Speed of Capturing SPI Data and Writing to Disk without Data Loss

### Dylan Kreth | Source code available at <https://github.com/dkreth/pi_test>

# Overview

This report aims show the feasibility of using a Raspberry Pi to capture SPI data and write it to a disk without any data loss, and measure the speed at which it can do so. The report is broken down into 3 sections:

1. Measuring speed of capturing data with the SPI interface with varying block sizes
2. Measuring speed of writing data to the Raspberry Pi with different file sizes
3. Measuring speed of doing both operations—SPI capture and write to disk—in series

# Specs

Compiler: GNU GCC Compiler  
Hardware: Raspberry Pi 3 B+, OS installed on 16GB SanDisk microSD card, 8GB USB A drive  
OS: Raspbian  
Libraries used: *stdlib.h*, *iostream*, *fstream*, *bcm2835.h*, *sys/time.h*

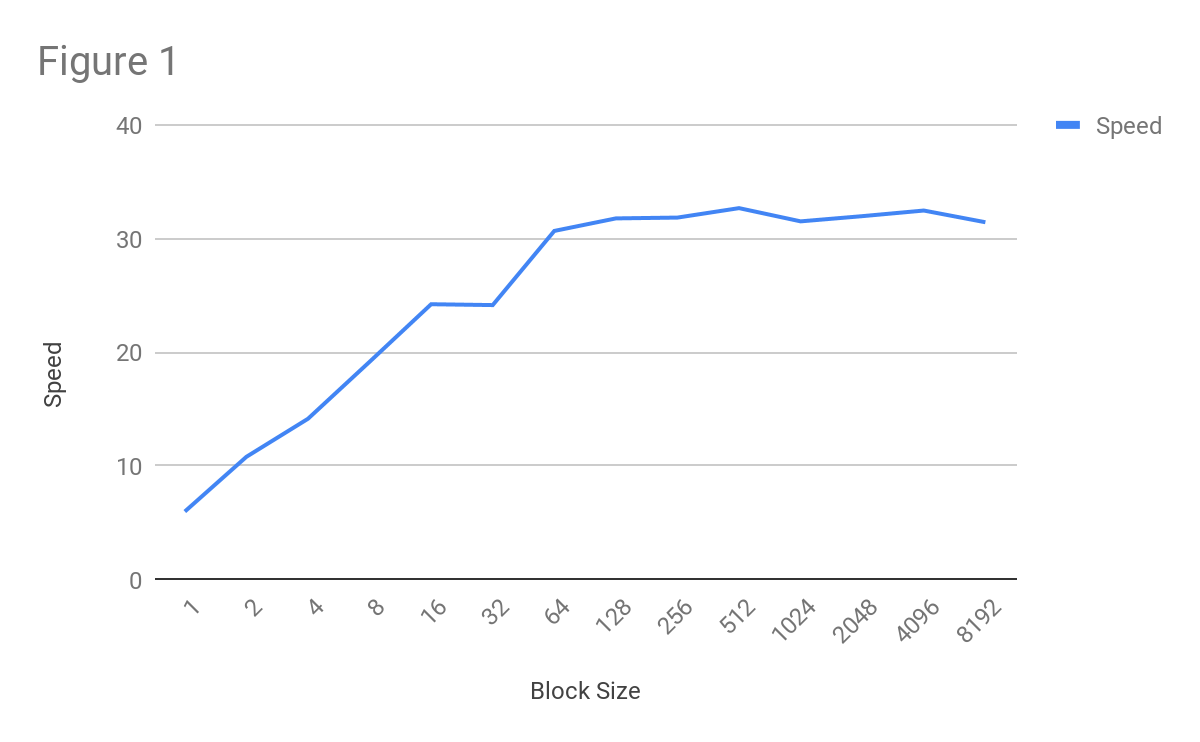
# Summary of Results

* Maximum clock speed that could reliably be used was the 8x clock divider (50MHz on Raspberry Pi 3). Any faster would make the Pi glitch or miss data
* Max speed of SPI transfer was approximately 32 Mb/s and could be accomplished with blocks of 256 bytes. Larger blocks increased latency without affecting speed. CPU usage for SPI\_TEST was 20-25%.
* No clear difference in write speed could be determined between writing files of 1MB and 256 MB. CPU usage for MEM\_TEST was 5-8%, with occassional fluctuations to 15%.
* Performing a SPI transfer followed by writing the data to a binary file can be performed reliably (without data loss) at a rate of about 23.6 Mb/s. CPU usage for COMBO\_TEST was 20-25%.

# SPI test

Through tests, it was found that the maximum clock speed that could reliably be used was the 8x clock divider (50MHz on Raspberry Pi 3). Any faster would make the Pi glitch or miss data.

Then, various blocks sizes (powers of 2 from 1 to 8192) were tested and timed to see how long it took to send a total of 8KB of data (e.g. spin send 1 byte 8192 times, 2 bytes 4096 times, 4 bytes 2048 times, 8 bytes 1024 ... 2048 bytes 4 times, 4096 bytes 2 times, and 8192 bytes 1 time). As expected, sending data in blocks 1 byte at a time was the slowest, averaging around 5.98Mb/s. Increasing the block size corresponded to an increase in the speed until it leveled out to a max of about 31-32Mb/s, as seen in the Figure 1. Max speed could be accomplished with blocks of 256 bytes.

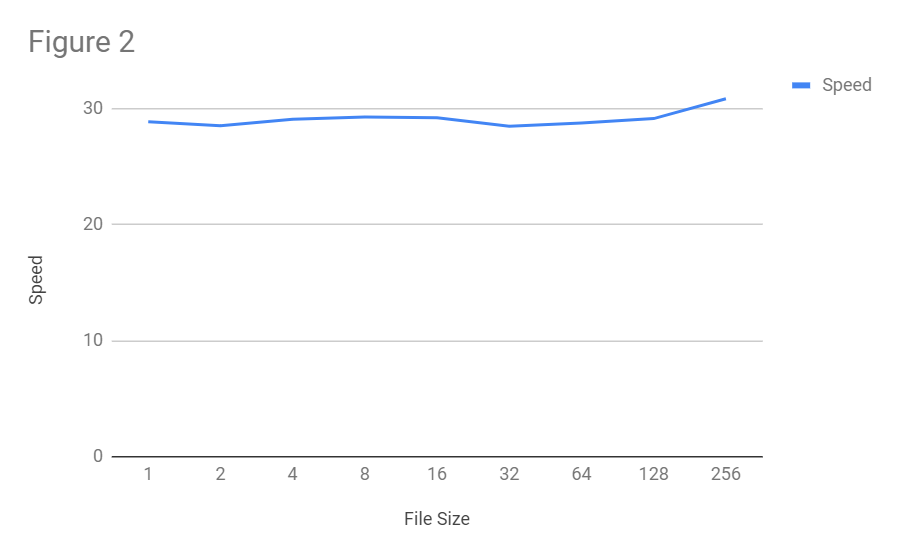


# Memory Test

Varying the size of files (again using powers of 2), the time was measured for how long it took to write enough files so that 256 MB of data were written. (e.g. write 1 256MB file, 2 128MB files, 4 64MB files, 8 32 MB files, 16 16MB files, 32 8MB files, 64 4MB files, 128 2MB files, and 256 1MB files).

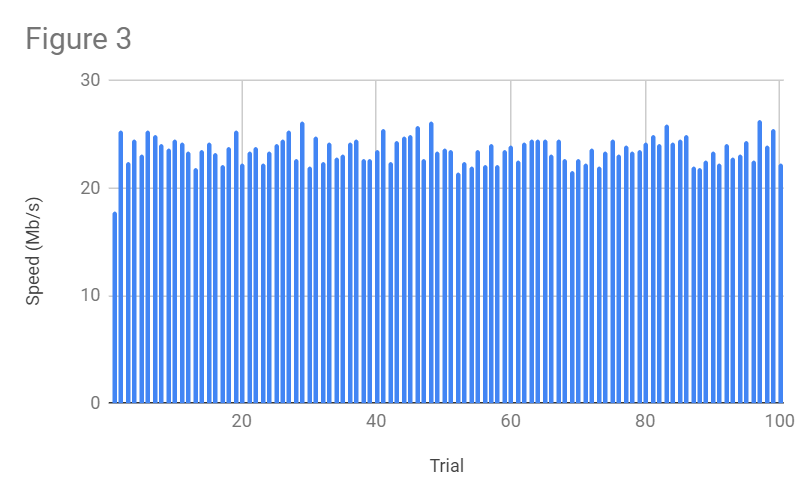
The test was run with an increasing file size (i.e. starting with 1 256MB file and ending with 256 1MB files), and then again but in reverse order (the latter being MEM\_BACKWARDS\_TEST ).

There was minimal variation in the trials, showing no clear correlation between file size and write speed (Figure 2).



# Combo Test (SPI & Memory)

In this test, the SPI is sent blocks of 256 bytes, the data is returned via a loopback, and this returned data is then written to a .bin file. This is done 75,000 times in a row fill the file with about 19.2MB of data. This is performed for 100 files and then averaged to eliminate fluctuations. This resulted in a max speed of about 26.3 Mb/s, min speed of about 17.8 Mb/s, a mean speed of about 23.6 Mb/s, and a median speed of about 23.6 Mb/s. See Figure 3 below.



# Other Notes

The testing of accuracy of data received by the SPI transfer was done by a different program. All of the files were written correctly, with 0 mistakes recorded in the 100 files.