

Multi-view video acquisition with synchronized IP cameras

A Degree Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Jennifer Valle

In partial fulfilment

of the requirements for the degree in

Audiovisual systems ENGINEERING

Advisor: Josep R. Casas

Barcelona, February 2016

Abstract

The media world has changed as we know it. A few years ago we could not imagine having interactive televisions. We can now surf the web from the TV and save our favourite program to enjoy later. In this project we propose a new way to display our favourite events. With this system, the client can watch an event from the point of view that you want to live. The customer is no longer passive and becomes a participant of the action is happening. The main objective is to create a simple application for the user to choose the camera from which to observe the event.

It is for this reason that synchronization is the cornerstone of the project. We studied different possibilities to carry out this idea and to solve the problems that have arisen. We find problems when performing streaming with two cameras, but will find a solution to this inconvenience. On the other side we can control the cameras and configure the parameters of interest.

It is a fairly comprehensive and complex project and we could not get a final product for the customer by the time limitation. Therefore it will have to make further improvements.

It has proved to be an interesting work, where I have learned many useful things for future projects. Besides all the work that I do can be helpful for other projects with these cameras. It could also be a final product that may interest companies that offer live events on Internet.

Resum

El món multimèdia ha canviat molt des de que va sorgir. Fa uns anys era impensable pensar que tindríem una televisió interactiva, que podríem navegar per la xarxa i, fins i tot, gravar els nostres programes favorits per veure'ls més tard. Amb aquest projecte volem oferir una nova forma de visualitzar els esdeveniments importants. Des d'aquest sistema el client pot escollir el punt de vista que desitgi per veure un succés com, per exemple, un partit de futbol. El client deixa de ser un element passiu a ser particip de l'acció que està succeint en aquell moment. Per tant, l'objectiu principal és crear una aplicació simple des de la qual la persona pugui seleccionar la càmera que més li interessi en aquell moment.

Per aquest motiu, la sincronització és molt important en aquest projecte. Hem estudiat les diferents opcions per realitzar aquesta idea i els problemes que aniran sorgint. Trobarem problemes sobretot amb la part d'streaming amb dues càmeres i trobarem una manera per arreglar-ho. Però per una altra banda, podrem controlar perfectament les càmeres que utilitzarem i configurar-les amb els valors més adients.

Ha resultat ser un treball molt complet i a la vegada, molt complex i no s'ha pogut obtenir una aplicació o producte final per la falta de temps. Encara que es podran fer millores més endavant.

Ha sigut un projecte interessant i del qual he après moltes coses útils que m'ajudaran en futurs projectes que realitzi. A més a més, es podrà aprofitar aquest treball per a futurs projectes que es realitzin amb aquestes càmeres. Com a última conclusió, indicar que podria ser un producte que podria interessar a les empreses que es dediquen a difondre esdeveniments en directe per internet.

Resumen

El mundo multimedia ha cambiado mucho desde surgió. Hace unos años era impensable pensar que tendríamos una televisión interactiva, que podríamos navegar por la red y que incluso, podríamos gravar nuestros programas favoritos para verlos más tarde. Con este proyecto queremos ofrecer una nueva forma de visualizar los eventos importantes. Con este sistema el cliente puede escoger el punto de vista desde el que quiere ver un suceso como por ejemplo, un partido de fútbol. El cliente deja de ser un elemento pasivo a ser partícipe de la acción que está sucediendo en ese momento. Por lo tanto, el objetivo principal es crear una aplicación simple desde la cual la persona pueda seleccionar la cámara que más le interese.

Por este motivo, la sincronización es muy importante en este proyecto. Hemos estudiado las diferentes opciones para realizar esta idea y los problemas que irán surgiendo. Encontraremos problemas sobre todo en la parte de streaming con dos cámaras y conseguiremos una manera de solucionarlo. Por otro lado, podremos controlar perfectamente las cámaras que usaremos y configurarlas con los valores que creamos mejores.

Ha resultado ser un trabajo muy completo y a la vez, muy complejo y no se ha podido llegar a obtener una aplicación por la falta de tiempo. Aunque creemos que se podrán hacer mejores más adelante.

Ha sido un proyecto interesante i del cual he aprendido muchas cosas útiles que me ayudarán en futuros proyectos que realice. Además, se puede aprovechar este trabajo para futuros proyectos que se realicen con estas cámaras. Como última conclusión, indicar que podría ser un producto que podría interesar a las empresas que se dedican a difundir eventos en directo por internet.



Dedication

I dedicate this degree thesis to my family and friends that support me in all my decisions and because they make me smile every day.

Thanks to all.

Acknowledgements

First of all I would like to sincerely thank this project Josep R. Casas for all the support you have provided me. It has helped me to realize this project and has given me confidence in myself when things did not go as we had planned.

I would also like to thank Josep Pujal all the technical support that he has given me to do this job. And Albert Gil for help as quickly as possible every time I had a question about programming or the working environment.

I also thank all the support that Elphel provided us to solve our problems that have arisen in the project and the rapid reply to the messages we sent.

Finally thank my family and friends for support and give me a quiet place to work.

Thanks to all, without your helping it would not have been possible to do this project.

Revision history and approval record

Revision	Date	Purpose
0	11/01/2016	Document creation
1	16/01/2016	First index
2	17/01/2016	Modify index
3	19/01/2016	First topic
4	20/01/2016	Second topic
5	21/01/2016	Third topic
6	22/01/2016	Conclusions
7	23/01/2016	Abstract and Results
8	24/01/2016	Revision 1
9	25/01/2016	Last Revision

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Jennifer Valle	jennifer.valle@alu-etsetb.upc.edu
Josep R. Casas	josep.ramon.casas@upc.edu


Written by:		Reviewed and approved by:	
			
Date	21/01/2016	Date	25/01/2016
Name	Jennifer Valle	Name	Josep R. Casas
Position	Project Author	Position	Project Supervisor

Table of contents

Abstract	1
Resum	2
Resumen	3
Dedication	4
Acknowledgements	5
Revision history and approval record	6
Table of contents.....	7
List of Figures.....	9
List of Tables.....	10
List of Equations.....	12
1. Introduction	13
1.1. Statement of purpose	13
1.2. Objectives.....	13
1.3. Requirements and specifications	14
1.3.1. Requirements.....	14
1.3.2. System specifications	14
1.3.3. Camera specifications.....	15
1.4. Methods and procedures	16
1.5. Work plan	16
1.5.1. Work packages	17
1.5.2. Milestones	20
1.5.3. Gantt diagram	21
1.6. Description of the deviations from the initial plan and incidences that may have occurred ...	20
1.6.1. Initial plan: RTSP protocol	20
1.6.2. Imsgsv.....	20
2. State of the art of the technology used or applied in this thesis:	21
2.1. Introduction.....	21
2.2. Server/Eclipse	21
2.3. Elphel cameras.....	21
2.3.1. Introduction.....	21
2.3.2. C++.....	22
2.3.3. Trigger	22
2.3.4. Synchronization.....	22



2.3.5.	Protocol RTSP	22
2.3.6.	Parameters editor.....	23
2.3.7.	API PHP	24
2.3.8.	Imgsrv (curl and exif)	25
3.	Methodology / project development:	26
3.1.	Introduction.....	26
3.2.	Work environment	26
3.2.1.	Server.....	26
3.2.2.	Positioning of cameras.....	27
3.3.	Setting cameras.....	28
3.3.1.	Parameter editor	28
3.3.2.	API PHP and get/set functions	29
3.4.	C++ programming	29
3.4.1.	RTSP protocol.....	30
3.4.2.	Imgsrv.....	31
3.5.	Tests.....	32
3.5.1.	Synchronization tests.....	34
4.	Results	38
5.	Budget.....	39
6.	Conclusions and future development:.....	43
7.	Bibliography	44
8.	Appendices	46
A.	Conversation with Elphel's support list.....	46
B.	Setting cameras	54
C.	Configure cameras	56
9.	Glossary.....	60

List of Figures

<i>Figure 1: Block diagram</i>	16
<i>Figure 2: Gantt diagram</i>	21
<i>Figure 3: Camera Elphel NC353L [14]</i>	21
<i>Figure 4: Framerate Elphel cameras</i>	22
<i>Figure 5: Cameras map</i>	27
<i>Figure 6: Wiring diagram on cameras</i>	27
<i>Figure 7: Camera Control Interface</i>	28
<i>Figure 8: Parameter Editor</i>	29

List of Tables

<i>Table 1: System Specifications</i>	14
<i>Table 2: Camera Specifications</i>	15
<i>Table 3: Milestones</i>	20
<i>Table 4: Important Elphel Parameters</i>	24
<i>Table 5: Example to change the framerate command line</i>	24
<i>Table 6: Libcurl init</i>	31
<i>Table 7: Curl execution</i>	31
<i>Table 8: Finish command curl</i>	31
<i>Table 9: Test1 and Test2</i>	32
<i>Table 10: Test3, Test4 and Test5</i>	32
<i>Table 11: Test5 and Test6</i>	33
<i>Table 12: Test7, 8, 9, 10 and 11</i>	33
<i>Table 13: Test12</i>	34
<i>Table 14: Result Test12</i>	34
<i>Table 15: Test14</i>	35
<i>Table 16: Result Test14</i>	35
<i>Table 17: Test16</i>	36
<i>Table 18: Result Test16</i>	36
<i>Table 19: Results Test17</i>	36
<i>Table 20: Results Test18</i>	37
<i>Table 23: Test19</i>	37
<i>Table 24: Costs Table</i>	39
<i>Table 25: Cameras amortization</i>	40
<i>Table 26: Server amortization</i>	40
<i>Table 27: Computer amortization</i>	40
<i>Table 28: Budget</i>	41
<i>Table 29: Jennifer's first email</i>	46
<i>Table 30: Support-list 's first email</i>	47
<i>Table 31: Jennifer's second email</i>	47
<i>Table 32: Support-list 's second email</i>	48
<i>Table 33: Jennifer's third email</i>	48
<i>Table 34: Support-list 's third email</i>	49
<i>Table 35: Jennifer's fourth email</i>	49



<i>Table 36: Support-list 's fourth email</i>	50
<i>Table 37: Support-list 's fifth email</i>	51
<i>Table 38: Jennifer's fifth email</i>	52
<i>Table 39: Support-list 's sixth email</i>	52
<i>Table 40: Jennifer's sixth email</i>	53
<i>Table 41: Support-list 's seventh email</i>	53
<i>Table 42: test_set function</i>	54
<i>Table 43: test_get function</i>	55
<i>Table 44: config_Elphel function</i>	59

List of Equations

<i>Equation 1: TRIG_PERIOD formula</i>	38
<i>Equation 2: Hours spent on to the project</i>	39
<i>Equation 3: Hours/months spent on the project</i>	39
<i>Equation 4: Hours/day spent on the project</i>	39
<i>Equation 5: Researcher wage</i>	39
<i>Equation 6: Software Advisor wage</i>	40
<i>Equation 7: Technical Advisor wage</i>	40
<i>Equation 10: Residual value formula</i>	41
<i>Equation 11: Amortization/year formula</i>	41
<i>Equation 12: First year amortization price formula</i>	41
<i>Equation 13: Second year amortization price formula</i>	41

1. Introduction

In this first section we will explain the main objectives of the project, the system requirements, the final work plan and the issues that have emerged throughout the project.

1.1. Statement of purpose

In this project we want to get an application for multi-view capture. The user could be able to choose from what point of view he/she wants to observe the events happening in a smartroom. The smartroom is equipped with a camera network with 10 to 12 cameras, which are connected to each other for synchronization purposes. The ultimate goal would be to build an interactive user interface allowing switching the camera view in real time or to record video by selecting a certain number of cameras.

1.2. Objectives

The project main goals are:

1. Develop a capture application from cameras IP Elphel NC353L.
2. Develop a capture application from two synchronized IP cameras.
3. Comparison of the performance of IP capture on remote server and local light PC.
4. Extension to simultaneous and synchronized capture from 10-12 cameras.
5. Do a validation by an application for spatiotemporal navigation and processed off-line.
6. Do another validation by an application for spatiotemporal navigation and processed on-line (real-time).
7. Finally, test all the system and validate it.

1.3. Requirements and specifications

In this section we discuss the different requirements and specifications required to develop this project.

1.3.1. Requirements

- The final application must be able to synchronize all the cameras that we have in the smartroom with a timestamp indicator.
- We have to get an off-line navigation with a delay smaller than 10 seconds.
- Online navigation must be in real time, so that the user obtains the number of dropped frames is minimized and the scene in the room can be monitorized.
- Navigation must be intuitive for the user to facilitate its use.
- The evaluation of the whole system has to be done quantitatively for more precision in the result.

1.3.2. System specifications

Specification	Value
Work environment	
Platform	Cloud of servers
Access services	Remote desktop client or D5 terminals
Software Quality Assurance	Unit Test Framework
Eclipse version	Kepler Service Release 1
Synchronization	
Individual frames should be timestamped	Synchronized with a network time protocol (NTP)
Streaming Media	Live555(OpenRTSP)
Evaluation	
Delay offline navigation	Less than 10 seconds.

Table 1: System Specifications

1.3.3. Camera specifications

Specification	Value
Elphel cameras	
IP Elphel cameras	NC353L
Camera sensor high-resolution	CMOS
Mount lenses	C/CS/M12
Format lenses	35mm(o larger)
System memory	64MB (SDRAM)
System flash memory	128MB (NAND flash)
Image memory	64MB (DDR SDRAM)
Sensor interface	uses LVTTTL 2.5V signals instead of 3.3V
Sensor output	12Bit ADC
Connections	Ethernet cable to camera 100Mbit/s
Default camera IP address	192.168.0.9
local light PC	ZOTAC ZBOX ID83
Dynamic Range	70 dB (76 db at 2x2 binning)
Effective number of pixels	2592x1936 (5,018,112 pixels)
Full resolution	2592x1944 @15fps
Full HD video	1080p@25fps
Recording formats	Quicktime, OGM, JPEG image sequence, JP4 RAW image
User/developer friendly	HTML, JavaScript, PHP, CGI, C/C++, Verilog

Table 2: Camera Specifications

1.4. Methods and procedures

This project is not a continuation of any work. A similar project with non-IP analog cameras was developed in 2007 by Albert Gilbut timestamps were not possible to obtain due to the unknown delay of several frames in the Osprey capture boards [1]. This proposal for multi-camera capture was explained in a TPA (*Tecnologia i Producció Audiovisual*) course that I did. On this subject some international projects exploiting smartroom facilities were explained and there was one on multi-view that interested me in this topic.

1.5. Work plan

In this section we discuss the final work plan and the modifications that have occurred regarding the *Critical Review* template. Now we show the block diagram of our project. We add new **internal tasks** and in **other tasks** we have reduced their execution time.

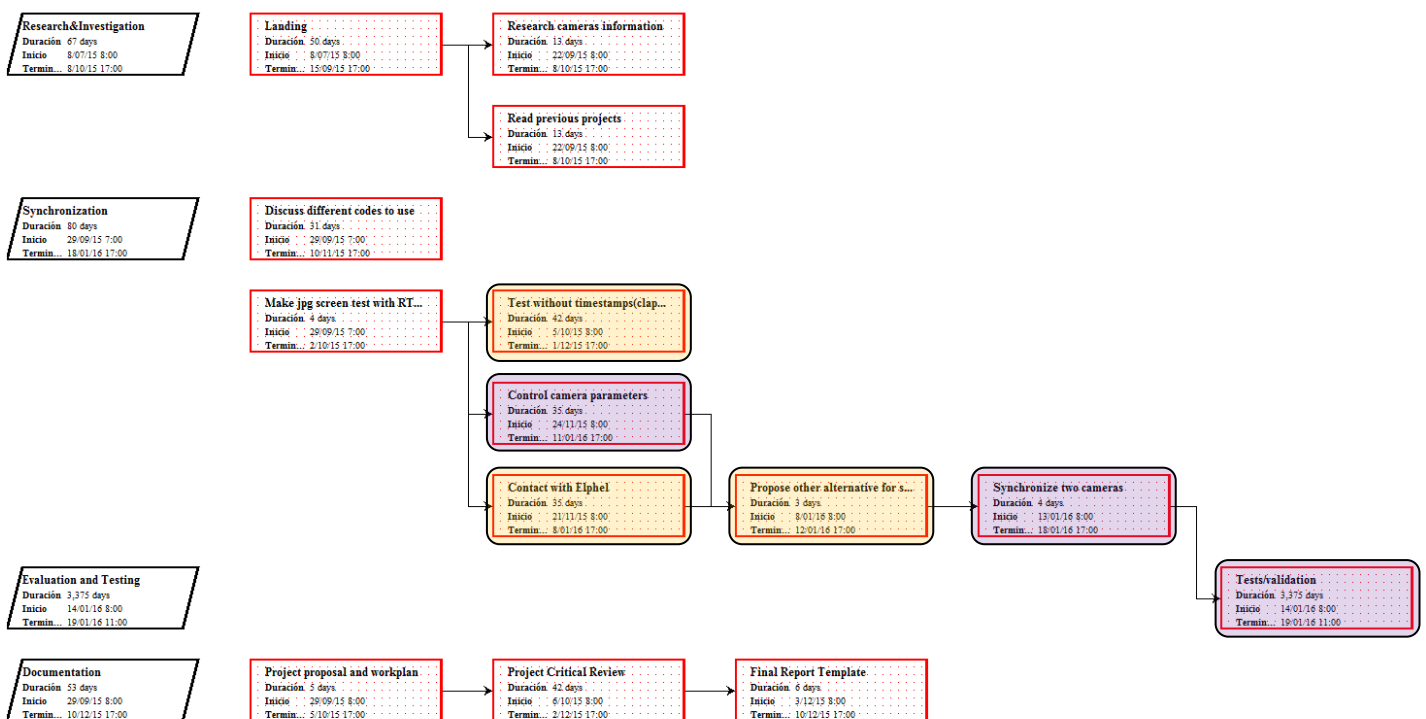


Figure 1: Block diagram

1.5.1. Work packages

Project: Research and Investigation	WP ref: (#1)	
Major constituent: Documents	Sheet 1 of 4	
Short description: Research about different topics needed to accomplish all the tasks of the project. We will communicate through the forum of The Trac Open Source Project.	Planned start date: 08/07/15 Planned end date: 08/10/15	
	Start event: 10/07/15 End event: 08/10/15	
Internal task T1: Landing Internal task T2: Read and research all the information about Elphel cameras. Internal task T3: Read all the previous projects	Deliverables:	Dates:

Project: Synchronization	WP ref: (#2)	
Major constituent: Software	Sheet 2 of 4	
Short description: This is the most important part of the project. We have to get synchronize cameras that are in the laboratory. We program in Eclipse. The landing platform will be a cloud of servers.	Planned start date: 29/09/15 Planned end date: 13/01/16	
	Start event: 29/09/15 End event: 18/01/16	
Internal task T1: Discuss different codes to use in this project. Internal task T2: Make jpg screen tests. Internal task T3: Test without timestamps (clapboard). Internal task T4: Control camera parameters. Internal task T5: Contact with Elphel. Internal task T6: Propose other alternative for synchronization. Internal task T7: Synchronize two cameras.	Deliverables:	Dates:

Project: Evaluation and Testing	WP ref: (#3)	
Major constituent: Software	Sheet 3 of 4	
Short description: In this block, we will evaluate the limitations of synchronization and we will check the appropriate functioning of the system according to the requirements.	Planned start date: 08/01/16	
	Planned end date: 13/01/16	
	Start event: 14/01/16	
	End event: 19/01/16	
Internal task T1: Tests/validation.	Deliverables:	Dates:

Project: Documentation	WP ref: (#4)	
Major constituent: Documents	Sheet 4 of 4	
Short description: Create all the different documents that should be delivered to throughout the project.	Planned start date: 15/09/15	
	Planned end date: 25/01/16	
	Start event: 22/09/15	
	End event: 25/01/16	
	Deliverables:	Dates:
Internal task T1: Project proposal and Workplan	Project Proposal	05/10/2015
Internal task T2: Project Critical Review	Critical Review	01/12/2015
Internal task T3: Final Report Template	Final template	25/01/2016

In the **Synchronization block** we added the following internal tasks:

- **Test without timestamps (clapboard):** We had to find an alternative to evaluate the system because we had incorrect timestamps. Therefore, we tested with clapboard. We closed it twice to be able to compare the same frame in the two camera recordings.
- **Control camera parameters:** The control camera parameters utility embedded in the camera seemed to be a good tool. With this we can control the camera settings in simple manner (frame rate, resolution, etc.) without having to access the camera GUI. This task was already on the critical review document. The difference is the time taken to do it, because it has become a more complex task than we expected.
- **Contact with Elphel:** We contacted the camera manufacturer (Elphel) to help us solve the problems that arose in the project.



- **Propose other alternative for synchronization:** With the alternative proposed by Elphel, we changed the code to use the imgsrv tool, proposed as an alternative to streaming.
- **Synchronize two cameras:** This task was already in the previous template but has reduced execution time. Also it has eliminated the task of expanding to all cameras for lack of time.

In the **Evaluation and Testing block** we modify the internal task:

- **Tests/validation:** This task was already in the previous template but has reduced execution time

1.5.2. Milestones

WP#	Task#	Short title	Milestone / deliverable	Date (week)
		Research and Investigation		
1	1	Landing	-	15/09/2015
	2	Research cameras information	-	08/10/2015
	3	Read all the previous projects	-	08/10/2015
		Synchronization		
2	1	Discuss codes to use in this project.	-	10/11/2015
	2	Make jpg screen tests.	-	02/10/2015
	3	Test without timestamps(clapboard)	-	01/12/2015
	4	Control camera parameters	-	11/01/2016
	5	Contact with Elphel	-	08/01/2016
	6	Propose other alternative for synchronization	-	12/01/2016
	7	Synchronize two cameras	-	18/01/2016
		Evaluation and Testing		
3	1	Tests/validation	-	19/01/2016
		Documentation		
4	1	Project proposal and Workplan	ProposalTemplate	05/10/2015
	2	Project Critical Review	CriticalReview	01/12/2015
	3	Final Report Template	Final template	25/01/2016

Table 3: Milestones

1.5.3. Gantt diagram

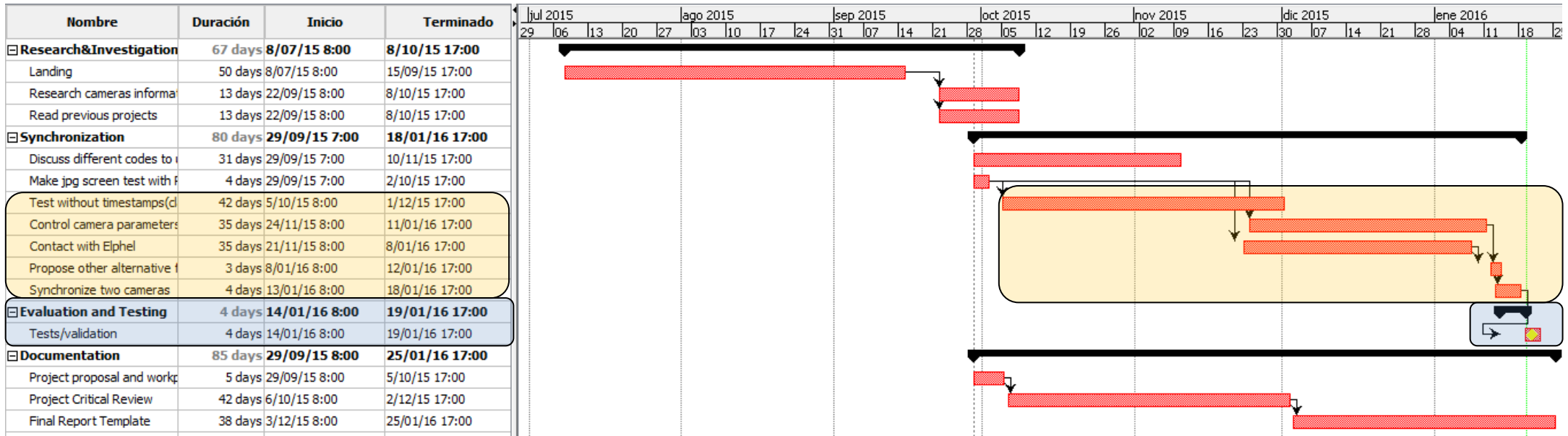


Figure 2: Gantt diagram

We can see that over the previous Gantt, there have been some modifications. We have added new internal tasks and thus block synchronization is longer now because we use a different tool (*imgsrv*). It is for this reason that the following blocks will now have a shorter duration.

1.6. Description of the deviations from the initial plan and incidences that may have occurred

This section describes the initial plan we designed and the problems that have arisen over the implementation of the project.

1.6.1. Initial plan: RTSP protocol

At first, we decided to exploit the RTSP streaming protocol [18]. We found the live555 library [17] useful for streaming. We downloaded live555 code and chose two functions for RTSP streaming. Simply by adding new methods to the already existing examples from the Elphel's wiki or in the examples provided with the live555 library. This would be independent of the transport protocol and safe.

The **problem** arose when we wanted to validate the synchronization between two cameras: the timestamp present in every captured frame was incorrect. We devoted enough time to try to find out why this problem occurred. Eventually I wrote an email to the Elphel cameras support list (See in Appendice A: **Table 28: Support-list 's first email**). They claimed that there was a problem with the timestamp in the original code in the camera streaming engine and that they could not guarantee to solve it before February.

We re-contacted Elphel's support (See Appendice A: **Table 31: Jennifer's third email**) and they advised to use some alternative tools (*camogm* [2], *imgsrv* [3], *event logger* [4]) to capture synchronized images with the "correct" timestamp.

1.6.2. *imgsrv*

We investigated all the tools advised by Elphel to choose the best alternative for our particular project. We decided to use *imgsrv*. This tool is a simple and fast HTTP server to provide metadata and still images acquired by the camera. I explain this tool in subsequent sections.

2. State of the art of the technology used or applied in this thesis:

2.1. Introduction

In this section we present the technologies that we used to realize this project. We explain the development environment in which we worked and the cameras that we used, the programming language and tools to develop the work.

2.2. Server/Eclipse

To develop this project, we use a cloud of servers as a development platform of the **Image Processing Group (GPI)** [5]. Additionally libraries and tools have been added to a group called **ImagePlus** [6] which is where we begin to program.

The working environment can be found in the computers or terminals in the D5 building of the Campus Nord, or can be **accessed remotely** [7]. There is a project management tool in which we can create "tickets" and write down notes or to create to-dos. The tool is called **Trac** [8] and, apart from the ticket management tool and the access to the software repository and documentation, it includes a wiki in **WikiFormat** [9]. There is a set of introductory pages in track website where you see the steps that you must follow to make a step by step "**landing**" into the development environment [10]

2.3. Elphel cameras

2.3.1. Introduction

The cameras we use for this project belong to the company **Elphel** [11]. This company was established in 2001 and is dedicated to the production of cameras with open source and high performance. They use **GNU General Public License 1.3**. [12] to cover all your software. They also have a wiki (Elphel wiki [13])

We have chosen the model NC353L. You can see its specifications in the **Table 2: Camera Specifications** .



Figure 3: Camera Elphel NC353L [14]

We can find a **quick guide** about how the cameras work in Elphel wiki [15]. These cameras contain a GUI (Camera Control Interface), a PHP API, a terminal, disk recorder, text and parameters editor and a system preferences.

We will also list the limitations of the camera when we doing testing (3.5 section). Then we show a picture of the **Elphel Wiki** [16] on framerates maximum obtainable according to its resolutions and the output format of the data.

	JPEG mode	JP4 mode
Performance	53 Mp/s	80 Mp/s
2592x1936	15 FPS	
1920x1088 (FullHD)	24 FPS	30 FPS
1280x720 (720p)	57FPS	60 FPS
800x608	90 FPS	
640x480	126 FPS	
320x240	310 FPS	

Figure 4: Framerate Elphel cameras

2.3.2. C++

C++ is a programming object-oriented language and is an extension of C language I decided to program in this language because I think it was very useful for the idea that I had raised and also it is language rather used within Image Processing Group for computer vision applications. Finally, C++ is important because the cameras could also be controlled through this language.

2.3.3. Trigger

The triggering allows synchronize two or more cameras. In this way we can capture an image from several cameras at the same specific moment.

In the case of our Elphel cameras, the 10369 board is required for synchronization. In addition, the cameras can have an internal or external trigger; this means that the capture of each frame is "triggered" using their own trigger or using the trigger of the camera to which it is connected, respectively. In other sections I explain the parameters of the cameras that depend on the trigger.

2.3.4. Synchronization

As explained in the previous section, the trigger is a cornerstone for cameras synchronization. The basic function is to obtain the same timestamps in the frames obtained in 2 different cameras. The timestamps are a time references which may be of microsecond.

The synchronization is therefore the most important part of this project so that the viewer chooses the camera that he wants to watch in the corresponding timestamp.

2.3.5. Protocol RTSP

Real Time Streaming Protocol is a network control protocol is not connection oriented. It will be useful because it was created specifically to control the delivery of real-time data. It controls the player with requests http: DESCRIBE, SETUP, PLAY, PAUSE and TEARDOWN.

It is a protocol that works in conjunction with RTP protocol.

We will use the **Live555**[16] library for this project. We will use the Live555 Streaming Media library for this project. It is a C ++ library that can open and use the RTP / RTCP, RTSP and SIP protocols.

When we downloaded this library we get a number of functions and files which would have to choose the right one for our case. Among them are two functions: **open_RTSP** [17] and **test_RTSPClient**. The first function is an executable (command-line program) which has a number of operations and may open a URL. This will be useful to indicate the IP camera. The other function is a smaller and simpler version of the previous function.

2.3.6. Parameters editor

Each camera has a URL from which you can access. On this website, we can find the GUI, the parameters editor, system preferences, among others.

For parameter editor, we find a fairly comprehensive list of all the parameters that you can modify the camera. It is quite complex so I'll explain only the parameters that have used or modified:

Parameter	Elphel Explanation
FP1000SLIM	FPS limit as integer number of frames per 1000 seconds
WOI WIDTH	Window of interest width. Should be multiple of 16 (divided by decimation if any). This parameter is modified by the driver according to the sensor capabilities, so if you put 10000 this value will be reduced to the full sensor width.
WOI HEIGHT	Window of interest width. Should be multiple of 16 (divided by decimation if any). This parameter is modified by the driver according to the sensor capabilities, so if you put 10000 this value will be reduced to the full sensor width.
FPSFLAG	FPS limit mode-bit 0-limit fps (not higher than), bit 1-maintain fps(not lower than)
TRIG	Trigger mode enable. Currently 0 - free running, 4 - triggered by external signal or internal FPGA timing generator.
TRIG_PERIOD	FPGA trigger sequencer output sync period (32 bits, in pixel clocks). 0-stop. 1 - single, >=256 repetitive with specified period (values 2..255 are reserved for programming timestamp communication)
TRIG_DELAY	FPGA trigger sequencer trigger delay, 32 bits in pixel clocks
TRIG_OUT	FPGA trigger sequencer trigger condition, 0 - internal, else dibits ((use<<1) level) for each GPIO[11:0] pin). Example:0x200000 - input from external connector (J15 - http://wiki.elphel.com/index.php?title=10369#J15_SYNC_external.29), 0x20000 - input from internal (J13/J14 - http://wiki.elphel.com/index.php?title=10369#J13_SYNC_internal.2C_slave.29)
TRIG_COND	level_when_active). Bit 24 - test mode, when GPIO[11:10] are controlled by other internal signals. Example: 0x800000 - output to external (J15 - http://wiki.elphel.com/index.php?title=10369#J15_SYNC_external.29) connector, 0x80000 - to internal (J12 - http://wiki.elphel.com/index.php?title=10369#J12_SYNC_internal.2C_master.29)
AUTOEXP_ON	1-autoexposure enabled when, 0- autoexposure disabled. Autoexposure daemon in DAEMON_EN is turned off-in the latter case the whole autoexposure daemon will be disabled, including white balancing and hdr mode also.
DAEMON_EN_AUTOEXPOSURE	0-turns autoexposure daemon off, 1-on. When off-not just autoexposure, but also white balance and HDR are disabled.
EXTERN_TIMESTAMP	When 1 camera will use external timestamp (received over inter-camera

	synchronization cable) if it is available (no action when external synchronization is not connected), when 0-local timestamp will be used.
XMIT_TIMESTAMP	Specify output signal sent through internal/external connector (defined by TRIG_OUT). 0-transmit just sync pulse (8*(TRIG_BITLENGTH+1) pixel clock periods long), 1-pulse+timestamp 64*(TRIG_BITLENGTH+1) pixel clock periods long.

Table 4: Important Elphel Parameters

Brief explanation of these parameters:

- FP1000SLIM: It is the framerate multiplied by one thousand.
- WOI WIDTH, WOI HEIGHT: They are the image resolution.
- FPSFLAG: It will always be 1 to maintain our framerate.
- TRIG [19]: To indicate is to use internal camera trigger or the other camera trigger.
- TRIG_PERIOD [19]: To indicate the period of trigger (96000000/FPS).
- TRIG_OUT [19]: It depends on the connection we make between cameras.
- TRIG_COND [19]: It depends on the connection we make between cameras.
- AUTOEXP_ON: If we have this enabled, the frame rate varies depending on the room lighting.
- DAEMON_EN_AUTOEXPOSURE: If we have this enabled, the frame rate varies depending on the room lighting.
- EXTERN_TIMESTAMP: To indicate is to use internal camera timestamp or the other camera timestamp.
- XMIT_TIMESTAMP: To indicate if we only send a sync pulse or if we send a pulse and timestamp.

2.3.7. API PHP

PHP is a programming language suitable for open source development. The cameras have an API PHP from which we can create a simple code to set its parameters.

Elphel have a website where we indicated all the functions that can be performed in PHP and how they are used [20].

For example:

```
<html>
  <head>
    <title>PHP Test</title>
  </head>
  <body>
    <?php
      $rate=$_GET['fps1000'];
      elphel_set_P_value(ELPHEL_FP1000SLIM, $rate);
    ?>
  </body>
</html>
```

Table 5: Example to change the framerate command line

2.3.8. **Imgsrv (curl and exif)**

“This server was developed to increase the transfer rate of individual images acquired by the Elphel 353 series cameras as the connected through the universal web failed to reach the top speed of the 100mbps network”.

To learn how to use this tool we can access your website [3].

3. Methodology / project development:

3.1. Introduction

From the moment I will explain how I developed this project and focus on the tools used and the problems encountered. First of all, a work environment had to be created to start code development for this project. Furthermore, in this case, working with servers can be seen as an added value.

3.2. Work environment

In order to begin this work, I needed an account on the Image Processing Group servers so I could access the servers and use developer's tools. I did the landing as recommended, in order to understand the environment and use it the right way.

I download a remote access program (x2Go [21]) to remotely access the network from home or from another computer that was not in the D5 lab. I discovered a new way to communicate with my colleagues through the project management tool (Trac) tickets in GPI. This way, when writing the different steps for the project, they could advise me of the best way to realize it.

I have learned to use Eclipse [22], which was an IDE program that I had never used. I also discovered the usefulness of conducting commits from time to time, in order to save the code for future mistakes. In addition to making commits I provided a new tool that my colleagues may use in the future.

I created tickets to keep track of my work. Main tickets were named: **Instalación live555** [22] (ticket # 1308) which was closely linked with a ticket from Albert Gil: **Install / build live55** [23] (ticket # 979).

3.2.1. **Server**

At first, it was a bit difficult to get used to this different way of working. I had never been within a working group in which there are several developers. Once you've got it already is quite useful because you can access tools from other developers to use in your work.

3.2.2. Positioning of cameras

We used the cameras in the smartroom, which may not be so easily accessible. We wanted to capture a scene from the smartroom so that the cameras covered as much space as possible.

With the teacher's assistance, it was possible to place 18 cameras from different points of view, as we can see in the picture.

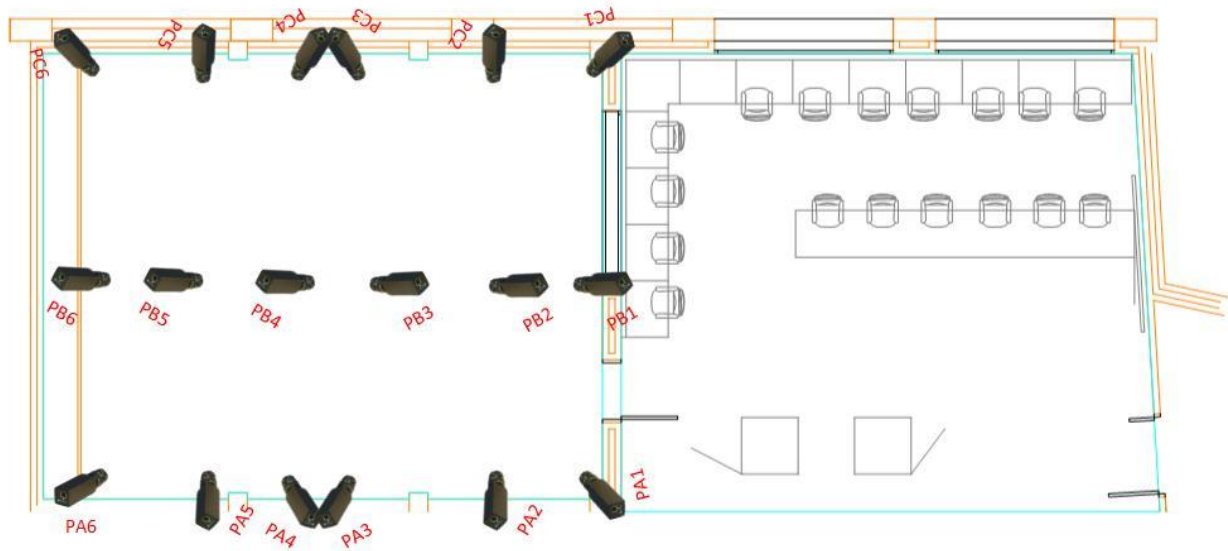


Figure 5: Cameras map

As you can see in the picture, we assign a name to each IP camera to be easier to use and can know all the time that specific position it is.

To connect the cameras between them, we followed the following diagram:

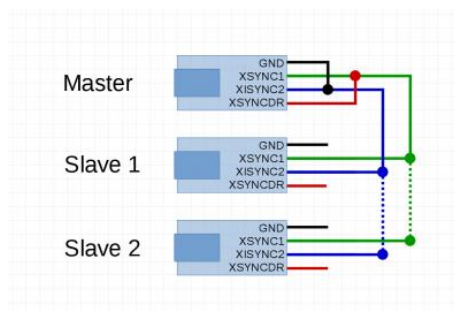


Figure 6: Wiring diagram on cameras

This way of connecting is called J15-SYNC (external) [19] and we will be of help in configuring the parameters.

3.3. Setting cameras

Once the work environment was ready, we focused on explaining the camera settings. This work is the part that resulted to be the most complex to perform and the longest in time assigned, as shown in **Figure 2: Gantt diagram**.

3.3.1. Parameter editor

At first we use the camera GUI (**Figure 7: Camera Control Interface**) because we thought it would be easier to understand and would be useful for tests.

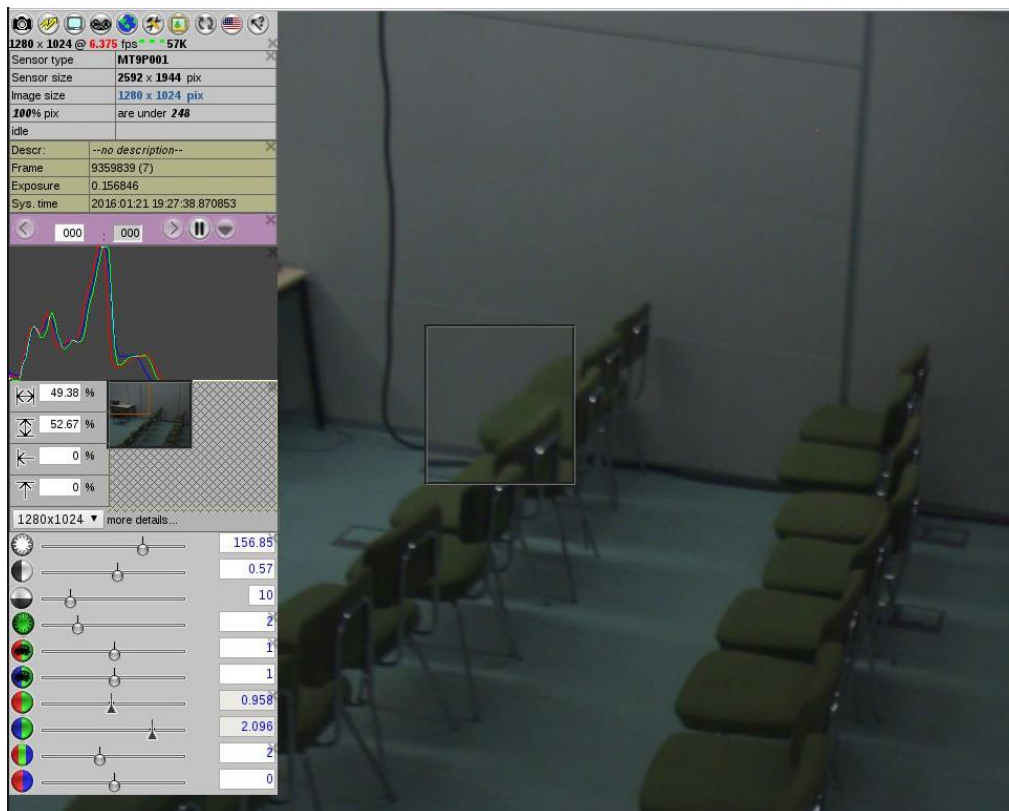


Figure 7: Camera Control Interface

We discovered a problem with the GUI tool: in some cases, when changing the value of a parameter, it did not really change anything. But we found that there were other more useful ways to change parameters: the Parameter Editor tool.

Select All	Bit	Name	Description
<input type="checkbox"/>	0	init	Initialization
<input type="checkbox"/>	1	woi	Window of Interest (WOI)
<input type="checkbox"/>	2	image	Image color, quality, ...
<input type="checkbox"/>	3	histwnd	Histogram Window
<input type="checkbox"/>	5	autoexposure	Autoexposure
<input type="checkbox"/>	6	whiteBalance	White Balance
<input type="checkbox"/>	7	streamer	Streamer
<input type="checkbox"/>	8	camftp	FTP Upload
<input type="checkbox"/>	9	camogm	In-camera recording
<input type="checkbox"/>	10	vignet	Vignetting correction
<input type="checkbox"/>	11	multisensor	Muti-sensor parameters
<input type="checkbox"/>	12	trigger	Camera trigger modes
<input type="checkbox"/>	21	persistent	Global parameters that survive sensor initializaion
<input type="checkbox"/>	22	unsafe	Not safe yet
<input type="checkbox"/>	23	diagn	Diagnostics, debug
Deselect All	View/Edit Current <input checked="" type="checkbox"/> Include image		

Figure 8: Parameter Editor

This image in Figure8 is a sample of the Parameter Editor tool. We find a table and, if we click to select one of the categories (0 to 23), we show all the parameters that can be modified for this category.

It was a bit complicated because many parameters depended on others, and learning this was mainly by trial and error because the information provided by Elphel about this editor was quite short. For example, to obtain a fixed frame rate (20fps for example), we have to set the following values:

- TRIG=4
- FPSFLAG=1
- TRIG_PERIOD=4800000 (96000000/fps)
- AUTOEXPO_ON=0
- DAEMON_EN_AUTOEXPOSURE=0
- EXTERNAL_TIMESTAMP=0
- XMIT_TIMESTAMP=1

3.3.2. API PHP and get/set functions

Later we decided to create a PHP file to manipulate or control parameters from a command line. We accessed the PHP web camera in Elphel [20] and I created the **test_set** (Table 40: *test_set function*) and **test_get** (Table 41: *test_get function*) files.

3.4. C++ programming

We explain here the main concepts related to the programming in this project. As I mentioned before, we have used the C ++ language and we have developed two different proposals due to several problems caused by the base code provided by Elphel.

3.4.1. RTSP protocol

Initially we downloaded the code live555 library [17] to use the RTSP protocol and keep the functions that most interested us. From the text example in the RTSP library named **test_RTSPClient**, we developed a new function: **capture_rtsp**. We created a new tool and called it by this name.

At first the source code was copied without changing anything for the initial commit.

We wanted to save disk frames initially to observe their timestamps. So we kept the frames with the variable name corresponding to time instant in Linux time code (**RepresentationTime**). With this, we noted that some timestamps we obtained were incorrect, because the dates were in 2080.

We thought it could have been our mistake, or that we had not understood the **RepresentationTime** variable. Therefore, we investigated the entire code to discover there was no error, and the timestamp we could read from the filename matched the timestamp obtained by the library. This took some time because it is a code that has many references to other functions and was a bit confusing to understand.

Since this was a test, we changed the base function from the live555 library and started using **openRTSP** as the base executable. It had an option to indicate the **RepresentationTime** in the filename. Only I had to put in the command line.

We saw that the mistake was repeated and tried to look at this code to find the error, but we realized that it was much more complex than the previous one, as it involved many more functions.

We returned to the previous capture RTSP code again. As we did not find the error, we decided to write to the manufacturers of the cameras to find out how to fix it. They had a support list for questions.

In the first message we received from them, they misunderstood our situation and did not solve anything. So we explain our case more clearly. In the second post, we explained that there was an error in the code of the protocol, but they answered that this error could not be fixed before the end of our project due to an accumulation of work from their side.

Since we were not given any proposal try to make a subjective validation of camera timings. I stood in front of cameras and closed twice a clapboard, count the time between closures and then check if these times were the same in each camera. This experiment also went wrong and we tried to focus on creating a tool to configure the cameras. Using this tool (**config_Elphel: Table 42: config_Elphel function**) we could create a configuration file in which indicate the values of the parameters that we wanted to modify.

We rewrote to Elphel to give us an alternative to streaming. They advised not to use RTSP as there were other options, including *imgsrv*.

3.4.2. Imgsrv

With this tool we can get as many frames as we would like just by indicating a URL, by getting a pointer to an internal buffer of the camera and copying it for writing. To use this from a C ++ program, we use the library **libcurl** [24]. I create a file called c ++ which **capture_imgsrv** and used this library. With this library one can easily read URLs, as itsupports HTTP, so that we can apply *imgsrv*.

For libcurl, the first think is to initialize the library by the **init** command of curl as follows.

```
curl = curl_easy_init();
```

Table 6: Libcurl init

We enter the URL desired with this command and execute it.

```
curl_easy_setopt(curl, CURLOPT_URL, url.c_str());
res = curl_easy_perform(curl);
```

Table 7: Curl execution

We check that is buggy and clean everything to start over again.

```
/* Check for errors */
if(res != CURLE_OK)
    fprintf(stderr, "curl_easy_perform() failed: %s\n",curl_easy_strerror(res));

/* always cleanup */
curl_easy_cleanup(curl);
```

Table 8: Finish command curl

Finally we wanted to know the information in the **exif** header of the image, which conveys of the features from the capture device. There is an "exif" command in linyx to get the exif header of the image. We note that, in the exif header, we have a field called **Data and Time Original** and **SubSecond Original Time**. With this information we will know the capture timestamp written by the camera on each frame of each camera (precision is in units of microseconds).

3.5. Tests

We have carried out several tests throughout this project. First of all we tried different values of the parameters of a **single camera** (pa2):

	Test1	Test2
FP1000SLIM	25000	25000
FPSFLAG	1	0
WOI_WIDTH	1280	1280
WOI_HEIGHT	1024	1024
TRIG	0	0
TRIG_PERIOD	-	-
Room Lighting	On-off	On-off
AUTOEXP_ON	0	0
DAEMON_EN_AUTOEXPOSURE	0	0
Framerate	24.95fps	40.58fps

Table 9: Test1 and Test2

Table 9 shows the parameters for Test1 and Test2. In these tests we set the AUTOEXP_ON and the DAEMON_EN_AUTOEXPOSURE to be independent of room lighting. In addition, we set TRIG to 0 to indicate that it is free running. The only value that changes between the two tests is the FPSFLAG. If it is equal to 0 ("not higher than") framerate is 40.58fps but it is equal to 1 ("not lower than") we obtain approximately the desired framerate 24.95fps.

	Test3	Test4	Test5
FP1000SLIM	25000	25000	10000
FPSFLAG	1	1	1
WOI_WIDTH	1280	1280	1280
WOI_HEIGHT	1024	1024	1024
TRIG	4	4	4
TRIG_PERIOD	9600000	3840000	3840000
Room Lighting	On-off	On-off	On-off
AUTOEXP_ON	0	0	0
DAEMON_EN_AUTOEXPOSURE	0	0	0
Framerate	10fps	25fps	25fps

Table 10: Test3, Test4 and Test5

Table 10 shows the parameters for Tests 3, 4 and 5. In these three tests we wanted to check how it affects the trigger on the camera framerate. As stated above the TRIG_PERIOD = 96 million / fps. We can see that if the trigger is activated, the framerate is independent of FP1000SLIM because it is controlled by the TRIG_PERIOD. We also see that this feature gets an accurate framerate.

	Test5	Test6
FP1000SLIM	25000	25000
FPSFLAG	1	1
WOI_WIDTH	1280	1280
WOI_HEIGHT	1024	1024
TRIG	0	0
TRIG_PERIOD	-	-
Room Lighting	On	Off
AUTOEXP_ON	1	1
DAEMON_EN_AUTOEXPOSURE	1	1
Framerate	24.9525fps	2.1655fps

Table 11: Test5 and Test6

In this case we can check the importance of room lighting to choose a framerate. If we put AUTOEXP_ON and DAEMON_EN_AUTOEXPOSURE equal to 1 when we turn on the light the framerate is 25fps but when we turn off the light the framerate decrease.

	Test7	Test8	Test9	Test10	Tes11
FP1000SLIM	20000	20000	20000	20000	20000
FPSFLAG	1	1	1	1	1
WOI_WIDTH	1280	1280	2048	640	320
WOI_HEIGHT	1024	1024	1536	480	240
TRIG	4	4	4	4	4
TRIG_PERIOD	3200000	2400000	2400000	2400000	2400000
Room Lighting	On-off	On-off	On-off	On-off	On-off
AUTOEXP_ON	0	0	0	0	0
DAEMON_EN_AUTOEXPOSURE	0	0	0	0	0
Framerate	29.999fps	20fps	13.33-20fps	20fps	20fps

Table 12: Test7, 8, 9, 10 and 11

In Tests 7, 8, 9, 10 and 11 (Table 12), we vary the image resolution and increase the TRIG_PERIOD to observe the limitation in framerate. In Test7 we increase the framerate to 30 and the result is not accurate (29.99). In Test8 we increase framerate to 40fps and we obtain a framerate equal to 20fps. There is a limitation of framerate due to the maximum speed of the output network connection of the camera (100 Mbit). If we decrease the resolution of the image we obtain the same result.

3.5.1. Synchronization tests

To test whether the synchronization is successful we will test **two cameras**.

Test12:

Data:

	Pa1	Pa2
FP1000SLIM	20000	20000
WOI_WIDTH	1280	1280
WOI_HEIGHT	1024	1024
FPSFLAGS	1	1
TRIG	0	4
TRIG_PERIOD	4800000	4800000
AUTOEXP_ON	0	0
DAEMON_EN_AUTOEXPOSURE	0	0
EXTERN_TIMESTAMP	0	1
XMIT_TIMESTAMP	1	1

Table 13: Test12

Result:

Camera	Framerate
Pa1	19.97084fps
Pa2	20fps

Table 14: Result Test12

To test the camera synchronization (Test 12 in Table 13), we set trigger in camera pa2 and to indicate that it uses an external timestamp. We observe in the fps results in Table 14 that the pa1 camera gets a framerate of 19,97fps following the FP1000SLIM value, and pa2 shows an accurate framerate (20fps) following the TRIG_PERIOD value. The fps values are computed from the precise timestamps in the exif headers from the frames written to disk.

Test13:

If we change the EXTERN_TIMESTAMP value to 1 in the pa1 camera, we obtain the same framerates in both cameras.

Test14:

Data:

	Pa1	Pa2
FP1000SLIM	20	20
WOI_WIDTH	1280	1280
WOI_HEIGHT	1024	1024
FPSFLAGS	1	1
TRIG	4	4
TRIG_PERIOD	4800000	4800000
AUTOEXP_ON	0	0
DAEMON_EN_AUTOEXPOSURE	0	0
EXTERN_TIMESTAMP	0	0
XMIT_TIMESTAMP	1	1

Table 15: Test14

Result:

Camera	Framerate
Pa1	20fps
Pa2	20fps

Table 16: Result Test14

For Test 14 (Table 15) we set trigger equal to 4 in both cameras. We obtain the framerate that we have set in the TRIG_PERIOD value, as shown in Table 16.

Test15:

If we do the same test as in test14 but indicating that the two cameras we will use an external trigger, we get the same framerates than in the previous tests.

Test16:

Data:

	Pa1	Pa2
FP1000SLIM	20	20
WOI_WIDTH	1280	1280
WOI_HEIGHT	1024	1024
FPSFLAGS	1	1
TRIG	4	4
TRIG_PERIOD	4800000	4800000
AUTOEXP_ON	0	0
DAEMON_EN_AUTOEXPOSURE	0	0
EXTERN_TIMESTAMP	0	1
XMIT_TIMESTAMP	1	1

Table 17: Test16

Result:

Camera	Framerate
Pa1	20fps
Pa2	20fps

Table 18: Result Test16

For Test 16, in Tables 17 and 18, we will follow the advice provided by Elphel in one of their last messages. We will set both cameras with internal trigger and the pa2 camera with EXTERN_TIMESTAMP equal to 1. This yields a framerate of 20fps in both cameras.

Test17:

We will test with the same parameters as above but changing the framerate of pa1 to 2fps. We obtain the following results:

Camera	Framerate
Pa1	2fps
Pa2	2fps

Table 19: Results Test17

We note that the framerate has changed in pa2 camera. It has followed the framerate of the pa1 camera. What happens is that the pa2 reads the first frame at 20fps but waits until the pa1 done reading it.

Test18:

We do the same test as in Test16 but changing the framerate of pa2 to 2fps.

Camera	Framerate
Pa1	20fps
Pa2	2fps

Table 20: Results Test18

In this case, the pa1 is too fast and the camera pa2 cannot read the image at the same speed.

Test19:

We connected the cameras following the J15 model (**Figure 6: Wiring diagram on cameras**). Therefore we have set in both cameras TRIG=4, TRIG_COND = 0x200000 and TRIG_OUT = 0x800000.

Data:

	Pa1	Pa2
FP1000SLIM	20	20
WOI_WIDTH	1280	1280
WOI_HEIGHT	1024	1024
FPSFLAGS	1	1
TRIG	0	4
TRIG_PERIOD	4800000	4800000
TRIG_COND	0x200000	0x200000
TRIG_OUT	0x800000	0x800000
AUTOEXP_ON	0	0
DAEMON_EN_AUTOEXPOSURE	0	0
EXTERN_TIMESTAMP	0	1
XMIT_TIMESTAMP	1	1

Table 21:Test19

We could not get results because the camera CPU hangs and need to be reset every time we set these values (See **Table 37: Support-list 's sixth email**).

4. Results

From the series of tests presented in the previous section, we can derive some conclusions. When performing the first tests with a camera we have seen that the combination of parameters affects in different ways its framerate.

The first important parameter is FPSFLAGS, because depending on its value we obtained very different framerates. Therefore, this parameter will have to be always enabled to obtain the target framerate.

In the following tests we discovered that when we activate the trigger of the camera, the exact frame rate depends more on the TRIG_PERIOD parameter rather than the parameter FP1000SLIM.

With this we can get exactly get the desired frame rate using the TRIG_PERIOD formula.

$$TRIG_PERIOD = \frac{96000000}{fps}$$

Equation 1: TRIG_PERIOD formula

To complete testing with a single camera we wanted to discover the framerate limitation of varying resolution and TRIG_PERIOD. We could observe that we can increase TRIG_PERIOD to 40fps at high resolution; but the framerate we obtained was limited to 20fps. We could not achieve that frame rate for high image resolution.

In a second series of tests, we tested synchronization with two cameras joined by a trigger cable. We wanted to check which values were important when two cameras were synchronized. We observed in the test16 that the two cameras should have activated the external trigger parameter to communicate with each other. In subsequent tests we found that if the master camera (pa1) has a framerate lower than the slave camera (pa2), then the slave camera is able to reach the same frame rate than the master.

Despite these tests, we discovered that we kept getting different timestamps for each frame of each camera. The problem may arise in that the two cameras may not have the same exact time, i.e. differing from a few seconds. The other possibility for this error would be that we could be missing to change some camera parameter to indicate that they were physically connected. We contacted Elphel and they explained how to set the connections and to determine specific values for TRIG_COND and TRIG_OUT (**Table 21:Test**).

The problem occurs when the indicated values were set. In camera pa2 we set these values and it worked properly but in pa1 camera whenever we changed the values, the camera was blocked.

It is for this reason that we could not make a final test to check if the synchronization cable was able to transport the timestamps of the frames.

5. Budget

In this section we explain the budget needed for this project. The high cost is due to the purchase of cameras and servers because, as we were using open software, we did not need to invest anything. The programs that we use are public domain.

First of all show a table of costs for each product.

Components	Cost
Hardware	
14 Cameras	14x1.300USD=18.200USD=16.854,84€
Telephone wire(50m)	3x13,60€= 40,80€
Servers	2x10.000€=20.000€
Computer	1.000€
Software	
Eclipse	0€
Trac	0€

Table 22: Costs Table

We have 14 Elphel cameras and two servers (x1 and x2) from which we can access the network. We bought telephone cable used to connect the cameras for synchronization. As we have a fairly large room, we bought 3 rolls of 50 m of cable. Also we need a computer to program and to access to network.

Regarding the software, Eclipse is a public domain program. On the other hand, Trac project manager can be used provided that their **licensing requirements** [25] are met.

This project is a UPC subject which consists of 24 ECTS. Each ECTS is equivalent to 30h dedicated to perform the project. The course duration is 4 months so:

$$30\text{hours} * 24\text{ECTS} = 720\text{ hours in 4 months}$$

Equation 2: Hours spent on to the project

We believe this project can last up to two years to solve the problems that we have encountered.

$$\frac{720\text{ hours}}{4\text{ months}} = 180\text{ hours/month}$$

Equation 3: Hours/months spent on the project

$$\frac{180\text{ hours}}{4\text{ weeks}} = 45\frac{\text{hours}}{\text{week}} = \frac{45\text{ hours}}{5\text{ days}} = 9\text{ hours/day}$$

Equation 4: Hours/day spent on the project

According to previous calculations we employ 9 hours a day to realize this project. From here we can calculate approximately the price that you pay a junior UPC engineer in a year. In UPC, The maximum number of working days is 210 days a year. Also we work 7.5 hours a day.

$$\text{Researcher} = \frac{14\text{€}}{\text{hour}} * \frac{7,5\text{hours}}{\text{day}} * \frac{210\text{days}}{\text{year}} = 22.050\text{€/year}$$

Equation 5: Researcher wage

Besides the researcher also we need a software advisor for future problems or concerns about the work programming. Furthermore, this person is responsible of maintaining the Trac web of GPI.

$$\text{Software Advisor} = \frac{20\text{€}}{\text{hour}} * \frac{4\text{hours}}{\text{day}} * 210 \frac{\text{days}}{\text{year}} = 16.800\text{€/year}$$

Equation 6: Software Advisor wage

We also need a technical supervisor for maintenance of servers and solve hardware problems.

$$\text{Technical Advisor} = \frac{20\text{€}}{\text{hour}} * \frac{4\text{hours}}{\text{day}} * \frac{210\text{days}}{\text{year}} = 16.800\text{€/year}$$

Equation 7: Technical Advisor wage

Cameras, servers and computers have a life time and therefore amortization of product. Then we show the amortization of these devices for two-year duration of the project.

Cameras amortization:

	Total price	Shelf life	Residual value (10%)	Amortization/year	First year amort. price	Second year amort. price
Cameras amort.	16.854'84€	10 years	1.685'48€	1.516'94€	15337'9€	-
Cameras amort. new	16.854'84€	10 years	1.685'48€	1.516'94€	-	13820'96€

Table 23: Cameras amortization

For cameras we put a shelf life of 10 years. But for the server a useful life of 3 years because it is used by many more people a day.

Server amortization:

	Total price	Shelf life	Residual value (10%)	Amortization/year	First year amort. price	Second year amort. price
Server amort.	20.000€	3 years	2.000€	6000€	14.000€	-
Server amort. new	20.000€	3 years	2.000€	6000€	-	8.000€

Table 24: Server amortization

Computer amortization:

	Total price	Shelf life	Residual value (10%)	Amortization/year	First year amort. price	Second year amort. price
PC amort.	1.000€	5years	100€	180€	820€	-
PC amort. new	1.000€	5years	100€	180€	-	640€

Table 25: Computer amortization

The above values are calculated from the following formulas:

$$\text{Residual value (10\%)} = \frac{\text{Total price}}{10}$$

Equation 8: Residual value formula

$$\frac{\text{Amortization}}{\text{year}} = \frac{\text{Total price} - \text{Residual value}}{\text{Shelf life}}$$

Equation 9: Amortization/year formula

$$\text{First year amortization price} = \text{Total price} - \text{Amortization/year}$$

Equation 10: First year amortization price formula

$$\text{Second year amortization price} = \text{Total price} - 2 * \text{Amortization/year}$$

Equation 11: Second year amortization price formula

	First Year	Second Year
Hardware		
Cameras Amortization	15337,9€	13820'96€
Telephone wire	40,80€	0€
Servers Amortization	14.000€	8.000€
Computer Amortization	820€	640€
Software		
Eclipse	0€	0€
Trac web	0€	0€
Team		
Project researcher	22.050€	22.050€
Software Advisor	16.800€	16.800€
Technical Advisor	16.800€	16.800€

Table 26: Budget

Viability

As we can see in the table above the price of the material we buy it is quite high. The cameras will use about 10 years so will the profitable and servers will use about 3 years because they are shared with other projects in different fields. It is a research project and it is not profitable unless we have some income in terms of technology transfer. It is for this reason that we need to find a way to make this project will be viable.

We want to obtain a final product and promote it commercially. We could make money selling this product or providing a service. This service is to offer a new way to view an event from different perspectives. The companies that are dedicated to recording live events could be our future customers.

If we think long term we will have to buy servers for exclusive use for the project and not for sharing with other developers.

Another idea to make this project feasible is that the materials used can be leveraged to current and future projects. That is, even if this project does not become a final product that other projects can use these devices for other purposes.

6. Conclusions and future development:

This section presents the conclusions we have reached after the completion of the project. To do this I think it would be nice to look back at the goals we set at the beginning and argue the changes we have made.

At first we wanted to create an application from which the customer could choose the viewpoint that prefers. This could be from a video recorded on disk or in real time. From this moment we could make an objective assessment of the results obtained.

We wanted to perform streaming using the RTSP protocol but, as we have seen, we could not do it this way due to the faulty implementation of the RTSP streaming server in the camera hardware. This was the biggest drawback of the project because it took a long time to complete an application with RTSP and we did not know that the hardware would not allow perform the streaming correctly. Therefore, when we contact with Elphel the second time, they proposed us to use a new tool to perform it (*imgsrv*). With this tool we have had little time to investigate thoroughly and for this reason some goals have not been met. We have only used two cameras and it has not been possible to make the application in real time just from a recording disk.

With the setting of the parameter there were also emerging problems, but have managed to solve these. The only downside is that we could not make one last try to improve synchronization because the camera was blocked for some combinations of parameters.

Despite problems encountered I think it has been an interesting job which I learned a lot. I used a work environment with a cloud of servers which had never worked. I learned a little programming in C ++ and working with a research team.

As a further improvement, it could be useful to develop the capture application with RTSP protocol when Elphel have solved the problems of the code. I think it's a good way to capture video and also offers the possibility of doing in real time.

If Elphel does not solve this problem, further improvement could include the current **exif library** code to avoid having to use the command line. It could also create an intuitive user interface to select cameras and observe their characteristics.

7. Bibliography

- [1] A. Gil-Moreno, *Sistema de gestió de vídeo off-line per a una smart-room*, Barcelona: Universitat Politècnica de Catalunya, 2007.
- [2] O. Dzhimiev, A. Filippov and M. Malyshko, "Camogm," 3 June 2011. [Online]. Available: <http://wiki.elphel.com/index.php?title=Camogm>. [Accessed 2015].
- [3] A. Poltorak and A. Filippov, "Imgsrv," 17 February 2012. [Online]. Available: <http://wiki.elphel.com/index.php?title=Imgsrv>. [Accessed 2015].
- [4] A. Filippov and A. Poltorak, "Event logger," 20 July 2015. [Online]. Available: http://wiki.elphel.com/index.php?title=Event_logger. [Accessed 2015].
- [5] U. UPC, "Image Processing Group," UPC,UB, 1999. [Online]. Available: <https://imatge.upc.edu/web/>. [Accessed 2015].
- [6] GPI, "ImagePlus," UPC, [Online]. Available: <https://imatge.upc.edu/trac/wiki/ImagePlus>. [Accessed 2015].
- [7] I. P. Group, "Remote Access," [Online]. Available: <https://imatge.upc.edu/trac/wiki/DevelopmentPlatform/RemoteAccess>. [Accessed July 2015].
- [8] GPI, "Track Links," 2013. [Online]. Available: <https://imatge.upc.edu/trac/wiki/TracLinks>. [Accessed 2015].
- [9] GPI, "Wiki Formatting," 2011. [Online]. Available: <https://imatge.upc.edu/trac/wiki/WikiFormatting>. [Accessed 2015].
- [10] GPI, "Landing," UPC, 2014. [Online]. Available: <https://imatge.upc.edu/trac/wiki/Landing>. [Accessed July 2015].
- [11] Elphel, "Elphel," Elphel,Inc, 2013. [Online]. Available: <http://www3.elphel.com/>. [Accessed 2015].
- [12] "GNU Operating System," Free Software Foundation, Inc.; Electronic Frontier Foundation, 2012. [Online]. Available: <http://www.gnu.org/licenses/gpl.html>. [Accessed 2015].
- [13] faris, A. Filippov, Ilya, J. Pfenniger, S. Pichelhofer, A. Poltorak and S. Khlutchin, "Elphel Wiki," Elphel,Inc, 2011. [Online]. Available: http://wiki.elphel.com/index.php?title=Main_Page. [Accessed 2015].
- [14] I. Elphel, "Elphel," 2013. [Online]. Available: <http://www3.elphel.com/nc353>. [Accessed 2016].
- [15] O. Dzhimiev, Flavio, A. Flippov and S. Pichelhofer, "Elphel 353 series quick start guide," Elphel wiki, 2010. [Online]. Available: http://wiki.elphel.com/index.php?title=Elphel_353_series_quick_start_guide. [Accessed 2015].

- [16] Kimstik, "Elphel Wiki framerate," Elphel, Inc, 2009. [Online]. Available: http://wiki.elphel.com/index.php?title=Elphel_353_framerate. [Accessed 2015].
- [17] "Live555 Streaming Media," Live Networks, Inc, [Online]. Available: <http://www.live555.com/liveMedia/>. [Accessed 2015].
- [18] "openRTSP," Live Networks, Inc, [Online]. Available: <http://www.live555.com/openRTSP/>. [Accessed 2015].
- [19] O. Dzhimiev, A. Filippov and S. Pichelhofer, "Trigger," Elphel, Inc, 2010. [Online]. Available: <http://wiki.elphel.com/index.php?title=Trigger>. [Accessed 2016].
- [20] M. Aschauer, Bburke, Dimitrios, A. Filippov, S. Pichelhofer, A. Poltorak and T. kriegler, "PHP in Elphel cameras," Elphel, Inc, 2007. [Online]. Available: http://wiki.elphel.com/index.php?title=PHP_in_Elphel_cameras. [Accessed 2015].
- [21] "X2Go," 2014. [Online]. Available: <http://wiki.x2go.org/doku.php>. [Accessed July 2015].
- [22] J. Valle, "Trac ticket 1308," October 2015. [Online]. Available: <https://imatge.upc.edu/trac/ticket/1308>. [Accessed January 2016].
- [23] A. Gil, "Track ticket 979," 2014. [Online]. Available: <https://imatge.upc.edu/trac/ticket/979>. [Accessed 2015].
- [24] "Libcurl," [Online]. Available: <http://curl.haxx.se/libcurl/>. [Accessed 2016].
- [25] "Trac License," Edgewall Software, 2003. [Online]. Available: <http://trac.edgewall.org/wiki/TracLicense>. [Accessed 2016].

8. Appendices

A. Conversation with Elphel's support list

From: Jennifer Valle **To:** support-list **Date:** 21/11/2015

Hello,

My name is Jennifer Valle and I am doing the final degree project in UPC with some of its cameras, in particular, NC353L Elphel cameras with the original code of Live555 about the RTSP protocol. I want to synchronized different cameras and I need its timestamps.

In this code, there are a `testProgs_rtsp` function with a struct `timeval` called `PresentationTime` and I think that it's the same that timestamp of the server (cameras) when the RTCP is synchronized.

I have a problem because when I execute the program, and RTCP is synchronized, I obtain negatives times and i don't understand why this occurs.

The part of the code that I want to modify is in `DummySink::afterGettingFrame` function.

```
void DummySink::afterGettingFrame(unsigned frameSize, unsigned numTruncatedBytes, struct timeval
presentationTime, unsigned /*durationInMicroseconds*/) {
    // We've just received a frame of data. (Optionally) print out information about it:
#ifdef DEBUG_PRINT_EACH_RECEIVED_FRAME
    if (fStreamId != NULL) envir() << "Stream \"" << fStreamId << "\"; ";
    envir() << fSubsession.mediumName() << "/" << fSubsession.codecName() << ":Received " << frameSize << " bytes";
    if (numTruncatedBytes > 0) envir() << " (with " << numTruncatedBytes << " bytes truncated)";
    char uSecsStr[6+1]; // used to output the 'microseconds' part of the presentation time
        sprintf(uSecsStr, "%06u", (unsigned)presentationTime.tv_usec);
    envir() << ".Presentation time: " << (int)presentationTime.tv_sec << "." << uSecsStr;
    if (fSubsession.rtpSource() != NULL && !fSubsession.rtpSource()->hasBeenSynchronizedUsingRTCP()) {
        envir() << "(not RTCP-synchronized!); // mark the debugging output to indicate that this presentation time is not
RTCP-synchronized
    }
#endif
    envir() << "\tNPT: " << fSubsession.getNormalPlayTime(presentationTime);
#endif
    envir() << "\n";
#endif
```

And I obtain "incorrects" `PresentationTimes`:

```
Stream "rtsp://pa1"; video/JPEG: Received 97775 bytes. Presentation time: 1448129165.229872(not RTCP-
synchronized!) NPT: 0.000000
Stream "rtsp://pa1"; video/JPEG: Received 98174 bytes. Presentation time: 1448129165.265949(not RTCP-
synchronized!) NPT: 0.000000
Stream "rtsp://pa1"; video/JPEG: Received 98264 bytes. Presentation time: -760859674.598727 NPT: 0.000000
Stream "rtsp://pa1"; video/JPEG: Received 98233 bytes. Presentation time: -760859674.634804 NPT: 0.000000
Stream "rtsp://pa1"; video/JPEG: Received 98228 bytes. Presentation time: -760859674.670881 NPT: 0.000000
Stream "rtsp://pa1"; video/JPEG: Received 98644 bytes. Presentation time: -760859674.706947 NPT: 0.000000
```

How could you get the correct timestamp the server gives each frame? Is it the `PresentationTime` after RTCP is synchronized or is an another variable? If I have to use the `PresentationTime`, why do I get incorrect values?

Kind regards.

Table 27: Jennifer's first email

From: support-list **To:** Jennifer Valle **Date:** 21/11/2015

Hello Jennifer,

The original Live555 implementation is rather old - I did not know it still works, there are some newer RTSP streamers. That first streamer (written by Ross himself) was made before the FPGA was providing the exact time stamp of each image. The FPGA is providing JPEG-compressed data in the circular buffer, aligning each new frame to 32 bytes and leaving additional gaps between frames, and the extra data after each frame includes 2 of the 32-bit words - seconds and microseconds of the time when the frame acquisition was started. You may find the code that extracts the timestamp here:

<http://elphel.cvs.sourceforge.net/viewvc/elphel/elphel353-8.0/apps/astreamer/video.cpp?view=markup> (line 254)

What is the overall software version in the camera that you are using?

Andrey

Table 28: Support-list 's first email

From: Jennifer Valle **To:** support-list **Date:** 23/11/2015

Hello,

I think there has been confusion, when I referred to the "original" code live555, I was referring to the latest version without changing code. The problem is that when I connect to the camera with the instruction `openrtsp -m`, I get files with the timestamp (PresentationTime) in the filename and these values are incorrect.

video-JPEG-1-1448298848.836308
video-JPEG-1-3534277304.245330
video-JPEG-1-3534277304.281407
video-JPEG-1-3534277304.317484
video-JPEG-1-3534277304.353561
video-JPEG-1-3534277304.389627

The first file has a correct timestamp, but in the following files have incorrect "times" (year 2081).

Kind regards,

Jennifer

Table 29: Jennifer's second email

From: support-list **To:** Jennifer Valle **Date:** 1/12/2015

Hello Jennifer,

Were you able to solve your problem?

I think there has been confusion, when I referred to the "original" code live555, I was referring to the latest version without changing code. The problem is that when I connect to the camera with the instruction `openrtsp -m`, I get files with the timestamp (PresentationTime) in the filename and these values are incorrect.

video-JPEG-1-1448298848.836308
video-JPEG-1-3534277304.245330
video-JPEG-1-3534277304.281407
video-JPEG-1-3534277304.317484
video-JPEG-1-3534277304.353561
video-JPEG-1-3534277304.389627

The first file has a correct timestamp, but in the following files have incorrect "times" (year 2081).

I was able to reproduce your problem. Installed live555 version (dated 2015-11-09).

The first correct timestamp is the timestamp of the PC.

Might be something with the RTSP server implementation in the camera. To find out I need to have a closer look at the code: <http://elphel.cvs.sourceforge.net/viewvc/elphel/elphel353-8.0/apps/astreamer/>

Will take some time - busy with other things.

Best
Oleg Dzhimiev

regards,

Table 30: Support-list 's second email

From: Jennifer Valle **To:** support-list **Date:** 09/01/2016

Good Morning,

A month ago I wrote a message about the problems I had with timestamps that obtained with the RTSP protocol live555. It is a project of validation of cameras in space-time and the synchronization is very important. I would be very grateful if you could solve these problems before February, when I present the final degree project.

Kinds regards,

Jennifer Valle

Table 31: Jennifer's third email

From: support-list **To:** Jennifer Valle **Date:** 09/01/2016

Hello Jennifer,

Unfortunately we did not had enough resources to get into the videostreamer code - we do not use it ourselves because of the multiple limitation of the streaming. It strips frames of the Exif headers that we use not only for the time stamping (with 1 microsecond resolution), but for other metadata too (like IMU or GPS data). Additionally videostream is designed for just synchronous streaming, while camera has a large videobuffer and it is possible to control it precisely, for example acquiring higher resolution/quality using longer time and then "catching up" either skipping frames completely or requcing resolution/quality.

Can you please describe your overall application, it may be possible to achieve the same goals differently, using approach similar to what we use ourselves?

There are several tools available for capturing and synchronizing images:

1. camogm (<http://wiki.elphel.com/index.php?title=Camogm>) allows to record sequences of images, each precisely timestamped and having optional GPS data (if GPS is connected). Multiple cameras can be synchronized together with microsecond precision, so each camera sequence of images will have frames marked with the absolutely the same timestamps.

2. imgsrv (<http://wiki.elphel.com/index.php?title=Imgsrv>) - optimized image server available at http://<camera_ip>:8081 - it has a mini-help page when you open this URL without additional parameters. This is the fastest way to grab images over the network that fully utilizes 19MB buffer in the camera, can be used with the client-based scripts or the in-camera PHP code. Among the operation modes it has multi-part JPEG (/mimg suffix in imgsrv request) that is supported by multiple browsers - it is very easy to decode as it is just a sequence of individual JPEG images with complete Exif headers.

3. Event logger (http://wiki.elphel.com/index.php?title=Event_logger) is the FPGA-based system that combines image data timestamping (from multiple cameras), GPS data (including precise absolute timing) and high-speed IMU (we use 2.5K samples/s) that allows absolute timestaping of each image with the accuracy not possible with just internal camera oscillator. Additionally the logger accepts external signals over the dedicated input and/or over the network (with network the timing uncertainty of the logged event will be higher - around 2 ms).

Andrey

Table 32: Support-list 's third email

From: Jennifer Valle **To:** support-list **Date:** 10/01/2016

Hello,

Thank you very much for the information and quick response. We study in detail what you proposed.

I'm going to explain a little what it will be our project.

Our goal is a multi-view synchronized capture for 3D analysis with 16 Elphel cameras (An example that has already been made in this university)

These analyzes can be of recordings stored on disk or in real time. Therefore the timestamps are very important in our project.

Millisecond resolution would almost enough for what we want to do, but if we have a microseconds resolution would be better.

The real-time processing does not want to do in camera; we want to do on remote machine (inside a LAN) GPUs, etc.

What would be your recommendation for this particular project?

Kind regards

Table 33: Jennifer's fourth email

From: support-list **To:** Jennifer Valle **Date:** 10/01/2016

Jennifer, how do you currently synchronize the cameras, do these cameras have 10369 (<http://wiki.elphel.com/index.php?title=10369>) interface boards?

In any case I would use wget (or just function in Python, PHP, ...) to repetitively read images from imgsrv (as explained here: http://wiki.elphel.com/index.php?title=Imgsrv#imgsrv_usage) - each image will have timestamp with microsecond resolution that you can extract from the Exif header.

As long as the average image size * frame rate does not exceed the network bandwidth (if it does you may need to decrease image quality or frame rate) cameras will tolerate some delays on the computers as long as the internal camera 19MB buffer is not overrun.

Imgsrv will wait (as explained on our wiki) for the next image ready so the host computer does not need to do anything for synchronization but read files (and match the files with the same timestamps (should be exact same value) later.

Alternatively you may use camogm over NFS (camera will "see" the host computer HDD/SSD as its own).

If the cameras do not have 10369 I/O boards, timestamps will be based on each camera clock that is based on a crystal oscillator with +/-50ppm accuracy. You can increase precision by comparing the timestamps at the beginning and end of a longer interval and calculate each clock correction.

With 10369 boards it is much easier - you can connect all cameras to one "master" camera using the telephone cables/connectors, wired as shown here : http://wiki.elphel.com/index.php?title=10369#J15_-_SYNC_.28external.29 The cameras should be placed in external trigger mode (it may limit the maximal frame rate if there is insufficient light, as in triggered mode exposure and readout are sequential, while in free-running rolling-shutter mode they overlap), and the "master" will set pace to all the cameras (itself included), the synchronization cable both provides the trigger signal and is also used to transmit a 52 (32+20) bit timestamp to all the cameras, so each of them will be triggered simultaneously and have exactly the same timestamp value in Exif header regardless of the individual camera clock rate.

And if you use data for 3d reconstruction I would recommend using JP4 format (not JPEG) and use green pixels for processing. You may use color too, of course, but raw Bayer data will be more convenient than JPEG YCbCr that is subject to artefacts introduced by a simple de-mosaic algorithm in the camera.

Andrey

Table 34: Support-list 's fourth email

From: support-list **To:** Jennifer Valle **Date:** 11/01/2016

Jennifer,

In our records I found that your cameras all have 10369 I/O boards, ordered specifically for synchronization of the cameras so I suppose you or others in your team know how to use them.

Cameras should be placed in triggered mode, where they performs as two independent devices: one just generates periodic trigger pulses (with attached timestamps), and the other (the camera itself) waits for the trigger and then initiates frame acquisition sequence (so if individual cameras have different exposures, the beginning of each exposure will match, not the center). The synchronization cable wiring defines the "master" - output pair of wires on its connector are soldered to all input pairs in parallel (including input on the master itself) - that means that all the cameras can be programmed identically as "masters" - outputs from all but the real "master" will be just not used.

The electronic rolling shutter sensors have two running (in scanline order) pixel pointers - erase pointer and readout pointer, and the delay between the erase pointer and the readout one define exposure time. In free running mode (not synchronized) the pointers can be in different frames - while the readout is in frame #0, the erase one may already be in frame #1 (next one), the longest exposure time in that mode is just the frame period - as soon as pixel is read out, it is erased and exposure starts for the next frame.

In triggered mode this is not possible, so when trigger arrives the sensor starts erasing line by line, starting from the first line. Exposure time later the readout pointer starts, and erasing of the next frame can not start until the full frame is read out. This makes the minimal frame period (the value programmed to the FPGA sync pulses generator) equal to the sum of the exposure time (or maximal anticipated exposure time if you use the default autoexposure mode).

Sensor readout time can be calculated using sensor datasheet available for download on On Semiconductor web site. Pixel clock is 96MHz (96MPix/s). This time includes not just pixel readout, but also vertical (rather small) and horizontal (large) blanking;

FPGA processes 53.33Mpix/s in JPEG mode and 80 MPix/s in JP4 mode (we recommend this mode that provides you with raw Bayer mosaic data), and FPGA has the full frame period to compress image - and there is no "blinking", so for the FPGA the minimal synchronization period is just slightly above total number of fixels divided by FPGA pixel rate (53.3M and 80M). JP4 mode is always faster than the sensor, JPEG is slower for large frames, but smaller for the smaller ones.

Other frame rate limiting factor is the network bandwidth and while sensor readout and FPGA processing do not depend on compression quality, the bandwidth sure does. Sending images over the 100Mbps Ethernet provides approximately 10MB/s of data. If you are trying to push as much data as you can, you can adjust image quality by acquiring a test image of the scene, looking at the file size - the maximal frame rate will be just 10MB/s divided by the image size.

10 MB/s is achieved if the network can handle 100Mbps from each camera. As you have 16 of them you will need to split them in 2 groups (if you are using GigE switches). On the host PC you need to run either multi-threaded or just run one script (that reads from the imgsrv) per camera simultaneously.

This will provide you with the sets of images precisely timestamped, each channel will have images with exactly the same timestamp value so you should not have any problems matching images from different channels.

There are multiple programs available to process JP4 format (an perform client-side demosaic) - in C, Java, and even JavaScript code that converts images using just HTML5 canvas.

Andrey

Table 35: Support-list 's fifth email

From: Jennifer Valle **To:** support-list **Date:** 18/01/2016

Hi,

Thanks for the information that you have provided us. It was very useful and we will use the tool imgsrv. Now we would like to ask you about camera settings.

We captured two cameras and get different timestamps for each frame. Can it be depends on the parameters of trig_cond and trig_out (which currently are at 0). We have trig = 4 mode with a trig_period = 4800000 in both. In the master camera we have external trig = 0 and the slave 1 and trig_delay equal to 0 in both.

What parameters would have to be modified to get the same timestamp of the two cameras for a framerate of 25fps for example?

Kind regards,

Jennifer

Table 36: Jennifer's fifth email

From: support-list **To:** Jennifer Valle **Date:** 18/01/2016

Jennifer,

The simplest setup is to program all (master and slaves) identically, the master camera should also be triggered by the external trigger as all the other cameras. There are 4 pins on the connector - 2 are output pair, 2 - input pair . Cabling should be as described on the wiki page - http://wiki.elphel.com/index.php?title=10369#J15_-_SYNC_.28external.29 , here is the diagram itself:http://wiki.elphel.com/index.php?title=File:Sync_cable.png

<http://wiki.elphel.com/index.php?title=Trigger> shows settings/examples. TRIG_CONDITION selects - which signal the camera is listening to, and there are 3 valid options : 0 (use own FPGA), 0x200000 - use external connector (4-pin telephone style connector) and 0x20000 - when camera boards are connected with flex cables - this option is used in multi-camera systems such as Eysis4pi, not in the regular cameras. TRIG_OUT can be 0 (no output), 0x800000 to send signal to the telephone connector (OK to have on "slave" cameras - those pins are just not connected, so no harm to drive them) and 0x80000 - drive internal connectors (like in Eysis4pi)

As for the frame rate I would start with some slow rate (like 1 fps - trig_period = 96000000) to make it easier to compare timestamps. And only when you get what you expect - increase the frame rate. And make sure you have enough light (exposure time is added to the frame readout time).

Andrey

Table 37: Support-list 's sixth email

From: Jennifer Valle **To:** support-list **Date:** 22/01/2016

Hello,

We have followed the configuration of J15 to connect the camera but when I put the values of the parameters (<http://wiki.elphel.com/index.php?title=Trigger>) in the master camera does not respond and I have to restart the

camera. Instead the slave camera takes the parameters correctly.
What is the problem?

Kind regards,

Jennifer

Table 38: Jennifer's sixth email

From: support-list **To:** Jennifer Valle **Date:** 22/01/2016

Hello Jennifer,

I'm sorry you are having these problems. Here are several notes:

1) If the camera stops responding when using external trigger, it usually means that all the PHP instances that serve requests are waiting for the event that never happens. You can telnet to the camera and run:

```
killall -9 lighttpd
```

lighttpd - the camera web server, that launches PHP instances, when you kill web server, it kills all the PHP instances, and then it gets automatically restarted.

2) Make sure that you first (or in the same command) start trigger output, then make camera listen to the external trigger.

3) The main thing - you need to make sure that the cable is correct, plugged in the correct connector (synchro connector is smaller than serial, so it can fit in the wrong port).

Can you send the photo of the cable?

Is the "master" end marked?

For testing you just need the "master" end of the cable, master camera should only work (not getting stuck) if the connector is plugged in. It is possible to make camera output a 5VDC output on the connector so you can check it with multimeter.

Andrey

Table 39: Support-list's seventh email

B. Setting cameras

- test_set:

```

<html>
  <head>
    <title>PHP Test</title>
  </head>
  <body>
    <?php
      $rate=$_GET['fps1000'];
      $width=$_GET['width'];
      $height=$_GET['height'];
      $fpsflag=$_GET['fpsflag'];
      $trig=$_GET['trig'];
      $trig_period=$_GET['trig_period'];
      $auto=$_GET['auto_expo'];
      $daemon=$_GET['daemon_expo'];
      $e_timestamp=$_GET['extern_timestamp'];
      $x_timestamp=$_GET['xmit_timestamp'];
      elphel_set_P_value(ELPHEL_FP1000SLIM, $rate);
      elphel_set_P_value(ELPHEL_WOI_WIDTH,$width);
      elphel_set_P_value(ELPHEL_WOI_HEIGHT, $height);
      elphel_set_P_value(ELPHEL_FPSFLAGS, $fpsflag);
      elphel_set_P_value(ELPHEL_TRIG,$trig);
      elphel_set_P_value(ELPHEL_TRIG_PERIOD, $trig_period);
      elphel_set_P_value(ELPHEL_AUTOEXP_ON,$auto);
      elphel_set_P_value(ELPHEL_DAEMON_EN_AUTOEXPOSURE,$daemon);
      elphel_set_P_value(ELPHEL_EXTERN_TIMESTAMP,$e_timestamp);
      elphel_set_P_value(ELPHEL_XMIT_TIMESTAMP,$x_timestamp);
    ?>
  </body>
</html>

```

Table 40: test_set function

- test_get:

```
<html>
  <head>
    <title>PHP Test</title>
  </head>
  <body>
    <?php
      echo "FP1000SLIM =";
      echo elpheel_get_P_value(ELPHEEL_FP1000SLIM)/1000;
      echo " WOI_WIDTH =";
      echo elpheel_get_P_value(ELPHEEL_WOI_WIDTH);
      echo " WOI_HEIGHT =";
      echo elpheel_get_P_value(ELPHEEL_WOI_HEIGHT);
      echo " FPSFLAGS =";
      echo elpheel_get_P_value(ELPHEEL_FPSFLAGS);
      echo " TRIG =";
      echo elpheel_get_P_value(ELPHEEL_TRIG);
      echo " TRIG_PERIOD =";
      echo elpheel_get_P_value(ELPHEEL_TRIG_PERIOD);
      echo " AUTOEXP_ON =";
      echo elpheel_get_P_value(ELPHEEL_AUTOEXP_ON);
      echo " DAEMON_EN_AUTOEXPOSURE =";
      echo elpheel_get_P_value(ELPHEEL_DAEMON_EN_AUTOEXPOSURE);
      echo " EXTERN_TIMESTAMP =";
      echo elpheel_get_P_value(ELPHEEL_EXTERN_TIMESTAMP);
      echo " XMIT_TIMESTAMP =";
      echo elpheel_get_P_value(ELPHEEL_XMIT_TIMESTAMP);
    ?>
  </body>
</html>
```

Table 41: test_get function

C. Configure cameras

- config_Elphel:

```
#define IMAGEPLUS_TOOL_CONFIG_NAME tool_config_example

#define IMAGEPLUS_TOOL_CONFIG_GROUP examples

#include <curl/curl.h>

using namespace std;

using namespace imageplus;

IMAGEPLUS_TOOL_CONFIG_BRIEF()

    "Here you should write a brief description of your tool...\n";

IMAGEPLUS_TOOL_CONFIG_DESCRIPTION()

    "This is the detailed description of your tool.      \n" +
    "What it does, how, why...                          \n" +
    "                                                    \n" +
    "You can add blank lines as the previous and the next one.\n" +
    "                                                    \n" +
    "\tAlso indented lines like this...                  \n" +
    "                                                    \n" +
    "This information will be shown in the help message.  \n";

IMAGEPLUS_TOOL_CONFIG_PARAMETERS()

// Your Options:

int    width;

int    height;

int    fps1000;

int    fpsflag;

int    trig;

int    trig_period;

int    auto_expo;

int    daemon_expo;

int    extern_timestamp;

int    xmit_timestamp;

// Your Flags:
```

```

bool verbose;

// Your Arguments

std::string  name_camera;

IMAGEPLUS_TOOL_CONFIG_DEFAULT_VALUES()

width= 1280;
height= 1024;
fps1000= 20000;
fpsflag= 1;
trig= 0;
trig_period= 9600000;
auto_expo= 0;
daemon_expo=0;
extern_timestamp=0;
xmit_timestamp=1;

IMAGEPLUS_TOOL_CONFIG_OPTIONS()

IMAGEPLUS_TOOL_CONFIG_OPTION_ABV ( fps1000 , "Framerate of camera in 1000s", 'r' );
IMAGEPLUS_TOOL_CONFIG_OPTION_ABV ( width , "WOI width of camera", 'w' );
IMAGEPLUS_TOOL_CONFIG_OPTION_ABV ( height , "WOI height of camera", 'h' );
IMAGEPLUS_TOOL_CONFIG_OPTION_ABV ( fpsflag , "0-limit fps, 1-maintain fps", 'f' );
IMAGEPLUS_TOOL_CONFIG_OPTION_ABV ( trig , "Disable(0)/Enable(4) trigger camera", 't' );
IMAGEPLUS_TOOL_CONFIG_OPTION_ABV ( trig_period , "Period of trigger camera(96000000/fps)", 'p' );
IMAGEPLUS_TOOL_CONFIG_OPTION_ABV ( auto_expo , "Autoexposure (0-disable, 1-enable)", 'e' );
IMAGEPLUS_TOOL_CONFIG_OPTION_ABV ( daemon_expo , "Autoexposure daemon(0-disable, 1-
enable)", 'd' );
IMAGEPLUS_TOOL_CONFIG_OPTION_ABV ( extern_timestamp , "Extern Timestamp(0-local timestamp, 1-
extern timestamp)", 'l' );
IMAGEPLUS_TOOL_CONFIG_OPTION_ABV ( xmit_timestamp , "Xmit Timestamp(0-pulse, 1-
pulse+timestamp)", 'x' );

IMAGEPLUS_TOOL_CONFIG_FLAGS()

```

```

IMAGEPLUS_TOOL_CONFIG_ARGUMENTS()
IMAGEPLUS_TOOL_CONFIG_ARGUMENT( name_camera , "The camera IP" );
IMAGEPLUS_TOOL_CONFIG_READ_PARAMETERS()
    IMAGEPLUS_TOOL_CONFIG_READ( fps1000 );
    IMAGEPLUS_TOOL_CONFIG_READ( width );
    IMAGEPLUS_TOOL_CONFIG_READ( height );
    IMAGEPLUS_TOOL_CONFIG_READ( fpsflag );
    IMAGEPLUS_TOOL_CONFIG_READ( trig );
    IMAGEPLUS_TOOL_CONFIG_READ( trig_period );
    IMAGEPLUS_TOOL_CONFIG_READ( auto_expo );
    IMAGEPLUS_TOOL_CONFIG_READ( daemon_expo );
    IMAGEPLUS_TOOL_CONFIG_READ( extern_timestamp );
    IMAGEPLUS_TOOL_CONFIG_READ( xmit_timestamp );
IMAGEPLUS_TOOL_CONFIG_MAIN()
{
    if (argc < 2) {
        std::cout<<"Error:se necesitan argumentos.";
        return 1;
    }
    std::cout << "Camera name          " <<cfg.name_camera    << std::endl;
    std::cout << "Framerate:          " << cfg.fps1000        << std::endl;
    std::cout << "Width:             " << cfg.width          << std::endl;
    std::cout << "Height:           " << cfg.height         << std::endl;
    std::cout << "Fpsflag:         " << cfg.fpsflag        << std::endl;
    std::cout << "Trigger:         " << cfg.trig           << std::endl;
    std::cout << "Trigger period:  " << cfg.trig_period    << std::endl;
    std::cout << "Autoexposure:    " << cfg.auto_expo      << std::endl;
    std::cout << "Autoexposure daemon: " << cfg.daemon_expo  << std::endl;
    std::cout << "Extern Timestamp: " << cfg.extern_timestamp << std::endl;
    std::cout << "Xmit Timestamp:  " << cfg.xmit_timestamp  << std::endl;
    CURL *curl;
    CURLcode res;
    curl = curl_easy_init();

```

```

if(curl) {
    int rate=cfg.fps1000;
    int w=cfg.width;
    int h=cfg.height;
    int flag=cfg.fpsflag;
    int trigger=cfg.trig;
    int t_period=cfg.trig_period;
    int a_expo=cfg.auto_expo;
    int d_expo=cfg.daemon_expo;
    int e_timestamp=cfg.extern_timestamp;
    int x_timestamp=cfg.xmit_timestamp;
    std::string url =
curl_easy_setopt(curl, CURLOPT_URL, url.c_str());
    res = curl_easy_perform(curl);
}
if(res != CURLE_OK)
    fprintf(stderr, "curl_easy_perform() failed: %s\n",curl_easy_strerror(res));
/* always cleanup */
    curl_easy_cleanup(curl);

return 0;
}
IMAGEPLUS_TOOL_CONFIG_END()

```

Table 42: config_Elphel function

9. Glossary

- GPI: Image Processing Group
- TRIG: trigger
- Imgsrv: image server
- RTSP: Real-Time Streaming Protocol
- Pa1: Camera 192.1.168.51 IP
- Pa2: Camera 192.1.168.52 IP