# Display TFT

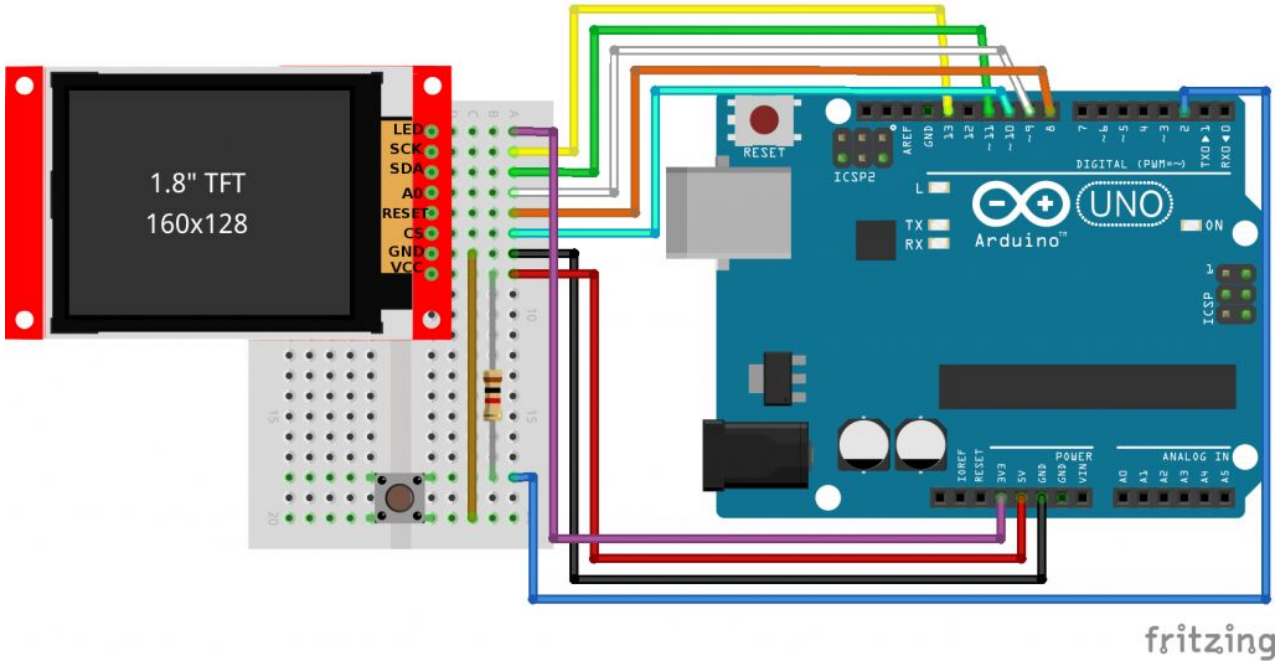
Eduardo Rodrigues - 86637
Keyla Rocha - 89387
Eloisa Morais - 89390

# O que é TFT

TFT é a tecnologia de telas mais comum usada em celulares de baixo e alto desempenho. É uma variação do Display de Cristal Líquido (LCD) que utiliza a tecnologia Transistor de Película Fina (TFT) para controlar cada pixel, e não linhas e colunas como era feito antigamente. Oferecendo assim melhor qualidade de imagem e maiores resoluções quando comparado com a geração de telas LCD anterior.

# Como utilizar o TFT

Montagem do circuito:



# Como utilizar o TFT

```
#include <SPI.h>
```

Controle de pinos

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

Controle dos gráficos e funções

```
#include <Adafruit_ST7735.h>
```

Biblioteca específica da tela

```
#include <SD.h>
```

Controle do cartão SD

Bibliotecas usadas:
- SPI
- AdaFruitGTX
- AdaFruitST7735
- SD(opcional)

# Funções básicas

Biblioteca Adafruit_GFX:

**Funções gráficas:**

- drawPixel (x, y, color): desenha um único pixel na cor desejada;
- drawLine (x0, y0, x1, y1, color): desenha linha entre as coordenadas fornecidas;
- drawFastHLine(x, y, width, color): desenha linha horizontal;
- drawFastVLine (x, y, height, color): desenha linha vertical;
- Funções para desenhar figuras geométricas mais comuns;

# Funções básicas

**Funções de texto:**

- setCursor (x, y): posição onde o texto será escrito;
- print(text) ou println(text): mostra o texto na tela;
- setTextColor (color): determina a cor do texto;
- setTextSize (fontsize): tamanho da fonte (vai de 1 a 10);

**Funções da tela:**

- fillScreen (color): preenche a tela com a cor desejada;

# Flappy bird

...

# Modificações

- Adição de novas telas:
  - Mensagem de parabenização caso o recorde máximo seja quebrado;
  - Mensagem para mostrar o recorde máximo;

- Gravação do recorde máximo na memória não volátil (EEPROM) do Arduino.

- Facilitação do jogo:
  - A cada 3 pontos, um cano não aparecerá;

# Flappy Bird - modificado

```
// ---------------
// game init
// ---------------
void game_init() {
  // clear screen
  TFT.fillScreen(BCKGRDCOL);
  // reset score
  score = 0;
  if (recordeMAX != 0) {
    oldrecorde = recordeMAX;
  }
  // init bird
  bird.x = 20;
  bird.y = bird.old_y = TFTH2 - BIRDH;
  bird.vel_y = -JUMP_FORCE;
  tmpx = tmpy = 0;
  // generate new random seed for the pipe gape
  randomSeed(analogRead(0));
  // init pipe
  pipe.x = TFTW;
  pipe.gap_y = random(20, TFTH-60);
}
```

Função para inicio do jogo:
- Preenche a tela com um fundo pré-definido;
- Grava a pontuação máxima, caso exista;
- Inicializa a posição do pássaro na tela;
- Gera uma seed para a posição do espaço do cano e inicializa o cano.

# Flappy Bird - modificado

```
while (1) {
  loops = 0;
  while( millis() > next_game_tick && loops < MAX_FRAMESKIP) {
    // ===============
    // input
    // ===============
    if ( !(PIND & (1<<PD2)) ) {
      // if the bird is not too close to the top of the screen apply jump force
      if (bird.y > BIRDH2*0.5) bird.vel_y = -JUMP_FORCE;
      // else zero velocity
      else bird.vel_y = 0;
    }

    // ===============
    // update
    // ===============
    // calculate delta time
    // ---------------
    old_time = current_time;
    current_time = millis();
    delta = (current_time-old_time)/1000;

    // bird
    // ---------------
    bird.vel_y += GRAVITY * delta;
    bird.y += bird.vel_y;
```

Loop do jogo:
- Recebe a entrada do pushbutton e faz o pássaro subir se ele não estiver muito perto da borda superior;

- Calcula o tempo que o pássaro levará para cair de acordo com a gravidade colocada.

# Flappy Bird - modificado

```
// if the bird hit the ground game over
if (bird.y > GAMEH-BIRDH) break;
// checking for bird collision with pipe
if (bird.x+BIRDW >= pipe.x-BIRDW2 && bird.x <= pipe.x+PIPEW-BIRDW) {
  // bird entered a pipe, check for collision
  if (bird.y < pipe.gap_y || bird.y+BIRDH > pipe.gap_y+GAPHEIGHT) break;
  else passed_pipe = true;
}
// if bird has passed the pipe increase score
else if (bird.x > pipe.x+PIPEW-BIRDW && passed_pipe) {
  passed_pipe = false;
  // erase score with background color
  TFT.setTextColor(BCKGRDCOL);
  TFT.setCursor( TFTW2, 4);
  TFT.print(score);
  // set text color back to white for new score
  TFT.setTextColor(ST7735_WHITE);
  // increase score since we successfully passed a pipe
  score++;
  if (score > recorde1) {
    recorde1 = score;
    if (EEPROM.read(0)==0 && recorde1 > EEPROM.read(0)){
        EEPROM.write(0, recorde1);
    }

    if (recorde1 > EEPROM.read(0)) {
       EEPROM.update(0, recorde1);
  }
```

Casos de colisão:

- Finaliza o jogo se o pássaro atingir o solo ou se houver colisão com o cano;
- Desenha a tela de derrota com a pontuação e o highscore;
- Aumenta a pontuação se não houve nenhuma colisão;
- Verifica se há um novo highscore.

# Flappy Bird - modificado

```
void novorecorde() {
  TFT.fillScreen(ST7735_YELLOW);
  TFT.setTextColor(ST7735_BLACK);
  TFT.setTextSize(2);
  // half width - num char * char width in pixels
  TFT.setCursor( TFTW2 - (9*6), TFTH2 - 4);
  TFT.print("PARABENS!");
  TFT.setTextSize(0);
  TFT.setCursor( 10, TFTH2 - 14);
  TFT.print("Novo recorde: ");
  TFT.println(recorde1);
  TFT.setCursor( TFTW2 - (12*3), TFTH2 + 12);
  TFT.println("aperte o botao");
  while (1) {
    // wait for push button
    if ( !(PIND & (1<<PD2)) ) break;
  }
}
void game_recorde() {
  TFT.fillScreen(ST7735_BLACK);
  TFT.setTextColor(ST7735_WHITE);
  TFT.setTextSize(2);
  TFT.setCursor( TFTW2 - (9*6.5), TFTH2 - 4);
  TFT.println ("Recorde:");
  TFT.setCursor( TFTW2 - (12*2) - 1, TFTH2 + 20);
  TFT.println(EEPROM.read(0));
  TFT.setTextSize(0);
  TFT.setCursor( TFTW2 - (12*4) - 1, TFTH2 + 40);
  TFT.print("aperte o botao");
   while (1) {
```

Highscores:

Função novorecorde:

- Desenha a tela de novo recorde;

Função game_recorde

- Desenha a tela final e mostra o recorde máximo;

# Sensores e Gráficos

...

# Valores de sensor

```
TFT.fillScreen(ST7735_WHITE);
TFT.setTextColor(ST7735_BLACK);
TFT.setTextSize(1);
TFT.setCursor(0, 10);
TFT.print("Distance:");
TFT.print(cmMsec);
TFT.print(" cm");

TFT.setCursor(0, 25);
TFT.print("Time:");
TFT.print(hour());
printDigits(minute(t));
printDigits(second(t));

TFT.setCursor(0, 40);
TFT.print("Minimum:");
TFT.println(minimum);
TFT.print("Maximum:");
TFT.println(maximum);
TFT.print("average:");
```

- Inicializa a tela com a cor branca e a fonte na cor preta;
- Imprime o valor do sensor, o contador, valor mínimo, máximo e média de valores;

# Gráfico com valores de sensor

```
#define trigger 4
#define echo 5
Ultrasonic ultrasonic(trigger, echo);

int xPos = 0;


void setup(){
  TFTscreen.begin();
  TFTscreen.background(0,0,0);
}

void loop(){
  long microsec = ultrasonic.timing();
  int sensor = ultrasonic.convert(microsec, Ultrasonic::CM);
  int graphHeight = map(sensor,0,90,0, TFTscreen.height());
  TFTscreen.stroke(255,255,255);
  TFTscreen.line(xPos, TFTscreen.height() - graphHeight, xPos, TFTscreen.height());
  if (xPos >= 160)
  {
    xPos = 0;
    TFTscreen.background(0,0,0);
  }
  else
    xPos++;
  delay(100);
```

- Inicializa a tela com fundo preto;

- Desenha o gráfico com a função *map* da biblioteca TFT;

# Mostrando imagens do SDCard

# Mostrando imagens


Para mostrar imagens com uma tela TFT, é necessário:
- Um cartão SD, ou um adaptador de MicroSD;
- Biblioteca SD instalada.
- Imagens em bmp(é possível converter qualquer imagem para esse formato com um editor de imagens);

# Mostrando imagens

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

  tft.initR(INITR_BLACKTAB);

  Serial.print("Carregando imagens...");
  if (!SD.begin(SD_CS)) {
    Serial.println("Falha ao carregar o SD, tente novamente.");
    return;
  }
  Serial.println("Concluido!");

  tft.setRotation(1); // Landscape

}

void loop() {

  bmpDraw("dpi.bmp", 0, 0);
  delay(3000);
  bmpDraw("ufv.bmp",0,0);
  delay(3000);
  bmpDraw("arduino.bmp",0,0);
  delay(3000);

}
```

Na função setup:
- Inicialização do SD;

Na função loop:
- Abertura da imagens por meio da função bmpDraw;

# Mostrando imagens

```
// Parse BMP header
if(read16(bmpFile) == 0x4D42) { // BMP signature
  Serial.print("File size: "); Serial.println(read32(bmpFile));
  (void)read32(bmpFile); // Read & ignore creator bytes
  bmpImageoffset = read32(bmpFile); // Start of image data
  Serial.print("Image Offset: "); Serial.println(bmpImageoffset, DEC);
  // Read DIB header
  Serial.print("Header size: "); Serial.println(read32(bmpFile));
  bmpWidth  = read32(bmpFile);
  bmpHeight = read32(bmpFile);
  if(read16(bmpFile) == 1) { // # planes -- must be '1'
    bmpDepth = read16(bmpFile); // bits per pixel
    Serial.print("Bit Depth: "); Serial.println(bmpDepth);
    if((bmpDepth == 24) && (read32(bmpFile) == 0)) { // 0 = uncompressed

      goodBmp = true; // Supported BMP format -- proceed!
      Serial.print("Image size: ");
      Serial.print(bmpWidth);
      Serial.print('x');
      Serial.println(bmpHeight);

      // BMP rows are padded (if needed) to 4-byte boundary
      rowSize = (bmpWidth * 3 + 3) & ~3;

      // If bmpHeight is negative, image is in top-down order.
      // This is not canon but has been observed in the wild.
      if(bmpHeight < 0) {
        bmpHeight = -bmpHeight;
        flip      = false;
      }
```

Função bmpDraw:
- Abre um arquivo bmp, lê pixel a pixel da imagem, verifica se a imagem está com extensão correta e “desenha” a imagem na tela.

# Perguntas?

...