# Music Player

**Thursday 10:00 lab**

**Date: 06.06.2019**

# Embedded Systems Team F12:

# Octavio Sales (team leader): 902769,

# Marcos Suarez: 902674,

# Aimeric Martin: 902916,

# Sonny Miel: 902918.

# Information Technology, IFE 4th semester

[Music Player 1](#_heading=h.gjdgxs)

[1. Devices Used: 3](#_heading=h.1t3h5sf)

[2. Task Assignment: 3](#_heading=h.4d34og8)

[3. Project Description 4](#_heading=h.2s8eyo1)

[4. Peripherals and interface configuration 5](#_heading=h.17dp8vu)

[4.1. GPIO 5](#_heading=h.3rdcrjn)

[4.2. I2C 6](#_heading=h.26in1rg)

[4.3. SPI 6](#_heading=h.lnxbz9)

[4.4. OLED 7](#_heading=h.35nkun2)

[4.5. FAT File System and SD card interface 8](#_heading=h.1ksv4uv)

[4.6. Joystick and Buttons 9](#_heading=h.44sinio)

[4.7. Timer and RTC 10](#_heading=h.2jxsxqh)

[4.8. Decode WAV files 10](#_heading=h.z337ya)

[5. Failure Mode and Effect Analysis 11](#_heading=h.3j2qqm3)

[6. References: 11](#_heading=h.1y810tw)

[7. Other information: 12](#_heading=h.2xcytpi)

# Devices Used:

* LPCXpresso Base Board with LPC1769 Cortex-M3,
* External SD card.

**Peripherals used:** EEPROM, OLED, joystick, buttons, timer, ADC, buzzer, SD card.

**Interfaces used:** I2C, SPI.

# Task Assignment:

Octavio Sales:

* EEPROM,
* Timer
* SD card

Marcos Suarez:

* OLED,
* Buttons (Interrupt)
* RTC

Sonny Miel:

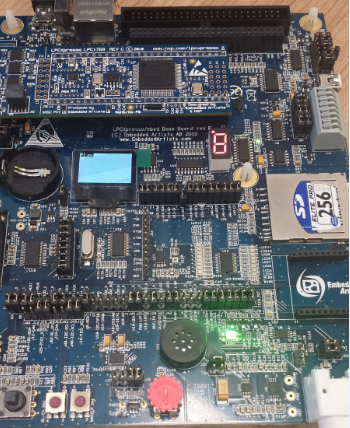
* DAC
* Buzzer
* WAV file decoder

Aimeric Martin:

* Joystick
* WAV file decoder
* UART

# Project Description

The project is an embedded music player, which allows you to enjoy all your music stored in a SD card. The system can play any song through the buzzer, in wav format, showing your playlist on the OLED and can navigate through the menu with the joystick and buttons which also allow you to resume/pause the music.



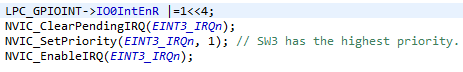
The application would not be able to perform all these functions if it did not have and configured all these peripherals that we explain below:

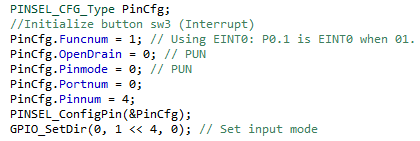
# Peripherals and interface configuration

# GPIO

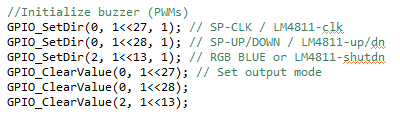
The system uses the GPIO to connect different devices as button, joystick, OLED, buzzer or DAC. Most part of them is already configured in the libraries as “Lib\_EaBaseBoard” so the application uses these functions included in it.

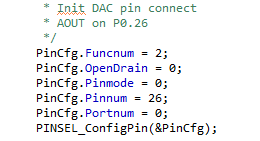
We configure the button SW3 as interrupt:





Also to connect the buzzer and DAC:

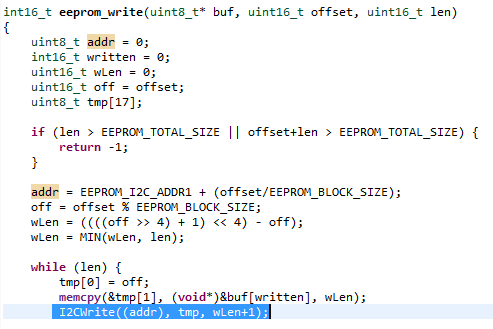




The OLED and Joystick configuration are in the functions included by the libraries previously mentioned: “oled\_init()”, “joystick\_init()”.Where also you can choose the interface of communication to use in OLED.

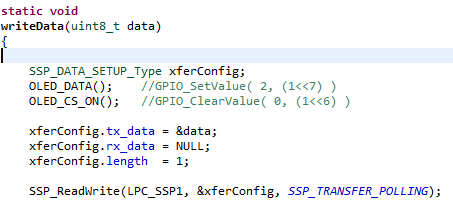
# I2C

The I2C bus is used to transfer data from/to the EEPROM, the system uses the library “Lib\_EaBaseBoard”, where as we can see it is everything defined:



# SPI

SSP is used by the OLED and also by the SD card interface. In the picture bellow we can see how the application writes the data in OLED via SSP:

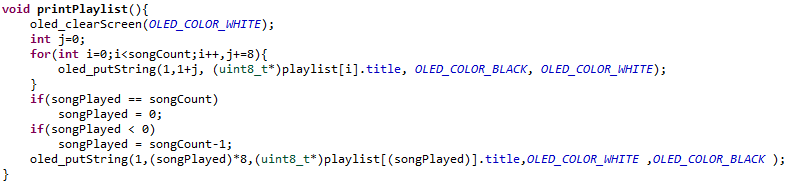


We will talk in 4.5 about the SD card and how it works.

# OLED

The OLED display can be connected either to the SPI-bus or the I2Cbus. Always insert a jumper in J44 in order to allow GPIO\_29 to control the OLED-voltage (about 11V). As mentioned before application uses SPI interface (default), must be inserted jumpers in J42, J43, J45 pin1-2 and J46 pin 1-2.

The OLED is used as main user interface, to show the SD card content as a playlist, which song is playing, and all the options. This is a fragment of the code to show the playlist:



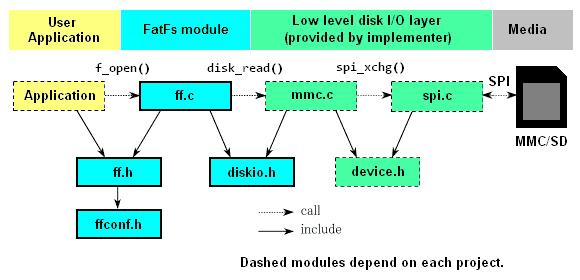
And the playlist showed on the OLED:



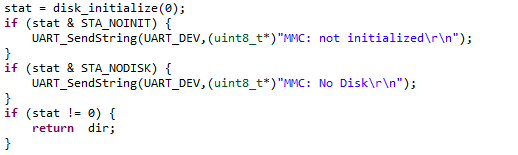
# FAT File System and SD card interface

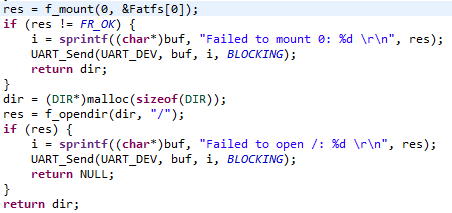
The SD/MMC memory card interface connector, J40, can be connected to the SPI bus. Insert all five jumpers in J39, in order to connect all SPI signals to J40.

The application get the music from the SD card, so before run “the user interface” the application has to read all the data from SD card. To carry this out, we use the library "Lib\_Fat\_Fs\_Sd", which gives us a module of FAT FS and a low level disk I/O layer implemented, as the scheme bellow:



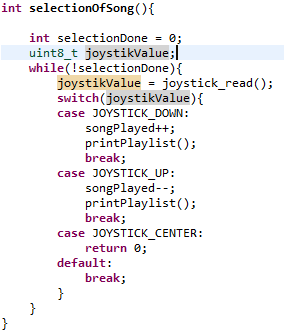
First of all, the application initializes the SD card interface and then mounts the FAT file system to manage all the data:



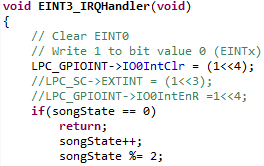


# Joystick and Buttons

The joystick and the buttons are other user interface to manage the application. The joystick allows the user to navigate from the OLED, and buttons are able pause/resume the song which is playing. Here is one example of the joystick functionality:

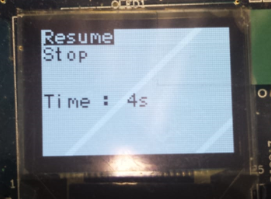


The ISR of SW3 button is shown below:



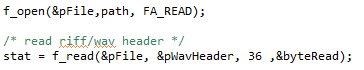
# Timer and RTC

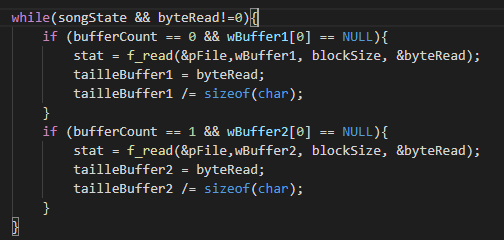
The system uses the timer and the Real Time Clock on measure time or produce delays. The main functionality of the RTC is to show the elapsed time of the song which is playing. Timer is used to produce delays for synchronize all communications.

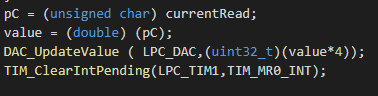


# Decode WAV files

This function consists of opening the file, reading the wav header file to make sure the file is in the good format. After that, the main process read the data from the wavFile while a timer throw an exception to make the dac read a value from the buffer. Finally, the value read by the DAC plays through the buzzer. We show Three fragments of the code where you can see how it is read the header file, and how the data is read and sent to the DAC:







# Failure Mode and Effect Analysis

The table below describes how each of the possible risks on the system's hardware would be considered.

|  |  |
| --- | --- |
| **RISKS** | **GRAVITY** |
| Microcontroller fails | Critical |
| Failure in power supply | Critical |
| SD drive | Critical |
| Buzzer | Critical |
| Joystick | Critical |
| OLED | Medium |
| Button | Negligible |
| RTC | Negligible |

In each risk considered critical, the system could not work. In the event that the microcontroller fails there would be no solution to consider. In other critical cases you could always replace/fix the previously mentioned devices, but not all of them is a good solution because the cost of replace them. However if the OLED is broken, the application could continue playing music, but it would be unable to see which song is playing or choose one of the playlist. In the case of buttons, the user can continue using the application using the joystick. And if the RTC does not work, the only thing that would have repercussions does not know how long it has been playing a song.

# References:

* <https://www.nxp.com/docs/en/data-sheet/LPC1769_68_67_66_65_64_63.pdf>
* <https://www.nxp.com/docs/en/user-guide/UM10360.pdf>
* <http://skl.it.p.lodz.pl/~morawski/SCR&ES/Guides&Schematics/LPCXpresso_BaseBoard_rev_B_Users_Guide_Rev_A.pdf>
* <http://elm-chan.org/fsw/ff/00index_e.html>
* <http://www.support.code-red-tech.com/CodeRedWiki/NXPDriverLibraries>

# Libraries used:

* MCU
* FatFs\_SD
* EaBaseBoard
* CMSISv1p30\_LPC17xx

# Other information:

In consideration with the final schedule, it has been possible to perform all the functionalities successfully:

* Joystick
* Buzzer
* OLED
* EEPROM
* DAC
* RTC
* Interrupts
* Timer
* SD card