CSE 321 Term Project: DripBox

Authors: **Pritesh Gupta, Tejas Vyas**

Table of Contents:

Intel Edison on Arduino Kit

Problem Details

There are several solutions available for hosting and sharing files amongst our devices and family members. However, we couldn't come across any free, simple and efficient way of file transfer and sharing.
Sharing files usually involve steps extremely dependent on the internet bandwidth, hosting service and account registrations. Thus, a simple image share needs following steps to be followed in order to be shared:

- Check whether internet speed is optimal,
- Check whether hosting service is up and running,
- Log-in to your specific account,
- Check if you have sufficient space available on the hosting website, (if not pay to get more space)
- Upload the file,
- Make the other use log-in to his account, and finally,
- Download the file.

For sharing files such as family videos and images, there should not be such limitations present and numerous steps involved.

Why We Chose Dripbox?

That's how DripBox was born. DripBox is essentially a DropBox like website, but instead of relying on DropBox servers, the files are hosted on the device itself (Intel Edison [plus flash storage/memory card]). In addition to self-hosting your files, DripBox has following advantages:

- More secure and control (you own and host the files).
- Free.
- No downtime as the device runs as long as it has power and is dedicated only to your files, independent of the file hosting servers and their respective speeds.
- Unlimited storage as storage can be scaled with inexpensive flash drives and/or memory cards which do not need to be dedicated to the hosting service.
- Much faster (no need of Internet, upload/download files upto your network bandwidth).
- Easy UI and UX, no authentication required (the UI can only be accessed by people who are already connected to the same wifi network).

Design of System

The system is designed to be used as a simple file-hosting service dedicated to any device with internet and JavaScript capabilities inside a specific Wi-Fi network.

The idea behind the system is to use Edison Arduino as an http server which acts as a hosting device by using its inbuilt wireless capabilities.

The basic steps involved in the system are:

I. Start Arduino using a power source and connect to your Wi-Fi network

II. Enter a dedicated IP Address for Arduino to use for UI

III. Upload file from a device by going to the network address at which Arduino operates

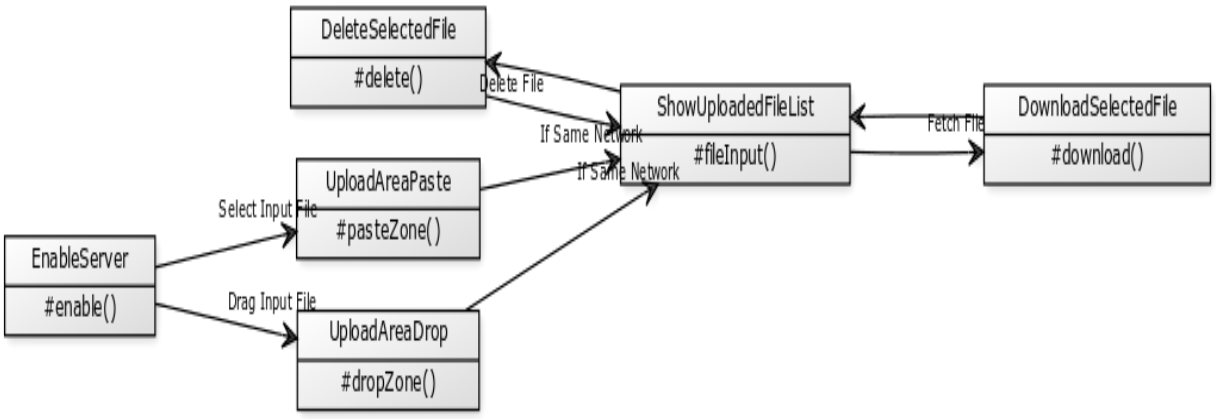
A. Uploading may be performed using Drag and drop or Selecting the file via browsing the drives.

IV. The file should be saved on Arduino and may be accessed from any device around which can fetch the network address.

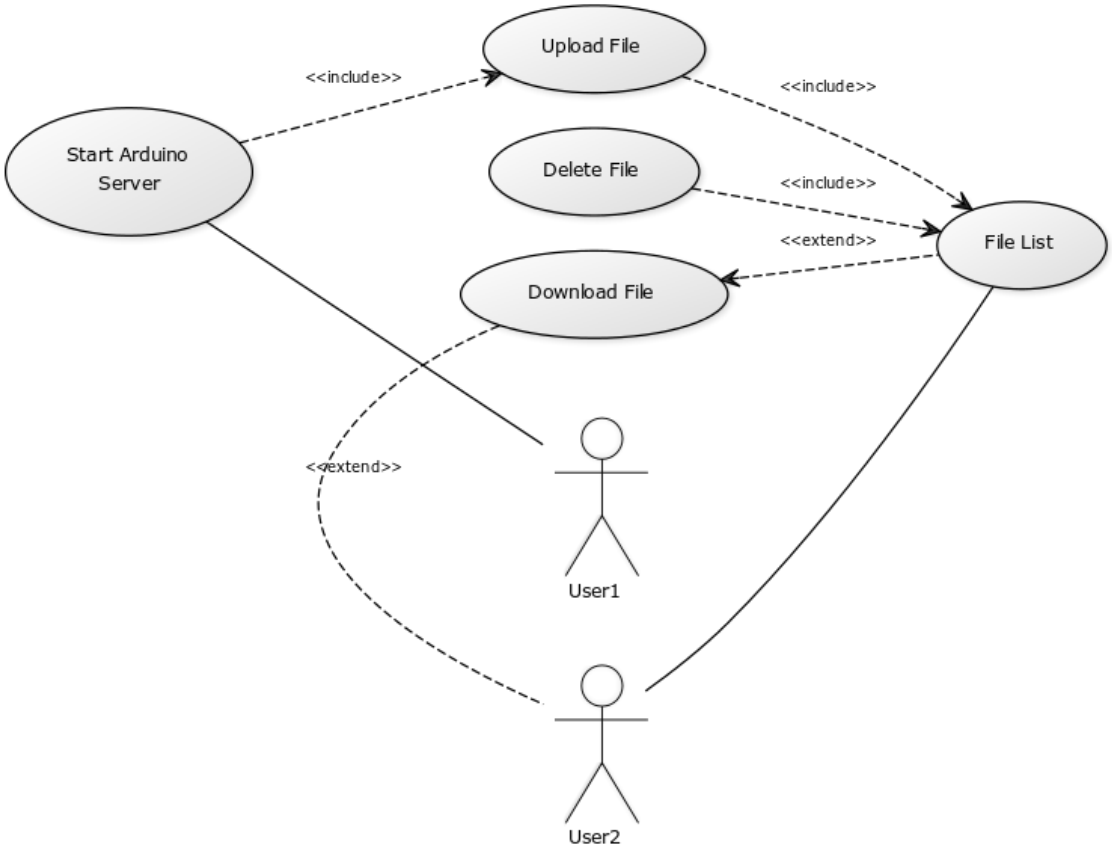
V. Using the design above to upload and download files on any system, i.e., file sharing via Wi-Fi.

UML Diagrams

Use Class Diagram



Use Case Diagram



Implementation

Linux Custom Build in C

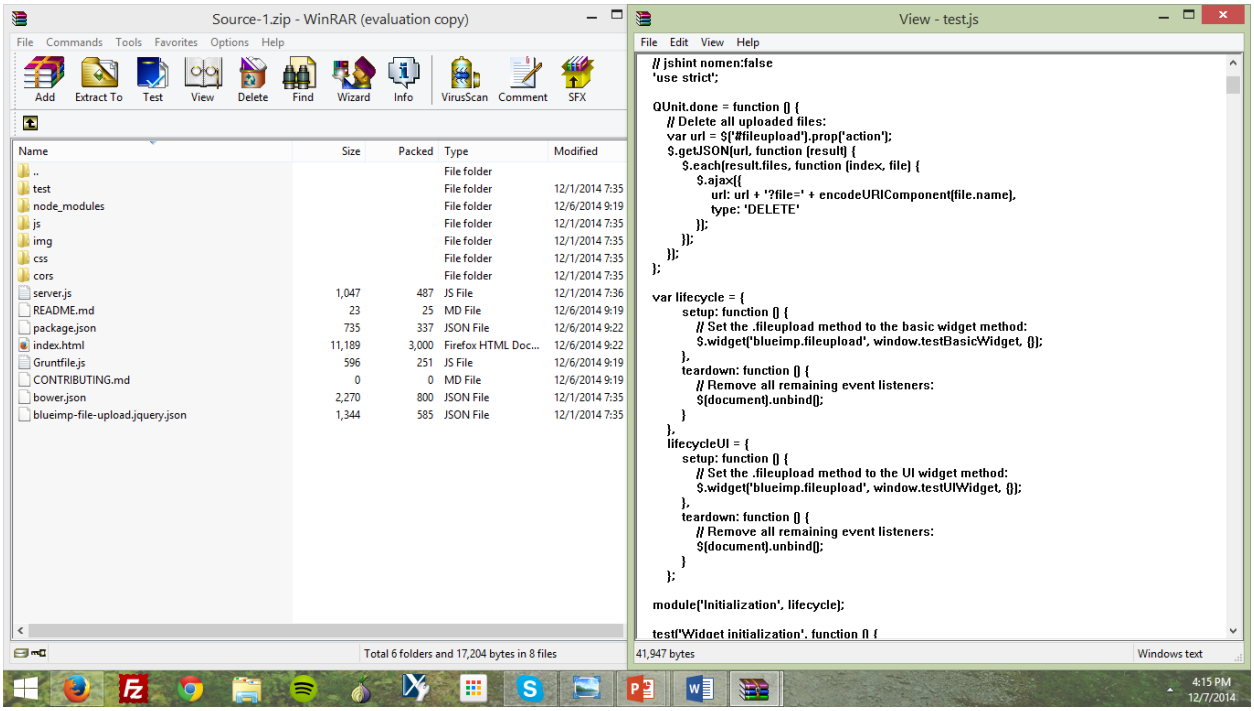
Intel Edison (on Arduino) predominantly supports certain variants of Linux only. Therefore a custom build of Yocto Linux was flashed onto the hardware, and then relevant drivers were enabled and installed (Wi-Fi/network drivers, SSH enabling, USB support/drivers, etc.) along with relevant compilers and technologies (GCC, NODE, NPM, etc.).

Node JS and Main Project

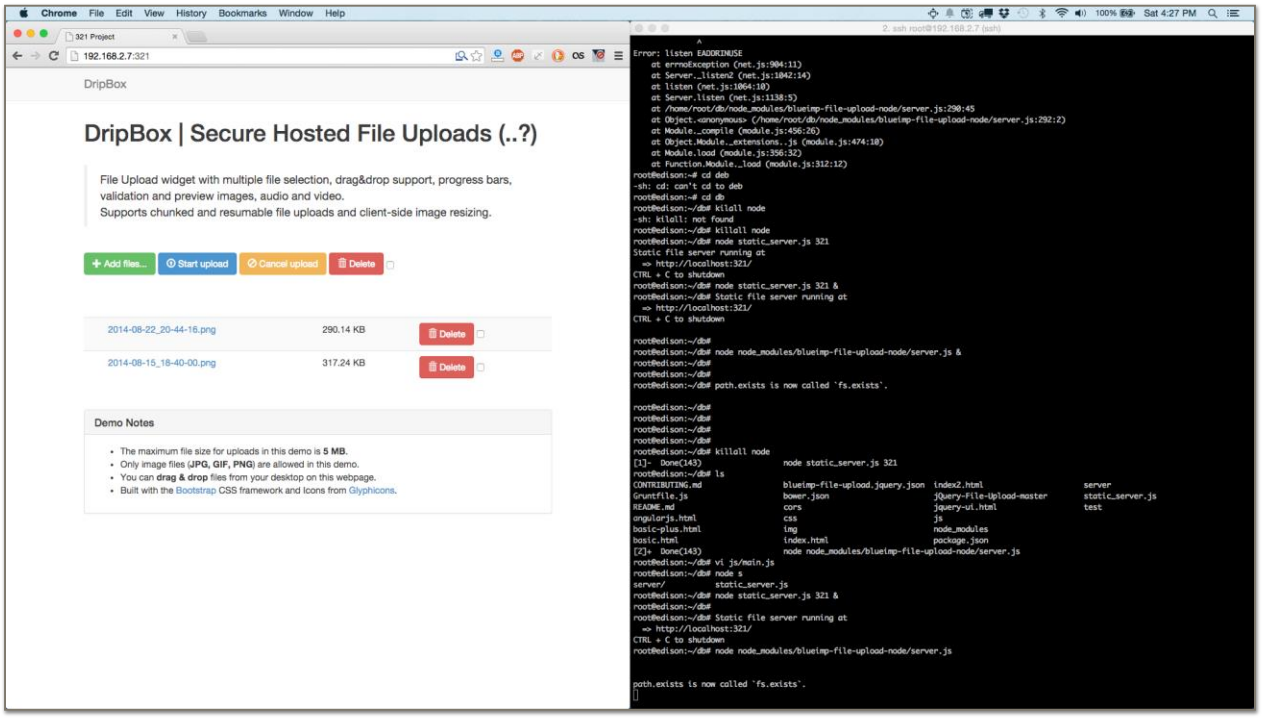
Since this project required extensive file manipulation and handling, therefore using a programming language which supported asynchronous I/O (or non-blocking I/O) made the most practical sense for the scope of technological challenges. Therefore we chose to do all the server side file handling using NodeJS and its relevant NPM modules. For an intuitive UI for frontend, JavaScript jQuery (for dragging/dropping files, upload/loading bars, etc.) library was used with HTML/CSS for an aesthetically appealing look.

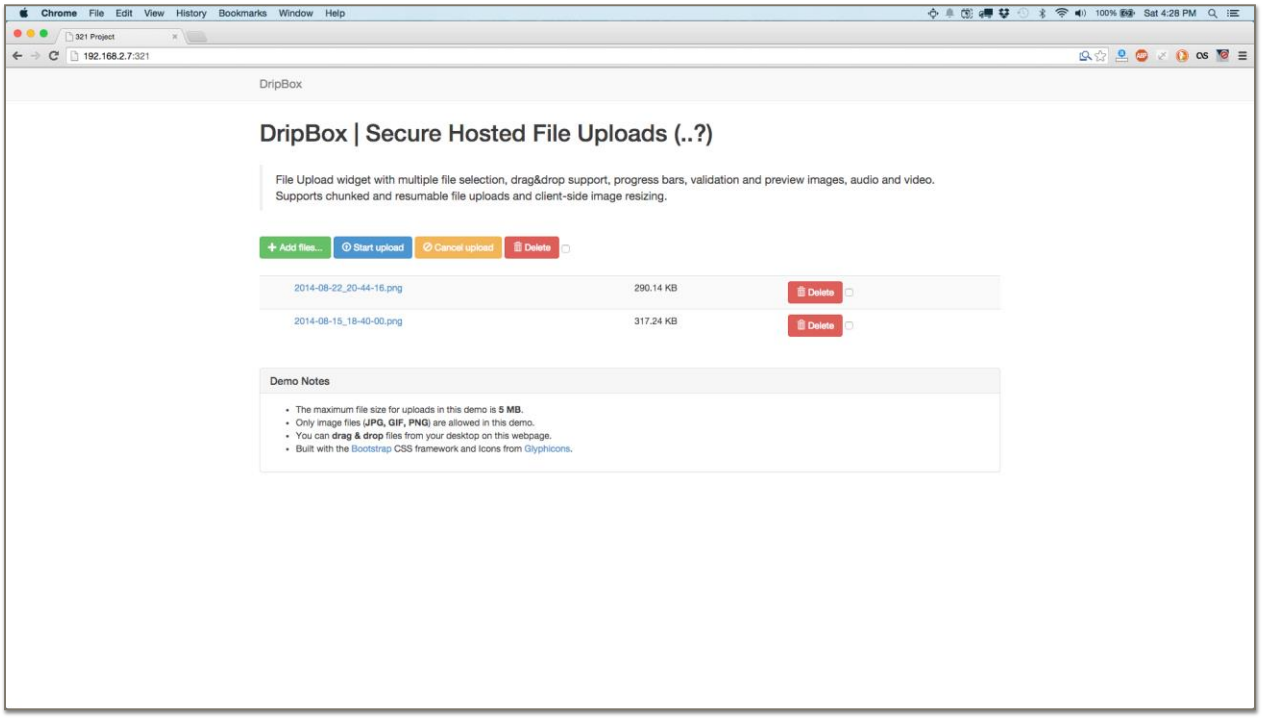
Multiple functions such as enable, disable, delete, fileInput, DropZone, PasteZone, download etc. have been written which manage each part of the process described in the designs section above.

*All source code files are available in the Source.zip file.



Screenshots





*All images included in the directory as a reference.

How It Works

The initial setting up of the device requires it to be connected to a computer, after that, it can be connected to a 5v -12v power adapter and will continue to operate stand alone. The initial setting up includes connecting to Wi-Fi and starting the Node servers on the appropriate ports.

To access the files (for uploading + downloading), the users would just need to go to the specific address:port on which the server was started.

Once the server is running has started and the device is connected to the Wi-Fi, we no longer ever need to physically connect it to the computer, we can simply SSH into the device remotely and manage it from there.

Below is the demonstration YouTube video: (In case embedded video does not work, here's the link: https://www.youtube.com/watch?v=gCfg6v8JgoQ&feature=youtu.be)