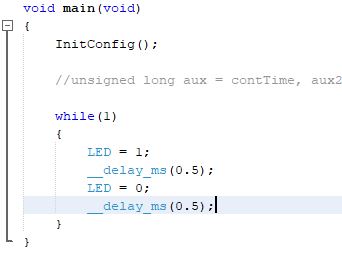
Henrique Miranda Gusmão

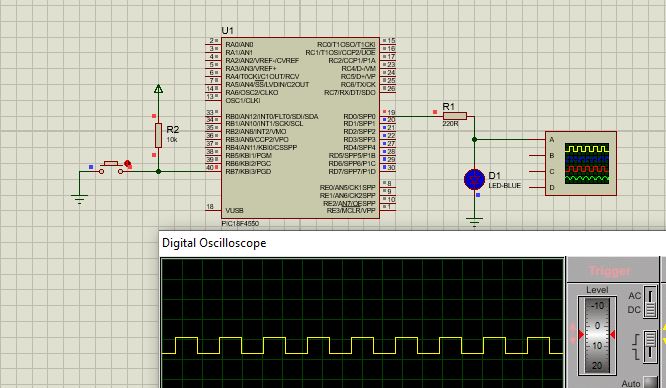
**Atividade Calouros (Respostas):**

**Questão 1:**

Antes de elaborar o código para a realização desta questão foi necessário efetuar o cálculo para descobrir o período de onda quadrática formada a partir da frequência de 1kHz. Assim, utilizando a fórmula de frequência, na qual , substituindo os valores e isolando o t, temos , que resulta no tempo de onda de 1ms (0,001s). A partir disso foi elaborado o código abaixo para o microcontrolador PIC 18F4550, no MPLabX IDE v5.35:

****

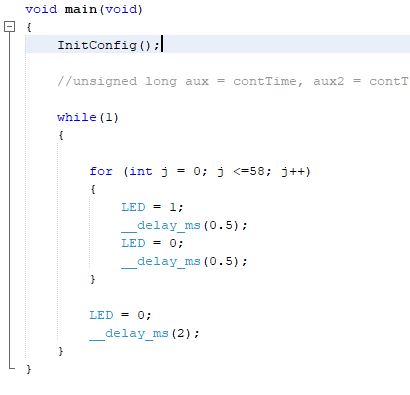
Para simular, o LED irá piscar na frequência desejada. Como 1ms é o tempo de onda completo, foi preciso deixar o LED ligado por 0,5ms e depois desligado pelo mesmo tempo, completando o período de onda. Segue a simulação realizada no software Proteus com o auxílio do osciloscópio do próprio aplicativo:



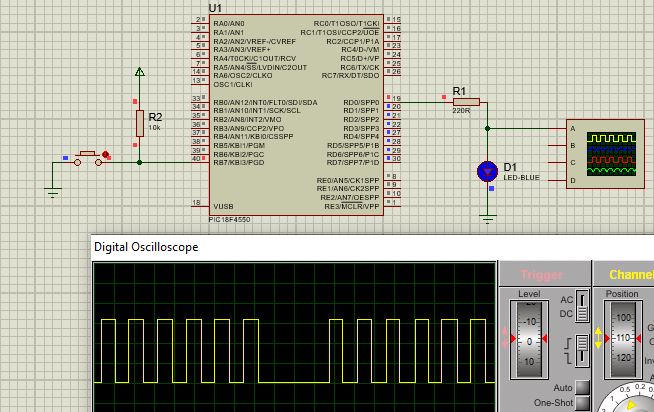
Supondo que esta frequência fosse a mesma do sinal de rotação do veículo em questão, o motor estaria numa rotação de 1000 rpm.

**Questão 2:**

Para a elaboração deste exercício foi apenas necessário incrementar na lógica da programação os 60 dentes da roda fônica do carro, levando em conta que 2 desses dentes são retirados para serem lidos como sinais de falha. Dessa forma, a lógica criada foi a seguinte:



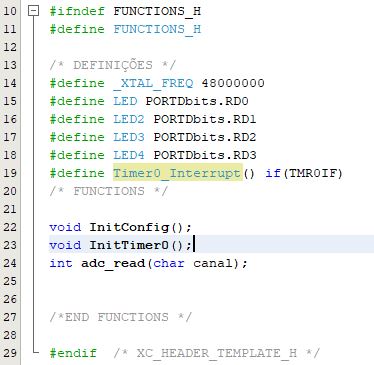
Seguindo os mesmos princípios da lógica passada, simplesmente foi acrescentado o comando “for”, assim simulando os 58 dentes existentes na roda fônica, e quando o laço do “for” for concluído, ele fará o LED ficar desligado pelo tempo de 2 períodos, retratando os dois dentes de falha. Segue a simulação criada:

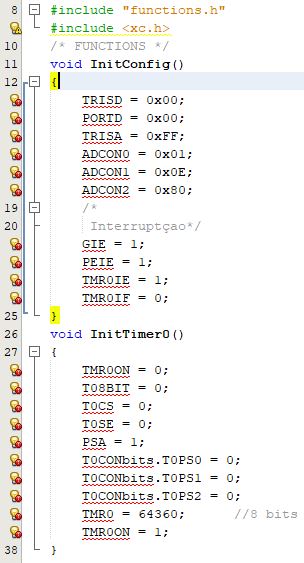


A diferença existente entre as duas primeiras questões se encontra justamente nos dois dentes de falha que foi incrementado na questão 2, podendo simular o funcionamento da roda fônica do carro. Tal falha tem como função primordial localizar em que ponto a roda fônica está e identificar o final do giro, reconhecendo uma rotação.

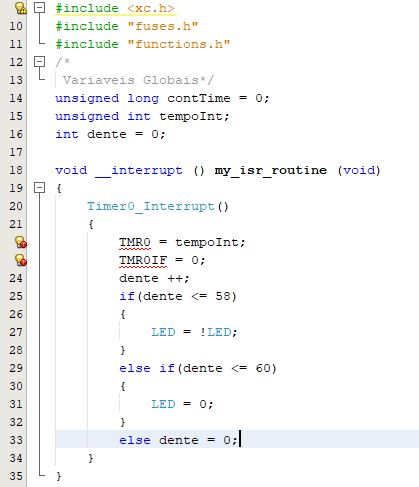
**Questão 3:**

A partir de agora a lógica precisou de muitas alterações. Devido a falta de precisão existente no compilador XC8 para utilizar tempos de “delay” muito altos, foi necessário aplicar uma lógica usando os “timers” do microcontrolador. O timer é um recurso via hardware existente que faz a contagem de ciclos de máquina, podendo utilizar uma contagem de tempo mais precisa e com períodos maiores. Assim, foi necessário iniciar o Timer0 na função de configurações iniciais e, posteriormente, configurar o próprio Timer0. A página de funções ficou do seguinte modo, juntamente com a página de declarações de funções e definições.

Definições e Declarações

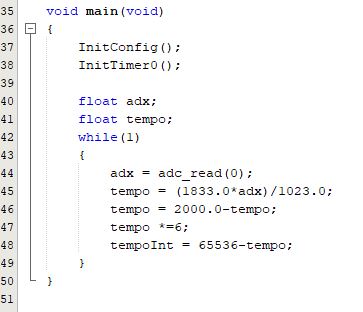
Página de Funções

A lógica criada ficou do seguinte modo:

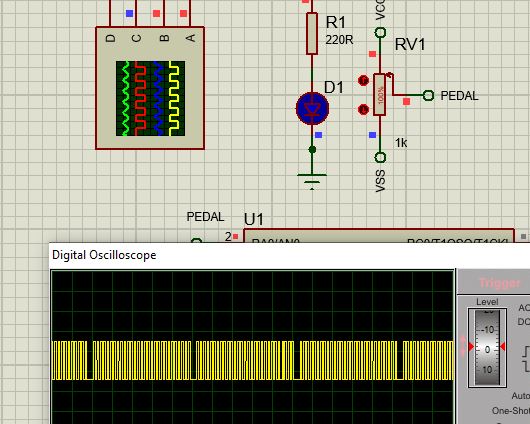


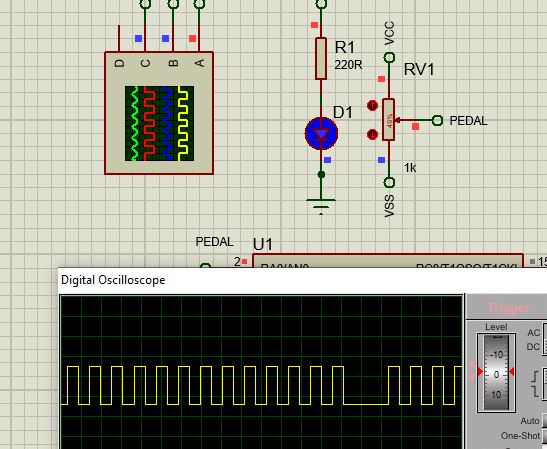
Inicialmente, na página principal, temos as declarações das variáveis globais utilizadas. Depois temos a função de interrupção do Timer. Ao ocorrer o estouro do Timer0, ele primeiramente irá configurar o novo tempo em que o Timer irá contar, isso através do cálculo realizado na conversão A/D na função principal. A lógica que segue é basicamente a que foi feita na questão anterior, com poucas adaptações para ser utilizada na função de interrupção.

Na função principal temos as chamadas das funções de configurações iniciais e de inicialização do Timer0 e as declarações das variáveis utilizadas nos cálculos. Agora no loop (“while 1”), apenas será feita a conta de conversão A/D, aplicando uma simples regra de três para converter o valor lido num valor que será o tempo de estouro do Timer, variando entre 166us e 2ms (respectivamente 6000rpm e 500rpm).



Segue a imagem de teste do programa:



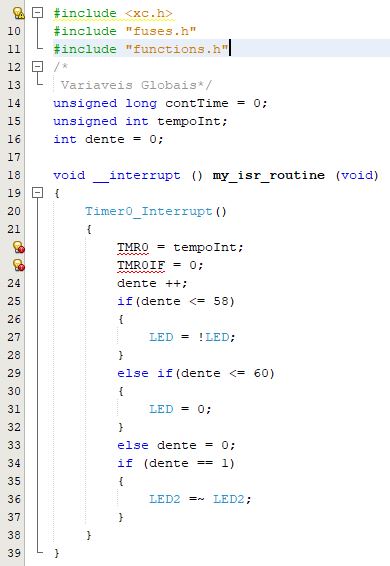


Diferença no comprimento de onda

Do mesmo modo que é funcionamento do potenciômetro nesta lógica, assim o pedal do carro também funciona: quanto mais é pressionado o pedal, maior será a rotação do carro e, consequentemente, a ação contrária irá reduzir a rotação.

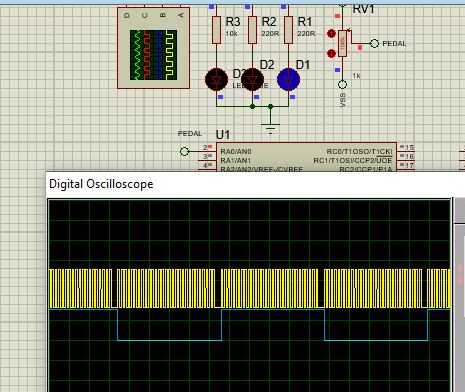
**Questão 4:**

Neste processo o código não sofreu alterações significativas, apenas foi necessário acrescentar ao hardware um ou LED para simular o funcionamento do sinal de fase. Segue a lógica:



Simplesmente foi acrescentado este comando de comparação para que sempre que ele identificar o primeiro “dente”, ele mude o estado lógico do LED2 (Sinal de Fase).

Segue a imagem de simulação:



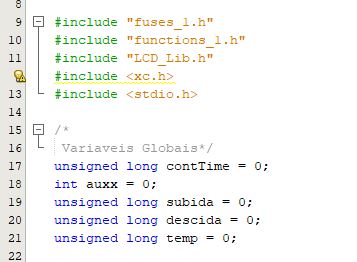
Sinal de Rotação

Sinal de Fase

O sensor de fase tem como função principal informar em qual posição está o primeiro cilindro, para assim determinar a sequência da execução dos tempos em todos os cilindros. Além disso, na partida do motor, o sensor de fase faz rapidamente o trabalho de sincronia entre o virabrequim e o comando de válvulas.

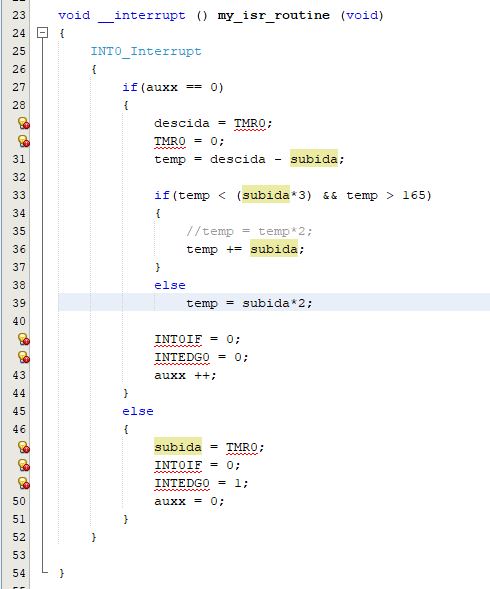
**Questão 5:**

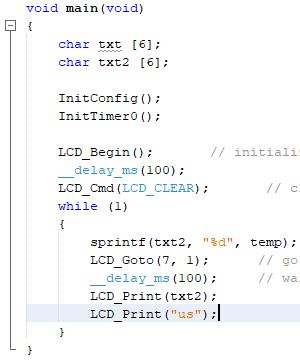
A partir de agora foi criado um segundo projeto no MPLabX para fazer a análise do sinal e exibir no Display LCD, assim como a atividade propõem. O segundo projeto ficou do seguinte modo (As páginas de funções e definições do programa e de hardware e do LCD serão colocadas no final):



Declarações das variáveis globais.

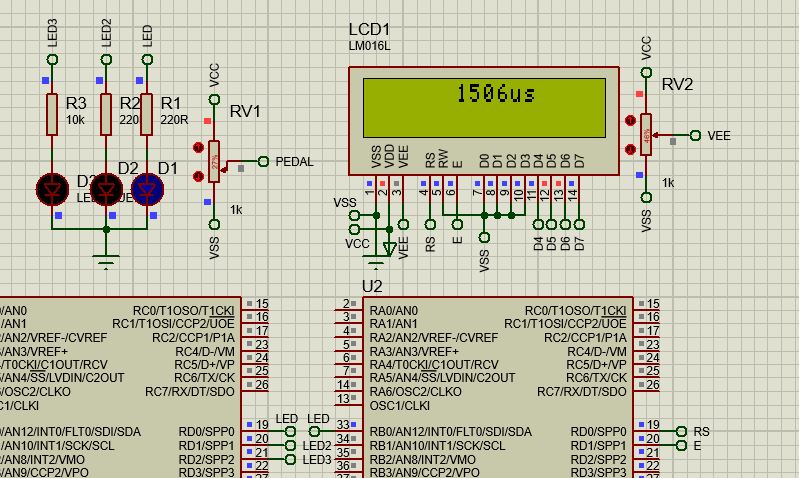
A função de interrupção do Timer0 neste projeto está configurado para o modo borda de descida/subida. Assim, na interrupção, ele primeiramente irá gravar na variável “descida” o valor total do tempo de onda, configurar o timer0 para 0 e exibir a diferença entre o valor de “descida” e “subida” (tempo de meia-onda). Depois será feita a comparação para analisar se o tempo correspondente não faz parte da falha, o que deixaria o período do pulso errado em relação aos outros, e guardar na variável “temp”, além de configurar o Timer0 para modo borda de descida. No último “else” analisado na imagem guarda na variável “subida” o tempo de meia-onda, além de configurar o Timer0 para detecção de borda de subida.





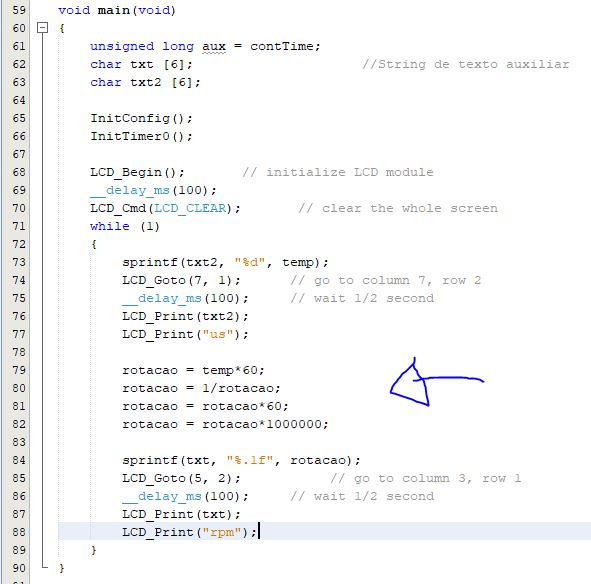
Na função principal ele fazer a chamada das funções de configurações iniciais e de inicialização do Timer0. Depois será inicializado o LCD e limpo. Na rotina de loop será transformado o valor de “temp”, que é o período do pulso, em uma variável de texto e ficará escrevendo no LCD esse valor em us (microssegundos).

Segue a imagem de simulação:



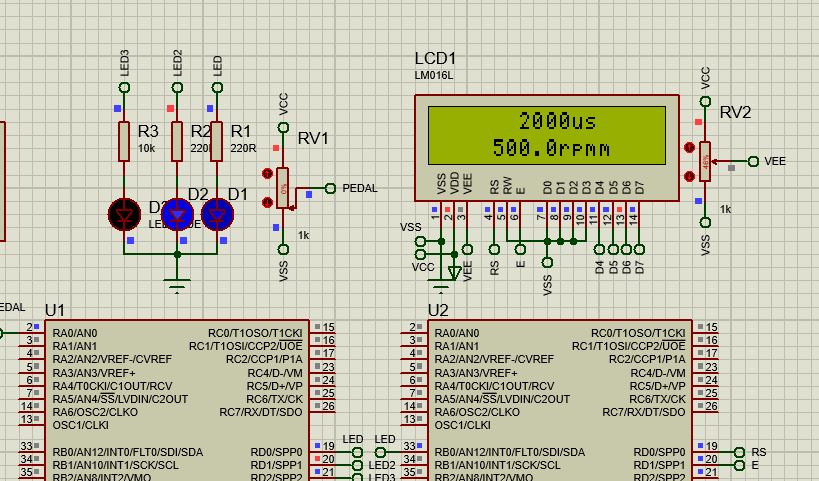
**Questão 6:**

Para realizar cálculo para encontrar o RPM é necessário saber o período do pulso, o qual é lido a partir da questão anterior. Desse modo, o cálculo fica: . Segue a programação:



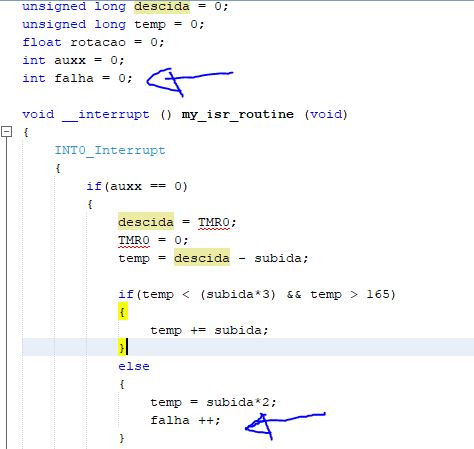
A única diferença entre as questões 5 e 6 é apenas a incrementação da conta e da escrita do RPM no LCD. É necessário dividir a variável “rotacao”, que guarda o valor de RPM, por 1000000, pois o período de pulso está em microssegundos.

Segue a imagem de simulação:



**Questão 7:**

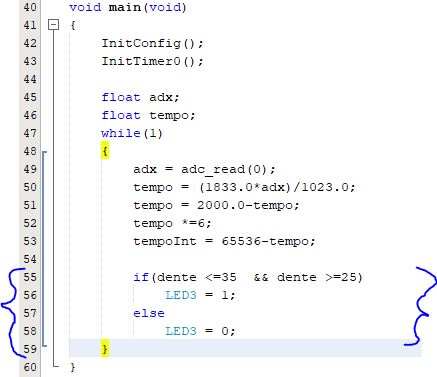
Na realização desta questão o programa não teve alterações significativas. Abaixo está a lógica:



Nesta lógica foi necessário apenas criar uma variável do tipo inteira e incrementar 1 nela sempre ocorrer a falha. Esse ponto de falha está localizado no “else” apontado pela seta.

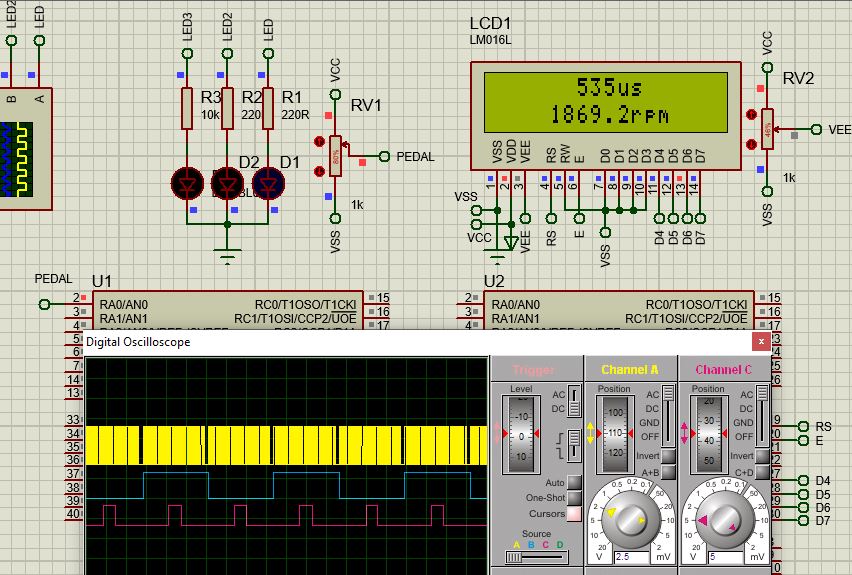
**Questão 8:**

Nesta última questão foi preciso retornar ao primeiro projeto criado. Segue a lógica da programação na função principal na imagem abaixo:



A única alteração realizada no “void main” foi adicionar um comando de comparação do dente em que se encontra a roda fônica. Se localizado aproximadamente da metade do curso de dentes, o pistão entra em PMS. Assim, sempre que estiver nesse meio o “LED3” é ativado, indicando o PMS do primeiro pistão.

Ao realizar pesquisas sobre o assunto pude chegar à conclusão de que dependendo do sinal de fase e do sinal de rotação, o PMS será encontrado, seja ele PMS no momento de injeção, ou seja ele no momento de compressão/explosão. Segue a imagem de teste:



**Dificuldades e conclusão:**

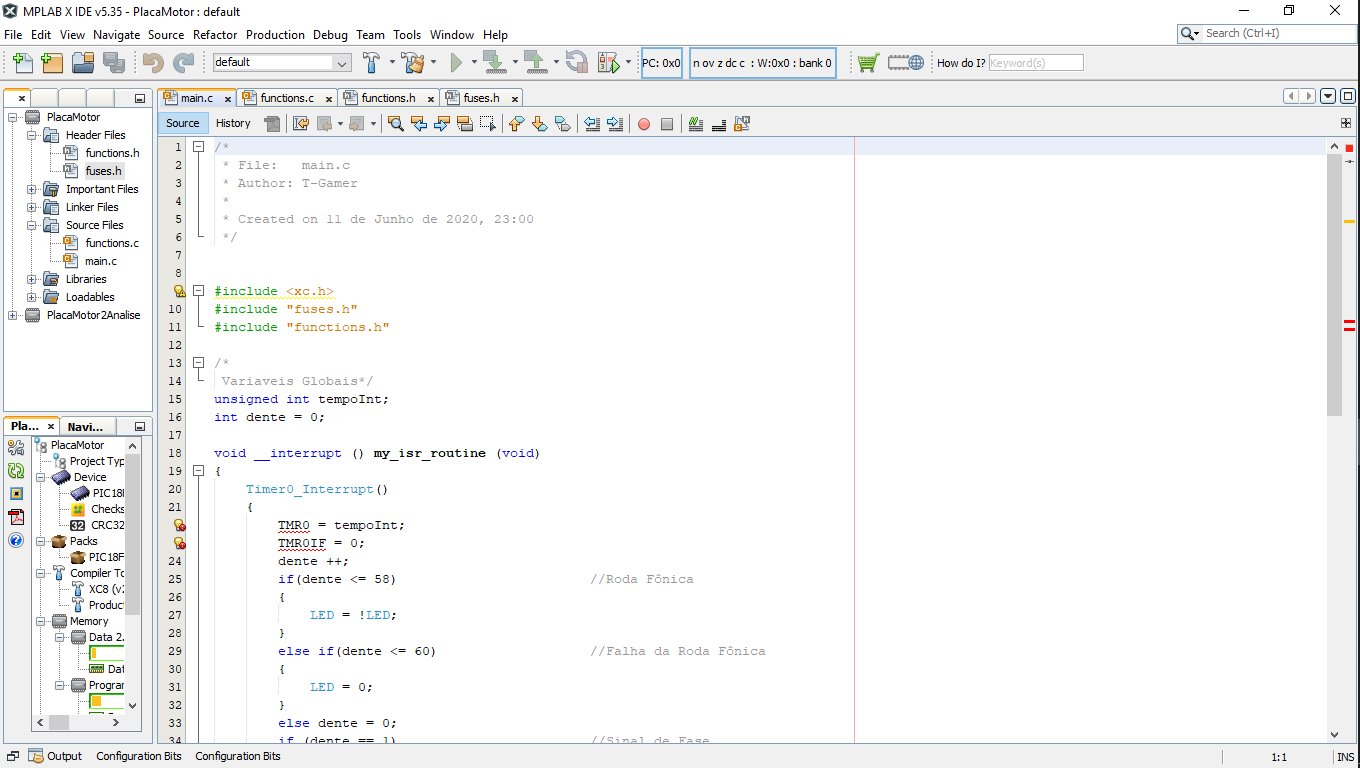
Durante o período de realização dos exercícios, diversas dificuldades surgiram, principalmente em relação ao hardware, pois possuía um conhecimento limitado sobre a estruturação e o uso do PIC 18F455, além de seus vastos detalhes. Para entender e aplicar na minha lógica esses detalhes busquei por fontes confiáveis na internet e, principalmente, a ajuda de meu irmão William Miranda Gusmão, que já possuía um conhecimento mais amplo sobre o hardware do microcontrolador utilizado.

Com relação a lógica desenvolvida, a mesma teve seus maiores contratempos em relação a comunicação dos recursos do hardware (Timers) com o firmware. Nesse quesito ainda pude obter a ajuda de meu irmão e dos sites. Porém, devido a experiencia que obtive no curso de Mecatrônica no SENAI, o resto da programação se tornou mais simples, apenas deixando algumas dúvidas com relação a estruturação do software MPLabX e de seu compilador (XC8) utilizado para programar, dos quais eu nunca havia trabalhado. Por isso foi necessário buscar em fontes as informações necessárias.

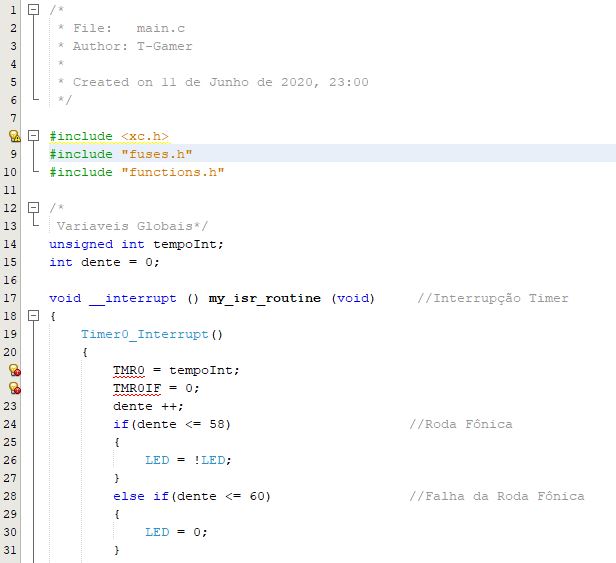
Em relação ao gerenciamento do motor, foi necessário pesquisas para compreender como realmente funcionavam as ações do mesmo, para assim gerar de modo prático as simulações necessárias, visto que ingressei recentemente no curso de eletrônica automotiva da Fatec Sorocaba e não cheguei nas matérias requisitadas. Porém, com a realização desta atividade fui incentivado a pesquisar e entender a atuação de um eletrônico automotivo e tudo que o engloba, além de ampliar meus conhecimentos de programação em linguagem C e microcontroladores.

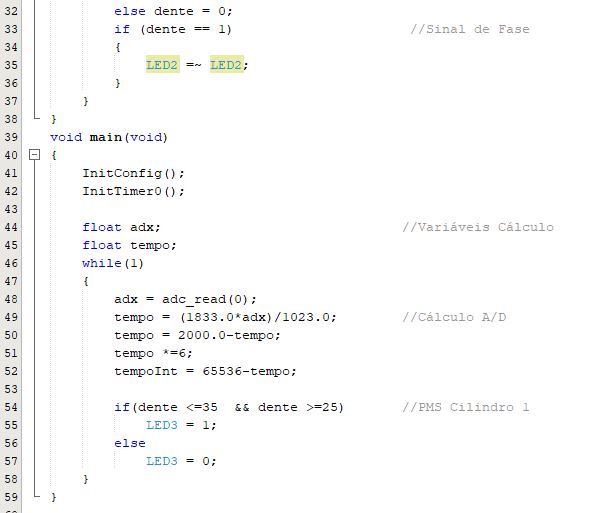
**Programações na íntegra**

**Lógica Final Projeto 1 (Sinais Gerados):**

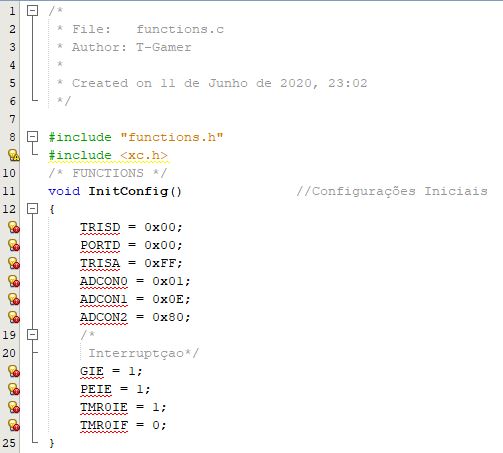
****

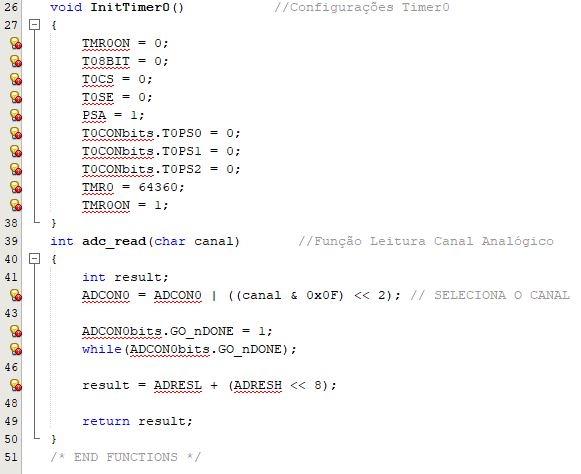
(main.c)



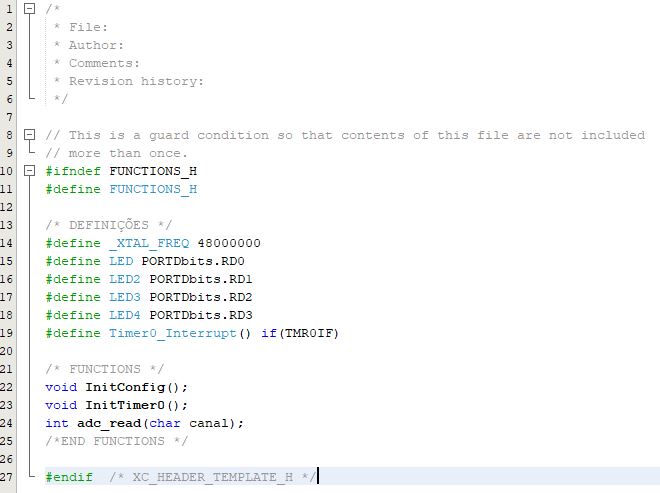


(Functions.c)

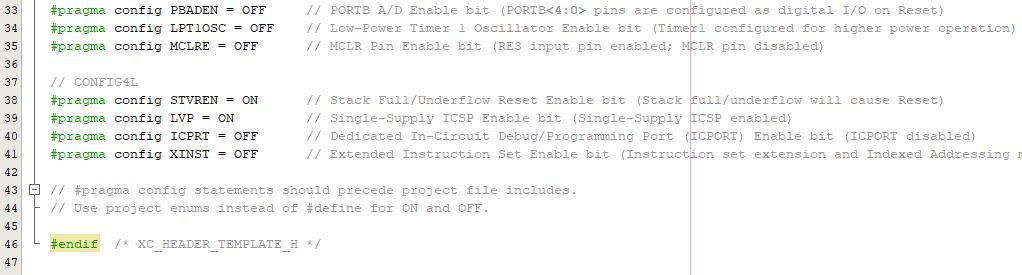
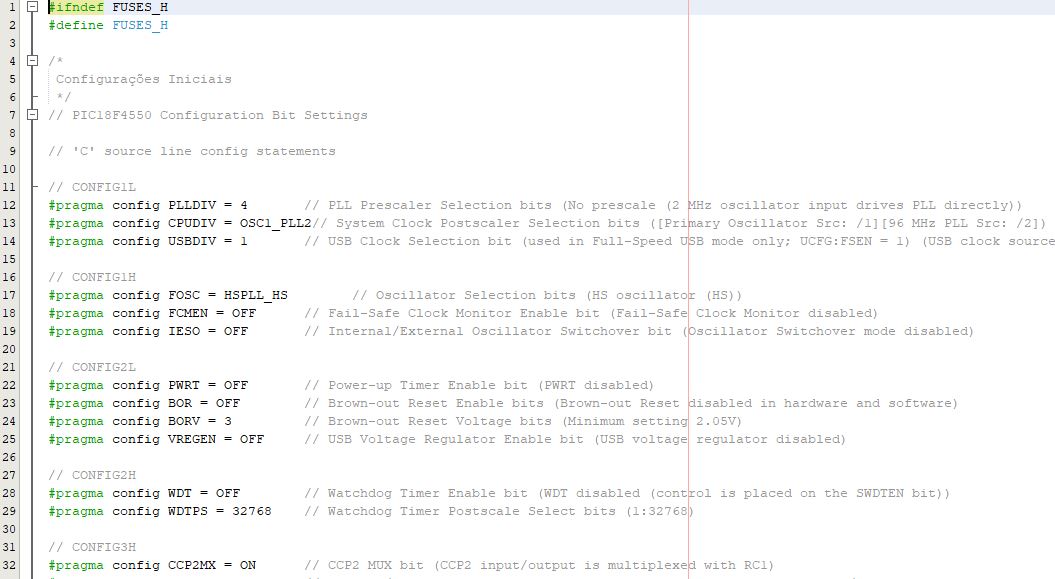




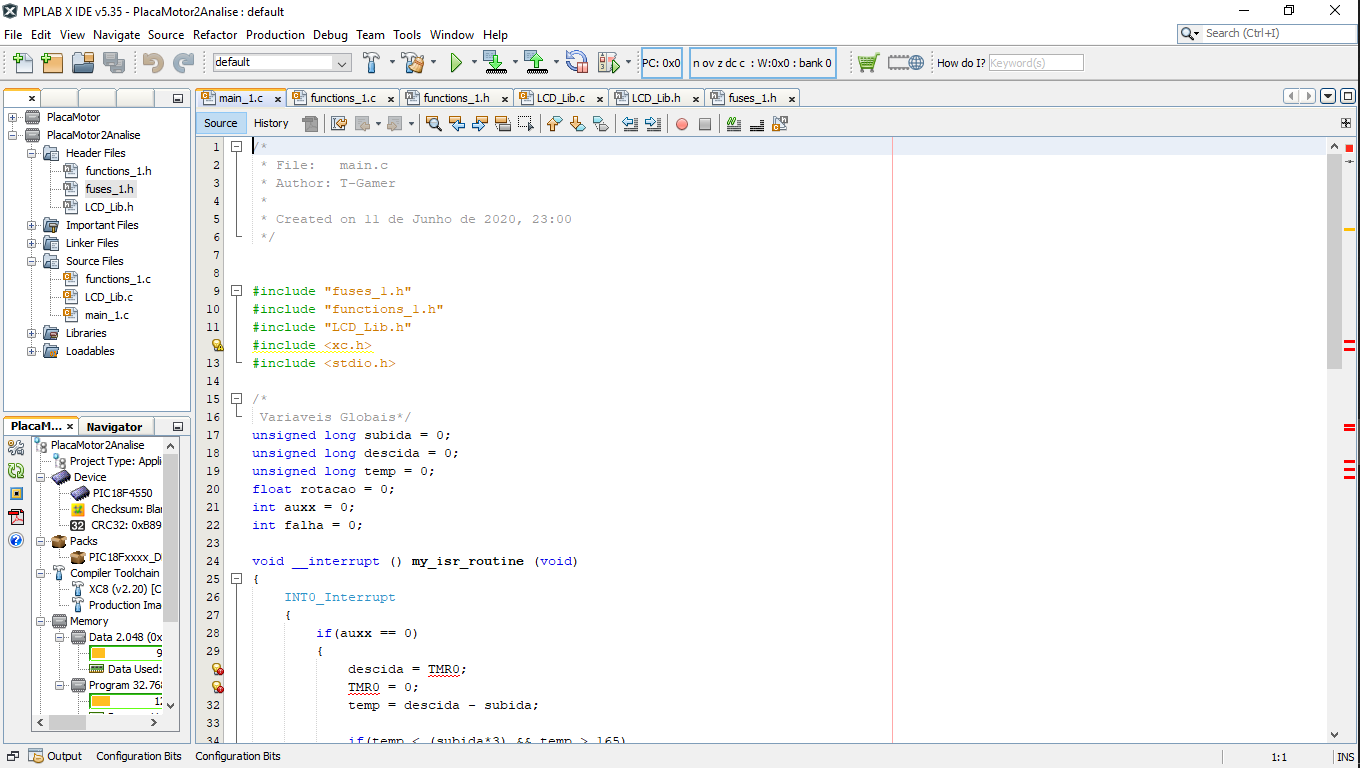
(Function.h)



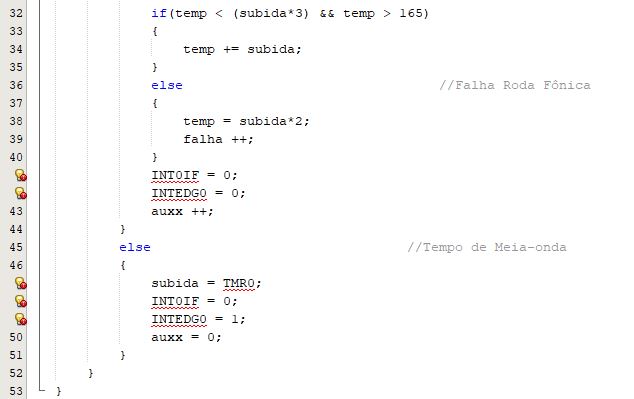
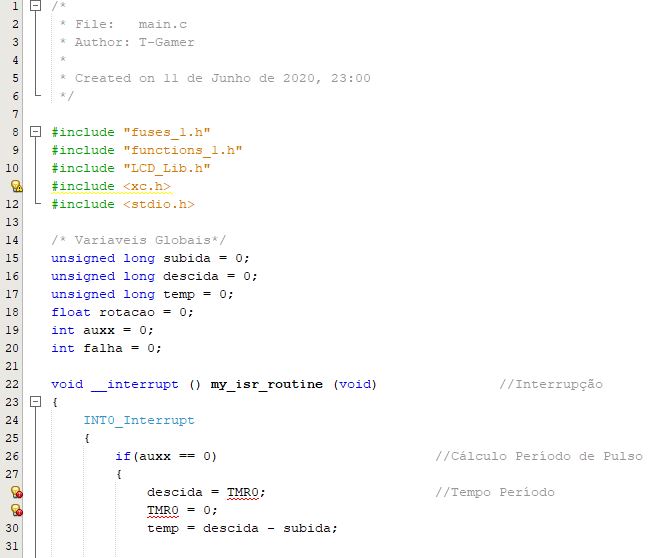
(Fuses.h)

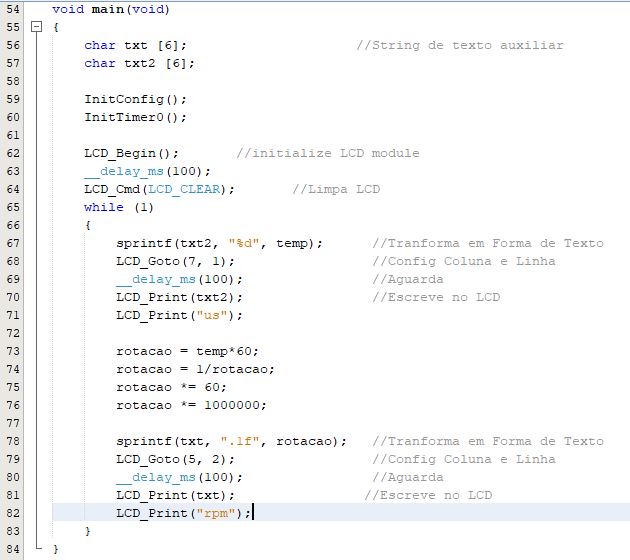


**Lógica Final Projeto 2 (Análise de Sinais):**

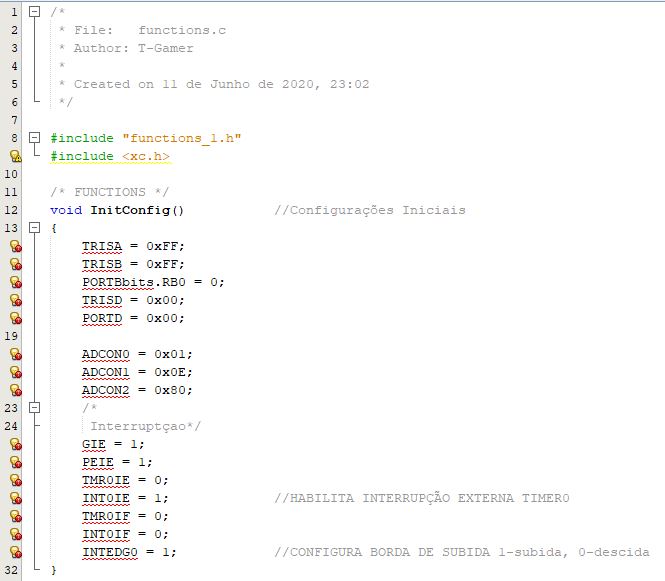
****

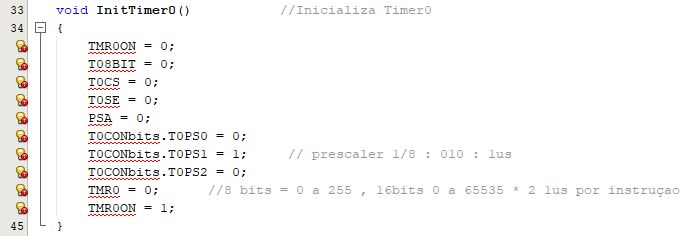
(Main.c)



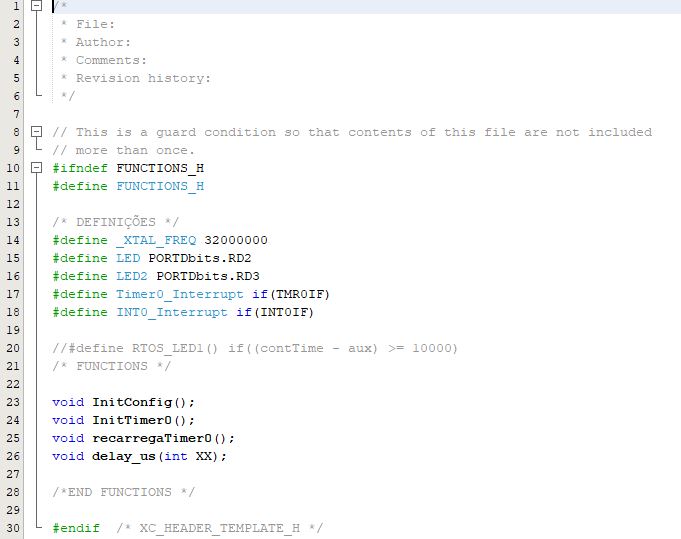


(functions.c)

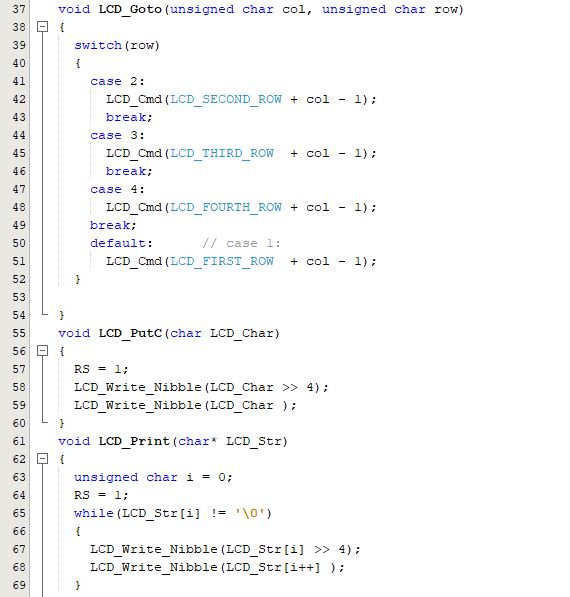
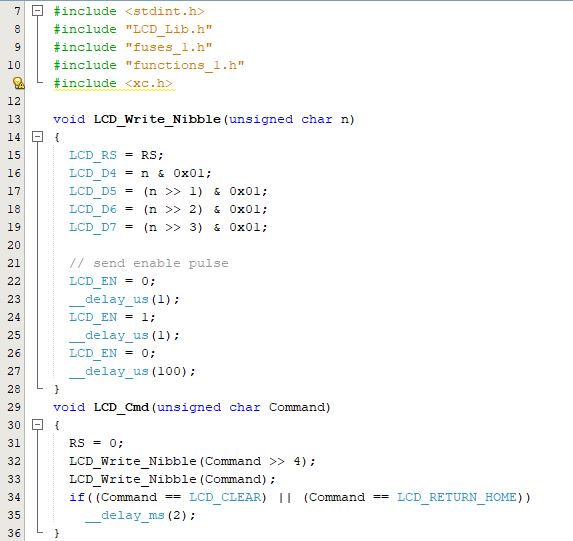


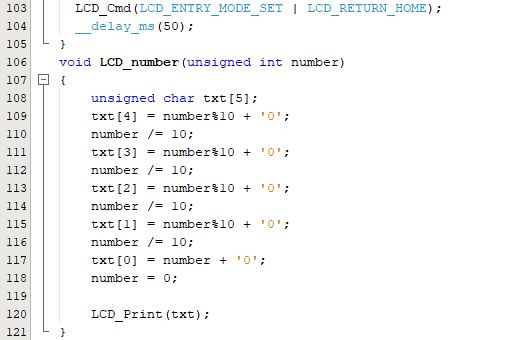
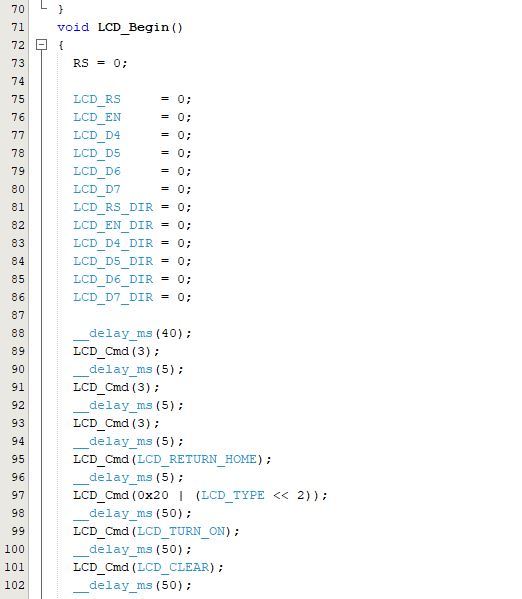


(functions.h)

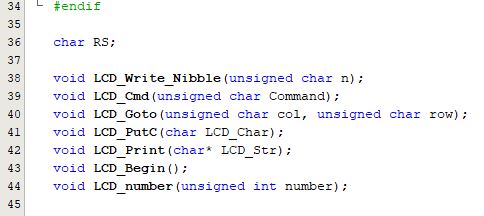


(LCD.c)

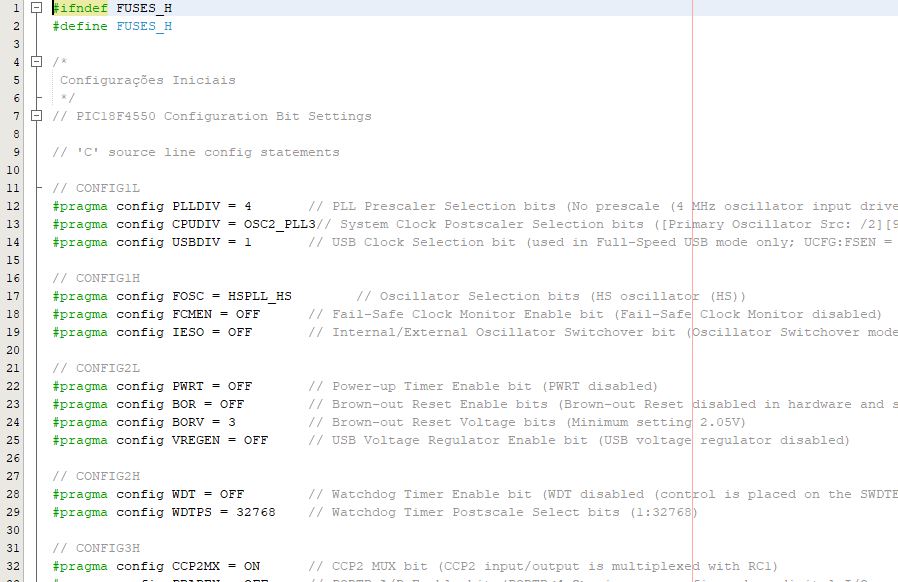


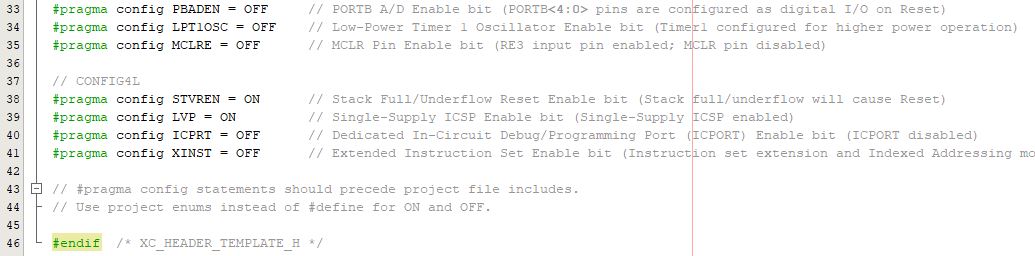


(LCD.h)



(fuses.h)





**Referências:**

Datasheet PIC 18F4550

https://www.canaldapeca.com.br/blog/sabe-encontrar-o-pms-do-motor/

https://www.minutoseguros.com.br/blog/sensor-de-fase-como-funciona/#:~:text=Fun%C3%A7%C3%A3o%20do%20sensor%20de%20fase,sinal%20do%20sensor%20de%20rota%C3%A7%C3%A3o.

https://blog.abmpecas.com/ponto-morto-superior/

http://www.cplusplus.com/reference/cstdio/sprintf/

http://linguagemc.com.br/variaveis-em-linguagem-c/

https://www.clubedohardware.com.br/topic/609686-como-fazer-para-um-float-ficar-s%C3%B3-com-2-casas-decimais/

https://simple-circuit.com/pic-microcontroller-mplab-xc8-lcd/

https://www.microchip.com/forums/

https://www.youtube.com/channel/UCazAvTtoRlOrFDWDJDB2DKQ - WRKits, Eng. Wagner Rambo.

https://www.youtube.com/watch?v=4WaB\_Lj8FnY

https://www.youtube.com/watch?v=RgTMbQ6lssI

https://www.youtube.com/watch?v=vyDyAYZroXI

https://www.youtube.com/watch?v=lqCdQqniD8Y