Arduino Data Logger: Serial Monitor Alternatives

Forget the serial monitor, here's 5 other ways to do Arduino data logging

The Arduino serial monitor is usable when you want to watch data from an Arduino. However, it does not have a built-in method for saving the data. Here are some ideas if you want to build an Arduino data logger with or without a PC.

by James Lewis https://www.baldengineer.com/arduino-data-logger-options.html

Important note on Arduino Data Logger examples

With all of these examples, please remember that whenever you open the Arduino's serial port, the board will reset. So if your log file shows "Initializing SD card…" with a few data lines in between, it is because there is a reset happening.

```
1
2 Initializing SD card…initialized.
3 Temp: 34, Time: 03:24:44
4 Temp: 33, Time: 03:24:45
5 Temp: 34, Time: 03:24:46
6 Tem
7
8 Initializing SD card…initialized.
9 Temp: 34, Time: 03:24:50
10 Temp: 34, Time: 03:24:51
11 Temp: 33, Time: 03:24:52
Temp: 34, Time: 03:24:53
```

In that code you can see data logging started and then restarted. What happened is that after programming, the board starts logging. Then when you open the Serial Monitor, the data logger restarts.

To solve this issue, either disable auto-reset, add a 3-4 second delay at the start of setup(), wait for a character to be received, or wait for a button press. That will give you time to open the Serial Monitor.

SD Card

If you can add an SD card to your project, that is an obvious option to create an Arduino data logger. Some Ethernet shields come with an SD Card reader. Alternatively, you can get an adapter to add SD Cards.

SD Cards use a form of the SPI protocol, so it is easy to interact with them from an Arduino. Just make sure it is a slower class. I use Class 4 cards with my projects. And I make sure then are 16GB or less. Newer or faster cards use a different protocol and won't work.

I will not provide a code example here. The IDE comes with an SD Example called "Datalogger." Guess what it does.

Better Serial Terminal

In the past, I mentioned some Arduino serial monitor alternatives. I find these tools useful for real-time interaction. For Windows there is PuTTY, and for all operating systems there is the command-line tool 'screen.'

Most terminal apps will offer a "save buffer." Others provide controls for keeping serial output. Personally, I prefer the simplicity of "screen."

For an Arduino data logger, just use the -L command line option. This switch will save a file called "screenlog.0″, where the 0 is the number of the screen. In most cases, it will be 0.

```
1
2
3 MacMan:~ james$ screen -L /dev/tty.usbserial-AE01AX15 > text.txt
4 6:41:23 PM 4/3/2017,1481
5 6:41:24 PM 4/3/2017,1481
6 BinBoo Release 0.3 - James Lewis (james@baldengineer.com  )
7 Use date +%s to get current time in seconds
8 RTC has set the system time
9 6:41:27 PM 4/3/2017,1481
10 6:41:28 PM 4/3/2017,1481
11 6:41:29 PM 4/3/2017,1481
12 6:41:30 PM 4/3/2017,1481
13 [screen exited]
14
15 MacMan:~ james$ more screenlog.0
16 6:41:23 PM 4/3/2017,1481
17 6:41:24 PM 4/3/2017,1481
18 BinBoo Release 0.3 - James Lewis (james@baldengineer.com  )
19 Use date +%s to get current time in seconds
20 RTC has set the system time
21 6:41:27 PM 4/3/2017,1481
22 6:41:28 PM 4/3/2017,1481
6:41:29 PM 4/3/2017,1481
6:41:30 PM 4/3/2017,1481
```

MQTT

If you have WiFi, like with an ESP8266, you could use MQTT. Set up a broker on your PC (or Raspberry Pi). Then publish the messages to the MQTT broker. When I use this method, I use the topic "debug." Why debug? Because "Arduino Data Logger" is too long! Of course, you will need something to capture the content. You could use mosquitto_sub on the broker to capture it directly to a file.

```
MacMan:~ james$ mosquitto_sub -h rasppib -t debug >> mqtt-example.txt
^C
MacMan:~ james$ more mqtt-example.txt
181, Lights Turning On (Receive: 1)
182, Lights Turning Off (Receive: 0)
183, Lights Turning On (Receive: 1)
184, Lights Turning Off (Receive: 0)
185, Lights Turning Off (Receive: 0)
186, Warning: Lights Alright Off, not doing that again!
187, Lights Turning Off (Receive: 0)
188, Warning: Lights Alright Off, not doing that again!
189, Lights Turning Off (Receive: 0)
190, Warning: Lights Alright Off, not doing that again!
MacMan:~ james$
```

The downside to this approach is that you will not get to see the saved data in real-time. If you want to see the messages on-screen while saving to a file, you could run screen -L from above and then mosquitto_sub.

Python (and other scripting languages)

Another option for an Arduino data logger is to use a scripting language. Open up the serial port and save the contents to a file. This approach has the added benefit that you can also act on the incoming data. For example, you could use PyQt, like from the [Raspberry Pi GUI tutorial], to create a trend graph of that data.

You may need to install "pyserial." Run pip or pip3, depending on your Python version. Or to be safe, run both.

```
ModuleNotFoundError: No module named 'serial'

MacMan:~ james$ pip install pyserial
MacMan:~ james$ pip3 install pyserial
```

Here's a bit of code that will capture the serial port, print to the screen and write to a file. (Overwrites if the file exists.) When you are done logging, hit ctrl-c to exit the script.

```
#!/usr/local/bin/python3

import serial, io

device   = '/dev/tty.usbserial-AE01AX15' # serial port
baud     = 9600                         # baud rate
filename = 'bald-log.txt'               # log file to save data in

with serial.Serial(device,baud) as serialPort, open(filename,'wb') as
outFile:
    while(1):
        line = serialPort.readline() # must send \n! get a line of log
        print (line)                 # show line on screen
        outFile.write(line)          # write line of text to file
        outFile.flush()              # make sure it actually gets written
out
```

Python will complain, like in the output below, but that is okay. outFile.flush() makes sure the file's contents are saved.

```
1
2 MacMan:2017-04-05 datalogger options james$ ./python-serial-logger.py
3 b'BinBoo Release 0.3 - James Lewis (james@baldengineer.com  )\r\n'
4 b'Use date +%s to get current time in seconds\r\n'
5 b'RTC has set the system time\r\n'
6 b'7:16:53 PM 4/3/2017,6191\r\n'
7 b'7:16:54 PM 4/3/2017,6191\r\n'
8 b'7:16:55 PM 4/3/2017,6191\r\n'
9 b'7:16:56 PM 4/3/2017,6191\r\n'
10 b'7:16:57 PM 4/3/2017,6191\r\n'
11 ^CTraceback (most recent call last):
12   File "./python-serial-logger.py", line 11, in <module>
13     line = serialPort.readline() # must send \n! get a line of log
14   File "/usr/local/lib/python3.6/site-packages/serial/serialposix.py",
15line 483, in read
16     ready, _, _ = select.select([self.fd, self.pipe_abort_read_r], [],
17[], timeout.time_left())
18KeyboardInterrupt
19
20MacMan:2017-04-05 datalogger options james$ more bald-log.txt
21BinBoo Release 0.3 - James Lewis (james@baldengineer.com  )
22Use date +%s to get current time in seconds
23RTC has set the system time
24 7:16:53 PM 4/3/2017,6191
25 7:16:54 PM 4/3/2017,6191
 7:16:55 PM 4/3/2017,6191
 7:16:56 PM 4/3/2017,6191
 7:16:57 PM 4/3/2017,6191
```

I think compared to the other options, using Python is a bit of overkill. However, if you want to extend the script to respond to the logged values, Python creates an excellent one-stop solution.

Conclusion

Whether you want an Arduino data logger with or without a PC, you have several options. Personally, MQTT and screen are my favorite two methods. MQTT on an ESP8266 does not require a tethered PC.

I am curious about how difficult it would be to make a PyQt based GUI that logs and graphs the data it is receiving. Any takers?