

WORKSHOP / ACD

LED - Laboratórios de Educação Digital¹ EXPLORAÇÃO DE RECURSOS LED

Tema: Utilização e programação da plataforma **Arduino Uno**

Destinatários: Professores do ensino básico e secundário (área de Física e Química)

Objetivos:

- Entender o conceito de microcontrolador;
- Realizar a sua programação utilizando a linguagem gráfica **Snap!**;
- Integrar a plataforma Arduino em ensaios laboratoriais específicos da área de *Física e Química*.

Material: Computador portátil ou equivalente, com sistema operativo Windows (10 ou superior) e plataforma de desenvolvimento **Arduino Uno** e kit com alguns sensores.

Repositório:

https://github.com/jpcoelhoATipbDOTpt/LED_Laboratorios_de_Educacao_Digital.git



Programa:

Parte 1:	Instalação do Software.....	3
1.1 -	Descrição:.....	3
1.2 -	Acerca do Scratch.....	3
1.3 -	Snap!	3
1.4 -	Snap4Arduino.....	4
1.5 -	Procedimento de Instalação.....	4
Parte 2:	O microcontrolador e a plataforma Arduino Uno	11

¹ Recurso formativo da autoria de [João Paulo Coelho](#), disponibilizado no contexto do workshop.

2.1 -	Um bicho de 7 cabeças...	11
2.2 -	Arduino Vs. Microcontrolador	12
2.3 -	Caraterísticas da plataforma Arduino Uno	13
2.4 -	Ligação do Arduino ao computador	14
2.5 -	Portos de I/O	15
Parte 3:	Ligando o Arduino Uno ao Snap4Arduino	19
3.1 -	Preparação do Arduino Uno (aconselhado).....	19
3.2 -	Preparação do Arduino Uno (alternativo)	21
3.3 -	Interligação do Arduino Uno e do Snap4Arduino	27
Parte 4:	Programando o Arduino Uno com o Snap4Arduino	30
4.1 -	Os primeiros <i>sketches</i>	30
4.2 -	Variáveis, ciclos e condições.....	36
4.3 -	Controlando o ator e o palco usando Arduino Uno	39
Parte 5:	Interface com o mundo físico usando o Arduino Uno	47
5.1 -	Medição da iluminância	47
5.2 -	Medição da temperatura do ar	49
Parte 6:	Conclusão	52

Parte 1: Instalação do Software

1.1 - Descrição:

Nesta primeira parte pretende-se instalar o software que será utilizado para programar a plataforma **Arduino Uno**. Dada a audiência a que este curso se destina, pretende-se evitar a utilização de linguagens de programação baseadas em texto estruturado como é o caso do C/C++ e, em vez disso, recorrer à linguagem gráfica **Snap!** que deriva do **Scratch**. A escolha desta estratégia de programação tem uma segunda vantagem que se prende com o facto desta já ser utilizada nos *currículos* de algumas disciplinas do ensino básico/secundário o que poderá facilitar a sua utilização, *a posteriori*, pelos próprios alunos.

1.2 - Acerca do Scratch

O Scratch é uma linguagem de programação visual desenvolvida pelo *Lifelong Kindergarten Group* do **MIT Media Lab** tendo sido desenhada para apresentar conceitos de programação de uma forma divertida e interativa. Em particular, o **Scratch** serve frequentemente como ferramenta para promover e simplificar a entrada no mundo da programação e ensinar pensamento lógico e computacional.



Uma das particularidades do **Scratch** assenta na ideia de se criarem programas informáticos baseados na interligação de **blocos**. Em particular, o **Scratch** recorre a uma forma de programação gráfica que envolve arrastar e empilhar blocos com funções distintas para criar *scripts*.

Cada bloco representa um conceito ou ação de programação, tais como ciclos, condições e criação de variáveis, que se encaixam para formar instruções. O **Scratch** é principalmente baseado na Web, podendo ser acedido através do link² <https://scratch.mit.edu/>.

1.3 - Snap!



O **Snap!** e o **Scratch** compartilham muitas semelhanças dado que o primeiro deriva do segundo. Ambas são linguagens de programação visuais desenhadas para apresentar conceitos de programação a iniciante. No entanto, o **Snap!** possui recursos e capacidades mais avançados permitindo criar blocos e extensões personalizadas.

² É necessária a criação de uma conta (gratuita) para ter acesso à plataforma on-line.

Tal como o **Scratch**, o **Snap!** é disponibilizado via web browser e acessível através do URL:

<https://snap.berkeley.edu/snap/snap.html>

1.4 - Snap4Arduino

O **Snap4Arduino**³ é uma extensão do **Snap!** adaptado especificamente para programar placas de desenvolvimento **Arduino**. Tal como as linguagens referidas anteriormente, permite criar programas usando uma linguagem gráfica que envolve a interligação de blocos com funções primitivas. Possui blocos específicos para interagir com os portos I/O das plataformas **Arduino** permitindo, deste modo, o desenvolvimento **scripts** capazes de controlar sensores, atuadores e outros componentes conetados à placa. Através da interface gráfica, blocos podem ser facilmente inseridos no ambiente de trabalho facilitando a criação de programas complexos sem a necessidade de escrever código tradicional.



Para mais detalhes e informação sobre esta extensão sugere-se a visita à página de suporte do **Snap4Arduino** localizada no endereço:

<https://snap4arduino.rocks/>.

Para este curso, o **Snap4Arduino** será a base utilizada para a programação da plataforma **Arduino Uno** e, para este fim, deverá ser instalado localmente. Para isso, é possível aceder ao repositório GIT deste projeto cujo endereço é:

<https://github.com/bromagosa/Snap4Arduino/releases>

À data, a versão mais atual é a **9.1.1** que será a versão utilizada no decorrer deste curso.

Assim, a primeira parte deste curso envolve a instalação deste ambiente de desenvolvimento. Na seção que se segue apresentam-se a etapas necessárias à sua instalação.

1.5 - Procedimento de Instalação

Antes de mais, e para suporte a este curso, foi criado um repositório onde é possível encontrar toda a documentação e software a ser utilizado. Este repositório pode ser acedido via o endereço:

https://github.com/jpcoelhoATipbDOTpt/LED_Laboratorios_de_Educacao_Digital

³ **Snap4Arduino** tem vindo a ser desenvolvido e mantido por *Joan Guillén* e *Bernat Romagosa*.

Apresentam-se os seguintes passos a executar a fim de instalar o software:

Passo 1: *Clicar* no link apresentado anteriormente e aceder ao repositório.

Passo 2: *Clicar* na pasta com nome “**software**”

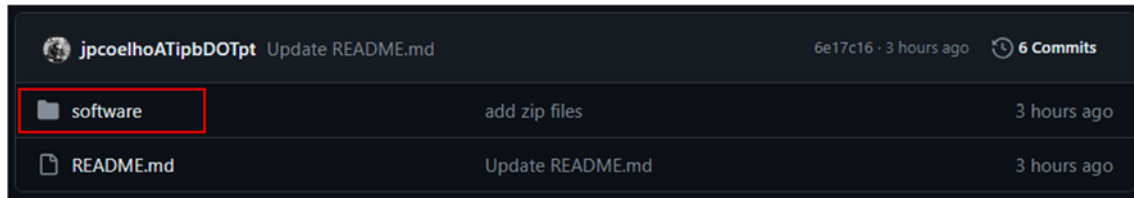


Figura 1 - Pasta "software" existente no repositório do curso.

Passo 3: Na área **software** encontram-se duas versões do software destinadas a duas versões possíveis de sistemas operativos: uma de **32 bits** e outra de **64 bits**. Clique naquela que se adapta ao sistema operativo instalado no computador.

Como ver qual o meu tipo de sistema operativo?

Windows 10 and Windows 8.1

1. Select the **Start** button, then select **Settings** > **System** > **About**.

[Open About settings](#)

2. At the right, under **Device specifications**, see **System type**.

Windows 7

1. Select the **Start** button, right-click **Computer**, and then select **Properties**.
2. Under **System**, see the system type.

Figura 2 - Adaptado da informação disponibilizada [neste](#) site.

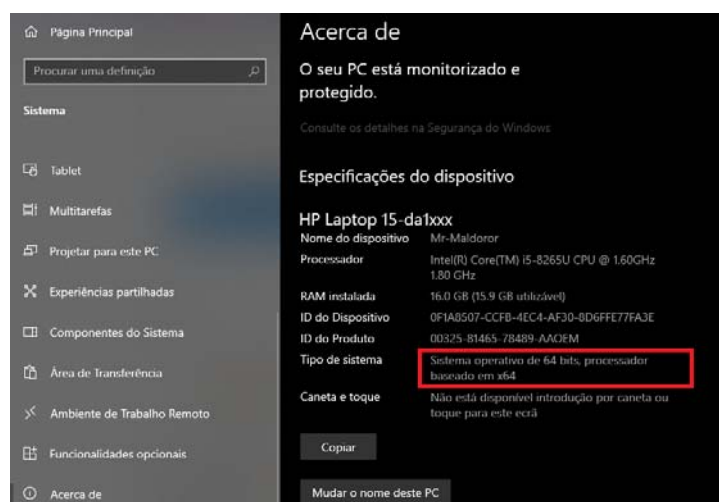


Figura 3 - Exemplo do resultado da execução dos passos descritos na figura anterior (Windows 10).

Name	Last commit message	Last commit date
..		
README.md	New Files	3 hours ago
Snap4Arduino_desktop-win-installer-32_9.1.1...	add zip files	3 hours ago
Snap4Arduino_desktop-win-installer-64_9.1.1...	add zip files	3 hours ago
Snap4Arduino_desktop-win-installer-64_9.1.1.zip		
README.md		

Figura 4 - Conteúdo da pasta de Software.

Passo 4: Clicar no ícone de **download** (ao lado de RAW).

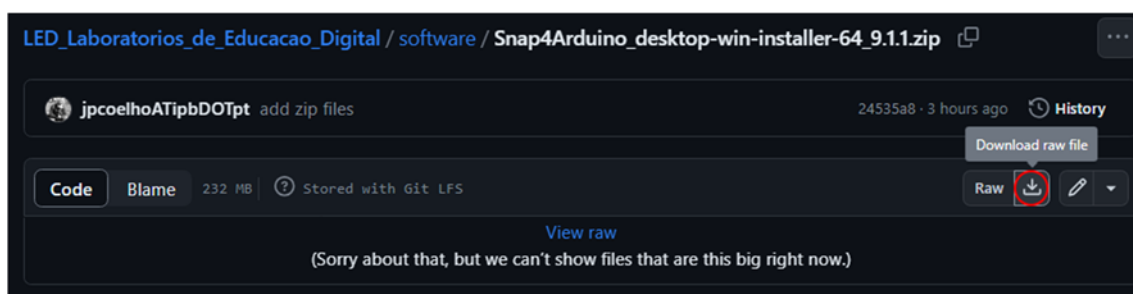


Figura 5 - Procedimento para download do software a partir do repositório.

Neste momento, na pasta de transferências local do computador, deverá existir uma pasta zipada como se mostra na Figura 6.

Nome	Data de modificação	Tipo	Tamanho
Hoje (2)			
Snap4Arduino_desktop-win-installer-64_9.1.1.zip	27/12/2023 19:52	Pasta comprimida...	237 703 KB
Última semana (3)			
19/12/2023 13:21	19/12/2023 13:21	Documento do A...	1 233 KB
19/12/2023 13:21	19/12/2023 13:21	Documento do A...	1 206 KB
19/12/2023 13:20	19/12/2023 13:20	Documento do A...	794 KB
Anteriormente neste mês (3)			
02/12/2023 12:20	02/12/2023 12:20	rar	398 KB
02/12/2023 12:19	02/12/2023 12:19	Ficheiro BDV	860 KB
02/12/2023 12:20	02/12/2023 12:20	Pasta de ficheiros	
Último mês (2)			
20/11/2023 14:03	20/11/2023 14:03	Documento do Mi...	591 KB

Figura 6 - Resultado do processo de descarga do software a partir do repositório do curso.

Passo 5: Descomprimir pasta. Clicando com o botão direito do rato sobre o nome do ficheiro selecione a opção “Extrair Todos...”. No caso de existir algum software instalado localmente para lidar com arquivos comprimidos, pode usar-se esse software para descomprimir a pasta. No entanto, por defeito, as novas versões do Windows⁴ já permitem lidar diretamente com arquivos comprimidos do tipo ZIP.

⁴ O Windows tem a capacidade de lidar com arquivos ZIP nativamente desde o Windows XP SP 2 (2004)

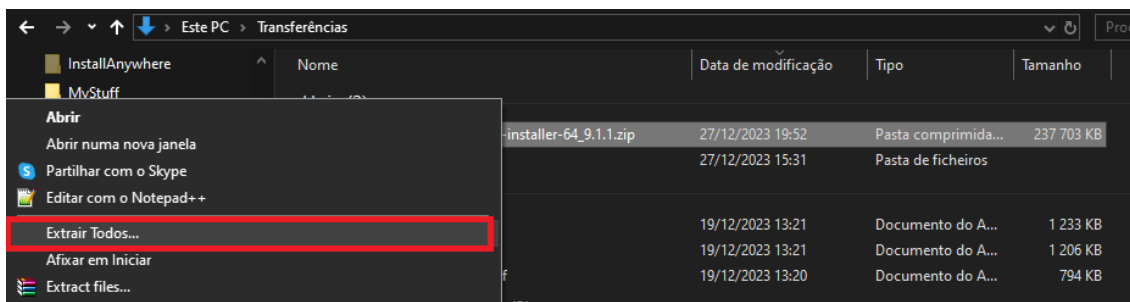


Figura 7 - Descomprimir ficheiro ZIP descarregado a partir do repositório.

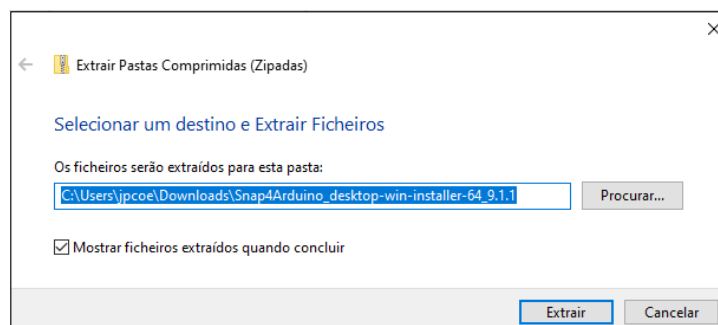


Figura 8 - Caminho (localização) onde o ficheiro será descomprimido. Pode deixar-se o caminho que aparece por defeito.

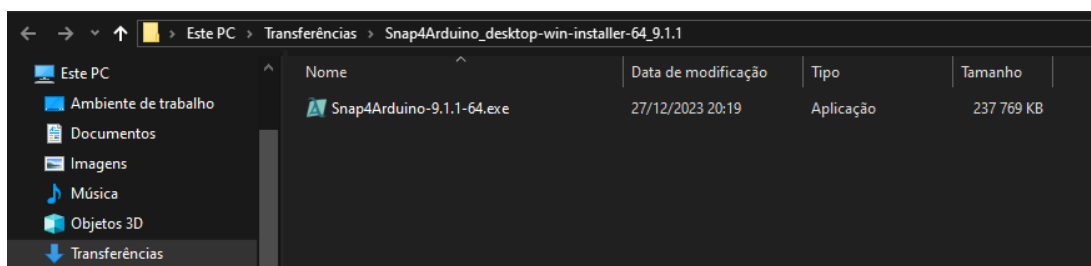


Figura 9 - Resultado final do processo de descompressão do ficheiro descarregado a partir do repositório.

Passo 6: Execução do programa de instalação. Fazendo duplo clique sobre o nome do ficheiro (Snap4Arduino-9.1.1-64.exe), seguir as instruções.

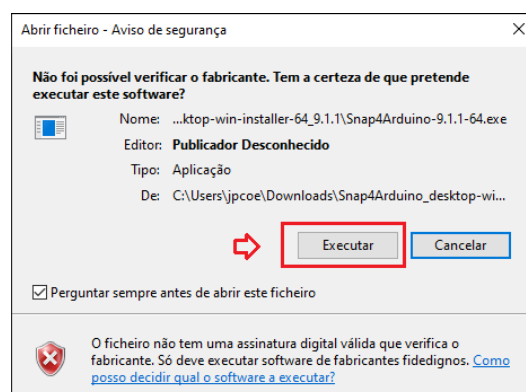


Figura 10 - Caso apareça esta alerta, clicar em "Executar".

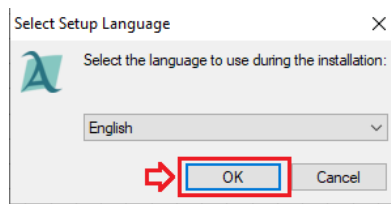


Figura 11 - Escolha do idioma utilizado no processo de instalação.

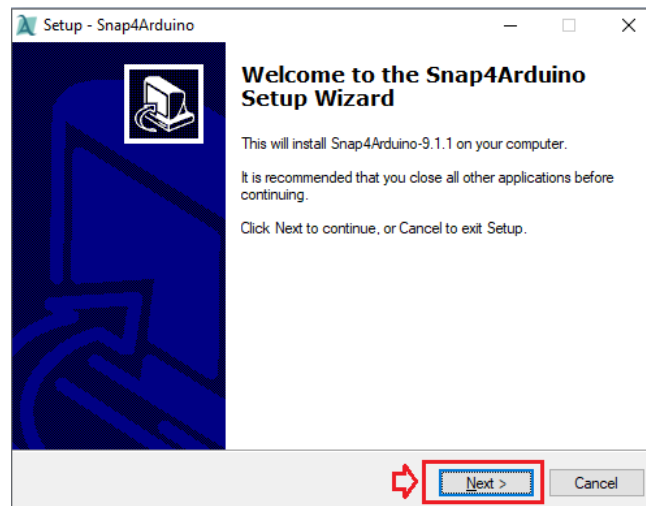


Figura 12 - Janela informativa. Clicar em "Next".

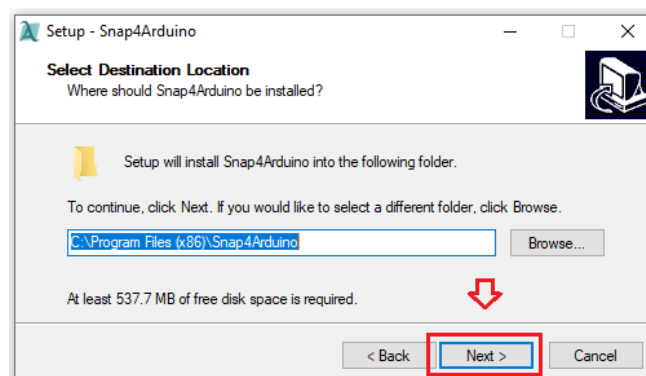


Figura 13 - Localização da pasta onde o software será instalado. Manter o que está por defeito e clicar em "Next".

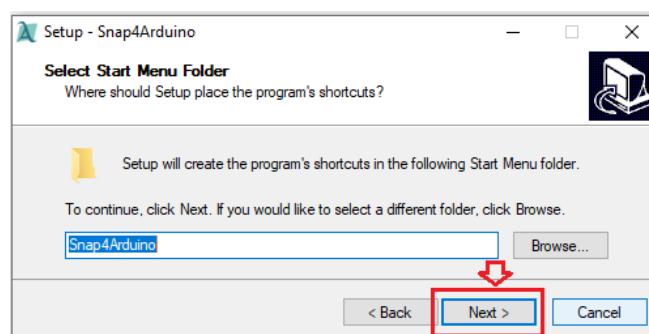


Figura 14 - Nome do atalho associado ao programa. Manter o que está por defeito e clicar em "Next".

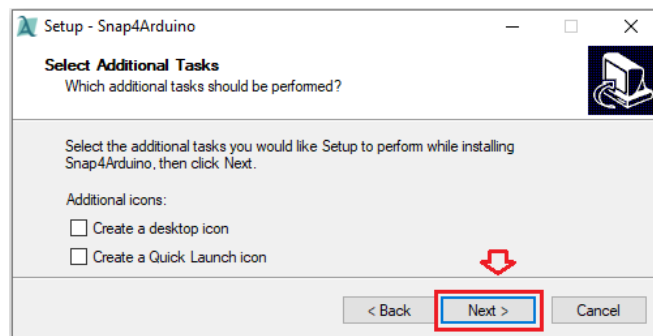


Figura 15 - Caso pretenda criar um atalho no ambiente de trabalho ou na barra de tarefas, selecione a check box apropriada. Manter o que está por defeito e clicar em "Next".

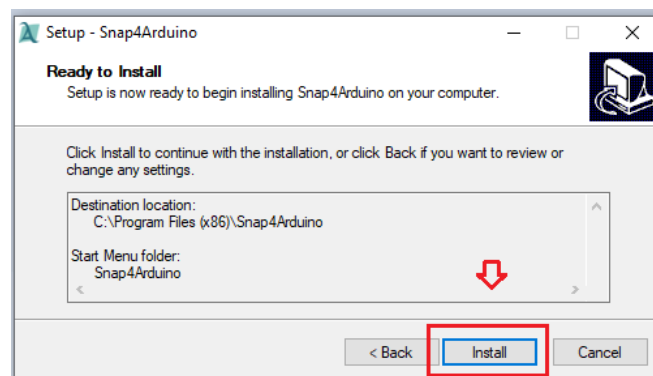


Figura 16 - Janela informativa. Clicar em "Next".

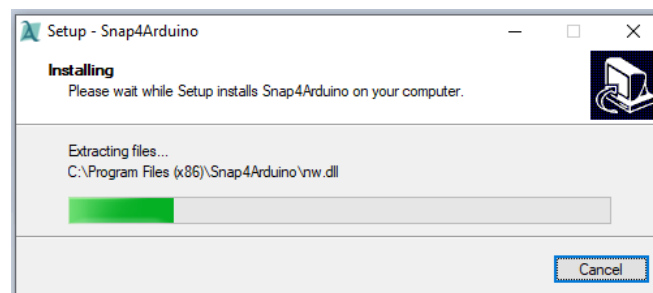


Figura 17 - Processo de instalação. Poderá demorar mais ou menos tempo dependendo do tipo de configurações do computador. Aguardar que esteja concluído o processo de instalação.

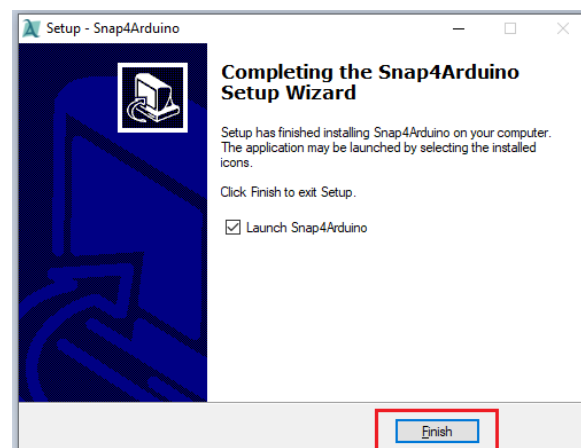


Figura 18 - Processo de instalação concluído. Clicar em "Next".

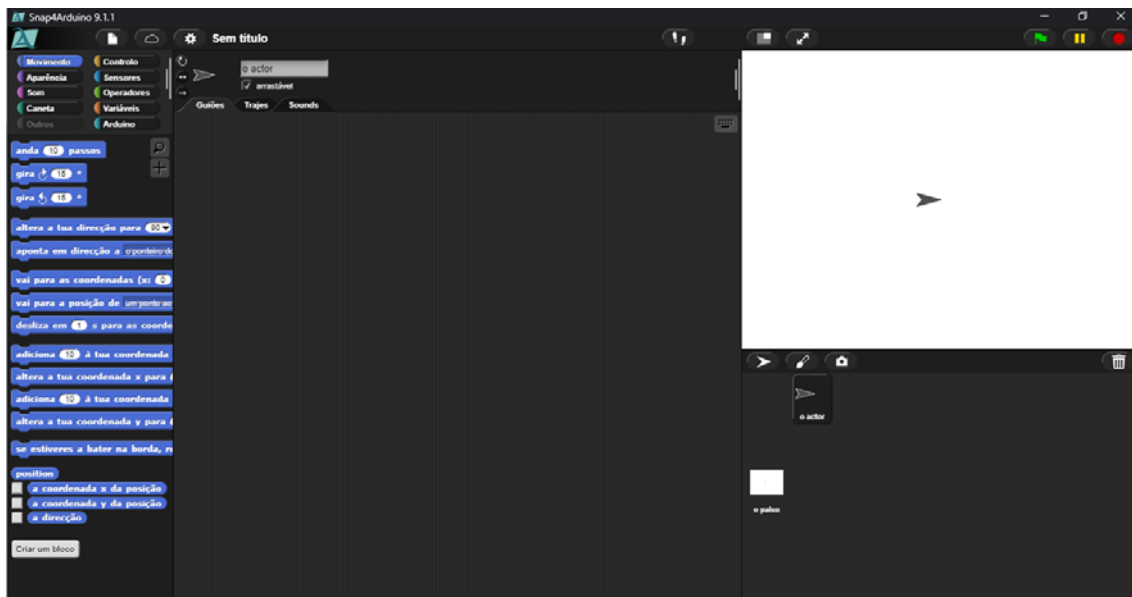
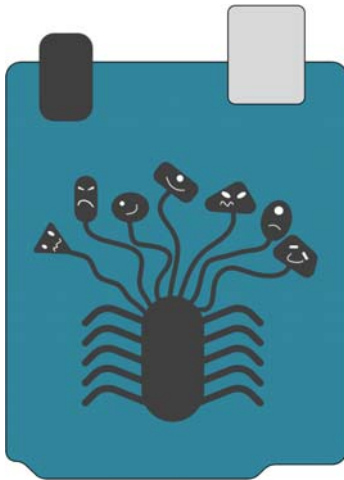


Figura 19 - Este deverá ser o aspeto da etapa final do processo de instalação. O Snap4Arduino é aberto e apresenta a sua interface gráfica.

Com isto se conclui o processo de instalação do *software Snap4Arduino* que servirá para programar a plataforma **Arduino Uno** usando a linguagem gráfica **Scratch**. Na próxima parte, apresenta-se a plataforma **Arduino Uno** e realiza-se o primeiro programa usando o *software* que acabou de ser instalado. □

Parte 2: O microcontrolador e a plataforma Arduino Uno

2.1 - Um bicho de 7 cabeças...



Quando se fala em **Arduino**, fala-se em todo um ecossistema que envolve componentes de hardware e software. A origem do projeto remonta a 2005 onde um grupo de desenvolvedores italianos tomou como objetivo criar toda uma plataforma de desenvolvimento, destinada a pessoas sem formação em engenharia, para criação de projetos eletrônicos.

Atualmente, o projeto **Arduino** conta com diversas plataformas de hardware sendo que a **Uno** é a mais simples e mais amigável para iniciantes. A Figura 20 mostra o aspecto físico da plataforma **Arduino Uno**. Cada uma das diferentes opções de *hardware* existentes atualmente possui seu próprio conjunto de recursos e valências. Para mais informação acerca das soluções atualmente disponíveis pode consultar-se o website do projeto em <https://www.arduino.cc/en/hardware>.



Figura 20 - Aspecto da plataforma Arduino Uno.

Uma das características mais interessantes é a possibilidade de estender a capacidade das plataformas **Arduino Uno** através da adição de outros blocos de *hardware* designados por **shields**. Por exemplo, existem **shields** capazes de adicionar funcionalidades como WiFi, Bluetooth, controle de motor e sensores. A adição de um **shield** a uma plataforma **Arduino Uno** é feita facilmente através da instalação da primeira nos pinos de ligação do segundo através de encaixe. A Figura 21 mostra este processo considerando um **shield** que contém um LCD e um conjunto de botões de pressão montado sobre um **Arduino Uno**.



Figura 21 - Adição de um shield LCD a uma placa Arduino Uno.

Em termos de software, o projeto **Arduino** inclui um ambiente de desenvolvimento integrado (IDE) que permite, utilizando uma versão simplificada da linguagem de programação C++, escrever, compilar e fazer *upload* de código. Neste curso, este IDE não será utilizado ficando toda a parte de codificação entregue ao **Snap4Arduino**.

Finalmente, é importante salientar que, tanto o *hardware* como o *software* do projeto **Arduino** é *open source*. Ou seja, os esquemáticos do *hardware* são de domínio público assim como o código fonte do *software*. Devido a este fato, existe uma vasta comunidade de utilizadores e desenvolvedores que contribuem com bibliotecas e outros recursos que fazem com que haja constantes atualizações e melhorias.

2.2 - Arduino Vs. Microcontrolador

Um **microcontrolador** é um **circuito integrado** que possui a capacidade de manipular informação sobre a forma digital. É uma espécie de processador como o que existe nos computadores pessoais, mas desenhado com um objetivo diferente. Tal como os processadores, possui um conjunto de instruções, memória e outros elementos que faz com que seja possível executar diferentes tipos de operações. No entanto, o seu grande objetivo é interligar-se ao mundo físico através da disponibilização de diferentes interfaces de comunicação. Na prática essas interfaces de comunicação são acessíveis através dos **pinos** desse **circuito integrado**. A Figura 22 mostra diferentes encapsulamentos de microcontroladores onde se pode ver um conjunto de pinos que permitem, entre outras funções, a **troca de informação** com o exterior.

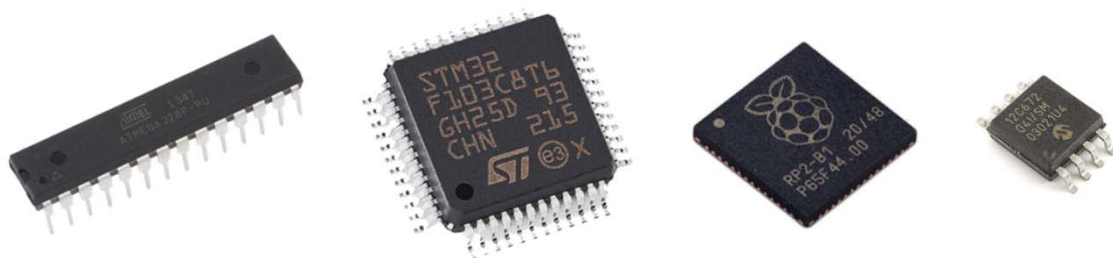


Figura 22 - Exemplos de diferentes microcontroladores com diferentes encapsulamentos.

Normalmente, um microcontrolador não funciona sozinho e precisa de outros elementos à sua volta para poder operar convenientemente. É isso que o hardware das plataformas **Arduino** oferece. O **microcontrolador** juntamente com outros **circuitos integrados** e dispositivos eletrónicos que formam um conjunto pronto para funcionar. A Figura 23 mostra a localização do microcontrolador na placa Arduino Uno⁵ assim como os restantes circuitos integrados e componentes necessários ao seu funcionamento.

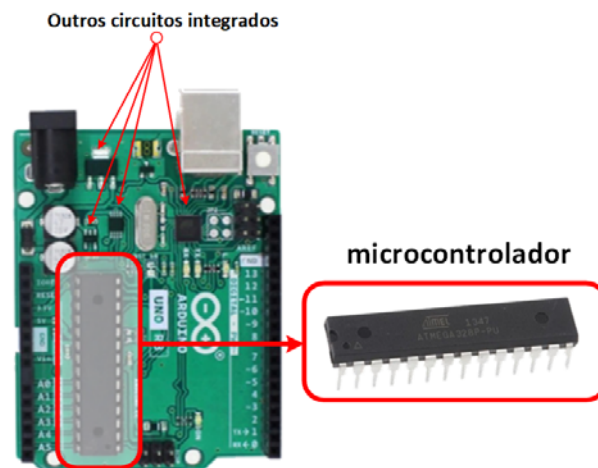


Figura 23 - Identificação do microcontrolador e outros circuitos integrados na plataforma Arduino Uno.

Em conclusão, um **microcontrolador** é um circuito digital integrado numa pastilha de silício que realiza operações de acordo com um **programa** que é colocado na sua **memória**. Esse **microcontrolador** não pode funcionar sozinho e requer outros componentes para poder operar. A plataforma **Arduino**, para além do microcontrolador, inclui todos esses componentes o que facilita a utilização do **microcontrolador** por pessoas leigas na área da eletrónica.

2.3 - Características da plataforma Arduino Uno

Agora que já se conhece o aspeto físico da placa de desenvolvimento **Arduino Uno** e a sua relação com o **microcontrolador**, é importante detalhar algumas das duas características. Nomeadamente os portos de entrada e saída e a forma como esta se interliga com o computador pessoal para poder ser programada e testada. A Figura 24 indica alguns dos elementos que fazem parte do **Arduino Uno**. Em particular, o porto USB que será responsável pela comunicação entre o microcontrolador e o computador pessoal e os portos de entrada e saída (**I/O**) responsáveis pela interação da plataforma com o mundo físico. Quando o **Arduino Uno** está ligado a um computador, a alimentação da placa deriva do próprio computador. No entanto,

⁵ Existem diversas revisões da plataforma Arduino Uno envolvendo microcontroladores com diferentes encapsulamentos.

nos casos em que a plataforma deve operar de forma isolada, existe um conector (*jack*) que permite a sua ligação a uma fonte alternativa como por exemplo baterias⁶.

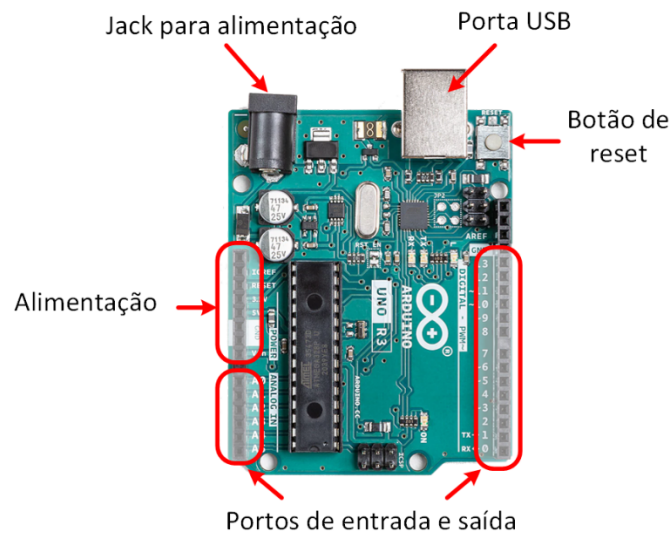


Figura 24 - Organização da placa Arduino Uno.

Um botão de pressão de *reset* é também incluído e serve para reiniciar o programa existente no **microcontrolador** sempre que este for premido. Finalmente, os pinos de alimentação são utilizados para, entre outros fins, alimentar os **shields** que são instalados sobre o **Arduino**. Existem ainda outros pinos, que não foram identificados nesta figura, mas que não são relevantes para o atual curso.

2.4 - Ligação do Arduino ao computador

A ligação do **Arduino Uno** ao computador é feita via um cabo USB (tipo A/B) como se mostra na Figura 25.

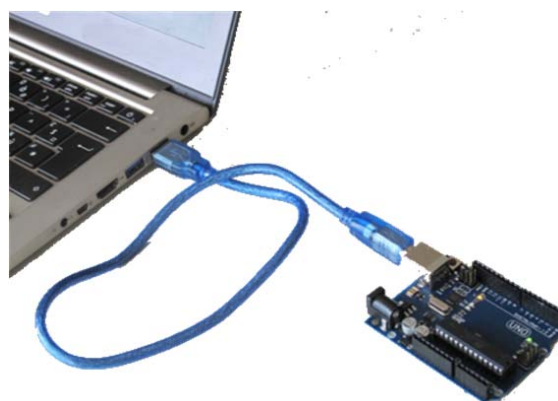


Figura 25 - Ligação da plataforma Arduino Uno ao computador via cabo USB.

⁶A gama de tensões de operação neste caso deverá estar entre os 6V e os 20V.

Com o sistema operativo Windows 10 ou superior, o hardware é reconhecido automaticamente e todos os elementos necessários (drivers) instalados automaticamente. No final do processo, a placa é reconhecida (enumerada) como um porto COM como se mostra na Figura 26.

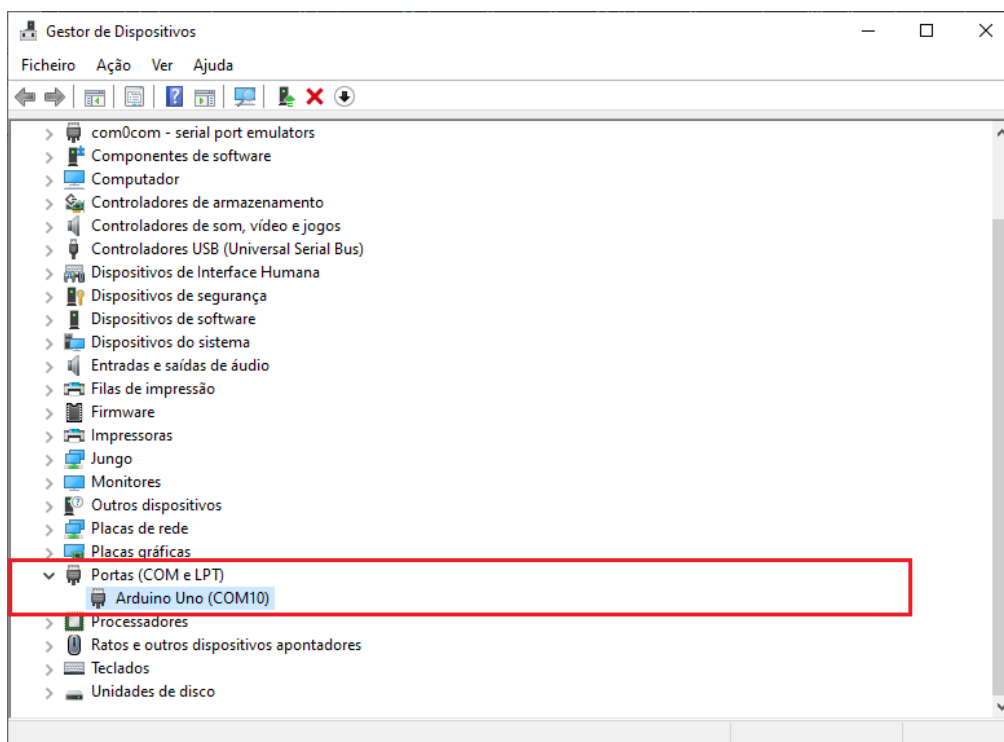


Figura 26 - Resultado do processo de ligação da placa Arduino ao computador via porto USB.

Depois de conetado e reconhecido pelo computador, é possível estabelecer comunicação com o **Arduino Uno** permitindo, entre outros, a sua programação.

2.5 - Portos de I/O

Um dos detalhes mais importantes a compreender para a utilização do **Arduino Uno** é a forma como este se pode interligar com o mundo físico. Esta interligação é feita através dos seus portos de I/O. Na Seção 2.3 apresentou-se a localização desses portos e nesta seção serão detalhadas algumas das suas características e funcionalidades. Este tema é relativamente complexo dado que cada um dos pinos associados a esses portos pode ter diferentes funcionalidades permitindo a comunicação entre a plataforma **Arduino** e o exterior com diferentes protocolos. No entanto, dado o carater introdutório deste curso, a descrição dos portos de I/O será reduzida ao seu nível mais básico.

À direita do Arduino Uno, e como se pode ver na Figura 27, existe um conjunto de **13 portos de entrada e saída** do tipo **digital**. Desses 13 portos, e por razões que para já são irrelevantes, apenas serão utilizados os pinos ② a ⑬.

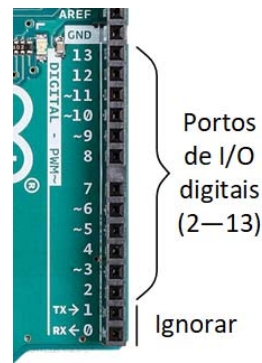


Figura 27 - Portos de I/O digitais associados à plataforma Arduino Uno.

Neste momento, abre-se um parêntese para falar de alguns termos introduzidos no parágrafo anterior. Em particular a ideia de **digital** e o conceito de **entrada** e **saída**.

Considere-se uma lâmpada numa divisão qualquer de uma casa. Suponha-se que esta lâmpada pode estar apenas num de dois **estados** possíveis: “acesa” ou “apagada”. Neste caso, e como a lâmpada apenas permite dois **estados**, o seu comportamento é **digital**. O estado associado a uma grandeza digital pode assim ser apenas uma de duas possibilidades. Ainda que “aceso” e “apagado” seja uma designação perfeitamente válida para designar os estados da lâmpada, de forma mais comum, os dois estados são designados por “0” e “1” ou “Baixo” e “Alto”. Por exemplo, o estado “0” associado à lâmpada pode indicar que esta se encontra apagada e o estado “1” que se encontra acesa.

Em relação aos portos 2 a 13, estes são digitais no sentido de que apenas permitem a existência nesses pinos de um dos dois estados possíveis: “0” e “1”. Em **sistemas digitais eletrónicos**, com frequência os níveis lógicos “0” e “1” são representados fisicamente por diferentes níveis de **tensão elétrica**. Na plataforma **Arduino Uno** que será utilizada neste curso, o estado “0” é representado por uma diferença de potencial, relativamente à referência (designada por GND ou *ground*), igual a **0V**. Por outro lado, o valor lógico “1” é representado por uma tensão elétrica de **5V** (em relação à mesma referência).

Agora a noção de entrada e saída tem a ver com a direção em que a **informação**, sobre a forma digital, flui. Se um dos portos estiver configurado como **entrada**, então o microcontrolador executa operações de “**leitura**” sobre esse porto. Ou seja, o **microcontrolador** pode *aceder* a esse porto e verificar qual o nível lógico que está presente. Por outro lado, se um porto estiver configurado como **saída**, o valor lógico que lá está presente será estabelecido pelo **microcontrolador** através de um processo de “**escrita**”. Ou seja, num determinado instante, o estado de um porto configurado como saída é determinado pelo programa que está a ser executado no **microcontrolador**. Ainda que estes conceitos pareçam complicados, serão

tornados mais transparentes quando, no próximo capítulo, forem apresentados alguns exemplos ilustrativos.

À esquerda do **Arduino Uno**, existem também um conjunto de portas, identificados por **0** a **5**, que, não sendo **digitais**⁷, são designados por **analógicos**. A Figura 28 ilustra a sua localização.

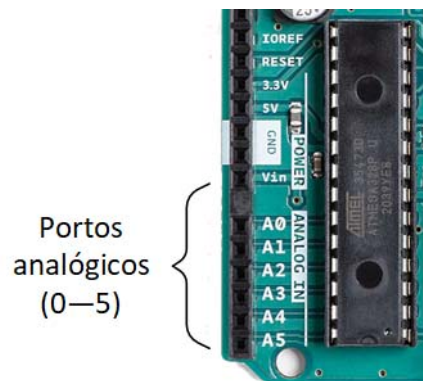


Figura 28 - Portos analógicos associados à plataforma Arduino Uno.

Estes pinos, quando a funcionar como portos analógicos, são **apenas de entrada** e não de saída o que significa que a direção da informação é do exterior para o interior do microcontrolador. Tal como aconteceu anteriormente, faz-se aqui uma pausa para se apresentar o conceito de **analógico**. Imagine-se que no quarto onde a lâmpada referida anteriormente se encontra, existe também um termómetro de mercúrio. Um termómetro de mercúrio é um dispositivo que assenta na dilatação dos metais com a temperatura. Um capilar ligado, a um reservatório de vidro, é preenchido por mercúrio. Um aumento de temperatura expande o mercúrio que é obrigado a subir pelo capilar. A Figura 29 ilustra esta ideia.

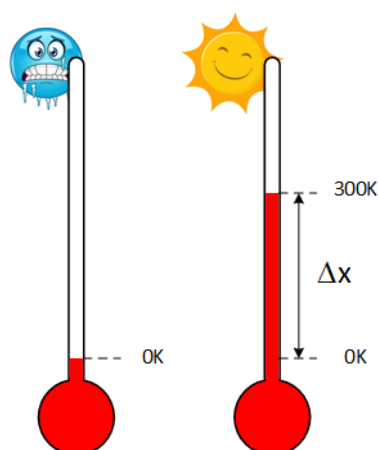


Figura 29 - Altura da coluna de mercúrio num termómetro como função da temperatura.

⁷ Na realidade, podem ser configurados como pinos de I/O digitais, mas isso é uma outra história...

Quanto maior a temperatura maior é a expansão do mercúrio e mais alta é a sua posição no capilar. A altura da coluna de mercúrio, relativamente a uma referência, é designada por Δx e é uma **variável real** pelo que pode tomar **qualquer valor** entre **0** e o comprimento total do capilar. Neste caso, Δx é uma variável **analógica**. Ao contrário de uma variável digital que pode apenas tomar dois estados distintos, uma variável analógica possui um número **infinito** de estados distintos. Na prática, no entanto, devido ao limite de resolução dos instrumentos de medida, não é possível determinar o valor de Δx com precisão infinita pelo que o número de estados, ainda que seja muito elevado, não é infinito. Por exemplo, se Δx fosse medida com uma régua vulgar, onde a resolução é 1 mm (com um erro absoluto de $\pm \frac{1}{2}$ mm), e se o comprimento máximo do capilar fosse 1 cm, o número de estados possíveis, tendo em conta a apresentação da leitura com dois dígitos significativos, seriam apenas 101 (valores que se estendem de 0.0, 0.1, ..., até 1.0).

No **Arduino Uno**, uma entrada **analógica** converte um sinal elétrico, com **valores reais** entre [0,5] V, aplicado à sua entrada num valor **inteiro** entre **0** e **1023** (ou seja, 1024 estados diferentes). Ainda que, mais uma vez, este conceito pareça complicado, adiante, e através de alguns exemplos, espera-se dissipar qualquer dúvida acerca disto. □

Parte 3: Ligando o Arduino Uno ao Snap4Arduino

3.1 - Preparação do Arduino Uno (aconselhado)

Neste momento já foi instalado o **Snap4Arduino** e já se apresentaram alguns detalhes operacionais associados à plataforma **Arduino Uno**. Agora é preciso **juntar tudo**. **No entanto**, e na **primeira vez** que um **Arduino** é utilizado com o **Snap4Arduino** é necessária uma etapa adicional que consiste em **carregar** um programa especial.

Comece por **ligar** o **Arduino Uno** à porta USB do computador. A seguir, clique no ícone da engrenagem e selecione a opção “**more supported devices**” como se mostra na Figura 30.

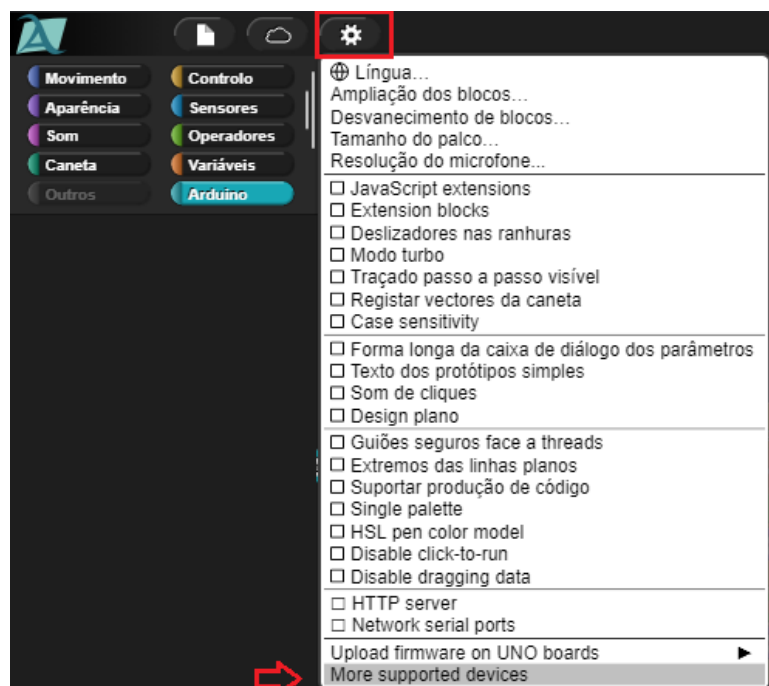


Figura 30 - Menu de configurações do Snap4Arduino.

Na janela que, entretanto, se vai abrir, selecione a opção **upload** garantindo que a opção **FirmwareSAS5 Tone** está selecionada. A Figura 31 detalha este procedimento.

Supported Devices

Snap4Arduino requires boards with Firmata firmware uploaded.

UNO boards

You can upload Firmata firmwares directly from Snap4Arduino (with both desktop and online versions) to UNOs compatible boards. Or just here:

- Be sure you are using Chromium/Chrome/Edge browser and you are under <https>
- Plug your UNO by USB
- Choose your firmware and just upload it!

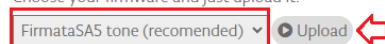


Figura 31 - Janela de diálogo para upload de software no Arduino Uno.

Na Figura 32 apresenta-se a janela que aparece para se proceder à escolha do porto de comunicação e a Figura 33 a janela que se abre para confirmar a execução correta do processo de carregamento do software.

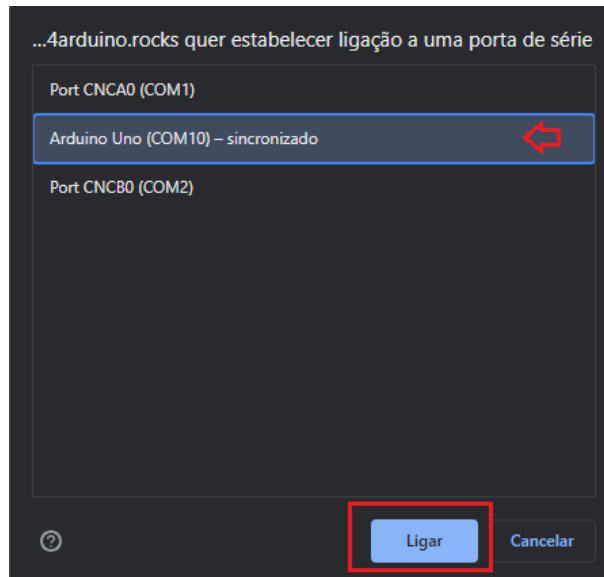


Figura 32 - Seleção do porto de comunicação.

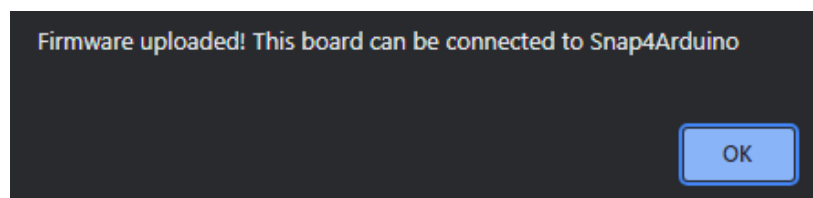


Figura 33 - Resultado final de uma operação de gravação com sucesso.

Neste momento, a plataforma **Arduino Uno** está pronta a comunicar com o **Snap4Arduino**. Se, por alguma razão, não conseguir executar este procedimento com este método, a seção que se segue mostra uma forma alternativa independente de carregar o software necessário no **Arduino Uno** sem recorrer ao **Snap4Arduino**.

3.2 - Preparação do Arduino Uno (alternativo)



Na eventualidade do processo ilustrado no ponto anterior não funcionar, apresenta-se nesta seção uma forma alternativa de carregar o Arduino Uno com o programa necessário para que este possa comunicar com o Snap4Arduino. O procedimento não é complicado, mas envolve a execução de um conjunto de passos que podem criar alguma confusão. Por este motivo, todo o processo é ilustrado através de uma sequência ilustrada com figuras.

Passo 1: Fazer o download do ficheiro com nome **workshop.hex**. Para isso, basta aceder ao [repositório](#) apresentado no primeiro capítulo e clicar na pasta **Firmware**.

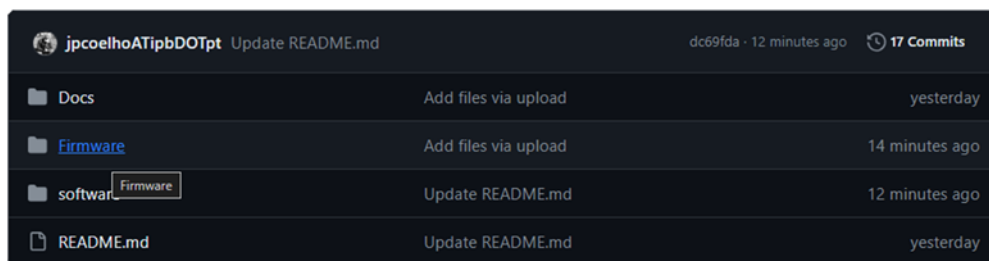


Figura 34 - Pasta no repositório onde se encontra guardado o ficheiro **workshop.hex**.

Passo 2: Dentro dessa pasta existe um ficheiro com o nome **workshop.hex**. Coloque o ponteiro do rato sobre ele e clique no botão **direito** e escolha a opção “**Guardar link como...**”. A Figura 35 ilustra este procedimento.

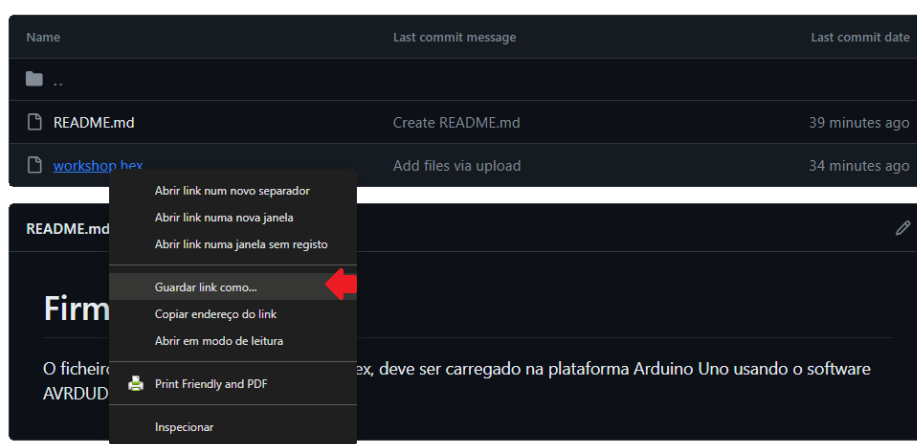


Figura 35 - Procedimento de descarga do ficheiro **workshop.hex**.

Selecione a opção “**Guardar**” na janela que se abre depois de seleccionar a opção anteriormente referida e verifique, na pasta de transferências, que possui o ficheiro **workshop.hex**. A Figura 36 mostra o resultado final.

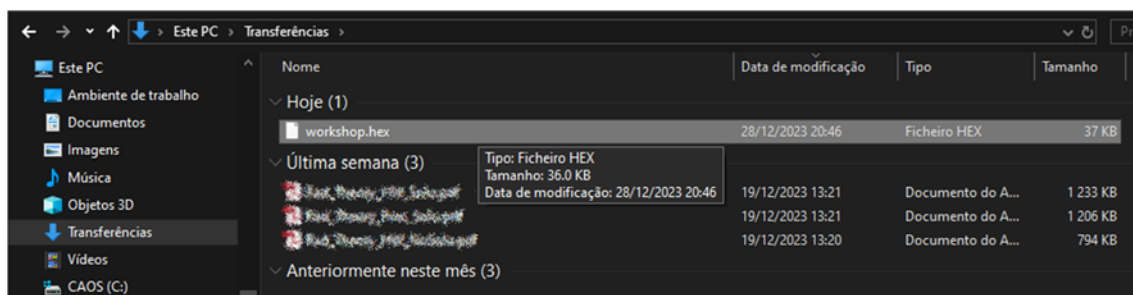


Figura 36 - Resultado final da operação de descarga do ficheiro workshop.hex.

Passo 3: Descarregar do repositório o programa **AVRDUDESS**. Para isso deve aceder ao endereço:

https://github.com/jpcoelhoATipbDOTpt/LED_Laboratorios_de_Educacao_Digital/tree/main/software

e clicar com o botão **esquerdo** do rato sobre o nome do arquivo como se mostra na Figura 37.

Name	Last commit message	Last commit date
..		
AVRDUDESS-2.14-portable.zip	Add files via upload	1 hour ago
README.md	Update README.md	1 hour ago
Snap4Arduino_desktop-win-installer-32_9.1.1...	add zip files	yesterday
Snap4Arduino_desktop-win-installer-64_9.1.1...	add zip files	yesterday

Figura 37 - Localização do arquivo AVRDUDESS.

Tal como aconteceu no primeiro capítulo, deverá seleccionar o ícone de descarga como se mostra na Figura 38.

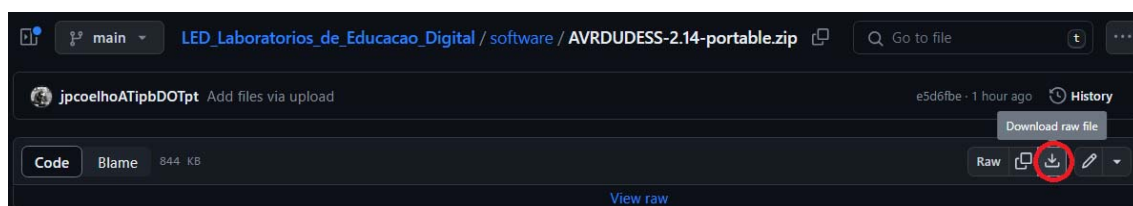


Figura 38 - Processo de download do arquivo AVRDUDESS.

Depois do ficheiro descarregado, deve confirmar-se que este se encontra na área de transferências como se mostra na Figura 39.

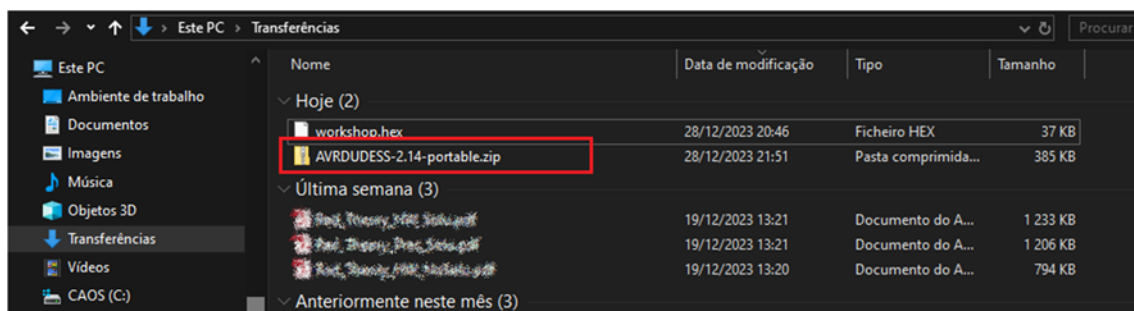


Figura 39 - Resultado após o download do AVRDUDESS.

Passo 4: Conectar o Arduino Uno à porta USB do computador.



Figura 40 - Conexão PC - Arduino Uno

Passo 5: Descomprimir a pasta **AVRDUDESS-2.14-portable.zip**. Dentro da pasta descomprimida encontra-se o programa **avrdudess.exe** que deverá executar através de **duplo clique** sobre o nome do ficheiro.

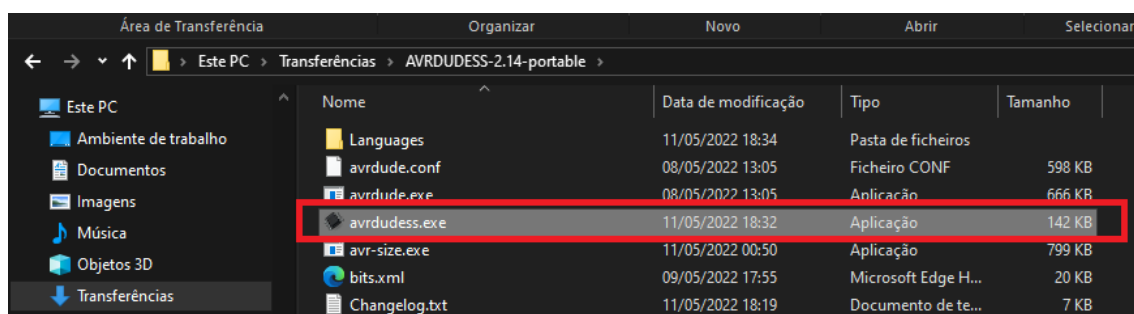


Figura 41 - Conteúdo da pasta comprimida **AVRDUDESS-2.14-portable.zip**.

Passo 6: Dentro do programa, em “**Select a programmer...**” escolher “**Arduino**” e em “**Select an MCU...**” procurar e seleccionar **ATMega328P**. No campo “**Port**” seleccione o porto associado ao Arduino Uno que está ligado ao computador. Será um porto que começa com as letras **COM** seguido de um número. É muito provável que só apareça uma única opção com estas características. No entanto, em caso de dúvida, desconecte o Arduino Uno e veja qual dos portos desapareceu. Assim, quando o

voltar a ligar já sabe qual o número que lhe será atribuído. As três figuras que se seguem pretendem ilustrar as ações a realizar neste passo.

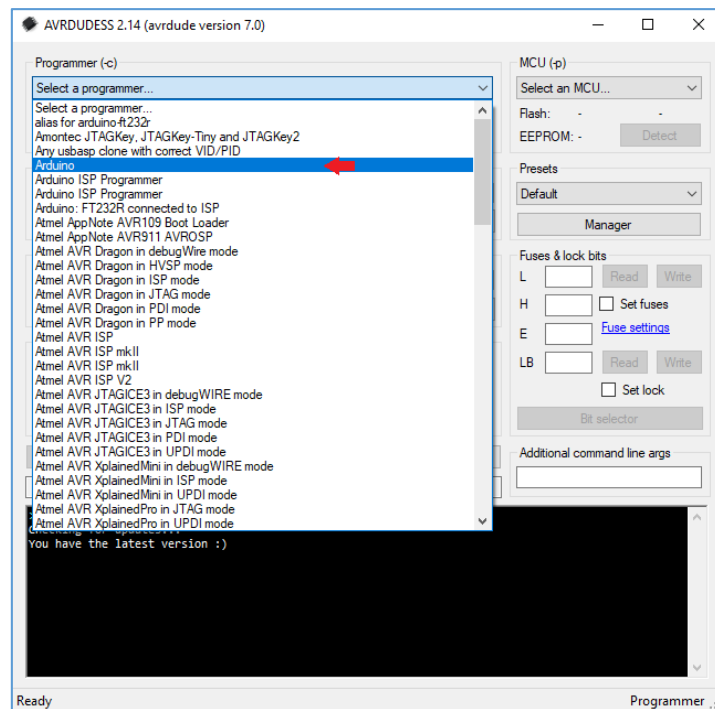


Figura 42 - Seleção do tipo de programador.

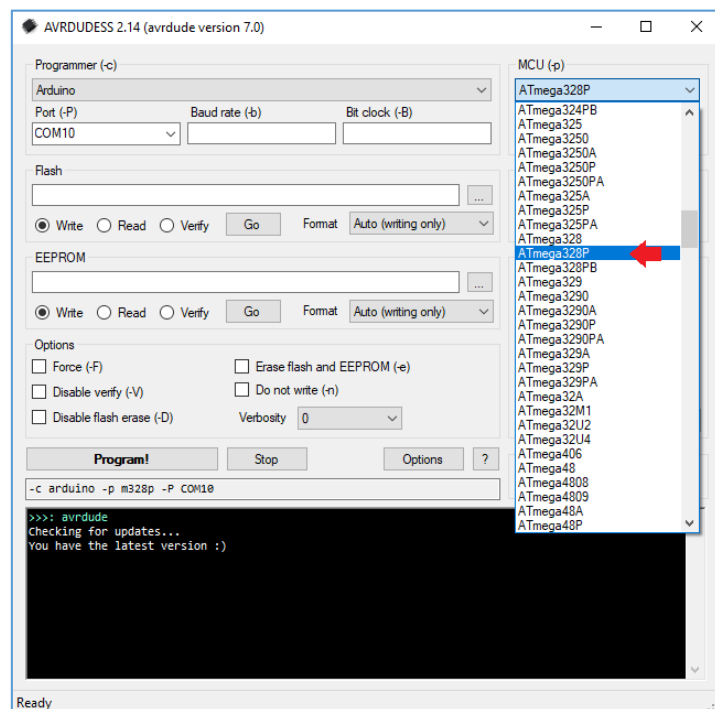


Figura 43 - Seleção da referência do microcontrolador.

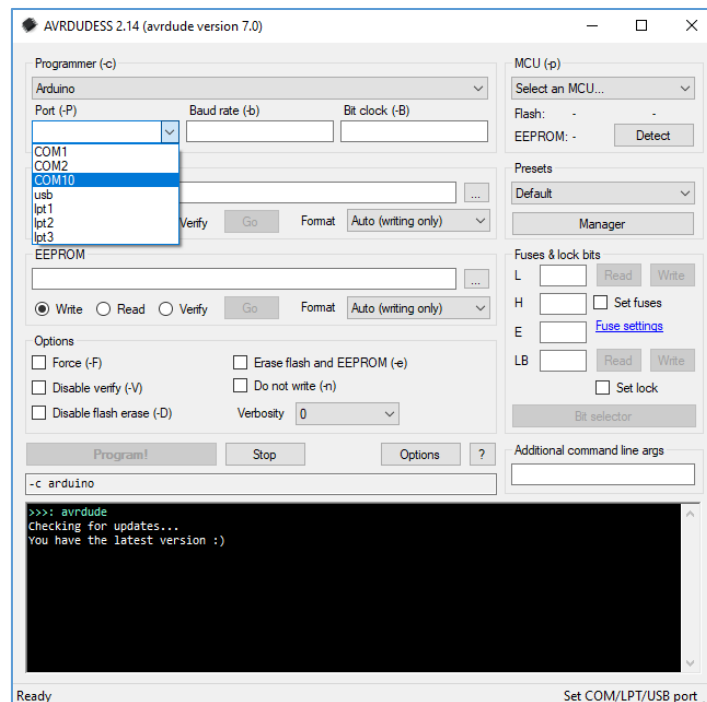


Figura 44 - Seleção da porta de comunicação com o Arduino Uno.

Passo 7: Selecionar no campo “Flash” a localização do ficheiro **workshop.hex** que foi descarregado no início deste capítulo. Para isso, siga as imagens apresentadas nas duas figuras que se seguem.

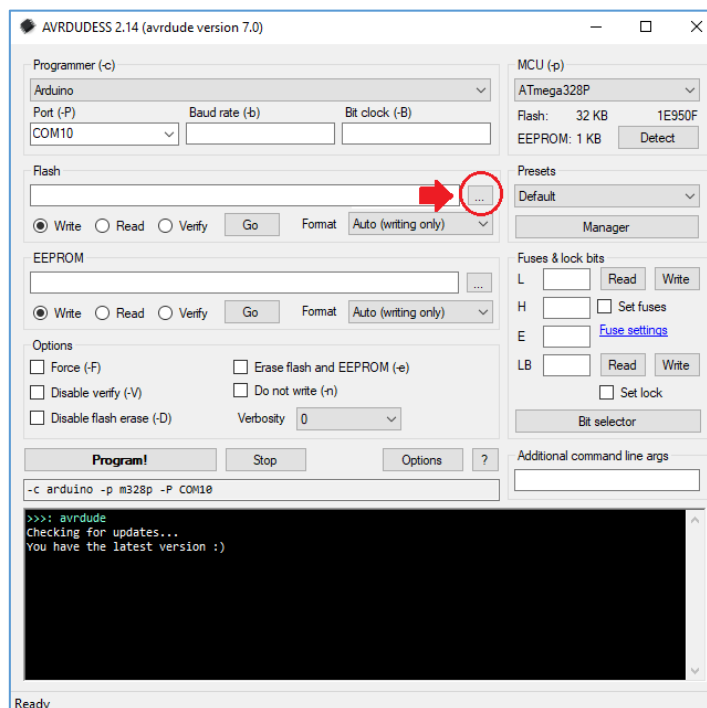


Figura 45 - Abrir localização do ficheiro HEX.

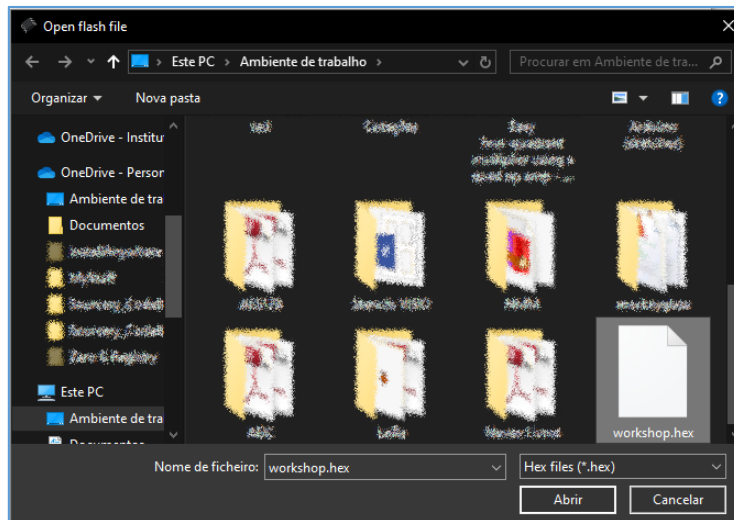


Figura 46 - Navegar até à área de transferências (local onde guardou o ficheiro) e selecionar workshop.hex.

Passo 8: Finalmente (ufa!), premir o botão **“Program!”** e aguardar a conclusão do processo de descarga. A Figura 47 mostra a localização desse botão e a Figura 48 apresenta o diálogo final após o processo de carregamento.

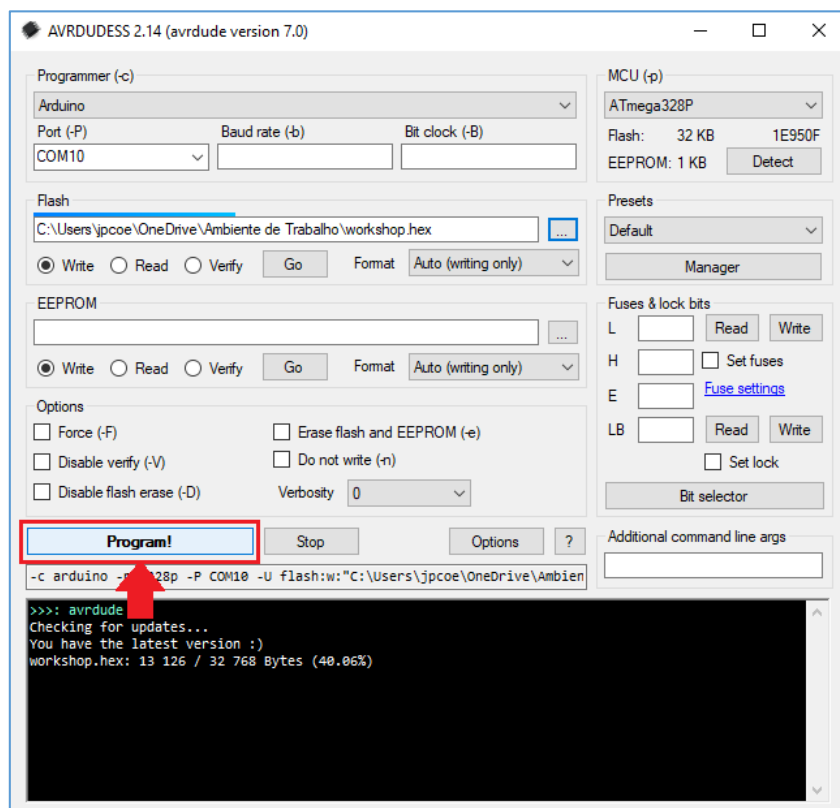


Figura 47 - Ação de programação do Arduino Uno.

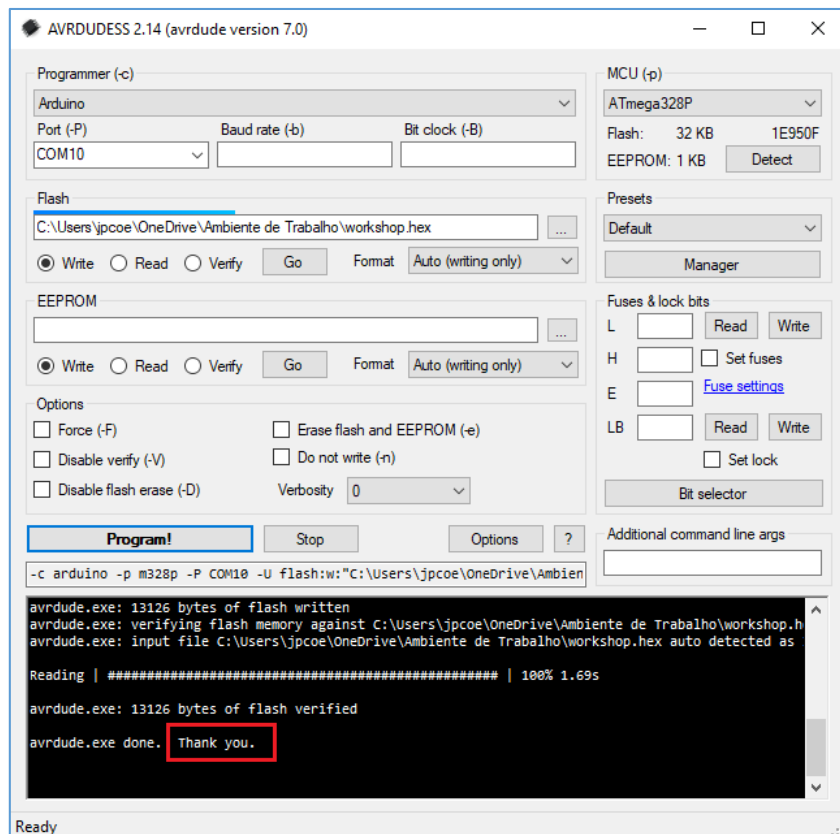


Figura 48 - Resultado da conclusão do processo de upload do workshop.hex na plataforma Arduino Uno.

Antes de terminar esta seção deixam-se aqui duas considerações. A primeira é de que seria possível carregar este programa de comunicação entre o Arduino Uno e o Snap4Arduino recorrendo a uma estratégia alternativa. Essa abordagem iria requer a instalação do software **Arduino IDE** e o restante processo não seria mais simples do que se enumerou ao longo desta seção. Deste modo, optou-se por este método que evita a instalação de um software com uma dimensão muito maior. A segunda é que este processo **é apenas necessário ser executado uma única vez**⁸! Depois do **Arduino Uno** ter o programa guardado em memória este **mantém-se armazenado** e não é eliminado mesmo depois de ser desligado.

3.3 - Interligação do Arduino Uno e do Snap4Arduino

Nesta seção, e depois de toda a *stack* de *software* instalado, será apresentado a forma como a comunicação entre o **Arduino Uno** e o **Snap4Arduino** é formalizada. Para isso, e com o **Arduino** conectado ao computador via porto USB, abrir o software **Snap4Arduino**. Se não houver um atalho no ambiente de trabalho para o programa, pode fazer-se a procura utilizando o motor de busca do Windows como se mostra na Figura 49.

⁸ Para cada diferente plataforma Arduino Uno que se pretenda interligar com o Snap4Arduino.

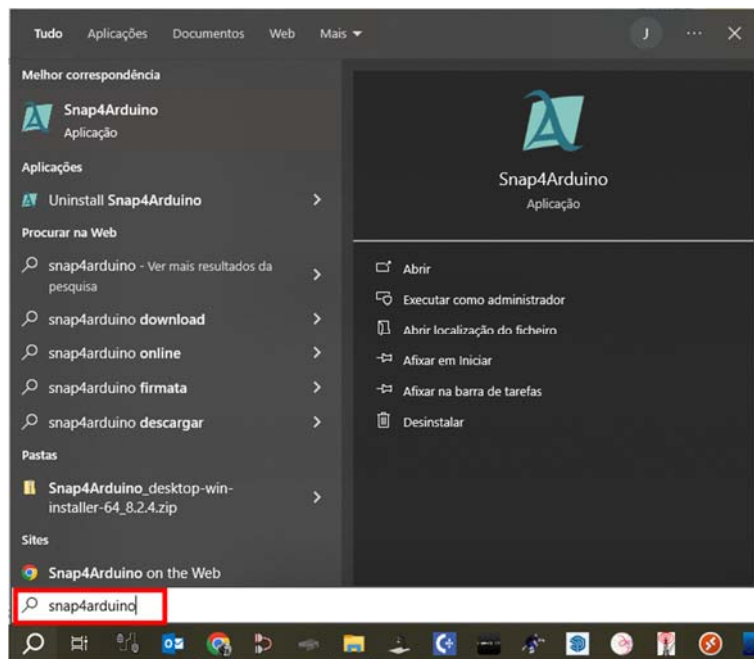


Figura 49 - Encontrar o Snap4Arduino utilizando o motor de pesquisa do Windows.

Depois do **Snap4Arduino** aberto, procurar o botão com a designação “**Arduino**” no canto superior esquerdo e clicar.

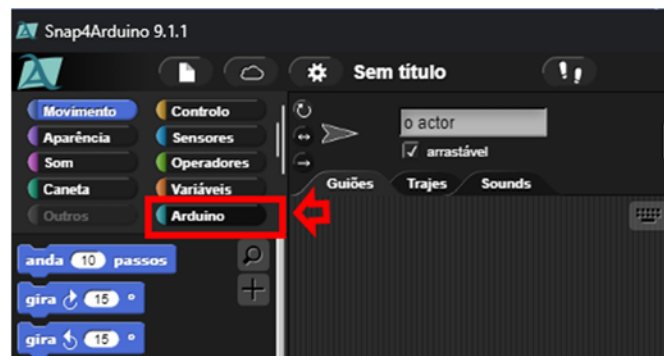
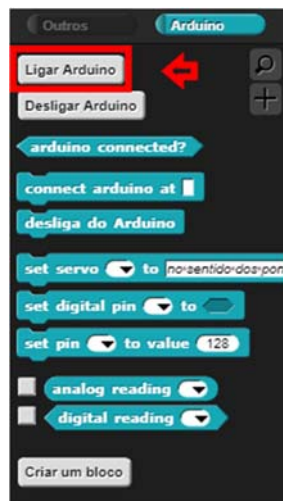


Figura 50 - Localização do botão "Arduino" no Snap4Arduino.

Decorrente desta ação, deverá aparecer, na zona inferior, uma paleta de operações associadas ao **Arduino** como se mostra na Figura 51a. Para além disso, e como a mesma figura ilustra, deverá ser premido o botão “**Ligar Arduino**” a fim de se estabelecer a comunicação entre o software e a plataforma de desenvolvimento **Arduino**. Desta ação resulta a abertura de uma lista com todos os portos **COM** instalados no computador. Deverá ser selecionado aquele que se encontra associado ao **Arduino Uno** como se mostra na Figura 51b.

Durante o processo de ligação, uma janela informativa é apresentada ao utilizador como aquela que se mostra na Figura 52a. Após escassos segundos, e em caso de sucesso, essa janela é substituída por aquela ilustrada na Figura 52b.



(a)

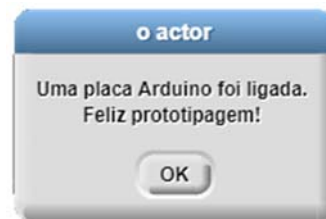


(b)

Figura 51 – (a) Paleta de operações e objetos disponibilizados quando a opção "Arduino" é ativada. (b) Seleção da porta de comunicação (COM) associada ao Arduino Uno.



(a)



(b)

Figura 52 – em (a), janela apresentada durante o processo de conexão e, em (b), a janela que aparece após ligação com sucesso.

Caso se obtenha uma mensagem de **erro** ou o porto COM não apareça como opção deverá verificar:

- se o **Arduino Uno** se encontra ligado;
- se o cabo USB utilizado está em condições de operação;
- se o ficheiro **workshop.hex** foi devidamente instalado.

Em caso de sucesso, **parabéns!**

Agora é altura de começar a programar o “**bichinho de 7 cabeças**”⁹ ... □

⁹ a.k.a. Arduino Uno < ☺

Parte 4: Programando o Arduino Uno com o Snap4Arduino

4.1 - Os primeiros sketches

O principal objetivo deste workshop é aprender a programar o **Arduino Uno** utilizando **Snap!/Scratch**. Neste momento toda a cadeia de *software* necessária ao processo já se encontra instalada. Em particular, existe uma ligação lógica entre o **Snap4Arduino** e o *hardware* **Arduino Uno**. Assim, o passo que se segue consiste em realizar um primeiro programa (designado por **sketch**) que permita interagir com o **Arduino**. É de salientar que tópicos associados à programação em **Snap!** Serão introduzidos sempre no contexto da programação com o **Arduino** e apresentados à medida que os desafios vão sendo lançados. Neste quadro de referência, começa-se com um primeiro desafio que será colocado nos seguintes termos:

Desafio 1. Controlo do estado de um LED

Associado ao porto ⑬ do **Arduino Uno** existe uma pequena “luzinha” cujo estado (aceso/apagado) pode ser controlado. Essa “luzinha” é, na realidade, um dispositivo semicondutor designado pelo acrónimo LED¹⁰ (do Inglês *Light Emitting Diode*). Quando se mandar “escrever” o valor lógico 1 no porto ⑬, o LED ligado a esse porto irá acender. Por outro lado, se o valor lógico a ser “escrito” no porto ⑬ for “0”, o LED desliga.

O objetivo deste primeiro desafio é criar um **sketch** que seja capaz de controlar o estado do LED. Para isso, no **Snap4Arduino**, identifique na paleta associada ao **Arduino** o objeto



Utilizando o rato, enquanto se pressiona o botão esquerdo do rato em cima desse objeto, desloca-se para o ambiente de trabalho como se mostra na Figura 53. Depois de posicionado, o passo seguinte é parametrizar este objeto. Existem dois parâmetros a definir sendo que o primeiro é o número do porto ao qual se pretende aceder e o segundo o valor lógico que se pretende “escrever”. Relativamente ao primeiro, quando se clicar no item **pin** do objeto será aberto um *menu* onde poderá escolher o número do porto. Neste caso, será selecionado o ⑬ pois é nesse que o LED se encontra instalado. O segundo parâmetro aparece depois da palavra “to” e, com o ponteiro do rato no interior do hexágono, clicar até que a cor **verde** apareça. No Snap4Arduino, a cor **verde** está associada ao valor lógico “1” e a cor **vermelho** ao valor lógico “0”. A Figura 54 mostra a parametrização deste objeto.

¹⁰ Não confundir com LED de Laboratórios de Educação Digital ©

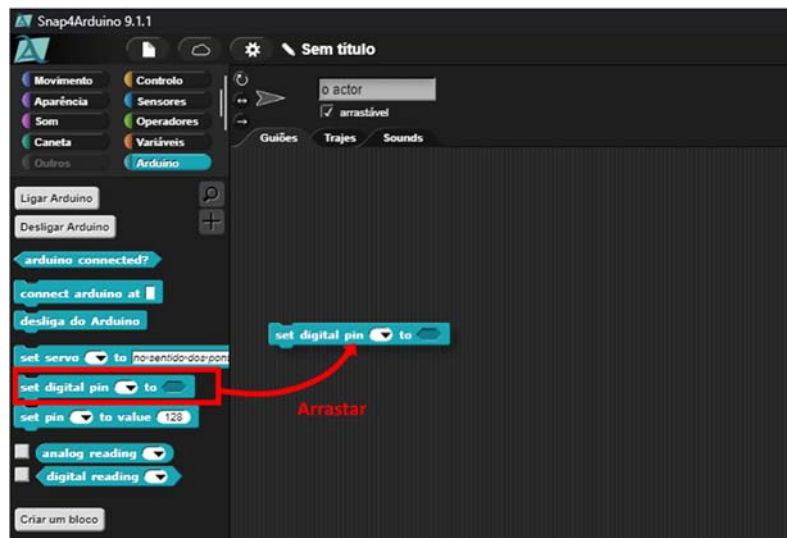


Figura 53 - Colocação do objeto "set digital pin" no ambiente de trabalho.

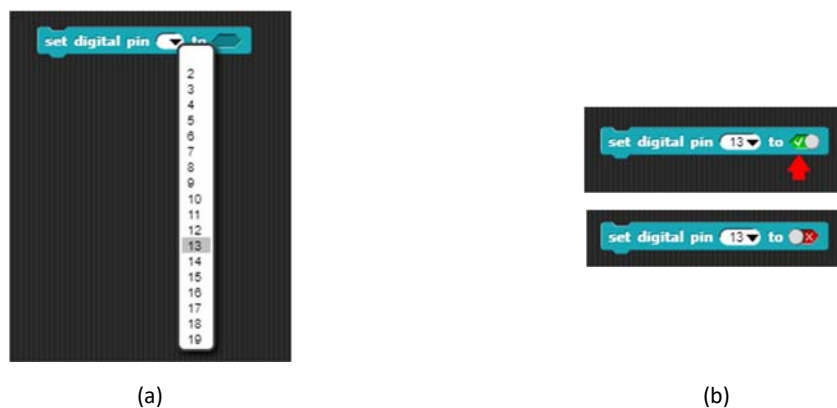


Figura 54 - Em (a), parametrização do porto de I/O e em (b) definição do valor lógico associado ao porto.

O último passo consiste em ligar a este objeto um controlo que servirá para “disparar” a ação que foi colocada no ambiente de trabalho. Para isso, deve seleccionar-se o botão **Controlo** no canto superior esquerdo e, de entre a paleta que se abre, seleccionar e arrastar com o rato o objeto **Quando alguém clicar em**.



Figura 55 - Adicionar controlo "Quando alguém clicar em" ao objeto anterior.

Neste momento o primeiro sketch está concluído. No **Snap4Arduino** não existe a necessidade da execução de qualquer ação para executar um **sketch**. Por defeito, a execução é realizada ao longo da própria criação do **sketch**. Assim, para testar o programa que acabou de ser construído basta usar o rato e clicar em cima do objeto de controlo **Quando alguém clicar em**.

A Figura 56 mostra a localização e o estado do LED após a ação referida no parágrafo anterior. Para apagar o LED basta selecionar a cor vermelha no hexágono e clicar novamente no objeto

Quando alguém clicar em. Confirme, experimentalmente, esta ação.



Figura 56 - Localização e estado do LED depois da execução do sketch associado ao Desafio 1.

Desafio 2. Controlo do estado de um LED (parte II)

No **Snap4Arduino**, é possível ter diversas ações a terem lugar de forma concorrente. Por exemplo, na Figura 57 existem dois controlos idênticos que disparam duas ações distintas: a da esquerda faz com que o LED acenda e a da direita faz com que este apague. Este **sketch** pode ser obtido facilmente, sem ter que repetir o processo de arrasto dos objetos, copiando o que já existe do **Desafio 1** clicando, para isso, sobre os objetos com o botão direito do rato e escolher a opção “**duplicar**”.

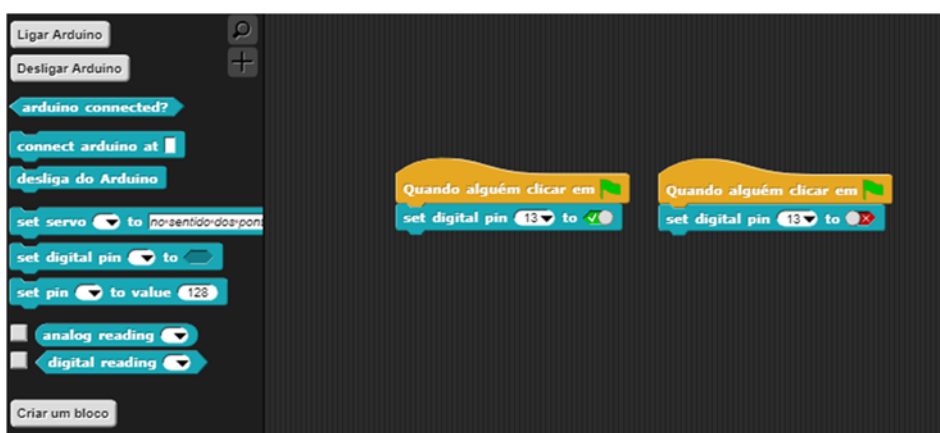


Figura 57 - Execução concorrente de ações.

Agora, de forma alternada, clique com o rato em cada um dos objetos amarelos e verifique que, quando seleciona o da esquerda, o LED acende e, quando seleciona o da direita, o LED apaga.

Desafio 3. *Leitura do estado de um porto digital*

O **sketch** que foi desenhado no primeiro desafio pode ser extrapolado para qualquer um dos diferentes portos disponíveis. Basta apenas para isso que selecione um valor diferente no parâmetro “pin”. Neste novo desafio, o objetivo não é “escrever”, mas sim “ler” de um porto. Para este ensaio, para além do **Arduino Uno**, será necessário um pequeno condutor¹¹ como aquele que se mostra na Figura 58.



Figura 58 - Condutor de ligação necessário para a realização do Desafio 3.

O objetivo deste desafio é “ler” o valor lógico presente no porto ②. Se esse valor lógico for “0” o LED no porto ⑬ deverá apagar. Caso contrário, se o valor lógico associado ao porto ② for “1”, o LED deverá acender. Para isso, será realizado um **sketch** utilizando o objeto **digital reading** para se conhecer o valor lógico do porto. Tal como aconteceu nos desafios anteriores, este bloco deve ser parametrizado com o número do porto que se pretende “ler” (neste caso ②). O **sketch** completo pode ser observado na Figura 59.

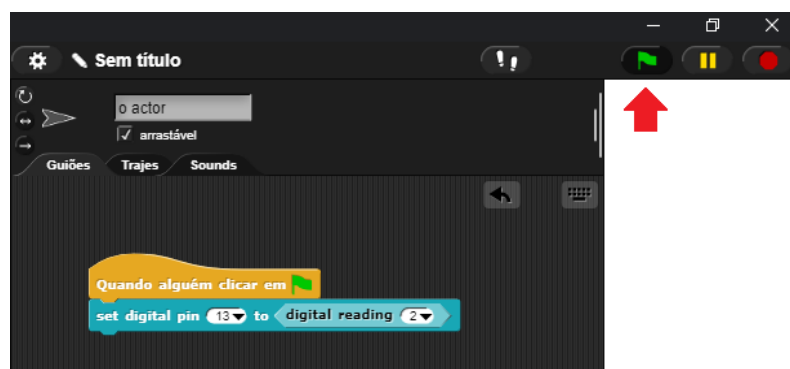



Figura 59 - Sketch associado ao Desafio3.

Para a execução deste programa, comece por fazer a ligação do porto ② do **Arduino Uno** ao pino designado por **GND**. Esta ligação está ilustrada na Figura 60a. Execute o programa

¹¹ Muitas vezes designado por *jumper*.

carregando na bandeira verde  ou no controlo amarelo. Verifique que o LED se encontra apagado. Agora, altere a ligação do fio condutor do pino **GND** para o pino **5V** como se mostra na Figura 60b. Execute novamente o **sketch** e confirme que o LED agora se acende. Repita as vezes que achar necessário para entender o conceito por detrás da ação realizada.

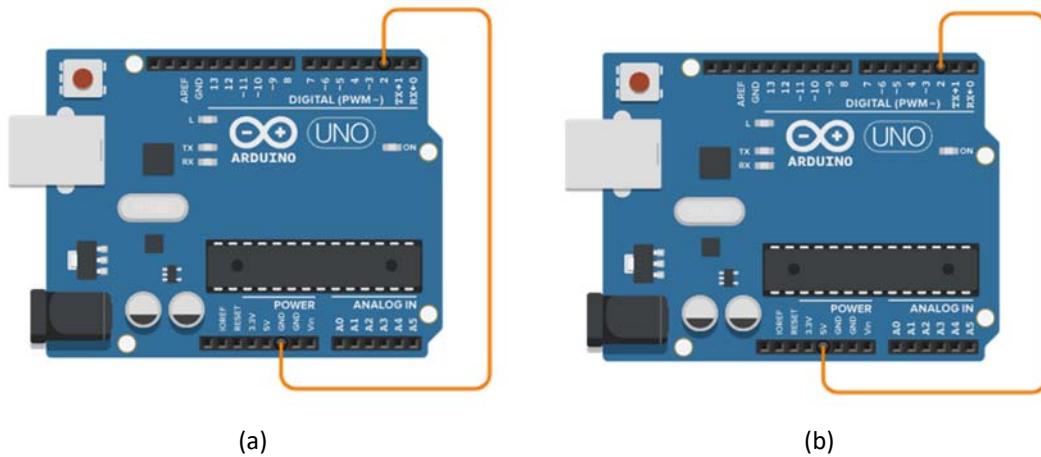


Figura 60 - Referências para ligação do fio condutor.

Desafio 4. Leitura do estado de um porto analógico

O **Arduino Uno** permite também **ler** um sinal analógico desde que representado como uma tensão elétrica entre **0** e **5V**. Internamente, o **microcontrolador** converte os valores de tensão num número inteiro entre **0** e **1023**. A Figura 61 mostra a forma como o mapeamento da tensão elétrica é realizado.

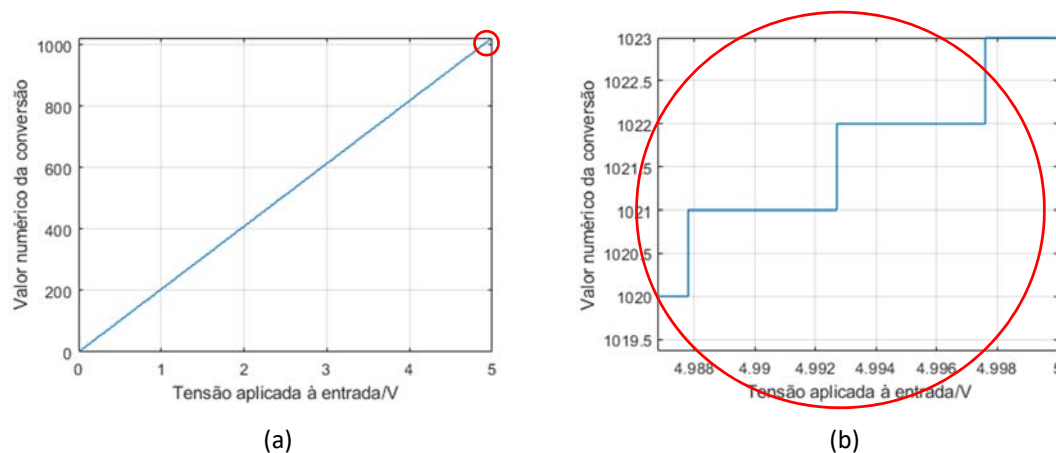





Figura 61 - Mapeamento da tensão elétrica aplicada à entrada analógica do Arduino Uno e a respetiva conversão num número inteiro. Em (b), detalhe do extremo da função de transferência.

Neste novo desafio, iremos confirmar este facto através de um ensaio que envolve a utilização do objeto `analog reading` . Este objeto tem como argumento um dos portos seis portos de entrada analógicos identificados pelos números de **0** a **5** e retorna um número inteiro que representa a conversão do valor de tensão colocado à sua entrada. A apresentação, na tela do

computador, do valor analógico convertido para inteiro será feita através da utilização do bloco  existente na paleta de .

Assim, e no ambiente de trabalho, coloque os blocos representados na Figura 64 e empilhados na forma como se mostra¹².

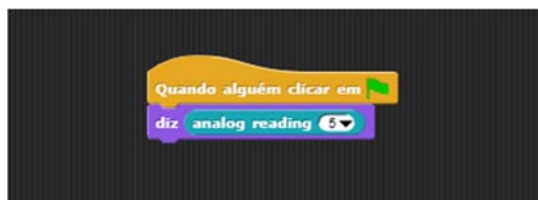



Figura 62 - Diagrama de blocos associado ao Desafio 4.

A seguir, utilizando o condutor elétrico do desafio anterior, faça a ligação que se mostra na Figura 63a (do porto analógico 5 para o pino de 5V). Finalmente, carregando na bandeira verde  ou no controlo amarelo, verifique que o “ator” associado ao projeto apresenta a mensagem com o valor 1023. Faça agora a ligação que se mostra na Figura 63b e repita a execução do programa. Deverá observar que o valor foi substituído pelo número 0. A Figura 64 ilustra o resultado que deverá obter para cada uma das duas situações.

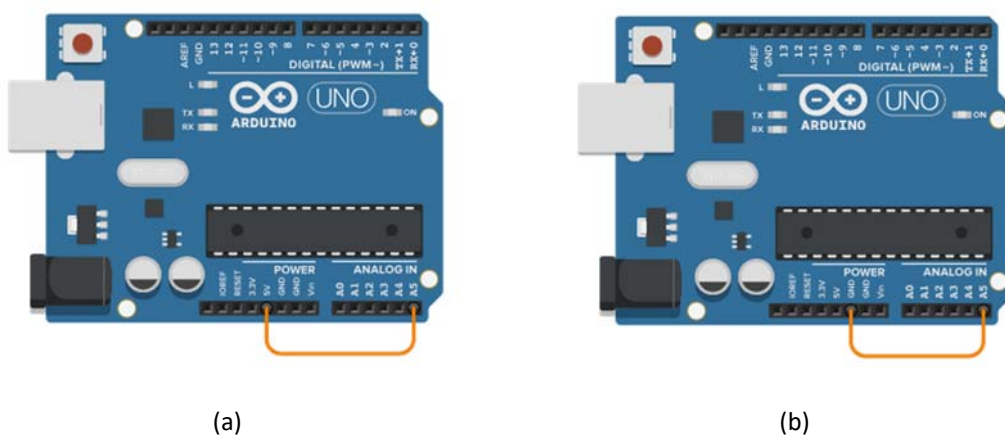




Figura 63 - Diagrama de ligação do Arduino Uno para Desafio 4.

Deligue o fio condutor e vá executando o programa. Verifique que agora aparecem valores distintos, arbitrários e aleatórios, de cada vez que o bloco é executado.

Desafio 5. Leitura do estado de um porto analógico (parte II)

Em vez de estar constantemente a premir a ação de execução, será melhor automatizar este processo fazendo com que a leitura da entrada analógica seja realizada periodicamente a cada

¹² Deve arrastar o bloco “analog reading” para dentro do campo do objeto “diz”.

segundo. Esta alteração é simples de fazer dado que a linguagem **Snap!** possui blocos que permitem realizar ciclos. Neste caso, será selecionado o bloco  juntamente com o bloco  para realizar um ciclo temporizado. A Figura 65 mostra a alteração a fazer ao diagrama de blocos da Figura 62. Observe que, agora, a mensagem apresentada pelo “ator” muda automaticamente a cada segundo.

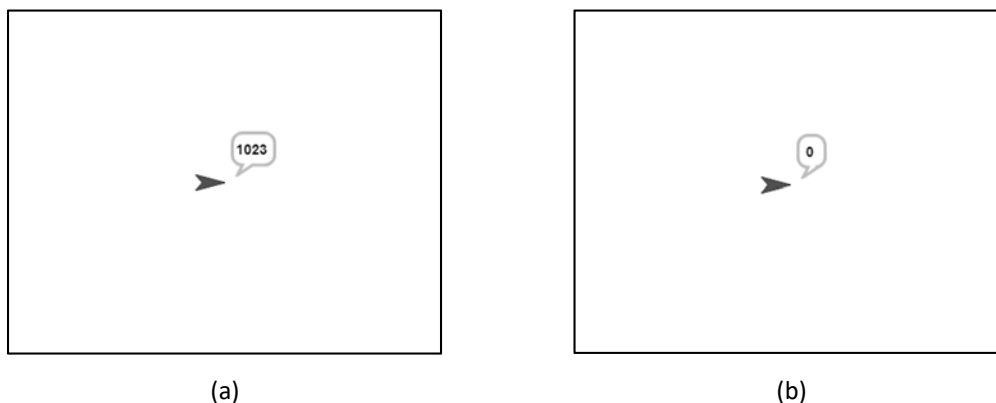


Figura 64 – Resultado da execução do Desafio 4. Em (a) o Resultado obtido quando 5 estiver ligado a 5V e em (b) quando 5 for ligado a GND.

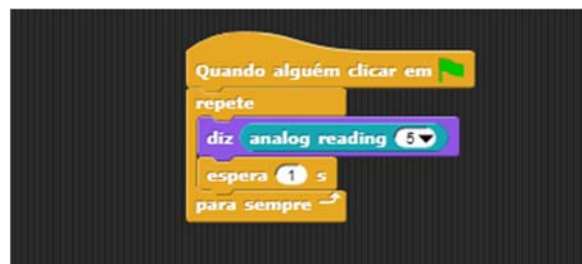


Figura 65 - Diagrama de blocos do Desafio 5.

Com este desafio, encerra-se esta seção onde se apresentou um conjunto de casos de estudo simples com os quais se pretendeu ilustrar a forma como os portos do **Arduino Uno** poderiam ser acedidos e utilizados. Esta seção deve ser repetida as vezes que forem necessárias até que os conceitos que se pretende transmitir fiquem claros. Nomeadamente, a diferença entre portos analógicos e digitais, assim como a diferença entre modo de entrada e de saída. Nas seções que se seguem, apresentam-se desafios de complexidade crescente. Alguns deles envolvem a utilização de funções nativas do **Snap!** que, dado o âmbito deste workshop, apenas serão apresentadas brevemente na seção que se segue.

4.2 - Variáveis, ciclos e condições

Nativamente, o **Snap!** inclui uma panóplia de funções e blocos capazes de controlar atores e palcos. Adicionalmente, e ao contrário do **Scratch**, o **Snap!** permite implementar blocos personalizados. Nesta seção, apresenta-se uma breve descrição dos blocos mais comuns

utilizados na programação em **Snap!** e que serão importantes no desenvolvimento de programas para a plataforma **Arduino Uno**. Neste contexto, começa-se por apresentar a paleta de funções associada à criação de **Variáveis**. Um excerto¹³ desta paleta, ilustrada na Figura 66, permite a criação, destruição e alteração do valor das variáveis.

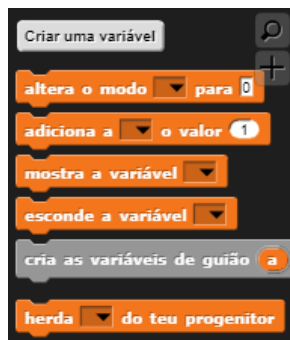


Figura 66 - Excerto dos blocos existentes na paleta de variáveis.

Estas variáveis podem ser manipuladas através de diferentes **Operadores**. Por exemplo, comece por criar três variáveis: **x**, **y** e **z**. Defina **x** com o valor 10 e **y** com o valor 20. O valor de **z** será a função seno da soma de **x** e **y**. A Figura 67 mostra a forma de selecionar a função seno de entre as diferentes funções disponibilizadas. A Figura 68 mostra o **sketch** final que executa a operação pretendida.



Figura 67 - Seleção da função seno.

Tal como qualquer linguagem de programação, o **Snap!** possui elementos para controlo de fluxo. Por exemplo, a existência de blocos que permitem repetir um conjunto de ações um determinado número de vezes.

¹³ Existem também variáveis do tipo lista que não serão abordados neste workshop.

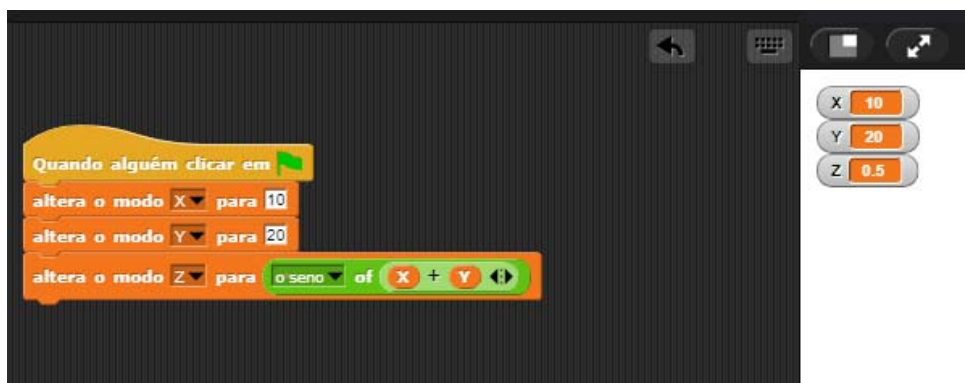


Figura 68 - Diagrama de blocos onde se realiza uma operação algébrica com base no valor existente em duas variáveis.

Estas operações podem ser encontradas na paleta da operação de **Controlo**. No final da seção anterior já se apresentou o exemplo de um bloco que permite a repetição indefinida de um conjunto de ações. No entanto, em determinadas circunstâncias, é necessário que a repetição de ações aconteça um número pré-determinado de vezes ou até que uma dada condição ocorra. Na Figura 69 mostra-se o exemplo de cada um desses dois tipos de ciclos.



(a)



(b)

Figura 69 - Diferentes tipos de ciclos. Em (a) um ciclo que se repete um número pré-determinado de vezes e, em (b), um ciclo que se repete enquanto uma condição se verificar.

A Figura 70 mostra dois exemplos de aplicação onde cada um dos dois tipos distintos de ciclos é utilizado.

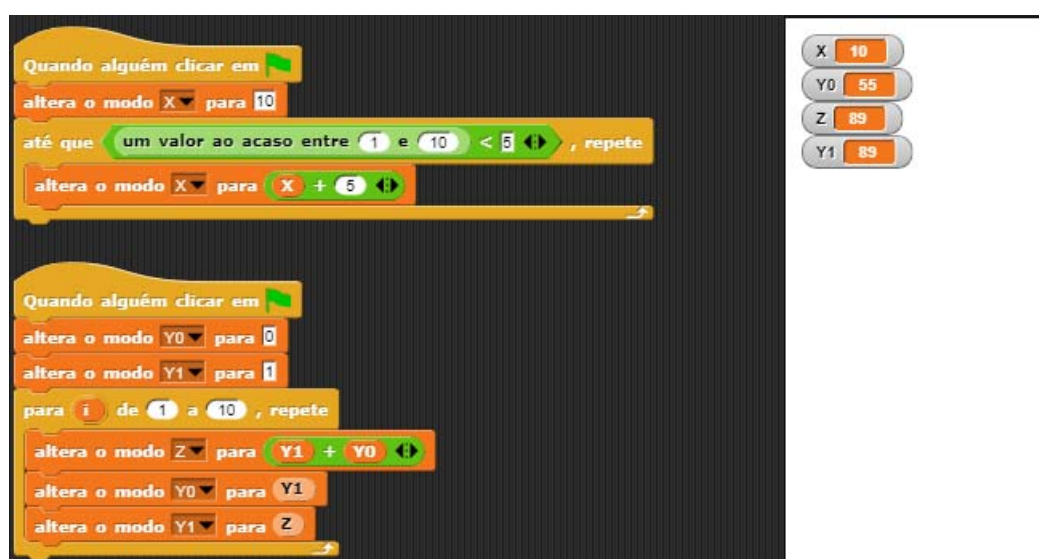


Figura 70 - Exemplos de aplicação dos blocos de execução cíclica.

Finalmente, com frequência, é executar diferentes ações dependendo de um determinado conjunto de condições. O bloco **se... então... senão** permite testar condições e reagir de forma diferente em função de cada situação. A ilustra a sua utilização com um exemplo.



Figura 71 - Exemplo de utilização de condições se... então....

4.3 - Controlando o ator e o palco usando Arduino Uno

Uma das características mais interessantes quando se utiliza o **Snap4Arduino**, quando comparado com outras estratégias de interação com o Arduino Uno, é a facilidade com que se integra a informação proveniente dos portos do microcontrolador e eventuais ações gráficas. Por exemplo, na seção anterior mostrou-se como apresentar, sobre a forma de um **balão de diálogo** associado ao **ator**, o valor adquirido por uma das entradas analógicas do **Arduino Uno**. Nesta seção vamos apresentar mais alguns desafios que envolvem a interação entre o **Snap4Arduino** e o hardware que poderão ser mais tarde utilizados para criar cenas visualmente mais apelativas.

No **Snap4Arduino**, existe um **palco** (quadrado branco no canto superior direito) no centro do qual se encontra o **ator** (com a forma de uma seta preta a apontar para a direita). Por defeito, o palco possui uma dimensão de 480x360 sendo que o seu centro tem a coordenada espacial (0,0). Assim, o canto superior esquerdo encontra-se em (-240,180) e o canto inferior direito em (240,-180).

O ator tem associado três componentes: um **guião**, os **trajes** e **sons**. Por sua vez, e tal como o **ator**, o **palco** tem também três componentes onde, neste caso, o **traje** é substituído pelo cenário. Estas três componentes podem ser alteradas programaticamente recorrendo aos blocos existentes nas áreas de **Movimento**, **Aparência** e **Som**. É de salientar que no **palco** é possível ter mais do que um **ator**, cada um deles com os seus próprios atributos (**guião**, **traje** e **som**).

Para ganhar intuição sobre a forma como controlar os atributos de um **ator**, comece por criar o **guião** (para o **ator**) representado pelo diagrama de blocos da Figura 72. Execute o **guião** e tente entender o papel que cada um dos blocos desempenha no movimento do **ator**.

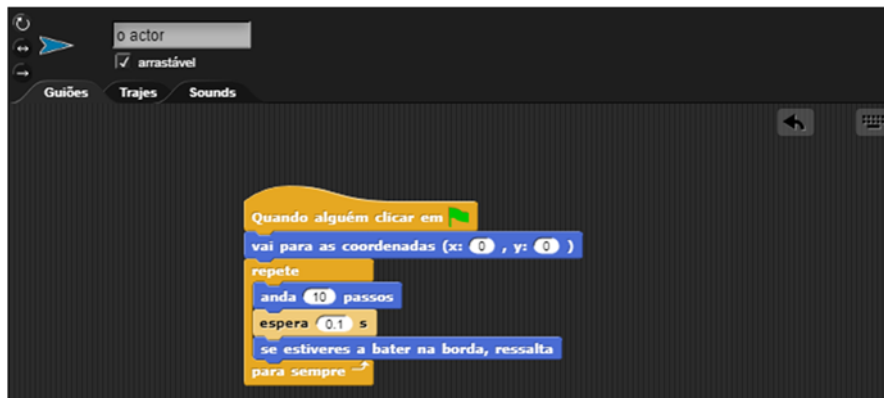


Figura 72 - Movimento do ator com ressalto.

Desafio 6. Movimento de um projétil

Considere que o **ator** é um projétil lançado com uma velocidade inicial v_0 e um ângulo (tomado a partir da horizontal) θ . Pretende-se desenhar um **sketch** considerando as seguintes equações da cinemática:

$$\begin{cases} x = x_0 + v_0 \cos(\theta)t \\ y = y_0 + \tan(\theta)(x - x_0) - \frac{1}{2}g\left(\frac{x - x_0}{v_0 \cos(\theta)}\right)^2 \end{cases} \quad (1.1)$$

Para isso, são criadas 8 variáveis que serão responsáveis por registar os valores da posição do corpo ao longo do tempo assim como outros parâmetros necessários para a simulação tais como a aceleração da gravidade, o ângulo e posições iniciais. A Figura 73 mostra o diagrama de blocos associado à simulação do lançamento de um projétil.

Como se sabe, e na ausência de outras forças sobre o corpo, a trajetória do projétil é parabólica. Essa forma pode ser registada através da utilização da **Caneta**. O diagrama de blocos da Figura 74 mostra o perfil da trajetória parabólica traçada usando a caneta.

O **traje** que o **ator** possui por defeito pode não estar devidamente adaptado ao contexto. Por exemplo, imagine-se que se pretendia que o aspeto do **ator** fosse uma bola vermelha e não uma seta. É possível definir múltiplos **trajes** para um **ator** e altera-los dinamicamente. A forma de o fazer envolve aceder ao separador **traje** do **ator** e carregar, ou desenhar, os **trajes** que se entenderem necessários.

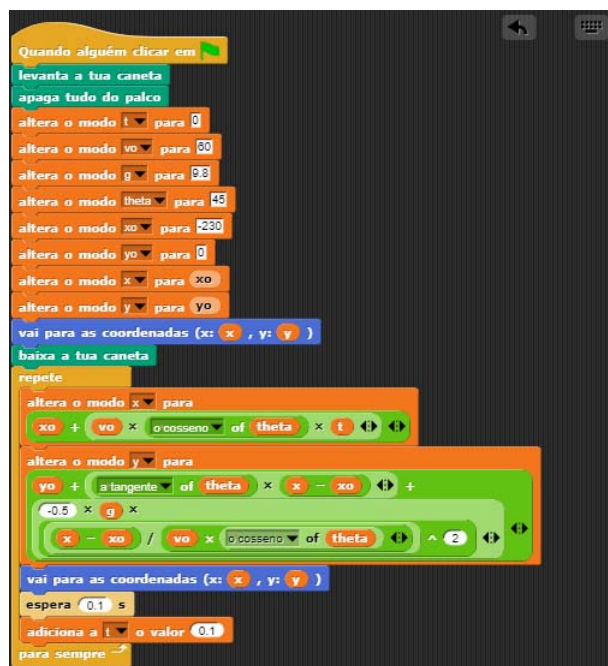


(a)

vo 60 yo 0 xo -230 y 89.632834 x -17.86796 theta 45 g 9.8
t 5

(b)

Figura 73 - Simulação do lançamento de um projétil.



(a)

vo 60 yo 0 xo -230 y 89.632834 x -17.86796 theta 45 g 9.8
t 5

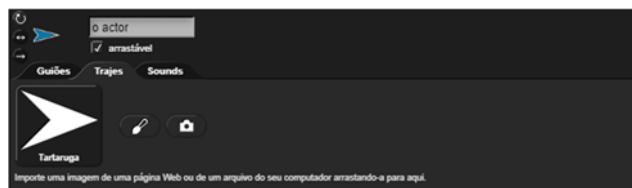
(b)

Figura 74 - Simulação do lançamento do projétil e representação da sua trajetória usando a caneta.

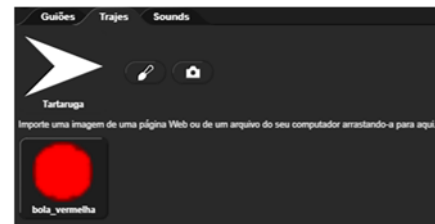
Comece por aceder ao **repositório** do workshop, na pasta **Imagens**, subpasta **Bolas**, descarregue a imagem associada à [bola vermelha](#). A seguir, utilizando o rato, arraste a imagem da área de Transferências para o separador **Trajes** como se mostra na Figura 75

Agora, adicione o bloco **muda o traje para** ao diagrama de blocos da Figura 74 e escolha como parâmetro **bola_vermelha**. A Figura 76 mostra o resultado final do diagrama de blocos. Depois

destas alterações, realize a simulação e verifique que agora o **ator** tem o aspeto de uma bola vermelha.

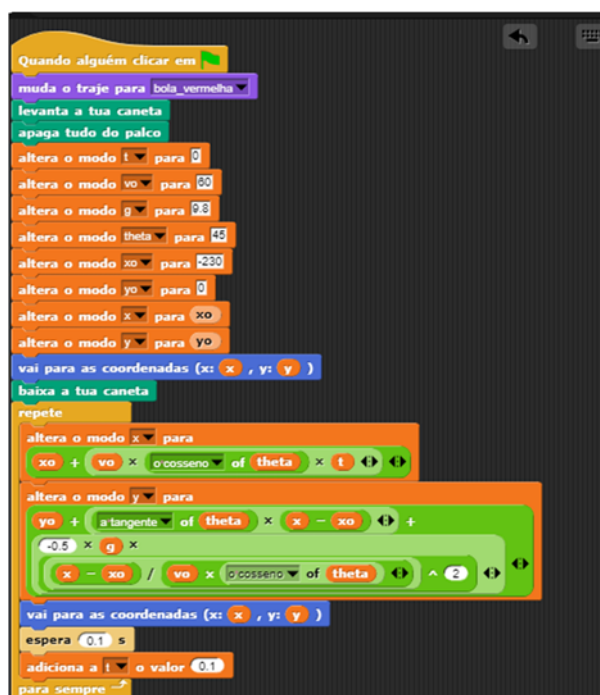


(a)



(b)

Figura 75 - Upload de um traje a partir de uma imagem.



(a)



(b)

Figura 76 - Simulação do lançamento de um projétil utilizando um traje diferente para o ator.

TPC 1. Altere o diagrama de blocos de modo que a bola não passe abaixo da linha $y=0$.

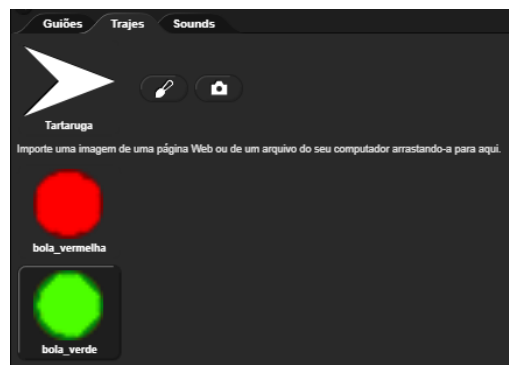
Desafio 7. Alteração do traje do ator com I/O digital

No início deste capítulo, viu-se como controlar o estado de um LED usando uma entrada digital do Arduino Uno. Neste desafio a ideia é utilizar o mesmo porto para controlar o **traje** de um **ator**. Em particular, um determinado **ator** terá dois **trajes** distintos, um com a forma de uma bola vermelha e outro com a forma de uma bola verde. Ambas as imagens se encontram no

repositório (diretório **Imagens/Bolas**), devem ser descarregadas e colocadas no separador de **trajes** do **ator**. Crie o sketch que se apresenta na Figura 77 e teste o seu funcionamento.



(a)



(b)

Figura 77 - Trajes e guião associado ao ator do desafio 7.

Desafio 8. Representação gráfica de um sinal sinusoidal.

Neste desafio pretende-se representar um sinal sinusoidal com a forma:

$$y(t) = A \cdot \sin(\omega \cdot t + \varphi) \quad (1.2)$$

A amplitude, frequência e fase serão variáveis cujos valores podem ser alterados pelo utilizador, em tempo real, usando *sliders*. Estas variáveis devem estar balizadas pelos seguintes valores: $A \in [10, 100]$, $\omega \in [0.5, 5]$ e $\varphi \in [-90, 90]$. Comece por criar cinco variáveis com os nomes **y**, **t**, **phi**, **omega** e **A**.

Essas variáveis aparecem no palco e devem ser parametrizadas de acordo com o que foi dito anteriormente. Para isso, clique com o botão direito do rato sobre cada uma das variáveis **phi**, **omega** e **A**. Desta ação, será aberta uma janela com um conjunto de diferentes opções como se mostra na Figura 78. Para cada uma das três variáveis defina o formato como “potenciômetro” e parametrize os valores mínimo e máximo de acordo com os limites dos respetivos intervalos.

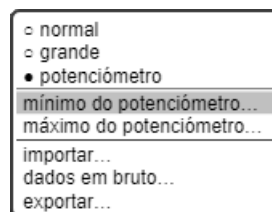


Figura 78 - Menu associado às variáveis que se pretendem alterar dinamicamente.

O passo seguinte consiste em alterar o **cenário** do **palco** e o **traje** do **ator**. No repositório do workshop poderá encontrar, no diretório **Imagens/Gráficos**, um ficheiro com nome

ponteiro.png e outro com nome `quadriculado_invertido.png`. Descarregue ambos e associe o primeiro ao **traje** do **ator** e o segundo ao **cenário** do **palco**. O passo seguinte, consiste em construir o **guião** do **ator** de acordo com o diagrama de blocos da Figura 79.

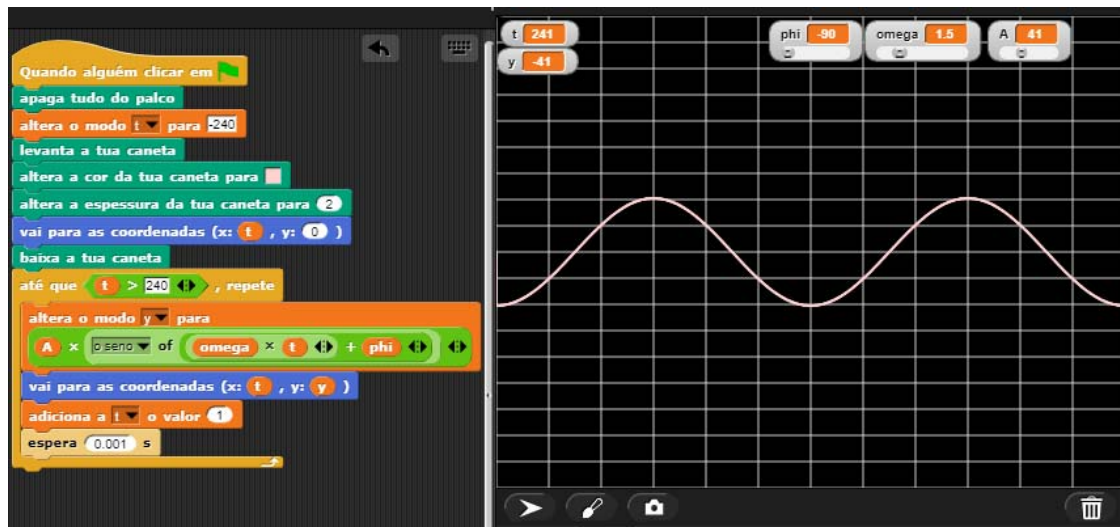


Figura 79 - Diagrama de blocos associado ao Desafio 8 e respetivo resultado da execução.

Desafio 9. Osciloscópio com Arduino Uno (parte I).

Neste novo desafio, o objetivo é construir um osciloscópio rudimentar com o **Arduino Uno**. Este osciloscópio será capaz de mostrar a forma da onda, no domínio do tempo, do sinal aplicado à entrada analógica 5. Neste caso, e como se mostra na Figura 80, será realizado um mapeamento entre os limites fornecidos pela entrada analógica (0 e 1023) e os limites da janela de visualização (-180 a 180).

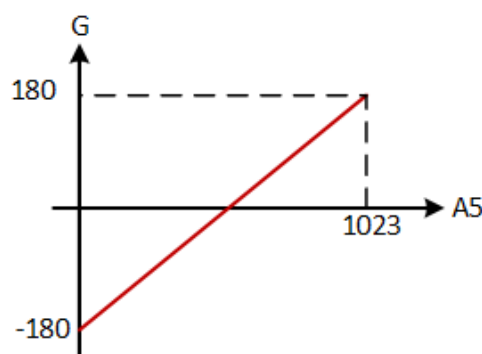


Figura 80 - Mapeamento entre o valor inteiro fornecido pelo Arduino Uno e a dimensão da janela de visualização.

Este mapeamento é obtido através da seguinte equação da reta:

$$G = \frac{360}{1023} A_5 - 180 \quad (1.3)$$

No cenário a utilizar para o palco, deverá descarregar o ficheiro **osciloscópio_invertido.png** do repositório. Neste cenário, cada divisão, ao longo das ordenadas, é constituída por 36 pixéis. Como 360 pixéis vão corresponder a 5V então cada divisão da grelha corresponde a 0.5V. Ou seja, o osciloscópio apresenta uma resolução vertical de 0.5V/div. Com base nesta informação, construa o diagrama de blocos da Figura 81.

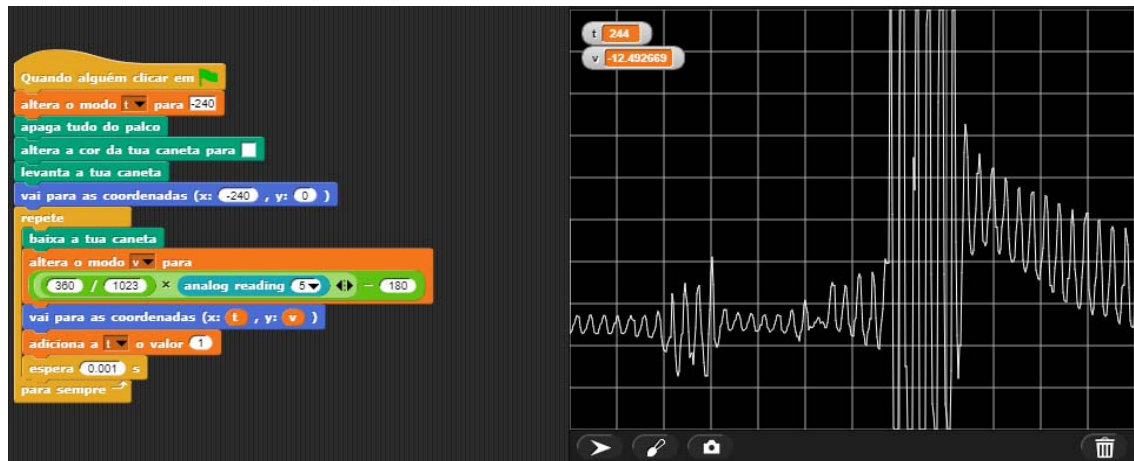


Figura 81 - Diagrama de blocos e exemplo de resultado do Desafio 9.

Neste exemplo, apenas tem lugar um único varrimento. No desafio que se segue o diagrama de blocos será alterado de modo a permitir varrimentos contínuos, como acontece num osciloscópio real.

Desafio 10. Osciloscópio com Arduino Uno (parte II).

Neste desafio será utilizada uma resistência variável (**potenciômetro**) ligada à entrada **5** do **Arduino Uno** como se mostra na Figura 82.

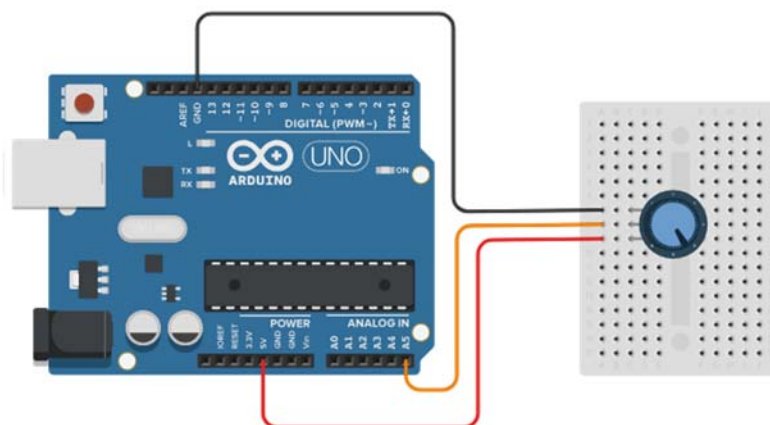


Figura 82 - Ligação de um potenciômetro à entrada analógica 5.

O diagrama de blocos da Figura 81 é alterado como se mostra na Figura 83. As alterações permitem o varrimento contínuo da tela do osciloscópio e acrescentam um balão informativo com o valor da tensão presente na entrada 5 (aproximado às décimas).

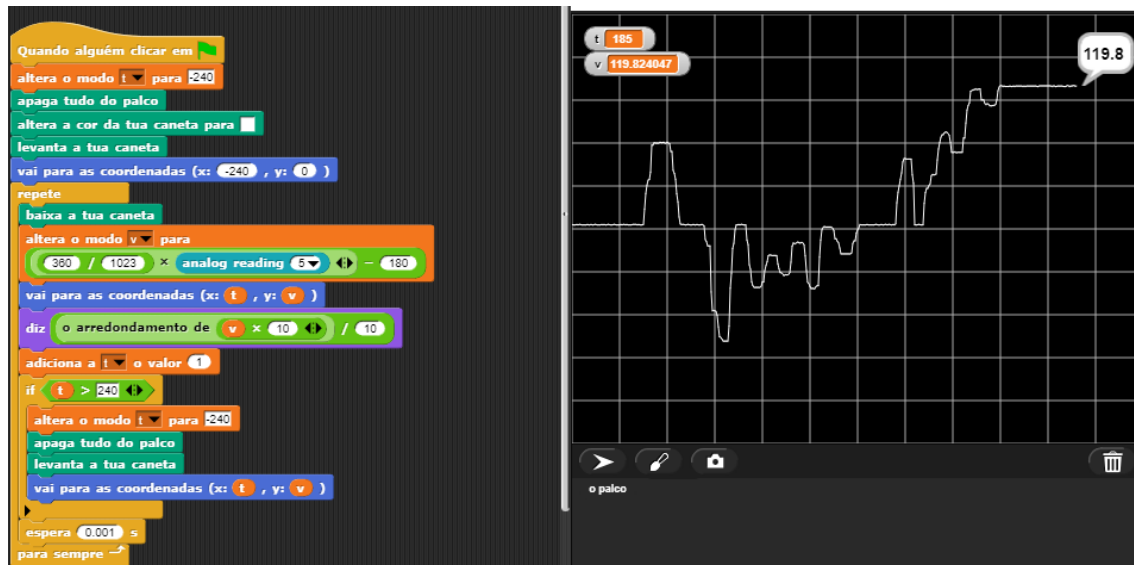


Figura 83 - Diagrama de blocos associado ao Desafio 10.

Execute o **sketch** e gire o **potenciômetro** para um e outro lado observando como o traço, na janela de visualização, se altera em função disso.

Em ciência, é muitas vezes necessária a medição de diferentes variáveis físicas. O Arduino Uno é uma plataforma que permite, através da interligação com **shields** e outros dispositivos eletrônicos, a medição de um elevado número de grandezas como é o caso da temperatura, humidade do ar, aceleração, etc. Na próxima parte, serão apresentados alguns desafios que envolvem a interligação do Arduino Uno com sensores. □

Parte 5: Interface com o mundo físico usando o Arduino Uno

5.1 - Medição da iluminância

Neste primeiro ensaio, será utilizado a *breakout board*¹⁴ com referência KY-018¹⁵ constituído por uma resistência dependente da luz¹⁶ (LDR) em série com uma resistência de polarização de 10kΩ. A Figura 84a mostra o aspeto da breakout board e a Figura 84b o esquema elétrico equivalente.

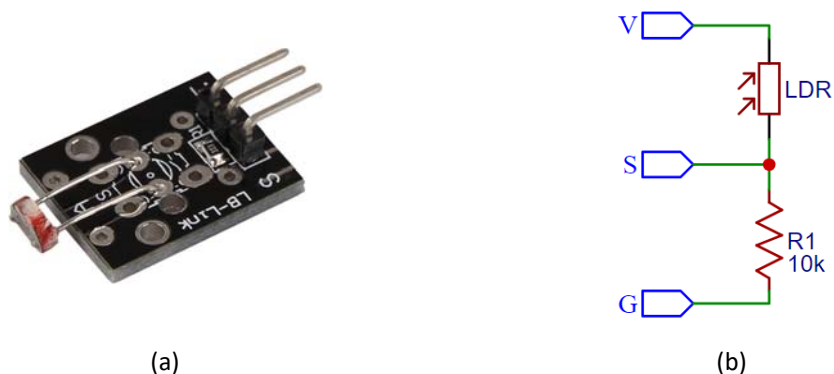


Figura 84 - Em (a) aspeto da breakout board KY-018 e em (b) o esquema elétrico equivalente.

Um LDR é sensível à iluminância convertendo variações dessa grandeza em variações de resistência elétrica. A iluminância é medida, no sistema internacional de unidades, em **lux** (lx) e descreve a forma como o fluxo luminoso (medido em **lúmen**) se dispersa por uma determinada área. Em particular, 1 lx é igual a 1 lm sobre uma área de 1m². De modo a ganhar-se intuição acerca desta unidade, na Tabela 1 apresenta-se na uma lista com os valores de iluminância típica em diferentes contextos ao longo de um dia sem nebulosidade.

Tabela 1 - Valores típicos de iluminância para diferentes instantes ao longo de um dia.

Contexto	Iluminância/lx
Lua cheia (num dia sem nebulosidade)	1
Por ou nascer do Sol (num dia sem nebulosidade)	400
Dia ensolarado (sem nebulosidade)	120,000

A relação entre a resistência entre os dois terminais de um LDR e a iluminância é uma função não linear cuja curva de calibração é apresentada na Figura 85. Esta curva pode ser aproximada pela equação:

¹⁴ Uma *breakout board* é uma placa de circuito impresso (PCB) que simplifica a conexão de componentes ou dispositivos eletrônicos, fornecendo conectores ou terminais acessíveis para ser interligada a outras PCB.

¹⁵ À data deste workshop, os participantes têm acesso a um kit de sensores como [este](#).

¹⁶ Também designada por fotorresistência.

$$R_{LDR}(\phi) = R_0 \cdot \phi^{-\gamma} \quad (1.4)$$

A aproximação desta função à curva de calibração leva a que $R_0 = 130k\Omega$ e $\gamma = 0.7785$.

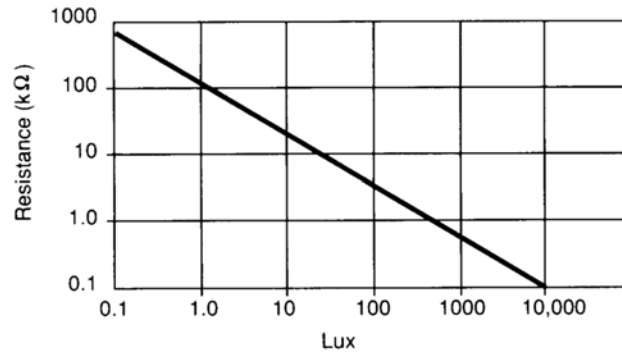


Figura 85 - Curva de calibração de um LDR típico (adaptado de RS datasheets, 232-3816, Março de 1997).

O **Arduino Uno** não é capaz de medir diretamente a resistência elétrica. Na verdade, a resistência elétrica é estimada, de forma **indireta**, a partir da medição da diferença de potencial entre os terminais **S** e **G** do esquema da Figura 84b admitindo uma diferença de potencial igual a **5V** entre **V** e **G**. Nesse caso, a diferença de potencial entre **S** e **G** é igual à queda de tensão na resistência de polarização que facilmente se deriva a partir das leis de *Ohm* e *Kirchhoff* como sendo:

$$U_{SG} = \frac{R_{10k}}{R_{10k} + R_{LDR}(\phi)} \times 5 \quad (1.5)$$

Onde U_{SG} se refere à queda de tensão na resistência de 10kΩ.

O **Arduino Uno** é capaz de medir, de forma direta, U_{SG} . Utilizando a expressão anterior é possível estimar o valor de $R_{LDR}(\phi)$ e, com base na equação (1.4), determinar a iluminância. Neste quadro de referência,

$$\begin{cases} R_{LDR}(\phi) = \frac{R_{10k}}{U_{SG}} 5 - R_{10k} \\ \phi = \left(\frac{R_{LDR}(\phi)}{R_0} \right)^{-\left(\frac{1}{\gamma}\right)} \end{cases} \quad (1.6)$$

Desafio 11. Construção de um luxímetro

Neste desafio o objetivo é utilizar o conhecimento apresentado anteriormente para criar um luxímetro usando a plataforma **Arduino Uno** e o **Snap4Arduino**.

Com o **Arduino Uno desligado** da porta USB do computador, faça as ligações entre a *breakout board* e a plataforma como se mostra na Figura 86.

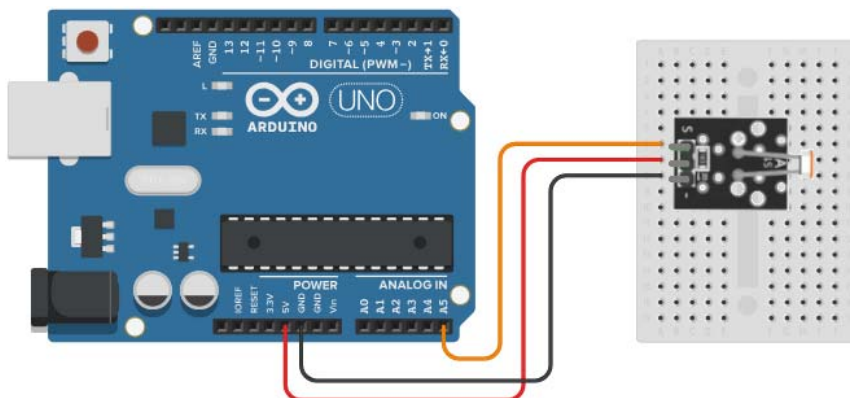


Figura 86 - Diagrama para ligação do KY-018 ao Arduino Uno.

Em seguida, ligue o **Arduino Uno** à porta USB e ative a comunicação com o **Snap4Arduino**. Posteriormente, construa o diagrama de blocos representado na Figura 87 e execute o programa.

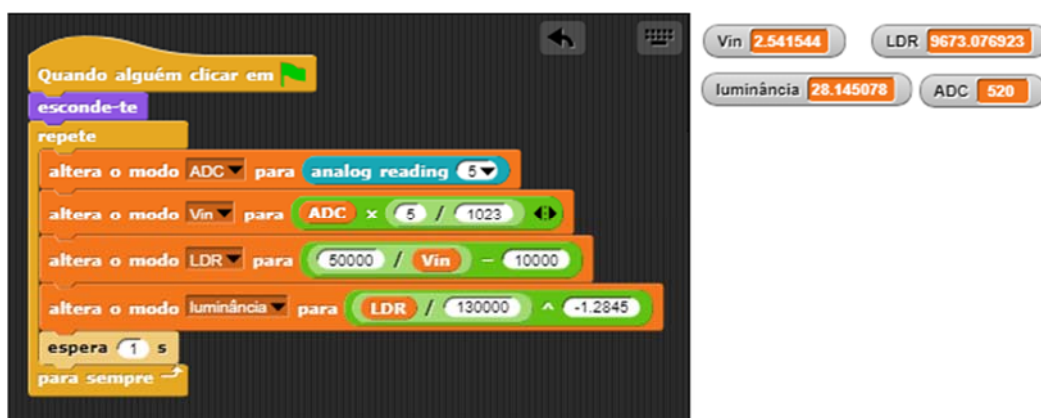


Figura 87 - Diagrama de blocos do Desafio 11.

Comprove o seu funcionamento dirigindo o LDR para diferentes fontes luminosas.

5.2 - Medição da temperatura do ar

A temperatura está diretamente relacionada com a energia cinética dos átomos que formam a matéria. Neste segundo ensaio, a *breakout board* com referência KY-013, constituído por um termistor NTC¹⁷ em série com uma resistência de polarização de 10kΩ, será utilizada para a construção de um termómetro.

A Figura 84a mostra o aspeto da *breakout board* e a Figura 84b o esquema elétrico equivalente.

¹⁷ NTC – Coeficiente de temperatura negativo.

Um termistor é um sensor resistivo com dois terminais. A resistência elétrica entre esses terminais é dependente da temperatura de acordo com a equação:

$$R(T) = R_0 \exp \left(\beta \left(\frac{1}{T} - \frac{1}{T_0} \right) \right) \quad (1.7)$$

onde T se refere à temperatura absoluta em *kelvin*, T_0 uma temperatura de referência (normalmente 298.15 K correspondendo a 25°C) e R_0 a resistência do termistor à temperatura de referência T_0 . O coeficiente β representa a sensibilidade do sensor à temperatura. Para o KY-013, $\beta = 3950$ e $R_0 = 10k\Omega$ e a gama de medida estende-se de -55°C a 125°C.

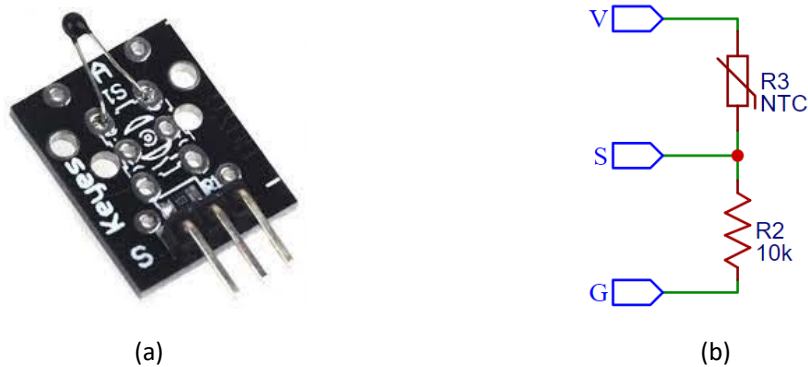


Figura 88 - Em (a) aspeto da breakout board KY-013 e em (b) o esquema elétrico equivalente.

Tal como aconteceu no desafio anterior, o Arduino Uno será utilizado para medir a diferença de potencial aos terminais da resistência R2 e, com base nesse valor, determinar o valor da resistência do termistor. Tendo em consideração a relação entre a resistência do termistor e a temperatura, conhecendo a resistência permite estimar o valor da temperatura.

Desafio 12. Construção de um Termómetro

Neste desafio o objetivo é utilizar o KY-013 para criar um termómetro usando a plataforma **Arduino Uno** e o **Snap4Arduino**. Tal como aconteceu no desafio anterior, comece por **desligar** o **Arduino Uno** da porta USB do computador e faça as ligações entre a *breakout board* e a plataforma como se mostra na Figura 89.

Em seguida, depois de ligar o Arduino Uno ao computador e ativar a sua ligação com o snap4arduino, construa o diagrama de blocos representado na Figura 90 e execute o programa.

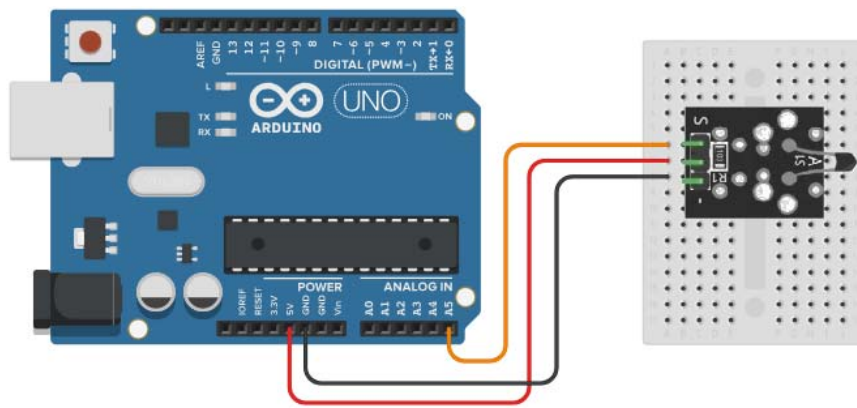


Figura 89 - Diagrama para ligação do KY-013 ao Arduino Uno.

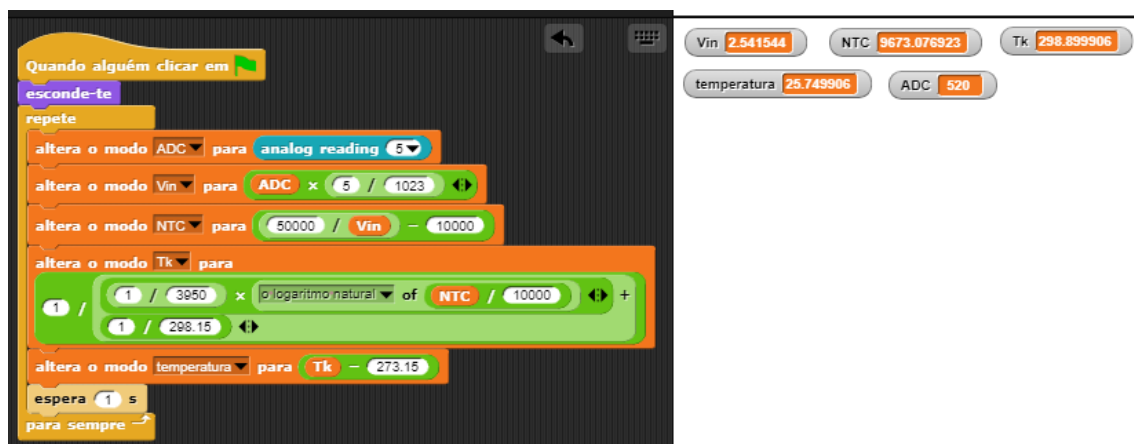


Figura 90 - Diagrama de blocos do Desafio 12.

Parte 6: Conclusão

Como foi dito logo no início, o principal objetivo deste workshop foi aprender a utilizar o **Arduino Uno** sem toda a bagagem de programação em texto estruturado como normalmente acontece. Em particular, e devido ao fato do **Scratch** ser uma linguagem já ensinada no ensino básico/secundário, considerou-se que a sua transposição para a programação da plataforma **Arduino Uno** seria vantajosa no que se refere à redução do impacto necessário para aprender uma linguagem de programação comum como é o caso do **C/C++**. No entanto, há que deixar claro que a facilidade que o **Snap4Arduino** oferece tem um preço a pagar. Por exemplo, muitas das suas funções encontram-se significativamente limitadas e há a necessidade constante de manter a comunicação com o computador. No entanto, apesar de todas estas restrições, acredita-se que este primeiro contacto com a programação de microcontroladores acabe por despertar a curiosidade e, com isto, que a vontade se abra para a aprendizagem de outras formas menos restritivas de lidar com o **Arduino Uno**.

Espera-se que este workshop, antes de tudo, seja um catalisador para que os docentes comecem a ver potencialidade na integração da plataforma **Arduino Uno** como ferramenta pedagógica. Para além disso, que os exemplos de digitalização que foram realizados, possam eventualmente ser replicados, adaptados e alterados para poderem funcionar em sala de aula. Nada seria mais gratificante do que ver este conhecimento conduzido até aos vossos alunos e integrado como ferramenta de aprendizagem nas diferentes disciplinas. No final de contas, dada a sua versatilidade, este **bicho de 7 cabeças** é mais (muito mais), o **homem dos 7 instrumentos**.