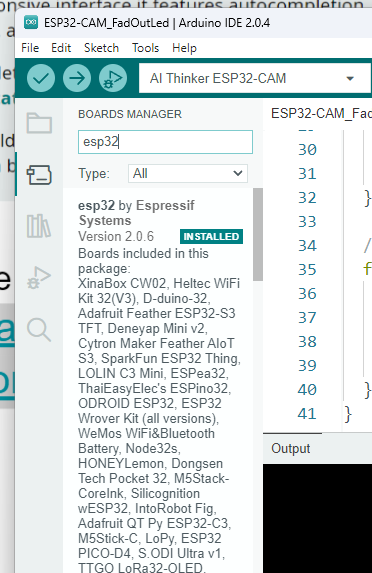
### **24/MARÇO**

Passos iniciais:

1. Conectar o cabo usb na placa gravadora do ESP32-CAM e no PC;
2. Instalar o ambiente de programação do Arduino: arduino.cc e escolha a versão do O.S. desejada



1. Cole esse link em PREFERÊNCIAS que fica no menu principal FILE / ARQUIVO <https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json>
2. Após colar o link, vá em TOOLS, depois BOARD, depois BOARDS MANAGER, daí vai abrir uma barra na lateral direita, onde vc busca por “esp32” e vai aparecer o pacote de placas da ESPRESSIF, e vc deve instalá-lo. Considere a última versão e não a versão 2.0.6 que aparece nesse print

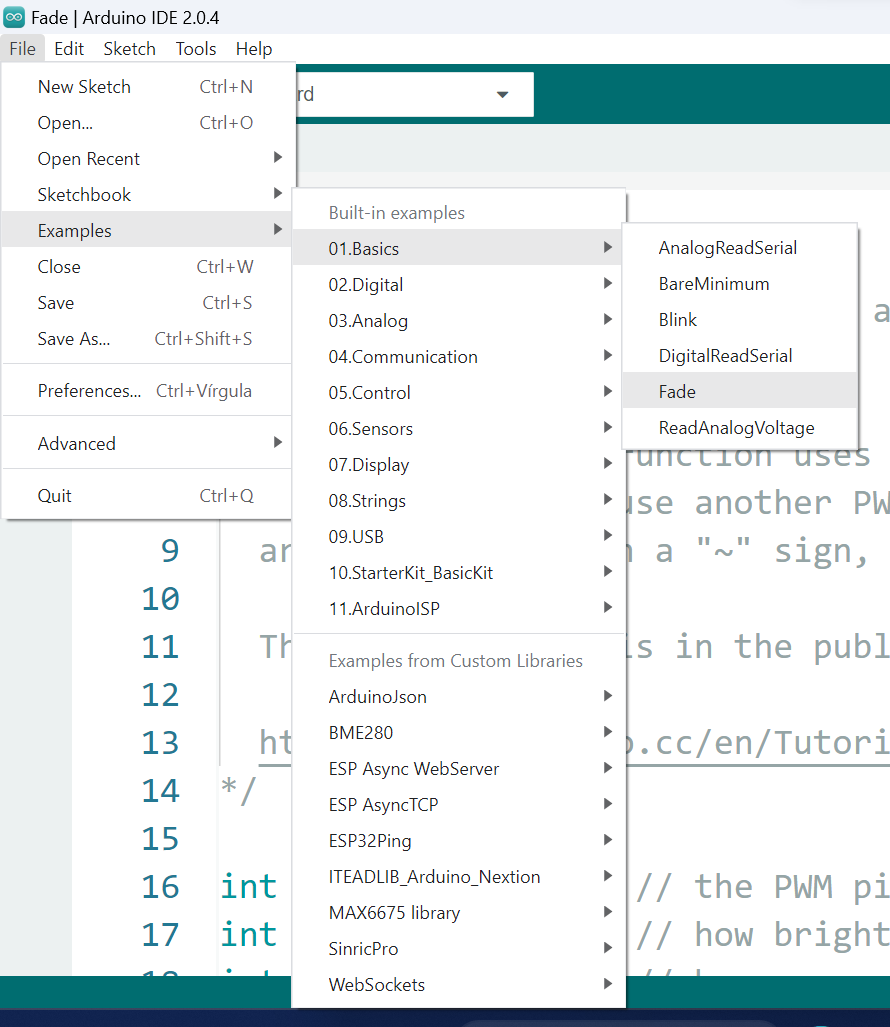


1. Crie o seu programa da seguinte forma: ARQUIVO / FILE e depois NEW SKETCH

### **31/MARÇO**

Correção do checkpoint 01

Questão 1 - Comentar o código-exemplo fade



int led = 4; //declaração de vars do tipo inteiro

int brightness = 0; //declaração de vars do tipo inteiro

int fadeAmount = 5; //declaração de vars do tipo inteiro

void setup() { //essa função ou estrutura define as default features ao energizar

pinMode(led, OUTPUT); //define o comportamento do pino do LED, nesse

//caso, será saída. Os tabs não são obrigatórios

}

void loop() { //essa função roda infinitamente em loop enquanto estiver

//energizado

analogWrite(led, brightness); //essa linha liga o LED em intensidade de 0

//a 1023. Nesse caso, o LED está em

//brightness=0

brightness = brightness + fadeAmount; //1º volta brightness = 5

//2º volta brightness = 5+5

//3º volta brightness = 5+5+5

//4º volta brightness = 5+5+5+5

if (brightness <= 0 || brightness >= 255) { //essa condição só será

//satisfeita se tivermos

//brightness <= 0 ou >= 255

//a condição >=255 ocorrerá na //51º

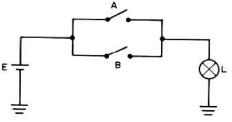
fadeAmount = -fadeAmount;

}

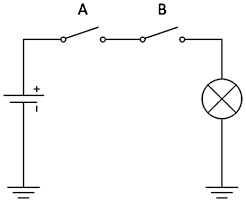
delay(30); //aguarde 30 milissegundos, o valor entre () é sempre dado em ms

} //fim do loop

Comportamento do programa: a cada volta, a variável brightness acrescenta +5 em seu valor e aciona o pino do LED, aumentando o seu brilho em passos de +5. O limite seria 1023, mas o programa não alcança esse valor para evitar o brilho máximo do flash. As condições do if só são satisfeitas a cada 51º voltas, ou seja, o brightness atinge 255 em 51 voltas e depois, ele decrementa até atingir 0 com outras 51 voltas, e assim, o LED aumenta e diminui seu brilho infinitamente até ser desernergizado. Por causa do delay, esse aumentar e diminuir de brilho é em “câmera lenta”. Sem o delay, os olhos humanos não perceberia a câmera lenta e seu comportamento parecia com o *digitalWrite* que é booleano (liga / desliga)

 → condição OU (uma das condições precisa ser

satisfeita)

 → condição E (as 2 condições precisam ser

satisfeitas)

Questão 2 → cada dupla terá um feedback da sua proposta.

Questão 3 → fazer um pisca-pisca

1 segundo ligado

aguarda 1 segundo

fazer isso 2x

500ms ligado

500ms aguardando

fazer isso 2x

retorna o loop

Código-fonte o original do pisca-pisca exemplo (Blink)

void setup() { //idem do código acima

pinMode(4, OUTPUT); //idem do código acima, porém, a variável

//LED\_BUILTIN não funciona para o ESP32-CAM

//tem que trocar por “4” (que é o pino)

}

void loop() { //idem do comentário anterior

digitalWrite(4, HIGH); //digitalWrite(pino, nível), nesse caso, nível máximo

delay(1000); //aguarda 1000 ms ou o mesmo que 1s no estado ligado

digitalWrite(4, LOW);

delay(1000); //aguarda 1000 ms ou o mesmo que 1s no estado desligado

digitalWrite(4, HIGH); //digitalWrite(pino, nível), nesse caso, nível máximo

delay(1000); //aguarda 1000 ms ou o mesmo que 1s no estado ligado

digitalWrite(4, LOW);

delay(1000); //aguarda 1000 ms ou o mesmo que 1s no estado desligado

digitalWrite(4, HIGH); //digitalWrite(pino, nível), nesse caso, nível máximo

delay(500); //aguarda 500 ms ou o mesmo que 0,5s no estado ligado

digitalWrite(4, LOW);

delay(500); //aguarda 500 ms ou o mesmo que 0,5s no estado desligado

digitalWrite(4, HIGH); //digitalWrite(pino, nível), nesse caso, nível máximo

delay(500; //aguarda 500 ms ou o mesmo que 0,5s no estado ligado

digitalWrite(4, LOW);

delay(500); //aguarda 500 ms ou o mesmo que 0,5s no estado desligado

}

BLINK SEM DELAY

/\*

Blink without Delay

Turns on and off a light emitting diode (LED) connected to a digital pin,

without using the delay() function. This means that other code can run at the

same time without being interrupted by the LED code.

The circuit:

- Use the onboard LED.

- Note: Most Arduinos have an on-board LED you can control. On the UNO, MEGA

and ZERO it is attached to digital pin 13, on MKR1000 on pin 6. LED\_BUILTIN

is set to the correct LED pin independent of which board is used.

If you want to know what pin the on-board LED is connected to on your

Arduino model, check the Technical Specs of your board at:

https://www.arduino.cc/en/Main/Products

created 2005

by David A. Mellis

modified 8 Feb 2010

by Paul Stoffregen

modified 11 Nov 2013

by Scott Fitzgerald

modified 9 Jan 2017

by Arturo Guadalupi

This example code is in the public domain.

https://www.arduino.cc/en/Tutorial/BuiltInExamples/BlinkWithoutDelay

\*/

const int ledPin = LED\_BUILTIN; // the number of the LED pin

// Variables will change:

int ledState = LOW; // ledState used to set the LED

// Generally, you should use "unsigned long" for variables that hold time

// The value will quickly become too large for an int to store

unsigned long millisAnterior = 0; // will store last time LED was updated

// variávle long pode ir de -2.147.483.648 a 2.147.483.647

// constants won't change:

const long intervalo = 1000; // interval at which to blink (milliseconds)

void setup() {

// set the digital pin as output:

pinMode(ledPin, OUTPUT);

digitalWrite(ledPin, HIGH); //forçando o led iniciar ligado assim a placa

//for energizada

}

void loop() {

// check to see if it's time to blink the LED; that is, if the difference

// between the current time and last time you blinked the LED is bigger than

// the interval at which you want to blink the LED.

unsigned long millisAtual = millis();

if (millisAtual - millisAnterior >= intervalo) {

millisAnterior = millisAtual;

// if the LED is off turn it on and vice-versa:

if (ledState == LOW) {

ledState = HIGH;

} else {

ledState = LOW;

}

// set the LED with the ledState of the variable:

digitalWrite(ledPin, ledState);

}

}

### **14/ABRIL**

void setup() {

Serial.begin(9600);

}

void loop() {

if (Serial.available() > 0) { //consultando a memória volátil

//lê o byte da porta serial

int byteLido = Serial.read();

//converte o byte em um número inteiro

//buffer [1.1, 1.2, 1.3, 1.4] --> isso é um vetor de 4 posições

char buffer[4]; //buffer para armazenar a string

buffer[0] = (char)byteLido;

buffer[1] = '\0'; //adiciona o caractere nulo para fazer o atoi parar de converter

//nessa posição

int valorInteiro = atoi(buffer);

//faz cálculos com o valor inteiro

int resultado = valorInteiro \* 2;

Serial.println(resultado);

}

}

Exercícios:

1. Elaborar um código-fonte que faça as 4 operações matemáticas com um número inserido no MONITOR SERIAL. Descarregue esse código no seu ESP32-CAM;

void setup() {

Serial.begin(9600);

}

void loop() {

if (Serial.available() > 0) { //consultando a memória volátil

// lê o byte da porta serial

int byteLido = Serial.read();

// converte o byte em um número inteiro

// buffer [1.1, 1.2, 1.3, 1.4] --> isso é um vetor de 4 posições

char buffer[4]; // buffer para armazenar a string

buffer[0] = (char)byteLido;

buffer[1] = '\0'; // adiciona o caractere nulo

int valorInteiro = atoi(buffer);

if(valorInteiro == 1 || valorInteiro == 2 || valorInteiro == 3 || valorInteiro == 4 || valorInteiro == 5 || valorInteiro == 6 || valorInteiro == 7 || valorInteiro == 8 || valorInteiro == 9)

{

// faz cálculos com o valor inteiro

float resultado = valorInteiro \* 2;

Serial.println(resultado);

}

}

}

1. Elaborar um código-fonte que faça a operação da exponencial com um número inserido no MONITOR SERIAL. Descarregue esse código no seu ESP32-CAM;

valor = x ^ 2;

Serial.printf(“%E”, valor); //serve para mostrar um valor no formato Notação

//Científica

### 10/MAIO

→ COMENTÁRIO DO CÓDIGO RECONHECEDOR DE OBJETOS.INO

#include <WebServer.h> //vc deve incluir as libraries adequadas para o seu projeto

#include <WiFi.h> //vai dar BO dizendo “arquivo no included” se não puser essas biblios

#include <esp32cam.h> //na pasta DOCUMENTOS/ARDUINO/LIBRARY

const char\* WIFI\_SSID = "nome da sua rede";

const char\* WIFI\_PASS = “senha do seu wifi";

WebServer server(80); //invocando um objeto HTTP usando a porta 80

static auto loRes = esp32cam::Resolution::find(320, 240); //alocação de RAM para guardar os pixels

static auto midRes = esp32cam::Resolution::find(350, 530);

static auto hiRes = esp32cam::Resolution::find(800, 600);

void serveJpg() //função chamada serverJpg

{

auto frame = esp32cam::capture(); //a variável frame está recebendo pixels por meio da função capture()

if (frame == nullptr) { //a var frame está com zero pixels

Serial.println("CAMERA NAO ESTA FUNCIONANDO");

server.send(503, "", "");

return;

}

Serial.printf("CAPTURE OK %dx%d %db\n", frame->getWidth(), frame->getHeight(), static\_cast<int>(frame->size())); //se tudo estiver bem, o MONITOR SERIAL vai imprimir CAPTURA OK

server.setContentLength(frame->size()); //preparando os bytes para serem enviados via html

server.send(200, "image/jpeg"); //preparando os bytes para serem enviados via html

WiFiClient client = server.client(); //preparando os bytes para serem enviados via html

frame->writeTo(client); //aqui eu envio os bytes

}

void handleJpgLo() //função que será chamada pela câmera e pela conexão WiFi para ajustar a taxa de fluxo de pixels

{

if (!esp32cam::Camera.changeResolution(loRes)) {

Serial.println("SET-LO-RES FAIL");

}

serveJpg();

}

void handleJpgHi() //função que será chamada pela câmera e pela conexão WiFi para ajustar a taxa de fluxo de pixels

{

if (!esp32cam::Camera.changeResolution(hiRes)) {

Serial.println("SET-HI-RES FAIL");

}

serveJpg();

}

void handleJpgMid() //função que será chamada pela câmera e pela conexão WiFi para ajustar a taxa de fluxo de pixels

{

if (!esp32cam::Camera.changeResolution(midRes)) {

Serial.println("SET-MID-RES FAIL");

}

serveJpg();

}

void setup(){

Serial.begin(115200); //nessa linha estamos iniciando o MONITOR SERIAL com 115200bps

Serial.println();

{

using namespace esp32cam;

Config cfg;

cfg.setPins(pins::AiThinker);

cfg.setResolution(hiRes);

cfg.setBufferCount(2);

cfg.setJpeg(80);

bool ok = Camera.begin(cfg);

Serial.println(ok ? "CAMERA OK" : "CAMERA FAIL");

}

WiFi.persistent(false);

WiFi.mode(WIFI\_STA);

WiFi.begin(WIFI\_SSID, WIFI\_PASS);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

}

Serial.print("http://");

Serial.println(WiFi.localIP());

Serial.println(" /cam-lo.jpg");

Serial.println(" /cam-hi.jpg");

Serial.println(" /cam-mid.jpg");

server.on("/cam-lo.jpg", handleJpgLo);

server.on("/cam-hi.jpg", handleJpgHi);

server.on("/cam-mid.jpg", handleJpgMid);

server.begin();

}

void loop()

{

server.handleClient();

}

**16/AGOSTO - AULA PRÁTICA COM ESP32-CAM**

Revisões:

1. Cuidado com o valor do baldrate do monitor serial. Ele tem que ser igual ao do valor Serial.begin(XXXX);
2. Se a sua placa não aceitar gravações, pressione os botões Reset ou Boot;
3. Tenha certeza que o cabo USB é de dados;
4. As bibliotecas #include <X> devem estar na pasta padrão C:\user\Documents\Arduino\libraries\ e descompactadas. Você pode até incluir as bibliotecas.zip que o Arduino IDE descompacta;
5. Modelo da placa: AI Thinker ESP32-CAM;
6. O nome do \*.ino tem que ser exatamente igual ao da sua pasta;
7. Lembre-se de selecionar a porta com do Arduino IDE para conectar a placa;
8. Grave um pisca-pisca usando o pino 4. Troque o “LED\_BUILTIN” por “4”;

https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package\_esp32\_index.json