Ing. Hernán Badel [a], Ing. Gustavo Suenaga [b]

[a] Universidad de La Marina Mercante. Facultad de Ingeniería. hernan.badel@docentes.udemm.edu.ar

Diseño de una impresora 3D controlada de manera remota

Resumen: *El presente trabajo consiste en el rediseño de una impresora 3D que fue desarrollada por un grupo de docentes y alumnos en la UdeMM. Esta nueva versión, en la que también participan activamente alumnos de Ingeniería, a diferencia de la anterior, es administrada y controlada a través de un sitio web, lo cual posibilita el acceso a todas sus funcionalidades de manera remota ya sea en una red interna o en un sitio publicado en internet.*

Se incorporó para ello una placa Raspberry Pi y un software Open Source llamado OctoPrint, embebido dentro de este hardware. Ambos elementos tienen un gran potencial.

La placa es, por su versatilidad, casi un ordenador de reducidas dimensiones, y el software al ser un proyecto de código abierto, es sumamente adaptable a las necesidades e inquietudes de los usuarios.

De este modo la facultad de ingeniería dispone de un laboratorio integrado, en donde además de contar con los recursos para poder realizar todo el proceso completo, desde el diseño hasta la impresión de la pieza, también posibilita sumarlo a otro proyecto de la UdeMM, el denominado Laboratorio Remoto.

Palabras Claves: Impresora 3D, manufactura aditiva, Open Source, Laboratorio Remoto, Prototipo.

1. Introducción

El presente trabajo es la continuación del proyecto de construcción de una impresora 3D, cuya primera etapa requirió de adaptar partes mecánicas y electrónicas en conjunto con el firmware y software asociado de manera que la misma pueda imprimir piezas a partir de un diseño tridimensional realizado en algún programa del tipo CAD [1].

En esta instancia se hará foco en el software y hardware involucrado, los protocolos utilizados y la relación entre los mismos.

Actualmente se está utilizando como software de control de la impresora instalado en una PC de escritorio el Repetier Host, el cual dispone de configuraciones, entre las cuales podemos destacar la configuración de la impresora, su modo de comunicarse con la misma, y el nexo con un programa de slice, el cual permite seccionar la pieza en capas para su posterior impresión, en nuestro caso adoptamos el Slic3r, en éste se cuenta con múltiples configuraciones que afectan a la impresión de la pieza, su terminación, siendo que a mayor calidad en la misma, le llevará más tiempo a la impresora su terminación.

Continuando con el Software-Host, es a través del Usb que se comunica con la impresora, dialogando con el firmware instalado en la placa Rumba, la misma fue diseñada específicamente para controlar la mecánica, los periféricos asociados a una impresora 3D y es compatible con el proyecto Arduino [3], el software grabado en su microprocesador que interrelaciona dicha placa con la PC es el Firmware Marlin. En dicho firmware se configuran aspectos que hacen a la mecánica involucrada en la impresora.

2. Software Host

El software que se utiliza para realizar el control de la impresora es el Repetier Host, el mismo fue elegido entre los disponibles al momento del inicio del armado de la impresora, se decidió por

éste debido a su filosofía de software libre (se ha podido bajar el proyecto completo, el cual compila correctamente, de manera de poder introducir mejoras y nuevas funcionalidades) y por ser el más versátil entre sus competidores, ver figura 1.

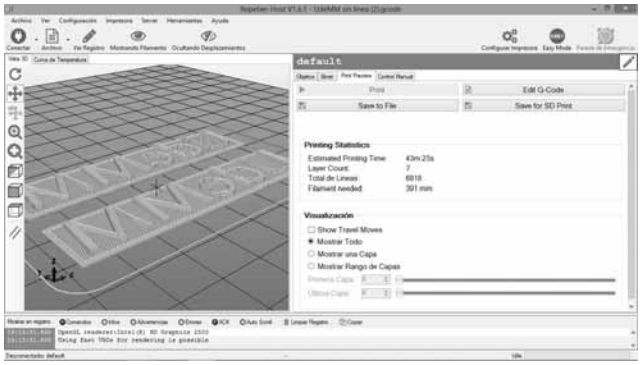

Fig. 1, Aspecto del Repetier Host, mostrando una figura lista para imprimir.

En el mismo se configuran aspectos como la conexión a la PC, velocidades de los ejes, extrusores, dimensiones y espacio de impresión, entre otros.
En la actualidad se puede observar que el proyecto abandonó el estado de libre con las fuentes disponibles para su descarga, con lo cual el programa sigue estando disponible de manera gratuita, ofreciéndose funciones adicionales con costo.
Es por esto que el grupo investigador comenzó a buscar alternativas al Repetier Host, pues adhiere a la filosofía de software libre.

3. Software Slice

El software slice utilizado en este proyecto es el Slic3r [4], elegido por ser también de tipo open source, el mismo permite dividir la pieza en capas para su posterior impresión y se integra perfectamente con el Repetier Host
En el mismo se configuran varios aspectos que modificarán la calidad, tiempo de impresión y resistencia estructural de la pieza, pues permite variar el tipo de relleno entre capas.
La cantidad de ajustes es tan extensa que excede al alcance de este documento, pero se destacan los siguientes, a saber, altura de la capa de impresión, tipo de relleno (diferentes formas geométricas), material de soporte para piezas en voladizo, retracción del filamento (evita la formación de hilos entre diferentes objetos de la pieza).
En la figura 1, se puede observar la división en capas de la pieza a imprimir, mientras que en la figura 2 se muestra una de las solapas de configuración.

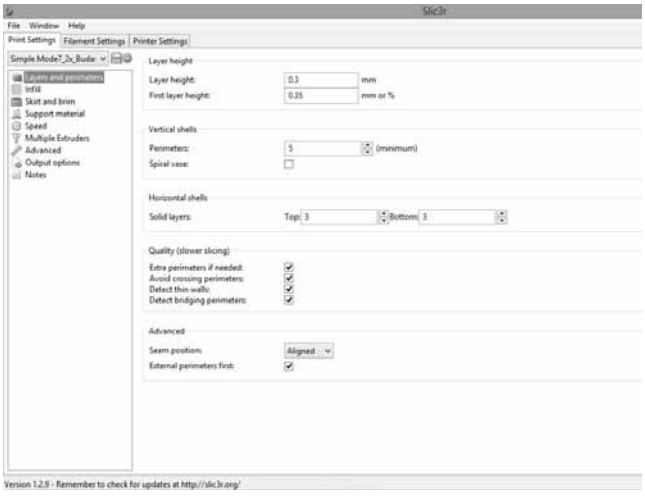

Fig. 2, una de las tantas configuraciones del Slic3r, los ajustes de la impresora.

4. Firmware Marlin

Marlin es un firmware de código abierto, destinado a controlar impresoras 3D [5] de la familia RepRap [2], se lo denomina firmware, porque es un programa que se graba en el microprocesador de la placa que controla el hardware de la impresora, en nuestro caso la placa en cuestión es la Rumba.
Antes de optar por el Marlin se probó con el firmware de Repetier, el cual presentó inconvenientes a la hora de configurar la velocidad de trabajo de los motores.
Otro de los aspectos destacables del Marlin es que fue adoptado es por varias impresoras 3D comerciales ya establecidas como: Printrbot, Ultimaker, Lulzbot y Prusa Research, también es capaz de manejar equipos de CNC y por sobre todas las cosas es de código abierto y gratuito a partir del 2011.
En el mismo se configuran aspectos específicos del hardware, para el correcto funcionamiento de la impresora, en general se realizan ajustes y configuran parámetros, no es necesario realizar agregados al mismo, ejemplo de configuración en la solapa configuration.h correspondiente a los finales de carrera y los límites de recorrido de los sensores:

```
// ENDSTOP SETTINGS:
// Sets direction of endstops when homing; 1=MAX, -1=MIN
#define X_HOME_DIR 1 // -1
#define Y_HOME_DIR 1 // -1
#define Z_HOME_DIR 1 // -1

#define min_software_endstops true // If true, axis won't move to coordinates less than HOME_POS.
```

#define max_software_endstops true // If true, axis won't move to coordinates greater than the defined lengths below.

// Travel limits after homing (medidas de la impresora)
#define X_MAX_POS 260
#define X_MIN_POS 0
#define Y_MAX_POS 260
#define Y_MIN_POS 0
#define Z_MAX_POS 134 //174 //155 //Impresora 3D Udemm
#define Z_MIN_POS 0

5. Placa controladora Rumba

RUMBA (R.eprap U.niversal M.ega B.oard con controlador A.llegro), es una placa controladora, en concordancia con el proyecto RepRap, destinada al control del hardware de impresoras 3D, es una de las más completas, permite, entre otras opciones, conectar hasta 3 extrusores [6] , ver figura 3. En la figura 4 puede apreciarse un esquema de las conexiones entre la placa y sus periféricos.

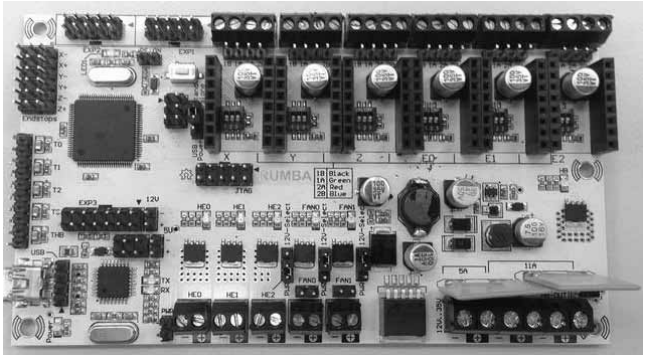

Fig. 3, Placa Rumba (extraído de la página del proyecto RepRap).

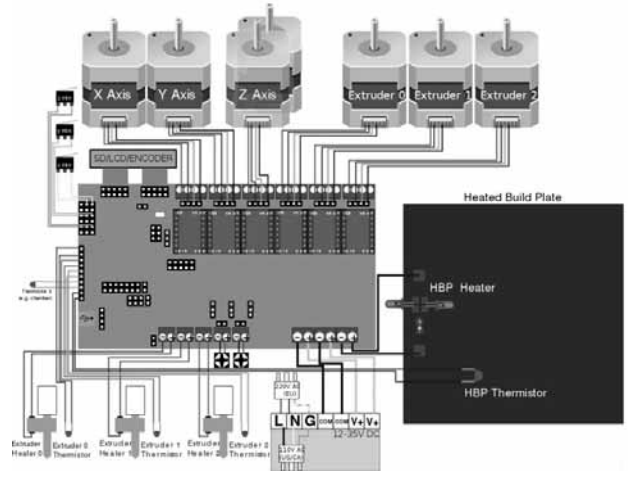

Fig. 4, Placa Rumba, conexionado (extraído de la página del proyecto RepRap).

6. Plataforma Raspberry Pi

La placa Raspberry Pi [7], es un hardware desarrollado en el Reino Unido por Raspberry Pi Fundation con el objetivo de estimular la enseñanza de informática en las escuelas, ver figura 5.

Podemos mencionar, a grandes rasgos, que la misma se comporta como una PC de escritorio, con recursos de memoria y procesamiento limitados, con lo cual permite instalar versiones simplificadas y adaptadas de sistemas operativos destinados a computadoras convencionales, como ser: Raspbian (basado en Debian de Linux), Pidora (basado en Fedora de Linux) y Windows IoT Core (basado en Windows 10), entre otros.

La placa cuenta con 1 GB de memoria RAM, un procesador basado en la familia de ARMv8 con 4 núcleos, una velocidad reloj de 1.2 Ghz, e incluye conectividad inalámbrica a través de Wi-Fi y Bluetooth 4.1.

El interés en dicha placa consiste en la posibilidad de instalar el software Host en ella, y, como ya se había mencionado, puesto que el software Repetier Host abandonó la filosofía open Source, el grupo investigador encontró una alternativa interesante en el software OctoPrint, cuyo destino de hardware es la Raspberry Pi.

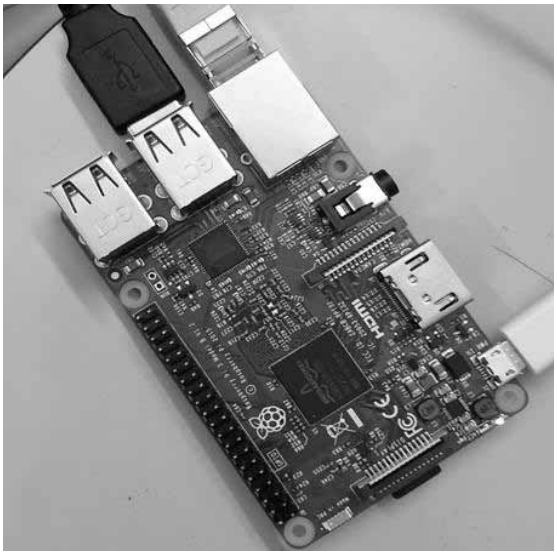

Fig. 5, Placa Raspberry Pi del proyecto en prueba.

7. Software OctoPrint

El software OctoPrint [8], es del tipo Open Source y sus fuentes están disponibles para que cualquier persona pueda modificarlo de acuerdo a sus necesidades o simplemente ver cómo está diseñado. Se basa en el lenguaje Python, ver figura 6.

Controla y monitorea en tiempo real cada aspecto de la impresora 3D y sus trabajos de impresión directamente desde un browser, ver figura 6:

- Permite acceder a una cámara web y observar de forma remota cómo la impresora está fabricando las piezas.
- Tiene integrado un visualizador de GCode que permite ver las instrucciones en ese formato que se envían a la placa Rumba.
- Mantiene un monitoreo continuo sobre la temperatura del HotEnd y del HotBed. Los modifica dinámicamente durante la impresión (ver figura 8).
- Mueve el cabezal de impresión a lo largo de los 3 ejes, también el extrusor. Se pueden agregar controles personalizados(custom) para resolver necesidades particulares.

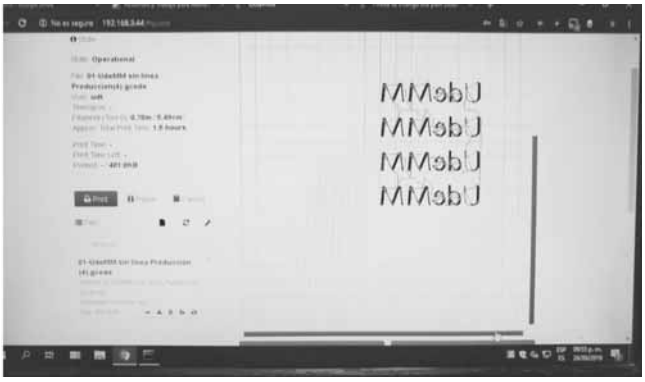

Fig. 6, Pantalla del programa Octoprint, donde se observan las piezas que se están imprimiendo.

El software está compuesto de varios módulos:

7.1. *Control de acceso*

Según el tipo de usuario, el sistema permite realizar más o menos funciones. Si por ejemplo se accede en forma anónima se podrá conocer:

- El estado de la impresión
- Las temperaturas de los dispositivos
- Ver el GCode (no modificar)
- Ver la salida de los comandos
- Tiempos transcurridos durante la impresión

Cuando un usuario introduce sus credenciales y tienen rol Administrador, puede acceder a todas las funciones de configuración.

También es posible deshabilitar este módulo, pero no se recomienda, porque en ese caso cualquier usuario anónimo podría cambiar la configuración del sistema.

7.2. *Controles Custom*

Los controles custom permiten agregar botones para ejecutar comandos GCode simples y también más complejos donde se les puede pasar parámetros.

Para ello se debe editar un archivo llamado config.yaml, donde se configura en una estructura jerárquica todos los controles custom.

Por ejemplo, supongamos un control para establecer la velocidad del ventilador, la estructura sería la siguiente:

controls:
- **name**: Fan
layout: horizontal
children:
- **name**: Enable Fan
command: M106 S%(speed)s
input:
- **name**: Speed (0-255)
parameter: speed
default: 255
slider:
min: 0
max: 255
- **name**: Disable Fan
command: M107

Y en la página web se vería lo siguiente:

7.3. *Scripts GCODE*

Se pueden definir scripts custom en GCode para ocasiones específicas. Por ejemplo, cuando la impresora arranca o cuando OctoPrint se conecta a la impresora.

Para ello, OctoPrint busca los scripts por defecto en la carpeta scripts/gcode. Obviamente esta carpeta puede ser cambiada en el archivo config.yaml.

Para programar scripts GCode, se utiliza Jinja2 que es un motor de plantilla web para el lenguaje de programación Python.

7.4. *Scripts Predefinidos*

Son enviados automáticamente por OctoPrint

- afterPrinterConnected: enviado exactamente luego que OctoPrint se conectó a una impresora. Por defecto está vacío.
- beforePrinterDisconnected: enviado antes de que OctoPrint cierre una conexión activa a una impresora. Por defecto está vacío. No será enviado si el corte se produjo por un error en la conexión USB.
- beforePrintStarted: enviado exactamente antes que comience el trabajo de impresión

- afterPrintCancelled: enviado exactamente luego de que el trabajo de impresión es cancelado.
- afterPrintDone: enviado exactamente luego de que el trabajo de impresión finaliza. Por defecto es un script vacío
- afterPrintPaused: enviado exactamente luego de que el trabajo de impresión sea pausado. Por defecto es un script vacío.
- beforePrintResumed: enviado exactamente antes que el trabajo de impresión sea reanudado.

7.5. *Snippets*

Para hacer que pequeños fragmentos GCODE (Snippets) sean reutilizables en una plantilla (por ejemplo, para deshabilitar todos los hotends) hay un comando adicional de plantilla Jinja {% snippet '<snippet name>'%} disponible que permite incluir fragmentos almacenados bajo la carpeta scripts/gcode/snippets en el directorio de configuración de OctoPrint. Son totalmente compatibles con todo el espectro del lenguaje de plantillas Jinja2.

7.6. *Contexto*

Todos los scripts GCODE tienen acceso a una lista de variables de contexto.

- printer_profile: incluye información tal como cantidad de extrusores, diámetro del filamento, etc.
- last_position: última posición reportada por la impresora. Utiliza para ello el mensaje M114 de GCODE.
- last_temperature: temperaturas actuales y de trabajo de todos los hotends y de la cama caliente. Se expresan como una tabla de pares clave-valor. Las claves de los hotends se numeran empezando por el cero y la clave para la cama caliente es una “b”. Las temperaturas están expresadas en grados Celsius.

Ejemplo de Scripts:

```
afterPrintCancelled:
    ; deshabilita motores
    M84

    ;deshabilita hotends y hotbeds
    {% snippet 'disable_hotends' %}
    {% snippet 'disable_bed' %}

    ;apaga el ventilador
    M106 S0
```

7.7. *Comandos Action*

Los comandos Action son una característica definida para el protocolo de comunicación RepRap basado en GCODE. Para distinguir un comando de este estilo, se les antepone //, de la siguiente manera:

// action:*comando*

Por ejemplo:

pause: Cuando se recibe este comando desde la impresora, OctoPrint pausará un trabajo de impresión actual como si se hubiera hecho clic en el botón “Pausa".

paused: Cuando se recibe este comando desde la impresora, OctoPrint pausará un trabajo de impresión actual pero sin activar ningún script GCODE. Esto puede ser interesante para el firmware que desea enviar una señal a OctoPrint de que una impresión debe pausarse, pero sin ninguna interferencia de control de OctoPrint, por ejemplo, en el caso de un cambio de filamento totalmente gestionado por el firmware.

resume: Cuando se recibe este comando desde la impresora, OctoPrint reanudará un trabajo de impresión actual como si se hubiera hecho clic en el botón “Reanudar".

resumed: Cuando se recibe este comando de la impresora, OctoPrint reanudará un trabajo de impresión actual, pero sin activar ningún script GCODE. Esto puede ser interesante para el firmware que quiere enviar una señal a OctoPrint de que se debe reanudar una impresión, pero sin ninguna interferencia de control de OctoPrint, por ejemplo, en el caso de un cambio de filamento totalmente gestionado por el firmware.

disconnect: Cuando se recibe este comando de la impresora, OctoPrint se desconectará de inmediato.

cancel: Cuando se recibe este comando desde la impresora, OctoPrint cancelará un trabajo de impresión actual como si se hubiera hecho clic en el botón “Cancelar".

7.8. *Plugins*

Los plugins o complementos, se pueden usar para ampliar la interfaz web de OctoPrint, ya sea para ejecutar tareas específicas en el inicio y apagado del servidor, para proporcionar interfaces de usuario completas con funcionalidades especiales, para responder adecuadamente a eventos del sistema, informes de progreso, o también para agregar soporte para slicers adicionales.

Los plugins pueden descargarse de diferentes sitios, como por ejemplo plugins.octoprint.org, y también en el repositorio de fuentes en GitHub.

Por otro lado, también es posible desarrollar complementos propios si es que se necesitan características especiales no cubiertas por los existentes.

Existen diversas maneras de instalarlos, desde copiarlos manualmente en una carpeta específica para plugins, utilizando el plugin manager o sino como un módulo python.

Si el plugin se encuentra en el Python Package Index (PyPi), la instalación es tan simple como introducir:

pip install <plugin_name>

Y si estuviese en el repositorio GIT del proyecto:

pip install https://github.com/OctoPrint/OctoPrint-Growl/archive/master.zip

7.9. *Modo Seguro*

La gran ventaja que supone que Octoprint soporte plugins, también trae como efecto colateral que muchas veces aparezcan errores en el funcionamiento que son adjudicados al proyecto, y en realidad son problemas introducidos por los plugins, en su mayoría desarrollados por terceras partes.
Para poder detectar estos casos fácilmente, Octoprint puede ser ejecutado en Modo seguro, que inicia el programa original y los plugins propios, y deja pausados los plugins de terceros. Permitiendo de este modo identificar fácilmente el origen de algún comportamiento fallido.
Existen 3 maneras de iniciar Octoprint en Modo Seguro, ver figura 7:

- Desde el menú "System", Restart Octoprint in Safe Mode.
- Modificar el flag "server.startOnceInSafeMode" poniéndolo en valor "true", esto es dentro del archivo config. yaml
- Iniciando Octoprint desde la consola de comandos:

octoprint serve --safe

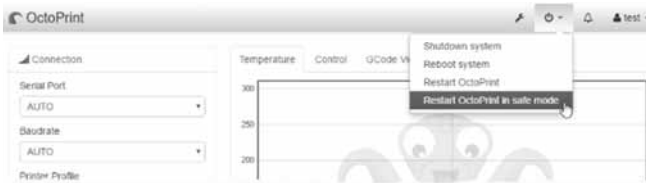

Fig. 7, se observa el menú desplegable para pasar a funcionar en Safe Mode (modo seguro)

7.10. *API REST*

Autorización:
Cuando el Control de Acceso se encuentra habilitado, la API de Octoprint pide una clave (API key) por cada Request. Puede ser una clave global para todos los usuarios o individual.
La clave es enviada en el HTTP header como X-Api-Key:

```
GET /api/files HTTP/1.1
Host: example.com
X-Api-Key: abcdef...
```

Es importante destacar que si el Control de Acceso se encuentra desactivado, cualquier usuario anónimo podrá enviar Request con derechos de Administrador. Se recomienda por ello no desactivar este plugin.
Tipo de Contenido:
El tipo de datos por defecto que retorna Octoprint es JSON.

```
Content-Type: application/json
```

Peticiones realizadas desde otras plataformas:
Para poder acceder a la API de Octoprint desde cualquier sitio web diferente de la interface web de Octoprint, debe habilitarse CORS (Cross-Origin Resource Sharing).
Para habilitar esta característica, se pone en **true** el valor allowCrossOrigin del archivo config.yaml.

```
api:
  enabled: true
  key: ...
  allowCrossOrigin: true
```

Si CORS no está habilitado se produce un error del siguiente tipo:

```
XMLHttpRequest cannot load http://localhost:8081/api/files. No 'Access-Control-Allow-Origin'
header is present on the requested resource.
```

Fig. 8, se observa las temperaturas del HotEnd y del HotBed.

8. Trabajo de Campo: Desarrollo de la aplicación

Pensando en el objetivo de lograr acceder y controlar la impresora 3D remotamente se investigaron diferentes proyectos existentes, de todos ellos resultó elegido el Octoprint por prestaciones, sencillez de manejo y ser de tipo libre con fuentes disponibles para su descarga, puesto que el mismo fue concebido para correr en una placa Raspberry PI, se adquirió la misma e instaló el software en ella.
Luego de configurar algunos parámetros básicos del programa se logró imprimir algunas piezas con aceptable calidad, pero se observó la necesidad de adaptar algunas de sus características. Era necesario investigar intrínsecamente el software.
Por este motivo, el siguiente paso fue instalar el entorno de programación del proyecto Octoprint en una PC para poder realizar algunas modificaciones convenientes antes de grabarlo en la Raspberry PI, ver figura 9.

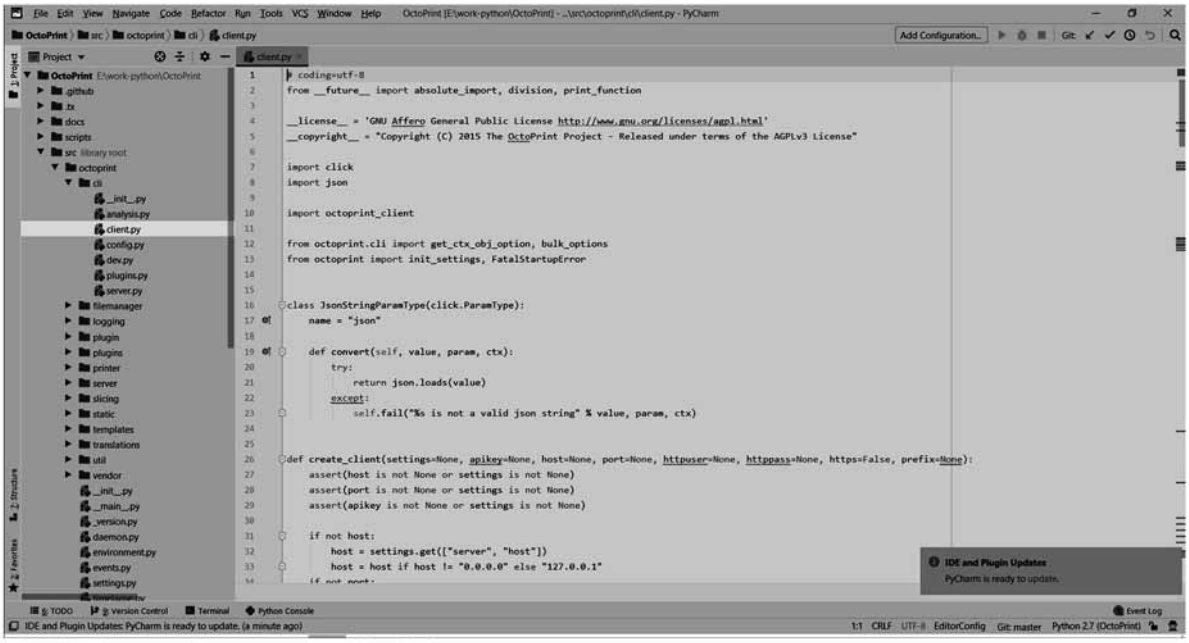

Fig. 9, Entorno de programación del proyecto.

El proyecto en sí mismo es Open Source y se lo puede descargar libremente del repositorio GitHub.
El lenguaje de programación utilizado es JavaScript para el front end, y Python 2.7 para el back end. El IDE (**I**ntegrated **D**evelopment **E**nvironment) que se eligió para trabajar en Python es PyCharm, que es gratuito.
Una facilidad interesante que ofrece el proyecto Octoprint, es poder configurar una impresora virtual, y de ese modo depurar el programa sin tener la necesidad de conectarse todo el tiempo a una impresora real.

8.1. *Resultados*

Se cumplieron los objetivos planteados en esta primera etapa, que eran poder imprimir de manera remota para lograr que la impresora se integrara al proyecto de la UdeMM, "Laboratorio Remoto", ver figura 10.

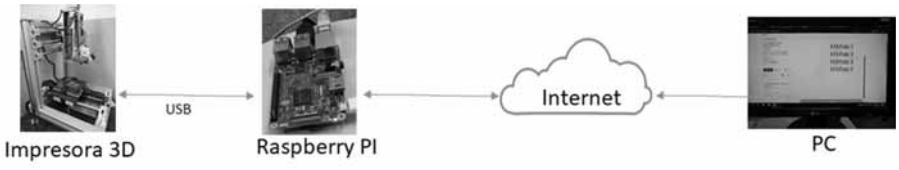

Fig. 10, Diagrama esquemático de interconexión.

El programa Octoprint instalado en la Raspberry Pi se pudo conectar a la impresora 3D y realizar algunas impresiones de prueba de manera exitosa.

9. Conclusiones

El software sirvió para poder acceder a la impresora 3D, realizar y monitorear la impresión desde una interface web, permitirá su control a través de una red interna, para dar acceso a los alumnos y/o profesores dentro del ámbito de la UdeMM que así lo requieran, y, a través de un acceso a internet, su control de manera remota.

10. Trabajos Futuros

Se modificará el programa para alcanzar algunas funcionalidades que están presentes en el proyecto Repetier y que nos parecen útiles.
Se configurará el sistema para monitorear la impresión con una cámara web en tiempo real.

11. Referencias

[1] “Desarrollo de una impresora 3D basada en tecnología FDM”, (Noviembre 2017), H, Badel; G. Suenaga; Federico García, Atenea 2017, ISSN: 1668-348X. Proyecto RepRap,[online] disponible en: http://reprap.org/. Proyecto Arduino,[online] disponible en: https://www.arduino.cc/

[2] Proyecto Slic3r, G-code generator for 3D printers,[online] disponible en: http://slic3r.org/. Proyecto Marlin, Open-Source 3D Printer Firmware,[online] disponible en: http://marlinfw.org/. Placa Rumba, características e información disponible en: https://reprap.org/wiki/RUMBA#Where_to_get_it.3F. Placa Raspberry Pi, características y aplicaciones disponible en: https://www.raspberrypi.org/. Proyecto Octoprint, [online] disponible en: https://octoprint.org/