

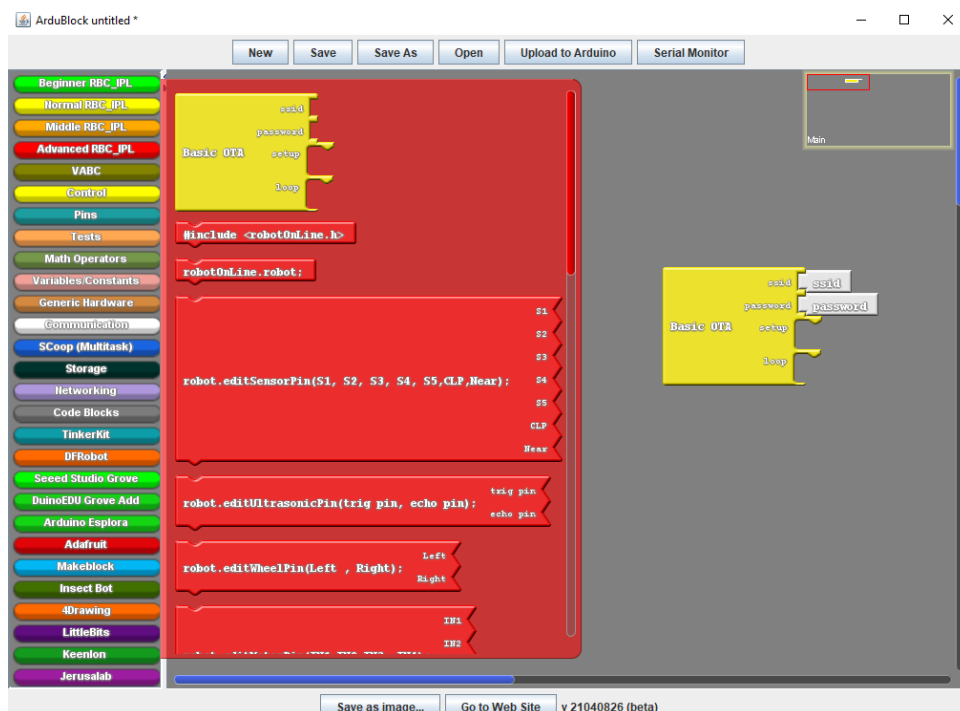


INSTITUTO DE ENGENHARIA DE
SISTEMAS E COMPUTADORES
(INESC) DE COIMBRA



Tutorial – Nível Avançado

Programar por blocos



Alunos:

Abel Teixeira - 2180522

Samuel Lourenço - 2180356

Docente: Luís Conde

Carlos Neves

Ano letivo: 2019/2020

Índice

Nível Avançado (Adanced)	2
1. Blocos do Setup	3
1.1. Bloco “#include <robotOnLine.h	3
1.2. Bloco “robot.ediSensorPin(S1, S2, S3, S4, S5, CLP, Near);”	3
1.3. Bloco “robot.editUltrasonicPin(trig pin, echo pin);”	3
1.4. Bloco “robot.editWheelPin(Left , Right);”	3
1.5. Bloco “robot.editMotorPin(INT1, INT2, INT3, INT4);”	4
1.6. Bloco “robot.begin();”	4
2. Blocos dos loop	4
2.1. Blocos “robot.getLeftEncoderCount();” e “robot.getRightEncoderCount();”	4
2.2. Bloco “robot.clearEncoderCount();”	4
2.3. Bloco “robot.distance();”	4
2.4. Blocos “robot.forward();” e “robot.reverse();”	4
2.5. Bloco “robot.leftWheel();” e “robot.rightWheel();”	5
2.6. Blocos “robot.rotateLeft();” e “robot.rotateRight();”	5
2.9. Bloco “robot.setSpeeds(Fast, Average, Slow)”	5
2.10. Bloco “robot.steerLeft(byte);” e “robot.steerRight(byte);”	6
2.11. Blocos da condução autónoma	6
2.11.1. Bloco “robot.autoDrive(byte);”	6
2.11.2. Bloco “robot.beginAutoDrive();”	6
2.11.3. Bloco “robot.readCLP();”	7
2.11.4. Bloco “robot.readNear();”	7
2.11.5. Bloco “robot.readS1();”	7
2.11.6. Bloco “robot.readS2();”	7
2.11.7. Bloco “robot.reaS3();”	7
2.11.8. Bloco “robot.readS4();”	8
2.11.9. Bloco “robot.readS5();”	8
2.11.10. Bloco “robot.disableCLP();”	8
2.11.11. Bloco “robot.disableNear();”	8
2.11.12. Bloco “robot.disableUltrasonic();”	8
2.11.13. Bloco “robot.endAutoDrive();”	8
2.11.14. Bloco “robot.noLineDelay(uint);”	9
2.12. Funções também muito utilizadas na programação	9

Nível Avançado (Adanced)

No nível avançado já é como se estivesse a programar. Os blocos vermelhos não necessitam de biblioteca são funções que se usam na linguagem C++ e os blocos roxos são da biblioteca robot, esta biblioteca facilitara a programação do robô.

1. Blocos do Setup

1.1. Bloco “#include <robotOnLine.h

Este bloco chama a biblioteca robotOnLine.h que é necessário para que os outros blocos vermelhos funcionem e também define o nome robot para os vários blocos vermelhos que iniciam com o nome “robot.”



```
#include <robotOnLine.h>
```

1.2. Bloco “robot.editSensorPin(S1, S2, S3, S4, S5, CLP, Near);”

Este bloco define os pinos da placa IR 74HC14. Os pinos S1, S2, S3 e S4 são dos pinos dos sensores de infravermelhos, o pino CLP corresponde ao fim-de-curso da placa e o Near corresponde ao sensor infravermelho que esta a frente da placa.



1.3. Bloco “robot.editUltrasonicPin(trig pin, echo pin);”

Este bloco define os pinos que estão ligados ao sensor ultrassom (HC-SR04).



1.4. Bloco “robot.editWheelPin(Left , Right);”

Este bloco define os pinos que estão ligados aos encodares. Left ao encodar da roda esquerda e Right ao encodar da roda direita.



1.5. Bloco “robot.editMotorPin(INT1, INT2, INT3, INT4);”

Este bloco define os pinos que estão ligados no modulo (L293D) que controla os motores que estão acopladas as rodas do robô.



1.6. Bloco “robot.begin();”

Este bloco é necessário para iniciar a biblioteca e a configuração dos pinos.



2. Blocos dos loop

2.1. Blocos “robot.getLeftEncoderCount();” e “robot.getRightEncoderCount();”

O bloco “robot.getLeftEncoderCount();” corresponde ao encoder do lado esquerdo e o “robot.getRightEncoderCount();” corresponde ao encoder do lado direito. Guardando o valor na variável com o nome que colocar no bloco branco.



2.2. Bloco “robot.clearEncoderCount();”

Este bloco serve para colocar os valores guardados pelos encodares das duas rodas a zero.



2.3. Bloco “robot.distance();”

Este bloco verifica a distância e guardada na variável com o nome que colocar no bloco branco.



2.4. Blocos “robot.forward();” e “robot.reverse();”

O bloco “robot.forward();” faz com que o robô ande para frente e o bloco “robot.reverse();” faz com que o robô recue. Há frente de cada bloco tem de escolher a velocidade que vai andar de 0 a 255



2.5. Bloco “robot.leftWheel();” e “robot.rightWheel();”

Estes blocos controlam as rodas. O bloco “robot.leftWheel();” controla a roda esquerda e o bloco “robot.rightWheel();” controla a direita. A frente de cada bloco coloca a velocidade de -255 a 255, sendo que se for um número negativo a roda respetiva recua e se for positiva ele avança.



2.6. Blocos “robot.rotateLeft();” e “robot.rotateRight();”

O bloco “robot.rotateLeft();” faz rodar para a esquerda a velocidade determinada pelo Duty-Cycle (Modelação de Largura de Pulso) inserido. O bloco “robot.rotateRight();” faz rodar para a direita a velocidade determinada pelo Duty-Cycle inserido.



2.7. Blocos “robot.turnLeft(degrees);” e “robot.turnRight(degrees);”

O bloco “robot.turnLeft();” faz rodar o robô para a esquerda a os graus inserido. O bloco “robot.turnRight();” faz rodar o robô para a direita a os graus inserido.



2.8. Bloco “robot.stopMotors();”

Este bloco faz com que o robô pare ambos os motores.



2.9. Bloco “robot.setSpeeds(Fast, Average, Slow)”

Este bloco serve para definir a velocidade máxima, media e mínima entre 255 a 1 que o robô vai andar.



2.10. Bloco “robot.steerLeft(byte);” e “robot.steerRight(byte);”

O bloco “robot.steerLeft();” fazer com que o robô ande para a esquerda e o bloco “robot.steerRight ();” fazer com que o robô **vire ligeiramente** para a direita. Há frente de cada bloco tem de escolher a velocidade que vai andar de 0 a 255.



2.11. Blocos da condução autónoma

2.11.1. Bloco “robot.autoDrive(byte);”

Este bloco deve ser usado apenas se quiser a função de condução automática proporcionada pela biblioteca. Este bloco deve ser usado com o bloco “robot.beginAutoDrive();”. Este bloco recebe e devolve valores, os valores devolvidos significam o seguinte:

- 0 - nada a reportar;
- 1 - múltiplas linhas encontradas;
- 2 - obstáculo encontrado;
- 3 – não foram encontradas linhas

Os valores que podem ser enviados são:

- 1- Rodar para a direita;
- 2 – Rodar para a esquerda;
- 3 – Seguir em frete;
- 4 – Retroceder.



2.11.2. Bloco “robot.beginAutoDrive();”

Este bloco serve para iniciar a condução automática, apenas deve ser usado caso queira usar a função de condução automática proporcionada pela biblioteca.
Este bloco deve ser usado com o bloco “robot. autoDrive(byte);”.



robot.beginAutoDrive();

2.11.3. Bloco “robot.readCLP();”

Este bloco faz a leitura do fim de curso CLP da placa IR 74HC14. Devolvendo 0 ou 1.



robot.readCLP(); Variable_Name None

2.11.4. Bloco “robot.readNear();”

Este bloco faz a leitura do infravermelho que se localiza á frente do robô da placa IR 74HC14. Devolvendo 0 ou 1.



robot.readNear(); Variable_Name None

2.11.5. Bloco “robot.readS1();”

Este bloco faz a leitura do sensor mais à direita da placa IR 74HC14. Devolvendo 0 ou 1.



robot.readS1(); Variable_Name None

2.11.6. Bloco “robot.readS2();”

Este bloco faz a leitura do sensor à direita da placa IR 74HC14. Devolvendo 0 ou 1.



robot.readS2(); Variable_Name None

2.11.7. Bloco “robot.readS3();”

Este bloco faz a leitura do sensor meio da placa IR 74HC14. Devolvendo 0 ou 1.



2.11.8. Bloco “robot.readS4();”

Este bloco faz a leitura do sensor à esquerda da placa IR 74HC14. Devolvendo 0 ou 1.



2.11.9. Bloco “robot.readS5();”

Este bloco faz a leitura do sensor mais a esquerda da placa IR 74HC14. Devolvendo 0 ou 1.



2.11.10. Bloco “robot.disableCLP();”

Este bloco desabilita a leitura do sensor CLP da placa IR 74HC14 caso esteja a usar a função de condução automática proporcionada pela biblioteca.



2.11.11. Bloco “robot.disableNear();”

Este bloco desabilita a leitura do sensor Near da placa IR 74HC14 caso esteja a usar a função de condução automática proporcionada pela biblioteca.



2.11.12. Bloco “robot.disableUltrasonic();”

Este bloco desabilita a leitura do sensor ultrassons HC-SR04 caso esteja a usar a função de condução automática proporcionada pela biblioteca.



2.11.13. Bloco “robot.endAutoDrive();”

Este bloco destinado à funcionalidade da condução autónoma e necessita de ser chamada uma vez para desassociar as interrupções previamente configuradas.

`robot.endAutoDrive();`


2.11.14. Bloco “robot.noLineDelay(uint);”

Este bloco serve para especificar o tempo (em milissegundos) que o robô aguarda para parar os motores após ter identificado que nenhum dos sensores detetou uma linha.

`robot.noLineDelay(uint);` unit 0

2.12. Funções também muito utilizadas na programação

Pode investigar as várias abas e descobrir novas funções que possa adaptar no seu programa. Quando se sentir já preparado passa para o Arduino IDE sem o Ardublok e comece a programar linha a linha.



Arrows point from the following blocks in the sidebar to their respective reference links:

- if** → <https://www.arduino.cc/reference/pt/language/structure/control-structure/if/>
- if/else** → <https://www.arduino.cc/reference/en/language/structure/control-structure/else/>
- while** → <https://www.arduino.cc/reference/en/language/structure/control-structure/while/>
- do while** → <https://www.arduino.cc/reference/en/language/structure/control-structure/dowhile/>

3. Exemplo de um programa

3.1. Robô vai andar em frente

No programa em baixo no Setup:

Bloco “#include <robotOnLine.h>” chama a biblioteca;

Bloco “robotOnLine.robot;” define o nome do bloco como robot;

Bloco “robot.editMotorPin(IN1, IN2, IN3, IN4);” de define os pinos que vão controlar as rodas;

Bloco “robot.begin();” configura a biblioteca e os pinos;

No loop:

Bloco “robot.forward();” faz com que o robô ande para a frente.

