HW2 – Raspberry Pi Exploration

Table of Contents

[Objective 2](#_Toc474272920)

[Areas Explored 2](#_Toc474272921)

[Q & A 2](#_Toc474272922)

[1. Linux version 2](#_Toc474272923)

[2. GPU firmware version 2](#_Toc474272924)

[3. SoC for Raspberry Pi 3 2](#_Toc474272925)

[4. Multi-thread operation support? 3](#_Toc474272926)

[5. CPU frequency 4](#_Toc474272927)

[6. Communication between multiple Pi - Serial data transfer 4](#_Toc474272928)

[7. Pin control library 4](#_Toc474272929)

[8. Controlling LED with GPIO 6](#_Toc474272930)

[9. Heat 7](#_Toc474272931)

[10. Performance 7](#_Toc474272932)

[11. SD Benchmarks 9](#_Toc474272933)

[12. Read and write speed 9](#_Toc474272934)

[13. Power drawn 9](#_Toc474272935)

[Conclusion 10](#_Toc474272936)

[References 10](#_Toc474272937)

# Objective

After basic setup for Raspberry Pi 3, this report focus on the exploration of raspberry pi 3’s capabilities, functionalities, hardware and benchmarks. This report aim to explore the options the Pi provide for final project.

# Areas Explored

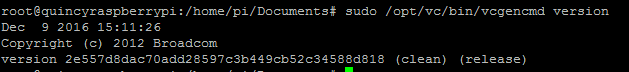
* Hardware components (HAT)
* GPIO pin function and libraries to control GPIO (PWM)
* Communication
* Power consumption
* Benchmarks

# Q & A

1. Linux version



1. GPU firmware version



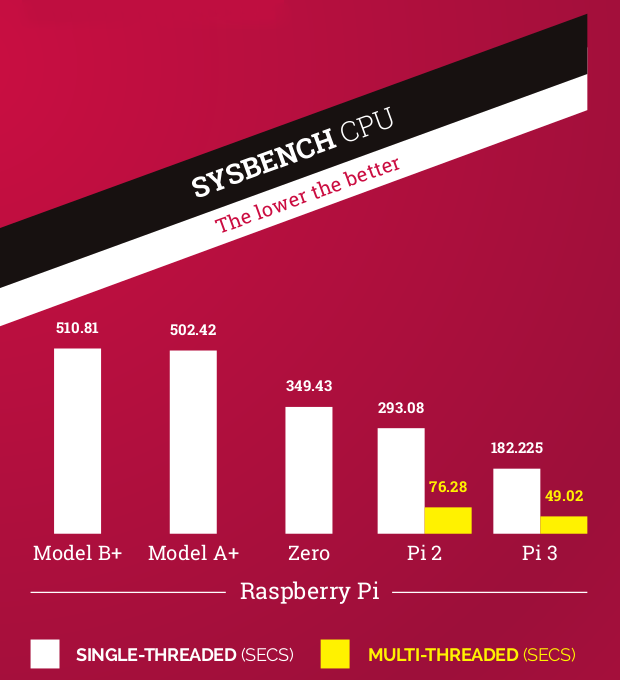
1. SoC for Raspberry Pi 3

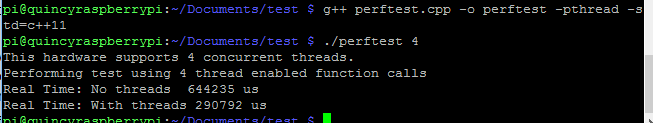
The Raspberry Pi 3 uses a Broadcom BCM 2837 SoC includes 1.2 GHz 64-bit high-performance quad-core ARM Cortex-A53 processor. It has 32kb on L1 cache, and 512kb on L2 cache.

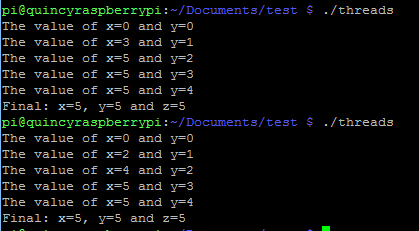
**SoC:** Broadcom BCM2837  
**CPU:** 4× ARM Cortex-A53, 1.2GHz  
**GPU:** Broadcom VideoCore IV  
**RAM:** 1GB LPDDR2 (900 MHz)  
**Networking:** 10/100 Ethernet, 2.4GHz 802.11n wireless  
**Bluetooth:** Bluetooth 4.1 Classic, Bluetooth Low Energy  
**Storage:** microSD  
**GPIO:** 40-pin header, populated  
**Ports:** HDMI, 3.5mm analogue audio-video jack, 4× USB 2.0, Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI)

1. Multi-thread operation support?

The Pi is supported for multi-thread operation, and comparing to pi 2, the performance is much better according to the graph below. Utilizing ‘perftest.cpp’ provided on chp. 6 on Molloy’s book, it shows that using multi-core to solve a problem would be much faster. On solving larger problem, multithreading provide a great tool to increase the speed. Both ‘perftest.cpp’ and ‘pthreads.cpp’ were used to test on multi-thread operations.

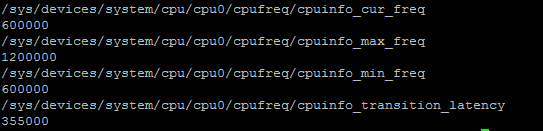






Multi-thread was used and it is shown in the example above.

1. CPU frequency



The CPU frequency can be change via overclocking. However, the application that I am planning to do does not require heavy computation speed. Therefore no overclocking was needed.

1. Communication between multiple Pi - Serial data transfer

“Serial is a data link between the Pi and one other device.” GPIO 14 and 15 (pin 8 and 10)

The Pi uses GPIO 14 (pin 8) for TXD and GPIO 15 (pin 10) for RXD.

Data is normally transmitted in 8-bit bytes with a start bit, eight data bits, no parity, and one stop bit.

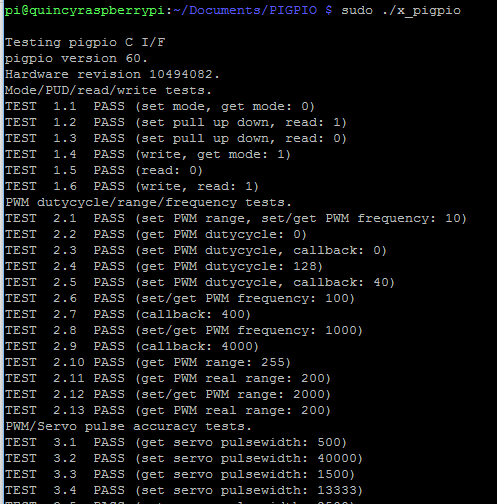
1. Pin control library

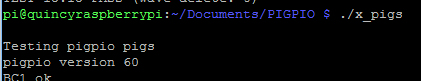
I installed both BCM2835 and PiGPIO pin library stated in homework. For PIGPIO’s test, it tested for PWM duty cycles and its pulse width. This would be useful to confirm the pin’s functionality before attaching HAT such as servo.

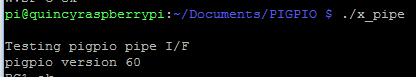
Pin Control Library

Usage

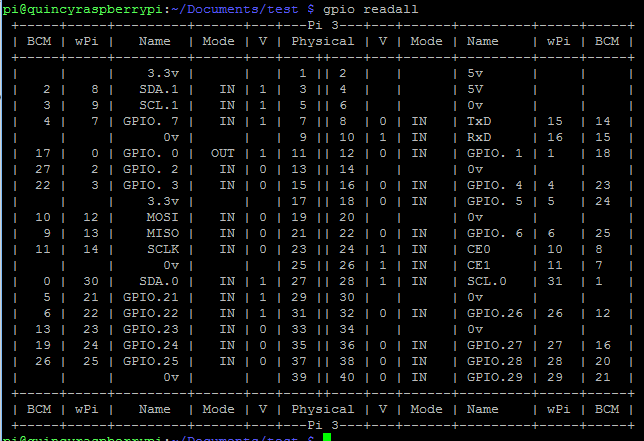
After Include <pigpio.h> in source files









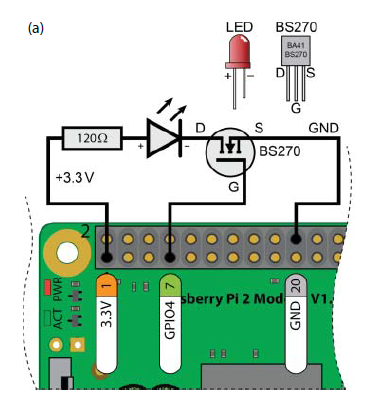


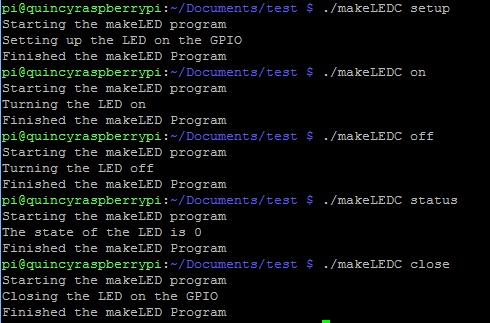
1. Controlling LED with GPIO

Following the textbook, I successful build the circuit below and used the Pi to control the LED. Furthermore, ‘LEDflash.c’ and ‘makeLED.c’ was also tested for the led functionality. It has basic control of the led and was able to turn on and off by the c program.

Pin 1 – Vdd, Pin 7 – GPIO4, Pin 20 – GND





 Sample makeLEDC operations

1. Heat

This reference (I, II and III) allow me to monitor the temperature of the RPi throughout the day. I am trying to run the raspberry pi 24/7. And the Pi3’s processor can get really hot, and damage the board.

However, I have not yet figure out how to monitor the temperature following the steps provided in the reference, more exploration is needed.

I can get the current temperature on the processor by using the command below.

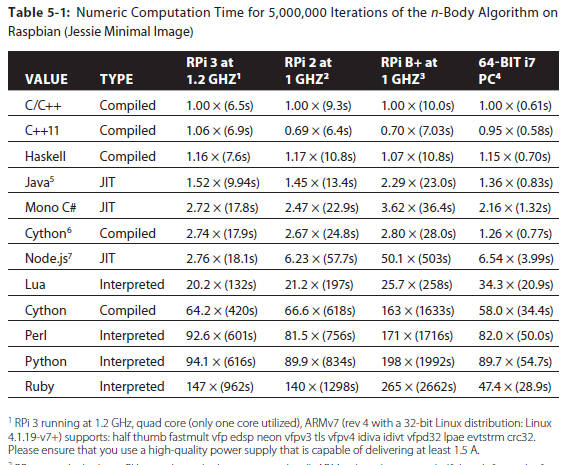
or

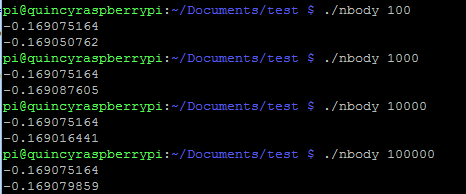


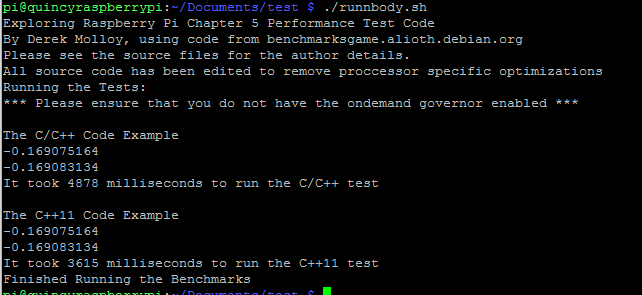
On the spec, “the LAN9512 is specified by the manufacturers as being qualified from 0 - 70°C, while the AP (Application Processor (Broadcom BCM2835), CPU of the board) is qualified from -40°C to 85°C.”

1. Performance

Following is the performance table of different language’s benchmarks on RPis. Although C or C++ is the best in performance, fast prototyping would be easier using other Object oriented language. Both ‘n-body.cpp’ and ‘n-body2.cpp’ in Cpp was the only language requirement that I tested. It shows similar result compare to the reference below.







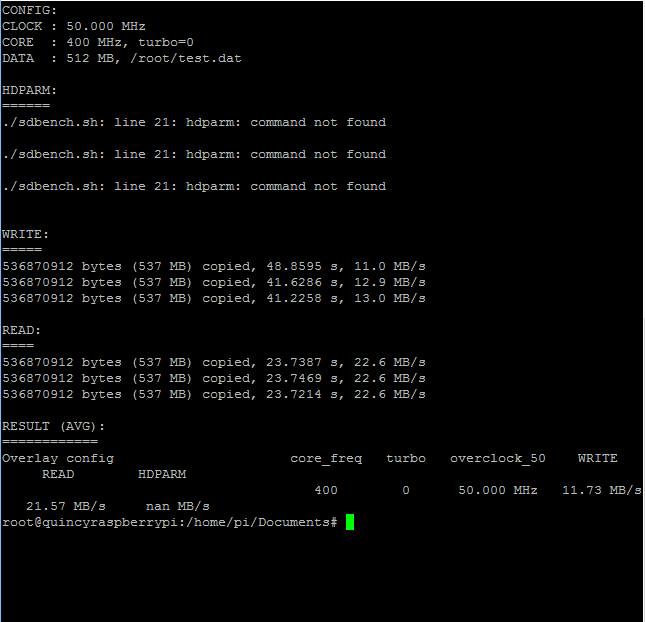
The first result of 4878 millisecounds was using ‘n-body.cpp’ and 36615 milliseconds was using ‘n-body2.cpp’.

1. SD Benchmarks

Following the SD card benchmark, the result is shown below. (Sdbench.sh)

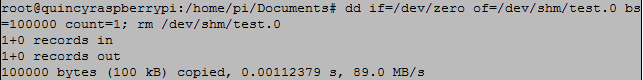
The result shown the read and write speed on the SD card, without overclocking the SD card to 100 MHz. The SD card can fully utilize by overclocking it in with command

Two different card was tested under the benchmark. The 64Gb UH1 Samsung SD card outperform the 16Gb Class 4 SanDisk. The following result shown is testing on the Samsung UH1 SD card.



1. Read and write speed

The read and write speed tested by copying a file would be testing for cache’s speed. This speed can be change by overclocking the RPi, however it might also reduce the lifespan of the RPi. Many people have shown the overclocking RPi benchmarks online. However, my application does not require overclocking.



1. Power drawn

According to the raspberry pi official cite, the 3.3 V rail can supply a maximum of 50 mA. And the 5V rail “appears to passed straight through from the USB and the current is therefore limited to whatever the USB port can supply minus the current being drawn by the board.”

|  |  |  |
| --- | --- | --- |
|  |  | Pi3 B (Amps) |
| Boot | Max | 0.75 |
|  | Avg | 0.35 |
| idle | Avg | 0.30 |
| Video playback (H. 264) | Max | 0.55 |
|  | Avg | 0.33 |
| Stress | Max | 1.34 |
|  | Avg | 0.85 |

# Conclusion

The RPi3 is an impressive device with lots of supported development language with great performance. It has many features like multi-threading and overclocking that is great for quick prototyping. In this exploration process, I was able control the LEDs through programming the RPi3. This allow me to visually see the interaction with the board. The media with GPIO pins was a very low level programming language, and it was a bit difficult to understand and write working program. The UH1 SD card that I am using for my raspberry provided promised file read and write speed. However, as the iteration of read and write to the SD card, and the operations increase on the RPi3, the temperature measured on the Pi has slowly increased, but it was well under the safety operation range. Measuring the temperature on the Pi was the main part of this exploration, as I am trying to set the RPi on 24/7. The power draw on the RPi on idle was 0.3A. And for the application that I am doing, the RPi would be on idle over 90% of the time. If I decided to use batteries instead of wall outlet, the idle current of 0.3A should allow the application to run for a sufficient amount of time.

# References

1. <http://www.erijas.com/monitoring-cpu-temperature-raspberry-pi/>
2. <https://www.cyberciti.biz/faq/howto-linux-unix-start-restart-cron/>
3. <http://lazydroid.com/2013/05/raspberry-pi-monitoring-cpu-temperature-with-rrdtool/>
4. <http://abyz.co.uk/rpi/pigpio/download.html>
5. <https://www.raspberrypi.org/documentation/configuration/device-tree.md>
6. <https://github.com/stripcode/pigpio-stepper-motor>
7. <https://www.raspberrypi.org/magpi/raspberry-pi-3-specs-benchmarks/>
8. <https://www.raspberrypi.org/forums/viewtopic.php?f=63&t=144493>
9. <http://www.nmacleod.com/public/sdbench.sh>
10. <http://raspberrypi.stackexchange.com/questions/5477/why-is-my-pi-running-at-700mhz-all-the-time>
11. <http://raspberrypi.stackexchange.com/questions/51615/raspberry-pi-power-limitations>
12. <https://www.raspberrypi.org/help/faqs/#powerReqs>
13. <http://raspberrypi.stackexchange.com/questions/51615/raspberry-pi-power-limitations>
14. <https://www.raspberrypi.org/help/faqs/#performanceOperatingTemperature>
15. <https://www.raspberrypi.org/forums/viewtopic.php?f=91&t=34994>
16. Exploring Raspberry Pi Interfacing to the Real World with Embedded Lunux – Derek Molloy