



Trabalho Final - Grupo 3

Bola no Tubo

Herbert Tavares de Oliveira Silva - 190014300

Júlio César S. Carvalhaes - 190090332

Pedro Sampaio Dias Rocha - 211043745

Rafael Brasileiro V. R. da Costa - 190044110

Disciplina: Eletrônica Embarcada

Curso: Engenharia Eletrônica

FCTE - Faculdade Gama

Universidade de Brasília

Fevereiro, 2025



1 Resumo

No presente documento, será apresentado o desenvolvimento do projeto "Bola no Tubo" que consiste no desenvolvimento de um firmware para manter a altura de uma bola flutuando dentro de um tubo vertical, utilizando um fluxo de ar gerado por uma ventoinha. O sistema é controlado por um microcontrolador PIC16F1827, que gerencia a medição da altura da bola através de um sensor ultrassônico HC-SR04, o controle da ventoinha via PWM, e a abertura de uma válvula motorizada controlada por um motor de passo 28BYJ-48. Além disso, o sistema permite comunicação Bluetooth, através do módulo HC-05, para monitoramento e controle remoto.

2 Metodologia

O projeto foi dividido em quatro partes assim como foi sugerido no roteiro:

- Medição da altura
- Movimento do motor de passo e detecção do fim de curso
- Controle PI ou PID
- Comunicação Serial-BT

Cada integrante do grupo ficou responsável por uma dessas partes e ao final, as mesmas foram unidas em um único programa e todos os membros do grupo, juntos, foram otimizando e desenvolvendo melhor o Firmware.

Integrante	Área responsável
Pedro Sampaio	Comunicação Serial-BT
Hebert tavares	Movimento do Motor
Júlio César	Medição da altura
Rafael Brasileiro	Controle PI ou PID

Tabela 1: Distribuição entre os integrantes

O desenvolvimento do projeto foi realizado no MPLAB X e o desenvolvimento do firmware foi registrado no GitHub e nos vídeos publicados no Youtube.

3 Desenvolvimento

Seguindo o roteiro disponibilizado pelo professor e os esclarecimentos obtidos em aula, cada etapa do projeto foi desenvolvida e testada por meio da ferramenta de simulação do MPLAB. No entanto, devido a certas limitações e dificuldades enfrentadas na simulação, nem todos os testes puderam ser plenamente aproveitados. Por esse motivo, optou-se por realizar testes em bancada, garantindo previamente que os procedimentos não causariam danos ao projeto.

4 Fluxograma de Projeto

O sistema de controle de posição da bola no tubo segue um ciclo baseado em sensoriamento (medidas), processamento, controle e comunicação. Inicialmente, os sensores coletam dados do ambiente, incluindo o tempo de eco do ultrassom e a temperatura. A partir dessas informações, a posição da bola é calculada.

Com base na altura determinada, o sistema ajusta a válvula e a ventoinha para manter a bola na posição desejada. Os dados do sistema são enviados via UART para monitoramento, enquanto comandos externos podem ser recebidos para modificar o modo de operação, permitindo controle manual ou automático. Esse processo se repete continuamente para garantir a estabilidade do sistema. O fluxograma abaixo:

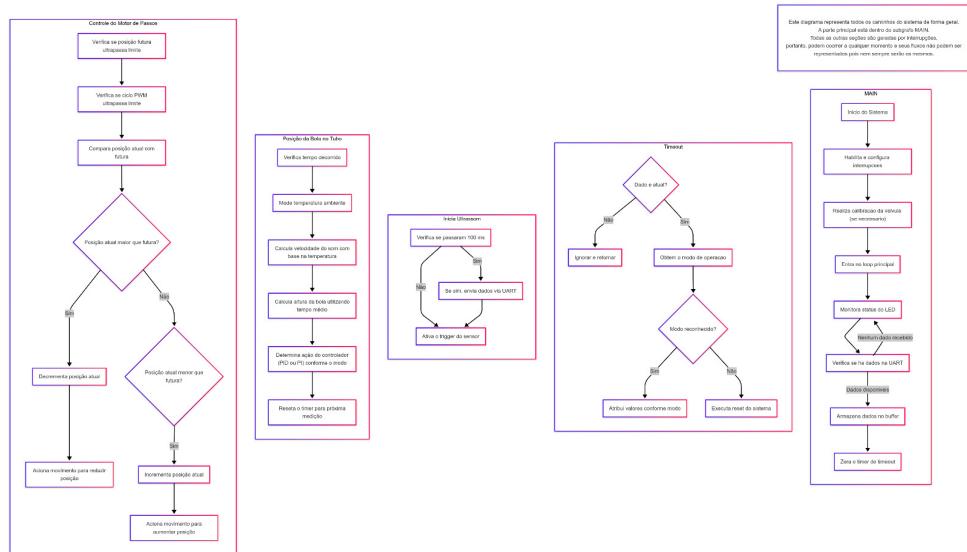


Figura 1: Fluxograma do projeto

5 Resultados

A seguir serão apresentados os resultados das simulações do firmware para cada parte dividida entre os integrantes responsável pelas mesmas. O detalhamento de cada função pode ser encontrada no Doxygen.

5.1 Pedro Sampaio

Funções desenvolvidas: Dados_recebidos, ProcessaDados, EnviaTX, selecao_do_modo.

5.1.1 Resultados

Iniciamos a simulação com o estímulo externo com o arquivo que pode ser visto na figura 2, e, na figura 3, os dados sendo recebidos na variável "Dado" que representa o RxBuffer que é um array unidimensional com 7 posições e a função de seleção de modo já funcionando selecionando o modo 01 como foi pedido no estímulo.

```
Video.txt          Estimulos.txt
Arquivo  Editar  Exibir

Wait 130 ms
0x01
0x03
0x20
0x03
0x20
0x01
0x03
0x20
0x03
0x20
0x00
0x03
0x20
0x03
0x20
0x01
0x03
0x20
0x03
0x20
```

Figura 2: Estimulo de dados a serem recebidos

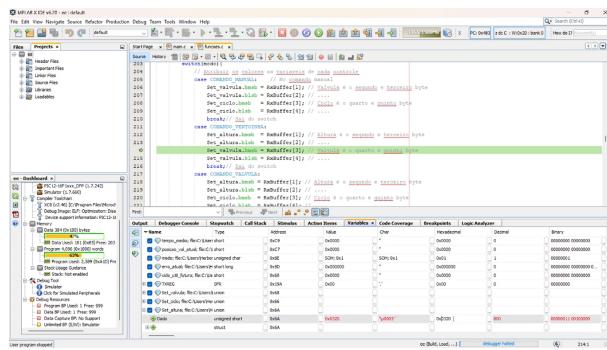


Figura 3: Recebimento dos dados e seleção do modo

A seguir temos o segundo funcionamento do estímulo onde é requisitado que entre no modo 02 (modo válvula) e a seleção ocorre assim que chega na função.

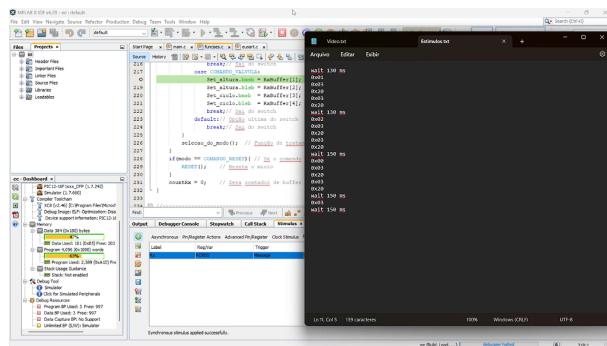


Figura 4: Segunda recepção de dados do estímulo

Por fim vemos o envio final de dados realizado pela função EnviaTx, onde a cada ação da função, a função EUSART_Write recebe o bit a ser escrito que no momento da figura 5 é o envio do modo 02.

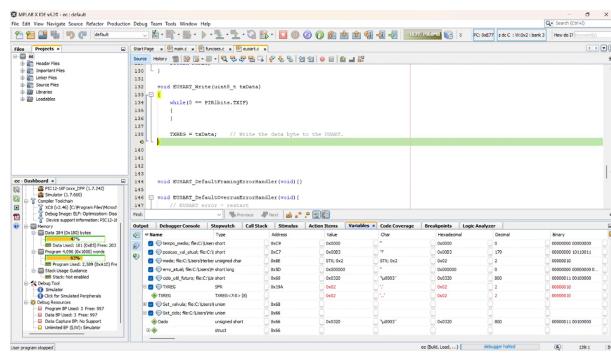


Figura 5: Envio de dados

E dessa forma é realizado o recebimento e envio de dados pelo pino Tx e Rx respectivamente.

5.2 Herbert Tavares

Funções desenvolvidas: DRMotordePassos, funcao_diminui_num_passos, funcao_aumenta_num_passos, Passo1, Passo2, Passo3, Passo4.

5.2.1 Resultados

A simulação do motor de passo começa com a interrupção do Timer 6 que ocorre a cada 3 ms, como pode ser visto na aba stopwatch na figura 6, iniciando a função DRMotordePasso.

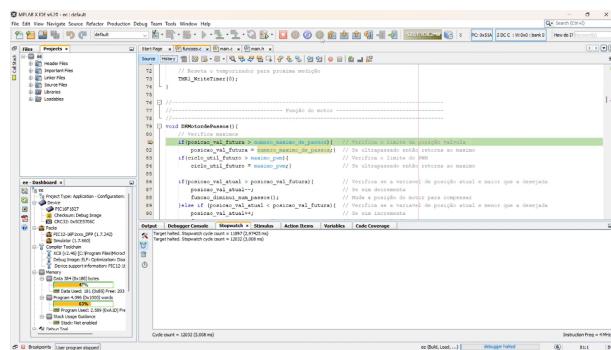


Figura 6: Interrupção do Timer 6

A seguir é simulada a limitação de valores do Duty Cicle do PWM e de posição da valvula. Na imagem 7 é forçado um valor de 1026 e 2066 para a valvula e o PWM respectivamente e na imagem 8 é visto que o programa corrige para os valores máximos de 460 e 1023.

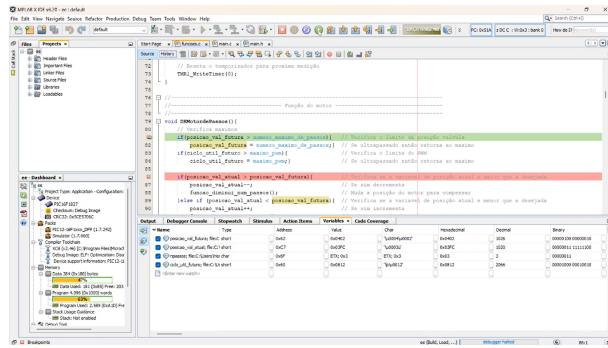


Figura 7: Variáveis recebendo valor maior que o máximo

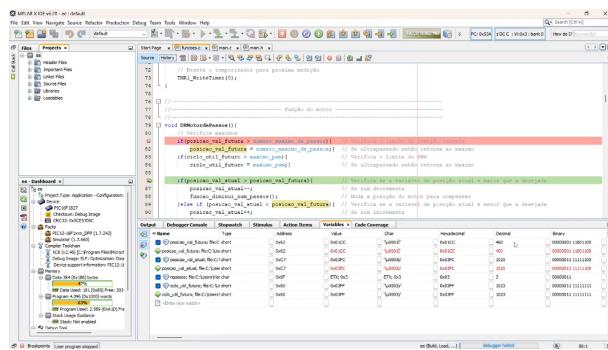


Figura 8: Programa adequando a entrada para o valor máximo possível

Agora colocando um valor de posição futura igual a 10 na válvula e atual igual a zero, temos funcionamento da função de aumentar o número de passos para alcançar esse valor desejado de 10.

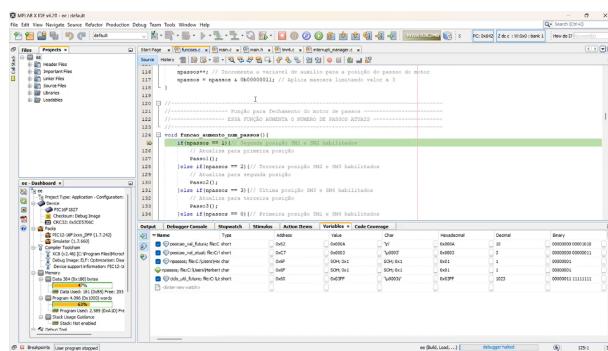


Figura 9: Função de aumento de passo

Por fim, caso coloquemos que a posição futura é 0 e a atual é um valor maior, como 2, a função que será chamada é a de diminuir passos, até que o valor da posição futura seja igual à posição atual.

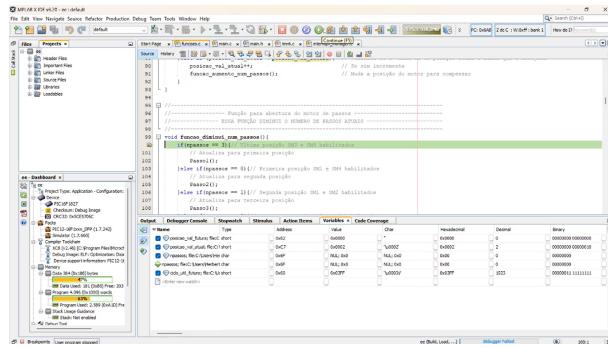


Figura 10: Função de diminuição de passo

5.3 Júlio César

Funções desenvolvidas: inicia_ultrassom, velocidade_som, media, posicao_bola_tubo.

5.3.1 Resultados

É possível visualizarmos a relação do ECHO e Trigger



Figura 11: Envio de dados

5.4 Rafael Brasileiro

Funções desenvolvidas: Comando_PiEd, modo_ventoinha, modo_valvula.

5.4.1 Resultados

A simulação do PID se inicia zerando todas as variáveis relacionadas ao seu cálculo (integrativo, proporcional, derivativo e erro) e com o recebimento de todas as variáveis necessárias para realizar o controle do sistema.

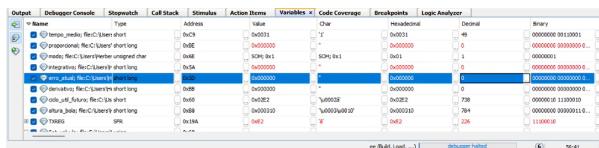


Figura 12: Variáveis relacionadas ao PID zeradas

Agora, partindo para a simulação, forçando os valores de setpoint de altura da bola de 1000 e de altura real da bola de 784. Obtemos um erro de 226, com derivativo bem alto, forçando a diminuição ainda maior do erro e tendendo a linearidade.

```

// Main PID loop
for(;;)
{
    // Read sensor values
    sensor_value = readSensor();
    error = target - sensor_value;

    // Compute PID control signal
    output = (error * Kp) + (integral * Ki) + (derivative * Kd);

    // Write output to actuator
    writeActuator(output);
}

// Function to calculate average of last 4 sensor readings
float calculateMean()
{
    float sum = 0;
    for(int i=0; i<4; i++)
        sum += sensor_values[i];
    return sum / 4;
}

void updateMedidas()
{
    // Calculate average of last 4 sensor readings
    mediaMedidas = calculateMean();

    // Calculate average of last 4 sensor readings
    mediaMedidas = (mediaMedidas * 2000) + (tempo_medidas * 2000) / 4000;
}

```

Figura 13: Valores calculados pelo PID após coleta

Cada vez que o erro se aproxima de zero o, o derivativo se aproxima mais de 0, enquanto o integrativo, graças ao seu ganho permanece com valores sempre baixos.

Forçando valores altos e negativos, podemos ver o comportamento do controle, com o derivativo tentando sempre compensar esse erro e tendendo a linearidade.

Name	Type	Address	Value	Char	Hexadecimal	Decimal	Binary
tempo_medidas	float	0x00000000	0.000000	'0'	0x00000000	0	0000000000000000
integral	float	0x00000004	-0.000000	'-'	0x00000000	-4000	1111111111000000
derivative	float	0x00000008	0.000000	'0'	0x00000000	0	0000000000000000
error	float	0x0000000C	0.000000	'0'	0x00000000	0	0000000000000000
output	float	0x00000010	0.000000	'0'	0x00000000	0	0000000000000000

Figura 14: Teste do PID forçando um erro bastante alto e negativo

O PID possui uma difícil simulação em software, conseguindo apenas visualizar os valores de derivativo, integrativo, proporcional e erro variando após as implementações. O teste em bancada é mais eficaz pois lá é possível identificar outros fatores, como quanto tempo o sistema demora para linearizar ou quão grande está sendo o overshoot, etc.