



Projeto 10

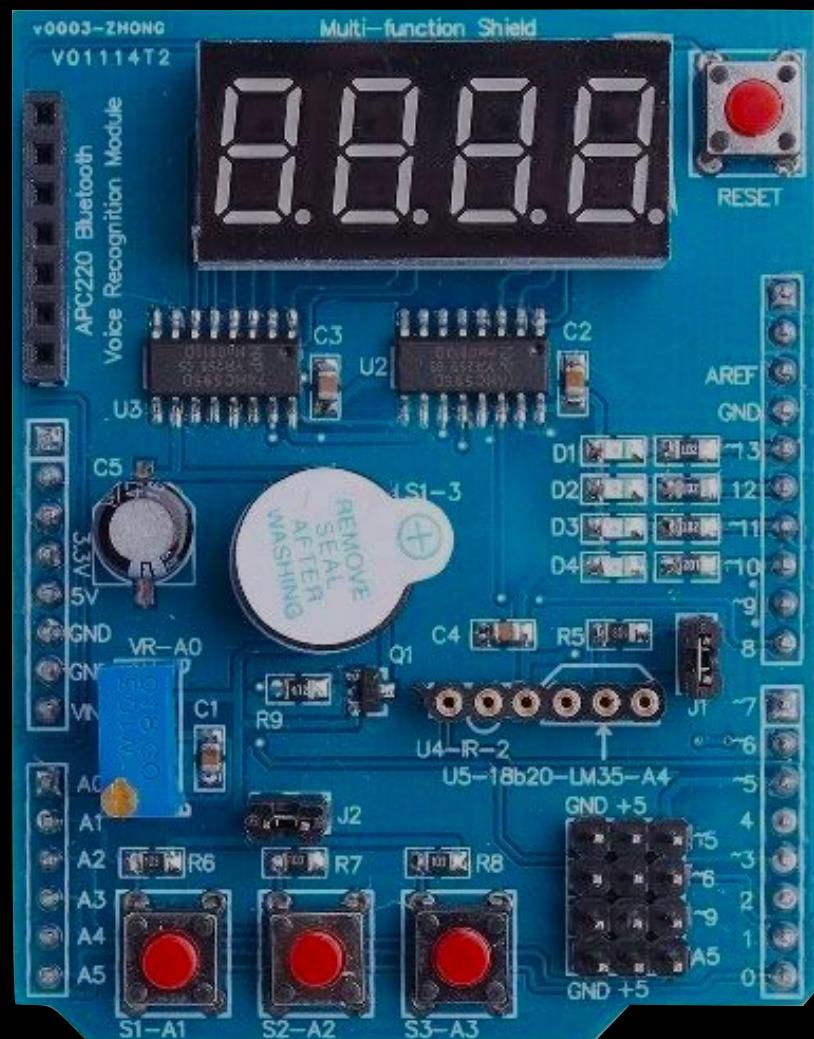
Controle Gráfico – Teoria

Jan K. S. – janks@puc-rio.br

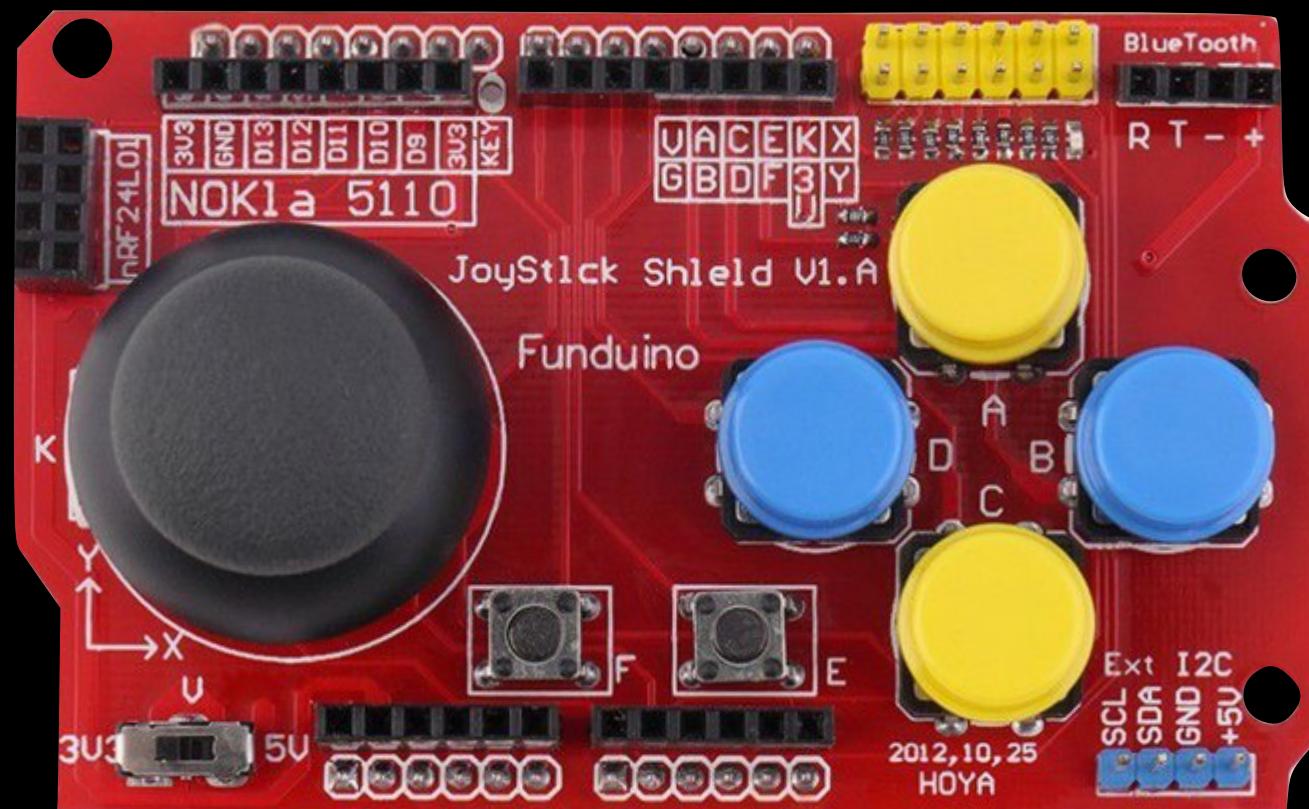
ENG1419 – Programação de Microcontroladores

Hardware

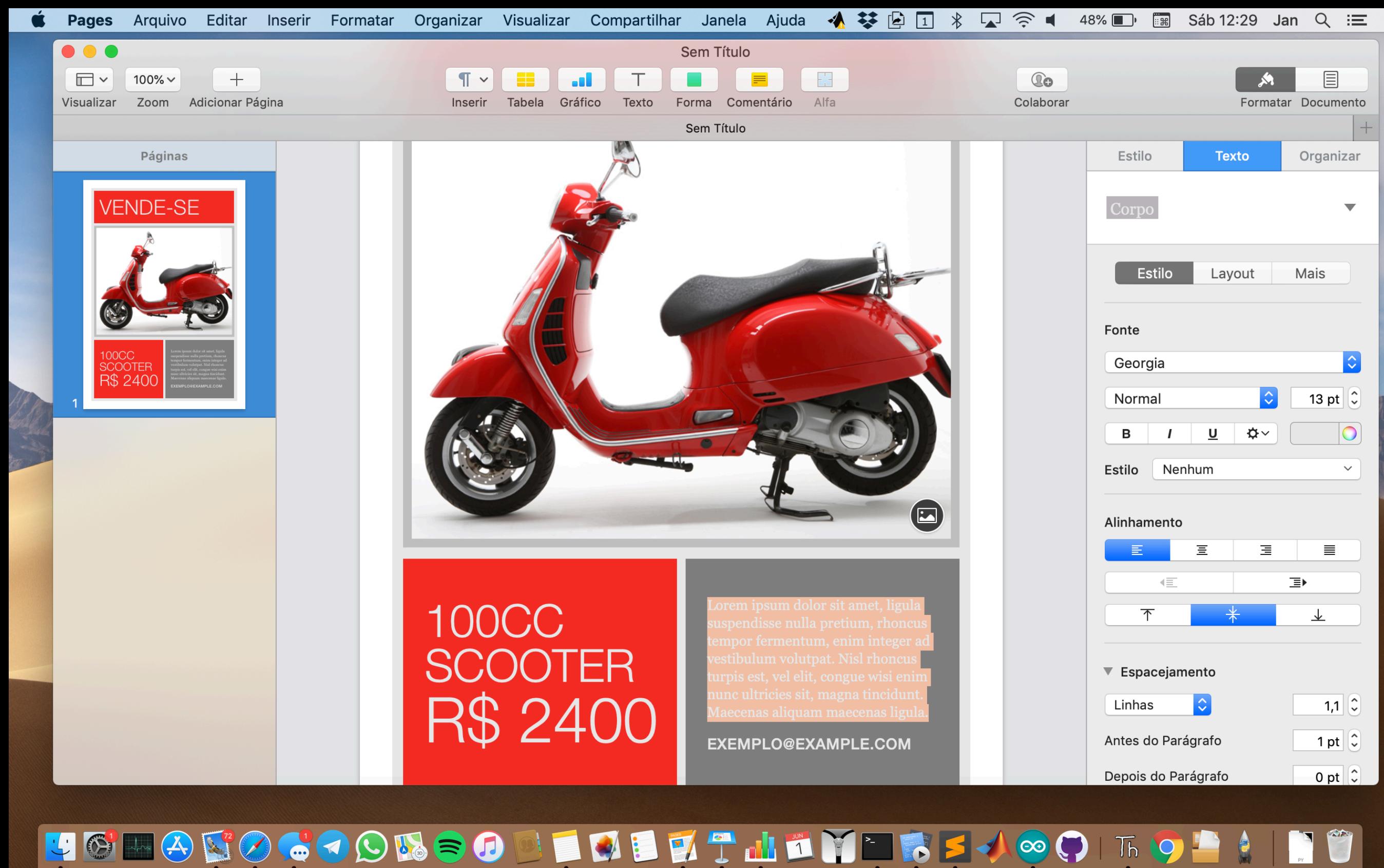
controles fixos



display e buzzer
poucos botões

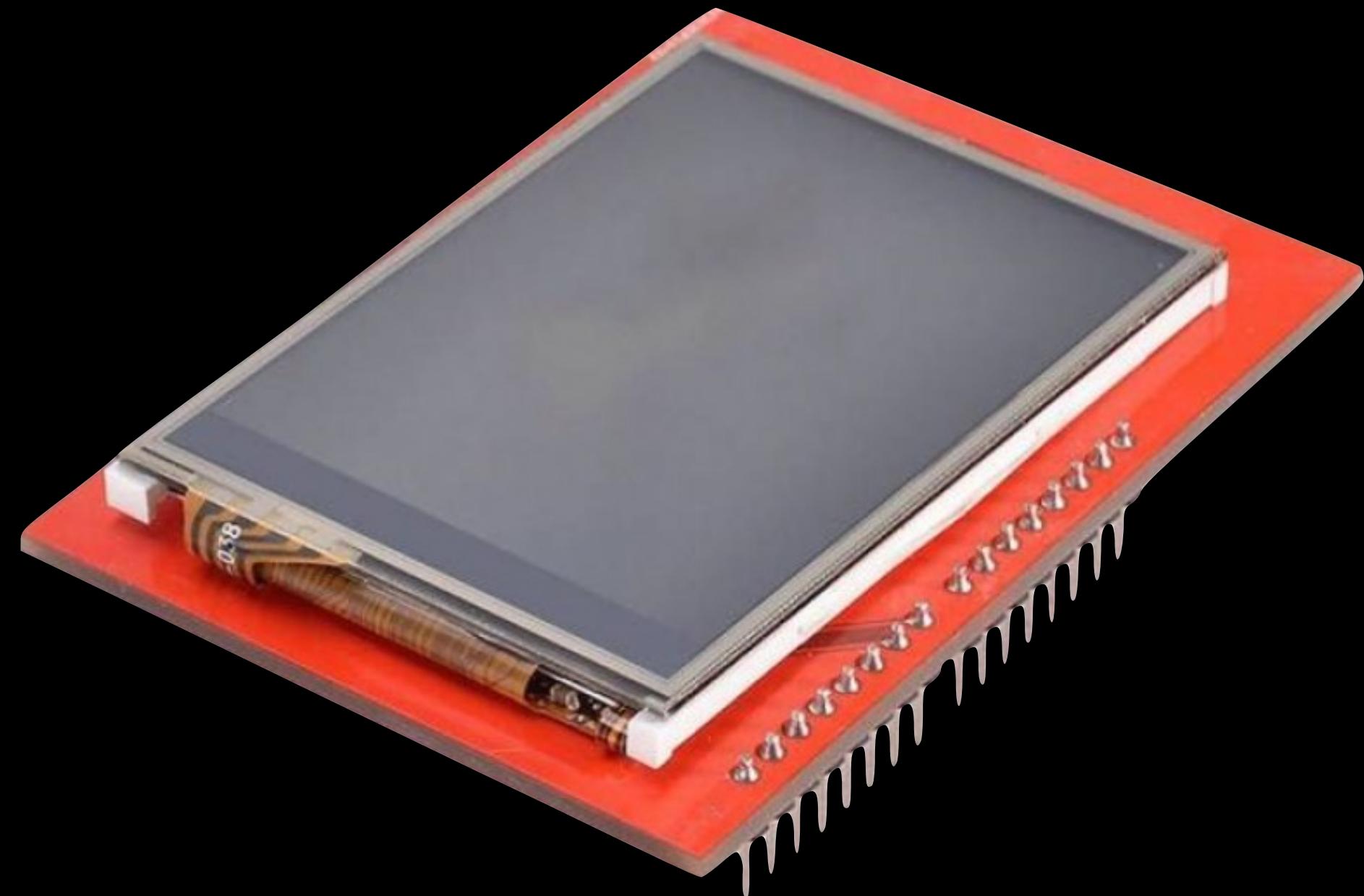


analógico e vários botões
nenhum display



Interfaces Gráficas

Liquid-Crystal Display
Thin-Film-Transistor
2.4 polegadas



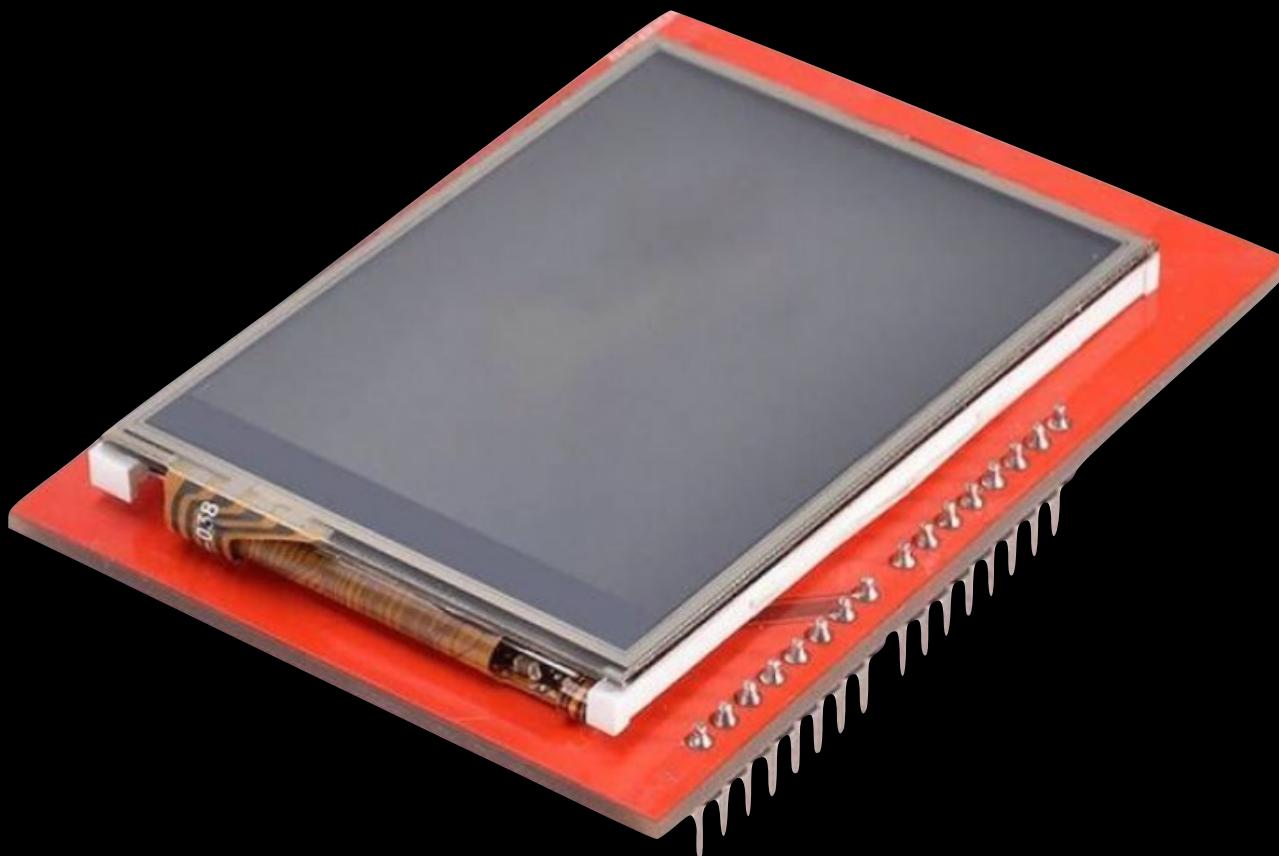
Shield LCD TFT 2.4"



Toque impreciso
com o dedo.



Tela Touch Resistiva



Não é mágica como a dos smartphones.

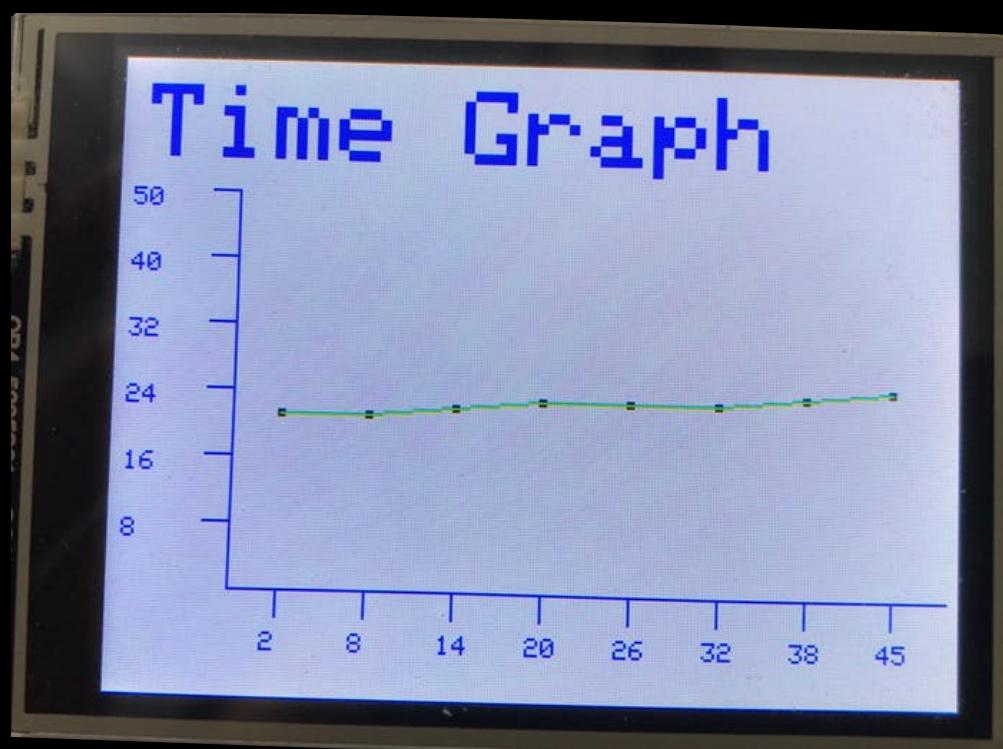
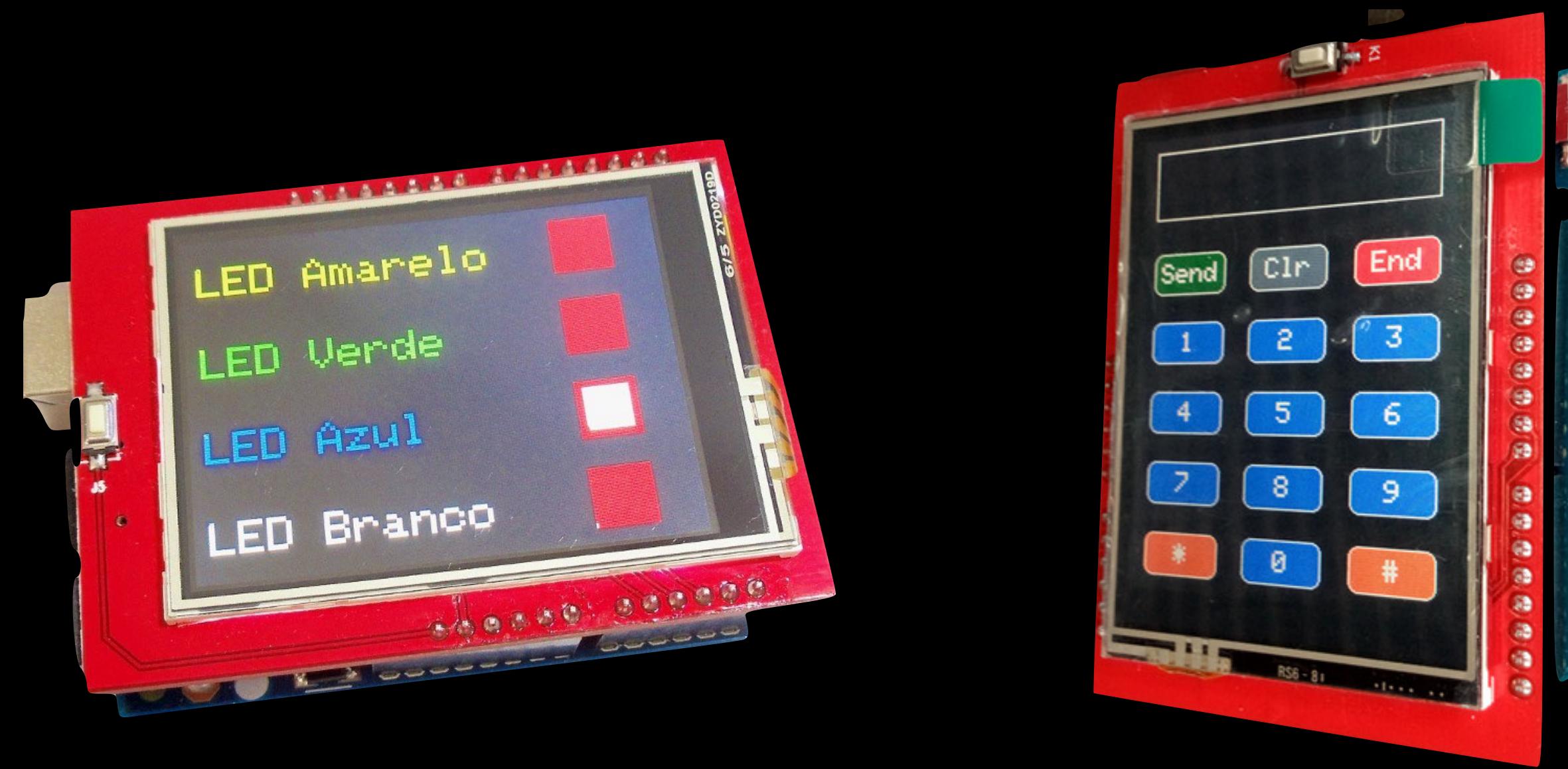
Não precisa de caneta, mas é recomendada.

Resolução de 320 x 240 pixels.

Não tem multitouch.

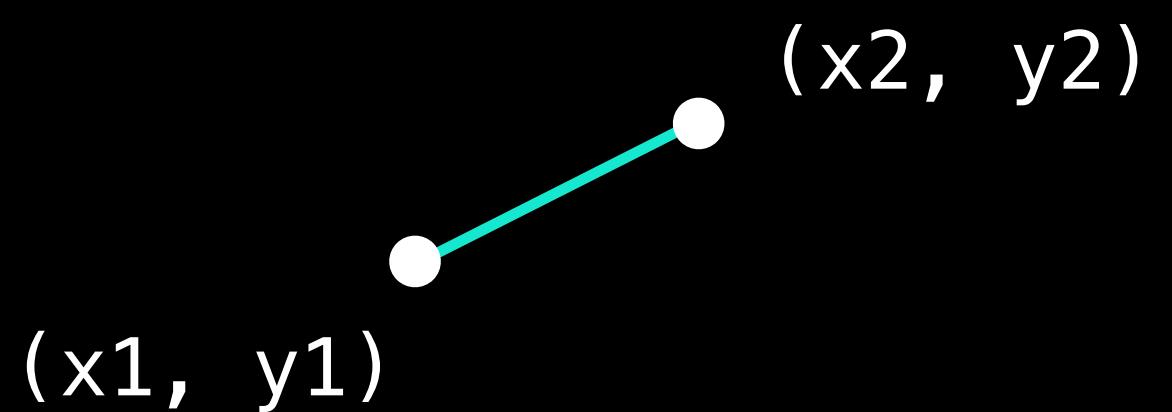
Sem patentes!

Shield LCD TFT



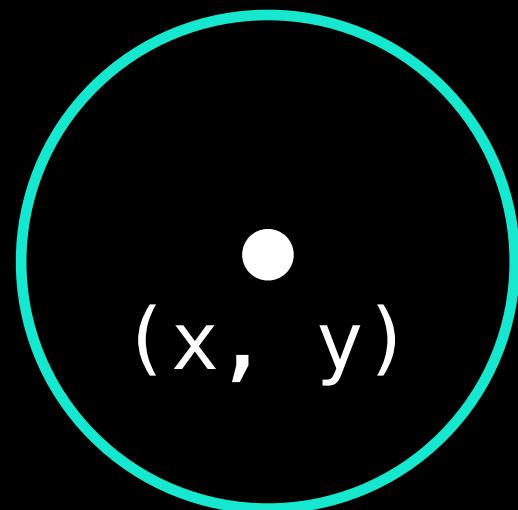
Exemplos de Interface no Shield LCD TFT

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



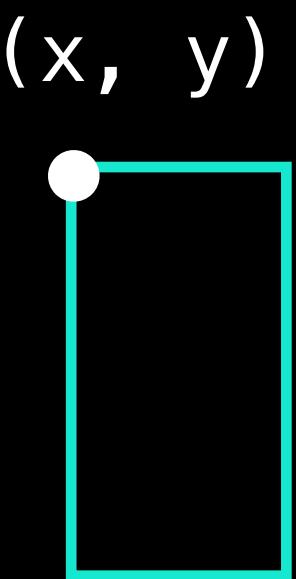
```
tela.fillCircle(x, y, raio, cor);
```

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



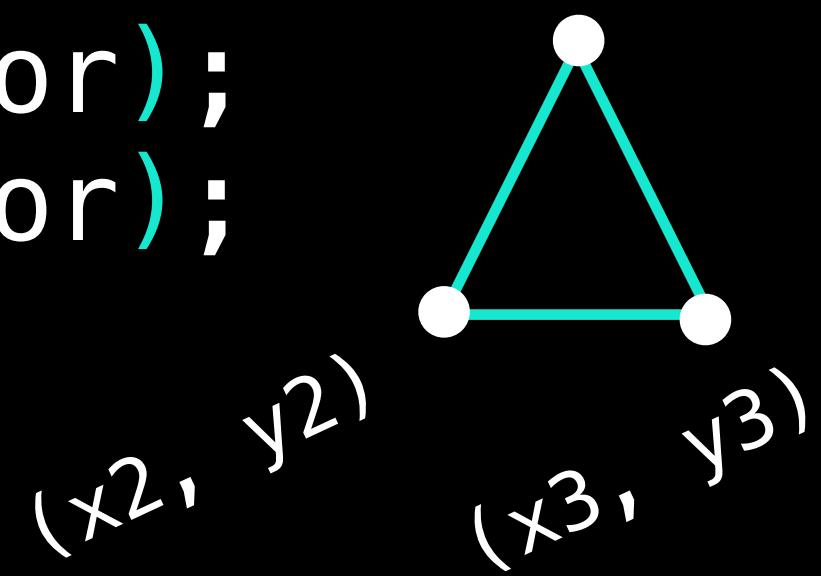
```
tela.fillRect(x, y, comprimento, altura, cor);
```

```
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);

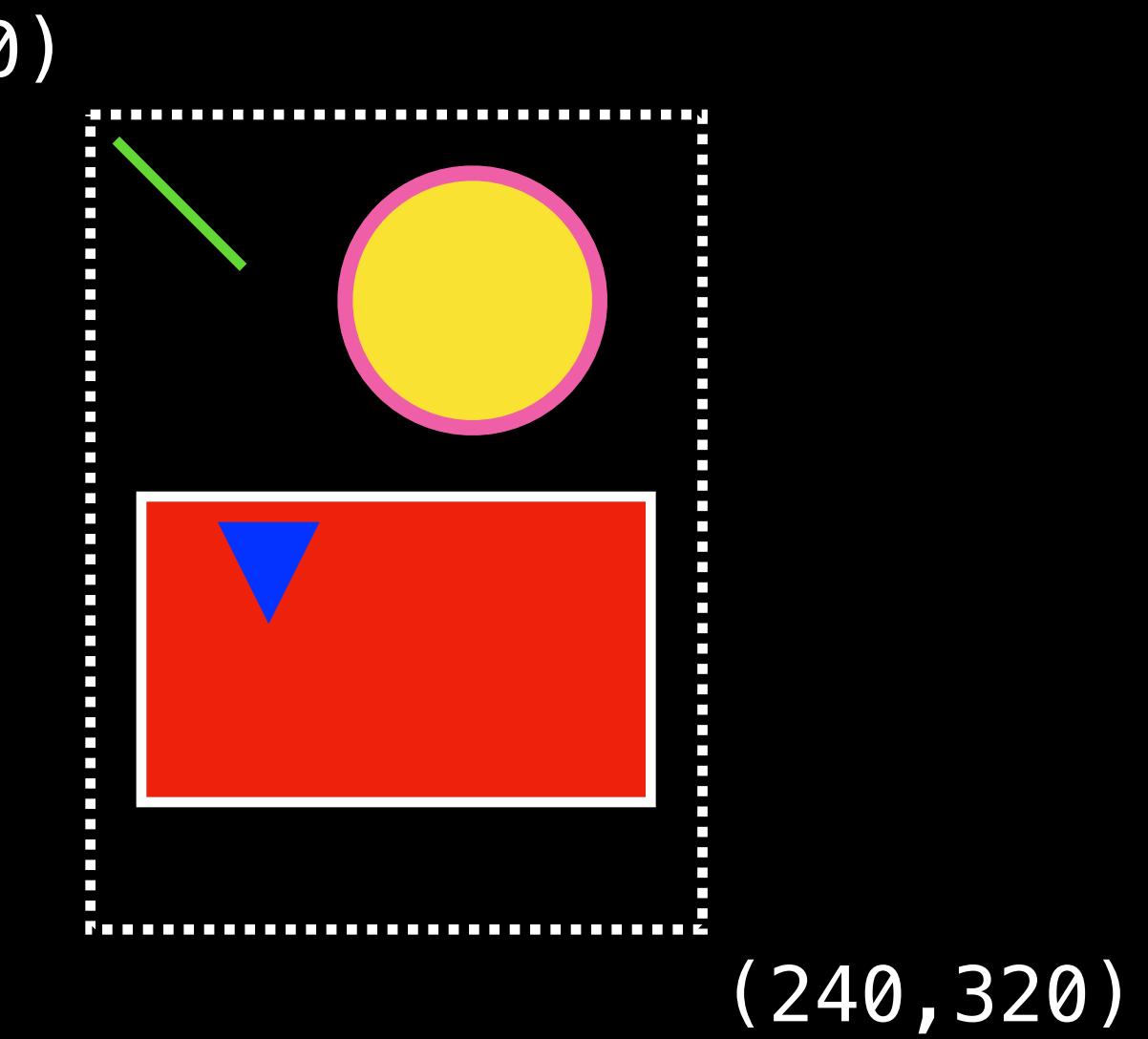
    tela.fillCircle(150, 70, 50, TFT_ORANGE);
    tela.drawCircle(150, 70, 50, TFT_PINK);

    tela.fillRect(20, 150, 200, 120, TFT_RED);
    tela.drawRect(20, 150, 200, 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");

}

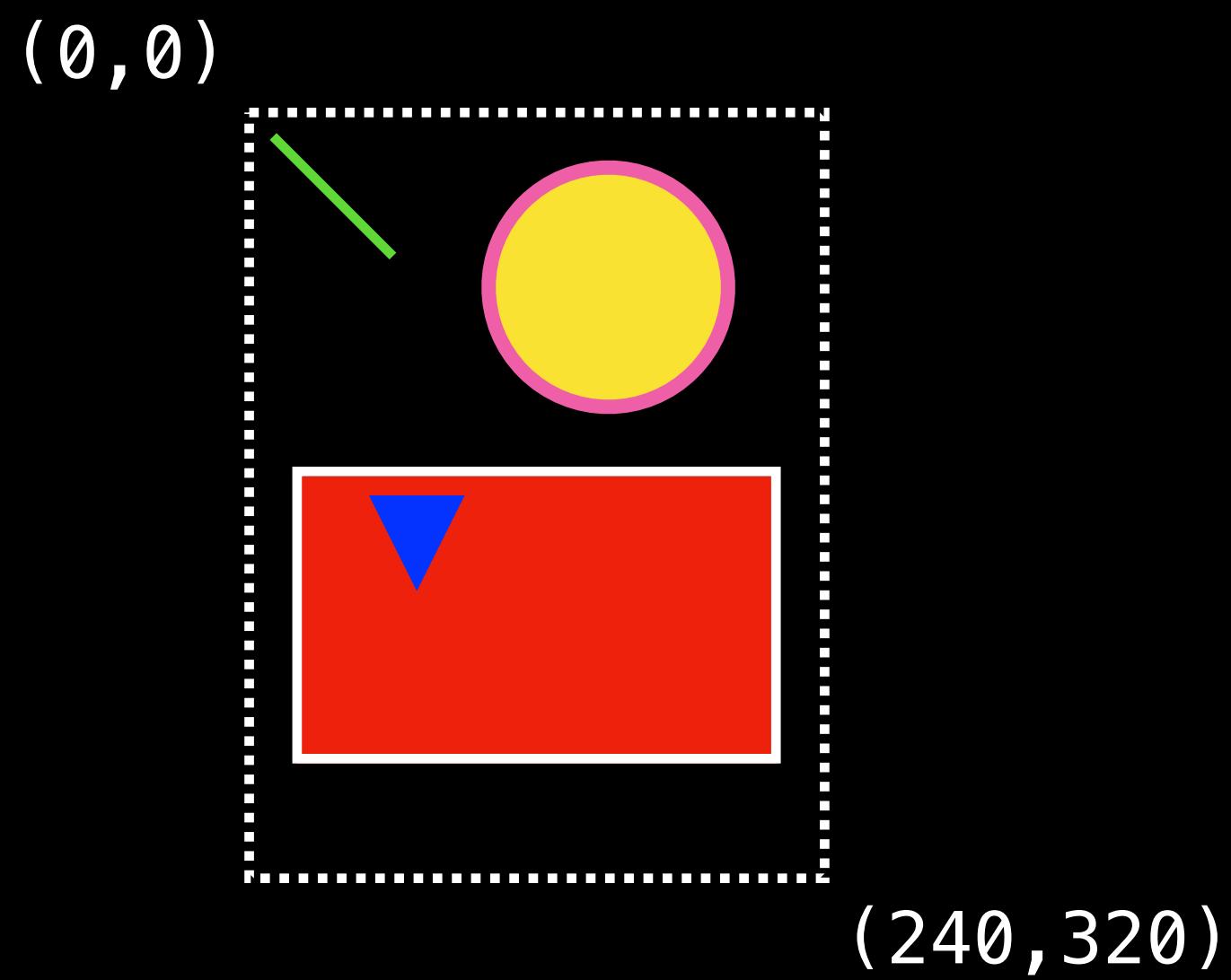
void loop () {
```

Jan K. S.

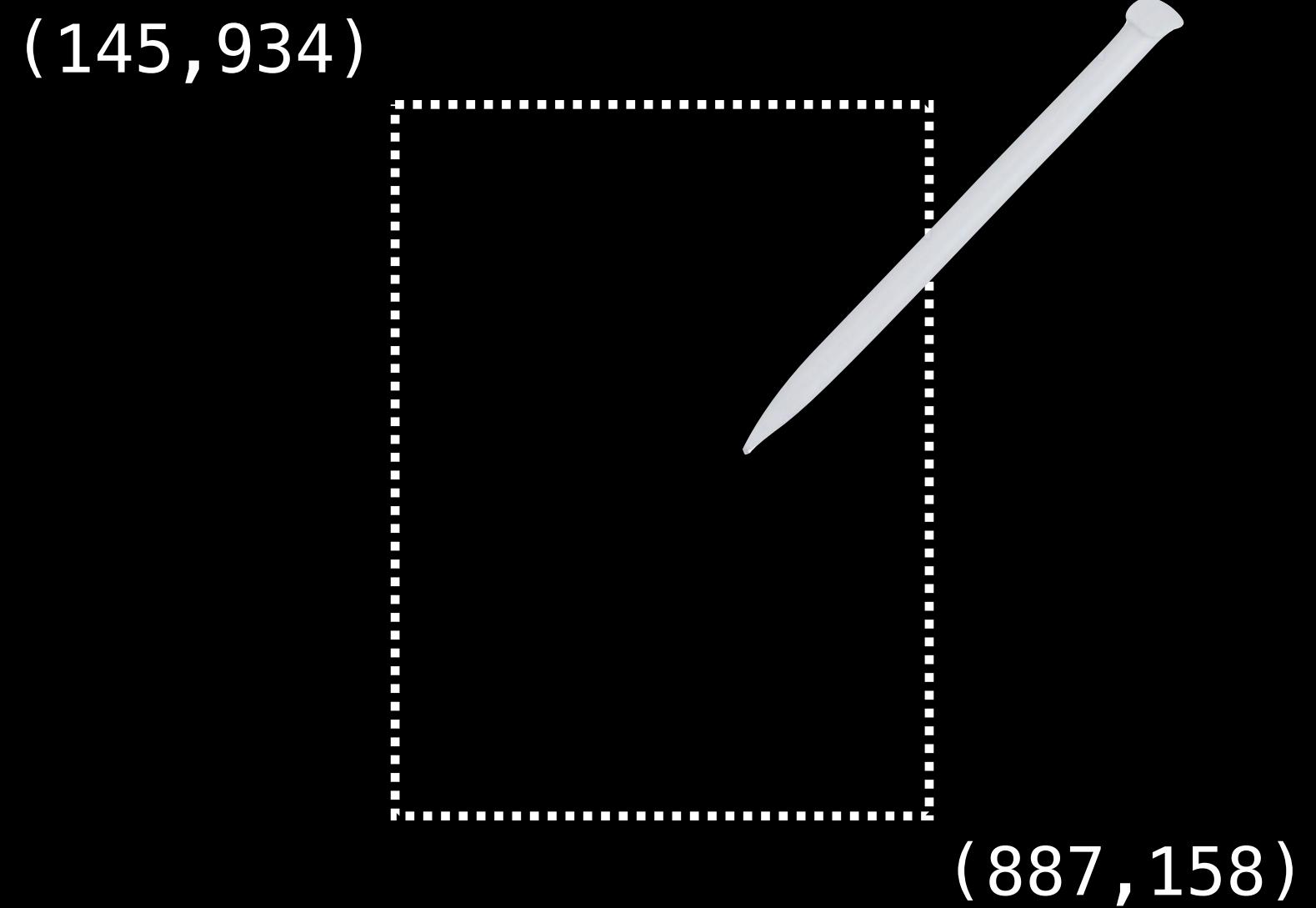
Microcontrol
adores

(240,320)

Exemplo com Funções de Desenho para Texto



coordenadas
gráficas



coordenadas
do touch

Coordenadas Gráficas vs Coordenadas do Sensor de Toque

```

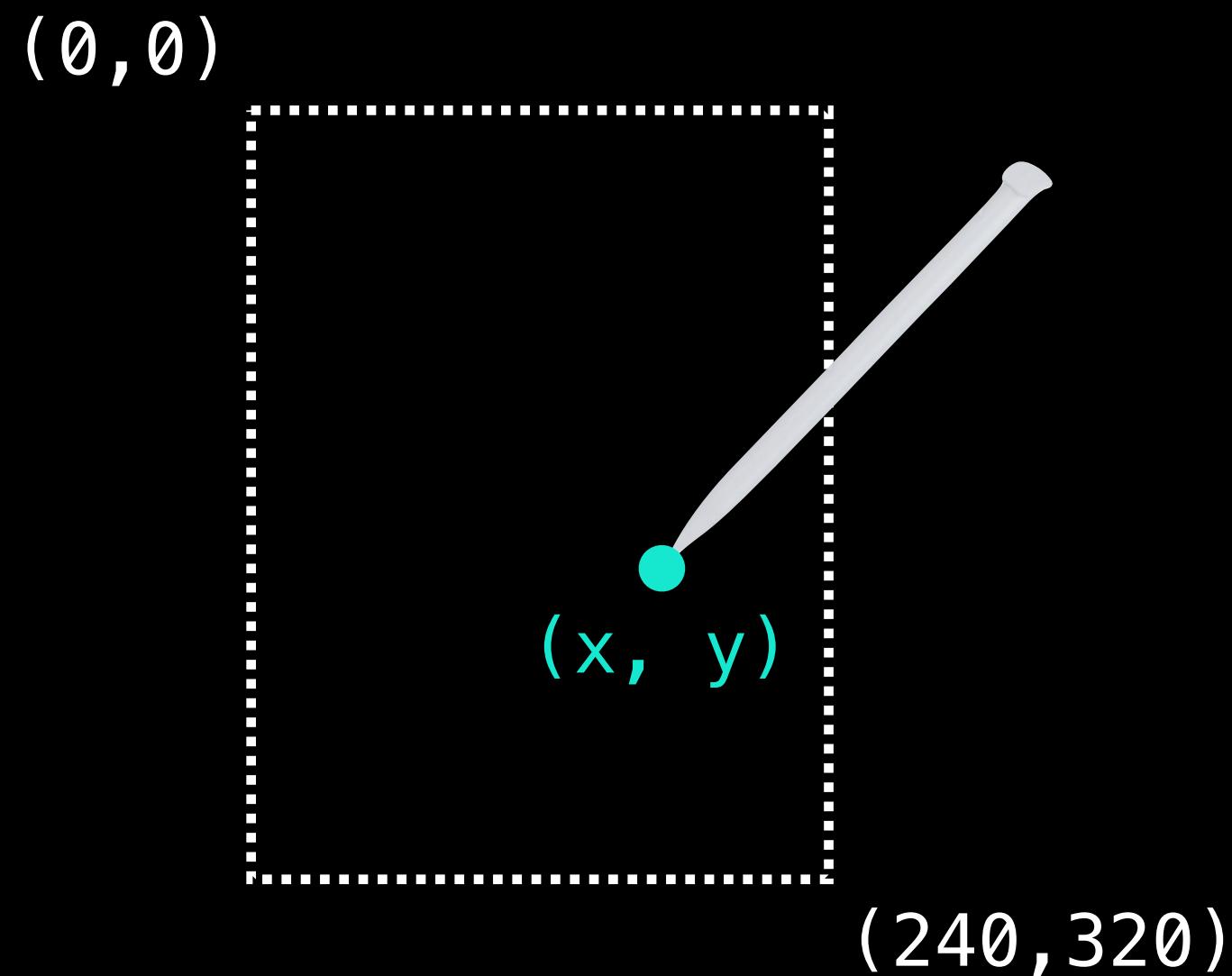
#include <TouchScreen.h>

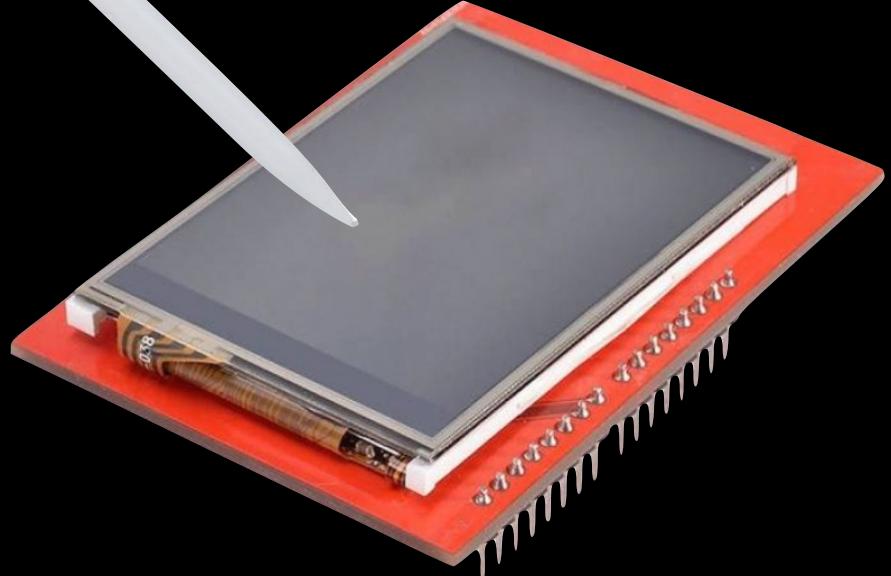
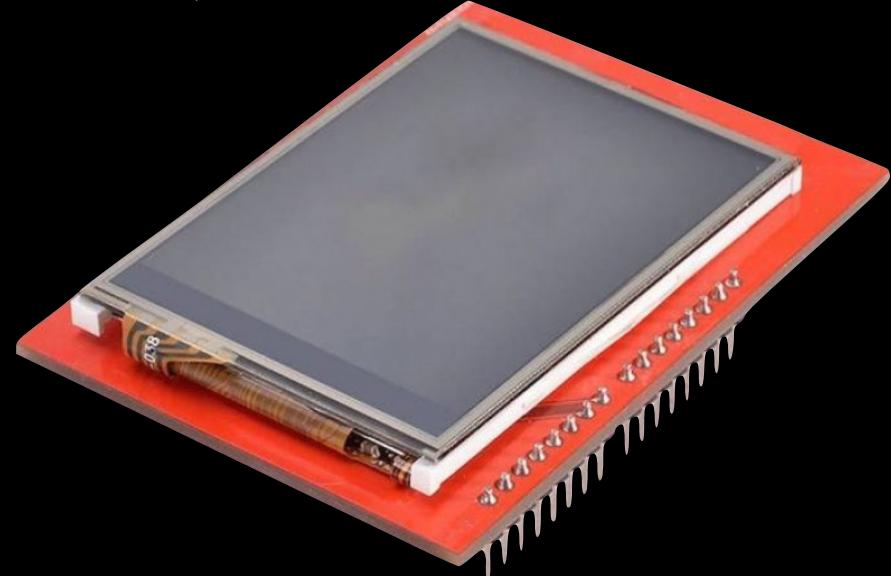
TouchScreen touch(6, A1, A2, 7, 300);
const int TS_LEFT = 145, TS_RT = 887,
          TS_TOP = 934, TS_BOT = 158;

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

void loop () {
    TSPoint ponto = touch.getPoint();
    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

```

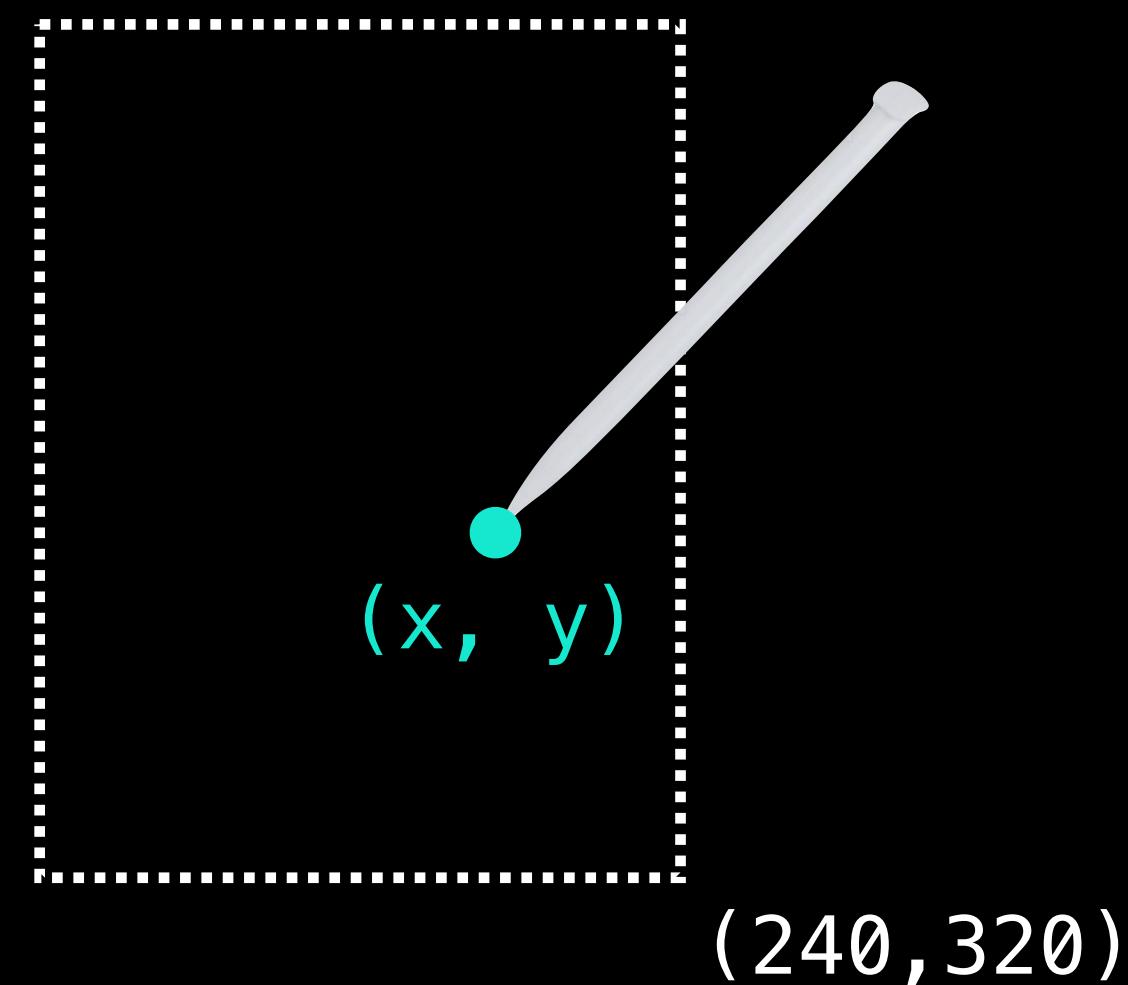
#include <TouchScreen.h>

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

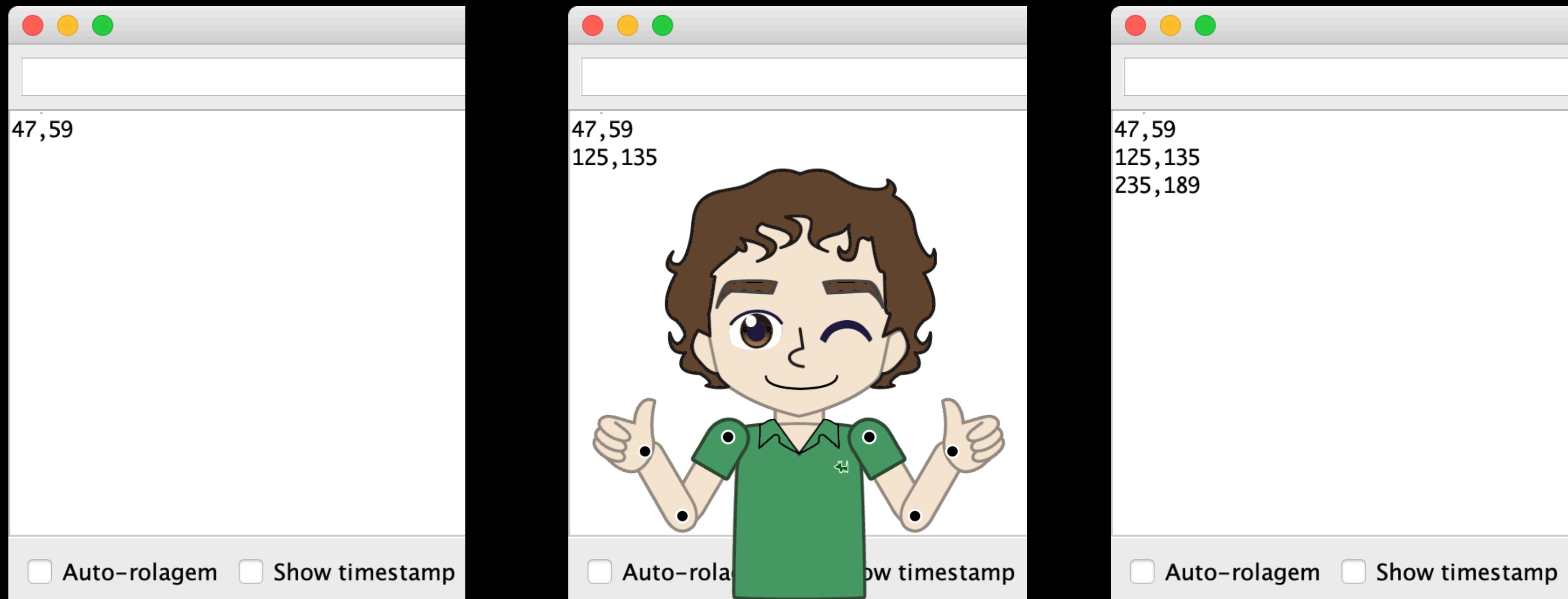
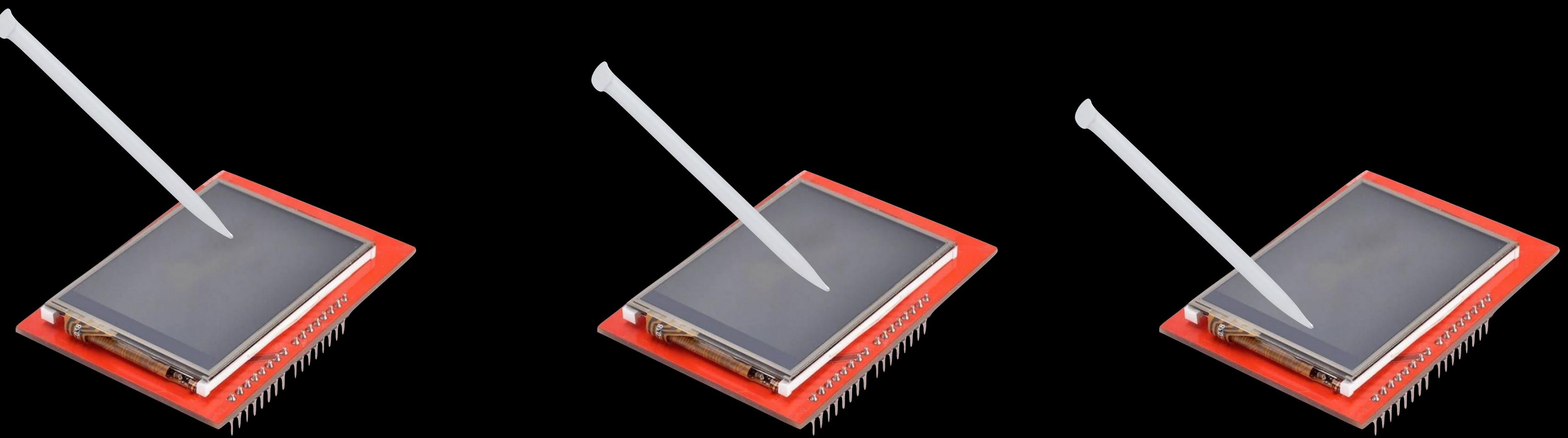
void setup () {
    Serial.begin(9600);
}

void loop () {
    TSPoint ponto = touch.getPoint();
    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"

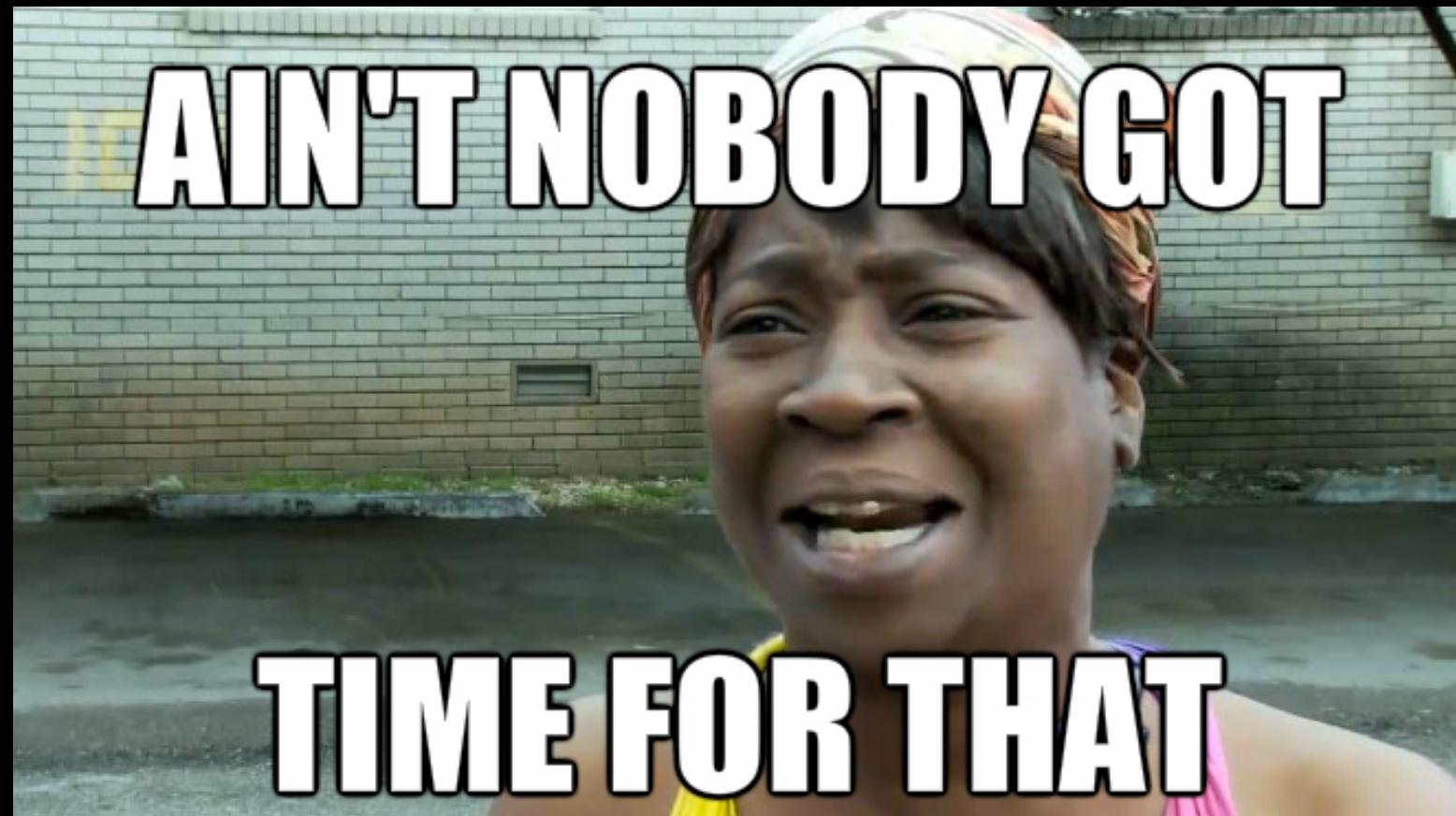


Detecção do Ponto de Toque sem Repetição e "Bounce"

Botão 1

Botão 1

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



```

#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);
}

```

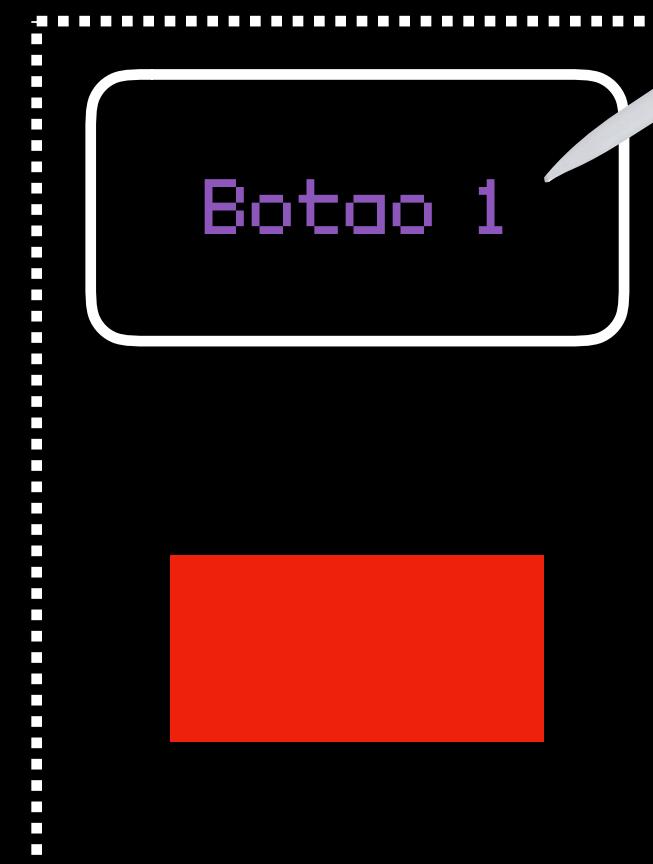
(0,0)



Botao 1

(240,320)

(0,0)



Botao 1

(240,320)

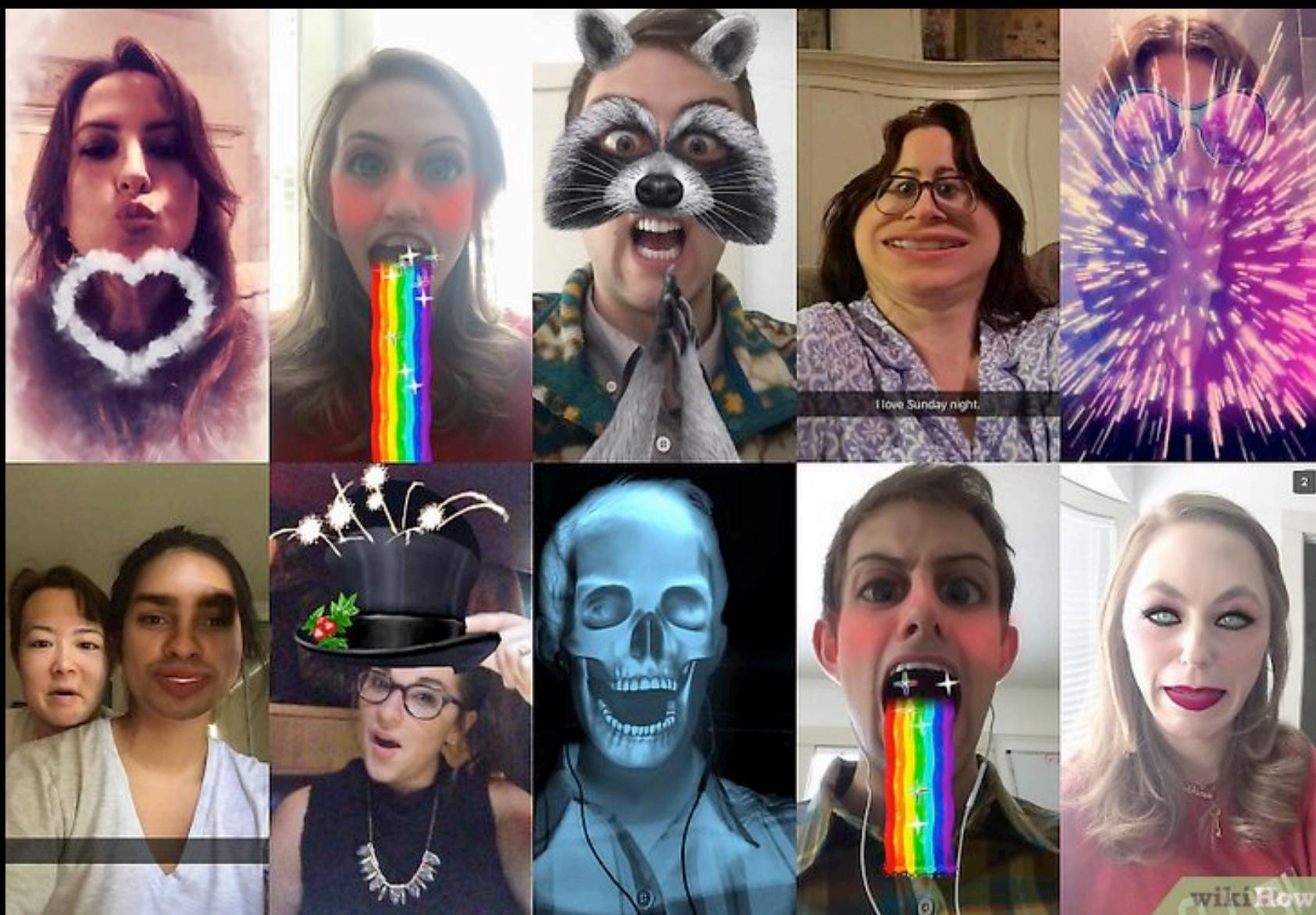
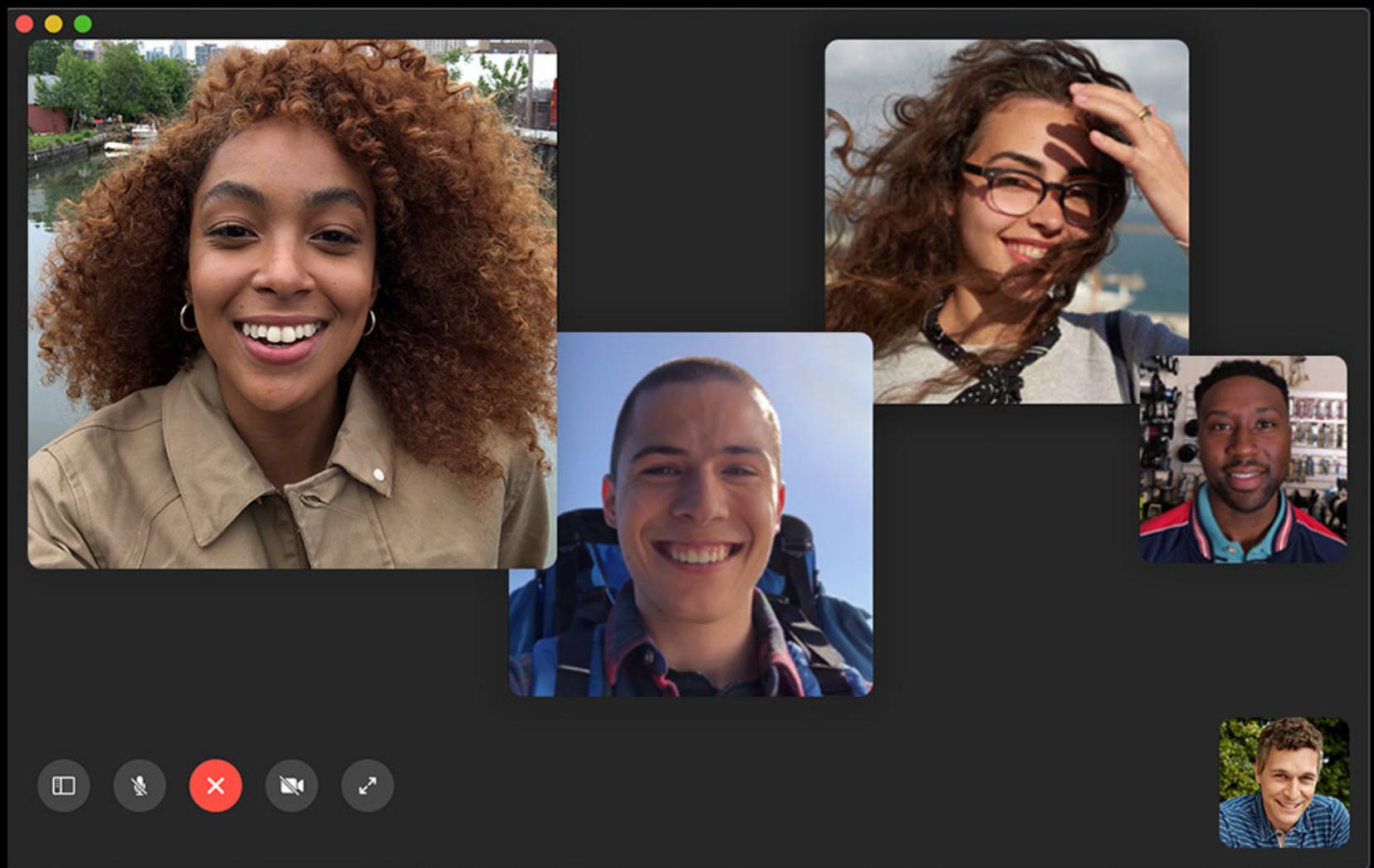
Exemplo de Botão com a JKSButton

Software



Como faz para
capturar 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.

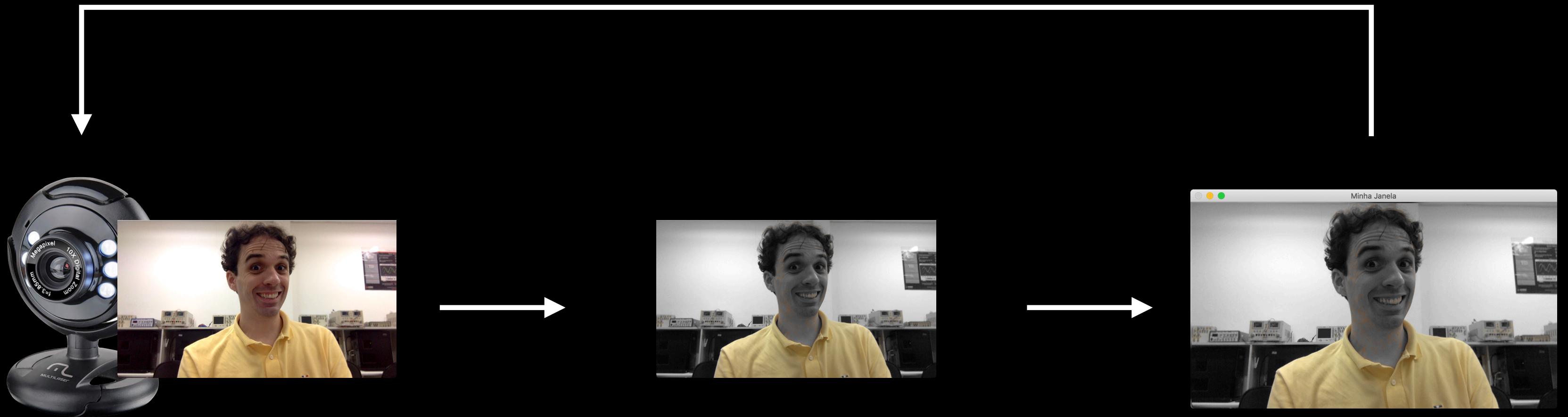
OpenCV (*Open Source Computer Vision Library*). Originalmente, desenvolvida pela **Intel**, em **2000**, é uma **biblioteca** multiplataforma, totalmente livre ao uso **acadêmico** e **comercial**, para o desenvolvimento de aplicativos na área de **Visão computacional**, bastando seguir o modelo de **licença 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)
    # interrompe quando tecla "q" for pressionada
    if waitKey(1) & 0xFF == ord("q"):
        break
stream.release()
destroyAllWindows()
```



```
from cv2 import *

stream = VideoCapture(0)

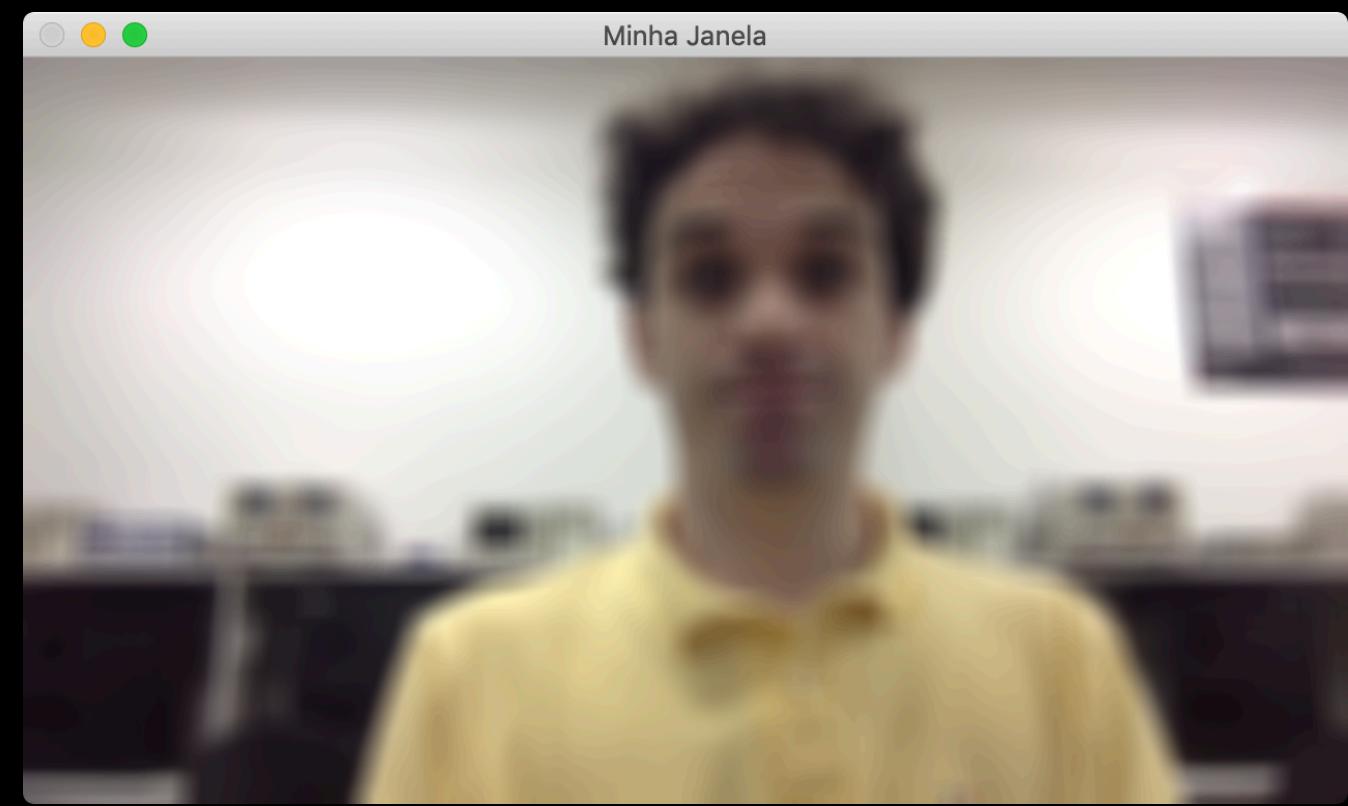
while True:
    _, imagem = stream.read()

    imagem2 = blur(imagem, (50, 50))

    imshow("Minha Janela", imagem2)

    # interrompe quando tecla "q" for pressionada
    if waitKey(1) & 0xFF == ord("q"):
        break

stream.release()
destroyAllWindows()
```



intensidade do desfoque
horizontal e vertical

```
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)

    # interrompe quando tecla "q" for pressionada
    if waitKey(1) & 0xFF == ord("q"):
        break

stream.release()
destroyAllWindows()
```



muda escala de cor para preto e branco,
e depois volta para escala colorida

```
from cv2 import *

stream = VideoCapture(0)

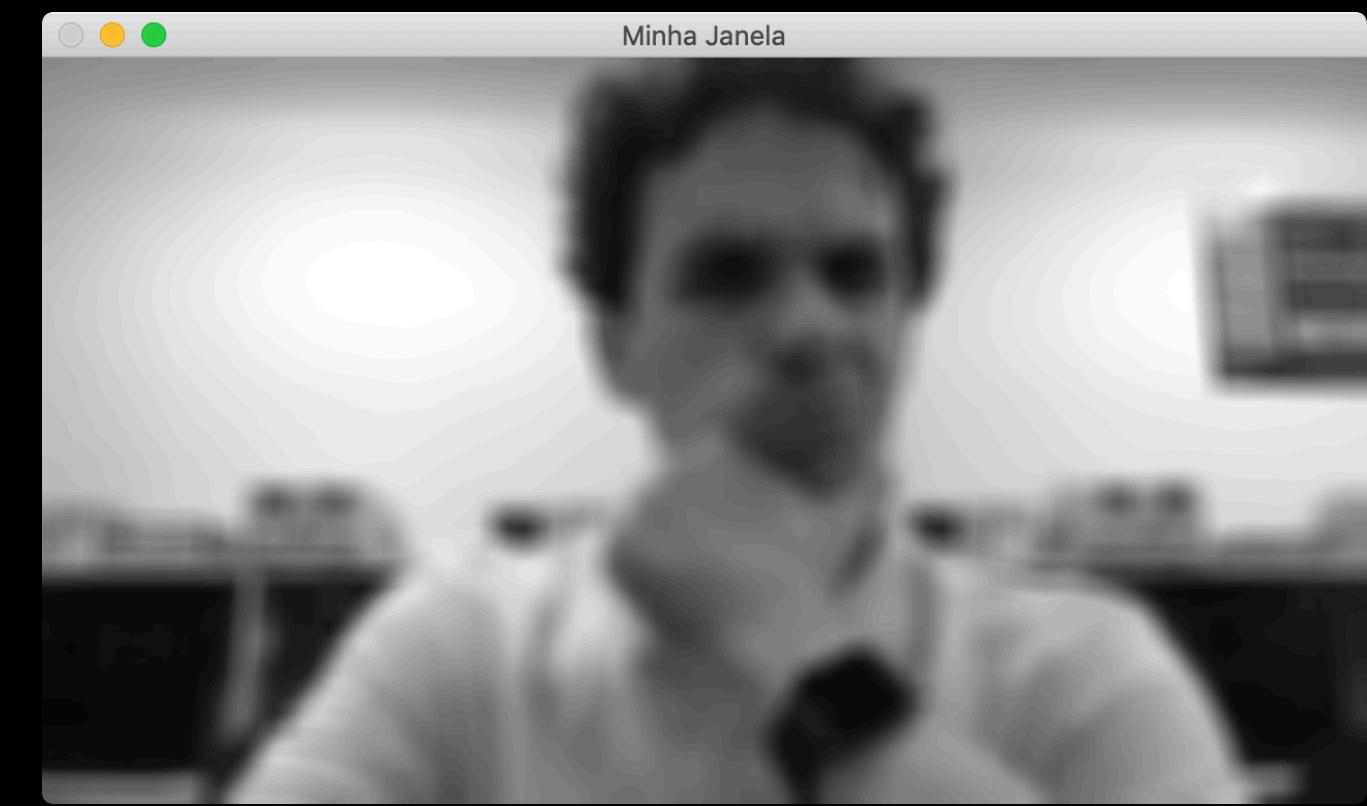
while True:
    _, imagem = stream.read()

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

    imshow("Minha Janela", imagem3)

    # interrompe 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()

    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)

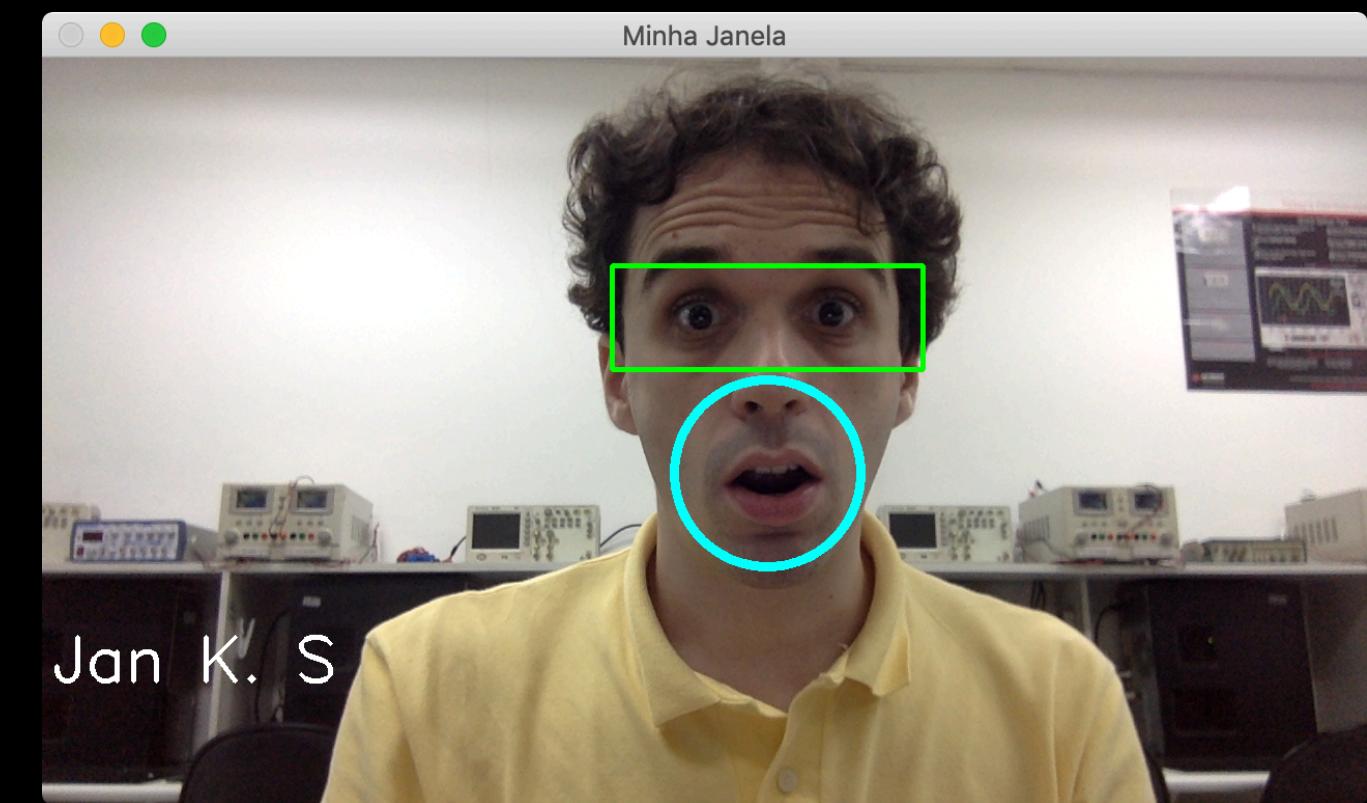
    putText(imagem, "Jan K. S", (10,600), color=(255,255,255),
thickness=4, fontFace=FONT_HERSHEY_SIMPLEX, fontScale=2)

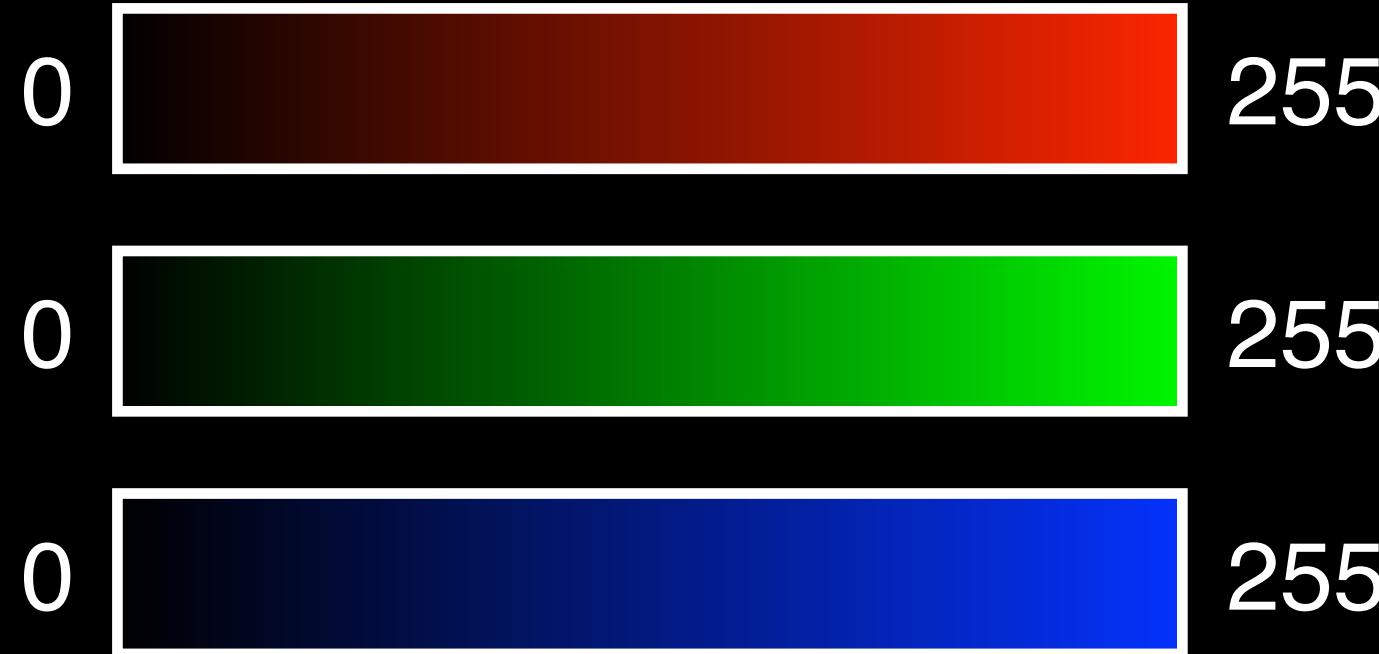
imshow("Minha Janela", imagem)

# interrompe quando tecla "q" for pressionada
if waitKey(1) & 0xFF == ord("q"):
    break

stream.release()
destroyAllWindows()

```





Espaço RGB

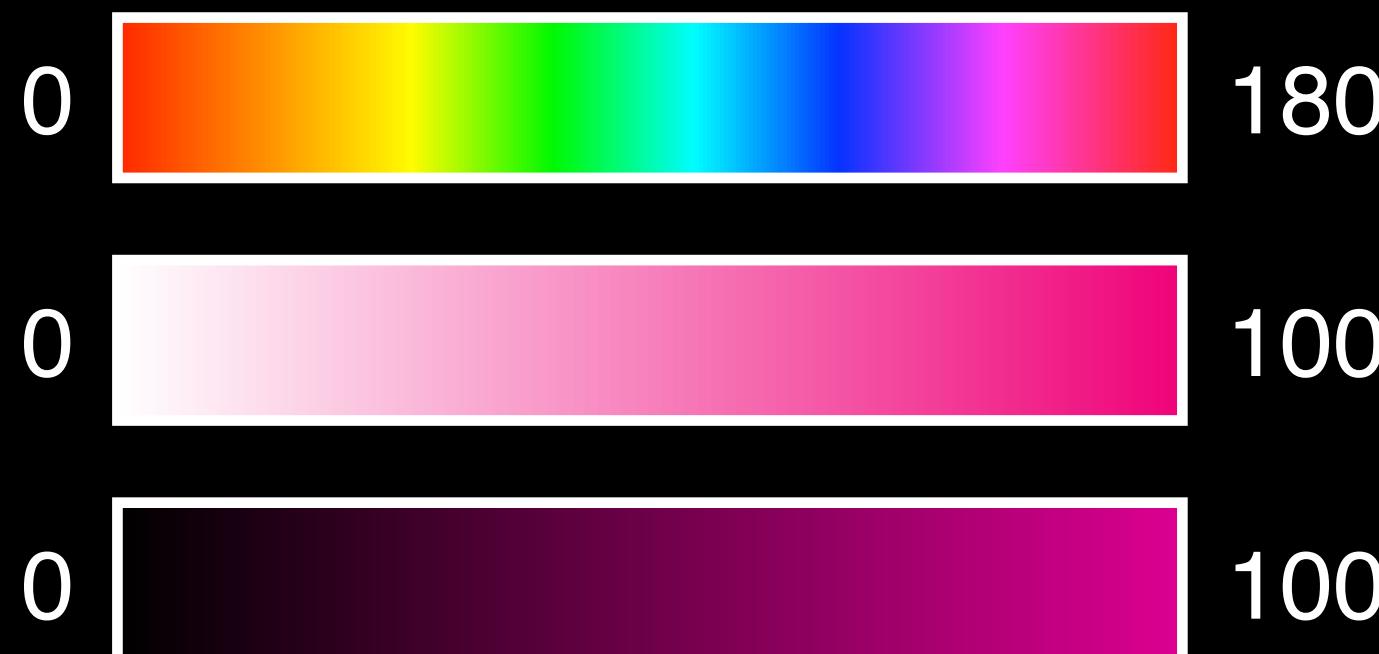
componentes vermelha + verde + azul



Espaço Gray

componente escala de cinza

útil para selecionar uma faixa de cores



Espaço HSV (ou HSB)

componentes matiz, saturação e valor de brilho

Espaço de Cores

```
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)

    imshow("Minha Janela", mascara)

    # interrompe 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()

    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)

    # interrompe 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()

    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)

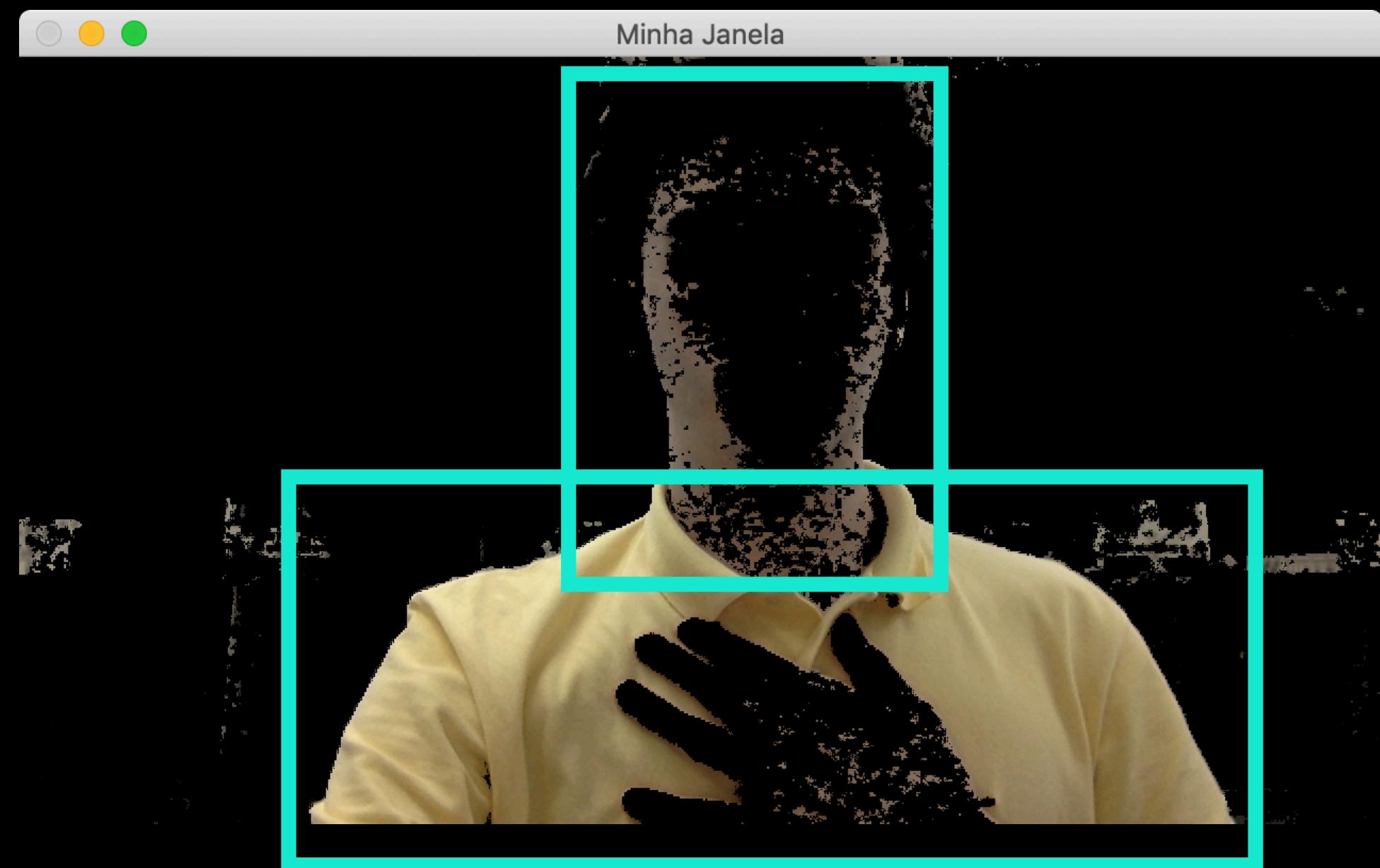
    imagem2 = bitwise_and(imagem, imagem, mask=mascara2)

    imshow("Minha Janela", imagem2)

    # interrompe quando tecla "q" for pressionada
    if waitKey(1) & 0xFF == ord("q"):
        break

stream.release()
destroyAllWindows()
```





Detecção de Objetos por Cor

```
from cv2 import *

stream = VideoCapture(0)

while True:
    ...
    contornos = findCountours(mascara, RETR_TREE,
CHAIN_APPROX_SIMPLE)
    for contorno in contornos:
        x, y, comprimento, altura = boundingRect(contorno)
        print(x, y, comprimento, altura)
    ...


```

Resumo da Ópera

Funcionalidade

Comandos

Sensor Ótico

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

Shield Motor
[acessar 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
[acessar documentação](#)

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

Serial no Arduino
[acessar documentação](#)

```
Serial.begin(9600); Serial1.begin(9600);
Serial.setTimeout(10);
Serial.print("Olá"); Serial1.println(2);
String texto = Serial.readString().trim();
```

PySerial
[acessar 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()
```

Resumo para o Projeto 09

Funcionalidade

Leitura Analógica
[acessar documentação](#)

Servomotor
[acessar documentação](#)

Braço Mecânico
[acessar documentação](#)

EEPROM
[acessar 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
[acessar documentação](#)

Interrupção
[acessar documentação](#)

Contagem
de Tempo
[acessar documentação](#)

Encoder Rotativo
[acessar 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);  
attachInterrupt(digitalPinToInterrupt(20), funcao, CHANGE);  
attachInterrupt(digitalPinToInterrupt(21), funcao, CHANGE);  
encoder.tick(); int posicao = encoder.getPosition();
```

Funcionalidade

Revisão de C++

Print Serial

Escrita/Leitura acessar documentação

GButton acessar documentação

ShiftDisplay acessar documentação

Timer1 acessar documentação

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;  
}
```

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

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

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

```
#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);
```

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