placa de desenvolupament Arduino.

```
.global main

/* Això és un comentari
de més d'una línia */

/* Comença el programa principal */
main:
    ;; configurem tots els bits del PORT B com a sortides
    ldi 16,0xFF
    out 0x4,16
loop:
    ;; assignem a tots els bits del PORT B un valor 0
    ldi 16,0x00
    out 0x05,16
    ;; assignem a tots els bits del PORT B un valor 1
    ldi 16,0xFF
    out 0x05,16
    rjmp loop
```

Exercici 1 Calculeu el temps en què el LED està en ON i en OFF.

Exercici 2 Useu l'oscil·loscopi per a verificar aquest resultat.

### 4.2 El segon programa: *define*

El següent codi és una variació del primer programa que usa *definicions*.

```
.global main

/* Això és un comentari
de més d'una línia */

/* Aquí es fan algunes definicions fent servir dues directives equivalents */
mregistre = 16
.set DDRB_o , 0x4
.equ PORTB_o , 0x5

/* Comença el programa principal */
main:
    ldi mregistre,0xFF
    out DDRB_o,mregistre
loop:
    ldi mregistre,0x00
    out PORTB_o,mregistre
    ldi mregistre,0xFF
    out PORTB_o,mregistre
    rjmp loop
```

Exercici 3 Ha canviat el temps en què el LED està en ON i en OFF?

### 4.3 El tercer programa: *toggle*

El següent codi és una variació del primer programa que en lloc de fixar el valor a '0' o '1' indica que *canviï* de valor (això es coneix com a *toggle*).

```
.global main

/* Aquí es fan algunes definicions fent servir tres directives equivalents */
    mregistre = 16
    .set DDRB_o , 0x4
    .equ PINB_o , 0x3

/* Comença el programa principal */
main:
    ldi mregistre,0xFF
    out DDRB_o,mregistre
loop:
    out PINB_o,mregistre
    rjmp loop
```

Exercici 4 Ha canviat el temps en què el LED està en ON i en OFF?

## Referències


[Ard] Arduino UNO; http://arduino.cc/en/Main/ArduinoBoardUno

[ATmega328p] ATmega328P datasheet; https://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061B.pdf

[AtmAss] Microchip Assembler documentation; http://ww1.microchip.com/downloads/en/DeviceDoc/40001917A.pdf

[Instr] Joc d'instruccions dels AVR; http://ww1.microchip.com/downloads/en/devicedoc/atmel-0856-avr-instruction-set-manual.pdf

[AS] Manual de referència del Assemblador del GNU; http://sourceware.org/binutils/docs/as/

[ihex] Format de fitxer Intel HEX; http://en.wikipedia.org/wiki/Intel_HEX