**Hello I’m Elena and this is the final delivery of our structured programming project. My team mates are: Rodrigo Moguel, Marcos Morales and Víctor Ortiz.**

**I will start by saying that the project is 100% finished, because all the requirements defined at the beginning have been implemented and the program works with the hardware we build.**

**Next, I will give a quick explanation of these requirements with the help of the code generated from them.**

**Functional Requirement 000: When you hit a sensor, an input data is generated.**

**Functional Requirement 001: The system constantly reads the input data and stores it in a memory space.**

**These requirements are implemented specifically on line number 136 with the help of the analogread() function which, as its name says, reads the analog signal and stores it in the velocity field of the main vector.**

**Functional Requirement 002: In case where the stored data is valid, a command based on the MIDI protocol will be generated.**

**This requirement is implemented in the validatedata() and senddata() functions, since in the validation and sending process the values form the command based on the MIDI protocol are obtained.**

**Non-Functional Requirement 000: Each sensor has a minimum force assigned so the data generated by residual vibrations or interference are not valid.**

**This minimum volume is defined in line 47 and is used in the validatedata() function.**

**Non-Functional Requirement 001: The volume of the generated musical note is associated with the strength that the sensor receives when has been struck.**

**The volume of the note is represented in the program as the variable velocity.**

**Non-Functional Requirement 002: Each sensor has a unique musical note respects to the rest of the sensors.**

**Here we do the mapping of the sounds, assigning a label to the byte that defines the tone in the MIDI mapping of FL studio.**

**Functional Requirement 003: The generated command is sent to the output port of the device.**

**This requirement is implemented within the senddata() function thanks to the MIDI.sendNoteOn() and MIDI.sendNoteOff() functions.**

**Functional Requirement 004: By hitting two or more sensors at the same time, the sounds will be heard simultaneously.**

**MIDI communication is serial; therefore the data transmission is carried out one after the other, but this transmission is executed so fast that it seems that the sounds are reproduced simultaneously.**

**Non-Functional Requirement 003: The system accepts a maximum of six sensors defined from the beginning.**

**We decided to use only 5.**

**Diagram of use cases**

**The interaction between the user and the system consists in the generation of an analog signal from a stroke in a drumpad. This signal is interpreted by the system as an input data. This input data are constantly stored, validated and a command based on the MIDI protocol is generated only if the data meet the requirements.**

**Documentation**

**For documentation we decided to start using a coding standard called Cambridge, which we define ourselves and is available in the repository.**

**For this last delivery we use Doxygen as a tool to generate documentation, this document is in the repository as well as in the source code.**

**Modularity**

**Regarding modularity, our code is ordered as follows:**

**First Arduino runs, then the memory spaces of the constants and variables are reserved, then the function setup() is called, here is where all the variables and the main vector will be initialized,** **at the end of the execution of setup(), the loop() function is called, within loop() the function validatedata() is called and within validatedata(), the input data is received and validated. If the input data is correct, it will be sent to the senddata() function where it will be sent by the USB port to the PC, to be interpreted by the audio editor.**

**Our code was divided in this way, since in case we want to change some aspect of the reading or validation of data, the senddata() function will remain intact and will be able to continue functioning as it does.**

**Development process**

**Individual productivity is divided into 4 aspects, attendance, responsability, know how to follow instructions and proactivity. Taking into account these four aspects, the individual qualifications divided by delivery were as shown in the table.**