Micro SD Card Module

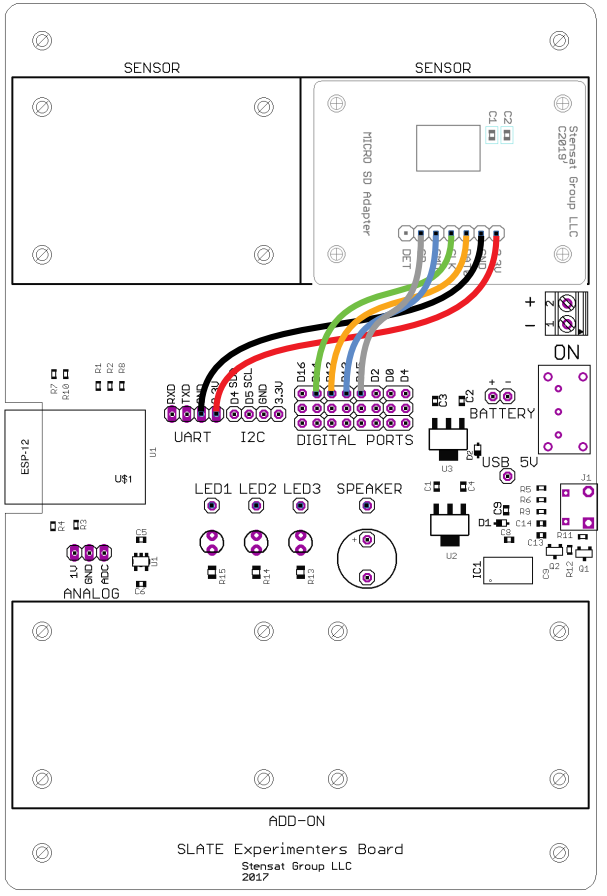
Introduction

- The MicroSD card module allows you to use a microSD card for data storage. The SD library supports the FAT file system which has a 2 GB size limit.
- The interface to the Micro SD card is through the SPI port or Serial Peripheral Interface port. Specific pins are used.
- The CS pin can be assigned to any pin.
- The other pins cannot be moved. They are dedicated to the SPI port.

Signal	Description	Pin
CS	Chip Select	15
DI	Data In	13
DO	Data Out	12
SCLK	Serial Clock	14

Connecting

- Make the connections as shown.
  - 3.3V from the UART to 3.3 on the module
  - GND to GND
  - DAT0 to D12
  - CLK to D14
  - CMD to D13
  - CD to D15



Requirements

- The SD library is included in Arduino.
- The library does not have a function to format the SD card. It must be formatted on another computer and must be in the FAT format. No other formats are supported.
- The SD card must be installed when powering up before the program accesses the SD card.

Example Program

- The example will create a file and write a sentence and then open the file and read the sentence back.
- At the top, the library for the SPI port is included along with the SD library.
- A file object fd is declared.

```
#include <SPI.h>
#include <SD.h>

File fd;

void setup() {
  Serial.begin(115200);
  Serial.println("\nStarting code");
  if(!SD.begin(15)) {
    Serial.println("Error init");
    return;
  }
  fd = SD.open("test.txt",FILE_WRITE);
  if(fd) {
    fd.println("This is a sentence2");
    fd.close();
  } else Serial.println("Error openning");
  fd = SD.open("test.txt");
  if(fd) {
    Serial.println(fd.size());
    while(fd.available()) {
      Serial.write(fd.read());
    }
    fd.close();
  } else Serial.println("File can't open");
}

void loop() {
}
```

Example Program

- The SD has to be initialized first. This is not formatting the SD card but setting up the interface. The CS pin is specified.
- If there is an error in setting up the interface, the program exits the **setup()** function without attempting to write to a file.

```
#include <SPI.h>
#include <SD.h>

File fd;

void setup() {
  Serial.begin(115200);
  Serial.println("\nStarting code");
  if(!SD.begin(15)) {
    Serial.println("Error init");
    return;
  }
  fd = SD.open("test.txt",FILE_WRITE);
  if(fd) {
    fd.println("This is a sentence2");
    fd.close();
  } else Serial.println("Error openning");
  fd = SD.open("test.txt");
  if(fd) {
    Serial.println(fd.size());
    while(fd.available()) {
      Serial.write(fd.read());
    }
    fd.close();
  } else Serial.println("File can't open");
}

void loop() {
}
```

Example Program

- Next, a file is opened for writing. The file name is in quotes and **FILE_WRITE** specifies writing.
- If the file is successfully opened, the sentence is written into the file and the file is then closed. If there was an error, an error message would be displayed on the serial monitor.
- If the file exists, anything written to the file is added to what exists in the file.

```
#include <SPI.h>
#include <SD.h>

File fd;

void setup() {
  Serial.begin(115200);
  Serial.println("\nStarting code");
  if(!SD.begin(15)) {
    Serial.println("Error init");
    return;
  }
  fd = SD.open("test.txt",FILE_WRITE);
  if(fd) {
    fd.println("This is a sentence2");
    fd.close();
  } else Serial.println("Error openning");
  fd = SD.open("test.txt");
  if(fd) {
    Serial.println(fd.size());
    while(fd.available()) {
      Serial.write(fd.read());
    }
    fd.close();
  } else Serial.println("File can't open");
}

void loop() {
}
```

Example Program

- The file is opened again. This time, only the file name is specified. This automatically sets the file to be read.
- If the file is successfully opened, the size of the file is displayed.
- the **while()** loop reads all the data in the file.
- When the **while()** loop exits, the file is closed.
- If the file cannot be opened, an error message is displayed.

```
#include <SPI.h>
#include <SD.h>

File fd;

void setup() {
  Serial.begin(115200);
  Serial.println("\nStarting code");
  if(!SD.begin(15)) {
    Serial.println("Error init");
    return;
  }
  fd = SD.open("test.txt",FILE_WRITE);
  if(fd) {
    fd.println("This is a sentence2");
    fd.close();
  } else Serial.println("Error openning");
  fd = SD.open("test.txt");
  if(fd) {
    Serial.println(fd.size());
    while(fd.available()) {
      Serial.write(fd.read());
    }
    fd.close();
  } else Serial.println("File can't open");
}

void loop() {
}
```

Other Functions

- The SD library includes other functions
  - SD.exists(“filename”) will return a 1 if the file exists or 0 if it does not.
  - SD.mkdir(“directory”) will create a directory on the SD card. Multiple levels of directories are allowed.
  - SD.remove(“filename”) will delete the specified file.
  - SD.rmdir(“directory”) will delete the specified directory.
  - size() will return the size of the file that is opened.

Data Logging

- This code collects analog voltage from the ADC port at 10 samples per second for 10 seconds. It will save the data with a time tag into file data.csv. You can take the SD card to another computer and retrieve the data and import it into a spreadsheet as a CSV formatted file.

```
#include <SPI.h>
#include <SD.h>

File fd;
int count = 0;

void setup() {
  Serial.begin(115200);
  Serial.println("\nStarting code");
  if(!SD.begin(15)) {
    Serial.println("Error init");
    return;
  }
  fd = SD.open("data.csv",FILE_WRITE);
  if(!fd) {
    Serial.println("Error openning");
    return;
  }
}

void loop() {
  int a = analogRead(0);
  unsigned long t = millis();
  fd.print(t);fd.print(“,”);
  fd.println(a);
  count++;
  if(count > 1000) {
    fd.close();
    Serial.println(“Done”);
    while(1) delay(100);
  }
  delay(100);
}
```

Data Logging

- Variable **count** is initialized to zero. This will count the number of samples to save.
- In **setup()**, the SD card is initialized and the file is created or opened if it exists.
- If an error occurs, the program exits the **setup()** function. Nothing else will execute properly.

```
#include <SPI.h>
#include <SD.h>

File fd;
int count = 0;

void setup() {
  Serial.begin(115200);
  Serial.println("\nStarting code");
  if(!SD.begin(15)) {
    Serial.println("Error init");
    return;
  }
  fd = SD.open("data.csv",FILE_WRITE);
  if(!fd){
    Serial.println("Error openning");
    return;
  }
}

void loop() {
  int a = analogRead(0);
  unsigned long t = millis();
  fd.print(t);fd.print(“,”);
  fd.println(a);
  count++;
  if(count > 1000) {
    fd.close();
    Serial.println(“Done”);
    while(1) delay(100);
  }
  delay(100);
}
```

Data Logging

- In the **loop()** function, the analog port value is read.
- The next line reads the time in milliseconds. The **millis()** function returns the number of milliseconds since the start of the program.
- Next, the data is written to the file the same was as it would be sent to the serial interface.
- A comma is inserted between the time and the ADC value. This is the csv format.

```
#include <SPI.h>
#include <SD.h>

File fd;
int count = 0;

void setup() {
  Serial.begin(115200);
  Serial.println("\nStarting code");
  if(!SD.begin(15)) {
    Serial.println("Error init");
    return;
  }
  fd = SD.open("data.csv",FILE_WRITE);
  if(!fd) {
    Serial.println("Error openning");
    return;
  }
}

void loop() {
  int a = analogRead(0);
  unsigned long t = millis();
  fd.print(t);fd.print(",");
  fd.println(a);
  count++;
  if(count > 1000) {
    fd.close();
    Serial.println("Done");
    while(1) delay(100);
  }
  delay(100);
}
```

Data Logging

- In this section of the code, the count is incremented by one.
- The count is checked to see if more than a 1000 samples has been collected.
- If more than 1000 samples have been collected, the file is closed the and program loops forever executing the **delay(100)**. This is required to keep the processor from crashing. The delay function allows the processor to handle other background operations. It's just how the software works. A watchdog timer is reset in the background. If it isn't the processor reboots and the program will restart.

```
#include <SPI.h>
#include <SD.h>

File fd;
int count = 0;

void setup() {
  Serial.begin(115200);
  Serial.println("\nStarting code");
  if(!SD.begin(15)) {
    Serial.println("Error init");
    return;
  }
  fd = SD.open("data.csv",FILE_WRITE);
  if(!fd) {
    Serial.println("Error openning");
    return;
  }
}

void loop() {
  int a = analogRead(0);
  unsigned long t = millis();
  fd.print(t);fd.print(",");
  fd.println(a);
  count++;
  if(count > 1000) {
    fd.close();
    Serial.println("Done");
    while(1) delay(100);
  }
  delay(100);
}
```

Data Logging

- The last line of code is the delay(100). This sets the loop() operation to occur at a 10 Hz rate so samples are collected 10 times per second.
- Connect the temperature sensor to the ADC port. Upload and run the program. While the program is running, touch the temperature sensor to change the temperature. Let it go and touch it at different intervals over about a 100 second period until the program displays done in the serial monitor.
- Remove the SD card and connect it to another computer and import the file into a spreadsheet. You should be able to plot the data with the first column time and the second the ADC value. You can convert the ADC value to a temperature in the spreadsheet.

```
#include <SPI.h>
#include <SD.h>

File fd;
int count = 0;

void setup() {
  Serial.begin(115200);
  Serial.println("\nStarting code");
  if(!SD.begin(15)) {
    Serial.println("Error init");
    return;
  }
  fd = SD.open("data.csv",FILE_WRITE);
  if(!fd) {
    Serial.println("Error openning");
    return;
  }
}

void loop() {
  int a = analogRead(0);
  unsigned long t = millis();
  fd.print(t);fd.print(“,”);
  fd.println(a);
  count++;
  if(count > 1000) {
    fd.close();
    Serial.println(“Done”);
    while(1) delay(100);
  }
  delay(100);
}
```

Long Duration Open Ended Data Collection

- This example shows how to collect data at a slow interval and for an unknown duration. This will allow you to stop the program at any time and retrieve the SD card without the worry of corrupting the SD card.
- The library for using the SD card will buffer a certain amount in memory before writing. Removing the card before it is written cause the loss of data.
- Removing the SD card before the file is closed can corrupt the data file.
- The next example shows the sequence to avoid loss of data and corruption of the file.

Long Duration Open Ended Data Collection

- The code is similar to before but samples the temperature sensor once every 10 seconds.
- In the **setup()** function, only the SD card interface is initialized.
- In the **loop()** function, the temperature sensor is sampled and the time is captured in milliseconds.
- Next, the file is opened. If the file does not exist, it will be created. If it exists, the new data is appended to the existing file.

```
#include <SPI.h>
#include <SD.h>

File fd;

void setup() {
  Serial.begin(115200);
  Serial.println("\nStarting code");
  if(!SD.begin(15)) {
    Serial.println("Error init");
    return;
  }
}

void loop() {
  int a = analogRead(0);
  unsigned long t = millis();
  fd = SD.open("long.csv",FILE_WRITE);
  if(!fd){
    Serial.println("Error opening");
    return;
  }
  fd.print(t);fd.print(“,”);
  fd.println(a);
  Serial.println(“Wrote sample”);
  fd.close();
  delay(10000);
}
```

Long Duration Open Ended Data Collection

- The file is then closed right after sending the data to the SD card. This forces the file to be immediately updated and properly closed.
- The program then waits 10 seconds before capturing the next sample.
- You can run the program and let it run for several hours in a room or outside and collect temperature data over time.
- You can the same thing with the light sensor connected. Point it out a window and see how the daylight varies over time.
- To relate it to actual time, you need to manually record the time you started the program.

```
#include <SPI.h>
#include <SD.h>

File fd;

void setup() {
  Serial.begin(115200);
  Serial.println("\nStarting code");
  if(!SD.begin(15)) {
    Serial.println("Error init");
    return;
  }
}

void loop() {
  int a = analogRead(0);
  unsigned long t = millis();
  fd = SD.open("long.csv",FILE_WRITE);
  if(!fd){
    Serial.println("Error opening");
    return;
  }
  fd.print(t);fd.print(“,”);
  fd.println(a);
  Serial.println(“Wrote sample”);
  fd.close();
  delay(10000);
}
```

Reading a Data File

- The python code will open a text file and read each line. A dialog window will open asking to select the file.
- The first part is importing the needed modules and creating object **top** for **Tk**.

```
from tkinter import *
from tkinter import filedialog as fd

top = Tk()

def readfile():
  filen = fd.askopenfilename()
  fd = open(filen,”r”)
  while 1:
    line = fd.readline()
    if not line:
      break
    print(line);
  fd.close()

l1 = Label(top,text="File Reader")
b1 = Button(top,text="Read Data",command=readfile)

l1.pack()
b1.pack()
mainloop()
```

Reading a Data File

- The next part is creating a function to read the file and display the contents.
- A request for the filename is made using the **askopenfilename()** function.
- The selected file is then opened with the **open()** function. The open function takes two arguments, a file name and how to open it.
  - “r” is read
  - “w” is write or overwrite existing file.
  - “a” is append to existing file

```
from tkinter import *
from tkinter import filedialog as fd

top = Tk()

def readfile():
  filen = fd.askopenfilename()
  fd = open(filen,”r”)
  while 1:
    line = fd.readline()
    if not line:
      break
    print(line);
  fd.close()

l1 = Label(top,text="File Reader")
b1 = Button(top,text="Read Data",command=readfile)

l1.pack()
b1.pack()
mainloop()
```

Reading a Data File

- In the while loop, one line is read. **readline()** will read in text until an end of line character is detected such as carriage return or line feed.
- Notice the **if not line:** if variable **line** is empty, that means the end of the file so the break command is executed which causes the program to exit the while loop. The file is closed and the function ends execution.

```
from tkinter import *
from tkinter import filedialog as fd

top = Tk()

def readfile():
  filen = fd.askopenfilename()
  fd = open(filen,”r”)
  while 1:
    line = fd.readline()
    if not line:
      break
    print(line);
  fd.close()

l1 = Label(top,text="File Reader")
b1 = Button(top,text="Read Data",command=readfile)

l1.pack()
b1.pack()
mainloop()
```

Reading a Data File

- Continuing with the program, a label is created and a button is created. When the button is clicked, the **readfile()** function will execute.
- The last three lines set up the window with the label above the button.
- **mainloop()** scans for the button to be pressed and executes the **readfile()** function.

```
from tkinter import *
from tkinter import filedialog as fd

top = Tk()

def readfile():
  filen = fd.askopenfilename()
  fd = open(filen,”r”)
  while 1:
    line = fd.readline()
    if not line:
      break
    print(line);
  fd.close()

l1 = Label(top,text="File Reader")
b1 = Button(top,text="Read Data",command=readfile)

l1.pack()
b1.pack()
mainloop()
```

Reading a Data File

- Insert the SD card into the computer and run the python program.
- Select a file from the SD card.
- The contents of the file should be displayed in the python IDE window.

```
from tkinter import *
from tkinter import filedialog as fd

top = Tk()

def readfile():
  filen = fd.askopenfilename()
  fd = open(filen,”r”)
  while 1:
    line = fd.readline()
    if not line:
      break
    print(line);
  fd.close()

l1 = Label(top,text="File Reader")
b1 = Button(top,text="Read Data",command=readfile)

l1.pack()
b1.pack()
mainloop()
```