MICROCONTROLADORES E INTERFACES

Positional Tracker

[Rastreador de posição]

Mestrado integrado em Engenharia Física

Alexandre Silva A89128

Pedro Teixeira A75049

Equipa Docente:

Profº Paulo Mendes

Contents

Introdução

O projeto a desenvolver na UC de Microcontroladores e Interfaces tem como objetivo a criação de um sistema capaz de medir e apresentar graficamente o deslocamento de um ser humano ao longo de um intervalo de tempo.

Os componentes utilizados neste sistema foram:

1) Uma placa “Curiosity HPC” da MicroChip, com o microcontrolador PIC18F47Q10;
2) Uma placa com um acelerómetro MMA7361L;
3) Programa MATLab para processamento e plotagem dos dados amostrados;

Por último serão retiradas as respetivas conclusões face à criação deste sistema de rastreamento posicional.

O nosso trabalho consistiu na projeção de uma interface que utiliza o pic18f47q10 como base de recolha de dados com a ajuda de um acelerómetro de três eixos que capta as acelerações nas direções x,y e z.

O microcontrolador por sua vez fará a discretização destes mesmos dados e serão enviados pela porta série. Sendo que o microcontrolador apenas tem na sua constituição um adc(conversor analógico-digital) também será implementada a técnica de multiplexagem de modo a poder recolher a informação proveniente de cada eixo em questão sem comprometer a frequência de amostragem estabelecida. Recebendo estes dados no computador com uma ferramenta adequada, no nosso caso usamos o matlab(podíamos ter escolhido outra linguagem de programação), faremos o plot em tempo real das acelerações nos três eixos e a sua dupla integração que consequentemente originará a posição nos diferentes eixos e com a aplicação do módulo ficaremos a saber a posição em cada instante.

Isto obviamente constituiu uma ferramenta amplamente usada no dia a dia como nos gps dos carros e nos telemóveis. Sendo assim com os conhecimentos adquiridos durante o semestre vamos construir um protótipo rústico desses dispositivos.

Sistema de Aquisição

O sistema de aquisição consiste numa placa com um acelerómetro (MMA7361L) com 3 outputs, sendo que cada output corresponde a um sinal de tensão que posteriormente será convertido num valor de aceleração instantânea no eixo dos xx (Output 1), no eixo dos yy (Output 2) e no eixo dos zz (Output 3).

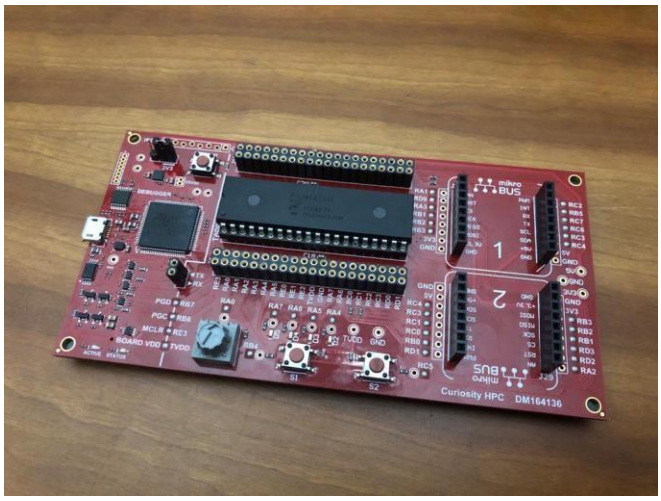


Figura 1 – Placa Curiosity HPC.

Os três outputs da placa com um acelerómetro são ligados a três entradas digitais (RD7/RD6/RD5) da nossa placa "Curiosity HPC" da MicroChip, com o microcontrolador PIC18F47Q10. A amostragem dos valores será feita com uma frequência de 93 Hz em cada pino.

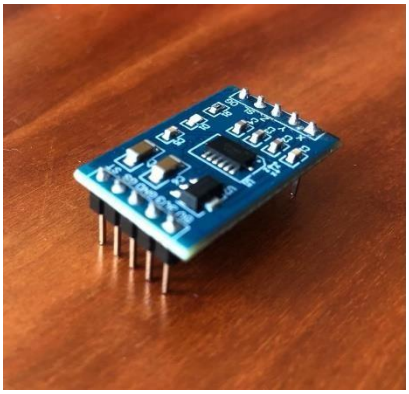


Figura 2 – Placa com acelerómetro.

Após ser feita a amostragem dos valores, os mesmos serão convertidos de valores decimais para binários e de seguida enviados para o computador através da EUSART1 presente na Placa Curiosity HPC.

Os valores fruto da amostragem poderão ser observados através de aplicativos como o CoolTerm no computador do utilizador, obtendo o seguinte resultado:

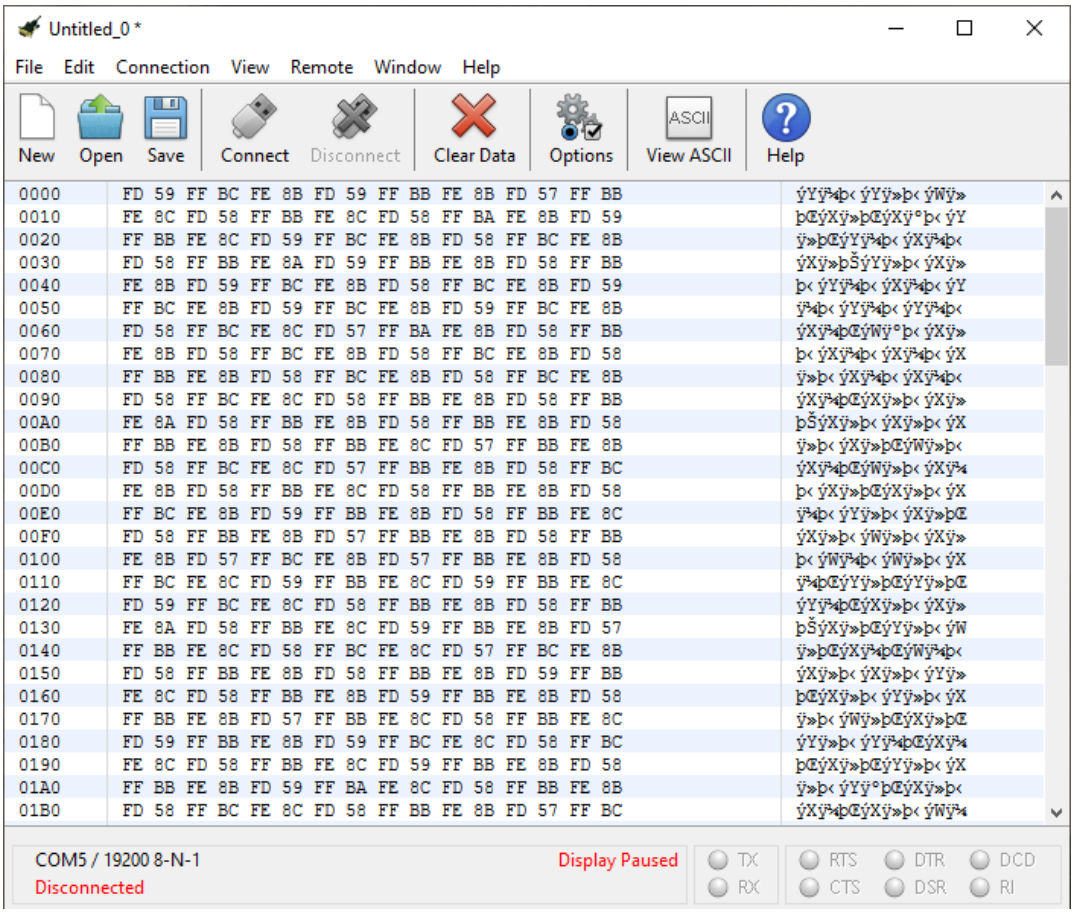


Figura 3 – Janela CoolTerm.exe (em Windows).

O descodificar dos valores observados no CoolTerm serão explicados na seção de tratamento de dados.

Microcontrolador PIC18F47Q10

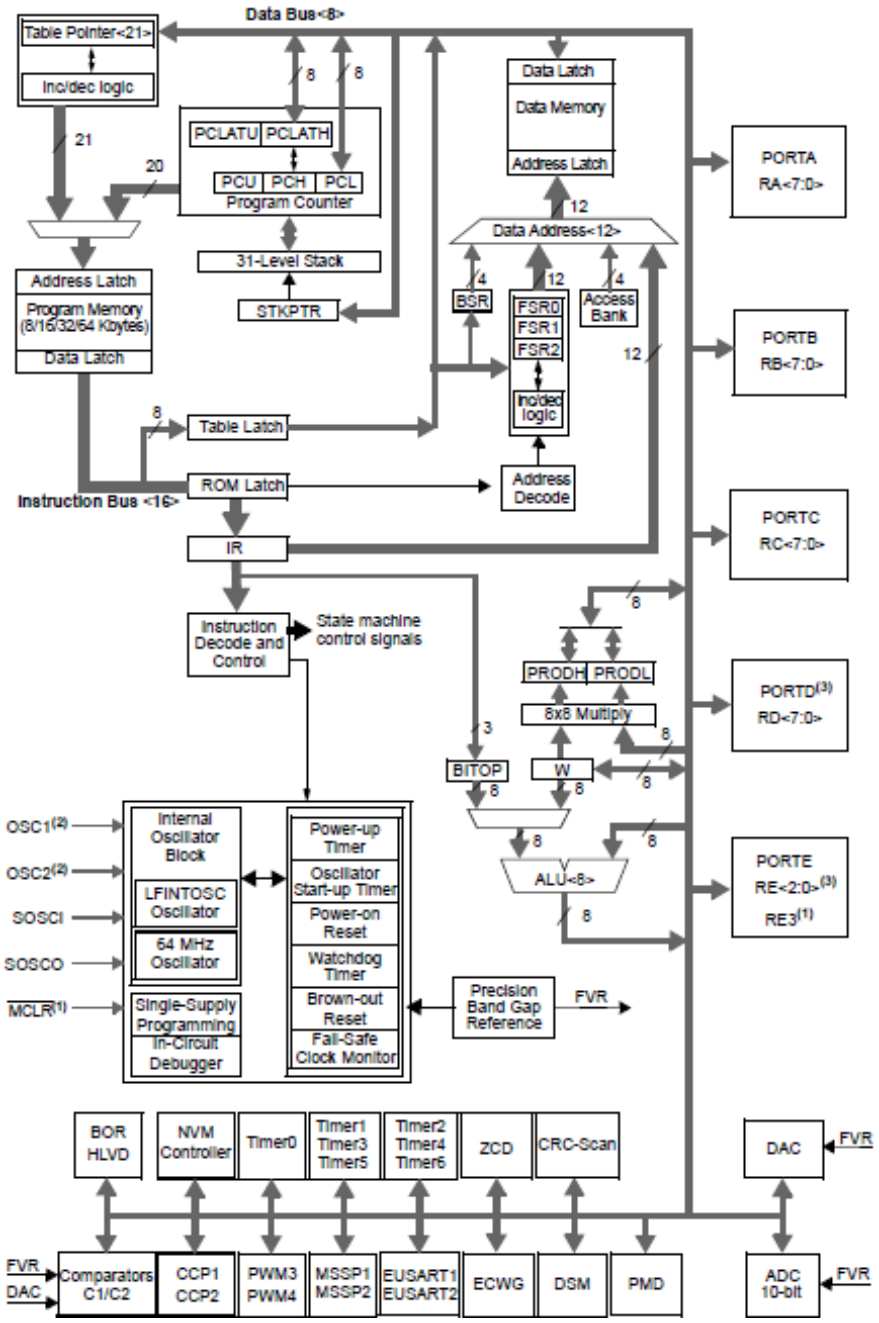


Figura 4 – Esquemático do microcontrolador.

Tivemos de recorrer à utilização de um microcontrolador PIC18F47Q10 devido às suas funcionalidades de ADC e USART que nos permitem converter o sinal analógico para palavras em binário e enviar o sinal convertido para um computador onde a informação pode ser processada. Algumas das caraterísticas relevantes para o trabalho realizado são as seguintes:

O PIC10F47Q10 é um microcontrolador de arquitetura RISC capaz de executa 75 instruções, possui 128 KBytes de memória Flash, 1 KByte de capacidade da memória EEPROM

e 3615 Bytes de memória SRAM. Ele possui 35 registos de serviço geral que podem ser acedidos em somente um ciclo de clock, ou seja, em 1 microssegundo ou mais caso alteremos a frequência de clock do sistema.

O PIC18F47Q10 contém 3 timers, registos que permitem executar programas de acordo um dado timing, dois dos quais tem um tamanho de 8-bits e outro de 16-bits.

O conversor analógico para digital presente neste microcontrolador pode operar em 40 canais diferentes com diversos modos de operação e modos de ganho diferencial selecionáveis e tem uma resolução de 10-bits.

O PIC18F47Q10 possui também 2 transmissores e 2 recetores cujas características de funcionamento podem ser configuradas.

O microcontrolador dispõe também de várias interrupções (incluindo o RESET), A maioria destas instruções pode ser programada. A prioridade de uma interrupção é definida pela sua posição na memória.

O PIC18F47Q10 opera com frequência de clock a 1MHz após ser ligado, este clock pode ser substituído recorrendo a um oscilador interno ou externo, no nosso caso escolhemos um oscilador interno de frequência 32 MHz.

Acelerómetro

O acelerómetro foi utilizado tendo em consideração os seguintes parâmetros:

- Pino GS (G Select) = 0
  Logo a escala do acelerómetro encontra-se no intervalo [-1.5g;1.5g]
  (Se Pino GS = 1, [-6g;6g].)
- Sensibilidade: 800 mV/g
- 0g–X_OffSet = 1.655V (= 0g)
  Medido com um multímetro e dentro do intervalo fornecido pelo fabricante.
- 0g–Y_OffSet = 1.755V (= 0g)
- Medido com um multímetro e dentro do intervalo fornecido pelo fabricante.
- 0g–Z_OffSet = 1.91V (= 0g)
  Medido com um multímetro e dentro do intervalo fornecido pelo fabricante.

Ligações

O microcontrolador estando ligado a uma placa “Curiosity HPC” da MicroChip, o mesmo já se encontra corretamente alimentado. O acelerómetro é alimentado a 3.3V no seu pino VDD e no pino Sleep.

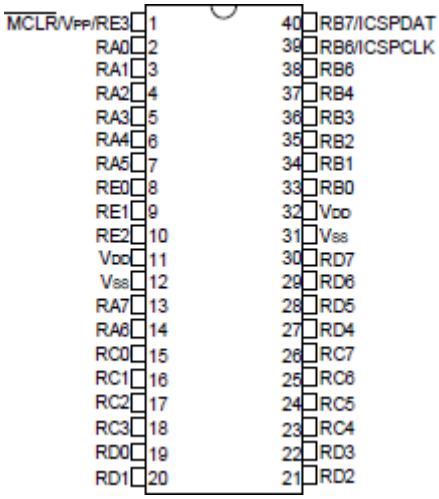
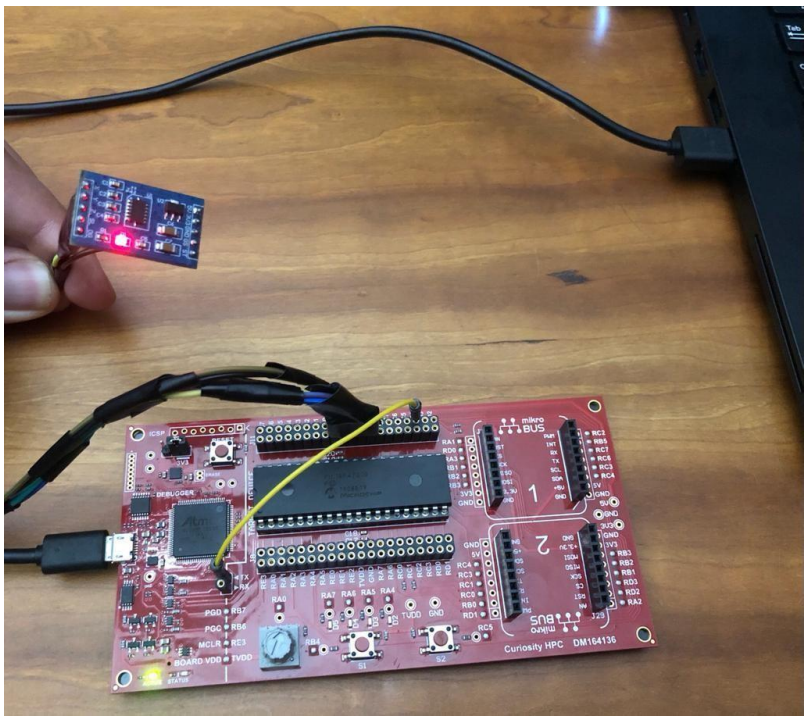


Figura 5 – PINOUT do Microcontrolador.

A ligação entre a placa “Curiosity HPC” e o computador é realizada via USB type A para Micro USB.



ADC

Foi usado o ADC do microcontrolador que permite a conversão de um sinal analógico para um sinal digital. Este ADC possui 10 bits, dos quais nós apenas utilizamos 8, e também possui 8 canais, o que permite converter 8 sinais analógicos independentes através dos pinos do porto , dos quais precisamos de usar 1 alterna (Alternando entre o ADC0 e ADC1). A conversão é efetuada iterativamente, isto é, através de aproximações sucessivas.

Relativamente à referência do ADC, esta pode ser interna ou externa. No caso deste trabalho foi utilizada uma referência externa positiva de 3.3V.

Com uma referência de 3.3V a resolução do nosso ADC será de aproximadamente 12.9mV (3.3V repartidos por 256 valores possíveis de tensão).

A palavra de 10 bits resultante de uma conversão é armazenada nos registos ADRESH e ADRESL, que iram conter respetivamente os bits mais e menos significativos do resultado da conversão.

CÓDIGO DE CONFIGURAÇÃO ADC

```
BANKSEL ADREF        ;Seleciona o Banco com o registo ADREF.
MOVLW 0b00000000     ;V_ref(-) = AV_ss, V_ref(+) = V_dd.
MOVWF ADREF,1
BANKSEL ADCLK        ;Seleciona o Banco com o registo ADCLK.
MOVLW 0b00001111     ;ADCS=001111, ADC_CLK = 32 MHz/32 = 1 MHz.
                     ;1us para converter 1 bit, 11.5us para 10 bits.
MOVWF ADCLK,1
BANKSEL ADCON0       ;Seleciona o Banco com o registo ADCON0.
MOVLW 0b00000000     ;ADC desativo, ADC_CLK=Fosc/div, results left adjusted
                     ;Conversion not in progress.
MOVWF ADCON0,1
```

A conversão do ADC é iniciada com a seguinte instrução:

```
BSF ADCON0,7         ;ENABLE ADC
```

USART

É necessário um modo de enviar os dados via porta série para o computador, para tal é utilizado o USART (Universal Synchronous Asynchronous Receiver Transceiver) que opera em full-duplex (ou seja é capaz de receber e transmitir dados simultaneamente), com frame, velocidade de envio e modo de paridade configuráveis. O PIC18F47Q10 apresenta 2 USART's, mas para a realização do projeto só necessitamos de um, por isso resolvemos usa o USART 1.

A informação é enviada numa frame que contem 1 start bit, uma palavra de 5 a 9 data bits, pode conter 1 bit de paridade e 1 ou 2 stop bits, conforme a frame for configurada. A frame terá no total 10 bits.

Para este projeto decidimos utilizar uma frame de 8 databits, sem paridade par e 1 stopbit. A baud-rate selecionada foi a standard 19200 bps, que é uma velocidade de transmissão mais que suficiente para a nossa frequência de amostragem (279Hz).

Para o cálculo do débito binário sabemos que a transmissão de dados é feita a partir do envio de uma palavra binária de 1byte, ou seja, 8 bits. A nossa frequência de amostragem é de 279Hz, logo temos um débito binário de 8bits*250Hz=2232bps.

CÓDIGO DE CONFIGURAÇÃO EUSART

```
movlw X              ;Move X to Baud Rate Generator.
                     ;Expected a baud of 19200.
BANKSEL SP1BRGL      ;Seleciona o Banco com o registo SP1BRGL.
movwf SP1BRGL        ;Passa o valor de X(25) para o registo SP1BRGL.
movlw 0x00           ;Passa o valor de 0x00 para o registo SP1BRGH.
movwf SP1BRGH        ;"1" for USART 1, since we have 2 USART available.
movlw 0b10100000     ;8 data bits, TX enabled, master clock selected.
BANKSEL TX1STA
movwf TX1STA         ;Low speed SPBRG mode.
movlw 0b10010000     ;Ativar  o usart, ativar a recepçao,8 bits.
        BANKSEL RC1STA
        movwf RC1STA         ;Receiver enabled
        MOVLW 0b00000000
        MOVWF BAUD1CON
```

USART

	BaudRate (bits/s)	Data	Palavra	T_envio (1bit)	T_envio (Palavra)
Experiência 1	9600	8 bits	10 bits	0.00010417	0.001041667
Experiência 2	19200	8 bits	10 bits	5.2083E-05	0.000520833

Timer de 8-Bit

Os timers estão presentes na maioria dos microcontroladores, incluindo o PIC18F47Q10, e podem ser utilizados para gerar interrupções no programa respondendo a uma sub-rotina. Pode ser definido de modo simples como sendo um registo ou como um simples contador.

No nosso trabalho configuramos o Timer 0 que, sendo de 8 bits, permite uma contagem até 255.

	F_osc (HZ)	CLK Timer (HZ)	PreScaler	Pre_Bin	Comparator	Comp_Bin
Experiência 1	32000000	8000000	2048	1011	2	10
Experiência 2	32000000	8000000	2048	1011	7	111
Experiência 3	32000000	8000000	4096	1100	32	100000

	PostScaler	Comp_Bin	F_ams (Hz)	T_ams (s)	F_ams.x (HZ)	T_ams.x (s)
Experiência 1	2	1011	976.5625	0.001024	325.5208333	0.003072
Experiência 2	2	1011	279.0178571	0.003584	93.00595238	0.010752
Experiência 3	4	11	15.25878906	0.065536	5.086263021	0.196608

Os valores da linha “Experiência 2” foram os valores utilizados para uma correta plotagem dos valores de tensão e aceleração dos ficheiros matlab “PROJ_plot_aceleracoes.m” e “PROJ_fast_plot_tensao.m”.

Os valore da linha “Experiência 3” foram utilizados para uma plotagem aproximada dos valores de posição após dupla integração dos valores de aceleração no ficheiro matalab “PROJ_plot_posicao.m”.

CÓDIGO DE CONFIGURAÇÃO TIMER0

```
BANKSEL T0CON0          ;Seleciona o Banco com o registo T0CON0.
MOVLW 0b00000001        ;Modulo desativo, TMR0 é 8bit, 1:2 PostScaler.
MOVWF T0CON0,1
BANKSEL T0CON1          ;Seleciona o Banco com o registo T0CON1.
MOVLW 0b01001011        ;CLK=Fosc/4, TMRO sincrono em Fosc/4.
                        ;PreScaler =1011 (2048).
                        ;CLK=32M/4/2048/2/7=279.01Hz.
                        ;4 ms period between Timer interruptions.
MOVWF T0CON1,1
BANKSEL TMR0L           ;Seleciona o Banco com o registo TMR0L.
CLRF TMR0L              ;Coloca os 8 bits menos significativos do contador
                        ;do TIMER0 a 0 (Limpa o nosso contador.)
BANKSEL TMR0H           ;Seleciona o Banco com o registo TMR0H. (Counter)
MOVLW 0b0000111         ;Seleciona 7 como valor a comparar. (Comparater)
MOVWF TMR0H
```

Programação do Microcontrolador

Para configurar o funcionamento do microcontrolador este teve de ser programado em linguagem assembly. Este código é carregado para o microcontrolador através de um cabo micro USB que estabelece a ligação entre o microcontrolador e um computador. O programa utilizado para essa finalidade é o MPLab X IDE v5.45.

O objetivo do trabalho é visualizar os 3 sinais de aceleração num gráfico temporal e de seguida realizar uma dupla integração de modo a obter a trajetória de um utilizador num gráfico 3D. Para tal foi necessário converter os sinais provenientes dos 3 pinos do acelerómetro e de seguida enviar o resultado para o computador identificando a que eixo se refere a conversão, para tal realizamos uma multiplexagem do ADC do PIC e após a conclusão da conversão o valor resultante é enviado pela USART. De modo a garantir uma correta identificação do eixo a que se referente o resultado da conversão, foi criado um protocolo em que o primeiro Byte a ser enviado identificaria o eixo. Assim sendo, 0xFF corresponderia ao eixo do XX, 0xFE ao eixo dos YY e 0xFD ao eixo dos ZZ. Utilizamos um timer com interrupção para realizar a multiplexagem e mudança de canal de conversão.

CÓDIGO MULTIPLEXAGEM

```
BANKSEL PIR0
BCF PIR0,5                  ;clear timer_int flag.
canal1:
   MOVLW 0b00000001
   CPFSEQ count             ;se for igual ao W skip.
   GOTO canal2              ;Vai para o canal Y.
      BANKSEL ADPCH
   MOVLW 0b00011101         ;Set RD5 as ADC, input X.
   MOVWF ADPCH,1
      MOVLW 0b00000010      ;Passa o canal Y para count.
      MOVWF count
      MOVLW 0b11111111      ;Identificador do canal X.
      MOVWF protocolo       ;Coloca identificador no protocolo.
   CALL SENDPROTOCOLO       ;Envia protocolo.
      GOTO fim
canal2:
   MOVLW 0b00000010
```

```
    CPFSEQ count                ;se for igual ao W skip.
    GOTO canal3                 ;Vai para o canal Z.
    BANKSEL ADPCH
    MOVLW 0b00011110        ;Set RD6 as ADC,input Y.
    MOVWF ADPCH,1
        MOVLW 0b00000011            ;Passa o canal Z para count.
        MOVWF count
        MOVLW 0b11111110            ;Identificador do canal Y.
        MOVWF protocolo             ;Coloca identificador no protocolo.
        CALL SENDPROTOCOLO          ;Envia protocolo.
    GOTO fim
canal3:
    MOVLW 0b00000011
    CPFSEQ count                    ;Se for igual ao W skip.
    nop
    BANKSEL ADPCH
    MOVLW 0b00011111                ;Set RD7 as ADC input z.
    MOVWF ADPCH,1
        MOVLW 0b00000001            ;Passa o canal X para count.
        MOVWF count
        MOVLW 0b11111101            ;Identificador do canal Z.
        MOVWF protocolo             ;Coloca identificador no protocolo.
        CALL SENDPROTOCOLO ;Envia protocolo.
    GOTO fim
fim:
```

Explicação do programa em Assembly

Agora que temos o diagrama de blocos vamos explicar com mais profundidade o processo utilizado.

• Começamos por configurar as portas A,B,C,D. Na portd vamos ligar os 3 eixos do acelerómetro e deste modo temos de definir esses três pinos (ra5 ra6 ra7) como sendo analógico e de input. A portc é onde vamos colocar o usart mais concretamente no pino rc4 que depois ligamos através de um fio ao pino TX de modo a ser enviado para o computador. No portc é onde estará o clock para o podermos visualizar no osciloscópio. No portb vamos colocar a interrupção externa (INT0);

• De seguida configuramos o Timer0, definimo-lo como sendo de 8 bits e colocámos um prescaler e postcaler adequados de modo a este demorar o tempo que queremos até ocorrer a interrupção. Serve de nota que com o prescaler diminuímos a rapidez de contagem do timer, ou seja, o número de impulsos de relógio necessários para que o contador incremente uma unidade. Já o prescaler é a frequência á qual é gerada uma interrupção (de 2 em 2 vezes por exemplo). Também temos de definir a fonte de relógio que usamos sendo que no trabalho selecionamos a Fosc/4. Com estes dados é possível calcular o tempo da interrupção, seja 4/Fosc o tempo que demora um ciclo de relógio basta multiplicar este tempo pelo prescaler e postcaler e pelo número de incrementos que irão ser feitos. Para estes incrementos, o timer0 tem dois registos, um no qual começa a incrementar e outro com o qual irá fazer a comparação sendo que para assim que ambos forem iguais. A fórmula utilizada encontra se abaixo:

$$Ttimer = Postcaler * Prescaler * número\ de\ incrementos * 4\ /Fosc$$

• Partimos agora para a configuração do adc. Configuramos o ADC como leftjustified, ou seja, os 8 bits mais significativos são armazenados no registo ADRESH e os 2 menos significativos no registo ADRESL. Selecionamos como sinal de relógio do adc a Fosc/128 de modo que o tempo de conversão de um bit (Tad) fosse 4us o que segundo o datasheet é um tempo aceitável. Com isto retiramos também o tempo que o adc demora a converter os 10 bits através de uma 17 simples regra de 3 simples. Sabendo que 10 bits demoram 11.5Tad a converter (segundo o datasheet) então no nosso caso este tempo vai ser da ordem dos 11.5us.

• Configuramos o eusart, onde vamos mover o valor no registo X para o registo SP1BRGL de modo a selecionar a baud pretendida de 19600. Vamos também selecionar o uasart1 uma vez que temos dois usart disponíveis para o nosso uso. Selecionamos uma

comunicação de 8 bits pois no modo como configurámos o método foi explicitado que enviamos 8 bits de cada vez. Também ativamos tx de modo a permitir o envio de dados;

• Vamos agora para a parte das interrupções na qual limpamos as flags das interrupções a usar de modo a poder usá-las. Também temos de estabelecer prioridades nelas, no nosso caso vamos conceder ao INT0, ADC e timer0 alta prioridade de modo que sempre que um destes cause uma interrupção, o pic possa parar o que estava a fazer e execute uma função previamente definida para o caso em que isso acontece.

Chegamos ao fim das configurações dos registos, agora vamos construir o processo que se desenrola quando as interrupções são geradas.

Primeiramente pressionamos o botão, isto gera uma interrupção e o pic para de fazer a tarefa atual. De seguida verifica qual a interrupção que foi gerada que nesta instância foi a do INT0. Põe o timer a contar e move o valor 0b00000001 para o registo seleciona. Finalmente volta ao que estava a fazer antes da interrupção.

Quando o timer acaba uma segunda interrupção é gerada e o pic faz o mesmo processo de descobrir qual a interrupção tal como na interrupção anterior. Agora o pic vai comparar o valor no registo seleciona com valores predefinidos (0b00000001 /0b00000010 /0b00000100). Se for igual seleciona o canal de entrada do adc pré-definido e move um novo valor para o registo seleciona de modo que numa nova interrupção outro canal seja selecionado. Tirando no caso se ser igual a 0b00000100 onde o valor que movemos é 0b00000001 de modo a reiniciar o ciclo. Além disso também chamamos a função sendchar que envia por porta série o valor do canal que está a ser convertido.

Finalmente coloca o adc a converter e volta da interrupção. 18 Finalmente temos a interrupção do adc que basicamente quando é gerada coloca os 8 bits mais significativos num registo e envia por porta série.

Este processo repete-se indefinidamente enquanto o utilizador desejar.

MATLAB

A nossa interface foi concebida em MATLAB. O principal objetivo é gerar uma interface onde consigamos observar os 3 sinais de tensão de cada pino e de seguida os 3 sinais de aceleração após conversão segundo a sensibilidade de 0.8mV/g e tendo em consideração o g-offset descritivos no datasheet do acelerómetro.

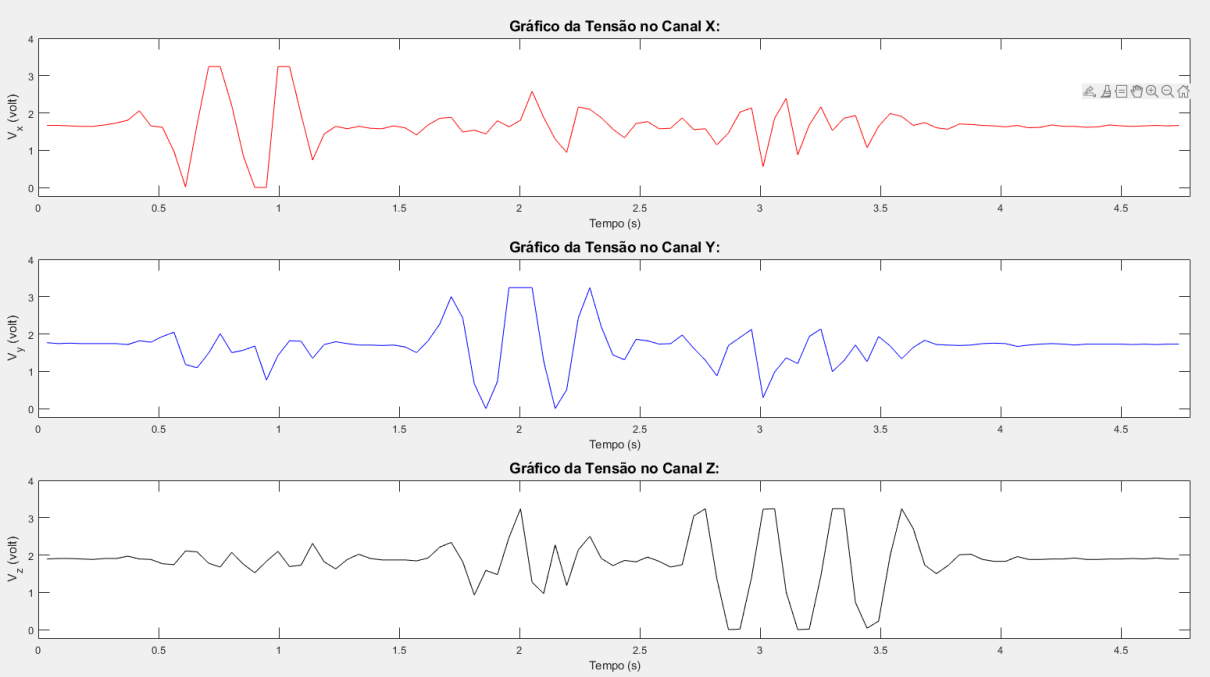


Figura 6 - Interface em MatLab PROG_plot_tensão

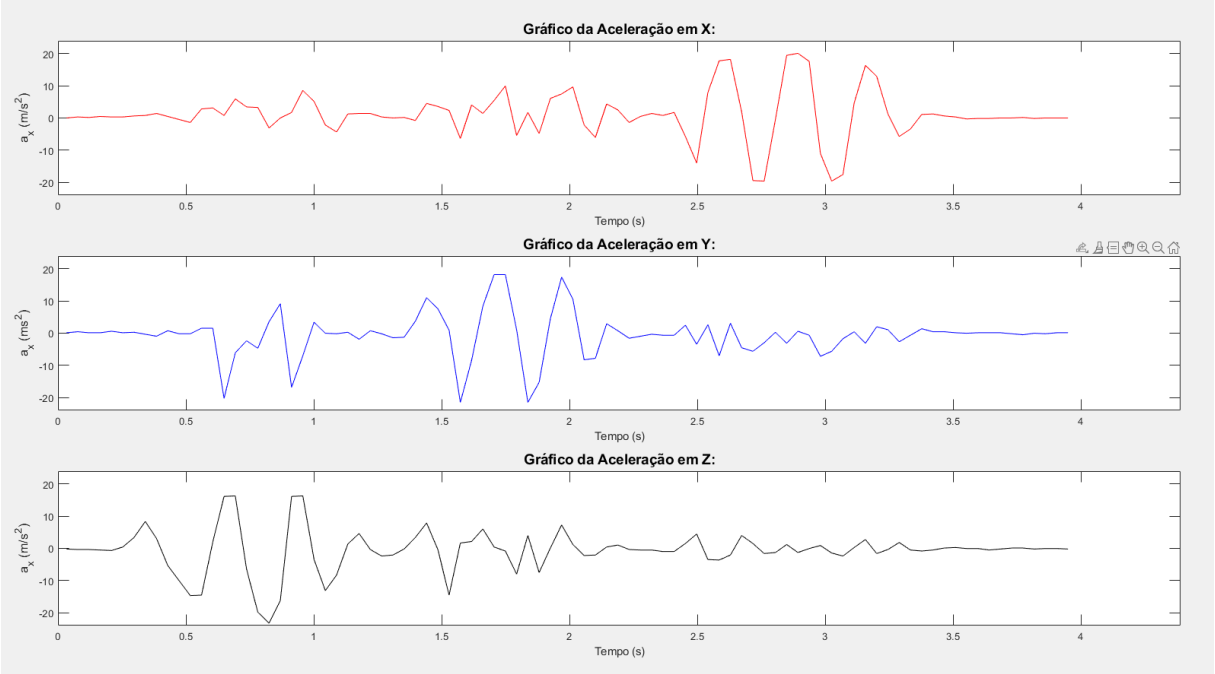


Figura 7 - Interface em MatLab PROG_plot_aceleracoes

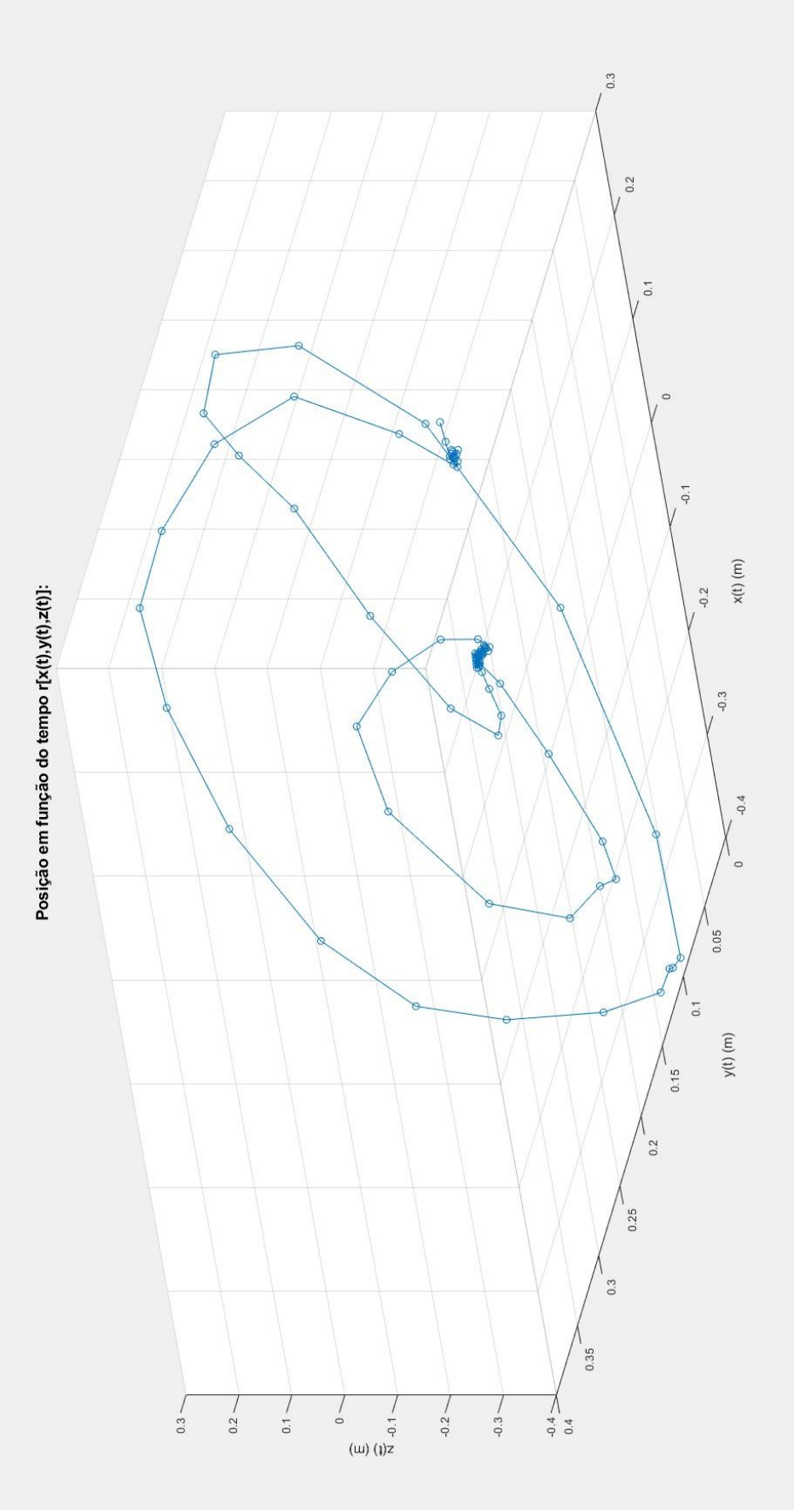
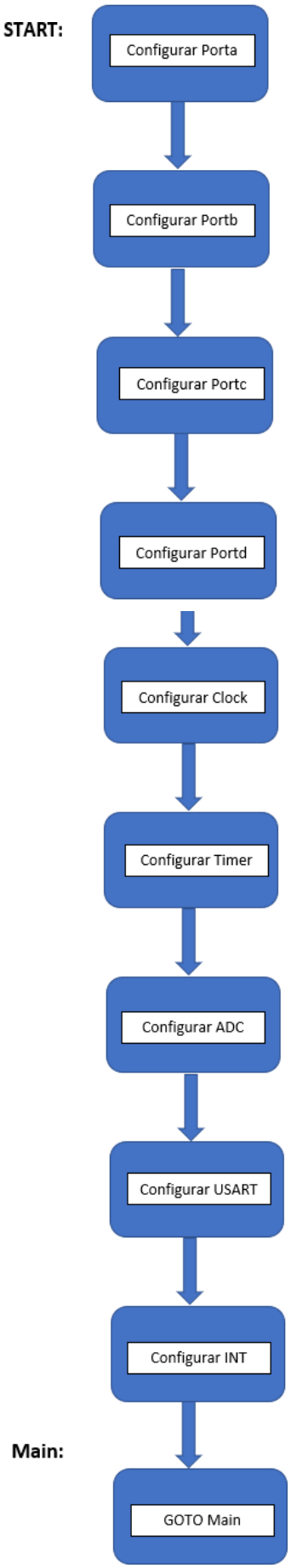


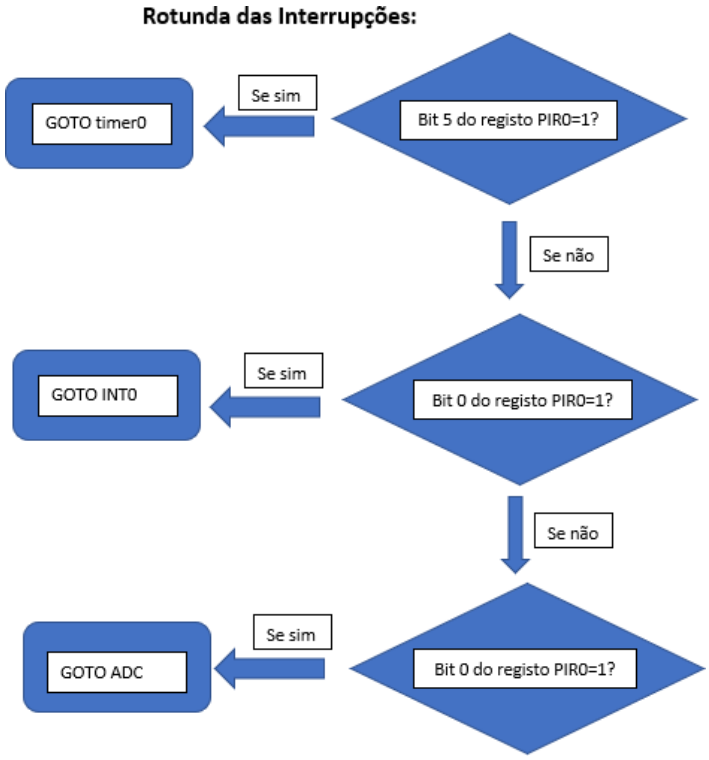
Figura 8 - Interface em MatLab PROG_plot_posicoes

Fluxograma

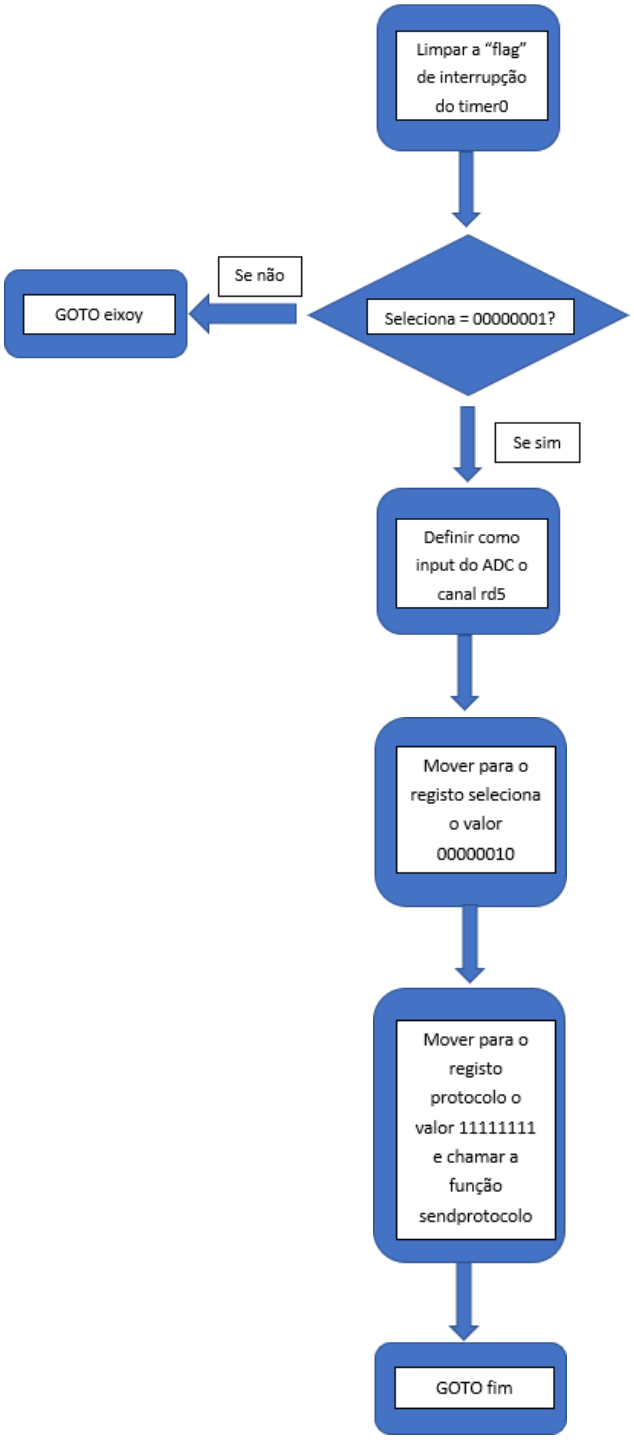
Começamos por fazer um fluxograma do nosso código em assembly com o qual programamos o PIC18F47Q10, de modo a explicar o método com o qual realizamos a primeira parte do trabalho.

Asssembly

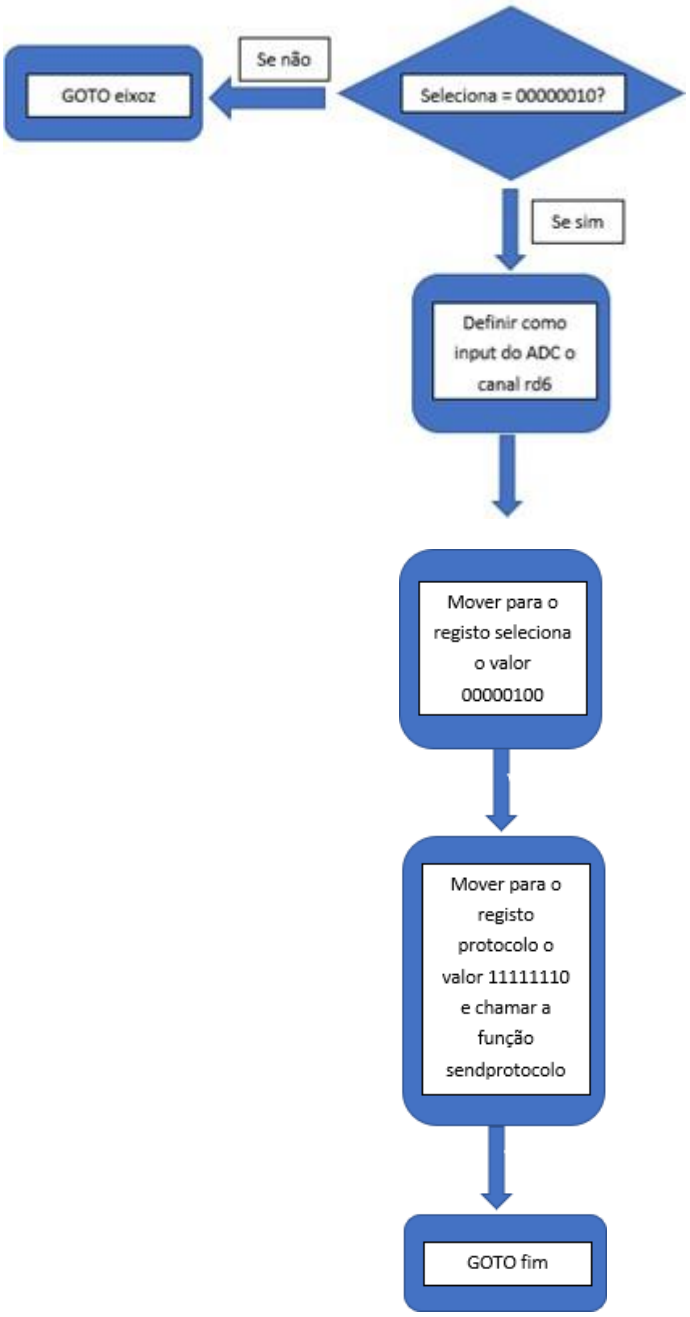




Timer0:



Eixoy:



ADC:

MATLAB

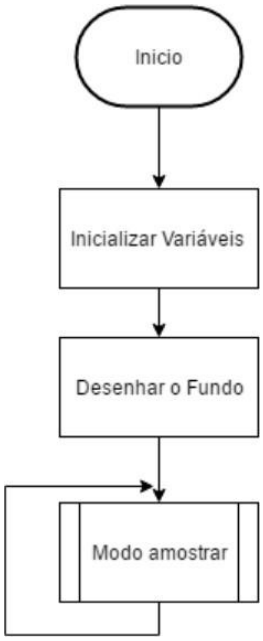


Figura 9 - Fluxograma MATLAB

Referências

[1] Datasheet PIC18F47Q10 MicroChip;

[2] Datasheet MMA7361L RoHS;

[3] https://www.mathworks.com/help/matlab/ref/serial.html;

[4] https://www.mathworks.com/help/matlab/ref/trapz.html;

[6] https://www.microchip.com/DevelopmentTools/ProductDetails/PartNO/DM164136#additional-summary;

[7] https://asm51.eng.br/phpbb/viewtopic.php?f=10&t=17925;

[8] Digital Signal Processing, **ISBN:** 9780128150719;

Conclusão

Em suma o objetivo do trabalho foi parcialmente atingido, isto é, conseguimos criar um sistema capaz de amostrar valores de tensão proveniente de 3 pinos do acelerómetro, converter os valores em aceleração e de seguida realizar uma dupla integração e obter a trajetória de um utilizador, sendo que as precisões dos resultados obtidos deixam a desejar.

Uma forma de melhor a precisão e a exatidão nos resultados seria adicionar um subsistema com GPS e assim realizar correções nos valores obtidos pela acelerómetro.

No entanto com o sistema construído conseguimos detetar a forma do movimento realizado pelo utilizador (retilíneo, circular ou complexo) (figura 8), a direção do movimento e o sentido. Algo que servirá de motivação para futuros projetos.

Pelo caminho descobrimos algumas limitações na nossa implementação, sendo elas:

1) Grande frequência de amostragem leva a uma maior quantidade de valores amostrados que por sua vez atrasa bastante o processamento dos dados e a sua plotagem em tempo real.
2) Filtragem com hardware, mesmo adicionando ruído ao sinal (que depende da potência desse mesmo ruído), permite obter os valores já filtrados, sendo que filtragem com software implica maior processamento por parte do computador o que leva a um atraso no “display” dos valores, tornando por fazes o tempo de display 40x maior que o tempo de coleção dos valores.
3) O tempo utilizado para o timer0 deve ter em consideração o tempo de conversão do ADC e o tempo de envio de dados pela USART.
4) As limitações do acelerómetro devem ser tidas em consideração antes de realizar qualquer tipo de cálculo de frequência de amostragem, assim como na conversão de valores de tensão para aceleração.

Para terminar resta dizer que num futuro próximo iremos olhar para microcontroladores e qualquer tipo de sensores com muito mais respeito, atenção e submissão.

Anexos

Código Assembly

```
; Código que permite configurar vários PORT, o clock, o ADC, e as interrupções
; implementa também a leitura do ADC utilizando um mecanismo baseado em
; interrupções.
; configura também uma interrupção externa ligada ao switch que liga ao pino RB4
; e acende um LED em resposta a carregar no switch.
; configura também o envio do valor do ADC pela porta série usando a USART
; (usa o pino RC4 como TX da porta série) que liga/desliga o envio por interrupção
; gerada pelo S1.
; neste código vamos configurar o PIC para funcionar como um sistema de aquisição
; e transmissão de dados para o computador que provêm do acelerómetro de 3 eixos
; para isso vamos construir um sistema de multiplexagem que permita fazer a ;
aquisição nos 3 eixos sem perturbar a frequência de amostragem designada para o
efeito


#include <pic18f47q10.inc>
#include <xc.inc>

CONFIG FEXTOSC=0b100        ;Deactivate external oscillator (to allow write to RA7).
CONFIG CSWEN=0b1            ;Allow editing NDIV and NOSC for CLK config.
CONFIG WDTE=OFF             ;Required to avoid WDT restarting micro all the time.

#define adc_read 0
#define count 1
#define protocolo 2
#define X 25                ;25 for baud 19200 for CLK at 32MHz


PSECT code
ORG 0x0000
   goto start               ;The start of the code (position 0x00 in the memory)
```

```
                        ;goes to function 'start'

ORG 0x0008
    goto CMEMS          ;Position 0x08 is attributed to interruption response.

ORG 0x0030              ;Main program starts at position 0x30
start:

   ;===============
   ;CONFIGURE PORTA
   ;===============
   BANKSEL LATA         ;Seleciona o Banco com o registo LATA.
   CLRF LATA,1          ;Coloca todos os pinos de saída do porto A a "0".
   MOVLW 0b00000000     ;Configura os RA<7:0> como saida.
                        ;(Define direção dos Pinos. 1 = In, 0 = Out.)
   BANKSEL TRISA        ;Seleciona o Banco com o registo TRISA.
   MOVWF TRISA,1
   MOVLW 0b00000000     ;Configura os RA<7:0> como saida digital.
   BANKSEL ANSELA       ;Seleciona o Banco com o registo ANSELA.
   MOVWF ANSELA,1


   ;===============
   ;CONFIGURE CLOCK
   ;===============
   BANKSEL OSCCON1      ;Seleciona o Banco com o registo OSCCON1.
   MOVLW 0b01100000     ;NOSC=0110 (Oscilador Interno de Alta Frequência).
                        ;(HFINTOSC).
                        ;NDIV=0000 (Divider=1, CLK divided by 1).
   MOVWF OSCCON1,1
   BANKSEL OSCFRQ       ;Seleciona o Banco com o registo OSCFRQ.
   MOVLW 0b0000110      ;HFFRQ=0110 -> CLK=32 MHz => 32/1<- NDIV.
   MOVWF OSCFRQ,1
   BANKSEL OSCEN        ;Seleciona o Banco com o registo OSCEN.
```

```
MOVLW 0b01000000        ;HFINTOSC Enabled @freq=OSCFRQ ativo.
MOVWF OSCEN,1

;===============
;CONFIGURE PORTB
;===============
BANKSEL LATB            ;Seleciona o Banco com o registo LATB.
CLRF LATB,1             ;Coloca todos os pinos de saída do porto B a "0".
BANKSEL TRISB           ;Seleciona o Banco com o registo TRISB.
CLRF TRISB,1            ;Coloca todos os pinos de saída do porto A a "0".
BSF     TRISB,4         ;Excepto RB4, que será usado como botão de fonte de
                        ;interrupcão.
BANKSEL ANSELB          ;Seleciona o Banco com o registo ANSELB.
CLRF ANSELB,1           ;Configura os RB<7:0> como saida digital.
BANKSEL RB7PPS          ;Seleciona o Banco com o registo RB7PPS.
                        ;Selecionar os INPUTS e OUTPUTS dos respectivos
                        ;perifericos.
MOVLW 0x14              ;CLKR só pode ter o OUTPUt
                        ;direcionado para PORTB ou C.
MOVWF RB7PPS            ;CLKR_output no pino RB7.
BANKSEL INT0PPS         ;Seleciona o Banco com o registo INT0PPS.
MOVLW 0x0C              ;0x0C for RB4.
MOVWF INT0PPS           ;INT0_input ligado ao pino RB4.

;===============
;CONFIGURE PORTD
;===============
BANKSEL LATD            ;Seleciona o Banco com o registo LATD.
CLRF LATD,1             ;Coloca todos os pinos de saída do porto D a "0".
BANKSEL TRISD           ;Seleciona o Banco com o registo TRISD.
MOVLW 0b11100000        ;Configura os RD<7:5> como entradas e
                        ;RA<4:0> como saídas.
MOVWF TRISD,1
BANKSEL ANSELD          ;Seleciona o Banco com o registo ANSELD.
```

```
MOVLW 0b11100000        ;Configura os RD<7:5> como entrada analógica
                        ;RD<4:0> como saídas digitais.
MOVWF ANSELD,1

;===============
;CONFIGURE PORTC
;===============
BANKSEL LATC            ;Seleciona o Banco com o registo LATC.
CLRF LATC,1             ;Coloca todos os pinos de saída do porto C a "0".
BANKSEL TRISC           ;Seleciona o Banco com o registo TRISC.
MOVLW 0b10000000        ;Configura os RC<7> como entrada
                        ;e RC<6:0> como saida.
MOVWF TRISC,1           ;Todos os pinos são de output exceto
                        ;o RC7 que é input.
BANKSEL ANSELC          ;Seleciona o Banco com o registo ANSELC.
CLRF ANSELC,1           ;Configura os RC<7:0> pinos digital.
BANKSEL RC4PPS          ;Seleciona o Banco com o registo RC4PPS.
MOVLW 0x09              ;0x09 for EUSART1(TX/CK).
MOVWF RC4PPS            ;EUSART1(TX/CK)_output in RC4.
MOVLW 0x17              ;EUSART1_Receive_input ligado ao RC7.
MOVWF RX1PPS

;================
;CONFIGURE TIMER0
;================
BANKSEL T0CON0          ;Seleciona o Banco com o registo T0CON0.
MOVLW 0b00000001        ;Modulo desativo, TMR0 é 8bit, 1:2 PostScaler.
MOVWF T0CON0,1
BANKSEL T0CON1          ;Seleciona o Banco com o registo T0CON1.
MOVLW 0b01001011        ;CLK=Fosc/4, TMRO sincrono em Fosc/4.
                        ;PreScaler =1011 (2048).
                        ;CLK=32M/4/2048/2/7=279.01Hz.
                        ;4 ms period between Timer interruptions.
MOVWF T0CON1,1
```

```
BANKSEL TMR0L           ;Seleciona o Banco com o registo TMR0L.
CLRF TMR0L              ;Coloca os 8 bits menos significativos do contador
                        ;do TIMER0 a 0 (Limpa o nosso contador.)
BANKSEL TMR0H           ;Seleciona o Banco com o registo TMR0H. (Counter)
MOVLW 0b0000111         ;Seleciona 7 como valor a comparar. (Comparater)
MOVWF TMR0H

;=============
;CONFIGURE ADC
;=============
BANKSEL ADREF           ;Seleciona o Banco com o registo ADREF.
MOVLW 0b00000000        ;V_ref(-) = AV_ss, V_ref(+) = V_dd.
MOVWF ADREF,1
BANKSEL ADCLK           ;Seleciona o Banco com o registo ADCLK.
MOVLW 0b00001111        ;ADCS=001111, ADC_CLK = 32 MHz/32 = 1 MHz.
                        ;1us para converter 1 bit, 11.5us para 10 bits.
MOVWF ADCLK,1
BANKSEL ADCON0          ;Seleciona o Banco com o registo ADCON0.
MOVLW 0b00000000         ;ADC desativo, ADC_CLK=Fosc/div, results left adjusted
                        ;Conversion not in progress.
MOVWF ADCON0,1

;========================
;configure serial port
;========================
movlw X                 ;Move X to Baud Rate Generator.
                        ;Expected a baud of 19200.
BANKSEL SP1BRGL         ;Seleciona o Banco com o registo SP1BRGL.
movwf SP1BRGL           ;Passa o valor de X(25) para o registo SP1BRGL.
movlw 0x00              ;Passa o valor de 0x00 para o registo SP1BRGH.
movwf SP1BRGH           ; "1" for USART 1, since we have 2 USART available.
movlw 0b10100000        ;8 data bits, TX enabled, master clock selected.
BANKSEL TX1STA
movwf TX1STA            ;Low speed SPBRG mode.
```

```
movlw 0b10010000        ;Ativar  o usart, ativar a recepçao,8 bits.
BANKSEL RC1STA
movwf RC1STA            ;Receiver enabled
MOVLW 0b00000000
MOVWF BAUD1CON



;=================
;ENABLE INTERRUPTS
;=================
BANKSEL PIR0
BCF PIR0, 5             ;clear timer interrupt flag
BANKSEL PIR1
BCF PIR1,0              ;clear ADC interrupt flag
BANKSEL PIE0
BSF PIE0,5              ;enable timer int
BSF PIE0,0              ;enable INT0
BCF PIR0,0              ;clear INT0 interrupt flag
BANKSEL PIE1
BSF PIE1,0              ;enable adc int
BANKSEL PIE3
BSF PIE3,5              ;ativar a interrupção de recebimento
BANKSEL INTCON
BSF INTCON,5            ;liga a prioridade das interrupções
BSF INTCON,7            ;enable peripheral interruptions
BANKSEL IPR0
MOVLW 0b00100001        ;timer e INT0 alta prioridade
MOVWF IPR0
BANKSEL IPR1
MOVLW 0b00000001        ;adc alta prioridade
MOVWF IPR1
BANKSEL IPR3
MOVLW 0b00000000        ;interrupção de recebimento de baixa prioridade
MOVWF IPR3
```

```
    BANKSEL T0CON0
    BANKSEL ADCON0
    BSF ADCON0,7            ;ENABLE ADC
    BSF INTCON,6            ;enable global interruptions - do this after
                            ;configurations are SET


;=======================
;Main code (do nothing)
;=======================
main:
    nop
    nop
    BCF PORTA, 6
    nop
    goto main

CMEMS:
    BANKSEL PIR0
    BTFSC PIR0, 5           ;(se for zero salta)
                            ;check if the timer0 interrupt flag is set.
                            ;If so, go to timer0_int_handler. If not, skip.
    goto timer0_int_handler

    BTFSC PIR0, 0           ;check if the INT0IF.
                            ;If so, go to INT0_int_handler.
                            ;If not, skip.
    goto INT0_int_handler

    BANKSEL PIR1
    BTFSC PIR1,0            ;check if the ADC interrupt flag is set.
                            ;If so, go to adc_int_handler.
                            ;If not, skip.
    goto adc_int_handler
```

```
timer0_int_handler:
   BANKSEL PIR0
   BCF PIR0,5                    ;clear timer_int flag.
   canal1:
      MOVLW 0b00000001
      CPFSEQ count               ;se for igual ao W skip.
      GOTO canal2                ;Vai para o canal Y.
         BANKSEL ADPCH
      MOVLW 0b00011101           ;Set RD5 as ADC, input X.
      MOVWF ADPCH,1
         MOVLW 0b00000010        ;Passa o canal Y para count.
         MOVWF count
         MOVLW 0b11111111        ;Identificador do canal X.
         MOVWF protocolo         ;Coloca identificador no protocolo.
      CALL SENDPROTOCOLO         ;Envia protocolo.
         GOTO fim
   canal2:
      MOVLW 0b00000010
      CPFSEQ count               ;se for igual ao W skip.
      GOTO canal3                ;Vai para o canal Z.
      BANKSEL ADPCH
      MOVLW 0b00011110           ;Set RD6 as ADC,input Y.
      MOVWF ADPCH,1
         MOVLW 0b00000011        ;Passa o canal Z para count.
         MOVWF count
         MOVLW 0b11111110        ;Identificador do canal Y.
         MOVWF protocolo         ;Coloca identificador no protocolo.
         CALL SENDPROTOCOLO      ;Envia protocolo.
      GOTO fim
   canal3:
      MOVLW 0b00000011
      CPFSEQ count               ;Se for igual ao W skip.
      nop
```

```
        BANKSEL ADPCH
        MOVLW 0b00011111        ;Set RD7 as ADC input z.
        MOVWF ADPCH,1
            MOVLW 0b00000001    ;Passa o canal X para count.
            MOVWF count
            MOVLW 0b11111101    ;Identificador do canal Z.
            MOVWF protocolo     ;Coloca identificador no protocolo.
            CALL SENDPROTOCOLO ;Envia protocolo.
        GOTO fim
    fim:
    BANKSEL ADCON0
    BSF ADCON0,0            ;Inicia conversao do ADC.
    BTG PORTA,4
    RETFIE                  ;return from interruption

INT0_int_handler:
    BANKSEL PIR0
    BCF PIR0, 0             ;Clear the INT0 intrrupt test bit
    BANKSEL PORTA
    BTG PORTA,5             ;Toggle the LED state (ON if OFF, OFF if ON).
    BTG T0CON0,7            ;Toggle TIMER0 ON<->OFF.
    MOVLW 0b00000001        ;Passa o canal X para count.
    MOVWF count
    RETFIE                  ;Return from interruption.

adc_int_handler:
    BANKSEL ADRESH
    MOVFF ADRESH, adc_read  ;Copy the 8 MSBs from the ADC
                            ;conversion to a variable.
    CALL SENDCHAR           ;The value to send is in address adc_read
    BANKSEL PIR1
    BCF PIR1,0              ;clear the ADC interrupt flag.
    RETFIE                  ;Return from interruption.
```

```
;============================
;Send a char using USART1
;============================
SENDCHAR:
   movlw 0x2F                ;number to send by usart 1
   movwf 0x0F
   BANKSEL PIR3
   btfss PIR3,4              ;TXIF| Check, is TX buffer full?
   bra SENDCHAR              ;IF not THEN try again.
   BANKSEL TX1REG
   movff adc_read,TX1REG     ;ELSE copy to USART TX register.
   RETURN

SENDPROTOCOLO:
   movlw 0x2F                ;number to send by usart 1
   movwf 0x0F
   BANKSEL PIR3
   btfss PIR3,4              ;TXIF ; Check, is TX buffer full?
   bra SENDPROTOCOLO          ;IF not THEN try again
   BANKSEL TX1REG
   movff protocolo,TX1REG     ;ELSE copy to USART TX register
   RETURN
end
```

Codigo MATLAB PROJ_plot_tensao.m

```
%Configuração da PORTA Série em MATLAB
if ~isempty(instrfind)
   fclose(instrfind);
   delete(instrfind);
end
close all;
clc;
s = serial('COM5','BaudRate',19200);
set(s,'FlowControl','none');
set(s,'Parity','none');
set(s,'InputBufferSize',2);
s.Terminator="";
s.ReadAsyncMode='continuous';
set(s,'Databits',8);
set(s,'StopBit',1);
set(s,'Timeout',100);
fopen(s);
flushinput(s);

%Definição das variáveis.
i=1;
j=1;
N=1200;
xt=1;
yt=1;
zt=1;
ts=0.012;
a=zeros(2,1);
x=zeros(1,1);
xv=zeros(2,1);
y=zeros(1,1);
yv=zeros(2,1);
z=zeros(1,1);
zv=zeros(2,1);

%Enlarge figure to full screen.
set(gcf, 'Units', 'Normalized', 'OuterPosition', [0, 0.04, 1, 0.96]);

%Ciclo de amostragem e plot.
tic
while(i<=N/12)
    %Ciclo de 'REFRESH' do plot após N/12 amostragens.
```

```
if (i==N/12)
     i=1;
     a=zeros(2,1);
     x=zeros(1,1);
     xv=zeros(2,1);
     y=zeros(1,1);
     yv=zeros(2,1);
     z=zeros(1,1);
     zv=zeros(2,1);
     xt=1;
     yt=1;
     zt=1;
end
while(j<=12)
    idn=fscanf(s);
    int=dec2bin(idn);
    dec=bin2dec(int);
    a(1,i)=dec(1);
    a(2,i)=dec(2);
    %Distribuição dos valores de aceleração em x, y e z para
    %as respetivas listas.
    if a(1,i)== 255
     x(1,xt) = a(2,i);
     %Conversão de valor binário em tensão de x.
     xv(2,xt)= (x(1,xt)*3.24/255);
     xv(1,xt)= (xt-1)*ts;
     xt=xt+1;
    end

    if a(1,i) == 254
     y(1,yt) = a(2,i);
     %Conversão de valor binário em tensão de y.
     yv(2,yt)= (y(1,yt)*3.24/255);
     yv(1,yt)= (yt-1)*ts;
     yt=yt+1;
    end

    if a(1,i) == 253
     z(1,zt) = a(2,i);
     %Conversão de valor binário em tensão de z.
     zv(2,zt)= (z(1,zt)*3.24/255);
     zv(1,zt)= (zt-1)*ts;
     zt=zt+1;
    end
    j=j+1;
end
%Plot das tensões dos 3 eixos.
j=1;
subplot(311)
plot(xv(1,4:4:i*4),xv(2,4:4:i*4),'r')
xlabel('Tempo (s)')
ylabel('V_x (volt)','FontSize',12)
title('Gráfico da Tensão no Canal X:','FontSize',14)
axis([0 ((N/3-1)*ts) -0.25 4])
subplot(312)
plot(yv(1,4:4:i*4),yv(2,4:4:i*4),'b')
xlabel('Tempo (s)')
ylabel('V_y (volt)','FontSize',12)
title('Gráfico da Tensão no Canal Y:','FontSize',14)
axis([0 ((N/3-1)*ts) -0.25 4])
subplot(313)
```

```
    plot(zv(1,4:4:i*4),zv(2,4:4:i*4),'k')
    xlabel('Tempo (s)')
    ylabel('V_z (volt)','FontSize',12)
    title('Gráfico da Tensão no Canal Z:','FontSize',14)
    axis([0 ((N/3-1)*ts) -0.25 4])
    pause(0.000001);
    i=i+1;
end
toc
```

Codigo MATLAB PROJ_plot_aceleracoes.m

```
%Configuração da PORTA Série em MATLAB
if ~isempty(instrfind)
    fclose(instrfind);
    delete(instrfind);
end

close all;
clc;
s = serial('COM5','BaudRate',19200);
set(s,'FlowControl','none');
set(s,'Parity','none');
set(s,'InputBufferSize',2);
s.Terminator="";
s.ReadAsyncMode='continuous';
set(s,'Databits',8);
set(s,'StopBit',1);
set(s,'Timeout',100);
fopen(s);
flushinput(s);

%Definição das variáveis.
i=1;
j=1;
N=1200;
xt=1;
yt=1;
zt=1;
ts=0.011;
a=zeros(2,1);
x=zeros(1,1);
xa=zeros(2,1);
y=zeros(1,1);
ya=zeros(2,1);
z=zeros(1,1);
za=zeros(2,1);

%Enlarge figure to full screen.
set(gcf, 'Units', 'Normalized', 'OuterPosition', [0, 0.04, 1, 0.96]);

%Ciclo de amostragem e plot.
tic
while(i<=N/12)
    %Ciclo de 'REFRESH' do plot após N/12 amostragens.
```

```
 if (i==N/12)
     i=1;
     a=zeros(2,1);
     x=zeros(1,1);
     xv=zeros(2,1);
     y=zeros(1,1);
     yv=zeros(2,1);
     z=zeros(1,1);
     zv=zeros(2,1);
     xt=1;
     yt=1;
     zt=1;
 end
 %Distribuição dos valores de aceleração em x, y e z para
 %as respetivas listas.
 while(j<=12)
    idn=fscanf(s);
    int=dec2bin(idn);
    dec=bin2dec(int);
    a(1,i)=dec(1);
    a(2,i)=dec(2);
    if a(1,i)== 255
     x(1,xt) = a(2,i);
     %Conversão de valor binário em aceleraçãode x.
     xa(2,xt)= ((x(1,xt)*3.24/255)-1.60)*9.81/0.8;
     xa(1,xt)= (xt-1)*ts;
     xt=xt+1;
    end
    if a(1,i) == 254
     y(1,yt) = a(2,i);
     %Conversão de valor binário em aceleraçãode y.
     ya(2,yt)= ((y(1,yt)*3.24/255)-1.755)*9.81/0.8;
     ya(1,yt)=(yt-1)*ts;
     yt=yt+1;

    end
    if a(1,i) == 253
     z(1,zt) = a(2,i);
     %Conversão de valor binário em aceleraçãode z.
     za(2,zt)= ((z(1,zt)*3.24/255)-1.91)*9.81/0.8;
     za(1,zt)= (zt-1)*ts;
     zt=zt+1;
    end
    j=j+1;
 end
%Plot das acelerações dos 3 eixos.
j=1;
subplot(311)
plot(xa(1,4:4:i*4),xa(2,4:4:i*4),'r')
xlabel('Tempo (s)')
ylabel('a_x (m/s^2)','FontSize',12)
title('Gráfico da Aceleração em X:','FontSize',14)
axis([0 ((N/3-1)*ts) -24 24])
subplot(312)
plot(ya(1,4:4:i*4),ya(2,4:4:i*4),'b')
xlabel('Tempo (s)')
ylabel('a_x (ms^2)','FontSize',12)
title('Gráfico da Aceleração em Y:','FontSize',14)
axis([0 ((N/3-1)*ts) -24 24])
subplot(313)
plot(za(1,4:4:i*4),za(2,4:4:i*4),'k')
```

```
    xlabel('Tempo (s)')
    ylabel('a_x (m/s^2)','FontSize',12)
    title('Gráfico da Aceleração em Z:','FontSize',14)
    axis([0 ((N/3-1)*ts) -24 24])
    pause(0.000001);
    i=i+1;
end
toc
```

Codigo MATLAB PROJ_plot_posicao.m

```
%Configuração da PORTA Série em MATLAB
if ~isempty(instrfind)
    fclose(instrfind);
    delete(instrfind);
end

close all;
clc;
s = serial('COM5','BaudRate',19200);
set(s,'FlowControl','none');
set(s,'Parity','none');
set(s,'InputBufferSize',2);
s.Terminator="";
s.ReadAsyncMode='continuous';
set(s,'Databits',8);
set(s,'StopBit',1);
set(s,'Timeout',100);
fopen(s);
flushinput(s);

%Definição das variáveis.
i=1;
j=1;
N=1200;
a=zeros(2,1);
xa=zeros(2,1);
xv=zeros(2,1);
xx=zeros(2,1);
ya=zeros(2,1);
yv=zeros(2,1);
yy=zeros(2,1);
za=zeros(2,1);
zv=zeros(2,1);
zz=zeros(2,1);
xt=1;
yt=1;
zt=1;
ts=0.1966;

%Enlarge figure to full screen.
set(gcf, 'Units', 'Normalized', 'OuterPosition', [0, 0.04, 1, 0.96]);

%Ciclo de amostragem e plot.
tic
```

```
while(i<=N)
    if (i==200)
        i=1;
        a=zeros(2,1);
        x=zeros(1,1);
        xa=zeros(2,1);
        xv=zeros(2,1);
        xx=zeros(2,1);
        y=zeros(1,1);
        ya=zeros(2,1);
        yv=zeros(2,1);
        yy=zeros(2,1);
        z=zeros(1,1);
        zv=zeros(2,1);
        za=zeros(2,1);
        zz=zeros(2,1);
        xt=1;
        yt=1;
        zt=1;
    end
    for j=1:3
        idn=fscanf(s);
        int=dec2bin(idn);
        dec=bin2dec(int);
        a(1,i)=dec(1);
        a(2,i)=dec(2);
        %Distribuição dos valores de aceleração em x, y e z para
        %as respetivas listas.
        if a(1,i)== 255
            x(1,xt) = a(2,i);
            xa(2,xt)=((x(1,xt)*3.24/255)-1.655)*9.81/0.8;
            xa(1,xt)=(xt-1)*ts;
            if(xt~=1)
                %Integração dos valores de aceleração em velocidade e
                %de seguida posição.
                xv(2,xt)=ts*trapz(xa(2,(xt-1):xt));
                xv(1,xt)=(xt-1)*ts;
                xx(2,xt)=ts*trapz(xv(2,(xt-1):xt));
                xx(1,xt)=(xt-1)*ts;
            end
            xt=xt+1;
        end
        if a(1,i) == 254
            y(1,yt) = a(2,i);
            ya(2,yt)=((y(1,yt)*3.24/255)-1.755)*9.81/0.8;
            ya(1,yt)=(yt-1)*ts;
            if(yt~=1)
                %Integração dos valores de aceleração em velocidade e
                %de seguida posição.
                yv(2,yt)=ts*trapz(ya(2,(yt-1):yt));
                yv(1,yt)=(yt-1)*ts;
                yy(2,yt)=ts*trapz(yv(2,(yt-1):yt));
                yy(1,yt)=(yt-1)*ts;
            end
            yt=yt+1;
        end
        if a(1,i) == 253
            z(1,zt) = a(2,i);
            za(2,zt)=((z(1,zt)*3.24/255)-1.91)*9.81/0.8;
            za(1,zt)=(zt-1)*ts;
            if(zt~=1)
```

```
                %Integração dos valores de aceleração em velocidade e
                %de seguida posição.
                zv(2,zt)=ts*trapz(za(2,(zt-1):zt));
                zv(1,zt)=(zt-1)*ts;
                zz(2,zt)=ts*trapz(xv(2,(zt-1):zt));
                zz(1,zt)=(zt-1)*ts;
            end
            zt=zt+1;
        end
    end
   %Plot das 3 coordenadas.
   plot3(xx(2,:),yy(2,:),zz(2,:),'-o')
   grid on
   xlabel('x(t) (m)','FontSize',12)
   ylabel('y(t) (m)','FontSize',12)
   zlabel('z(t) (m)','FontSize',12)
   title('Posição em função do tempo
r[x(t),y(t),z(t)]:','FontSize',14)
   pause(0.00000001);
   i=i+1;
end
toc
```