



Workshop B > Aviso à Navegação

Adelina Machado, Pedro Coimbra – ESERO Portugal/Ciência Viva

André Breda – IST



O Kit

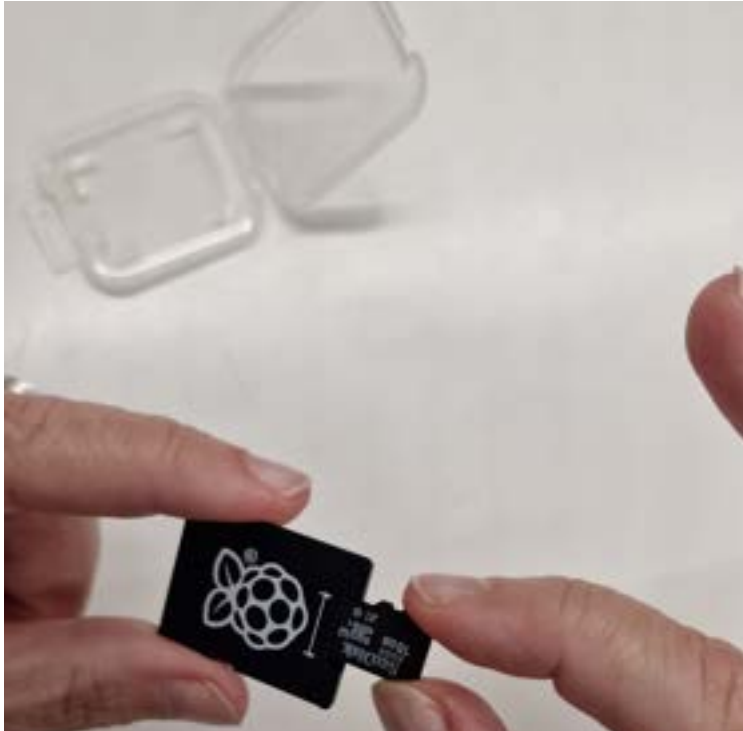


Composição:

1. Raspberry Pi 4 (4GB)
2. Cabo mini HDMI - HDMI
3. Cartão microSD com adaptador para SD
4. Fonte de alimentação para Raspberry Pi USB-C
5. *SenseHAT* (placa de expansão com conjunto de sensores e atuadores)
6. Componentes estruturais (8 parafusos + 4 espaçadores) e barra de pinos **curta**
7. Lente para câmara
8. Pi Camera HD
9. 3 Jumpers fêmea-fêmea
10. Barra de pinos **comprida**
11. Sensor de luz infravermelha (PIR)

Montagem do Kit

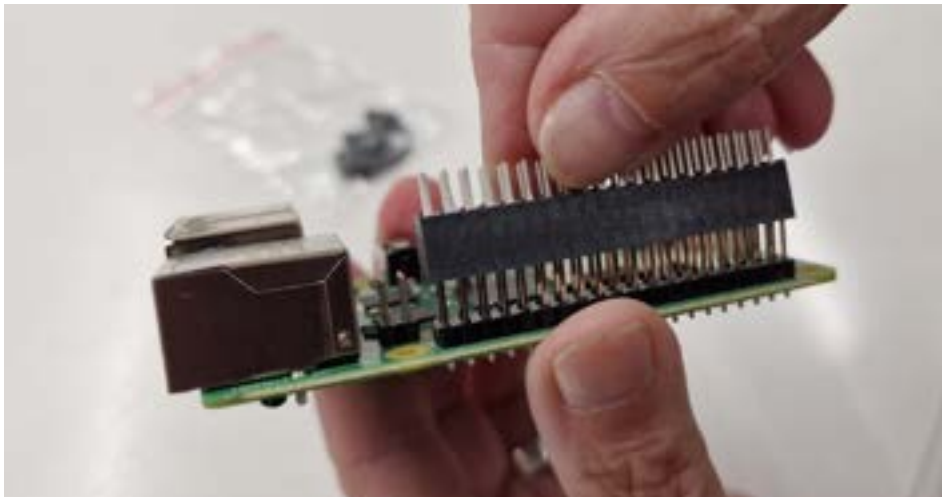
1. Separe o cartão microSD do adaptador



2. Insira o cartão microSD no RaspberryPi

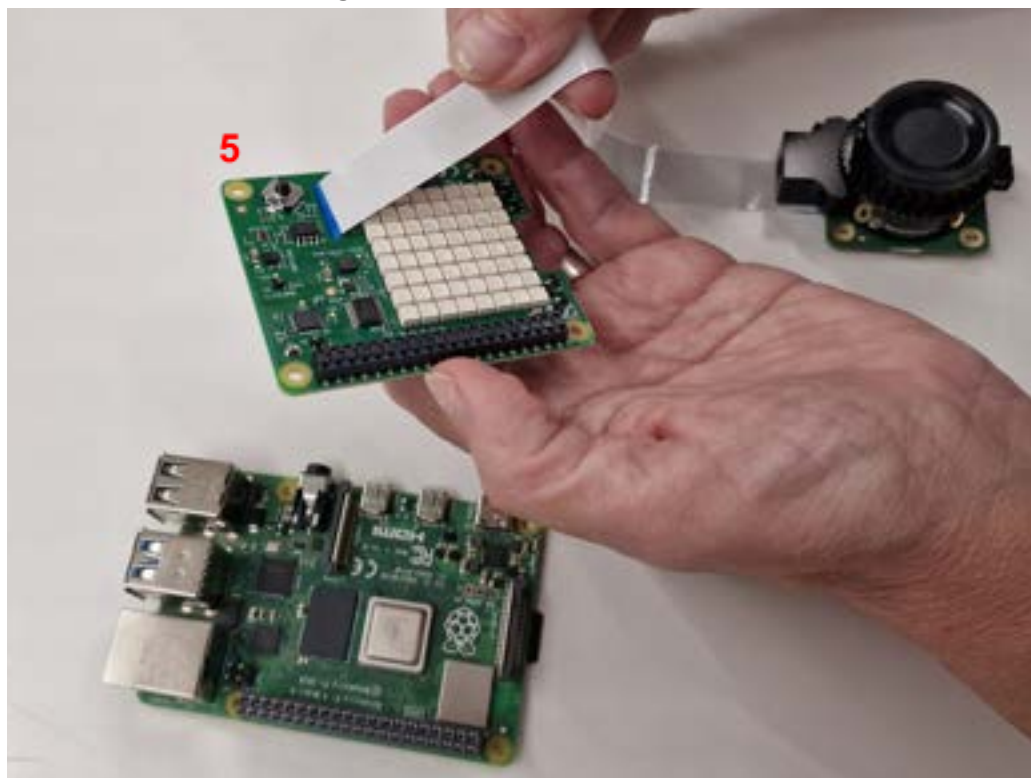


3. Encaixe a barra de pinos **curta** no Raspberry Pi.
Pode ter de empurrar ligeiramente de um lado de cada vez para conseguir encaixá-la totalmente.

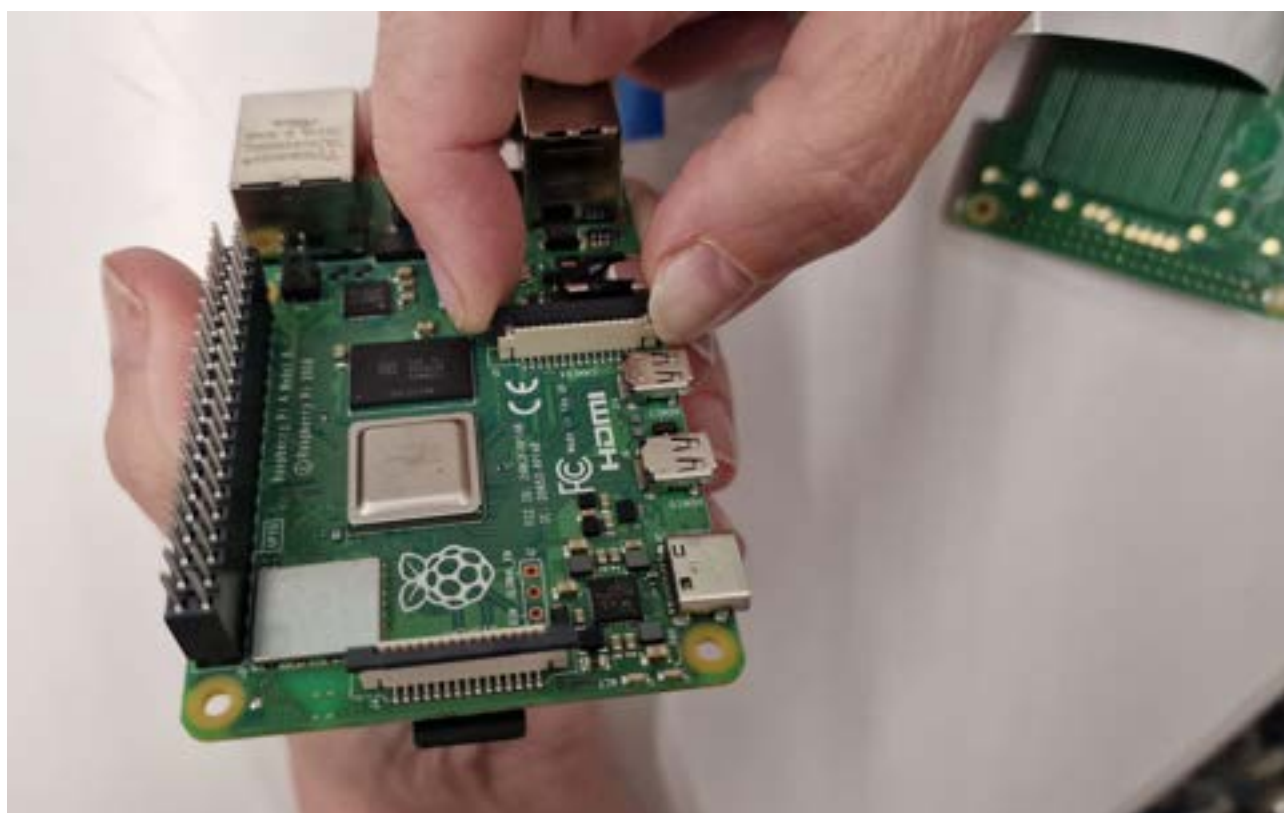


4. Passe o cabo da câmara pela ranhura presente no *SenseHAT* (identificado com “5” na imagem).
Garanta que o lado azul do cabo fica do lado oposto à matriz de LEDs (grelha de objetos brancos) como na imagem.

Cuidado: **este cabo é frágil**

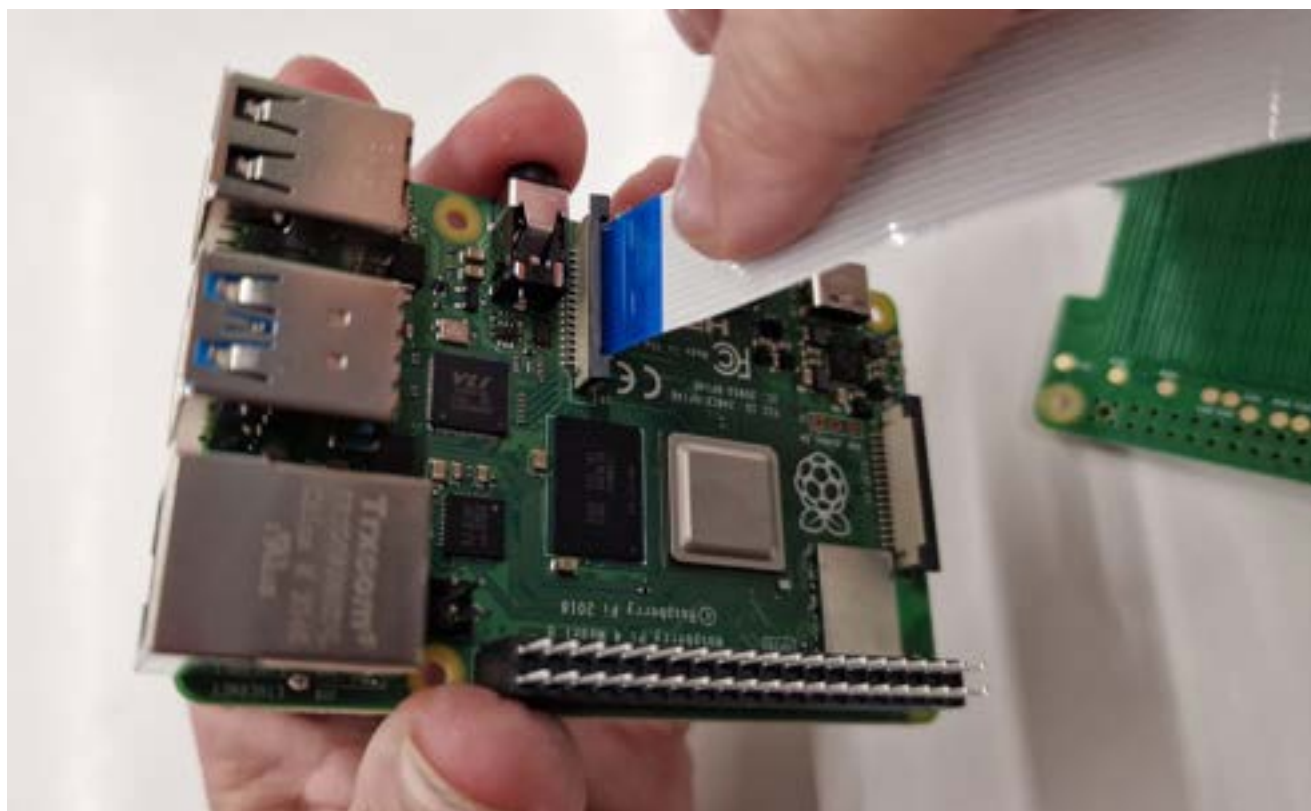


5. No Raspberry Pi, levante a patilha preta no conector da câmara como na imagem.
A patilha NÃO sai totalmente. Basta estar ligeiramente elevada de ambos os lados.
É mais fácil soltar um lado de cada vez antes de levantar.

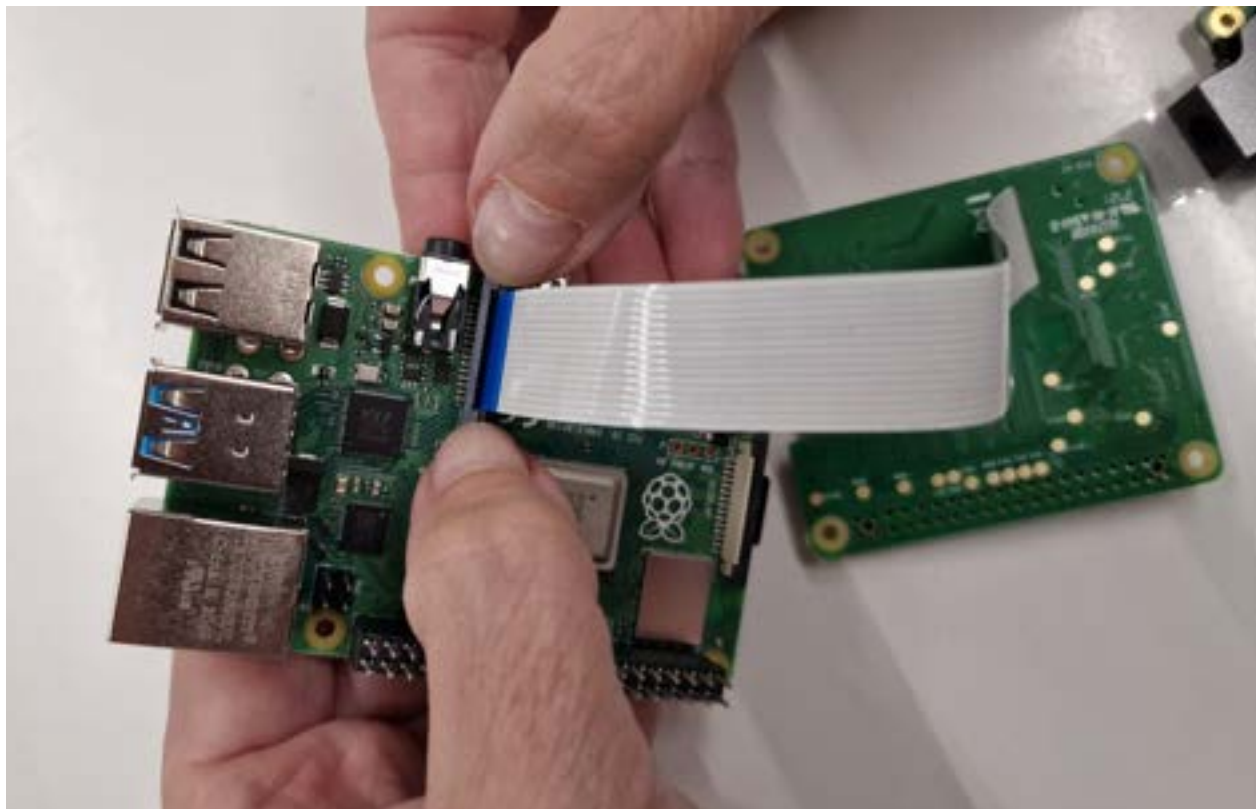


6. Ligue **sem usar força** o cabo da câmara que atravessou o *SenseHAT* no passo 4 ao conector respectivo no RaspberryPi.

É normal o cabo **NÃO entrar totalmente** no conector. Pode usar o lado não-azul como guia, e confirmar que os contactos metálicos estão só ligeiramente expostos como na segunda imagem abaixo. A película azul **NÃO** sai.

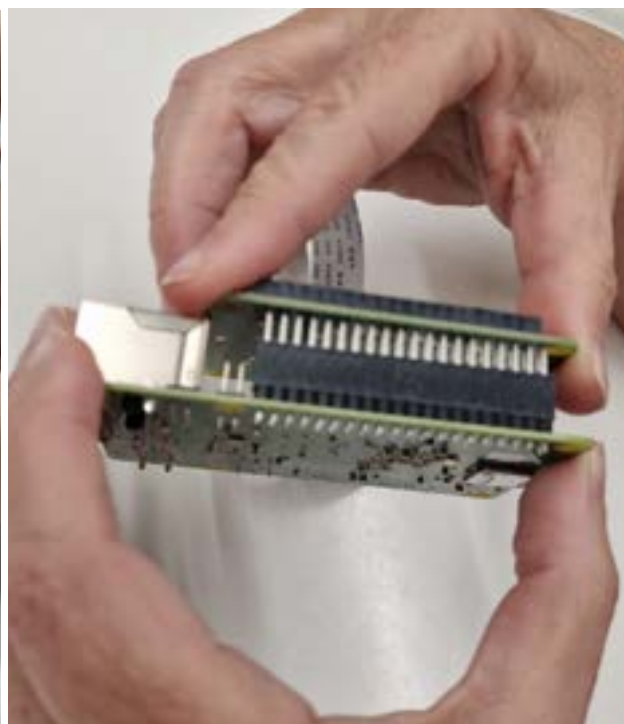


7. Pressione **nas pontas** da patilha preta do conector da câmara como na imagem para prender o cabo.



8. Encaixe o *SenseHAT* no Raspberry Pi com a matriz de LEDs voltada para cima. Deverá ouvir um clique.

Tome o cuidado de **não dobrar e trilhar o cabo da câmara entre o *SenseHAT* e o Raspberry Pi**.



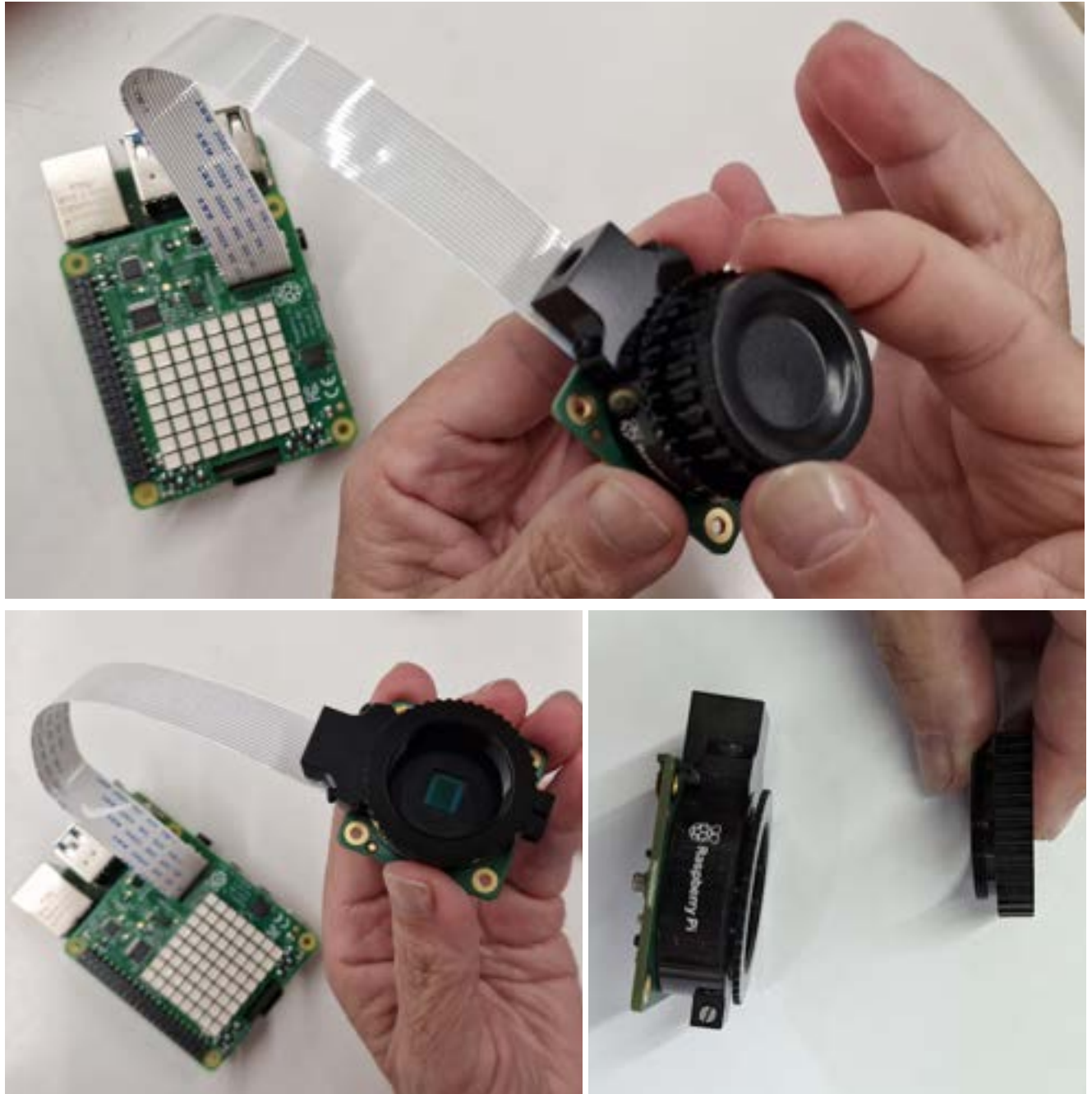
9. Pressione o conector preto no topo do *SenseHAT* para garantir que está totalmente encaixado.

10. Desenrosque a tampa e o adaptador de lente do módulo da câmara, rodando a roda dentada mais alta como na imagem.

O sensor da câmara vai ficar exposto. Mantenha-o virado para baixo e fora de contacto com outros objetos para não se sujar.

Garanta que o adaptador de lente é removido como mostrado na terceira imagem. Caso não seja removido, será impossível obter uma imagem focada com a lente do kit.

Note que há uma **pequena roda dentada no módulo da câmara**: esta serve para ajustar o foco e **deve permanecer ligada** ao módulo, mesmo que não totalmente enroscada.



11. Desenrosque a tampa da traseira da lente da câmara, rodando a proteção do lado que contém as marcas “NEAR / FAR” como na imagem.



12. Enrosque a lente da câmara no módulo da câmara.



13. A tampa da lente da câmara que resta não tem rosca e pode ser simplesmente retirada e reposta.

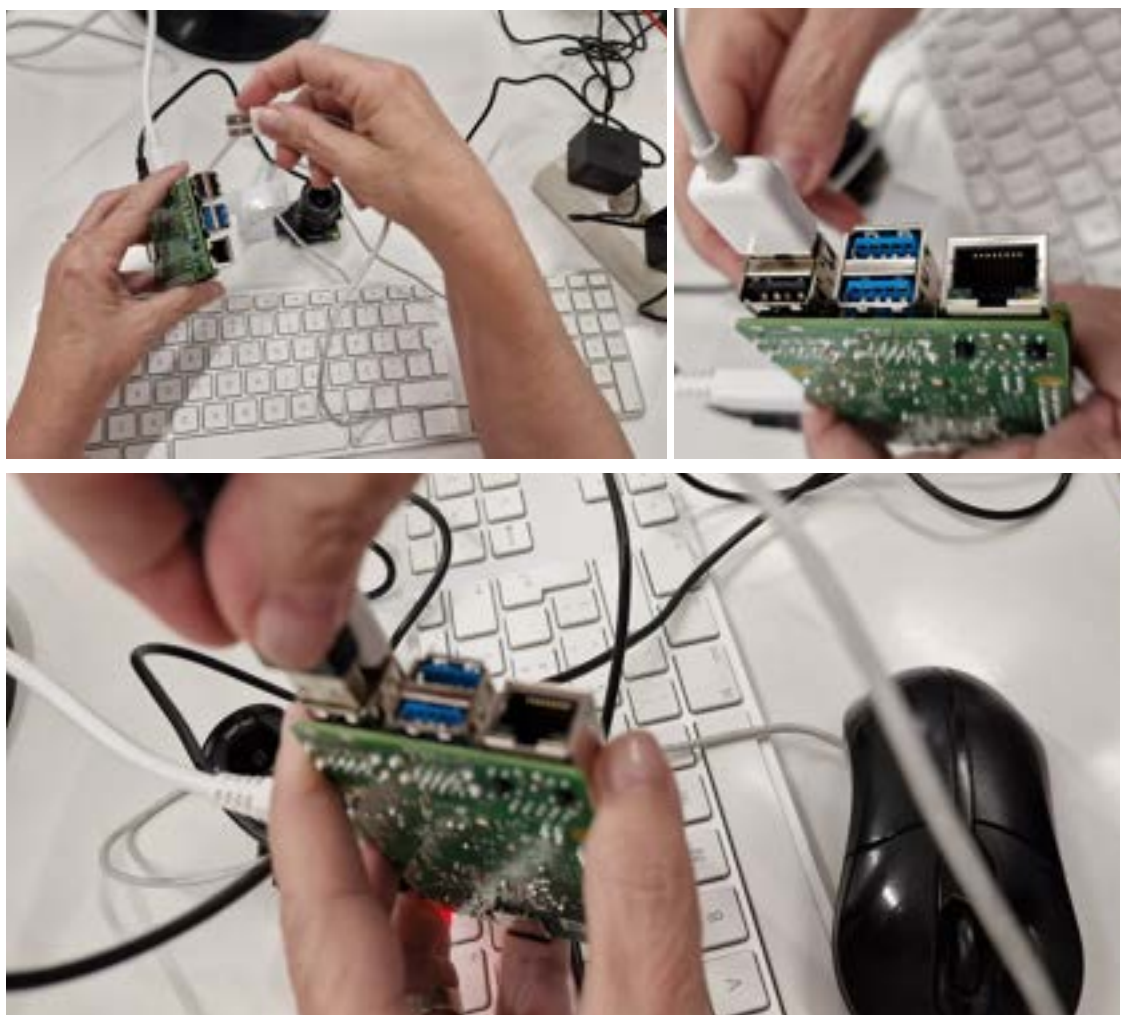
14. Ligue o cabo mini HDMI-HDMI ao Raspberry Pi. Pode ser ligado em duas portas, qualquer uma serve.



15. Ligue a outra ponta do cabo mini HDMI-HDMI ao seu monitor.



16. Ligue o rato e o teclado ao Raspberry Pi.

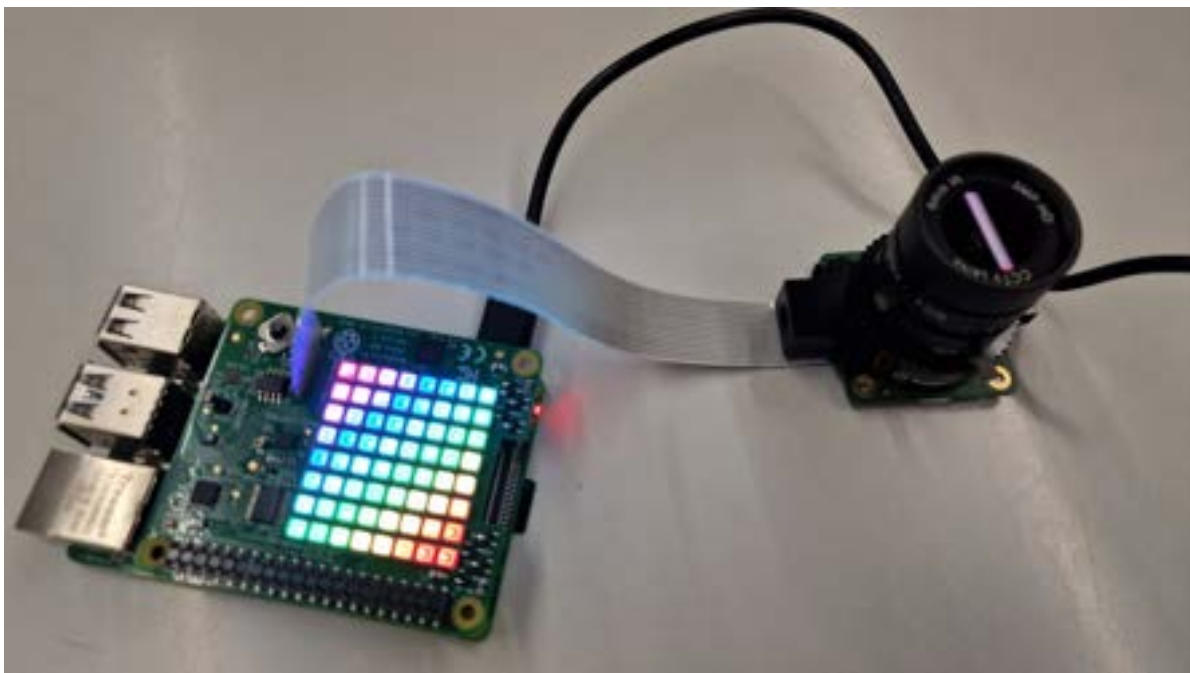


17. Ligue o cabo de alimentação ao Raspberry Pi, e ligue a fonte de alimentação a uma tomada.

A matriz de LEDs do *SenseHAT* vai mostrar um padrão arco-íris.

Passados alguns segundos, deverá ter imagem no monitor.

Nota: caso tenha ligado o Raspberry Pi à energia antes do monitor, vai ter de o reiniciar para que o monitor seja detetado. Desligue-o da energia, espere que as luzes estejam todas apagadas, e volte a ligar.



18. Caso não consiga escrever caracteres acentuados ou a letra “c” com cedilha, é provável que o teclado esteja configurado como Inglês (EUA)

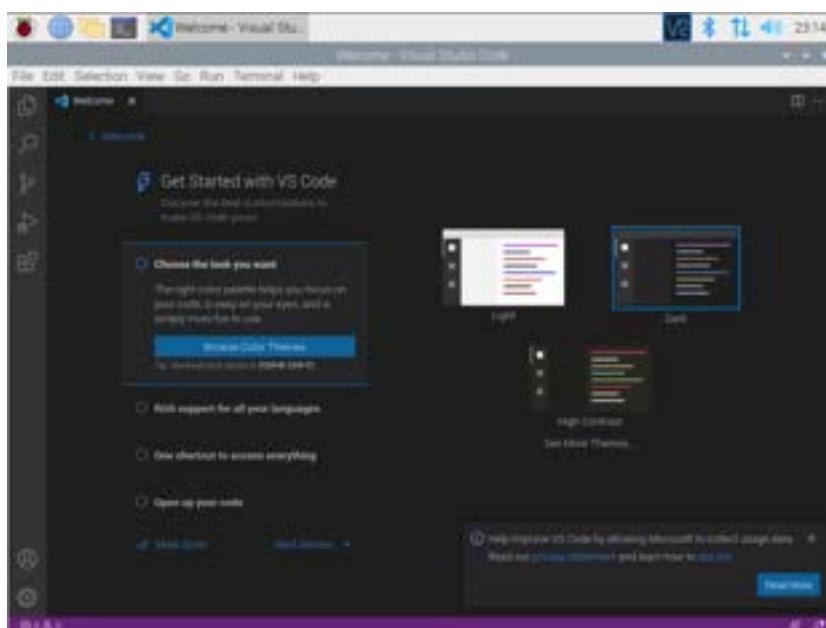
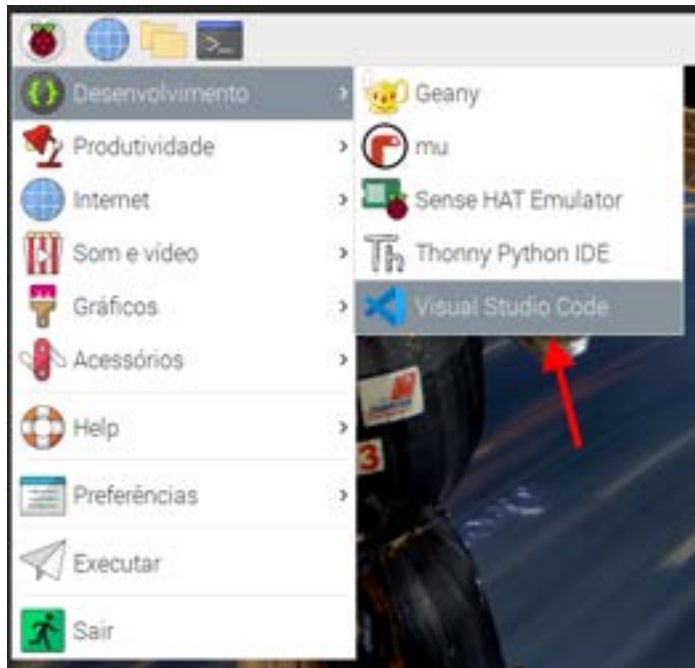
Para mudar para o esquema de teclado português, clique na framboesa do canto superior esquerdo, e escolha Preferências > Teclado e Rato. Navegue para a aba “Teclado”, e clique no botão “Keyboard Layout...”.

Selecione as opções “Model: Generic 105-key PC (intl)”, “Layout: Portuguese” e “Variant: Portuguese”.

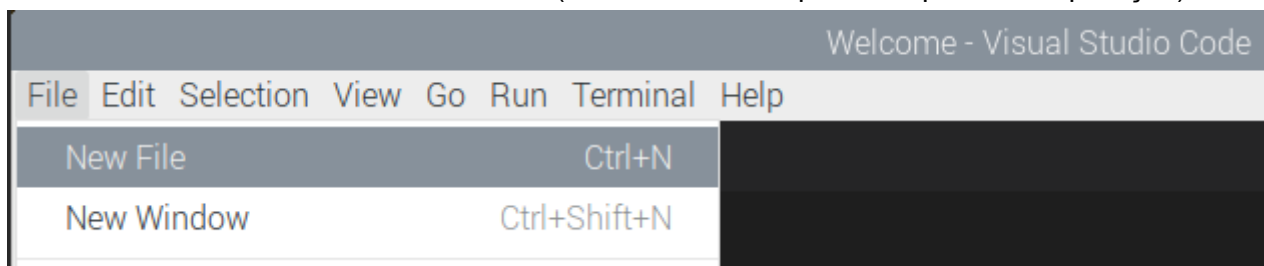
Programação

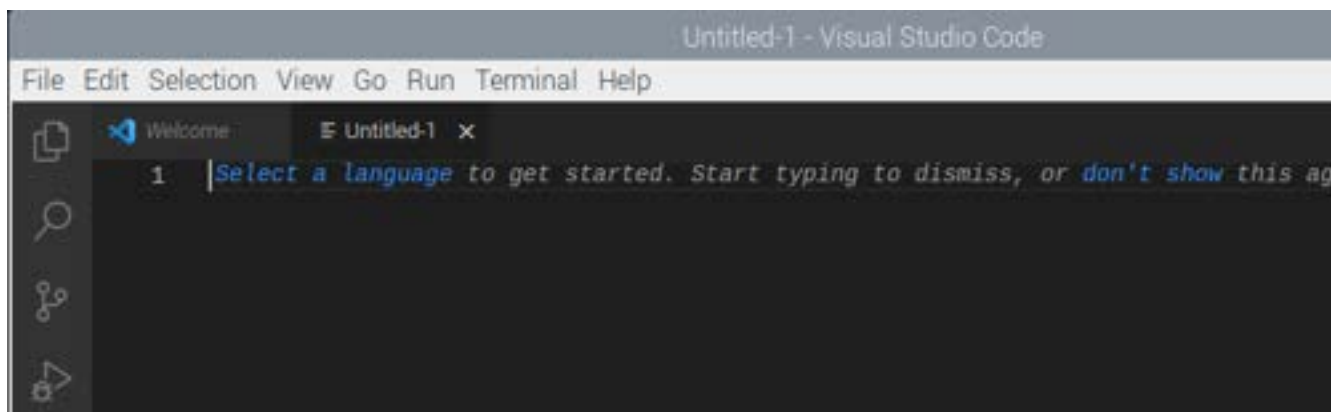
Mostrar Texto no *SenseHAT*

1. Abra um editor de texto (p.ex. Visual Studio Code) usando o menu do canto superior esquerdo.

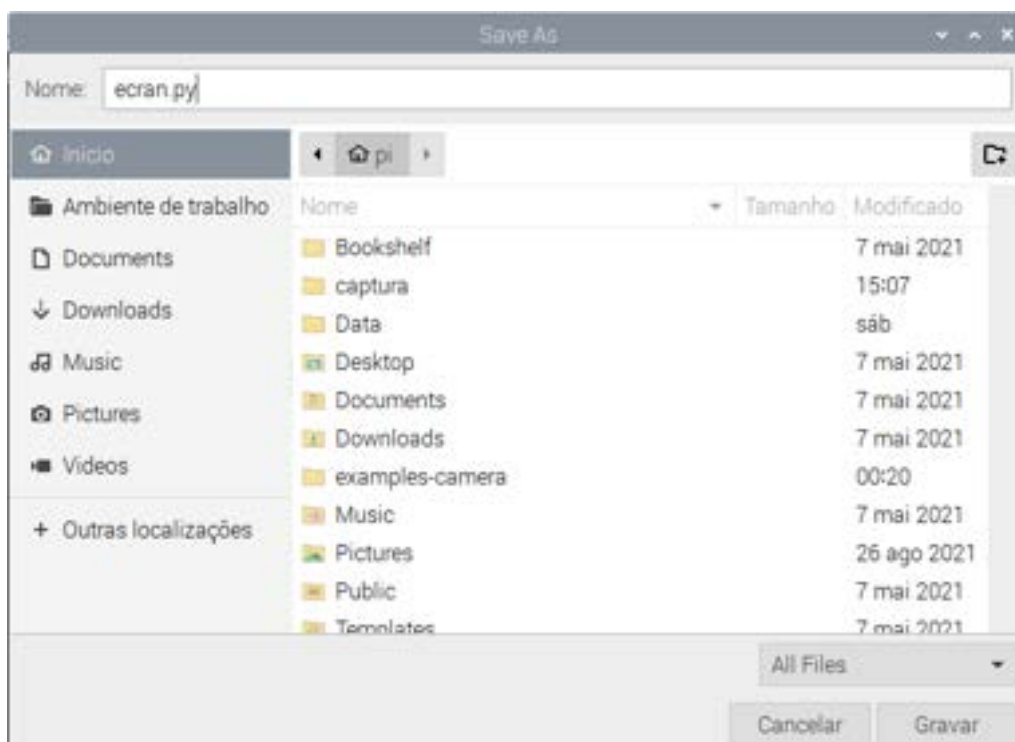
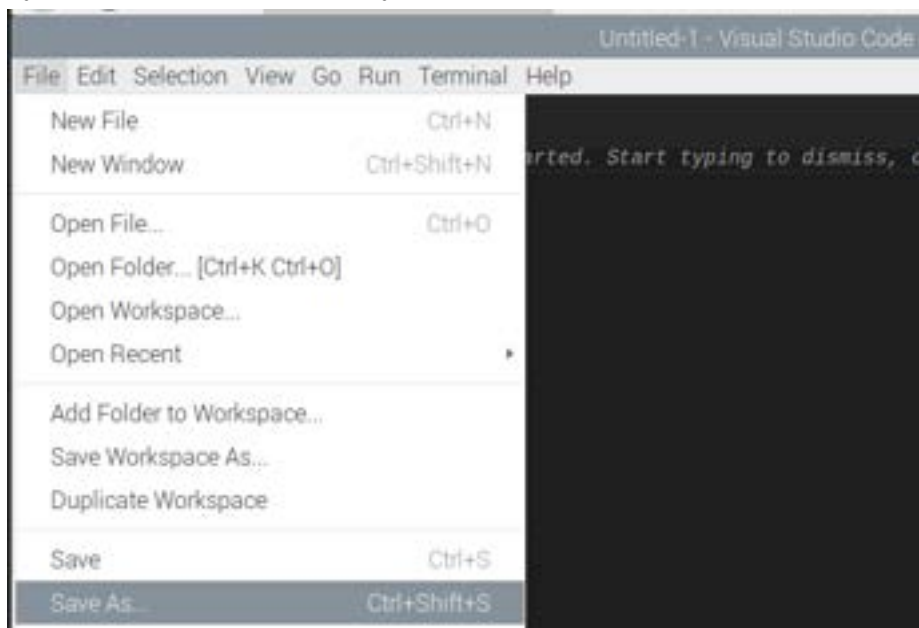


2. Crie um novo ficheiro com *File > New File* (menu no canto superior esquerdo da aplicação)



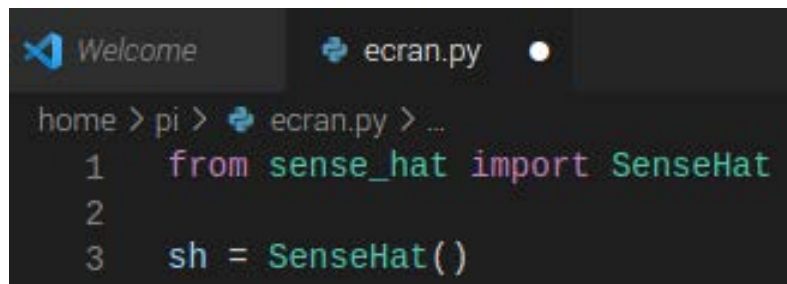


3. Vamos já guardar o ficheiro vazio com um nome específico com *File > Save As...*
O importante aqui é que o nome do ficheiro termine em “.py” para ser reconhecido pelo editor como Python. Vamos usar “ecran.py”.



4. Vamos programar! Começemos por importar a biblioteca do *SenseHAT* (linha 1) e criar uma variável “sh” para o representar (linha 3).

Dica: caso as sugestões de código estejam a substituir “SenseHat” por “sense_hat” prima a tecla ESC para fechar as sugestões sem as aplicar



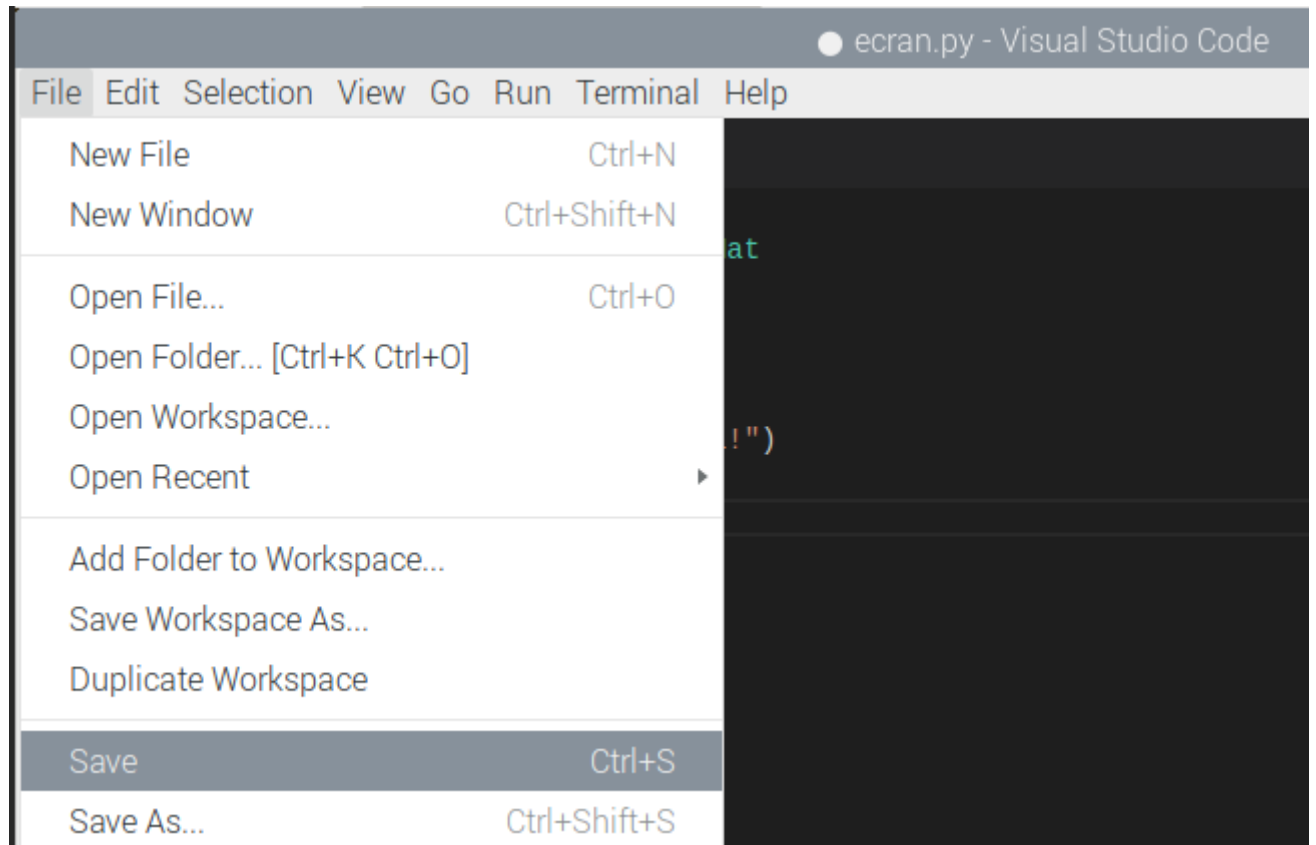
```
home > pi > ecran.py > ...
1  from sense_hat import SenseHat
2
3  sh = SenseHat()
```

5. Agora, vamos fazer o *SenseHAT* mostrar a mensagem “Bom dia!” indefinidamente. Para isso vamos usar o método *show_message* em ciclo infinito.



```
home > pi > ecran.py > ...
1  from sense_hat import SenseHat
2
3  sh = SenseHat()
4
5  while True:
6      sh.show_message("Bom-dia!")
7
```

6. Vamos guardar o nosso programa com *File > Save* (ou premindo as teclas *Ctrl* e *S*).



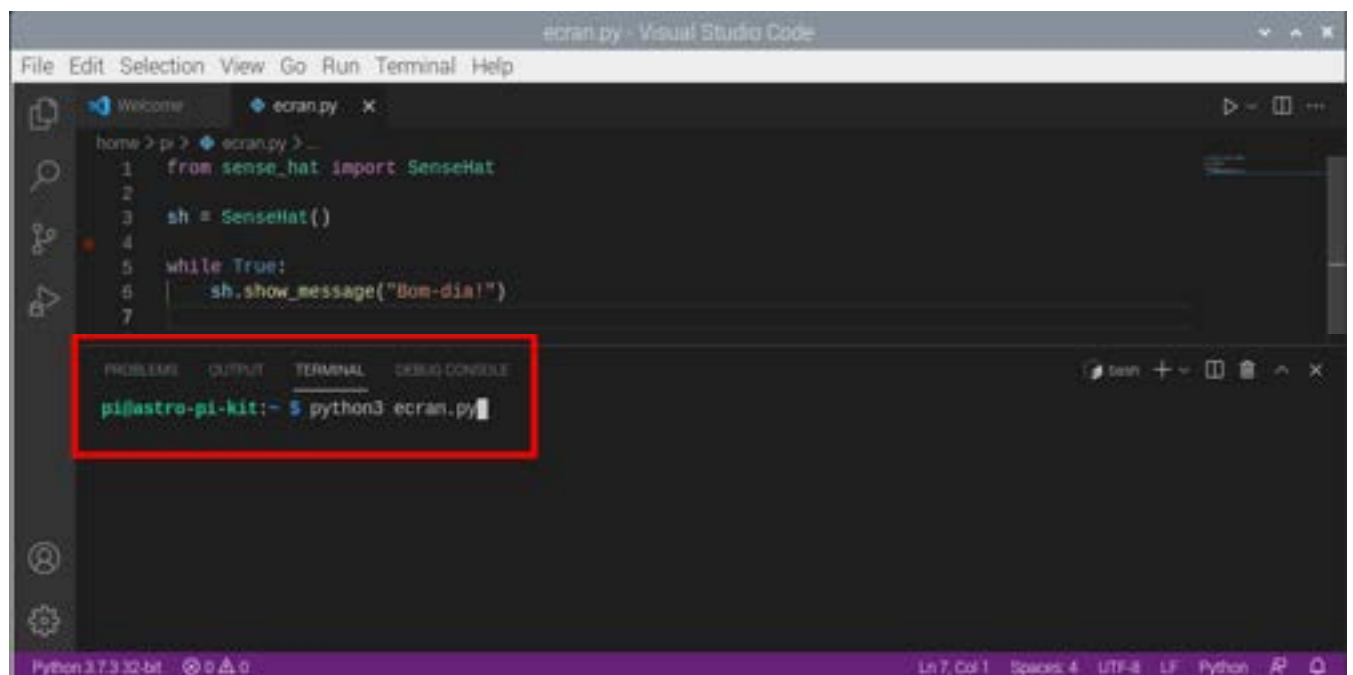
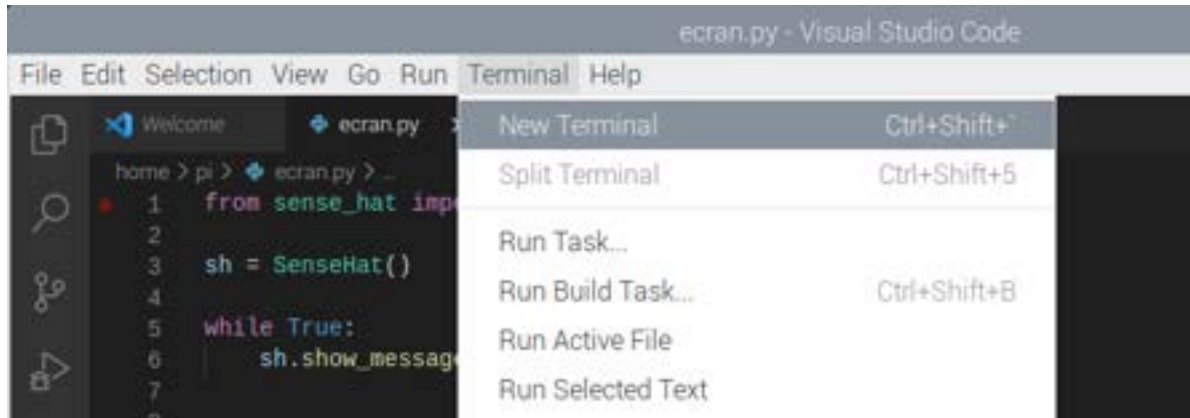
A bola branca antes do nome do ficheiro no topo do ecrã vai desaparecer.

7. Para correr o nosso programa e mostrar a mensagem, vamos abrir um terminal em *Terminal > New Terminal*, escrever “python3 ecran.py” (sem as aspas) e premir a tecla ENTER.

Nota: caso o ficheiro tenha outro nome (p.ex. “outro.py”), deve refletir isto no comando (“python3 outro.py”).

Nota: Na linha de comandos o cursor é um bloco (█). Caso veja apenas os contornos (□), é sinal que o foco não está no terminal: basta clicar nele com o rato para o focar e poder escrever comandos.

Dica: Premir a tecla TAB manda o computador completar o nome de comandos e ficheiros na linha de comandos.



8. Para parar o programa, prima as teclas Ctrl e C.

A prompt (`pi@astro-pi-kit:~ $`) vai ser novamente escrita no terminal, e o cursor colocado no fim, à espera do comando seguinte.

É normal aparecer um erro “KeyboardInterrupt” e um “Traceback” antes: esta combinação de teclas interrompe o programa de forma súbita.

Note que os LEDs do *SenseHAT* ficaram congelados no estado em que estavam quando parou o programa.

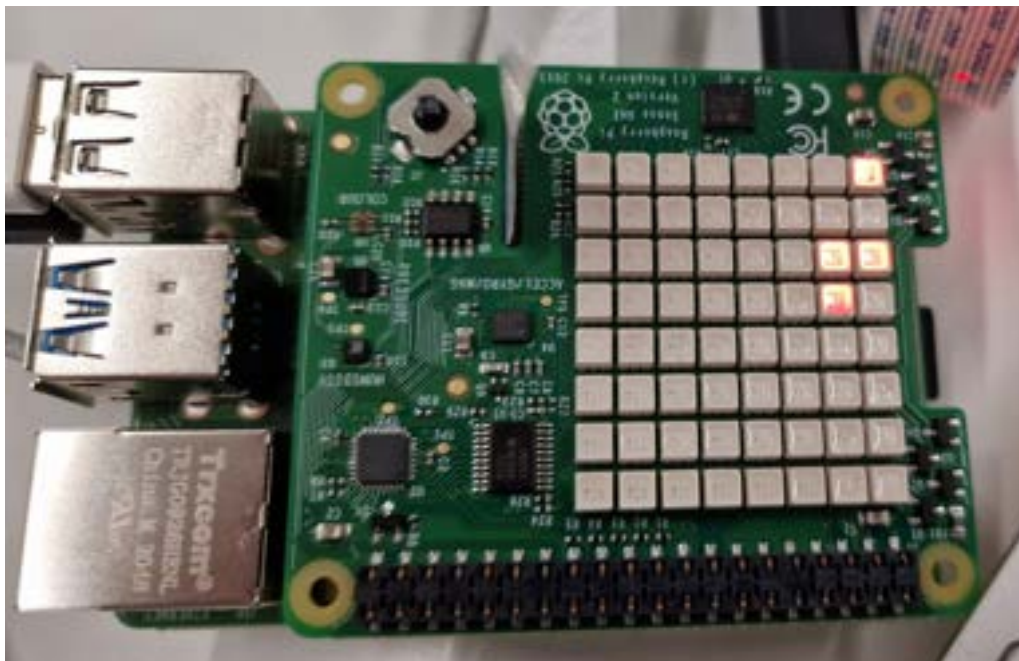
9. Os LEDs do *SenseHAT* são RGB, isto é, conseguem mostrar cor. Vamos mudar a cor do nosso texto para vermelho.

Para isto vamos criar uma variável RED para representar a cor vermelho, e usá-la quando mostramos o texto:

```
home > pi >  ecran.py > ...  
1  from sense_hat import SenseHat  
2  
3  sh = SenseHat()  
4  RED = [255, 0, 0]  
5  
6  while True:  
7      sh.show_message("Bom-dia!", text_colour=RED)  
8
```

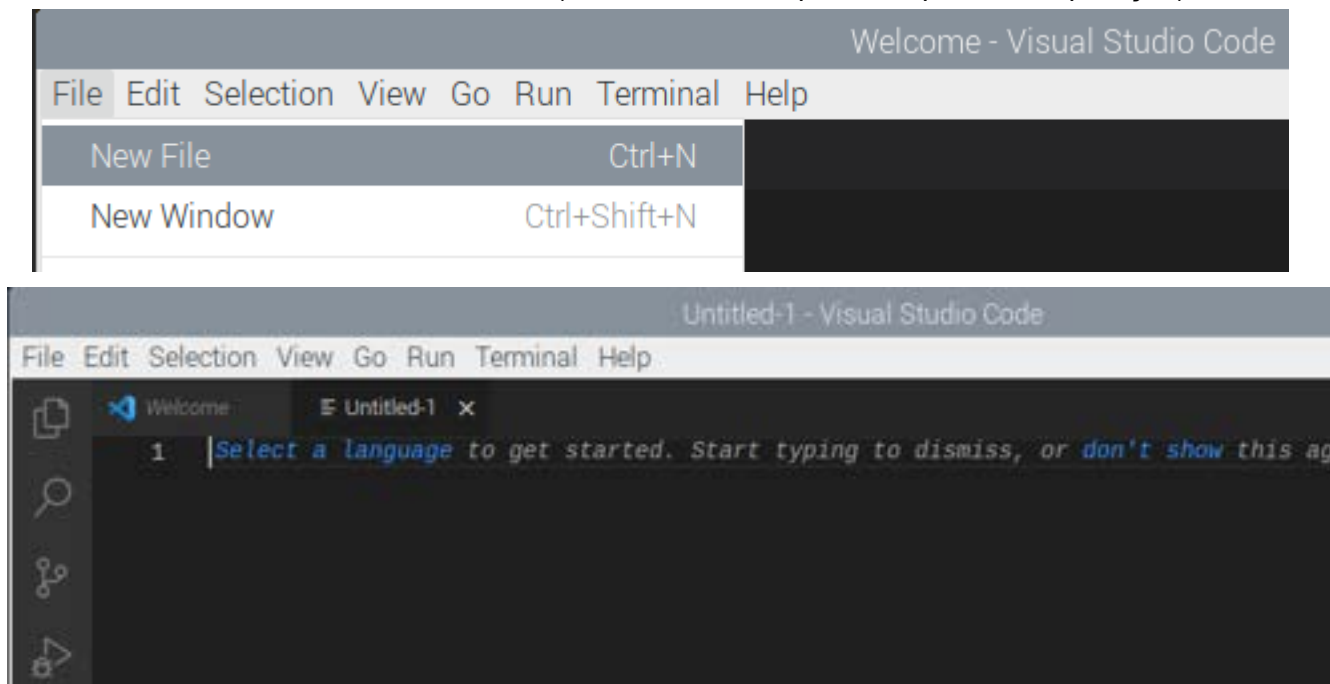
10. Guarde e corra novamente o programa com o mesmo processo dos passos 6 e 7.

Dica: em vez de escrever novamente o comando “python3 ecran.py” (sem incluir as aspas), pode premir a seta para cima no teclado para ir buscar comandos previamente usados.

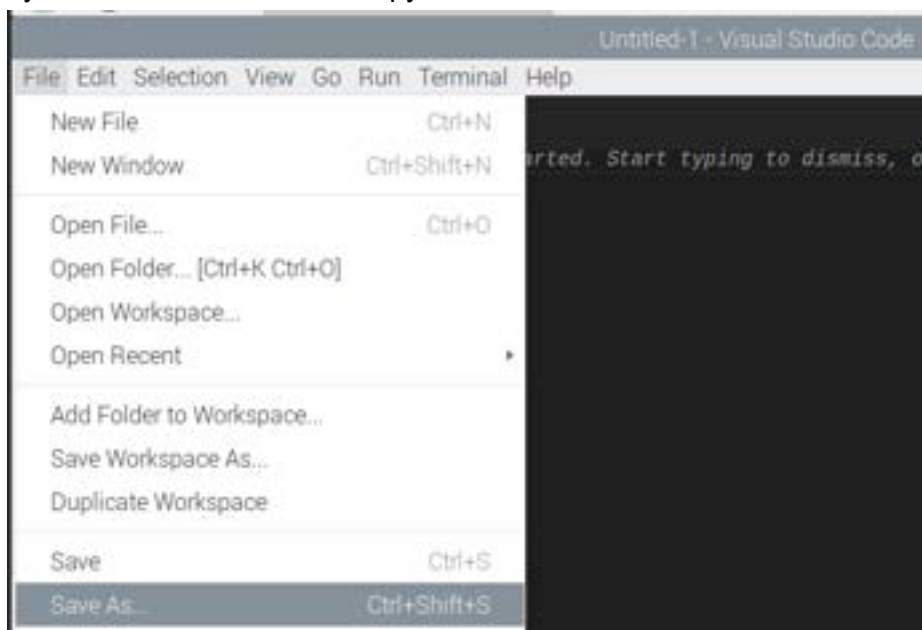


Tirar Fotografias com a Câmara

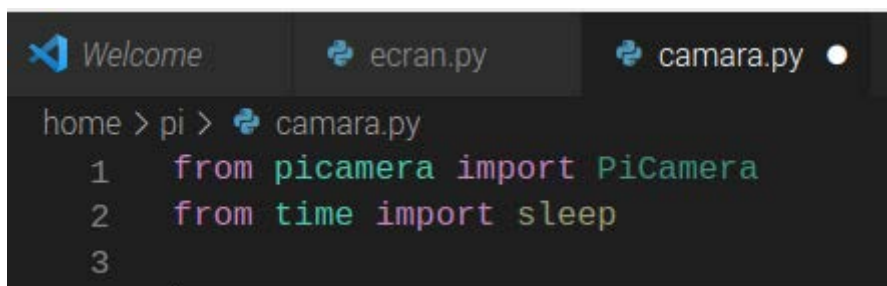
1. Crie um novo ficheiro com *File > New File* (menu no canto superior esquerdo da aplicação)



2. Vamos já guardar o ficheiro vazio com um nome específico com *File > Save As...*
O importante aqui é que o nome do ficheiro termine em “.py” para ser reconhecido pelo editor como Python. Vamos usar “camara.py”.



3. Para usar a câmara, vamos importar as bibliotecas “PiCamera” e “time” (linhas 1 e 2, respetivamente):



4. Vamos criar a variável “cam” para representar a câmara (linha 4), definir a sua resolução (escolhemos 1024x768, mas ela suporta mais!) (linha 5), e esperar 1 segundo para garantir que a câmara está inicializada (linha 6):

```
home > pi > camara.py > ...
1  from picamera import PiCamera
2  from time import sleep
3
4  cam = PiCamera()
5  cam.resolution = (1024, 768)
6  sleep(1)
7
```

5. Agora podemos capturar uma imagem com o método “capture”. Como argumento, vamos dar o nome do ficheiro que vai conter a imagem capturada, que deve ter a extensão jpg. Vamos usar “foto.jpg”.

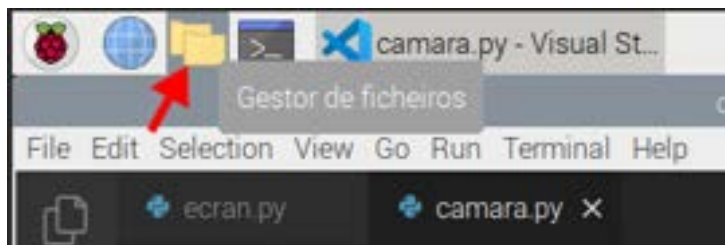
```
home > pi > camara.py > ...
1  from picamera import PiCamera
2  from time import sleep
3
4  cam = PiCamera()
5  cam.resolution = (1024, 768)
6  sleep(1)
7
8  cam.capture("foto.jpg")
9
```

6. À semelhança do guia anterior, vamos guardar o ficheiro (Ctrl+S ou *File>Save*), e correr o nosso programa no terminal (pode ser aberto em *Terminal>New Terminal*). Como o nosso ficheiro se chama “camara.py”, vamos correr “python3 camara.py” (novamente sem incluir as aspas).

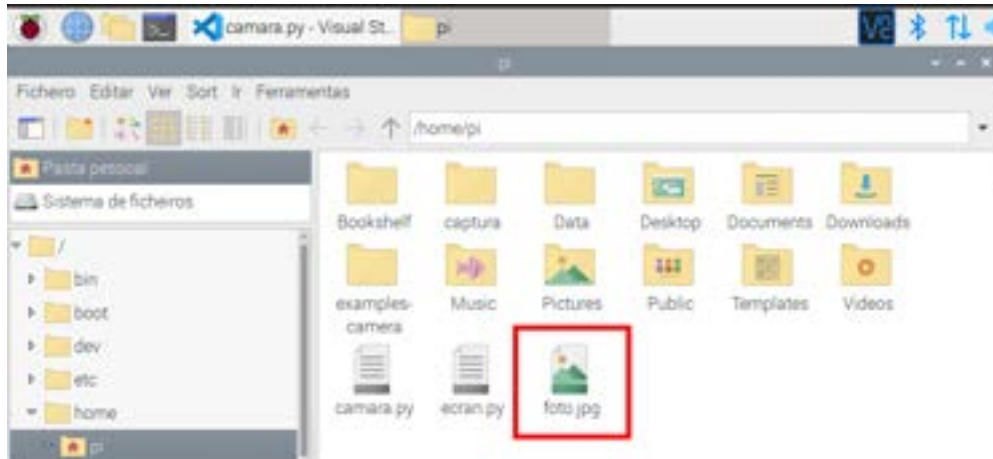
Nota: não se esqueça de tirar a cobertura da lente!

O programa termina sozinho, não sendo preciso premir Ctrl+C.

7. Para ver a imagem capturada, vamos abrir o gestor de ficheiros no menu do topo do ecrã:



8. Clique duas vezes em “foto.jpg” para a abrir



9. A imagem ficou desfocada?

Pode ajustar o foco da câmara em 2 sítios: na roda dentada encostada ao sensor (que pode ser bloqueado com a pequena chave incluída com o módulo da câmara apertando o parafuso deste), e na zona estriada da lente (que pode ser bloqueada rodando o parafuso marcado com NEAR / FAR também na lente).

O ajuste na roda dentada é usado para mudanças grande no foco, e o da lente para mudanças pequenas.

Pode também variar a quantidade de luz recebida pelo sensor rodando a parte mais exterior da lente (bloqueada rodando o parafuso marcado com OPEN / CLOSE na lente).

Utilização da Câmara para Reconhecimento de Objetos

Demo: Reconhecimento de Objetos com Aprendizagem Automática

O que é Aprendizagem Automática

Aprendizagem automática (*Machine Learning* em inglês) é um ramo da Inteligência Artificial, provavelmente o mais popular hoje.

De um modo geral, consiste em criar de forma automática um **modelo** para previsões com base num conjunto de exemplos anotados. Este processo de afinação com base em exemplos é conhecido como o **treino** do modelo.

Um modelo pode habitualmente ser decomposto em duas partes: estrutura e coeficientes. A estrutura do modelo é definida pelo programador (qual o tamanho dos dados de entrada e saída, e que operações são feitas para transformar um no outro), parametrizadas por um conjunto de coeficientes. Os coeficientes são ajustados durante o processo de treino.

Um tipo de modelo popular são as **redes neurais**:

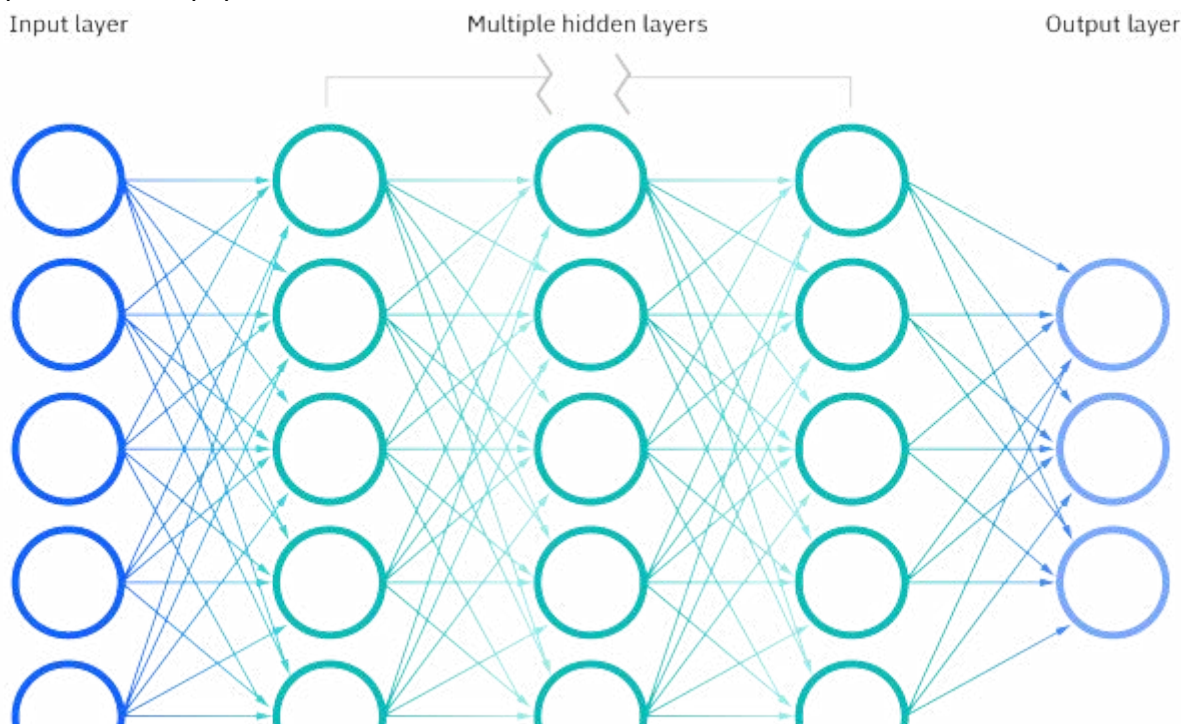


Imagem retirada de <https://www.ibm.com/topics/neural-networks>

Em cada círculo (nó) da entrada introduz-se um valor de entrada (numérico). O valor dos círculos de cada camada seguinte são calculados com base nos da camada anterior ponderados pelos coeficientes (um coeficiente diferente por ligação).

O processo de treino consiste em modificar os coeficientes de modo a que a saída da rede neuronal se aproxime do valor real para cada conjunto de dados de entrada, usando algoritmos de optimização. Não é uma ciência exata, porque queremos que seja próximo o suficiente para ser uma boa aproximação, mas não ao ponto de copiar os dados de treino (chamado *overfitting*).

Martelar números até coincidir vagamente com o esperado parece uma loucura, mas o que é certo é que costuma produzir uma aproximação boa o suficiente!

Para avaliar estes modelos (e garantir que não há *overfitting*), é costume reservar uma parte do conjunto de dados anotados que **não serão usados para treino** para quantificar a sua precisão.

Por exemplo, um modelo para identificar se uma imagem é de um gato ou cão podia ter como entrada uma imagem (se tiver resolução 300x300, corresponde a 300x300x3 valores, para capturar os 3 canais de cor), e como saída um único valor, em que, por exemplo, 0 significa cão e 1 significa gato.

Importância de Pré-Processamento

300x300 é um tamanho muito mais pequeno do que estamos habituados para imagens, mas um tamanho maior faz o modelo ficar também maior, e mais difícil de treinar e usar.

Por isto, é comum investir tanto tempo no pré-processamento dos dados (que pode envolver outros modelos!), como no próprio modelo que se pretende criar.

Exemplo: em língua natural é comum eliminar todas as preposições do texto como um dos passos de pré-processamento.

Modelos Generativos

Para gerar texto ou imagem, é comum aplicar estes modelos duas vezes, uma primeira para “sumarizar” a entrada, e uma segunda com o modelo ao contrário (da saída para a entrada), para gerar um novo texto/imagem.

Observou-se que se alterarmos ligeiramente a versão sumarizada que está no meio, produzimos texto/imagem semelhantes ao original excepto numa ou outra característica.

O que é o Coral USB Accelerator

É um coprocessador desenhado especificamente para correr operações matemáticas comuns em modelos (aritmética com números decimais “*floating point*”).

Este em particular é desenhado para uma biblioteca popular de machine learning chamada *TensorFlow* (*TPU* significa *Tensor Processing Unit*).

Para além de coprocessadores especializados como este, que são recentes, é comum usar placas gráficas para acelerar o processo de treino/uso de modelos de IA. Estas têm melhor desempenho neste tipo de operações que os processadores tradicionais, embora não com a mesma eficiência de *hardware* especializado.

A Coral especializa-se em trazer aprendizagem automática a dispositivos do consumidor (“*edge*”), e disponibiliza vários exemplos e modelos pré-treinados: <https://coral.ai/examples/>

É possível afinar alguns destes modelos com dados próprios.

Como correr um exemplo de deteção e classificação de objetos

Abrir um novo terminal e correr (um comando por linha):

```
git clone https://github.com/google-coral/examples-camera.git
cd examples-camera
sh download_models.sh
cd gstreamer
sh install_requirements.sh
python3 detect.py
```

O primeiro comando transfere o programa-exemplo, os do meio transferem modelos e instalam pré-requisitos, e o último corre o programa-exemplo.