NOVA SCHOOL OF
SCIENCE & TECHNOLOGY
Departamento de Engenharia Eletrotécnica e de Computadores

Sistemas de Aquisição de Dados

Trabalho prático Nº3

C L A S S I F I C A Ç Ã O :	

Trabalho realizado pelos alunos:

Nome: Tiago Manuel Morais Zorro Número: 57390

Nome: Tiago Alexandre Nunes Lopes Número: 57733

Nome: Francisco Abade dos Santos Número:57901

Mestrado Integrado Em Engenharia Eletrotécnica e de Computadores

ANO LETIVO 2022 / 2023

Índice

Índice de Figuras

I. Requisitos do dispositivo

O terceiro trabalho prático da unidade curricular de Sistemas de Aquisição de Dados (SAD), tem como objetivo continuar o desenvolvimento do sistema de aquisição de dados desenvolvido no segundo trabalho, tendo-se adicionado mais componentes e funcionalidades.

O PIC24 utilizado anteriormente no segundo projeto, continuará também a ser usado para o terceiro trabalho como um componente que funciona como Dispositivo de Aquisição de Dados (DAD). Haverá uma componente com Sensores Digitais (SD), ligada ao Arduino, que irá ler os valores dos sensores e enviá-los através do protocolo I2C para o DAD. Nesta comunicação o SD irá comportar-se como um *slave* e o DAD como *master*.

Será também desenvolvida uma aplicação com interface gráfica para o computador, que permitirá com que o computador e o DAD interajam através de uma comunicação serial, de maneira a receber mensagens JSON de monitorização, enviar mensagens JSON de configuração e guardar os dados localmente num ficheiro. A aplicação também permitirá enviar os dados recebidos para um Servidor Remoto através do método HTTP POST, com mensagens XML.

II. Comportamento do dispositivo

PIC24

O Comportamento do sistema é controlado pelo DAD. Quando este é ativado ele irá realizar a inicialização da UART do I2C e do Timer. Depois das inicializações serem feitas ele irá entrar dentro do *loop*, onde se verifica se já passou tempo suficiente para realizar-se uma nova leitura dos sensores. Caso possa realizar as medições, o DAD irá pedir ao SD para lhe enviar os valores lidos. A comunicação entre o SD e o DAD é feita através de protocolo I2C. Os valores recebidos pelos sensores são guardados. Depois verifica-se se o número de amostras feitas é igual ao número definido. Quando esta condição se verifica, é gerada e seguidamente enviada a mensagem de monitorização para a porta série onde o computador estará a escutar.

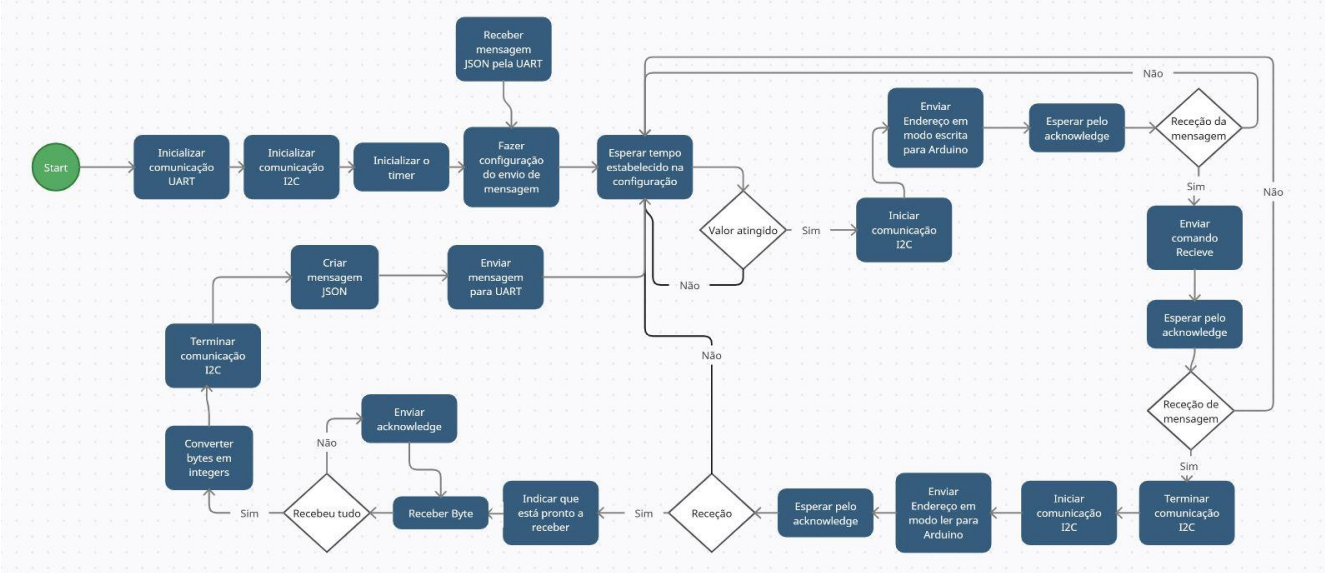

Figura 1 - Fluxograma com a lógica do PIC24

Arduino

O componente SD funciona como *slave* no nosso sistema, pelo que este apenas comunica com o DAD quando lhe é pedido. Quando este recebe o comando 0xAC (valor em hexadecimal), proveniente do Arduino, ele mede os valores dos sensores e envia-os para o DAD.

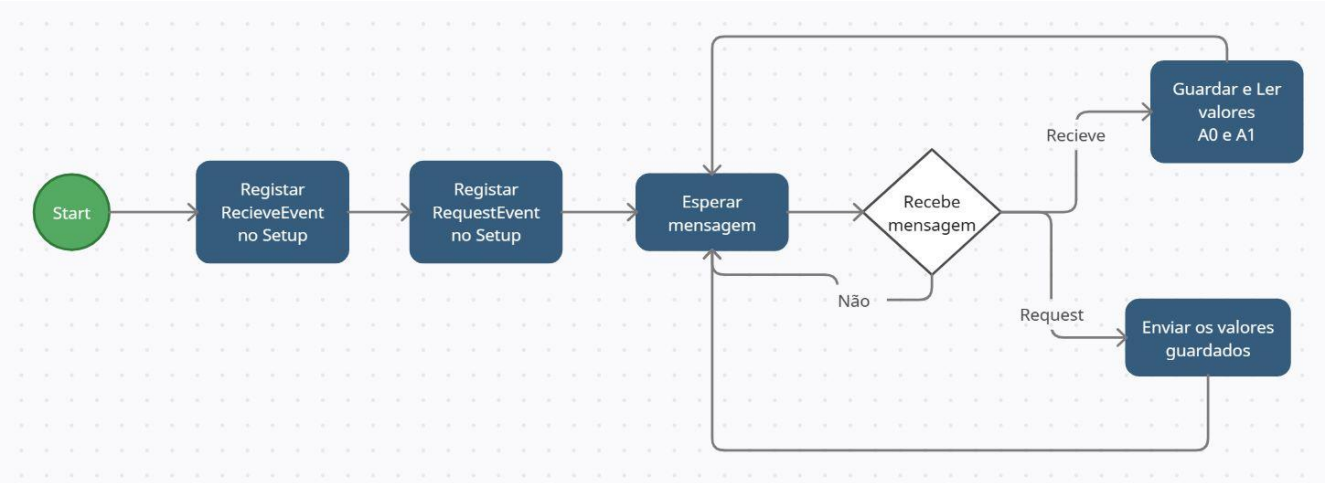

Figura 2 - Fluxograma com a lógica do Arduino

APP

Neste trabalho foi desenvolvida uma aplicação para o computador. Esta aplicação serve de interface gráfica entre o utilizador, o computador e o DAD. Quando a interface está a ser executada esta irá receber mensagens de monitorização pela porta COM do computador e armazenar as mensagens num ficheiro localmente. A aplicação também é responsável por enviar mensagens no formato XML, com os valores lidos para um servidor remoto. Caso o utilizador pretenda alterar as configurações do DAD, a interface está munida com caixas de seleção, que alteram os parâmetros quando utilizadas.

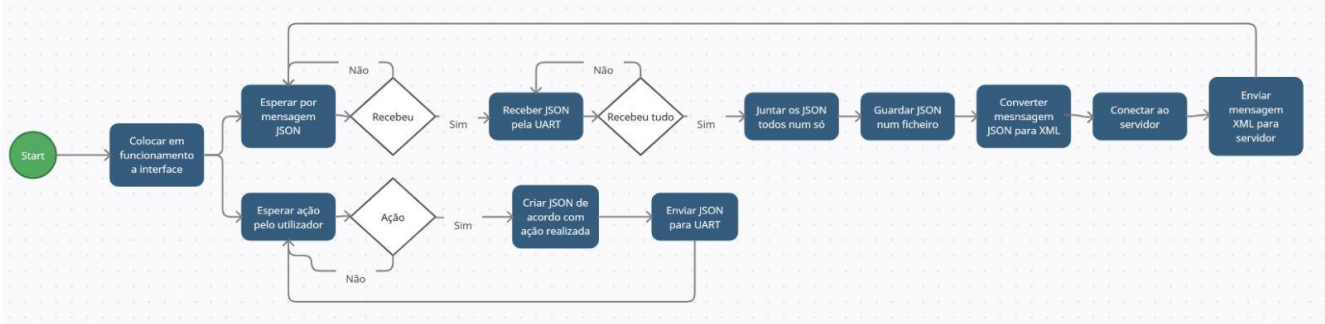

Figura 3 - Fluxograma com a lógica da APP

III. Descrição do dispositivo

PIC24

- ### I2C

De maneira a configurar o I2C do PIC24, no nosso trabalho temos uma função chamada de *I2C_init*. Esta função possui duas utilidades. Uma delas serve para habilitar o uso do I2C. O outro propósito é definir o *Baud Rate* da comunicação. No nosso caso como pretendemos utilizar um *Baud Rate* de 100 Hz enviamos o valor 39 (de acordo com o *datasheet* do PIC24), mas caso se pretendesse utilizar uma frequência diferente bastava alterar-se o número.

- ### UART

Para configurar os *interrupts* da recepção de dados pela UART, utilizamos a função *UART_init*, já adicionada no trabalho anterior. Nessa função adicionamos a funcionalidade da rotina de atendimento de *interrupts*. Quando o registo da UART recebe dados, a *flag* do estado do registro é ativada, o que vai despertar a função que trata do *interrupt.* Nessa função guardamos os dados num vetor. Quando chegamos ao final da mensagem, é chamada a função dedicada à alteração da configuração do dispositivo.

Arduino

Neste trabalho o Arduino implementa o SD. O SD e o DAD funcionam numa configuração *slave/master*. O SD quando pedido deve ler os valores dos sensores e enviá-los para o DAD. Para tal existem duas funções que implementam estas funcionalidades, o *receiveEvent* e o *requestEvent*. Quando estas funções não são chamadas, o Arduino fica a escrever os valores lidos anteriormente.

- ### *receiveEvent*

Quando o Arduino recebe uma transmissão do DAD, a função *receiveEvent* é chamada. Esta função recebe como parâmetro o número de bytes lidos pelo DAD, não retornando nada. Nesta função o Arduino quando receber o comando 0xAC irá ler e guardar os valores dos sensores.

- ### *requestEvent*

Quando o DAD pede para ler os dados do SD é chamada a função *requestEvent*. Nesta função o Arduino irá enviar para a linha de dados 5 informações: O endereço do dispositivo, o byte menos significativo do primeiro valor, o byte mais significativo do primeiro valor, o byte menos significativo do segundo valor, o byte mais significativo do segundo valor.

APP

Para este trabalho foi desenvolvida uma interface gráfica para o utilizador. Para tal foi usada uma *Windows Form App*, no Visual Studio. A app permite receber e enviar dados pela porta COM. Esta Porta COM é pré-definida e não pode ser alterada. Também permite armazenar dados num ficheiro localmente e fazer HTTP POST para enviar os dados para um servidor remoto.

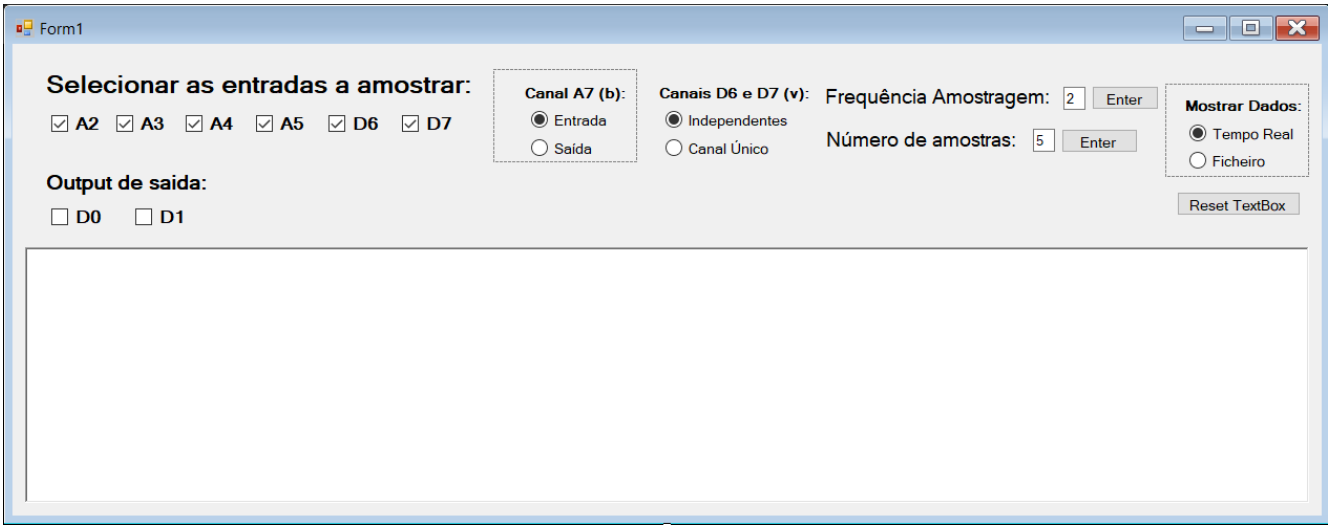

Figura 4 - Interface gráfica da aplicação desenvolvida

- ## Envio de dados

Para o trabalho anterior, quando se pretendia enviar dados do computador para o PIC24, escrevia-se mensagens JSON no monitor do Arduino IDE e através de uma comunicação serial estas eram enviadas para o PIC24. Para este trabalho a lógica mantém-se semelhante, continuando-se a enviar mensagens JSON com os parâmetros de configuração, mas em vez de se escrever as mensagens, apenas é necessário selecionar na interface as características que se pretende alterar. A lógica por detrás da aplicação cria a mensagem JSON e adiciona os parâmetros selecionados, enviando-as depois para o PIC24.

Quando a interface é lançada pela primeira vez, já vêm selecionadas as opções padrão definidas no trabalho anterior.

- ## Receção de dados

Tal como para o envio de dados no trabalho, o método usado da receção de dados continua a usar os mesmos princípios do trabalho anterior. A nossa aplicação vai receber as mensagens de monitorização do PIC24 pela porta COM do computador. Na interface gráfica da aplicação existe uma caixa de texto, que depois de recebermos a mensagem, irá mostrar a mensagem.

- ## Armazenamento de dados

A aplicação desenvolvida para este trabalho também possui a funcionalidade de armazenar os dados recebidos. Quando uma mensagem JSON é recebida pela porta COM do computador, esta é guardada localmente num ficheiro .json. Este ficheiro .json está localizado numa diretoria pré-definida no código e não pode ser alterado durante a execução do trabalho.

- ## Envio para o servidor

Neste trabalho era pedido para enviar os dados recolhidos pelo SD para um servidor remoto, através de uma aplicação para o PC. Os dados têm de ser enviados numa mensagem XML para o servidor remoto. Na aplicação desenvolvida pelo nosso grupo, esta funcionalidade é executada quando recebemos a mensagem. A mensagem vem num formato JSON. Pelo que é necessário processar a mensagem e convertê-la num formato XML. É também adicionado o número de aluno dos elementos do grupo à mensagem de maneira a ser mais fácil distinguir das outras mensagens no servidor. Depois de obtermos a mensagem no formato XML esta é enviada para o servidor por um HTTP POST.

IV. Testes

Depois de obtermos o sistema montado e programado foi necessário testar as funcionalidades implementadas. Para tal recorremos aos seguintes testes de maneira a garantir que o dispositivo estava bem programado e a funcionar de acordo com os requisitos.

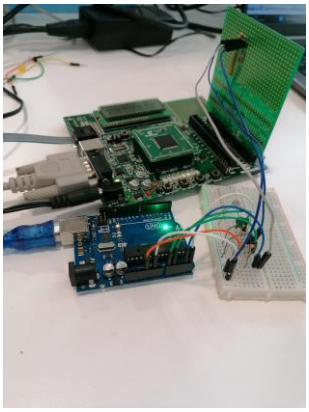

Figura 5 - Montagem física dos componentes

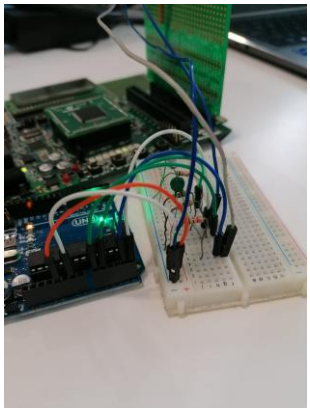

Figura 6 - Ligações do I2C entre os SD e o DAD e dos sensores do SD

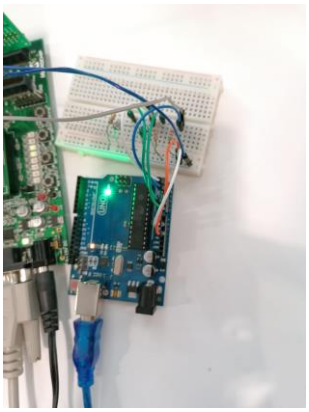

Figura 7 - Implementação física do SD

Figura 8 - Implementação física do DAD

Primeiro começamos por testar o funcionamento da interface gráfica. Pelo que podemos observar das duas mensagens abaixo, a interface gráfica está a funcionar. Apresenta as várias caixas de seleção para escolhermos as configurações dos canais que pretendemos amostrar. Também envia mensagens de monitorização com os valores lidos.

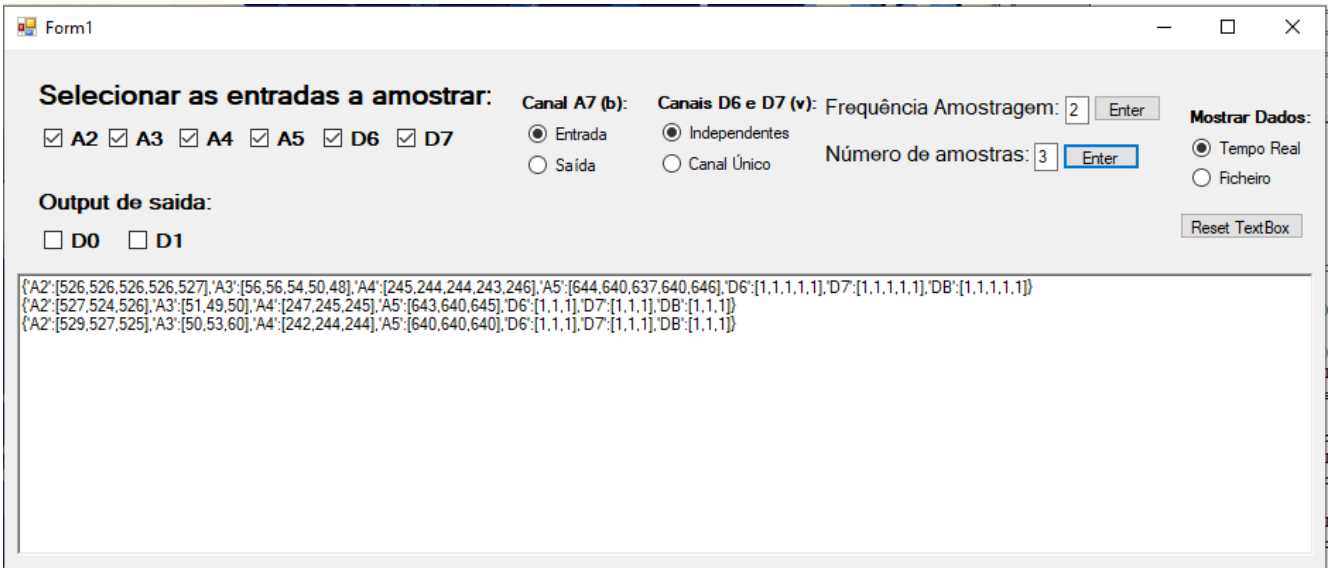

Figura 9 - Interface da aplicação com mensagens de monitorização

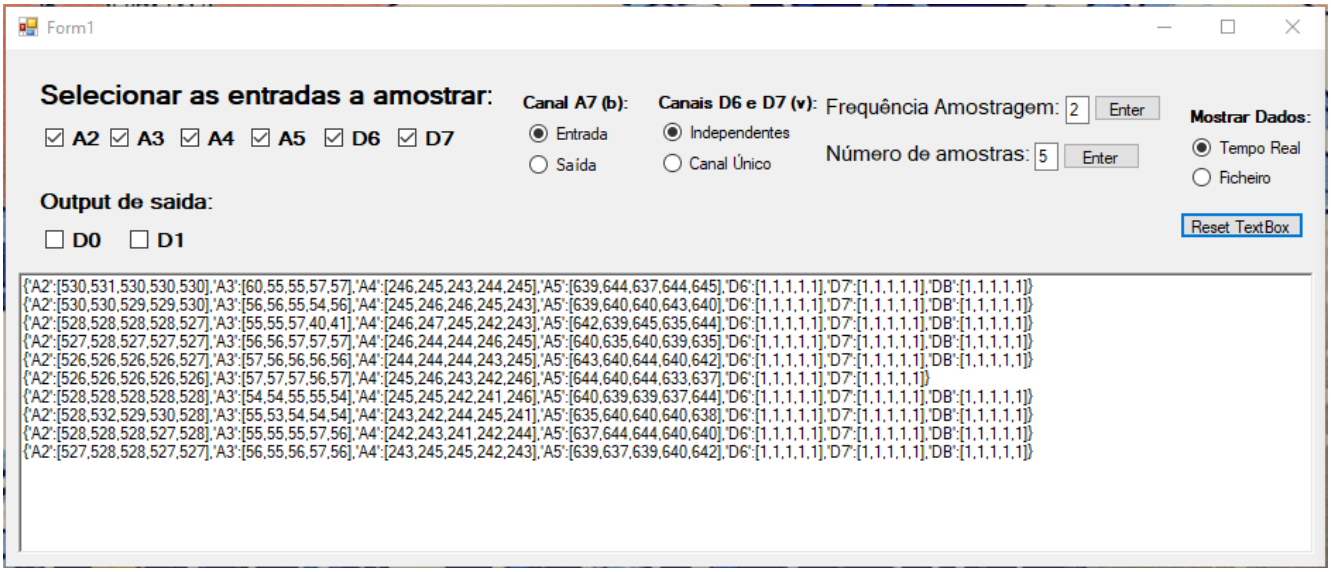

Figura 10 - Interface da aplicação com mensagens de monitorização

Em seguida testamos a funcionalidade do envio de dados do computador através da nossa aplicação para o DAD. Para tal decidimos remover as entradas A4 e A5. Com estes testes estávamos à espera de que nas próximas mensagens enviadas seriam enviadas as mesmas entradas que havia na mensagem anterior excepto estas duas. Ao observarmos a imagem abaixo conseguimos ver que foi precisamente isso que aconteceu.

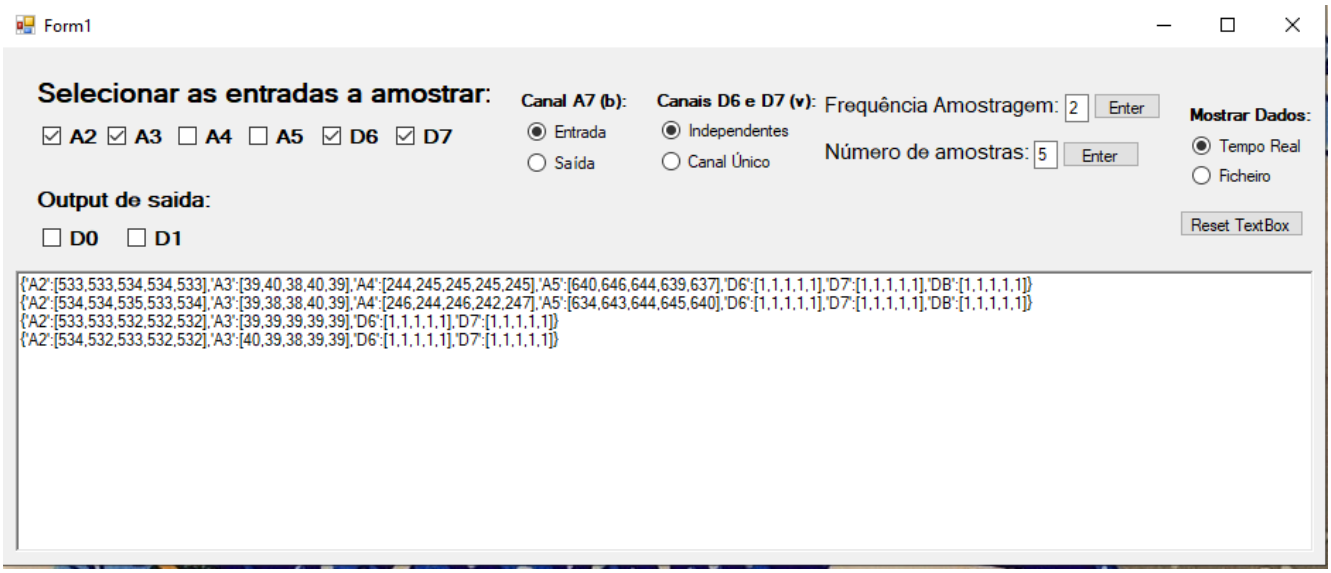

Figura 11 - Alteração das entradas de amostragem

Em seguida realizamos uma variante do teste anterior. Nesta em vez de tirarmos apenas duas entradas, fomos tirando várias aos poucos até termos uma mensagem vazia.

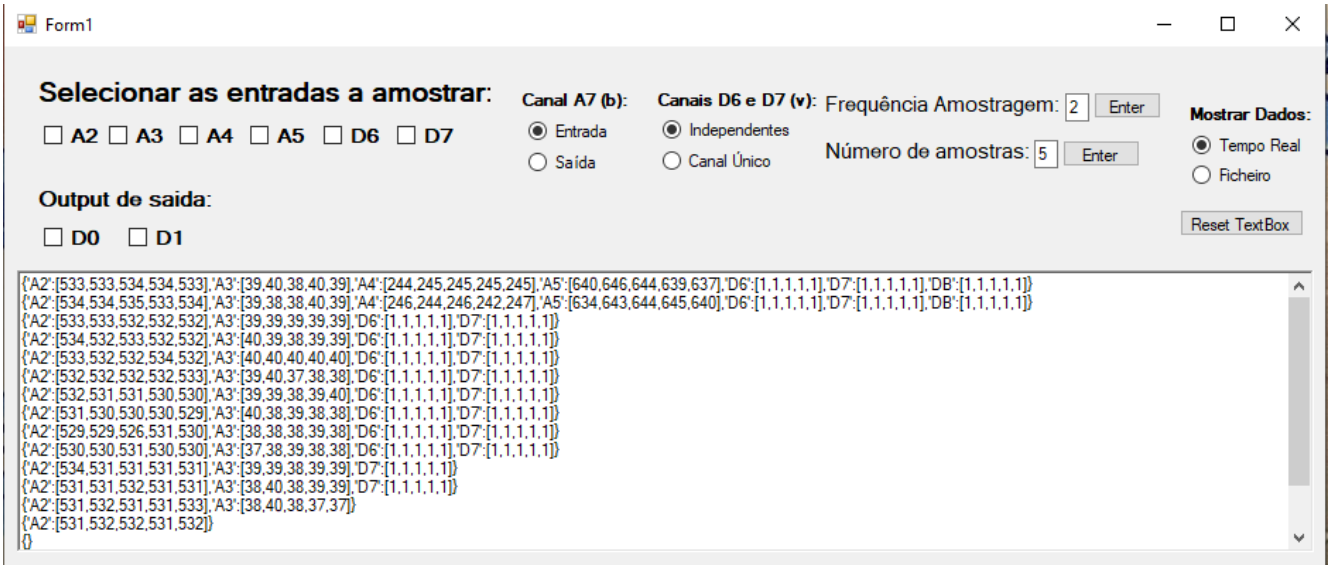

Figura 12 - Remoção das entradas de amostragem até se obter uma mensagem vazia

Ao ter-se verificado que se estava a receber corretamente as mensagens de monitorização, decidiu-se verificar se as mensagens estavam a ser corretamente recebidas no servidor remoto. Para tal acedeu-se a um ficheiro que guarda as mensagens recebidas pelo servidor nesse dia.

Ao observar a figura abaixo, verifica-se que à uma série de mensagens identificadas pelos números de aluno dos elementos do grupo" <57390_57901_57733>...</57390_57901_57733>". As mensagens aparecem no formato XML, com os valores lidos pelos sensores, tal como se esperava.

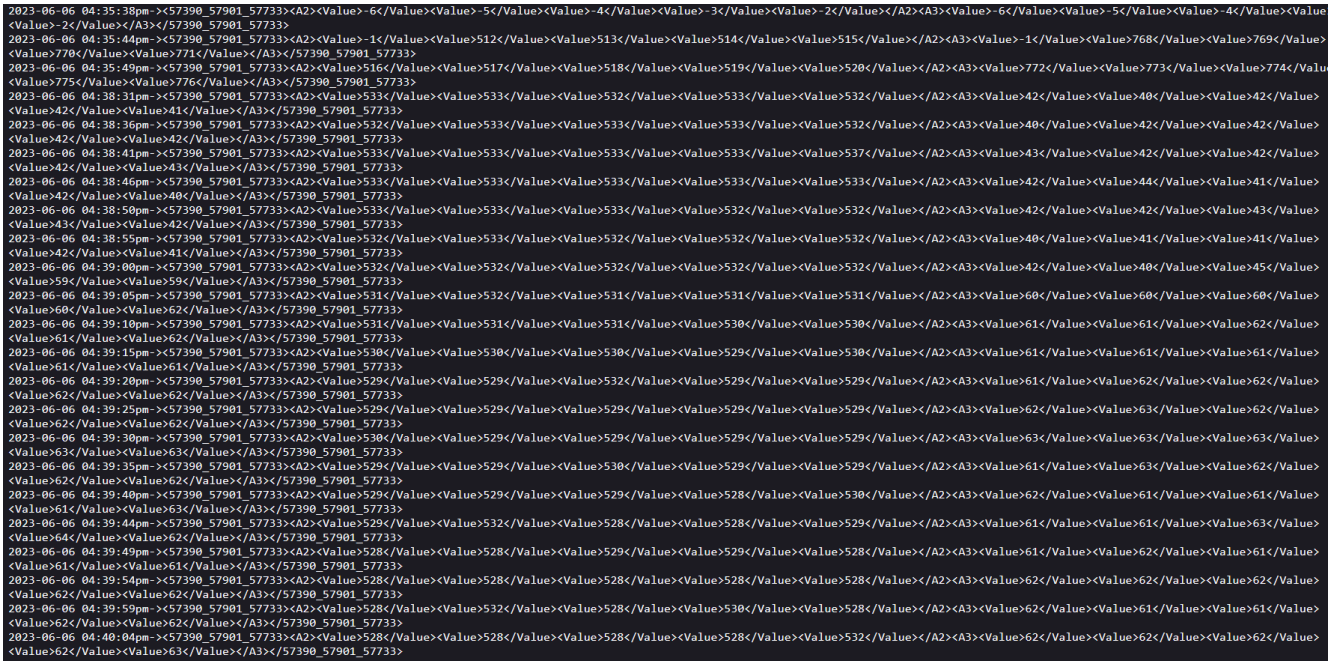

Figura 13 - Mensagens em formato XML no servidor remoto

Também foi testada a funcionalidade de guardar as mensagens recebidas pelo computador num ficheiro local. Para testar esta funcionalidade selecionou-se a opção "Ficheiro" na parte "Mostrar Dados:" da interface. Quando se selecciona esta opção, a aplicação vai escrever toda a informação que guardou no ficheiro, na caixa de texto. Ao observar-se a figura abaixo, podemos ver que a aplicação guardou com sucesso as mensagens num ficheiro local e também conseguiu exibir os dados armazenados na interface gráfica.

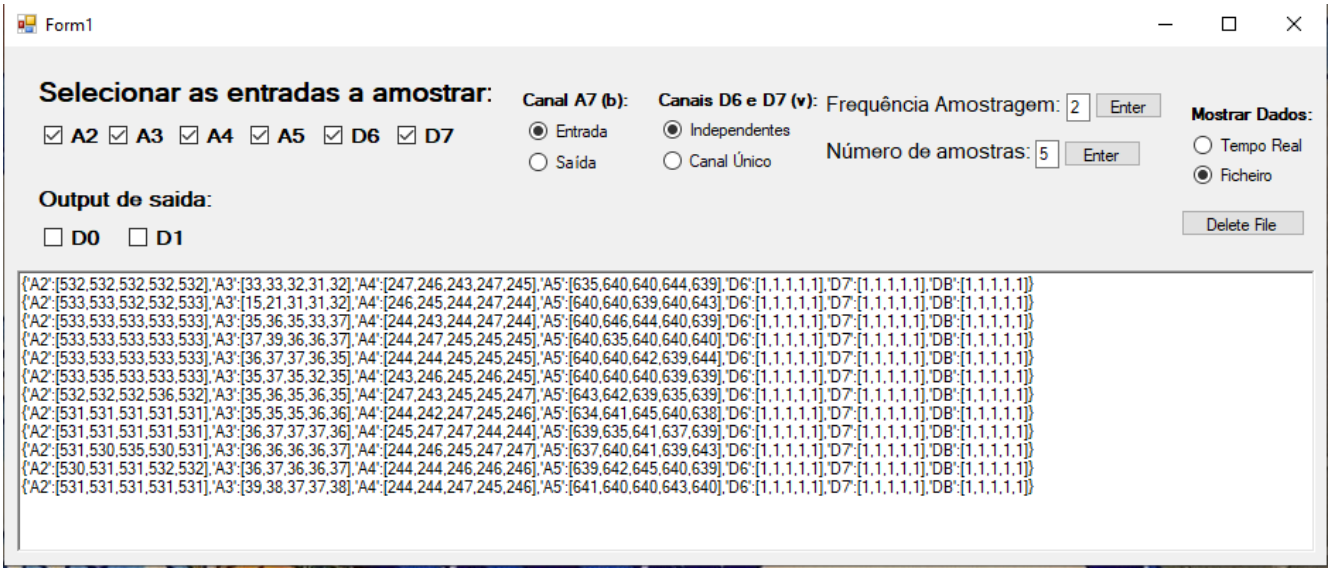

Figura 14 - Mensagens de monitorização guardadas no ficheiro local, a serem visualizadas na app

Para finalizar fez-se um teste para verificar se os valores lidos pelo Arduino eram os mesmos que chegavam às mensagens de monitorização da aplicação. De maneira a realizar-se este teste, usámos o monitor serial do Arduino IDE para verificar os valores lidos pelo SD. Se os valores lidos pelo Arduino forem os mesmos recebidos nas mensagens na aplicação então não há nenhum problema no envio destas. Ao observarmos a imagem abaixo conseguimos ver que os valores lidos pelo arduino são os mesmos que aparecem nas mensagens.

Figura 15 - Valores lidos no Arduino, juntamente com as mensagens de monitorização

V. Conclusões

Este trabalho teve como objetivo, a continuação do desenvolvimento do trabalho anterior sobre um sistema de aquisição de dados. Pretendia-se adicionar um módulo de Sensor Digital, implementado por um Arduino e uma aplicação para PC, que permita receber e enviar mensagens do DAD para o computador via RS-232. A aplicação também necessitava de conseguir enviar mensagens em XML para um servidor remoto.

O objetivo geral deste trabalho foi cumprido, pois o dispositivo de aquisição de dados consegue comunicar com o sensor digital através do protocolo I2C. E a aplicação desenvolvida possui todas as especificações pedidas, o que se pode comprovar pelos resultados obtidos na secção de testes do relatório.

Na realização do nosso trabalho achamos interessante a nossa implementação da aplicação. Decidimos criar uma interface gráfica, em vez de uma interface por consola. Que permite facilitar o utilizador na escolha dos parâmetros de configuração, não tendo de escrever a mensagem, apenas necessitando de se selecionar os atributos a mudar.

Um dos problemas que detetámos no nosso trabalho, que ficou por resolver do trabalho anterior, foi a troca do período de amostragem com a frequência de amostragem na condição que verifica a passagem do tempo antes da amostragem. Causando os problemas referidos no relatório do trabalho anterior. Uma forma de resolver este problema seria inverter o valor da frequência de amostragem recebida das mensagens de configuração, sem que assim tivéssemos de enviar diretamente o período de amostragem através dessas mensagens, obtendo-se assim o comportamento desejado. Ou seja, como está implementado através da aplicação com interface gráfica é necessário enviar-se o período de amostragem na caixa destinada à frequência de amostragem, e não a própria frequência de amostragem (como deveria de ter sido feito de acordo com o pedido pelo enunciado). No entanto, consideramos que tendo o utilizador conhecimento deste aspeto ao utilizar a aplicação com interface gráfica, o funcionamento de todo o sistema acaba por não ser comprometido.

Um dos aspetos que poderíamos ter melhorado neste trabalho seria, por exemplo, adicionar a funcionalidade à aplicação de permitir escolher a porta COM do PC, pois com a configuração que possui atualmente a porta COM está definida *hardcoded*. Num contexto semelhante, também poderíamos adicionar a opção do utilizador escolher qual a diretoria onde se pretenderia guardar o ficheiro .json com as mensagens recolhidas. Também achamos que seria interessante abordar o armazenamento dos valores lidos das mensagens de monitorização de maneira diferente. No nosso trabalho estamos a guardar as mensagens de monitorização num ficheiro localmente, mas poderíamos ter utilizado uma base de dados para guardar os valores.

De salientar que os aspetos a melhorar referidos no último parágrafo acabam por dar alguma inspiração para trabalhos futuros relacionados com os dois trabalhos desenvolvidos nesta disciplina.

VI. Anexos

Código do PIC24

```
#include <p24fxxxx.h>
#include <stdio.h>
#include <string.h>

#define MAX_SAMPLES 100
#define MAX_ANALOG 4
#define MAX_CHANNELS 8
//------------------------Global Variables------------------
int counter=0;
int seconds=0;
int MessageEnded =0;
int n=0;
char Message[100];
//----------------------Funciton Declaration-----------------------------
void ADC_init();
void LED_init();
void UART_init();
void Timer_init();
void I2C_init();
int DigitalOutputA0_Conf(char * str);
int DigitalOutputA1_Conf(char * str);
void digitalOutputA0_On();
void digitalOutputA0_Off();
void digitalOutputA1_On();
void digitalOutputA1_Off();
void bidiretionalOutput_On();
void bidiretionalOutput_Off();
void readString(char* str);
void sendString(char *str);
int ADC_read(int chanel);
void Channel_configuration(char * str,  int * inputs);
int Samples_number(char * str);
int Frequency_configuration(char * str);
int Virtual_channel(char * str);
int Bidiretional_channel(char * str);
void Configuration(int * channels_to_sample, int *frequency, int *samples_num, int *virtual_ch, int *bidiretional_ch);
int readDigitalChannel();
void I2CReceive();
void I2CSend();
// Configuration Bits
#ifdef __PIC24FJ64GA004__ //Defined by MPLAB when using 24FJ64GA004 device
_CONFIG1( JTAGEN_OFF & GCP_OFF & GWRP_OFF & COE_OFF & FWDTEN_OFF & ICS_PGx1 & IOL1WAY_ON)
```

```
_CONFIG2( FCKSM_CSDCMD & OSCIOFNC_OFF & POSCMOD_HS & FNOSC_PRI & I2C1SEL_SEC)
#else
_CONFIG1( JTAGEN_OFF & GCP_OFF & GWRP_OFF & COE_OFF & FWDTEN_OFF & ICS_PGx2)
_CONFIG2( FCKSM_CSDCMD & OSCIOFNC_OFF & POSCMOD_HS & FNOSC_PRI)
#endif


int main(void){

   int lastSeconds = 0;
   int j=0, k=0;
   int virgula = 0;;

   int samples_counter=0;

   int A2[MAX_SAMPLES];
   int A3[MAX_SAMPLES];
   int A4[MAX_SAMPLES];
   int A5[MAX_SAMPLES];
   int D6[MAX_SAMPLES];
   int D7[MAX_SAMPLES];
   int DB[MAX_SAMPLES];
   int DV[MAX_SAMPLES];

   char str[1000];
   char buffer[100];
   char num_buffer[100];

   //Default Configuration
   int frequency = 1;
   int samples_num = 5;
   int virtual_ch= 0;
   int bidiretional_ch = 1;
   int channels_to_sample[] = {1,1,1,1,1,1,0};
   //{"A2":1, "A3":1, "A4":1, "A5":1, "D6":1, "D7":1, "DB":1, "DV":0}

   int A2Value;
   int A3Value;

   ADC_init();
   LED_init();
   UART_init();
   Timer_init();
   I2C_init();

   while(1){

         int a,b;
```

```
	for (a=0;a<2000;a++)
		for (b=0;b<300;b++);

	if ( !PORTDbits.RD13 ){
		/*sendString("Please configure the following parameter of the device:\n(If copying from the paper -> \"\" <- won't work. Need to write them manually\n ");
		sendString(" - Channels to Sample,\n - Frequency,\n - Number of Samples,\n - Virtual Channel,\n - Bidirectional Channel\n -> ");

		Configuration(channels_to_sample, &frequency, &samples_num, &virtual_ch, &bidiretional_ch);  */
	}

	if(MessageEnded == 1){
		n=0;
		MessageEnded = 0;

		Configuration(channels_to_sample, &frequency, &samples_num, &virtual_ch, &bidiretional_ch);
	}

	if ( (seconds - lastSeconds) >= frequency ){
			I2CSend();
			I2CReceive(&A2Value,&A3Value);

			if(channels_to_sample[0] == 1){
				A2[samples_counter]  = A2Value;
			}
			if(channels_to_sample[1] == 1){
				A3[samples_counter]  = A3Value;
			}
			if(channels_to_sample[2] == 1){
				A4[samples_counter]  =ADC_read(0x0004);
			}
			if(channels_to_sample[3] == 1){
				A5[samples_counter]  =ADC_read(0x0005);
			}
			if(channels_to_sample[4] == 1 && virtual_ch == 0){
				D6[samples_counter] = PORTDbits.RD6;
			}
			if(channels_to_sample[5] == 1 && virtual_ch == 0){
				D7[samples_counter] = PORTDbits.RD7;
			}
			if(channels_to_sample[6] == 1 && bidiretional_ch == 1){
				DB[samples_counter] = PORTAbits.RA7;
			}
			if(channels_to_sample[7] == 1 && virtual_ch == 1){
				DV[samples_counter] = readDigitalChannel();
```

```
                }

        samples_counter++;

        lastSeconds = seconds;
}

// enviar apenas quando todos os samples estão registados
if(samples_counter==samples_num){
        virgula = 0;
        strcpy(str,"{");

        for(j=0;j< MAX_CHANNELS;j++){
                if(channels_to_sample[j] == 1){

                        if (virgula == 1) {
                strcat(str, ",");
                }

                        if((j+2) == 6 || (j+2) == 7 || (j+2) == 8 || (j+2) == 9){
                                if((j+2) == 8 ){
                                        sprintf(buffer,"'DB':[");
                                        strcat(str,buffer);
                                }
                                if((j+2) == 9 ){
                                        sprintf(buffer,"'DV':[");
                                        strcat(str,buffer);
                                }
                                if((j+2) == 6 ){
                                        sprintf(buffer,"'D6':[");
                                        strcat(str,buffer);
                                }
                                if((j+2) == 7 ){
                                        sprintf(buffer,"'D7':[");
                                        strcat(str,buffer);
                                }
                        }else{
                                sprintf(buffer,"'A%d':[",(j+2));
                                strcat(str,buffer);
                        }

                        int arrayCpy[100];

                        switch(j+2){
                                case 2:    memcpy(arrayCpy,A2, sizeof(A2));
                                                break;
                                case 3: memcpy(arrayCpy,A3, sizeof(A3));
                                                break;
```

```
                                        case 4: memcpy(arrayCpy,A4, sizeof(A4));
                                                break;
                                        case 5: memcpy(arrayCpy,A5, sizeof(A5));
                                                break;
                                        case 6: memcpy(arrayCpy,D6, sizeof(D6));
                                                break;
                                        case 7:    memcpy(arrayCpy,D7, sizeof(D7));
                                                break;
                                        case 8: memcpy(arrayCpy,DB, sizeof(DB));
                                                break;
                                        case 9: memcpy(arrayCpy,DV, sizeof(DV));
                                                break;
                                }
                                for(k=0;k<samples_num;k++){

                                        sprintf(num_buffer, "%d",arrayCpy[k]);
                                        strcat(str,num_buffer);
                                if (k < samples_num - 1) {
                                strcat(str, ",");
                                }
                                }

                                strcat(str, "]");
                                virgula = 1;
                        }
                }
                strcat(str, "}\n");
                sendString(str);
//              sendString("\n");

                for(j = 0;j < samples_num; j++){
                        A2[j] = 0;
                        A3[j] = 0;
                        A4[j] = 0;
                        A5[j] = 0;
                        D6[j] = 0;
                        D7[j] = 0;
                        DB[j] = 0;
                        DV[j] = 0;
                }
                samples_counter=0;
        }
   }
}
/*
*Function: Configuration - Reads the message from the user and processes it. It also applys the changes to the system
*
```

```
*Input: Recieves the channels_to_sample,frequency, samples_num, virtual_ch, *bidiretional_ch
*
*Output:None
*/
void Configuration(int *channels_to_sample, int *frequency, int *samples_num, int *virtual_ch, int *bidiretional_ch){

/*    sendString("Entered in Configuration \n");

   sendString(Message);*/

   switch (Message[2]){
        case'A':Channel_configuration(Message, channels_to_sample);
                        *bidiretional_ch = channels_to_sample[6];

                        if(*bidiretional_ch == 0){
                                TRISAbits.TRISA7 = 0;
                                bidiretionalOutput_On();
                        }else{
                                TRISAbits.TRISA7 = 1;
                                bidiretionalOutput_Off();
                        }

                        *virtual_ch = channels_to_sample[7];

                        if(*virtual_ch ==1){
                                channels_to_sample[4] = 0;
                                channels_to_sample[5] = 0;
                        }

                        break;

        case'f':*frequency = Frequency_configuration(Message);
                        break;

        case'n':*samples_num = Samples_number(Message);
                        break;

        case'v':*virtual_ch = Virtual_channel(Message);
                        channels_to_sample[7] = *virtual_ch;

                        if(*virtual_ch ==1){
                                channels_to_sample[4] = 0;
                                channels_to_sample[5] = 0;
                        }
                        if(*virtual_ch ==0){
                                channels_to_sample[4] = 1;
                                channels_to_sample[5] = 1;
```

```
                        }
                        break;

        case'b':*bidiretional_ch = Bidiretional_channel(Message);
                        channels_to_sample[6] = *bidiretional_ch;

                        if(*bidiretional_ch == 0){
                                TRISAbits.TRISA7 = 0;
                                bidiretionalOutput_On();
                        }else{
                                TRISAbits.TRISA7 = 1;
                                bidiretionalOutput_Off();
                        }
                        break;

        case'D': if(Message[3] == '0' ){
                                if(DigitalOutputA0_Conf(Message)){
                                        digitalOutputA0_On();
                                }else{
                                        digitalOutputA0_Off();
                                }
                         }

                         if(Message[3] == '1'){
                                if(DigitalOutputA1_Conf(Message)){
                                        digitalOutputA1_On();
                                }else{
                                        digitalOutputA1_Off();
                                }
                         }
   }
}
/*
*Function: DigitalOutputA0_Conf - Reads and processes the message that configures the digital output A0
*
*Input: The string to be read
*
*Output: returns the values that defines if the digital output is ative or inactive
*/
int DigitalOutputA0_Conf(char * str){
   int input;

sscanf(str,"{\"D0\":%d}",&input);

   return input;
}
/*
*Function: DigitalOutputA1_Conf - Reads and processes the message that configures the digital output A1
```

```
*
*Input: The string to be read
*
*Output: returns the values that defines if the digital output is ative or inactive
*/
int DigitalOutputA1_Conf(char * str){
   int input;

sscanf(str,"{\"D1\":%d}",&input);

   return input;
}
/*
*Function: readDigitalChannel - Reads the individual inputs from digital input 6 and 7 and returns a int values
                                                                    with the 2 bit channel outpu
*
*Input:None
*
*Output: 0, 1, 2, 3 as ints for the virtual channel output
*/
int readDigitalChannel(){
   int number=0;
   int biValues[2]={0,0};

   biValues[0]=PORTDbits.RD6;
   biValues[1]=PORTDbits.RD7;

   if(biValues[0] == 0 && biValues[1] == 0 ){
         number = 3;
   }
   if(biValues[0] == 0 && biValues[1] == 1 ){
         number = 2;
   }
   if(biValues[0] == 1 && biValues[1] == 0 ){
         number = 1;
   }
   if(biValues[0] == 1 && biValues[1] == 1 ){
         number = 0;
   }

   return number;
}
/*
*Function: Channel_configuration - Reads and processes the message that configures the channels to read
*
*Input:The string to be processed and where the input channels to read are going to be stored
*
*Output:None
```

```
*/
void Channel_configuration(char * str, int * inputs){
   //sendString("Entered in Channel Configuration \n");
sscanf(str, "{\"A2\":%d, \"A3\":%d, \"A4\":%d, \"A5\":%d, \"D6\":%d, \"D7\":%d, \"DB\":%d, \"DV\":%d}",
          &inputs[0], &inputs[1], &inputs[2], &inputs[3], &inputs[4], &inputs[5], &inputs[6],&inputs[7]);
}
/*
*Function: Frequency_configuration - Reads and processes the message that configures the frequency
*
*Input: The string to be read
*
*Output: the number of the frequency
*/
int Frequency_configuration(char * str){
   int input;

sscanf(str,"{\"f\":%d}",&input);

   return input;
}
/*
*Function: Samples_number - Reads and processes the message that configures the numeber of samples
*
*Input:The string to be read
*
*Output:the numbers to sample
*/
int Samples_number(char * str){
   int input;

sscanf(str, "{\"n\":%d}",&input);

   return input;
}
/*
*Function: Virtual_channel - Reads and processes the message that configures the virtual channel
*
*Input:The string to be read
*
*Output: 1 or 0 that defines of the channel is an input or an output
*/
int Virtual_channel(char * str){
   int input;

sscanf(str, "{\"v\":%d}",&input);

   return input;
}
```

```
/*
*Function: Bidirectional_channel-Reads and processes the message that configures the bidirectional channel
*
*Input: The string to be read
*
*Output: 1 or 0 that defines of the channel is an input or an output
*/
int Bidiretional_channel(char * str){
   int input;

sscanf(str, "{\"b\":%d}",&input);

   return input;
}

/*
*Function:Handles the interrupt off the timer. Increments the counter and the seconds
*
*Input:None
*
*Output:None
*/
void __attribute__((__interrupt__, __shadow__)) _T1Interrupt(void)
{
   //char str[10];
   counter++;
   if(counter==60){
           seconds++;
           //sprintf(str, "%d\n", seconds);
           //sendString(str);
           counter=0;
   }
   IFS0bits.T1IF = 0; //Reset Timer1 interrupt flag and Return from ISR
}

void __attribute__((__interrupt__, auto_psv)) _U2RXInterrupt(void)
{
   IFS1bits.U2RXIF = 0;

   Message[n] = U2RXREG;

   if(Message[n]  == '\n'){
           MessageEnded =1;
   }
   n++;
}

/*
```

```
*Function:Timer_init - Handles the initialization to use the timer and timer interrupt
*
*Input:None
*
*Output:None
*/
void Timer_init(){
   T1CON = 0x00; //Stops the Timer1 and reset control reg.
   TMR1 = 0x00; //Clear contents of the timer register
   PR1 = 0xFFFF; //Load the Period register with the value 0xFFFF
   IPC0bits.T1IP = 0x01; //Setup Timer1 interrupt for desired priority level
                                                  // (This example assigns level 1 priority)
   IFS0bits.T1IF = 0; //Clear the Timer1 interrupt status flag
   IEC0bits.T1IE = 1; //Enable Timer1 interrupts
   T1CONbits.TON = 1; //Start Timer1 with prescaler settings at 1:1 and
                                                 //clock source set to the internal instruction cycle
}

/*
*Function:LED_init - Handles the initialization to use the LEDs with index 0 and 1
*
*Input:None
*
*Output:None
*/

void LED_init(){
   TRISAbits.TRISA7 = 1;
   TRISDbits.TRISD6 = 1;
   TRISAbits.TRISA0 = 0;
   TRISAbits.TRISA1 = 0;
}

/*
*Function:UART_init - Handles the initialization to use the UART
*
*Input:None
*
*Output:None
*/
void UART_init(){
   U2BRG=25;               //Set Baudrate
   U2STA = 0;
   U2MODE = 0x8000;    //Enable Uart for 8-bit data
                     //no parity, 1 STOP bit
   U2STAbits.UTXEN = 1;  //Enable Transmit

   //receive interrups from pic
```

```
   IEC1bits.U2RXIE =1;
}

/*
*Function:ADC_init - Handels the initialization to use the ADC
*
*Input:None
*
*Output:None
*/
void ADC_init(){

   AD1PCFG = 0xFFC3; //ffc3 para o trabalho- quais as entradas analógicas a ler
   AD1CON1 = 0x0000;

   AD1CSSL = 0;
   AD1CON3 = 0x0002;
   AD1CON2 = 0;
   AD1CON1bits.ADON = 1;

}

/*
*Function: ADC_read - Reads the input from a give ADC chanel. Requires ADC initialization
*
*Input:Number of the channel in Hexa
*
*Output:The value read by the ADC as an int
*/
int ADC_read(int channel){
   int i;
   AD1CHS = channel; //canal que queremos ler

   AD1CON1bits.SAMP=1;
   for( i = 0 ; i < 1000 ; i++){};
   AD1CON1bits.SAMP=0;
   while (!AD1CON1bits.DONE) {};
   int    ADCValue=ADC1BUF0;

   return ADCValue;
}

/*
*Function: readString - Reads a string from UART. Requires UART initialization
*
*Input:String
*
*Output:None
```

```
*/

void readString(char* str) {
int i = 0;
char letra='a';

// Wait for a string to be received

// Read the string
        while(letra != '\n' && i < 99){

        // Wait for a character to be received
        if (U2STAbits.URXDA == 1) {

        // Read the character
        letra = U2RXREG;

        // Add the character to the string
        str[i] = letra;
        i++;
         }
   }

   str[i] = '\0';

}

/*
*Function:sendingStrings - Sends a String throught the UART N.2. Requires UART initialization
*
*Input:String
*
*Output:None
*/

void sendString(char *str){

   char *add = str;         //!!!!!!!Mudei de int para char*, ver se depois nao ha problema
   while (*str != '\0')
   {
         while (U2STAbits.UTXBF); // Wait for TX buffer to be empty
         U2TXREG = *str;          // Send character
         str++;             // Increment pointer to next character
   }
   str=add;
}

/*
```

```
*Function: digitalOutputA0_On - Turns on the digital Output A0
*
*Input:None
*
*Output:None
*/
void digitalOutputA0_On(){

   LATAbits.LATA0 = 1;

}

/*
*Function: digitalOutputA0_Off - Turns off the digital Output A0
*
*Input:None
*
*Output:None
*/
void digitalOutputA0_Off(){

   LATAbits.LATA0 = 0;

}
/*
*Function: digitalOutputA1_On - Turns on the digital Output A1
*
*Input:None
*
*Output:None
*/
void digitalOutputA1_On(){

   LATAbits.LATA1 = 1;

}
/*
*Function: digitalOutputA1_Off - Turns off the digital Output A1
*
*Input:None
*
*Output:None
*/
void digitalOutputA1_Off(){

   LATAbits.LATA1 = 0;

}
```

```
/*
*Function: bidiretionalOutput_On - Turns on the LED when the bidirectional channel is defined as output
*
*Input:None
*
*Output:None
*/
void bidiretionalOutput_On(){
   LATAbits.LATA7 = 1 ;
}
/*
*Function: bidiretionalOutput_Off - Turns off the LED when the bidirectional channel is defined as output
*
*Input:None
*
*Output:None
*/
void bidiretionalOutput_Off(){
   LATAbits.LATA7 = 0 ;
}


/*
*Function: I2CSend - Responsable for starting I2C communication. Sends the command "AC" for slave to read sensor values.
*                                    Requires I2C initialization
*Input:None
*
*Output:None
*/
void I2CSend(){

          I2C2CONbits.SEN=1;

          while(I2C2CONbits.SEN);//Esperara que a linha volte a zero para comecar a comunicacao

         I2C2TRN = (0x48<<1) + 0;//enviar address
          //enviar r/w bit juntamente com o address

          while(I2C2STATbits.TBF); //Verificamos de o o hardware acabou de enviar os dados

          while(I2C2STATbits.TRSTAT); //hardware clear at end of slave Acknowledge

          if( I2C2STATbits.ACKSTAT)//recebemos ack
          {
                    sendString("ACK not recieved.\n");
          }
  //    else
```

```
                //sendString("ACK recieved.\n");
        I2C2TRN=0xAC; //enviamos o commando

        while(I2C2STATbits.TBF); //Verificamos de o o hardware acabou de enviar os dados

        while(I2C2STATbits.TRSTAT); //hardware clear at end of slave Acknowledge

        if( I2C2STATbits.ACKSTAT)//recebemos ack
        {
                sendString("ACK not recieved.\n");
        }
        //else
        //    sendString("ACK recieved.\n");
        I2C2CONbits.PEN=1;
        while(I2C2CONbits.PEN)
        {};
}
/*
*Function: I2CReceive - Responsable for receiving sensor values from I2C communication.
*                                       Requires I2C initialization
*Input:None
*
*Output:None
*/
void I2CReceive(int *A2Value,int *A3Value){

        I2C2CONbits.SEN=1;

        while(I2C2CONbits.SEN);//Esperara que a linha volte a zero para comecar a comunicacao

       I2C2TRN = (0x48<<1) + 1;//enviar address
        //enviar r/w bit juntamente com o address

        while(I2C2STATbits.TBF); //Verificamos de o o hardware acabou de enviar os dados

        while(I2C2STATbits.TRSTAT); //hardware clear at end of slave Acknowledge

        if( I2C2STATbits.ACKSTAT)//recebemos ack
        {
                sendString("ACK not recieved.\n");
        }
  //    else
        //    sendString("ACK recieved.\n");


        char data[5]={0,0,0,0,0};
        int i;
        for(i=0;i<5;i++){
```

```
                I2C2CONbits.RCEN =1;
                while(!I2C2STATbits.RBF); //Verificamos de o o hardware acabou de enviar os dados

                data[i] = I2C2RCV;    //recebemos dados
                while(I2C2CONbits.RCEN);

                if(i<4)
                        I2C2CONbits.ACKDT = 0;
                else
                        I2C2CONbits.ACKDT = 1;
                I2C2CONbits.ACKEN = 1;
                while(I2C2CONbits.ACKEN);
        }

        I2C2CONbits.PEN=1;
        while(I2C2CONbits.PEN)
        {};

        char str3[10];
        /**sprintf(str3, "Data1_1: %d,%d\n",data[1], data[2]);
        sendString(str3);*/

        *A2Value= (data[1] << 8) | data[2];
        char str1[10];
        char str2[10];

        /*sprintf(str1, "Data1: %d\n",*A2Value);
        sendString(str1);*/

        *A3Value = (data[3] << 8) | data[4];

        /*sprintf(str2, "Data2: %d\n",*A3Value);
        sendString(str2);*/
}
/*
*Function: I2C_init - Handels the initialization to use the I2C communication.
*
*Input:None
*
*Output:None
*/
void I2C_init(){
   I2C2BRG=39; //Baud rate value set to 100 KHz
   I2C2CONbits.I2CEN =1; //I2C enable

}
```

Código da App

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Configuration;
using System.Data;
using System.Drawing;
using System.IO.Ports;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Xml.Linq;
using Newtonsoft.Json;  //adicionado
using Newtonsoft.Json.Linq; //adicionado
using System.Diagnostics;
using System.Threading;
// adicionado para por no servidor
using System.Net.Http;
using System.Xml;
using System.Text.RegularExpressions;

// adicionado para criar um ficheiro
using System.IO;



namespace WindowsFormsApp1
{
public partial class Form1 : Form
{
        //output
        private int digital0 = 0;
        private int digital1 = 0;
        //input
        private int analog2 = 1;
        private int analog3 = 1;
        private int analog4 = 1;
        private int analog5 = 1;
        private int digital6 = 1;
        private int digital7 = 1;
        private int Dbidirecional = 0;
        private int Dvirtual = 0;
        //other values
        private string frequency;
        private string samples;
```

```
private string printText = "";


public Form1()
{
InitializeComponent();
}

private void Form1_Load(object sender, EventArgs e)
{
serialPort1.BaudRate = 9600; // Set the baud rate to match your device configuration
serialPort1.Open();

// Register the event handler
serialPort1.DataReceived += serialPort1_DataReceived;
}


private void Configuration_input(object sender, EventArgs e)
{
CheckBox clickedCheckbox = (CheckBox)sender;
int i;
if (clickedCheckbox.Checked)
{   i = 1;
} else   {   i = 0;  }

switch (clickedCheckbox.Name)
{
case "checkBox1":
            analog2 = i;
            break;
case "checkBox2":
            analog3 = i;
            break;
case "checkBox3":
            analog4 = i;
            break;
case "checkBox4":
            analog5 = i;
            break;
case "checkBox5":
            digital6 = i;
            break;
case "checkBox6":
            digital7 = i;
            break;
case "checkBox7":
            Dbidirecional = i;
```

```
                break;
        case "checkBox8":
                Dvirtual = i;
                break;
        }


        string send = "{\"A2\":" + analog2 + ", \"A3\":" + analog3 + ", \"A4\":" + analog4 + ", \"A5\":" + analog5
        + ", \"D6\":" + digital6 + ", \"D7\":" + digital7 + ", \"DB\":" + Dbidirecional + ", \"DV\":" + Dvirtual +
"}"+'\n';

        richTextBox1.Text = send;
        serialPort1_DataSender(send);

        }

        private void checkBox7_CheckedChanged(object sender, EventArgs e)
        {
        CheckBox clickedCheckbox = (CheckBox)sender;
        if (clickedCheckbox.Checked)
        {
        digital0 = 1;
        }
        else
        {
        digital0 = 0;
        }
        string send = "{\"D0\":" + digital0 + "}" + '\n';
        richTextBox1.Text = send;
        serialPort1_DataSender(send);

        }

        private void radioButtonV_CheckedChanged(object sender, EventArgs e)
        {
        RadioButton selectedRadioButton = (RadioButton)sender;

        if (selectedRadioButton.Checked)
        {
        Dvirtual = 0;
        }
        else { Dvirtual = 1; }

        string send = "{\"v\":" + Dvirtual + "}" + '\n';
        richTextBox1.Text = send;
        serialPort1_DataSender(send);

        }
```

```
private void radioButtonB_CheckedChanged(object sender, EventArgs e)
{
RadioButton selectedRadioButton = (RadioButton)sender;

if (selectedRadioButton.Checked)
{
Dbidirecional= 1;
}
else { Dbidirecional = 0; }

string send = "{\"b\":" + Dbidirecional + "}" + '\n';
richTextBox1.Text = send;
serialPort1_DataSender(send);
}


private void button1_Click(object sender, EventArgs e)
{
frequency = textBox3.Text;
string send = "{\"f\":" + frequency + "}" + '\n';
richTextBox1.Text = send;
serialPort1_DataSender(send);
}

private void button2_Click(object sender, EventArgs e)
{
samples = textBox4.Text;
string send = "{\"n\":" + samples + "}" + '\n';
richTextBox1.Text = send;
serialPort1_DataSender(send);
}

private void button3_Click(object sender, EventArgs e)
{
richTextBox1.Text = "";
if (radioButton5.Checked)
{
printText = "";
}
if (radioButton6.Checked)
{

var filePath = @"C:\Users\Tiago Zorro\Desktop\Faculdade\4ºAno\2ºSemestre\SAD\Trabalho3\file.json";
if (File.Exists(filePath)) // Check if file exists
{
        // Delete the file
```

```
            File.Delete(filePath);
        }

        }
        }

        private void radioButton6_CheckedChanged(object sender, EventArgs e)
        {
        if (radioButton6.Checked)
        {
        button3.Text = "Delete File";

        string fileName = @"C:\Users\Tiago Zorro\Desktop\Faculdade\4ºAno\2ºSemestre\SAD\Trabalho3\file.json";
        string fileContent = File.ReadAllText(fileName);
        richTextBox1.Text = fileContent;
        }

        }

        private void radioButton5_CheckedChanged(object sender, EventArgs e)
        {
        if (radioButton5.Checked)
        {
        button3.Text = "Reset TextBox";
        richTextBox1.Text = printText;
        }

        }




        private void serialPort1_DataSender(String Message)
        {
        // Check if the serial port is open
        if (serialPort1.IsOpen)
        {
        try
        {
            serialPort1.Write(Message);
            // Alternatively, you can use WriteLine to send the data with a newline character at the end
            // serialPort1.WriteLine(configurationData);

            // Optionally, perform any post-sending operations or update the UI
            MessageBox.Show("Data sent successfully!");
        }
        catch (Exception ex)
```

```
{
        // Handle any errors that occur during data sending
        MessageBox.Show("Error sending data: " + ex.Message);
}
}
else
{
MessageBox.Show("Serial port is not open!");
}
}

private StringBuilder receivedDataBuffer = new StringBuilder();

private void serialPort1_DataReceived(object sender, SerialDataReceivedEventArgs e)
{

// Check for a delimiter, such as a newline character, to determine when the complete message has been received
string receivedData = serialPort1.ReadExisting();

receivedDataBuffer.Append(receivedData);

// Check for a delimiter, such as a newline character, to determine when the complete message has been received
string delimiter = "\n";
int delimiterIndex = receivedDataBuffer.ToString().IndexOf(delimiter);

if (delimiterIndex >= 0)
{
// Extract the complete message up to the delimiter
string completeMessage = receivedDataBuffer.ToString().Substring(0, delimiterIndex + delimiter.Length);

// Remove the processed message from the buffer
receivedDataBuffer.Remove(0, delimiterIndex + delimiter.Length);

printText += completeMessage;

string JSONString = completeMessage.Replace("\n", "");

Debug.WriteLine( "&" + JSONString + "&");

JObject jsonObject = JObject.Parse(JSONString);

//XmlDocument doc = (XmlDocument)JsonConvert.DeserializeXmlNode(completeMessage);
XmlDocument xmlDoc = new XmlDocument();

// Create the root element of the XML
```

```
XmlElement rootElement = xmlDoc.CreateElement("57390_57901_57733");
xmlDoc.AppendChild(rootElement);

// Loop through the properties of the JSON object
foreach (JProperty property in jsonObject.Properties())
{
	// Check if the property name is "A2" or "A3"
	if (property.Name == "A2" || property.Name == "A3")
	{
	// Create a new XML element with the property name
	XmlElement element = xmlDoc.CreateElement(property.Name);

	// Get the array of values from the property value
	JArray valuesArray = (JArray)property.Value;

	// Loop through the values and add them as child elements to the XML element
	foreach (JToken value in valuesArray)
	{
	XmlElement valueElement = xmlDoc.CreateElement("Value");
	valueElement.InnerText = value.ToString();
	element.AppendChild(valueElement);
	}

	// Add the XML element to the root element
	rootElement.AppendChild(element);
	}
}

// Convert the XmlDocument to a string
string xmlString = xmlDoc.OuterXml;
//enviar para servidor
SendXmlMessage(xmlString);

// Process the received JSON data
try
{
	if (radioButton5.Checked)
	{
	Invoke((MethodInvoker)delegate
	{
	richTextBox1.Text = printText;
	});
	}

	//guardar em ficheiro
	SaveDataToFile(completeMessage, @"C:\Users\Tiago Zorro\Desktop\Faculdade\4ºAno\2ºSemestre\SAD\Trabalho3\file.json");
```

```
        }
        catch (JsonException ex)
        {
                // Handle JSON parsing errors
                Console.WriteLine("Error parsing JSON: " + ex.Message);
        }
        }
        }

        /// tem que ser mudado de acordo com o enunciado
        private async void SendXmlMessage(string xmlMessage)
        {
        // Create an HttpClient instance
        using (HttpClient httpClient = new HttpClient())
        {
        // Set the remote server URL
        string serverUrl = "http://193.136.120.133/~sad/";

        try
        {
                // Create the HTTP content with the XML message
                HttpContent httpContent = new StringContent(xmlMessage, Encoding.UTF8,
"application/xml");

                // Send the HTTP POST request
                HttpResponseMessage response = await httpClient.PostAsync(serverUrl, httpContent);

                // Check the response status
                if (response.IsSuccessStatusCode)
                {
                // Request successful
                Console.WriteLine("XML message sent successfully!");
                }
                else
                {
                // Request failed
                Console.WriteLine("Failed to send XML message. Status Code: " + response.StatusCode);
                }
        }
        catch (Exception ex)
        {
                // Handle any exceptions that occur during the HTTP request
                Console.WriteLine("Error sending XML message: " + ex.Message);
        }
        }

        }
```

```
private void SaveDataToFile(string jsonString, string fileName)
{
// Open the file in Append mode
using (FileStream file = new FileStream(fileName, FileMode.Append))
{
using (StreamWriter writer = new StreamWriter(file))
{
        // Append the data to the file
        writer.Write(jsonString);
}
}
}

private string ReadDataFromFile(string fileName)
{
// Open the file in Read mode
using (FileStream file = new FileStream(fileName, FileMode.Open))
{
using (StreamReader reader = new StreamReader(file))
{
        // Read the data from the file
        string fileContent = reader.ReadToEnd();
        return fileContent;
}
}
}

private string CleanReceivedData(string data)
{
// Replace unwanted characters
data = data.Replace("\r\n", "\n");
data = data.Replace("\r", "\n");
data = data.Replace("\n\n", "\n");

// Remove any leading or trailing spaces
data = data.Trim();

return data;
}

private void button4_Click(object sender, EventArgs e)
{
serialPort1.PortName = textBox4.Text;

}

private void textBox1_TextChanged(object sender, EventArgs e)
{
```

```
        }

        private void checkBox8_CheckedChanged(object sender, EventArgs e)
        {
        CheckBox clickedCheckbox = (CheckBox)sender;
        if (clickedCheckbox.Checked)
        {
        digital1 = 1;
        }
        else
        {
        digital1 = 0;
        }
        string send = "{\"D1\":" + digital1 + "}" + '\n';
        richTextBox1.Text = send;
        serialPort1_DataSender(send);
        }

        private void radioButton2_CheckedChanged(object sender, EventArgs e)
        {

        }
}
}
```