VRIJE
UNIVERSITEIT
BRUSSEL

In order to be awarded the Master's Degree in industrial science:
Electronic-ICT with specialization embedded systems

Benchmarking single board computers for embedded systems

Zineddine Wakrim

August 26, 2018

Jan Lemeire

Engineering science

Abstract

Acknowledgements

Contents

List of Figures

List of Tables

1 Introduction

This thesis will benchmark vision algorithms on several single board computers. The purpose of the thesis is to check wich single board computers are optimal for real time image processing.

Vision is a hype for the moment in the IT world, autonomous car, camera with number plate reconision,...

With the advent of the Arduino a lot of hoobbies want to put vision in robots or in other application. Because a microcontroller like the one on the Arduino is not powerful enough to perform real time video processing It is mandatory to combine it with a more powerful computer or a single board computer. in severals application the single board computer will send the image to a more powerful computer or server and it is this one that will perform the imageprocessing. For our application all the image processing will be done on the single board computer.

Two users were defined: students or person that are not ease with programming. In this case the supposition is that the user will use python. The second user is a person that is more alaise with coding and that will use or C. it will be interesting to see if there is a big difference between python algorithms and C. For both uses cases we want to find the optimal choise of embedded computer.

Last but not least is also the power consumption. Because most of the application like robots needs a battery to be powered it will be interesting to check the different power consumption during the test. Concluding, the most important criteria for the evaluation are processing capacities, easy-to-use and energy consumption.

The benchmark suite will be made generic such that other tests can easily be added.

2 Single board computers

2.1 board comparison

This thesis will compare 5 boards. The chosen boards all run an arm architecture. The boaérd are selected because of they popularity, price(depending on use case) and power consumption:

- **Raspberry Pi 3**: The most famous single board computer of our list. The cost of the board is around $35. It is possible to run several operating system on it like Linux or Windows 10 IOT core. [1]
- **Udoo**: The Udoo Dual is a computer that can be used with Android and Linux. It is a powerful prototyping board because it contains a processor and a Arduino Due micro-controller. To summerize the Udoo we can say that it bring the best of an Raspberry Pi and an Arduino togheter in-to one single board. **maksimovic2014raspberry** On the market we can find three type of Udoo that combine a processor and an Arduino. In this thesis we will discuss the Udoo dual. The dual is the mid-range Udoo and the cost of it is around $115 [2].
- **Odroid Xu4**: The Odroid X4 is and powerful Single board computer. It combine two processor a Samsung Exynos5422 Cortex A15 (2 GHz) and a Cortex A7 Octa core CPU (1.4 Ghz). This two processors are combined with 2 GB of RAM. **abdullah2016position** The board can run several flavors of Linux. The computing power of the X4 seems to be seven times faster than the Raspberry Pi3. The Xu4 can be found on the market at the price of $59. [3]
- **Dragonboard 410c**: The Dragonboard 410 c is a single board computer basd on the mid-tier Qualcomm® Snapdragon™ 410E processor. It can run Windows 10 IOT core and Linux(Andoid, Debian). The price is around $75. [4]
- **Nvidia jetston tx2**: Jetson is an NVIDIA® high performane computer. it is low-power and it is ideal for embedded compute-intensive projects like deep learning or computer vision. [5] This thesis will compare the Jetson TX2 Developer Kit with other boards. On the market the TX2 Devloper kit can bi found for $599.

  Table 2.1 compares the technical specs of all the cited boards.

	udoo dual	**raspberry pi 3**	**odroid Xu**	**Dragonboard 410c**	**NVIDIA jetston TX2 devlopper kit**	**Pixycam**
cpu	NXP i.MX 6 DualLite Atmel SAM3X8E (Arduino Due)	ARM Cortex-A53 Quad-core	Samsung Exynos5422 Cortex™A15 2Ghz + Cortex™-A7 Octa core CPUs	ARM Cortex-A53 Quad-core	HMP Dual Denver 2/2MB L2 + Quad ARM A57/2MB L2	NXP LPC4330
# cores	2	4	8	4	6	2
clk(GHz)	1	1.2	2	1.2	2	0.204
ram(GB)	1	1	2	1	8	0.000264
memory(GB)	unlimitted (sd card)	unlimitted (sd card)	unlimitted (sd card)	unlimitted (sd card)	32	1M
ipc(instruction per cycles)	1	1	1	1	1	1
GFLOPs	2	4.8	16	4.8	12	0.408
Power supply (DC V)	6	5	5	6.5	19	
GPU	yes Vivante GC 880 + Vivante GC 320	yes broadcom videocore 4	yes mali T628	yes Qualcomm Ardeno 306	yes NVIDIA Pascal, 256 NVIDIA CUDA Cores	no
opencl/cuda	maybe	no	yes	yes	yes(Cuda)	
core clk (mhz)	400	250	600	450		
pixel/rate(Mpixel/sec)	80	1000				
operating System	Linux debian	linux windows10 debian	linux debian	linux windows debian	linux debian	no
camera plug	yes	yes	yes	yes	yes	
python /opencv	yes	yes	yes	yes	yes	
price(€)	110	40	75	75	599	
arduino compatible (connector for Arduino)	yes	no	no	yes	no	yes

Table 2.1: Comparison board table [6] [7] [8] [5]

2.2 Working and first run experience

A lot of information is available on the internet for most of the boards. Some boards have a bigger community like the Rasperry Pi. My personal opinion is that with the Raspberry Pi everything works directly. The most difficult board is the Dragonboard. small community, slow board and less memory. I was not able to install Opencv on this board.

An other point is the fact that all the board used a the same power supply socket except the dragonboard. I suggest to buy the boards with they power supply. As example for the Odroid xu4 i tried an other power supply and on some point it was not working. This is because the Dragonboard needs a maximum of 4 ampere to work. 5V 4A is not an common power supply to find.

About the other boards the installation documents are very complete.

3 Related Work

3.1 Existing benchmarks

On the market there are a lot of single board computers. They all have different capabilities and cost.[9] Because this big diversity the question that we have to ask is how should we decide which board is right for our needs? Benchmarkt [9] created two categories: Boards under $100 and boards over the $100. The boards under the $100 is more for DIY maker. With this kind of boards it is possible to do a lot be the boards are limited in computing power. Over the $100 the boards are more powerful and built for more specific purpose like machine vision and robotics.[9]

The result of benchmark [9] can be found in appendix .2. The conclusion of this benchmark is that each single board computer comes with his own capabilities, and one isn't perfect for all applications.

An other interesting benchmark is the one from Phoronix.com. The Phoronix Test Suite is an benchmarking platform available for Linux, Solaris, Mac OS X, and BSD operating systems. [10] Benchmark [11] tested several single board computers, some similar two boards are similar as the one tested in this thesis, The jetston TX2 and the Raspberry Pi 3. The benchmark tested performance and performance-per-dollar. In all the performance benchmark the jetson TX2 did it better than the other boards. About cost the Jetson is the best one when using some graphic [11]

3.1.1 Running existing benchmark

Eight benchmarks were tested. The same as[11]

FFTW

The FFTW benchmark is a C subroutine for computing DFT in one of more direction. Figure 3.1 show the result of the benchmark. The Jetston TX 2 scored the best with 1697 Mflops. The Udoo board was the worst one.

Figure 3.2 shows the FFTW benchmarkt devide by the price of the board. The Udoo board is still the worst one. The best one is the Odroid xu4 that is more than 4 times better than the second one(Rasperry pi 3).

Figure 3.1: FFTW benchmark.

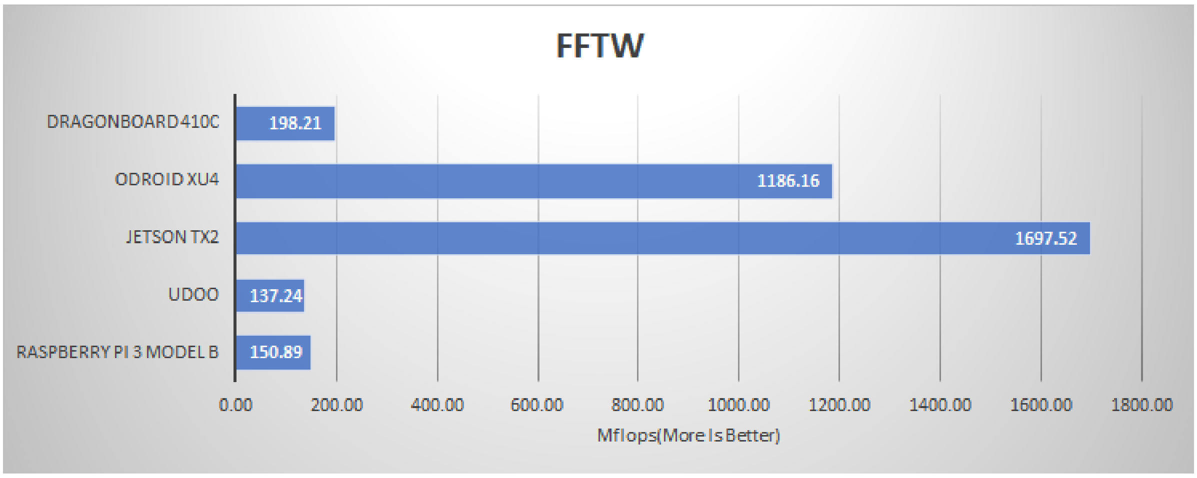
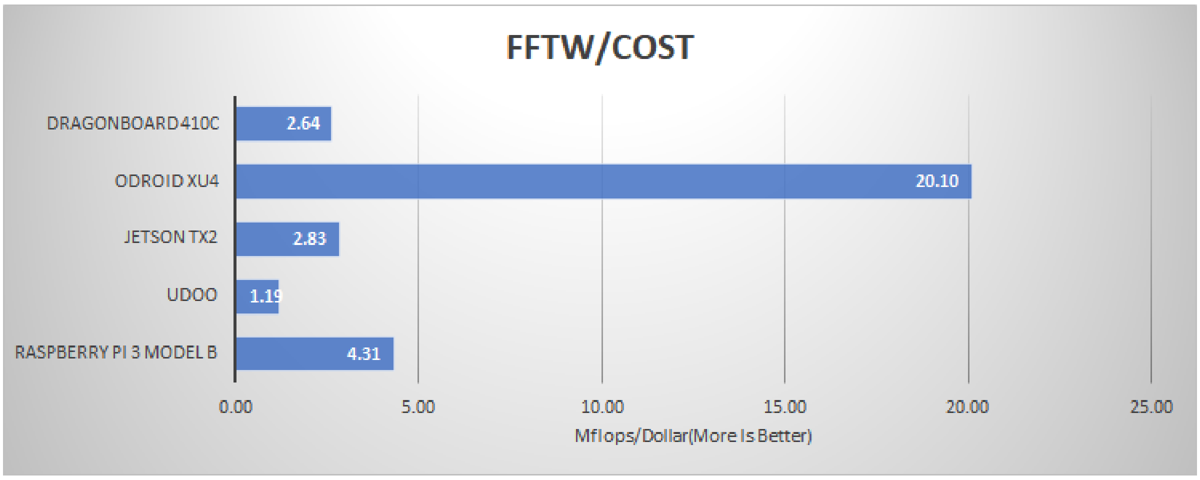


Figure 3.2: FFTW/Cost benchmark.



John The Ripper

John The Ripper algorithm is a password cracker. The odroid Xu4 scored the best in the benchmark as well as the cost benchmark . The worst one is the Udoo board(3.3). The worst value for money board is the Jetson TX2 (3.4).

Figure 3.3: John The Ripper benchmark.

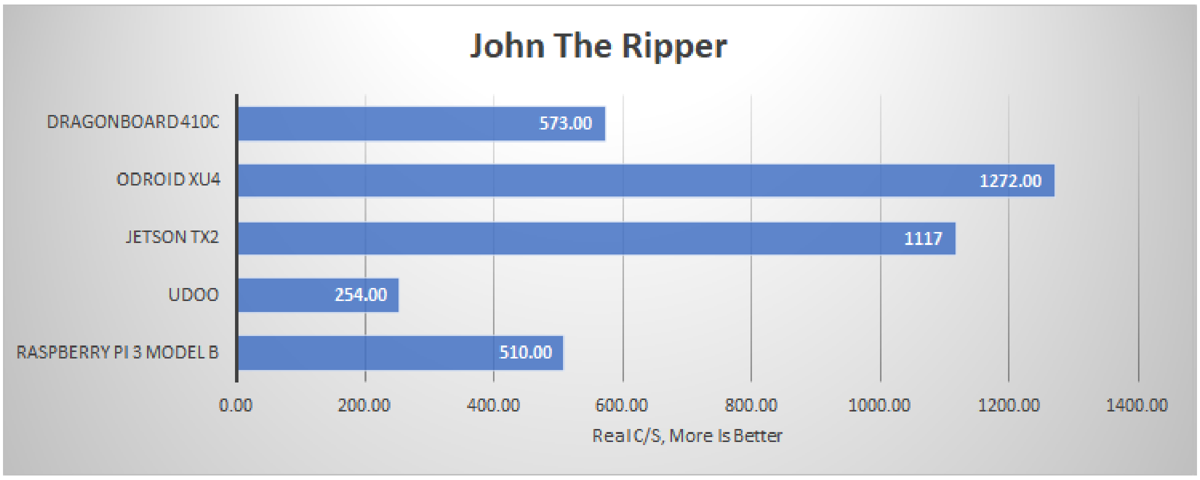
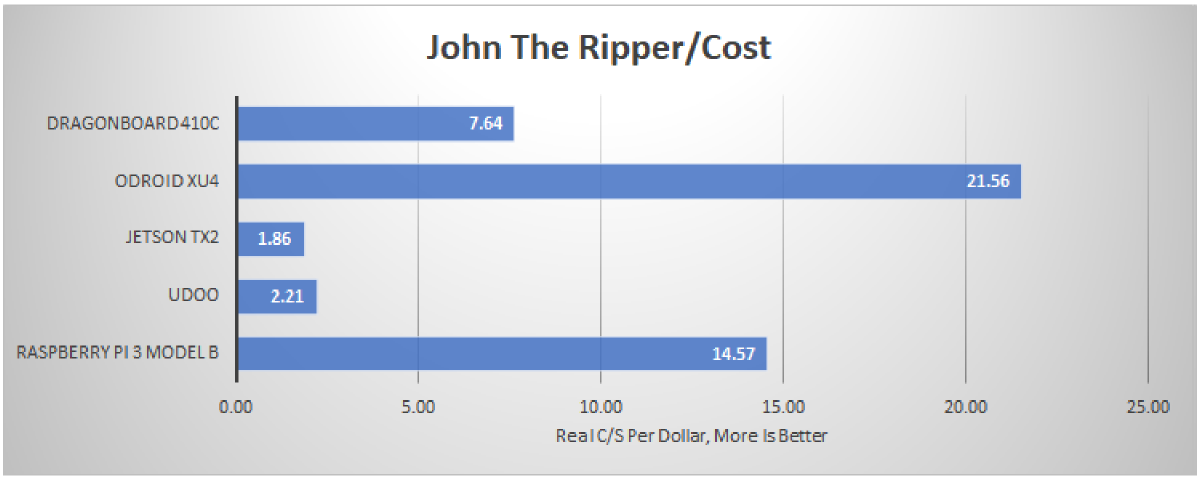


Figure 3.4: John The Ripper / Cost benchmark.



C-ray

The C-Ray is a raytracer it is used to test the floating-point CPU performance. The one used on this benchmark is multi-threaded(16 threads per core). It will generate a 1600 x 1200 image and will shoot 8 rays per pixel for anti-aliasing. [12]

The Jetson TX2 scored the best on the benchmark but the Odroid Xu4 Has the best price value score.

Figure 3.5: C-Ray benchmark.

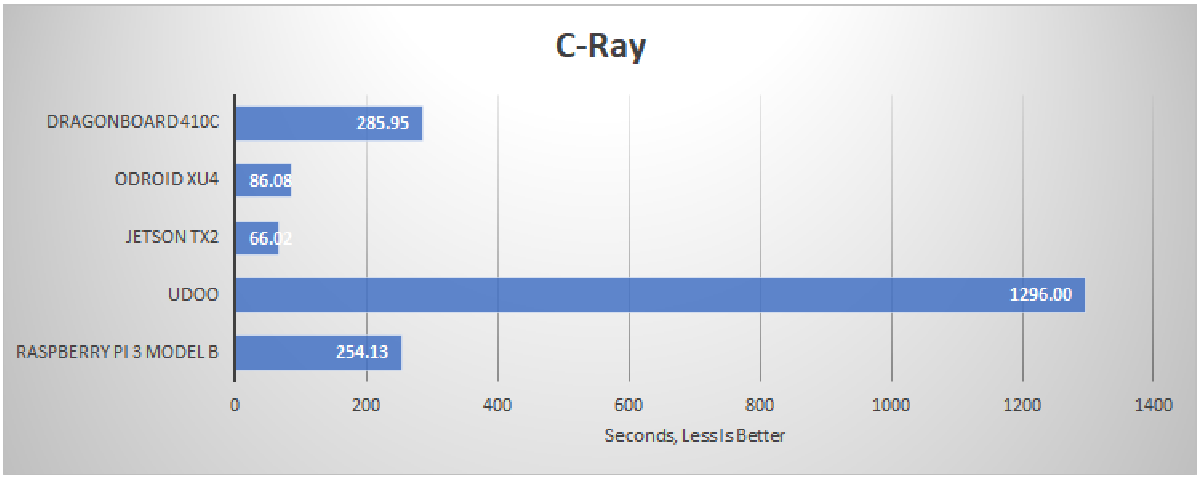
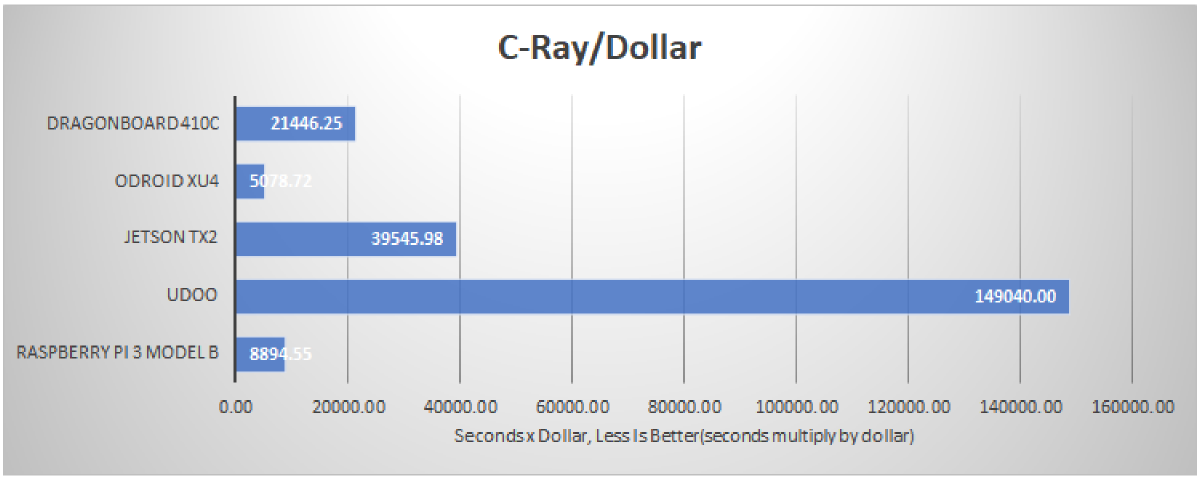


Figure 3.6: C-Ray/Cost benchmark.



Smallpt

The Odroid Xu4 scored the best it has the best result in for the value benchmark but the jetson has the best performance.

Figure 3.7: smallpt benchmark.

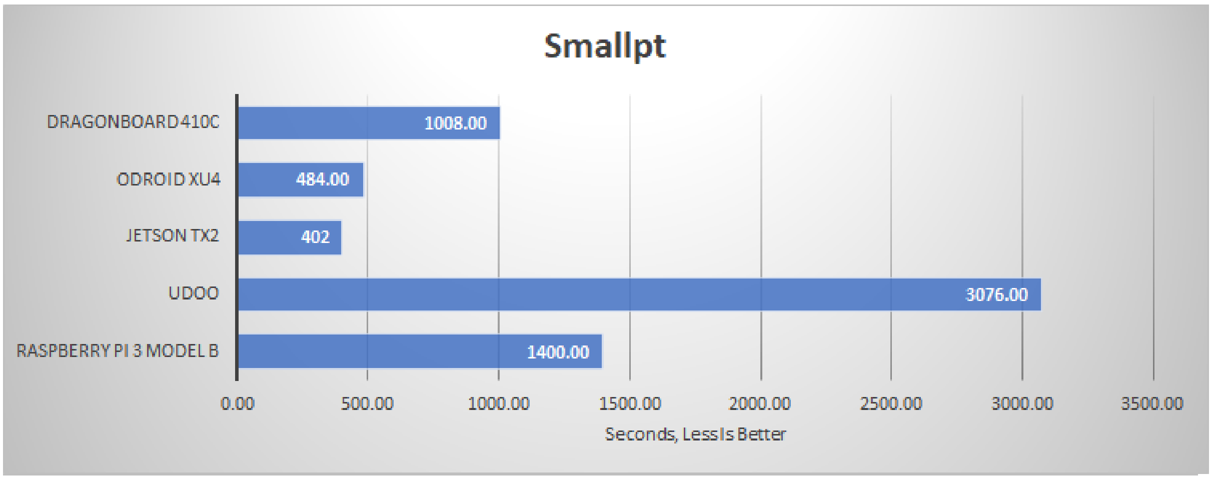
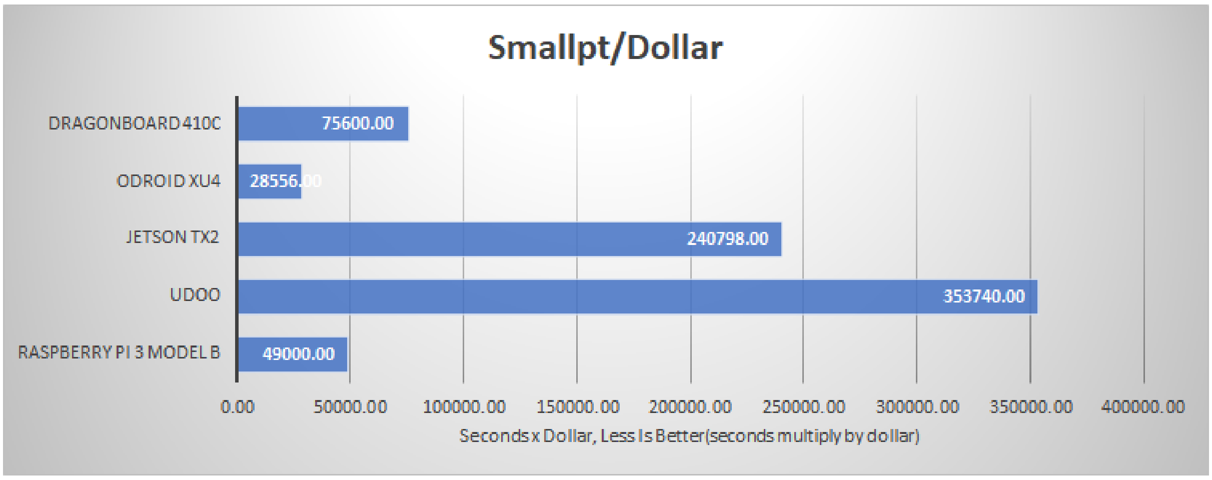


Figure 3.8: smallpt/cost benchmark.



FLAC Audio Encoding

Figure 3.9: flac audio benchmark.

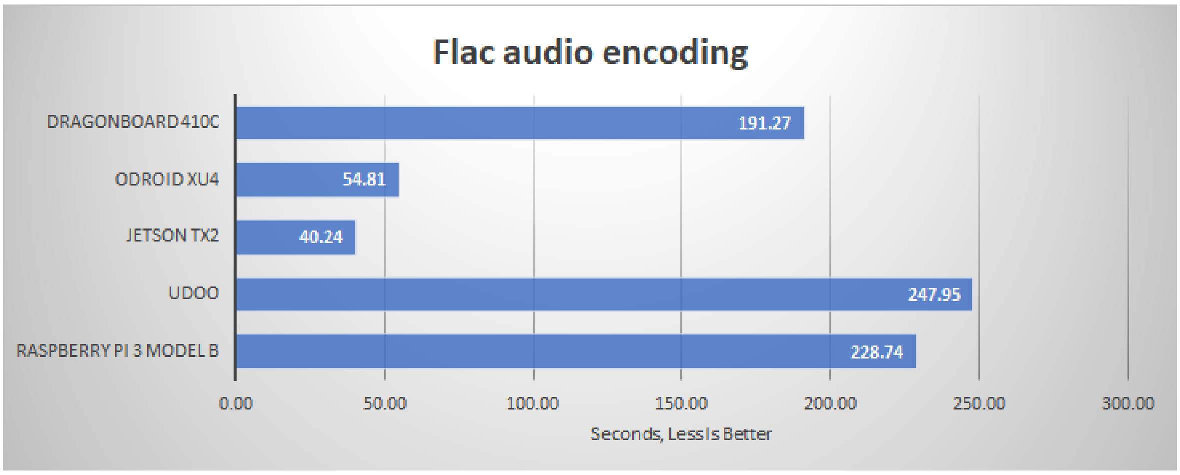
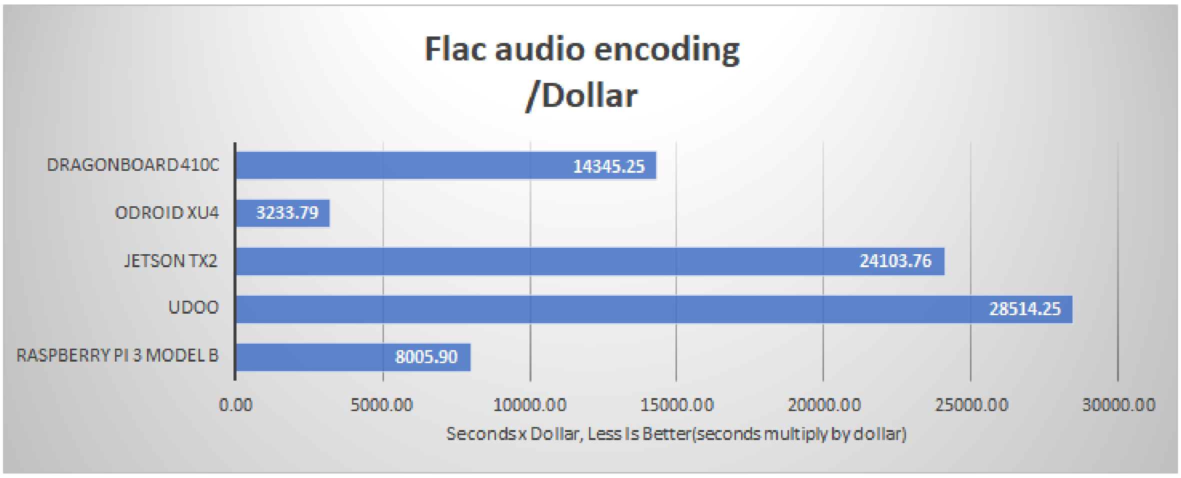


Figure 3.10: flac audio /cost benchmark.



Redis

Figure 3.11: Redis(Set) benchmark.

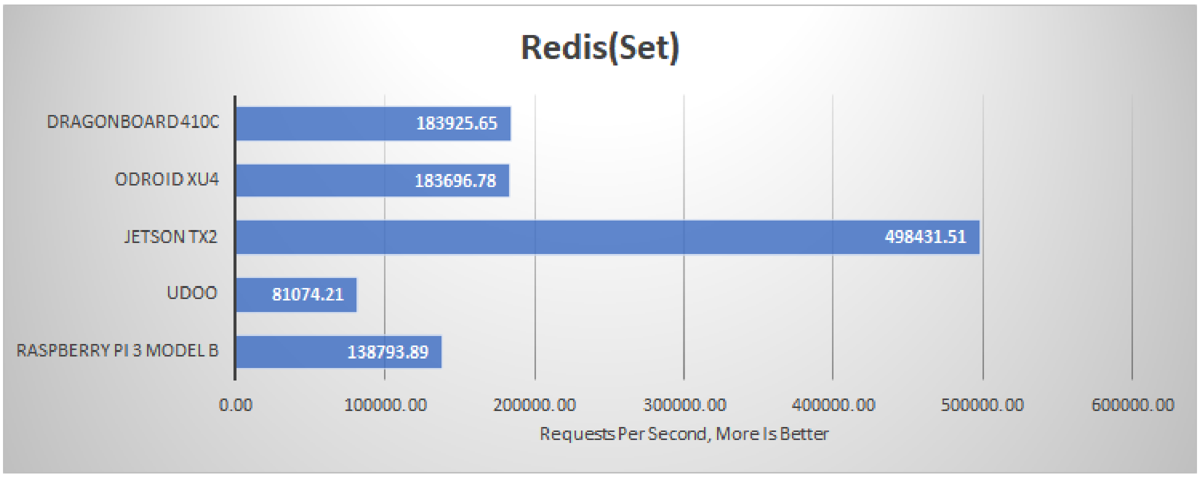


Figure 3.12: Redis(set)/cost benchmark.

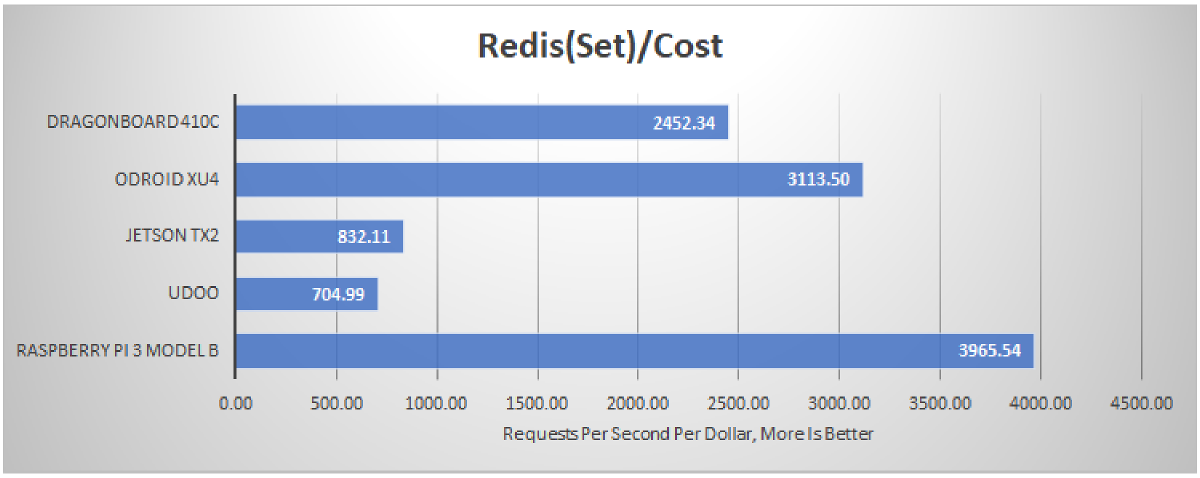


Figure 3.13: Redis(Get) benchmark.

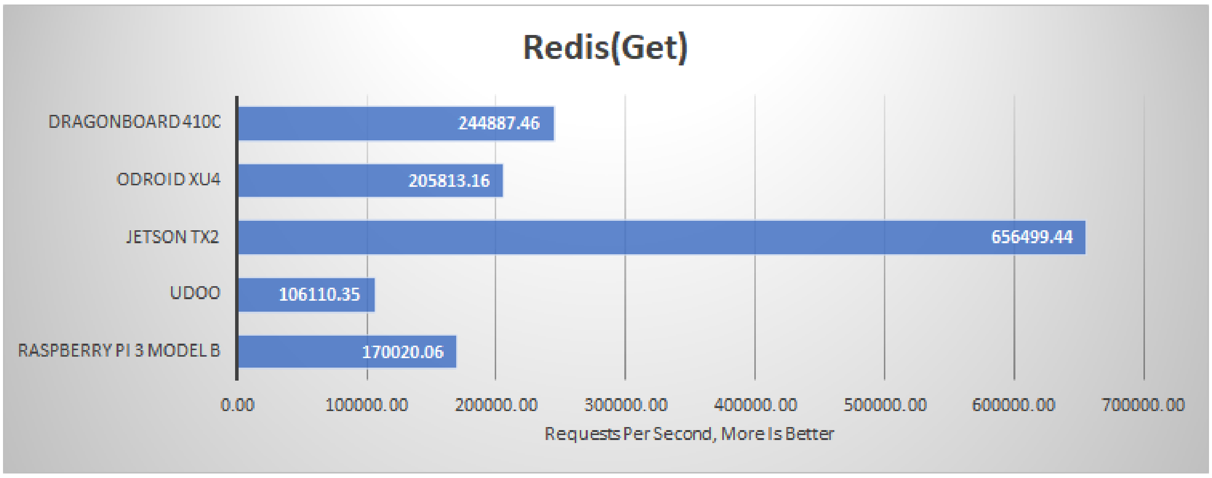
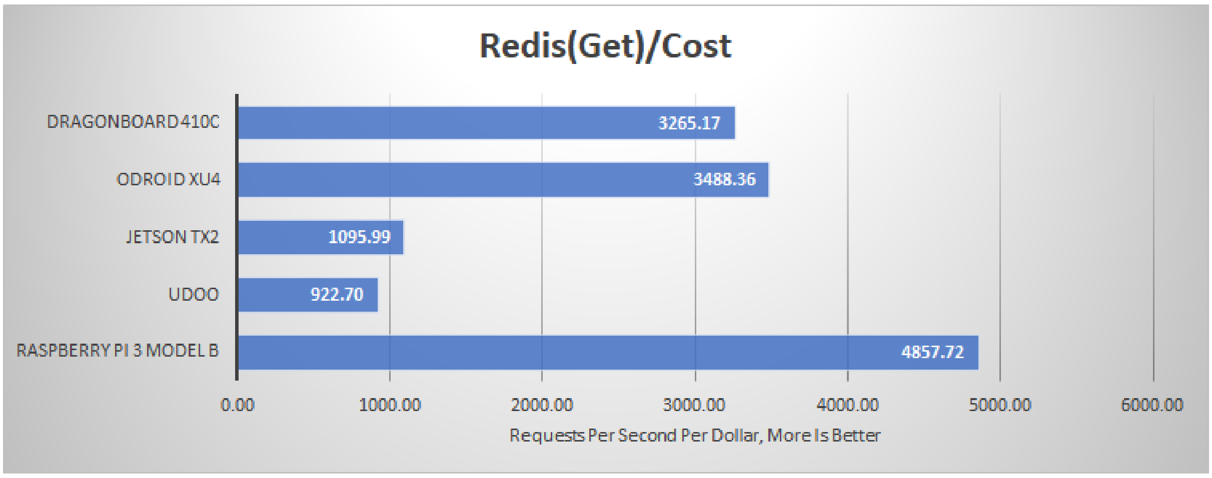


Figure 3.14: Redis(Get)/cost benchmark.



Himeno

Figure 3.15: Himeno benchmark.

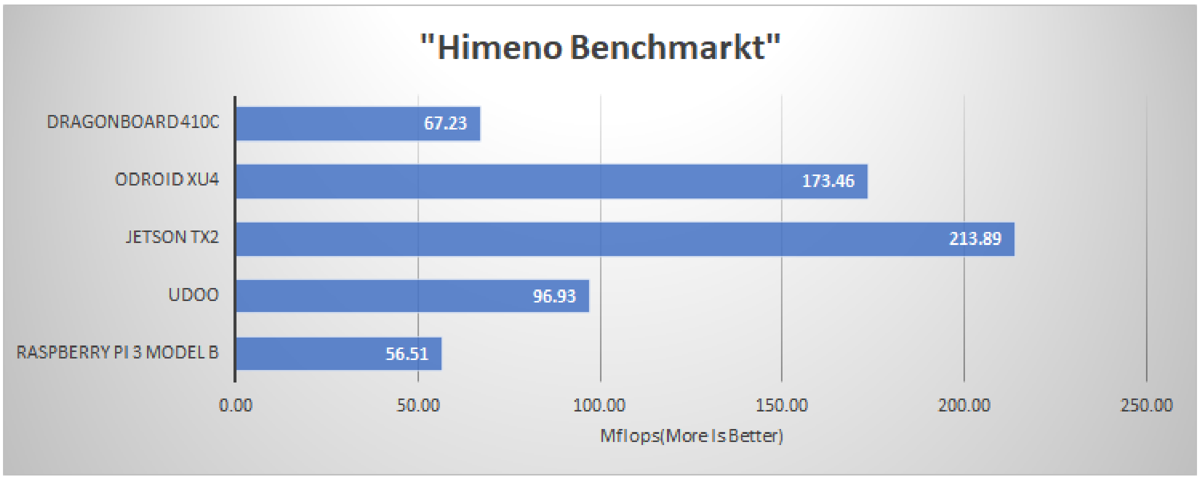
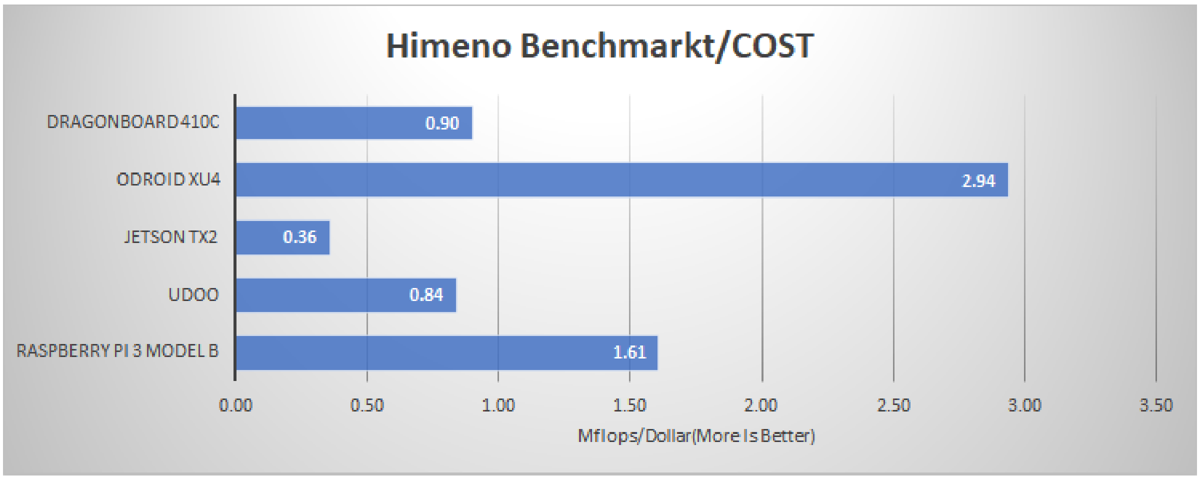


Figure 3.16: Himeno/cost benchmark.



conlusion

In all the processors benchmarks the best board is without surprise the Jetson TX2. The reason is because this board is more powerful it has 8Gb of rams compared to the Odroid Xu4 it has 4 times the amount of ram memory. The best price value board is the Odroid Xu4 for performance. Its scores much better than all the other board for the Performance benchmarks.

For the Redis benchmarks wich is a system benchmark. The Jetson still score the best but on the Rasperry Pi3 score the best in term of price value. All the result are available on following link : https://openbenchmarking.org/result/1805273-FO-1805279FO65

The user has do decides if he want to save money and work with a board like a Raspberry Pi 3 or get maximum performance and work with a jetson board.A good compromise is the

Odroid Xu4. This board looks powerful and is low cost.

About vision we can find a lot of benchmarks that compares algorithms to do some specific task like face-detection. There are also more general benchmarks that compares single boards computers (3.1), but there is no benchmark that will compare boards on how they are good to do real time image processing.

4 Use cases

4.1 Pixy cam

The pixycacam is an vision sensor. This board can help an arduino or simal board to do fast image processing. the pixicam will process the image and send only useful information to the microcontroller. The frame rate is 50Hz and it can use several interfaces to send the inforamtion:UART serial, SPI, I2C, USB, or digital/analog output. In our use cas the pixi will do the iamge processing while the micronctorller(Arduino) can use his CPU for other tasks.[13]

Technical specs:

- Processor: NXP LPC4330, 204 MHz, dual core
- Image sensor: Omnivision OV9715, 1/4”, 1280x800
- Lens field-of-view: 75 degrees horizontal, 47 degrees vertical
- Lens type: standard M12 (several different types available)
- Power consumption: 140 mA typical
- Power input: USB input (5V) or unregulated input (6V to 10V)
- RAM: 264K bytes
- Flash: 1M bytes
- Available data outputs: UART serial, SPI, I2C, USB, digital, analog
- Dimensions: 2.1” x 2.0” x 1.4
- Weight: 27 grams

4.2 Limit of Pixy

The Pixy can only be used for color detection not for object detection. The second problem is that the Pixy can only detect flashy colors like red yellow blue (figure: 4.1). If the object has any rebound color the Pixy will not detect the object (figure:4.2). Last point is that the pixy will not detect white colors (figure: 4.3).

Figure 4.1: Red color detection.

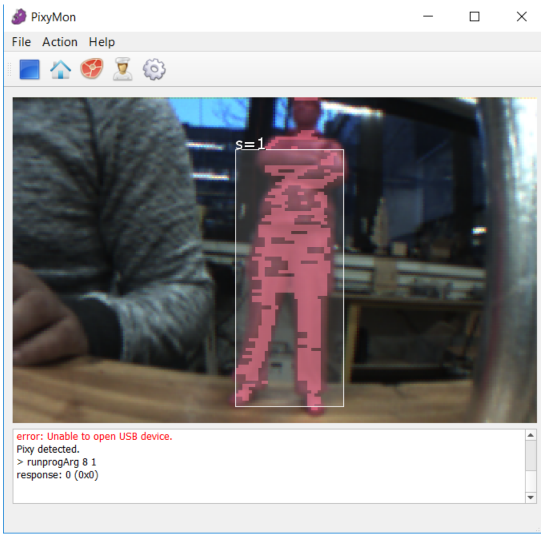


Figure 4.2: Rebound object detection.

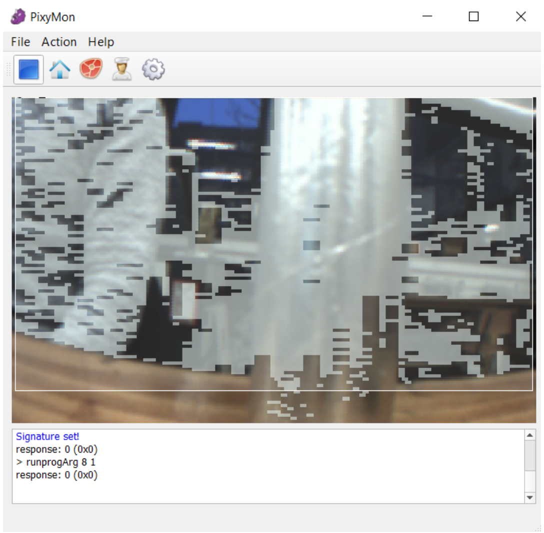
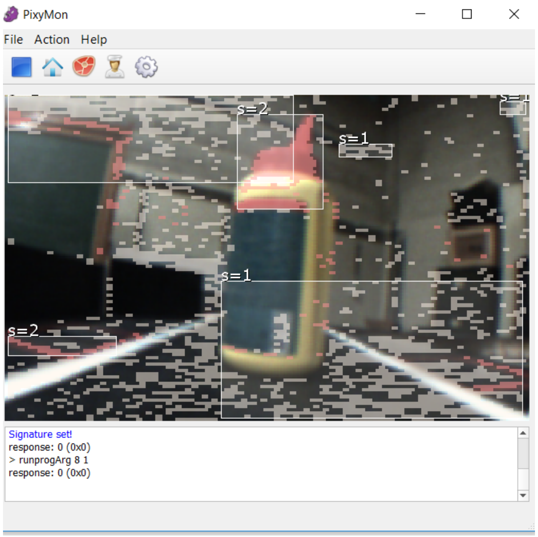


Figure 4.3: white color detection.



5 Benchmark suite

5.1 Algorithms

Several algorithms were written to do the benchmark. Each algorithm is written in python and C++. For this proof of concept 4 algorithms were written: face detection, convolution (filter 5 by 5), convolution (filter (20 by 20), pixel operation. They all use openCV. The algorithms where chosen because some are memory limited and other are compute limited.

Each algorithm runs for 20 seconds and is tested 4 times with a different resolution. The suite will measure the effective calculation time of the image-processing,the current and so later we can easely calculate the power and the usage of the CPU.

5.2 Framework board side

Before running the benchmark several libraries has to be installed.

- OpenCV : This library is used for all the vision algorithms. Not easy to install on single boards computers. The installation of OpenCv will take most of the time.
- Serial: Used to read serial port. With this library the benchmark is able to monitor the current consumption.
- psutil : psutil (python system and process utilities) is a library for retrieving information on running processes and system utilization. With this library we will be able to monitor the CPU utilization.
- subprocess: with this library the benchmark is able to spawn new processes and connect their input and outputs.
- jsonlines: This library help the benchmark to save in one json file several jsons. Each line is a json. This technique is used in a lot of big data application.

The main code for the benchmark is the benchmark.py algorithm. This will control the benchmark suite. Benchmark.py will first read two txt files that contain the terminal command to launch a algorithm. One text file for the python algorithms and an other for the C++ ones. Benchmark.py will read the terminal command and with the help of subprocess library it will run run the terminal command. Five second before and 5 seconds after running the command it will measure the idle mode of the single board computer.

Approximately each second the benchmark.py will measure the CPU utilization, temperature if possible and current. Each second it will provide a json.

Example:

```
{"node": "raspberrypi", "Temp": {}, "counter": 12, "percent_cpu": [55.9, 58.3,
    51.5, 51.0], "Current": 0.684, "algoritme": "facedetection_python", "time":
    "2018-08-24 10:13:06", "resulotion": 320}
```

On the end the benchmark provide one main json file and for each algorithm a json.
The output of the algorithm code is a jsonl with several keys and value:

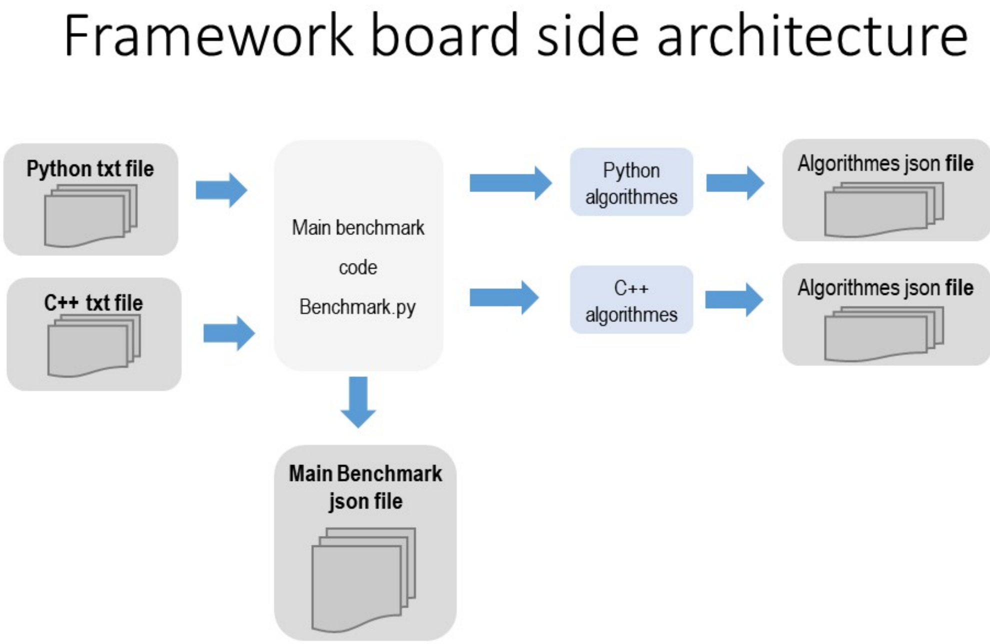
- ratiCalTime: The ratio between the effective calculation time of the algorithm divide by the total time of the algorithm.
- type: python or C++
- Second: each second a jsonline is created this can be use as an unique id
- algorithm: name of the algorithm
- fps: The frames per second at that moment
- resolution: the resolution of the algorithm
- node: computer's network name (supposed to be unique)

example:

```
   {"ratioCalTime": 8.937491320338326, "Type": "Python", "Second": 14,
      "algoritme": "Convolution_python", "fps": 24, "resolution":
      176,"node":raspberrypi}
```

Each algorithm create an json file with several jsonlines.
figure 5.1 show schematically how the benchmark works.

Figure 5.1: Schematic board framework.



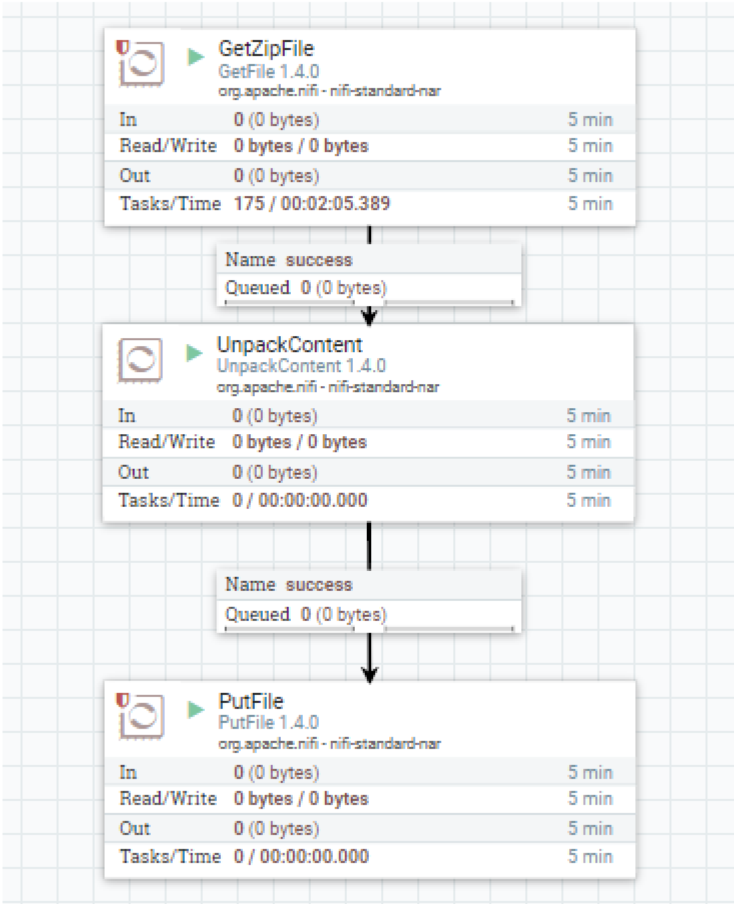
5.3 Framework Server side

once the benchmark is done the json are zipped and the zip file need to be uploaded on the server side. To do this a server on Google cloud was created.

On the server side three main tools were installed Nifi, Zeppelin and PostgreSQL.

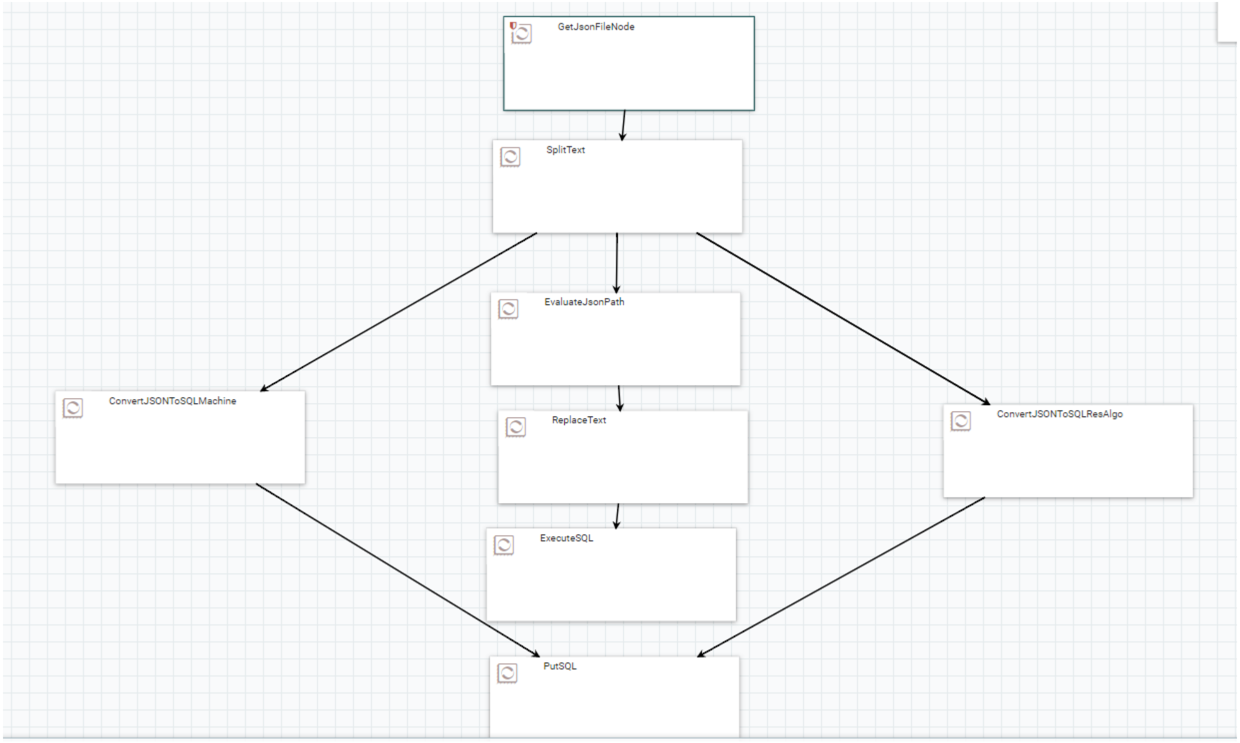
- Apache Nifi: Nifi is a software from Apache build to automate dataflow between systems. In this thesis Nifi is used to transform the data from the zipfile to a SQL query and put the data in a database. The reason of the chose of Nifi is because it is simple to install and run. No code is needed to transform the data either for putting the data in the database. To transform the data the nifi UI is used. The UI provide some blocks that does the job automatically. Figure 5.2 shows the nifi process to get the zip file and unzip it.

Figure 5.2: Get and unzip nifi.



Once the data is unzipped and moved to an other folder the second part of process can begin. Figure 5.3 is a screen capture of the nifi process to transform and add each json line in the database.

Figure 5.3: Transform and put in database.

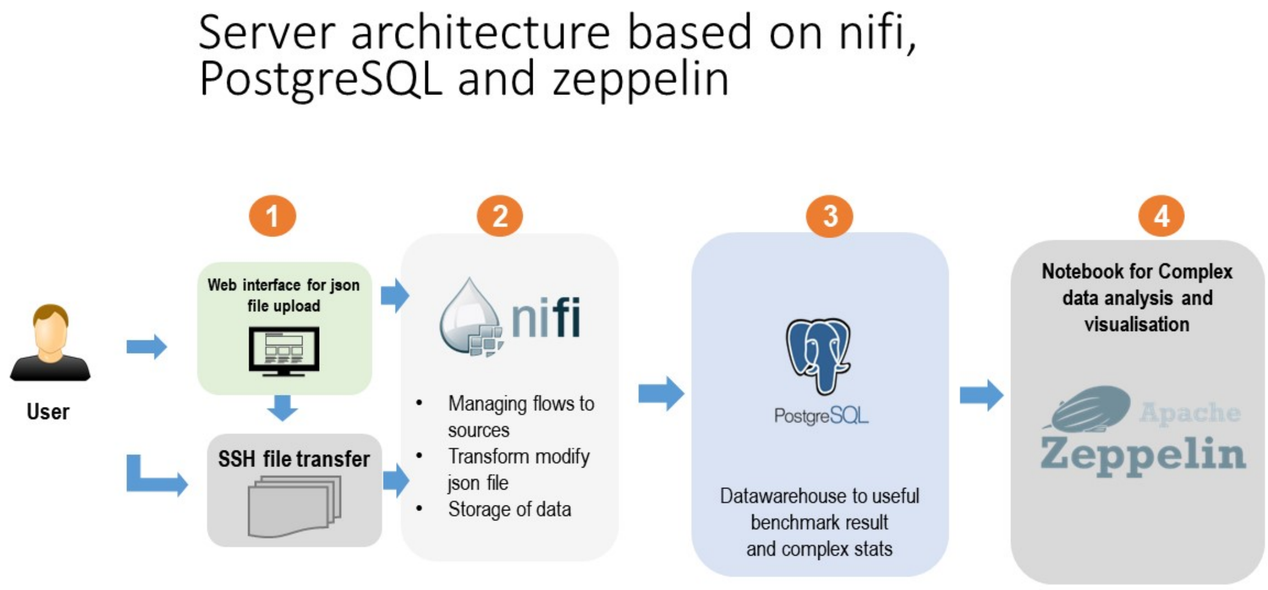


- PostgreSQL : PostgreSQL is an database system that extend the SQL language. It is open-sources like all the other softwares used in this thesis. The advantage of PostgreSQL is that it is good for analytics and data mining. Three tables where created "machines", "resBenchmarkt" and "resAlgo". The first table is a listing of all the tested boards. "resBenchmarkt" contains the result of the benchmark.py code and "resAlgo" contains the result of each algorithm.

- Zeppelin: Zeppelin is also a software devlopped by Apache. Zeppelin is a web-based notebook a little bit like Jupyter. The main difference is that zeppelin support different languages and with Zeppelin it is easy to create a dashboard and share it. In this thesis Zeppelin is used to create the tables in the PostgreSQL database and is also use to query the database and visualize the data.

For uploading the file the user can use SSH transfer or more simple if the user has access to the console platform on the Google cloud website it is possible to upload the file via the a web browser. once this is done the rest goes automatically. Figure 5.4 shows the server architecture.

Figure 5.4: Schematic server framework.



5.4 Current measurement

To measure the current a Sparkfun 70C multimeter is used. The multimeter chip is a FS9922-DMM4 from Fortune Semiconductor. This chip is a low power Analog to digital converter. To send data via usb the multimeter use an CP2102 USB to Serial converter. (source a rajouter plus tard

The datasheet of the FS9922-DMM4 (p31) descripe how to read the bytes coming from the serial port. By using the Arduino IDE it was simple to get the right baudrate, 2400 baud.

If a serial port is given as argument to the benchmark main code, the code will try to read the value coming from the serial port. (rajouter un example)

On some boards it was not possible to read the current cause the cp210x(USB to Serial convert) driver was not activated in the kernel. An improvement will be to activate the driver on all the single boards computers.

5.5 User guide

The first step before running ”benchmark.py” is to install OpenCV. The best way to do that is to search how to install opencv for a specific single board computer.

If the user just want to test python OpenCV algorithms then it is possible to install opencv with the following command.

```
pip install opencv-python
```

Second step is to install all the python libraries to monitor the single board computer. This can be done easily by running a pip install command. example:

```
pip install serial
```

The third step is to compile C++ code. To do that the following command can be used. In this exaple the pixel transformation code will give and output ”pxltransform”.

```
g++ -std=c++11 pixeltransform.cpp -o pxltransform `pkg-config --cflags --libs
    opencv`
```

ones all the C++ algorithms are compiled the user can launch the benchmark. The ttyUSB0 is the serial port used by the multimeter. This argument is not mandatory.

```
python benchmark.py ttyUSB0
```

5.5.1 Adding algorithm to benchmark

To add a new code to the suite the user should modify the text file. As example to add new python code the user should open the ”python_algorithm.txt” and add the command line for running the benchmark. the main code benchmark.py will automatically run the command line and monitor the singe board computer. The user is free to create some output in the added algorithm.

6 Results

7 Conclusion

Bibliography

[1] *Raspberrypifaqs-frequentlyaskedquestions*, https://www.raspberrypi.org/help/faqs/, (Accessedon05/02/2018), (cit. on p. 11).

[2] *Dualquad-shop–udoo*, https://shop.udoo.org/eu/quad-dual.html?___from_store=other&popup=no, (Accessedon05/02/2018), (cit. on p. 11).

[3] *Odroid—hardkernel*, http://www.hardkernel.com/main/products/prdt_info.php, (Accessedon05/02/2018), (cit. on p. 11).

[4] *Dragonboard410cbyarrowdevelopmenttools—embeddedsystemdevelopmentboardsandkits—arrow.co* https://www.arrow.com/en/products/dragonboard410c/arrow-development-tools, (Accessedon05/02/2018), (cit. on p. 11).

[5] *Jetsonfaq—nvidiadeveloper*, https://developer.nvidia.com/embedded/faq, (Accessedon05/04/2018), (cit. on pp. 11, 12).

[6] *Samsungexynos5420octasoc-notebookcheck.nettech*, https://www.notebookcheck.net/Samsung-Exynos-5420-Octa-SoC.103633.0.html, (Accessedon05/04/2018), (cit. on p. 12).

[7] *Videocore®iv3darchitecturereferenceguide*, https://docs.broadcom.com/docs/12358545, (Accessedon05/04/2018), 5300CaliforniaAvenue•Irvine,CA92617: BROADCOM, (cit. on p. 12).

[8] *I.mx6solo/6dualliteapplicationsprocessorsforindustrialproductsdatasheet*, https://www.nxp.com/docs/en/data-sheet/IMX6SDLIEC.pdf, (Accessedon05/04/2018), (cit. on p. 12).

[9] NickPowers, *Benchmarkingpopularsingleboardcomputers—arrow.com*, https://www.arrow.com/en/research-and-events/articles/benchmarking-popular-single-board-computers, (Accessedon05/09/2018), Feb. 2017 (cit. on pp. 14, 38, 39).

[10] MichaelLarabelandMatthewTippett, *Phoronixtestsuitev7.8.0*, https://www.phoronix-test-suite.com/documentation/phoronix-test-suite.pdf, (Accessedon05/14/2018), (cit. on p. 14).

[11] MichaelLarabel, *Benchmarksofmanyarmboardsfromtheraspberrypitonvidiajetsontx2-phoronix*, https://www.phoronix.com/scan.php?page=article&item=march-2017-arm&num=5, (Accessedon05/14/2018), Mar. 2017 (cit. on p. 14).

[12] *Openbenchmarking.org - c-ray test profile*, https://openbenchmarking.org/test/pts/c-ray-1.1.0, (Accessed on 06/04/2018), (cit. on p. 16).

[13] *Overview-CMUcam5Pixy-CMUcam:OpenSourceProgrammableEmbeddedColorVisionSensors*. [Online]. Available: http://cmucam.org/projects/cmucam5 (visited on 05/01/2018) (cit. on p. 24).

8 Appendix

.1 Appendix 1 : Result benchmark [9]

4 CPUs/1 Parallel Process

Benchmark	Units	BBB	DragonBoard	Jetson	Joule	Raspberry Pi 3
Dhrystone 2 using register variables	lps	3675629.7	6412076.1	16062505.4	20379093.6	4409370.5
Dhrystone 2 using register variables	DMIPS	2092.0	3649.4	9142.0	11598.8	2509.6
Double-Precision Whetstone	MWIPS	191.3	726.3	1939.8	3366.4	765.9
Execl Throughput	lps	420	584.3	1708.2	1012.1	547.4
File Copy 1024 bufsize 2000 maxblocks	KBps	67210.2	175556.2	178734.4	536476.1	147852
File Copy 256 bufsize 500 maxblocks	KBps	20736.3	52613.5	50675.1	175890.5	44358
File Copy 4096 bufsize 8000 maxblocks	KBps	154563.2	444977.6	560630.5	1136087.5	363495
Pipe Throughput	lps	159914.8	448412.7	326667.6	1221011.2	313126.2
Pipe-based Context Switching	lps	24435.6	39752.1	70472.7	56665.2	55660.8
Process Creation	lps	1081.5	1733.9	4638.9	2171.4	2533.6
Shell Scripts (1 concurrent)	lpm	737.4	1381.3	2195.5	4384.9	2171.9
Shell Scripts (8 concurrent)	lpm	94.8	712.4	1264.7	1856.7	590.7
System Call Overhead	lps	348986.1	851850	777035.1	1963668.6	699242.9
System Benchmarks Index Score:		131.2	320	500.7	802.4	306.8

.2 Appendix 1 : Result benchmark [9]

4 CPUs/4 Parallel Process

Benchmark	Units	BBB	DragonBoard	Jetson	Joule	Raspberry Pi 3
Dhrystone 2 using register variables	lps	N/A	18546851.3	65127838.2	66693865.6	17635586.6
Dhrystone 2 using register variables	DMIPS	N/A	10556.0	37067.6	37958.9	10037.3
Double-Precision Whetstone	MWIPS	N/A	2125.1	7802.5	11602.4	3063.1
Execl Throughput	lps	N/A	2984.7	6243.4	8878.1	2277.1
File Copy 1024 bufsize 2000 maxblocks	KBps	N/A	200079.7	254376.7	629679.3	246720.9
File Copy 256 bufsize 500 maxblocks	KBps	N/A	54394.5	75190.3	184008.3	64040.1
File Copy 4096 bufsize 8000 maxblocks	KBps	N/A	560188.4	726453.7	1503381.1	579721.3
Pipe Throughput	lps	N/A	1216158.7	790668.8	4066808.5	1249299.5
Pipe-based Context Switching	lps	N/A	174856.2	250735.2	631020.7	210350.8
Process Creation	lps	N/A	7602.8	8602.8	27367.6	5097.1
Shell Scripts (1 concurrent)	lpm	N/A	4564.4	10213	16146.5	4954.9
Shell Scripts (8 concurrent)	lpm	N/A	588.1	1305.8	2038.1	639.2
System Call Overhead	lps	N/A	2321293.3	2868714.1	5160156	2689482.6
System Benchmarks Index Score:		N/A	744	1225.4	2490.2	780.6