



UNIVERSIDADE FEDERAL DE MINAS GERAIS  
ESCOLA DE ENGENHARIA

PROJETO DE SISTEMAS EMBUTIDOS

PROJETO IoT FINAL  
PROF. DIÓGENES CECILIO DA SILVA JÚNIOR

---

**Sistema de leitura de ingressos -  
Grupo 3**

---

*Alunos:*

*Camila Santana Braz*

*Elias Faria Silva*

*Giovanni Martins de Sá Júnior*

*Pedro Robles Dutenhofner*

*Matrícula:*

*2019027423*

*2018127254*

*2017001850*

*2018072557*

23 de dezembro de 2023

# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>2</b>
1.1	Descrição . . . . .	2
1.2	Motivação . . . . .	3
1.3	Justificativa . . . . .	4
1.4	Organização do documento . . . . .	4
<b>2</b>	<b>Desenvolvimento</b>	<b>4</b>
2.1	Gerenciamento de projeto . . . . .	4
2.2	Custos . . . . .	5
2.3	Arquiteturas . . . . .	6
2.3.1	Arquitetura do Sistema . . . . .	6
2.3.2	Arquitetura do Hardware . . . . .	6
2.3.3	Arquitetura do Software . . . . .	8
2.4	Entradas e saídas . . . . .	9
<b>3</b>	<b>Experimento</b>	<b>11</b>
3.1	Circuito . . . . .	11
3.1.1	Montagem . . . . .	11
3.1.2	Código . . . . .	12
<b>4</b>	<b>Conclusão</b>	<b>20</b>

# 1 Introdução

Com o avanço da tecnologia digital, a necessidade de sistemas eficientes e seguros para controle de acesso a eventos tornou-se extremamente necessária. Neste contexto, este relatório apresenta o desenvolvimento de um sistema embarcado dedicado à leitura de QR Codes presentes em ingressos para controle de entrada em eventos. O objetivo principal do sistema é automatizar e acelerar o processo de verificação e controle de entrada, garantindo maior segurança e eficiência operacional.

A importância de tais sistemas no cenário atual é indiscutível, onde a digitalização de processos se torna cada vez mais relevante, não apenas para a conveniência, mas também para a segurança dos eventos. A implementação deste sistema representa um avanço significativo na gestão de eventos, oferecendo um método confiável e eficiente para o controle de acesso.

Este relatório detalha as etapas do desenvolvimento, começando pela concepção do projeto, passando pelo design e implementação do hardware e software, e finalizando com os testes e a validação do sistema. Iremos explicar detalhadamente o projeto desenvolvido em termos de hardware, isto envolve os equipamentos utilizados e os circuitos projetados. Além disso, explicaremos a implementação do código necessário para execução correta das tarefas propostas e o correto funcionamento do sistema.

## 1.1 Descrição

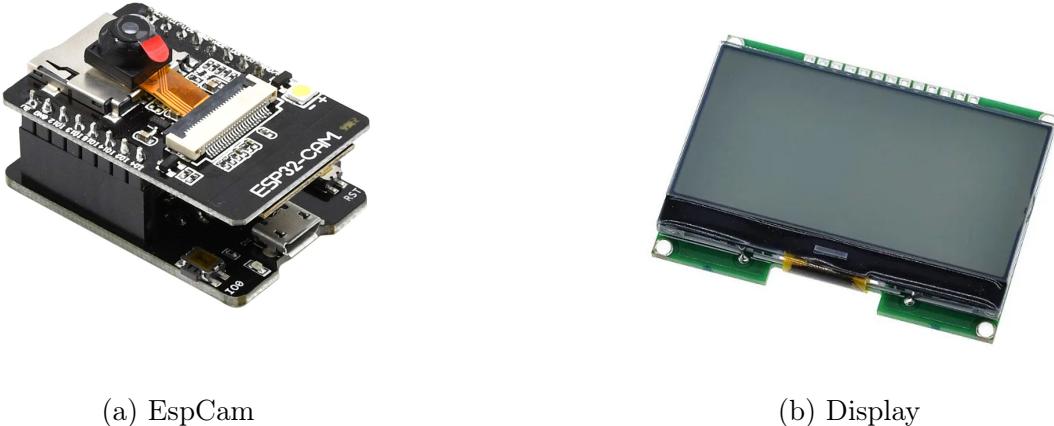
Com o crescente uso de tecnologia digital em vários setores, a eficiência e segurança dos sistemas de controle de acesso a eventos tornaram-se cruciais. Nesse contexto, este relatório detalha o desenvolvimento de um sistema embarcado inovador para leitura de QR Codes, utilizado em ingressos para gerenciar o acesso a eventos. O sistema foi projetado com o objetivo principal de automatizar e agilizar o processo de verificação de entradas, assegurando uma operação mais segura e eficiente.

A relevância de tais sistemas no cenário contemporâneo é inquestionável, considerando a crescente digitalização de processos. Isso não se limita apenas à conveniência, mas estende-se significativamente à segurança em eventos. A implementação deste sistema embarcado é um passo importante na modernização da gestão de eventos, oferecendo uma solução confiável e eficiente para o controle de acesso.

Além de melhorar a segurança e a eficiência, o sistema proposto também oferece outras vantagens significativas. Primeiramente, ele reduz significativamente o tempo de espera na entrada dos eventos, melhorando a experiência do usuário. Ao digitalizar o processo de entrada, também minimiza a necessidade de interações manuais, reduzindo o risco de erros humanos e aumentando a precisão no controle de acesso. Outro aspecto crucial é a capacidade do sistema de armazenar dados de forma segura, permitindo uma análise posterior para melhorar futuros eventos.

Neste contexto, o sistema embarcado desenvolvido será empregado nas entradas de eventos, oferecendo maior comodidade, agilidade e segurança para ambas as partes envolvidas: os organizadores e os participantes. Cada participante receberá um QR Code exclusivo, que será enviado para o seu smartphone assim que a compra do

ingresso for confirmada. O sistema utiliza uma ESPCam para ler e decodificar estes QR Codes. Quando um QR Code é reconhecido, o sistema consulta um banco de dados para verificar a validade do ingresso e determinar quais áreas o portador tem permissão para acessar. Estas informações são exibidas em um display integrado ao microcontrolador, fornecendo orientações claras tanto para o visitante quanto para os funcionários do evento.



(a) EspCam

(b) Display

Figura 1: Componentes do sistema.

## 1.2 Motivação

As motivações deste trabalho incluem:

- **Agilidade no Check-In:** Reduzir filas e tempo de espera na entrada, proporcionando uma experiência mais agradável para os visitantes.
- **Segurança:** Garantir que apenas pessoas autorizadas acessem determinadas áreas, aumentando a segurança do evento.
- **Eficiência de Custo:** Reduzir a necessidade de grande número de funcionários para verificar manualmente os ingressos.
- **Dados em Tempo Real:** Coletar dados em tempo real sobre o fluxo de visitantes, o que pode ajudar na gestão do evento e em planejamentos futuros.



(a) Evento em que o sistema pode ser utilizado



(b) Filas grandes: dores de organizadores e participantes de eventos

Figura 2

### 1.3 Justificativa

O desenvolvimento desse sistema é justificado pela crescente demanda por soluções tecnológicas em eventos de grande escala. Além disso, o uso de ESPCam e displays para a leitura e apresentação de informações de QR Codes é uma solução de baixo custo, eficiente e pode ser escalável. Isso não apenas melhora a experiência do usuário, mas também oferece uma forma eficaz e moderna de controle de acesso e gerenciamento de eventos.

### 1.4 Organização do documento

Este relatório detalha as etapas de desenvolvimento, desde a concepção do projeto até a implementação prática do sistema, incluindo os desafios enfrentados e as soluções adotadas. Desta forma, ele está organizado da seguinte maneira: a próxima seção apresenta o desenvolvimento do sistema, apresentando os requisitos e custos, os diagramas de gerenciamento de projeto, as arquiteturas e a montagem e o código do circuito. A seção 3 descreve e discute os resultados do projeto e, finalmente, a seção 4 apresenta a conclusão e os trabalhos futuros.

## 2 Desenvolvimento

### 2.1 Gerenciamento de projeto

Para gerenciar o projeto, utilizou-se o Project Model Canvas (PMC). O PMC é uma metodologia de gerenciamento de projetos que oferece uma forma visual e intuitiva de planejar e comunicar aspectos-chave de um projeto. Inspirado no Business Model Canvas, ele foi desenvolvido para simplificar e tornar mais eficiente o processo de criação e gestão de projetos. O modelo consiste em um quadro dividido em várias seções, cada uma representando um componente vital do projeto, como objetivos, stakeholders, equipe, recursos, riscos, entregas, e cronograma. O uso do Project Model Canvas facilita a colaboração e o engajamento da equipe, permitindo que todos os envolvidos visualizem e entendam o panorama geral e os detalhes do projeto de forma rápida e clara. Ele é particularmente útil nas fases iniciais de planejamento,

onde ideias podem ser rapidamente esboçadas, discutidas e ajustadas, promovendo uma maior agilidade e adaptabilidade na gestão de projetos. A figura 3 apresta o PMC do projeto.

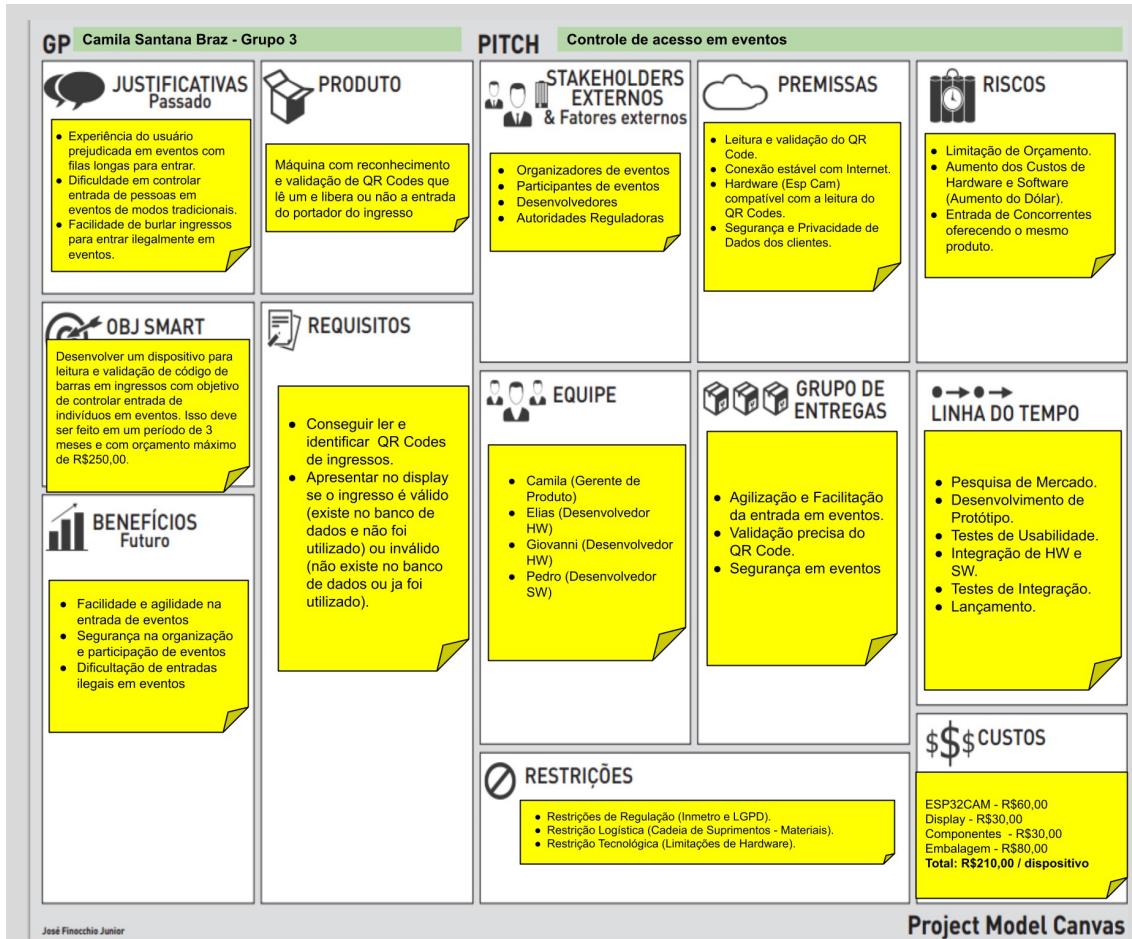


Figura 3: PMC do projeto.

## 2.2 Custos

O sistema foi projetado para ter um custo máximo de R\$210,00. Isso representa a soma do custo de cada componente da tabela 1.

Componentes	Custo
Esp 32 com câmera OV2640	R\$ 60,00
Display LCD	R\$ 30,00
Embalagem	R\$ 80,00
Componentes Eletrônicos + Conectores + Protoboard	R\$ 30,00

Tabela 1: Lista de Componentes e Custos

## 2.3 Arquiteturas

### 2.3.1 Arquitetura do Sistema

A arquitetura de sistemas refere-se à estrutura organizacional de um sistema, incluindo seus componentes, as relações entre esses componentes e os princípios e diretrizes que orientam seu design e evolução.

O sistema embarcado representado na figura abaixo é composto por quatro componentes principais: o módulo da câmera, o sistema central de processamento (microcontrolador), a interface com o usuário e a base de dados.

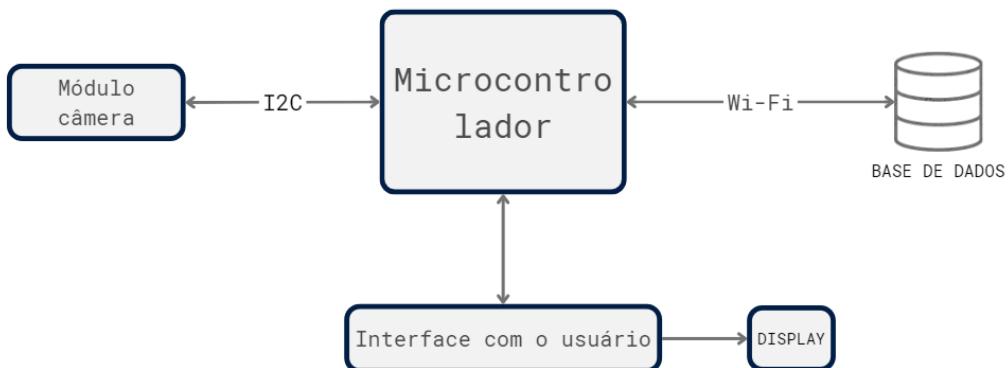


Figura 4: Arquitetura do sistema embarcado com ESPCam32 e display.

A ESPCam32 é uma placa de desenvolvimento que combina um módulo de câmera com um microcontrolador ESP32. Esta placa é amplamente utilizada em projetos de IoT devido à sua capacidade de processamento, conectividade Wi-Fi integrada e facilidade de uso. Além disso, a ESPCam32 pode ser equipada com um display LCD, permitindo a criação de interfaces de usuário interativas.

O módulo de câmera da ESPCam32 é capaz de capturar imagens e vídeos de alta qualidade. A comunicação entre o módulo de câmera e o sistema central é realizada através do protocolo I2C, que permite uma transferência de dados eficiente e confiável.

A ESPCam32 possui conectividade Wi-Fi que permite a comunicação sem fio com a base de dados. Isso possibilita o armazenamento e a análise dos dados capturados pela câmera em tempo real.

A interface com o usuário é gerenciada através do display LCD acoplado à ESP-Cam32. Este display pode mostrar informações relevantes para o usuário ou servir como uma tela de interação para controlar o sistema.

### 2.3.2 Arquitetura do Hardware

A arquitetura de hardware refere-se à organização e design dos componentes físicos de um sistema de computador. Ela determina como o hardware funciona e interage para formar um sistema de computação. A figura 5 apresenta a arquitetura de hardware do sistema desenvolvido.

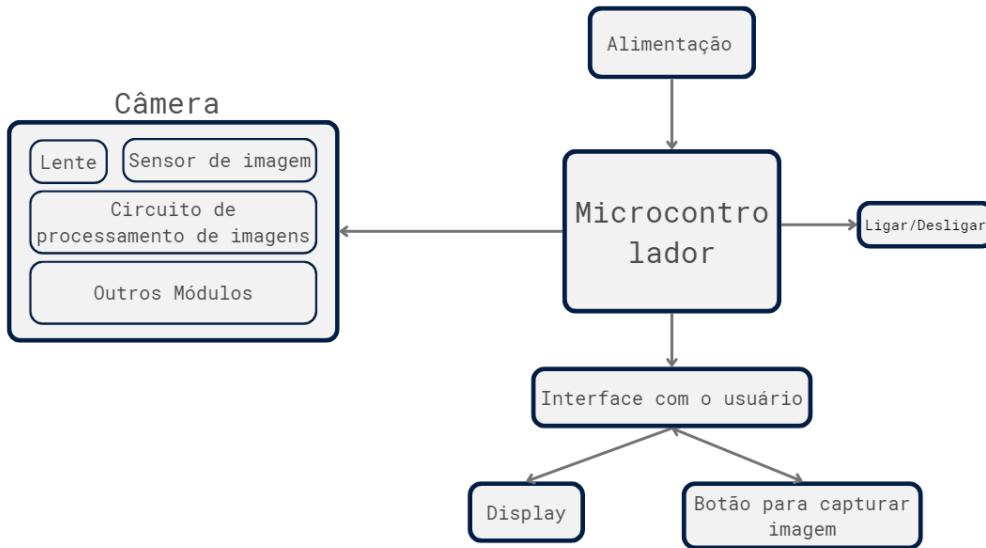


Figura 5: Arquitetura de hardware.

Essa arquitetura é um sistema de câmera integrado com uma interface de usuário. Na seção da câmera, há uma lente responsável por captar a luz e focá-la no sensor de imagem, que por sua vez converte essa luz em sinais elétricos. Esses sinais são então processados pelo circuito de processamento de imagens, que ajusta aspectos como cor e brilho para criar a imagem digital. Além disso, existem outros módulos como:

- **Sensor de Imagem:** O sensor, como o OV2640, OV7670, ou similares, é responsável por converter a luz em sinais eletrônicos. Cada sensor tem características específicas em termos de resolução, sensibilidade à luz e capacidade de captura de cor.
- **Lente:** A lente é montada na frente do sensor de imagem. Ela foca a luz no sensor e pode variar em termos de qualidade óptica, distância focal e capacidade de foco (fixo ou ajustável).
- **Circuito de Processamento de Imagem:** Alguns módulos de câmera incluem circuitos integrados para processar o sinal de vídeo capturado pelo sensor. Este processamento pode incluir conversão de formato, compressão de imagem, e ajustes automáticos de imagem.
- **Interface de Comunicação:** Uma interface de comunicação, como I2C ou SPI, é usada para conectar o módulo de câmera ao ESP32. Através desta interface, o ESP32 pode enviar comandos para o módulo e receber dados de imagem.
- **Conectores e Pinagem:** O módulo terá pinos ou conectores para a integração com o ESP32. Isso inclui pinos para alimentação (VCC e GND), comunicação de dados e, às vezes, sinais adicionais para controle.
- **Regulador de Tensão e Componentes de Filtragem:** Para garantir que

o módulo de câmera opere na tensão correta e com estabilidade, um regulador de tensão e componentes para filtrar ruídos elétricos podem estar presentes.

- **LEDs ou Iluminação Auxiliar:** Alguns módulos podem incluir LEDs para iluminação suplementar, ajudando na captura de imagens em ambientes com pouca luz.

O microcontrolador é o núcleo central que gerencia todas as operações do sistema, desde o processamento dos dados da imagem até a execução do software que permite ao usuário operar a câmera. Ele se conecta a um sistema de alimentação que fornece energia necessária para todos os componentes eletrônicos. É possível também ligar e desligar o sistema.

Para a interface com o usuário, o sistema conta com um display que permite ao usuário verificar o QR Code é válido ou não, e um botão para capturar imagem, que quando pressionado, instrui a câmera a realizar uma captura de foto. O microcontrolador atua como um elo entre a entrada do usuário, através do botão, e a captura e processamento da imagem pela câmera, além de enviar o status de validação do QR Code.

### 2.3.3 Arquitetura do Software

A arquitetura de software é um conjunto de princípios, técnicas e práticas usadas para projetar e organizar sistemas de software. Ela se refere à estrutura de alto nível de um sistema de software e ao modo como essa estrutura proporciona uma base para o desenvolvimento e a manutenção do software.

A imagem 6 descreve um diagrama de fluxo para o sistema.

- **Código:** Onde as principais operações são coordenadas.
- **Identificar QR Code:** Componente responsável pela detecção inicial de um QR Code.
- **Captura da imagem:** Ação associada à obtenção de uma imagem que contém o QR Code.
- **Validar QR Code:** Verifica a autenticidade do QR Code, seu prazo de validade e se já foi utilizado.
- **Acessar Banco de Dados:** Operação de consulta ou atualização em um banco de dados relacionado ao QR Code.
- **Leitura dos botões:** Interpreta a interação do usuário com a interface física, como botões.
- **Conexão Wi-Fi:** Necessária para a comunicação de rede, como acesso ao banco de dados online.
- **Renderizar a UI no display:** Apresenta a interface de usuário em um display.

- **Gerar da mensagem do status do QR Code (válido/inválido):** Mostra o resultado da validação do QR Code.

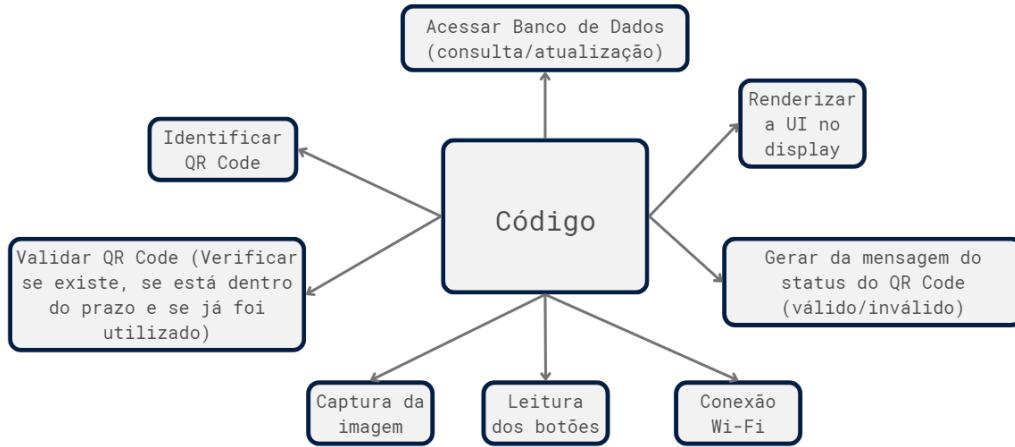


Figura 6: Arquitetura de hardware.

## 2.4 Entradas e saídas

As relações entre os componentes de entrada e saída estão disponíveis nas tabelas 2, 3 e 4.

Conexão	ESP32
PINO 1	GPIO12
PINO 2	GND

Tabela 2: Conexões do Push Button

Canal/Sinal	ESP32
Canal de dados da câmera (D0-D7)	D0-D7
Clock externo para a câmera (XCLK)	XCLK
Clock de pixel para a câmera (PCLK)	PCLK
Sinal de sincronização vertical (VSYNC)	VSYNC
Sinal de referência horizontal (HREF)	HREF
Linha de controle de I2C (SIOC)	SIOC
Linha de dados de I2C (SIOD)	SIOD

Tabela 3: Conexões da Câmera

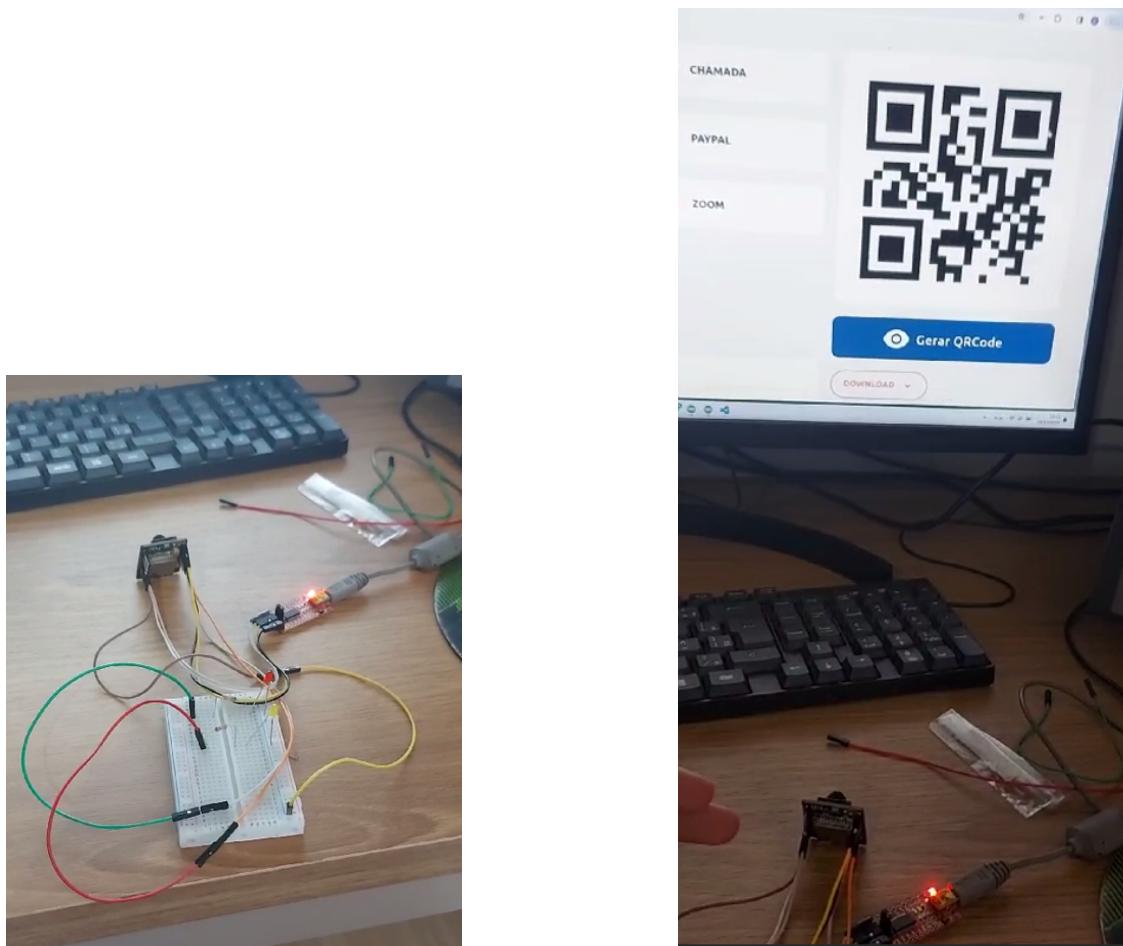
Conexão	ESP32
VSS	GND
VDD	5V
V0	10K POT
RS	GPIO19
RW	GND
E	GPIO23
D4-D7	GPIO18-GPIO15
A	5V
K	GND

Tabela 4: Conexões do Display

### 3 Experimento

Neste terceiro momento, será discutido a implementação feita do projeto do leitor de ingressos. Assim, esta terceira seção se divide no circuito em si, mostrando fotos da implementação do mesmo, a discussão da montagem efetuada para o protótipo, e por último, uma discussão pontual explicando o funcionamento do código como um todo.

#### 3.1 Circuito



(a) Montagem do Circuito para a leitura do QR Code

(b) Câmera capturando QR Code apontado para tela

Figura 7

##### 3.1.1 Montagem

A respeito da montagem, ela se baseou na associação do Esp com dois leds principais, e que confirmam ou recusam a leitura, e um terceiro led associado com o flash da câmera.

### 3.1.2 Código

Abaixo, será descrita em passos a implementação desenvolvida no código:

- Importação de Bibliotecas:** bibliotecas necessárias para o projeto, como a biblioteca da câmera, a biblioteca para processamento de QR Code (Quirc), a biblioteca WiFi, a HTTPClient para fazer solicitações HTTP, e a ArduinoJson para manipulação de dados JSON.

```
/* Including the libraries. */
#include "esp_camera.h"
#include "soc/soc.h"
#include "soc/rtc_CNTL_REG.h"
#include "quirc.h"
#include "WiFi.h"
#include <HTTPClient.h>
#include <ArduinoJson.h>
/* ===== */
```

- Configuração Inicial:** Após a declaração das bibliotecas são feitas as configurações dos Leds utilizados no projeto, além da configuração da conexão e da tarefa a ser executada.

```
#define LED_RED 12
#define LED_YELLOW 13
#define LED_LED 4

const char *ssid = "TELEVISAO";
const char *password = "agoratemtvboa";

// creating a task handle
TaskHandle_t QRCodeReader_Task;
```

- Seleção do modelo da câmera e configuração de GPIO da Câmera escolhida:** Para a execução do trabalho, foram tratadas a utilização e configuração de diferentes modelos de câmeras, contudo, para este trabalho, foi escolhido a modelo *CAMERA\_MODEL\_AI\_THINKER*.

```
/* ===== Select camera model ===== */
//#define CAMERA_MODEL_WROVER_KIT
//#define CAMERA_MODEL_ESP_EYE
//#define CAMERA_MODEL_M5STACK_PSRAM
//#define CAMERA_MODEL_M5STACK_WITHOUT_PSRAM
//#define CAMERA_MODEL_M5STACK_WITHOUT_PSRAM
#define CAMERA_MODEL_AI_THINKER
/* ===== */

/* ===== GPIO of camera models ===== */
#if defined(CAMERA_MODEL_WROVER_KIT)
#define PWDN_GPIO_NUM -1
```

```
#define RESET_GPIO_NUM -1
#define XCLK_GPIO_NUM 21
#define SIOD_GPIO_NUM 26
#define SIOC_GPIO_NUM 27

#define Y9_GPIO_NUM 35
#define Y8_GPIO_NUM 34
#define Y7_GPIO_NUM 39
#define Y6_GPIO_NUM 36
#define Y5_GPIO_NUM 19
#define Y4_GPIO_NUM 18
#define Y3_GPIO_NUM 5
#define Y2_GPIO_NUM 4
#define VSYNC_GPIO_NUM 25
#define HREF_GPIO_NUM 23
#define PCLK_GPIO_NUM 22

#elif defined(CAMERA_MODEL_ESP_EYE)
#define PWDN_GPIO_NUM -1
#define RESET_GPIO_NUM -1
#define XCLK_GPIO_NUM 4
#define SIOD_GPIO_NUM 18
#define SIOC_GPIO_NUM 23

#define Y9_GPIO_NUM 36
#define Y8_GPIO_NUM 37
#define Y7_GPIO_NUM 38
#define Y6_GPIO_NUM 39
#define Y5_GPIO_NUM 35
#define Y4_GPIO_NUM 14
#define Y3_GPIO_NUM 13
#define Y2_GPIO_NUM 34
#define VSYNC_GPIO_NUM 5
#define HREF_GPIO_NUM 27
#define PCLK_GPIO_NUM 25

#elif defined(CAMERA_MODEL_M5STACK_PSRAM)
#define PWDN_GPIO_NUM -1
#define RESET_GPIO_NUM 15
#define XCLK_GPIO_NUM 27
#define SIOD_GPIO_NUM 25
#define SIOC_GPIO_NUM 23

#define Y9_GPIO_NUM 19
#define Y8_GPIO_NUM 36
#define Y7_GPIO_NUM 18
#define Y6_GPIO_NUM 39
#define Y5_GPIO_NUM 5
#define Y4_GPIO_NUM 34
```

```

#define Y3_GPIO_NUM      35
#define Y2_GPIO_NUM      32
#define VSYNC_GPIO_NUM   22
#define HREF_GPIO_NUM    26
#define PCLK_GPIO_NUM    21

#elif defined(CAMERA_MODEL_M5STACK_WITHOUT_PSRAM)
#define PWDN_GPIO_NUM    -1
#define RESET_GPIO_NUM   15
#define XCLK_GPIO_NUM    27
#define SIOD_GPIO_NUM    25
#define SIOC_GPIO_NUM    23

#define Y9_GPIO_NUM      19
#define Y8_GPIO_NUM      36
#define Y7_GPIO_NUM      18
#define Y6_GPIO_NUM      39
#define Y5_GPIO_NUM      5
#define Y4_GPIO_NUM      34
#define Y3_GPIO_NUM      35
#define Y2_GPIO_NUM      17
#define VSYNC_GPIO_NUM   22
#define HREF_GPIO_NUM    26
#define PCLK_GPIO_NUM    21

#elif defined(CAMERA_MODEL_AI_THINKER)
#define PWDN_GPIO_NUM    32
#define RESET_GPIO_NUM   -1
#define XCLK_GPIO_NUM    0
#define SIOD_GPIO_NUM    26
#define SIOC_GPIO_NUM    27

#define Y9_GPIO_NUM      35
#define Y8_GPIO_NUM      34
#define Y7_GPIO_NUM      39
#define Y6_GPIO_NUM      36
#define Y5_GPIO_NUM      21
#define Y4_GPIO_NUM      19
#define Y3_GPIO_NUM      18
#define Y2_GPIO_NUM      5
#define VSYNC_GPIO_NUM   25
#define HREF_GPIO_NUM    23
#define PCLK_GPIO_NUM    22

#else
#error "Camera model not selected"
#endif

```

4. **Struct do QRCode:** responsável por armazenar os dados relacionados ao QR Code que foi lido.

```

    struct QRCodeData
    {
        bool valid;
        int dataType;
        uint8_t payload[1024];
        int payloadLen;
    };

    struct quirc *q = NULL;
    uint8_t *image = NULL;
    camera_fb_t * fb = NULL;
    struct quirc_code code;
    struct quirc_data data;
    quirc_decode_error_t err;
    struct QRCodeData qrCodeData;
    String QRCodeResult = "";

```

- 5. Função Setup() e Loop():** A função **setup** é chamada uma vez, quando o microcontrolador for inicializado. O setup é responsável basicamente por executar a configuração inicial e inicialização dos periféricos.

A primeira configuração feita, é a desativação do Detector de Brownout, que é um mecanismo de proteção contra baixa tensão na alimentação do dispositivo. Em seguida, é efetuado a comunicação serial com uma taxa de baud de 115200, e a configuração da saída de depuração para comunicação serial.

Na sequência, efetuam-se a configuração dos Leds de saída, da conexão de Wi-fi fornecendo as credenciais definidas anteriormente, e a configuração da câmera. Por último, ainda na função **setup**, é também criada uma tarefa que executa a leitura de um QR Code em tempo real em paralelo com o loop principal.

Para a função **loop**, ela é executada continuamente enquanto o microcontrolador se mantém ligado. Nela, foi feita apenas uma implementação que executa uma pausa de 1 milisegundo.

```

/* VOID SETUP() */
void setup() {
    // put your setup code here, to run once:

    // Disable brownout detector.
    WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);

    /* Init serial communication speed (baud rate). */
    Serial.begin(115200);
    Serial.setDebugOutput(true);

```

```

Serial.println();

pinMode(LED_RED, OUTPUT);
pinMode(LED_YELLOW, OUTPUT);
pinMode(LED_LED, OUTPUT);

// Connect to WiFi
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
}
Serial.println("Connected to WiFi");
Serial.println("IP Address: " + WiFi.localIP().toString());
/* ----- */

/* Camera configuration. */
Serial.println("Start configuring and initializing the
camera...");
camera_config_t config;
config.ledc_channel = LEDC_CHANNEL_0;
config.ledc_timer = LEDC_TIMER_0;
config.pin_d0 = Y2_GPIO_NUM;
config.pin_d1 = Y3_GPIO_NUM;
config.pin_d2 = Y4_GPIO_NUM;
config.pin_d3 = Y5_GPIO_NUM;
config.pin_d4 = Y6_GPIO_NUM;
config.pin_d5 = Y7_GPIO_NUM;
config.pin_d6 = Y8_GPIO_NUM;
config.pin_d7 = Y9_GPIO_NUM;
config.pin_xclk = XCLK_GPIO_NUM;
config.pin_pclk = PCLK_GPIO_NUM;
config.pin_vsync = VSYNC_GPIO_NUM;
config.pin_href = HREF_GPIO_NUM;
config.pin_sscb_sda = SIOD_GPIO_NUM;
config.pin_sscb_scl = SIOC_GPIO_NUM;
config.pin_pwdn = PWDN_GPIO_NUM;
config.pin_reset = RESET_GPIO_NUM;
config.xclk_freq_hz = 10000000;
config.pixel_format = PIXFORMAT_GRAYSCALE;
config.frame_size = FRAMESIZE_QVGA;
config.jpeg_quality = 15;
config.fb_count = 1;

#if defined(CAMERA_MODEL_ESP_EYE)
    pinMode(13, INPUT_PULLUP);
    pinMode(14, INPUT_PULLUP);
#endif

```

```

esp_err_t err = esp_camera_init(&config);
if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x",
                  err);
    ESP.restart();
}

sensor_t * s = esp_camera_sensor_get();
s->set_framesize(s, FRAMESIZE_QVGA);

Serial.println("Configure and initialize the camera
               successfully.");
Serial.println();

xTaskCreatePinnedToCore(
    QRCodeReader,           /* Task function. */
    "QRCodeReader_Task",   /* name of task. */
    10000,                 /* Stack size of task */
    NULL,                  /* parameter of the task
                           */
    1,                     /* priority of the task
                           */
    &QRCodeReader_Task,    /* Task handle to keep
                           track of created task */
    0);                   /* pin task to core 0 */

/* -----
 */
}

/* VOID LOOP */
void loop() {
    // put your main code here, to run repeatedly:
    delay(1);
}

```

6. **Função QrCodeReader():** A função será utilizada pela task anteriormente mencionada. Com isso, ela efetuará a captura de imagens da câmera, processando os QR Codes em tempo real, exibindo os resultados.

```

// This function is to instruct the camera to take or
// capture a QR Code image, then it is processed and
// translated into text.
void QRCodeReader( void * pvParameters ){
/* -----
*/
Serial.println("QRCodeReader is ready.");
Serial.print("QRCodeReader running on core ");
Serial.println(xPortGetCoreID());
Serial.println();
/* ----- */

/* ----- Loop to read

```

```

QR Code in real time. */
while(1){
    q = quirc_new();
    if (q == NULL){
        Serial.print("can't create quirc object\r\n");
        continue;
    }

    fb = esp_camera_fb_get();
    if (!fb)
    {
        Serial.println("Camera capture failed");
        continue;
    }

    quirc_resize(q, fb->width, fb->height);
    image = quirc_begin(q, NULL, NULL);
    memcpy(image, fb->buf, fb->len);
    quirc_end(q);

    int count = quirc_count(q);
    if (count > 0) {
        quirc_extract(q, 0, &code);
        err = quirc_decode(&code, &data);

        if (err){
            Serial.println("Decoding FAILED");
            QRCodeResult = "Decoding FAILED";
        } else {
            Serial.printf("Decoding successful:\n");
            dumpData(&data);
        }
        Serial.println();
    }

    esp_camera_fb_return(fb);
    fb = NULL;
    image = NULL;
    quirc_destroy(q);
}

```

- 7. Função dumpData():** A função terá o papel de exibir informações acerca do QR Code detectado e realiza a requisição HTTP para a validação do código.

```

// Function to display the results of reading the QR Code
// on the serial monitor.
void dumpData(const struct quirc_data *data)
{
    Serial.printf("Version: %d\n", data->version);
    Serial.printf("ECC level: %c\n", "MLHQ"[data->ecc_level

```

```

    ]);

Serial.printf("Mask: %d\n", data->mask);
Serial.printf("Length: %d\n", data->payload_len);
Serial.printf("Payload: %s\n", data->payload);

HTTPClient http;
http.begin("https://ticketauthenticatorufmg-096d21475c99
            .herokuapp.com/validar_ingresso/" + String(
            reinterpret_cast<const char*>(data->payload)));

int httpCode = http.GET();
if (httpCode > 0) {
    if (httpCode == HTTP_CODE_OK) {
        String payload = http.getString();

        // Parse JSON usando ArduinoJson
        DynamicJsonDocument doc(1024); // Tamanho mximo
                                         esperado do JSON
        deserializeJson(doc, payload);

        // Agora voc pode acessar os dados do JSON
        const char* status = doc["status"];
        const char* mensagem = doc["mensagem"];

        Serial.println("Status: " + String(status));
        Serial.println("Mensagem: " + String(mensagem));

        controle_led(String(mensagem));
    } else {
        Serial.printf("[HTTP] Falha na requisio, cdigo de
                     erro: %d\n", httpCode);
    }
} else {
    Serial.printf("[HTTP] Falha na conexo\n");
}

http.end();
}

```

8. **Função controle\_led(String result):** A função controlará os Leds indicadores com base no resultado da validação do QR code.

```

void controle_led(String result)
{
    if (result == "Ingresso j utilizado") {
        Serial.println("Chegou aqui 1");
        digitalWrite(LED_RED, LOW);
        digitalWrite(LED_YELLOW, HIGH);
        digitalWrite(LED_LED, LOW);
        delay(10000);
    }
}

```

```

    } else if (result == "Ingresso válido") {
        Serial.println("Chegou aqui 2");
        digitalWrite(LED_RED, LOW);
        digitalWrite(LED_YELLOW, LOW);
        digitalWrite(LED_LED, HIGH);
        delay(10000);
    } else if (result == "Ingresso inválido") {
        Serial.println("Chegou aqui 3");
        digitalWrite(LED_RED, HIGH);
        digitalWrite(LED_YELLOW, LOW);
        digitalWrite(LED_LED, LOW);
        delay(10000);
    }
}

```

## 4 Conclusão

Acerca dos resultados obtidos, foi possível observar um funcionamento satisfatório para a proposta do trabalho, apesar de termos enfrentado dificuldades acerca do desenvolvimento. A respeito do funcionamento, foi possível observar a identificação correta dos QR Codes, tanto os válidos, quanto os inválidos e os já utilizados, executando a devida sinalizações nos LEDs. Com isso, o projeto se mostrou uma prova de conceito interessante para o contexto em que pode ser aplicado.

Apesar do resultado final do projeto ter se mostrado adequado, deve-se comparar o que foi proposto com o que foi entregue. A ideia inicial era implementar um sistema de leitura de códigos de barras em produtos de supermercado para agilidade e facilitação da tarefa de realizar compras. Entretanto, ler e identificar códigos de barras é uma tarefa complexa para componentes como a EspCam e o valor de um leitor de código de barras é elevado no mercado brasileiro. Portanto, foi feita uma pivotagem dessa ideia inicial para leitura de QR Codes em um contexto de entrada de eventos. A ideia era, além do sistema ler e identificar o QR Code, que o sistema tivesse um display para feedback em tempo real de seu status, com mensagens como "enviando QR Code para ser validado" ou "QR Code validado". O display poderia, inclusive, mostrar a imagem sendo capturada com a EspCam para facilitar a captura do QR Code. Entretanto, não foi possível conectar o display disponível com o microcontrolador, e isso foi contornado com o uso de LEDs vermelhos, amarelos e verdes para indicar se o ingresso é válido, inválido ou utilizado.

Em contraponto às dificuldades enfrentadas, o projeto inicial traz também a possibilidade de penitenciais melhorias visando a otimização do protótipo, podendo ser ainda readequado para utilização comercial. Nesse sentido, dentre as potenciais melhorias a serem listadas, podem ser citadas, a própria integração do display, a montagem de uma carcaça para comportar os componentes do projeto, além da procura de componentes mais baratos, visando uma potencial melhoria do ROI (Retorno sobre Investimento). Além disso, melhorias do software também podem ser feitas, visando a diminuição da latência na identificação do código, por exemplo.