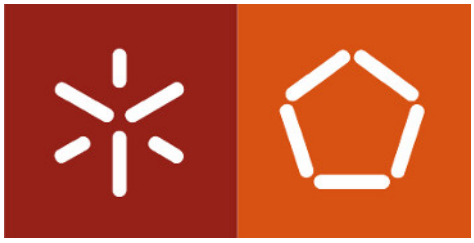
# Projeto Final

Leander Reascos
A89154

Tiago Pereira
A85893

Dezembro 2020



**-Microcontroladores e Interfaces-**
**Docente**:
· Paulo Mendes

# Índice

# 1 Introdução ao Projeto

Nos dia de hoje, a automação de sistemas implica o controlo e monitorização de variáveis que estão em constante mudança. Essas mudanças físicas são detetadas através da utilização de sensores que as traduzem em sinais elétricos. Estes sinais de natureza usualmente analógica requerem um tratamento para poderem ser lidos e posteriormente processados. As unidades que processam estes sinais não necessitam de um grande poder de processamento, pelo que, são geralmente utilizados microcontroladores que recebem informação dos sensores e as enviam para unidades com poder de processamento mais elevado.

No projeto em questão, o sensor trata-se de um acelerómetro, o qual traduz em três sinais analógicos distintos as acelerações nos três eixos espaciais que atuam no dispositivo. Este sensor é ligado a um microcontrolador PIC18F47Q10 que trata da multiplexagem dos sinais analógicos para conversão por um ADC e, posteriormente, formata as informações convertidas e as envia de acordo com o protocolo de comunicação série definido para um computador, que processa esta informação de acordo com a aplicação desejada.

## 1.1 Aplicações

Com base no sensor disponível, existiam algumas aplicações que poderiam ser adotadas, tais como o *tracking* da posição, deteção de picos aceleração, deteção de orientação, etc. O objetivo principal do projeto foi o *tracking* da posição do acelerómetro, tendo sido também desenvolvidas aplicações focadas na deteção da aceleração e orientação do microcontrolador.

## 1.2 Montagem do Circuito

Para assegurar a comunicação entre todos os módulos do sistema, foi necessário conecta-los fisicamente, uma vez que, não se utilizam módulos para comunicação sem-fios. Com isto, foi necessário assegurar a alimentação de todos os componentes, bem como ligação de cada *output* analógico do acelerómetro ao microcontrolador e a ligação deste último ao computador para comunicação serial, como exposto na figura 1.

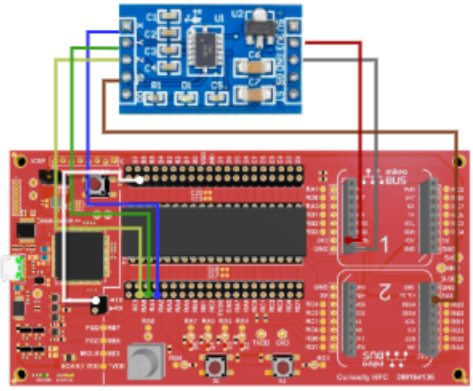


Fig. 1: Esquema de montagem do microcontrolador e acelerómetro.

De um ponto de vista da comunicação entre módulos, o microcontrolador é um elemento fulcral tendo em conta que establece a "ponte" de comunicação entre o sensor e o computador (figura 2).


Fig. 2: Diagrama de blocos do sistema.

## 2 Programação do Microcontrolador

O microcontrolador sendo uma ponte de comunicação entre sensor e computador tem de ser cuidadosamente programado para exercer as suas funções. Assim, o modo de funcionamento deste pode ser decomposto em três partes:

- Conversão dos valores analógicos do acelerómetro.
- Multiplexagem entre eixos do acelerómetro.
- Formatação e envio dos Chars com informação para o computador.

De um modo geral, o seu funcionamento pode ser descrito pelo fluxograma da figura 3.

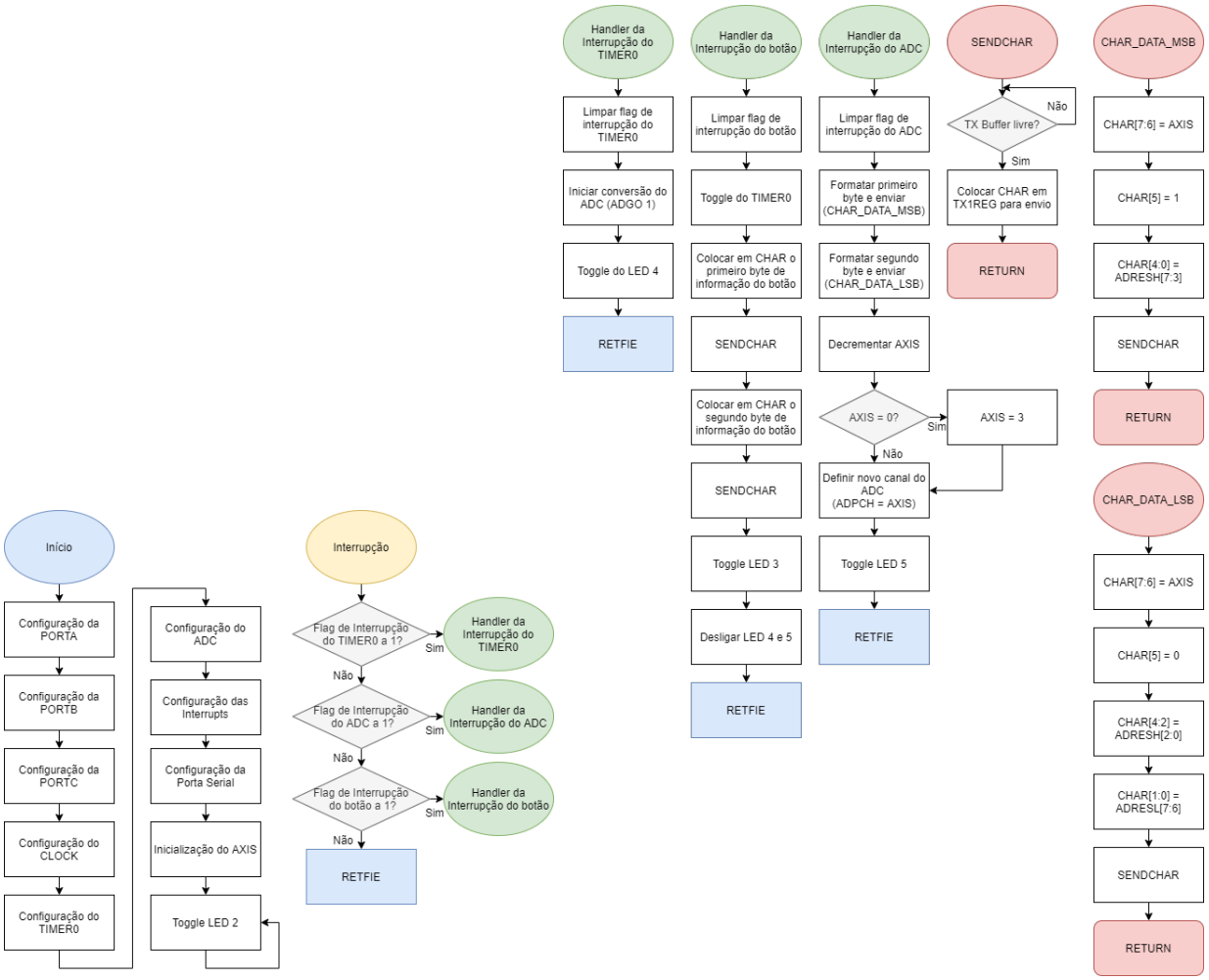



Fig. 3: Fluxograma do programa do microcontrolador.

## 2.1 Amostragem

De forma a termos o nosso sitema com os tempos de conversão, amostragem, comunicação e representação, sem que haja problemas como aliasing e garantir que a informação representada posteriormente pela aplicação desejada não apresente nenhum tipo de perdas significativas, fez-se a análise dos sinais.

O computador nas diferentes aplicações desenvolvidas tem duas frequências para representar os dados obtidos, isto é 20fps e 60fps. Outro dado a ter em conta são os tipos de movimentos de interesse, estes movimentos são executados por um utilizador que é incapaz de produzir sinais de frequência elevada, logo podemos limitar a largura de banda do nosso sistema a 10 Hz. Assim, a frequência de Nyquist seria:

$$f_{\text{Nyquist}} = 20\,Hz \tag{1}$$

A representação gráfica do sinal pode ser interpretada como uma decimação do sinal, pois como se garante que nao há perdas significativas ou nenhuma perda na receção e tratamento dos dados pelo computador, a representação seria uma nova amostragem das sequências previamente amostradas. Desta forma, precisou-se ter em atenção a frequencia à qual o sinal analógico é amostrado inicialmente.

Como para a representação gráfica do sinal a 60 fps vai ocorrer um redução da frequência máxima de amostragem de pelo menos 2 vezes, ou seja, decimar o sinal por um fator mínimo de 2, temos um possivel valor para a frequência de amostragem do sinal.

$$f_{\text{amostragem}} = 120\,Hz \tag{2}$$

Esta frequência é 6 vezes superior à frequencia de Nyquist e garante também que a decimação feita a 20 Hz seja feita de forma a não apresentar aliasing em relação à gama de frequências definidas, pois a redução de 120 Hz para 20 Hz implica que a frequência máxima do sinal seja de $f_{\text{amostragem}}/6$, ou seja, 20 Hz que é a $f_{\text{Nyquist}}$.

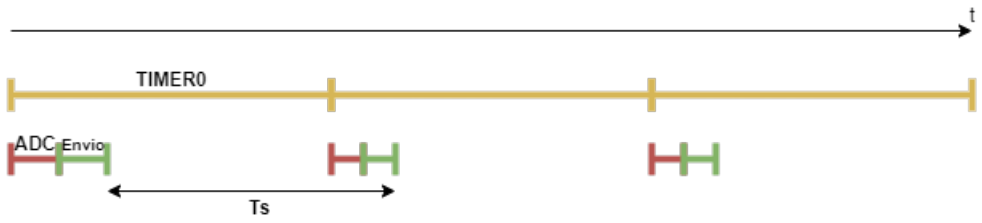



Fig. 4: Esquema temporal dos processos do ADC, não à escala. Repare-se que $T_S = TIMER0$.

Para se obter esta frequência de amostragem desejada, é preciso configurar o Timer do microcontrolador, pois é este quem define o tempo no qual os sinais são convertidos (figura 4). Os 3 sinais são convertidos ciclicamente, portanto a frequência de amostragem terá de ser 3 vezes superior.

$$f_{\text{TimerInterrupt}} = 3 \times f_{\text{amostragem}} = 360\,Hz \tag{3}$$

Como existe uma interrupção do timer a cada 2 ciclos deste (contagem de 0 a 1), a frequência do timer terá de ser 2 vezes superior à frequência da interrupção.

$$f_{\text{Timer}} = 2 \times f_{\text{TimerInterrupt}} = 720\,Hz \tag{4}$$

A amostragem dos valores de tensão provenientes do acelerómetro é levada a cabo pelo ADC conectado ao microcontrolador. De forma à amostragem se assemelhar a uma amostragem feita por trem de impulsos, ou seja, uma amostragem ideal, a frequência do ADC tem de ter um valor para o

qual o seu tempo de conversão possa ser desprezado na análise, para este caso em concreto, foi procurado um tempo de conversão de 0.15% do tempo de amostragem. Este ADC tem as suas limitações, pelo que, é necessária a sua configuração de forma a manter a fiabilidade das conversões efetuadas. Como tal, definiu-se um *clock* do ADC que respeitasse os tempos de conversão mínimos requiridos para um resultado de 10 bits.

Dadas as limitações do Hardware para ser configurado a valores precisos, mas de forma a garantir a análise feita, as frequências finais são as seguintes:

$$\begin{aligned} f_{\text{Timer}} &= 781.25\,Hz \\ f_{\text{TimerInterrupt}} &= 390.63\,Hz \\ f_{\text{Amostragem}} &= 130.20\,Hz \\ f_{\text{ADC}} &= 250\,KHz \end{aligned}$$

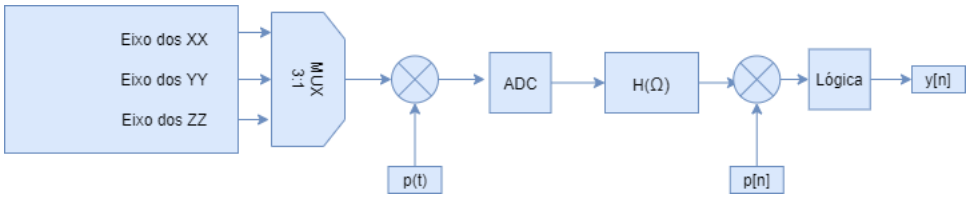



Fig. 5: Esquema do sistema que permite a amostragem do sinal e posterior filtragem e processamento usado como modelo para o cálculo das frequências de funcionamento.

## 2.2 Comunicação

Para a comunicação entre o microcontrolador e o computador, foi utilizado o *USART* no qual foi necessário configurar vários aspetos de forma a se garantir os requisitos necessários para um envio de informação eficiente que permitisse o funcionamento fiável do programa.

### 2.2.1 Definição do *CHAR*

Um dos aspetos a definir foi a informação contida em cada *char* enviado desdo microcontrolador. Uma vez que cada *char* é uma palavra de 8 bits e no total existem 10 bits que correspondem à conversão efetuada pelo ADC, a informação respeitante a cada amostra não pode ser enviada de uma vez só. Para além do mais, cada amostra pode pertencer a três eixos distintos e é necessário distingui-los. Face a estes problemas optou-se pela formatação do *char* divindindo os 8 bits em três partes dinstintas, como ilustrado na figura 6.

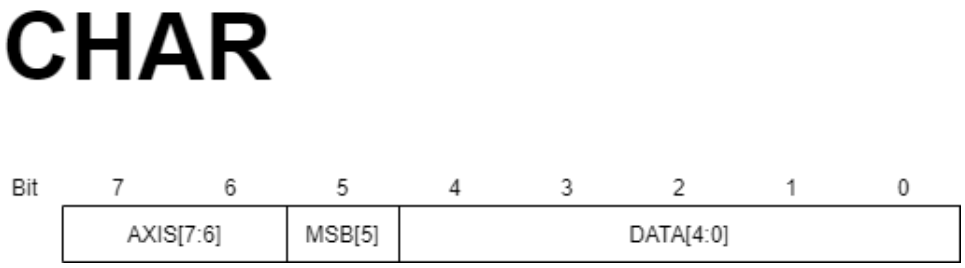



Fig. 6: Organização dos bits em cada *char*

Assim, cada secção do *char* contém informações distintas (ver Anexo A):

- **AXIS**: indica se o *char* contém informação acerca do eixo xx, yy, zz ou sobre o acionamento dos botões.
- **MSB**: indica se o *char* contém os bits mais significativos ou menos significativos da data.
- **DATA**: são os bits de *data* do *char*.

Com isto, a cada 2 *char* enviados obtém-se a informação referente a uma amostra realizada para ser posteriormente processada sem ambiguidades.

### 2.2.2 *Baud Rate*

Como visto na figura 3, a cada conversão do ADC são enviados os carateres ao computador, isto é, 16 bits no total por cada conversão. Como também foi visto na análise da amostragem, a frequência de amostragem e por consequência a frequência a que se enviam os valores é dada pela $f_{\text{TimerInterrupt}} = 390.63\,Hz$, ou seja, a cada $t_{\text{envio}} = 2.56\,ms$ são enviados 16 bits. De forma a garantir que não há perdas nem sobrecarga da porta série o baudrate mínimo será dado por:

$$\text{baudrate} = 16 \times f_{\text{TimerInterrupt}} = 6250\,bps \tag{5}$$

De forma a obter-se 0% de erros na comunicação, como é dado no datasheet, o baudrate para o relógio do microcontrolador a $8\,MHz$ é 10417, que é superior ao baudrate mínimo necessário para não existir problemas na comunicação.

## 2.3 Configurações do Microcontrolador

Face aos aspetos discutidos, configurou-se o microcontrolador de tal forma a respeitar todas as condições para o seu bom funcionamento. Nas tabelas 1 à 8 encontram-se expostos os parâmetros de configuração necessários referentes aos blocos de configuração no fluxograma da figura 3.

| *Registo* | *Valor* | *Configuração* |
|---|---|---|
| LATA | 0b00000000 | Limpar valores guardados na *Latch* A |
| TRISA | 0b00001111 | Definir RA[0:3] como inputs |
| ANSELA | 0b00001111 | Definir RA[0:3] como portas analógicas |

Tabela 1: Configurações da PORTA.

| *Registo* | *Valor* | *Configuração* |
|---|---|---|
| LATB | 0b00000000 | Limpar valores guardados na *Latch* B |
| TRISB | 0b00010000 | Definir RB[4] como input |
| ANSELB | 0b00000000 | Definir toda a PORTB como portas digitais |
| RB6PPS | 0bxxx01001 | Definir como output do pino RB6 o EUSART1 (TX/CK) |
| INT0PPS | 0bxxx01100 | Definir RB4 como input do Peripheral Interrupt 0 |

Tabela 2: Configurações da PORTB.

| *Registo* | *Valor* | *Configuração* |
|---|---|---|
| LATC | 0b00000000 | Limpar valores guardados na *Latch* C |
| TRISB | 0b00100000 | Definir RC5 como input |
| ANSELB | 0b00000000 | Definir toda a PORTC como portas digitais |
| INT1PPS | 0bxxx10101 | Definir RC5 como input do Peripheral Interrupt 1 |

Tabela 3: Configurações da PORTC.

| *Registo* | *Valor* | *Configuração* |
|---|---|---|
| OSCFRQ | 0b00000011 | Selecionar 8MHz para HFINTOSC |
| OSCCON1 | 0bx1100000 | Definir HFINTOSC como novo oscilador com uma divisão do CLOCK de 1 |
| OSCEN | 0b010000xx | Habilitar o HFINTOSC com as configurações já definidas |

Tabela 4: Configurações do CLOCK.

| *Registo* | *Valor* | *Configuração* |
|---|---|---|
| T0CON0 | 0b0x000100 | Timer0 desligado, timer de 8 bits e postscaler 1:5 |
| T0CON1 | 0b01101011 | HFINTOSC como source, TMR0 sincronizado com FOSC/4, prescaler 1:2048 |
| TMR0L | 0b00000000 | Limpar o contador do timer de 8 bits |
| TMR0H | 0b00000001 | Definir o valor de comparação do timer de 8 bits como 1. |

Tabela 5: Configurações do TIMER0.

| *Registo* | *Valor* | *Configuração* |
|---|---|---|
| ADPCH | 0b00000011 | Definir RA3 como input do ADC |
| ADREF | 0bxxx0xx00 | Definir $V_{REF-}$ como $V_{SS}$ e $V_{REF+}$ como $V_{DD}$ |
| ADCLK | 0bxx001111 | Definir clock do ADC como FOSC/$(2 * (15 + 1))$=FOSC/32 |
| ADCON0 | 0b10x0x0x0 | Habilitar o ADC |

Tabela 6: Configurações do ADC.

| *Registo* | *Valor* | *Configuração* |
|---|---|---|
| PIR0 | 0bxx0x0000 | Limpar flags de interrupção do TMR0 e do INT0 |
| PIR1 | 0b00xxxx00 | Limpar flag de interrupção do ADC |
| PIE0 | 0bxx01x001 | Habilitar interrupções do TMR0 e do INT0 |
| PIE1 | 0b00xxxx01 | Habilitar interrupções do ADC |
| INTCON | 0b110xx000 | Habilitar interrupções globalmente e interrupções de periféricos |

Tabela 7: Configurações das Interrupções.

| *Registo* | *Valor* | *Configuração* |
|---|---|---|
| SP1BRGL | 0b00001011 | Bits menos significativos para derivar baud rate do USART1 $\frac{F_{OSC}}{64\times(SP1BRG+1)} \approx 10417$ |
| SP1BRGH | 0b00000000 | Bits mais significativos para derivar baud rate do USART1 $\frac{F_{OSC}}{64\times(SP1BRG+1)} \approx 10417$ |
| TX1STA | 0b10100000 | Clock interno, transmissão 8 bits, habilitado, assíncrono, baixa velocidade |
| RC1STA | 0b10000000 | Habilitar porta serial |

Tabela 8: Configurações da Porta Serial.

# 3 Processamento de Dados

Todas as aplicações desenvolvidas apresentam a mesma base de tratamento dos dados recebidos pela porta série. Inicialmente quando conseguida com sucesso a ligação entre o microcontrolador e o computador, é iniciado um proceso em *background* para a leitura assíncrona com o programa principal dos dados recebidos no buffer da porta série. O programa fica em *standby* até receber o sinal do botão que vem do microcontrolador, este sinal indica ao computador que vai começar o programa para o qual deve calibrar os valores que recebe quando o acelerómetro está em "repouso". Neste momento continua o programa principal e a leitura em background, a qual faz o tratamento e armazenamento dos dados.

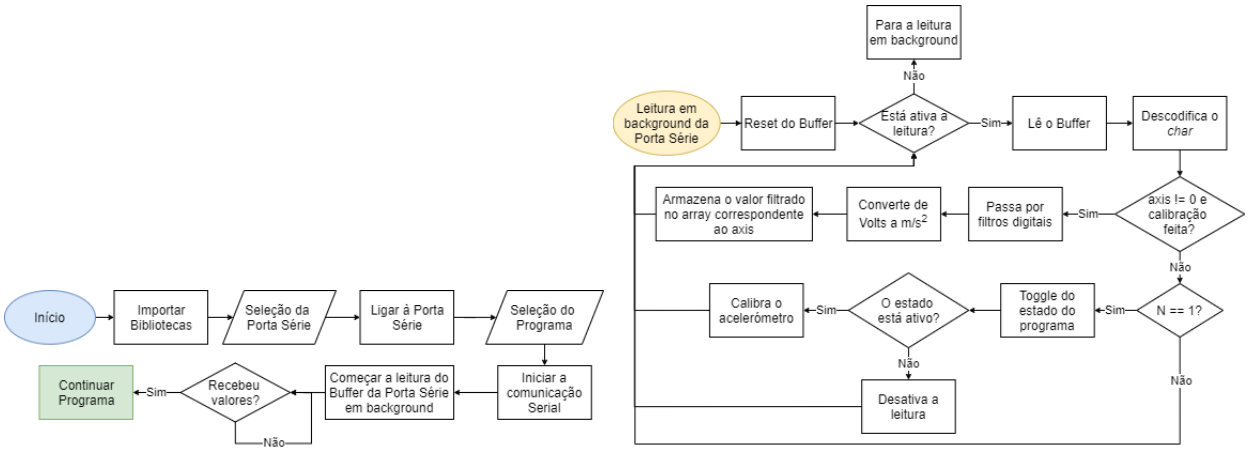



Fig. 7: Fluxograma da inicialização do programa e da leitura de dados em background.

## 3.1 Processamento Digital do Sinal

Dada a limitação no número de componentes físicos, optou-se por uma filtragem e processamento digital do sinal recebido. Desta forma, a única limitação seria a capacidade de processamento do computador em questão ou a complexidade do filtro aplicado.

O sinal recebido foi processado três vezes antes da sua conversão (figura 8) para valores de aceleração "úteis": filtragem passa baixo, redução do ruído branco em torno do valor médio do sinal e filtragem com média móvel.

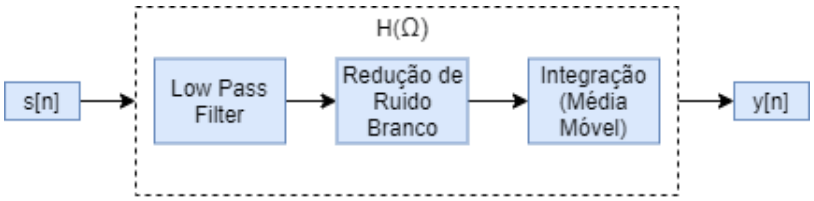



Fig. 8: Diagrama de blocos do sistema de processamento digital do sinal recebido, s[n].

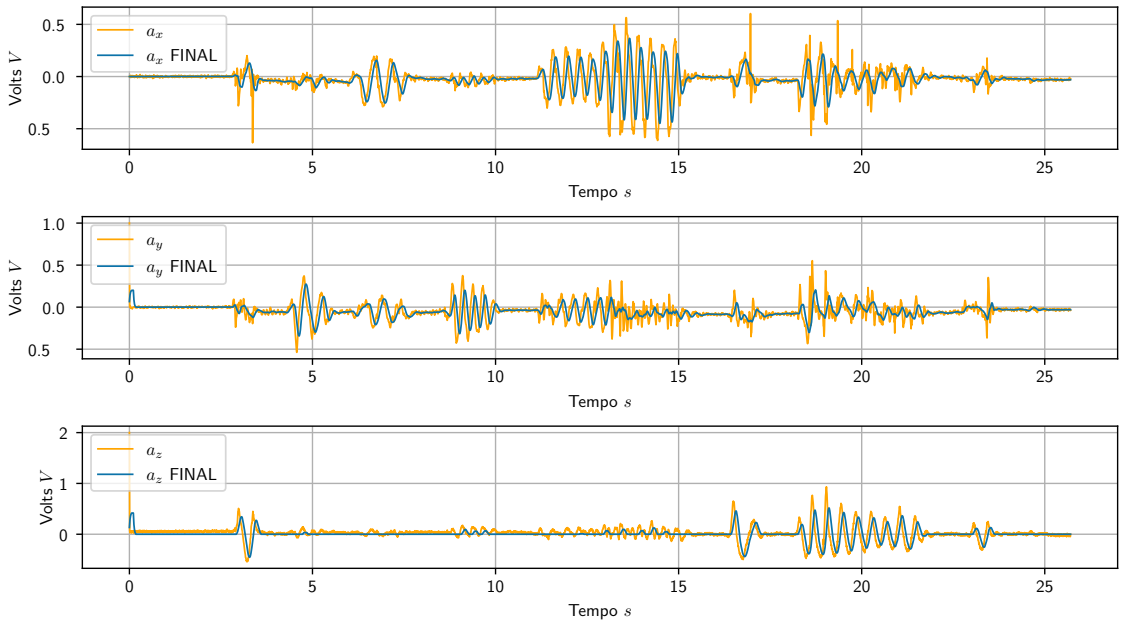



Fig. 9: Tensão (sem componente DC média) em função do tempo do sinal processado (azul) e por processar (laranja) nos três eixos de orientação.

### 3.1.1 Low Pass Filter

Em qualquer sistema de aquisição é crucial a redução da largura de banda do sinal. Como tal, utilizou-se um filtro digital passa baixo com a equação de diferenças

$$y[n] = \alpha y[n-1] + (1-\alpha)\, x[n] \tag{6}$$

onde a frequencia de corte é dada por

$$f_c = f_s \frac{1-\alpha}{2\pi\alpha} \tag{7}$$

e $\alpha$ por

$$\alpha = \frac{f_s}{2\pi f_c + f_s} \tag{8}$$

Fixou-se $f_c = 10Hz$ e aplicou-se o filtro, o que resultou numa diminuição da amplitude do espetro do sinal para frequências mais elevadas (figura 10).

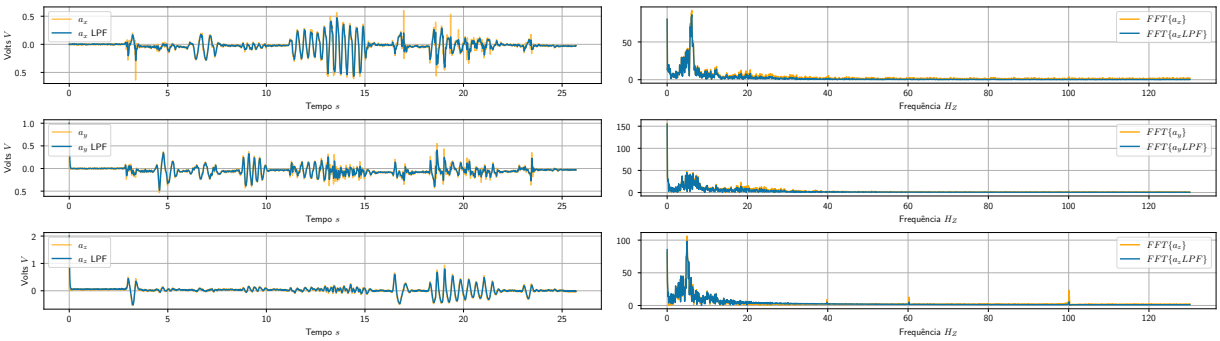



Fig. 10: Tensão (sem componente DC média) em função do tempo do sinal antes e após de ser aplicado o LPF (esquerda) e o seu respetivo espetro (direita).

### 3.1.2 Redução do Ruído Branco

Mesmo com o acelerómetro parado e nivelado, existem flutuações da tensão do sinal. Para mitigar o efeito destas oscilações, calcula-se para um dado número de valores a média e respetivo desvio padrão, $\sigma$. Assim, todos os valores de tensão que estejam dentro de um dado intervalo dependente do desvio serão descartados, correspondendo a uma aceleração de 0 $m/s^2$.

$$y\left[n\right] = \begin{cases} x\left[n\right] & , \left|x\left[n\right]\right| \geq 1.5\sigma \\ 0 & , otherwise \end{cases} \tag{9}$$

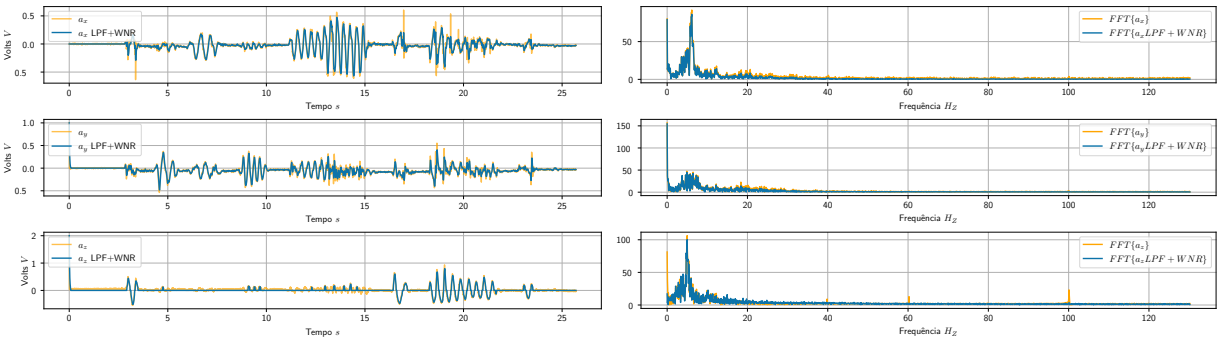



Fig. 11: Tensão (sem componente DC média) em função do tempo do sinal antes e após de ser aplicada a WNR (*White Noise Reduction*) (esquerda) e o seu respetivo espetro (direita).

### 3.1.3 Integração por Média Móvel

Um dos problemas do sinal proveniente do acelerómetro é que por vezes existem picos de tensão elevada que não descrevem corretamente o movimento efetuado. De forma a suavizar estes eventos, implementou-se um filtro de média móvel tal que a sua equação de diferenças é dada por

$$y\left[n\right] = \frac{1}{M}\sum_{j=0}^{M-1} y_1\left[n-j\right] \tag{10}$$

o que permitiu uma suavização das tensões e em simultâneo reduziu bastante o ruído de frequências mais elevadas (figura 12).

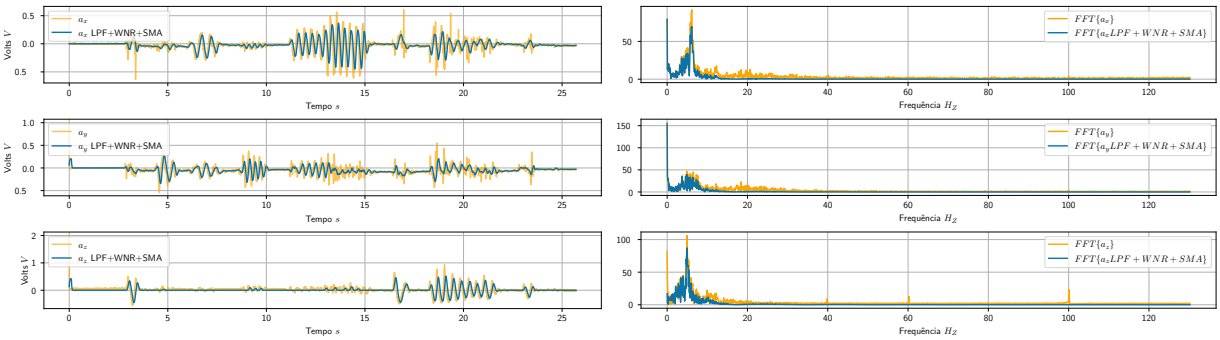



Fig. 12: Tensão (sem componente DC média) em função do tempo do sinal antes e após de ser aplicado o SMA (esquerda) e o seu respetivo espetro (direita).

## 3.2 Aplicações

### 3.2.1 Aceleração em *Tempo Real*

Após a calibração ser feita, cria os gráficos vazios, que serão atualizados frame a frame a uma frequência de 20fps lendo os últimos valores da aceleração que foram armazenados em background e atualiza os gráficos mostrados.

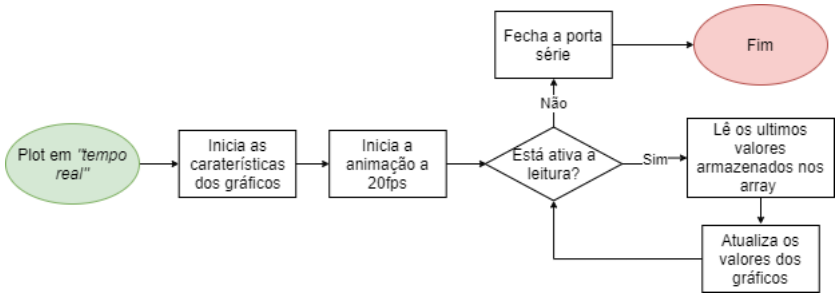



Fig. 13: Fluxograma da aplicação de exibição da aceleração em "tempo real".

### 3.2.2 Controlo por Orientação

O acelerómetro está por defeito configurado a $1g$ em repouso no eixo dos zz. Fazendo uso disto, pode-se calcular a orientação na qual se encontra o acelerómetro de acordo com o grafico da figura 14.

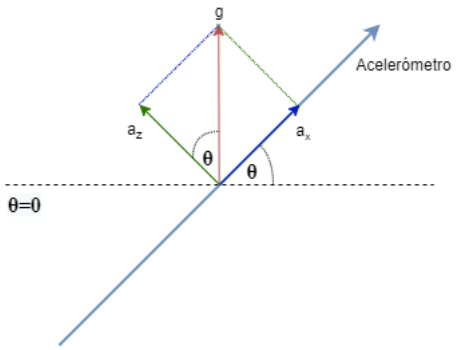



Fig. 14: Esquema de forças que atuam no acelerómetro em função da orientação.

Assim, o ângulo de inclinacâo é dado por:

$$\theta = \arcsin \frac{a_x}{g} \tag{11}$$

Aplicando esta lógica para o eixo dos yy calcula-se a orientação com respeito ao plano em xx e yy. O eixo dos zz indica em que sentido se encontra virado o acelerómetro, pois este pode encontrar-se virado para baixo e unicamente com os ângulos dos eixos xx e yy não é posivel determina-lo. Desta forma, pode-se ter várias combinações de comandos que podem ser programados e executados pela personagem *BB8* na implementação desenvolvida. Neste caso em concreto, tem-se os seguintes comandos:

| Aceleração | | |
|---|---|---|
| ZZ | XX | YY |
| >0 | Movimenta e orienta a personagem e à câmara proporcionalmente à inclinação | |
| >1.2g | Projétil no sentido do movimento e com velocidade proporcional à inclinação | |
| <-0.7g | Roda a câmara de acordo com à inclinação | |

Fig. 15: Condições para o movimento e acionamento de funções da aplicação.

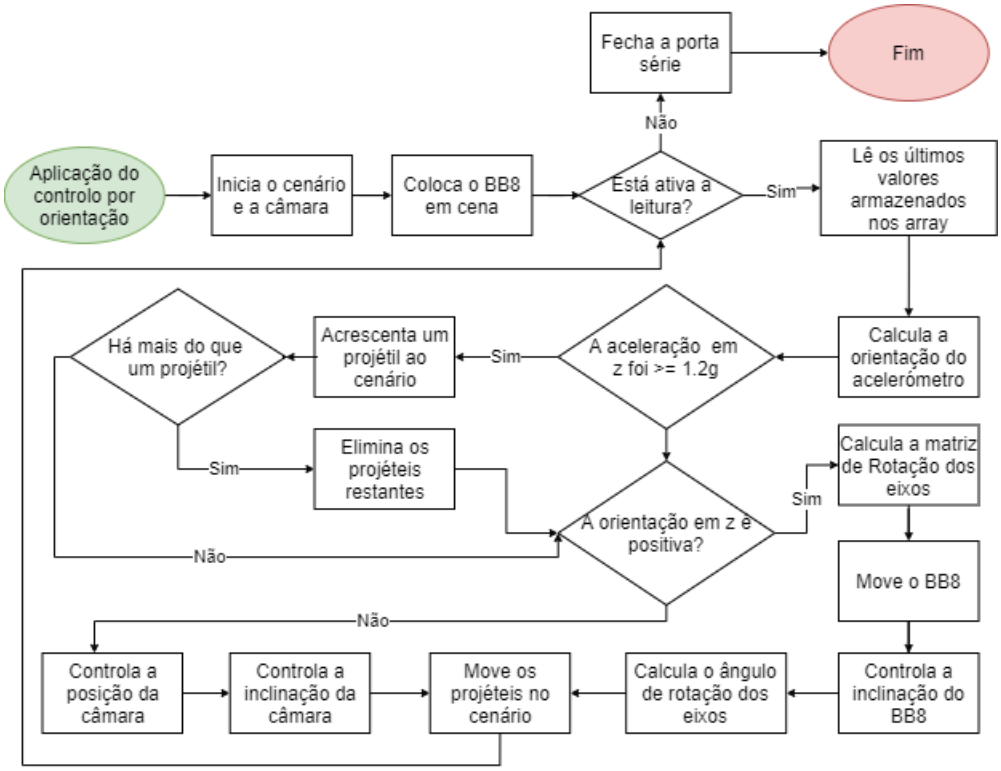



Fig. 16: Fluxograma da aplicação de controlo com base na orientação do acelerómetro.

### 3.2.3 *Tracker* de posição

O principal desafio neste ponto do trabalho foi, fazendo somente uso do acelerómetro, aproximar a posição do mesmo com a melhor precisão e exatidão possível, utilizando-se tudo o que já foi anteriormente discutido de forma a melhorar e corrigir os problemas que se foram apresentando para cumprir este objetivo.

Assim, de forma a calcular a posição, ou seja, integrar pelo método de Newton a aceleração com um $dt$ dado pela frequência de amostragem e de representação, chegando a gráficos de velocidade e finalmente da posição. Foram aplicadas várias correções aos valores de aceleração pois estas não eram exatas o suficiente para, por exemplo, detetar travagens totais do movimento, ou mesmo a própria orientação do acelerómetro ao devolver valores constantes de aceleração, pois para o programa implementado isto era considerado como acelerar a uma taxa constante. Assim, para corrigir isto foram usadas parte das funcionalidades anteriormente descritas como o cálculo da orientação do acelerómetro e correção das acelerações obtidas, enquanto para a velocidade, partindo do princípio de que não seria possível ter, para a aplicação em causa, velocidades constantes por um longo periodo de tempo, esta é corrigida quando acontece isto no software. As equações usadas para o efeito foram as seguintes:

$$v\,[n] = v\,[n-1] + dt\,a\,[n] \tag{12}$$

$$x\,[n] = x\,[n-1] + dt\,v\,[n] \tag{13}$$

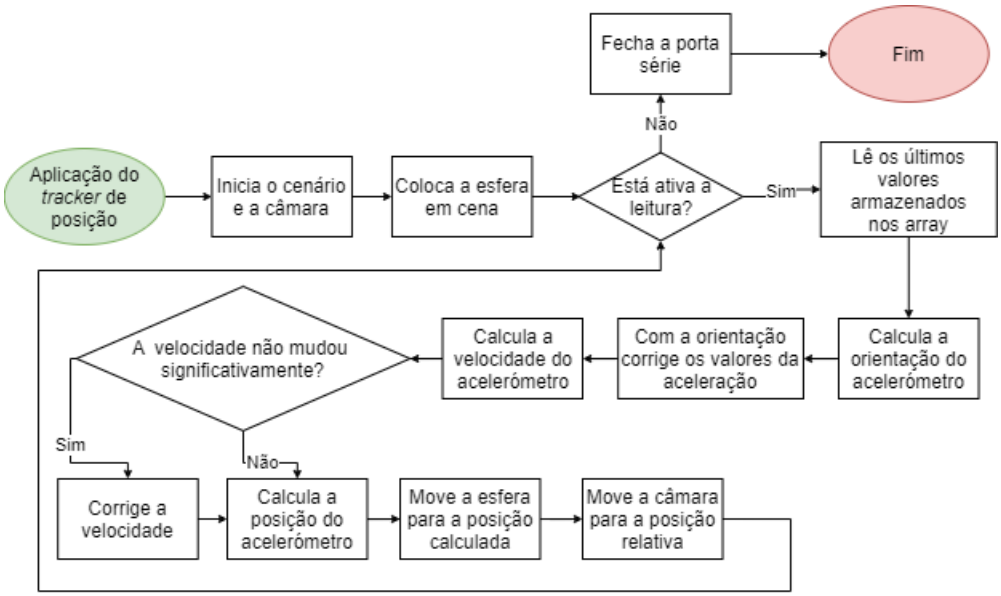



Fig. 17: Fluxograma da aplicação de *tracking* da posição.

# 4 Conclusão

De forma geral, o objetivo do projeto foi alcançado, tendo-se estabelecido uma comunicação serial eficiente e fiável entre o microcontrolador e o computador, respeitando todas as implicações que o espetro do sinal e o teorema de Nysquist trouxeram de forma a evitar a perda de informação importante.

Do ponto de vista do condicionamento do sinal, dada a falta de material para a criação de filtros analógicos, os filtros digitais aplicados conseguiram substituir eficientemente o comportamento dos primeiros. Estes mostraram-se de elevada importância para a obtenção de um sinal mais "limpo" e correspondente à realidade percetível, ajudando a obter melhores resultados nas posteriores aplicações.

As diferentes aplicações desenvolvidas mostraram que um acelerómetro, por si só, tem capacidades limitadas. Este mostrou-se suficiente para a representação de acelerações em tempo real, conseguindo-se um gráfico aceleração/tempo que correspondia satisfatoriamente aos movimentos do acelerómetro. Desta forma, poderia ser utilizado em várias áreas onde fosse necessária a monitorização de acelerações sofridas por um dado objeto, por exemplo um carro, e detetar *thresholds* de aceleração para o acionamento de funcionalidades.

Obteve-se também resultados bastante positivos no que toca à monitorização da orientação do sensor, sendo-se capaz de com base na inclinação que o acelerómetro tem e a ação da força da gravidade nos diferentes eixos, executar variadas funções. A obtenção da inclinação de um acelerómetro pode ser útil em vários ramos, desdo controlo de objetos (como feito na aplicação neste projeto), até à deteção de inclinações em estruturas, como se de um nível digital se tratasse.

No que toca à capacidade do acelerómetro como sensor para efetuar o *tracking* de objetos, este não é fiável sendo de muito difícil execução a obtenção de uma posição precisa com recurso a esta única ferramenta. O acelerómetro não tem precisão nem exatidão suficiente para uma integração dupla do seu sinal de forma a obter a posição em tempo real.

# 5 Bibliografia

[1] "Curiosity High Pin Count (HPC) Development Board User's Guide".
[2] "Curiosity-High-Pin-Count-Development-Board-User-Guide-40001856C.pdf".[Em linha]. Disponível em:
https://ww1.microchip.com/downloads/en/DeviceDoc/Curiosity-High-Pin-Count-Development-Board-User-Guide-40001856C.pdf.
[3] "Curiosity_HPC_Schematics_rev2.pdf". [Em linha]. Disponível em:
https://ww1.microchip.com/downloads/en/DeviceDoc/Curiosity_HPC_Schematics_rev2.pdf.
[4] "MMA7361LC.pdf". [Em linha]. Disponível em:
https://www.nxp.com/docs/en/data-sheet/MMA7361LC.pdf.
[5] "PIC18F27-47Q10-Data-Sheet-40002043E.pdf". [Em linha]. Disponível em:
https://ww1.microchip.com/downloads/en/DeviceDoc/PIC18F27-47Q10-Data-Sheet-40002043E.pdf.

# Anexo A - Formatação do CHAR

## CHAR

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | AXIS[7:6] | | MSB[5] | DATA[4:0] | | | | |

| AXIS[7:6] | |
|---|---|
| Value | Description |
| 11 | CHAR com data do eixo XX |
| 10 | CHAR com data do eixo YY |
| 01 | CHAR com data do eixo ZZ |
| 00 | CHAR de acionamento de botão |

| MSB[5] | | |
|---|---|---|
| Value | Condition | Description |
| 1 | AXIS ≠ 00 | CHAR com 5 MSB data |
| 0 | AXIS ≠ 00 | CHAR com 5 LSB data |
| 1 | AXIS = 00 | Primeiro CHAR de acionamento do botão |
| 0 | AXIS = 00 | Segundo CHAR de acionamento do botão |

| DATA[4:0] | | |
|---|---|---|
| Value | Condition | Description |
| 0 to 1F | AXIS ≠ 00 | Data bits |
| 0 to 1F | AXIS = 00 | Identificação do botão |

# Anexo B - Código dos Handlers das Interrupções

```
TIMER0_INT: ; Handler da Interrupt do TIEMR0
    BANKSEL PIR0
    BCF    PIR0   ,5   ; Limpar flag de interrupt do TIMER0
    BANKSEL ADCON0
    BSF    ADCON0   ,0   ; Comecar nova conversao do ADC
    BTG    LATA, 6    ; "Piscar" LED.
    RETFIE
```

```
ADC_Finish_Interrupt: ; Handler da Interrupt do ADC
    MOVLW 0b01000000
    MULWF AXIS     ; Shift Left 6 casas
    MOVFF PRODL, AUX   ; Mover AXIS para variavel auxiliar
     call   CHAR_DATA_MSB
     call   CHAR_DATA_LSB
    BTG    LATA, 7    ; "Piscar" LED.
    DCFSNZ  AXIS     ; Decrementar, skip se != 0.
     call   AXIS_RESET   ; Voltar a definir AXIS = 0b00000011
    BANKSEL ADPCH
    BTFSC 0X00,1
    BSF    ADPCH,1
    BTFSC 0X00,0
    BSF    ADPCH,0
    BTFSS 0X00,1
    BCF    ADPCH,1
    BTFSS 0X00,0
    BCF    ADPCH,0
    BANKSEL PIR1
    BCF    PIR1,0     ; Limpar flag do ADC
    ; ADC em standby
    RETFIE
```

```
BUTTON_0:      ; Handler da Interrupt do botao
    BANKSEL LATA
    BSF    LATA     ,6
    BTG    T0CON0   ,7   ; Toggle TIMER0 (Ligar/Desligar comunicacao)
    MOVLW 0X20     ; AXIS: 00, MSB: 1, DATA: 00000 - (Ligar + Calibrar) ou Desligar
    MOVWF CHAR
     call   SENDCHAR
    MOVLW 0X01     ; AXIS: 00, MSB: 0, DATA: 00001 - (Ligar + Calibrar) ou Desligar
    MOVWF CHAR
     call   SENDCHAR
    BANKSEL LATA
    BTG    LATA   ,5   ; "Piscar" LED
    BCF    LATA     ,6
    BCF    LATA     ,7
    BANKSEL PIR0
    BCF    PIR0, 0    ; Limpar INT0 interrupt test bit
    RETFIE
```

# Anexo C - Código das Funções

```
SENDCHAR:
    BANKSEL PIR3
    BTFSS PIR3,    4 ; Ver se TX Buffer esta disponivel (TXIF)
    bra    SENDCHAR  ; Se n o , branch para tentar de novo
    BANKSEL TX1REG
    MOVFF CHAR,    TX1REG  ; Colocar CHAR no USART TX register para envio
    BANKSEL LATA
    return
```

```
CHAR_DATA_MSB:
    BANKSEL ADRES
    MOVFF ADRESH, CHAR_MSB
    MOVLW 0b00100000
    MULWF CHAR_MSB
    MOVFF PRODH, CHAR_MSB ; 5 MSB em [4:0]
    BSF    CHAR_MSB, 5 ; Set bit "MSB"
    MOVFF CHAR_MSB, CHAR
    MOVF  AUX , 0    ; Colocar AXIS no WREG
    ADDWF CHAR, 1    ; Somar os bits de AXIS a CHAR, guardar em CHAR
    call  SENDCHAR   ; Enviar CHAR
    return
```

```
CHAR_DATA_LSB:
    BANKSEL ADRES
    MOVFF ADRESH, CHAR_MSB
    BANKSEL ADRES
    MOVFF ADRESL, CHAR_LSB
    MOVLW 0b00000100
    MULWF CHAR_MSB
    MOVFF PRODL, CHAR_MSB ; Guardar temporariamente 3 LSB de ADRESH em [4:2]
    MOVLW 0b00000100
    MULWF CHAR_LSB
    MOVFF PRODH, CHAR_LSB ; 2 LSB de ADRES em [1:0]
    MOVF  CHAR_MSB, 0 ; Colocar no WREG
    ADDWF CHAR_LSB, 1 ; CHAR_MSB + CHAR_LSB, guardado em CHAR_LSB - 5 LSB em [4:0]
    BCF    CHAR_LSB, 5 ; Clear bit "MSB"
    BCF    CHAR_LSB, 6 ; Limpar bit 6
    BCF    CHAR_LSB, 7 ; Limpar bit 7
    MOVFF CHAR_LSB, CHAR
    MOVF  AUX , 0    ; Colocar AXIS no WREG
    ADDWF CHAR, 1    ; Somar os bits de AXIS a CHAR, guardar em CHAR
    call  SENDCHAR   ; Enviar CHAR
    return
```

```
AXIS_RESET:
    MOVLW 0b00000011
    MOVWF AXIS
    return
```