OPTA-PROG-USBSTICK

PT-BR

Programação para salvar dados em um pendrive tipo C no OPTA

1. Informações Gerais

- Versão / Revisão: Revisão 1
- Data de Desenvolvimento: 05/02/2024
- Autor: Daniel Arcos
- Linguagem: C++
- Plataforma / Hardware: Finder OPTA

2. Objetivo da Programação

Esta programação tem a função de criar 3 pastas com um arquivo em cada uma, sendo tudo salvo em um pendrive conectado ao OPTA

3. Requisitos e Dependências

- Instalar o Arduino IDE em sua última versão disponível
- Instalar os drivers de Hardware do OPTA no Arduino IDE
- Instalar biblioteca do OPTA
- Instalar bibliotecas utilizadas na programação
- Pendrive USB-C

4. Passo a Passo da Implementação

Preparação do Hardware:

1. **Para armazenamento USB**:
    - Conecte um pen drive formatado em FAT32 à porta USB da placa
2. **Para armazenamento interno**:
    - Nenhum hardware adicional necessário

Configuração do Código:

1. Selecione o tipo de storage descomentando a linha apropriada:

```
//SDStorage storage;          // Para SD card
USBStorage storage;          // Para USB storage (selecionado)
//InternalStorage storage;    // Para armazenamento interno
```

1. Compile e faça upload do código para a placa
2. Abra o Serial Monitor configurado para 115200 baud

5. Código completo

Todo o arquivo da programação está disponível para download nos arquivos.

Visão Geral

Definições e Includes:

```
#define ARDUINO_UNIFIED_STORAGE_DEBUG  // Habilita mensagens de de
bug
```

```
#include "Arduino_UnifiedStorage.h"    // Biblioteca principal
```

Função printFolderContents:

```
void printFolderContents(Folder dir, int indentation = 0) {
  // Obtém lista de pastas e arquivos
  std::vector<Folder> directories = dir.getFolders();
  std::vector<UFile> files = dir.getFiles();

  // Imprime diretórios com recuo para hierarquia visual
  for (Folder subdir : directories) {
    for (int i = 0; i < indentation; i++) {
      Arduino_UnifiedStorage::testPrint("  "); // Indentação
    }
    Arduino_UnifiedStorage::testPrint("[D] "); // Indicador de diretório
    Arduino_UnifiedStorage::testPrint(subdir.getPath()); // Caminho
    printFolderContents(subdir, indentation + 1); // Recursão para subdiretóri
os
  }

  // Imprime arquivos
  for (UFile file : files) {
    for (int i = 0; i < indentation; i++) {
      Arduino_UnifiedStorage::testPrint("  "); // Indentação
    }
    Arduino_UnifiedStorage::testPrint("[F] "); // Indicador de arquivo
    Arduino_UnifiedStorage::testPrint(file.getPath()); // Caminho do arquivo
  }
}
```

Seleção do Tipo de Storage:

```
//SDStorage storage;            // Para armazenamento em cartão SD
USBStorage storage;           // Para armazenamento USB (selecionado)
//InternalStorage storage;      // Para armazenamento interno da placa
```

Função setup (Configuração Principal):

```
void setup() {
  beginRS485(115200);  // Inicializa comunicação serial (presumivelmente)

  // Habilita output de debug
  Arduino_UnifiedStorage::debuggingModeEnabled = true;

  // Inicializa o storage selecionado
  // storage = InternalStorage();
  // storage = SDStorage();
  storage = USBStorage(); // USB storage selecionado

  if(!storage.begin()){
    Arduino_UnifiedStorage::testPrint("Error mounting storage device.");
  }

  // Obtém diretório raiz do dispositivo
  Folder root = storage.getRootFolder();

  // Cria estrutura de diretórios
  Folder subdir1 = root.createSubfolder("subdir1");
  Folder subdir2 = root.createSubfolder("subdir2");
  Folder subdir3 = root.createSubfolder("subdir3");

  // Cria arquivos de texto nos diretórios
  UFile file1 = subdir1.createFile("file1.txt", FileMode::WRITE);
  UFile file2 = subdir2.createFile("file2.txt", FileMode::WRITE);
  UFile file3 = subdir3.createFile("file3.txt", FileMode::WRITE);

  // Escreve conteúdo nos arquivos
  file1.write("This is file 1.");
  file2.write("This is file 2.");
  file3.write("This is file 3.");

  // Prepara para leitura dos arquivos
  Arduino_UnifiedStorage::testPrint("Reading data from files using seek and
```

```
available:");

  // Altera modo dos arquivos para leitura
  file1.changeMode(FileMode::READ);
  file2.changeMode(FileMode::READ);
  file3.changeMode(FileMode::READ);

  // Lê e exibe conteúdo dos arquivos
  file1.seek(0); // Reposiciona ponteiro para início
  while (file1.available()) {
    char data = file1.read();
    Arduino_UnifiedStorage::testPrint(String(data)); // Caractere por caracter
e
  }
  Arduino_UnifiedStorage::testPrint("");

  file2.seek(0);
  while (file2.available()) {
    char data = file2.read();
    Arduino_UnifiedStorage::testPrint(String(data));
  }
  Arduino_UnifiedStorage::testPrint("");

  file3.seek(0);
  while (file3.available()) {
    char data = file3.read();
    Arduino_UnifiedStorage::testPrint(String(data));
  }
  Arduino_UnifiedStorage::testPrint("");

  // Exibe estrutura completa de diretórios
  printFolderContents(storage.getRootFolder());
}
```

Função loop (Loop Principal):

```
void loop() {
  // Pisca LED para indicar operação contínua
  digitalWrite(LEDG, !digitalRead(LEDG));
  delay(500);
}
```

5. Funcionamento do Sistema

Este código demonstra um sistema completo de gerenciamento de arquivos com:

Operações Principais:

1. **Inicialização de Storage**: Montagem do dispositivo selecionado
2. **Criação de Diretórios**: Estrutura hierárquica de pastas
3. **Criação de Arquivos**: Geração de arquivos de texto
4. **Escrita de Dados**: Armazenamento de conteúdo nos arquivos
5. **Leitura de Dados**: Recuperação e exibição do conteúdo
6. **Navegação de Arquivos**: Uso de seek() e available()
7. **Listagem de Diretórios**: Exibição da estrutura completa

Características Técnicas:

- **Interface Uniforme**: API consistente para diferentes tipos de storage
- **Modos de Acesso**: Suporte a diferentes modos de arquivo (leitura/escrita)
- **Manipulação de Ponteiro**: Controle de posição dentro do arquivo
- **Operações Recursivas**: Navegação em árvore de diretórios
- **Debug Integrado**: Sistema de logging para desenvolvimento

Fluxo de Execução:

1. Inicializa comunicação serial
2. Configura e monta o dispositivo de storage
3. Cria estrutura de diretórios (root → subdir1, subdir2, subdir3)

4. Cria arquivos em cada subdiretório
5. Escreve conteúdo específico em cada arquivo
6. Lê e exibe o conteúdo de cada arquivo
7. Lista a estrutura completa de diretórios
8. Entra em loop indicando operação com LED

Aplicações Práticas:

- Sistemas de logging de dados
- Armazenamento de configurações
- Coleta de dados em campo
- Backup de informações
- Sistemas de arquivos embarcados

Considerações Importantes:

1. **Formatação**: Dispositivos devem estar formatados em FAT32
2. **Capacidade**: Limites de tamanho dependem do hardware
3. **Performance**: Velocidade de acesso varia entre tipos de storage

---

EN

Programming to save data to a type C pendrive in OPTA

1. General Information

- Version / Revision: Revision 1
- Development Date: February 5, 2024

- Author: Daniel Arcos
- Language: C++
- Platform / Hardware: Finder OPTA

2. Programming Objective

This program creates three folders with one file in each, all of which are saved to a USB flash drive connected to the OPTA.

3. Requirements and Dependencies

- Install the latest version of the Arduino IDE
- Install the OPTA hardware drivers in the Arduino IDE
- Install the OPTA library
- Install libraries used in programming
- USB-C flash drive

4. Implementation Step-by-Step

Hardware Preparation:

1. **For USB storage**:

- Connect a FAT32-formatted flash drive to the board's USB port.

1. **For internal storage**:

- No additional hardware required.

Code Configuration:

1. Select the storage type by uncommenting the appropriate line:

```
//SDStorage storage;          // Para SD card
USBStorage storage;          // Para USB storage (selecionado)
```

```
//InternalStorage storage;    // Para armazenamento interno
```

1. Compile and upload the code to the board.
2. Open Serial Monitor and set it to 115200 baud.

5. Complete Code

The entire programming file is available for download in the archives.

Overview

Definitions and Includes:

```
#define ARDUINO_UNIFIED_STORAGE_DEBUG  // Habilita mensagens de de
bug
#include "Arduino_UnifiedStorage.h"    // Biblioteca principal
```

printFolderContents function:

```
void printFolderContents(Folder dir, int indentation = 0) {
  // Obtém lista de pastas e arquivos
  std::vector<Folder> directories = dir.getFolders();
  std::vector<UFile> files = dir.getFiles();

  // Imprime diretórios com recuo para hierarquia visual
  for (Folder subdir : directories) {
    for (int i = 0; i < indentation; i++) {
      Arduino_UnifiedStorage::testPrint("  "); // Indentação
    }
    Arduino_UnifiedStorage::testPrint("[D] "); // Indicador de diretório
    Arduino_UnifiedStorage::testPrint(subdir.getPath()); // Caminho
    printFolderContents(subdir, indentation + 1); // Recursão para subdiretóri
os
```

```
  }

  // Imprime arquivos
  for (UFile file : files) {
    for (int i = 0; i < indentation; i++) {
      Arduino_UnifiedStorage::testPrint("  "); // Indentação
    }
    Arduino_UnifiedStorage::testPrint("[F] "); // Indicador de arquivo
    Arduino_UnifiedStorage::testPrint(file.getPath()); // Caminho do arquivo
  }
}
```

Storage Type Selection:

```
//SDStorage storage;            // Para armazenamento em cartão SD
USBStorage storage;            // Para armazenamento USB (selecionado)
//InternalStorage storage;      // Para armazenamento interno da placa
```

Setup function (Main Configuration):

```
void setup() {
  beginRS485(115200);  // Inicializa comunicação serial (presumivelmente)

  // Habilita output de debug
  Arduino_UnifiedStorage::debuggingModeEnabled = true;

  // Inicializa o storage selecionado
  // storage = InternalStorage();
  // storage = SDStorage();
  storage = USBStorage(); // USB storage selecionado

  if(!storage.begin()){
    Arduino_UnifiedStorage::testPrint("Error mounting storage device.");
  }

  // Obtém diretório raiz do dispositivo
```

```
  Folder root = storage.getRootFolder();

  // Cria estrutura de diretórios
  Folder subdir1 = root.createSubfolder("subdir1");
  Folder subdir2 = root.createSubfolder("subdir2");
  Folder subdir3 = root.createSubfolder("subdir3");

  // Cria arquivos de texto nos diretórios
  UFile file1 = subdir1.createFile("file1.txt", FileMode::WRITE);
  UFile file2 = subdir2.createFile("file2.txt", FileMode::WRITE);
  UFile file3 = subdir3.createFile("file3.txt", FileMode::WRITE);

  // Escreve conteúdo nos arquivos
  file1.write("This is file 1.");
  file2.write("This is file 2.");
  file3.write("This is file 3.");

  // Prepara para leitura dos arquivos
  Arduino_UnifiedStorage::testPrint("Reading data from files using seek and
available:");

  // Altera modo dos arquivos para leitura
  file1.changeMode(FileMode::READ);
  file2.changeMode(FileMode::READ);
  file3.changeMode(FileMode::READ);

  // Lê e exibe conteúdo dos arquivos
  file1.seek(0); // Reposiciona ponteiro para início
  while (file1.available()) {
    char data = file1.read();
    Arduino_UnifiedStorage::testPrint(String(data)); // Caractere por caracter
e
  }
  Arduino_UnifiedStorage::testPrint("");

  file2.seek(0);
  while (file2.available()) {
    char data = file2.read();
```

```
    Arduino_UnifiedStorage::testPrint(String(data));
  }
  Arduino_UnifiedStorage::testPrint("");

  file3.seek(0);
  while (file3.available()) {
    char data = file3.read();
    Arduino_UnifiedStorage::testPrint(String(data));
  }
  Arduino_UnifiedStorage::testPrint("");

  // Exibe estrutura completa de diretórios
  printFolderContents(storage.getRootFolder());
}
```

Loop function (Main Loop):

```
void loop() {
  // Pisca LED para indicar operação contínua
  digitalWrite(LEDG, !digitalRead(LEDG));
  delay(500);
}
```

5. System Operation

This code demonstrates a complete file management system with:

Main Operations:

1. **Storage Initialization**: Mounting the selected device
2. **Directory Creation**: Hierarchical folder structure
3. **File Creation**: Generating text files
4. **Data Writing**: Storing content in files
5. **Data Reading**: Retrieving and displaying content

6. **File Navigation**: Using seek() and available()
7. **Directory Listing**: Displaying the complete structure

Technical Features:

- **Uniform Interface**: Consistent API for different storage types
- **Access Modes**: Support for different file modes (read/write)
- **Pointer Manipulation**: Controlling position within the file
- **Recursive Operations**: Tree navigation Directory Management
- **Integrated Debug**: Logging system for development

Execution Flow:

1. Initializes serial communication
2. Configures and mounts the storage device
3. Creates directory structure (root → subdir1, subdir2, subdir3)
4. Creates files in each subdirectory
5. Writes specific content to each file
6. Reads and displays the contents of each file
7. Lists the complete directory structure
8. Enters a loop indicating operation with an LED

Practical Applications:

- Data logging systems
- Configuration storage
- Field data collection
- Information backup
- Embedded file systems

Important Considerations:

1. **Formatting**: Devices must be formatted in FAT32
2. **Capacity**: Size limits depend on the hardware

3. **Performance**: Access speed varies between storage types