# Laboratório de sistemas microprocessados

# Projeto Final

**Aluno:** David Gonçalves Mendes

**Matrícula:** 190056967

## Objetivo

Este projeto tinha como objetivo o uso dos conhecimentos obtidos ao longo da disciplina de Sistemas Microprocessados e do laboratório para desenvolver, utilizando a plaquinha MSP430F5529, um carrinho motorizado de controle remoto controlado via bluetooth por um dispositivo externo (celular). Este carrinho terá como particularidade um sensor de ultrassom que será utilizado para identificar obstáculos no caminho e, se for o caso, limitar as ações do usuário, fazendo com que o carrinho para e altere a sua trajetória, impedindo a colisão.

## Componentes

Além da MSP, os componentes que foram necessários para o desenvolvimento foram:

- Módulo bluetooth serial SPP-C (Compatível HC-05/HC-06) - Conexão com o celular;
- KIT Chassi para Robô 2WD – Base para o Carrinho (acompanha motor);
- Módulo Driver Motor com dupla Ponte H L298 – Controlar o sentido de rotação dos motores;
- Módulo Sensor Ultrassônico HC-SR04 – Medir a distância para o obstáculo;
- Jumpers Macho-Fêmea
- Jumpers Fêmea-Fêmea
- Jumper Macho-Macho
- Protoboard
- Power Bank

## Funcionamento

- **Motores**

O acionamento de cada motor foi feito por intermédio de uma ponte H, configurados, inicialmente com um PWM de 50%. Para aumentar a velocidade dos motores, foi criada uma variável para indicar a velocidade de rotação, que varia de 3 (PWM = 30%) a 9 (PWM = 90%). Inicialmente, a ideia era variar o PWM de 10% a 90%, porém, ao efetuar alguns testes, foi

observado que abaixo de 30% os motores ficam tão lentos que o carrinho praticamente nem sai do lugar.

Para gerar os sinais de PWM, foram usados os timers TA2.2 para o EnA (Motor direito) e TB0.6 para o EnB (Motor esquerdo) e ambos foram configurados no modo OUT 6 (Toggle/Set) com o modo UP de contagem.

A ponte H também nos possibilita controlar o sentido de rotação dos motores, e para isso, utilizamos mais 4 pinos da MSP, sendo eles: P8.1 - IN1, P8.2 - IN2, P4.2 - IN3, P4.1 - IN4. Segue abaixo a tabela verdade para a rotação do motor ao fazer o uso do módulo L298:

| En A (En B) | In 1 (In 3) | In 2 (In 4) | Função |
|---|---|---|---|
| 0 | X | X | Livre |
| 1 | 1 | 0 | Direto |
| 1 | 0 | 1 | Reverso |
| 1 | 0 | 0 | Freio |
| 1 | 1 | 1 | Freio |

A partir da tabela acima, fica evidente que, para mudar o sentido de rotação dos motores, só é necessário inverter os valores dos pinos In's e manter o PWM dos enables normalmente.

Além disso é necessário fornecer duas entradas de 5V para a ponte H, uma que irá alimentar o circuito lógico digital e outra que irá alimentar os motores em si. Como a plaquinha possui apenas dois pinos de 5V, foi optado por conectar a alimentação do motor direto com 5V, sem nenhum outro componente em paralelo, para evitar a divisão de corrente e todos os outros componentes conectados em paralelo com o outro pino de 5 V que sobrou.

- **Terminal serial**

O terminal serial é de suma importância para o funcionamento do projeto, já que é ele quem nos possibilita controlar de forma prática as ações do carrinho por meio de comandos simples que podem ser enviados pelo próprio celular de forma assíncrona (UART). Mais especificamente, fazemos o uso de um módulo bluetooth, que será conectado aos pinos da MSP que também têm a função de transmitir e receber dados de forma assíncrona. Dessa forma iremos conectar o pino P3.3 (TXD da MSP) ao RXD do módulo bluetooth e o pino P3.4 (RXD da MSP) ao TXD do módulo.

Como o módulo bluetooth vem por padrão configurado a 9600bps, e foi optado por utilizar o ACLK, o UART foi configurado com BRW = 3, BRS = 3, BRF = 0 e UCOS16 = 0. No entanto, essa configuração tem um erro máximo de –21.1% a 15,2% para a transmissão e de –44,3% a 21,3% para a recepção, o que é um erro consideravelmente grande. Embora esse erro não tenha ocasionado em nenhum problema para o projeto, para evitar comportamentos imprevisíveis, uma configuração alternativa que poderia ter sido utilizar o SMCLK com BRW = 109, BRS = 2, BRF = 0 e UCOS16 = 0, que daria um erro máximo de -0.2% a 0.7% para a transmissão e de –1% a 0.8% para a recepção, o que é um erro consideravelmente menor.

O módulo bluetooth, apesar de ser alimentado em 5V, tem as suas entradas TXD e RXD alimentadas com 3,3V, portanto não será necessário fazermos o uso de um divisor resistivo para esse componente.

- **Sensor ultrassom**

O sensor de ultrassom HC-SR04 é a principal particularidade desse projeto, pois é com ele que podemos medir a distância até algum obstáculo e, assim, impedir alguma colisão iminente que possa vir a acontecer.

Antes de conectar o sensor ultrassom, é preciso se atentar que o sensor é alimentado em 5V, portanto é necessário o uso de um divisor resistivo no retorno do echo para não queimar o pino da launchpad (ligar o echo em um resistor de 4K7 e o resistor em paralelo com um resistor de 10K (ligado na terra) e com o pino da plaquinha).

O pino da plaquinha utilizado para capturar o Echo será o P2.0 (timer TA1.1), já para o trigger, iremos utilizar o pino P1.5 (Timer TA0.4).

Para verificarmos a distância do sensor até um objeto, foi ativada uma interrupção que verifica se foi capturado um flanco, verifica se foi de descida ou subida e, caso seja de descida, armazena o valor do contador do timer naquela posição e quando o flanco capturado for de subida, será subtraído o valor do contador do timer na nova posição para obter o tamanho do echo. Com isso, já conseguimos obter a distância e, caso o carrinho esteja a menos de 15 centímetros de bater, ele deve parar, dar uma ré e fazer uma pequena curva para um lado.

Um dos grandes problemas desse sensor é que, como ele é um sensor de som, caso a superfície seja muito irregular ou caso a angulação do sensor para a superfície seja muito diferente de 90 graus, é bem provável que o módulo não irá funcionar e o carrinho irá bater, porém eu não fui capaz de em uma boa solução para contornar esse problema.

- **Programa principal**

Para o programa principal, foram feitos 11 comandos que podem ser enviados pela porta serial:

**h**: help. Exibe a mensagem com todos os comandos;

**L**: Left. Desliga o motor direito e deixa apenas o motor esquerdo acionado;

**R**: Right. Desliga o motor esquerdo e deixa apenas o motor direito acionado;

**S**: Stop. Desliga os dois motores;

**M**: Modo 1. Define o estado do carrinho como andando para frente;

**m**: Modo 2. Define o estado do carrinho como andando para trás (ré);

**A**: Acelerar. Aumenta a velocidade do carrinho para cada vez que for mandado esse comando (velocidade máxima = 9);

**a**: Desacelerar. Diminui a velocidade do carrinho para cada vez que for mandado esse comando (velocidade mínima = 3);

**<**: Curva leve a esquerda. Aumenta o PWM do motor direito para que o carrinho vire levemente a esquerda, pressionar várias vezes para aumentar a velocidade da curva. Caso o carrinho esteja em velocidade máxima (9), o PWM do motor esquerdo que é diminuído.

**>**: Curva leve a direita. Aumenta o PWM do motor esquerdo para que o carrinho vire levemente a direita, pressionar várias vezes para aumentar a velocidade da curva. Caso o carrinho esteja em velocidade máxima (9), o PWM do motor direito que é diminuído.

**N**: Normal. Iguala o PWM que possa ter sido alterado nos dois comandos anteriores para deixar igual entre os dois motores e permitir novamente que ele ande reto.

## Possíveis melhorias

Nesse projeto conseguimos botar em prática alguns tópicos da disciplina de Sistemas Microprocessados, tais como Timers, GPIO e UART e fazer um projeto real e divertido. No entanto, apesar de já funcionar bem, sempre há espaço para melhorias e, ao longo do projeto, percebi alguns detalhes que poderiam ser corrigidos/melhorados, mas que não foram arrumados durante o desenvolvimento. Portanto, finalizo esse texto apresentando alguns pontos que ainda podem ser revisados para que esse projeto fique ainda melhor e mais sofisticado:

- Quanto a montagem do carrinho, algo que me incomodou bastante foi que eu não encontrei espaço no chassi para acomodar a plaquinha direito, então ela acabou ficando solta em cima da protoboard, sendo segurada apenas pelos fios de conexão. Acredito que eu poderia ter feito algum tipo de plataforma para colocar a plaquinha ao carrinho para aumentar a estabilidade da mesma e evitar possíveis acidentes.
- Outro problema que surgiu foi quanto ao alinhamento do carrinho, pois quando a velocidade era mais baixa (PWM de cerca de 50% para menos), o carrinho quase sempre puxa para um dos lados, enquanto que com uma velocidade mais alta, isso raramente ocorria.
- Em relação ao comportamento do carrinho ao se aproximar muito de um objeto, foi utilizado delay por software pois todos os timers já estavam sendo utilizados e eu acabei não conseguindo tempo para voltar nessa parte e tentar fazer utilizando timer. Sofisticando ainda mais, seria interessante que, caso o carrinho fosse bater, ele fizesse esse comportamento, mas depois voltasse a andar. Acabei não implementando isso por medo do carrinho bater devido as limitações do módulo HC-SR04.
- Por fim, nos modos de curva leve para um lado ou outro, eu não consegui testar muito bem pelo fato de que o carrinho raramente anda reto quando a velocidade é baixa e eu não tenho muito espaço para testar com o carrinho em uma velocidade mais elevada (acabei ficando com medo de bater o carrinho e danificar algum componente).