Universidade de Brasília
Faculdade UnB-Gama

Projeto Bola no Tubo

Disciplina: Eletrônica Embarcada 2-2024

Integrantes:
Jessica Kamily Oliveira de Sousa - 110123646
Karen Julia Araujo Espindola de Almeida - 202017648
Sofia Victoria Bispo da Silva - 202063542
Tifany de Paula Severo - 170115038

Brasília, Fevereiro de 2025

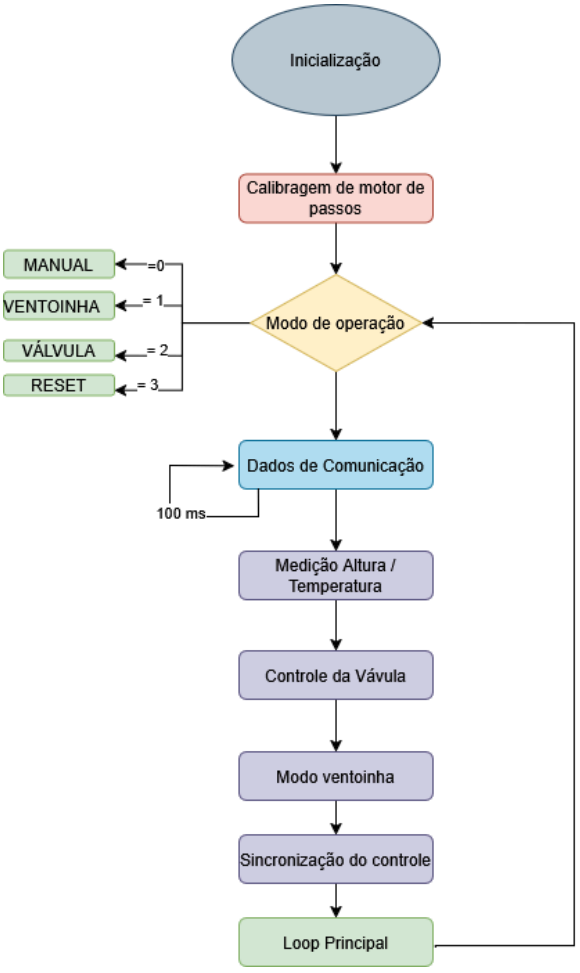
1. Introdução

O projeto tem como objetivo desenvolver um sistema de controle automatizado capaz de controlar a altura de uma bola de isopor no tubo. O sistema utiliza o microcontrolador **PIC16F1827** para ajustar o fluxo de ar gerado por uma ventoinha e controlar da válvula acoplada a um motor de passo, além disso, integra sensores, como o ultrasônico **(HC-SR04)** para realizar a medição da altura da bola, sensor óptico **(TCRT-5000)** para identificar início e fim do curso da válvula, e um sensor de temperatura **(LM35)** para garantir a precisão das medições da altura de acordo com a velocidade do som. A comunicação serial **Bluetooth** permite a transmissão e recepção de dados no formato hexadecimal, proporcionando o controle do sistema.

Neste relatório, serão descritas as metodologias abordadas para realização de cada etapa do projeto, de acordo com os requisitos definidos pelo roteiro.

2. Fluxograma

A lógica do sistema de controle é representada na Figura 1, onde é possível visualizar a representação dos principais processos descritos para ajuste de controle e funcionamento do projeto.



3. Atividades Desenvolvidas

Cada parte do projeto foi designada de acordo com o sugerido pelo roteiro, visando o cumprimento dos objetivos do projeto e participação de todos os membros da equipe. O projeto foi dividido em 4 partes, sendo elas:

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

3.1. Medição de altura de bola

A integrante **Jessica Sousa** trabalhou no desenvolvimento do firmware necessário para o cálculo da altura e da temperatura. No loop principal, foi configurado um contador que varia de **0 a 100 ms**, determinando o instante exato em que cada medição do sensor de altura é realizada. Ao final desse ciclo, a transmissão do quadro de comunicação é enviada via **Bluetooth**. Com essa contagem, é possível identificar o momento em que o **Timer 1** registra o tempo de voo da onda sonora, permitindo assim o cálculo da altura da bola.

O sensor **HC-SR04**, em conjunto com o sensor **LM35**, tem como objetivo estimar a altura da bola de isopor por meio da Equação 1:

$$\text{Altura} = \frac{\text{Tempo de Voo} \times \text{Velocidade do Som}}{2}$$

O **HC-SR04** mede o tempo de voo da onda sonora refletida, enquanto o **LM35** auxilia na compensação da variação da velocidade do som de acordo com a temperatura ambiente, garantindo maior precisão no cálculo da altura.

A altura é expressa em milímetros. O tempo de voo corresponde ao intervalo entre a emissão e a recepção do sinal de echo, sendo representado pela contagem de ciclos do timer. A velocidade do som, medida em metros por segundo, varia de acordo com a temperatura ambiente, que é monitorada pelo **sensor LM35** para garantir maior precisão no cálculo.

O cálculo da velocidade do som depende da temperatura. Para otimizar o processamento, foi utilizado apenas valores pré-determinados da velocidade do som correspondentes a temperaturas dentro da faixa de 0°C a 50°C. Isso reduz a necessidade de operações complexas, melhorando o desempenho do sistema. Esses valores pré-calculados foram escritos em uma tabela LUT no código. Para calcular a velocidade do som foi utilizado a equação abaixo:

$$C = C_0 \sqrt{\frac{\tau}{\tau_0}}$$

3.2. Movimento do motor de passo e detecção do fim de curso

A integrante **Karen Júlia** foi responsável pelo controle do motor de passo e pela detecção do fim de curso da válvula, desenvolvendo o firmware necessário para essas funções. O acionamento do motor de passo foi configurado em quatro modos, conforme ilustrado na imagem abaixo.

Posição	SM1	SM2	SM3	SM4
1	■	■		
2		■	■	
3			■	■
4	■			■

Para ativar o motor de passo, a posição da válvula é monitorada pelo sensor óptico **TCRT500**. A partir dessa leitura, verifica-se se a válvula está na posição desejada, previamente configurada por meio da comunicação serial, a saída da leitura é feita pelo **LED** azul que acende ao detectar válvula aberta. Em seguida, um **switch case** determina o modo de acionamento do motor, regulando assim a abertura da válvula. Para essa decisão, foi utilizado o periférico comparador **CMP1**, que compara a posição atual da válvula com o limite de 420.

3.3. Controle PID

A integrante **Tífany de Paula** foi responsável pelo controle **PID** do sistema, com o objetivo de ajustar a altura da bola e estabilizá-la na posição desejada. Para isso, foram implementados dois modos de operação:

- **Modo Ventoinha**: Regula o **duty cycle** do **PWM** para alterar a velocidade da ventoinha, ajustando assim a altura da bola.
- **Modo Válvula**: Controla a abertura da válvula para gerenciar o fluxo de ar disponível para a ventoinha, influenciando a altura da bola.

O firmware desenvolvido baseou-se inicialmente em uma estrutura **if-else** para verificar o modo de operação ativo e, a partir disso, acionar a função de controle correspondente. O controle **PID** foi implementado com base nas equações apresentadas a seguir.

$$e_k = sp_k - y_k \qquad u_k = K_c \left(e_k + \frac{T}{2T_i}(e_k + e_{k-1}) \right) + u_{k-1}$$

Para implementar essa equação, foi calculada a diferença (**delta**) entre a altura atual e a altura desejada, determinando assim o erro. Com esse valor, foram aplicadas as constantes do controle **PID**, conforme ilustrado na imagem abaixo.

```
//Função para calcular o erro e retornar o ajuste que deve ser feito na ventoinha ou na válvula
int ControleAjuste() {
    if (data.setpoint_height.total > data.height.total){
        erroAtual = (data.setpoint_height.total - data.height.total); //Mede a diferença entre a altura desejada e a altura lida
    }
    else {
        erroAtual = (data.height.total - data.setpoint_height.total); //Mede a diferença entre a altura desejada e a altura lida
    }
    erroAcumulado += erroAtual; //Soma o erro ao longo do tempo para eliminar o erro estacionário.
    uint16_t deltaErro = erroAtual - erroAnterior; // Mede a taxa de variação do erro, ajudando a evitar oscilações excessivas.
    erroAnterior = erroAtual;

    return (int)(ganhoK * erroAtual + tempoK * erroAcumulado + ganhoD * deltaErro); //Cálculo da saída do controlador PI
}
```

Após obter o valor de ajuste, ele é somado ou subtraído do **PWM** (no modo ventoinha) ou da posição da válvula (no modo válvula), de acordo com a necessidade de correção. Os valores das constantes utilizadas no cálculo foram determinados empiricamente por meio de testes práticos na bancada, seguindo um processo de tentativa e erro para otimizar a resposta do sistema.

3.4. Comunicação serial-BT

A integrante **Sofia Victoria** foi responsável pela implementação da comunicação serial do sistema. Para isso, foi desenvolvido um firmware utilizando a interface **UART**, um protocolo assíncrono para transmissão de dados. A taxa de transmissão (**baud rate**) foi configurada para **115200 bps**.

A detecção do fim de um quadro de comunicação é feita por meio de um **timeout**, considerando a transmissão completa após **40 ms** sem novos dados recebidos. O controle temporal é gerenciado por um único **timer**, que gera interrupções a cada **1 ms** e utiliza variáveis de controle para monitorar os eventos do sistema.

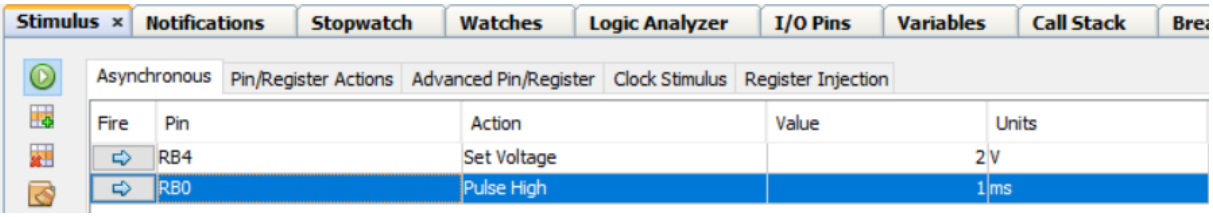
A leitura dos dados ocorre periodicamente dentro do **loop principal**, garantindo acesso exclusivo às informações e evitando a concorrência. Quando o **timer** atinge **100 ms**, os dados são enviados pelo pino **Tx**. Além disso, após **40 ms** do envio de um dado, ele é considerado como completamente transmitido. Para assegurar que apenas os **7 dados enviados** sejam processados, foi implementado um contador que verifica se o número de leituras permanece dentro desse limite.

4. Simulações e testes

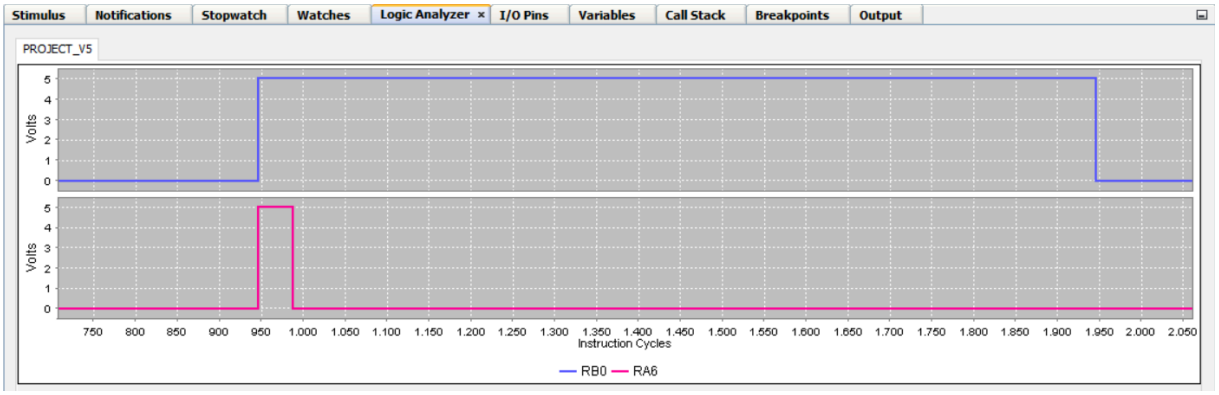
Nessa etapa de simulações e testes, foram aplicadas diferentes condições de simulação para cada fase do projeto, com objetivo de avaliar o desempenho e a funcionalidade do sistema. Os resultados obtidos serão apresentados e detalhados neste tópico.

4.1. Medição da altura:

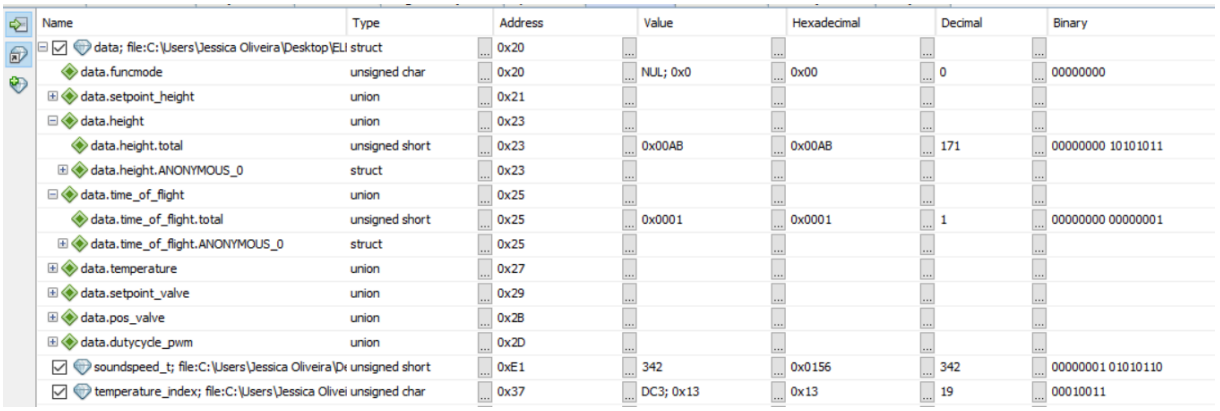
Para simulação do código referente à altura da bola, primeiramente é necessário simular a bola e o efeito que ela causa no sensor ultrassônico, sendo assim, é necessário aplicar um estímulo no tempo na porta destinada ao “Echo” do sensor



Com isso é possível obter na barra de Logic Analyser, os pulsos relacionados ao sensor ultrassônico, o pulso enviado a cada 10us na porta RB6 destinada ao trigger, e o pulso medido pelo “Echo” na porta RB0:



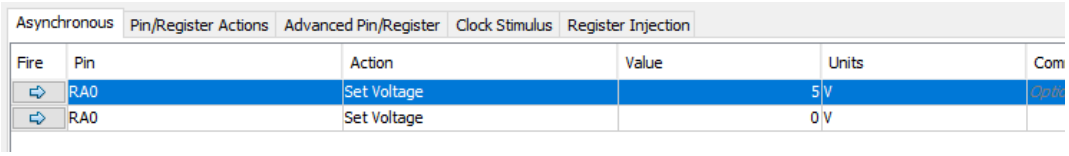
O tempo de voo é escrito na váriavel data.time_of_flight.total referente ao valor em ms, em temperatura é dado o calor em ºC e após alguns passos seguintes é dado o valor em mm, ou seja, a altura estimada é de 171mm.



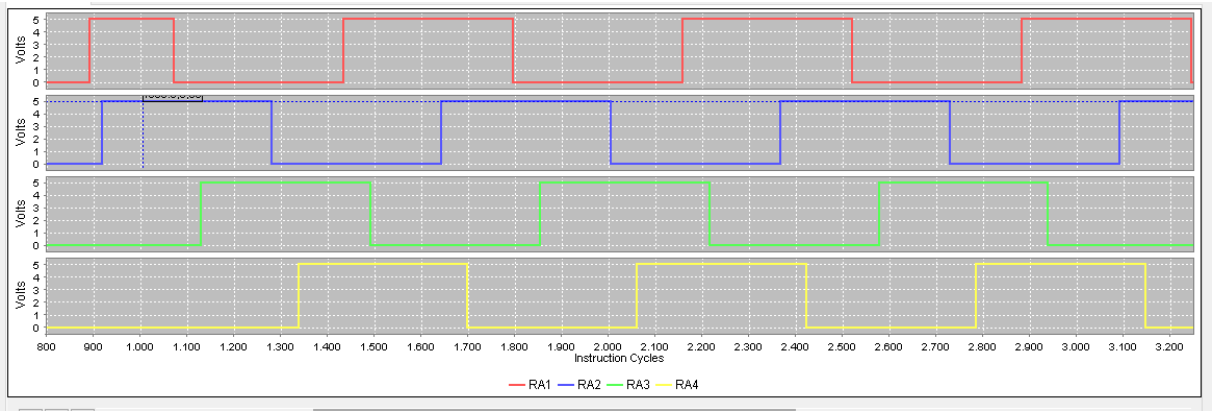
Name	Type	Address	Value	Hexadecimal	Decimal	Binary
data; file:C:\Users\Jessica Oliveira\Desktop\ELI	struct	0x20				
data.funcmode	unsigned char	0x20	NUL; 0x0	0x00	0	00000000
data.setpoint_height	union	0x21				
data.height	union	0x23				
data.height.total	unsigned short	0x23	0x00AB	0x00AB	171	00000000 10101011
data.height.ANONYMOUS_0	struct	0x23				
data.time_of_flight	union	0x25				
data.time_of_flight.total	unsigned short	0x25	0x0001	0x0001	1	00000000 00000001
data.time_of_flight.ANONYMOUS_0	struct	0x25				
data.temperature	union	0x27				
data.setpoint_valve	union	0x29				
data.pos_valve	union	0x2B				
data.dutycycle_pwm	union	0x2D				
soundspeed_t; file:C:\Users\Jessica Oliveira\D	unsigned short	0xE1	342	0x0156	342	00000001 01010110
temperature_index; file:C:\Users\Jessica Olivei	unsigned char	0x37	DC3; 0x13	0x13	19	00010011

4.2. Motor:

A validação de testes e simulação foram realizadas pelo **MPLAB X IDE,** para isso foi definido 2 tipos de estímulos na entrada **RA0** que corresponde o **Lim**, que é capaz de realizar a leitura entregando o valor digital de 1 ou 0, ou seja, quando válvula aberta detectada é retornado 1, caso contrário retorna 0.



A imagem abaixo representa o formato de onda de cada etapa correspondente do sentido horário e anti-horário do motor de passo, onde seus valores são pré-definidos de acordo com o sentido que o motor deve se mover para melhor atender o controle da válvula.



Sentido Horário

Sentido anti-horário

Ao utilizar a ferramenta de visualização **I/O pins** é possível identificar o momento que o **LED** acende a partir do pino **RA7** ao identificar a válvula**,** além disso conseguimos acompanhar em qual estado o motor de encontrar pelos pino RA1,

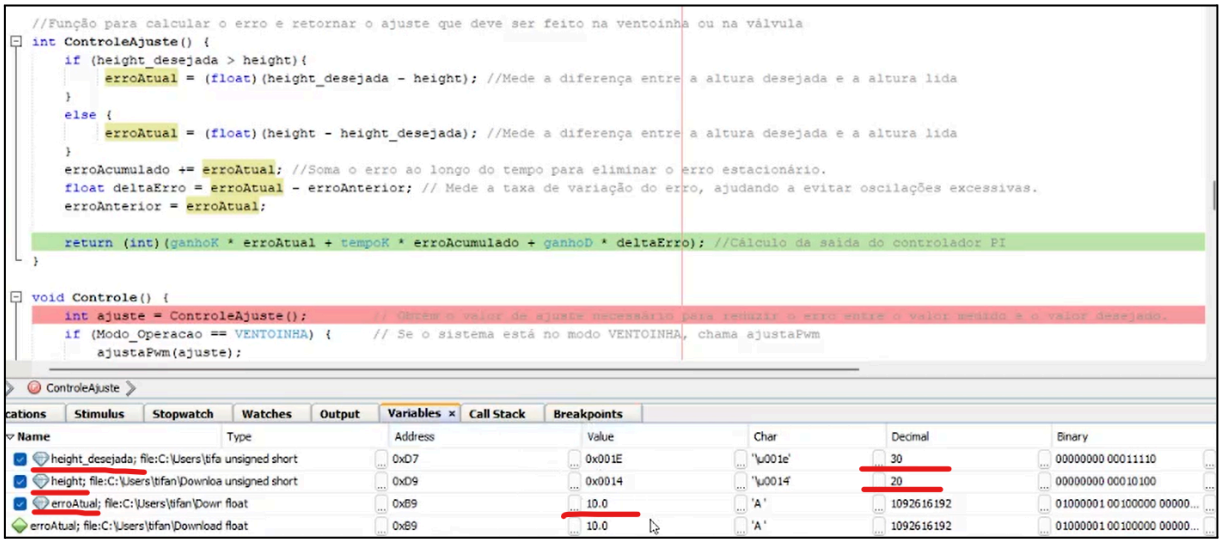
RA2, RA3 e RA4, na imagem abaixo o seu estado é 2.

Pin	Mode	Value	Owner or *Mapping*
RA0	Ain	5.0V	**RA0**/AN0/CPS0/C12IN0-/SDO2
RA7	Dout	0	**RA7**/OSC1/CLKIN/P1C1/CCP21/P2A1
RA1	Dout	0	**RA1**/AN1/CPS1/C12IN1-/SS2
RA2	Dout	0	**RA2**/AN2/CPS2/C12IN2-/C12IN+/VREF-/DACOUT
RA3	Dout	1	**RA3**/AN3/CPS3/C12IN3-/C1IN+/VREF+/C1OUT/CCP3/SRQ
RA4	Dout	1	**RA4**/AN4/CPS4/C2OUT/T0CKI/CCP4/SRNQ

Verificação I/O pins

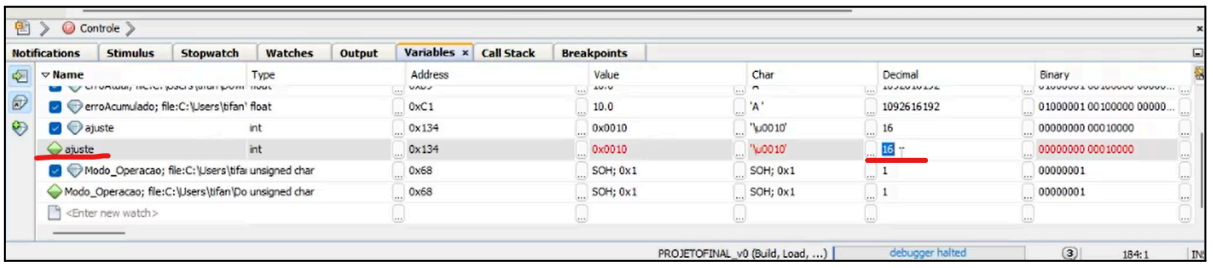
4.3. Controle:

Para simular o controle, definiu-se a altura desejada como 30 e a altura medida como 20. Assim, o erro esperado (delta) seria 10. Como mostrado na imagem abaixo, o cálculo do erro foi realizado corretamente.



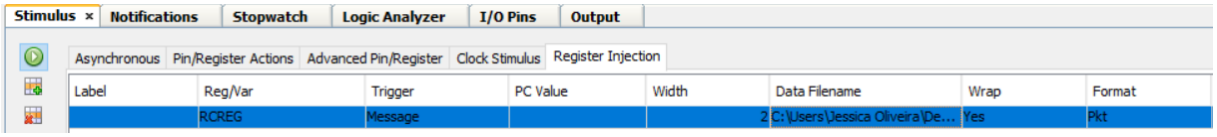
Na sequência da simulação, é necessário calcular o ajuste a ser aplicado no controle por meio do PID, utilizando a multiplicação pelos coeficientes correspondentes. O valor esperado seria: (0,1 × 10) + (1,5 × 10) + (0,005 × 10) = 16,05.

A imagem abaixo confirma que o cálculo foi realizado corretamente, evidenciando o funcionamento adequado do controle PID na simulação.

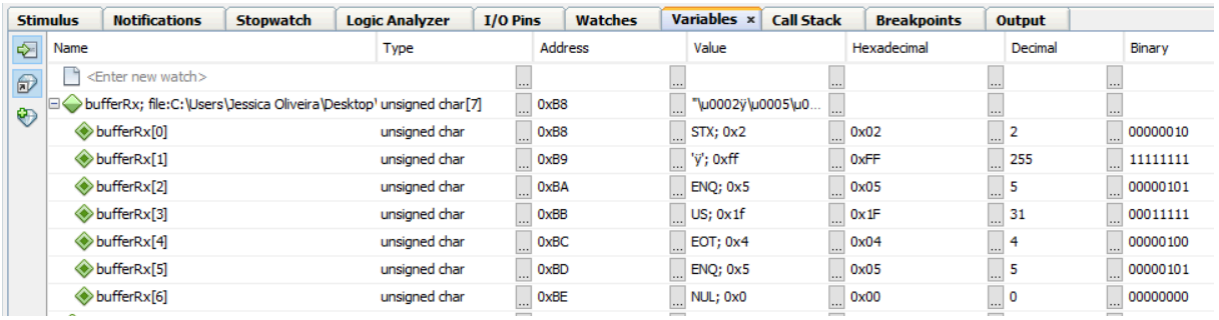


4.4. Comunicação:

Para o teste de comunicação, foi aplicado um estímulo no Register Injection no registrador RCREG e observado o valor armazenado no buffer.



A sequência utilizada: 0x02, 0xFF, 0x05, 0x1F, 0x04, 0x05, 0x00



O valor sendo transmitido por TxREG, correspondendo aos valores de 15bytes.