Posted by u/frankcohen just now

File Transfer using Bluetooth Classic, ESP32, SD SPI, and Android

I am engineering a watch to play home videos of my children on my wrist - my Reflections project. It seems to me that everyone with the wrist watch will also be carrying an Android or iOS mobile device with Internet service. My goal is to produce a gateway app that the wrist watch (an ESP32 using Bluetooth Classic) uses to get the media files from a Cloud service and store them on an SD card on the ESP32 SPI bus. The media files are binary and are 1 to 40 MB in size. This is my developer's journal of the problems that I encountered and the work-arounds I used.

I distribute the source code to this project under a GPL version 3 open source license. I chose tar and GZip as widely used standards to build and compress an archive of binary (for example, Using TAR files in ESP32 with SD applications for easy multiple file transfers) and text (for example, JSON encoded procedural scripting command) files.

Bluetooth has these advantages over Wifi for this application:

- Less energy, more battery play time.
- Pairing between devices is easier. My wrist watch has no easy user input for selecting Wifi networks (using SSID) and entering passwords.
- Fast enough speed for data transfer. My application does not require real-time data streaming. Tests show average transfer rates with the solution presented here are 74,381 bytes per second. A 2.5 Mbyte file transfers in 32 seconds and a 48 Mbyte file transfers in 13 minutes.

I had another idea to have a Bluetooth session initiate the transfer and fire up Wifi temporarily to move the data. I put this aside as it is complicated to implement. Bluetooth will do just fine.

The project requirements top goals:

- Make this easy for the average Arduino IDE and ESP32 user to understand. Specifically, do not code the ESP32 side using tools outside of Arduino IDE make/compile system.
- Implement an automation pairing and discovery mechanism.
- Implement a command protocol to initiate file send and receive operations.
- Implement a data transfer protocol to validate sent and received data. While it was tempting to implement the FTP standard protocols over Bluetooth transports, using our own data transfer protocol seemed simpler.

Getting Bluetooth to work with large data

Bluetooth surprised me. Where is the file transfer protocol? I found many examples of small data transfer - for example, transferring the moisture content from a sensor in a potted plant. These are

works in reverse of what I expected. One offers an OBEX server on the mobile device and pushes data to the mobile device. For reference I looked at how Apple implements the Bluetooth Transport subclass of the OBEXSession object. I found a lot of overhead to implement an OBEX object. If Meghan Trainer is "all about the bass" then I am about "easy".

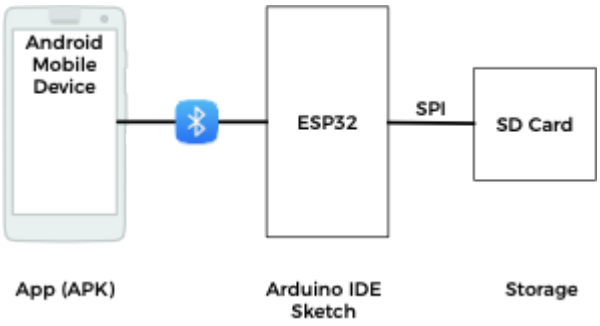
Bluetooth implements a Serial Profile (SPP) to push raw data with some framing to a device. Some tests I did show SPP is faster with better flow-control. I read that some firmware updates get pushed to mobile devices using SPP.

For framing I create a small header to label the packets and a checksum to minimize data integrity risks. Framing also allows for retries and error handling during data transport. There are best practices to fit a payload exactly into the Bluetooth MTU for best throughput, so adding a few bytes to the header is direct and understood.

I also found the ESP32 software distribution includes esp32/hardware/esp32/libraries/BluetoothSerial/SerialToSerialBTM to bridge between Serial and Bluetooth (SPP). Espressif uses this library to deliver Over The Air (OTA) uploads of sketches.

Don't expect much from the official ESP32 doc site. I found no detailed notes on OBEX. https://www.espressif.com/sites/default/files/documentation/esp32_bluetooth_architecture_en.pdf

Hardware



Wiring guide to the Reflections project is at https://github.com/frankcohen/ReflectionsOS/Seuss Display Prototype wiring.pdf.

When initializing the SPI bus and SD card I use:

```
SD.begin( 4, SPI, 80000000 )
```

Low quality (cheap) SD cards may need the bus speed to be lower to work.

Arduino IDE Sketch Using ESP32

https://github.com/frankcohen/ReflectionsOS/blob/main/Gateway/FileTransferGateway.ino

Espressif makes software libraries available to build and deploy ESP32 sketches with Arduino IDE. Software drivers and utilities for ESP32 are at:

`https://github.com/espressif/arduino-esp32`

`https://github.com/nodemcu/nodemcu-firmware/tree/dev-esp32`

Follow these instructions to install ESP32 support to Arduino IDE on a Mac.
https://github.com/espressif/arduino-esp32/blob/master/docs/arduino-ide/mac.md

The important libraries needed for this sketch are part of the above packages.

```
#include <BLEDevice.h>
#include <BLEUtils.h>
#include <BLEServer.h>
#include <BLE2902.h>
#include "BluetoothSerial.h"
These provide the Bluetooth serial support.
```

I created UUID values manually for the service, transmit, and receive hooks. You will need to create your own for your application.

The majority of the code implements a set of callbacks for the Bluetooth stack. Everything is being pushed from the Android app, so the ESP32 initializes the Bluetooth stack to advertise its file transfer service. Then a set of handlers receive the framing, handle the data buffers, and save the received data over SPI to the SD card.

There is no retry logic added (yet). There is no checksum or validation either. I tested the sketch in Arduino IDE 1.8.13. It should work without change in Arduino IDE 2 beta, which just launched a few weeks ago.

With the sketch uploaded and running, use the Serial Monitor to see status. Type ‘send’ and click the Send button to initiate the file transfer.

Android Application

The Android side of this project is a standard application built with Android Studio 4.1.1 on MacOS 11.2.3 (Big Sur). It uses Gradle as a make environment. You need to add a Gradle Scripts -> local.properties file with an entry to the sdk.dir to build. For example:

```
sdk.dir=/Users/frankcohen/Library/Android/sdk
```

I use the Build -> Build Bundle(s) / APK (s) -> Build APK (s) command to create an APK installer file. I transfer the APK to the Android mobile device, turn-on Bluetooth, pair with the ESP32 device, and select a file to transfer (by touching the file transfer user interface element, a file selector appears).

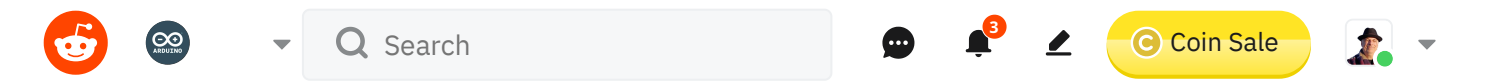
Where to go next?

- From community discussions in the Bluetooth community SPP should transfer 200,000 bytes per second. The implementation here delivers 74,381 bytes per second. Faster transfer is theoretically possible by optimizing the packages and headers.
- I could go file-transfer-crazy and implement the File Transfer Protocol (FTP) to work over Bluetooth, and implement an FTP client working over Bluetooth with Trees.
- This architecture makes it easy to pass commands and data from an ESP32 device to an Android mobile device. It's also ideal to forward the commands and data onto a Cloud based service. That's where I am going next.
- The technique presented works well on Android - I tested it on a OnePlus 7 and a RockIT mobile device. It should also work on an Apple iOS device like an iPad or iPhone. I'd be glad to receive an iOS implementation and contribute it to the project.

Conclusion

In my experience Bluetooth Classic, ESP32, SD SPI, and Android are excellent technology for doing large file transfers.

My watch project is called Reflections and is hosted here.


Search
3
Coin Sale