# Analysis of Sorting Algorithms Using a WSN and Environmental Pollution Data based on FPGA

Guillermo Montesdeoca *, Víctor Asanza *, Kevin Chica *, Diego H. Peluffo-Ordóñez †‡

*Facultad de Ingeniería en Electricidad y Computación,
Escuela Superior Politécnica del Litoral, ESPOL,
Campus Gustavo Galindo km 30.5 Vía Perimetral, P.O. Box 09-01-5863, Guayaquil, Ecuador
(e-mail: {guianmon,vasanza,kechica}@espol.edu.ec)
†Smart Data Analysis Systems Group (SDAS Research Group - www.sdas-group.com),
Ben Guerir 47963, Morocco (e-mail: {diego.peluffo}@sdas-group.com)
‡Modeling, Simulation and Data Analysis (MSDA) Research Program,
Mohammed VI Polytechnic University, Ben Guerir 47963, Morocco
(e-mail: peluffo.diego@um6p.ma)

2022 International Conference on Applied Electronics (AE) | 978-1-6654-9482-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/AE54730.2022.9920090

***Abstract*—Wireless Snesor Network (WSN) based systems with a focus on Internet of Things (IoT) applications generate a large amount of data. Many applications that need to process data in real time make use of microcontroller-based architectures with sequential programming. Systems based on sequential programming can emulate parallelism up to a certain number of instructions, which is not the case with Field Programmable Gate Array (FPGA). The main objective of this work is to monitor a network of 40 CO2 sensors and to perform real-time sorting of all data. In addition, the run time analysis of two sorting algorithms is performed: bubble sort and insertion sort. For this purpose, an FPGA-based architecture is implemented, controlled by a finite state machine(FSM), which executes each of the sorting algorithms. The results show that the insertion sort algorithm is faster than the burbble sort, but consumes more hardware resources in the FPGA.**

***Keywords*—FPGA, Sorting Algortims, WSN, Sensor Network.**

## I. Introduction

Today the need to send data over the internet for Internet Of Things(IOT) applications has increased dramatically, as by 2020 there will be 30 trillion connected devices generating data that must be processed [1].

One of the biggest challenges coming from the massive increase in data is that data centers will not be able to handle the traffic generated by all these devices. One proposal to solve this problem is EDGE computing [2].

This solution explores the alternative of processing data on local devices far away from the cloud in order to reduce data traffic to them. One of the biggest problems is that these devices must be able to process information in real time without using large computational resources [2].

For these purposes, alternatives to the often used microcontrollers or microcontrollers must be sought. Field Programmable Gate Array (FPGA) become beneficial when algorithms need to be executed concurrently or parallelize processes [3].

FPGAs, which were invented more than 3 decades ago and have since advanced exponentially in both complexity and number of logic blocks, comprise the next step in computer system design and control as they perform better for multi-threaded tasks due to parallelism [4].

In combination with FPGA cards, sensor technology has advanced to the point where we can use sensors such as the MG-811. This uses a chemical reaction accompanied by an electronic circuit to generate an analog signal that allows us to know the level of CO2 in the environment [5].

The reaction used by the sensor is detailed as follows: Anode side reaction:

$$2Li + +CO_2 + 1/2O_2 + 2e^- \rightarrow Li_2CO_3 \quad (1)$$

Cathode side :

$$2NA + +1/2O_2 + 2e^- \rightarrow Na_2O \quad (2)$$

General chemical reaction:

$$Li_2CO_3 + +2NA+ = Na_2O_2Li + +CO_2 \quad (3)$$

The Nernst equation shows what would be the PEF generated from the reaction

$$FEM = Ec - (RxT)/(2F)ln(P(CO_2)) \quad (4)$$

With these characteristics we obtain a sensor that delivers a higher voltage the less Co2 has the environment in which it is located. A great advantage of this sensor is that it is very sensitive to Co2 gas but it is not affected when it comes in contact with other gases such as carbon monoxide [5].

Finally, one more thing to keep in mind is that since it is an electrochemical sensor, it must be calibrated before use. The instructions to perform this calibration were provided by the manufacturer and came with the sensor [5].

In this work we propose to perform an analysis of various data processing algorithm based on FPGA and present the data in charts so hat a comparison may be made with other popular hardware for iot proyects such as raspberry pi or other ARM microprocessors.

## II. Related research

There are many works related to IoT sensor networks based on real-time monitoring of data provided by sensors and implemented with microcontrollers. J. Capelo et al [6],analyzed the optimal parameters for shrimp culture, using a cyber-physical system, including sensors and an ARM processor-based embedded system, to maintain the quality of each pool under optimal conditions. A. Maisincho-Jivaja et al. [7],conducted the monitoring of parameters of a turkey farm to improve the quality of life in the hatchery. Among the monitored parameters are Temperature, Humidity, Ammonia Emission and Lux. This implementation is based on a cyber-physical system composed of a sensor network, a controller and an actuator based on microcontrollers.

Asanza et al. [8]performed the comparison of embedded systems based on ARM processors with FPGA. This work tested the execution of database servers on development boards, such as Raspberry or FPGA. Demonstrating that the analyzed performance metrics such as response time, memory and CPU usage, were more efectively used in the FPGA.

Yuan [9] makes a comparison of the time it takes to sort a large amount of data on both a computer and an ARM-based development board, the Raspberry pi. For this it uses the algorithms of bubble sorting, selection sorting, heap sorting and fast sorting. With this he manages to obtain a comparison between the sorting capabilities of the systems and the efficiency of the algorithms he uses.

Rajput et al. [10]make an in-depth analysis of fast sorting algorithms and fast sorting algorithms applying parallelism. As variables for comparison he uses the time these algorithms take to run, the number of comparisons and the speed of the algorithms. It obtains graphs of each of these metrics using matlab software and makes an analysis of the complexity of each algorithm.

## III. Experimental Methodology

To implement the sorting algorithms with concurrent execution, an Intel FPGA will be used on the DE 10 nano development board from Terasic (www.terasic.com.tw) and the Quartus Prime V19.1 design software.

The sensor network-based system consists of 40 end-devices and a point-to-multipoint coordinator. The end-devive consists of an ESP-32 LORA module connected to an MG-811 sensor. This combination will allow us to measure the Co2 levels in the environment of about one kilometer around the faculty of electrical engineering and computer science of the ESPOL, which is shown in Fig. 1.

On the other hand, the coordinator node consists of a receiver which is another ESP-32 LORA module connected using GPIO interface to a DE 10 NANO board.

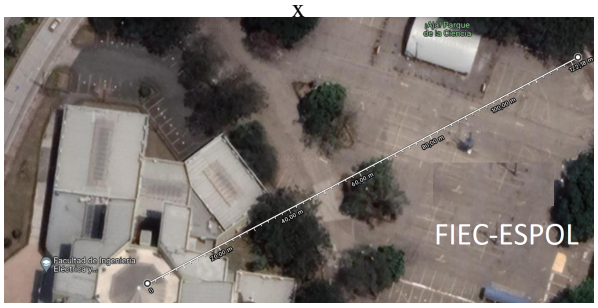


Fig. 1: *Location where environmental data were obtained*

For the communication between the coordinator node and the end-device an esp-32 lora TTGO module will be used. The wireless communication network would be as shown in Fig. 2.

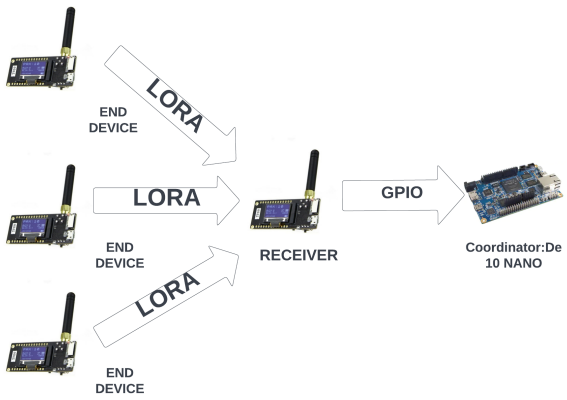


Fig. 2: *Diagram showing communication between gateway and end devices.*

4. As seen in fig 2 the receiver is conected to 40 End Devices each equipped with an air quality sensor. end devices then send the data via lora to a single receiver that then sends the data to the progarmable logic part(PL) of the FPGA using a custom GPIO interface.

fig 4 represents the time the FPGA has to receive, store and sort all the data.

- **Universal Asynchronous Receiver-Transmitter(UART):** is a block implemented in VHDL that has a transition pin and a reception pin that allows us to communicate via UART protocol with other peripherals. In this system it was used to communicate the FPGA with the TTGO. The VHDL code used to implement the MSI UART block can be found in the following repository

- **FSM Insertion :** Is the synchronous sequential machine (SSM) that acts as the controller of the digital system. It follows the behavior of the algorithm 1 for data sorting by the insertion method.
- **Random Access Memory (RAM):** Acts as a buffer and a volatile memory that contains the data coming from the sensor and is subsequently rewritten by the digital system with the sorted data.
- **Counters:** They are blocks of VHDL code that store a numeric value that increases or decreases as required by the MSS to continue the algorithm.
- **FSM bubble sort :** Is the synchronous sequential machine that acts as the controller of the digital bubble sorting system which follows the behavior of the algorithm 2 for data sorting.
- **comparators :** Is a block that allows us to know if a binary value is: greater than, equal to or less than another. It is especially useful to know if the counter has reached its maximum value or to compare two values stored in RAM memory.

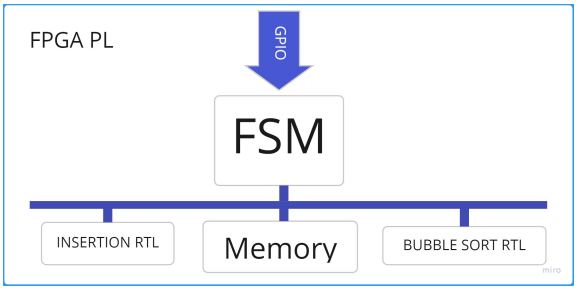


Fig. 3: *FPGA architecture.*

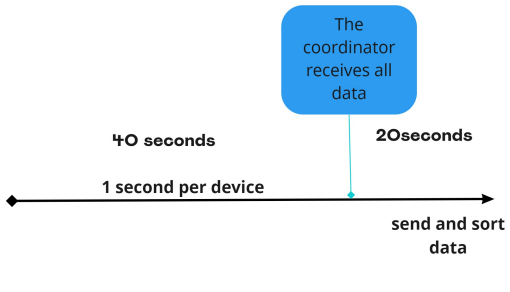


Fig. 4: *Time diagram.*

All source codes used in this work are available in the following repository: https://github.com/vasanza/FPGA-based-Co2-Monitor.

## IV. Discussion and Conclusions

The design uses the FPGA distributed memory to store the 40 values received by the sensors. Inmediately after both sorting algoritms start working at the same time this way we can measure the performance of each implementation as well as the amount of logic elements that are used.

In the figures 5 and 6,we can see the performance of the insertion algorithm. The 'x' axis shows the number of data to be sorted, and the 'y' axis shows the time in microseconds that the algorithm takes to perform the sorting. In both graphs, a linear regression was performed to predict the scalability of the sensor network using the proposed architecture. The analysis allows us to determine the execution times with a network of 10000 $CO_2$ sensors.

To perform the linear regression, the WSN was tested with 10, 20, 28 and 40 $CO_2$ sensors. The results obtained are shown in the tableI.

**Algorithm 1:** insertion sort

```
sorting using the insertion sort algorithm
while i < data to be proccesed do
    contador j = contador j +1
    regi = Ram(i)
    while j < data to be proccesed do
        Regj = Ram(j)
        if Regj < Regi then
            swap(Ram(i),Ram(j))
        j++
    end
    i++
end
```

**Algorithm 2:** bubble sort

```
sorting using the bubble sort algorithm
while start=0 do
end
i=0 j=1 counter=data to be proccesed
 Regi=Ram(i) Regj=Ram(j) while contador>0
 do
    while j < data to be proccesed do
        if j<i then
            swap(Ram(i),Ram(j))
        i=i+1 j=j+1
    end
    counter=counter-1
end
```

TABLE I: Comparison of sorting algorithms

| | *Run time in us* | |
|---|---|---|
| ***Number of End Devices (CO2 sensors)*** | ***incertion*** | ***bubble*** |
| 10 | 3.34 | 2.85 |
| 20 | 11.63 | 12.06 |
| 28 | 21.99 | 23.76 |
| 40 | 43.75 | 48.88 |

The regression performed with the data obtained using the insertion algorithm is shown in the equation 5. Where 's' represents the number of End Devices and 't' represents the run time in microseconds.

$$t = 0,0469s^{1,8538} \tag{5}$$

The regression performed with the data obtained using the bubble algorithm is shown in the equation 6. Where 's' represents the number of End Devices and 't' represents the run time in microseconds.

$$t = 0,0272s^{2,0315} \quad (6)$$

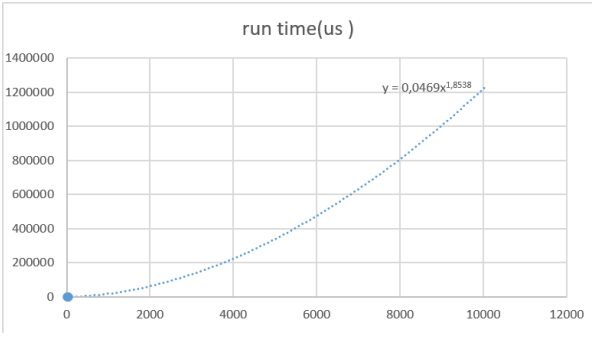


Fig. 5: *Diagram showing results corresponding to Algorithm 1* .

In the fig 5 we can see the results of the experiment with the insertion algorithm. Using the data obtained we were able to obtain an equation that allows us to extrapolate the behavior of the algorithm for a much larger number of data. this equation corresponds to a strictly increasing function whose curve resembles a potential.

Using the equation 5 we can estimate that with a netwotk of 10000 sensors, the time it would take to sort all the data received by the sensors is 1.2 seconds.

In the fig 6 we can see the results of the experiment with the bubble sort algorithm. This also has a potential behavior but more pronounced than the previous graph. With the refeq:02 regression equation, we can estimate that in a network of 10000 CO2 sensors, the ordering times with this algorithm are up to 3.5 seconds.

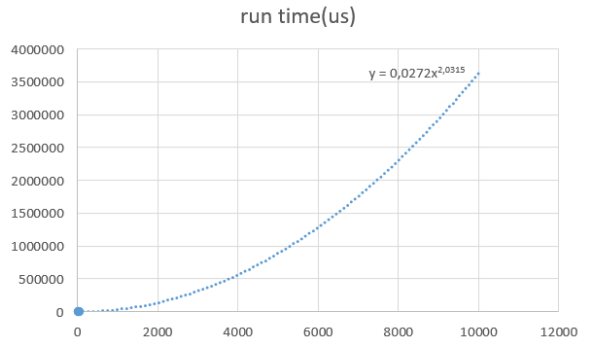


Fig. 6: *Diagram showing the results corresponding to Algorithm 2* .

In the case of the logical elements used, a comparison was made with a structure that supports a maximum of 250 data. the result obtained is: 2824 logical elements used in the insertion algorithm and 1094 elements used in the bubble algorithm.

It is concluded that the better algorithm of the two corresponds to the insertion sort algorithm since we can demonstrate using the equation found that it will take less time to sort a very large amount of data. With the exception that if the amount of data is less than 10, the bubble sort algorithm is faster.

if we make a comparison beetween our implementation and a raspbarry pi using the data from Yuan [9]'s work we can see that the raspberry pi took on average 8.53 seconds for the bubble sort and 1.87 seconds for the insertion Algorithm. In both cases the raspberry pi lost to the implementation in VHDL. therefore we can conclude that FPGA's can be used in data procesing aplications with advantages over conventional processors.

However we also need to take into account the amount of logic elements used to perform the sorting each algoritm uses. since the incertion algortim uses around 3% more of the total amount of logical element we can conclude that using the incertion sort will have faster a faster performance but the cost of the proyect will be higher since the amount of hardware needed to implement will be higher.

One of the limitations of this project is the hardware used, since the greater the number of data, the greater the number of logic elements required. Therefore, it is proposed as future work to repeat the experiment with an FPGA containing a larger number of required logic elements [11].

## REFERENCES


[1] Gomes, T., Pinto, S., Tavares, A., Cabral, J. (2015, September). Towards an FPGA-based edge device for the Internet of Things. In 2015 IEEE 20th Conference on Emerging Technologies Factory Automation (ETFA) (pp. 1-4). IEEE.

[2] Varghese, B., Wang, N., Barbhuiya, S., Kilpatrick, P., Nikolopoulos, D. S. (2016, November). Challenges and opportunities in edge computing. In 2016 IEEE International Conference on Smart Cloud (SmartCloud) (pp. 20-26). IEEE.

[3] Rodríguez, A., Valverde, J., Portilla, J., Otero, A., Riesgo, T., De la Torre, E. (2018). Fpga-based high-performance embedded systems for adaptive edge computing in cyber-physical systems: The artico3 framework. Sensors, 18(6), 1877.

[4] Dhote, S., Charjan, P., Phansekar, A., Hegde, A., Joshi, S., Joshi, J. (2016, September). Using FPGA-SoC interface for low cost IoT based image processing. In 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI) (pp. 1963-1968). IEEE.

[5] Aziz, M. H., Saptiani, P., Iryanti, M., Aminudin, A. (2019, November). Design of carbon dioxide level measures on peat soil with MG 811 sensor. In Journal of Physics: Conference Series (Vol. 1280, No. 2, p. 022061). IOP Publishing.

[6] J. Capelo et al., "Raspberry Pi-based IoT for shrimp farms Real-time remote monitoring with automated system," 2021 International Conference on Applied Electronics (AE), 2021, pp. 1-4, doi: 10.23919/AE51540.2021.9542907.

[7] A. Maisincho-Jivaja et al., "Monitoring a turkey hatchery based on a cyber-physical system," 2021 International Conference on Applied Electronics (AE), 2021, pp. 1-6, doi: 10.23919/AE51540.2021.9542899.

[8] V. Asanza, R. Estrada, J. Miranda, L. Rivas and D. Torres, "Performance Comparison of Database Server based on SoC FPGA and ARM Processor," 2021 IEEE Latin-American Conference on Communications (LATINCOM), 2021, pp. 1-6, doi: 10.1109/LATINCOM53176.2021.9647742.

[9] Yuan, Y. (2015). Performance Evaluation of Sorting Algorithms in Raspberry Pi and Personal Computer.

[10] Rajput, I. S., Kumar, B., Singh, T. (2012). Performance comparison of sequential quick sort and parallel quick sort algorithms. International Journal of Computer Applications, 57(9).

[11] V. A. Armijos, N. S. Chan, R. Saquicela and L. M. Lopez, "Monitoring of system memory usage embedded in FPGA," 2020 International Conference on Applied Electronics (AE), 2020, pp. 1-4, doi: 10.23919/AE49394.2020.9232863.