CURSO:	ENGENHARIA ELETRÔNICA
DISCIPLINA:	Eletrônica Embarcada TURMA: A
SEMESTRE:	2°/2019
PROFESSORA:	Gilmar Silva Beserra
ALUNO:	Lucas Gonçalves Campos e Wemerson Fontenele Sousa
MATRÌCULA:	170016757 e 170024130

Ponto de Controle 3

1. Resumo

No atual ponto de controle, especificaremos os testes realizados até o momento, no que diz respeito à implementação deste projeto na MSP430, através do ambiente de desenvolvimento integrado Code Composer Studio. Explicando as etapas para adaptação do programa desenvolvido para o ponto de controle 1, para o código em C, desenvolvido até o momento. Assim como as dificuldades encontradas. Explicando como o grupo deseja contornar estas dificuldades e o que ainda precisa ser feito para que se conclua o projeto.

2. Desenvolvimento

2.1 Adaptação Energia - Code Composer Studio

Foi adaptado para linguagem C, o código desenvolvido para a plataforma energia IDE. Tendo novamente como base, a máquina de estados desenvolvida inicialmente para o projeto (figura 1).

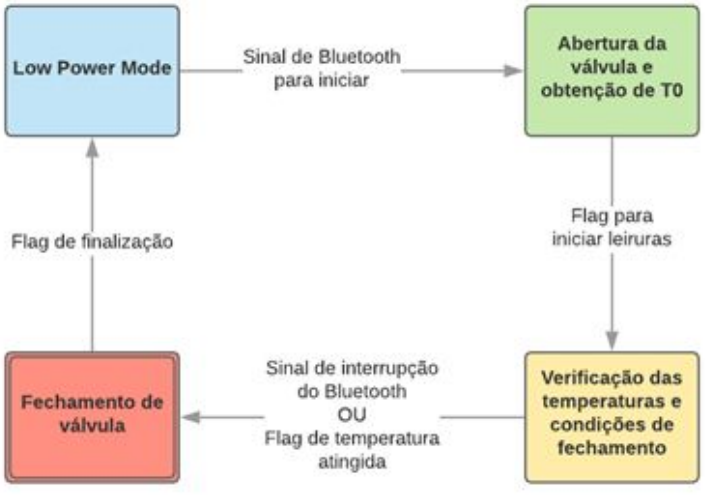

Figura 1: máquina de estados do projeto

2.2. Testes realizados

Até o exato momento, foi desenvolvido apenas o código do projeto para linguagem C. Tornando possível os testes de todas as etapas do projeto, porém, sem utilizar a linguagem assembly, já que serão realizadas posteriormente duas sub rotinas nesta linguagem: uma responsável pela conversão A/D do sensor de temperatura e outra pela realização do atraso, função que será melhor explicada posteriormente, mas que foi bastante usada.

Primeiramente, foram desenvolvidas três funções convenientes para o controle do motor de passo: uma função para girá-lo em sentido horário (abertura), uma função para girá-lo em sentido anti-horário (fechamento) e uma função de atraso de X ms, sendo X determinado pelo usuário, utilizando o timer da placa, sendo esta função necessária para o controle de velocidade do motor.

Após isso, foi desenvolvido uma função (leitura) para medir a temperatura medida pelo sensor LM35, vinte vezes, e após isso, tirar a média dessas medições para se obter maior precisão dessas medidas, sendo essa média necessária para contornar o erro encontrado no ponto de controle 2, com valores randômicos atrapalhando o valor final da leitura. Sendo esta função configurada de maneira a ser realizada uma medida inicial que é guardada para comparações e após isso, entrando em um loop de medições, até que seja detectada uma interrupção que acionaria o fechamento da válvula, ou que a temperatura varie um determinado delta T, que também acionaria o fechamento da válvula. Entre as leituras, também foi usada a função de atraso para diminuir erros por alta velocidade de processamento.

Após isso, foram desenvolvidas as bibliotecas necessárias para o uso do módulo bluetooth (HC-06), sendo elas:

- A função Init_Uart, responsável por iniciar as flags necessárias para o uso das portas RX e TX para tal meio de comunicação e escolher o baud rate para o módulo bluetooth HC-06, no caso, 9600;
- A função Receive_Data, responsável por realizar a leitura de cada caractere dos dados enviados à placa.
- A função Read_string, responsável por traduzir os dados obtidos pela função anterior em uma String;
- A função Send_Data responsável pelo envio de dados;
- A função Send_String, usada para enviar mensagens maiores para o aplicativo;
- A função Send_int responsável pelo envio de dados numéricos do tipo Int para o aplicativo;

Após isso, foi realizada a união de todas estas funções. Para isso, foi desenvolvido o código de maneira a placa permanecer em LPM1, até receber uma interrupção. A única rotina de interrupção vem do bit RX, recebido quando enviada uma mensagem pelo aplicativo de celular, essa rotina identifica os dados recebidos e através de uma nova função (cmp_str), que identifica se é uma mensagem para a válvula ser aberta, fechada; após isso, essa rotina compara o comando enviado e o estado atual e identifica se este

comando deve abrir a válvula, fechá-la, ou se este é um comando inválido (seja por estar em um estado errado, ou pela mensagem ser inválida).

Os resultados obtidos durante todas as etapas destes testes foram como esperado, e o projeto está funcional.

3. Dificuldades Encontradas

Até o exato momento, a maior dificuldade é o fato de que ainda não foi desenvolvida uma função “Send_Float”, para que sejam enviados os dados de temperatura com uma maior resolução, já que estes estão sendo enviados como variáveis do tipo inteiro.

Outra dificuldade encontrada é o fato de não ter sido desenvolvido nenhuma sub rotina em assembly, como é pedido para o projeto. Para a conclusão deste, a sub rotina a ser desenvolvida será a responsável pelo atraso de 1 ms. E caso, seja possível, será desenvolvida em assembly também, a sub rotina responsável pelo conversor AD necessário para que após obtidos esses dados, seja calculada a temperatura.

4. Conclusão

Todo o código já escrito para a plataforma energia IDE foi traduzido para a linguagem C do MSP430, utilizada pelo Code Composer Studio. Assim como no ponto de controle 2, o projeto teve suas funcionalidades testadas e aprovadas. Entretanto, restam alguns pontos a serem aprimorados no código, tais como a precisão dos dados de temperatura através do envio de sinais tipo float e a otimização de uma parcela do código para o Assembly, da forma que foi requisitado pelo projeto.

Para a finalização do trabalho, resta, além das otimizações explicitadas acima, a obtenção dos materiais para a construção do protótipo, no que diz respeito a sua estrutura e a elaboração deste para a apresentação final, pontos que serão trabalhados até o próximo ponto de controle.