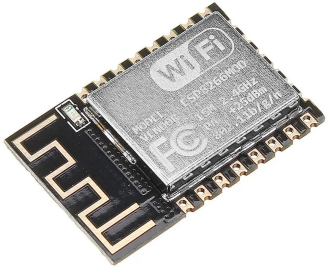
LittleFS

File System Storage

- There are times when storing data in a file can be useful. The data can be collected from sensors or can be configuration data. Whatever it is, the LittleFS library provides a way to allocate some of the SLATE program memory for storing files.
- The SLATE processor board can be set up to support a small file system called **LittleFS**. Details can be found at
- https://arduino-esp8266.readthedocs.io/en/latest/filesystem.html
- File names are limited to 31 characters.
- You program can create, write read, delete and rename files.

File System Storage

- SLATE boards purchased before Fall 2018 have 1Mbyte of Program memory. New SLATES have 4Mbytes. This memory can be divided between the program and the file system.
- In the Tools menu, select the **Flash Size** menu Item. A selection of memory configurations are provided. The file system size is indicated by the **FS:** in parenthesis. For the 1Mbyte SLATE board, up to half the memory can be allocated for the file system. The 4MByte SLATE can have up to ¾ of the memory allocated for the file system.
- Select one. Start with 128K for the 1MByte SLATE and 1M for the 4MByte slate.



4 Mbyte Version
Black Color
1 Mbyte Version
Blue Color

File System Uploader

- There is a tool that can be added to the Arduino IDE to let you upload files. You can create files for your program to use and upload them after uploading your program.
- Go to https://github.com/earlephilhower/arduino-esp8266littlefs-plugin/releases

  and download **ESP8266LittleFS-2.6.0.zip** or higher revision.
- Unzip the file. It will create a directory called **esp8266LttleFS-2.6.0**. Open the directory. There should be **ESP8266LittleFS** directory.
- In the Arduino sketchbook directory where all your programs are stored, called **Arduino**, create a directory named **tools** if it does not exist. The **Arduino** directory should be in the **Documents** directory.
- Move the **ESP8266LittleFS** directory to the tools directory.
- Close the Arduino IDE and restart it. In the Tools menu, you should see **ESP8266 LittleFS Data Upload**
- Remember, you have to upload your program with the filesystem space configured before uploading files.

First File

- First thing to do is include the SPIFFS library.
- In **setup()**, the file system is initialized with **LittleFS.begin()**. This must be done before accessing any files.
- This example, the file writing and reading are done in **setup()** so it is only executed once.

```
#include <LittleFS.h>

void setup()
{
  Serial.begin(115200);
  LittleFS.begin();
  File f = LittleFS.open("/file.txt","w");
  f.println("This is the first line");
  f.println("This is the second line");
  f.close();
  delay(1000);
  f = LittleFS.open("/file.txt","r");
  while(f.available()) {
    String line = f.readStringUntil('\n');
     Serial.println(line);
  }
  f.close();
}

void loop()
{
}
```

Writing and Reading

- First, a file is created by opening a file for writing. object **f** is used to access the file. **File** is the object for connecting to the file in the file system.
- The first argument is the file name. The forward slash is required and is used to indicate the top of the file system.
- **“w”** indicates writing to the file.
- **f.println()** writes to the file similar to how **Serial.println()** works to the serial terminal.
- **println(), print(), write()** functions can be used to send data to the file.
- Lastly, the file needs to be closed.

```
#include <LittleFS.h>

void setup()
{
  Serial.begin(115200);
  LittleFS.begin();
  File f = LittleFS.open(“/file.txt”,”w”);
  f.println(“This is the first line”);
  f.println(“This is the second line”);
  f.close();
  delay(1000);
  f = LittleFS.open(“/file.txt”,”r”);
  while(f.available()) {
    String line = f.readStringUntil('\n');
     Serial.println(line);
  }
  f.close();
}

void loop()
{
}
```

Writing and Reading

- The second half is reading the file that was just written.
- The file is opened using the same file name and replacing **“w”** with **“r”**.
- Then a while loop is used to read through the whole file one line at a time. While there is data to read from the file, the while loop keeps executing. Once the end of the file is reached, the while loop exits and the file is closed.

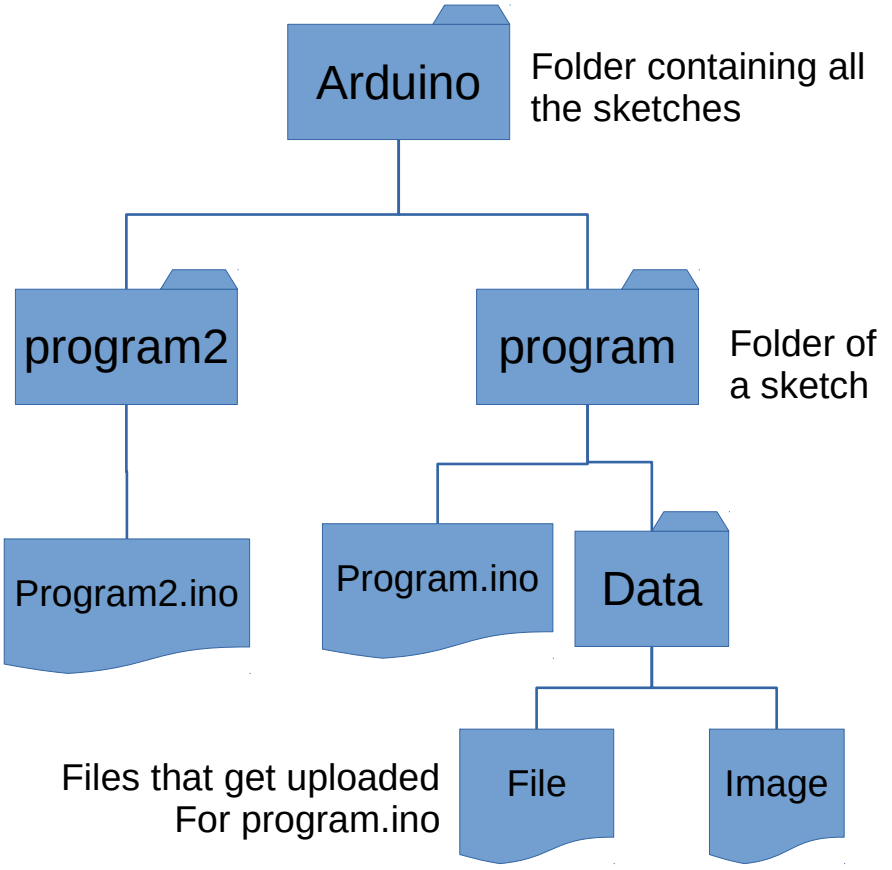
```
#include <LittleFS.h>

void setup()
{
  Serial.begin(115200);
  LittleFS.begin();
  File f = LittleFS.open(“/file.txt”,”w”);
  f.println(“This is the first line”);
  f.println(“This is the second line”);
  f.close();
  delay(1000);
  f = LittleFS.open(“/file.txt”,”r”);
  while(f.available()) {
    String line = f.readStringUntil('\n');
     Serial.println(line);
  }
  f.close();
}

void loop()
{
}
```

File System Upload

- To upload a file, you need to create a directory in the program sketch directory called **data**. Any files placed in this directory will be uploaded when you click on **ESP8266 Sketch Data Upload** menu under **Tools** menu.
- This is useful for uploading html files, images and javascript libraries.



Other File System Functions

- LittleFS.exists(“filename”);
  - This returns true if the file exists.
- LittleFS.remove(“filename”);
  - Deletes the file.
- LittleFS.rename(“oldfile”,”newfile”);
  - Changes the file name from oldfile to newfile.

File Upload Test

- In a text editor, create a short text file called **mytext.txt** and enter some sentences.
- Create a new Arduino program. The program will open the file, read the text and send it to the serial monitor.
- Save the program with the name **readtext**.
- Create the **data** folder in the **readtext** folder.
- Copy **mytext.txt** file to the **data** folder.

```
#include <LittleFS.h>

void setup()
{
  Serial.begin(115200);
  LittleFS.begin();
  while(Serial.available() == 0) delay(1);
  File f = LittleFS.open("mytext.txt","r");
  while(f.available()) {
    String line = f.readStringUntil('\r');
     Serial.println(line);
  }
  f.close();
}

void loop()
{
}
```

File Upload Test

- Compile the program and upload.
- Next, click on the **Tools** menu and select **ESP8266 Data Upload**.
- The text file will upload to the file storage space.
- Open the serial monitor.
- At the top enter any character and click send. This will tell the program to start. The contents of mytext.txt should be displayed.

```
#include <LittleFS.h>

void setup()
{
  Serial.begin(115200);
  LittleFS.begin();
  while(Serial.available() == 0) delay(1);
  File f = LittleFS.open(“mytext.txt”,”r”);
  while(f.available()) {
    String line = f.readStringUntil('\r');
     Serial.println(line);
  }
  f.close();
}

void loop()
{
}
```

File Upload Test

- The program is partially copied from earlier with the write portion of the code deleted.
- The first highlighted line of code is added to allow you to open the serial monitor before the file contents get displayed.
- In the second highlighted line, the readStringUntil() includes the carriage return code. For Mac users, replace that with **\n**.

```
#include <LittleFS.h>

void setup()
{
  Serial.begin(115200);
  LittleFS.begin();
  while(Serial.available() == 0) delay(1);
  File f = LittleFS.open("mytext.txt","r");
  while(f.available()) {
    String line = f.readStringUntil('\r');
     Serial.println(line);
  }
  f.close();
}

void loop()
{
}
```