Mini Projeto - Braço Robótico

José Filipe Alves Chaves - 202006295, Luís Silva Sousa - 202007472

I. Introdução

O objetivo deste mini projeto consiste em construir um braço robótico capaz de separar blocos LEGO distribuídos de forma aleatória numa área selecionada de acordo com a sua cor. Os componentes necessários para a realização deste projeto incluem:

- Microcontrolador Raspberry Pi Pico (MC);
- 4x Servos SG90 9g 180°;
- Pico Servo Driver;
- Sensor de cor RGB (TCS34725);
- Sensor de distância TOF (VL53L0X);
- Transformador de corrente (5V/1.55A);
- Cabos do tipo ”Jumper”;
- Peças de montagem do braço robótico (plástico);

O ”budget”necessário para adquirir todos estes componentes rondou os 30€ em compras no AliExpress com portes incluídos. De notar que o MC e o material para construção do braço robótico foram fornecidos pelo docente da unidade curricular.

II. Sensores

A. Sensor RGB (TCS)

O sensor de cor (figura 1) comporta uma matriz de fotodíodos (fotodíodos sensíveis a vermelho, verde e azul - RGB) e conversores ADC para converter o valor de corrente dos fotodíodos em valores digitais. A comunicação deste sensor é efetuada recorrendo à comunicação I2C, para a qual a livraria *Wire1* é utilizada.

Começamos por criar o *setup* correspondente ao sensor definindo o tempo de integração e o ganho do mesmo, para os quais o valor inicial foi 24ms (quanto maior o tempo de integração, menor é a influência do ruído nos valores obtidos aumentando assim a precisão com o custo de ser menos responsivo) e ganho 4x (só terá um impacto significativo mediante condições de iluminação fracas), respetivamente.

A função, `getRawData_noDelay()`, foi utilizada como principal meio de obtenção dos valores RGB detetados. Esta realiza um *delay* com o tempo escolhido no tempo de integração após a medição do sensor, porém como no nosso caso utilizamos um intervalo de espera entre cada *loop*, não necessitamos esperar pelo próximo ciclo de integração (temos é de garantir que o intervalo usado é maior que o tempo de integração).

Seguidamente, recorrendo à livraria *Wire1*, inicializamos o sensor colocando o sinal do relógio no pino GPIO 19 e o sinal de dados no pino GPIO 18, auxiliado de uma condição para saber se o sensor foi conectado com sucesso ou não.

Dentro do *loop* principal, chamamos a função `getRawData_noDelay()`, que nos vai permitir obter os valores RGB para cada *loop* e associar à cor de interesse.

Figura 1. Sensor RGB (TCS)

B. Sensor de distância (TOF)

O sensor TOF (figura 2) vem equipado com um emissor luminoso VCSEL (*vertical cavity surface-emitting laser*) e uma matriz de díodos designada por SPAD (*single photon avalanche diodes*) que deteta a luz refletida e é com base neste tempo de voo da luz (ida → reflexão → volta) que o sensor consegue determinar as distâncias pretendidas. Por este último motivo, foi usada uma base preta (na área onde se encontram distribuídas as peças) para evitar reflexões indesejadas da luz, que se poderiam traduzir em deteções imprecisas das distâncias.

No *setup* deste sensor, começamos por definir os pinos associados ao sinal do relógio (pino GPIO 21) e ao sinal de dados (pino GPIO 20), especificamos a livraria associada a este sensor que é a *Wire* (como os sensores de distância e cor se encontram ligados em pinos diferentes para evitar interferências, é importante fazer esta distinção, porque o sensor de cor usa a livraria *Wire1*) e associamos um endereço I2C diferente do padrão na livraria *Wire* (numa configuração inicial tínhamos ambos os sensores ligados nos mesmos pinos de relógio e dados, o que implicaria o uso de um endereço diferente para cada sensor, porém isso provou ser um problema mais à frente como se explica na secção III, desta forma ligaram-se os sensores a pinos diferentes recorrendo para isso a duas livrarias *Wire*, pelo que o endereço poderia ser o padrão em ambos). Ainda foi definido um *Timeout* que atua como um tempo de espera para o sensor estabelecer ligação com o microcontrolador e uma condição para saber se o sensor foi conectado com sucesso ou não.

A medição em si foi efetuada recorrendo à função `tof.readRangeAvailable()` presente no início do *loop* principal e que averigua se há medições disponíveis a ser feitas e atribui à variável **prev_distance** o valor da distância efetuada no final do loop anterior e atribui à variável **distance** o valor da medição obtida pela função `tof.readRangeMillimeters()`. Estas duas variáveis são

importantes na deteção de uma peça como é explicado da secção V. No final de cada loop iniciamos uma nova medida com a função `tof.startReadRangeMillimeters()`.

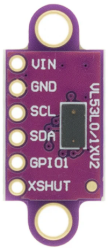

Figura 2. Sensor de distância (TOF)

III. Montagem

No que diz respeito à montagem deste projeto, é possível observar na figura 3 as ligações a fazer. Esta ligações correspondem a ligar os motores aos respetivos pinos de controlo que se encontram na parte inferior da placa, estando estes ligados na seguinte ordem: Motor de movimento da base - GPIO 0; Motor de controlo da extensão do braço - GPIO 2; Motor de movimento da pega/gancho - GPIO 4; Motor de controlo da altura do braço - GPIO 6. Uma vez que estes motores necessitam de cerca de 2A a 3A de corrente para trabalhar corretamente, é necessário introduzir uma fonte de tensão extra pois a corrente debitada pelos pinos da placa não é o suficiente (corrente máxima debitada de 16mA). A placa que nos permite controlar os motores tem uns pinos que permitem a ligação desta fonte de corrente extra e que está ligada eletricamente ao pino VBUS do MC.

Quanto ao sensores, a informação necessária é obtida recorrendo ao protocolo de comunicação I2C. Inicialmente tentámos utilizar a mesma linhas de comunicação I2C, uma vez que esta permite a comunicação de um ”master”com vários ”slaves”. Contudo tivemos alguns problemas pois bastava uma ligação de um dos sensores falhar que causava problemas no envio da informação em geral. De forma a resolver este problema utilizamos as duas linhas de I2C (I2C0 e I2C1) presentes no MC de forma a controlar cada sensor separadamente.

O material necessário para a montagem do braço foi fornecido pelo docente responsável por esta unidade curricular. As peças foram todas obtidas por impressão 3D em plástico. A base que serve de suporte para segurar o braço e o pequeno bloco para pousar o sensor de cor foram implementados por nós.

É necessário realçar que a posição do sensor é alta de mais para peças que se possam vir a utilizar portanto a necessidade da elevação da base onde as peças se encontram distribuidas é necessária e tal é feita com um simples caderno de capa preta.

IV. Calibração dos sensores

Antes de iniciarmos quaisquer projeção de máquina de estados fizemos as calibrações que achamos necessárias dos

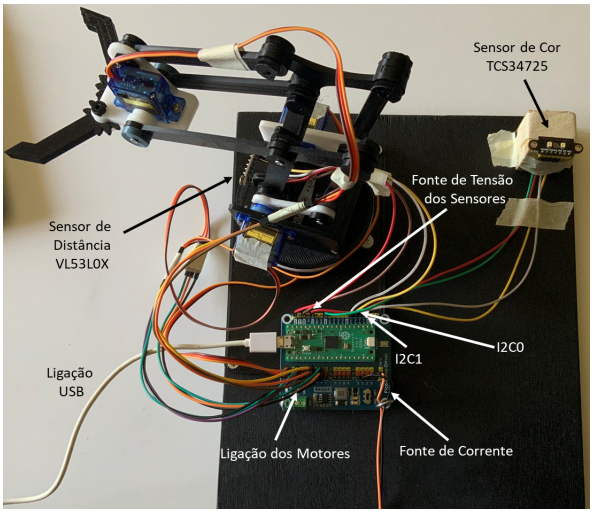

Figura 3. Esquema de ligações entre motores, sensores, microprocessador e fonte de corrente/tensão

sensores que viríamos a implementar. No caso, após termos montado o braço e os motores, através de um simples script de leitura da porta série, pudemos controlar as posições dos 4 motores presentes. Dada a natureza do objetivo final do projeto, a separação das peças é feita por 4 cores. Uma vez que 3 destas cores correspondem diretamente às variáveis de leitura do sensor de cor, bastou averiguar qual a variável com maior valor e associar a respetiva cor ao objeto. Por exemplo, colocamos uma peça LEGO azul em frente ao sensor. O maior valor das 3 variáveis de cor R (red), G (green) e B (blue) será B, e portanto se (B $>$ G e B $>$ R), concluímos que a peça é azul. A quarta cor em estudo é o amarelo, e para tal definimos um ganho do sensor (fixo ao longo do funcionamento de todo o projeto) e definimos algumas razões entre as 3 cores primárias. Daqui obtivemos que uma peça seria dita como amarela se R $>$ G, R $>$ B e G $<$ 1.5B.

Para sabermos as posições angulares dos motores de controlo de altura e extensão, fizemos uma recolha experimental que consistiu em definir o intervalo de distâncias das quais o nosso braço consegue atingir. Para tal concluiu-se o seguinte:

	Distância (cm)	Extensão (deg)	Altura (deg)
Máximo	18.5	170	115
Mínimo	8.8	110	30

Sabendo esta gama fizemos as mesmas medições para outros pontos entre estes valores limites o que nos permitiu obter 2 equações para converter a distância a que a peça está do sensor para a posição do braço de extensão, e sabendo esta posição de extensão conseguimos averiguar a altura a que este deve estar através de uma segunda equação. Os gráficos da recolha de dados e os parâmetros estão presentes em anexo. As equações que daqui resultaram foram então as seguintes:

$$\alpha = 590 \times d + 54; \quad (1)$$

$$\beta = 1.3 \times \alpha - 112; \quad (2)$$

em que α corresponde ao ângulo do motor de controlo de extensão, β é o ângulo do motor de altura (ângulos em graus) e d é a distância a que a peça se encontra.

V. Máquina de Estados

A. *Especificação dos estados*

- S0 - Estado 0: Neste estado temos o braço na sua posição inicial de movimento e damos ”attach”dos pins dos motores aqui;
- S1 - Estado 1: Estado no qual o braço faz a procura por peças através do movimento da base e detenções de distância do sensor;
- S2 - Estado 2: Calcula as posições angulares dos motores necessárias para pegar numa peça eventualmente detetada;
- S3 - Estado 3: Serve simplesmente para agarrar a peça por controlo de um motor;
- S4 - Estado 4: Levanta a peça para uma posição ”standard”de altura e extensão (iguais à do estado 0);
- S5 - Estado 5: Movimentar o braço para colocar a peça sobre o sensor de cor;
- S6 - Estado 6: Permite obter a cor da peça e guarda esta cor por alteração de valor booleano num array de 4 cores.
- S7 - Estado 7: Controla os motores para uma posição onde larga a peça de acordo com a cor detetada;
- S8 - Estado 8: Voltar às posições padrão para continuar a procura por peças;
- S9 - Estado 9: Estado IDLE, isto é, se não detetar nenhuma peça damos ”detach”dos pinos de modo a parar o sistema por completo;

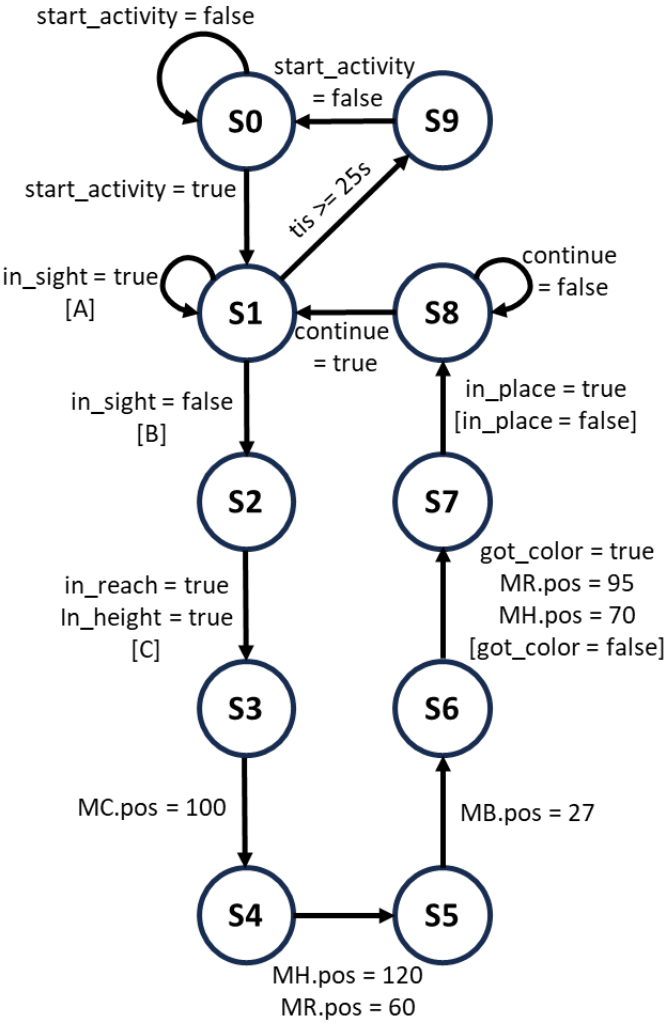

Figura 4. Máquina de estados utilizada para o funcionamento do braço robótico

B. *Evolução da máquina de estados*

A máquina de estados inicia no estado 0. Para sairmos deste estado de repouso temos de inserir na porta série o símbolo ”**+**”. Ao longo da evolução da máquina se precisarmos de voltar a este estado por necessidade basta inserir o símbolo ”**-**”. Estes comandos servem basicamente para podermos para/iniciar o funcionamento geral da máquina de estados. Portanto temos que **start_activity = true** quando pressionamos ”**+**” e **start_activity = false** quando pressionamos ”**-**”.

Sendo estabelecida esta condição passamos para o estado 1. A variável **in_sight** corresponde a um conjunto de condições que correspondem a averiguar se a distância que o sensor está a medir está dentro do intervalo definido na calibração e se duas medidas consecutivas não diferem mais do que 2cm. Deste modo é possível verificar quando encontramos uma peça. Sempre que este estado é verificado temos um conjunto de operações [A] que consistem em armazenar a medida obtida pelo sensor, a posição da base e o número da medição. Assim que **in_sight = false**, ou seja, assim que a distância medida pelo sensor estiver fora do nosso limite ou diferir de 2 cm temos que avançar para o estado 2. Nesta passagem temos um conjunto de operações [B] que correspondem à distância média a que a peça se encontra, à posição média da base e uns ajustes de offset adicionados perante testes.

No estado 2 o braço vai ter com a peça. Só avançamos para o 3 quando os motores de altura e extensão estiverem nas suas posições calculadas aquando da deteção da pela, no caso quando **in_height = true** e **in_reach = true**, respetivamente. Agora nos estado 3 fechamos a pega e colocamos a 0 algumas variáveis necessárias, o conjunto de operações [C]. Quando fechada **MC.pos = 100**, avançamos para o estado 4. Neste estado recolhemos o braço para a posição padrão definida como **MH.pos = 120** e **MR.pos = 60**. Agora no estado 5 levamos o braço para a posição do sensor em **MB.pos = 27**. Seguidamente no estado 6 averiguamos a cor da peça e colocamos **got_color = true**. Ainda neste estado também verificamos se as posições dos motores estão corretas pois a deteção deve ser feita num sítio específico dada a pequena dimensão do sensor de cor.

Verificada a cor, o braço leva a peça para uma posição predefinida no estado 7. Quando a peça for largada temos que **in_place = true** o que nos leva a passar para o estado 8. Nesta passagem precisamos de alterar para falso o valor desta variável. Estando a agora no estado 8, o braço volta à sua posição padrão e quando a atingir com **continue = true**, este irá continuar a sua procura por peça naturalmente.

Um efeito de IDLE foi adicionado. Este acontece se nenhuma peça for encontrada por algum tempo. Isto é, passado 25 segundos no estado 1, se nenhuma peça for detetada vamos para o estado 9 onde damos ”detach”dos pinos dos motores de modo a reduzir consumo energético caso o braço seja esquecido em modo de pesquisa. Para recomeçar basta enviar para a porta série os símbolos mencionados na explicação da iniciação do braço.

VI. Estrutura do Código

Começamos por importar todas as livrarias necessárias, criar as variáveis globais, as variáveis necessárias ao funcionamento dos sensores de cor e distância, as variáveis associadas à máquina de estados e, para concluir, criamos ainda variáveis associadas a cada servo.

O corpo principal do código é composto pelas seguintes funções:

- `calc_next_state()` → Calcula o próximo estado da nossa máquina de estados tendo em conta o estado atual da mesma;
- `update_state()` → Atualiza o estado atual da máquina de estados;
- `calc_outputs()` → Atribui os valores às variáveis representativas dos ângulos dos servos segundo as posições idealizadas em cada estado;
- `update_outputs()` → Dá *write* dos valores dos ângulos nos servos, isto é, transmite aos servos as posições para as quais queremos que eles se movam;
- `setup()` → Define todas as condições iniciais do sistema tais como os pinos dos motores, a largura do pulso elétrico em micro-segundos que codifica a posição do motor, inicializa a máquina num estado, define-se um intervalo de 40ms entre cada conjunto de operações da função `loop()` e define-se também os pinos dos sensores bem como as sua inicializações;
- `loop()` → Chama todas as funções anteriores seguindo a ordem: `calc_next_state()` → `update_state()` → `calc_outputs()` → `update_outputs()` (**esta ordem é importante**, pois não queremos atualizar os estados sem primeiro saber para qual estado atualizar e o mesmo se aplica aos *outputs*). Nesta função também está definida uma condição para inicializar o braço apenas quando se premir o botão ”**+**”e parar quando se prime o botão ”**-**”, assim como o código necessário aos sensores de cor e distância para efetuar medições e, por último, os `Serial.print` que nos permitem ter uma visão geral do que está a acontecer.

De forma a facilitar a leitura do código criamos uma estrutura de dados para os motores, isto é, a cada motor temos associada a variável de posição e direção. Esta segunda é apenas útil para o movimento da base. Os motores são definidos no código como:

- **MB** → Motor da posição base;
- **MR** → Motor de controlo de extensão ou ”reach”;
- **MH** → Motor de controlo de altura ou ”height”
- **MC** → Motro de controlo da garra ou ”catching motor”;

É possível ver as bibliotecas usadas, a folha Excel das calibrações e o código do projeto no seguinte link: https://github.com/LuisSousa3/ACE_RoboticArm.git.

VII. Demonstração e Trabalho Futuro

Aqui está presente um link do Youtube para uma gravação acelerada do projeto a funcionar: https://www.youtube.com/watch?v=zKAURG0bCNk. Concluímos aqui que o braço estava a trabalhar relativamente bem e como o esperado. Contudo algumas melhorias poderiam ter sido feitas.

No que diz respeito à estrutura do braço, as peças da impressão 3D tinham um defeito que deixavam a pega um pouco inclinada e algumas delas eram mais finas do que o esperado. Em geral isto causou imensas folgas na montagem o que levou a alguns problemas nos movimentos do braço. Quanto aos motores, verificou-se que muitas vezes estes não tinham torque suficiente para fazer mexer o braço, portanto uma melhoria seria a alteração dos motores por uns de maior qualidade. É de notar ainda que o transformador usado (que nos permitiu trabalhar no projeto em casa) era de 1.5A, o que não é suficiente para todos os motores. Portanto para melhor funcionamento e para medições mais precisas na calibração esta deve ser feita em boas condições de corrente para garantir que os motores funcionam corretamente (gerador de tensão do laboratório).

APÊNDICE

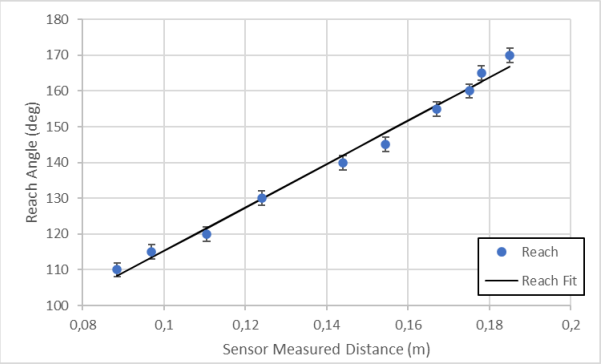

Figura 5. Gráfico de pontos recolhidos para ajuste de ângulo de extensão em função da distância detetada

m	608	54	**b**
u(m)	21	3	**u(b)**
r^2	0,99	2	**u(y)**

Figura 6. Parâmetros de ajuste para a equação 1

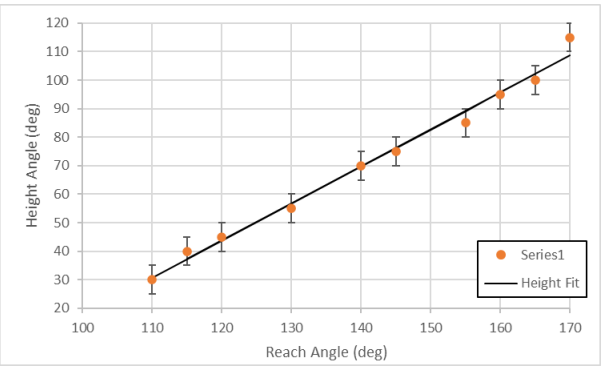

Figura 7. Gráfico de pontos recolhidos para ajuste de altura do braço em função do ângulo de extensão

m	1,30	-112	**b**
u(m)	0,05	7	**u(b)**
r^2	0,99	3	**u(y)**

Figura 8. Parâmetros de ajuste para a equação 2

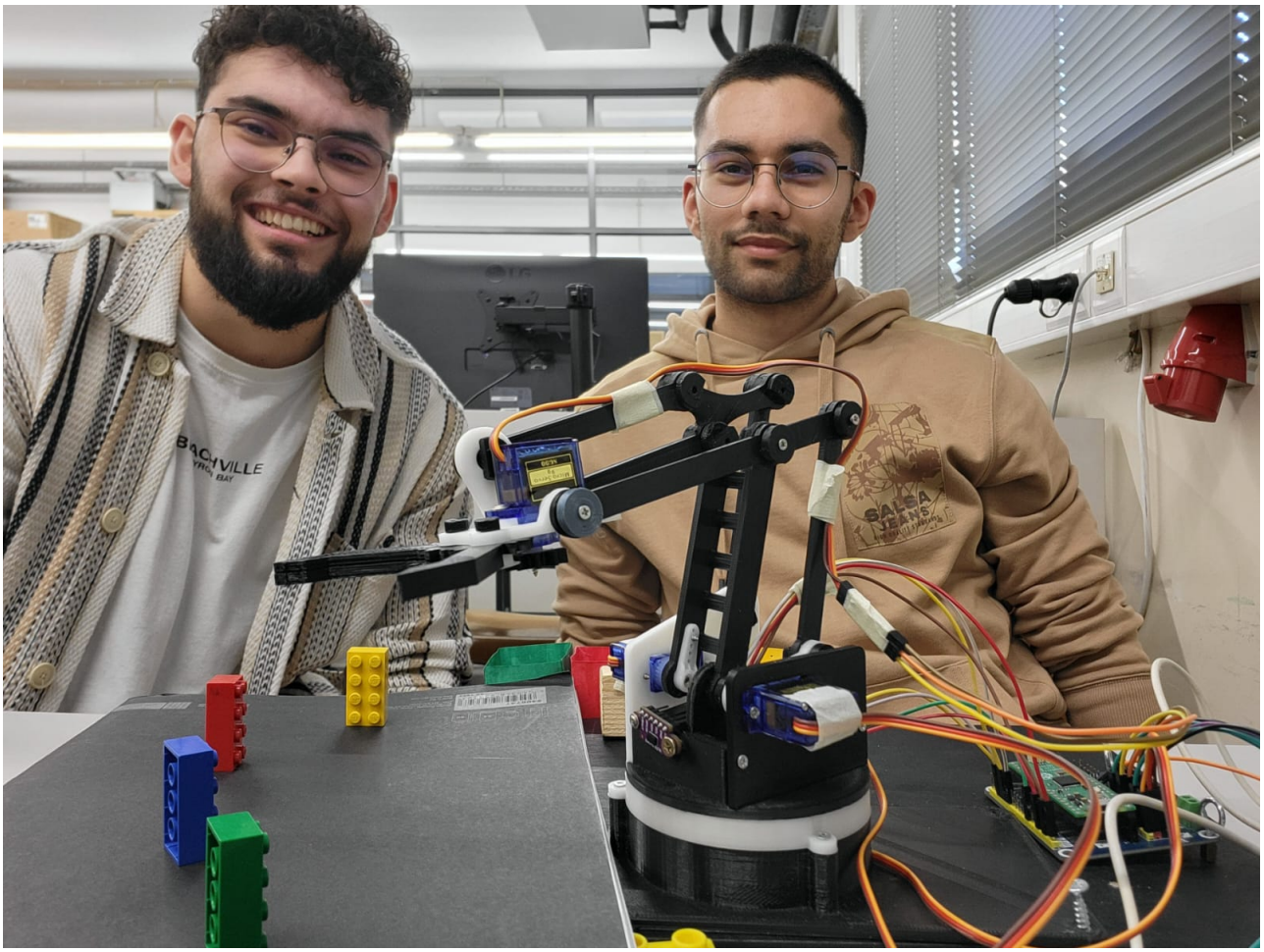

Figura 9. Fotografia de grupo (José Chaves na esquerda e Luís Sousa na direita)