

# Hexaquest

## guia para interface lúdico

um jogo de Eva Vital e Marco Conceição  
um projeto Universidade Lusófona — ECATI (licenciatura em Videojogos)  
& Casa Pia de Lisboa – CED Jacob Rodrigues Pereira  
disponível em <<http://educacaoacessivel.ulusofona.pt>>

A BUSCA DE JACOB é um videojogo que pode ser jogado com um normal teclado  
ou com um interface original concebido pelos autores do jogo.  
Este documento contém as indicações fundamentais para a sua reprodução.



This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License

## COMPONENTES

1 x Arduino Micro

<https://www.ptrobotics.com/plataformaarduino-e-modelos-alternativos-equivalentes/1776-arduino-micro.html>

1 x Placa Veroboard 95x130mm 36x50 Buracos

<https://www.ptrobotics.com/pcb/411-veroboard-95x130mm-36x50buracos.html>

— Necessita corte à medida para a interface

12 x Push Button SPST 24V 50mA 2Pin

<https://www.ptrobotics.com/tactile-switch/1890-push-button-spst-24v-50ma-2pin.html>

2 x Tactile Button 12mm w/out cap

<https://www.ptrobotics.com/tactile-switch/6449-tactile-button-12mm-wout-cap.html>

1 x Easy Go PLA bq 1,75mm Sky Blue 1Kg

<https://www.ptrobotics.com/impressao-3d/3250-easy-go-pla-bq-175mm-sky-blue-1kg.html> — Botões laterais

1 x Ultimaker PLA 2.85 750g Blue

<https://www.ptrobotics.com/impressao-3d/4504-ultimaker-pla-285-750g-blue.html> — Corpo do modelo

1x Easy Go PLA bq 1,75mm Vitamine Orange 1Kg

<https://www.ptrobotics.com/impressao-3d/3253-easy-go-pla-bq-175mm-vitamine-orange-1kg.html> — Botão central

7x Crimp Connector 1x1Pin

1 x Wire 20AWG Black 1m

<https://www.ptrobotics.com/fio-electrico/5425-wire-20awg-black-1m.html>

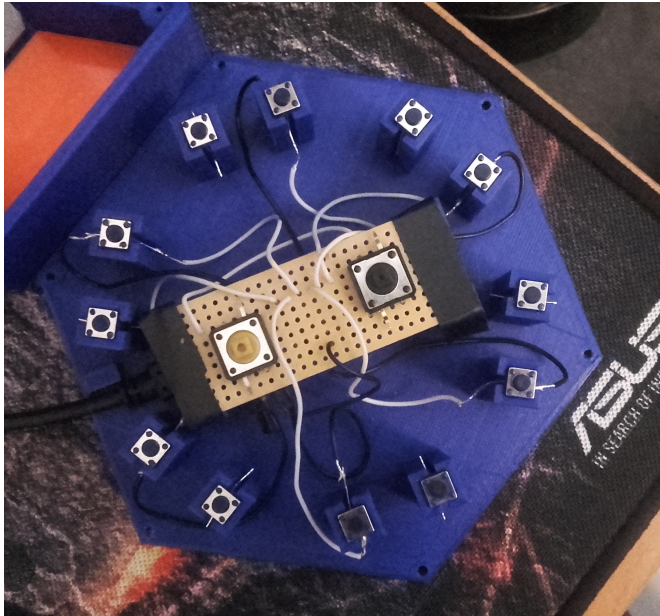
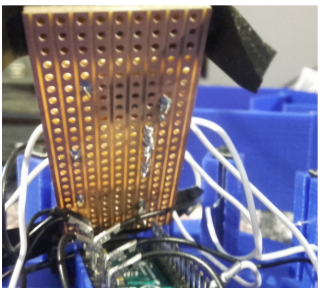
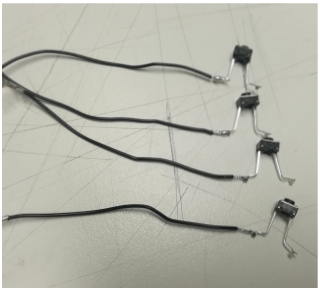
1 x Wire 20AWG Grey 1m

<https://www.ptrobotics.com/fio-electrico/5427-wire-20awg-grey-1m.html>

1 x Fita Elétrica - para prender a Placa Veroboard em posição

## Montagem

Primeiro, soldamos os conectores dos sete switches. O maior é diretamente soldado na veroboard. Neste caso, os fios brancos são o Ground e os pretos são os Input. Apenas 6 dos 12 Push Buttons terão fios soldados.



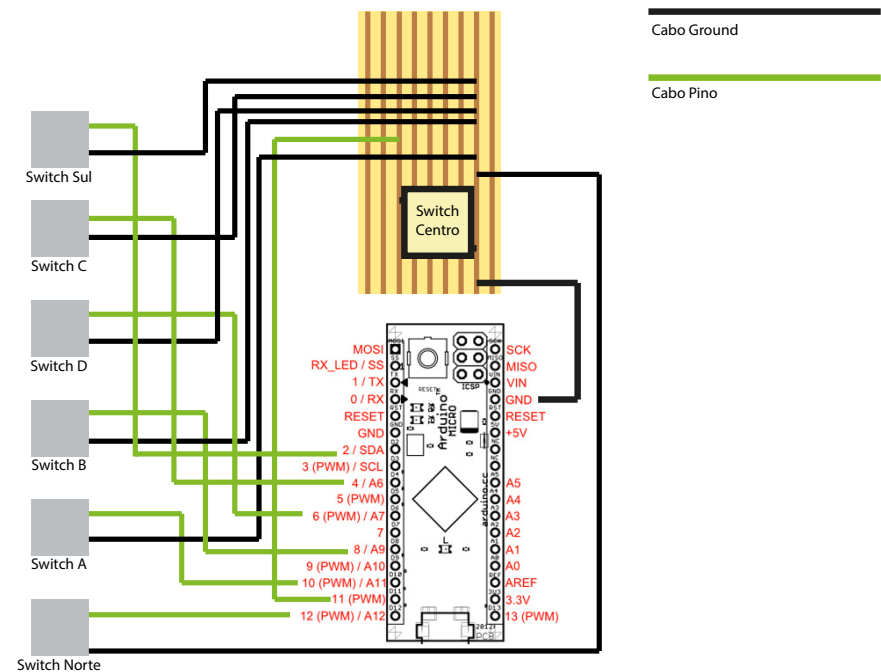
Após soldar ambos os fios passa a soldar os Grounds numa única linha na veroboard em que no final tem um fio ligado ao pino GND no Arduino, e depois colocar os dois Tactile Buttons (apenas um tem ligações GND e Input, o outro serve a função de equilibrar a placa do botão quando pressionada).

Com o pino GND no veroboard ligado, colocam-se os Push Buttons nas ranhuras conforme mostrado na imagem acima, onde se vai colocando os botões alternando entre os que têm contactos soldados e os que servirão para equilibrar as placas dos botões.

A seguir, ligue os botões dos Push Buttons aos pinos correspondentes no Arduino Micro conforme o esquema apresentado abaixo.

Depois do processo de montagem dos componentes internos, ligamos o cabo USB ao Arduino e fechamos o interface ao colocar a peça superior com as placas dos botões já colocadas na mesma..

Após este processo fechámos o Interface com parafusos.



## Programação Arduino

Primeiramente decidimos associar o Arduino às teclas do teclado em vez de o ligar diretamente ao Unity.

Desta forma programámos da seguinte forma:

Começamos por determinar os pinos no Arduino Micro que serão usados como comunicação dos botões

```
#include <Keyboard.h>

void setup() {
  //start serial connection
  Serial.begin(9600);
  //South
  pinMode(2, INPUT_PULLUP);
  //C
  pinMode(4, INPUT_PULLUP);
  //D
  pinMode(6, INPUT_PULLUP);
  //B
  pinMode(8, INPUT_PULLUP);
  //A
  pinMode(10, INPUT_PULLUP);
  //Confirm
  pinMode(11, INPUT_PULLUP);
  //North
  pinMode(12, INPUT_PULLUP);
}
```

De seguida, registámos o valor dos botões para uma variável e emitimos o valor de cada botão.

```
void loop() {
  //read the pushbutton value into a variable
  int sensorS = digitalRead(2);
  int sensorC = digitalRead(6);
  int sensorD = digitalRead(4);
  int sensorB = digitalRead(8);
  int sensorA = digitalRead(10);
  int sensorConf = digitalRead(11);
  int sensorN = digitalRead(12);

  //print out the value of the pushbutton
  Serial.println(sensorS);
  Serial.println(sensorC);
  Serial.println(sensorD);
  Serial.println(sensorB);
  Serial.println(sensorA);
  Serial.println(sensorConf);
  Serial.println(sensorN);
}
```

Atendendo ao elemento INPUT\_PULLUP, a lógica dos botões é invertida, ou seja, quando é premido um botão, o sensor está em estado LOW, e quando não é premido, está em estado HIGH

Atendendo ao parágrafo anterior, vamos indicar para cada sensor qual a tecla do teclado que é para ser emitida.

Neste caso usamos Keyboard.press() e Keyboard.release() em vez de apenas Keyboard.print() quando o sensor está no estado LOW, uma vez que alguns jogos não aceitam keystrokes programadas, mas aceitam inputs programados que tenham um espaço de tempo entre o premir do botão e o momento em que solta o botão

```
if (sensorS == LOW) {  
  Keyboard.press('s');  
  //digitalWrite(13, LOW);  
} else {  
  Keyboard.release('s');  
  //digitalWrite(13, HIGH);  
}
```

```
if (sensorC == LOW) {  
  Keyboard.press('a');  
  //digitalWrite(13, LOW);  
} else {  
  Keyboard.release('a');  
  //digitalWrite(13, HIGH);  
}
```

```
if (sensorD == LOW) {  
  Keyboard.press('d');  
  //digitalWrite(13, LOW);  
} else {  
  Keyboard.release('d');  
  //digitalWrite(13, HIGH);  
}
```

```
if (sensorB == LOW) {  
  Keyboard.press('e');  
  //digitalWrite(13, LOW);  
} else {  
  Keyboard.release('e');  
  //digitalWrite(13, HIGH);  
}
```

```
if (sensorA == LOW) {  
  Keyboard.press('q');  
  //digitalWrite(13, LOW);  
} else {  
  Keyboard.release('q');  
  //digitalWrite(13, HIGH);  
}
```

```
if (sensorConf == LOW) {  
  Keyboard.press((char) 32);  
  //digitalWrite(13, LOW);  
} else {  
  Keyboard.release((char) 32);  
  //digitalWrite(13, HIGH);  
}
```

```
if (sensorN == LOW) {  
  Keyboard.press('w');  
  //digitalWrite(13, LOW);  
} else {  
  Keyboard.release('w');  
  //digitalWrite(13, HIGH);  
}  
}
```