Departamento de Engenharia Eletrotécnica

Sistemas de Aquisição de Dados

2º Semestre 2018/2019

Relatório 2º Trabalho Prático

Chapéu de sol inteligente

Trabalho Realizado por:

Diogo Lograd nº 43697

Rui Martins nº 45174

Conteúdo

Introdução

Neste relatório vamos falar sobre os objetivos deste trabalho, o funcionamento, a implementação e as conclusões obtidas.

Vamos utilizar a placa Explorer 16 mais especificamente o PIC24FJ128GA010 sendo o objetivo criar um sistema de um chapéu de sol inteligente.

Objetivo

Neste trabalho queremos criar um chapéu de sol que consiga “procurar” o sol de modo a que o utilizador não esteja exposto. Utilizando sensores LDR vamos verificar de que lado o sol se encontra. O sensor com a leitura mais alta vai ser do lado para que o motor vai rodar.

Vai ainda existir um modo que o utilizador pode escolher onde o objetivo é estar exposto ao sol (“bronzear”). Um botão é utlizado para alternar entre estes modos.

Temos ainda um sensor de temperatura que quando é atingido um certo valor é acionada uma ventoinha. A velocidade desta ventoinha é controlada pelo potenciómetro.

O sistema só pode ser controlado por utilizadores que saibam a password. Para ativar o chapéu de sol tem que se carregar num botão e depois inserir uma password de 4 letras, previamente definida, através da porta UART. Se a password estiver correta o sistema fica ativo.

Funcionamento

1. Quando o botão B4 for premido deve-se inserir a password que vai ser enviada para a UART. O sistema vai verificar a password e se estiver errada o chapéu de sol continua parado e o utilizador é notificado, se estiver certa o sistema começa a funcionar.
2. O sistema tem dois modos, “normal” e “bronzear”. No modo “normal” o sistema vai procurar o lado de onde está o sol e posiciona o chapéu através do uso de um motor. Quando o chapéu estiver na posição correta o utilizador é notificado através da UART.
3. Feedback do sistema é necessária. O utilizador vai poder saber os valores dos sensores LDR, temperatura, estado da ventoinha e a sua velocidade (potenciómetro). Estes valores são transmitidos para a UART (“T” – temperatura, “P” – potenciómetro, “L” – sensores LDR).
4. Quando o sensor de temperatura tiver valores acima dos 30ºC, a ventoinha é ligada com uma velocidade definida pelo potenciómetro e o utilizador é notificado através da UART.
5. No modo “bronzear”, que pode ser ativado pelo botão B2, o chapéu vai deixar de tapar o sol e deixa o utilizador exposto. Se se pretender voltar ao modo “normal” carrega-se no botão B2.

Implementação

Inicialização ADC

Esta função vai inicializar o registo PCFG (p.10 ADC) com as portas analógicas usadas neste trabalho deixando o registo CSSL (p.10 ADC) a zero visto que não pretendemos fazer scan. (AD1CON p.6 ADC).

```
void inicializar_adc(){

     AD1PCFG = 0xFFD1;     //1FFD1    //todos os registos
necessarios para o potenciometros, LDR's e temperatura, em
hexadecimal (AN1, AN2, AN5, AN3)
     AD1CON1 = 0x0000; // SAMP bit = 0 ends sampling
     // and starts converting
     AD1CSSL = 0;     // in this example AN2 is the input
     //TRISAbits.TRISA0 = 0;    //output

     AD1CON1bits.ADON = 1; // turn ADC ON
}
```

Inicialização UART

É preciso definir o Baud Rate a utilizar, neste caso será 9600, e no datasheet diz para colocar U2BRG (p.9 UART) a 25, valor este que depende da frequência do oscilador da placa. Foi ainda colocado no modo de 8 bits para poder tratar dos dados enviados. Colocou-se então o U2MODE (p.3 UART) com o valor de 0x8000.

```
void inicializar_UART(){
      U2BRG = 25; //Valor calculado atraves da formula
presente no datasheet, para um baud rate de 9600
      U2STA = 0;
      U2MODE = 0x8000; //Enable Uart for 8-bit data
      //no parity, 1 STOP bit
      U2STAbits.UTXEN = 1; //Enable Transmit
}
```

Colocar caracter e string na UART

Funções que colocam caracteres/string na UART.

```
void putCharUART( char c) {
      int i;
//    while( U2STAbits.UTXBF);     //UTXBF = registo que
determina se o buffer ta cheio ou nao. Se for = 1, o buffer
esta cheio e nao se pode escrever
                                                  // se for = 0, o
buffer nao esta cheio e pode escrever-se pelo menos um
caracter
      U2TXREG = c;                //buffer onde se mete o
caracter escolhido. TXREG = transmission register

      for(i=0;i<2000; i++);    //é necessário ter este for que
serve como delay, sem este for o ecrã só recebe lixo, pois é
rápido demais
}

 void putStringUART( char *str ) {          //*str ->
 apontador para a string str

       int i, string_lenght = strlen(str);        //strlen =
 tamanho da string

       for(i=0; i<string_lenght; i++){
            putCharUART( *str );
            *str++;
       }
 for( i = 0 ; i < 2000 ; i++){}; //delay
 }
```

Potenciómetro

Nesta função temos como retorno o valor do potenciómetro. O valor do potenciómetro representa a velocidade atual da ventoinha.

```
int valor_potenciometro(){
      int i = 0;

      AD1CHS = 5; // Connect AN5 as CH0 input

      AD1CON1bits.SAMP = 1; // start sampling...
      for( i = 0 ; i < 20000 ; i++){};//delay
      AD1CON1bits.SAMP = 0; // start Converting
      while (!AD1CON1bits.DONE); // conversion done?
      return ADC1BUF0; // yes then get ADC value  (1-1024)
}
```

Sensores LDR

Os sensores LDR vão nos dizer de que lado temos leituras mais altas. A função devolve o valor lido pelo sensor variando entre 1 e 1024.

```
int valor_LDR1(){
      int i = 0;

      AD1CHS = 1; // Connect AN1 as CH0 input

      AD1CON1bits.SAMP = 1; // start sampling...
      for( i = 0 ; i < 20000 ; i++){};//delay
      AD1CON1bits.SAMP = 0; // start Converting
      while (!AD1CON1bits.DONE); // conversion done?
      return ADC1BUF0; // yes then get ADC value  (1-1024)
}
```

Sensor de temperatura

Função que devolve o valor da temperatura atual.

```
int sensor_temperatura(){
int i = 0;


      AD1CHS = 0x0002; // Connect AN2 as CH0 input

      AD1CON1bits.SAMP = 1; // start sampling...
      for( i = 0 ; i < 20000 ; i++){};//delay
      AD1CON1bits.SAMP = 0; // start Converting
      while (!AD1CON1bits.DONE); // conversion done?
      return ADC1BUF0; // yes then get ADC value  (1-1024)
}
```

Modo “normal”

Sistema procura a posição do sol comparando os valores dos sensores e movendo o motor para o lado com leituras mais altas.

```
void modo_normal(int sensor_esquerda, int sensor_direita){


      int i;

      if(sensor_esquerda > sensor_direita){
            //move o motor para a esquerda

            LATAbits.LATA0 = 1;              //    PORTAbits.RA0
= 1;
            LATAbits.LATA1 = 0;         //    PORTAbits.RA1 = 0;
            for( i = 0 ; i < 20000 ; i++){};
            LATAbits.LATA0 = 0;              //    PORTAbits.RA0
= 0;
            LATAbits.LATA1 = 0;              //    PORTAbits.RA1
= 0;
      }
      else{
            //move o motor para a direita
            LATAbits.LATA0 = 0;              //    PORTAbits.RA0
= 0;
            LATAbits.LATA1 = 1;              //    PORTAbits.RA1
= 1;
            for( i = 0 ; i < 20000 ; i++){};
            LATAbits.LATA0 = 0;              //    PORTAbits.RA0
= 0;
            LATAbits.LATA1 = 0;              //    PORTAbits.RA1
= 0;
      }
}
```

Modo “bronzear”

Ao contrário do modo “normal” vai mover o motor para o lado com valores mais baixos, para a pessoa se bronzear.

```
void modo_bronzear(int sensor_esquerda, int sensor_direita){


      int i;

      if(sensor_esquerda > sensor_direita){
            //move motor para a direita
            LATAbits.LATA0 = 0;              //    PORTAbits.RA0
= 0;
            LATAbits.LATA1 = 1;              //    PORTAbits.RA1
= 1;
            for( i = 0 ; i < 20000 ; i++){};
            LATAbits.LATA0 = 0;              //    PORTAbits.RA0
= 0;
            LATAbits.LATA1 = 0;              //    PORTAbits.RA1
= 0;
      }
      else{
            //move motor para a esquerda
            LATAbits.LATA0 = 1;              //    PORTAbits.RA0
= 1;
            LATAbits.LATA1 = 0;        //    PORTAbits.RA1 = 0;
            for( i = 0 ; i < 20000 ; i++){};
            LATAbits.LATA0 = 0;              //    PORTAbits.RA0
= 0;
            LATAbits.LATA1 = 0;              //    PORTAbits.RA1
= 0;
      }
}
```

Conclusão

Com este trabalho conseguimos aprender aspetos importantes da placa Explorer 16 e conhecer melhor as formas de comunicação entre dispositivos.

Conseguimos criar um sistema com várias funcionalidades onde o utilizador consegue controlar os modos de funcionamento e tem conhecimento dos valores medidos pelos sensores e dos vários componentes do projeto.