Formula Student Racing Championship: implementation of the on-board video acquisition and encoding system

Sílvia Fernandes Calunda, IST

Abstract—A complete on-board audio and video acquisition and encoding system, to be integrated in a Formula Student racing prototype, was developed and implemented in the scope of this dissertation. Real-time on-board video scenes, taken from several possible positions in the car, are captured with an off-the-shelf low cost webcam and subsequently transmitted to the pits by using a dedicated Wi-Fi communication infrastructure. To comply with the strict restrictions imposed by this application environment, the conceived system was implemented by using a mini-ITX embedded board, equipped with an ultra low-power Via C7 processor, running a small Linux distribution installed on a flash-based memory device. According to the conducted experimental assessment, the conceived system is able to capture and encode video streams at a rate of 14 fps using the MPEG-4 video standard.

Index Terms—On-board Audio and Video, Multimedia Signals Acquisition and Encoding, Multimedia Streaming, Portable Embedded Systems.

I. INTRODUCTION

A. Scope

The FST Project was created in 2001 by a group of students from Instituto Superior Técnico (IST), in order to design and implement a prototype to participate in the European Formula Student competitions, also known as the "University Formula One".

B. Industrial and Commercial Prototypes

There are not many commercial products available of in-car video and audio communication systems. Usually, Formula One establishes partnerships with the multinational companies which provide these services, and the technologies that they use are usually kept secret. Nevertheless, some prototypes are publicly available. As an example, Racelogic, a designer and manufacturer of cutting edge electronic systems for the automotive industry, developed the Video VBOX in-car video system. This equipment combines a powerful Global Positioning System (GPS) data logger with a high quality solid-state video recorder.

C. Objectives

The main goal of this project is to develop a set of software components, integrated in a single application, that handles the interface with the hardware that deal with the audio/video acquisition, as well as with the communication system installed in the car, in order to acquire audio and images/video data and send it to the pits through a wireless connection. All the acquired data should be properly encoded and compressed before transmission, so that the required bandwidth can be as reduced as possible. Moreover, the team engineer should also be able to remotely control the data transmission using a simple interface tool. Therefore, the developed functionalities are as follows:

- Two-way audio communication;
- One-way image/video communication.

To achieve this goal, the following tasks had to be accomplished:

- Signal acquisition using a microphone and a webcam;
- Signal encoding;
- Signal transmission through the communication channel.

To circumvent the physical constraints of the application (e.g.: vibration, space restrictions, etc.), as well as to handle the access and the interface with the peripherals, an embedded Linux distribution should be adopted. It is also necessary to develop an efficient user interface of this system to be used at the base station.

D. Original Contributions

The main contributions of this project are:

- A low-cost and open source solution for in-car audio and video systems: these kind of systems usually cost about 10 times more than the budget for this project;
- Offer of many design options for data encoding: the characteristics of the conceived system may be simply changed and/or configured by the programmer, by changing a simple line of code where the parameters of the codecs are set;
- Flexibility, quality and low transmission rates.

II. SYSTEM ARCHITECTURE

The system architecture is based on a client/server structure, implemented using a direct host-to-host communication network. It is composed by two nodes: the mobile station (installed in the racing car) and the base station (Pit Wall). Figure 1 shows a block diagram that illustrates the several implemented software modules that constitute the mobile station. Figure 2 shows a block diagram that illustrates the several software modules that constitute the base station.

A. Requirements

1) Physical constraints: Similarly to most components in a Formula One racing car, the in-car audio/video communication system must be as small and light as possible. In addition, the system must be robust to vibration.

1In Formula One Pit Wall is where the team owner, managers and engineers stay during the race and communicate with the driver.
Fig. 1. The mobile station.

Fig. 2. The base station.
2) Communication system: The data/messages exchanged in the car/pit network must use a pre-defined protocol. Since the communication is achieved through a Wi-Fi connection, network limitations such as a short communication range (about 60 m) and a 54 Mbps maximum data rate must be taken into account when implementing the system.

3) Bandwidth Requirements: All the acquired data must be properly compressed before transmission, so that the required bandwidth can be as reduced as possible.

4) Computational restrictions: The executed adopted software applications shall be as “light” as possible and the processor must be fully compatible with Linux operating systems, in order to easily handle the access to the peripherals.

5) Operating system: Due to the car rough conditions, such as vibration, the operating system should be installed in a USB flash-disk, rather than in a conventional hard drive. Moreover, USB flash disks allow the storage of the acquired data, since they are not read-only devices. The adopted operating system must also offer the required drivers for audio/video acquisition and for the webcam.

6) User interface: The user interface shall be as simple as possible and must allow the user to configure the most important parameters concerning audio and image/video: audio sampling rate, audio sample resolution, frame resolution, frame rate, image resolution and image refresh rate.

7) Webcam: The adopted camera must have a USB interface and must be able to acquire images/video at a high frame rate. Another important feature to be referred as well is that the camera must be USB Video Class (UVC) compatible.

8) Budget: The budget imposed by the FST championship is very low. Consequently, the hardware that composes the communication system should be acquired with particular attention to this requisite.

B. Racing Car (Mobile Station)

The mobile station includes an ultra compact Pico-ITX board (shown in Figure 3) connected to a Wi-Fi Universal Serial Bus (USB) network adapter. This computational board is responsible for acquiring the audio and image/video signals from the car (through a microphone and a webcam), encode and compress the acquired signals, store the encoded data in a local flash memory and/or send it to the laptop/monitor at the pits through the wireless network. The flash memory installed, also contains the operating system and the application software that controls both the microphone and the webcam and encodes the acquired signals.

1) Operating System: One of the advantages of using an operating system is that it greatly facilitates the access to the peripherals, along with other procedures that control the resources of the board.

When choosing the operating system to be adopted, the relevant features that should be considered for this particular project are:

- The ability to be installed in a USB flash-disk drive, with a minimalistic configuration, in order to circumvent the physical constraints (in particular, the vibrations).

Since the cost is a main concern of this project, the first choice favored the selection of a Linux distribution over a Microsoft operating system, because its cost is almost free. Initially, there were two possibilities that were considered: the iMedia Embedded Linux [8] and Knoppix [11] distributions. They have quite similar features concerning the requirements of this project. Nevertheless, Knoppix was the chosen distribution, mainly because it is more popular and presents a higher development activity.

![Fig. 3. EPIA-P700 accessories.](image)

C. Pit Wall (Base Station)

The base station consists of a PC (laptop), with an integrated Wi-Fi interface. The laptop at the pit wall is used to control, monitor and store the data received from the board installed in the racing car.

The user controls the signal acquisition by “remote control”, through a simple interface. In other words, the user at the pit controls, in real time, what is happening on track, by controlling the behavior of the board in the car. He decides which data to acquire, when and how the data is received (in what format).

D. Signal Encoding

Signal encoding is the process of putting a sequence of data into a special format suitable for transmission or storage purposes. The main goal is to represent data as efficient as possible (less amount of bits), with the best quality.

After acquiring the data there is the need to compress it, mainly because it is required to reduce the memory space and the transmission bandwidth needed to stream it. However, when transmitting compressed data, one must ensure that both the sender and the receiver understand the encoding scheme. Designing this data compression schemes involves many factors such as:

- the degree of compression;
- the amount of distortion that may be introduced;
- the computational resources required to compress and uncompress the data.

Usually, audio can be easily compressed at a 10:1 ratio without perceptible loss of quality. For video the compression ratio is even higher, about 100:1. Such rates are more difficult to be achieved for still images: with a 10:1 ratio the quality loss is already perceptible.
1) Audio: Along the past decades, several audio compression algorithms have been proposed to enable audio data to be saved more efficiently. A compressor codec takes an original uncompressed audio track and exploits its intrinsic proprieties in order to reduce its size. Hence, because of the smaller size of the encoded files, the speed requirements of the storage devices may be greatly reduced, as well as transmission bandwidth. To playback the compressed audio data, a decompression algorithm is generally used.

Pulse-Code Modulation (PCM) is often used in the context of digital telephony, to represent speech signals (8000 Hz sampling rate, 8-bit quantization accuracy and 64 kbps data rate). To reduce the bandwidth or storage space required to accommodate PCM audio signals, several data compression techniques may be applied. Among the most often used compression formats are the MPEG-1 Layer III (MP3) [1], the Advanced Audio Coding (AAC), the Ogg Vorbis [2], etc. These are lossy encoding techniques and they are quite efficient at compressing data.

2) Images/Video: There are several different techniques in which image data can be compressed. These techniques may be lossless or lossy. For Internet use, as well as for other applications with limited channel capacity, the two most common compressed image formats are the Joint Photographic Experts Group (JPEG) format (usually lossy) and the Graphic Interchange Format (GIF) format (lossless).

Video compression can be seen as a combination of spatial image compression and temporal motion compensation techniques to reduce the redundancy and irrelevancy of the image data. The redundancy is presented as correlated and predictable information between neighboring pixels (spatial redundancies), or similarities between adjacent frames (temporal redundancies). Its reduction generally conducts to lossless compression methods. On the other hand, the compression methods that reduce irrelevancy take advantage of the fact that part of the information is imperceptible to the human eye or even insignificant to the human brain. As a consequence, these techniques give rise to lossy compression methods.

Several video coding standards have been proposed in the last three decades, since the ITU-T H.120, presented in 1984 to the ITU-T H.264, presented in 2003.

Below is a list that represents the evolution of video coding standards:
- ISO/IEC MPEG-1 (1990)

As it was referred before H.264 is the latest digital video standard and is also expected to become the video standard of choice in the coming years. However, despite presenting a much greater encoding efficiency than MPEG-4 Part 2 (it needs fewer than 50% of the bits to encode a video sequence), it requires much higher computational requirements [6]. As a consequence, the strict computational restrictions imposed by the adopted computational system led to the selection of MPEG-4 Part 2 encoder to be implemented in the conceived system.

E. Communication Link

Nowadays there is a wide variety of different wireless data communication technologies available. Some examples are Ultra-wideband (UWB), Bluetooth, ZigBee, and Wireless USB, suitable for use in Wireless Personal Area Network (WPAN) systems. They are usually applied for short range communications, between paired devices that are typically controlled by a single user. For broadband applications, Wi-Fi is the most successful technology intended for use as a Wireless Local Area Network (WLAN) system. For such reason, this project was focused on the Wi-Fi technology.

III. AUDIO ACQUISITION AND ENCODING

This section addresses the basics of audio programming in Linux. The three main operations that will have to be considered are capturing (recording), playing and encoding. The first two are implemented by handling the Digital Signal Processor (DSP) device. The aim is to implement a number of applications to record, play and store the sound data for further processing. Then, it will be presented the adopted solution to compress and store the audio information. At the respect, it will be described the usage of some of the current publicly available libraries of applications to encode and compress the sound signal.

A. Interaction with the Operating System

The most common approach to control a device under Linux consists in using a small set of functions, usually called system calls, implemented by the operating system’s kernel. The open system call establishes the access to the device, returning a file descriptor to be used for subsequent calls. The read and write functions receive data from and send data to a device, respectively. Finally, the ioctl routine is a catch-all function to perform other operations that do not fit in the read/write model.

B. Linux Sound

Most Linux distributions include one of the following two sound drivers: the older Open Sound System (OSS) [13], which works with every UNIX-like system, and the newer Advanced Linux Sound Architecture (ALSA) [4]. ALSA provides better support for Linux and offers more features. Moreover, it is regarded as allowing a faster driver development. For such reason, this project was focused on the ALSA system.

C. Sound Programming in Linux

Programming the sound interface in Linux usually corresponds to the development of software code that implements the capturing (and recording) and playback procedures.
D. Compressed Audio Encoding

MP3 is currently the most popular. Nevertheless, unlike MP3, Ogg Vorbis is patent and license-free.

For encoding using the MP3 file format, LAME [2] software framework is currently considered one of the best encoders at mid-high bitrates and at variable bitrate (VBR).

Oggenc is part of vorbis-tools and it is the official Ogg Vorbis encoder by the Xiph.org Foundation. Both the LAME and the oggenc codecs are open-source.

Table I shows the variation of the bit rate of the compressed files for several sampling rates. Results show that LAME compresses with a lower bit rate than oggenc most of the time, and with the same quality. As a consequence, it was considered that the LAME encoder is the best solution, taking into account the objectives of this project.

<table>
<thead>
<tr>
<th>Sampling Rate (kHz)</th>
<th>Average Bit Rate (kbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>13</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>44.1</td>
<td>64</td>
</tr>
</tbody>
</table>

### Table I: Comparison of the LAME and Oggenc Encoders.

E. Implementation of the Audio Acquisition and Encoding Module

The audio acquisition process and the audio encoding process can both be executed in parallel, each one running in a different thread. To transfer the acquired sound data from the acquisition (capture) process to the encoding process, it was decided to adopt a “named” pipe, provided by the operating system. Such approach allowed an easy integration of the developed software with the adopted open-source encoding tools that were selected to conceive the implemented system. As such, the capture program records audio data and writes it to a “named” pipe (NP1), while the encoder reads the data from the pipe and compresses it. Meanwhile, the compressed data can be sent to another pipe (NP2). Subsequently, such data will be read from NP2 and sent to the communication channel.

In order to achieve this goal, the following tasks must be accomplished:

1) Define a static buffer to temporarily hold the sound data (unsigned samples);
2) Create the pipes;
3) Open the DSP device for reading;
4) Set the audio parameters with the required ioctl system calls. The parameters to be set are the sample resolution (8-bit), the number of channels (1) and the sampling rate (configurable between 8 kHz and 44.1 kHz);
5) Open the “named” pipes with the open() system call. NP1 is opened for writing and NP2 is opened for reading;
6) Run the processing loop. Within the loop, execute the following tasks:
   a) Read the sound data into the buffer;
   b) Write the data to the pipe (NP1);
7) Run the encoder (in another process) to encode the data from NP1 and send the compressed data to NP2.
8) Send the data from NP2 to the base station.

IV. IMAGE ACQUISITION AND ENCODING

This section discusses the implementation of a computer program to acquire and encode a still image captured with a camera device in the Linux operating system. The conceived program makes use of an internal kernel API designed for such purpose: the Video4Linux2 application programming interface (API) [5]. The aim is to implement an application that programs the capture interface (a Video4Linux2 device) in order to acquire images from the webcam. Then, two alternatives will be presented to store the image information: compressed and uncompressed. Finally, it will be described a tool to encode and compress the image data.

A. Interaction With the Operating System

1) Drivers (V4L, V4L2): V4L2 is the second version of the V4L API. Essentially, the V4L API is a kernel interface for analog video capture and output drivers. V4L2 was designed to support a wide variety of devices, although only some of which are truly “video” in nature [7]. The video capture interface grabs video data from a tuner or camera device. This interface will be further emphasized in this section.

2) Usage of the V4L2 Devices: Overview: The interaction between the applications and the driver follows four major steps:
   - Opening and closing devices;
   - Querying capabilities;
   - Image format negotiation;
   - Read from or write to a device - Input/Output (I/O).

3) Input/Output: The V4L2 API defines several different methods to read from or write to a device. All drivers exchanging data with the applications must support at least one of them. The classic I/O method based on the read and the write system calls is automatically selected after opening a V4L2 device. When the driver does not support this method, attempts to read or write will fail. Moreover, there are two streaming methods, either with memory mapped or user buffers.
B. Capture Interface

With the video capture interface, applications can control the capturing process and move images from the driver into user space. In the following, the common API elements and basic concepts, particularly applied to this interface, will be presented.

1) Driver Parameterization:
   a) Opening and Closing: To open and close V4L2 devices, the applications use the open and the close system calls, respectively.
   b) Querying Capabilities: In order to read images, the device must support at least one of the read/write or streaming I/O methods. In the particular case of the Logitech QuickCam Pro webcam device, which was used in this project, only memory mapping is supported.
   c) Image Format Negotiation: The result of a capture operation is determined by cropping and by the image format parameters. The former selects an area of the acquired frame while the latter define how images are stored in memory (i.e. Red, Green, Blue (RGB) or YUV formats), the number of bits per pixel, the image width and height. Together they also define how images are scaled in the process.

C. Image Formats

YUV is the native format of TV broadcast and composite video signals. It separates the brightness information (Y) from the color information (U/Cb and V/Cr). The color information consists of blue and red color difference signals. The green component is reconstructed by subtracting U and V from the brightness component.

There are several subsampling schemes for the YUV format, but only three are considered relevant for this project - 4:4:4, 4:2:2 and 4:2:0. Despite the wide set of digital image formats available, the webcam device that was considered for this project only supports the YUV 4:2:2 format.

Moreover, most image formats (e.g.: Portable Pixel Map (PPM), etc.) adopt the RGB color space in their encoding, which requires the implementation of a conversion procedure between the YUV 4:2:2 and the RGB sample formats. For such conversion it is necessary to first convert the YUV data to YUV 4:4:4, and then convert from YUV 4:4:4 to RGB.

D. Compressed Image Encoding

Several image compression formats are currently available, some of them represent the original image without introducing any degradation, such as the GIF, Portable Network Graphics (PNG) and the Tagged Image File Format (TIFF). However, their lossless nature compromises the achieved compression level. As a consequence, some lossy compression image formats have been considered, such as the JPEG, providing a much greater encoding efficiency.

A useful property of JPEG is that the degree of lossiness can be varied by adjusting some compression parameters. This means that the image maker can trade off the file size against the output image quality. Another interesting and important aspect of JPEG is that decoders can trade off the decoding speed against image quality, by using faster but inaccurate approximations to the required calculations.

For encoding using the JPEG format, the cjpeg encoder was adopted. It is part of a coder/decoder package (cjpeg and djpeg respectively). Cjpeg compresses the image file and produces a JPEG/JFIF file on the standard output. The currently supported input file formats are: PPM, Portable Gray Map (PGM), GIF, Truvision Advanced Raster Graphics Array (Targa), and run-length encoding (RLE). The image quality is adjusted by setting the -quality option.

Table II shows the resulting file size, after applying the cjpeg encoder to encode a lossless uncompressed PPM image file, considering the amount of image data being acquired with the -quality option set to default (75).

By comparing between compressed and uncompressed data, it is possible to verify that the results are better than the initially expected before testing (the 10:1 ratio previously described in Section II-D).

<table>
<thead>
<tr>
<th>RESOLUTION</th>
<th>DATA RATE (KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>QCIF (176 x 144 pixels)</td>
<td>74.277 3.16</td>
</tr>
<tr>
<td>CIF (352 x 288 pixels)</td>
<td>297.027 6.885</td>
</tr>
<tr>
<td>640 x 480</td>
<td>900.027 26.256</td>
</tr>
</tbody>
</table>

E. Implementation of the Image Acquisition and Encoding Module

The image acquisition and encoding module was implemented by a software routine divided in two functional blocks. The first is responsible for the configuration of the video capture device and related negotiations with the V4L driver, required after opening and closing the device; querying capabilities, image format negotiation, etc. The second block is responsible for acquiring and encoding the captured frames. To properly control this cycle, so that images are acquired in a certain and periodic interval, a software timer and the corresponding signal handler routine was implemented. This periodic interval is denoted the image refresh rate.

In order to achieve this goal, the following tasks must be accomplished:

1) Setup both the interval-timer and the signal handler routine;
2) Open the video capture device (/dev/video0);
3) Initialize the video capture device:
   a) Query the device capabilities;
   b) Negotiate the image format;
4) Initialize the memory-mapping scheme:
   a) Request the driver to accommodate the required frames in memory mapping mode;
   b) Setup the buffers characteristics;
5) Run the acquisition loop, by executing the following tasks:
   a) Image capture:
      i) Request the driver to enqueue the frame buffer;
ii) Start the acquisition chain;
iii) Wait and dequeue the acquired frame;
iv) As soon as the frame is processed, re-enqueue the buffer again;
b) Image encoding:
i) Convert the acquired YUV 4:2:2 image into the YUV 4:4:4 pixel format;
ii) Convert the YUV 4:4:4 image into RGB color space;
iii) Add the PPM header to the RGB image data;
iv) Run the cjpeg encoder, in order to encode the PPM image data into a JPEG image;
c) Send the encoded data to the base station;
d) Wait until the next interruption of the timer.

V. VIDEO ACQUISITION AND ENCODING

This section discusses the main concepts about video programming in Linux. In the following, it will be provided a brief description about the main operations that had to be considered: video acquisition, and encoding.

A. Video Programming in Linux

Video can be defined as an ordered sequence of still images. As such, the involved procedures included the implementation of a software application that programs and interacts with the video capture interface provided by the V4L2 driver. A detailed presentation of the interface provided by this driver was already presented and discussed in the previous section. In order to acquire video frames from the webcam, the involved tasks and configurations are exactly the same: opening and closing the device, querying the device capabilities, image format negotiation and read the frames. The only difference refers to the implementation of the acquisition loop, where the adopted timer has to be properly setup in order to achieve a triggering period corresponding to the desired frame rate.

As previously mentioned in Section IV-C, the frames that are acquired with the adopted webcam must be output in raw YUV format (more precisely YUV 4:2:2), due to the limitations of the camera. Nevertheless, these frames shall be used either to playback in the base station (laptop) or to be encoded and compressed. However, most video players and encoders only support the YUV 4:2:0 format in their most basic parameterizations. Therefore, the conceived application must start by converting the acquired frames, from the YUV 4:2:2 to the YUV 4:2:0 format

B. Compressed Video Encoding

The selected video compression standard was the MPEG-4 Part 2 video encoder. MPEG-4 Part 2 uses motion compensation followed by Discrete Cosine Transform (DCT), quantization and entropy coding to produce a compressed bit stream.

Video Codec

For encoding using the MPEG-4 Part 2 file format, FFmpeg is a very popular software framework that incorporates several codecs. It is a very fast video and audio converter. FFmpeg is a free software with a command line interface that is designed to be intuitive. Moreover, it is open source and is always being developed and improved.

C. Evaluation of the Video Codec Performance

Table III shows the resulting file size and data rate, after applying the ffmpeg encoder to execute two lossless uncompressed YUV 4:2:0 files: one with QCIF resolution, whose size is 14.5 MB, and the other with CIF resolution, whose size is 43.5 MB.

By comparing the sizes between both the compressed and the uncompressed video data, it is possible to verify that the results for compressed data using a quantizer scale value of 10 (or greater) are closer to the results that were initially expected (the 100:1 ratio previously described in Section II-D). Moreover, values greater than 13 produces low quality video. Nevertheless, this is still acceptable considering the compromise between bandwidth and video quality. However, if necessary it is possible to set this value to a satisfactory value or, instead of assigning a constant value, one can set a upper limit by setting the -qmax parameter to a desired value; 10 is a good value.

<table>
<thead>
<tr>
<th>RESOLUTION</th>
<th>Quantizer Scale</th>
<th>Compressed Size (KB)</th>
<th>Data Rate (kbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>QCIF</td>
<td>5</td>
<td>598</td>
<td>367.6</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>246</td>
<td>150.9</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>148</td>
<td>91.2</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>107</td>
<td>66.0</td>
</tr>
<tr>
<td>CIF</td>
<td>5</td>
<td>199</td>
<td>1143.0</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>575</td>
<td>473.1</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>355</td>
<td>291.2</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>263</td>
<td>215.3</td>
</tr>
</tbody>
</table>

TABLE III

BANDWIDTH OF VIDEO SIGNALS.

D. Implementation of the Video Acquisition and Encoding Module

The acquisition process and the encoding process shall run each in a different thread. Therefore, both processes may be running in parallel and shall communicate with each other through a “named” pipe. As such, the acquisition module captures the video data and writes it to a pipe (IN_PIPE). Meanwhile, the encoder compresses the YUV video data from IN_PIPE and writes it to another pipe (OUT_PIPE). Therefore, in order to achieve this goal, the following tasks must be accomplished:
1) Set both the interval-timer and the signal handler;
2) Create the pipes;
3) Open the video capture device (/dev/video0);
4) Initialize device and memory-mapping;
5) Run in a loop. Within the loop execute the following tasks:
a) Start capturing (same as in Section IV-E);
b) Encode the image data:
   i) Convert the YUV 4:2:2 frame to YUV 4:2:0;
   ii) Run the encoder in order to encode the YUV video data into MPEG-4;
c) Wait until the next interruption.
VI. SYSTEM INTEGRATION AND IMPLEMENTATION

In this section it is presented the integration of the several modules that were developed and described in the previous sections. The conceived system acquires all signals in parallel, encodes and compresses the sampled data and sends them to the other node of the communication link - this shall be the server of the system, which is installed in the mobile station (car). On the other hand, there is also a client which receives the data in the base station (pit), stores it in a media device and playback to the user. The communication is achieved through a TCP socket supported in a wireless link, as will be described further in Section VI-C. Finally, the conceived user interface will be described.

A. Developed Modules

The modules that were developed in the racing car node are the following:
- Audio decoding and playback module;
- Audio acquisition and encoding module;
- Image acquisition and encoding module;
- Video acquisition and encoding module;
- Controller - Receives the parameters (from the base station) that control the audio playback and the audio and image/video acquisition.

The modules that were developed in the pit wall node are the following:
- Image decoding and playback module;
- Video decoding and playback module;
- Image/video multiplexer module - This module selects which data to playback, between image and video. This procedure is useful, since the image and video shall be shown in the same display (see Figure 6);
- Audio decoding and playback module;
- Audio acquisition and encoding module;
- Parameter and control encoding module - Reads the parameters from the user.

Moreover, the message format and protocol module was also developed in both nodes. This module will be further discussed in Section VI-C3.

B. Modules Integration

1) Racing Car Node: The flow chart presented in Figure 4 illustrates the general operation of the server process. The set of tasks implemented in this procedure can be summarize as follows:
- Create a socket to establish the communication link with the client;
- Start the execution loop - each iteration comprises:
  - The server either receives the parameters to set up the devices and start capturing (if it is the first iteration), or receives the options from the client (pit) to control the acquisition of the signals. The parameters are: audio sampling rate, audio sample resolution, frame resolution, frame rate, image resolution, image refresh rate;
  - After acquiring and encoding the data, it is sent to the client trough the TCP socket. The integration of the acquisition and encoding module was previously described in sections III-E, IV-E and V-D.

Fig. 4. Flow chart of the server operation.

2) Pit Wall Node: The flow chart in Figure 5 depicts the operation of the client process. Its procedure can be summarize as follows:
- The program is initially blocked, waiting for the user to insert the required options which set up the parameters;
- Creation of the socket to communicate with the server. The program quits in case of failure, otherwise it sends the user options to the server.
- Start the operation loop. Within each iteration, it executes the following tasks:
  - Check if the user is changing the input parameters that control the acquisition process in the server side;
  - If a new parameter value is, the control message is sent to the server;
After the data is received, the storage and playback tasks are executed;  
The user may stop the loop by exiting the program, which actually exits both the client and the server processes.

In order to playback the received data (audio, image or video), it shall be written to a corresponding “named” pipe. Furthermore, mplayer (an external application) is used to read the data from these pipes. This application allows the user to directly visualize and/or listen to the received stream, since the decoding process is executed internally.

2) Communication Protocol: The communication between the client and the server is achieved through a stream TCP socket. These sockets are full-duplex byte streams, similar to pipes and they ensure that data is not lost or duplicated.

3) Message Format and Protocol: To guarantee that the data being transmitted from the car to the pit is not lost due to eventual connection losses of the wireless communication link, a reliable communication mechanism was developed by another Instituto Superior Técnico (IST) student in the scope of his MSc Thesis - Formula Student Racing Championship: design and implementation of the management and graphical interface of the telemetry system [9]. Such communication scheme was incorporated in the transmission module of this project in order to implement a reliable packeted protocol, whose packet structure is in Table IV.

The data messages exchanged between the car and the pit include other signals acquired by several sensors in the car. Their goal is to obtain several mechanical parameters, such as tire pressure and temperature, structure deformation and suspension displacement. Since all these signals are being transmitted at the same time, it was decided to define a set of priorities in order to establish a package discarding criteria whenever the offered bandwidth or the communication fidelity is not enough to assure the transmission of the whole set of signals. As such, it was considered that the signals coming from mechanical sensors should have higher priority, since they carry very important information. The multimedia signals generated in this project are assigned the high priority (images/video messages) and the medium priority (audio messages). The IDs assigned to these signals are as follows:

- 49152 (Medium) - Audio.
- 49153 (Low) - Images;
- 49154 (Low) - Video.

D. User Interface

In the base station the user is able to control the system through the interface shown in Figure 6. The interface allows the user to set the signals parameters and control the behavior of each signal. In other words, the user may control when to transmit or receive audio signal (whenever he wants to talk to the driver); when to receive an instant image (whenever he wants to get a snapshot) and also when to receive video frames.

1) Layout: Follows a brief description of the several pop-up menus, keys and the main display.

1) Mode - Pop-up menu to choose between video or snapshot (image).
2) Text box - Text box to type the value of the video frame rate (how many frames per second) or the refresh rate (in seconds) of the images.
3) Resolution - Pop-up menu to choose image/video resolution.
when the bandwidth is not enough. As such, it is possible to achieve a better balance in the system characteristics, such as the frame resolution for webcams. However, the bandwidth requirements are met. The problem of motion blur, when capturing moving objects, is affected by motion blur, when capturing moving objects, looking for the available sampling rates. The noise can be suppressed by comparing the signals from both sources.

Another aspect is concerned with the possibility to use another type of webcam device. In particular, the use of devices with (internal) flash memory, in order to avoid the problem of motion blur.

Since the signal quality was not the main concern in this project, in contrast to other aspects such as bandwidth and the system efficiency in portable embedded systems; another possible improvement to this project is to add more options to the user interface. In such a case, if quality is a main concern (for instance), the user may be able to configure parameters that are only available to the programmer. For example, the video standard and the parameterization of the codecs.

VII. PERFORMANCE ANALYSIS AND CONCLUSIONS

The designed system for in-car audio and video capture is now a reality. The conceived application provides an interface that allows the pit team to monitor the race through the camera installed in the car and to perform audio communication with the driver via Wi-Fi. The software running in the mobile station is installed in a USB flash-disk drive, running a Knoppix Linux distribution, in a low-power VIA EPIA-P700 board that meets all the requirements such as space and weight. In fact, all components that compose the system’s hardware in the car meet these requirements.

The bandwidth constraints were overcome by efficiently compressing the acquired signals before transmitting. The audio signal is captured in 8-bit PCM format and encoded to MP3, being transmitted at about 8 kbit/s with good quality. The video signal is acquired in YUV 4:2:0 pixel format at a maximum frame rate of 14 fps and encoded in MPEG-4 Part 2. The compression ratio is about 100:1, leading to about 150 kbps bit-stream to transmit 30 QCIF frames per second, with good quality. Occasionally, the captured images are affected by motion blur, when capturing moving objects, which is a problem that could not be solved with the adopted webcam. However, the bandwidth requirements are met. The compression ratio is about 10:1 for still images.

Finally, instead of encoding the whole set of the system parameters in the source code, the conceived user interface offers the possibility and flexibility to change and adapt the system characteristics, such as the frame resolution for image and video acquisition, or the sampling rate for audio acquisition. As such, it is possible to achieve a better balance when the bandwidth is not enough.

A. Budget Report

In order to comply with the strict budget restrictions, the entire software that was adopted was either cost free, including the codecs and the operating system, or was specially developed. The hardware was also acquired taking into account the ratio performance/price. Therefore, the total expense was €294.95.

B. Future Work

An important aspect that still deserves further research in the future is concerned with the possibility to incorporate audio noise-canceling technology. This can be done, for instance by using a sound card that provides more than one digital sampling device, /dev/dsp and /dev/dsp1. This way, one can use two microphones, one facing the driver to capture the speech and the other facing in the opposite direction, looking for noise. The noise can be suppressed by comparing the signals from both sources.

Another aspect is concerned with the possibility to use another type of webcam device. In particular, the use of devices with (internal) flash memory, in order to avoid the problem of motion blur.

Since the signal quality was not the main concern in this project, in contrast to other aspects such as bandwidth and the system efficiency in portable embedded systems; another possible improvement to this project is to add more options to the user interface. In such a case, if quality is a main concern (for instance), the user may be able to configure parameters that are only available to the programmer. For example, the video standard and the parameterization of the codecs.

REFERENCES