MicroObex: Uma biblioteca para a transferência de arquivos via Bluetooth em dispositivos embarcados

Lestaf Santiago Pereira Pimentel[1], **Roger L. Hoff Lavarda**[1]

[1]Instituto Federal de Ciência e Tecnologia do Rio Grande do Sul - Câmpus Ibirubá
Rua Nelsi Ribas Fritsch, 1111 – CEP: 98200-000 – Ibirubá – RS – Brasil

Abstract. *The Arduino, The ESP32 and other microcontroller platforms have a host of APIs and functions that facilitate Bluetooth communication, however to send a file using such methods requires a custom solution, which might prove too challenging to inexperienced programmers and hobbyists. Bluetooth file transfers could easily be done using the ObEx profile, however, the profile isn't present in the most popular microcontrollers' bluetooth stacks. In this course completion work, a light, easy to use library written in C that implements the ObEx profile was developed to facilitate file transfers between devices. The library contains several functions to perform connections, send files and parse information from ObEx packets following the structures and definitions imposed by the ObEx profile.*

Resumo. *A Arduino, a ESP32 e outras plataformas de microcontroladores possuem um leque de APIs e funções que facilitam a comunicação Bluetooth, no entanto para enviar um arquivo usando tais métodos requer uma solução customizada, o que pode ser demasiadamente complexo para programadores inexperientes. transferências de arquivos via Bluetooth podem ser realizadas facilmente utilizando o perfil ObEx, porem o perfil não está presente nos stacks Bluetooth dos microcontroladores mais populares. Neste trabalho de conclusão de curso, foi desenvolvida uma biblioteca leve e de fácil utilização que implementa o perfil ObEx, com o objetivo de facilitar transferências de arquivos entre dispositivos. A biblioteca contém várias funções para realizar conexões, enviar arquivos e interpretar informações de pacotes ObEx, seguindo as estruturas e definições impostas pelo perfil ObEx.*

1. Introdução

Devido à disponibilidade e a acessibilidade de microcontroladores voltados à prototipagem e uso em pequena escala, como os produtos da plataforma Arduíno e Raspberry, usuários os têm utilizado para as mais diversas aplicações, como para a automação domiciliar, *Dataloggers*, servidores de armazenamento, processamento de dados, entre outros. Muitas dessas aplicações necessitam realizar a transferência de arquivos entre dispositivos, como entre Arduinos e dispositivos celulares. Para tal, desenvolvedores podem optar por utilizar servidores de uma rede wireless ou cabeada devido à facilidade de implementação e à grande variedade de bibliotecas e perfis disponíveis para tal, como um servidor FTP[1].

[1]FTP - Sigla para File Transfer Protocol, protocolo de comunicação frequentemente utilizado sobre TCP/IP

Uma alternativa para a transferência de arquivos seria via Bluetooth, que é menos utilizado para esses propósitos, e mais para sinalização ou transferência de valores individuais devido à abundância de bibliotecas e implementações para estes fins. Outro motivo pelo qual não é tão amplamente utilizado para o envio e recebimento de arquivos é a falta de implementações e bibliotecas que o facilitem.

As opções disponíveis para a comunicação de curta distância entre microcontroladores para uso amador focam em transmissões curtas de dados desorganizados. Para realizar a transferência de arquivos em uma rede local, a tecnologia Bluetooth seria ideal, porém não existem métodos em microcontroladores que o facilitem. Para a realização de tal transferência, seria necessário criar uma aplicação própria que interpreta quando um arquivo estivesse sendo transferido, lê os dados recebidos e os salva. Dependendo da experiência do programador, tal método pode ser propenso a erros e falhas de comunicação.

Outro método disponível é a utilização de WiFi e internet, que necessita de um roteador que possa retransmitir os dados dos microcontroladores. Este método não é viável em redes espalhadas devido a limitações de alcance de tais dispositivos, assim como em locais remotos, que torna vital o baixo consumo de energia, como no trabalho de Spinelli e Gottesman (2019), em que foi utilizado hardware adicional que acarretou em um custo desnecessário para a compra de módulos de rede celular e para a compra dos chips para a conexão.

Segundo Ericsson (2021), em 2020, o número de dispositivos em aplicações *IoT* chegou a até 12,4 bilhões de unidades. Com esse crescimento, veio também a simplificação do processo de prototipagem, como por exemplo, na plataforma Arduino para usuários menos experientes. No entanto, algumas áreas do conhecimento ainda se demonstram de difícil acesso devido à falta de documentação, ferramentas e recursos educacionais, como a transferência.

Com o presente trabalho, foi desenvolvida e implementada uma biblioteca escrita na linguagem C que permite a transferência de arquivos entre dispositivos via Bluetooth em microcontroladores ESP32, utilizando os perfis, que são os protocolos de alto nível descritos na especificação Bluetooth, necessários para tal. A biblioteca oferece métodos simples e flexíveis para a formação de pacotes de transferência utilizando Bluetooth, que também podem ser adaptados para Infravermelho.

Este artigo está organizado da seguinte forma: na Seção 2 está delineado o problema; a Seção 3 apresenta o objetivo da biblioteca; A Seção 4 explana a revisão da literatura, entrando em detalhes sobre sistemas embarcados, comunicação, Bluetooth e seus protocolos; a Seção 5 discute os principais trabalhos correlatos a temática; a Seção 6 descreve a metodologia com a qual a biblioteca foi desenvolvida; a Seção 7 entra em detalhes sobre o desenvolvimento da biblioteca; a Seção 8 discute os resultados e limitações da biblioteca; e por fim, a Seção 9 apresenta considerações finais e trabalhos futuros.

2. Problema

Atualmente, a implementação de transferência de arquivos utilizando tecnologias de rede móveis necessita de roteadores e modems para permitir a comunicação. No trabalho de Spinelli e Gottesman (2019), o equipamento necessário para permitir a comunicação do dispositivo com o servidor acarretaria num custo de U$ 53,00 por dispositivo.

Muitas das bibliotecas disponíveis para a comunicação Bluetooth em microcontroladores são focadas em fluxos de dados não estruturados, como leituras de sensores e sinalizações. Bibliotecas como BluetoothSerial[2] e BTStack[3] tem como grande foco perfis como Serial Port Profile (SPP), que emula uma conexão serial com outro dispositivo.

Os perfis no contexto Bluetooth são especificações definidas pelo Grupo de Interesse Especial Bluetooth, que definem a interação entre os componentes do sistema Bluetooth. Além disso, também definindo formatos de dados e os comportamentos da aplicação quanto à comunicação entre dispositivos.

A comunicação serial como a implementada pelo Serial Port Profile, busca emular uma conexão serial como a do padrão RS-232, porém não é ideal para a transferência de arquivos, uma vez que se limita apenas a enviar e receber dados binários não estruturados e caberia ao dispositivo a interpretação destes dados. O perfil também não possui checagem de erros, o que pode deixar arquivos inacabados ou ilegíveis.

Para a comunicação com dispositivos celulares seria necessário um software adicional, como um terminal serial para a leitura e envio de sinais, pois os principais sistemas operacionais do mercado não possuem soluções nativas para a comunicação direta utilizando o perfil SPP. A solução seria utilizar o perfil ObEx, desenvolvido pela Associação de dados Infravermelho (IrDA), e foi incorporado à especificação Bluetooth e atualmente é o método padrão para a transferência de arquivos.

Segundo Deccio et al. (2003) o perfil Object Exchange (ObEx), é capaz de transferir, com segurança, grandes quantidades de dados, com alta velocidade, e não exige a instalação de uma nova aplicação em dispositivos celulares, pois o sistema operacional dos mesmos já possui implementações para o recebimento de tais arquivos. Uma implementação do perfil ObEx já foi desenvolvida por ZhangCheng e Shunxiang (2009), para a plataforma Windows Mobile, descontinuada em 2010, e sua implementação não é compatível com microcontroladores modernos.

3. Objetivo Geral

É objetivo do presente trabalho, o desenvolvimento de uma biblioteca que implemente os perfis necessários para a transferência de arquivos via bluetooth entre dispositivos que seja compatível com microcontroladores disponíveis atualmente.

3.1. Objetivos Específicos

- Desenvolver uma biblioteca compatível com os microcontroladores das famílias Arduino e ESP32
- Avaliar a confiabilidade e velocidade da biblioteca quanto ao envio de arquivos utilizando a tecnologia Bluetooth;
- Comparar a performance do perfil ObEx em relação a outras soluções;
- Criar um experimento com Arduino ou ESP32 e um aplicativo Android para estudo de caso de transferência de dados.

[2]BluetoothSerial - https://github.com/espressif/arduino-esp32/tree/master/libraries/BluetoothSerial
[3]BTStack - https://github.com/bluekitchen/btstack

4. Revisão da Literatura

Nesta seção está descrita a literatura utilizada para a realização do projeto, incluindo conceitos sobre o hardware utilizado e protocolos de software.

4.1. Sistemas embarcados

Segundo Heath (2002), um sistema embarcado é um sistema baseado em um microprocessador que é construído para exercer uma ou várias funções definidas dentro de um sistema eletromecânico, e não é projetado para ser reprogramado pelo usuário final, no sentido de que o usuário pode escolher entre as funcionalidades dispostas pelo sistema, mas não é possível alterar sua funcionalidade.

Sistemas embarcados atualmente compartilham certas semelhanças entre si, normalmente possuem dimensões reduzidas, permitindo o uso em sistemas maiores. Diferentes critérios podem influenciar a escolha de um microcontrolador, pois cada tipo atende a uma necessidade diferente, como apresentado no trabalho de Parai, Das e Das (2013), uma delas sendo a reatividade em tempo real, que é um fator de suma importância para a utilização de tais sistemas, como por exemplo, nos sistemas que regem a aceleração e freio em carros, que devem reagir à entrada do usuário e processar o sinal sem atrasos como apresentado no trabalho de Goud et al. (2006). Para obter essa operação em tempo real, utiliza-se um microcontrolador na maioria dos casos, mas também são utilizados FPGAs, *Field Programmable Grid Array*, ASICs, *Application Specific Integrated Circuits*, e microprocessadores. Para permitir sua interação com o sistema em que está inserido, normalmente essa interação é feita por sensores e atuadores e dispositivos de interação.

Segundo dados de Odunlade (2020), dois dos principais microcontroladores utilizados em sistemas embarcados modernos são o ESP32 e o ATmega, ambos são microcontroladores de baixo custo, porém, a principal utilização dos chips da série ATmega é na plataforma Arduino, que produz software e placas de prototipação baseadas no microcontrolador. A plataforma é conhecida por sua ampla acessibilidade, que, devido à sua interface de programação por USB, baixa demanda energética e API simplificada, a torna uma ótima ferramenta educacional, exemplificado no trabalho de Perenc, Jaworski e Duch (2019) ou como ferramenta para artistas e designers que não possuem experiência na área eletrônica.

O Microcontrolador ESP32 é um chip desenvolvido pela Espressif, o principal diferencial apresentado é sua conectividade, possuindo hardware para conexões Wi-Fi e Bluetooth sem necessitar equipamentos adicionais, facilitando o controle remoto de aplicações, como apresentado no trabalho de Pravalika e Prasad (2019) o que faz do produto uma opção atraente tanto para uso amador quanto em um contexto profissional. O microcontrolador possui capacidade de operação em tempo real através de sua implementação do FreeRTOS, o sistema operacional em tempo real utilizado no ESP32. Diferente do Arduino, o ESP32 é voltado para usuários mais experientes, embora seja possível utilizá-lo com a API Arduino, é recomendável utilizar o próprio framework Espressif, que requer certo conhecimento técnico, porém providencia um controle mais preciso sobre o dispositivo.

4.2. Comunicação entre dispositivos embarcados

No contexto da Internet das Coisas (IoT) atual, é essencial que dispositivos possuam capacidades de comunicação com outros dispositivos, seja para o simples envio de

dados para um servidor remoto ou em uma rede de dispositivos interligados. Existem várias soluções para efetivar a comunicação entre dispositivos, seja por fios e cabos, ou seja sem fio, utilizando frequências de rádio, micro-ondas e infravermelho. Para assegurar a compatibilidade da comunicação entre dispositivos é necessário a adoção de padrões e protocolos; assim, os dispositivos podem ser programados para receber, processar e responder corretamente aos diferentes tipos de dados que possa vir a receber ao longo da aplicação.

Na comunicação serial via cabos, é comum o uso do protocolo I2C[4], que utiliza uma linha bi-direcional para a transferência de dados e uma linha para a definição do clock da comunicação. O protocolo busca simplicidade e baixo custo, portanto é pouco apropriado para sistemas onde a velocidade de comunicação é uma prioridade. Também é comum o uso de Ethernet, que, comunicando-se com a internet, é capaz de providenciar alcance ilimitado com velocidades elevadas. Sua comunicação se dá por meio do protocolo TCP, cujos dados podem ser enviados livremente, ou estruturados utilizando protocolos como HTTP, que é baseado em requisições realizadas pelos dispositivos. Há também a opção de utilizar o protocolo FTP em conjunto com TCP, como no trabalho apresentado por Spinelli e Gottesman (2019), o protocolo é voltado à navegação e transferência de arquivos em um servidor, sendo pouco indicado para a comunicação de sinais e valores individuais devido a alta latência do protocolo.

Para conexões sem fio, há várias soluções disponíveis, uma das mais conhecidas é o Wi-Fi, que se comunica por meio de ondas de rádio, conectando-se à internet ou a uma rede local de forma semelhante à uma conexão ethernet, podendo utilizar os mesmos protocolos. Algumas das outras opções disponíveis aos usuários são Zigbee e Bluetooth. Zigbee é uma tecnologia semelhante ao Wi-Fi, possui uma baixa velocidade de transmissão com criptografia de dados, suas redes suportam até 65 mil dispositivos conectados que, devido ao seu baixo uso de energia, é mais adequado para aplicações de *IoT*, nas quais muitas vezes é necessário várias aplicações de baixo consumo enviando transmissões pequenas e intermitentes a outros dispositivos.

4.3. Bluetooth

De acordo com Muller (2003), Bluetooth é uma tecnologia de Rede de Área Pessoal Sem fio que permite a comunicação entre dispositivos a curtas distâncias. Seu baixo custo de implementação e baixo consumo de energia fazem da tecnologia uma ótima substituição para cabos convencionais, sendo assim uma opção de comunicação viável para aplicações de *IoT*, fones sem fio, serviços de localização, *wearables*, entre outros. A tecnologia permite a conexão de até 8 dispositivos em uma formação denominada *piconet*.

Para a interoperabilidade entre aplicações, a tecnologia adota uma série de especificações chamadas de perfis. Segundo Woolley (2019) os perfis definem as funções do sistema Bluetooth e também a estrutura da comunicação entre dispositivos. Quando os dispositivos envolvidos na comunicação implementam os requisitos do perfil, torna-se possível a interação entre os mesmos.

Alguns exemplos de especificações adotadas pelo grupo Bluetooth incluem o SPP, que, assim como outras, utiliza a camada de RFCOMM do protocolo bluetooth, que foi

[4]I2C - Circuito inter-integrado

desenvolvido para emular conexões seriais a cabo entre os dispositivos envolvidos, portanto a estrutura dos dados enviados é simples e mais adequada para a transferência intermitente de dados desestruturados. Outra especificação adotado pelo grupo é a IrObEx, originalmente desenvolvido pela *Infrared Data Association*, adaptada para bluetooth apenas como ObEx, que foi desenvolvida para a troca de objetos entre dispositivos, como arquivos ou cartões de visita. A Especificação possui uma estrutura semelhante ao protocolo HTTP, sendo baseado em requisições, com cabeçalhos contendo o método da requisição, seu tamanho, id da conexão e o nome do arquivo a ser transferido, seguido então pelo corpo contendo os dados a serem enviados.

4.3.1. Perfis e protocolos

Nesta seção são destacados os protocolos, perfis e especificações relevantes para o presente trabalho, com uma breve descrição da função de cada item.

Logical Link control and Adaptation Protocol(L2CAP), Gerencia serviços de dados às camadas superiores do *stack* utilizado e é o protocolo base da maioria dos perfis presentes na especificação Bluetooth. O L2CAP providencia também a multiplexação de dados com diferentes conexões, utilizando canais de comunicação. Implementa também um controle de fluxo de mensagens baseado em uma máquina de estado, assim como um sistema de verificação e recuperação de erros.

O *Service Discovery Protocol* (SDP), rege o anúncio e descoberta de serviços oferecidos por dispositivos. podendo realizar uma busca por todos os serviços oferecidos ou algum em específico, identificado por um número de 128-bits, retornando um pacote contendo as informações dos serviços requisitados. As informações disponíveis variam de serviço à serviço, e outros serviços podem ser adicionados por um programador com o devido conhecimento do protocolo.

Object Exchange Profile, é o perfil que especifica os métodos pra a transferência de objetos entre dispositivos, assim como a estrutura de pacotes para tal. Funciona de maneira similar ao HTTP, com base em códigos de operação, com informações contidas no cabeçalho da requisição, dependendo de um pacote de resposta para continuar a comunicação.

A especificação*Generic Object Exchange Profile* (GOEP), determina quais os métodos ObEx deverão ser usados pelas aplicações e futuros perfis que utilizem alguma forma de transferência de objetos. Especifica também quais características do protocolo deverão ser usados e como devem interagir com as camadas L2CAP e SDP.

Object Push Profile (OPP), se refere à especificação que define como apresenta-se a funcionalidade padrão para o envio e recebimento de objetos, definindo constantes a serem utilizadas e operações mandatórias e opcionais do protocolo, seguindo as definições da especificação GOEP.

5. Trabalhos Correlatos

Nesta seção estão presentes alguns trabalhos de diversos autores que se assemelham de alguma maneira com a biblioteca.

Em Ni e Liu (2010), os autores desenvolveram um método para a transferência de imagens utilizando o perfil ObEx, no entanto o trabalho se limitava à utilização do chip S3C2410, com uma implementação Linux embarcada. Os autores também utilizam a plataforma BlueZ para controlar o Bluetooth, portanto sua implementação não seria compatível com microcontroladores como o ESP32, que utiliza a plataforma BlueDroid.

No trabalho de Spinelli e Gottesman (2019), os autores buscaram criar alternativas de baixo custo para um medidor de fluxo e uma estação de monitoramento que possibilitasse o uso do software *CropManage*, para gerenciar a irrigação de diferentes campos. No trabalho, optaram por utilizar um módulo GSM da AdaFruit para conectar os microcontroladores à estação de monitoramento, a qual iria extrair os dados dos mesmos utilizando o protocolo FTP.

Na questão de bibliotecas que facilitem o desenvolvimento, Lee, Kim e Jeong (2014) que, devido à crescente utilização da Internet das Coisas e à relativa complexidade da API Bluetooth do sistema Android, buscaram desenvolver uma biblioteca para auxiliar o uso das funções Bluetooth disponíveis na API. Os autores então desenvolveram um simples aplicativo de teste utilizando sua biblioteca que se comunica de maneira serial com um computador. Utilizando apenas três classes, e com apenas 15 métodos, a biblioteca MoscaBT simplificou significativamente o desenvolvimento de aplicações para a plataforma, cuja API Bluetooth requer 61 métodos para executar as mesmas funções.

Portanto, com o desenvolvimento da biblioteca proposta neste trabalho, pretende-se facilitar o desenvolvimento de aplicações que demandam o envio e coleta de arquivos, reduzindo as chamadas necessárias para se executar tal tarefa. A abstração providenciada pela biblioteca proposta permitirá que usuários inexperientes e sem o total conhecimento dos protocolos necessários possam desenvolver aplicações eficientes e de baixo custo.

6. Metodologia

Inicialmente foi feito uma pesquisa quanto aos perfis e protocolos que possibilitam a comunicação entre dispositivos, como o perfil Obex em si, GOEP, OPP e outros. Em seguida foram desenvolvidos métodos para a transferência de arquivos entre os mesmos utilizando os perfis descritos. No próximo passo foram feitos testes e comparações dos métodos desenvolvidos com soluções já existentes, comparando a transferência de um arquivo de um celular para um computador e a transferência entre um ESP32 e um computador pessoal. Por fim foi desenvolvida uma biblioteca implementando tais métodos.

Durante os testes foi utilizado um método nativo para o envio de arquivos, para a comparação com a biblioteca. Na sequência, foi realizada a comparação entre métodos com relação à velocidade e a confiabilidade na transferência de arquivos de diversos tamanhos.

6.1. Estrutura da biblioteca

Como ilustrado pela Figura 1, a comunicação realizada pela biblioteca entre um dispositivo e outro se dá pela chamada de uma função, à qual a biblioteca enviará um pacote de requisição ao dispositivo pareado, que enviará um pacote de resposta, que será processado pela biblioteca que retornará um objeto ao dispositivo.

A Figura 2 representa o diagrama de sequência para o recebimento de um arquivo. O usuário inicia a comunicação chamando a função "*Connect*", a biblioteca então envia

para o dispositivo pareado uma requisição do tipo *CONNECT*, ao qual o dispositivo envia um pacote com um código de resposta contendo informações da conexão. Ao receber o código de resposta, a biblioteca envia uma requisição PUT, enviando os dados do arquivo, o dispositivo então envia uma resposta com o resultado da operação e requisitando o resto do arquivo. Um último pacote PUT é enviado com o final do arquivo, recebendo uma ultima resposta e retornando-a ao usuário. Caso não haja mais arquivos a serem enviados, o usuário chama a função Disconnect para realizar a desconexão com o dispositivo.

Figura 1. Funcionamento da biblioteca.

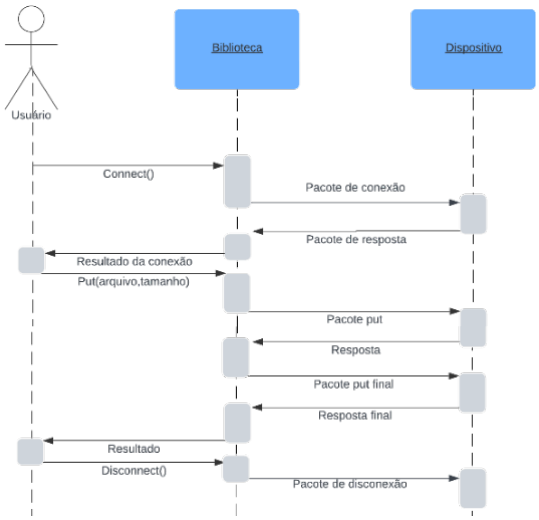

Figura 2. Diagrama de sequência da função put()

7. Desenvolvimento da MicroObex

A biblioteca foi escrita em C com personalização em mente, de modo que torna fácil a adaptação do código para diferentes dispositivos, para tal objetivo o desenvolvimento foi dividido em três arquivos: (I) o cabeçalho, que contém as constantes e estruturas necessárias para o perfil; (II) o código principal, que contém as funções necessárias para o cumprimento do perfil GOEP; (III) e o código auxiliar opcional, contendo funções para tornar ainda mais fácil o desenvolvimento. Ela opera com pacotes e cabeçalhos. Um pacote é uma unidade de informação a ser transferida ou recebida por um dispositivo.

No contexto Bluetooth, um pacote contém bytes carregando informações da qualidade da conexão, para qual endereço o pacote será enviado, de qual endereço veio. Assim como informações de protocolos associados, como o L2CAP, que gerencia múltiplas conexões e o SDP, que busca e interpreta informações de dispositivos.

Um cabeçalho, no contexto ObEx, é uma unidade de informação que pode conter diversas funções, como informar a quantidade de arquivos a serem transferidos, um ID para a multiplexação de conexões, assim como o corpo do arquivo em transferência.

7.1. Pacote de conexão

Inicialmente, foi desenvolvido o método para a criação do pacote de conexão. O mais simples comparado aos outros tipos de pacote ObEx, na maioria dos casos ocupando apenas 12 bytes que transmitem a informação da versão do protocolo, o tamanho máximo do pacote e a quantia de objetos que se deseja transferir. É utilizado para dar início e gerenciar uma conexão. Os dados são copiados byte a byte para um ponteiro providenciado à função pelo usuário após a função alocar corretamente a memória para o pacote. O dispositivo de destino deve enviar um pacote de resposta contendo suas informações e um ID de conexão, utilizado para gerenciar operações simultâneas.

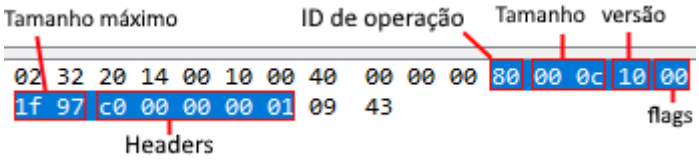

Figura 3. O texto selecionado representa a seção ObEx do pacote

7.2. Pacote de envio

O pacote de envio, PUT, contém os dados do arquivo em si assim como o ID da conexão referente a transação como cabeçalho do pacote. Caso o pacote não comporte todo o arquivo, múltiplos pacotes PUT podem ser enviados, aguardando uma resposta do servidor antes do próximo envio. Também pode ser enviado como cabeçalho uma requisição para habilitar o Modo de Resposta Única, que, para agilizar o envio de arquivos maiores, dispensa respostas do servidor.

Inicialmente planejava-se em fazer com que a função recebesse um descritor de arquivo e então lesse os dados conforme fosse realizado o envio, porém, como existem métodos alternativos para a utilização do perfil, optou-se por deixar os próprios dados a serem enviados como argumento para a função e foi feito uma função auxiliar para realizar a transferência de maneira direta. Ademais planejava-se utilizar um *Struct* para manter os dados da sessão, mas devido a problemas em algumas arquiteturas as informações devem ser enviadas manualmente à função.

7.3. Funções auxiliares

Durante o desenvolvimento da biblioteca, tornou-se evidente que funções adicionais deveriam ser desenvolvidas não só para auxiliar no desenvolvimento da biblioteca, mas também facilitar a sua utilização por usuários que não possuam boa compreensão dos protocolos envolvidos.

Uma das funções é para obter o tamanho de um cabeçalho de um pacote, como o perfil possui três tipos de cabeçalhos diferentes, devido às diferentes necessidades de cada tipo de dado. Um dos tipos de cabeçalho possui apenas um byte de ID e quatro de argumento, outro possui um byte de ID, dois bytes especificando o tamanho total do

cabeçalho e o resto de argumento. O último possui apenas dois bytes no total, para ID e argumento cada. Para identificar o tipo é necessário comparar os dois primeiros bits do ID, e com base nisso a função retorna o tamanho correto.

Outra função auxiliar é para inicializar um cabeçalho, recebendo o ID, um ponteiro para o argumento e um ponteiro para o tamanho, que pode ser 0 caso seja um cabeçalho de tamanho fixo. Após realizar a criação do cabeçalho, a função atualiza o ponteiro de tamanho e retorna um ponteiro para o cabeçalho.

Após alguns testes, observou-se que alguns valores eram recebidos em ordem reversa, após uma revisão da documentação, descobriu-se que o erro se dava ao fato da maioria dos dispositivos utilizarem *Least Significant Byte* (LSB), onde os bytes de um valor composto de múltiplos bytes são dispostos em ordem ascendente, enquanto Bluetooth e outras tecnologias de comunicação utilizam *Most Significant Byte* (MSB), onde os bytes são ordenados de maneira descendente, então foi desenvolvida uma função que inverte a ordem dos bytes enviados ao dispositivo.

Com a maioria dos problemas resolvidos, ainda assim se observou que os nomes de arquivos e seu *MIME TYPE*[5], eram interpretados erroneamente pelo dispositivo. Segundo a documentação ObEx o erro é devido a alguns cabeçalhos serem interpretados como uma *string* Unicode, de dois bytes, pelo dispositivo, enquanto a biblioteca os enviava como uma string ASCII, então foi desenvolvida uma função que transforma quaisquer dados em Unicode. Como o caractere Nulo em C é o prefixo Unicode para Latim, necessitou-se também da criação de uma função que interpretasse corretamente o tamanho de uma string em Unicode, levando em conta os dois bytes que compõem um único caractere.

7.4. Testes

Os testes foram estruturados utilizando um ESP32 enviando um arquivo a um OrangePi 3 LTS, verificando velocidade e confiabilidade da transferência. Durante os primeiros testes, o arquivo tratava-se de um simples arquivo de texto, contendo apenas 30 bytes. Testes subsequentes foram realizados com diferentes tipos de arquivos com diferentes tamanhos.

Inicialmente, houveram problemas para o envio de informações ao dispositivo devido a documentação precária das APIs Bluetooth do ESP32. Após resolvido, houveram problemas relacionados à conexão SDP com o dispositivo, portanto optou-se por realizar os testes com um código de exemplo providenciado pelo fabricante, adaptado para utilizar a biblioteca.

Durante os testes de envio tornou-se evidente que seria necessário o uso do cabeçalho contendo o ID de conexão, algo que considerava-se opcional, porém em alguns casos, o pacote contendo-o é recebido de forma errônea, trocando os bytes de *checksum* l2cap com o que deveria conter o ID de conexão, então o código foi adaptado para tais casos, caso o programa detecte que o cabeçalho esteja incorreto, altera a posição da qual extrai os bytes corretos.

Após solucionados os problemas, os pacotes foram enviados corretamente ao destinatário, o qual retornou pacotes de resposta indicando o sucesso da operação.

[5]MIME TYPE - Utilizado para distinguir o tipo de arquivo a ser enviado

Alguns dos cenários de testes propostos não foram realizados por serem irrelevantes ao contexto da biblioteca. Julgou-se desnecessário a comparação com o envio de arquivos via SPP, como o perfil não foi desenvolvido para este propósito e a velocidade e confiabilidade de tal método depende da implementação.

8. Resultados e limitações

A biblioteca comporta-se de maneira satisfatória, cumprindo seu propósito. Como se observa na Figura 4, o pacote de envio produzido pela biblioteca é virtualmente idêntico ao produzido por um dispositivo utilizando BlueDroid[6]. Nele se observa a formação do pacote contendo corretamente o tamanho do pacote, a versão do protocolo, o tamanho máximo suportado e o cabeçalho indicando a quantidade de objetos a serem transferidos. Por fim, a resposta é enviada pelo dispositivo contendo um cabeçalho com a Id de conexão, indicando a correta identificação do pacote e protocolo, como observado na Figura 5. ,

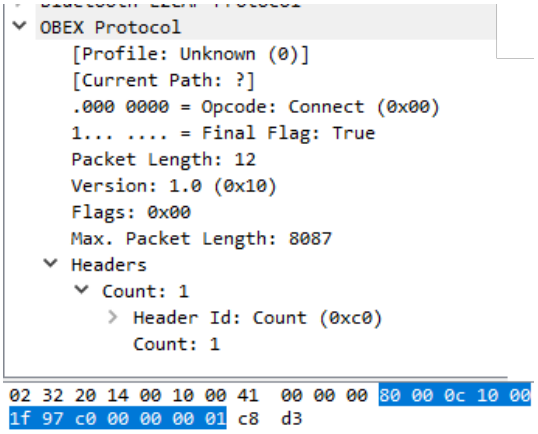

Figura 4. Correta formação do pacote

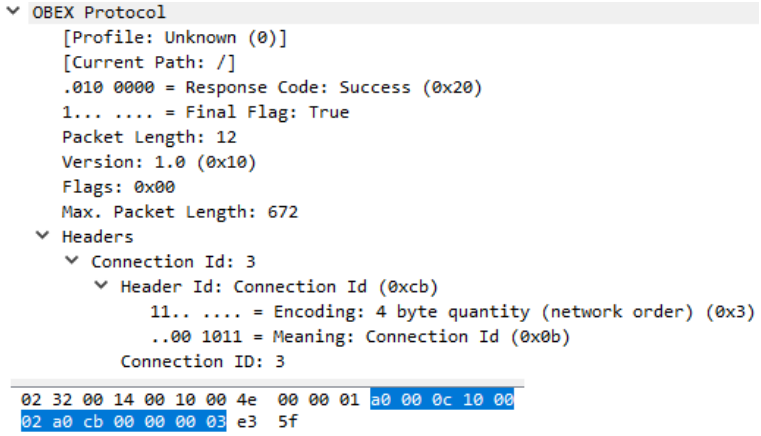

Figura 5. Resposta do dispositivo

Então o pacote subsequente contendo as informações do arquivo é enviado, contendo o Identificador de conexão, o nome do arquivo, tipo e conteúdo do arquivo, como visto na Figura 6. A resposta do dispositivo é recebida, indicando que foi reconhecido e requisitando o Modo de Resposta Única, como observado na Figura 7.

A biblioteca envia o pacote PUT final, indicando o fim do arquivo. O cabeçalho de final de arquivo possui valor opcional, portanto o valor não foi incluído na Figura 8. Então é recebido a resposta indicando o sucesso da transferência, visto na Figura 9.

Por fim, a biblioteca envia um pacote de desconexão, requisitando o fim da comunicação e o dispositivo responde com um pacote indicando o sucesso da operação

Com isso, observa-se que a biblioteca cria de maneira correta os pacotes e os envia ao dispositivo, que corretamente os interpreta e envia sua resposta ao usuário.

Devido à limitações de tempo e de equipamentos disponíveis, não foi possível realizar a adaptação e testes da biblioteca para a família de microcontroladores Arduino,

[6]Bluedroid - Stack bluetooth utilizado em sistemas Android

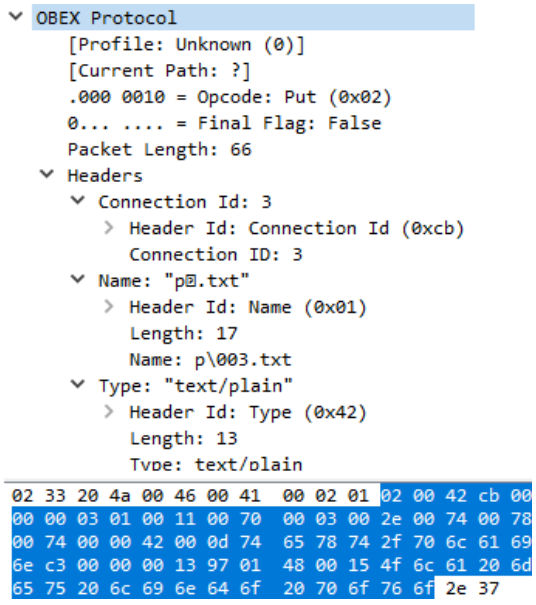

Figura 6. Pacote Put contendo arquivo

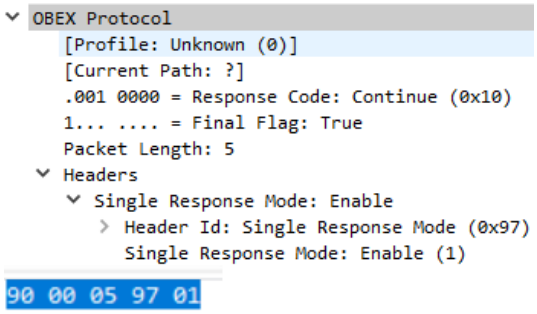

Figura 7. Resposta do dispositivo

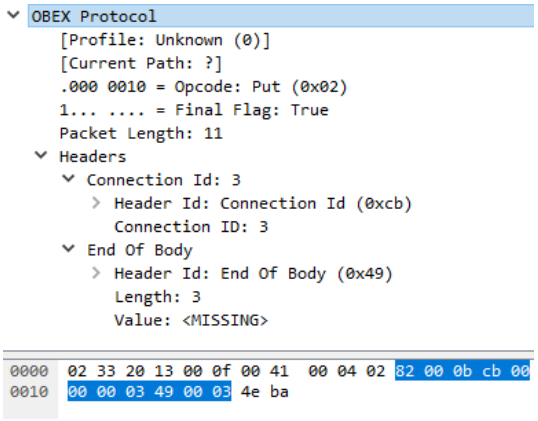

Figura 8. PUT final

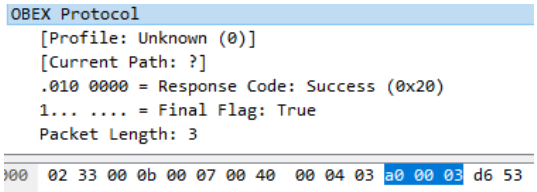

Figura 9. Resposta indicando sucesso

apenas no microcontrolador ESP32. O trabalho também limitou-se a implementar somente a função base do protocolo, deixando outras funções como FTP fora do escopo proposto. O código fonte da biblioteca se encontra no site de repositórios GitHub, disponível em **https://github.com/Lestafo/MicroObex.**

9. Conclusão

Neste trabalho, foi realizado o desenvolvimento de uma biblioteca que corretamente realiza operações ObEx para a transferência de arquivos. Como proposto, a biblioteca segue as especificações definidas pelo Grupo Bluetooth para fornecer ao usuário uma maneira fácil e rápida de utilizar suas funções. A biblioteca é altamente flexível e modular, podendo ser adaptada para quaisquer perfil ou arquitetura de microcontrolador disponível.

Inicialmente houveram dificuldades com travamentos que, após uma análise do

código fonte, determinou-se que estavam sendo causados devido a uma má interpretação do tamanho do pacote a ser enviado. A solução envolveu a reescrita do código referente ao processamento de pacotes. Mais tarde houveram problemas referentes à leitura do pacote pelo dispositivo de destino, mais especificamente com os cabeçalhos de nome do arquivo transferido quanto o cabeçalho contendo seu *MIME TYPE*, que estavam sendo interpretados com caracteres de outras línguas. Uma revisão da documentação OBEX revelou que tais cabeçalhos são interpretados como Unicode, que é um padrão de representação de caracteres que utiliza dois bytes para cada unidade, enquanto a linguagem com a qual a biblioteca foi desenvolvida utiliza ASCII, que utiliza apenas um byte, portanto foram desenvolvidos métodos para a conversão de uma string no padrão ASCII para o padrão Unicode.

Os resultados obtidos pela biblioteca são satisfatórios. Ao chamar suas devidas funções, observa-se que a biblioteca envia um pacote requisitando uma conexão com o dispositivo pareado, contendo todas as informações necessárias para uma conexão OBEX, e então recebe uma resposta contendo os parâmetros do dispositivo e um ID de conexão. Em seguida é enviado o pacote PUT referenciando a conexão formada anteriormente, contendo o nome do arquivo, seu tamanho, seu tipo, e seu conteúdo. A resposta enviada pelo dispositivo indica o sucesso da operação. Então a biblioteca envia o pacote final, uma vez que todo o arquivo foi transferido e então um pacote de desconexão, encerrando a transferência. Todos os pacotes enviados foram corretamente reconhecidos e interpretados pelo dispositivo, indicando que se seguem os padrões definidos pelos perfis OBEX, OPP e GOEP.

O presente trabalho simplifica o processo para envio de arquivos utilizando Bluetooth, que permite o desenvolvimento de aplicações remotas de baixo custo que necessitam recolher dados e enviá-los a outros dispositivos. Exemplos de aplicações da biblioteca seria em uma câmera de monitoramento selvagem, onde a presença frequente do usuário no dispositivo poderia afugentar possíveis animais. Utilizando a biblioteca bastaria o usuário aproximar-se da área em que se encontra o dispositivo para receber as imagens capturadas. A biblioteca também pode ser utilizada em um *stand* de informações, onde normalmente teria um QR code ou uma tela contendo as informações, que acarretaria em um custo adicional elevado, enquanto a biblioteca pode ser implementada em qualquer dispositivo com capacidade Bluetooth, providenciando maior comodidade aos usuários.

Devido à alta extensibilidade da biblioteca, pode-se propor os seguintes trabalhos futuros:

- Adaptar a biblioteca para outros microcontroladores disponíveis no mercado;
- Implementar outros perfis e protocolos que dependem do perfil ObEx;
- Implementar retrocompatibilidade com versões mais antigas do protocolo;
- Estender a biblioteca para facilitar ainda mais a sua utilização.

Referências

DECCIO, C. T. et al. A study of the suitability of irobex for high-speed exchange of large data objects. *IEEE*, 2003.

ERICSSON, T. L. *IoT connections outlook*. 2021. Disponível em: <https://www.ericsson.com/en/mobility-report/dataforecasts/iot-connections-outlook>.

GOUD, G. et al. Efficient real-time support for automotive applications: A case study. In: *12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'06)*. [S.l.: s.n.], 2006. p. 335–341.

HEATH, S. *Embedded Systems Design*. Elsevier Science, 2002. ISBN 9780080477565. Disponível em: <https://books.google.com.br/books?id=BjNZXwH7HlkC>.

LEE, T.-Y.; KIM, K.-H.; JEONG, G.-M. Design of an easy-to-use bluetooth library for wireless sensor network on android. *Contemporary Engineering Sciences*, 2014.

MULLER, N. J. *Networking A to Z*. [S.l.]: McGraw-Hill, 2003.

NI, Y.; LIU, J. Image acquirement with bluetooth technique under embedded linux environment. *IEEE*, 2010.

ODUNLADE, E. *Top 10 popular microcontrollers among makers*. 2020. Disponível em: <https://www.electronics-lab.com/top-10-popular-microcontrollers-among-makers/>.

PARAI, M. K.; DAS, B.; DAS, G. An overview of microcontroller unit: From proper selection to specific application. *International Journal of Soft Computing and Engineering*, 2013.

PERENC, I.; JAWORSKI, T.; DUCH, P. Teaching programming using dedicated arduino educational board. In: *Computer Application in Engineering Education*. [S.l.: s.n.], 2019. v. 27.

PRAVALIKA, V.; PRASAD, C. R. Internet of things based home monitoring and device control using esp32. *International Journal of Recent Technology and Engineering*, 2019.

SPINELLI, G. M.; GOTTESMAN, Z. L. A low-cost arduino-based datalogger with cellular modem and ftp communication for irrigation water use monitoring to enable access to cropmanage. *Elsevier*, 2019.

WOOLLEY, M. Bluetooth core specification v5. In: *Bluetooth*. [S.l.: s.n.], 2019.

ZHANGCHENG, X.; SHUNXIANG, W. File transferring via bluetooth based on the obex protocol. *International Conference on Wireless Networks and Information Systems*, 2009.