International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 11 (2020), pp. 3367-3371
 https://dx.doi.org/10.37624/IJERT/13.11.2020.3367-3371

On Screen Display Module

Esperanza Camargo Casallas[1], Luis Alberto Jaime Hernández[2], Cristian Ancizar Bermúdez Bello[3]

[1] Professor, Dept. of Telecommunications Engineering, Universidad Distrital Francisco Jose de Caldas, Bogotá -Colombia
[2] Student, Dept. of Telecommunications Engineering, Universidad Distrital Francisco Jose de Caldas, Bogotá -Colombia
[3] Student, Dept. of Telecommunications Engineering, Universidad Distrital Francisco Jose de Caldas, Bogotá -Colombia
[1]ORCID: 0000-0002-6320-4049, [2]ORCID: 0000-0003-2400-9205, [3]ORCID: 0000-0001-5672-5746

Abstract

Design and implementation of an image capture module with different perspectives in non-stationary space exploration probes. It was developed on a reduced plate where the processing and storage of information for four strategically located cameras is centralized, obtaining a multidirectional spatial vision. The four cameras are controlled by a Script developed in an interpreted programming language (Python), and a digital switch was used for the multiplexing sequence. In the development of the Script, various libraries were used to capture images and process the encapsulation on a virtual environment, complementing the images with data information considered input parameters to the module. The video sections with autosave functionality are generated through bursts of images for fixed periods.

Keywords: Cameras, demultiplexing, graphical interface, multiplexing, OSD (On Screen Display), sensors.

I. INTRODUCTION

The design of modules for monitoring is used in places that require supervision; the primary duty of the systems is to show the images of the set of cameras with different types of information. Video is a computationally intensive task, so when we need to display multiple video streams, we can run into problems due to the processing power required [1]. When the number of monitoring cameras for the system covers an increasing area, there are problems in displaying the information; consequently, basic techniques are used with various concepts in the framework of the development and implementation of tuning for the systems, offering a description specific of applied operation and possible failures of the technologies used to give efficiency and quality in the system.

The main aim of the project is to develop an OSD (On Screen Display) video processing module for non-stationary space exploration probes; this will allow recognizing the nearby atmosphere observing the current state and serving as informative visual support presenting a combination of information and images of video taken keeping a history of reference data for future missions or studies.

In 2013 the FAC launched the SUE probe, which only had HD video capture from a single camera, evidencing the non-existence of a video processing system; later in 2014, the SUE II probe's launch with the same video system as its predecessor. We evidenced the need to implement a module to capture images that allow recognizing the environment in detail. [2]

In the capture of images by a system, the terms of focus, resolution, and processing must be of the best quality, for this the Python OpenCV module and multiplexing schemes in embedded systems were used, as proposed by M. Ashourian in his article "A video multiplexing scheme using data embedding," where the decomposition of time for multiple images in the same communication channel is proposed. A pre-processing block and another post-processing are proposed, a robust coding scheme is developed for selected data in parameters and digital watermarks. [3]

II. METODOLOGY

In the OSD video processing module for non-stationary space exploration probes, the programming card on which it develops is of great importance; It must be capable of optimal image processing, low weight, and low energy consumption. The chosen microcontroller was Raspberry Pi B + for its processing capacity and robustness in hardware. The chosen software was Raspbian Jessie based on Linux, due to its ease in configuring the ports with the GPIO library for Python [4].

The cameras selected were the camera modules on the card due to the excellent image quality that offers their complete compatibility, dimensions, and physical characteristics.[5] These work in a cycle with a multi-adapter module using a CSI connector (serial interface for a camera) [6], in the reception of information that complements the images is obtained in SPI protocol due to its wide use in different microcontrollers. The module's output is HDMI, which provides excellent resolution and can be viewed on different projectors or screens. [7]

II.I Cameras cycle routines

The connection of the cameras is made with a CSI connector with a custom design; the cameras are controlled by SSH protocol from the Raspberry Pi [8]; once the system is updated, the configuration of the cameras can be enabled in option 5 of the tools software configuration as shown in figure 1.

International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 11 (2020), pp. 3367-3371
 https://dx.doi.org/10.37624/IJERT/13.11.2020.3367-3371

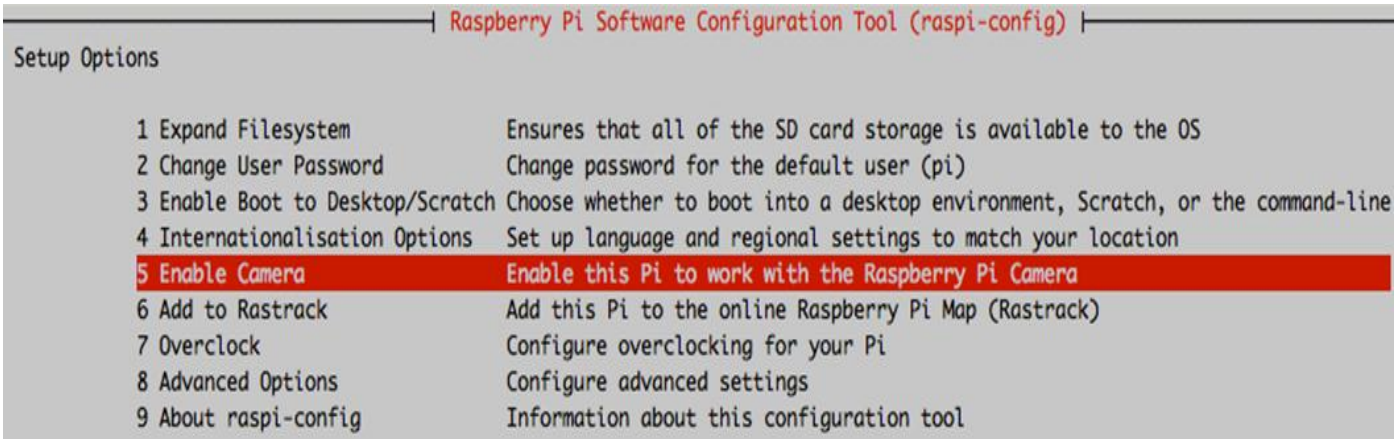

Fig. 1 Raspberry Pi Configuration window

After enabling the cameras' functionality by SSH protocol, JPEG images are generated and stored in the Raspbian execution directory; the image capture speed has a variable capture period with the '-timeout' command, this can be configured as required.

To process audio and video on a Raspberry Pi is omxplayer. This multimedia player runs from a terminal and allows us to control it easily from a simple ssh connection. For the video capture, OMXPLAYER was used to corroborate the saving of the video. This audio process was installed by default in Raspbian, so no additional package must be installed [9].

For the realization of this module, two phases were used; the first tests were carried out on a single camera's connection and processing; gradually, executions were done, adding cameras and evaluating the system's operability and response time.

II.II Multiplexing

For the mounting of the cameras, a multi-adapter was used, connecting the cameras to the available ports of the multi-adapter board and from this to the GPIO pins of the Raspberry Pi.

Fig. 2 Multi-adapter assembling with four cameras

In executing the second phase of video capture with the four cameras, it caused continuity problems due to the conflict between the alternate cameras' data lines and the reception of the Raspberry Pi's CSI port. For this reason, the video was generated taking a series of photos in a period, supporting the change of transmission lines generated by the digital switch on the CSI. Taking a photo every 60 seconds (60000 milliseconds) for 2 hours (7200000 milliseconds), it was resulting in a sequence of 120 images [10], with the command raspistill -o myimage_% 04d.jpg -tl 60000 -t 7200000, the "% 04d " will result in a four-digit number appearing in each file name.

In creating the video was carried out a "For" cycle, it was implementing the code in the Multi-adapter Module. By using functions of this type, we ensure the execution of a type of cycle with several non-simultaneous tasks.

II.III On Screen Display System

For the graphic interface, the Pygame library was used to create the OSD, adding text and images to the video. The method used for the images is "blit," and the text is "render" these functions are called within an infinite cycle to reserve shifts; each time the photo is taken, the cameras are turned on for their respective work and then turned off, in order to avoid conflict with the communication line.

Script mcfor.py was developed to couple the cameras with the Debian Linux distribution, each port worked correctly as individuals, but when operating the four ports with the multi-adapter, it did not work It was tested with the Raspberry card-carrying Rasbian Jessie, where the four cameras had been working, and the Multi-Adapter Module operated correctly; it was found that this error was caused by the Debian operating system, opting for the use of the Rasbian Jessie operating system for the use of the Script.

II.IV Visualization interface

The primary graphical interface was presented in a window with the cameras' video frame's parameter data. This Script used the raspistill command in the Rasbian terminal from the Python Script; this has little efficiency for image processing

International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 11 (2020), pp. 3367-3371
 https://dx.doi.org/10.37624/IJERT/13.11.2020.3367-3371

and was found with low processing speed, reaching about six fps (frames per second).
In a second display, the test was used to Open CV that is cross-platform, and there are versions for GNU / Linux, Mac OS X, and Windows. It contains more than 500 functions covering a wide range of areas in the vision process, such as object recognition (facial recognition), camera calibration, stereo vision, and robotic vision. Intel's built-in performance primitive's system can also be used, a set of low-level routines specific to Intel processors. [11]

For the installation of OpenCV on the Raspberry Pi board, it is necessary to download libraries such as GTK so that OpenCV can display images, build simple graphical interfaces, and perform different mathematical operations with the Numpy library. To be packaged and launched together with the operating system, the CMAKE tool was used.

The Script works under the same principle of an infinite cycle, adding functions to change cameras, expressed by the change of pins directly in the cycle, the CSI terminal is always active, managing to keep the cameras on permanently and without sudden changes in its electrical current. It also captures up to 32 fps, and text strings are added to each one and then arranged in an array in Numpy, creating a raw video frame stored in a cycle period.[12]

III. RESULT

III.II First Camera Phase

In this first phase, it was possible to see the capture of images in jpg format and video recording with a single camera, checking the Raspberry Pi's correct operation in the camera module.

III.II Cameras multiplexing phase

When trying to capture video with the four cameras of the previous stage, it caused continuity problems due to the conflict between the alternate cameras' data lines and the reception of the CSI port of the Raspberry pi.

The possibility of using the module on another operating system other than rabian was also reviewed. However, the Multi-adapter Module for cameras worked with anomalies on the Debian operating system, when different tests were carried out, showing that each port worked correctly individually. Nevertheless, at the time of using the four ports, the multiplexer worked erroneously, confusing port A with port B and port C with port D, obtaining images from only 2 cameras instead of 4, so the Rasbian Jessy system was definitely working.

The module was tested by adding elements that obtained information of interest to print over the image; the Raspberry Pi B + programming card was used; this card presented a high current deficiency due to the number of connected peripherals and processes ongoing. To cause the motherboard's processor will work in a less-than-optimal way, stopping the video frame and the OSD process in a short time. When showing short times for the purpose of this project, it was decided to use the Raspberry pi 3 board, which facilitated the process because when compared with the previous card, it does not go into a current deficiency in a short time and has a better processor that provides greater reliability in the use of the software for a more extended period as evidenced in tables 1 and 2.

Table 1. Processing video times using Raspberry pi B+.

Number of Cameras	Time
2 Cameras	3h 27m
3 Cameras	40m 20s
4 Cameras	4m 10s

Table 2. Processing video times using Raspberry pi 3.

Number of cameras	Time
2 Cameras	5h 00m
3 Cameras	1h 40m
4 Cameras	28m 22 s

III.III Multiplexing phase using Open CV.

In this phase, it was possible to obtain continuity in the frames generated with a more straightforward OSD design that did not demand from the system more processing capacity per image. Additionally, it allowed the handling of the data that will be entered to print on the image with less latency and without having inconveniences due to the current demand by the different input devices; in this phase, different tests of the device were carried out operating during travel, experiencing movement and several hours on.

III.III.I Test of device

1) Sierra Morena test in Ciudad Bolívar

Fig. 3 Triple camera test and sensor devices, Sierra Morena

International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 11 (2020), pp. 3367-3371
 https://dx.doi.org/10.37624/IJERT/13.11.2020.3367-3371

A problem was found with the photo's storage, a maximum of 122 photos were saved; afterward, they were rewritten; the code was structured so that more photos could be saved

3)Test on the way to Guadalupe

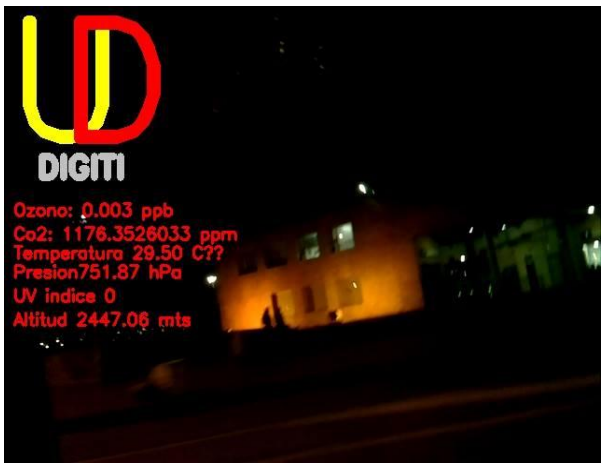

Fig. 4 Triple camera test with input sensors, road to cerro de Guadalupe- Bogotá

On the way to Guadalupe, due to the error of only storing 122 photos, it was not possible to have relevant captured, but the software was stable for 4 hours and 45 minutes.

4)Test on the way to Guadalupe II

In the second test in Guadalupe, with errors corrected, 99 photos stored every 5 minutes, four cameras in use, and all sensors connected, a tremendous current deficiency was found for the Raspberry PI due to the large amount of charge that is connected to its Gpio. The test lasted 2 hours and 57 minutes, software in perfect working order, 2754 photos are taken on the route.

Fig. 5 Photo taken at the start of the test, on the way Guadalupe II

Fig. 6 Photo number 1406, maximum height reached through the test, cerro de Guadalupe

Fig. 7 Final test photo, on the way to Guadalupe II

5) Test, El Ensueño

Hereby, we verified the efficient operation of Script and the HDMI video output along with the compound video output that was implemented in the Raspberry pi board; the different approaches of the cameras were also projected as to what can or can not be observed, for example, the upper chamber of the probe has the function of monitoring the balloon.

Fig. 8 Photo of the parachute and the probe taken by the probe's top camera

IV. CONCLUSION

In the development of the module with the Debian Linux distribution, each port worked right individually, but when using the four ports, the multiplexer worked erroneously, confusing port A with port B and port C with D, obtaining images only with 2 cameras instead of 4, so it is advisable to use multiplexer with the Raspbian Jessy distribution that does not present this problem.

Tests were carried out with input parameters to the system in Raspberry Pi B +, this card presented a high current deficiency due to the amount of connected peripherals and started processes, causing problems in the processor, stopping the video frame and the OSD processing. The results showed that the maximum operating time of the system in this version is 3 hours and 27 minutes, with the use of two cameras, which was not optimal. When demonstrating the functionality times, it was decided to use the Raspberry pi 3 board, since it presented a system functionality time improved in an additional 45% compared to the Raspberry Pi B +. A good power supply is necessary to avoid Raspberry pi to reboot due to power failure, it is recommended to use a 5 volt 1- or 2-Amp adapter for proper operation.

When working with OpenCV there is evidence of an improvement in image processing with OSD, using an array of matrices with Numpy, this facilitates a faster photo sequencing compared to the time it takes to use Pygame. It is advisable to work with OpenCV and the necessary libraries within a virtual environment immersed in the operating system; in it you can install all the required files without affecting the operating system.

REFERENCES

[1] V. Bautista and F. Gallego, GPU: Application for CCTV systems. Tecnol. Segur. (ICCST), Conf. Int. sobre Carnahan, 2014.

[2] FAC, 'Globo Sonda SUE II Desarrollo Aeroespacial Colombiano | Fuerza Aérea Colombiana', 2014. [Online]. Available: https://www.fac.mil.co/globo-sonda-sue-ii-desarrollo-aeroespacial-colombiano (accessed Jul. 25, 2020).

[3] M. Ashourian, 'A video multiplexing scheme using data embedding', in 2006 International Conference on Computing & Informatics, 2006, pp. 1–4, doi: 10.1109/ICOCI.2006.5276579.

[4] M. Pérez, 'TUTORIAL RASPBERRY PI – GPIO [PARTE 2]: CONTROL DE LEDs CON PYTHON - Geeky Theory', 2014. [Online]. Available: https://geekytheory.com/tutorial-raspberry-pi-gpio-parte-2-control-de-leds-con-python/(accessed Jul. 25, 2020).

[5] Raspberrypi, 'Buy a Camera Module V2 – Raspberry Pi', 2016. [Online]. Available: https://www.raspberrypi.org/products/camera-module-v2/ (accessed Jul. 25, 2020).

[6] L. Jackson, 'Multi Camera Adapter Module for Raspberry Pi - Arducam', 2015. [Online]. Available: https://www.arducam.com/multi-camera-adapter-module-raspberry-pi/ (accessed Jul. 25, 2020).

[7] Raspberrypi, 'Camera Modules - Raspberry Pi Documentation', 2014. [Online]. Available: https://www.raspberrypi.org/documentation/usage/camera/ (accessed Jul. 25, 2020)

[8] C. Enriquez, 'Acceso Remoto a la Computadora Raspberry Pi', 39, 2014. [Online]. Available: http://www.boletin.upiita.ipn.mx/index.php/ciencia/510-cyt-numero-39/365-acceso-remoto-a-la-computadora-raspberry-pi (accessed Jul. 25, 2020)

[9] R. Velasco, 'Cómo usar Omxplayer para ver películas en alta definición (HD)', 2014. [Online]. Available: https://www.redeszone.net/raspberry-pi/como-usar-omxplayer-para-ver-peliculas-en-alta-definicion-hd/ (accessed Jul. 25, 2020)

[10] Matt, 'Creating Timelapse Videos With The Raspberry Pi Camera-Raspberry-spy', 2013. [Online]. Available: https://www.raspberrypi-spy.co.uk/2013/05/creating-timelapse-videos-with-the-raspberry-pi-camera/ (accessed Jul. 25, 2020).

[11] OpenCV team, 'OpenCV'. [Online]. Available: https://opencv.org/ (accessed Jul. 25, 2020).

[12] A. Mordvintsev, 'OpenCV: Introducción a los tutoriales de OpenCV-Python', 2013. [Online]. Available: https://docs.opencv.org/master/d0/de3/tutorial_py_intro.html (accessed Jul. 25, 2020)