Sistemas Embebidos (PL1)
Mini-projeto: Robô móvel diferencial
Relatório global

Miguel Hirche Almeida (2021250125)
Leonardo Gonçalves (2020228071)

Conteúdo

1 Materiais usados

- Pololu 3500 (kit com estrutura para robô, motores, drivers, encoders e alimentação)
- Placa baseada em STM32 (2x)
- Pilhas AA NiMH (6x)
- Módulo joystick
- Módulo bluetooth HC-05 (2x)
- Sensor ultrassónico (HC-SR04)
- Fios condutores para ligações
- Resistências, díodos, condensadores e LEDs (componentes passivos)

2 Pinout e configuração

PB6 (SR04_Echo)	TIM4 CH1 (input capture direct mode)	Mede a largura do pulso.
PF13 (SR04_Trig)	GPIO Output	Inicia o pulso.
PA5	TIM2 CH1 (PWM generation)	Velocidade da roda direita.
PB0	TIM3 CH3 (PWM generation)	Velocidade da roda esquerda.
PE15 (DIR_L)	GPIO Output	Direção da roda esquerda.
PB10 (DIR_R)	GPIO Output	Direção da roda direita.
PD5	USART2 TX	RX do módulo bluetooth.
PD6	USART2 RX	TX do módulo bluetooth.
PA0	TIM5 (Encoder MODE T1 e T2)	TIM 1 do encoder da roda direita.
PA1	TIM5 (Encoder MODE T1 e T2)	TIM 2 do encoder da roda direita.
PE9	TIM1 (Encoder MODE T1 e T2)	TIM 1 do encoder da roda esquerda.
PE11	TIM1 (Encoder MODE T1 e T2)	TIM 2 do encoder da roda esquerda.

Tabela 1: Pinout projeto robô

PC0	ADC IN	Input 1 ADC 1.
PA3	ADC IN	Input 8 ADC 1.
PD5	USART2 TX	RX do módulo bluetooth.
PD6	USART2 RX	TX do módulo bluetooth.

Tabela 2: Pinout projeto joystick

- **Joystick**

  Para configurar o joystick ativamos o canal ADC1 que vai ler os dados do joystick por DMA, para isso também tivemos de ativar a interrupção do DMA.

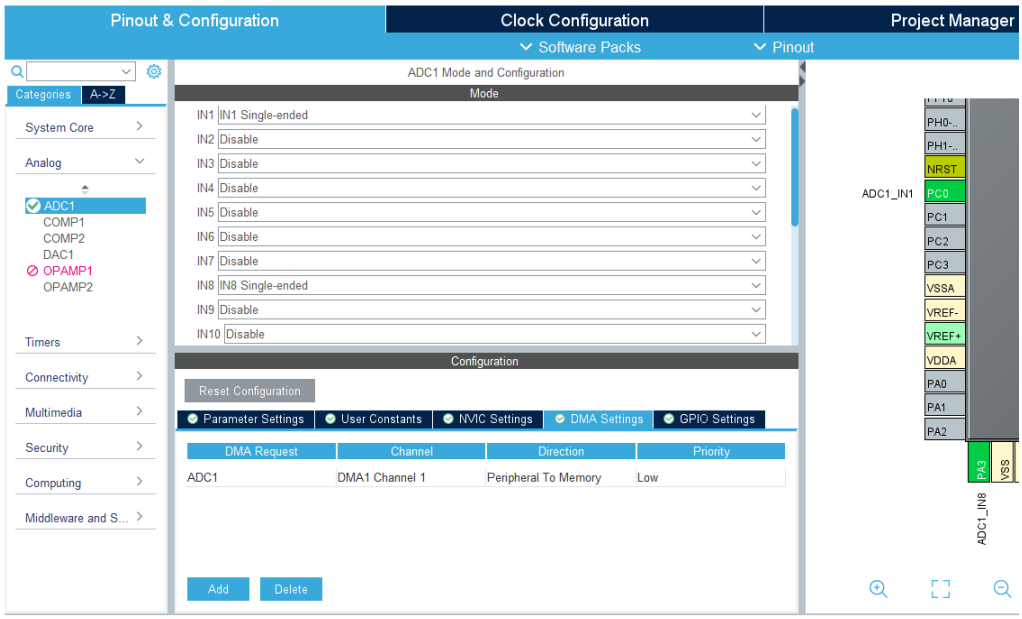

Figura 1: JoyStick ADC config

- **Bluetooth** Inicialmente tivemos de configurar os dois módulos bluetooth hc-05 pelos comandos AT, configurámos um dos módulos como master e o outro como slave, para isso utilizamos o BIND no master para o endereço do slave. Ainda configurámos a Baud Rate de ambos os módulos para 115200. De seguida, já na CubeIDE, configuramos a porta USART2, de modo a ligar os pinos TX e RX dos módulos bluetooth para permitir a comunicação Serial.

- **Motores** Os motores são acionados através de um sinal PWM e de um pino que determina a direção da rotação das rodas (FW/BW). Para gerar um pulso PWM é necessário configuramos um Timer como PWM generaton para cada uma das rodas. Configuramos o TIM2 CH1 para a roda direita e o TIM3 CH3 para a roda esquerda. E ainda os Pinos PE15 E PB10 como GPIO_Output para configurar o byte de controlo da direção das rodas.

- **Encoders** Os encoders romi são do género quadrature encoders com 2 canais de aquisição de dados, estes funcionam por Hall Effect. Os 2 canais adquirem pulsos que nos permitem obter a posição angular de um eixo através da contagem do número de pulsos emitidos. Para a aquisição dos dados dos encoders foi necessário configurar 2 Timers em Combined Channel como Encoder MODE T1 e T2, nomeadamente o TIM1(PE9 e PE11) e o TIM5(PA0 e PA1), um para a roda esquerda e o outro para a roda direita, respetivamente.

- **Sensor de Distância (SR-04)** O sensor de distância transmite uma onda sonora de alta frequência (40 kHz), e detecta sua reflexão. A partir do tempo que esse processo demora, é possível determinar a distância entre o sensor e um objeto. Para isso, foi configurado um timer em modo Input Capture (TIM4), que permite determinar a largura do pulso recebido no pin Echo. Com isso e a velocidade do som no ar, calcula-se a distância, que caso for menor que 10 cm, impede o robô de se movimentar para frente.

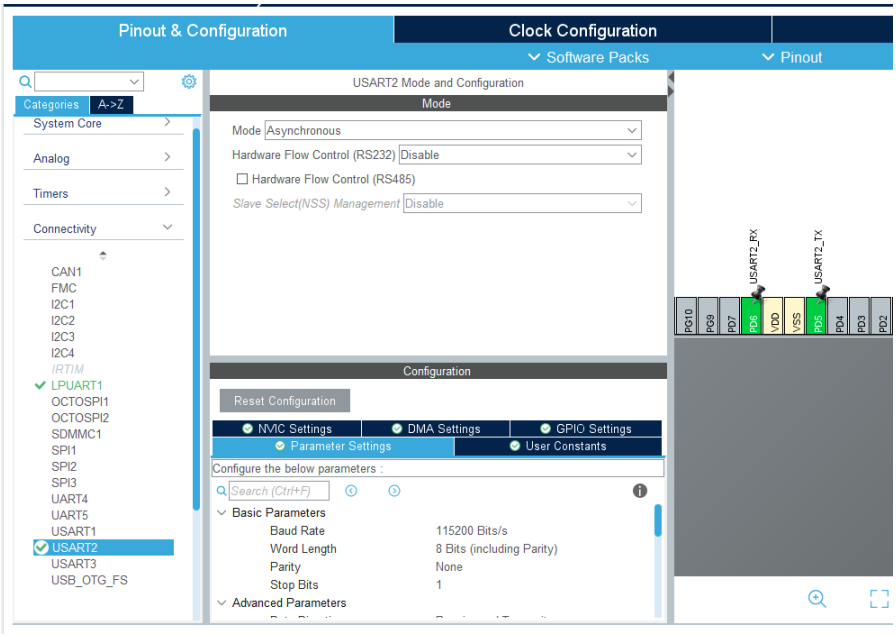

Figura 2: Bluetooth USART2 config

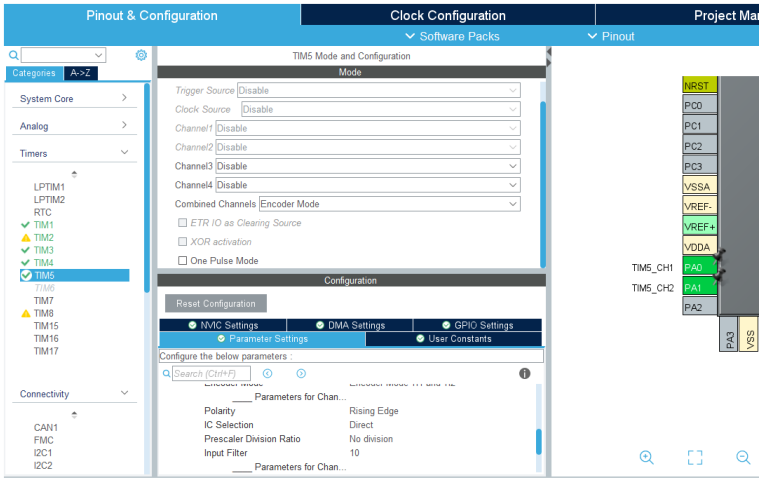

Figura 3: Timers Encoder Config

- **FreeRTOS Config** Vamos utilizar a funcionalida do FreeRTOS, que nos permite implementar um sistema operativo de tempo real robusto e eficiente. A configuração envolveu a criação das seguintes tarefas:
  - *DistanceSensor*;
  - *MovControl*;
  - *EncOdometry*;

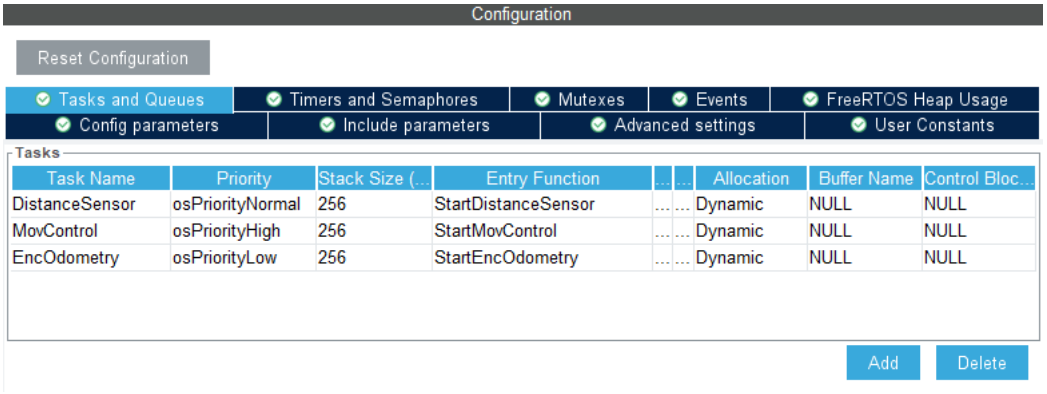

Figura 4: FreeRTOS Tasks Config

3 Arquitectura e funcionamento

Um robô diferencial é constituído por duas rodas independentes em um mesmo eixo. Isso permite que seja controlado pela diferença de velocidade e direção de rotação entre as rodas (por isso, ser diferencial). O kit Pololu 3500 possui um chassi, um eixo, duas rodas motorizadas e um suporte para pilhas. Desse modo, acoplando um micro-controlador STM32 ao kit permite realizar o controlo das rodas e, consequentemente, o movimento do robô.

A interface entre o usuário e o robô será feita através da porta série, por comunicação bluetooth. Um módulo bluetooth (joystick) estabelece comunicação com o outro (robô). Então, os dados do joystick são enviados ao microcontrolador, que realiza o controlo das rodas de acordo com o movimento desejado. Além disso, o robô terá uma competência de localização, que se resume em uma auto-localização obtida através do trajeto realizado desde o ponto inicial ao ponto final. Como o trajeto do robô é apenas determinado pela velocidade angular de cada roda conseguiremos calcular qual foi o trajeto feito pelo robô e a sua posição final.

4 Arquitetura de Software e Plano de Testes

A arquitetura de software permite-nos definir um conjunto de módulos base para uma implementação eficaz do robô móvel diferencial usando o micro-controlador STM32, sendo assim, a arquitetura de software deve de apresentar tarefas essenciais para a implementação do software, garantindo a fiabilidade e o sucesso do projeto do robô móvel diferencial.

Neste projeto será utilizado um sistema operativo de tempo real, o FreeRTOS, o qual funciona à base de tarefas, consequentemente que representam diversas funcionalidades necessárias para a implementação do robô. Definimos, à priori, as seguintes tarefas:

- Controlo de Movimento: Esta tarefa será responsável pelo controlo do movimento dos motores, processando os comandos enviados pelo utilizador, de modo a enviar para os motores os comandos necessários para controlar a velocidade e direção dos mesmos. O controlo de velocidade dos motores requer o uso de timer para geração de ondas PWM.
- Comunicação e Envio de Comandos: De modo a permitir a comunicação entre o robô e os dispositivos externos como o computador e o joystick. Pelo que é necessário configurar a comunicação entre as duas partes, neste caso usando os módulos Bluetooth.
- Aquisição de Dados por Odometria: Realiza as leituras dos encoders colocados nos motores do robô, de modo a obter as informações sobre o deslocamento e orientação do robô. A decodificação dos encoders é feita por timers (HAL encoder mode).
- Detetor de Proximidade: Calcula a distância entre o robô e um objeto à sua frente, e garante que não ocorram colisões.

Usaremos também um semáforo, disponibilizado pelo FreeRTOS, de modo a sicronizar a comunicação entre as tarefas:

- Semáforo de sincronização de tarefas: Garante o acesso exclusivo à tarefas que necessitam de precisão relativa à tempo em sua execução (controlo de movimento e sensor de distância).

Um plano de testes adequado é essencial para garantir um funcionamento correto de todas as funcionalidades que o robô móvel diferencial deve desempenhar. Logo sugerimos o seguinte plano de testes:

- Teste de Controle de Movimento: verificar os dados recolhidos pelo joystick ou pela interface de comunicação com utilizador são viáveis; confirmar se os dados recolhidos são corretamente transmitidos para os motores; verificar se a tarefa de controle de movimento está a receber corretamente os comandos de velocidade e direção; Verificar se os motores respondem corretamente ás ordens de execução;
- Teste de comunicação Bluetooth: verificar se as informações de controle são corretamente transmitidas via Bluetooth.
- Teste de Odometria: Verificar se os encoders estão a funcionar corretamente e se fornecem os dados de deslocamento e orientação do robô alinhados com o movimento real.
- Teste do sensor de proximidade: verifica se o sensor deteta os obstáculos e envia dados corretamente de modo a evitar a colisão com objetos.
- Teste de sincronização e comunicação entre tarefas: verificar se o semáforo está a ser usado corretamente, de modo a existir uma comunicação adequada entre as tarefas; Confirmar a prioridade entre as tarefas de modo a evitar bloqueios ou conflitos de recursos; Verificar se as tarefas estão a ser executadas nos intervalos de tempo desejados.
- Teste de fiabilidade: executar um teste final de longa duração, de modo a confirmar a fiabilidade e robustez do sistema; Verificar quaisquer falhas que porventura possam existir no sistema.

Tabela 3: Requisitos e planos de testes

ID do requisito	**Descrição do requisito**	**Plano de Teste**
REQ01	O sistema deve de receber e mapear entradas de um joystick conectado ao microcontrolador, e por sua vez mapeá-las para um sinal PWM.	Teste de verificação da entrada e mapeamento dos sinais obtidos pelo joystick.
REQ02	O sistema deve ser capaz de controlar a velocidade e sentido de rotação de cada uma das rodas do robô.	Teste de controlo da velocidade e sentido de rotação das rodas.
REQ03	O sistema deve de ajustar a velocidade e sentido das rodas em tempo real através dos dados de entrada do joystick.	Teste de controlo do movimento das rodas através do joystick.
REQ04	O sistema deve de comunicar entre as dois micro-controladores através do módulo bluetooth HC-05.	Teste de comunicação entre placas e conexão bluetooth.
REQ05	O sistema deve de receber os dados recolhidos pelos enconders e por sua vez obter as informações sobre o deslocamento e orientação do robô em tempo real.	Teste de verificação dos dados recolhidos pelos enconders e verificação da exatidão dos dados de deslocamento e orientação calculados.
REQ06	O sistema deve de detetar e evitar obstáculos através de um sensor de proximidade.	Teste de verificação de deteção de obstáculos.
REQ07	O sistema deve desempenhar todas as tarefas em tempo real.	Teste de simulação do sistema pela sua íntegra, testando todas as tarefas que cabem ao sistema desempenhar.

5 Interface com Elementos do Sistema e Implementação

5.1 Joystick

Os dados de entrada do joystick são recolhidos por DMA(Direct Access Memory). Depois de serem recolhidos na callback do ADC é chamada uma função que vai mapear os valores recolhidos pelo ADC para valores a serem transmitidos para o robô, de modo a gerarem um sinal PWM que permita controlar a velocidade e sentido de cada uma das rodas do robô em tempo real.
Primeiramente, inicializamos o ADC com DMA com o uso da função *HAL_ADC_Start_DMA(&hadc1, (uint32_t *) adcBuff, 2)*. Quando é feita uma nova leitura do ADC do joystick por DMA é chamada a Callback do DMA, aqui é chamada a função *Calc_PWM()* onde vai ser feito o mapeamento dos valores lidos pelo ADC para os respetivos valores do PWM.

5.2 Bluetooth - HC-05

Os módulos bluetooth são responsáveis por estabelecer a comunicação entre os dois microprocessadores. Esses foram configurados em modo master (joystick) e slave (master), com uma baud rate de 115200. A placa acoplada ao joystick envia os dados pré-processados (direção e velocidade das rodas) ao robô. Os valores recebidos via interrupção são atribuidos à variáveis, que atualizam o movimento do robô dentro da task de controlo.
Do lado do master, ou seja do joystick, a task *StartBluetoothTx* executa a função de transmitir os dados pelo bluetooth usanddo comunicação Serial, que é feita através da chamada da função *HAL_UART_Transmit(&huart2,tx_buff, 4, 100)* e envia os dados mapeados anteriormente pela função *Calc_PWM()* para o segundo módulo (slave) que se encontra no robô. Quando os dados são transmitidos, é ativada uma Interrupção neste segundo módulo que procede com a leitura dos dados na função *HAL_UART_RxCpltCallback()*

5.3 Sensor de distância - HC-SR04

Inicialmente, *SR04_Trig_Pin* é setado a LOW por 2 microsegundos. Isso evita ripples no pulso que será gerado posteriormente. Depois, esse pin é setado a HIGH por 10 microsegundos. Pelo modo Input Capture do TIM4, deteta-se o pulso e sua largura *diff* em microsegundos. A partir disso, é calculada a distância ao objeto *distance*, que é usada para o controlo do movimento do robô.

5.4 Controlo das Rodas

Os valores recebidos via bluetooth são atribuidos à variáveis, que atualizam o movimento do robô dentro da task de controlo. A direção é atribuida de forma direta. A velocidade é utilizada para gerar ondas PWM, cujo duty cycle corresponde ao percentual da velocidade dos motores. Vale ressaltar que o robô não realiza movimento caso esse implique em uma possível colisão.
Os dados do PWM recebidos pelo Bluetooth são transmitidos para as rodas através de sinais PWM na task *StartMovementControl()*.O sinal PWM é gerado através do uso da função *__HAL_TIM_SET_COMPARE()* e o byte de direção das rodas é ativado através da chamada da seguinte função *HAL_GPIO_WritePin()*, com os respetivos argumentos para cada caso.

5.5 Encoders

Os dados dos encoders são recolhidos pelos timers correspondentes a cada uma das rodas, estes que foram previamente configurados em Encoder Mode T1 e T2. Os encoders têm uma resolução de 1440 por cada rotação completa da roda. È chamada uma função que com base no número de counts que o encoder e com as medidas das dimensões do robô diferencial, nos permite calcular a sua posição e orientação atual usando as equações de odometria.
Usamos a task *Start_Encoder()* para chamar a função *get_robot_position()* e é nesta funçao onde é lido os dados dos encoders de cada uma das rodas e ainda é calculada a posição e a orientação do robô através do uso das equações de odometria.

6 Conclusão

Em conclusão, a implementação de um robô diferencial em STM32 no âmbito da disciplina de Sistemas Embebidos foi uma experiência bastante desafiadora e enriquecedora. Aprofundamos e aplicamos amplamente todos os conhecimentos adquiridos ao longo do semestre.
Consideramos que a implementação do robô diferencial foi um sucesso, pois este desempenhou todos os testes descritos no plano de testes relativos aos requisitos mínimos. Além disso o trabalho de equipa foi muito satisfatório, realçando o empenho de ambos os elementos da equipa, que desempenharam as tarefas em equipa de modo a progredirem com o desenvolvimento do projeto em conjunto com direção aos objetivos estabelecidos.

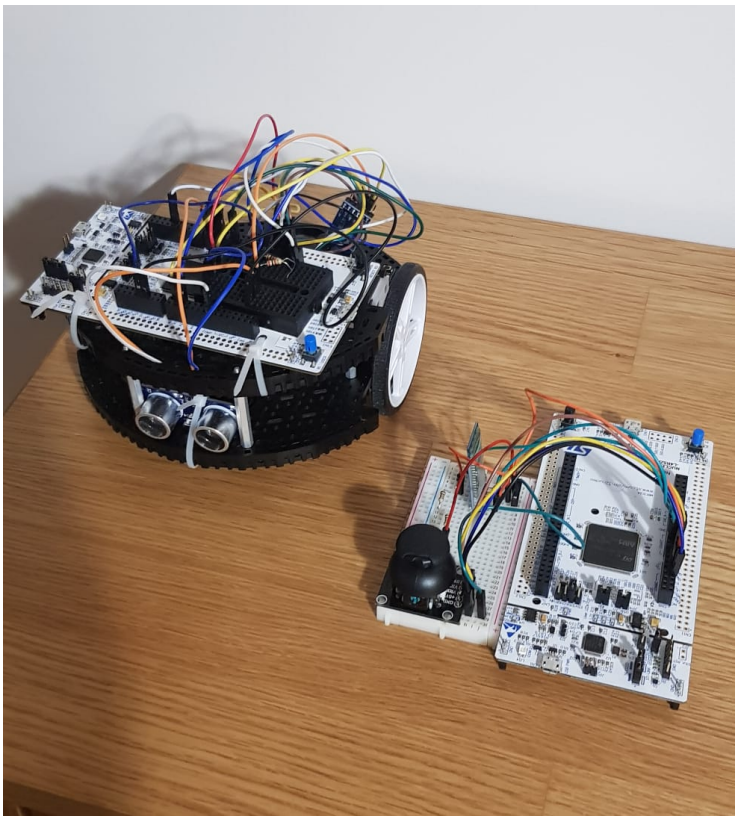

Figura 5: Robô Diferencial e Joystick