UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Enginyeria de Sistemes TIC

Comunicació sèrie

Dispositius Programables — Enginyeria de Sistemes TIC

Jordi Bonet Francisco del Águila

4 de desembre de 2022

Índex

1 Objectiu

L'objectiu d'aquesta pràctica és permetre la comunicació entre un Arduino i un ordinador personal usant el port sèrie.

2 Introducció teòrica

2.1 Comunicació sèrie asíncrona

Una de les formes que tenen els computadors per comunicar-se els uns amb els altres és amb un perifèric anomenat **USART** [**?**]. Aquest perifèric permet una comunicació de tipus sèrie. Usa dues línies de comunicació, una per a la recepció i l'altra per a la transmissió. Aquest tipus de comunicació generalment es fa en mode asíncron, és a dir, no hi ha cap línia amb un rellotge que reguli la comunicació. En una comunicació sèrie els bits (la informació) s'envien un rere l'altre. Per aquest motiu la quantitat de línies és mínima (en contrast amb una comunicació en paral·lel, en què cal una línia per bit).

En el cas de l'Arduino, la interfície de comunicació USART que es fa servir per comunicar-se amb l'ordinador personal està formada per la interfície USB. Així, la interfície USB queda configurada emulant el comportament d'una interfície USART. Per tant, en tenim prou amb interconnectar l'Arduino amb l'ordinador amb el mateix cable USB que fem servir per alimentar i per programar.

2.2 Programari de comunicació

L'ordinador que es connecta amb l'Arduino necessita d'un terminal que permeti la interacció entre l'usuari i el dispositiu amb el que es comunica. La funció d'aquest terminal és permetre que tot el que l'usuari tecleja sigui transmès al dispositiu connectat i també permet que allò que el dispositiu envia cap a l'ordinador pugui ser visualitzat. Una eina d'aquest tipus podria ser **picocom**.

Per instal · lar aquesta aplicació en una distribució tipus Debian s'ha d'executar la comanda

```
apt install picocom
```

Per saber com funciona aquesta aplicació es pot fer ús del seu manual: *man picocom*.

2.3 Mòdul USART

La majoria de microcontroladors AVR, i en particular l'ATmega328P que conté l'Arduino, utilitzen un mòdul USART que es pot configurar de diferents maneres. La descripció detallada d'aquest mòdul es troba a l'apartat 20 del manual de referència de l'ATmega328p [**?**].

La configuració que es fa servir per a la comunicació amb l'ordinador és en mode asíncron, amb un rellotge intern i sense habilitar el mode de multiprocessador.

La velocitat a la que poden anar els bits, així com el format del missatge, són paràmetres que es poden configurar tant en el costat de l'Arduino com per en el costat de l'ordinador. La comunicació serà possible si els dos dispositius tenen la mateixa configuració. En general aquesta configuració per defecte queda definida per 8 bits de dades, 1 bit de parada (*stop*) i sense paritat. Per tant, tret que es digui el contrari, aquesta serà la configuració que es farà servir tant a l'Arduino com en l'ordinador.

2.4 Codificació ASCII

Un estàndard molt simple per poder codificar en binari els caràcters d'un text és la codificació [**?**]. Aquesta codificació usa 7 bits per codificar 128 caràcters, un nombre insuficient per codificar, per exemple, els caràcters existents en llengües diferents de l'anglès. Per aquest motiu existeixen *extensions* de la taula ASCII que usen 8 bits per codificar 256 caràcters: els 128 de la taula ASCII i 128 més, diferents en cada extensió. Una d'aquestes extensions és la ISO-8859-1.

La comanda *man asccii* permet visualitzar la codificació dels 128 caràcters de la taula ASCII, i la comanda *man iso_8859-1* els caràcters de l'extensió ISO-8859-1.

La codificació binària que es fa servir per transmetre un caràcter a través d'una comunicació basada en un dispositiu USART és la codificació ASCII. Així, la transmissió de la lletra *A* queda codificada amb els bits *0b01000001*. Per cert, si voleu visualitzar aquests bits a l'oscil·loscopi, penseu que primer s'envia el bit de menor pes.

3 Exemple de programa

El següent programa, *p5-exemple.s*, implementa una comunicació sèrie de manera que l'Arduino es manté a l'espera fins que rep un byte. Quan rep aquest byte simplement es limita a tornar-lo a transmetre. Aquest tipus de comportament és el que s'anomena fer un eco.

```
.set DDRB_o , 0x4
.equ PORTB_o , 0x5
PORTD_o = 0x0b
DDRD_o = 0x0a

UDR0 = 0xc6
UBRR0H = 0xc5
UBRR0L = 0xc4
UCSR0C = Oxc2
UCSR0B = 0xC1
UCSR0A = 0xC0

.global main

/* rutina de recepció de bytes, el valor es recull al registre r16 */
```

```
rx: lds r16,UCSR0A
    sbrs r16,7
    rjmp rx
    lds r16,UDR0
    ret

/* rutina de transmissió de byte, el valor a transmetre està al registre r16 */
tx: lds  r17,UCSR0A
    sbrs r17,5
    rjmp tx
    sts UDR0,r16
    ret

main:
    /* set baud rate a 9600*/
    ldi r16, 0
    sts UBRR0H,r16
    ldi r16, 103
    sts UBRR0L,r16

    /* set frame format */
    /* tot i que el valor dels registres després d'un reset ja és correcte
    (asíncron, 8 bits de dades, 1 bit de parada, sense paritat,
    velocitat normal, comunicació no multiprocessor)
    assegurem aquesta configuració escrivint el valor als registres*/
    ldi r16, 0b00100000
    sts UCSR0A,r16

    ldi r16, 0b00000110
    sts UCSR0C,r16

    /* enable rx, tx, sense interrupcions */
    ldi r16, 0b00011000
    sts UCSR0B,r16

    /* configuració dels pins */
    ldi r16,0b00000010
    out DDRD_o,r16

loop:
    call rx
    call tx
    rjmp loop
```

4 Estudi previ

TASCA PRÈVIA 1 Llegiu detalladament l'apartat 20 de [?].

TASCA PRÈVIA 2 Descriviu detalladament què fa *p5-exemple.s*.

TASCA PRÈVIA 3 Modifiqueu *p5-exemple.s* de manera que la configuració de la comunicació passi de velocitat 9600 bps i 8 bits de dades sense paritat, a velocitat 115 Kbps i 7 bits de dades amb paritat. Anomeneu el fitxer modificat *p5-exemple-mod.s*.

TASCA PRÈVIA 4 Feu un programa que quan detecti que s'ha polsat la lletra *l* o *L* encengui el LED de l'Arduino i quan rebi qualsevol altre dada l'apagui. Useu la mateixa configuració del programa *p5-exemple.s*. Anomeneu aquest fitxer *p5-codi1.s*.

TASCA PRÈVIA 5 Feu un programa que quan detecti que s'ha polsat la lletra *n* o *N* respongui amb els caràcters corresponents als nombres 0,1,2,3,4,5,6,7,8,9. Quan es rebi qualsevol altre valor, respongui amb el caràcter *N*. Anomeneu aquest fitxer *p5-codi2.s*.

TASCA PRÈVIA 6 Feu un programa que quan detecti exactament la seqüència de lletres *led* encengui el LED i quan rebi qualsevol altre seqüència l'apagui. La resposta de l'Arduino cap al ordinador a cada pulsació hauria de ser el nombre de lletres que queden per teclejar fins aconseguir la seqüència *led*. Implementeu aquest programa com una màquina d'estats: definiu els estats, dibuixeu el diagrama d'estats i declareu qui mantindrà el valor de l'estat del sistema. Anomeneu aquest fitxer *p5-codi3.s*.

5 Treball pràctic

El treball al laboratori consisteix en la comprovació de les tasques de l'estudi previ. A continuació teniu algunes comandes del *Toolchain* de GNU, imprescindibles per passar els codis 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
```

5.1 Tasques

TASCA 7 Assembleu *p5-exemple.s* i comproveu el seu funcionament.

TASCA 8 Comproveu el correcte funcionament de *p5-codi1.s*.

TASCA 9 Comproveu el correcte funcionament de *p5-codi2.s*.

TASCA 10 Comproveu el correcte funcionament de *p5-codi3.s*.

Referències

[USART] Dispositiu USART; `http://en.wikipedia.org/wiki/Universal_asynchronous_receiver/transmitter`

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

[ASCII] Taula de caràcters ASCII; `http://en.wikipedia.org/wiki/ASCII`