



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y SISTEMAS DE TELECOMUNICACIÓN

PROYECTO FIN DE GRADO

TÍTULO: Improvement and addition of features in 3D printing Open Source Software OctoPrint

AUTOR: Javier Martínez Arrieta

TUTOR (o Director en su caso): Javier Martín Rueda

TITULACIÓN: Grado en Ingeniería Telemática

DEPARTAMENTO: Ingeniería Telemática y Electrónica

VºBº

Miembros del Tribunal Calificador:

PRESIDENTE: Irina Argüelles Álvarez

VOCAL: Javier Martín Rueda

SECRETARIO: Vicente Hernández Díaz

Fecha de lectura: de de 2016

Calificación:

El Secretario,

To my family, specially my mother, who supported me during all these years even when I was thinking about surrender after several failures in difficult subjects.

To my tutor, who supported me and this project even without almost knowing me and the topic of this project.

To the IEEE Student Branch UPM-Campus Sur, especially to two of the members who 'cheated' me to become one of them. It has been a great time that let me grow as a person and as a future engineer but, what is even more than important, gave me lots of experiences and friends.

To Gina Häußge, my 'second tutor', who help me achieving better results and integrating my proposals to the 'official' OctoPrint repository. This gratitude also includes the community members who helped me to refine the features developed.

Lastly, to RepRap, Clone Wars, Gina, Juan González Gómez (obijuan) and all community members who demonstrated that open source does not have to mean 'impossible to make benefit of it', 'not maintained' or other bad concepts associated with the concept. Thanks to all your work it was possible to me to do something I love, like 3D printing or software development.

Resumen

El propósito del presente documento es el de resumir el proceso seguido para la implementación de nuevas funcionalidades que serían integradas en OctoPrint, un proyecto software de código abierto existente que permite al usuario el control de una impresora 3D. Dicho proceso constó de un análisis de requisitos, el diseño y desarrollo de una solución que se adaptase a dichos requisitos y la realización de un conjunto de pruebas que permitiesen comprobar que el código implementado se ejecutase correctamente, cumpliendo así con los requisitos planteados. Cada funcionalidad desarrollada se trató de integrar posteriormente en el repositorio original, del cual descarga el software la mayoría de los usuarios.

A lo largo de los últimos años, la impresión 3D ha sufrido un fuerte auge que ha llevado consigo no sólo un mayor número de modelos de impresoras disponibles o de tiendas que disponen de material que permiten montar y configurar uno de los varios posibles modelos, sino que ha conllevado también la aparición de software que, entre otros, permiten controlar la impresora y que ésta genere los objetos diseñados.

OctoPrint es un servidor web, ejecutable en distintos sistemas operativos y hardware, que permite controlar distintos modelos de impresoras 3D además de almacenar los ficheros a imprimir entre otras numerosas funcionalidades de las que dispone y al cual se añadirán un total de tres, las cuales indicarán al usuario si se detectan errores en un fichero a imprimir porque alguno de los objetos se encontrase total o parcialmente fuera del área de impresión, generar un fichero listo para su impresión a partir de un fichero de diseño y programar impresiones a cualquier día y hora elegidas por el usuario. Todas las funcionalidades aquí mencionadas se encontrarán explicadas con mayor detalle a lo largo del documento.

Para el desarrollo del proyecto se ha hecho uso de una Raspberry Pi 2, un ordenador de pequeñas dimensiones que ejecuta Raspbian (una de las múltiples distribuciones de Linux) y que además resulta muy económico teniendo en cuenta el tamaño y prestaciones de los que dispone. Aunque OctoPrint es ejecutable en otros sistemas operativos como ya se ha mencionado, se eligió Linux para conseguir un mejor ajuste al tiempo de desarrollo del proyecto y que las funcionalidades implementadas pudiesen llegar a un mayor número de usuarios, ya que se cree que la mayoría de estos ejecutan en software en ordenadores con sistemas operativos Linux instalado.

Con la intención de que el lector pueda comprender mejor la temática de este proyecto, conceptos claves y el motivo por el que se desarrollaron las funcionalidades de una determinada manera, se introducirán una serie de apartados que tratarán de conseguir dicho objetivo.

Abstract

The aim of the present document is to summarise the process followed for the implementation of new features to be added to OctoPrint, an open source software project that lets the user controlling a 3D printer. That process consisted in the requirements analysis, the design and development of a solution to meet those requirements and a set of tests to verify the implemented features, verifying that the objectives had been accomplished. Every developed feature would be later tried to be integrated in the original repository as most of the users download the source code from there.

During the last years, 3D printing has achieved not only a wider variety of printer models or a higher number of shops where the different parts could be bought to later build and configure the printer, but also for software to be released that, among other possibilities, would allow controlling a printer that would generate the designed objects.

OctoPrint is a web server, executable in different operating systems and hardware that allows controlling several 3D printer models apart from storing files to be later printed among many other available functionalities, which will warn the user if an object from a file is going to be printed totally or partially outside of the printing area, generate a file ready to be used to print from a design and schedule prints to be done at any date and time set by the user. All of the mentioned features will be found explained in more detail through the pages of this document.

A Raspberry Pi 2 has been used for the project development, a computer of small dimensions that runs Raspbian (one of the many available Linux distributions), which is pretty economical taking into account its size and characteristics. Although OctoPrint can be run under other operating systems as it has been already mentioned, the Raspbian option was chosen so as to achieve a better adjustment to the development period and also because it is thought that most of the users running OctoPrint do it using a Raspberry Pi or another computer with Linux installed.

With the purpose of a better understanding of the project topic, key concepts and the reason why the features were developed in a particular way, some parts will be introduced with the intention of achieving that goal.

Index

Contents

| | |
|---|----|
| Resumen..... | 1 |
| Abstract | 2 |
| Index..... | 3 |
| Index of figures | 5 |
| Acronym list..... | 7 |
| Motivation | 8 |
| State of the art..... | 9 |
| History of CNC machines..... | 12 |
| History of OctoPrint | 12 |
| OctoPrint's license | 13 |
| 3D printer parts | 13 |
| 3D printing phases | 15 |
| Programming methodology used | 16 |
| Internal server structure | 17 |
| Languages used..... | 17 |
| Git..... | 20 |
| GitHub | 24 |
| Working environment | 27 |
| Features added to OctoPrint | 29 |
| Warn user in case an object to be printed exceeds the global printing area..... | 29 |
| Requirements..... | 29 |
| Development process..... | 29 |
| Tests done..... | 30 |
| Results | 31 |
| Integration of Slic3r as a plugin for OctoPrint | 31 |
| Requirements..... | 32 |
| Development process..... | 32 |
| Tests done..... | 33 |
| Results | 34 |

Index

| | |
|----------------------------------|----|
| Printing schedule | 35 |
| Crontab, cron and Wget | 35 |
| System date | 37 |
| Requirements..... | 37 |
| Development process..... | 38 |
| Tests done..... | 42 |
| Results | 43 |
| Conclusions and future work..... | 45 |
| References | 46 |
| Appendix | 48 |
| System date | 48 |

Index of figures

| | |
|---|----|
| Figure 1: Main screen of Repetier | 10 |
| Figure 2: Main screen of Cura | 10 |
| Figure 3: Main screen of Pronterface | 11 |
| Figure 4: Main web page of OctoPrint | 12 |
| Figure 5: 3D printer parts | 14 |
| Figure 6: Git rebase example using default mode (commit picking)..... | 22 |
| Figure 7: Git rebase example using interactive mode | 22 |
| Figure 8: Graphical explanation of some of the contents. | 23 |
| Figure 9: Code tab of the project repository | 25 |
| Figure 10: Example of a pull request | 26 |
| Figure 11: Example network tab of a repository..... | 27 |
| Figure 12: Possible equipment connections..... | 28 |
| Figure 13: G-code analysis result | 31 |
| Figure 14: Example error message in case of exceeding an axis | 31 |
| Figure 15: README file of OctoPrint-Slic3r repository | 34 |
| Figure 16: STL slice in Windows using slicer-console | 35 |
| Figure 17: Example crontab file | 36 |
| Figure 34: Example of conflict cases | 38 |
| Figure 35: Example crontab entry with access control enabled | 41 |
| Figure 36: Example crontab entry with access control disabled..... | 41 |
| Figure 37: Printer status workflow..... | 42 |
| Figure 38: File options | 43 |
| Figure 39: API key notification error | 43 |
| Figure 40: Scheduler menu | 44 |
| Figure 41: OctoPrint settings window for access control | 44 |
| Figure 42: Warning message to confirm or cancel print | 44 |
| Figure 18: Main configuration menu of a Raspberry Pi | 48 |
| Figure 19: Internationalisation options menu | 48 |
| Figure 20: Continents list for geographic area | 49 |
| Figure 21: Capital cities list for time zone | 49 |
| Figure 22: Main menu of Raspberry Pi settings..... | 50 |
| Figure 23: Advanced options menu..... | 50 |
| Figure 24: Habilitation of I2C interface | 50 |
| Figure 25: Confirmation message for I2C habilitation | 51 |
| Figure 26: I2C kernel modules load confirmation menu..... | 51 |
| Figure 27: I2C kernel modules load confirmation | 51 |
| Figure 28: Reboot menu | 52 |
| Figure 29: RTC connection with a Raspberry Pi | 52 |
| Figure 30: I2C detection..... | 53 |
| Figure 31: Modprobe command | 53 |
| Figure 32: Addition of the RTC module to modules file | 53 |

Index of figures

Figure 33: Addition of the RTC module in rc.local file 54

Acronym list

AMF – Additive Manufacturing File Format

ASCII – American Standard Code for Information Interchange

API – Application Programming Interface

CAD – Computer Aided Design

CNC – Computer Numerical Control

CSS – Cascading Style Sheets

GUI – Graphical User Interface

HTTP - Hypertext Transmission Protocol

I2C – Inter-Integrated Circuit

INI - Initialization

ISP – Internet Service Provider

JSON – JavaScript Object Notation

LESS – Language for End System Services

MIT – Massachusetts Institute of Technology

NC – Numerical Control

NTP – Network Time Protocol

OBJ - Object

OS – Operating System

PC – Personal Computer

RTC – Real Time Clock

SHA – Secure Hash Algorithm

STL – Surface Tessellation Language

URL – Uniform Resource Locator

VCS – Version Control System

Motivation

As a member of the University's Electronics Association AETEL, also an IEEE Student Branch, I had the opportunity of learning about 3D printing when the first of the current two available printers was built and configured right after some talks given during the Open Source Hardware Convention (OSHWCON) held in September of 2012 in Madrid. During the mentioned event, I could know about the Spanish RepRap community, known as Clone Wars. Thanks to the work of their members, every user can learn everything about 3D printing, most of it in his or her native language, without the need of special technical skills to build and maintain this kind of machines. Besides, people exchange information through forums and help new users achieving their goals, fixing errors or building a 3D printer for their own by donating 3D printed parts, with the condition of doing the same for another person when the machine is built and working, in exchange of a filament spool or some other sort of collaboration with the community. Those things made me eager to work in a final degree project related with 3D printing, sharing the work done with other community members instead of finishing with something 'lost in a drawer'.

With the growth of 3D printing, lots of open source printer variations and software appeared, letting the user decide among a wide variety depending on available features and plugins, ease of use, etc. With the assembly of the association's first 3D printer came the research of software that allowed doing the tasks as easy as possible by any of the printer users. With the research finished, Pronterface (see more information in 'State of the art') was the chosen one following the recommendations of some of the experienced community members. Unfortunately, most of the time prints were done using the computer of any of the members, so only one person could manage the printer and, what was worse, in case of willing to go somewhere else with his or her laptop, that would not be possible as the print must be finished before disconnecting the printer and finishing its control. Due to that fact, a new research for software that could solve that problem was done, being OctoPrint the best solution to cover the requests of the different users of the 3D printer. As an additional advantage in comparison with other software solutions, it was a web server executable under Windows, Linux (including Raspberry Pi, the target PC used for this project) and Mac OS, being much easier to transport the Raspberry Pi board to different places and at less cost in comparison with buying a laptop or a computer that also occupies much more space. For all of those reasons, OctoPrint seemed to be a good choice to work at and collaborate as it is widely supported by the community and permits more flexibility than other available open source software solutions.

Another motivation that made me want to work in this project was learning new programming languages commonly used in the current industry (JavaScript and Python), being able to apply the concepts learnt during the university degree to the new languages, being the main job to learn the syntax to be applied for those already known concepts. In addition, I was able to learn git, a very powerful tool for projects (mainly software, but also

for other types of repositories) that let me track changes done and repair errors occurred during the development process like recovering lost files.

About the language used for document writing, the reason why I decided to write these pages in English language was to keep practicing the language as well as making it readable for foreign readers interested in this topic as English is accepted as the universal language.

In conclusion, I believe I have chosen a good challenge as a final degree project that not only will let me widen my knowledge in some software languages and Version Control Systems (VCS), but will also be useful by the 3D printing community, having been able to participate in a real world project.

State of the art

3D printers have existed for more than twenty years for both personal and professional use, but it is not until 2004 that they became popular with the creation by Adrian Bowyer of the RepRap concept of a self-replicating, open source 3D printer that allowed this technology to be accessed by the public, becoming increasingly popular. With the growth of 3D printing usage, not only did new Computer Aided Design (CAD) tools and g-code generators appear, but also new printer models and software that permitted to control them, being able to generate almost any object that could be imagined. Besides, many of them are free, open source software maintained by companies and/or the user community.

As mentioned in ‘3D printing phases’, there is a phase where the design is ready to be printed. To complete that phase, many available software solutions cover the user needs, some of them with additional features to also complete the second phase of 3D printing (generation of g-code from an STL design). Some of the most known and used by the community are:

1. **Repetier:** This software presents different distributions depending on the purpose of the user. These distributions are:
 - a. Repetier host: To be used within the same computer where the printer is going to be controlled.
 - b. Repetier server: This is a web server that allows connections from clients that are outside of the host that controls the printer. The purpose of this distribution goes further from connecting from outside the host, permitting the individual control of one or multiple printers by just opening a new tab in the web browser.
 - c. As firmware: Every printer requires having as a part of it a control board that handles both electronics and communication with the controlling computer. It makes easier to the user configuring through a web page the firmware

State of the art

settings that will be uploaded to the printer's control board so as to achieve its goal.

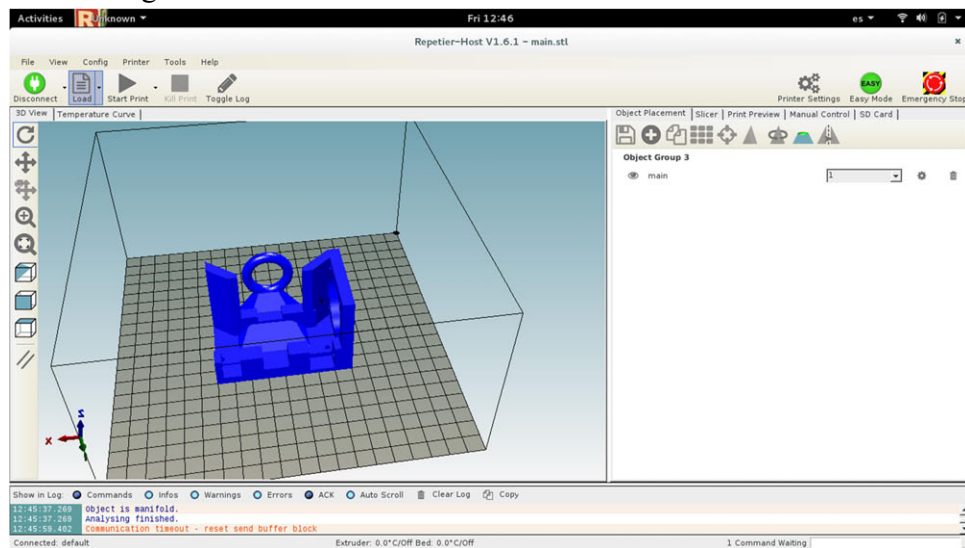


Figure 1: Main screen of Repetier

2. **Cura:** Created and maintained by Ultimaker, this software also integrates both second and third phases of 3D printing. To make it more usable to the end user and avoid obtaining bad results, it is possible to execute the program in simple or expert mode, changing the number of configurable settings available to the end user so as to fit the custom needs.

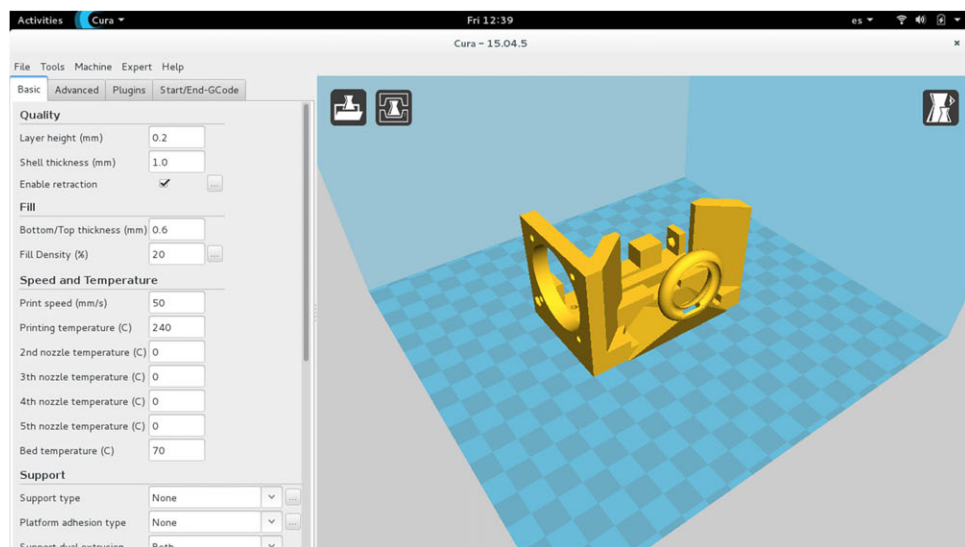


Figure 2: Main screen of Cura

3. **Pronterface:** Written entirely in Python language, it is a very simple program to use that, as others, lets the user do all parts but CAD designing. Printer control and file

management can be done within the same window and the menu can be changed by adding or removing menu options by just clicking a button.

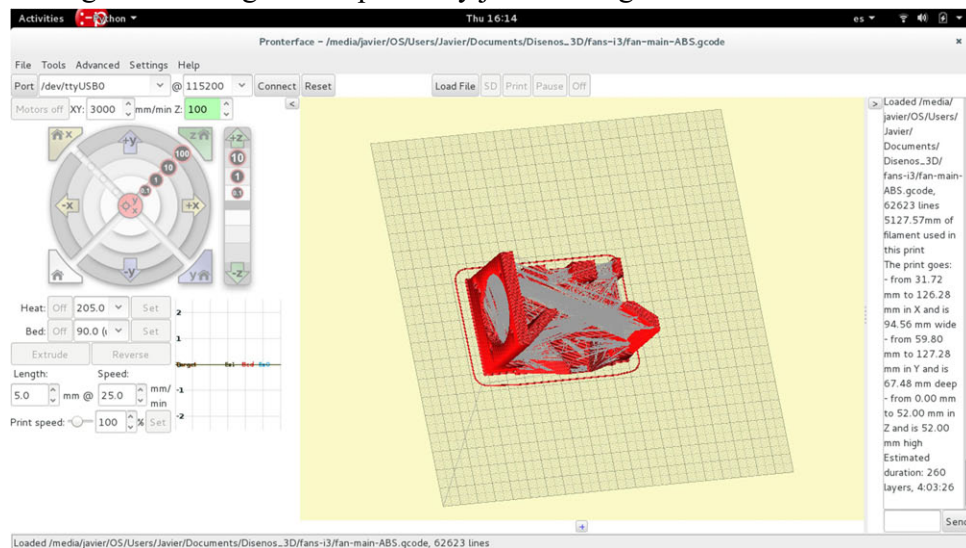


Figure 3: Main screen of Pronterface

4. **OctoPrint:** As many other software solutions, it lets the user control a 3D printer and also complete the second phase of 3D printing with the help of available plugins for the user. As the main difference with respect to the rest of options available, OctoPrint was designed to be run as a web server, being even accessible from outside the network where the server is running if no security restrictions avoid it, giving more flexibility and possibilities in comparison with other software solutions. Furthermore, a virtual printer can be configured, avoiding the need of a real one to develop and test new features or plugins.

History of CNC machines

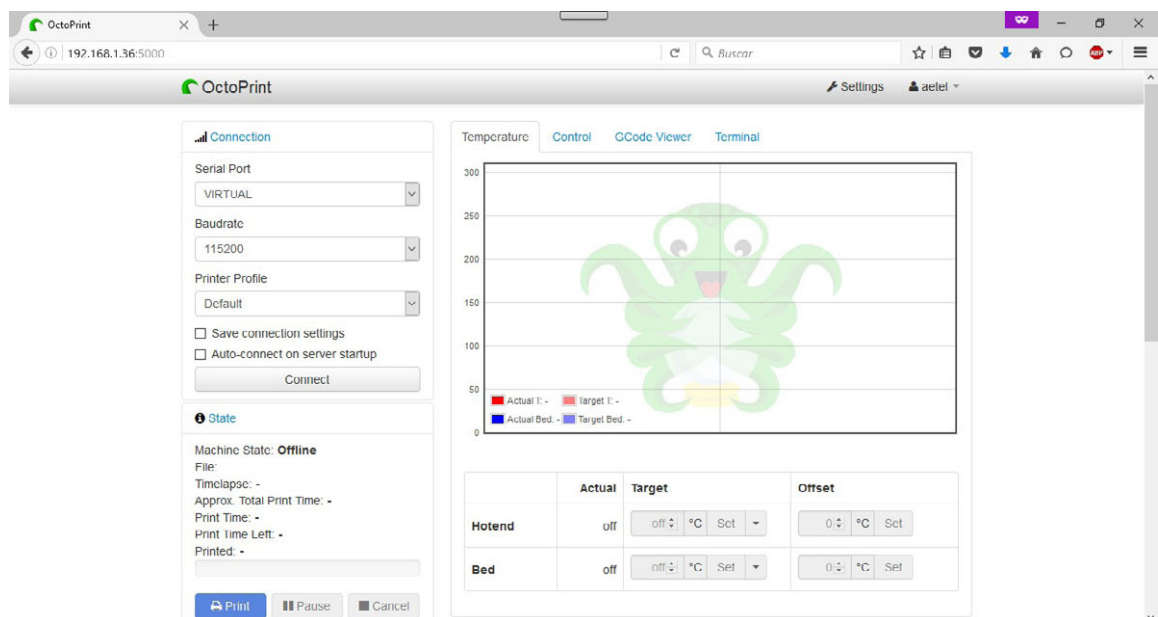


Figure 4: Main web page of OctoPrint

As can be seen, all of the available software tools give the user the control of every printer part, being the differences the type of printers that can be controlled, the Graphical User Interface (GUI) design and the extra features and plugins available in the software.

History of CNC machines

Computer Numerical Control (CNC) machines are the extension of Numerical Control (NC) ones, which were built for the first time during the 1940s and 1950s with the purpose of creating automated machines capable of doing certain tasks more precise and cost effective as less human intervention would be required. As the years passed by, the utilities of these sort of automated tools augmented to the level that nowadays it is possible to do tasks from semi-automatically manufacture cars to 3D print, the area related with this project.

History of OctoPrint

Created and developed by Gina Häußge, OctoPrint is an open source software application whose first commit was done in December of 2011. Although its main target Operating System is Raspbian, designed for Raspberry Pi (the computer used for this project), it can also be deployed under other Linux distributions, Windows or Mac OS. Besides, all the source code is publicly and freely available at GitHub platform since its beginning, so any user can not just use it, but also collaborate by fixing bugs or by adding functionalities. During part of its existence, OctoPrint was sponsored and maintained by the Spanish company bq with a small developer team led by her but, due to economic reasons of the

company, its author had to start searching for funding in portals like Patreon (for monthly donations) and Paypal (for single donations) that could help her creator or maybe a small team to keep working on OctoPrint, as happened when bq supported the project.

OctoPrint's license

OctoPrint has been licensed under the GNU Affero General Public License (AGPL). This license guarantees the developer to freely distribute and change the work, making sure that it remains free for all of the software uses. The terms and conditions to be remarked are:

1. If not specified, the developer can decide which version of the license applies to the software. If specified, the indicated version or an updated one must be applied.
2. Every work or variation based on the software must follow this license and cannot be changed for any of the parts of the work.
3. Developer has no obligation of bug fixing or software maintenance. However, it is possible to offer warranty and maintenance applying a fee for those services.
4. Copies can be distributed applying a fee or free or charge.
5. 'Free' does not mean 'at no charge', it means that the user has the freedom of distributing copies of the software, changing its code or adapting a part of it in new programs.
6. Installation instructions and software requirements must be easily accessible by the user.

For more information, all terms and conditions can be found at <http://www.gnu.org/licenses/agpl-3.0.html>

3D printer parts

With the purpose of a better understanding of 3D printing, the parts of a 3D printer are explained in this section. These parts are:

3D printer parts

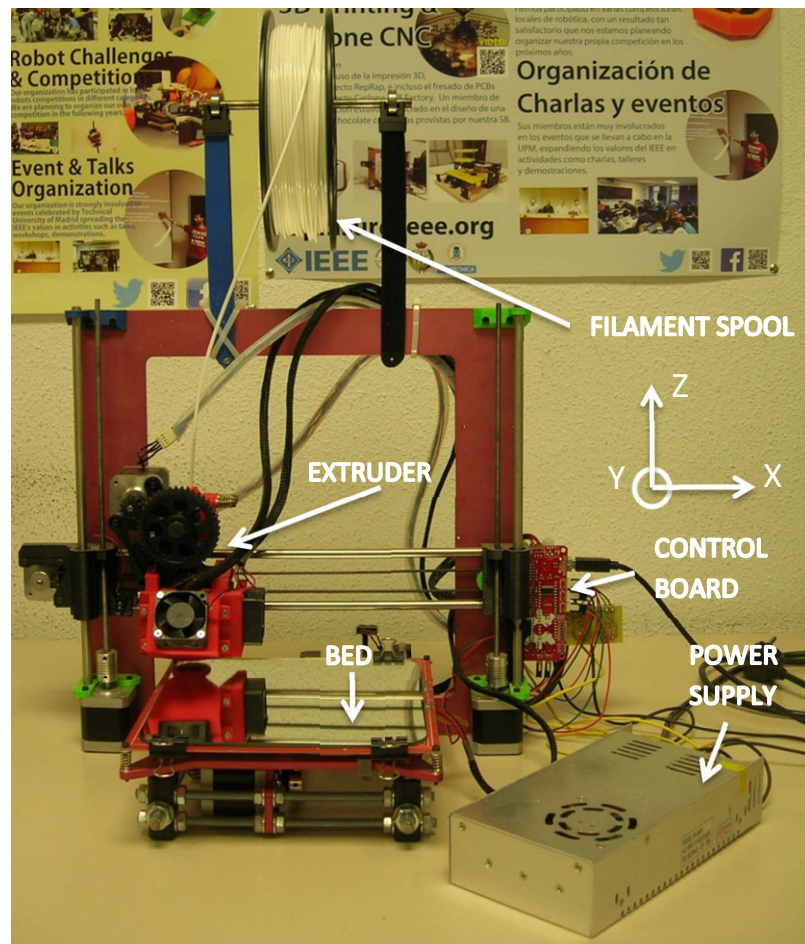


Figure 5: 3D printer parts

Extruder: This part is responsible for extruding the plastic that will be heated and placed in bed or retracting it. It has an axis assigned, named E.

Axes: The 3D printer follows the cartesian axes X, Y and Z, X and Y being for width and depth of the bed and Z for height.

Bed: This is the part where the object or objects printed are placed. The one that can be seen in the image is squared but the shape can be different depending on the printer, though.

Control board: This is the ‘brain’ of the machine, the part that communicates with the controlling computer. It translates the instructions into signals for the actuators or sends data received from sensors to the controller computer.

Sensors: These are electronic components used to measure temperature or to detect that the zero or starting point of an axis has been reached.

Actuators: These are motors that transform the signal received into movements or heat the nozzle, the part where the extruded material is heated.

3D printing phases

It may be thought that printing an object is as simple as printing a document, where just by clicking a button and setting a few parameters is enough to get our document or, in this case, our object or objects printed, but in reality it requires a longer process that can be divided in the following three phases:

1. **Object design:** Once the user wants to print one or multiple objects, the first step is the usage of a CAD tool that will allow transforming that idea into something real and printable, as the final object is placed in a flat bed. That means, not every object can be placed in the same way to be manufactured and not every object can be printed as it may have been thought (e.g. something to be printed in the air without support material). The design must be saved as an STL file, which contains information about the object's shape that will be used during the second phase.
In several cases, it is possible to get designs from webpages like Thingiverse, saving time and being able to print designs with good results as they have already been tested.
2. **G-code generation:** In this phase, the design is transformed into g-code instructions, a language understood by the CNC (a 3D printer in this case) machine which contains all of the instructions that must be completed by the printer in order to physically have the final object. All of the 3D printers share a set of settings full of options to control every single part of the print. However, every printer is different due to shape, axes length, printing material and many other reasons that make it impossible to have a unique configuration for all of them, which is why software solutions with configurable settings are required in this phase.
3. **Design printing:** With the G-code file generated, the last step is loading it to the program that takes control of the printer used and 'talks' with it sending instructions to the controller board and processing the received data so as to construct the design layer by layer, doing the perimeters first and the rest of the layer next.
As happens in many communication protocols, it is required to get an answer for every instruction in order to verify that it was received and completed correctly, though there may be cases where an instruction must be resent due to an error.

Programming methodology used

When developing software, several accepted standard methods can be applied depending on the type of project, objectives, specifications and the experience of the developers and project manager, the concrete method to be used will be different. Possible methodologies are:

1. **Waterfall:** Project is sequenced in different phases, paying special attention to planning, schedule, target dates, etc. These kinds of projects are supervised during all of their lifecycle, being a good choice for low experienced project teams or projects whose members vary and where the project progress can be measured. However, it entails lots of documentation and difficulties to make necessary changes usually detected during testing, so requirements must be well detected and defined since the project start.
2. **Prototyping:** Used in iterative development, where a first research development process leads to an evolving system design, coding and system requirements. On the one hand, this methodology allows a great flexibility along with user interaction, allowing a separation of a global task into small ones to be tested by the user whose feedback can help the developer in completing the code. On the other hand, cost estimation is not reliable as requirements suffer continuous changes (some of them pretty significantly as the final target is not clear), the design can be so poor that it would not lead to a solid, easy to maintain system, being also possible to produce a false feeling of a finished application to the end user.
3. **Incremental:** This is a combination of linear and iterative methods with the aim of reducing a single project into small ones in order to reduce the risks, so the development process can be controlled and flexibility is given to apply the necessary changes after tests are done, making sure that a good requirement analysis has been done.
4. **Spiral:** Also a combination of linear and iterative methods that makes possible reviewing risks and weighting considerations of the continuation of the project during its lifecycle. In contrast with other methodologies, it gives priority to risk avoidance, reliability and good implementation over functionality or resource consumption.
5. **Rapid Application Development:** An iterative methodology focused on a fast development without forgetting to achieve a high quality system at a relative low cost, giving more importance to the business needs than to engineering excellence. In order to be a successful method, the user must be involved during the process and documentation generation is required for future maintenance, being also important to adjust the delivery times.

From all of the possible methodologies, the one that has been followed in this project was prototyping. With the risks of a poor design and continuous requirement changes, it was the one that suits better as the feedback provided not just by the developer himself, but also from the tutor and other community members helped defining the changes to be done in the requirements as well as which parts had errors to be solved or could be improved for a better user focused designed.

Internal server structure

OctoPrint server follows the same structure as many other web servers, which is separated in two main parts: the frontend and the backend. The backend, developed in Python in this case, is the part of the server that is not visible to the user but performs several background tasks such as slicing a file, adding a user with specific roles or analysing a g-code uploaded by the user to be printed at any moment. In contrast, the frontend part, developed in JavaScript, jinja2, jQuery, CSS and LESS languages, is the one that is visible to the user and takes part of managing the responses sent from the backend as well as showing information or handling the different user inputs and petitions before sending the necessary data to the backend.

As can be seen, both parts are coded in different languages, so the data sent between them (using JSON in this case) must be understood by the languages involved. For that purpose, there are some functions for JavaScript and Python that allow transforming native data types into JSON (see in ‘Languages used’). As of Python, there is a framework called Flask that is in charge of accepting the requests received from the different clients and sending a response according to the result of the backend process. In order to send data to the frontend, a function called jsonify takes charge of internally adapting Python data (i.e. a dictionary or a list) into JSON data, whereas for getting the data received a function named get_json will obtain the JSON data sent in the request. For the case of JavaScript, there is an included function that transforms from JavaScript to JSON (stringify) and vice-versa (parse).

In order to communicate both entities, a library called sockjs-client is used. Its function is creating an object (concretely a web socket) to establish a connection between frontend and backend, managing the data to be converted to JSON in case of communicating from the frontend to the backend and parsing it otherwise.

Languages used

G-code

Languages used

Originally developed at MIT during the 1950s, it currently is the most used language for CNC machines, including 3D printers. The reason for this name is because most of the instructions start with the letter G. Commands can be used for several purposes, which can vary from setting temperatures to movements of one or more axes. If specified, it is possible to use a checksum to verify if an instruction was correctly sent, asking for a resend if necessary.

Instructions contain legible ASCII characters that follow the structure G/M/T + Code number + parameters (if any) with its values. The use of G, M or T will depend on the type of instruction, classified as follows:

1. G commands: Used for printer movements or settings related with that purpose (i.e. set position as 0 for a concrete axis or get to some point in space).
2. M commands: Settings not related with movements (e.g. powering a fan or managing an SD card) make use of this type of instructions.
3. T commands: Instructions starting with T are used to change the extruder used by the printer in case more than one is available.

Python

Python is a high level, object oriented programming language that easily permits file or user interaction and presents the following characteristics:

1. Indentation is required: Most of the programming languages known by the author of this book do recommend it to be used as it eases reading and understanding, but, if not done, will equally compile and/or execute. In contrast, Python will show an error to the user and code will not be executed until this error is solved.
2. No data type needs to be defined for variables. Python has methods to determine which data type a variable is. If, for some reason, the developer needs to verify it, calling the `type()` function will be enough to treat it as required.
3. ‘;’, ‘{’, and ‘}’ characters are no longer needed in if, for or while clauses. Instead, those sentences end with a ‘:’ character and the content of the clause is indented.

As a result, the code can be much cleaner and easier to be read by another person, also occupying less memory space.

Jinja2

According to documentation, *Jinja2 is a modern and designer-friendly templating language for Python, modelled after Django’s templates. It is fast, widely used and secure with the*

optional sandboxed template execution environment. The syntax used in this language can easily generate HTML and other markup formats commonly used for webpages, doing it dynamically according to the variables and expressions integrated within the templates. This way, webpages can avoid the problem of HTML and other similar languages which are based in static code so, when a more dynamic behaviour is needed according to the user input, jinja2, JavaScript and others must be used.

JavaScript

Designed by Netscape Communications Corporation and Mozilla Foundation in 1995, it is also an object oriented language, based on prototypes. Most of its code is developed on the client side as part of the web browser to permit a user interface and dynamic webpages, same as jinja2.

This language presents similarities with Python for object or data types definition, where the difference is that the keyword 'var' must be written preceding the object or variable name.

CSS and LESS

CSS is the acronym of Cascading Style Sheets whereas LESS is the acronym of Language for End System Services, an extension of the CSS language. Both languages are used for style design that will help to get a better user interaction in a pretty easy way. During the development of this project, no additional code had to be developed as all of the style settings required already existed, so the job during project development was finding the classes that must be used when developing the template.

JSON

Acronym of JavaScript Object Notation, it is an independent language (follows a text-only format) used to exchange data between the frontend and the backend so as to perform some tasks or to analyse data sent (i.e. to decide if the user must be notified of an error).

The syntax of this language is pretty simple and has the following characteristics:

1. Starts with the character '{' and ends with the character '}'
2. Data sent follows name and value pairs with the structure "name" : value, where name must be inside double quotes and values can be:
 - a. Numbers (without double quotes).
 - b. Strings (inside double quotes).
 - c. Boolean: true or false, without double quotes.
 - d. Array, whose content must be inside square brackets.
 - e. Object, whose content must be inside curly braces.

Git

Created by Linus Torvalds, also creator of the Linux Operating System, it is a tool similar to Subversion, Mercurial and many other VCS. Git allows creating different versions of a project, being able to track changes, create branches and where every person involved can work separately to finally merge all changes into a single, final project release. Thanks to the features provided, it is much easier to manage and maintain the projects hosted using git independently of its size.

In order to ease the understanding of some actions done during project development as well as some of the git characteristics, the following concepts are introduced:

1. **Repository:** A repository is where a project and its content are hosted. It can be created in the local part (in the PC or device where the user usually works with the files) or the remote part (GitHub in the case of this project).
2. **Index:** This is the part that tracks the changes to be done in the working tree during the next commit.
3. **Working tree:** A directory in the user's filesystem with a repository associated which also has associated its files and sub-directories.
4. **Commit:** A commit is a 'capture' of the working tree at a certain moment in time, so it contains information about the changes that have been done with respect to its previous commit. Every commit but the first always has a predecessor (parent commit), which is the reason why the concept of commit history exists. Besides, every commit has a different SHA1, used as a unique identifier.
5. **Fork:** A forked repository is a clone from an existing one into the cloner's. Unless authorization is granted to that user, every project has a restricted access so that only the owner of the repository can make changes or, in the case of a fork, to the user who forked the project.
6. **Branch:** It is usual that every project has different parts or features to be done so as to achieve a working application or version release. For efficiency purposes and separation of a global task into small parts, a branch can be (and usually is) created for every feature to be developed. Every repository has at least one branch, usually called master.
7. **Origin:** The URL where the repository is remotely hosted. This URL should point to the repository whose code is being developed by the user.
8. **Upstream:** The URL where the original repository is hosted. If that repository is not a fork, the URL will be equal to the one set in origin.
9. **Tree:** Objects that are part of a commit, also held by other commits and/or trees. An example of a tree is a directory that can contain sub-directories (another tree or trees) or files (blobs).
10. **Blob:** Objects that are part of tree objects. Every file is a blob that belongs to a tree.

11. **Head:** This is what defines what is selected or, better said, checked out. For example, supposing that there are three commits named A, B and C, by default the head will point to the last commit, in this case C. However, if commit B is checked out, head will point to that commit. It is very important to know where the head is when doing any operation (rebasing, merging, pushing ...) to avoid any possible error.
12. **Push:** An update of the remote repository by loading changes from local repository.
13. **Pull:** An update of the local repository getting the information required from the remote repository.
14. **Pull request:** When working in a team or, as in the case of this project, the user of a forked repository does not have a granted access to the original repository, a pull request is done, which is the equivalence of saying something like 'Please integrate my code into your code'.
15. **Merge:** Merges the content of two branches, which can belong the same repository or not. For example, imagine that a feature developed in a different branch is finished and the user wants to integrate it in the master branch or that the user is working in a forked repository and a concrete branch has been updated with more commits, so the user would like to apply that changes in his or her repository. For both cases, a merge is possible.
16. **Rebase:** This is a tool used to change commit history, which can be done in several ways. For example, it can be used when the original repository has suffered changes and the user is willing to update those changes in the forked repository. For example, imagine that a repository has branches master and devel, where master has a total of 2 commits (A and B) and devel has a total of 3 commits (F, G and H), with F pointing to A. Now, imagine that updates were made in the original repository and master branch has now two more commits (C and D), so the user wants to update its repository and make F commit in devel branch point to D from master branch. To do that, after doing a merge to apply that updates, a rebase would be done to make F point to D. With the rebase done, commits on devel branch will have its hash or identifier changed.

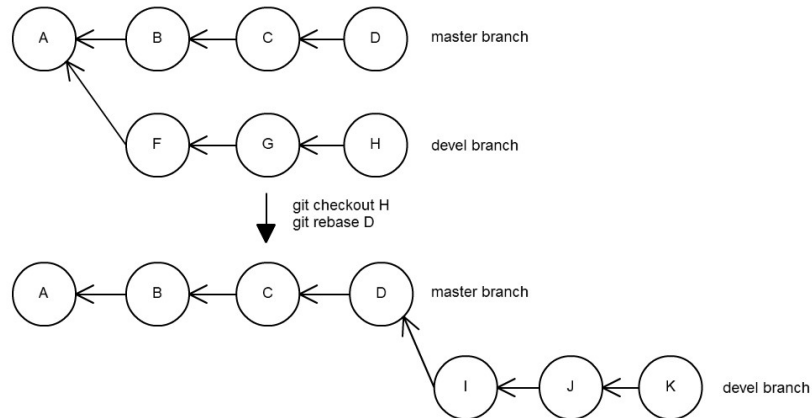


Figure 6: Git rebase example using default mode (commit picking)

Rebase not only does what was explained in the previous example (pick commits), it also lets the user change commit history by amending commits (that is, to change the message of a commit and/or the branch it belongs to), deleting commits or squashing them (that is, merge several commits into a single one), quite useful when making a pull request. Note that commits affected by rebasing will modify its identifier (hash).

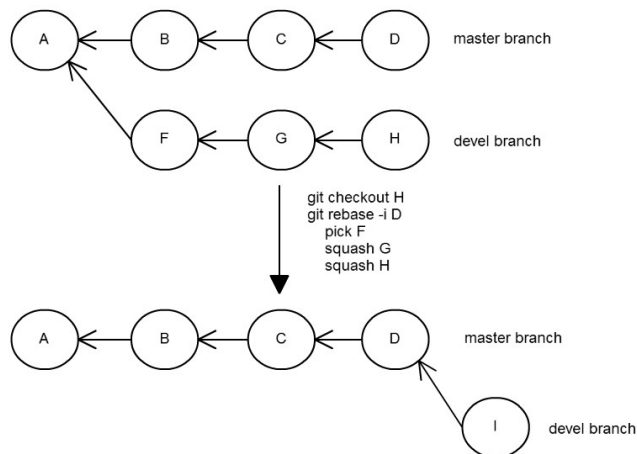


Figure 7: Git rebase example using interactive mode

17. **Ignore:** There might be some directories and type of files that may be user dependent (for example, *.pyc files) or that just are not wanted to be added to the index, so they can be added to the .gitignore file using the command `git ignore what_to_ignore`.

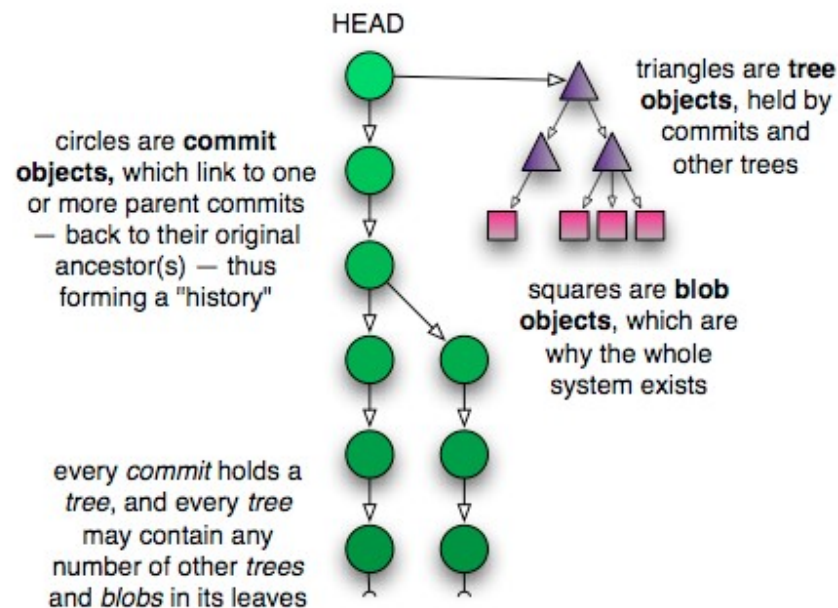


Figure 8: Graphical explanation of some of the contents.

Image taken from <https://jwiegley.github.io/git-from-the-bottom-up/1-Repository/5-the-beauty-of-commits.html>

Git characteristics:

1. Original repository and forked (cloned into another repository) ones are not synchronised: That is, when the original project is forked, the content of both will be the same but, after the first change in any of the repositories, the status and content of both will not be synchronised any longer, though the forked project can be easily synchronised with its parent by using a few commands (i.e. `git fetch upstream/name_of_repository`).
2. Every repository is divided in two, which are local and remote repositories, working both as individual entities, so changes applied in the local repository are not automatically updated in the remote repository and vice-versa. As happens between original and forked repositories, synchronization can be easily done by using a few commands.
3. Changes done to a repository can be reverted if necessary until thirty days after the change was done using `reflog`. This allows to solve possible problems occurred during a commit or a rebase for example.
4. Every blob, tree and commit is tracked. In case that any of them is lost, it is possible to search for it and recover.
5. Every file and directory has a unique hash that will remain the same even if that file or directory is removed and then created again with the same name.

6. If a content merge is attempted and conflicts which cannot be automatically solved are detected (e.g., somebody else edited the same lines of code), git will point out conflicts so they can be solved manually.

GitHub

GitHub is a secure and reliable web platform that uses git as Version Control System (VCS) which offers numerous services for different types of users. The main usage of the platform consists in storing software projects developed by one person or a team, although it does not have to necessarily be a software project to be hosted in this platform. Thanks to the possibility of storing projects free of charge (private repositories and business solutions are also available for a small fee) and all of the possibilities provided, this platform usually is the main choice for users and even companies willing to share code and ideas with the community. Besides, the owner of a repository can choose among several licenses (open source ones like Creative Commons or GNU General Public License).

The tools provided by GitHub platform for every repository to be remarked are:

1. **Code:** This is the main part of the repository, where its owner and authorised users can do any of the following:
 - a. Explore branches, commits, files and directories: For the last two it is also possible to see a text which is the message set for the last commit that has been done, so any user can know the files and directories affected by a commit.
 - b. New users can obtain information from README.md file content (shown to the user at the bottom part of the code tab) to set up the software and start using it. An alternative for this file that helps creating good documentation is the wiki.
 - c. Create a pull request: The owner of a forked repository can create a pull request so as to integrate his or her code inside the original author's. Before confirming it, it is possible to select the base fork (repository where the code would be added), the base branch (in which branch of the base fork would be integrated), the head fork (the user's repository) and the compare branch (the branch of the user that contains the code to be integrated). With the selection done, the developer can compare the changes to be applied before confirming the pull request.

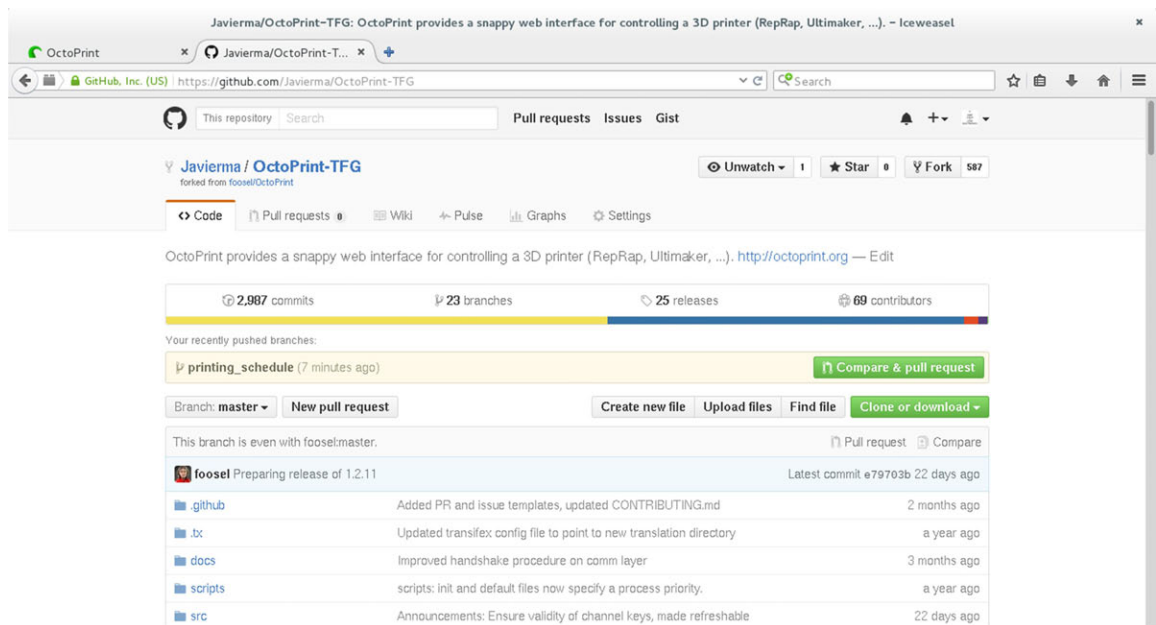


Figure 9: Code tab of the project repository

- d. Clone or download: Clone a repository (copy the repository to a PC using git clone) or download it as a zip file.
- e. Compare: Used also when a pull request is selected, this tool compares files with the same name located in different branches. Changes done are highlighted to the user and the parts of the code that did not suffer any change are 'hidden' to him or her, useful when comparing large files.
- f. Pull request: This tool is accessed the same way as when the user is in the code tab and clicks on 'New pull request'. Observing the following image, it can be seen that it includes the conversation or messages between both the owner of the target repository and the owner of the repository where the request is being done from, the number of commits affected by the pull request and the files changed, where it can be easily seen the modifications that have been done, as explained in the previous point.

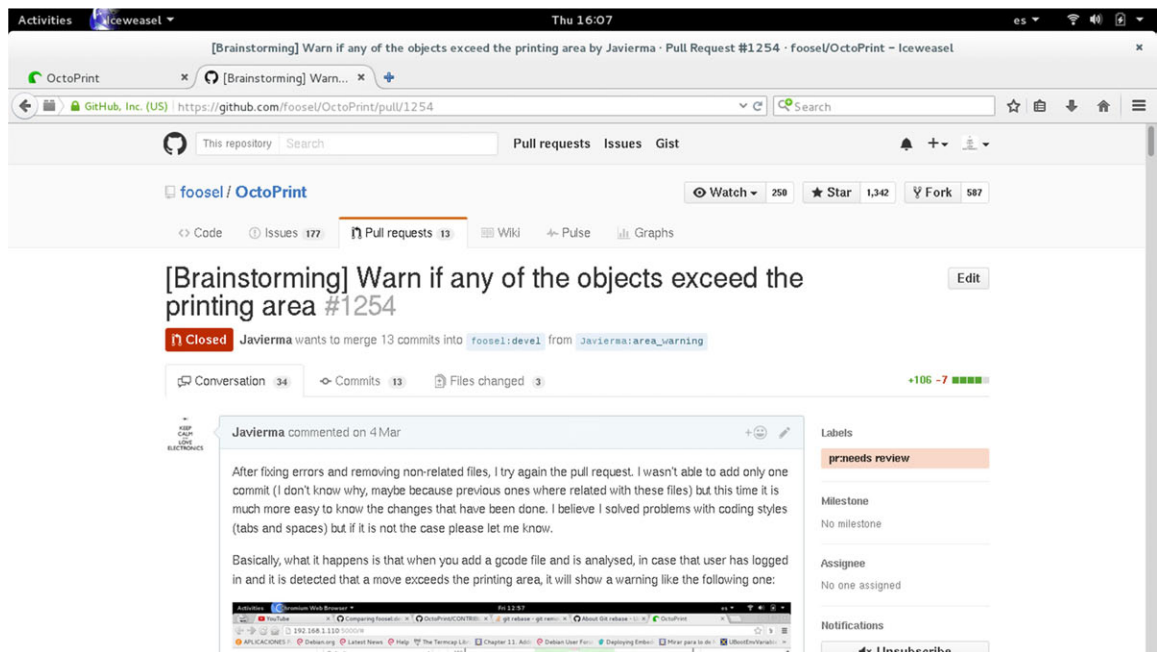


Figure 10: Example of a pull request

2. **Graphs:** The statistics of the repository, separated by, among others:
 - a. **Contributors:** Shows the contributions of the different users who forked the repository by the number of commits done and the total amount of additions and deletions.
 - b. **Traffic:** The number of visitors and also the number of users that cloned the repository, separating results by the total amount and by unique users. This statistic does not take into account the number of persons who selected 'Download as zip' option, though a script can be created to count it.
 - c. **Network:** A visual tool to observe the different branches of the users, including its commits and where are the different branches in comparison with the ones in the original repository.

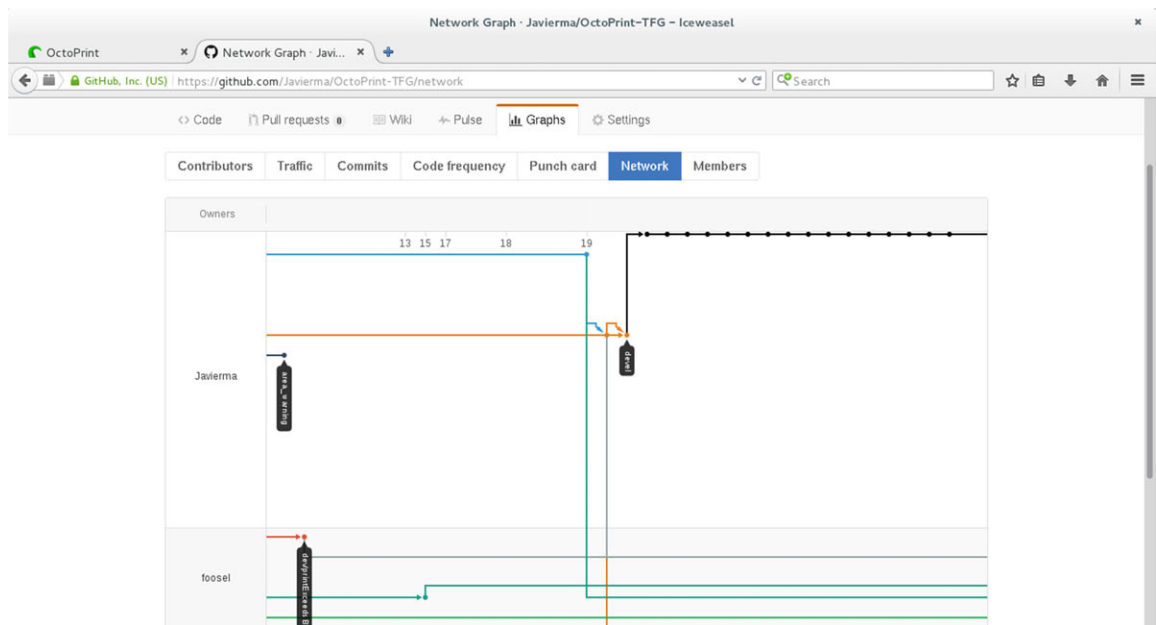


Figure 11: Example network tab of a repository

3. **Settings:** This is where the owner of the repository can change the global configuration of the repository. Some settings that can be changed are:
 - a. Repository name: The name of the repository.
 - b. Collaborators of the repository: The authorised users that can make changes to the repository.
 - c. Default and protected branches: Every repository has at least one branch, usually called master. In addition, the owner of the repository can decide if a branch must be protected against pull requests so, if other user tries it by selecting a protected branch in the based fork, the content to be added to the pull request will be forced to pass some tests set by the owner of the target repository.

Working environment

The project has been developed under a Raspberry Pi 2 with Raspbian Jessie Lite OS installed, a Linux distribution based on Debian Jessie adapted for the Raspberry Pi and its armhf architecture. In order to use this small computer, a SSH connection from a PC with Debian Jessie installed was used to remotely control it, navigating through its filesystem or using the terminal for example, being able to work as if the user was directly using the Raspberry Pi.

Depending on the availability of a router, the following two types of connections were done:

Working environment

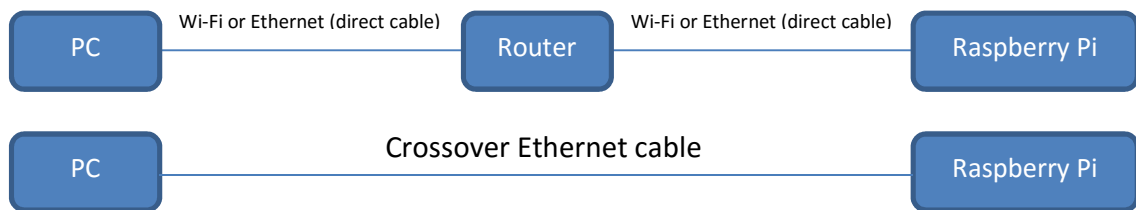


Figure 12: Possible equipment connections

Most of the times that the project was being developed, a router was available to connect with the Raspberry Pi, so it was possible to establish a connection through Wi-Fi or a direct Ethernet cable plus the router. However, there were sometimes that it was not possible to have a router and, therefore, Internet access. For those cases, a static configuration was set for the Ethernet Card in both extremes of the connection to create a network that will still allow remote connections, enough to keep coding and testing the different features. About the Ethernet cable, as connection is established through two devices of the same type, it is in theory required to use a crossover Ethernet cable, though most of network cards are nowadays capable of making the necessary changes by software, avoiding the need of using the cable type mentioned.

Other software and protocols were studied and tested for remote control purposes. For example, NoMachine, Remote Desktop Viewer (integrated by default in the Linux distributions) and Team Viewer allow remote connections using graphical applications to the target OS but, due to security reasons, numerous ports and protocols are blocked by the university network managers or the ISP and cannot be modified, so it was not possible to change the required configuration to establish the remote connection. In the case of Team Viewer, there was no problem with ports as it uses port 80 (same as HTTP, used for web surfing), but with the architecture of Raspberry Pi (armhf).

In addition to the settings configured to connect with the Raspberry Pi, the code was developed using gedit, the default editor that detects the language used and helps the user coding by changing the colour of the keywords or functions called. For the test process, iceweasel (default web browser) was selected as it provides the same developer tools for debugging purposes as other web browsers though with a better user interface and resources consumption.

Features added to OctoPrint

Warn user in case an object to be printed exceeds the global printing area

To print a file, an STL file must be uploaded to slice and generate the g-code file or directly the generated g-code file instead. Once the file has been loaded, an analysis starts in the backend, returning the following results to the user:

1. Filament used: Contains length and volume data (in m and cm^3 respectively) of the filament that is estimated to be used to print the content of the g-code file.
2. Estimated print time: The time that is expected to pass to build the content of the g-code file used.

Requirements

After the study of the data sent once the g-code analysis has been done, the requirements to implement this functionality were:

1. Evaluation of X, Y and/or Z parameters (if present) must be done if an extrusion is detected during g-code analysis.
2. Minimum and maximum values of each axis must be substituted with the value of the parameter of the axis if it is lower than the minimum or higher than the maximum.
3. Results must be sent to the frontend using JSON.
4. Results must be compared with values from the printer profile in the frontend.
5. Print origin must be detected. If origin is centre, minimum and maximum values must be divided by two in order to give reliable results in the new conditions.
6. If any of the objects is expected to be totally or partially printed outside of the printing area, a warning message must be shown to the user informing about the error detected, which will be done at the moment that the user selects to load or print the file with the implicated object.

Development process

To accomplish the objectives, the following was applied:

The first step was to edit the file containing the code which analysed the g-code file before retrieving the results to the frontend. Located in `src/octoprint/util/gcodeInterpreter.py`, the containing code reads the g-code file line by line and detects which concrete code is used in every instruction. This way, it is possible to do different tasks such as increment the value of the amount of filament used, the total print time for estimation or any other value that can be used for calculating the final value of the results.

Features added to OctoPrint

For the purpose of this feature, it is required to verify if the instruction is a G0 (rapid linear move) or G1 (linear move) command, as in their parameters are the positions to be reached for each axis, sometimes including an extrusion or retraction. If any of those instructions is used, it is compulsory to include at least one parameter, being an example instruction:

G1 X69.802 Y69.119 E1.84318

Basically, the meaning of this command is 'move to 69.802 in X axis, 69.119 in Y axis and extrude 1.84318 mm of the material used to print'. As can be seen, the Z axis was not included in the command, so Z min and Z max stored values would remain unmodified. Depending on the movement, there may be instructions where one or more parameters related with the different axis are not included. Knowing these factors, the following questions must be applied if instruction is G0 or G1:

Does the instruction include extrusion or retraction? To be any of them, the instruction has to include the parameter E. If that is not the case, no further analysis will be done for that instruction.

Does the instruction include a movement in any of the axis apart from E? If the value of this parameter is higher than zero, then the movement includes an extrusion, being retraction otherwise. In case of extrusion, it will be analysed if any of the parameters X, Y or Z is present. Sometimes a retraction is followed by an extrusion in order to achieve a better print quality, so the parameter E is the only one included in the command. As in these cases there is no movement, no maximum or minimum value would be modified for any of the axis.

With the analysis finished, the minimum and maximum values of the points reached for axis X, Y and Z will be appended to the results to be sent to the frontend, converting them to JSON.

At the moment that the data is received from the backend, the next step would be to obtain the printer profile settings (width, depth and height), and compare with the results of the analysis. If any of the axes is exceeded and the file has been selected for printing, a message will be shown to the user, informing him or her that the file must be revised and the comparison between the values from the profile and from the analysis, so the user will have knowledge about where the error would be.

Tests done

In order to check the proper execution of the new feature, the following tests were done applying all possible combinations of print origins (centre and lower left) and bed shapes (rectangular and circular), having successful results:

1. Load of a g-code file which objects will be printed inside the printing area. As expected, no warning message appeared.

2. Load of several modified g-code files, where the value of one or more modified axis is changed to make them exceed the printing area. As expected, if the file has been analysed and the detection has occurred, a warning message is shown to the user informing about the error at the moment the file is selected for printing.

Results

With the tests done, the feature could be considered completed so a pull request was done in order to integrate it to the official repository. Considering the feedback from its owner and other community members, some changes were applied to the code before a final approval. Including some changes applied by the main developer of OctoPrint after the pull request was accepted, the final version by observing the user interface was:

1. Model size was included to the file data.

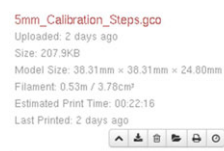


Figure 13: G-code analysis result

2. A notification warns the user about printing the selected file, giving information about the error found.



Figure 14: Example error message in case of exceeding an axis

Integration of Slic3r as a plugin for OctoPrint

As said in the second step of 3D printing, a g-code generator is required in order to convert a file with shape information (i.e. STL files) into instructions understood by the 3D printer according to custom printer settings (infill speed, bed temperature...). Slic3r is one of them and can be run using commands in a terminal or graphically in every known Operating System, translating AMF, OBJ and STL formats into g-code. From all those possible input files, STL is the one of interest as AMF and OBJ files are not admitted by OctoPrint.

For those users willing to install and execute Slic3r, it is possible to download the source code and build it or a compressed, already cross-compiled distribution ready to be executed in the target OS. Unfortunately, the armhf architecture is different from the target version

Features added to OctoPrint

applied in the cross-compilation for Linux systems, so Slic3r had to be downloaded and later compiled. As a consequence, the test process required a long time as it was not possible to easily change the version used without the need of compiling in every version change, usually taking at least twenty minutes or even close to one hour when doing the whole process for the first time, which included cloning the Slic3r repository and the download and installation of the required libraries.

The development of this plugin did not require starting from zero as there already was a repository (OctoPrint-Slic3r) that warned the user about using it due to unresolvable errors that for the moment were pending of being solved by someone else. In conclusion, the main task was to understand the existing code and find bugs to fix them so as to make the plugin distributable for Linux users.

Requirements

To achieve a distributable plugin, the following requirements must be met:

1. Slic3r versions 1.2.9 (most recent) and 1.1.7 (previous) stable versions for Linux distributions must work when using the plugin.
2. If an error occurs during the slicing process, the user must be notified, including in the message the error code. As it was already implemented by the owner of the original repository, no work was necessary to be done for this requirement.
3. The user must have the option to decide the name of the g-code file generated. As it was already implemented by the first user who started developing the plugin, no work was needed to be done for this requirement.
4. A user guide must be available to help installing and configuring the plugin under a Raspberry Pi 2, the target hardware of the project.
5. A shell script must be created to ease the installation of Slic3r in Raspbian, allowing the plugin to be more accessible by the community, including less skilled users.
6. The plugin must be able to be installed in any computer with Linux running OctoPrint. As there is already a plugin manager, its implementation could be avoided.

Development process

In order to meet the requirements and fix bugs, the following was done:

Profile settings

Slic3r separates its profile settings into three parts known as print, filament and printer, whose configuration values are stored in INI files. Nevertheless, it is possible to load a single file with all of the settings and their respective values so as to slice a file, which indeed is what the plugin does.

In addition to the Slic3r profile files, a default configuration, stored in a dictionary, exists in the code that is overridden during the profile import. Similar to maps in Java, a dictionary is a set of key and value pairs where the keys (a key is a setting in this case) have to be unique, avoiding any possible repetition. In comparison with the available settings dictionary used by the code of the plugin at development start, some changes were done to make them equal to the Slic3r settings, removing inexistent ones or existent ones (i.e. scale) that made no sense as they must depend on user input at the moment of slicing.

Profile import

Depending on the setting, the type of its value may vary among a percentage, an integer or a float, for example. Those settings were in a list so, when updating the value of any of those keys, it was searched in the list so as to evaluate the type of data. However, not all the settings required had been added, so the ones left were added during development. Besides, the list indicated that the values for the keys added could be a float or a percentage when in reality could also be an integer. With that change, it was required to modify the detection of the type of value as it was supposed that, if the value was not a percentage, it had to be a float then.

Slic3r installation

As the cross-compiled executable file was not runnable under the armhf architecture, a shell script was developed in order to ease the installation of Slic3r, especially for the non-skilled users that may face problems to download and set up the software. A shell script is a set of instructions (usually one per line) to be run using the terminal or another command line interpreter and has the capacity of ‘interacting’ with the user, changing its behaviour depending on the user input. In this case, the script takes charge of downloading the necessary libraries and packages so as to set up the version specified by the user. To sum up, it achieves a faster and reliable Slic3r installation.

Tests done

To verify that the plugin worked and could be considered installable for Raspbian and other Linux distributions, the following tests were done:

1. A script was run to compare the Slic3r settings with the ones available in a profile file created by the plugin. The result was that all settings were correctly imported for the versions specified in requirements.
2. A comparison between the g-code files generated directly by Slic3r and by plugin usage was done, also with the help of a script that accelerated the process, avoiding possible errors that could be committed by the human eye. Depending on the file, the result varied among the following ones:

Features added to OctoPrint

- a. Depending on directly using Slic3r graphically or through the console, the result g-code for the same file was different, being the difference 0.1 mm of filament length.
 - b. Most of the g-code files generated using the plugin and directly using Slic3r were exactly the same, with the only difference of an informative string that indicated the date when the slice was done. For the g-code files that were not exact, the difference was in filament length by no more than 2mm.
 - c. Every STL file with no errors was correctly sliced.
3. An installation was done through another Raspberry Pi running OctoPrint where the plugin plus Slic3r (using the created shell script) had to be installed. The process was successfully completed.

In conclusion, all the tests done indicated that the plugin was ready to be installed under computers running OctoPrint in Linux.

Results

With the plugin finished, a pull request was done in order to add OctoPrint-Slic3r as a plugin for all users running OctoPrint in Linux. At the time of writing this document, the pull request was still open, pending of some changes before officially adding it to the plugin repository, though users are still able to install from the repository where the pull request was done. The changes to be done were:

1. Improvement of the README file so as to achieve a cleaner user guide, warning the user about using the plugin at the moment in other OS than Linux distributions (excepting Slic3r version 1.2.9 in Windows).

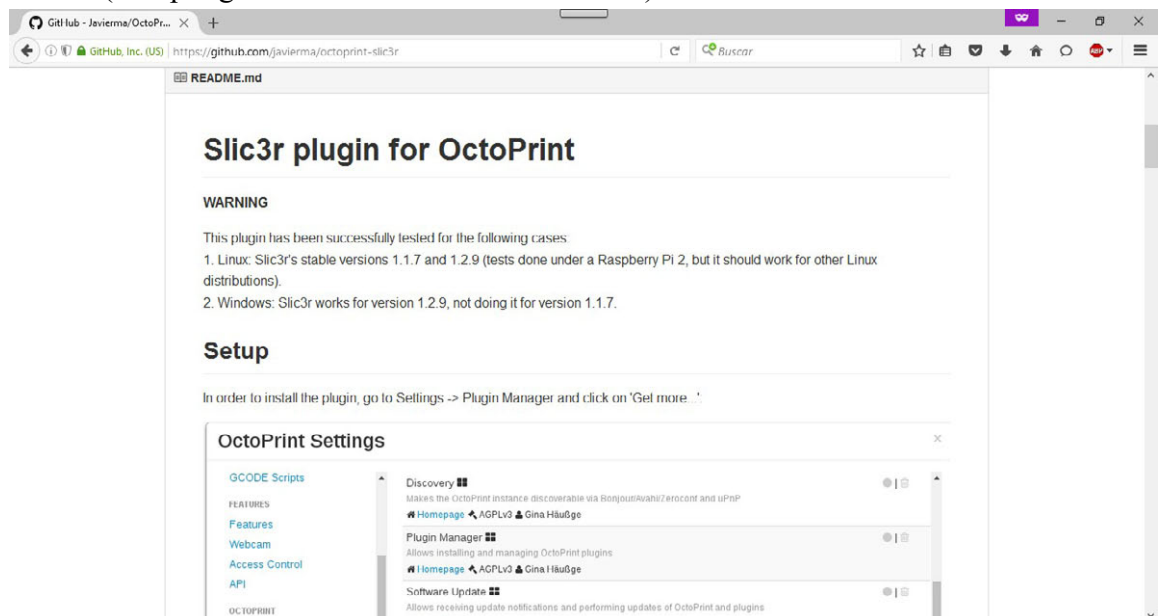
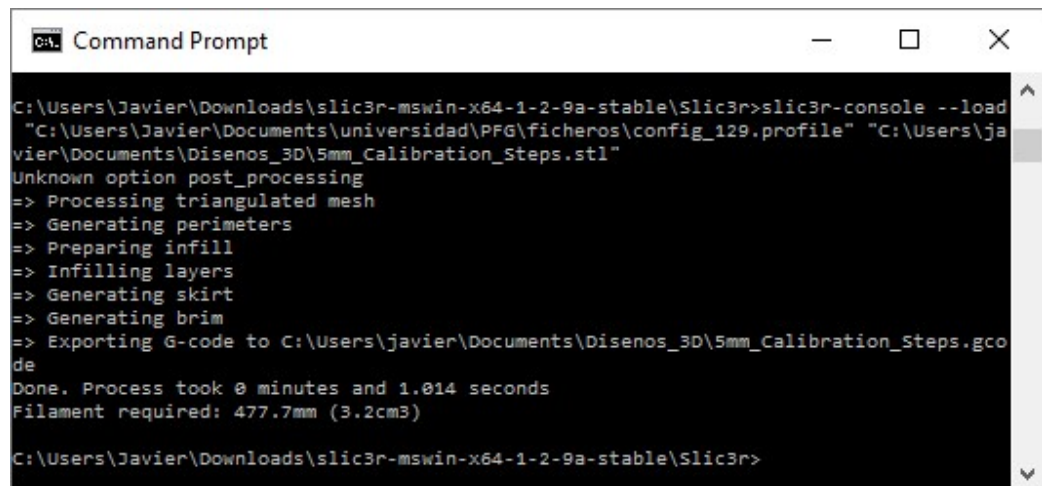


Figure 15: README file of OctoPrint-Slic3r repository

2. Although it was not a requirement for this project, a test was done using Slic3r under Windows as, according to the owner of the original repository, it was the OS that provokes most of the errors experienced. The result was successful for Slic3r version 1.2.9 but not for 1.1.7.



```
C:\Users\Javier\Downloads\slic3r-mswin-x64-1-2-9a-stable\Slic3r>slic3r-console --load
"C:\Users\Javier\Documents\universidad\PF6\ficheros\config_129.profile" "C:\Users\ja
vier\Documents\Disenos_3D\5mm_Calibration_Steps.stl"
Unknown option post_processing
=> Processing triangulated mesh
=> Generating perimeters
=> Preparing infill
=> Infilling layers
=> Generating skirt
=> Generating brim
=> Exporting G-code to C:\Users\javier\Documents\Disenos_3D\5mm_Calibration_Steps.gco
de
Done. Process took 0 minutes and 1.014 seconds
Filament required: 477.7mm (3.2cm³)

C:\Users\Javier\Downloads\slic3r-mswin-x64-1-2-9a-stable\Slic3r>
```

Figure 16: STL slice in Windows using slicer-console

Printing schedule

Every program known by the writer allows the user managing the printer and, among other options, start a print right at the moment that the user clicks it. However, none of them allow scheduling a print task so as to automatically start printing when the user wants to, being possible to program a single or a repetitive task. For that reason, this feature has been developed for the cases were OctoPrint runs under Raspbian.

Before explaining the requirements, development process, tests and results, several things are explained as an introduction, helping to achieve a better understanding of this feature and how it works.

Crontab, cron and Wget

Crontab is a file available in all Linux systems that permits the user to program one or more jobs in a pretty flexible way that will be executed at least once a year, though other options can be set. For jobs to be done only once, at is the solution as the entries are removed after being executed. In contrast with crontab, if a job was not done due to the fact that the Operating System was not running, it will be executed right after the next boot. Depending on the OS where at is running, jobs may be done in concrete times of the day (i.e. at 12:05 instead of 12:03) as entries might not be read every minute, therefore being less flexible than crontab.

Every user of the system has its own crontab file, and another one exists for the system itself. An example of a crontab file can be seen in the following figure:

Features added to OctoPrint

A screenshot of a terminal window titled 'javier@Javier-debian: ~'. The terminal shows the command 'cat /etc/crontab' being executed, displaying the contents of the crontab file. The output includes comments about the file's purpose, environment variables like SHELL and PATH, and several cron entries for hourly, daily, weekly, and monthly tasks, all running as root. The terminal ends with a prompt 'javier@Javier-debian:~\$' and a cursor.

```
javier@Javier-debian:/etc/network$ cd $HOME
javier@Javier-debian:~$ cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6 * * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
javier@Javier-debian:~$
```

Figure 17: Example crontab file

Doing a further analysis of the content, it can be seen that the crontab file is separated in cron jobs or entries (one per line), each one containing the following parameters, separated by the space character:

1. Minute: If value is a number, possible range goes from 0 to 59.
2. Hour: Possible values from 0 to 23 both included.
3. Day of month: Possible values from 1 to 31.
4. Month: From 1 to 12 or the first three letters of the month instead.
5. Day of week: From 0 (Sunday, which can also be 7) to 6 (Saturday) or the first three letters of the day of the week instead.
6. User: If modifying the system crontab, the user that will run the entry command.
7. Command: The command to be executed. If required, several commands can be written separating them by using ';' at the end of each command.

Date and time data can be numbers, wildcards (*) or divisors (i.e. */5), providing cron jobs to be very flexible. Besides, it is possible to substitute the date and time by text starting with the '@' character that would indicate the repetition of the job (i.e. @reboot or @daily).

Cron is a system service starting at boot that checks every minute if a crontab entry must be executed. After every check, the process sleeps until the next minute, when crontab entries are read again.

Wget is a web client that retrieves the file for the URL set as a parameter when executing the tool. As any other web client, it supports the same methods and protocols, being able to access any website with the particularity that it is executed using the terminal.

System date

There is an element in (almost) every computer that is used to obtain the date and time called RTC, which helps achieving a reliable system clock aided with the configuration of an NTP Server (a server available on the Internet that helps to synchronise system date and time). Nevertheless, the Raspberry Pi does not come with a RTC so as to achieve a less expensive and smaller computer. For that reason, it was needed to set up the system date in order to make this feature (printing schedule) work as it should. For more information about how was it done, you can consult the appendix.

Requirements

In order to make this feature distributable, the following requirements must be met:

1. The 'schedule' button must be added to the file options if it contains g-code instructions.
2. The schedule option must be enabled only if access control is enabled and the user is logged in. If access control is disabled, the option must be always enabled.
3. File to be scheduled and its estimated print time must be sent as parameters when the user opens the menu to schedule a job. For reliability purposes, the time sent must be:
 - a. The last print time if successful.
 - b. If the previous is not available, the estimated print time stored in the g-code analysis.
4. An option to modify or delete a print must be visible when at least one print of the same file has been programmed.
5. With the 'modify' option selected, a list with all already scheduled jobs for the file selected must appear, making clear to the user when are the jobs going to be done.
6. Date and time fields must be filled with current date and time data.
7. 'Save' button must be enabled in the following cases:
 - a. A job is selected for deletion.
 - b. In case of a repeated print, only one of the three possible repeating options (daily, weekly and monthly) is selected.
 - c. If the print to be scheduled is single, the date (day, month, year, hour and minute) must be a future and valid one.
 - d. None of the following conflicts can take place, independently of the kind of print to be scheduled and the type of the one already scheduled:
 - i. A print to be schedule starts during an already scheduled print.
 - ii. A print to be schedule ends during an already scheduled print.
 - iii. An already scheduled print starts during the print to be scheduled.
 - iv. An already scheduled print ends during the print to be scheduled.
 - v. A print to be scheduled starts before an already scheduled one and ends after that scheduled print.

Features added to OctoPrint

- vi. A print to be scheduled starts after an already scheduled one and ends before that scheduled print.

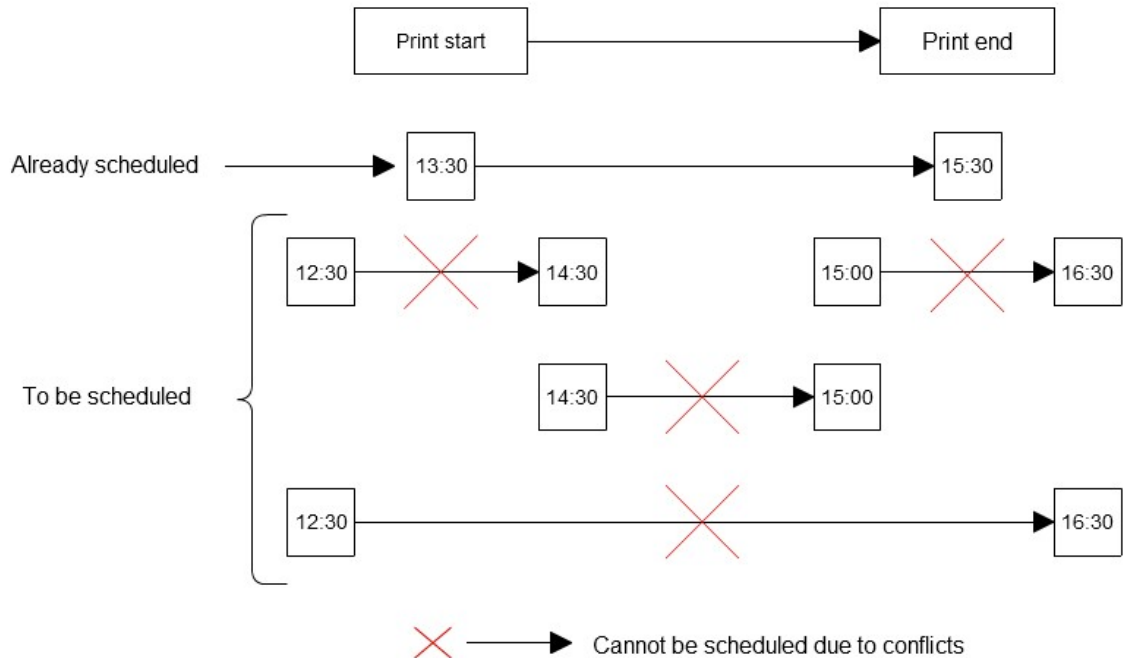


Figure 18: Example of conflict cases

- e. In case a conflict is found, the name of the file that conflicts with the one to be scheduled must be shown to the user.
- f. When the user clicks the 'schedule' button, the following information must be retrieved from the backend for every scheduled job:
 - i. Job name: The file to be printed
 - ii. Job start date: When a print is going to be done. In the case of a repeated print, the date must be adapted.
 - iii. Estimated print time: As in the case of the file selected, the most reliable time will be selected.
- g. When a print starts, check if it is a repetitive job, removing it if it is not.
- h. Do not delete single jobs if OctoPrint was not running when the job was going to be executed.
- i. OctoPrint must be running to start a print.
- j. A programmed print must not start if the printer is doing another job.
- k. If API key is not enabled, show the user an error indicating that it must be enabled in order to be able to schedule a job.

Development process

In order to meet the requirements, it was necessary to create extra files for both frontend and backend that took charge of user interaction plus a background process in charge of

managing the user crontab and, therefore, the prints to be done at the date set by the user. To complete the required process, the following things were done:

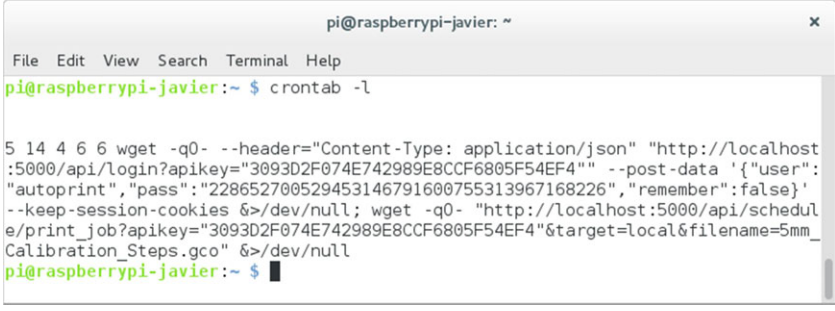
1. A button was added to the options for the g-code files.
2. If schedule option is selected, it is verified if an API key is enabled. If not, an error message is shown to the user.
3. A menu was created so the user can set when the file selected must be printed, having the following available options:
 - a. Modify job: This option will be visible in case that there are one or more files scheduled and invisible otherwise to avoid confusion. If visible, a list with information about each of the scheduled prints for the file appears, specifying the type of print, the date in case of a single print and the time for all of them.
 - b. Print after current job: A visible option if the machine is printing at the moment that the menu is opened.
 - c. Day, month and year, unless a repetition option is selected, where in that case:
 - i. If daily, the three fields are changed by '*'.
 - ii. If weekly, the three fields are changed by '*' and an option to select the day of the week appears.
 - iii. If monthly, month and year fields are changed by '*', so the user can decide the day of the month.
 - d. Time: The hour and minute when the print will start.
 - e. Conflicts: If the print to be scheduled would cause that an already scheduled print could not be completed, the name of the file that presents the conflict is shown to the user.
 - f. Print separation: The amount of minutes that must be the printer idle after finishing a previous print (if any) before starting the next one, being one minute the minimum separation time. This option makes sure that a print does not end in the same minute that other starts.
 - g. Repetition options: If the print must be done more than once, the user has the opportunity to print a file several times. By default, none of the options is selected.
 - h. 'Save' and 'Cancel' buttons: For the last thing, it is possible to cancel scheduling at all times, but 'Save' button will be enabled only if the specified requirements are met. For example, if the print to be done is single and the date and time selected are previous to the date when the scheduling is being done, or date selected is the 31st of February, it will not be possible to save settings as the print would not take place.
 - i. When the user selects saving a job, the following happens:

Features added to OctoPrint

- i. Before sending any data to the backend, it is verified if a user named 'autoprint' has been created. If not, a random password is generated and stored in a hidden file called '.autoprint_pass' (with read and write permissions only granted to the system user that is running OctoPrint) inside the hidden folder '.octoprint'. Finally, the new user is added to OctoPrint. In contrast to the rest of OctoPrint users, 'Update User' and 'Change password' options are disabled to the user under Settings -> FEATURES -> Access Control
- ii. If user exists or has been created because it did not, send the following data to the backend using JSON:
 1. The name of the file to be printed and its target (local or sd).
 2. If a single print, send the day of the month, month and the corresponding day of the week.
 3. If print is to be repeated daily, set '*' as value for day of the month, month and day of the week fields.
 4. If print is to be repeated weekly, set '*' as value for month and day of the month. The value for the day of the week will be the associated number of the day of the week selected through the menu (from 0 for Sunday to 6 for Saturday).
 5. If print is to be repeated monthly, set '*' as value for month and day of the week, whereas the day of the month will be the one selected by the user.
 6. Send the hour and minute when the print will start.
- iii. With the data received in the backend, the following data are retrieved from the configuration file:
 1. Port: The port where OctoPrint is executed (5000 by default).
 2. API key, required for directory and file access through the server.
 3. Password: The password that was generated for the user 'autoprint'.

From this point, all required data to setup a new cron entry has been obtained, so the next step is opening the Raspberry Pi current user (the system user executing OctoPrint) crontab, where the structure of the job command will be different depending on the access control settings:

1. With access control enabled:



```

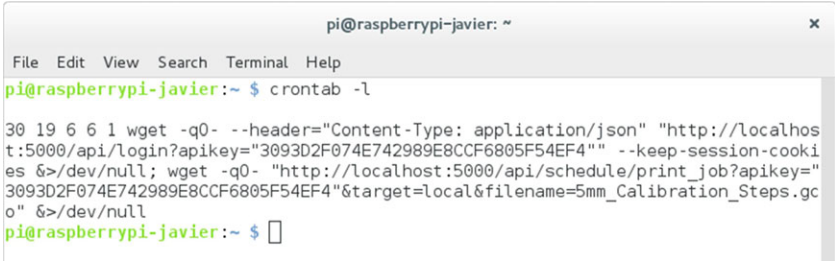
pi@raspberrypi-javier: ~
File Edit View Search Terminal Help
pi@raspberrypi-javier:~ $ crontab -l

5 14 4 6 6 wget -q0- --header="Content-Type: application/json" "http://localhost:5000/api/login?apikey="3093D2F074E742989E8CCF6805F54EF4"" --post-data '{"user": "autoprint", "pass": "228652700529453146791600755313967168226", "remember": false}' --keep-session-cookies &>/dev/null; wget -q0- "http://localhost:5000/api/schedule/print_job?apikey="3093D2F074E742989E8CCF6805F54EF4"&target=local&filename=5mm_Calibration_Steps.gco" &>/dev/null
pi@raspberrypi-javier:~ $

```

Figure 19: Example crontab entry with access control enabled

2. With access control disabled:



```

pi@raspberrypi-javier: ~
File Edit View Search Terminal Help
pi@raspberrypi-javier:~ $ crontab -l

30 19 6 6 1 wget -q0- --header="Content-Type: application/json" "http://localhost:5000/api/login?apikey="3093D2F074E742989E8CCF6805F54EF4"" --keep-session-cookies &>/dev/null; wget -q0- "http://localhost:5000/api/schedule/print_job?apikey="3093D2F074E742989E8CCF6805F54EF4"&target=local&filename=5mm_Calibration_Steps.gco" &>/dev/null
pi@raspberrypi-javier:~ $

```

Figure 20: Example crontab entry with access control disabled

As can be seen, the difference is sending the user and password through the POST method if access control is enabled. On the other hand, they have in common that the response and errors that may occur are sent to /dev/null, not showing any output to the user.

With the entry set, a response with an empty body is sent to the frontend, indicating that the process was successful.

At this point, a new cron entry has been set, and by the day and time specified in that entry, OctoPrint will be accessed to start a print if running. Supposing that condition is met, the printer status is checked so as to proceed differently depending on that status:

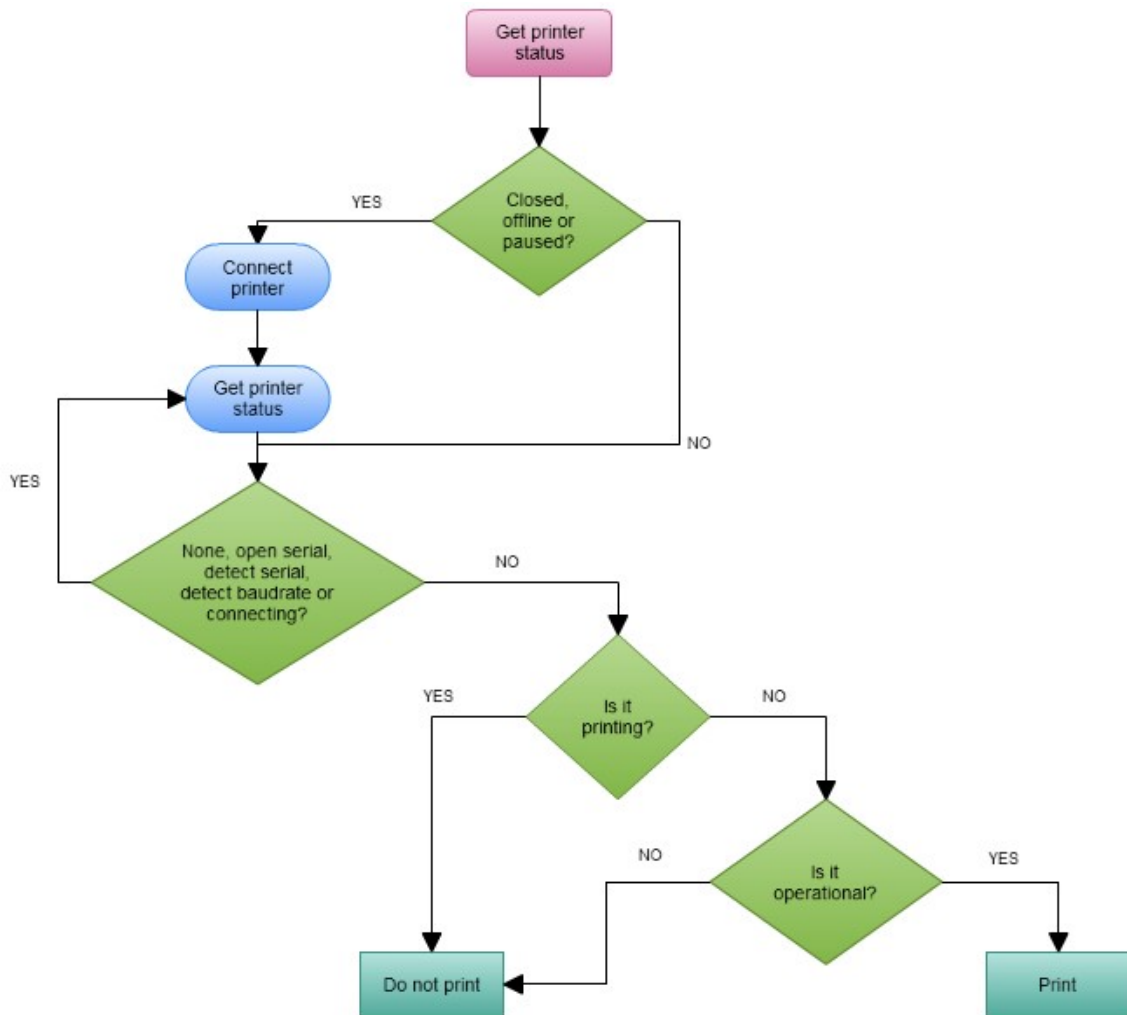


Figure 21: Printer status workflow

Depending on the status of the printer, if a print starts, it will be verified if the job was a single print so as to remove the crontab entry, whereas for repetitive jobs or single jobs not executed due to another print or the fact that OctoPrint is not running the entry will remain. If an entry must be removed, it will be verified that filename, date and time data are the same.

For the case where the user action is to modify or delete a job, what would be done is removing the file selected and adding a new entry if the user action is a modification. The data sent to the backend includes the name of the file plus date and time so as to verify that the entry to be removed clearly is the one that the user selected.

Tests done

With the purpose of verifying that the feature worked as expected, numerous thorough tests were done having satisfactory results:

1. With access control enabled, it was verified that the schedule option was enabled only when a user was logged in.
2. With access control disabled, it was verified that the schedule option was enabled.
3. 'Enable' option was unchecked at Settings -> FEATURES -> API verifying that, if the schedule option was selected, an error message was shown.
4. 'Enable' was checked again, verifying that a menu appeared.
5. It was verified that the option 'Print after current job' is visible only if a print is being done.
6. It was verified that the user could modify a job only if one or more prints of the same file were scheduled.
7. It was tested that, if there was no conflict, the value for conflicts field was 'No conflicts'.
8. It was tested that the name of the file that was already scheduled appeared if a conflict existed.
9. It was checked that the 'Save' button was enabled in the following cases and disabled otherwise:
 - a. A print had been selected for deletion
 - b. No print was selected for deletion and no conflicts were found.
 - c. Print to be scheduled was single and the date set was a valid one.
 - d. Print to be scheduled was repetitive and only one repeat option had been selected.
 - e. User had selected a deletion or a modification and a job had been selected.
 - f. With a print scheduled to be repeated weekly, a day of the week had been selected.
10. The 'Cancel' button was always enabled.
11. It was made sure that, when modifying or deleting a job by the user or by OctoPrint due to be a single job, the removed or modified entry was the expected one.

Results

Once the tests were finished, the applied result in the user interface was:

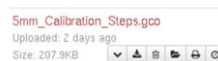


Figure 22: File options



Figure 23: API key notification error

Features added to OctoPrint

Schedule 5mm_Calibration_Steps

Schedule options

Date (dd / mm / yyyy) 18 / 6 / 2016

Time 14 : 24

Conflicts with No conflicts

Wait after previous job (if any) 1 min

Repeat job at the time set

☐ Every day

☐ Every week

☐ Every month

Cancel Save

Figure 24: Scheduler menu

OctoPrint Settings

PRINTER

- Serial Connection
- Printer Profiles
- Temperatures
- Terminal Filters
- GCODE Scripts

FEATURES

- Features
- Webcam & Timelapse
- Access Control**
- GCODE Visualizer
- API

OCTOPRINT

- Server

| Name | Active | Admin | Action |
|-----------|-------------------------------------|-------------------------------------|--------|
| aetel | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | |
| autoprint | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | |

+ Add user

About OctoPrint Cancel Save

Figure 25: OctoPrint settings window for access control

Warning

There is one or more scheduled jobs that will not success if this print is executed. Are you sure to continue?

Cancel Continue

Figure 26: Warning message to confirm or cancel print

With the feature tested and finished, an issue was opened in the OctoPrint repository commenting about the feature, wondering if it could be implemented as a feature as it only worked for Linux systems. Due to the fact that it was not possible to convert it to a plugin without extending the end date of the project, it was left in another branch of the repository so any user could make use of it by running OctoPrint from printing_schedule branch.

Conclusions and future work

With all the development process finished, including the response from the community, the following conclusions have been reached.

1. The developer has learned the basis of several new languages, being able to apply object oriented programming concepts.
2. The developer has learned git, a powerful VCS tool that allows collaborating in projects of any size and type.
3. Community users responded pretty positively to the new features, even giving some ideas that helped to improve the code developed and the user interface.
4. The writer of these pages could collaborate in a real project, adding some new ideas that will help having a better user experience in 3D printing.
5. Web server internal structure and communication was learnt, as server connections were done developing other software but not as deep as in the case of this project.

With all the conclusions, possible future work would be:

1. Enabling OctoPrint-Slic3r plugin to be used with OctoPrint installed under Mac OS.
2. Enabling the scheduler to be used under other OS than Linux, the target one used for the project development.
3. Modify OctoPrint so as to easily enable managing more than one printer. It can be done at the moment but requires knowledge that non-skilled users do not have.
4. Fix bugs that do not let some prints to be completed due to checksum errors, wasting time and material.
5. Translate OctoPrint into more languages.
6. Improve login security, especially when OctoPrint is accessed from outside of the network where the server is running.

References

References

- [1] Wiegley, John. *Git from the Bottom Up*, [consult: 5-06-2016]. Available at <https://jwiegley.github.io/git-from-the-bottom-up/>, public access.
- [2] Gina Häußge. *OctoPrint documentation* [online]. <<http://docs.octoprint.org/en/master/>>. [consult: 5-06-2016]
- [3] GitHub. *GitHub Help* [online] <<https://help.github.com/>>. [consult: 5-06-2016].
- [4] Git. *Git documentation* [online] <<https://git-scm.com/doc>>. [consult: 5-06-2016]
- [5] RepRap. *RepRap Wiki* [online] <<http://www.reprap.org/>>. [consult: 5-06-2016].
- [6] RepRap. *Proyecto Clone Wars* [online]. <http://www.reprap.org/wiki/Proyecto_Clone_Wars>. [consult: 5-06-2016]
- [7] RepRap. *G-code* [online]. <<http://reprap.org/wiki/G-code>>. [consult: 5-06-2016].
- [8] Pocoo. *Flask API* [online] <<http://flask.pocoo.org/docs/0.11/api/>>. [consult: 5-06-2016].
- [9] Pocoo. *Jinja2* [online]. <<http://jinja.pocoo.org/docs/dev/>>. [consult: 5-06-2016].
- [10] Python. *Python 2.7.11 documentation* [online]. <<https://docs.python.org/2/>>. [consult: 5-02-2016]
- [11] Knockout. *Knockout js* [online]. <<http://knockoutjs.com/>>. [consult: 5-06-2016].
- [12] Alessandro Ranellucci. *Slic3r* [online]. <<http://slic3r.org/>>. [consult: 5-06-2016].
- [13] GNU. *Bash reference manual* [online]. <<https://www.gnu.org/software/bash/manual/bashref.html#Shell-Scripts>>. [consult: 5-06-2016].
- [14] The Raspberry Pi foundation. *RASPI-CONFIG* [online]. <<https://www.raspberrypi.org/documentation/configuration/raspi-config.md>>. [consult: 5-06-2016].
- [15] Manpagez. *man crontab(5)* [online]. <<http://www.manpagez.com/man/5/crontab/>>. [consult: 5-06-2016].
- [16] GNU Operating System. *GNU Wget 1.17.1 Manual* [online]. <<https://www.gnu.org/software/wget/manual/wget.html>>. [consult: 5-06-2016].
- [17] The Pi Hut. *Adding a Real Time Clock to your Raspberry Pi* [online]. <<https://thepihut.com/blogs/raspberry-pi-tutorials/17209332-adding-a-real-time-clock-to-your-raspberry-pi>>. [consult: 5-06-2016].

- [18] W3 Schools. *JavaScript tutorial* [online]. <<http://www.w3schools.com/js/>>. [consult: 5-06-2016].
- [19] W3 Schools. *JSON tutorial* [online]. <<http://www.w3schools.com/json/>>. <<http://www.w3schools.com/js/>>. [consult: 5-06-2016].
- [20] Pilgrim, Mark. *Dive Into Python 3*, [consult: 5-06-2016]. Available at <<http://www.diveintopython3.net/>>, public access.
- [21] Department of health & human services - USA .Selecting a development approach, consult [5-06-2016]. Available at <<https://www.cms.gov/research-statistics-data-and-systems/cms-information-technology/xlc/downloads/selectingdevelopmentapproach.pdf>>, public access.
- [22] W3C. *Cascading Style Sheets home page* [online]. <<https://www.w3.org/Style/CSS/>>. [consult: 5-06-2016].
- [23] LESS. *LESS* [online]. <<http://lesscss.org/>>. [consult: 5-06-2016].
- [24] 3D printing industry. *The free beginner's guide* [online]. <<http://3dprintingindustry.com/3d-printing-basics-free-beginners-guide/history/>>. [consult: 5-06-2016].
- [25] Printron. *Pronterface* [online]. <<http://www.pronterface.com/>>. [consult: 5-06-2016].
- [26] Repetier. *Repetier software* [online]. <<https://www.repetier.com/>>. [consult: 5-06-2016].
- [27] OctoPrint. *OctoPrint* [online]. <<http://octoprint.org/>>. [consult: 5-06-2016].
- [28] Wikipedia. *History of Numerical Control* [online]. <https://en.wikipedia.org/wiki/History_of_numerical_control>. [consult: 5-06-2016].
- [29] Sockjs-client repository. *Sockjs-client* [online]. <<https://github.com/sockjs/sockjs-client>>. [consult: 5-06-2016].
- [30] GNU. *GNU Affero Public License v3* [online]. <<http://www.gnu.org/licenses/agpl-3.0.html>>. [consult: 5-06-2016].

Appendix

System date

As explained in the last implemented feature (printing schedule), the system date must be set in order to obtain the correct date and time. The steps required to set the time of Madrid (Spain) were:

1. With a terminal, the Raspberry Pi configuration was accessed by writing the command `sudo raspi-config` and selecting option 5 'Internationalization Options':

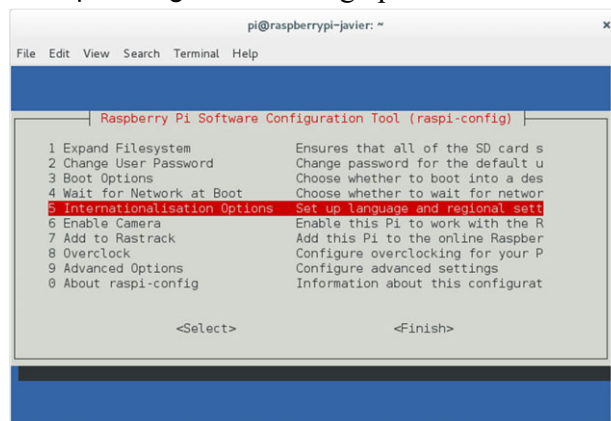


Figure 27: Main configuration menu of a Raspberry Pi

2. For the available options, the 'Change Timezone' option was selected:

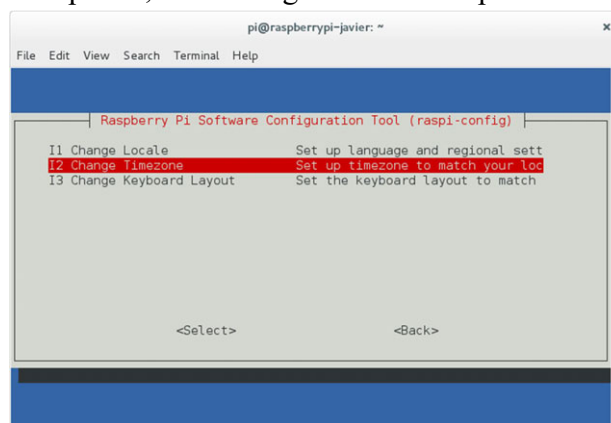


Figure 28: Internationalisation options menu

3. To set the Spanish time, 'Europe' and 'Madrid' were selected in the next menus:

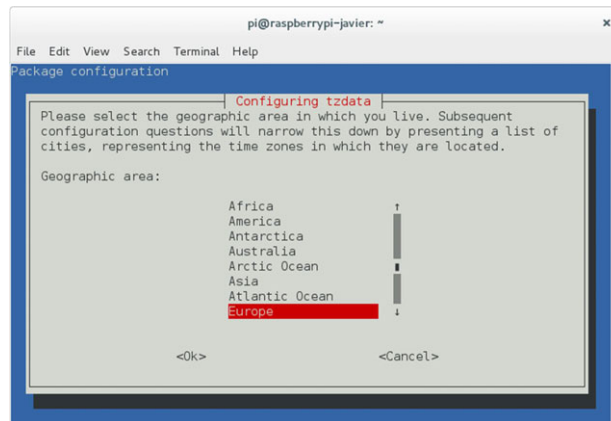


Figure 29: Continents list for geographic area

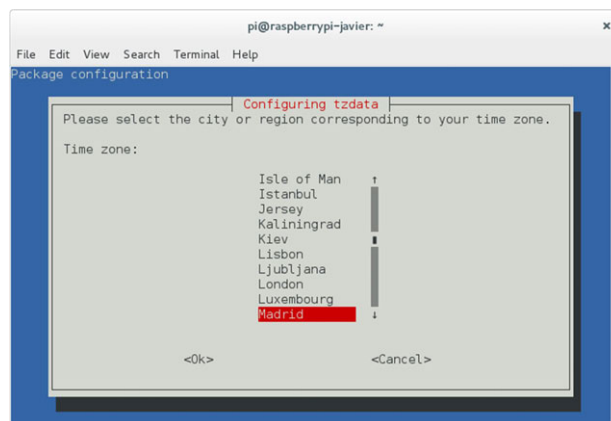


Figure 30: Capital cities list for time zone

4. Finally, date command was executed in order to verify that the system date was updated.

In order to make the feature able to be usable all of the time (previous configuration would not work in case of no Internet access), a RTC clock was set following the next steps:

1. Firstly, i2c was enabled following these steps:
 - a. Command `sudo raspi-config` was executed. In the menu, the option 9 'Advanced Options' was selected:

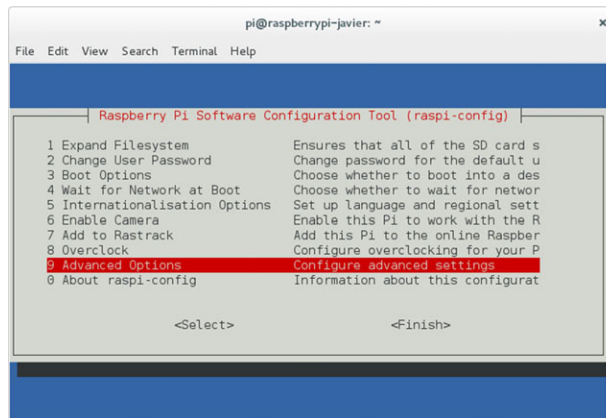


Figure 31: Main menu of Raspberry Pi settings

- b. In the options menu, option 'A7 I2C' was selected:

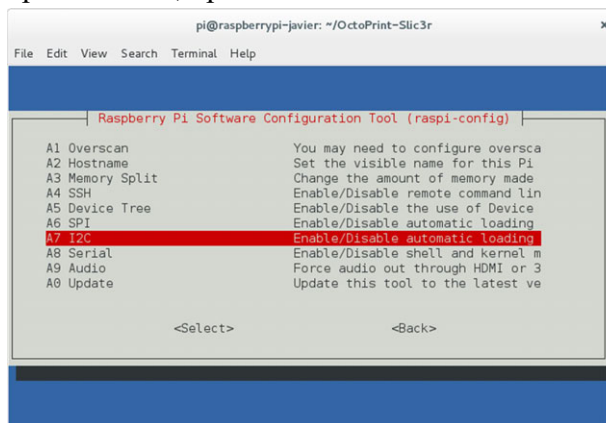


Figure 32: Advanced options menu

- c. When asked about enabling the I2C interface, the 'Yes' option was selected:

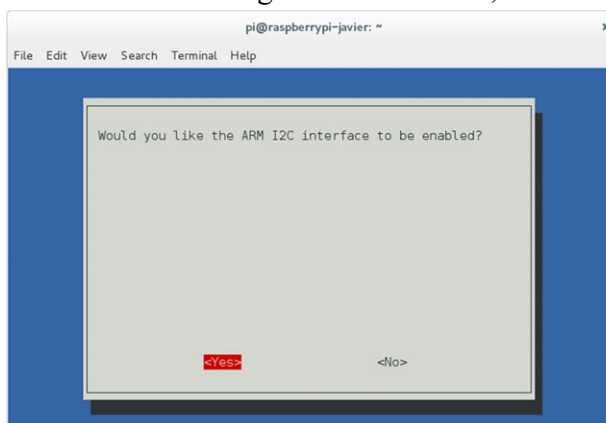


Figure 33: Habilitation of I2C interface

- d. A message to indicate that I2C will be enabled after rebooting was shown. To continue with the process, 'Ok' was pressed:

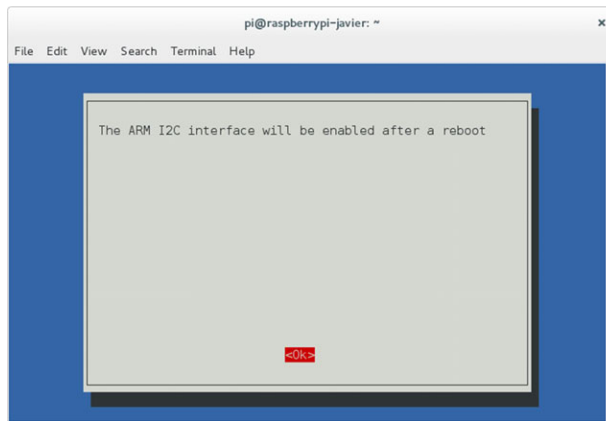


Figure 34: Confirmation message for I2C habilitation

- e. Next, an option about loading the I2C kernel module by default appeared. As it was necessary to make use of the RTC, the 'Yes' option was selected.

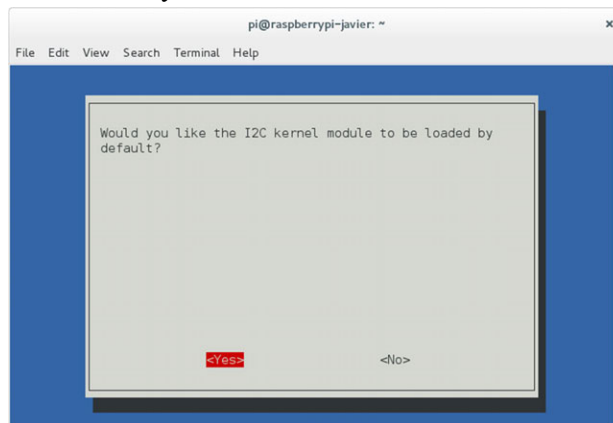


Figure 35: I2C kernel modules load confirmation menu

- f. A message was shown, confirming that the kernel module would be loaded by default from that moment.

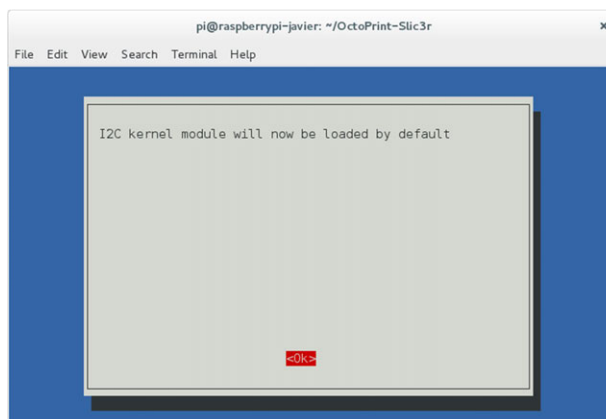


Figure 36: I2C kernel modules load confirmation

Appendix

- g. For the last thing, a reboot was required in order to apply changes. To finish, the 'Yes' option was selected.

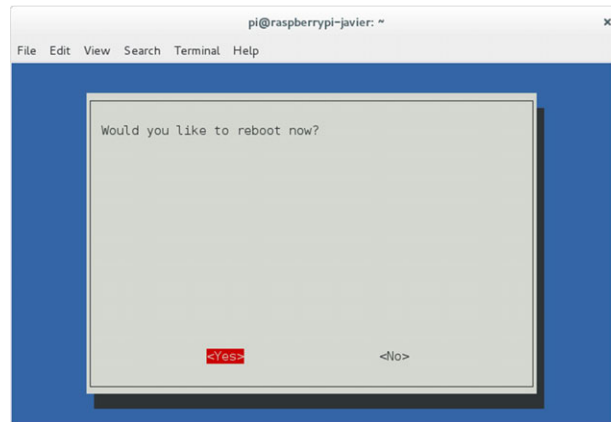


Figure 37: Reboot menu

2. With the reboot completed, the Raspberry Pi was shut down and wires were connected as in the following image before booting again:
- GND – GND
 - VCC – 5V
 - SDA – SDA
 - SCL – SCL

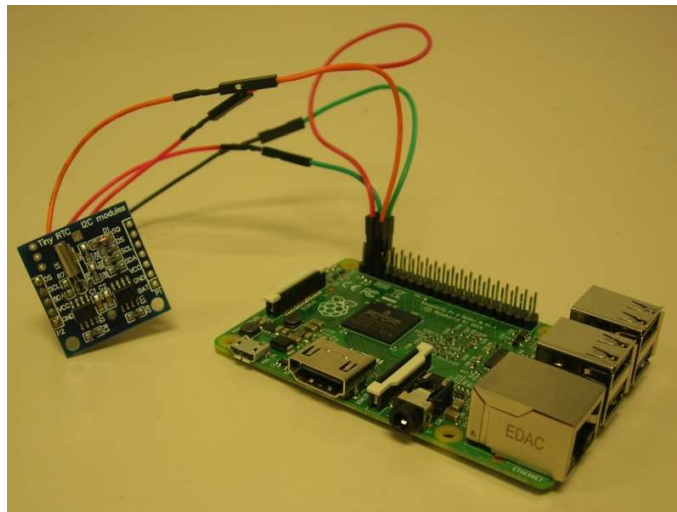


Figure 38: RTC connection with a Raspberry Pi

3. To verify that the device is detected, the command `ls /dev/i2c*` was executed, confirming the detection:



```

pi@raspberrypi-javier: ~
File Edit View Search Terminal Help
pi@raspberrypi-javier:~$ ls /dev/i2c*
/dev/i2c-1
pi@raspberrypi-javier:~$

```

Figure 39: I2C detection

- Next, the command `sudo modprobe rtc-ds1307` was executed (the parameter included will vary depending on the model).



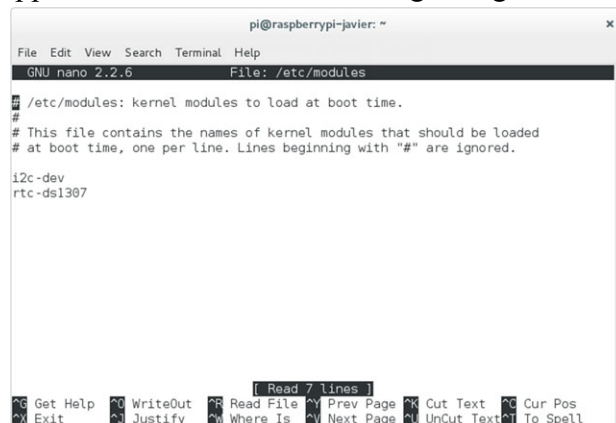
```

pi@raspberrypi-javier: ~
File Edit View Search Terminal Help
pi@raspberrypi-javier:~$ ls /dev/i2c*
/dev/i2c-1
pi@raspberrypi-javier:~$ sudo i2cdetect -y 1
sudo: i2cdetect: command not found
pi@raspberrypi-javier:~$ sudo modprobe rtc-ds1307

```

Figure 40: Modprobe command

- To continue with the configuration, the file `modules` stored in `etc` folder was edited using the command `sudo nano /etc/modules`. With the editor opened, the line `'rtc-ds1307'` was appended to the file before saving changes.



```

pi@raspberrypi-javier: ~
File Edit View Search Terminal Help
GNU nano 2.2.6 File: /etc/modules

# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.

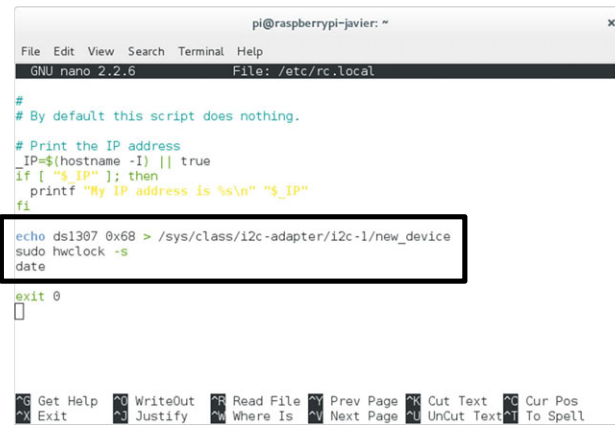
i2c-dev
rtc-ds1307

```

Figure 41: Addition of the RTC module to modules file

- With the file saved, `rc.local` file was edited by executing the command `sudo nano /etc/rc.local`, adding the lines indicated above the last line (the one containing `'exit 0'`) that adds the `i2c` device and gets the hardware date during the boot process. If the RTC used is not the same as in the example, it is possible for the device address (`0x68` in this case) to be different.

Appendix



```
pi@raspberrypi-javier: ~  
File Edit View Search Terminal Help  
GNU nano 2.2.6 File: /etc/rc.local  
  
# By default this script does nothing.  
  
# Print the IP address  
_IP=$(hostname -I) || true  
if [ "$_IP" ]; then  
    printf "My IP address is %s\n" "$_IP"  
fi  
  
echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-1/new_device  
sudo hwclock -s  
date  
  
exit 0  
^_
```

Figure 42: Addition of the RTC module in rc.local file