Implementação de um jogo estilo JumpMan usando Python e Embed embarcado para Arduino

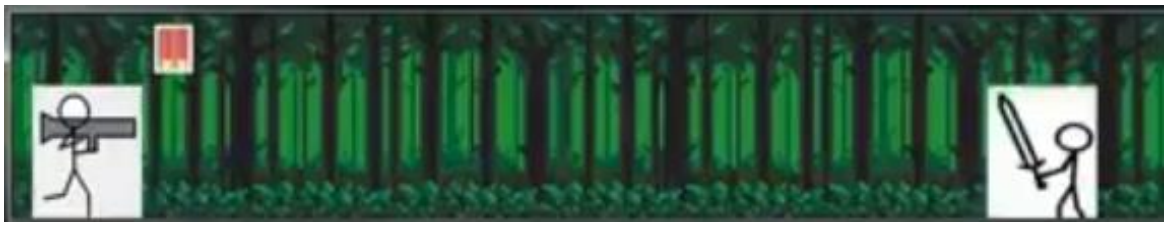


Figura 1 Visual inicial do jogo

1 Objetivo

Neste exemplo, iremos implementar as primeiras etapas da criação de um jogo estilo Jumpman. Um vídeo da versão inicial do jogo em funcionamento pode ser encontrado neste link.

A lógica do jogo, assim como seus gráficos, será implementada usando python e algumas de suas bibliotecas. O controle do jogo por parte do usuário será feito com uma placa Arduino e dois botões. Toda a lógica do Arduino foi implementada no SolidThinking Embed.

2 Implementação

2.1 Python

O primeiro passo será importar todas as bibliotecas do python que serão uteis neste projeto. A lista de bibliotecas pode ser vista abaixo.

```
from tkinter import *
from PIL import ImageTk, Image
import time
import serial
import random
```

As funções de cada biblioteca são:

Tkinter - Responsável por criar a interface gráfica.

ImageTk – Responsável por importar e trabalhar com arquivos de imagem.

Time – Responsável por criar contagens de tempo real. Será usada para definir a frequência de ciclo do programa.

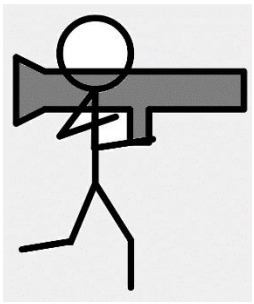
Serial – Responsável por estabelecer uma comunicação serial com alguma porta do computador. Será suada para estabelecer a comunicação com o Arduino.

Random – Responsável por criar gerações de numero randômicas. Será importante para definir se um novo inimigo deve ser criado ou não.

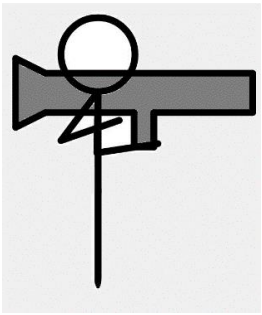
Com todas as bibliotecas importadas, passamos para a definição das variáveis que usaremos ao longo do programa. A lista de variáveis pode ser vista abaixo.

```
a=1
executando1=0;
executando2=0;
contagem=0;
count=0;
backcount=0;
enemyPos=[0,0,0,0,0,0,0,0,0,0];
difficulty=100;
over=1;
shotgun=2;
shot=0;
```

A variável ‘a’ irá variar constantemente entre zero e um a cada ciclo do programa. Seu valor irá determinar como o personagem do jogador aparecerá na tela, como pode ser visto nas figuras abaixo. O revezamento entre as duas imagens irá criar a ilusão de um personagem correndo.



a=1



a=0

As variáveis executando1 e executando2 irá determinar qual comando foi enviado pelo usuário e está sendo executado. Os comandos possíveis são pular ou atirar. Ambos os comandos acionam uma ação com uma duração maior que um ciclo do programa, de forma que essas variáveis são necessárias para que não haja sobreposição de ações. As variáveis contagem e count serão responsáveis por determinar a duração de cada uma das duas ações descritas anteriormente.

A variável backcount será a responsável por determinar o estado atual da imagem de fundo. Para criar a ilusão de movimento, uma imagem cíclica de fundo é criada e movida para a esquerda a cada ciclo do programa. Como a imagem é cíclica, ao chegar no final de seu movimento para a esquerda sua posição é reiniciada, e a ilusão de movimento continua.

O vetor enemyPos irá guardar a posição dos diversos inimigos existentes. Quando um inimigo novo é criado, o ultimo termo do vetor terá seu valor mudado para um e a cada ciclo os valores são movidos para a posição à esquerda.

A variável difficulty será a responsável por determinar a dificuldade do jogo. Quanto menor esta variável for, menor será o intervalo entre ciclos do programa, ou seja, mais rápido este ficará. A dificuldade do jogo é aumentada com o passar do tempo, até atingir uma dificuldade máxima onde esta variável se iguala a zero.

A variável over irá determinar se o jogo ainda está em execução ou se o jogador perdeu.

As variáveis shotgun e shot serão responsáveis por determinar o número de tiros que o jogador ainda possui e se os inimigos dentro do alcance da arma devem ser destruídos, respectivamente. O jogador possui direito a dois tiros, que serão recarregados a cada

intervalo preestabelecido de tempo. O intervalo de tempo é função da variável difficulty, de forma que quanto maior a dificuldade, maior será o intervalo.

Com as variáveis definidas, passamos para as preparações iniciais.

```
ser = serial.Serial('COM3', 9600)

def setboneco():
    global a
    if a:
        img = Image.open('boneco3.png')
        img=img.resize((50,65), Image.ANTIALIAS)
        a=0
        return img
    else:
        img = Image.open('boneco2.png')
        img=img.resize((50,65), Image.ANTIALIAS)
        a=1
        return img

root = Tk()
root.geometry("1000x100")
root.resizable(0,0)
```

Primeiramente iniciamos a comunicação serial com o Arduino, que foi previamente verificado usando a porta COM3. Definimos a função setboneco() que será a responsável por variar o valor da variável 'a' e escolher a imagem associada. Criamos também a variável root que será nossa janela principal, definimos seu tamanho e o deixamos fixo.

Com isso, todo o resto do programa será executado dentro de um loop infinito, criado pelo comando [while 1].

```
while 1:
    if over:
        if sum(enemyPos)<3:
            enemyPos=enemyPos[1:]+[round(random.randint(1,15)/29)]
        else:
            enemyPos=enemyPos[1:]+[0]

        for i in enemyPos:
            count=count+1;
            if i:
                enemy=Image.open('enemy.png')
                enemy=enemy.resize((50,65), Image.ANTIALIAS)
                enemy=ImageTk.PhotoImage(enemy)
                panel = Label(root, image = enemy)
                panel.place(relx=0.0+0.12*count, rely=0.35)
                root.update()

        count=0;

        if backcount<10:
            backcount=backcount+1
        else:
            backcount=0

        back = Image.open('back.png')
        back=back.resize((1500,110), Image.ANTIALIAS)
        back=ImageTk.PhotoImage(back)
        panel = Label(root, image = back)
        panel.place(relx=0.0-0.05*backcount, rely=-0.05)

        if shotgun > 1.99:
            shell=Image.open('shell2.png')
            shell=shell.resize((15,20),Image.ANTIALIAS)
            shell=ImageTk.PhotoImage(shell)
            panel = Label(root, image = shell)
            panel.place(relx=0.04+0.03, rely=0.05)

        elif shotgun >1:
            shell=Image.open('shell.png')
            shell=shell.resize((10,20),Image.ANTIALIAS)
            shell=ImageTk.PhotoImage(shell)
            panel = Label(root, image = shell)
            panel.place(relx=0.04+0.03, rely=0.05)
```

```
if ser.in_waiting and not executando1 and not executando2:
    if ser.read()==b'a':
        executando1=1
    else:
        executando2=1;
        if shotgun>0.9:
            shotgun=shotgun-1;
            shot=1;
elif executando1 and not executando2:
    if contagem<4:
        img = Image.open('boneco.png')
        img=img.resize((50,65), Image.ANTIALIAS)
        img=ImageTk.PhotoImage(img)
        panel = Label(root, image = img)
        panel.place(relx=0.01, rely=0.35-0.15*contagem)
        root.update_idletasks()
        root.update()
        contagem=contagem+1
    elif contagem<8:
        img = Image.open('boneco.png')
        img=img.resize((50,65), Image.ANTIALIAS)
        img=ImageTk.PhotoImage(img)
        panel = Label(root, image = img)
        panel.place(relx=0.01, rely=0.35-0.15*4+0.15*(contagem-4))
        root.update_idletasks()
        root.update()
        contagem=contagem+1
    else:
        contagem=0
        executando1=0
else:
    if executando2:
        if contagem<2 and shot:
            img2 = Image.open('atk.png')
            img2=img2.resize((500,20), Image.ANTIALIAS)
            img2=ImageTk.PhotoImage(img2)
            panel2 = Label(root, image = img2)
            panel2.place(relx=0.05, rely=0.43)
            contagem=contagem+1
            enemyPos[0]=0
            enemyPos[1]=0
            enemyPos[2]=0
            enemyPos[3]=0
        else:
            shot=0
            panel2.destroy()
            executando2=0
            contagem=0
    img=ImageTk.PhotoImage(setboneco())
    panel = Label(root, image = img)
    panel.place(relx=0.01, rely=0.35)
    #root.update_idletasks()
    root.update()
    if enemyPos[0]:
        over=0
```

```
        if difficulty>1:
            difficulty=difficulty-1

    else:
        img = Image.open('over.png')
        img=img.resize((500,100), Image.ANTIALIAS)
        img=ImageTk.PhotoImage(img)
        panel = Label(root, image = img)
        panel.place(relx=0.25, rely=0.05)
        root.update()

    if shotgun<2:
        shotgun=shotgun+0.0009*difficulty+0.01
    time.sleep(0.02+ 0.001*difficulty)
```

2.2 Embed

A implementação do controle se torna extremamente simplificada usando o Embed. O diagrama usado pode ser visto na Figura 2.

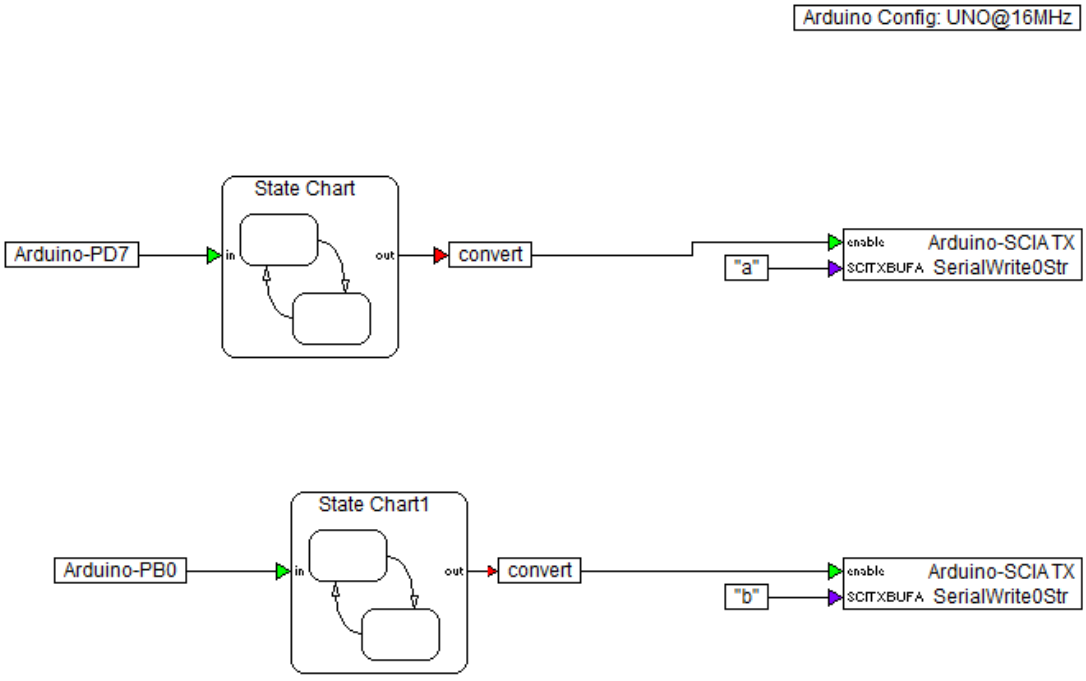


Figura 2 Diagrama de controle no Embed ST

Ao pressionar o botão de pulo, o Arduino deve enviar o char 'a' para o computador. O programa irá ler qualquer informação enviada pela porta serial, e quando receber o equivalente do char 'a' irá executar a ação de pulo. O mesmo vale para o comando de atirar, enviando desta vez o char 'b'.

Como queremos que apenas um char seja enviado quando pressionarmos o botão, independente do tempo de pressionamento, precisamos adicionar uma lógica de comando. Esta lógica foi facilmente criada usando os StateCharts que podem ser vistos na Figura 2. Sua estrutura pode ser vista na Figura 3.

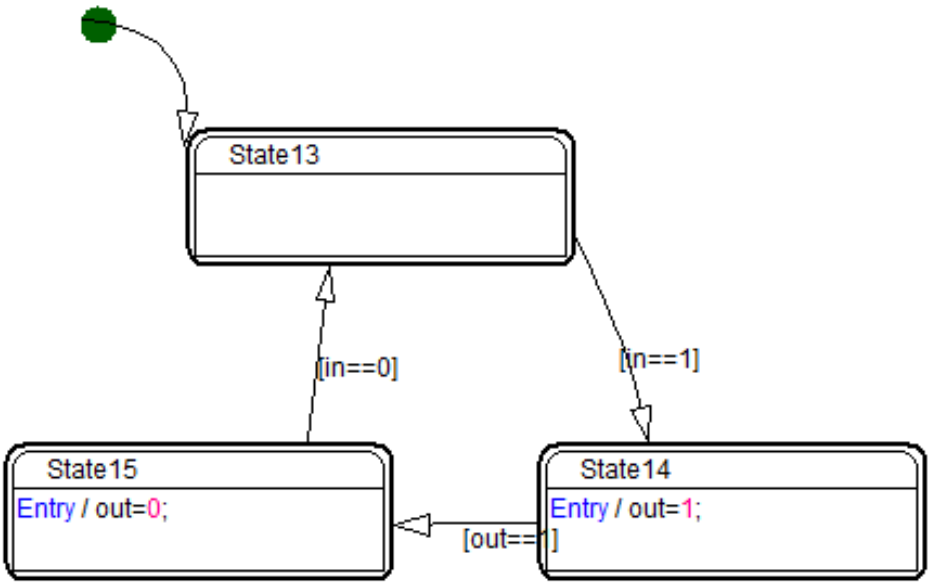


Figura 3 Estutura de estados no Embed

Desta forma, quando apertarmos o botão no Arduino, a variável in do StateChart correspondente irá assumir o valor 1, fazendo com que ele passe para o próximo estado. No segundo estado, a saída do StateChart será mudada para 1 e imediatamente passa para o próximo estado e volta a zero. Finalmente, quando a entrada voltar a zero, voltamos ao estado inicial e o ciclo pode ser repetido. Desta forma, criamos um pulso toda vez que o botão for pressionado. Este pulso irá habilitar o bloco de comunicação serial, que irá por sua vez enviar o char correspondente uma única vez.

O bloco de comunicação serial irá enviar os dados presentes em sua entrada "SXITXBUFA" sempre que a entrada "enable" receber um sinal alto. Lembrando que, antes de usar este bloco, a comunicação serial deve ser configurada. Neste exemplo usaremos uma conexão serial com velocidade 9600 e configurada para 8bits, como pode ser visto na Figura 4.

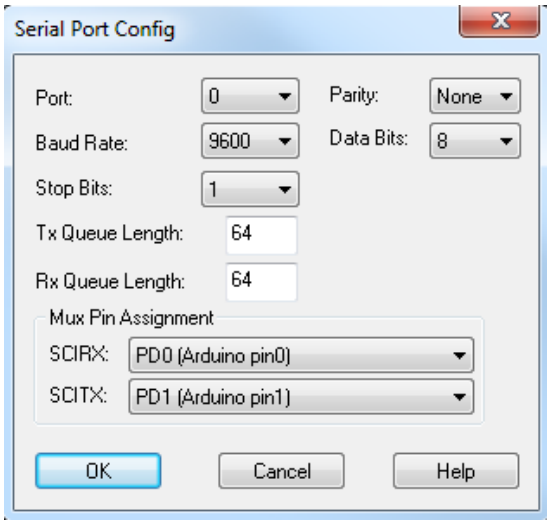


Figura 4 Configurações seriais usadas

2.3 Arduino

As conexões elétricas no Arduino podem ser vistas na Figura 5.

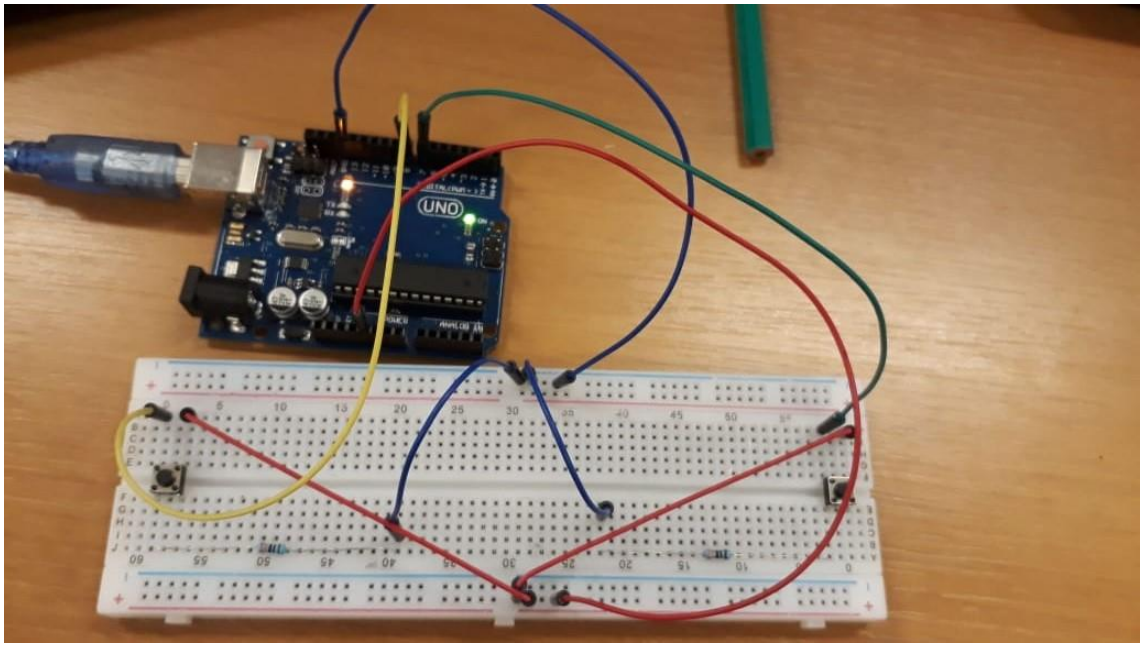


Figura 5 Montagem elétrica no Arduino com o auxílio de uma protoboard

As conexões foram feitas de forma que, enquanto um botão não estiver pressionado, o pino correspondente de entrada digital estará conectado ao GND. Quando o botão for pressionado, aquele pino será conectado aos 5V. Os resistores de 330Ω são necessários para garantir este funcionamento, uma vez que sem eles iriamos fechar um curto no circuito ao apertar o botão.

3 Resultados e conclusão

Um vídeo completo do jogo em execução pode ser encontrado neste link.

Muitas melhorias ainda podem ser feitas no código, tanto melhorias no funcionamento do jogo quanto em seu próprio gameplay. Mais controles podem ser adicionados no Arduino, novos inimigos, novos mapas e novas ações podem ser adicionadas no gameplay e melhorias gráficas e de desempenho podem ser feitas no código.

Este trabalho tentou mostrar as primeiras etapas do desenvolvimento de um jogo simples, misturando ferramentas como o Embed e o Python. O uso de bibliotecas específicas do python foi essencial para a operação bem-sucedida.