UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Guió de la pràctica al laboratori

Projecte
Computadors encastats a l'automòbil

(Annex)

1. Objectius

Durant aquesta sessió de laboratori es demana que demostreu el correcte funcionament del sistema d'assistència a l'aparcament que heu desenvolupat al vostre treball previ. També haureu d'implementar una interfície de comunicacions 1-wire utilitzant la tècnica de *bit-banging*.
Per últim, es demana que estructureu el vostre programa per tal d'integrar els diversos sistemes en un únic microcontrolador.

2. El bus 1-WIRE

1-wire és un protocol de comunicacions sèrie dissenyat per Dallas Semiconductor. Està basat en un bus amb un *master* i un o diversos *slaves* a una única línia de dades. Òbviament, es necessita una referència de tensió comuna per a tots els dispositius (GND).

- RESET o Inicialització

Totes les comunicacions en el bus 1-Wire comencen amb una seqüència de "Reset" i "presència" que permet al dispositiu que actua com a mestre sincronitzar tots els dispositius esclaus del bus. Es provoca un reset quan el dispositiu mestre col·loca la línia de dades a '0' durant uns 480µs, a continuació allibera el bus durant 60µs. Els dispositius esclaus indicaran la seva presencia col·locant la línia de dades a '0' durant un temps entre 60µs i 240µs.

- Lectura / Escriptura en el bus

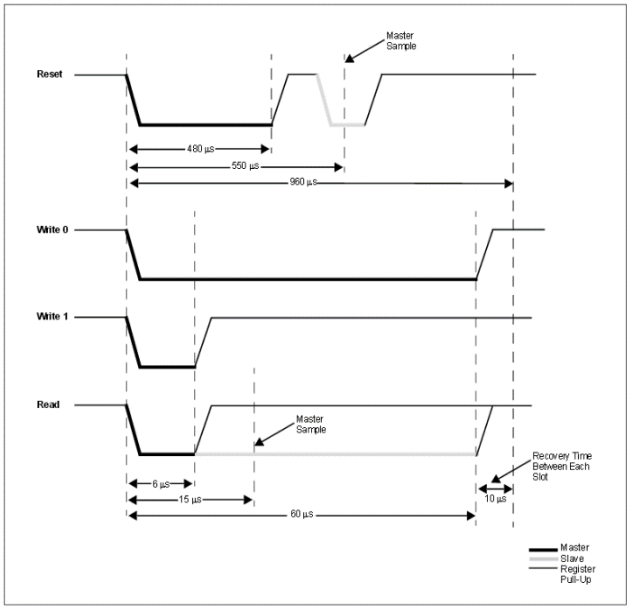
L'escriptura i la lectura de dades en el bus 1-Wire es fa mitjançant "slots" de temps. La generació d'aquests slots és responsabilitat del master. Quan el mestre vol llegir un bit d'informació ha de forçar la línia de dades a "0" durant 6µs, a continuació allibera el bus durant 9µs i fa una lectura de l'estat de la línia (el bit que ha transmès l'esclau). Per tancar el temps de l'slot farà una espera de 55µs.

El sistema per efectuar una escriptura d'un bit per part del master és similar. Si volem escriure un "0", el master col·locarà la línia a "0" durant 60µs, alliberarà el bus i a continuació farà una espera de 10µs. Si el que vol el master és escriure un "1", col·locarà la línia a "0" durant 6µs, alliberarà el bus i a continuació farà una espera de 60µs.
En la següent taula i figura es descriuen les operacions que heu d'implementar i el cronograma associat a cadascuna d'elles.

TABLE 1: 1-Wire® OPERATIONS

Operation	Description	Implementation
Reset	Reset the 1-Wire bus slave devices and get them ready for a command.	Drive bus low, delay 480 µs. Release bus, delay 70 µs. Sample bus: 0 = device(s) present, 1 = no device present Delay 410 µs.
Write 0 bit	Send '0' bit to the 1-Wire slaves (Write 0 slot time).	Drive bus low, delay 60 µs. Release bus, delay 10 µs.
Write 1 bit	Send '1' bit to the 1-Wire slaves (Write 1 slot time).	Drive bus low, delay 6 µs. Release bus, delay 64 µs.
Read bit	Read a bit from the 1-Wire slaves (Read time slot).	Drive bus low, delay 6 µs. Release bus, delay 9 µs. Sample bus to read bit from slave. Delay 55 µs.

FIGURE 2: 1-Wire® TIMING DIAGRAM



Per a la lectura de la temperatura podeu utilitzar la següent funció:

```
float Read_Temperature_DS1820(void)
{
        int temp;

        Reset();                    // master resets bus
        WriteByte(0xCC);            // skip ROM
        WriteByte(0x44);            // convert temperature

        Reset();                    // master resets bus
        WriteByte(0xCC);            // skip ROM(nomes un dispositiu)
        WriteByte(0xBE);            // read scratch pad
                                    // read 2 bytes from scratchpad
        temp= ReadByte();
        temp = (ReadByte() << 8) + temp; // sign bit set, temp is negative

       return (((float)temp)/2);
}
```

3. Cicle de control

Els microcontroladors estan dissenyats per a reduir el cost econòmic i el consum energètic d'un sistema en particular. Per això sovint un microcontrolador ha de controlar diversos processos. Programar un cicle de control ens pot ajudar a realitzar les diferents tasques, tant síncrones com asíncrones, de forma endreçada i sense bloquejar el flux d'execució d'una tasca per culpa d'una altra.
El *cicle de control* és una tècnica en la que programem un timer per obtenir una interrupció periòdica (la nostra base de temps). Mentre no es produeix la interrupció el microcontrolador realitza les diferents tasques **asíncrones**, com per exemple consultar pulsacions de botons per enquesta. Una vegada s'ha produït la interrupció passem a executar les tasques **periòdiques** que s'han d'executar cada n*base_de_temps.

```
unsigned int decimes=0; // comptador de decimes de segon transcorregudes

void interrupt rsi ( ) {
        if  (TMR0IE && TMR0IF) {
                TMR0= valor; //valor per obtenir una interrupció cada 100ms
                TMR0IF=0;
                decimes++;
                inici_cicle=1;
        }
}

void main ( ) {
        init_pic(); //inicialitzacions
        init_timer(); //Configurem el timer0 per generar una interrupció cada 100ms
        while(1) {
                //Arribem a aquest punt aprox cada 100ms
                if (decimes%2==0) {
                        //Arribem a aquest punt aprox cada 200ms
                }
                if (decimes%3==0)
                {       //Arribem a aquest punt aprox cada 300ms
                }
                inici_cicle=0;
                while (!inici_cicle) {
                        //Processos asíncrons
                }
        }
```

4. Pràctica al laboratori

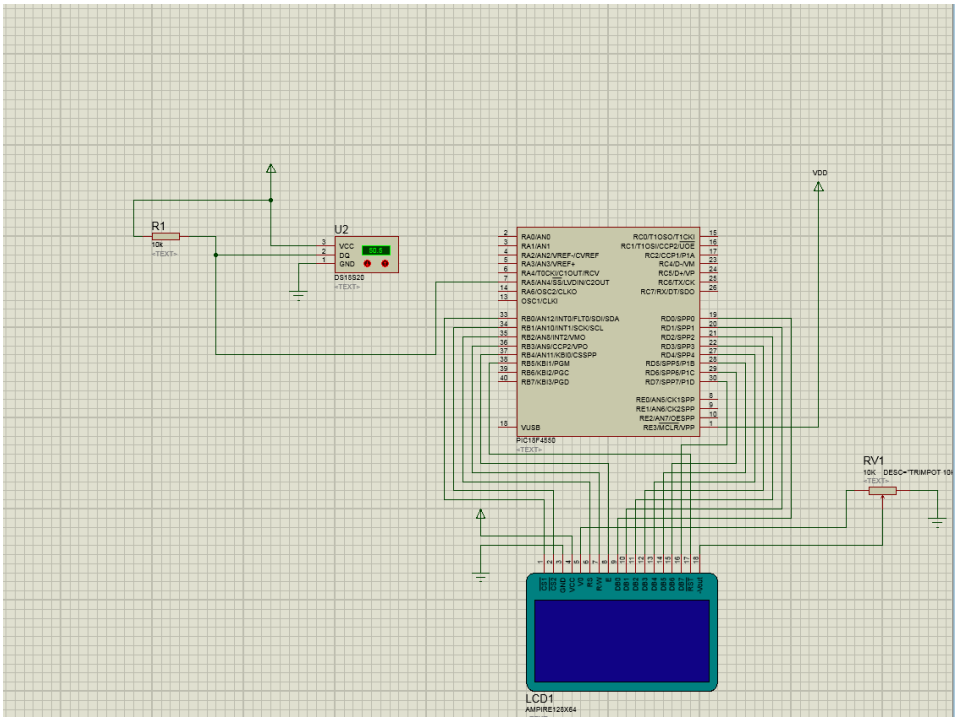
El treball a realitzar al laboratori consta dels següents apartats:

1) Mostrar el correcte funcionament del sensor d'aparcament sobre PROTEUS.

2) Comprovar el funcionament del programa realitzat, ara sobre la placa EASYPIC.

3) Implementeu la comunicació 1-wire amb el sensor de temperatura DS1820 present a la placa EasyPic.

3.1) Implementeu l'esquema electrònic sobre Proteus.



3.2) Realitzar un programa que en intervals d'1 segon es comuniqui amb el sensor de temperatura DS1820 (9 bits de resolució), implementant parcialment la comunicació sèrie asíncrona 1-Wire. El resultat de la lectura s'escriurà en el GLCD. La comunicació es farà mitjançant el pin d'entrada/sortida digital RA5 del microcontrolador. Cal implementar les següents funcions:

unsigned char Reset (void); // Genera un reset al dispositiu i indica si hi ha o no presència del dispositiu retornant 0: presència, 1: no presència

void WriteByte (unsigned char b); // Envia el byte b bit a bit pel pin RA5 segons el protocol 1wire.

unsigned char ReadByte(void); // Rep un byte del dispositiu pel pin RA5 segons el protocol 1wire.

El professor us facilitarà la informació necessària així com un esquelet de codi per a la implementació del protocol.

3.3) Implementeu un sistema de climatització (control de temperatura a l'habitacle del vehicle). Per fer-ho imaginarem que tenim un ventilador connectat a una sortida controlada mitjançant PWM (mòdul CCP). També tindrem dos botons que ens permetran triar la temperatura consigna a la que volem que es trobi l'habitacle. Si la temperatura que llegim del sensor és més alta que la temperatura consigna activarem el ventilador (imaginari) mitjançant el PWM. La consigna de PWM la podeu calcular mitjançant la següent fòrmula:

$$DUTY\_PWM\ (\%) = (Temp\_sensor - Temp\_Consigna) * 25$$

On DUTY_PWM és el % de duty cycle del PWM i com a màxim pot valer 100.

Visualitzarem la sortida de PWM fent servir l'oscil·loscopi i un led a la EasyPic.

4) Ara que tenim diversos sistemes (E/S, GLCD, sensor de temperatura 1-wire, sensor de distància AD, CCP) ens serà útil integrar totes les funcionalitats en una estructura tipus "cicle de control". Per fer-ho programeu un cicle de control fent ús del **Timer3** generant una interrupció periòdica cada 10ms. Executeu els processos asíncrons fora del cicle de control i sincronitzeu tots aquells processos periòdics. Podeu fer servir l'esquelet de codi facilitat en aquest document com a punt de partida.

MOLT IMPORTANT: A la propera sessió es continuarà treballant i fent créixer les funcionalitats del vostre codi. Sigueu curosos i genereu codi reaprofitable doncs us facilitarà molt la feina per a la propera sessió.