# Overview of embedded systems

Embedded systems are one of the key aspects of the Robot Orchestra project. Every instrument requires a controller, and they all need to interface with the conductor. At the heart of all embedded systems lies a microcontroller so additional care needs to be taken when selecting the appropriate one. Various parameters affect the choice of microcontroller, most notably, the clock speed, number of IO pins as well as the different interfaces used to connect peripherals. Each part of the orchestra also requires certain things, for example the conductor should be able to have a graphical user interface to allow friendly user interaction. However, there is a lot of flexibility when choosing microcontrollers for the instruments and therefore choice of microcontroller for the instruments will be based on other factors discussed next.

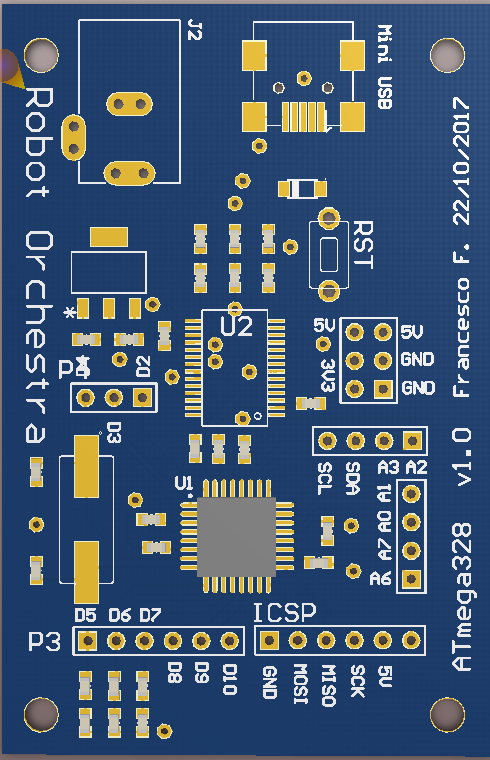
As the project is also used to showcase our skills as well as a great way to apply them whilst learning new things, the majority of the microcontrollers and peripherals will be chosen with that in mind. Moreover, using diverse programming languages, such as C,Python and Labview will also allow us to improve on languages we have previously used and learn new ones. To conclude, apart from the conductor which has specific needs to be met, the rest of the embedded systems will be selected with the academic aspect in mind.

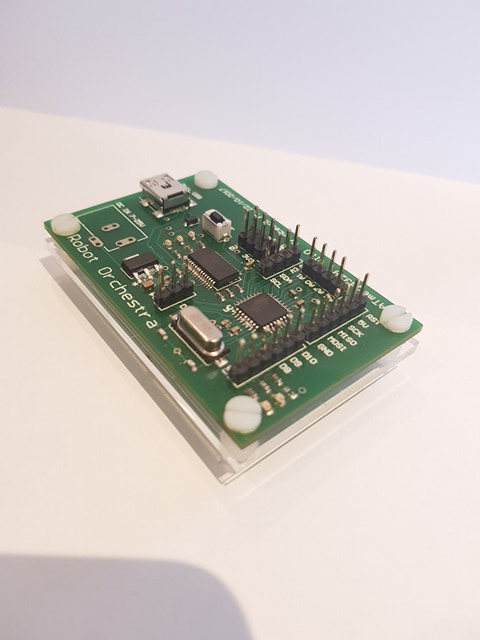
**Conductor Design**

The conductor has **three** tasks to perform. The first is to convert MIDI files into a format which can be interpreted by the various microcontrollers used (more in section **X**). The second is to transmit the converted MIDI files to the corresponding instruments, and lastly to synchronise the instruments. Furthermore, to make the conductor more elegant the team has decided to user a GUI to allow the user to interface with the conductor and must be relatively easy to transport as per the project specifications. Due to these restrictions, a computer was chosen for the task. More specifically a Raspberry Pi 3 (RPi) model B single board computer was chosen. Due to the familiarity of the team with the RPi and the complexity of the conductor the RPi was considered the best choice. The RPi is a System-On-Chip which runs a Linux distribution known as Raspbian [1]. The RPi is compatible with WiFi and Bluetooth as well as **40** GPIO pins [2]. It also has various other interfaces such as SPI and I2C which are commonly used to connect to peripheral devices. Python will be exclusively used to run scripts on the Pi and PyQt, a python based GUI builder to design the user interface. Since the RPi is essentially a computer, a screen/keyboard and mouse will all be connected.

**Instrument Microcontrollers**

Since the team will build four different instruments, at least 2 different microcontrollers will be used. As mentioned previously, since the load on the microcontrollers will be relatively low the choice will only be based on educational factors. The team has considerd the following microcontrollers for the instruments:

**ATmega328P** (Arduino Uno): The ATmega328P is the microcontroller used in the Arduino Uno platform. Arduino is an open-source platform consisting of both hardware and software. The hardware is based around the ATmega 8-bit microcontroller which is flashed with the Arduino bootloader, allowing Arduino programs, known as sketches to run on the microcontroller [3]. Arduino’s can be used to program simple tasks as well as more complex ones with relative ease, thanks to the Arduino language. The Arduino language is a simplified version of C++. Arduino also has vast online support and many libraries for various peripheral devices have been written for it, allowing fast development. The ATmega328P consists of 14 digital IO pins, 6 analogue pins, SPI/I2C/USART as well as other standard peripherals such as timers and interrupts [4]. It is also interfaceble with USB/WiFi/Bluetooth and XBee to name a few. This gives the team flexibility when developing the instruments. In order to customise the Arduino board to include various other integrated circuits needed for the instruments as well as to make it as professional as possible, a breakout board for the ATmega328P was made. The Arduino bootloader was then flashed onto it to allow the team to upload Arduino sketches to it. A prototype board was first developed. The prototype includes USB to UART FTDI chip which allows the user to upload sketches directly from a PC through USB without any additional external components as well as allowing the Arduino to transmit back to the serial monitor on the PC. Moreover SPI, I2C as well as all the digital and analogue pins have been broken out. Schematic and layout diagrams can be found in **APPENDIX X,** pictures of the final design can be found below in figure 1.1. The prototype was designed by looking at existing ATmega328 circuits as well as reading the datasheets and application notes of the various components used [5][6][7][8]. The prototype was tested and found to be working as expected. The next step will be to expand the board by adding other integrated circuits and components needed for each instrument.



*Figure 1.1: ATmega328P breakout*

**MyRio** – The MyRio is an ‘Embedded Device’ [9] developed by National Instruments. It consists of digital IO pins as well as analogue pins. It includes a Xilinx FPGA as well as a dual core ARM A9 processor. It can be programmed using both Labview as well as C [9]. The MyRio is also compatible with WiFi and Bluetooth standards and has all the peripherals which can be found on the ATmega328. The team has decided to use the MyRio as the main driver for the xylophone. The reason for using MyRio is in our project is purely educational. As the team doesn’t have any previous experience with MyRio and only limited experience using Labview it was seen as an opportunity to familiarise ourselves with it through the project. Labview has become an industry standard used by many engineering firms and learning the language is an asset for the team. Moreover, the MyRio is extremely powerful and can easily drive any of the instruments.

**NUCLEO-F303K8 –** The Nucleo board developed by ST is a breakout board for the ARM Cortex-M4 microcontroller. It can be programmed in both C and C++ (mbed) [10] although the team will use C in conjunction with ST’s hardware abstraction layer (HAL) to program it. This 32-bit microcontroller has replaced the PIC18 used by the University to teach students about microcontrollers. It is an extremely powerful 32-bit microcontroller with a vast range of peripherals. ARM has become an industry standard for microcontroller and microprocessors and the main reason the team chose this microcontroller once again purely academic. In the past team members have used the Teensy microcontroller, which is an ARM Cortex-M4 based platform compatible with Arduino. However, the team has limited, to almost no experience using Nucleo boards as well as HAL. This specific Nucleo board is the smallest one made by ST and is therefore perfectly suited for our project. Moreover, the design of the board means that a PCB can be designed to fit the Nucleo board as well as any other IC or peripheral devices needed to drive the instrument.

**High Level Overview**

Figure 1.2 High Level System Overview

**GUI 🡪 RPI 🡪 MIDI2Text🡪instruments:**

* **Channel 1: Xylophone**
* **Channel 2: Steppers**
* **Channel 3: Piano**
* **Channel 4: Pan Pipes**
* **Channel 5: Tesla**

**Conductor**

The structure of the embedded systems has been mentioned briefly in the previous section, however this serves as a more comprehensive description. Figure **X** depicts a flowchart showing the hierarchy of the system. First and foremost, the Raspberry Pi conductor takes input from the user through it’s GUI. The GUI takes in a few parameters, the file location of the MIDI file that the orchestra will play, the file location which the MIDI to text output will be saved. The GUI is kept relatively simple, however if more features need to be implemented they can added in the future. Once the MIDI file is selected the song is then converted after the user clicks on the ‘Convert’ button. More on the conversion can be found in **Section X.** The converted file can be found in the specified location. This is useful for the user to check that the output matches with what is expected. Finally, the user clicks the ‘Play’ button causing the conductor to send the corresponding text file to each instrument. For simplicity, the instrument channels are static and have been predefined. The user is expected to edit the MIDI file prior to uploading it to the Pi to make sure that each channel corresponds to the correct instrument.

**Wireless Link**

The link between the conductor and the individual instruments will be wireless. The team has considered a few different options. It should be pointed out that the team is currently working on developing the instruments and therefore is not actively developing the communication between the Pi and the instruments. A preliminary research of the available options was conducted. However, an in depth look at the various options as well as testing each one will be done in semester two.

Bluetooth:

The Raspberry Pi has an inbuilt Bluetooth module capable of transmitting to other Blouetooth enabled devices [2]. All apart from the MyRio would require Bluetooth modules as they don’t have any built in. To have the Raspberry Pi transmit to multiple devices at once a piconet is set up. Up to seven devices can be connected to the piconet as slaves, with the RPi acting as the master. The group doesn’t have any experience setting up piconets, making Bluetooth rather challenging. However, prior to conducting further research into the feasibility of Bluetooth, it won’t be ruled out.

WiFi:

After researching WiFi as a transmission method the team found that it is much easier to implement than Bluetooth. The Arduino platform as well as the Nucleo board are both capable of transmitting and receiving WiFi packets using an inexpensive additional module. Since the data transmitted will be relatively simple, a User Datagram Protocol (UDP) can be used with the Pi acting as a server and the instruments as clients. The Arduino platform provides a library which converts the Arduino into a client capable of receiving UDP packets over WiFi. The Nucleo board on the other hand uses the X-CUBE-WIFI1 add-on to the STM32Cube to accelerate development of WiFi to Serial programs. The MyRio has an inbuilt WiFi card and a the MyRio toolkit makes setting up the connection straightforward. Although this hasn’t been tested yet.

Radio:

Radio is the only method which the group has experience connecting multiple devices to. The NRF24L01+ is a 2.4GHz radio module interfaceable with both Arduino and the Raspberry Pi. This specific module allows for up to 127 different modules to be connected simultaneously over different channels. Arduino and Python libraries allow easy development on either platform. The NRF2L01+ uses SPI to communicate with the microcontroller making it compatible with the MyRio, however due to the high complexity of the device a library is needed for fast development and stable operation. Another benefit of using the NRF24L01+ module is that it is much cheaper than the WiFi/Bluetooth modules.

Conclusion

The above sections serve as an overview of the main embedded systems used and how they interact with each other. Detailed information for the embedded systems used by each component of the orchestra can be found in section **X**.

[1] <http://www.raspbian.org/>

[2] <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>

[3] <https://learn.sparkfun.com/tutorials/what-is-an-arduino>

[4] <https://store.arduino.cc/usa/arduino-uno-rev3>

[5] <http://www.atmel.com/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf>

[6] <http://www.atmel.com/Images/Atmel-2521-AVR-Hardware-Design-Considerations_ApplicationNote_AVR042.pdf>

[7] <https://www.arduino.cc/en/uploads/Main/ArduinoNano30Schematic.pdf>

[8] <http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf>

[9] <https://www.ni.com/en-gb/shop/select/myrio-student-embedded-device>

[10] https://os.mbed.com/questions/60867/what-are-the-programming-languages-suppo/