Development of an application for the conversion of G-Codes aimed at the use of a custom-made 3D-Printer.

The development of a custom-made 3D-Printer in a lab environment requires a dedicated tool for its use. In particular, in order to perform a print job, the required commands are given to the machine in the form of a file called G-Code. Such file contains elementary instructions which control all machine's parameters and lead to the completion of the task.

A code for the conversion of G-Codes into a form suitable for a custom machine has been developed. In order to make it readily available for all users of the lab environment, the code has been compiled into an executable application featuring a GUI. This way, no prevouis familiarity with coding is required for the use of the machine.

Context

Additive manufacturing is a family of technologies allowing to produce three-dimensional objects starting from a CAD model, without the use of any tool or fixture.
The final object is usually obtained by depositing several layers of material, until the whole geometry is built.

3D-Printers, however, are just CNC machines and are only able to deposit material through specific planar paths.
Consequently, the translation of CAD files into viable paths, along with the management of various parameters like nozzle temperature and cooling fans, is achieved through an external software known as a slicer. The slicer software analyzes the design, breaking it down into discrete layers, or slices, and subsequently generating the necessary planar paths.
All this information is sent to the 3D-Printer through a file called G-Code. Such file is actually a list of strings containing elementary instructions that the machine will follow. The whole sequence of instructions will lead to the production of the final piece.

When building custom machines, it is necessary to generate custom G-Codes too. However, due to the complexity of the algorithms involved in geometry analysis and path generation, which have been refined over decades, developing a custom slicer software is often an impractical option.
Instead, a more pragmatic approach is generating G-Codes using commercially available software and subsequently adapting them to suit the requirements of the custom machine. This approach avoids the need for developing an entirely new slicer software, while still allowing for perfect integration with the custom manufacturing process.

Aim of the work

This project focuses on the development of a user-friendly software solution specifically designed for the conversion of G-Codes. Given its intended utilization by all lab users, it is advantageous to compile it into an executable application featuring a graphical user interface (GUI). This approach is aimed to provide a readily available and intuitive tool that can be easily utilized by all users within the lab environment, eliminating the need to possess prior familiarity with Python code.

The need for such a tool arises from the necessity to use an in-house built 3D-Printer, featuring multiple nozzles and unconventional pieces of electronics.
The development of a software for G-Codes conversion allows to properly control the machine and to develop specific routines.

Machine functioning

The machine under investigation is a multi-nozzle 3D-Printer for the deposition of liquid materials. Such materials are contained into syringes and extruded through a nozzle.

In order to start printing, the selected head has to be activated (the print-head is lowered and deposition has to start). While, in order to switch head, the one in use has to be deactivated (the print-head is lifted and deposition has to stop) and, successively, the next one has to be activated.

The deposition is based on the application of compressed air through the use of solenoid valves, which are controlled from the motherboard. In order to start extruding material, the corresponding valve has to be turned on. In order to stop the deposition, such valve has to be turned off and a second one has to be opened in order to release existing pressure.

Finally, each time a layer is printed, a layer-change routine can be added. Such routine can include the piece to be moved under a UV light source. This process is used when printing UV-curable resins, in order to solidify the printed layer before printing the successive one on top of it.

The execution of all these commands is controlled by adding specific series of strings in the G-Code or by modifying existing ones.
The python code is able to recognize the place in which transformation has to be performed thanks to trigger strings present in the original G-Code.

The choice of strings to be modified and of ones to be added requires the knowledge of G-Code language, which is specific of CNC machines. The main commands to be used have been taken from https://marlinfw.org/meta/gcode/.

Python code

The python code developed allows to convert existing G-Codes, provided by the user, into G-Codes to be used on the custom printer.
As said, this is done by adding or modifying strings when specific triggers are met.
The transformation is performed basing on few parameters chosen by the user, while original files are not deleted: a new folder is created, containing the converted version of files.

The code is made of 5 main scripts:

- Main.py: it is the script setting up the graphical user interface (GUI) and recalling the actual core of the program, that is the script that actually converts the files;
- Gcode_mod.py: it contains the functions that create a new folder and actually convert G-Codes;
- Replacement.py: it contains, in the form of a dictionary, the strings to be converted and

the corresponding strings to be added in place of them;

- Toolchange.py: it contains the strings allowing to activate and deactivate single print-heads;

- Layerchange.py: it generates the series of strings allowing to perform a specific routine at each layer change, basing on the parameters inserted by the user.

The code has been compiled into an executable application, in order to be readily available to other users. At the moment, the compiled version of the application is available for Mac platform only.

Main.py

Relevant libraries imported:

- Tkinter: generation of GUI
- Platform: detection of platform on which the script is running. If it is running on a Mac, it imports the optimized version of Tkinter
- Tempfile: query of system's temporary file folder. User's settings are stored here in order to have them available when opening the application.

Outline of the script

A brief explanation of the code will be given.

Lines 35-45

As the application is launched, the code looks in the temporary file folder for a file containing previously stored settings. If the file exists and has the correct form, such information is recovered.
This way, the user finds the settings used the previous time.

Lines 200-382

The GUI is set. It contains entries and buttons necessary to set the parameters.

- A button allows to look for the folder containing the files to be converted, calling the "openfolder_function", defined in lines 172-179;
- Two checkboxes activate (thanks to functions defined in lines 184-196) other fields and allow to insert further parameters;
- Entries to insert numerical values. The input is filtered in order to allow only the insertion of numbers (letters and spaces are not accepted).
- "Convert" button: clicking on it, the function "run_function" is called.

The dimension and position of objects are expressed relatively to screen dimension. This way, the interface should be adaptive to different devices.

Lines 51-165

Run_function: function to be run when pressing "Convert". This function performs several operations:

- Close existing dialog (error) windows, if any (lines 58-63). This avoids opening many stacked windows if repeatedly clicking on the button “Convert”.

- Creating a list containing all user’s settings present on the GUI. Some settings can be modified in order to optimize coherence (lines 67-94).
  The modification is made to avoid useless routines during the printing process. Taking into account UV curing: if the UV curing option is selected, at each layer change the printed object is moved to the UV light source, UV LEDs are turned on and then the object is moved back to the previous position.
  If the user activates UV curing but inserts a UV curing duration equal to 0, the effect would be to move the object to the UV source and back without an actual reason. In this case, the UV curing option is automatically deactivated.

- Saving the list with settings into a file located into temporary files folder (lines 67-94). This allows to transfer the necessary setting to the core of the program and also to store the settings for future use of the program. The existing file is overwritten. This function is trivial at the moment, but can become very useful for future developments of the software, when many additional settings are added.

- Error dialog windows (lines 97-126): if necessary fields are left empty, an error message is shown and the execution of the function is interrupted, in order to avoid conflicts. In particular:

  - Folder path has to exist
  - UV time has to be specified if UV Checkbox is selected
  - Extra run distance has to be specified if Extra Run Checkbox is selected

  Error windows contain a message with the corresponding error and a button which closes the window.

- Dialog window (lines 129-142): if no errors are detected, a dialog window informs that the conversion has started.

- Calling the function “Conversion” (line 145). This calls the core function of the code, that is the one that actually converts files.

- Dialog update (lines 148-165): when the conversion ends, the user is informed about how many files have been converted.

GCode_mod.py

This script contains the functions that create a new folder and actually convert G-Codes. It is called when pressing the “Convert” button present in the GUI.
The whole script is defined as a function, in order to be easily called and/or interrupted.

Outline of the script

A brief explanation of the code will be given.

Lines 26-39

At the beginning of the script, user’s settings are retrieved from the file written by the “Run_function” contained in “Main.py” script.

Lines 45-54
A “New_Gcode” class is defined. The definition of a class is useful to rapidly get the file name, change the extension of the file, read it line by line etc…

Lines 84-92
The script parses through all files in the specified directory and selects all files with “.gcode” extension (calling the function “check_gcode”, lines 71-78).
If files are found, a new directory is created (function “mkdir”, lines 60-65), in which converted files will be saved.
If no file is found, the function is interrupted without creating any folder. This way, in case the user chooses a wrong folder path, the unwanted creation of new folder is avoided.
When no G-Codes are detected, the user is informed by the GUI.

Lines 100-112
The code goes inside single files and start reading them line by line.

Lines 119-130
The code keeps trace of the tool in use and of the state of the print (deposition started, tool change required etc…)

Lines 136-147
If tool has to be changed, the code adds a sequence corresponding to this task. If the print has already started, the tool currently in use is deactivated and the new one is activated (tool_change sequence).
If the print is not started yet, no tool is active, therefore deactivation is not needed and activation sequence only is added.

Lines 153-165
The sequence to be performed at each layer change is added.

Lines 172-200
Sequence relative to retraction (pause of the deposition, needed when the tool head has to move from one point to another without printing) and de-retraction are added when the corresponding trigger is met.

Lines 208-234
When using the custom printer, some commands that would be used by commercial ones become useless. Therefore, they are removed from the converted G-Code.
In particular, extrusion is controlled by motors pushing the material in commercial machines. In the custom one, extrusion is controlled by application of compressed air. As a consequence, all commands related to extruder motor are removed.

Lines 249-254
An intermediate version of the file is created.

Line 259-272
Lines related to unwanted features are removed.

Lines 286-296
The final version of the file is saved.

Other scripts
Secondary scripts are present. Their function is to create the sequences to be added during the execution of conversion function.
The logic at their base is specific of G-Code language, and the choice of strings to be added to sequences only depends on the desired behaviour of the machine.

Use of the code

In order to convert G-Codes, the application has to be opened (or the script Main.py has to be run) and, using the “Folder path” button, the folder containing the files has to be selected. Successively, the user has to select the desired options (if any) and enter required values.

Clicking on the “Convert!” button, G-Codes will be converted: a subfolder named “converted” will be created, containing converted versions of files.