



Projeto 10

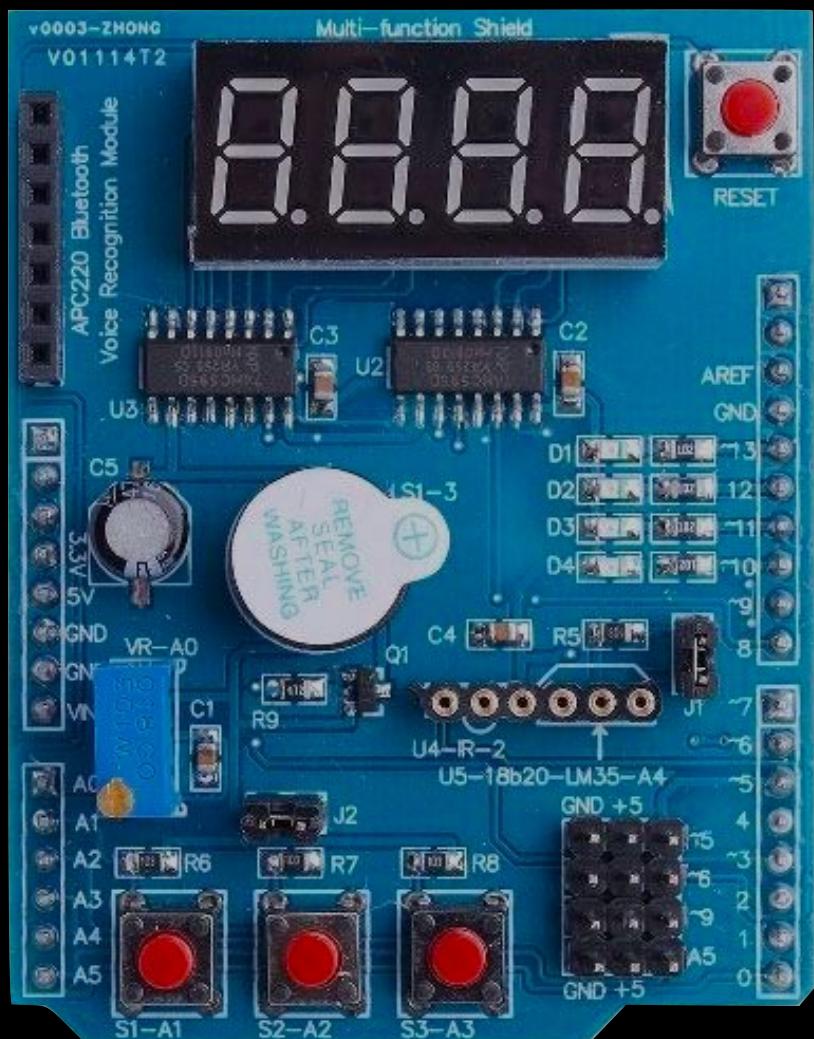
Controle Gráfico – Teoria

Jan K. S. – janks@puc-rio.br

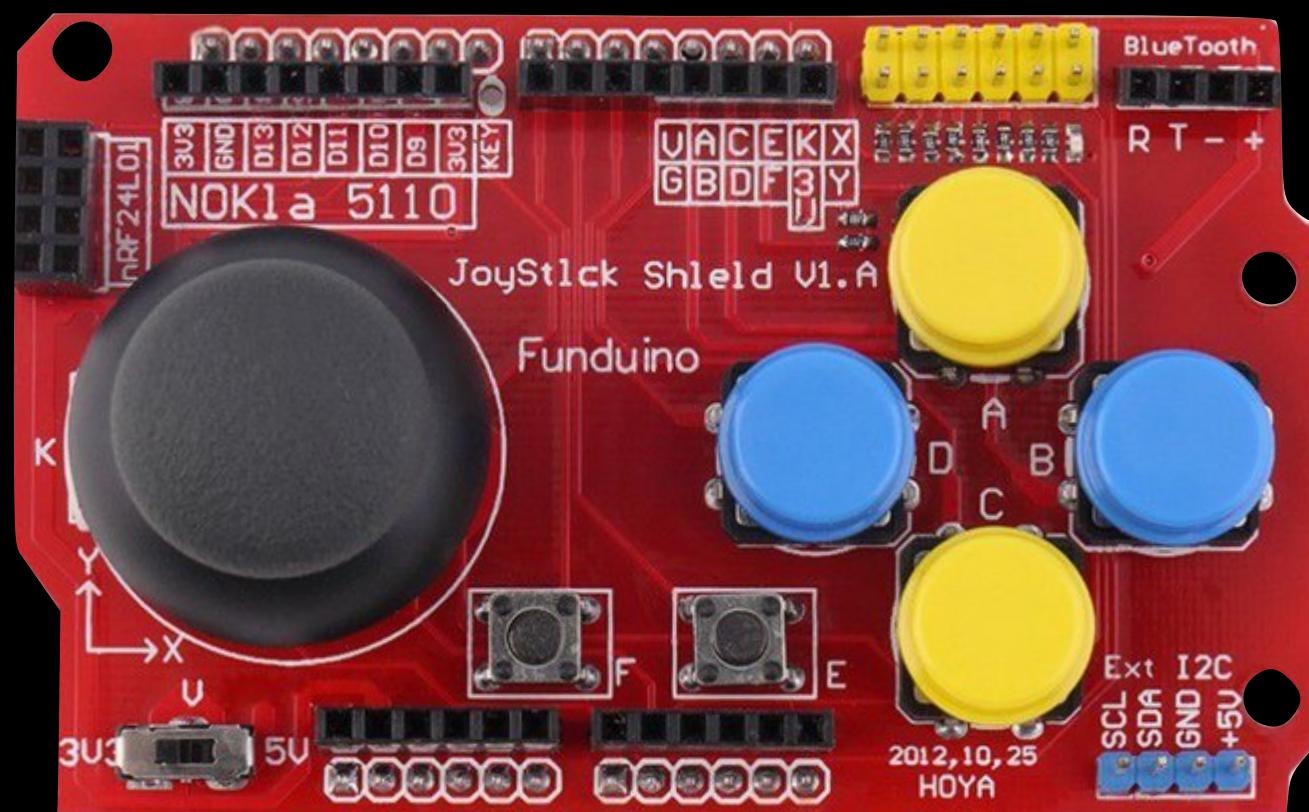
ENG1419 – Programação de Microcontroladores

Hardware

controles fixos



display e buzzer
poucos botões



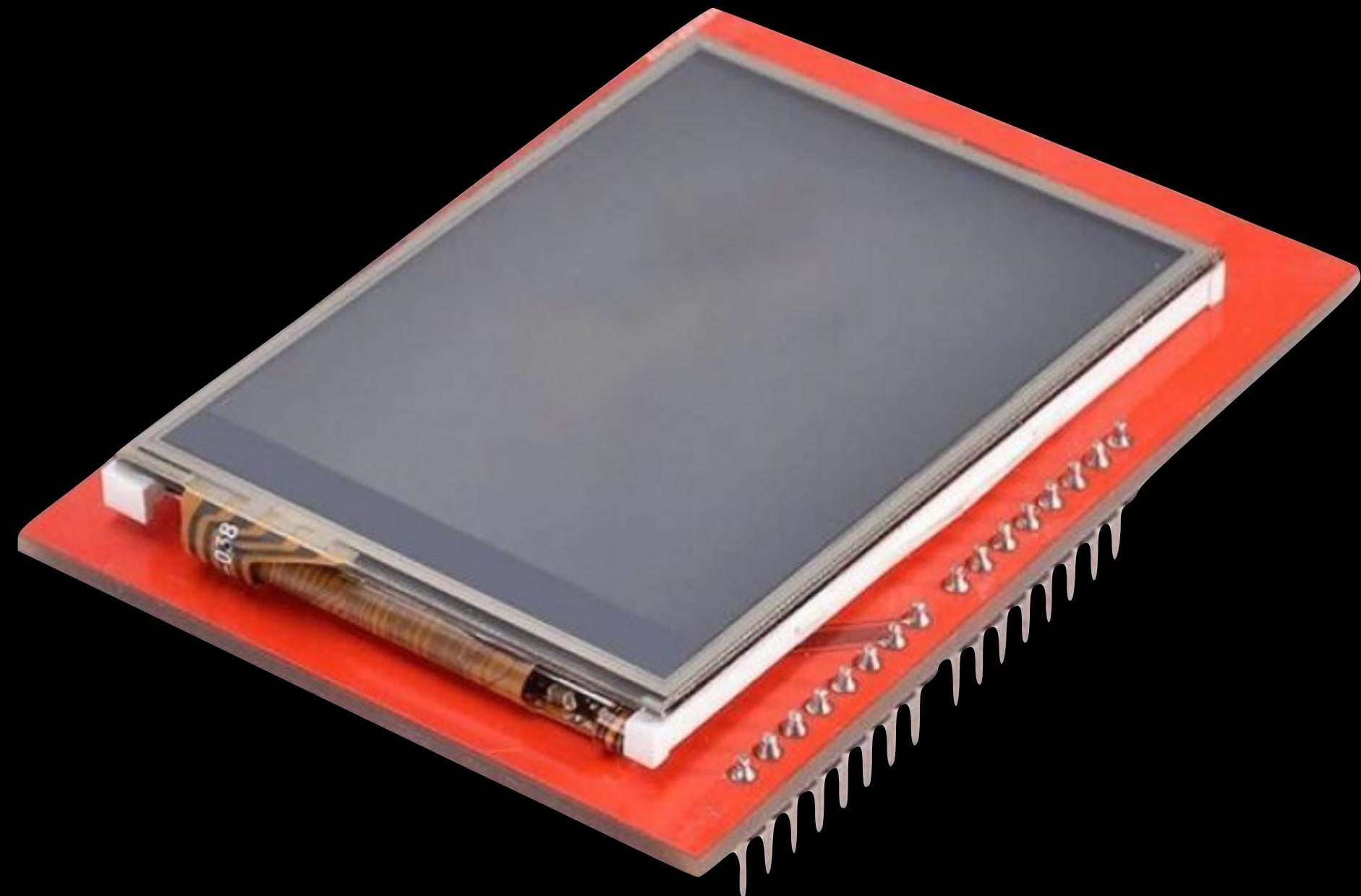
analógico e vários botões
nenhum display

Screenshot of a Mac desktop showing the Pages application open. The document is a flyer for a red scooter. The top bar shows the menu: Arquivo, Editar, Inserir, Formatar, Organizar, Visualizar, Compartilhar, Janela, Ajuda. The status bar shows the date as Sáb 12:29 Jan. The main content area displays a red Vespa-style scooter. Below it is a red box containing the text "100CC SCOOTER R\$ 2400". To the right of the scooter is a gray box with placeholder text: "Lorem ipsum dolor sit amet, ligula suspendisse nulla pretium, rhoncus tempor fermentum, enim integer ad vestibulum volutpat. Nisl rhoncus turpis est, vel elit, congue wisi enim nunc ultricies sit, magna tincidunt. Maecenas aliquam maecenas ligula." At the bottom of the page is the email "EXEMPLO@EXAMPLE.COM". The right side of the screen shows the Pages ribbon toolbar with icons for Inserir, Tabela, Gráfico, Texto, Forma, Comentário, Alfa, Colaborar, Formatar, and Documento. The sidebar on the left shows the page preview with the text "VENDE-SE" and "100CC SCOOTER R\$ 2400". The dock at the bottom contains various Mac application icons.



Interfaces Gráficas

Liquid-Crystal Display
Thin-Film-Transistor
Tela de 2.4 polegadas



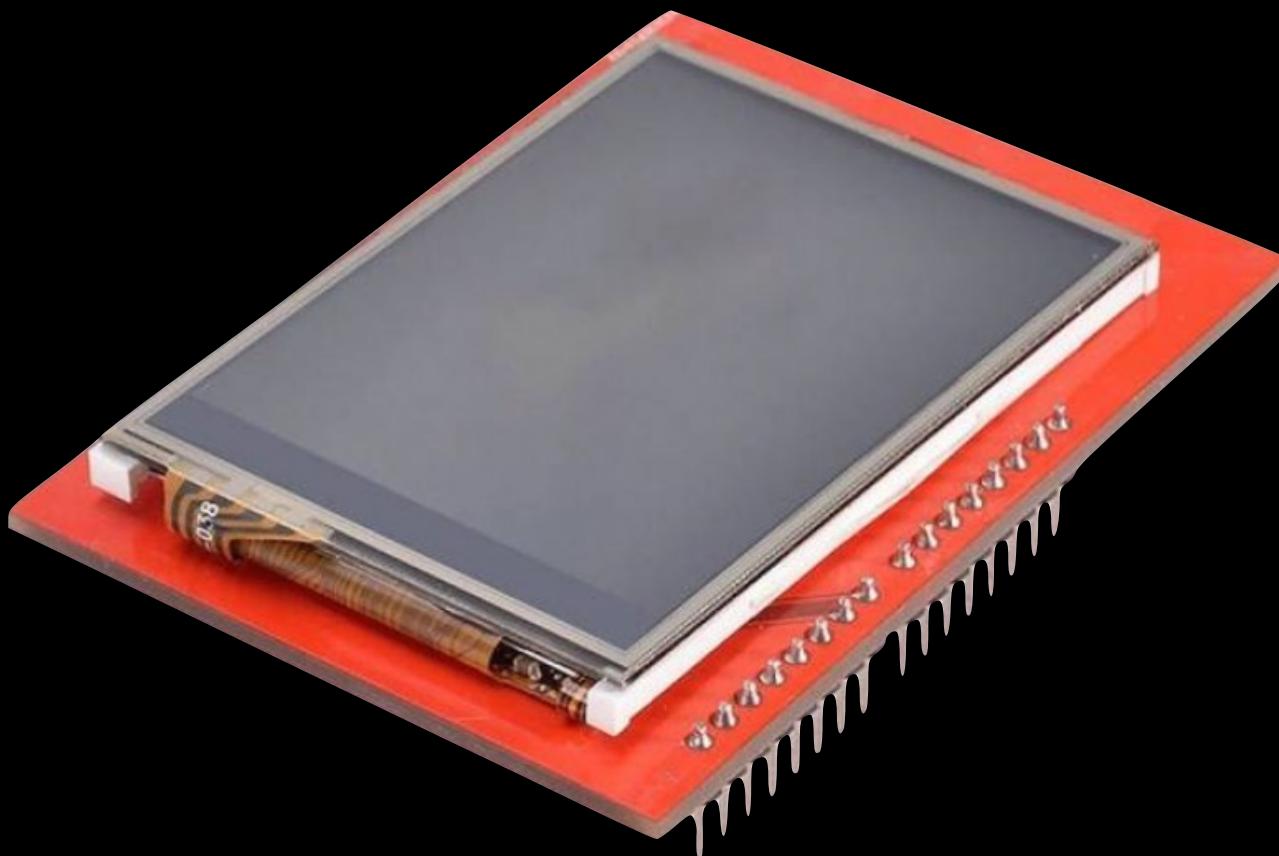
Shield LCD TFT 2.4"



Toque impreciso
com o dedo.



Tela Touch Resistiva



Não é mágica como a dos smartphones.

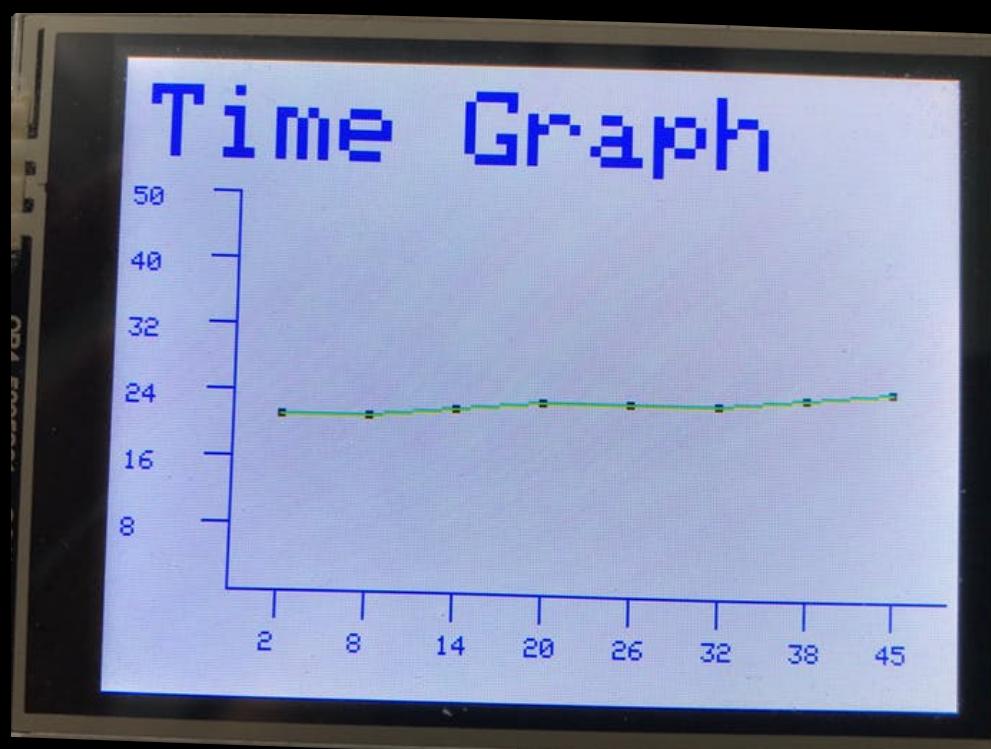
Não precisa de caneta, mas é recomendada.

Resolução de 320 x 240 pixels.

Não tem multitouch.

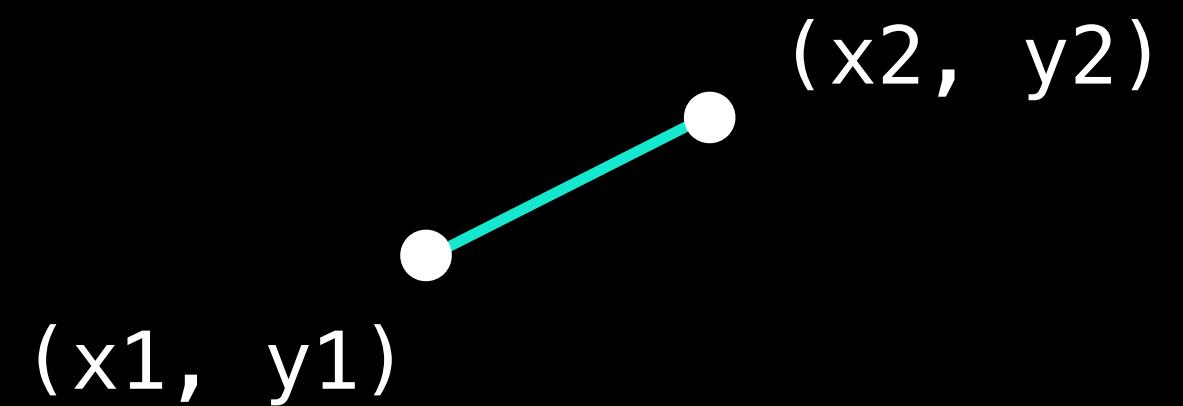
Sem patentes!

Shield LCD TFT



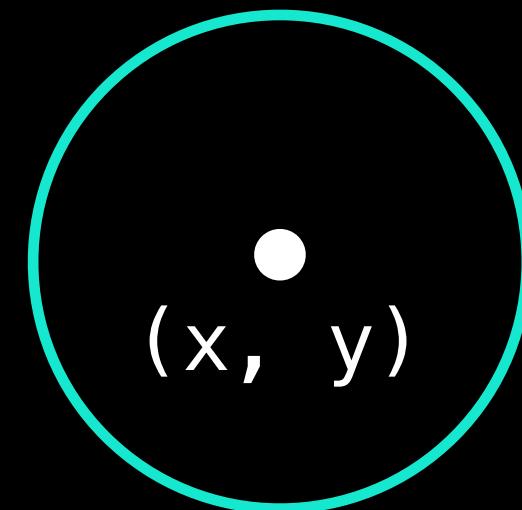
Exemplos de Interface no Shield LCD TFT

```
tela.drawLine(x1, y1, x2, y2, cor);
```



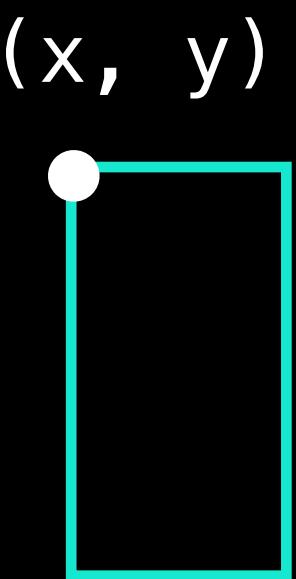
```
tela.fillCircle(x, y, raio, cor);
```

```
tela.drawCircle(x, y, raio, cor);
```



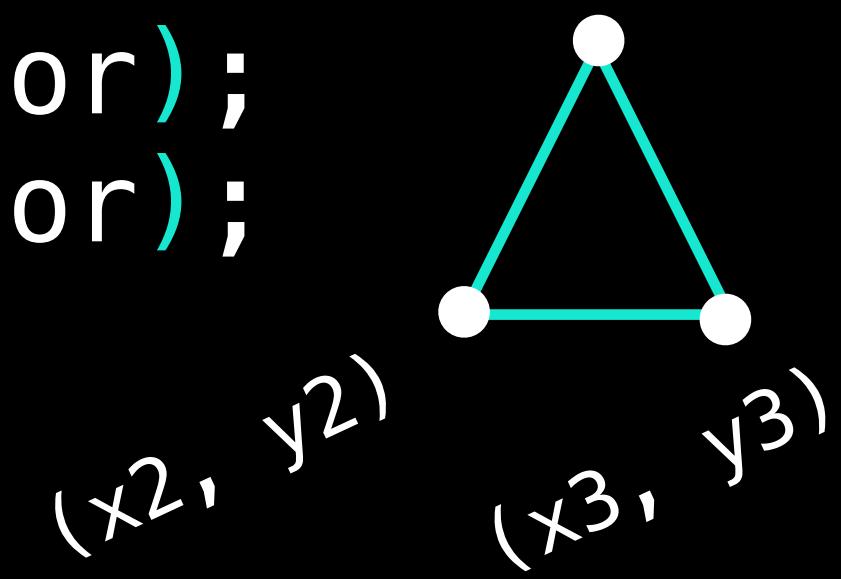
```
tela.fillRect(x, y, comprimento, altura, cor);
```

```
tela.drawRect(x, y, comprimento, altura, cor);
```



```
tela.fillTriangle(x1, y1, x2, y2, x3, y3, cor);
```

```
tela.drawTriangle(x1, y1, x2, y2, x3, y3, cor);
```



```

#include <Adafruit_GFX.h>
#include <MCUFRIEND_kbv.h>

MCUFRIEND_kbv tela;

void setup () {
    tela.begin( tela.readID() );
    tela.fillScreen(TFT_BLACK);

    tela.drawLine(10, 10, 60, 60, TFT_GREEN);

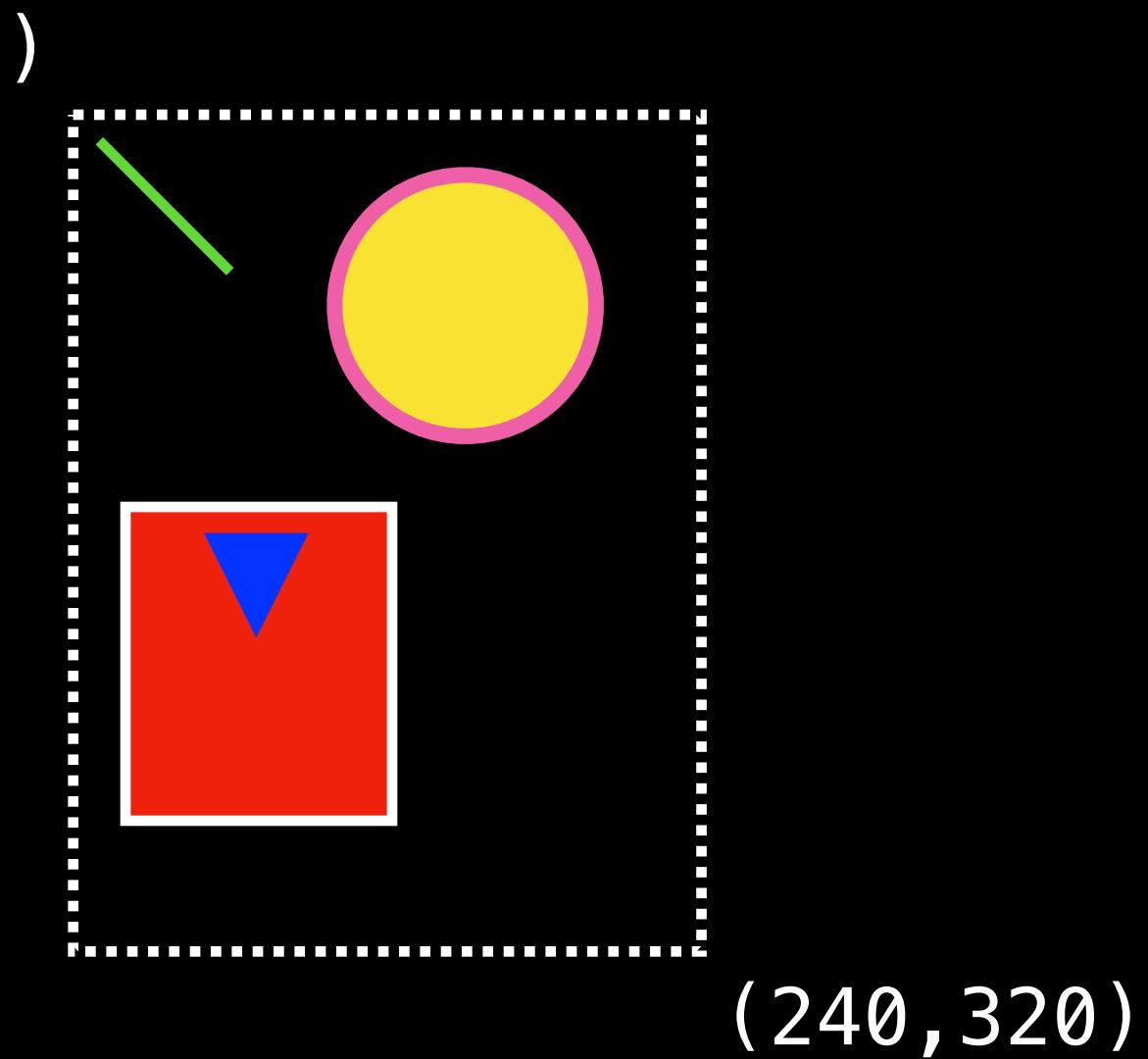
    tela.fillCircle(150, 70, 50, TFT_YELLOW);
    tela.drawCircle(150, 70, 50, TFT_PINK);

    tela.fillRect(20, 150, 100, 120, TFT_RED);
    tela.drawRect(20, 150, 100, 120, TFT_WHITE);

    tela.fillTriangle(50, 160, 70, 200, 90, 160, TFT_BLUE);
}

void loop () {
}

```



Exemplo com Funções de Desenho para Formas

(0,0)

```
#include <Adafruit_GFX.h>
#include <MCUFRIEND_kbv.h>

MCUFRIEND_kbv tela;

void setup () {
    tela.begin( tela.readID() );
    tela.fillRect(TFT_BLACK);

    tela.setCursor(20, 100);
    tela.setTextColor(TFT_YELLOW);
    tela.setTextSize(4);
    tela.print("Jan K. S.");

    tela.setCursor(20, 160);
    tela.setTextColor(TFT_CYAN);
    tela.setTextSize(3);
    tela.print("Microcontroladores");

}

void loop () {
```

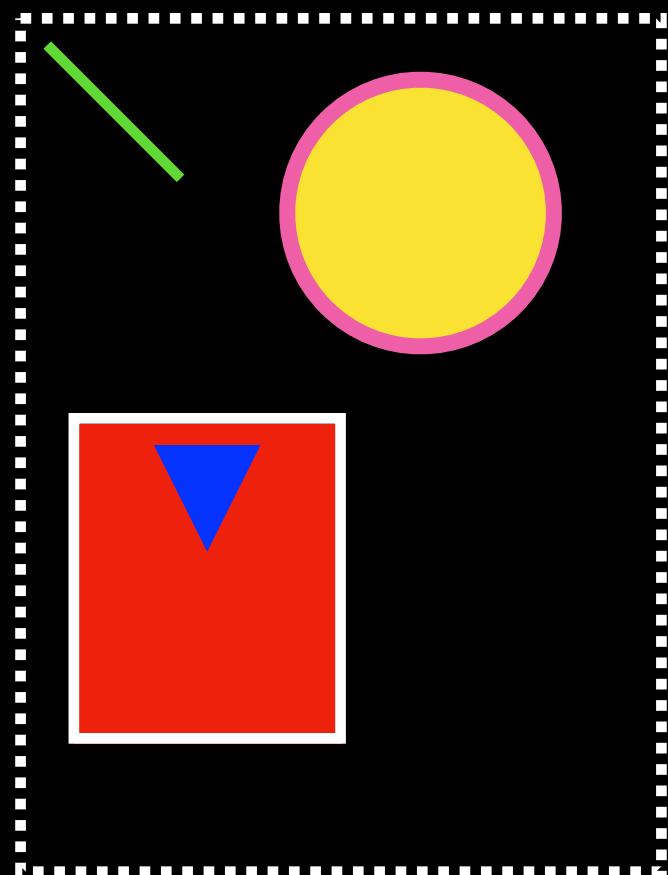
Jan K. S.

Microcontrol
adores

(240,320)

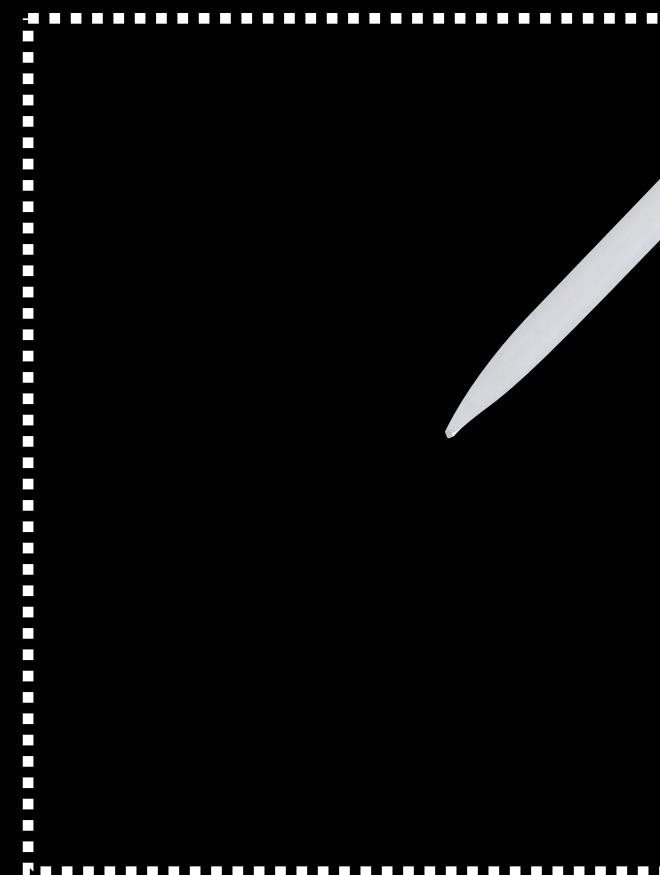
Exemplo com Funções de Desenho para Texto

$(0, 0)$



coordenadas
gráficas

$(145, 934)$



coordenadas
do touch

Coordenadas Gráficas vs Coordenadas do Sensor de Toque

```

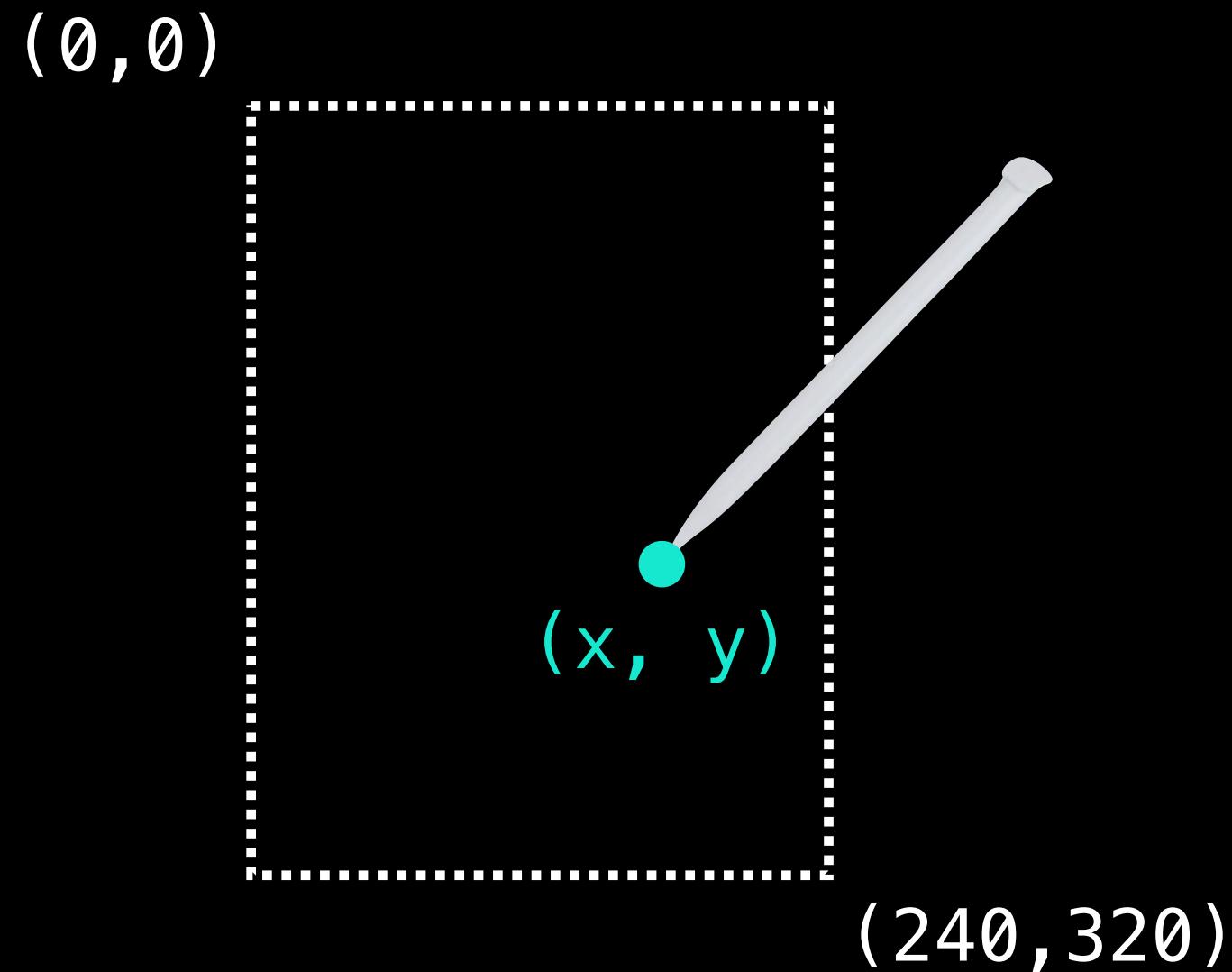
#include <TouchScreen.h>

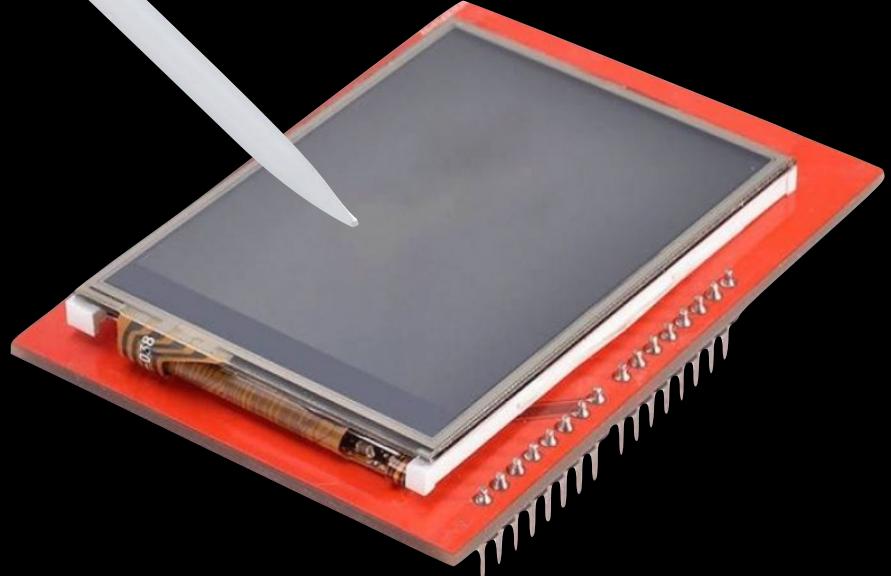
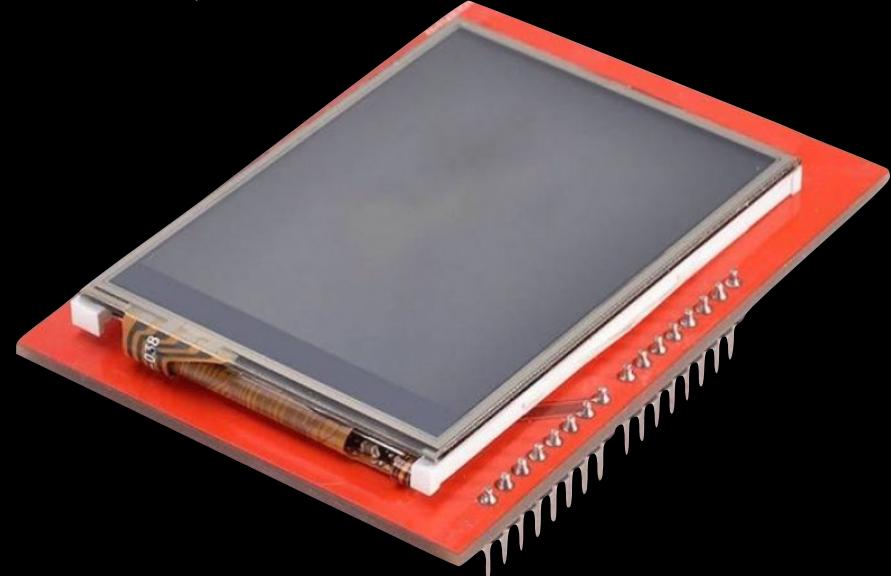
TouchScreen touch(6, A1, A2, 7, 300);
const int TS_LEFT = 145, TS_RT = 887,
          TS_TOP = 934, TS_BOT = 158;

void setup () {
    Serial.begin(9600);
}

void loop () {
    TSPoint ponto = touch.getPoint();
    pinMode(A1, OUTPUT); digitalWrite(A1, HIGH); // reconfigura pinos
    pinMode(A2, OUTPUT); digitalWrite(A2, HIGH); // para desenho
    int forca = ponto.z; // força aplicada na tela
    if (forca > 200 && forca < 1000) {
        int x = map(ponto.x, TS_LEFT, TS_RT, 0, 240);
        int y = map(ponto.y, TS_TOP, TS_BOT, 0, 320);
        Serial.println(x + String(",") + y);
    } else {
        Serial.println(String("forca = ") + forca);
    }
}

```





```
forca = 0  
forca = 0
```

Auto-rolagem Show timestamp

```
103,108  
103,108  
forca = 0  
forca = 0  
forca = 0  
103,108  
forca = 0  
forca = 0  
forca = 0  
forca = 0  
103,109  
103,109  
103,108  
forca = 0  
forca = 0
```

Auto-rolagem Show timestamp



Toque fixo gera um tipo de bounce constante...

"Bounce" do Touch

```

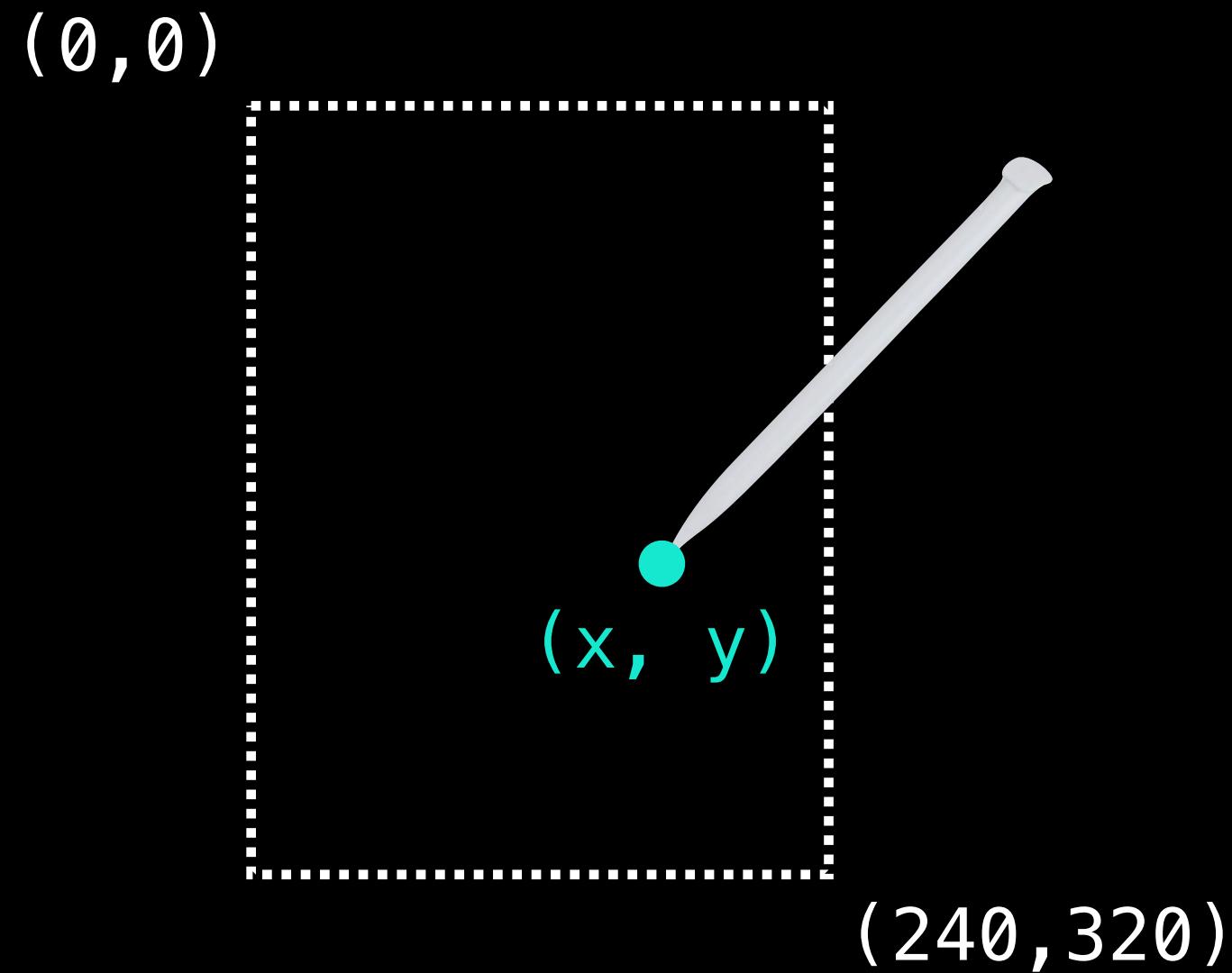
#include <TouchScreen.h>

TouchScreen touch(6, A1, A2, 7, 300);
const int TS_LEFT = 145, TS_RT = 887,
         TS_TOP = 934, TS_BOT = 158;
unsigned long instanteAnterior;

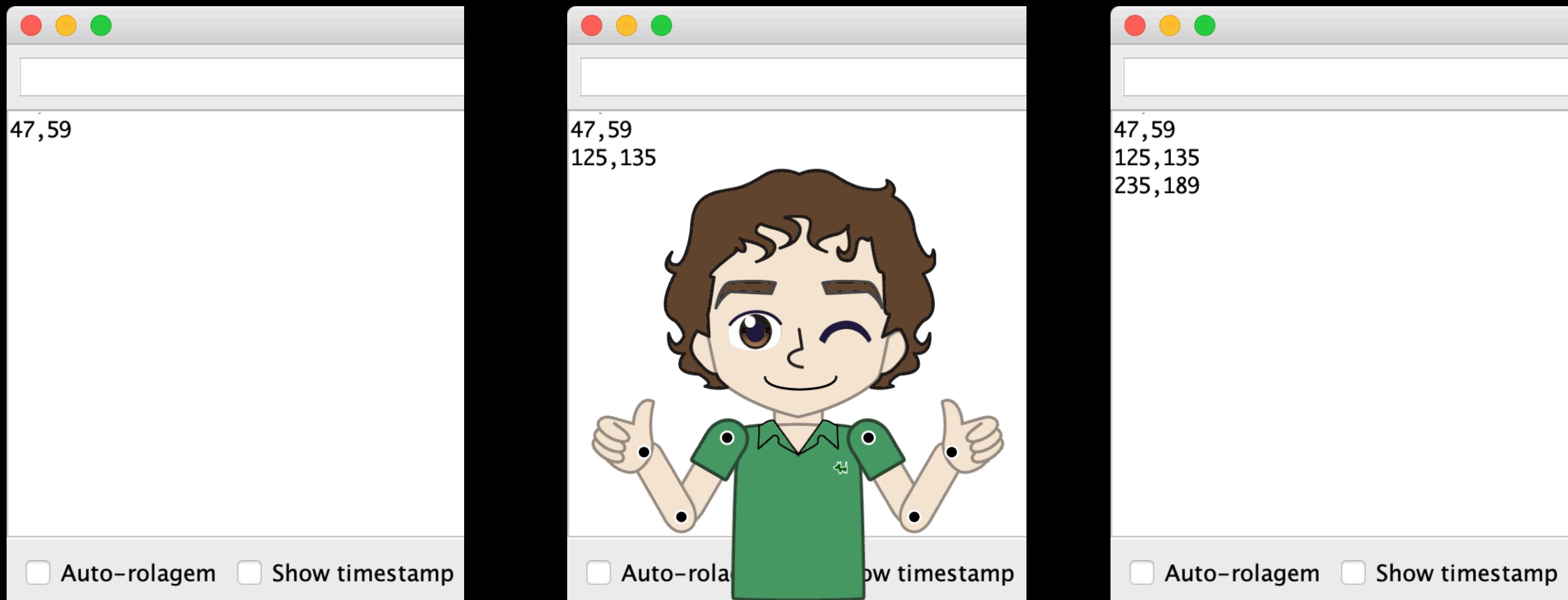
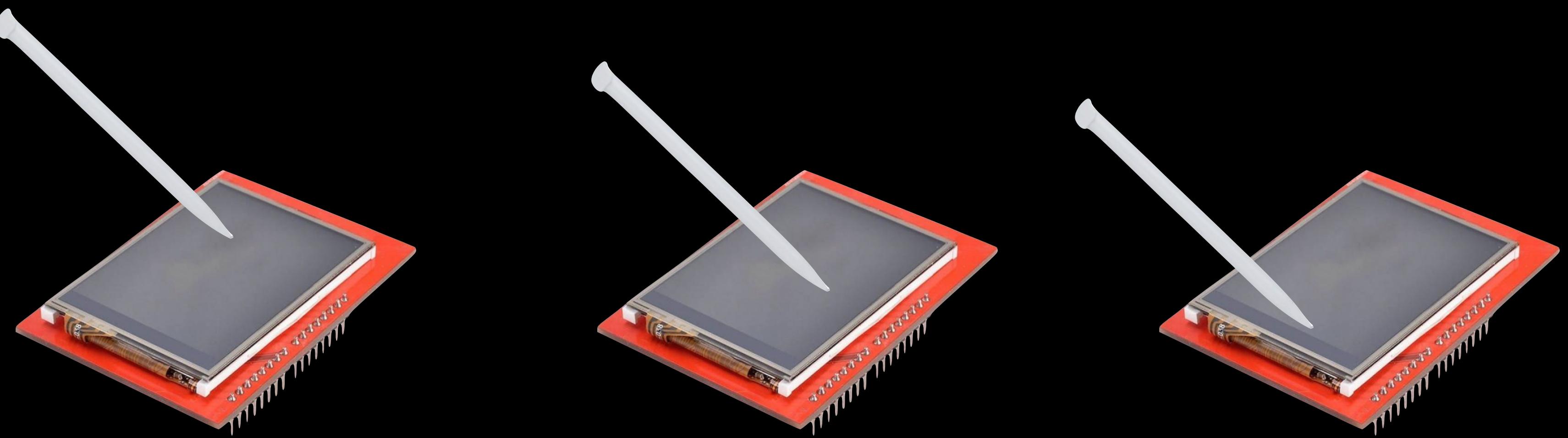
void setup () {
    Serial.begin(9600);
}

void loop () {
    TSPoint ponto = touch.getPoint();
    pinMode(A1, OUTPUT); digitalWrite(A1, HIGH); // reconfigura pinos
    pinMode(A2, OUTPUT); digitalWrite(A2, HIGH); // para desenho
    int forca = ponto.z; // força aplicada na tela
    if (forca > 200 && forca < 1000) {
        if (millis() - instanteAnterior > 300) {
            int x = map(ponto.x, TS_LEFT, TS_RT, 0, 240);
            int y = map(ponto.y, TS_TOP, TS_BOT, 0, 320);
            Serial.println(x + String(",") + y);
        }
        instanteAnterior = millis();
    }
}

```



Detecção do Ponto de Toque sem Repetição e "Bounce"

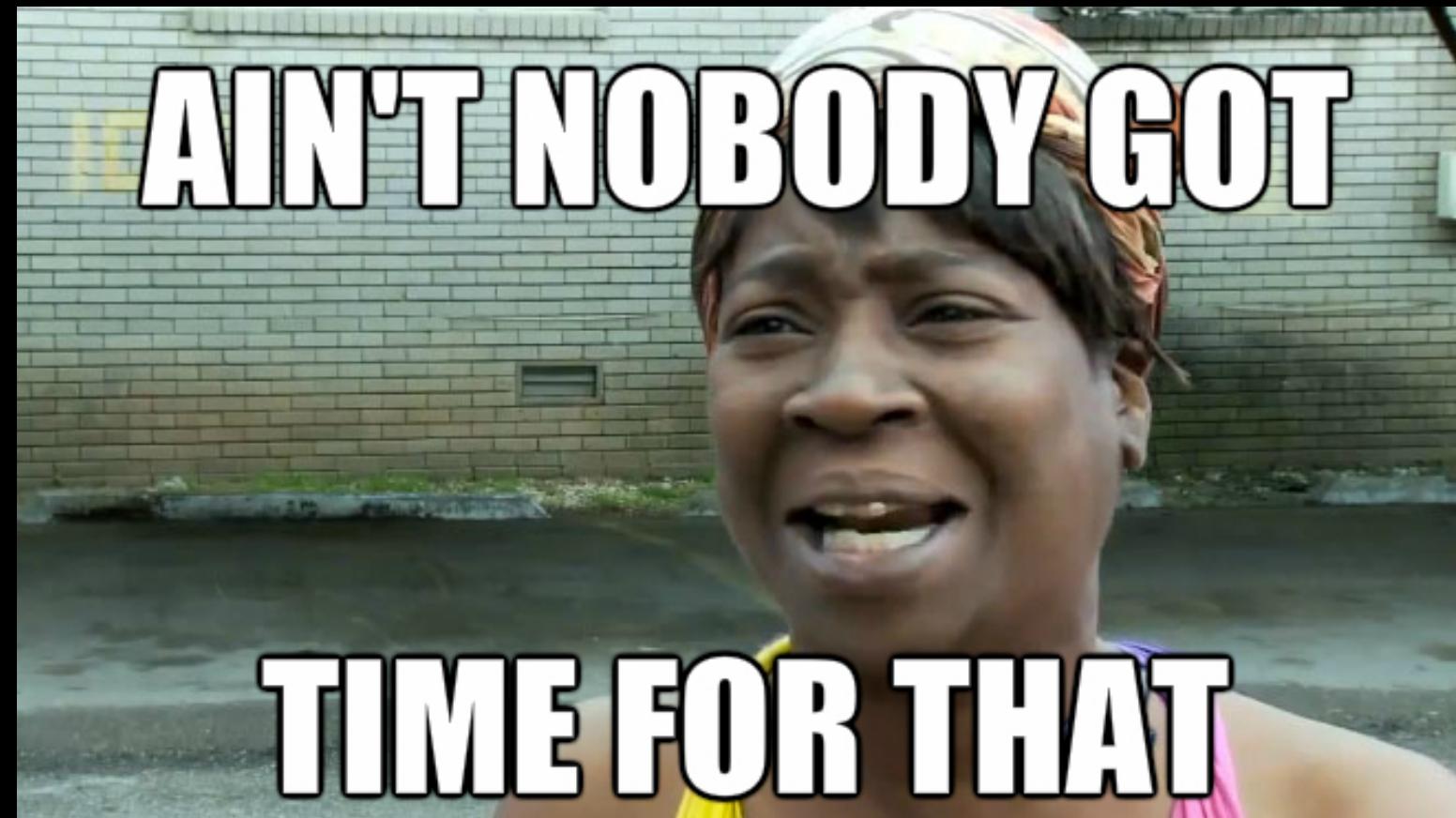


Detecção do Ponto de Toque sem Repetição e "Bounce"

Botão 1

Botão 1

- Desenhar um retângulo com texto dentro
- Desenhar versão do botão com toque
- Lidar com "bounce" do toque na tela
- Lidar com eventos de apertar, segurar e soltar



Adafruit_GFX_Button + GFButton = JKSBUTTON

```
JKSBUTTON botao;  
  
botao.init(  
    &tela, &touch,  
    xDoCentro, yDoCentro, comprimento, altura,  
    corDaBorda, corDoFundo, corDoTexto,  
    texto, tamanhoDoTexto  
);  
botao.setPressHandler(funcao1);  
botao.setReleaseHandler(funcao2);  
  
...  
  
botao.process()
```

```

#include <Adafruit_GFX.h>
#include <MCURIEND_kbv.h>
#include <TouchScreen.h>
#include <JKSButton.h>

MCURIEND_kbv tela; TouchScreen touch(6, A1, A2, 7, 300);
JKSButton botao;

void setup () {
    tela.begin( tela.readID() );
    tela.fillScreen(TFT_BLACK);

    botao.init(&tela, &touch, 120, 70, 200, 100, TFT_WHITE, TFT_PURPLE,
TFT_BLACK, "Botao 1", 2);
    botao.setPressHandler(desenhaRetangulo);
    botao.setReleaseHandler(apagaRetangulo);
}

void loop () {
    botao.process();
}

void desenhaRetangulo (JKSButton &botaoPressionado) {
    tela.fillRect(50, 200, 140, 70, TFT_RED);
}
void apagaRetangulo (JKSButton &botaoPressionado) {
    tela.fillRect(50, 200, 140, 70, TFT_BLACK);
}

```

(0,0)

Botao 1

(240,320)

(0,0)

Botao 1

(240,320)

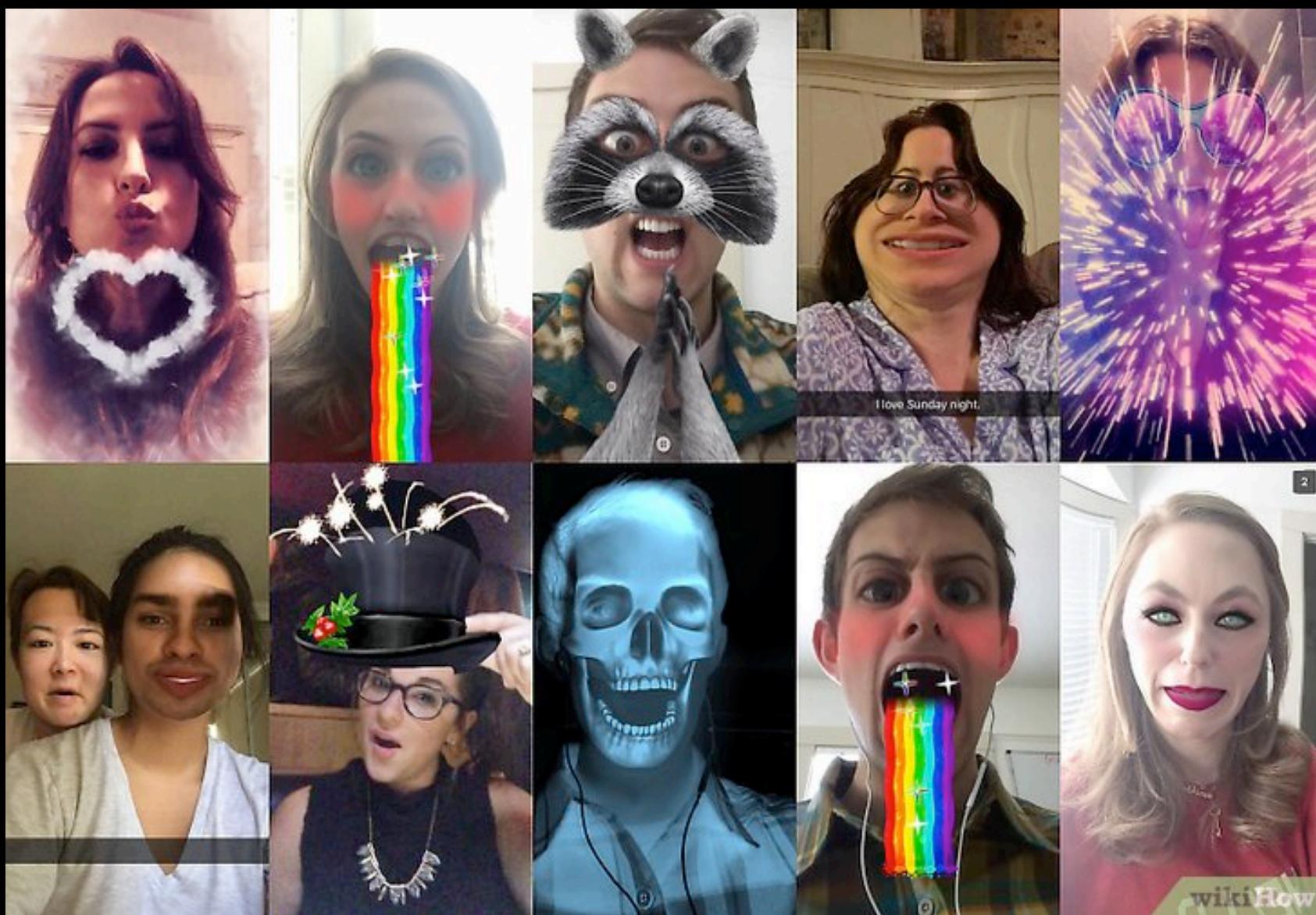
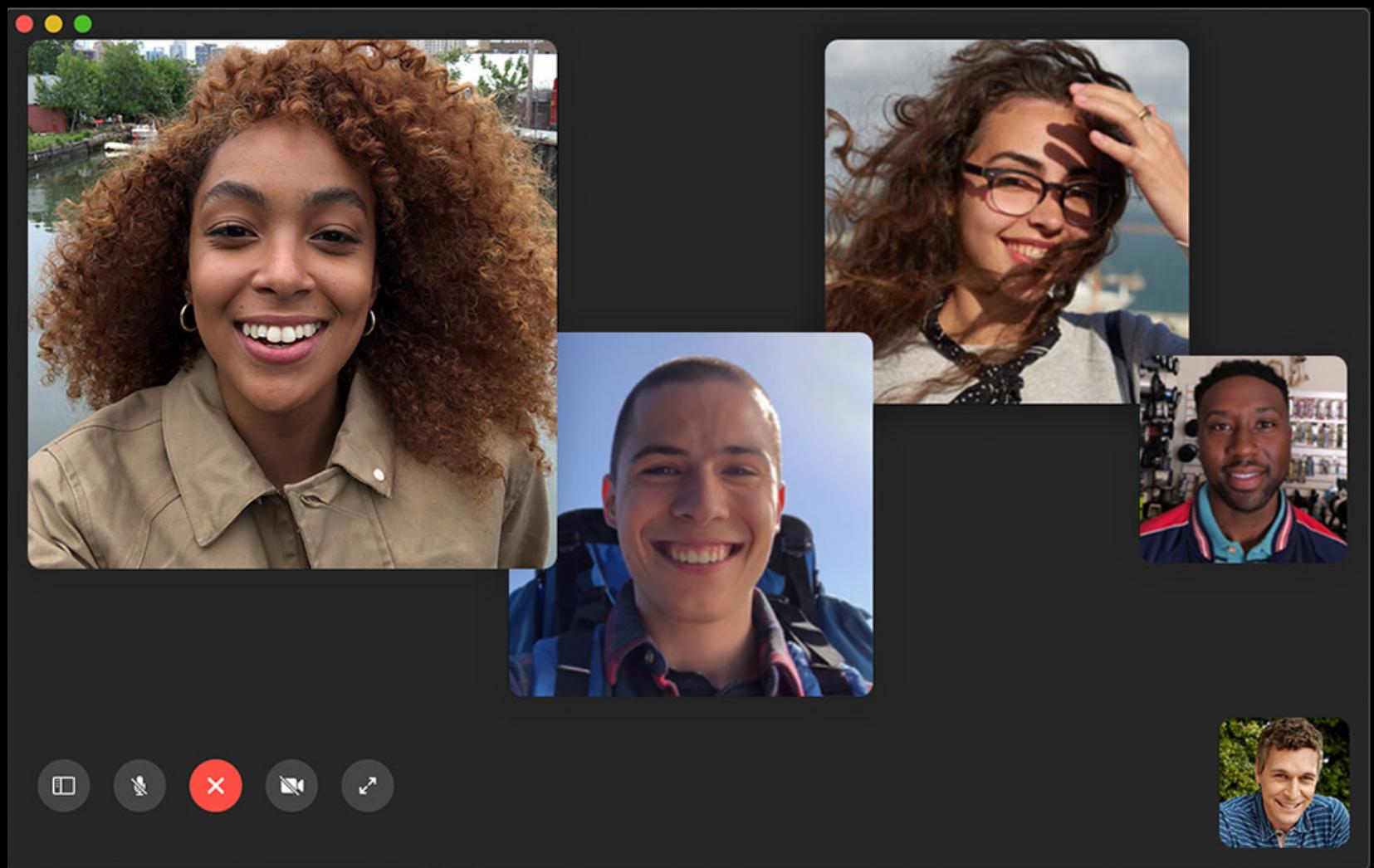
Exemplo de Botão com a JKSButton

Software



Como faz para
gravar vídeo?

De Volta à Webcam



Efeitos de Vídeo

Página principal Conteúdo destacado Eventos atuais Esplanada Página aleatória Portais Informar um erro Loja da Wikipédia

Colaboração Boas-vindas Ajuda Página de testes Portal comunitário Mudanças recentes Manutenção Criar página Páginas novas Contato Donativos

Noutros projetos Wikimedia Commons Imprimir/exportar

pt.wikipedia.org/wiki/OpenCV

OpenCV

[ocultar]

Origem: Wikipédia, a enciclopédia livre.

 As referências deste artigo **necessitam de formatação** (desde fevereiro de 2014). Por favor, utilize **fontes apropriadas** contendo referência ao título, autor, data e fonte de publicação do trabalho para que o artigo permaneça **verificável** no futuro.

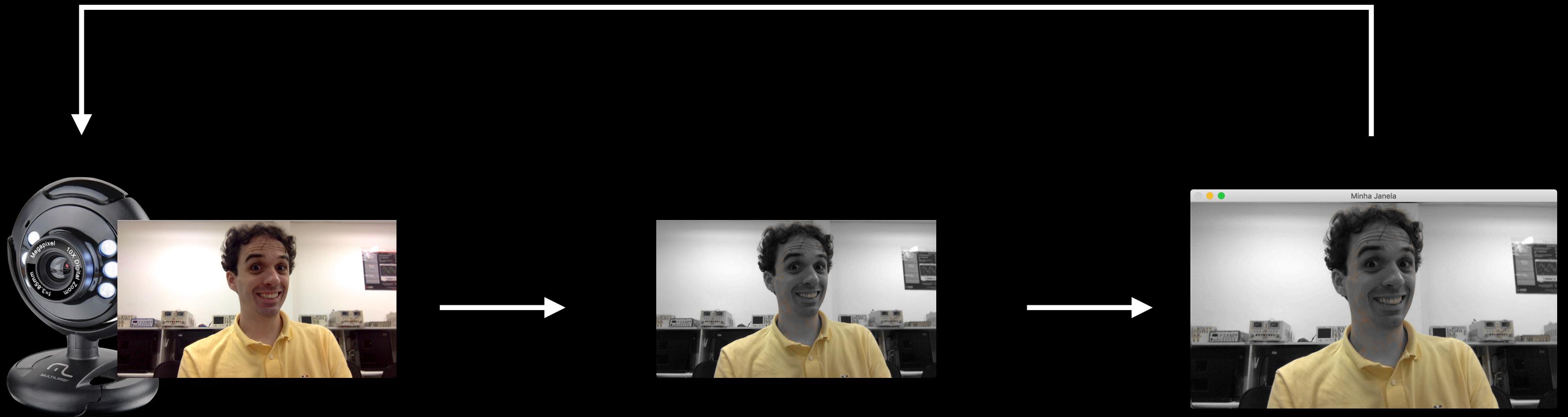
OpenCV (*Open Source Computer Vision Library*). Originalmente, desenvolvida pela **Intel**, em **2000**, é uma **biblioteca** multiplataforma, totalmente livre ao uso **acadêmico** e **comercial**, para o desenvolvimento de aplicativos na área de **Visão computacional**, bastando seguir o modelo de **licença BSD Intel**. O **OpenCV** possui módulos de **Processamento de Imagens** e **Video I/O**, **Estrutura de dados**, **Álgebra Linear**, **GUI** (Interface Gráfica do Usuário) **Básica** com sistema de janelas independentes, Controle de mouse e teclado, além de mais de 350 algoritmos de **Visão computacional** como: Filtros de **imagem**, **calibração de câmera**, **reconhecimento de objetos**, **análise estrutural** e outros. O seu **processamento** é em **tempo real** de **imagens**.

Esta biblioteca foi desenvolvida nas linguagens de **programação C/C++**. Também, dá suporte a **programadores** que utilizem **Java**, **Python** e **Visual Basic** e desejam incorporar a **biblioteca** a seus aplicativos. A versão 1.0 foi lançada no final de **2006** e a 2.0 foi lançada em setembro de **2009**.


OpenCV

Autor

Intel Corporation, Willow Garage, Itseez



captura quadro do
vídeo como imagem

processa imagem

exibe imagem

Processamento de Vídeo em Tempo Real

```
from cv2 import *
stream = VideoCapture(0)
while True:
    _, imagem = stream.read()
    imshow("Minha Janela", imagem)
    # mostra a imagem durante 1 milissegundo
    # e interrompe loop quando tecla q for pressionada
    if waitKey(1) & 0xFF == ord("q"):
        break
stream.release()
destroyAllWindows()
```



```
from cv2 import *

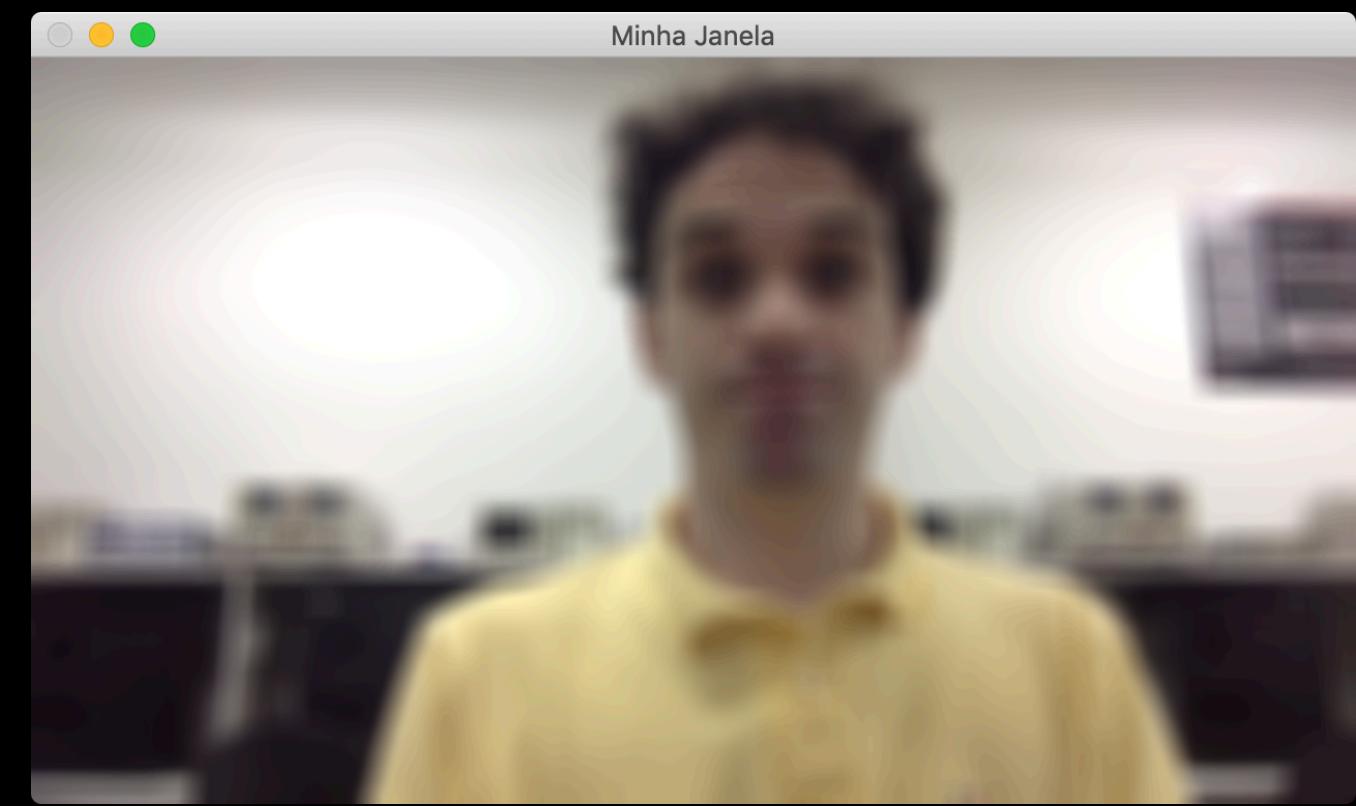
stream = VideoCapture(0)

while True:
    _, imagem = stream.read()

    imagem2 = blur(imagem, (50, 50))
    imshow("Minha Janela", imagem2

    if waitKey(1) & 0xFF == ord("q"):
        break

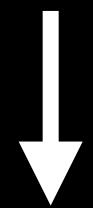
stream.release()
destroyAllWindows()
```



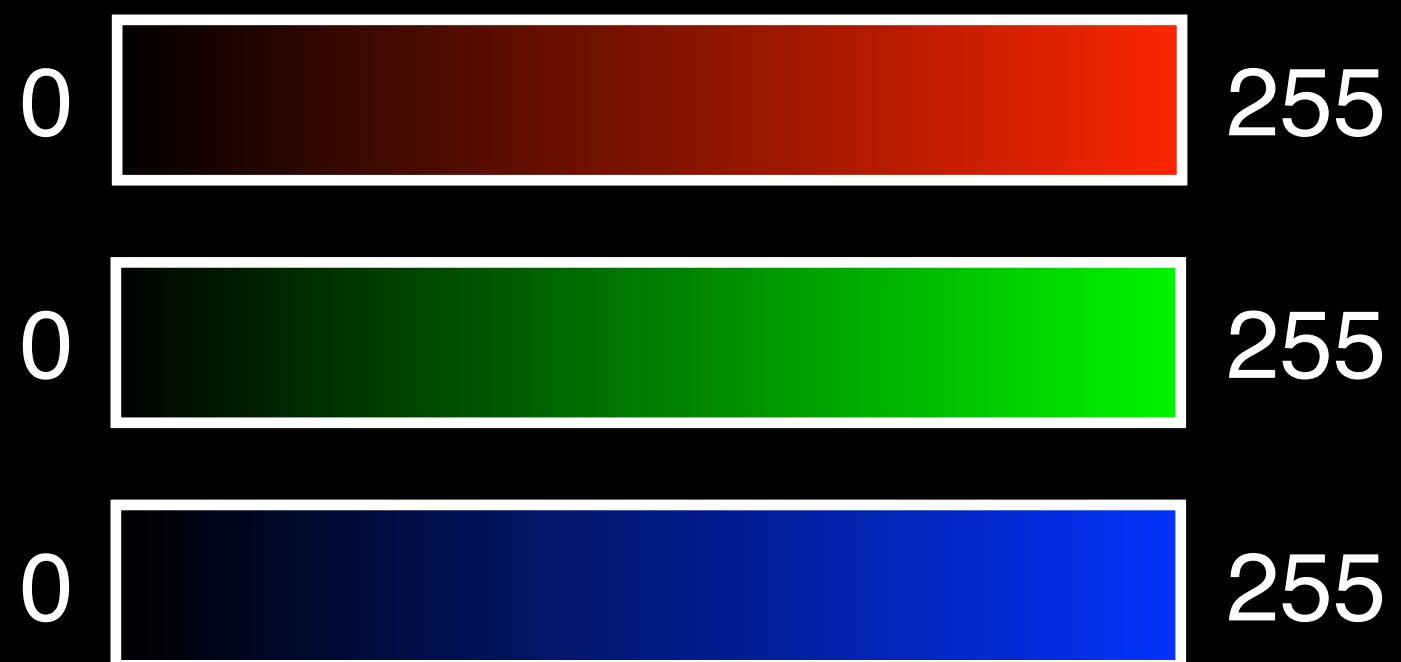
*intensidade do desfoque
horizontal e vertical*



= número?



Espaço de Cores RGB

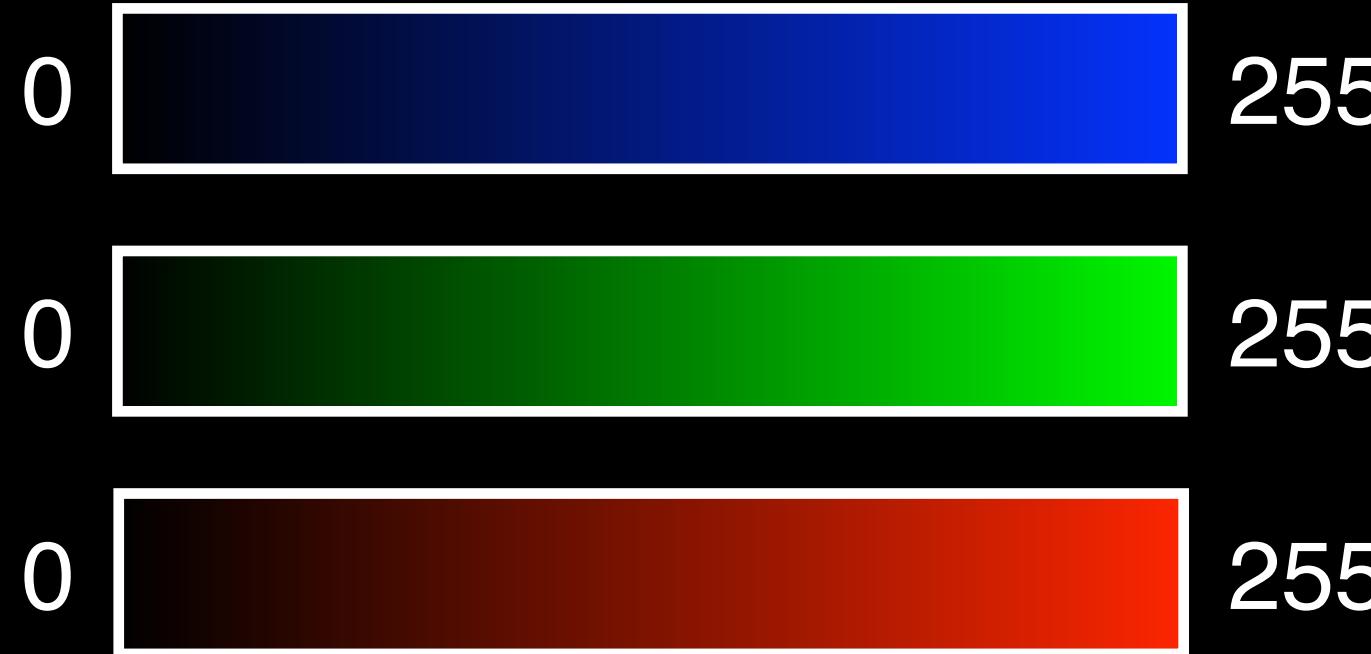


The diagram illustrates the decomposition of a pink color into its RGB components. On the left is a large pink square. To its right is an equals sign (=). Following the equals sign are four smaller squares: a red square, a green square, and a blue square, each followed by a plus sign (+). Below the red square is the number 215. Below the green square is the number 83. Below the blue square is the number 149.

$$\text{Pink} = \text{Red} + \text{Green} + \text{Blue}$$

215 83 149

Componentes da Cor no Espaço RGB



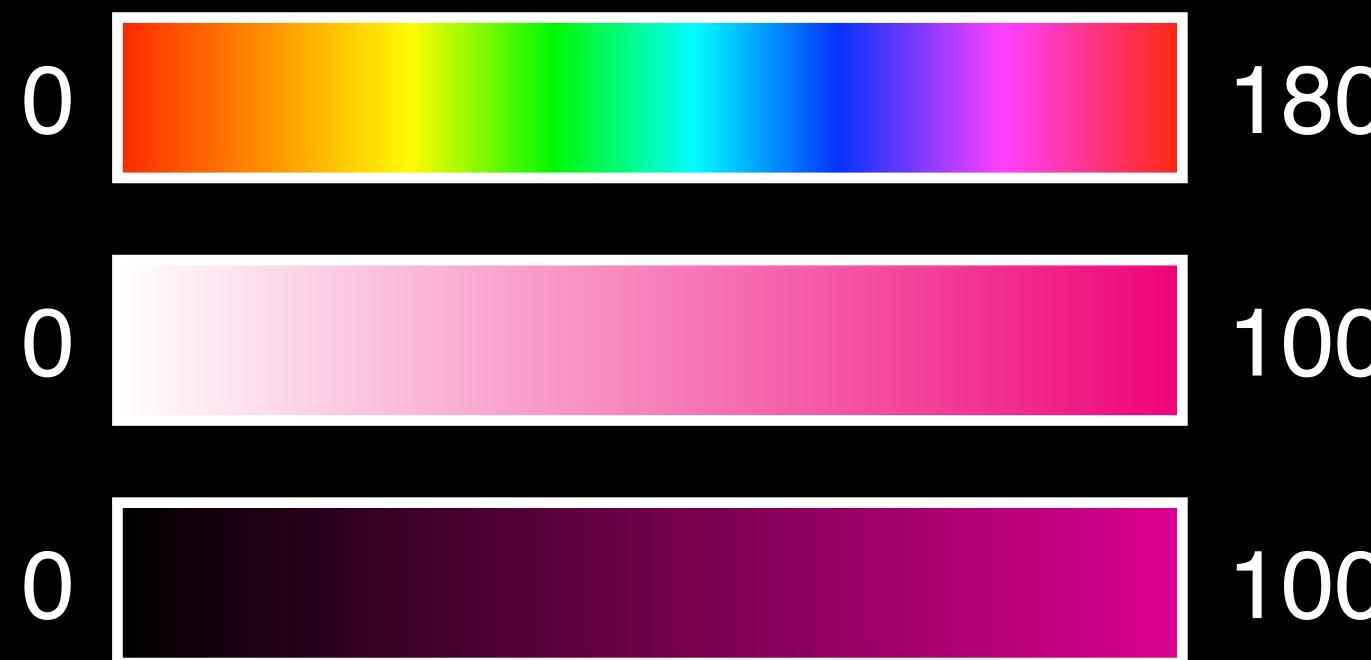
Espaço BGR

componentes azul + verde + vermelha



Espaço Gray

componente escala de cinza



Espaço HSV (ou HSB)

componentes matiz, saturação e valor de brilho

```
from cv2 import *

stream = VideoCapture(0)

while True:
    _, imagem = stream.read()

    imagem2 = cvtColor(imagem, COLOR_BGR2GRAY)
    imagem2 = cvtColor(imagem2, COLOR_GRAY2BGR)

    imshow("Minha Janela", imagem2)

    if waitKey(1) & 0xFF == ord("q"):
        break

stream.release()
destroyAllWindows()
```



volto para o espaço BGR para poder desenhar coisas coloridas depois, mas imagem a continua acimentada mesmo assim

Efeito Preto e Branco na Imagem

```
from cv2 import *

stream = VideoCapture(0)

while True:
    _, imagem = stream.read()

    imagem2 = blur(imagem, (30, 30))
    imagem3 = cvtColor(imagem2, COLOR_BGR2GRAY)
    imagem3 = cvtColor(imagem3, COLOR_GRAY2BGR)

    imshow("Minha Janela", imagem3)

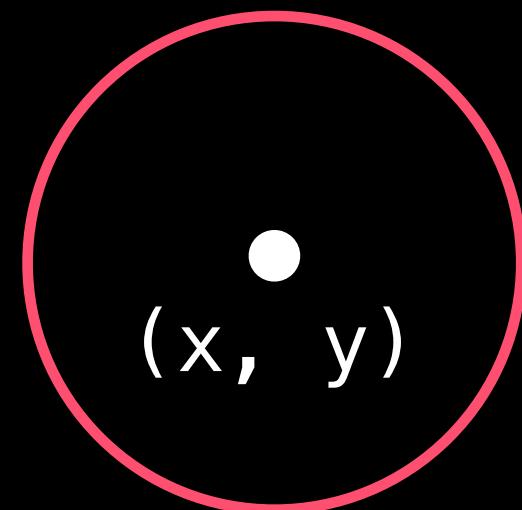
    if waitKey(1) & 0xFF == ord("q"):
        break

stream.release()
destroyAllWindows()
```

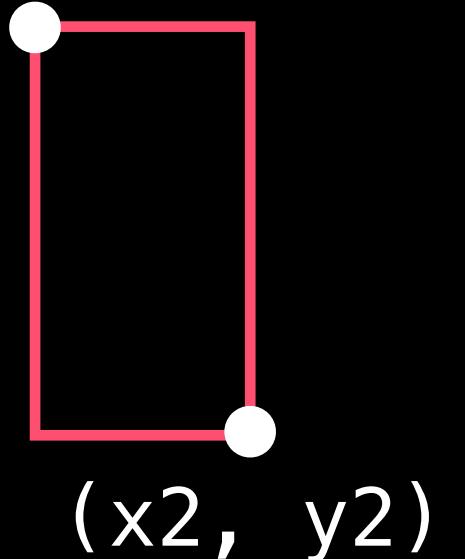


volto para o espaço BGR para poder
desenhar coisas coloridas depois, mas
imagem a continua acimentada mesmo assim

```
circle(imagem, (x,y), raio,  
       color=(r,g,b), thickness=espessura)
```

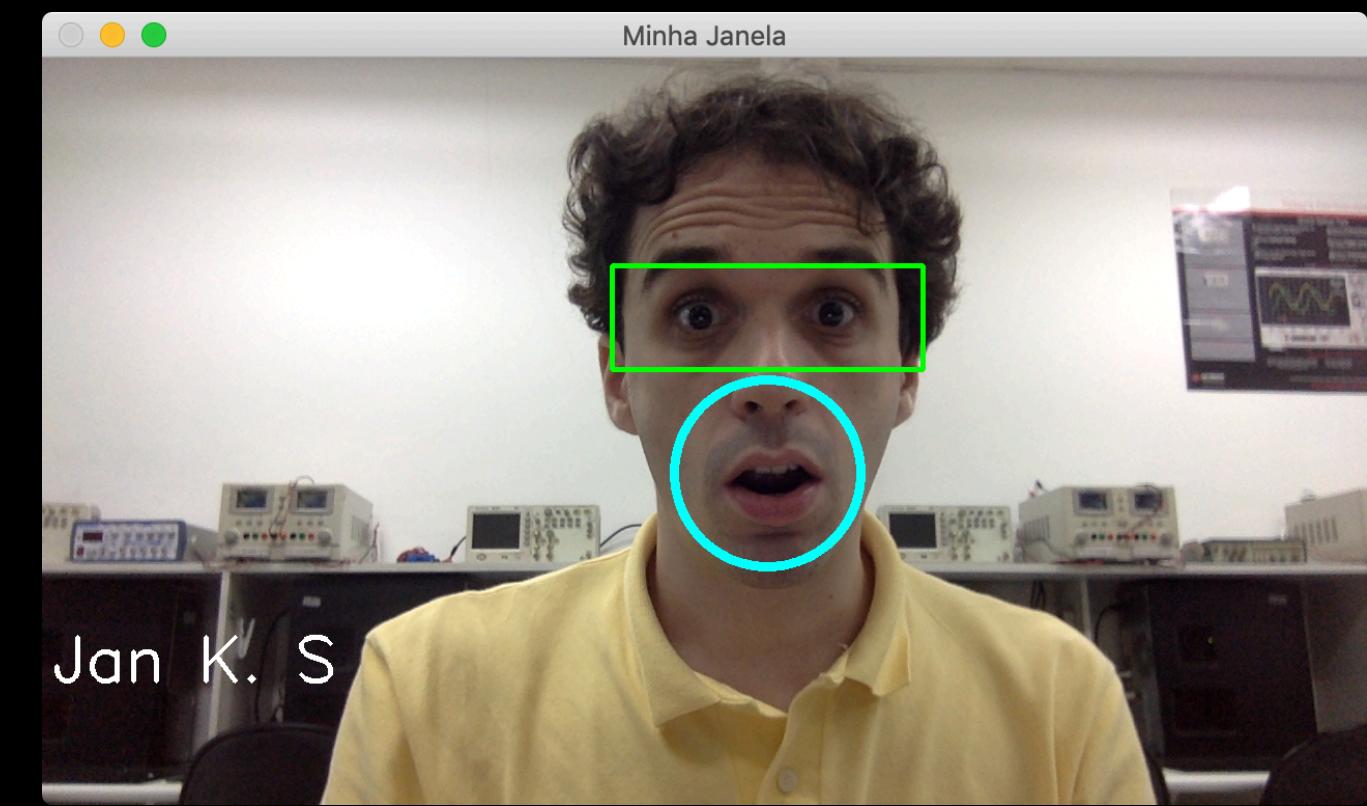


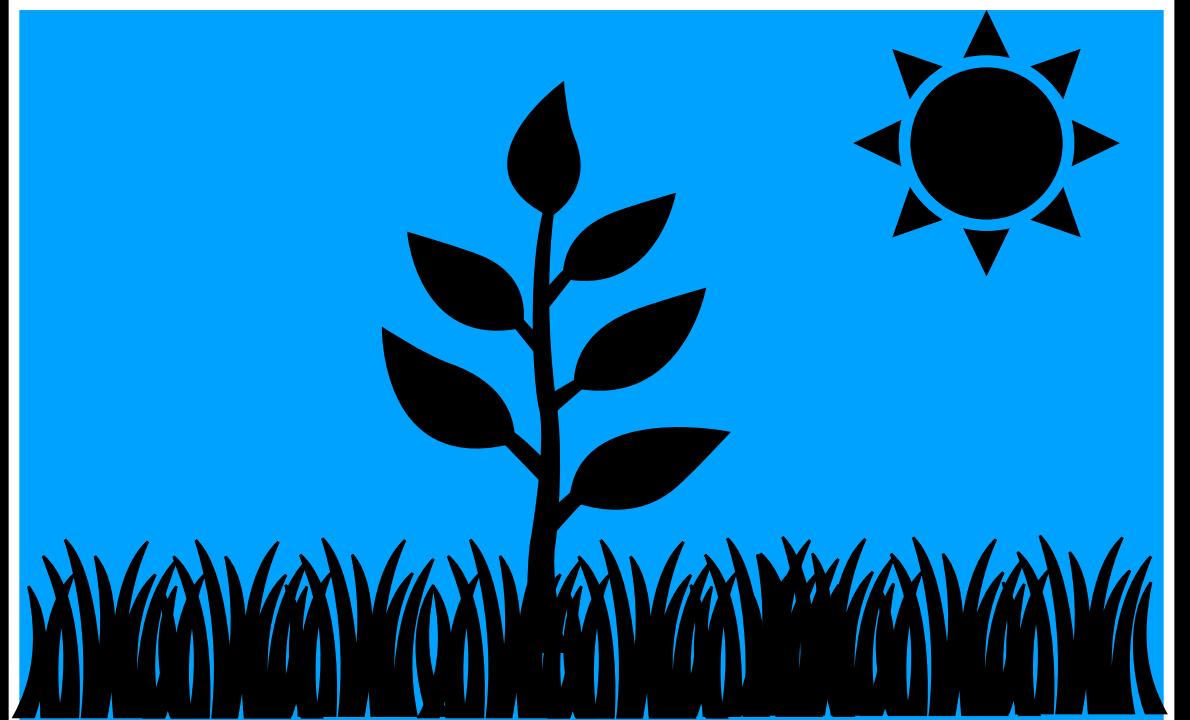
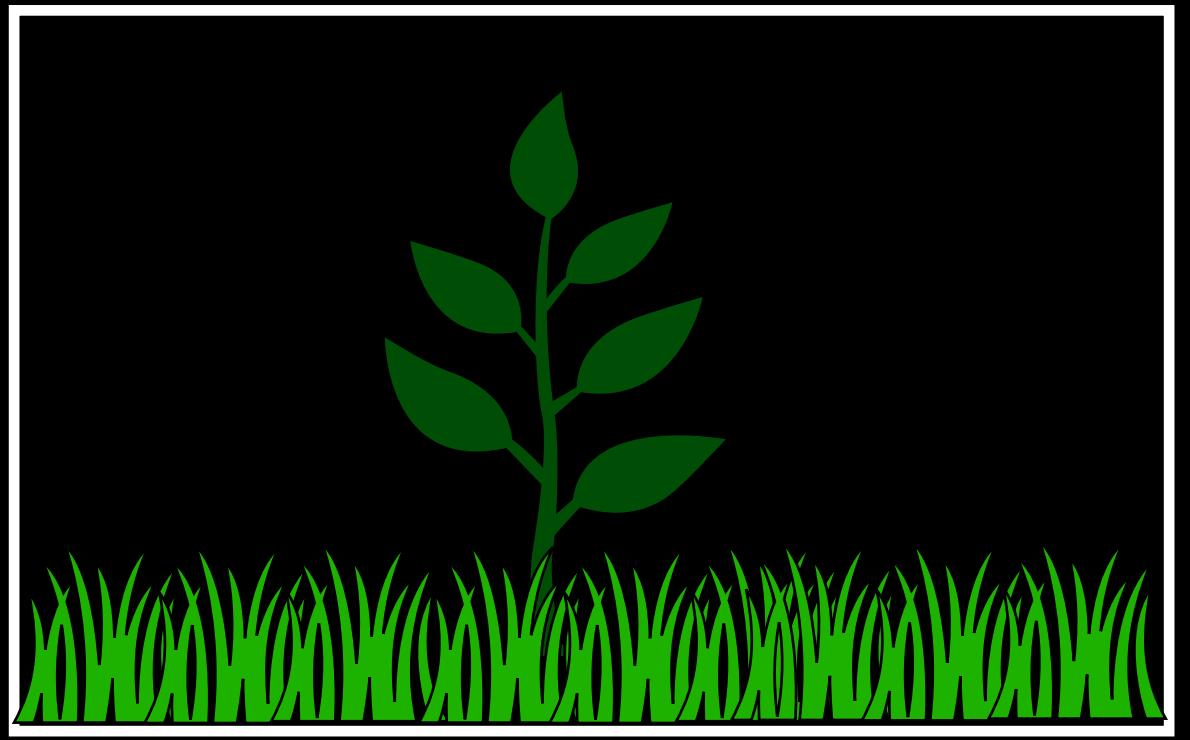
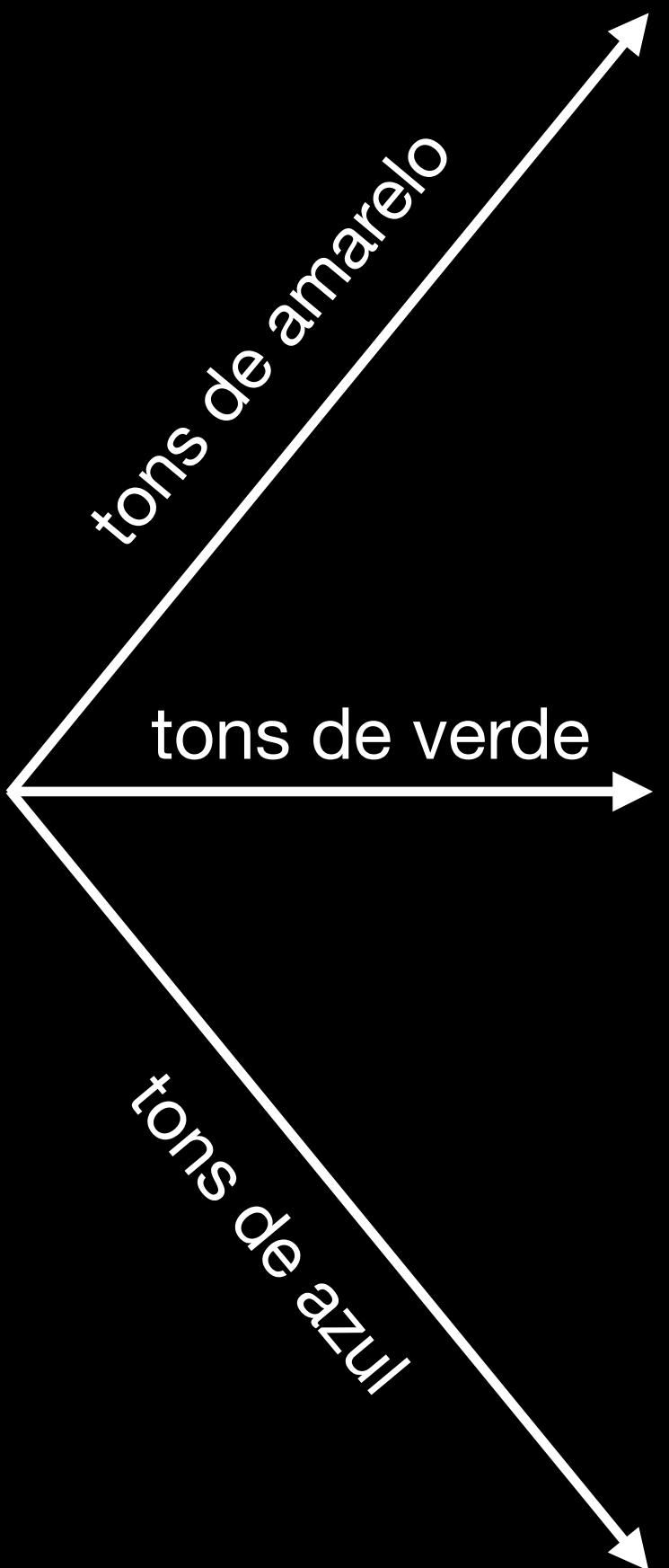
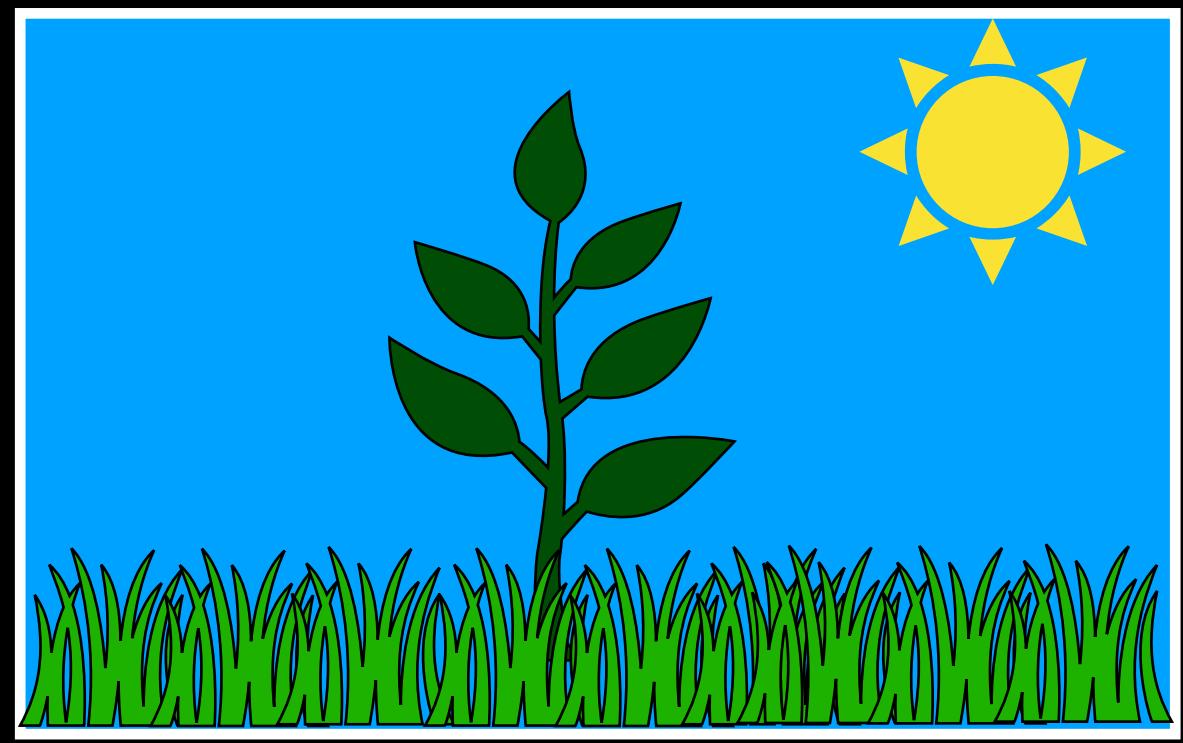
```
rectangle(imagem, pt1=(x1,y1), pt2=(x2,y2),  
          color=(r,g,b), thickness=espessura);
```



```
putText(imagem, "texto", (x,y),  
        color=(r,g,b), thickness=espessura,  
        fontFace=FONT_HERSHEY_SIMPLEX, fontScale=tamanho)
```

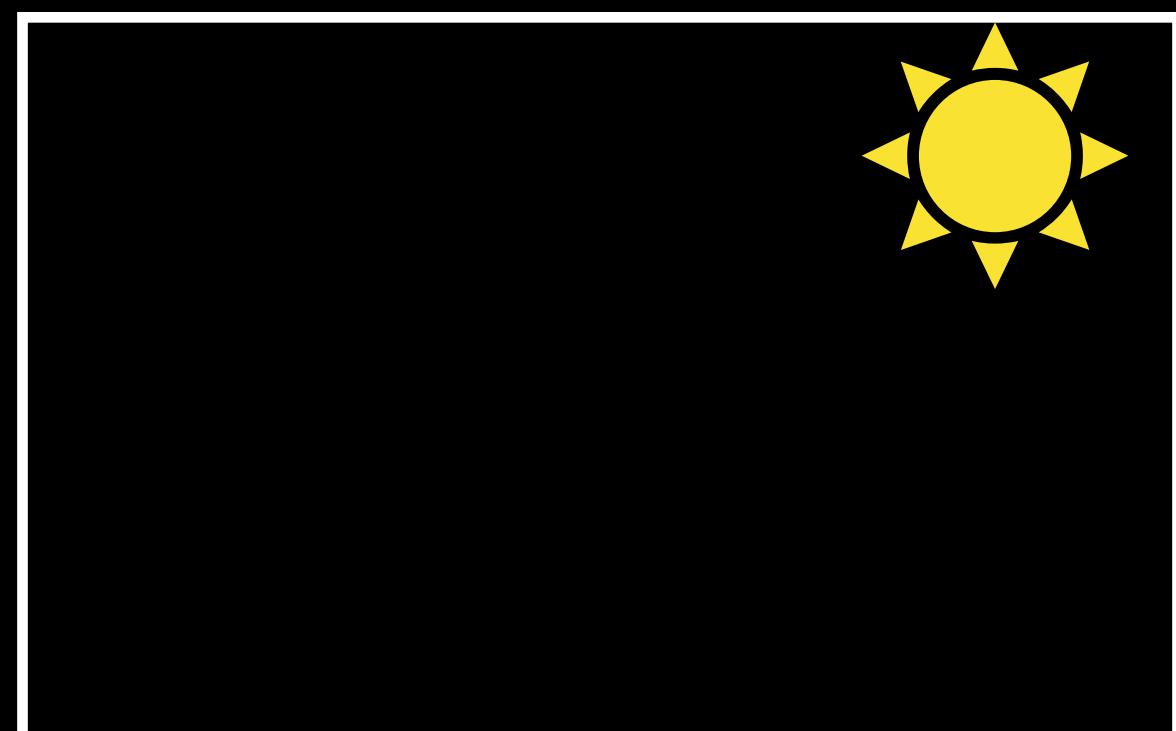
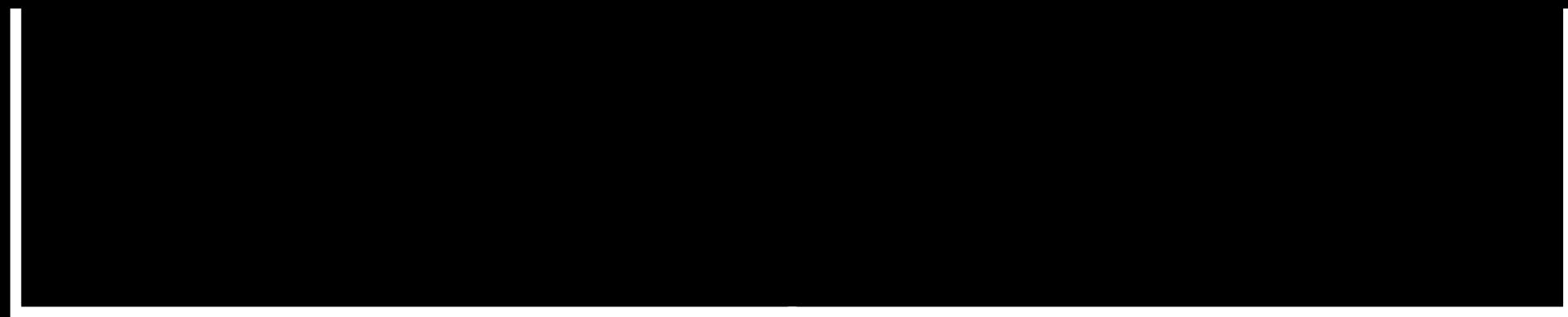
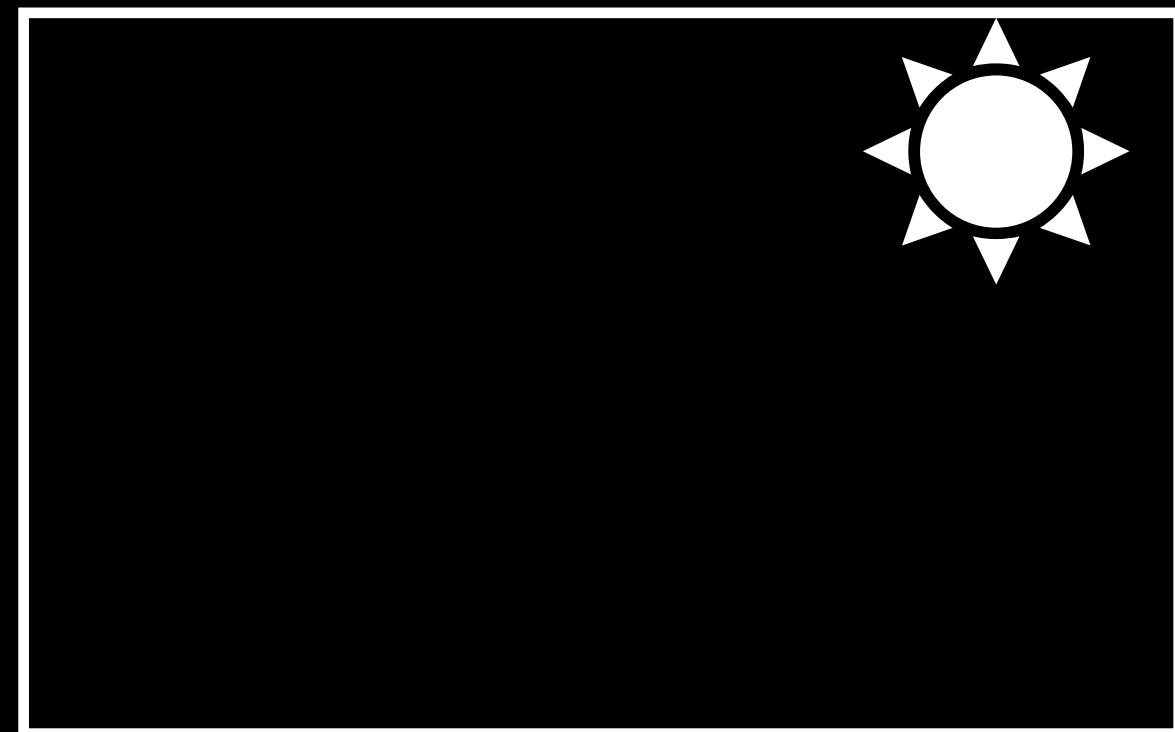
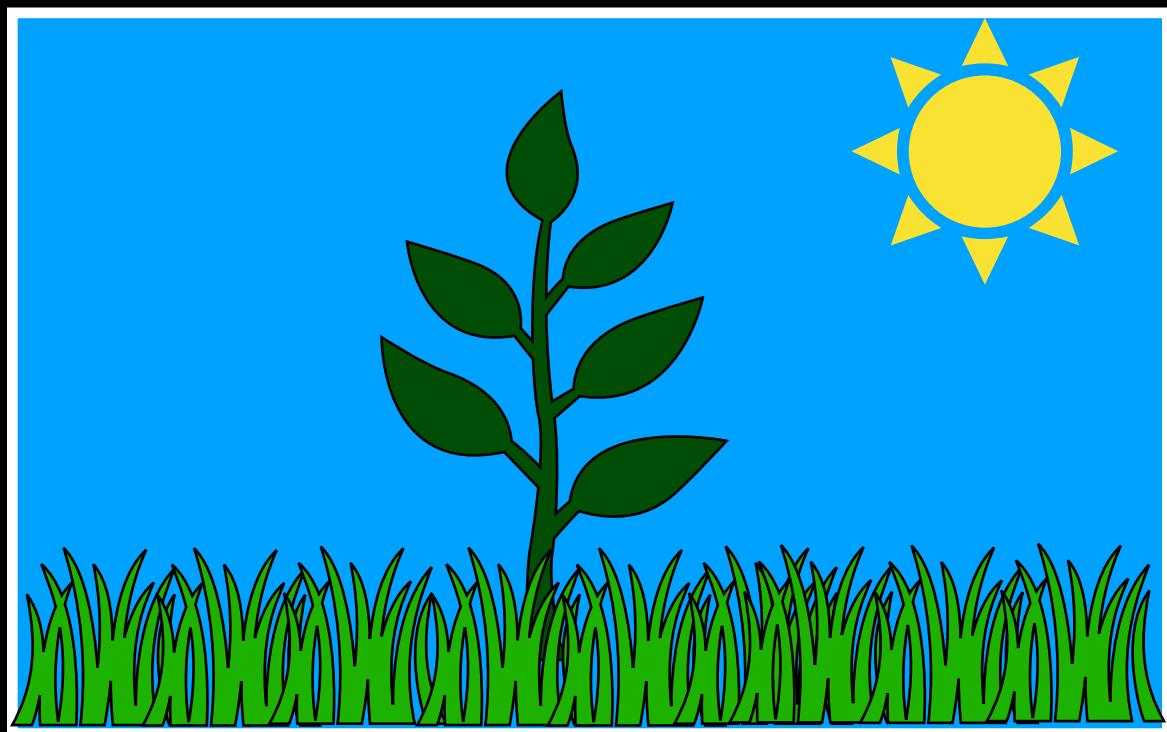
```
from cv2 import *
stream = VideoCapture(0)
while True:
    _, imagem = stream.read()
    rectangle(imagem, pt1=(550,200), pt2=(850,300), color=(0,255,0),
thickness=3)
    circle(imagem, (700,400), 90, color=(255,255,0), thickness=7)
    putText(imagem, "Jan K. S", (10,600), color=(255,255,255),
thickness=4, fontFace=FONT_HERSHEY_SIMPLEX, fontScale=2)
    imshow("Minha Janela", imagem)
    if waitKey(1) & 0xFF == ord("q"):
        break
stream.release()
destroyAllWindows()
```





Segmentação de Cor

máscara



Máscaras de Cor

```
from cv2 import *

stream = VideoCapture(0)

while True:
    _, imagem = stream.read()

    imagem_hsv = cvtColor(imagem, COLOR_BGR2HSV)
    light_yellow = (13, 37, 0) # valores no espaço HSV
    dark_yellow = (35, 189, 255) # valores no espaço HSV
    mascara = inRange(imagem_hsv, light_yellow, dark_yellow)

    imshow("Minha Janela", mascara)

    if waitKey(1) & 0xFF == ord("q"):
        break

stream.release()
destroyAllWindows()
```



```
from cv2 import *

stream = VideoCapture(0)

while True:
    _, imagem = stream.read()

    imagem_hsv = cvtColor(imagem, COLOR_BGR2HSV)
    light_yellow = (13, 37, 0)
    dark_yellow = (35, 189, 255)
    mascara = inRange(imagem_hsv, light_yellow, dark_yellow)

    imagem2 = bitwise_and(imagem, imagem, mask=mascara)

    imshow("Minha Janela", imagem2)

    if waitKey(1) & 0xFF == ord("q"):
        break

stream.release()
destroyAllWindows()
```



```
from cv2 import *

stream = VideoCapture(0)

while True:
    _, imagem = stream.read()

    imagem_hsv = cvtColor(imagem, COLOR_BGR2HSV)
    light_yellow = (13, 37, 0)
    dark_yellow = (35, 189, 255)
    mascara = inRange(imagem_hsv, light_yellow, dark_yellow)
    mascara2 = bitwise_not(mascara)

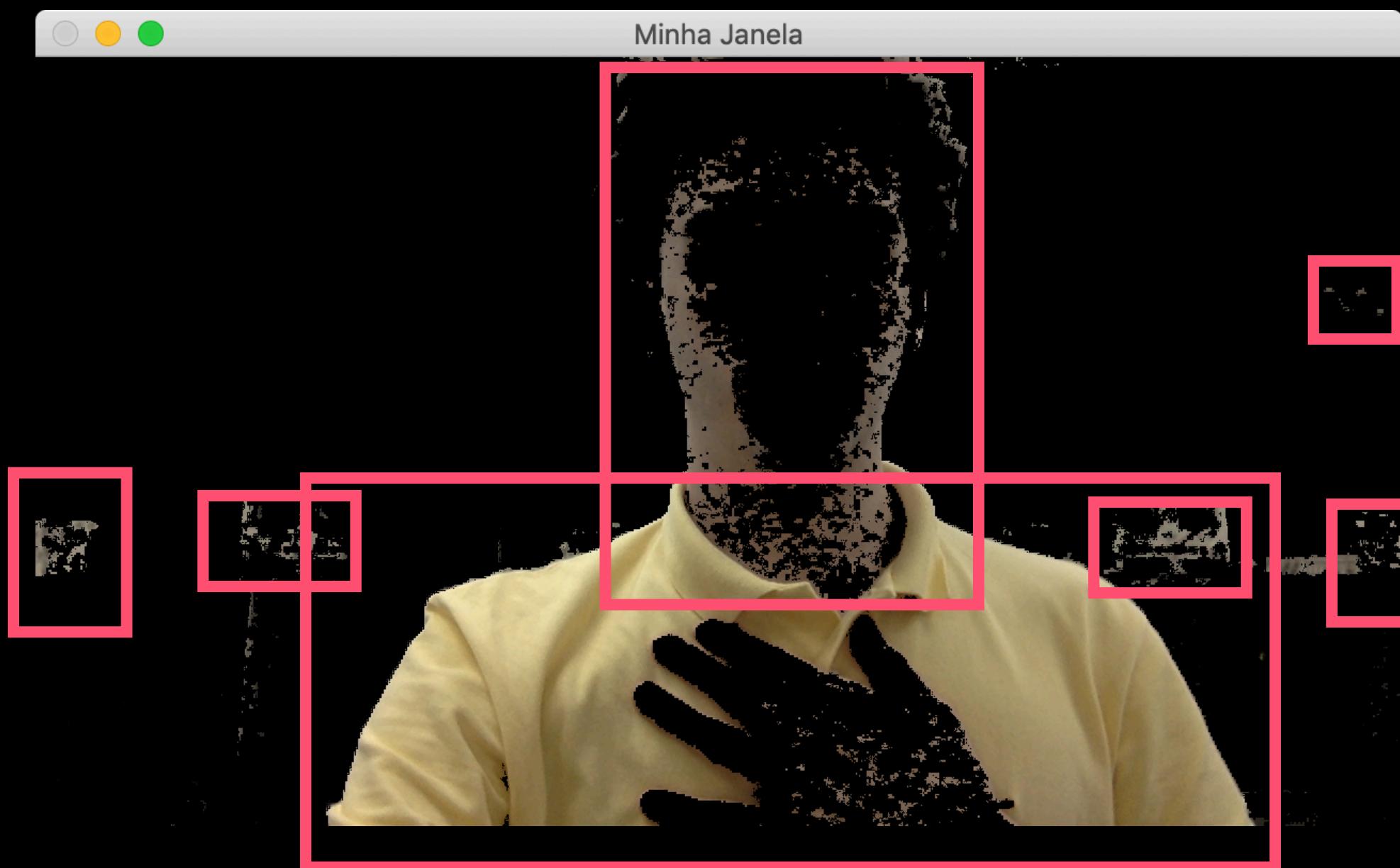
    imagem2 = bitwise_and(imagem, imagem, mask=mascara2)

    imshow("Minha Janela", imagem2)

    if waitKey(1) & 0xFF == ord("q"):
        break

stream.release()
destroyAllWindows()
```





Detecção de Regiões por Cor

```
from cv2 import *

stream = VideoCapture(0)

while True:
    imagem_hsv = cvtColor(imagem, COLOR_BGR2HSV)

    light_yellow = (13, 37, 0)
    dark_yellow = (35, 189, 255)
    mascara = inRange(imagem_hsv, light_yellow, dark_yellow)

    contornos,_ = findContours(mascara, RETR_TREE, CHAIN_APPROX_SIMPLE)
    for contorno in contornos:
        x, y, comprimento, altura = boundingRect(contorno)
        // desconsiderar regiões pequenas, desenhar retângulos, etc...
    ...
```

Resumo da Ópera

Funcionalidade

Comandos

Tela TFT

```
#include <Adafruit_GFX.h>    #include <MCUFRIEND_kbv.h>
                                MCUFRIEND_kbv tela;
tela.begin( tela.readID() ); tela.fillRect(x, y, comprimento, altura, TFT_GREEN);
                                tela.drawLine(x1, y1, x2, y2, TFT_RED);
                                tela.fillCircle(x, y, raio, TFT_ORANGE);
                                tela.drawCircle(x, y, raio, TFT_YELLOW);
                                tela.fillRect(x, y, comprimento, altura, TFT_BLUE);
                                tela.fillTriangle(x1, y1, x2, y2, x3, y3, TFT_PURPLE);
                                tela.drawTriangle(x1, y1, x2, y2, x3, y3, TFT_WHITE);
tela.setCursor(x, y); tela.setTextColor(TFT_DARKGREY);
tela.setTextSize(4); tela.print("Jan K. S.");
```

Touch

```
TouchScreen touch(6, A1, A2, 7, 300);
const int TS_LEFT=145, TS_RT=887, TS_TOP=934, TS_BOT=158;
TSPoint ponto = touch.getPoint();
pinMode(A1, OUTPUT); digitalWrite(A1, HIGH);
pinMode(A2, OUTPUT); digitalWrite(A2, HIGH);
int x = map(ponto.x, TS_LEFT, TS_RT, 0, 240);
int y = map(ponto.y, TS_TOP, TS_BOT, 0, 320);
```

JKSButton

```
JKSButton botao;
botao.init(&tela, &touch, xDoCentro, yDoCentro, comprimento, altura,
corDaBorda, corDoFundo, corDoTexto, texto, tamanhoDoTexto);
botao.setPressHandler(funcao1); botao.setReleaseHandler(funcao2);
botao.process()
```

Funcionalidade

Captura de Vídeo
[documentação](#)

Desenho de
Elementos
[documentação](#)

Espaços de Cor
e Segmentação
[documentação](#)

Detecção
[documentação](#)

Comandos

```
from cv2 import *
stream = VideoCapture(0)
while True:
    _, imagem = stream.read()
    imshow("Minha Janela", imagem)
    if waitKey(1) & 0xFF == ord("q"):
        break
    stream.release()
    destroyAllWindows()

circle(imagem, (x,y), raio, color=(b,g,r), thickness=espessura)
rectangle(imagem, pt1=(x1,y1), pt2=(x2,y2), color=(b,g,r),
          thickness=espessura);
putText(imagem, "texto", (x,y), color=(b,g,r), thickness=espessura,
        fontFace=FONT_HERSHEY_SIMPLEX, fontScale=tamanho)

{
    imagem_cinza = cvtColor(imagem, COLOR_BGR2GRAY)
    imagem_cinza = cvtColor(imagem_cinza, COLOR_GRAY2BGR)
    imagem_hsv = cvtColor(imagem, COLOR_BGR2HSV)
    mascara = inRange(imagem_hsv, cor_clara, cor_escura)
    mascara2 = bitwise_not(mascara)
    imagem2 = bitwise_and(imagem, imagem, mask=mascara)

contornos,_ = findContours(mascara, RETR_TREE, CHAIN_APPROX_SIMPLE)
for contorno in contornos:
    x, y, comprimento, altura = boundingRect(contorno)
```

Funcionalidade

Comandos

Sensor Ótico

```
int sensor = A11; pinMode(sensor, INPUT);
digitalRead(sensorOtico) == LOW // superfície é clara?
int valorAnalogico = analogRead(sensorOtico);
```

Shield Motor documentação

```
#include <AFMotor.h>
AF_DCMotor motorA(1); AF_DCMotor motorB(2);
motorA.setSpeed(255); motorB.setSpeed(128);
motorA.run(FORWARD); motorB.run(BACKWARD); motorA.run(RELEASE);
```

Strings documentação

```
String texto = "Olá!", trecho = texto.substring(0, 3);
String texto1 = String(numero), texto2 = "aaa" + texto1;
int numero2 = texto2.toInt() + 2;
texto2 == texto3; bool comecaCom = texto.startsWith("Olá");
texto.trim(); texto.replace("a", "A");
```

Serial no Arduino documentação

```
Serial.begin(9600); // ou Serial1, Serial2...
if (Serial.available() > 0) {
    String texto = Serial.readStringUntil('\n');
    texto.trim();
    Serial.println(texto);
}
```

PySerial documentação

```
from serial import Serial
meu_serial = Serial("/dev/serial0", baudrate=9600)
    texto = "Olá!" + "\n"
    meu_serial.write( texto.encode("UTF-8") )
texto_recebido = meu_serial.readline().decode().strip()
```

Funcionalidade

Leitura Analógica

[documentação](#)

Servomotor

[documentação](#)

Braço Mecânico

[documentação](#)

EEPROM

[documentação](#)

Comandos

```
int potenciometro = A10;  
pinMode(potenciometro, INPUT);  
  
int valorAnalogico = analogRead(potenciometro);  
int valorMapeado = map(valorAnalogico, 0, 1023, min, max);  
  
#include <Servo.h>  
  
Servo servo;  
servo.attach(pino); servo.detach();  
servo.write(anguloEmGraus);  
  
#include <meArm.h>  
  
int base = 12, ombro = 11, cotovelo = 10, garra = 9;  
meArm braco(  
    180, 0, -pi/2, pi/2, // ângulos da base  
    135, 45, pi/4, 3*pi/4, // ângulos do ombro  
    180, 90, 0, -pi/2, // ângulos do cotovelo  
    30, 0, pi/2, 0 // ângulos da garra  
);  
braco.begin(base, ombro, cotovelo, garra);  
braco.goDirectlyTo(x, y, z); braco.gotoPoint(x, y, z);  
braco.openGripper(); braco.closeGripper();  
braco.getX(); braco.getY(); braco.getZ(); braco.end();
```

```
#include <EEPROM.h>
```

```
EEPROM.get(endereco, minhaVariavel);  
EEPROM.put(endereco, minhaVariavel);
```

Funcionalidade

Campainha Passiva
[documentação](#)

Interrupção
[documentação](#)

Contagem
de Tempo
[documentação](#)

Encoder Rotativo
[documentação](#)

Comandos

```
int campainhaPassiva = 5;  
pinMode(campainhaPassiva, OUTPUT);  
int frequencia = 220; int duracaoEmMs = 500;  
tone(campainhaPassiva, frequencia);  
tone(campainhaPassiva, frequencia, duracaoEmMs);  
noTone(campainhaPassiva);
```

```
int sensorDeSom = 19;  
pinMode(sensorDeSom, INPUT);  
int origem = digitalPinToInterrupt(sensorDeSom);  
attachInterrupt(origem, minhaFuncao, RISING);
```

```
unsigned long instanteAnteriorDeDeteccao = 0;  
  
if (millis() > instanteAnteriorDeDeteccao + 10) {  
    instanteAnteriorDeDeteccao = millis();  
}
```

```
#include <RotaryEncoder.h>  
RotaryEncoder encoder(20, 21);  
int origem1 = digitalPinToInterrupt(20);  
attachInterrupt(origem1, tickDoEncoder, CHANGE);  
int origem2 = digitalPinToInterrupt(21);  
attachInterrupt(origem2, tickDoEncoder, CHANGE);  
encoder.tick(); int posicao = encoder.getPosition();
```

Funcionalidade

Revisão de C++

Comandos

```
int inteiro = 2; float decimal = 4.5; bool booleano = true;  
char texto[] = "Olá"; int listaDeInteiros[] = {1, 2, 3, 4};  
  
if (x > 0 && y > 0) {  
    z = 1;  
}  
else if (x < 0 || y < 0) {  
    z = 2;  
}
```

```
for (int i = 0; i < 5; i++) {  
    Serial.println(i);  
}  
float soma (float x) {  
    return x + 2;  
}
```

Print Serial

```
Serial.begin(9600); Serial.println("Olá"); Serial.println(2);
```

Escrita/Leitura documentação

```
int led = 13; pinMode(led, OUTPUT); digitalWrite(led, LOW);  
int campainha = 3; digitalWrite(campainha, HIGH);  
int botao = A1; pinMode(botao, INPUT); digitalWrite(botao) == LOW
```

GButton documentação

```
#include <GButton.h>  
GButton botao(A1); botao.isPressed(); botao.process();  
botao.setPressHandler(funcao); botao.setReleaseHandler(funcao);
```

ShiftDisplay documentação

```
#include <ShiftDisplay.h>  
ShiftDisplay display(4, 7, 8, COMMON_ANODE, 4, true);  
ShiftDisplay display(4, 7, 8, COMMON_CATHODE, 4, true);  
display.set(1234); display.set(4.21, 2); display.set("Erro");  
display.update(); display.show(1000); display.changeDot(0, true)
```

Timer1 documentação

```
#include <TimerOne.h>  
Timer1.initialize(1000000); Timer1.attachInterrupt(funcao);
```