Desenvolvimento de uma interface para comando e monitorização de Impressoras 3D no âmbito da indústria 4.0

Francisco Luís Machado Campos

Dissertação de Mestrado

Orientador na FEUP: Prof. António José Pessoa de Magalhães

Coorientador no INEGI: Eng. Tiago Abreu

Mestrado Integrado em Engenharia Mecânica

Ramo de Automação

Setembro de 2019

Resumo

O trabalho da presente dissertação foi desenvolvido no INEGI e consiste no estudo e desenvolvimento de *software* para comando e monitorização de impressoras 3D, com vista à criação de uma aplicação IoT que permite a adaptação destes equipamentos a sistemas de produção modernos e abertos idealizados pela Indústria 4.0. Trata-se de uma proposta inovadora que visa ultrapassar o conceito de máquina produtiva isolada inerente às impressoras 3D, com o fim de obter um sistema que permita a interligação com outros dispositivos ou sistemas, nomeadamente um *Cyber Physical System*.

Para esse efeito, foi contextualizado o processo de fabrico de impressão 3D de forma a compreender as dificuldades na monitorização de impressoras 3D, e daí enunciado, especificamente, o problema e apresentada a arquitetura de um sistema baseado num *Cyber Physical System* para o solucionar, que consiste na interposição de uma interface numa habitual comunicação entre a máquina e o seu PC de comando. Todas as tecnologias que permitem a construção deste sistema são, seguidamente, especificadas.

A aplicação IoT foi desenvolvida na ferramenta Node-Red. Esta permite obter dados de dois tipos de *firmware* de impressoras 3D, sendo depois enviados através do protocolo de comunicação MQTT para uma plataforma IoT, onde será efetuada a monitorização do equipamento. Para além disso, a aplicação IoT permite a comunicação simultânea entre a impressora 3D com a plataforma IoT e o seu PC de comando. Desta forma, obtém-se uma solução que durante uma impressão de uma peça, para além de realizar a monitorização do equipamento, permite também enviar comandos de controlo e de configuração simultaneamente.

A solução proposta é de natureza industrial, sendo, nessa medida, bastante robusta e, ao mesmo tempo, flexível. A sua estruturação permite uma simples implementação e a introdução de melhorias, sem que haja necessidade de alterar o sistema base.

Por fim, com a realização de testes, conclui-se que a solução desenvolvida cumpre os requisitos mínimos estabelecidos para ambos os *firmwares*, com os dados de uma impressão a serem obtidos com sucesso na plataforma IoT e a comunicação simultânea a ser implementada também com sucesso.

Palavras-chave: Internet das coisas, Sistemas Ciber Físicos, Impressão 3D, Interface de comando.

Development of an interface for 3D printer control and monitoring

Abstract

The work of the present dissertation was developed at INEGI and consists in the study and development of software for command and monitoring of 3D printers in order to create an IoT application that allows the adaption of these devices to modern and open production systems designed by Industry 4.0. This is an innovative that aims to go beyond the concept of isolated productive machine of 3D printers in order to obtain a system that allows interconnection with other devices or systems, namely a Cyber Physical System.

For this purpose, process of manufacturing, 3D printing, was contextualized in order to understand the difficulties in monitoring 3D printers and then, specifically, the problem was presented such as the architecture to solve it, which consists in the interposition of an interface in a normal communication between the machine and its control PC. All technologies that allow the construction of this system are then specified.

The IoT application was developed in the Node-Red tool. This application allows data to be obtained from two types of 3D printer firmware, and then sent via MQTT communication to an IoT platform where the equipment will be monitored. In addition, the IoT application allows simultaneous communication between the 3D printer with the IoT platform and the control PC. This provides a solution that during a one-piece printing not only monitors the machine but also allows control and configuration commands to be sent simultaneously.

The proposed solution is of industrial kind and thus it is robust and, at the same time, flexible. Its structure allows simple implementation and improvements, without having to change the base system.

Finally, through testing, it is concluded that the developed solution meets the minimum requirements established for both firmware, with the printing data to be successfully obtained on the IoT platform and the simultaneous communication to be successfully implemented as well.

Keywords: Internet of Things, Cyber Phisical Systems, 3D Printing, Control Interface.

Agradecimentos

Em primeiro lugar quero deixar o meu principal agradecimento ao meu pai, à minha mãe e ao meu irmão, como as pessoas mais importantes para mim e que sempre me acompanharam ao longo da minha vida e de todo o meu percurso académico. Sem o seu apoio incondicional, a minha jornada até este momento não teria sido possível.

Gostaria de agradecer ao Professor António Pessoa de Magalhães, ao Engenheiro Tiago Abreu e ao Engenheiro João Paulo Pereira, na qualidade de meus orientadores, pela disponibilidade, motivação e conhecimento transmitido, fundamentais para a realização da presente dissertação.

De uma forma geral, agradeço também a todos os colaboradores do INEGI que me ajudaram em diferentes momentos deste projeto. Ao Engenheiro Luís Oliveira por todas as dúvidas esclarecidas, boa disposição, motivação e, sobretudo, pelo apoio oferecido na implementação prática de inúmeras tarefas. Também ao Engenheiro João Sobral por todas as conversas, discussões abertas e críticas construtivas que muito me ajudaram na realização deste projeto.

Por último mas não menos importante, um grande obrigado a todos os meus amigos que de uma forma ou de outra me aconselharam e me motivaram durante este trabalho e por todo o companheirismo e amizade durante toda a minha jornada académica.

Índice de Conteúdos

Lista de Figuras

Lista de tabelas

Siglas e acrónimos

3D - Tridimensional
AMQP - *Advanced Message Queuing Protocol*
API - *Application Programming Interface*
CAD - *Computer Aided Design*
CNC - Comando Numérico Computorizado
CoAP - *Constrained Application Protocol*
CPS - *Cyber Physical System*
FDM - *Fused Deposition Modelling*
HTML - *HyperText Markup Language*
HTTP - *HyperText Transport Protocol*
IDE - *Integrated Development Environment*
IIoT - *Industrial Internet of Things*
IP - *Internet Protocol*
IPs - *Information Packets*
IoT - *Internet of Things*
JSON - *JavaScript Object Notation*
MES - *Manufacturing Execution System*
MQTT - *Message Query Telemetry Transport*
NoSQL - *Non Structured Query Language*
OEE - *Overall Equipment Effectiveness*
OPC-UA - *Open Platform Communications - Unified Architecture*
QoS - *Quality of Service*
RAMPS - RepRap Arduino Mega Pololu Shield
REST – *Representational State Transfer*
SBC - *Single Board Computer*
SQL - *Structured Query Language*
STL - *Stereolithography*
TCP - *Transmission Control Protocol*
XML - *Extensible Markup Language*

1 Introdução

O aparecimento das tecnologias de impressão 3D veio alterar a maneira como algumas peças são desenvolvidas e produzidas. Tendo começado como uma técnica de prototipagem rápida, a expiração das primeiras patentes em 2005 e, sobretudo do método FDM em 2009, possibilitou que a tecnologia se tornasse mais convencional e acessível ao público em geral, tendo como consequência um aumento exponencial da procura. Através de uma lógica de funcionamento intuitiva e inovadora, a impressão 3D permite uma produção de objetos com elevado nível de detalhe de uma forma rápida, barata e com uma rentabilização quase total do material usado [1].

A evolução recente do processo de impressão 3D coincide com uma cada vez maior presença da digitalização, tanto no quotidiano como na indústria. À crescente digitalização dos processos industriais está associado o fenómeno da Indústria 4.0. A também conhecida como "Quarta revolução industrial", com origem na Alemanha, aparece devido a uma necessidade de aumento de eficiência dos processos, de redução de custos nas empresas e também em virtude dos mais recentes avanços tecnológicos que possibilitaram esta transformação digital [2]. A Indústria 4.0 manifesta-se através de desenvolvimentos em muitas áreas diferentes como CPS (*Cyber-Physical Systems*) ou IIoT (*Industrial Internet of Things*), entre outras.

A presente dissertação enquadra-se nessa perspetiva de avanços tecnológicos que permitem a conjugação de métodos de produção da impressão 3D com os mais recentes desenvolvimentos nas tecnologias de informação e comunicação da Indústria 4.0. Assim sendo, o conceito de máquina produtiva isolada deixará de existir, sendo que os CPS ganharão cada vez maior importância para as capacidades produtivas de uma empresa. O termo CPS abrange uma nova geração de sistemas que integram capacidades computacionais e físicas que permitem monitorizar e controlar os processos físicos, normalmente, por mecanismos de *feedback*. Ou seja, os processos físicos afetam as ações computacionais e vice-versa [3]. O seu potencial sendo enorme, torna possível o seu uso em vários campos de aplicação, como por exemplo, no impedimento de colisões, cirurgia robótica, monitorização do estado de saúde de uma pessoa e muitos mais.

Um dos grandes objetivos da Indústria 4.0 é conectar os dispositivos físicos do nível fabril entre si e entre outros sistemas de nível de produção diferente (por exemplo, o MES (*Manufacturing Execution System*)) e permitir a correlação de informação. Para isso, o sistema de produção físico é transformado num CPS. Os CPS possibilitam a recolha de dados relevantes sobre o equipamento, sendo posteriormente enviados para uma plataforma externa onde serão analisados, garantindo a monitorização do processo. A presente dissertação insere-se nesse contexto. Desta forma, pretende-se transformar o sistema de produção isolado das impressoras 3D num CPS, que torna este equipamento físico como uma das componentes deste sistema.

Este trabalho será inovador, na medida em que não existe uma grande exploração em soluções que permitam a conversão de um modelo produtivo isolado de uma impressora 3D para um CPS. Para além disso, tendo em conta a expiração das patentes de impressoras 3D, cada vez mais existe exploração nesta área, com o número de impressoras a aumentar exponencialmente. Assim, com a evolução nas tecnologias de informação e comunicação será lógico criar uma solução que permita explorar as potencialidades de ambas as áreas e possibilite a sua aplicação num ambiente industrial.

Desta forma, o projeto tem um enorme potencial, já que a monitorização de impressoras 3D possibilita a otimização de uma tecnologia, que ainda tem margem de desenvolvimento, e a análise da eficiência da produção (como o OEE (*Overall Equipment Effectiveness*) para as máquinas CNC (Comando Numérico Computorizado)), o que irá conduzir à consolidação da impressão 3D enquanto processo de fabrico cada vez mais presente na indústria.

1.1 Indústria 4.0 no INEGI

A presente dissertação de mestrado foi desenvolvida no INEGI (Instituto de Ciência e Inovação em Engenharia Mecânica e Engenharia Industrial) na unidade DPS (Desenvolvimento de Produtos e Sistemas). O INEGI é um Centro de Interface Tecnológica (CIT) dedicado ao desenvolvimento de novas tecnologias e vocacionado para a realização de atividade de inovação e transferência de tecnologia orientada para o tecido industrial. Em atividade desde 1986, nasceu no seio do então Departamento de Engenharia Mecânica e Gestão Industrial (DEMEGI) da Faculdade de Engenharia da Universidade do Porto (FEUP).

A unidade DPS dedica-se ao processo de conceção e projeto de produtos e sistemas, estando presente em todas as suas fases, desde as mais concetuais até à produção de protótipos e arranque da produção, podendo estar também presente em outras áreas. Neste momento existe uma estratégia de investimento desta unidade que consiste na reformulação de um laboratório de Fabrico Aditivo, que contém várias impressoras 3D (uma equipa está destacada para trabalhar com as máquinas e os produtos deste processo), e que será dedicado à conceção de projetos na área da Indústria 4.0 [4]. Estes projetos são direcionados ao desenvolvimento de soluções aplicáveis na indústria. Desta forma, o trabalho a realizar foca-se no desenvolvimento de uma solução adaptável à indústria que possibilite a convergência do processo de impressão 3D com as tecnologias de informação promovidas pela Indústria 4.0, capaz de se adaptar às impressoras que constituem o laboratório.

1.2 Objetivos do projeto

Atendendo ao facto de o INEGI ter como missão implementar avanços tecnológicos nas áreas de Fabrico Aditivo e da Indústria 4.0, o trabalho desenvolvido está inserido num estudo sobre desenvolvimentos do laboratório de Indústria 4.0. Este projeto em particular insere-se numa estratégia de desenvolvimento de ferramentas para a adaptação de impressoras tridimensionais aos ambientes de produção sustentado pela Indústria 4.0, ou seja, possibilitar às impressoras 3D a troca de informação, referente ao seu processo, com outros dispositivos.

Nesse sentido, este trabalho terá como foco o desenvolvimento de *software* para um sistema embebido (interface) de forma a realizar a comunicação entre impressoras 3D e uma plataforma IoT (*Internet of Thing*s) externa que permita a sua monitorização. No final dos desenvolvimentos, deverá obter-se um sistema capaz de ser aplicado a várias impressoras, com o fim de monitorizar e comandar o processo que envolve as impressoras 3D.

A presente dissertação tem como principal objetivo o desenvolvimento de uma interface para comando e monitorização de Impressoras 3D no âmbito da Indústria 4.0. Sendo este trabalho de carácter exploratório, tenciona-se desenvolver um sistema que, do ponto de vista funcional, permita a obtenção de dados de impressoras 3D através de uma linguagem específica e própria destes equipamentos e enviá-los para uma plataforma IoT externa. Adicionalmente, pretende-se que a solução desenvolvida permita a comunicação simultânea entre uma impressora 3D com o seu PC de comando e um sistema externo de monitorização. Entre impressora e PC de comando a comunicação deverá ser bidirecional e unidirecional no sentido de impressora para plataforma.

Desta forma, a solução a criar deverá ser flexível, o mais independente do fabricante/modelo da máquina que for possível e possibilitar a realização de alterações e melhoramentos no futuro.

1.3 Linhas orientadoras do projeto

A metodologia adotada para a realização do projeto, numa primeira fase, centralizou-se numa pesquisa de soluções alternativas à que foi inicialmente sugerida pelo INEGI. Após aferir que a solução proposta pelo INEGI se constituía como a escolha mais viável, foi elaborado um estudo sobre todas as componentes tecnológicas referentes a esta escolha, nomeadamente, sobre o processo de impressão 3D (com o foco a incidir na máquina e na forma como esta comunica), a comunicação com a plataforma IoT e a ferramenta de programação a utilizar.

A partir do conhecimento sólido relativamente às componentes tecnológicas que envolveram este projeto, foram inicialmente realizados testes de interação entre uma impressora 3D e a aplicação a desenvolver. Posteriormente, foram desenvolvidos os meios para estabelecer comunicação entre uma impressora 3D e a plataforma IoT externa, na qual funcionalidades que possibilitam comunicar dados relevantes são também desenvolvidas. De seguida, foi estabelecida comunicação simultânea entre impressora 3D com o PC de comando e plataforma IoT. Por fim, foi adaptada a aplicação desenvolvida a outras impressoras 3D.

1.4 Estrutura da dissertação

A presente dissertação está organizada em seis capítulos distintos:

Neste capítulo inicial, “Introdução” é realizada uma introdução ao projeto, onde é feito um enquadramento, são descritos os principais objetivos e destacada a sua importância na instituição onde este foi desenvolvido.

O segundo capítulo, “Apresentação do problema”, pretende introduzir o problema que motivou esta dissertação. Neste é contextualizado o processo de impressão 3D, definido formalmente o problema associado às impressoras 3D, apresentada a hipótese de arquitetura física que irá sustentar o desenvolvimento da solução, abordadas as funcionalidades e potenciais benefícios que poderão ser obtidos da solução e ainda introduzidos os requisitos gerais que permitem responder ao problema enunciado.

O terceiro capítulo, “Requisitos de Engenharia”, são abordadas as tecnologias que serão utilizadas e que permitem cumprir os objetivos e requisitos do projeto. Desta forma, faz-se um levantamento de conteúdos dessas tecnologias, na qual estão descritos os *softwares* utilizados no processo de impressão 3D, os condicionalismos associados à plataforma IoT usada, o *hardware* de suporte à interface, a *stack* que permite correr a ferramenta de

programação que é utilizada no desenvolvimento da solução e os protocolos de comunicação que permitem a troca de informação entre diferentes elementos do sistema.

O quarto capítulo, "Desenvolvimento do *software*", foca-se na descrição dos trabalhos referentes ao desenvolvimento da aplicação IoT pretendida como solução do projeto.

No quinto capítulo. "Testes e Resultados", são apresentados os testes efetuados para validação da solução desenvolvida. Neste capítulo, os resultados decorrentes desses testes são também expostos e sujeitos a análise.

Por fim, o sexto capítulo "Conclusões e Trabalhos futuros", refere-se às conclusões da dissertação e as perspetivas de melhorias que se podem efetuar na solução final, em formato de trabalho futuro.

2 Apresentação do problema

Este capítulo introduz o problema que motivou a presente dissertação, bem como as linhas orientadoras da solução a conceber.

Como ponto de partida é explicada a tecnologia de impressão 3D, as dificuldades de monitorização das atuais impressoras 3D e a forma com este facto condiciona a integração destes equipamentos em ambientes de produção modernos e abertos. Daí emergirá a compreensão do problema a resolver nesta dissertação e respetivos desafios. De seguida, e mais formalmente, são apresentadas e justificadas as diretrizes da solução a desenvolver, em função do problema apontado, e referidas as suas funcionalidades. Os potenciais benefícios industriais resultantes do sistema a criar são também discutidos, sublinhando desde já as espectativas do trabalho a realizar. Por fim, são traçados os requisitos tecnológicos necessários ao projeto.

2.1 Impressão 3D

Para compreender as especificidades das impressoras 3D é necessário abordar primeiro o seu processo. Porém, o grande foco a ter é a integração da impressora 3D na sua vertente máquina, sendo que o processo é apenas uma contextualização para o trabalho realizado.

A produção tradicional de uma peça por impressão 3D envolve uma tecnologia que não se encontra na grande maioria dos métodos de fabrico. A técnica utilizada neste processo permite a conceção de partes, adicionando material camada a camada. Um exemplo ilustrativo do processo FDM, a tecnologia de Impressão 3D mais comum (principalmente para as impressoras de baixo custo), poderá ser observado na Figura 2.1. Nesta tecnologia, um extrusor aquece e deposita o filamento numa base (que estará a uma temperatura inferior à do extrusor) na qual arrefece o material depositado, com o auxílio de um ventilador acoplado à cabeça de extrusão, e o solidifica. O extrusor desta forma move-se para coordenadas, correspondentes aos contornos e enchimento da peça, definidas em cada camada até obter o produto final.

Neste processo de adição é impreterível a elaboração de um modelo digital da peça a produzir antes desta ser obtida fisicamente, assemelhando-se, neste contexto, à impressão de um documento em papel. Tal como na impressão de um livro ou folheto comum é necessária a elaboração de um documento digital na aplicação "*Word"* (ou outra com semelhantes funcionalidades), para depois se proceder à sua impressão em papel; na impressão 3D, através de um desenho CAD (*Computed Aided Design*), cria-se um protótipo digital que, por intermédio de uma série de procedimentos que serão de seguida explicados, se transforma num ficheiro com código que as impressoras 3D conseguem interpretar e executar, resultando na impressão de uma peça [5].

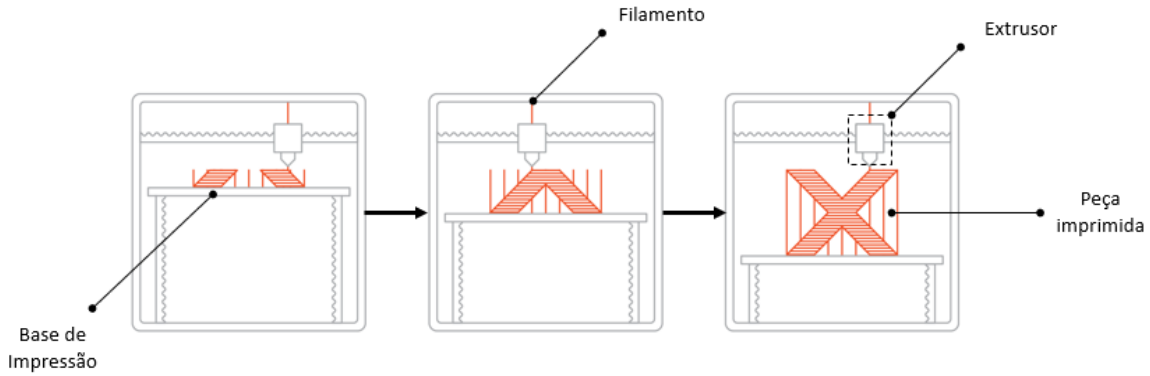


Figura 2.1 - Impressão 3D [6]

Embora haja diversas formas de imprimir tridimensionalmente, existe um conjunto de procedimentos que são comuns à grande maioria das tecnologias deste processo e que permitem a produção de uma peça. As etapas habituais de um processo tradicional de impressão 3D são apresentadas na Figura 2.2. As mudanças de cores nos blocos da Figura representam a transferência do ficheiro, que contém os dados da impressão, para outro *software*.

Figura 2.2 - *Workflow* do processo de Impressão 3D

Descrevendo o *workflow* da Impressão 3D, tal como já foi referido, após a conceção de um modelo CAD da peça a ser produzida, o ficheiro (CAD) terá que ser convertido para um outro que a impressora consiga ler. Para esse efeito, ainda no *software* de modelação CAD, será realizada a conversão do desenho CAD para um ficheiro STL (*Stereolithography).* Este formato consiste numa descrição geométrica simples do modelo 3D, pois aproxima as suas superfícies por malhas de triângulos. Quanto maior for o tamanho dos triângulos, menos precisa será a peça obtida. Posteriormente, o ficheiro será transferido para um outro programa chamado *Slicer*. Este programa tem como principal função obter um ficheiro *gcode*, que contém a informação necessária para a impressão da peça em código G. Este código consiste numa linguagem de programação/comunicação normalizada que é utilizada, habitualmente, para máquinas CNC, na qual são abrangidas as impressoras 3D. Antes de ser obtido o ficheiro

gcode, serão definidos no *Slicer* parâmetros que poderão ser ajustados e permitem otimizar a impressão (temperaturas de *set point* da base e do extrusor, espessura de cada camada, velocidade de extrusão, etc), estando estas definições incluídas no ficheiro final (*gcode*). Será exclusivamente através da linguagem de código G que se poderá comunicar com as impressoras 3D comuns, seja para comandar e controlar a máquina (como no caso da impressão de uma peça), seja para obter dados de variáveis que permitem monitorizar o desempenho da máquina (como a temperatura e a posição do extrusor). Na Figura 2.3 observam-se os formatos de informação da peça a produzir. Este último ponto será crucial para o trabalho a desenvolver.

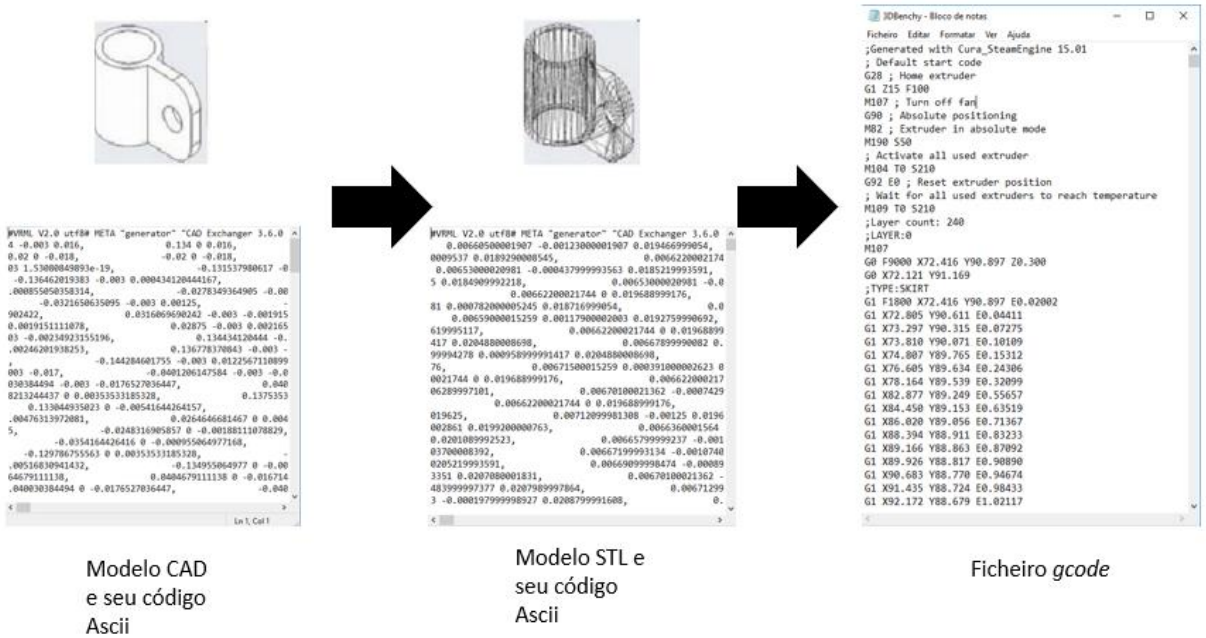


Figura 2.3 - Formatos de informação de uma peça para imprimir em 3D

De seguida, prossegue-se com a transferência do ficheiro *gcode* para a impressora 3D. Nesta etapa, uma de duas alternativas pode ser escolhida:

- O ficheiro será enviado para um cartão SD, onde posteriormente será inserido na impressora e, através de uma interface gráfica presente na máquina, o utilizador selecionará o *gcode* que pretende imprimir;
- O ficheiro é encaminhado para um *software* de controlo designado por *Host* instalado num PC (será chamado de PC *Host* a este conjunto de *hardware* e *software*). A aplicação inserida no PC *Host* consiste num programa que oferece uma interface gráfica interativa que possibilita ao utilizador enviar código G para a impressora 3D. A comunicação entre PC *Host* e impressora 3D é obtida através de uma ligação série assíncrona, sendo a porta que, normalmente, permite a entrada e saída de informação da impressora é uma porta USB-B. Para que seja realizada a impressão no PC *Host*, apenas terá de ser carregado o ficheiro *gcode* e ser premido o botão que dará início à impressão e, assim, enviar as linhas de código G em tempo real. Para além de impressões, do PC *Host* poderão ser enviados comandos de código G manuais através de uma janela disponibilizada pelo programa, que envia apenas a linha de código G específica que permite executar a função pretendida [7].

Do lado da impressora 3D, os comandos de código G enviados terão de ser analisados por um *software* incorporado na máquina, sendo que este código prevê três tipos de instruções: [8, 9]

- Os comandos de controlo, que tem por alvo os sistemas físicos da impressora (motores, aquecedores, ventiladores, atuadores, etc);
- Os comandos de monitorização, que solicitam variáveis que denotam o estado interno da impressora;
- Os comandos de configuração, que definem os parâmetros de construção, como por exemplo, unidade estabelecida para as coordenadas do extrusor ou se o posicionamento do extrusor é absoluto ou relativo ao anterior.

Ao *software* que interpreta e executa o código G, dá-se o nome de *firmware.* Do ponto de vista lógico, a arquitetura de uma impressora poderá ser representada pela Figura 2.4.

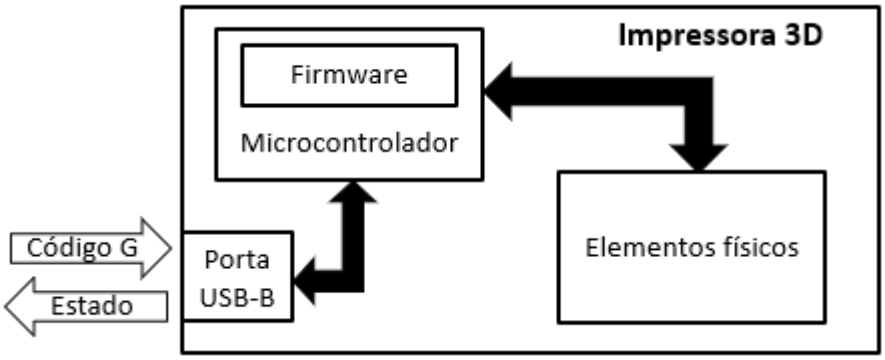


Figura 2.4 - Arquitetura do hardware e software de uma Impressora 3D. Adaptado de [10]

Contextualizado o funcionamento de um processo típico de impressão 3D, será de seguida abordada a razão do desenvolvimento e implementação de uma interface que motivou esta dissertação.

2.2 Definição do problema

A impressão 3D evolui cada vez mais no sentido de se tornar um processo de fabrico com presença assídua na indústria. A esta contínua evolução está subjacente a obtenção de dados e a sua consequente utilização na monitorização da condição e dos processos, visto que poderá contribuir na otimização de uma tecnologia que tem margem para explorar e inovar. Além disso, como foi referido no capítulo 1, estando o INEGI a investir num projeto de reformulação do laboratório de Indústria 4.0, acresce a necessidade de adaptar as impressoras 3D existentes no laboratório aos ambientes de produção modernos e abertos que a Indústria 4.0 proporciona.

Sendo assim, este trabalho irá centrar-se na obtenção dos dados que avaliam o desempenho de uma impressora 3D e enviá-los para uma plataforma IoT, onde será monitorizada a máquina. Uma plataforma IoT é uma tecnologia executada numa *cloud* ou num servidor local e tem como finalidade a comunicação, armazenamento, transformação, processamento e o controlo de dados, bem como a gestão dos dispositivos que os recolhem [11]. No entanto, uma máquina de impressão 3D comum apenas apresenta uma porta série (USB-B) para comunicação a um outro dispositivo, não tendo também capacidade de se conectar a uma rede. Desta forma, a obtenção de dados da impressora e envio dos mesmos para a plataforma IoT será o principal desafio deste projeto.

Após analisado o problema, na próxima secção serão apresentadas duas propostas de solução que permitem resolver o problema acima enunciado. Uma primeira proposta criada recentemente por Li et al. referida em [12] e uma segunda sugerida a criar pelo INEGI.

2.3 Arquitetura física de suporte à solução

Uma das formas de realizar a monitorização de impressoras 3D é proposta por Li et al. Nesse trabalho [12], foi desenvolvida uma aplicação IoT que permite monitorizar impressoras 3D (à semelhança do que será desenvolvido neste projeto) do modelo RepRap, conectando à impressora 3D um microcomputador Raspberry Pi com acesso à Internet. No Raspberry Pi foi instalado o Octoprint (tipo de PC *Host*), que recolhe dados das variáveis que se pretendem monitorizar para, de seguida, serem enviadas para uma plataforma IoT chamada ThingWorx. A ThingWorx, à semelhança de muitas outras plataformas IoT, é uma tecnologia *cloud* que permite a gestão de dispositivos e a recolha, processamento e visualização de dados para projetos IoT. Nesta plataforma, a aplicação IoT proposta por Li et al. é desenvolvida usando um programa IDE (*Integrated Development Environment)*, ThingWorx Composer 7. É através do programa IDE que são modelados os dados recolhidos e desenvolvida a interface gráfica para o utilizador. No entanto, a solução obtida com os desenvolvimentos deste trabalho não será viável, visto estar restringida a impressoras de *firmware* RepRap, plataforma IoT ThingWorx e PC *Host* Octoprint, o que não irá de encontro aos objetivos desta dissertação, que serão enunciados na subsecção 2.4.1.

No contexto da solução sugerida no INEGI, uma interface de comunicação será interposta entre a impressora 3D e o PC *Host.* A interface, neste caso, consiste num sistema embebido (conjunto de *hardware* e *software* com capacidades computacionais) que irá recolher e enviar dados de variáveis da impressora 3D para uma plataforma IoT (à semelhança do trabalho [12]), onde será realizada a monitorização remota da máquina. Para além da monitorização, a interface continuará a permitir uma possível comunicação entre a impressora 3D e o PC *Host*, na hipótese de o utilizador necessitar de enviar comandos manuais para a impressora ou imprimir pelo próprio programa. Será no *software* do dispositivo intermediário (interface) que será programada toda a solução (a obtenção dos dados da impressora, a sua modificação e a comunicação entre todos os elementos do sistema), e na qual deverá ser possível adaptar a qualquer tipo de plataforma IoT, *firmware* de impressoras 3D e PC's *Host.* Portanto, o facto de a interface não necessitar de um PC *Host* para monitorizar uma impressora 3D e de ser adaptável aos diferentes *softwares* e tecnologia (*firmwares,* plataformas IoT, PC's *Host*) permite aferir que esta solução será mais viável do que a que foi proposta em [12].

Assim, um exemplo esquemático das diferenças entre a comunicação de um processo comum de Fabrico Aditivo e a comunicação que se intenciona implementar poderá ser observado na Figura 2.5. A análise à comunicação entre os diferentes elementos da alínea b será apresentada na subsecção 2.5.3. Desta forma, em vez do conceito da impressora 3D como máquina produtiva isolada, esta passará a integrar um sistema que permite a interligação com outros dispositivos e sistemas.

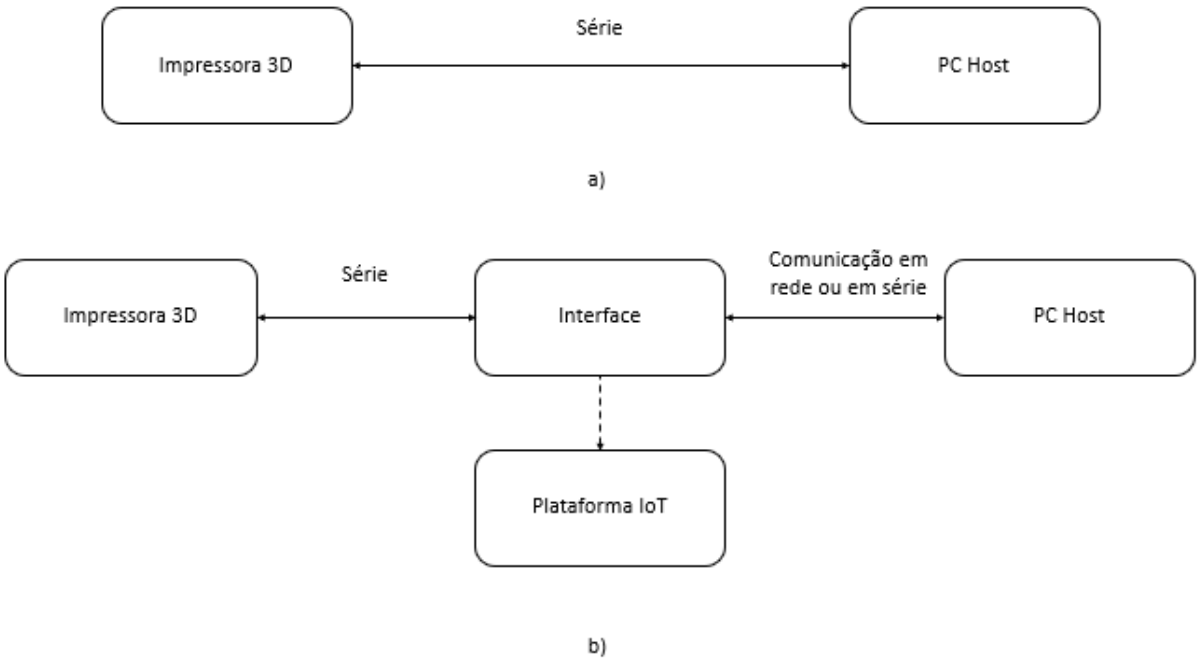


Figura 2.5 - Diferenças de comunicação: a) Comunicação típica de um sistema de impressão 3D; b) Comunicação a desenvolver no decorrer do projeto

A partir da associação de conceitos emergentes da Indústria 4.0 (CPS e IoT) será possível recolher dados relativos às impressoras 3D e, através da comunicação entre os diversos dispositivos e/ou sistemas, encaminhar a informação para uma plataforma IoT destinada ao seu armazenamento e que poderá ser processada pelo utilizador. Este modelo poderá ser inserido, num contexto industrial, a todas as máquinas que integram todo o ambiente de produção. Assim sendo, o que se poderá obter será um ambiente de produção em que todos os processos de uma fábrica são monitorizados e controlados remotamente e estão interligados entre si [13].

2.4 Funcionalidades da solução pretendida e potenciais benefícios

Apresentada a arquitetura física, importa referir as funcionalidades que guiarão a especificação funcional da aplicação IoT pretendida. Igualmente importante é definir desde já e de uma forma objetiva como se poderá retirar vantagens da solução a ser criada, ilustrando para isso a sua aplicação a casos concretos.

2.4.1 Funcionalidades pretendidas

Para obter a solução de monitorização e comando de uma impressora 3D, a aplicação que esta dissertação se propõe a desenvolver deverá exibir as seguintes funcionalidades:

- Permitir a comunicação bidirecional no sistema entre impressora 3D, interface e PC *Host*. No caso da comunicação entre interface e plataforma IoT, apenas existe comunicação no sentido de interface para plataforma, mas que poderá evoluir para comunicação bidirecional;
- Possibilidade de realizar um trabalho de impressão tanto via cartão SD, como via PC *Host*. Neste contexto, apenas a impressão com o cartão SD é considerada impreterível. Imprimir através PC *Host* será opcional, no entanto, qualquer que

seja o resultado, serão disponibilizadas as funções necessárias (no programa a desenvolver) que permitem iniciar essa impressão;

- Implementação obrigatória dos comandos para obtenção automática de dados das variáveis temperatura e posição (M105 e M114, respetivamente). Com o decorrer dos desenvolvimentos, se possível, poderão ser implementados mais comandos.
- A comunicação entre impressora 3D e PC *Host* não ser afetada. Desta forma, comandos manuais provenientes do PC *Host* terão que ser, obrigatoriamente, executados;
- Solução deverá ser possível de adaptar a qualquer tipo de *firmware*, plataforma IoT e PC *Host* de impressoras 3D;
- Após a conclusão dos desenvolvimentos da aplicação, esta terá que ser facilmente portável para um outro *hardware* adequado a um ambiente industrial.

2.4.2 Potenciais benefícios

Os benefícios que poderão ser retirados no imediato com desenvolvimento da aplicação pretendida na interface serão os seguintes:

- **Monitorização remota de uma impressora**. Este será um benefício que resulta diretamente da capacidade de obter dados e enviá-los para a plataforma IoT. Numa primeira instância, serão obtidos os dados da temperatura e da posição e, numa segunda, outro tipo de dados por definir como, por exemplo, o progresso da impressão, tempo restante de impressão ou velocidade atual dos extrusores. Na hipótese de no decorrer do projeto, não se implementarem outros comandos de monitorização, essa tarefa poderá ser realizada num trabalho futuro. Existe, assim, a possibilidade de aceder ao histórico de dados na plataforma IoT e analisá-los ou monitorizar remotamente através do PC.
- **Impressão 3D remota**. Consiste em mais um benefício direto da solução a criar, decorrente da capacidade de estabelecer uma conexão em rede entre a interface e o PC *Host*. Através desta conexão, trabalhos de impressão poderão ser iniciados remotamente, imprimindo diretamente do PC *Host* (terá de ser testada essa possibilidade após o desenvolvimento da solução).

Desta forma, num contexto futuro desta solução poderão emergir potenciais casos de uso proveitosos para a indústria, na qual podem ser enunciados os seguintes:

- **Análise da qualidade do produto**. Obtendo peças com irregularidades, poderão observar-se os dados recolhidos na impressão da peça e verificar se o problema é o desgaste dos componentes da máquina ou do modelo digital criado;
- **Monitorização em tempo-real** que, eventualmente, poderá evoluir para um sistema automático de deteção de erros. No caso de a informação recolhida indicar a instabilidade de uma impressão, ações como a inspeção ou manutenção da máquina poderão ser tomadas;
- **Gestão de uma unidade de produção por Impressão 3D.** A partir de uma plataforma central seriam enviados remotamente ficheiros para impressão de diferentes peças de forma a imprimir em toda uma unidade com várias impressoras, sendo que todos os processos seriam controlados e monitorizados nessa plataforma. Poderia, desta forma, ser integrado nos sistemas típicos

produtivos, como um MES que permite a gestão de processos ou a análise da performance.

2.5 Requisitos gerais da interface a desenvolver

Para responder ao problema enunciado e tendo em conta os objetivos a atingir, é necessário estabelecer os requisitos gerais subjacentes ao desenvolvimento da interface. Estes requisitos são aqui abordados de modo global e superficial, deixando para o próximo capítulo questões que justificarão escolhas tecnológicas concretas em torno dos mesmos.

2.5.1 Elementos de *hardware*

Sendo que a interface funcionará como um sistema embebido, esta irá conter elementos de *hardware* e *software*. Com o *hardware* pretende-se um elemento que possua capacidades de computação que permita executar o *software* onde será o desenvolvida a solução. Para além disso, o *hardware* deverá ser capaz de se conectar à rede, conter, pelo menos, uma porta para comunicação série assíncrona, compacta (para que seja facilmente conectada a diferentes impressoras 3D) e barata. De realçar que o *hardware* a utilizar durante o projeto terá como finalidade única o desenvolvimento da aplicação e não será destinado à aplicação final, em ambiente industrial.

2.5.2 Elementos de *software*

Elemento já referido no subcapítulo 2.5.1, será neste onde toda a programação, que permite cumprir os objetivos do projeto, reside. Existe uma diversa lista de opções onde se incluem diversas ferramentas com a finalidade de programação para um variado tipo de aplicações. O que é expectável, sobretudo, como característica fundamental da ferramenta escolhida será a sua portabilidade. Ou seja, após a conclusão dos trabalhos da presente dissertação, esta terá que ser capaz de integrar facilmente outro *hardware* adequado a ambiente industrial.

2.5.3 Elementos de comunicação

Para além de definir os elementos enunciados em 2.5.1 e 2.5.2, será necessário também definir os protocolos que irão permitir que a interface comunique tanto com as impressoras 3D, como com o PC Host e a plataforma IoT. Assim sendo, terão de ser estabelecidas três tipos de comunicação para a criação do sistema pretendido:

- Comunicação entre impressora 3D e interface;
- Comunicação entre interface e aplicação de PC Host;
- Comunicação entre interface e plataforma IoT;

No caso dos dois primeiros tipos de comunicação, através da análise da Figura 2.5 é possível observar que estes já foram estabelecidos. No primeiro ponto, a comunicação terá de ser impreterivelmente por uma ligação série, visto ser esta a única forma da grande maioria das impressoras 3D estabelecer comunicação com um dispositivo externo. No segundo ponto, a escolha deste tipo de comunicação está pré-determinada para série ou em rede, na qual é preferível a comunicação em rede, caso esta seja suportada pela aplicação do PC *Host*, devido ao facto de ser possível estabelecer uma conexão remota. Por fim, abordando o terceiro ponto, esta comunicação será estabelecida com um protocolo de comunicação adequado a soluções

IoT. Tendo em conta que existem diversos protocolos com aplicação na IoT, a escolha recairá para um que permita a comunicação com a generalidade das plataformas IoT.

2.6 Resumo do capítulo

Este capítulo permitiu contextualizar o processo de Impressão 3D e introduzir o problema que se centra, sobretudo, na incapacidade de as impressoras estabelecerem comunicação com uma plataforma IoT, onde será realizada a monitorização do equipamento, sendo incapazes de integrar os novos ambientes de produção orientados com as abordagens da Indústria 4.0.

Para solucionar de uma forma viável o problema, será introduzido um dispositivo (interface) a intermediar a comunicação entre impressora 3D e o seu programa de controlo no PC *Host*. Este dispositivo, ligado à Internet, deverá conseguir comunicar com uma plataforma IoT e enviar dados da impressora que permitem o seu controlo e monitorização. Esta capacidade de comunicação será desenvolvida numa ferramenta de programação instalada na interface. Tanto esta ferramenta, como o *hardware* da interface e a comunicação entre todos os elementos do sistema (impressora 3D, interface, PC Host e plataforma IoT) serão especificados tecnologicamente no próximo capítulo.

O objetivo principal será que a máquina de impressão funcione sem que o dispositivo implementado interfira na produção de uma peça, ao mesmo tempo em que serão recolhidos dados e enviados para a plataforma IoT.

3 Requisitos de Engenharia

No presente capítulo, serão abordados os requisitos de engenharia ou tecnológicos que permitem cumprir os objetivos e requisitos enunciados previamente e enquadrar a solução que será criada. Para esse efeito, foi efetuado um levantamento de conteúdos das tecnologias a serem apresentadas. Deverá salientar-se que as tecnologias selecionadas não são as únicas que permitem solucionar o problema, sendo que a razão dessa seleção será também abordada nas próximas secções.

Este capítulo será dividido em cinco tópicos distintos: Impressão 3D, Plataforma IoT, *Hardware* da interface, Node-Red e protocolos de comunicação. No tópico de Impressão 3D, serão abordadas diretamente as principais escolhas de aplicações de PC Host e *firmwares* disponíveis para este processo. Posteriormente, serão referidos os condicionalismos inerentes à utilização da plataforma IoT. De seguida, os elementos tecnológicos (sugeridos pelo INEGI) que compõem o *hardware* e a *stack* de *software* da interface (Raspberry Pi e Node-Red, respetivamente) serão descritos e justificados. Por fim, serão definidos os protocolos que garantem a comunicação entre todos os elementos do sistema.

3.1 Impressão 3D

Como foi referido anteriormente, a expiração de patentes das diferentes tecnologias de Impressão 3D, principalmente do FDM em 2009, permitiu que novos fabricantes (*Ultimaker* e *MakerBot*) e projetos (*RepRap Project* e *Fab@Home*) surgissem, aumentando, consequentemente, a popularidade e competitividade deste processo de fabrico [14, 15]. Neste contexto, poderão ser encontrados atualmente uma grande diversidade, não só de fabricantes de impressoras 3D, como também de produtores de *software* e componentes eletrónicos referentes a esta tecnologia. Nesta secção, serão apresentadas as principais aplicações para PC *Host* e de *firmware* de impressoras 3D disponíveis para o processo de Impressão.

3.1.1 Aplicações para PC *Host*

Conforme mencionado no capítulo 2, no processo de Impressão 3D, a máquina permite estabelecer comunicação apenas com um dispositivo externo que, tipicamente, será o PC *Host*. Para este projeto, essa comunicação estará intermediada pela interface a implementar (Figura 2.5 alínea b). A aplicação do PC *Host* (que poderá também incluir a funcionalidade de um *Slicer*) permite controlar todos os aspetos da impressora e dos trabalhos de impressão, como obter *feedbacks* constantes do progresso de uma impressão, monitorizar, ajustar as temperaturas para o extrusor ou base e parar ou pausar uma impressão. A Tabela 3.1 apresenta as mais populares aplicações de PC *Host* usadas num processo de Impressão 3D, na qual é indicada a sua função e algumas observações.

Tabela 3.1 - Aplicações de PC *Host* [16]

Nome	**Função**	**Observações**
Cura [17]	*Host, Slicer*	Desenvolvido pela *Ultimaker* (fabricante de impressoras 3D), mas que opera com grande parte das impressoras existentes. É um *software open-source Slicer*, podendo também funcionar como *Host*, com uma interface intuitiva e fácil de usar. Uma das melhores escolhas para principiantes, sendo que tem um modo para utilizadores mais experientes que permite um maior controlo sobre a configuração de uma impressão.
Simplify3D [18]	*Host, Slicer*	Considerado um *software premium* que ajuda a melhorar bastante a qualidade das impressões, sendo indicado para utilizadores mais experientes. Incorpora funcionalidades não existentes na maioria das aplicações de PC *Host* como reparação de ficheiros STL e capacidade de inserir e modificar o ficheiro de impressão (*gcode*). Para além disso, a sua execução é muito rápida, porém não é gratuito, tendo um custo de cerca de 133€.
Repetier-Host [19]	*Host, Slicer*	Um dos *softwares* mais antigos e também um dos favoritos entre a comunidade RepRap (mas compatível com a maioria das impressoras 3D comercializadas), é *open-source*, abrange desde os utilizadores intermédios até aos mais experientes. É rotulado como uma solução tudo-em-um, oferecendo suporte para vários extrusores (até 16) e permite a conexão por TCP/IP, caso esta seja suportada pela impressora.
Octoprint [20]	*Host, Slicer*	*Software open-source* que inclui uma versão que permite a instalação num Raspberry Pi. O microcomputador conectado à impressora por comunicação série possibilita o controlo remoto, através da interface *web* do Octoprint, num PC. Aceita ficheiros *gcode* de qualquer *Slicer* e possibilita a visualização do código G antes e durante a impressão. Poderão ser expandidas as funcionalidades deste *software* com a ativação de *plug-ins.*

3.1.2 *Firmwares* de impressoras 3D

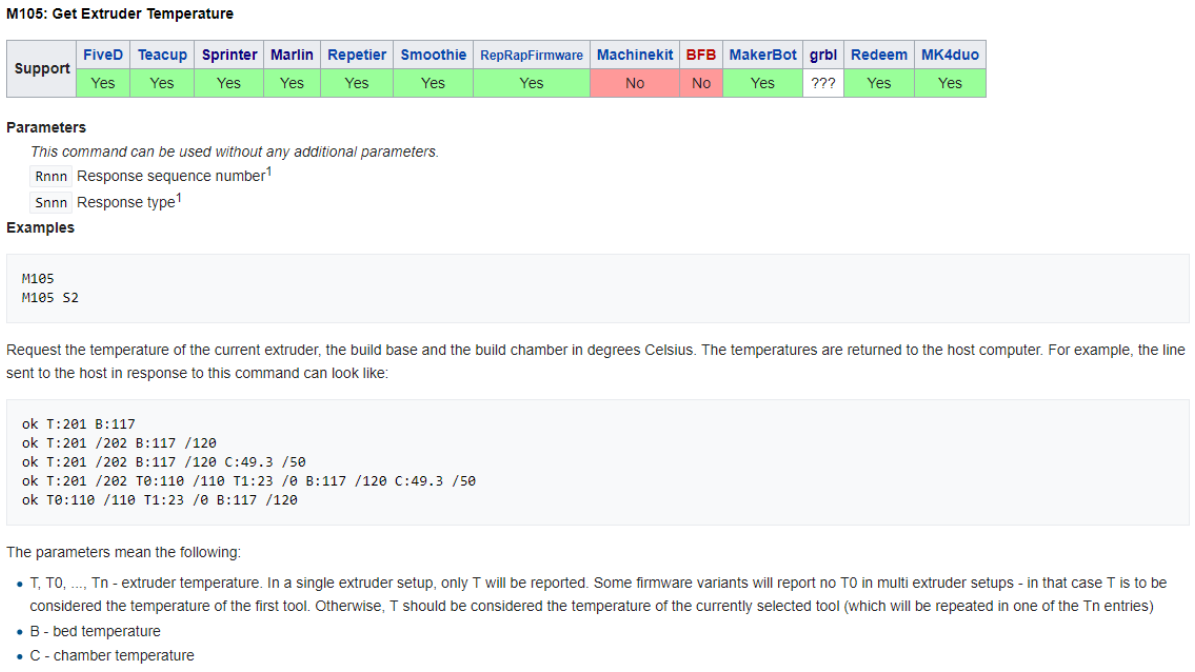
À semelhança das aplicações de PC Host, é possível abranger um certo leque de escolhas relativamente a um *firmware*. A Tabela 3.2 aborda alguns dos mais populares *firmwares* e observações sobre as suas características, bem como o *hardware* que é suportado. Cada um incorpora um pequeno conjunto de componentes eletrónicos, onde está incluído o amplamente usado RepRap Arduino Mega Pololu Shield (RAMPS). Todos os *firmwares* indicados a seguir são *open-source*.

Tabela 3.2 - Firmware para Impressoras 3D [21, 22]

Nome	**Hardware**	**Observações**
Sprinter [23]	RAMPS, Ultimaker, Sanguiololu, Gen6.	Oferece suporte para base aquecida e cartão SD, para além de oferecer aceleração constante ou exponencial que permita uma impressão rápida e controlada. Serve da base a outros *firmwares* como Marlin ou Repetier.
Marlin [24]	RAMPS, Ultimaker, Sanguinololu, Gen6 e semelhantes.	Baseado no *firmware* Sprinter, oferece suporte para múltiplos extrusores, bem como para painéis LCD, nivelamento automático da base, cartão SD e poderá ser adaptado para extrusão por mistura de cores. É usado por uma vasta gama de impressoras 3D
Repetier [25]	RAMPS, AzteegX3, Gen6, Sanguinololu, Gen7, Printrboard, RAMBo, e muitos mais.	*Firmware* reescrito do Sprinter (cerca de 80%), com aprimoramento de velocidade. Suporte para multi-extrusor, *dry-run* (permite detetar falhas não relacionadas com o extrusor) cartão SD e painéis LCD. É possível também extrudir por mistura de cores se devidamente adaptado.
Teacup [26]	RAMPS, Ultimaker, Sanguinololu, Gen6 e semelhantes.	Uma versão inicial de *firmware* para impressão 3D. Número arbitrário de aquecedores e sensores de temperatura e suporte multi-extrusor.
RepRap [27]	RAMPS, Duet, Alligator Board, RADDS	Um dos primeiros a suportar placas de 32-bits, foi planeado para ser um programa de controlo orientado ao objeto altamente modular, que permite uma grande variedade de movimentos na máquina. Permite comunicação através de uma interface *web* e reporta o estado de toda a impressora através de um único comando de código G, o M408.

Desta forma, para escolher um dado *firmware* de uma impressora, deve-se ter em conta características funcionais e componentes de *hardware*, contidos na impressora, compatíveis com cada *fimrware*.

Por vezes, certo código G poderá ser interpretado de maneira distinta (conteúdo das mensagens enviadas nem sempre se dispõe do mesmo modo) por diferentes *firmwares* ou não ser suportado de todo. Dando um exemplo concreto, o comando M85 é suportado pelos *firmwares* Sprinter, Marlin e Repetier, mas não pelo Teacup ou o RepRap. A disposição das variáveis numa mensagem poderá também ter influência na forma de programar a solução que se pretende criar para cada *firmware* ou até mesmo em versões diferentes de um mesmo *firmware*. Na referência [28] é apresentada uma lista extensa do código G e suas funcionalidades para cada *firmware* de impressoras 3D, que valida o que agora foi abordado neste parágrafo. À lista poderão ser adicionados mais códigos ou outras funcionalidades para um mesmo código G ao longo do tempo. Na Figura 3.1 é apresentado um excerto desta lista onde se visualiza o comando M105, que será implementado neste projeto.



M105: Get Extruder Temperature

Support	FiveD	Teacup	Sprinter	Marlin	Repetier	Smoothie	RepRapFirmware	Machinekit	BFB	MakerBot	grbl	Redeem	MK4duo
	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	Yes	???	Yes	Yes

Parameters

This command can be used without any additional parameters.

`Rnnn` Response sequence number[1]

`Snnn` Response type[1]

Examples

```
M105
M105 S2
```

Request the temperature of the current extruder, the build base and the build chamber in degrees Celsius. The temperatures are returned to the host computer. For example, the line sent to the host in response to this command can look like:

```
ok T:201 B:117
ok T:201 /202 B:117 /120
ok T:201 /202 B:117 /120 C:49.3 /50
ok T:201 /202 T0:110 /110 T1:23 /0 B:117 /120 C:49.3 /50
ok T0:110 /110 T1:23 /0 B:117 /120
```

The parameters mean the following:

- T, T0, ..., Tn - extruder temperature. In a single extruder setup, only T will be reported. Some firmware variants will report no T0 in multi extruder setups - in that case T is to be considered the temperature of the first tool. Otherwise, T should be considered the temperature of the currently selected tool (which will be repeated in one of the Tn entries)
- B - bed temperature
- C - chamber temperature

Figura 3.1 - Código G M105: Funcionalidades e *firmwares* compatíveis [28]

3.2 Plataforma IoT

A plataforma IoT utilizada pelo INEGI que possibilita a realização de testes para a solução a desenvolver é a ThingsBoard [29], sendo que esta plataforma abrange outros dispositivos de trabalhos paralelos a este que decorrem no INEGI. Para aceder às funcionalidades da ThingsBoard deverá escolher-se um dos produtos de duas edições distintas: Uma edição para a comunidade em geral, livre de custos e a utilizada pelo INEGI; Uma edição profissional onde terá de ser subscrito um plano de pagamento entre cinco modalidades existentes, em que o preço varia com o número de dispositivos que poderão comunicar com a plataforma.

Como referido previamente, uma plataforma IoT permite a recolha e visualização de dados, monitorizar e gerir as entidades IoT, processa os dados e estabelece relações em função desses dados recolhidos. Todos os dados enviados poderão ser armazenados numa base de dados fornecida por um sistema de gestão de dados *open-source* e NoSQL (*Non Structured Query Language*) denominado Cassandra (a plataforma também permite o suporte de bases de dados SQL (*Structured Query Language*)), podem ser acedidos através da interface gráfica da ThingsBoard ou por uma REST (*Representational State Transfer*) API (*Application Programming Interface*) e podem ser encaminhados por outros protocolos de comunicação (a serem apresentados ainda nesta secção) para outros *softwares*. As plataformas IoT são, normalmente, executadas na *cloud*, no entanto, a plataforma ThingsBoard também permite ser executada no servidor local, sendo que neste caso, é no servidor do INEGI que corre a plataforma. A Figura 3.2 mostra a disposição das diferentes componentes da interface gráfica da ThingsBoard.

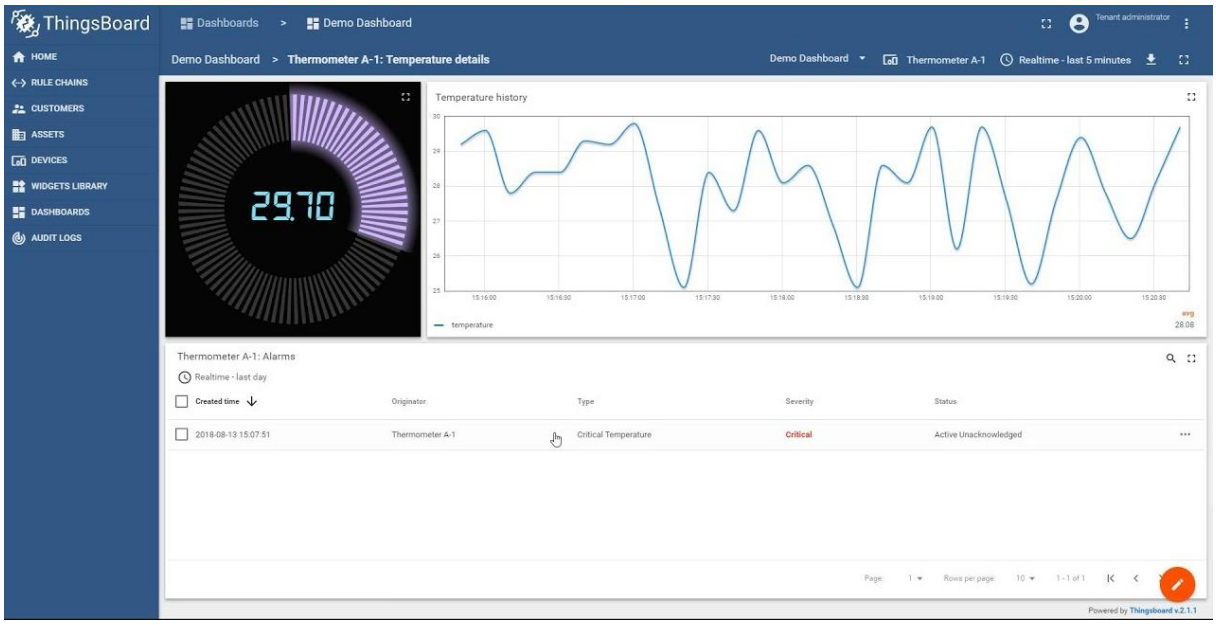


Figura 3.2 - Interface gráfica da ThingsBoard

Será do lado da ThingsBoard que os dados recolhidos da impressora (inicialmente da temperatura e da posição) serão armazenados e analisados, onde uma ação manual (e potencialmente automática) poderá ser tomada em função dessa análise.

Para que os dados sejam enviados para a plataforma, estes deverão estar dispostos num formato atributo-valor chamado JSON (*JavaScript Object Notation*). O formato JSON é um formato composto por duas estruturas [30]:

- Uma coleção de pares atributos-valores;
- Uma lista ordenada de valores, que pode ser traduzida com um *array* ou vetor.

Um exemplo de uma disposição de dados em formato JSON é apresentado na Figura 3.3:

```
{
"name":"John",
"age":30,
"cars":[ "Ford", "BMW", "Fiat" ]
}
```

Figura 3.3 - Exemplo de disposição de dados em formato JSON

Outras plataformas IoT tipicamente apresentam os mesmos requisitos no que toca ao formato dos dados enviados, podendo haver outras possibilidades, como o XML (*Extensible Markup Language*) ou o HTML (*HyperText Markup Language*). No entanto, o ponto comum entre este tipo de plataformas consiste nas mensagens serem comunicadas em formato JSON, visto ser mais *lightweight* comparado com o XML ou HTML [31, 32].

Para além do formato em JSON, a plataforma ThingsBoard requer que a conexão com um dispositivo seja realizada por um de três protocolos de comunicação: O MQTT (*Message Query Telemetry Transport*), o HTTP (*Hyper Text Transport Protocol*) ou o CoAP (*Constrained Application Protocol*). Para este projeto foi sugerido o uso do protocolo MQTT, que será abordado em 3.5.1. Este protocolo de comunicação, tal como o JSON para o formato

dos dados, é abrangido por várias plataformas IoT, dando por isso uma boa resposta para a adaptabilidade que se pretende da solução a desenvolver.

3.3 Hardware de suporte à interface

Como elemento de *hardware*, foi adotado para os desenvolvimentos da interface deste trabalho um Raspberry Pi, fazendo parte da sugestão do INEGI. O Raspberry Pi é uma placa SBC (*Single Board Computer*) ou microcomputador do tamanho de um cartão de crédito desenvolvido pela fundação Raspberry Pi. Com dimensões externas de 85 x 56 mm, esta placa SBC reúne todos os componentes necessários para ser usada como um PC comum [33]. O modelo do Raspberry Pi utilizado nos desenvolvimentos deste trabalho foi o 3B +, representado na Figura 3.4. Este dispositivo apresenta um processador 64-bit Broadcom BC 2837B0 integrado com gráficos Cortex A-53 (ARMv8). A memória RAM neste modelo é de 1GB e um cartão MicroSD é usado como disco rígido, sendo inserido numa ranhura adequada à sua leitura no dispositivo. O uso de um cartão MicroSD como disco rígido permite uma adaptação da capacidade do cartão às necessidades de cada utilizador e é neste onde será instalado o Sistema Operativo, sendo para projeto o Raspbian GNU/Linux 9. Adicionalmente, a este microcomputador é fornecida uma tensão de alimentação de 5V, através de um cabo MicroUSB e apresenta 4 portas USB 2.0, porta Ethernet e outros componentes não relevantes para este projeto, como portas para conectar uma câmara específica deste microcomputador e um visor *touchscreen*. A Tabela 3.3 apresenta uma lista com os parâmetros, alguns já mencionados, do modelo 3B+ do Raspberry Pi. De uma forma complementar, na Figura 3.4 é mostrada a organização e posicionamento de todos os componentes enunciados anteriormente.

Tabela 3.3 - Parâmetros do modelo 3B+ do Raspberry Pi [34]

Parâmetros	**Descrição**
Processador	Cortex A-53 1.4 GHz
Número de Núcleos	4 (*Quad Core*)
RAM	1GB
Memória	Cartão MicroSD
GPIO	40 pinos
Portas USB	4*2.0
Porta HDMI	Sim
Comunicação sem fios	*Ethernet, Wi-Fi, Bluetooth*
Sistema Operacional	Raspbian GNU/Linux 9

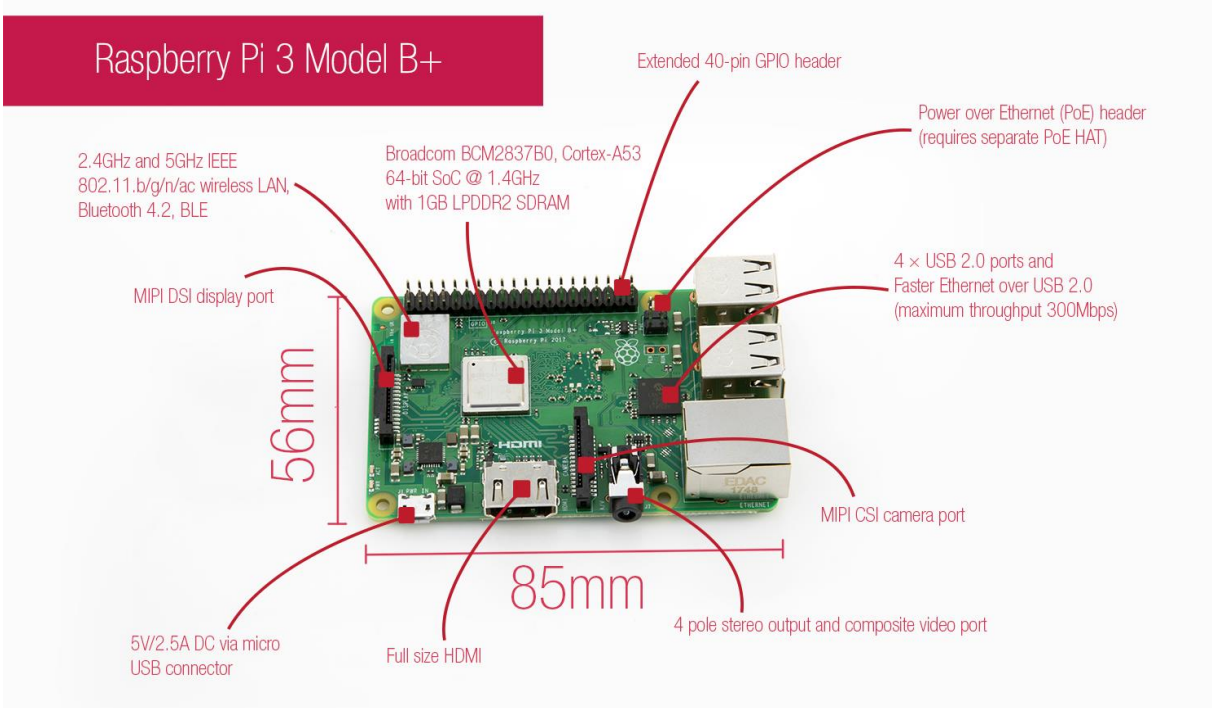


Figura 3.4 - Raspberry Pi 3B+ [35]

Desta forma, o Raspberry Pi é adequado para este projeto, pois permite estabelecer uma comunicação série (contém 4 portas USB) com uma impressora, conectar-se a uma rede (entrada para cabo *Ethernet* e possibilidade de conexão *Wi-Fi*) e possui capacidades computacionais e um Sistema Operativo que o possibilita suportar a ferramenta de programação escolhida, a ser abordada na próxima secção. Para além disso, o baixo consumo energético, custo reduzido (varia entre 30 e 35€) e o tamanho (compacto) deste microcomputador são outros fatores que contribuíram para a sua escolha [36]. Este *hardware* servirá como sistema de teste laboratorial que permite ser replicado numa solução industrial.

3.4 *Software-Stack*

A ferramenta de programação definida para a hipótese proposta, que permite os desenvolvimentos do *software* neste trabalho, foi o Node-Red. O Node-Red é uma ferramenta desenvolvida pela IBM que disponibiliza um ambiente gráfico para a programação principalmente de aplicações IoT, fazendo uso de um modelo baseado em fluxo. A abordagem dos modelos de programação baseado em fluxo visa as aplicações não como um processo sequencial e único, que começa a dado ponto e realiza uma ação de cada vez até que o processo termine, mas uma rede de processos assíncronos que comunicam recorrendo a fluxos de blocos de dados estruturados, chamados *"Information Packets"* (IPs). Estes IPs para o Node-Red denominam-se nós. Nesse ponto de vista, o foco reside na aplicação dos dados e nas transformações aplicadas a esses dados para produzir o *output* desejado [37].

O Node-Red é implementado sobre o *runtime lightweight* do Node.js, permitindo retirar vantagens do seu modelo orientado a eventos, sem bloqueios (tarefas poderão ser executadas sem que as anteriores estejam completas). Um sistema *runtime* é um componente de *software* responsável pela gestão da execução dinâmica de um programa e dos recursos concedidos pelo sistema operativo [38]. Um sistema *runtime* atua como um intermediário entre a aplicação desenvolvida e o sistema operativo inserido no *hardware*. Assim, o código desenvolvido no *software* Node-Red interage com o sistema operativo da Raspberry Pi por

meio do *runtime* do Node.js que converte o código num processo que será encaminhado para o sistema operativo (Raspbian), que o executa e envia comandos para os elementos de *hardware*. A *stack* de *software* que engloba o Node-Red está esquematizada na Figura 3.5.

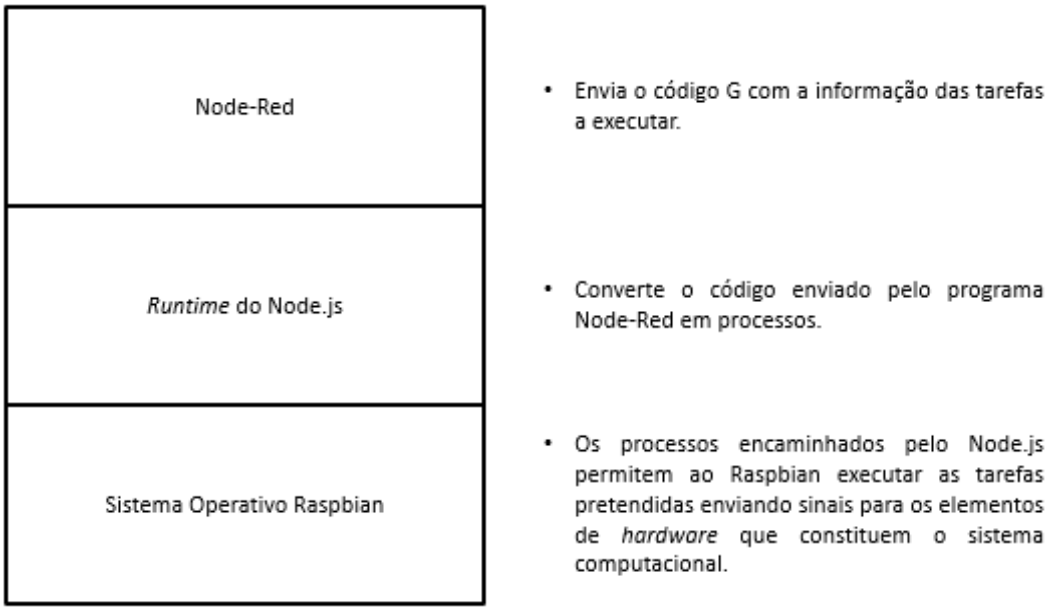


Figura 3.5 - *Stack* de *software* que envolve o programa Node-Red

A partir da *software stack* mostrada na Figura 3.5 (Node-Red, Node.js e Raspbian) será programada toda a solução que se pretende desenvolver. Assim sendo, o que se pretende programar baseia-se nos seguintes pontos:

- Programação das mensagens com instruções em código G a enviar para a impressora, com a inclusão de um número de linha e de um *checksum*, no início e final da mensagem, respetivamente;
- Gestão das mensagens provenientes da impressora para permitir comunicação simultânea entre a interface e o PC *Host* com a impressora 3D. Deverá ser identificado a origem do pedido, que corresponde uma determinada resposta da impressora, e encaminhar essa resposta para o *output* adequado;
- Conversão das mensagens destinadas à plataforma IoT para o formato JSON. Nesse sentido, será transformada inicialmente a mensagem para uma disposição dos dados em *array* que permite uma adaptação posterior a qualquer tipo de formato específico. De seguida, a mensagem é convertida para JSON, com a possibilidade em aberto de num trabalho ser realizada a conversão da mensagem para um outro tipo de formato de disposição de dados.

A Figura 3.6 esquematiza o essencial do que é pretendido com a implementação da interface.

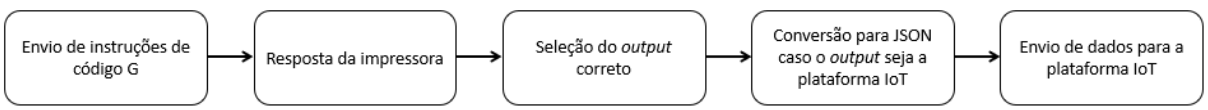


Figura 3.6 - Funcionamento essencial pretendido com a implementação da interface

Desta forma, o facto do *software* Node-Red estar assente numa *stack* composta também pelo Node.js e Raspbian permite uma eventual mudança de sistema operativo, visto ser o *runtime* do Node.js que transcreve o código enviado pelo Node-Red a um diferente tipo

de processos, adaptando-se às necessidades de cada sistema operativo. Assim, em futuros desenvolvimentos da aplicação, esta poderá ser executada num *hardware* com um sistema operativo diferente desde que suporte o Node.js. Dado que o Node.js é suportado por diversos sistemas operativos, poderá validar-se a condição de portabilidade para um sistema industrial mencionada em 2.5.3.

3.5 Protocolos de comunicação

Nesta secção, serão abordados os protocolos de comunicação utilizados na conexão dos diferentes elementos do sistema a criar. Será explicado e caracterizado cada protocolo abordado e, adicionalmente, a justificação do protocolo utilizado na comunicação entre interface e plataforma IoT.

3.5.1 MQTT

Para a comunicação entre interface e plataforma IoT, foi utilizado o protocolo MQTT, também imposto para a solução a desenvolver.

O MQTT é um protocolo de comunicação de nível 7 do modelo OSI desenvolvido pela IBM de forma a ser utilizado em sistemas IoT e que está assente no protocolo TCP (*Transmission Control Protocol*) /IP (*Internet Protocol*). O protocolo funciona num modelo de *publish/subscribe* formado pelos publicadores, *brokers* e subscritores. Os publicadores são os elementos que enviam e partilham a informação, enquanto que os subscritores apenas recebem essa mesma informação. O *broker* consiste num dispositivo central, que funciona como servidor, responsável por gerir a comunicação entre subscritores e publicadores, sendo que recolhe as mensagens provenientes dos publicadores e examina para quem é que a mensagem terá de ser encaminhada [39, 40].

O modo de funcionamento do protocolo é simples. Os publicadores publicam as mensagens para um *broker* MQTT, na qual é acedida pelos subscritores, podendo ser retida para futuros acessos a essa mensagem. Cada mensagem é enviada para um endereço, conhecido como tópico [41]. Os subscritores podem aceder a múltiplos tópicos e receber todas as mensagens publicadas para cada tópico.

Para cada tipo de mensagem existe uma funcionalidade integrada. Dos diferentes tipos de mensagens possíveis de trocar, podem-se destacar as seguintes:

- **Connect:** O cliente requisita uma conexão ao servidor. Será, de seguida, enviada uma resposta do servidor através de uma mensagem Connack a informar sobre o resultado da conexão;
- **Publish:** Este tipo de mensagens são sempre associadas a um tópico, onde são enviadas de um cliente para o servidor que depois distribui essa informação para outro tipo de clientes que subscrevem esse tópico. O servidor responde com um Puback;
- **Subscribe:** De forma a subscrever aos diferentes tópicos que o cliente pretende aceder, é enviada uma mensagem Subscribe para o servidor. O servidor responde com um Suback;
- **Disconnect:** Mensagem que desconecta o cliente do servidor.

Na Figura 3.7 apresenta-se um exemplo de um diagrama de sequência de troca de mensagens típico no protocolo MQTT, que ilustra o que foi, desde já, referido.

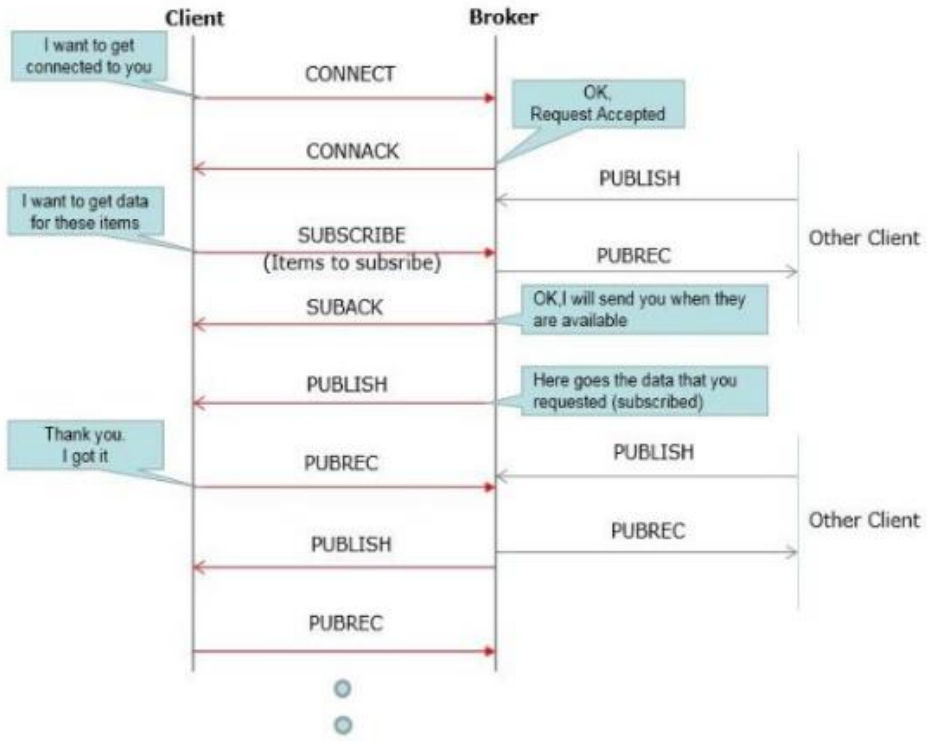


Figura 3.7 - Exemplo de um diagrama de sequência da troca de mensagens no protocolo MQTT [42]

A conexão dos clientes MQTT ao *broker*, tanto pelo publicador como pelo subscritor é realizada pelo protocolo TCP. No processo de conexão é estabelecido o QoS (*Quality of Service*), que define a fiabilidade de entrega da mensagem. Estão previstos três níveis de QoS, sendo estes [43]:

- **QoS 0 (*at most once*):** O que possui menor esforço, não havendo confirmações da entrega da mensagem. Não está definido uma retransmissão futura;
- **QoS 1 (*at least once*):** Neste nível já existe confirmação de entrega da mensagem. Porém, as mensagens são retransmitidas até que a sua chegada seja reconhecida, tendo como consequência que várias mensagens serão geradas;
- **QoS 2 (*exactly once*):** É garantido que a mensagem é entregue exatamente uma vez, com o envio de confirmação da entrega.

A Figura 3.8 esquematiza o que foi agora explicado sobre os três níveis de QoS.

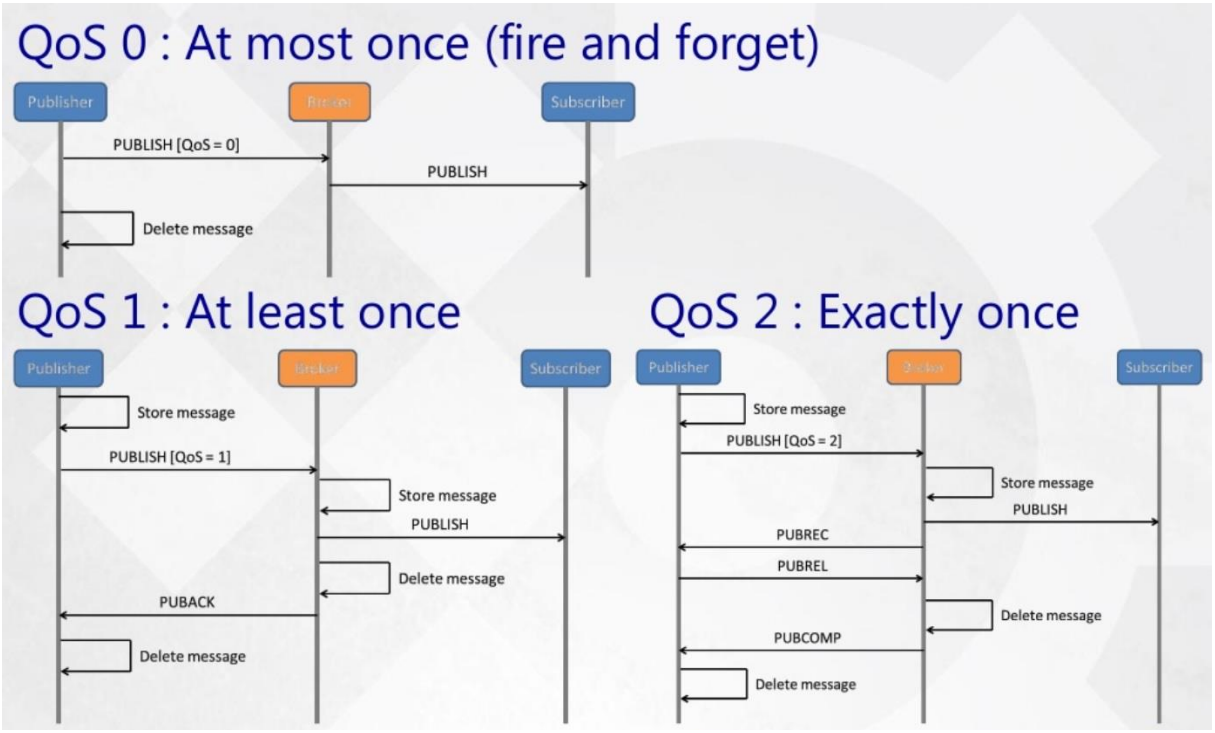


Figura 3.8 - Níveis de QoS

Para além do QoS, poderão ser definidos outros parâmetros como o *Retain* e *Last will message*. O *Retain* (definido como *true* ou *false*) possibilita ao servidor MQTT reter a última mensagem e o QoS correspondente para esse tópico. Normalmente, se um cliente MQTT subscreve um tópico num *broker*, não recebe as mensagens publicadas antes da subscrição. No caso do cliente definir o valor de *Retain* como *True* o *broker* irá reter a última mensagem emitida para esse tópico [44].

O *Last will message* consiste numa mensagem que é publicada quando o cliente desconecta do servidor inesperadamente, que indica o encerramento não espectável da conexão. Cada cliente pode especificar a sua *Last will message* quando se conectam ao *broker*. O *broker* guarda essa mensagem até que deteta que o cliente se desconecta acidentalmente. Em resposta a essa desconexão, o *broker* envia a *Last will message* para todos os clientes subscritos ao tópico dessa mensagem.

Poderá ainda ser definido outro parâmetro, que se trata do formato dos tópicos do protocolo MQTT. Como referido anteriormente, os tópicos deste protocolo são a forma de endereço que permite aos clientes MQTT partilhar informação. Nesse sentido, os tópicos MQTT estão estruturados numa hierarquia de pastas e ficheiros, semelhante a um sistema de ficheiros de um PC, usando a barra "/" como um delimitador entre os diferentes níveis. Esta estruturação é mostrada na Figura 3.9.

Figura 3.9 - Formato dos tópicos no protocolo MQTT [45]

De maneira a justificar a escolha deste protocolo, será abordado um trabalho realizado por Nitin Naik referenciado em [41]. Neste trabalho foram avaliados 4 dos principais protocolos de comunicação usados em aplicações IoT: O MQTT, o CoAP, AMQP (*Advanced Message Queuing Protocol*) e HTTP. Essa avaliação foi baseada em 5 parâmetros, nomeadamente, tamanho de mensagem e *overhead*; potência consumida (pelo dispositivo); largura de banda e latência; segurança; fiabilidade (na entrega da mensagem). Os resultados desta análise mostram que o MQTT é o protocolo que melhor combina todos os parâmetros, excetuando o da segurança (onde foi o protocolo que piores resultados apresentou), na qual nos parâmetros *overhead*, largura de banda e potência, foi o segundo protocolo que melhor resultados apresentou e o melhor protocolo referente ao parâmetro da fiabilidade. Estes resultados justificam assim uma estatística também indicada em [41], que mostram que o MQTT é o protocolo mais utilizado para aplicações IoT pelas empresas.

Assim, pela análise destes aspetos vantajosos do MQTT, é possível inferir que este protocolo de comunicação constitui uma boa solução para aplicações IIoT, onde pode ser abrangido este projeto, caracterizadas pela baixa potência disponível, capacidade de computação e memória limitada e largura de banda reduzida. Para além destas vantagens, a possibilidade de comunicação de 1 para N e o modelo *publish/subscribe* do MQTT faz com que este protocolo seja ideal também para sistemas de monitorização [13, 46].

3.5.2 TCP/IP

Protocolo de comunicação que engloba os níveis 3 (protocolo IP) e 4 (protocolo TCP), sobre o qual assentam muitos outros como o MQTT, o HTTP, o AMQP, etc. No caso deste trabalho em particular, poderá ser estabelecida uma ligação TCP/IP para conectar PC *Host* e interface.

O TCP/IP consiste num conjunto de protocolos de comunicação de suporte à transmissão de dados entre dois sistemas computacionais em rede. À semelhança do modelo OSI, a arquitetura protocolar TCP/IP é também representada por camadas, sendo que o modelo OSI é constituído por sete camadas e o TCP/IP, originalmente, por quatro camadas, mas que atualmente serão cinco camadas (camada física, camada de interface com a rede, camada da Internet, camada de transporte e camada de aplicação). Cada camada está incumbida de fornecer um conjunto de serviços bem definidos que passarão para a camada de nível superior, até chegar à camada mais alta que contacta diretamente com o utilizador (como o caso do MQTT ou HTTP), excetuando a camada física que apenas se refere ao *hardware* do dispositivo que pretende comunicar e, por isso mesmo, não participa no estabelecimento de comunicação.

Para as restantes camadas são usados quatro níveis de endereços por sistemas que aplicam o protocolo TCP/IP. Cada endereço está relacionado com uma camada, como é mostrado na Figura 3.10. Estes podem ser explicados, resumidamente, da seguinte forma [47]:

- **Endereço físico:** Aplicado Também conhecido como link, é o endereço de um nó conforme definido pelo seu LAN ou WAN;
- **Endereço IP:** Define de forma única um anfitrião que navega na Internet;
- **Endereço da porta:** Identifica o processo num anfitrião;
- **Endereço da aplicação:** Usado em algumas aplicações para fornecer ao utilizador uma interface simples.

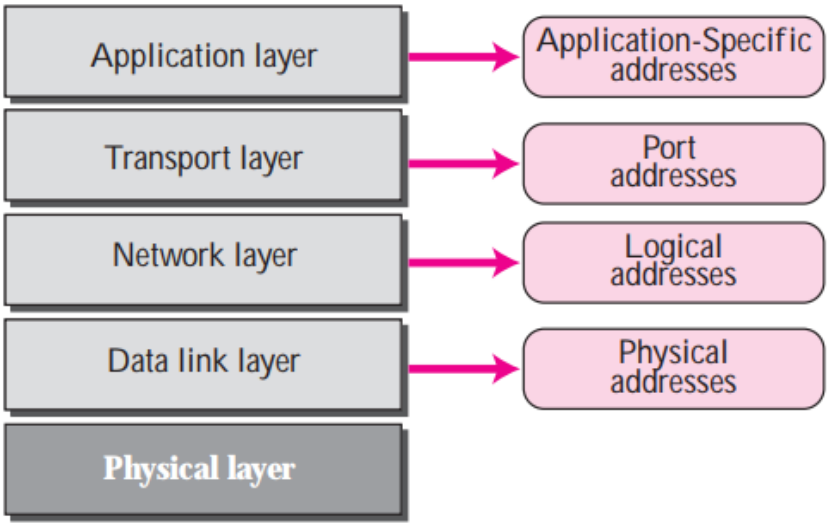


Figura 3.10 - Endereços no protocolo TCP/IP [47]

A conexão que será estabelecida entre interface e PC *Host* será através de um *socket* TCP/IP. Um *socket* TCP/IP é definido como a combinação de:

- Um endereço IP;
- Número de porta do protocolo de transporte (TCP).

Desta forma, será utilizada uma arquitetura cliente/servidor. Na aplicação do cliente é criado o *socket*, onde será especificado o endereço IP da aplicação servidor e a porta onde será realizada a comunicação. Na aplicação do servidor é também criado um *socket*, no entanto o servidor apenas necessita de associar o *socket* à porta que será realizada a comunicação e depois escolher a opção que permite escutar novas conexões de clientes.

3.5.3 Série assíncrona

Nesta secção é abordada o tipo de comunicação que será estabelecido entre a interface e a impressora 3D, que será uma comunicação série assíncrona.

A comunicação série constitui-se como o processo de envio de dados um bit de cada vez, de forma sequencial, através de um canal de comunicação. No caso do trabalho desenvolvido, a comunicação série usada foi assíncrona. Neste contexto, cada caracter (ou *byte*) é transmitido sob a forma de um sinal digital contendo as seguintes características apresentadas na Figura 3.11 [48]:

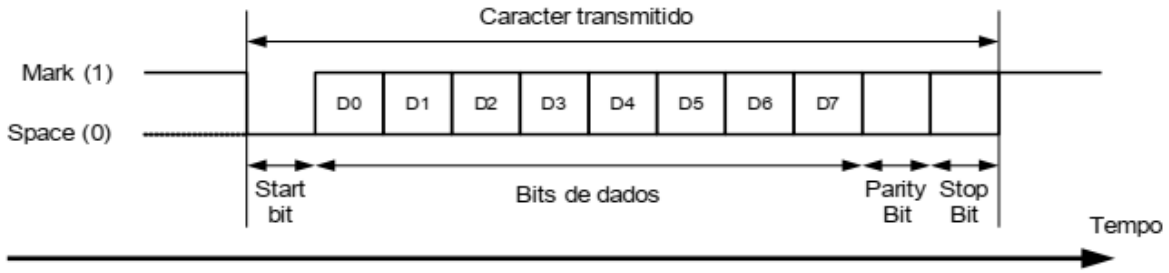


Figura 3.11 - Transmissão série assíncrona [48]

- *Start bit* (0 lógico): usado para assinalar a transmissão do caracter;

- *Bits* de dados (D0 a D7): Pode ser configurado de 5 a 9 *bits*, consoante o código utilizado.
- *Parity*: Deteta potenciais erros na transmissão. Pode ser configurado das seguintes formas:
    - Even: o número de 1's (lógicos) transmitidos (não contando com o *start bit*) é sempre par;
    - Odd: o número de 1's (lógicos) transmitidos (não contando com o *start bit*) é sempre ímpar;
    - Mark: é tomado o valor de 1 lógico;
    - Space: é tomado o valor de 0 lógico;
    - None: inexistente.
- *Stop bit*: usado para assinalar o fim da transmissão e assegurar o espaçamento entre caracteres. Toma os valores de 1, 1.5 ou 2 bits.

Outra característica que deve ser enunciada neste protocolo corresponde à taxa de transmissão (*Baud Rate*). A *Baud Rate* é definida pela velocidade de comunicação, medida pelo número de bits transferidos por segundo. Caso o recetor não esteja com este valor ajustado, a informação não é lida corretamente.

3.6 Resumo do capítulo

O capítulo abordou e descreveu os requisitos tecnológicos necessários à conceção da solução enunciada em 2.3 e a ser desenvolvida no capítulo 4. Numa primeira fase foi realizado um levantamento de conteúdos referentes a cada tecnologia em artigos, livros e páginas de documentação técnica. De seguida, procedeu-se à especificação dessas mesmas tecnologias. No caso da secção da Impressão 3D, foi apresentada uma listagem com as aplicações de PC *Host* e *firmwares* mais populares para este processo. De seguida, foi apresentada uma explicação sobre o que é uma plataforma IoT e que condicionalismos coloca no projeto. Posteriormente, foi especificado tecnicamente a Raspberry Pi como *hardware* da interface. Por fim, foi explicitado o funcionamento do *software-stack* que permite programar a solução e dos protocolos de comunicação usados para conectar os elementos do sistema a criar.

4 Desenvolvimento de *software*

Este capítulo tem como propósito a apresentação das soluções desenvolvidas no contexto desta dissertação, que permitem cumprir os objetivos e requisitos enunciados anteriormente. A aplicação será adaptada a dois tipos diferentes de *firmware*: o Marlin e o Repetier, sendo construída de forma a ser facilmente adaptada aos restantes *firmwares*.

No que toca ao desenvolvimento da aplicação para o Marlin, serão abordados e explicados todos os passos que permitiram obter a solução para este *firmware*. Para o Repetier a abordagem será distinta. Será estabelecido como ponto de partida os desenvolvimentos efetuados para o Marlin e, de seguida, serão adaptadas apenas as funções que são necessárias para obter no Repetier um funcionamento semelhante à solução desenvolvida para *firmware* Marlin.

Durante todo o desenvolvimento de *software* para esta dissertação, a aplicação de PC *Host* usada para testes foi o Repetier-Host. Outra aplicação deste tipo poderia ser usada, nomeadamente as que se encontram na Tabela 3.1, porém o Repetier-Host para além de ser livre de custos, suporta a conexão por TCP/IP com um outro dispositivo, dando resposta à comunicação que se pretende implementar entre interface e PC *Host*.

Sendo assim, a sequência lógica que pretende traduzir o funcionamento da aplicação a desenvolver é representada pela Figura 4.1.

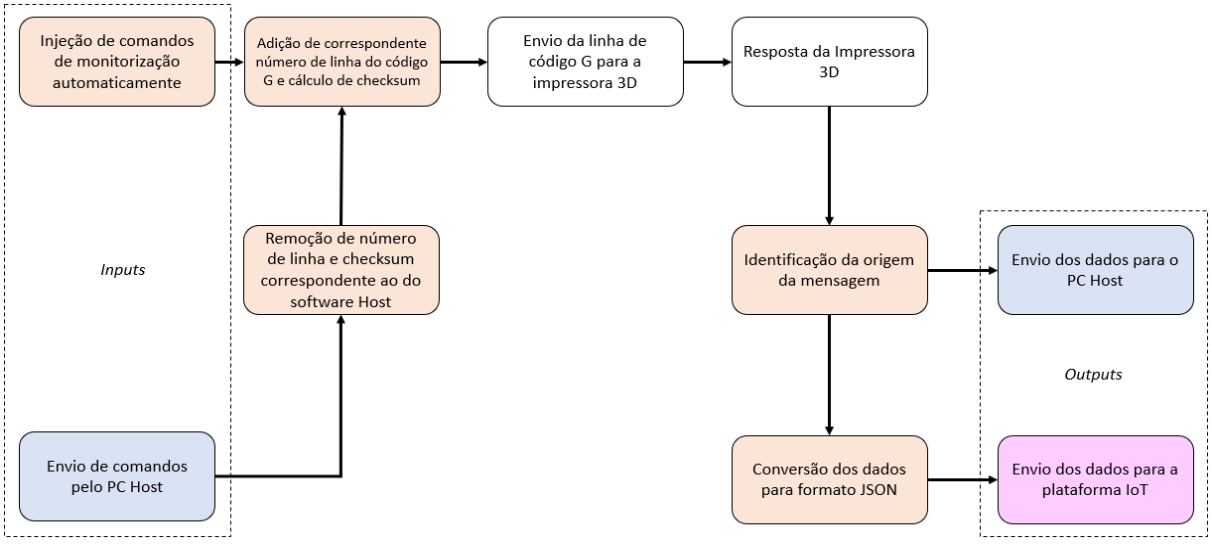


Figura 4.1 - Esquema que traduz o funcionamento da aplicação a desenvolver

Analisando a figura, é possível destacar diferentes cores nos blocos, que se justifica pela mais fácil distinção dos elementos do sistema que se pretende criar (Figura 2.5 b), sendo que as setas correspondem à transmissão de mensagens. Desta forma, as diferentes cores correspondem aos seguintes elementos do sistema:

- **Cor laranja:** Interface;
- **Cor azul:** PC *Host;*
- **Cor branca:** Impressora 3D;
- **Cor rosa:** Plataforma IoT.

Antes de ser apresentado o desenvolvimento da solução, será contextualizado o funcionamento da ferramenta de programação Node-Red para melhor entendimento da aplicação desenvolvida.

4.1 Node-Red

O Node-Red é uma ferramenta de programação visual *open-source* que utiliza um modelo baseado em fluxo para criar aplicações IoT. Este modelo de programação baseado em fluxo usa nós para representar ações. Cada nó tem um certo propósito e cada um atua sobre os dados o atravessam. Os dados ou são gerados por nós ou são recebidos de um nó anterior. Cada nó transforma os dados de acordo com a sua especificação e, de seguida, transmite para o próximo nó ou para fora do fluxo [49].

4.1.1 Nós

Os nós podem ter inputs e/ou outputs. Estes são independentes e a sua execução não afeta o comportamento de outros nós, sendo a modularidade a característica chave destes [50]. Portanto, cada nó tem a sua própria função separada que é intercambiável com os outros nós.

Pode, portanto, subdividir-se os nós em três tipos: entrada, processamento e saída.

Nós de entrada (*inputs*): Responsáveis pela entrada de dados no fluxo. Desta forma, todos os nós que estabelecem a entrada de dados a partir de fontes externas são considerados *inputs*. Um exemplo deste tipo de nós poderá ser o "TCP in", que permite a entrada de dados através de uma conexão TCP/IP.

Nós de processamento (*functions*): Os nós que alteram e manipulam os dados recebidos nos *inputs*. Após o processamento os dados são disponibilizados para uma determinada saída.

Nós de saída (*outputs*): Responsáveis pelo reencaminhamento dos dados para uma determinada saída no fluxo. Estes nós podem ser exemplificados através do nó "Serial out", na qual possibilita a saída de informação por uma porta série.

Tanto para os *inputs* como para os *outputs*, tipicamente é requerida uma configuração específica à fonte externa (Por exemplo, associar um *broker* e um tópico num nó MQTT).

No caso dos nós de processamento, é possível destacar um chamado "function" que permite ao próprio utilizador desenvolver código em Javascript. Quando é recebido por este nó uma mensagem, é executado o código na mensagem e é encaminhada para um, nenhum ou múltiplos *outputs*. Foi principalmente a partir deste nó que foi desenvolvida a parte crucial da solução que se pretende implementar, na medida em que é possível programar de acordo com as necessidades de uma certa aplicação. Para a aplicação que se pretende desenvolver, não existem nós que permitam implementar todas as funcionalidades requeridas, sendo que nesses casos será usado o nó "function" para desenvolver essas mesmas funcionalidades.

Para além dos nós já existentes e incorporados na instalação do próprio *software*, o Node-Red permite aceder a uma biblioteca que contém um conjunto grande de nós

desenvolvidos por uma extensa comunidade *open-source*, na qual permite introduzir facilmente novas funcionalidades, como por exemplo, nós de OPC-UA (*Open Platform Communications - Unified Architecture*) [51].

4.1.2 Ambiente gráfico

A Figura 4.2 mostra o ambiente IDE do Node-Red, onde se poderão identificar 5 áreas diferentes.

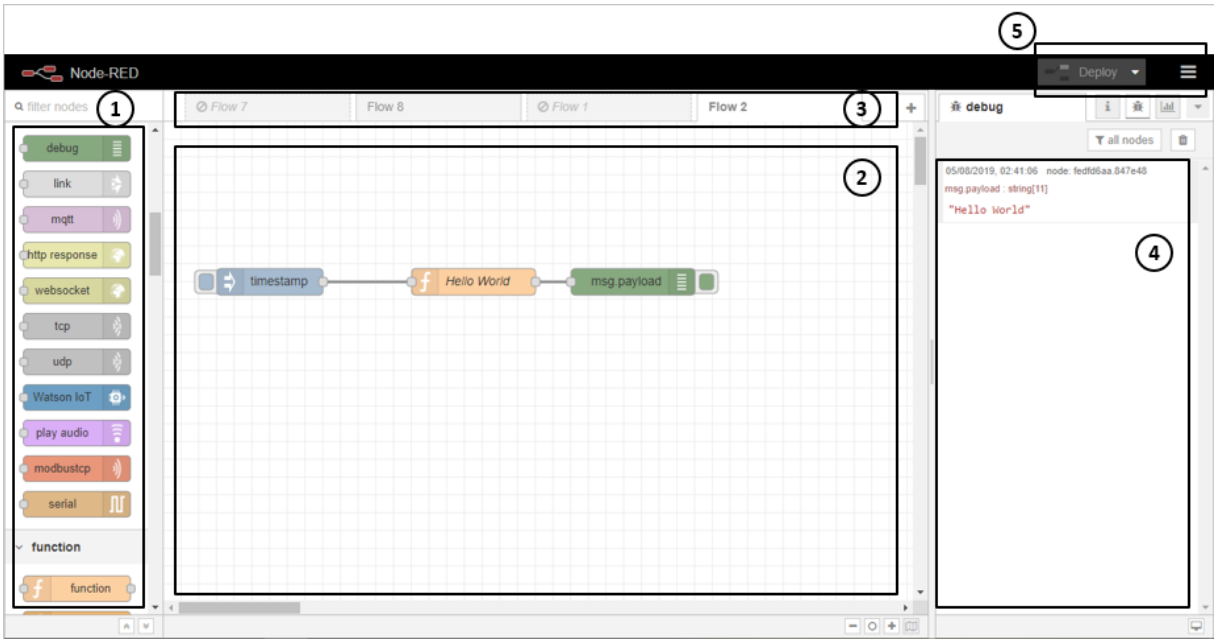


Figura 4.2 Ambiente gráfico do Node-Red

A partir da análise da Figura 4.2, é possível destacar cinco componentes distintas:

1. Painel onde estão situados os diferentes nós que podem ser usados para desenvolver a aplicação. Poderão ser englobados mais nós acedendo a uma biblioteca na qual existe uma grande variedade de nós criados pela comunidade *open-source*;
2. Fluxo onde os nós poderão ser arrastados e configurados de forma a criar uma aplicação. No exemplo estão representados três nós que permitem injetar uma mensagem, alterar o seu conteúdo para uma *string* "Hello World" e exibir no painel lateral do IDE identificado com o número 4;
3. Área que permite a seleção e gestão dos diferentes fluxos criados;
4. Painel lateral do IDE onde é exibido o conteúdo das mensagens ("msg.payload") ou, opcionalmente, mais informação associada a esse conteúdo, como tópicos ("msg");
5. Espaço que inclui o botão "Deploy" que efetiva os nós alterados ou criados no fluxo e o menu do programa Node-Red, onde se poderá aceder à biblioteca de nós, definições, entre outros.

Como já foi falado, a programação no Node-Red baseia-se no trânsito de fluxo de dados entre nós. É necessário, impreterivelmente, um nó input e um output. De uma forma geral, os três tipos de nós são utilizados para desenvolver uma aplicação. A sua conexão permite uma sequência de funcionamento que trabalha de forma cooperativa de modo a obter

um determinado resultado, originando um fluxo. Os dados seguem a estrutura desenhada pelo fluxo, onde percorre cada ponto e aí ocorre as determinadas ações. Neste último exemplo da Figura 4.2 podemos constatar um fluxo constituído pelos três grupos de nós mencionados em que o *input* injeta uma mensagem, o nó "function" converte o conteúdo da mensagem em uma *string* "Hello World" e o *output* que exibe a mensagem painel lateral do Node-Red.

Os próprios fluxos podem ser representados textualmente através do formato JSON, possibilitando ser facilmente importados e exportados do programa. Este aspeto permite também responder à fácil portabilidade da aplicação requerida para este projeto.

O Node-Red providencia, assim, um tipo de lógica mais intuitiva para facilitar a compreensão de não-programadores, e permite aos utilizadores passar rapidamente entre planeamento e implementação, reduzindo o tempo de desenvolvimento [51].

4.2 Desenvolvimentos para *firmware* Marlin

A solução começou por ser desenvolvida para o *firmware* Marlin. Para possibilitar estes desenvolvimentos a solução foi testada em duas impressoras distintas com este mesmo *firmware*. Uma impressora da marca Monoprice com a versão 1.0.9 deste *firmware* e uma outra impressora também da Monoprice com a versão 1.0.0. Nestes casos de diferentes versões de *firmware*, as únicas alterações nas mensagens poderão ser a nível de organização dos diferentes elementos, não diferindo no conteúdo quando for pedida uma mesma mensagem para as diferentes versões. Para este caso serão explicados todos os passos que culminaram criação da solução para este *firmware*, sendo que cada subsecção está estruturada com a amostragem da imagem do fluxo final desse passo, seguido da análise do que foi implementado.

Desta forma, os desenvolvimentos para *firmware* Marlin estarão divididos nos seguintes passos:

1. Implementação de comunicação série;
2. Comunicação com a plataforma IoT;
3. Introdução de comunicação com o PC *Host* e gestão de mensagens;
4. Implementação de número de linha e *checksum*;
5. Substituição de número de linha e *checksum* de pedidos do PC *Host;*
6. Implementação de comandos manuais.

4.2.1 Implementação de comunicação série

Como ponto de partida, será estabelecida a comunicação série com uma impressora. Para esse efeito, terão de ser configurados os nós mostrados na Figura 4.3.

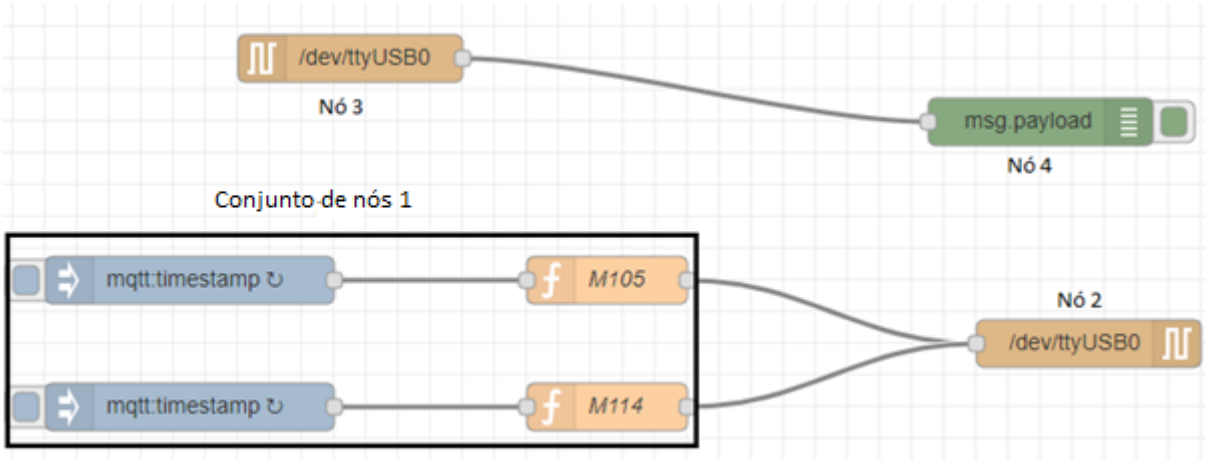


Figura 4.3 - Desenvolvimentos iniciais da aplicação IoT no *software* Node-Red

Assim, podemos definir os nós representados da seguinte forma:

Conjunto de nós 1: Conjunto de *inputs* e funções que correspondem à injeção de uma mensagem. Cada conjunto ("inject" + "function") representa a injeção de um *timestamp* e a modificação do seu conteúdo para uma instrução de código G que, no caso do projeto em questão, permitem obter dados da temperatura (M105) e da posição (M114). A injeção de mensagens será automática e realizada a cada segundo. Este conjunto de nós permite simular mensagens que poderão potencialmente ser enviadas no futuro pela plataforma IoT.

Nó 2: Nó que se caracteriza pela saída de informação da interface pela porta série USB (a interface disponibiliza 4 portas USB 2.0). Poderão ser definidos neste nó os parâmetros que configuram uma ligação série. É possível visualizar um tipo de configuração da ligação estabelecida entre a interface e a impressora 3D através da Figura 4.4.

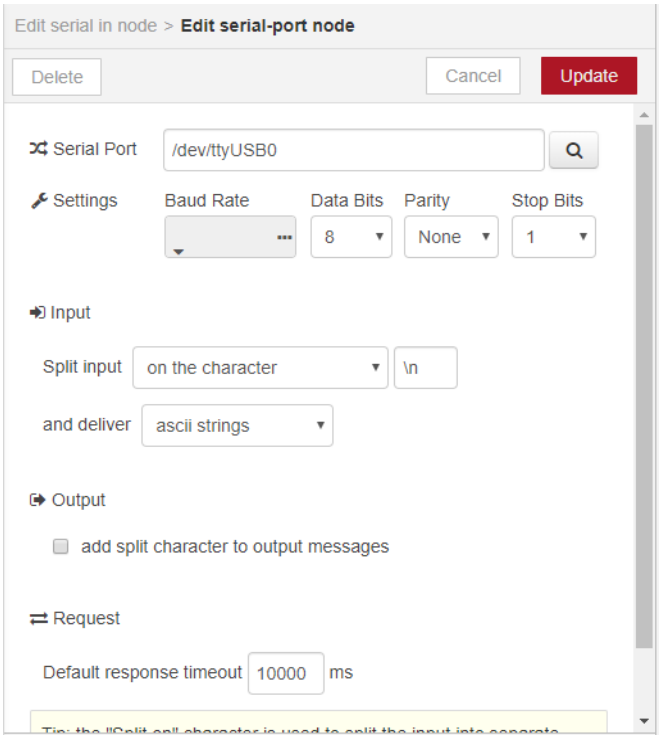


Figura 4.4 - Configuração de uma ligação série

A configuração da comunicação que é estabelecida, nomeadamente do valor da *Baud Rate*, poderá ser diferente de impressora para impressora.

Nó 3: Representa a resposta enviada pela impressora. Tal como para o nó 2, parâmetros típicos de uma ligação série como a *Baud Rate*, *data bits*, paridade e *stop bits* e a porta série poderão ser definidos.

Nó 4: Após o envio de uma mensagem (Conjunto de nós 1 e Nó 2), é esperada uma resposta da Impressora, que é lida através do Nó 3, na qual permite a este nó exibir essa mesma resposta no painel lateral do IDE do Node-Red. Tal como foi enunciado previamente, para além do conteúdo da mensagem poderá ser exibido outros conteúdos, como o tópico de uma mensagem. Com a implementação da comunicação série foi capaz de se testar a natureza de dois tipos de mensagens diferentes: Uma mensagem correspondente à temperatura do extrusor e da base (“M105”) e uma outra correspondente ao acompanhamento da posição do extrusor (“M114”). Porém, antes de se fazer esse teste, aquando da iniciação do sistema é divulgado um conjunto de mensagens pré-denominadas “echo” que relevam informações sobre as configurações iniciais da impressora. Na Figura 4.5 podem-se observar algumas mensagens deste tipo.

Figura 4.5 - Mensagens "echo" de configuração da Impressora

Após reveladas essas mensagens, testou-se a injeção das duas mensagens enunciadas anteriormente, sendo os resultados dessa injeção apresentados na Figura 4.6 e na Figura 4.7.

Figura 4.6 - Resposta ao envio da mensagem "M105”

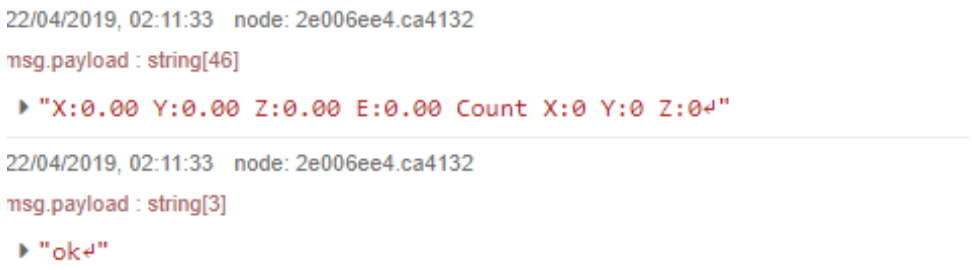


Figura 4.7 - Resposta ao envio da mensagem "M114"

Como é percetível na Figura 4.4, a ligação série está configurada para que as mensagens sejam recebidas no formato *String*. Como referido anteriormente, a plataforma IoT utilizada para este projeto, a ThingsBoard, está sujeita à imposição (enunciada em 3.2) do formato dos dados encaminhados ser em JSON. Desta forma, o formato da *string* recebida pela impressora terá que ser alterado para JSON de forma a que os dados possam ser encaminhados para a plataforma. Este passo é bastante importante e será especificado na próxima secção.

4.2.2 Comunicação com a Plataforma IoT

Para esta etapa serão adicionados os nós que permitem a comunicação com a plataforma IoT ThingsBoard.

Como referido anteriormente, terão que ser efetuadas alterações no que toca à disposição dos dados na *string* para o formato JSON. Para esse efeito, são criadas duas funções (Conjunto de nós 5) que alteram a disposição dos dados de forma a que seja possível transformá-los do formato *string*, para um *array* com os diferentes valores da mensagem (funções "trans M105" e "trans M114"). De seguida, é criada um outro nó "function" denominado "new json" que converte esse *array* num objeto JSON. Por fim, é adicionado um nó *output* MQTT (nó 7) que permite a comunicação com a plataforma IoT através do protocolo MQTT. A Figura 4.8 mostra o fluxo com a adição destes nós.

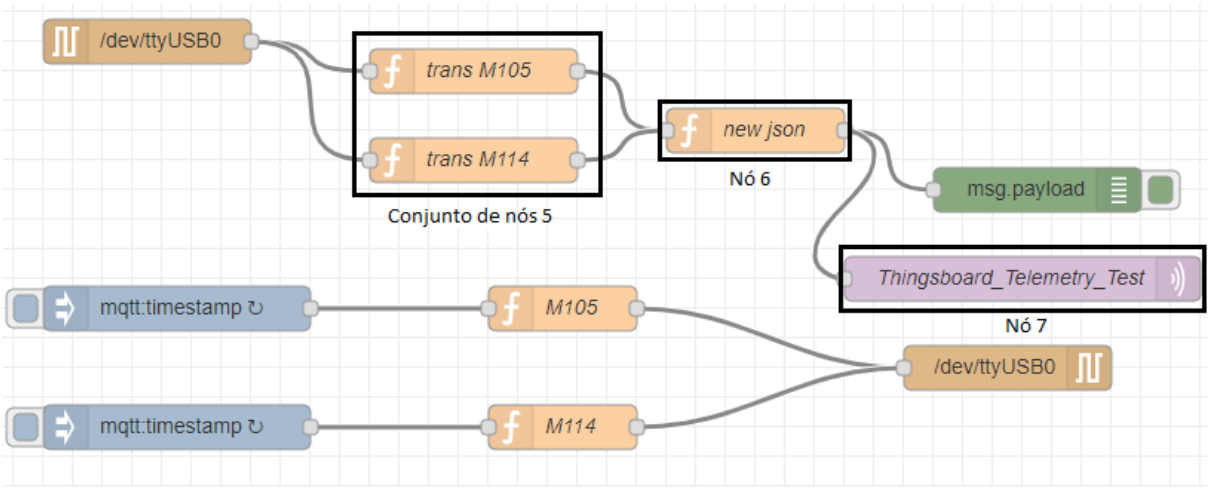


Figura 4.8 - Fluxo com funcionalidades que permitem comunicação com a plataforma IoT adicionadas

A criação separada do conjunto de nós 5 e do nó 6 deve-se, sobretudo, à facilidade de uma futura conversão para um outro formato que não o JSON, referido inicialmente em 3.4. Para essa futura conversão poderá manter-se o conjunto de nós 5 e substituir o nó 6 por um outro, adequado ao novo formato.

Assim, na Figura 4.9 e na Figura 4.10 é mostrado o resultado da conversão de dois comandos (M105 e M114) no formato JSON.

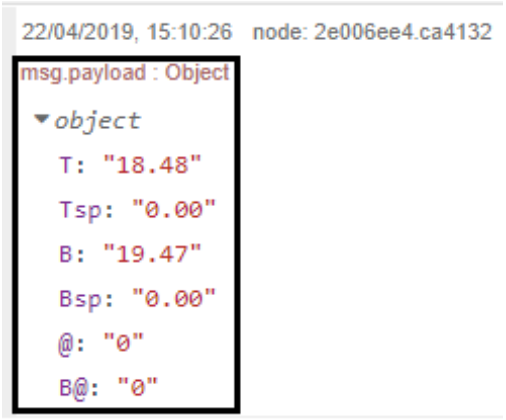


Figura 4.9 - Resposta da Impressora ao comando "M105" convertida em objeto JSON com as seguintes variáveis: T - Temperatura do extrusor; Tsp - Temperatura de *set point* do extrusor; B - Temperatura da base de impressão; Bsp - Temperatura de *set point* da base de impressão; @ - Coeficiente da conversão analógico-digital do sensor de temperatura do extrusor; B@ - Coeficiente da conversão analógico-digital do sensor de temperatura da base de impressão

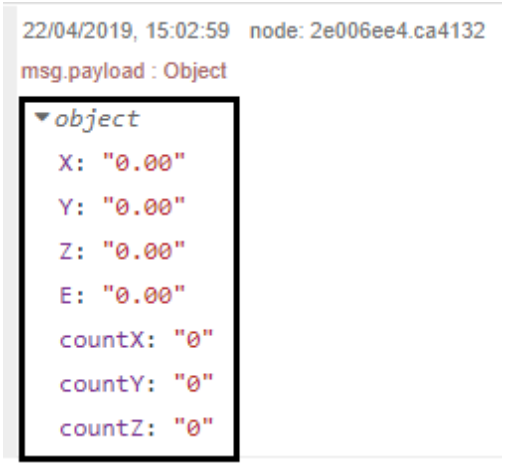


Figura 4.10 - Resposta da Impressora ao comando "M114" convertida em objeto JSON com as seguintes variáveis: X - Coordenada X do extrusor; Y - Coordenada Y do extrusor; Z - Coordenada Z do extrusor; CountX, CountY e CountZ - Coordenadas X, Y e Z, respetivamente, do motor passo-a-passo

Em relação ao nó 7, que permite a comunicação por MQTT para a plataforma IoT, poderão ser definidos os parâmetros que à semelhança dos nós 2 e 3 (comunicação série) que configuram a conexão MQTT. Na Figura 4.11 é apresentada uma imagem da configuração do nó de MQTT.

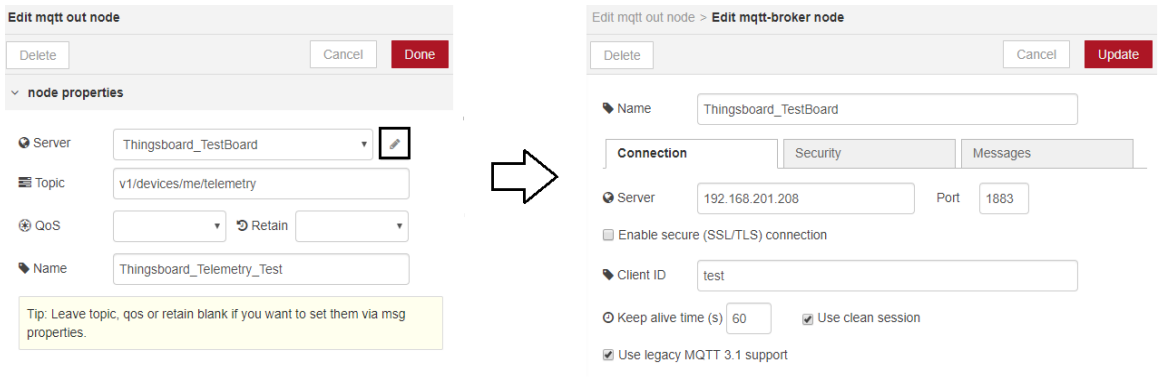


Figura 4.11 - Configuração da ligação MQTT

É possível observar que será na imagem da esquerda na janela com o nome "*Server*", onde se pode escolher o servidor para o qual serão publicados os dados de monitorização da Impressora. Na imagem mais à direita está representada a configuração desse mesmo servidor, onde se deverá preencher na janela de nome "*Connection*" o IP do servidor, a porta ao qual se irá conectar e, na janela "*Security*", as credenciais de acesso (nome do utilizador e palavra-passe se necessário) que definem o dispositivo do lado da plataforma ThingsBoard. Esta última informação sobre a janela "*Security"* é fornecida pelo INEGI.

4.2.3 Introdução de comunicação com PC Host e gestão de mensagens

Tal como enunciado previamente, uma das funcionalidades que a interface deverá dispor consiste em não afetar a comunicação entre a impressora e o PC *Host*. Sendo assim, nesta subsecção serão adicionados os nós que permitem estabelecer a comunicação por TCP/IP entre impressora e PC *Host* e também o encaminhamento das mensagens para os nós de processamento ou de *output* adequados.

Como ponto de partida nesta etapa serão adicionados dois nós: um *input* TCP e um *output* TCP, de forma a implementar a comunicação entre a interface e o PC *Host*. De seguida, tendo em conta que na etapa anterior não existe gestão de mensagens (o que impede a comunicação simultânea da interface e do PC *Host* com a impressora), serão adicionados os nós que permitem fazer a distinção das mensagens destinadas para o PC *Host* ou para a plataforma IoT, e também identificar para que função do conjunto de nós 5 será encaminhada uma mensagem destinada à plataforma IoT. Adicionalmente, foram adicionados outros nós que permitem auxiliar o encaminhamento correto das mensagens. Na Figura 4.12 é observável o fluxo com as alterações promovidas em relação à anterior etapa.

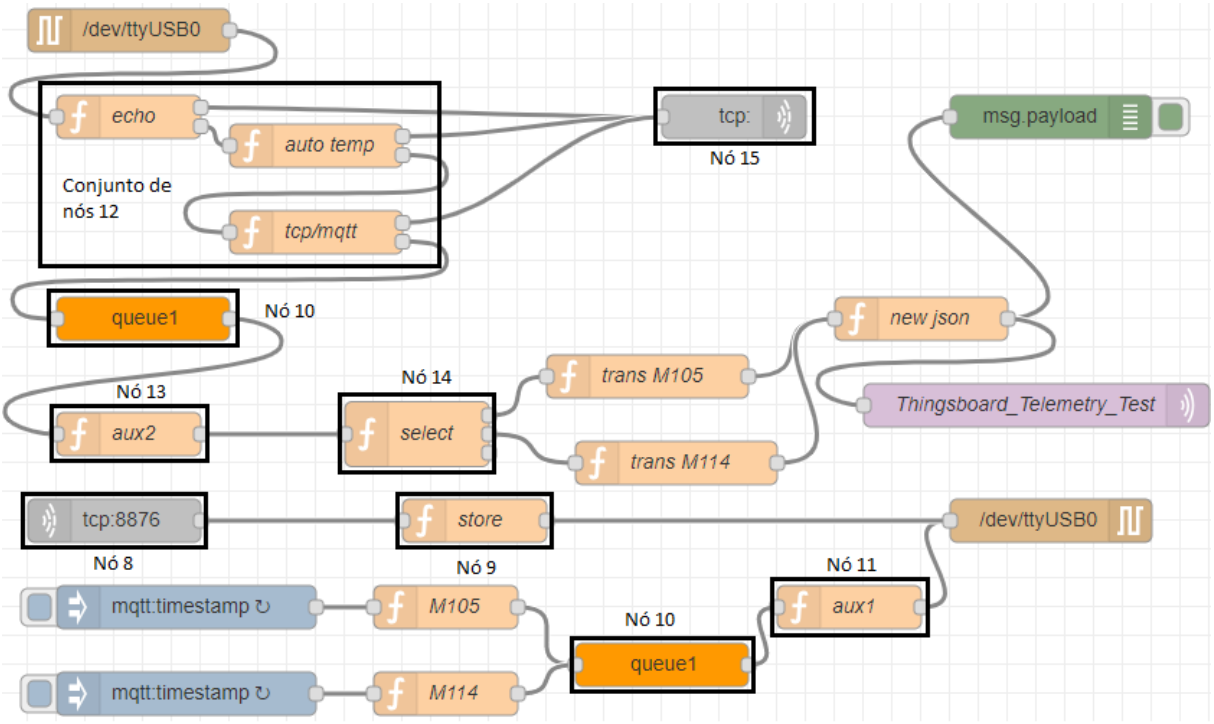


Figura 4.12 - Fluxo com a comunicação simultânea de interface e PC *Host* com a impressora

Comunicação com PC *Host*

Para estabelecer a conexão entre PC *Host* e impressora terão de ser configurados na interface, nomeadamente nos nós 8 e 15 (TCP *input e* TCP *output*, respetivamente) e na aplicação do PC *Host*, parâmetros que permitam essa conexão. No que diz respeito aos nós, é estabelecido o seguinte:

Nó 8: Definir o nó para aceitar pedidos de conexão recebidos. Terá de ser apenas definida a porta em que será realizada a comunicação com a PC *Host*;

Nó 15: Este nó na sua configuração contém uma opção do tipo "*Reply to TCP"*. Será configurado o nó para esta opção que, assim, responde a mensagens recebidas no nó 8.

Para além da configuração estabelecida para estes dois últimos nós, na aplicação do PC *Host* terá de ser definida a *socket* TCP/IP. A Figura 4.13 evidencia essa configuração.

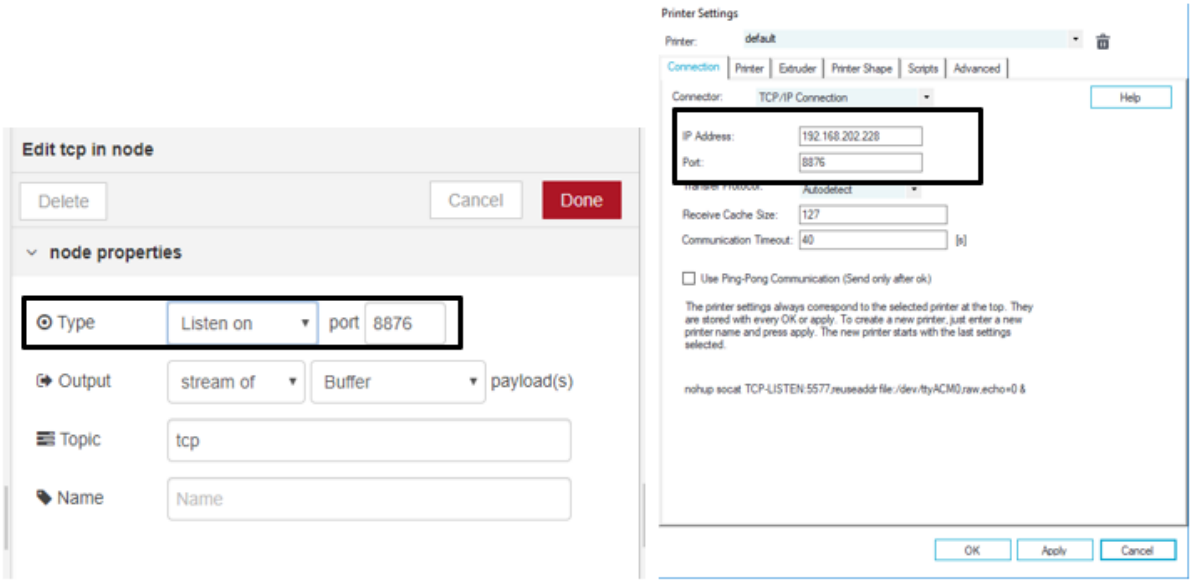


Figura 4.13 - Especificação da conexão TCP/IP na aplicação: a) Node-Red; b) Repetier-Host

Tal como é possível aferir, a interface através do Node-Red atende aos pedidos de conexão efetuados pela porta definida e na interface do da aplicação do PC *Host* terá que se especificar tanto o endereço IP da Raspberry Pi, como a porta por onde será realizada a comunicação, que terá de ser igual para ambas as aplicações.

Gestão de mensagens

Abordando a gestão de mensagens a realizar pela interface, os nós implementados no fluxo da Figura 4.12, à exceção dos nós 8 e 15, permitem encaminhar corretamente as mensagens para os destinos adequados. Poderão identificar-se diferentes tipos de mensagens de resposta por parte da impressora:

- Mensagens “echo”, já referidas previamente em 4.2.1;
- Respostas a pedidos da interface (M105 e M114);
- Respostas a pedidos do PC *Host*.

Dos nós que contribuem para a correta gestão das mensagens, distinguem-se os nós principais e auxiliares. Os nós auxiliares apenas introduzem variáveis no fluxo que permitem aos nós principais efetuar facilmente o encaminhamento correto das mensagens. Sendo assim, os nós com funções auxiliares para a gestão de mensagens dispõem as seguintes funcionalidades:

Nó 9: Permite guardar o valor da origem da mensagem (neste caso, PC *Host*), numa variável criada para o fluxo, ou seja, se são oriundas do PC *Host* ou da interface. Para esse efeito, é adicionada às funções “M105” e “M114” do conjunto de nós 1 a mesma funcionalidade que a este nó, identificando as mensagens oriundas da interface;

Nó 10: Permite a criação de uma fila que armazena mensagens recebidas e devolve-as de forma organizada, uma a uma;

Nó 11: Permite a criação de uma variável *array* que armazena os pedidos de código G para a impressora;

Nó 13: Permite a criação de uma variável *array* de igual dimensão ao criado no nó 11, onde serão armazenadas as respostas aos códigos G pedidos.

Focando agora nos nós que permitem separar os diferentes tipos de mensagens, estes poderão ser divididos nos que separam as mensagens que serão encaminhadas para o PC *Host* e para a plataforma IoT, e no que encaminha as mensagens (destinadas à plataforma IoT) para as diferentes funções de transformação do conjunto de nós 5 (Figura 4.8). Desta forma, estes nós apresentam as seguintes funcionalidades:

Conjunto de nós 12: Conjunto de funções que encaminham as mensagens para o PC *Host* ou para a plataforma IoT. É constituído por três nós "function":

- A função denominada "echo" identifica as mensagens de configuração inicial da impressora (Figura 4.5) encaminha-as para o PC *Host*;
- Uma função "auto temp" é também adicionada. A inclusão desta função deve-se ao envio por parte do PC *Host*, aquando do estabelecimento da comunicação com a impressora, de uma instrução de código G "M105 S3", solicitando o envio automático de dados da temperatura. Como passarão a ser enviadas automaticamente dados da temperatura sem qualquer tipo de solicitação do PC *Host* ou da interface, é adicionada esta função que separa os dados solicitados pelo utilizador, dos dados enviados automaticamente pela impressora que serão encaminhados para o PC *Host*;
- Por último a função "tcp/mqtt" que identifica a proveniência da mensagem através da variável mencionada no nó 9. Se esta variável assume o valor "tcp" é encaminhada para o PC *Host*, se apresenta o valor "mqtt" é encaminhada para a plataforma IoT.

Nó 14: Para as instruções de código G enviadas pela interface para a impressora, terão de ser encaminhadas corretamente as respostas para a função de transformação correspondente a cada código G (Conjunto de nós 5, Figura 4.8). Para esse efeito, é adicionada a função "select". O procedimento para o encaminhamento de mensagens da função "select" consiste na leitura de ambos os *arrays* criados nos nós 11 e 13, identificar o código G de cada linha do *array* do nó 11 e enviar a correspondente resposta do *array* 13 para a função de transformação adequada. Se porventura para uma determinada linha desses *arrays*, a resposta não for coincidente com o pedido (por exemplo, a um código G "M105" corresponder uma resposta com os dados da posição referentes ao código "M114"), nenhuma resposta é encaminhada para as funções de transformação. A lógica da função "select" é esquematizada na Figura 4.14, supondo uma dimensão de quatro linhas para cada *array*.

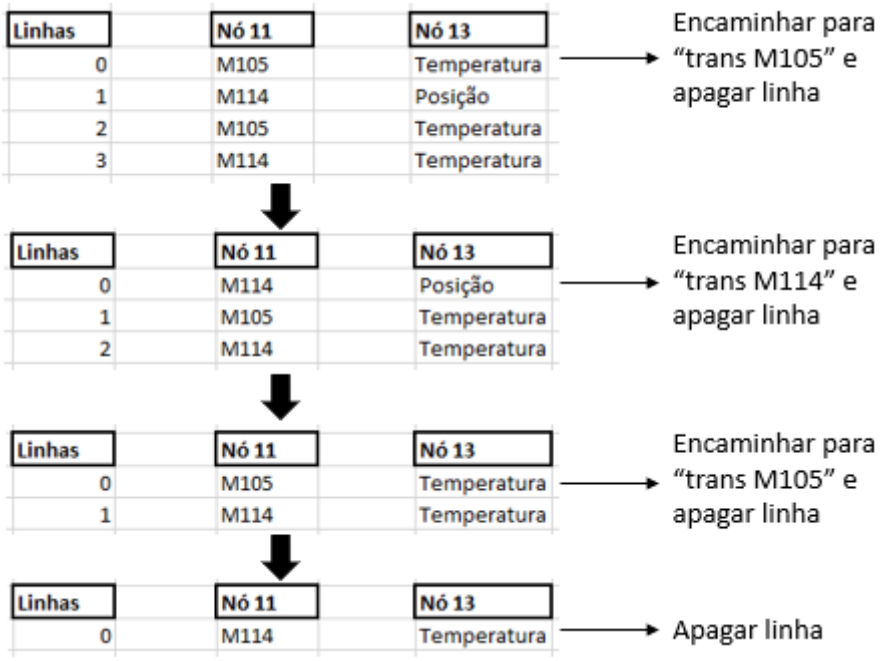


Figura 4.14 - Lógica do nó 14

Desta forma, dependendo da origem do pedido, as mensagens poderão seguir percursos diferentes ao longo do fluxo. A Figura 4.15 e a Figura 4.16 mostram o percurso seguido pelas mensagens enviadas pela interface e o PC *Host*, respetivamente.

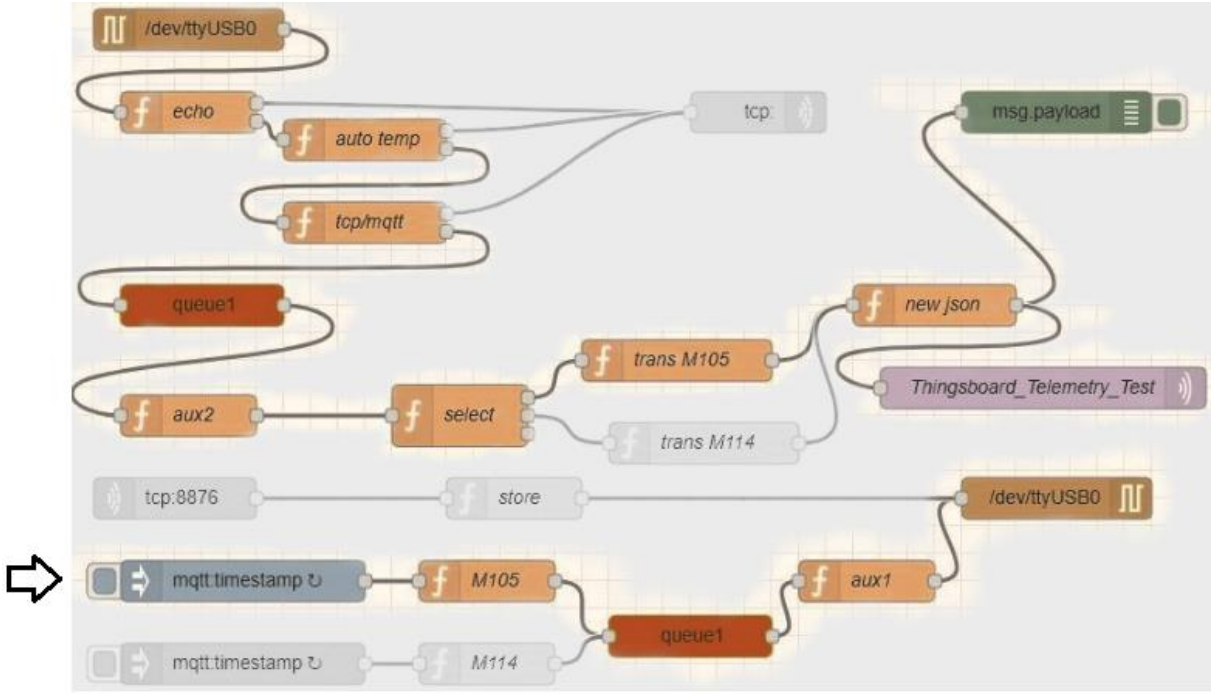


Figura 4.15 - Percurso das mensagens enviadas pela interface

A partir da análise da Figura 4.15, é possível aferir a injeção de uma instrução "M105" pela interface, passando pelas funções que permitem o encaminhamento para o *output*, terminando no nó 7, correspondente ao envio para a plataforma IoT, e também no nó 4.

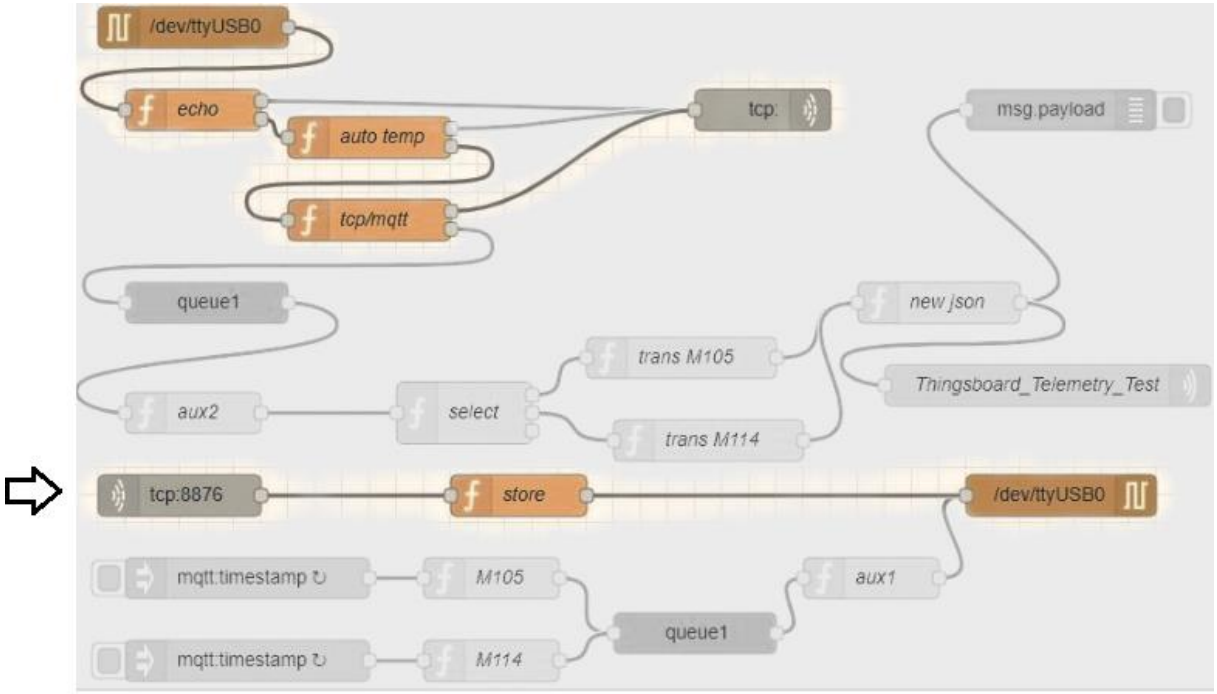


Figura 4.16 - Percurso das mensagens enviadas pelo PC *Host*

Testando o *software* desenvolvido nesta etapa, sem que um trabalho de impressão esteja a decorrer, verificou-se a correta separação das mensagens para o *output* correto. Quando testado com um trabalho de impressão a decorrer, o *software* separa corretamente as mensagens, porém a impressora pausa a impressão automaticamente ao final de cerca de 10 segundos após o seu início. Na próxima etapa (subsecção), o fluxo já irá incluir as funcionalidades que permitem corrigir este problema.

4.2.4 Implementação de número de linha e *checksum*

Tal como foi referido, o *software* desenvolvido até à etapa anterior foi testado sem que houvesse a impressão de uma peça, funcionando corretamente. Porém, um problema surgiu quando foi testado o *software* durante um trabalho de impressão. Cerca de 10 segundos após o início da impressão, a máquina pausava automaticamente, com uma mensagem "Click to resume" a aparecer na interface gráfica da impressora. Este problema é causado pela falta de dois valores na instrução de código G enviado para a impressora, que consistem no número de linha e do *checksum*. O *checksum* é um parâmetro usado na verificação da integridade dos dados e imposto pelo protocolo das impressoras, podendo ser usado para a comunicação com outro tipo de equipamentos. Este representa o número de *bits* de uma mensagem a transmitir. Este valor, depois de calculado, é adicionado no final da linha de código G que se pretende transmitir precedido de um caracter "*". Quando a mensagem é rececionada pela impressora, esta verificará primeiro o byte adicionado (*checksum*) para perceber se os dados permanecem ou não inalterados. Em caso positivo, os dados serão aceites, senão será enviada uma mensagem a pedir o reenvio dos dados. Terá de ser também adicionado o número de linha que precede sempre o código G, aumentado sempre mais um o seu valor consoante o envio de uma linha de código. Ambos os valores (número de linha e *checksum*) têm de estar presentes na linha de código G. Caso seja omitido um destes valores gera-se um erro. [28]

Assim, os nós que permitem corrigir o problema apresentado no final da subsecção anterior são adicionadas nesta etapa, obtendo-se o seguinte fluxo apresentado na Figura 4.17.

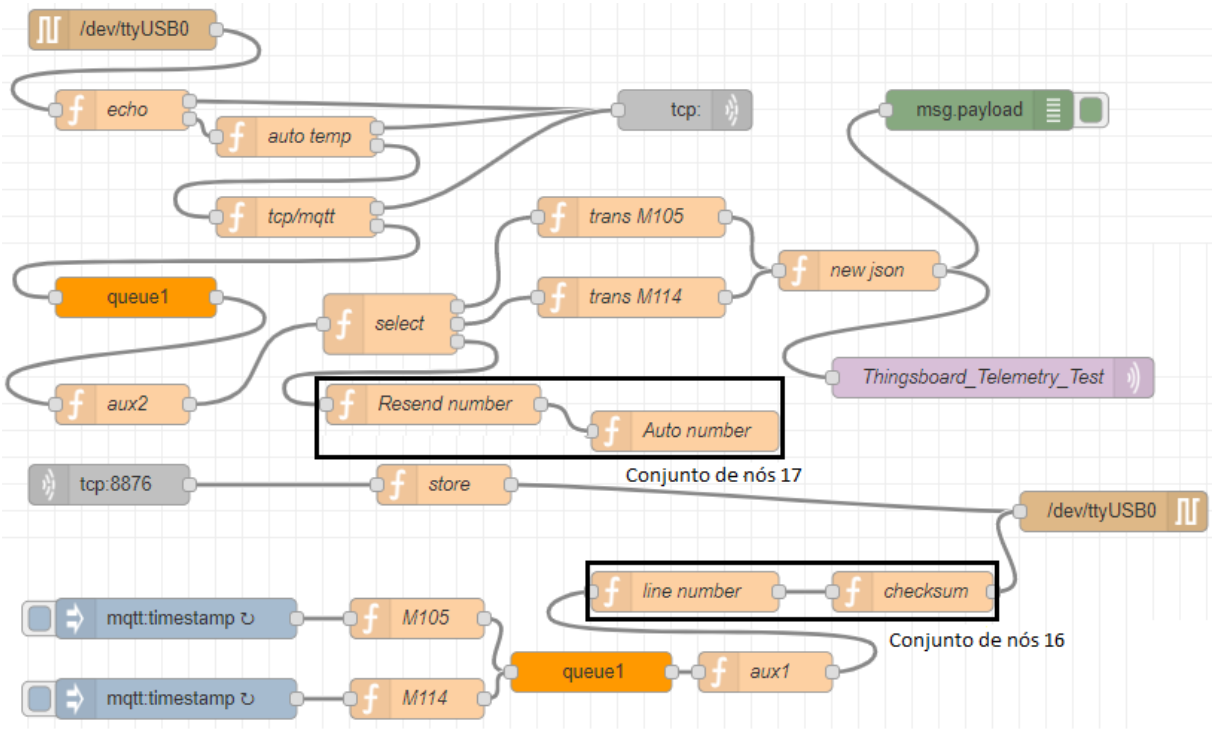


Figura 4.17 - Fluxo que possibilita a impressão por cartão SD

Poderão destacar-se dois conjuntos de nós implementados nesta etapa, sendo que um deles adiciona na instrução do código G enviada pela interface, um número de linha e um valor de *checksum*, e o outro, em caso de uma resposta de erro por parte da impressora, permite repetir o número de linha que esteve na origem desse erro. Poderá detalhar-se as funcionalidades dos dois conjuntos de nós da seguinte forma:

Conjunto de nós 16: Constituído pelas funções "Line Number" e "checksum" e tal como indicam os nomes, permitem adicionar um número de linha e um valor de *checksum*, respetivamente. Em relação à função "Line Number", o número de linha a enviar estará sempre dependente do número de linha da mensagem enviada imediatamente antes, sendo que acresce sempre mais um em relação a esta. Especificando sobre a função "checksum", numa primeira instância transforma a s*tring* num *array* de *bytes* e, de seguida, calcula o *checksum* realizando um XOR nos *bytes* que constituem a totalidade da mensagem (número de linha incluído). Depois de calculado, este valor é adicionado no final da instrução de código G precedido do caracter "*".

O código da função "checksum" em JavaScript está representado na Figura 4.18 onde estão também salientes os diferentes passos que permitem o cálculo do *checksum.*

Figura 4.18 - Código da função "checksum"

A partir da análise da Figura 4.18, é possível distinguir as duas componentes diferentes do código já referidas anteriormente:

1. Componente que inclui as linhas que permitem a conversão da mensagem do formato *string* para um *array* de bytes. Como a linguagem de programação JavaScript não tem uma função específica que permita a conversão direta da mensagem, recorre-se a ciclo "for" que corre todos os caracteres da mensagem e, em cada um, aplica-se a função "charCodeAt" que retorna o código de cada caracter e é adicionado a um *array* através da função *push*.
2. Na segunda parte do código através de um operador de bits XOR que percorre toda a extensão do *array* de *bytes*, calcula-se o *checksum* e, posteriormente, é adicionado no final da linha código G precedido de um "*". Após a adição do *checksum*, acrescenta-se dois caracteres correspondentes à indicação de final de linha.

Desta forma, a instrução de código G, gerada pela interface e que será enviada para a impressora, terá o seguinte formato percetível na Figura 4.19.

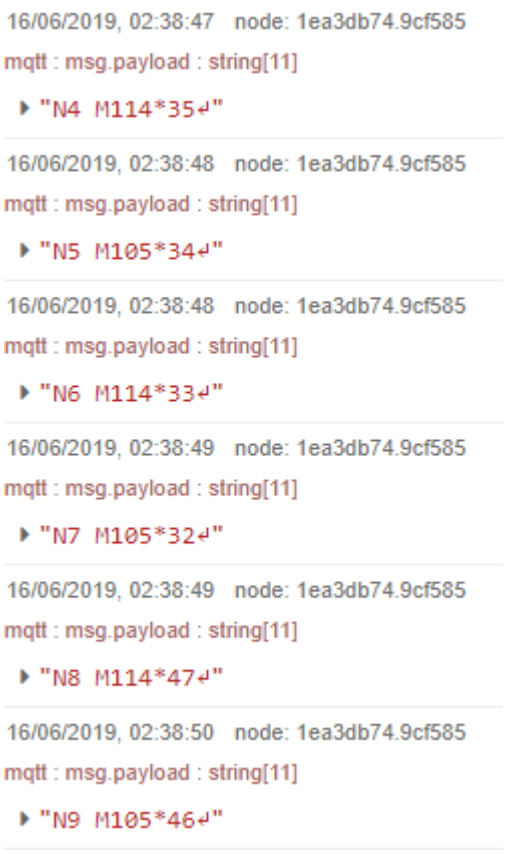


Figura 4.19 - Formato de código G enviado pela interface

Durante uma impressão, ocasionalmente poderá ser "ignorada" pela impressora uma instrução de código G, devido ao elevado processamento a que esta está sujeita (trabalho de impressão e resposta aos pedidos de código G). Poderá também, eventualmente, o cálculo do valor de *checksum* na aplicação desenvolvida diferir do calculado pelo *firmware* da impressora 3D. Em ambos os casos será emitida uma resposta de erro por parte da impressora. Essas respostas estão representadas na Figura 4.20.

Figura 4.20 - Respostas de erro: a) Quando o número de linha está errado; b) Quando valor do *checksum* calculado na aplicação difere do calculado pelo *firmware* da impressora

Consequentemente, deverão ser adicionadas funcionalidades à aplicação que permitam repetir o envio da linha de código G expressa pela impressora nas mensagens de erro. Desta

forma, é introduzido o conjunto de nós 17 que irá permitir adicionar as seguintes funcionalidades:

Conjunto de nós 17: Constituído pelas funções "Resend number" e "Auto number":

- Relativamente à função "Resend number", tem como propósito identificar as mensagens de erro (Figura 4.20), causadas pelo incorreto número de linha ou *checksum* (as mensagens enviadas pela primeira vez são perdidas), e enviá-las para a função seguinte, "Auto number". Será realizada uma adaptação na função "select" (nó 14, Figura 4.12), que baseia-se nas respostas da impressora que não coincidem com a instrução de código G enviada pelo conjunto de nós 1 (Figura 4.3), em vez de serem apagadas, como previsto inicialmente, serão encaminhadas para um terceiro *output* da função "select".
- No que diz respeito à função "Auto number", numa primeira instância obtém o número da linha que é pedido pela impressora para reenviar. De seguida, converte esse número de linha do formato *string* para o formato *number*. Finalmente, guarda esse valor na variável usada pela função "line number" (conjunto de nós 16, Figura 4.17) correspondente ao número de linha enviado anteriormente.

Com os nós adicionados nesta etapa foi possível verificar a impressão através do cartão SD, sem interrupções ou constrangimentos causados pela implementação da interface. No entanto, a adição destes nós impede a comunicação do PC *Host*, pois a sua aplicação faz a sua própria contagem dos números de linha. Consequentemente, quando o PC *Host* comunica em simultâneo com a interface, as suas mensagens irão transmitir um número linha errado, pois a contagem da impressora engloba as mensagens enviadas pelo PC *Host* e interface. Portanto, este aspeto será corrigido na próxima etapa.

4.2.5 Substituição de número de linha e *checksum* de pedidos do PC *Host*

Por fim, será adicionado nesta etapa o nó que permite, inicialmente, remover o número de linha e o valor do *checksum* da instrução de código G proveniente do PC *Host* para, posteriormente, conectar ao conjunto de nós 16, implementado na etapa anterior (Figura 4.17), que irá adicionar um novo número de linha e *checksum* numa contagem realizada através dos comandos enviados, tanto pela própria aplicação IoT desenvolvida, como pelo PC *Host*. A Figura 4.21 mostra a disposição do fluxo após efetuadas estas alterações.

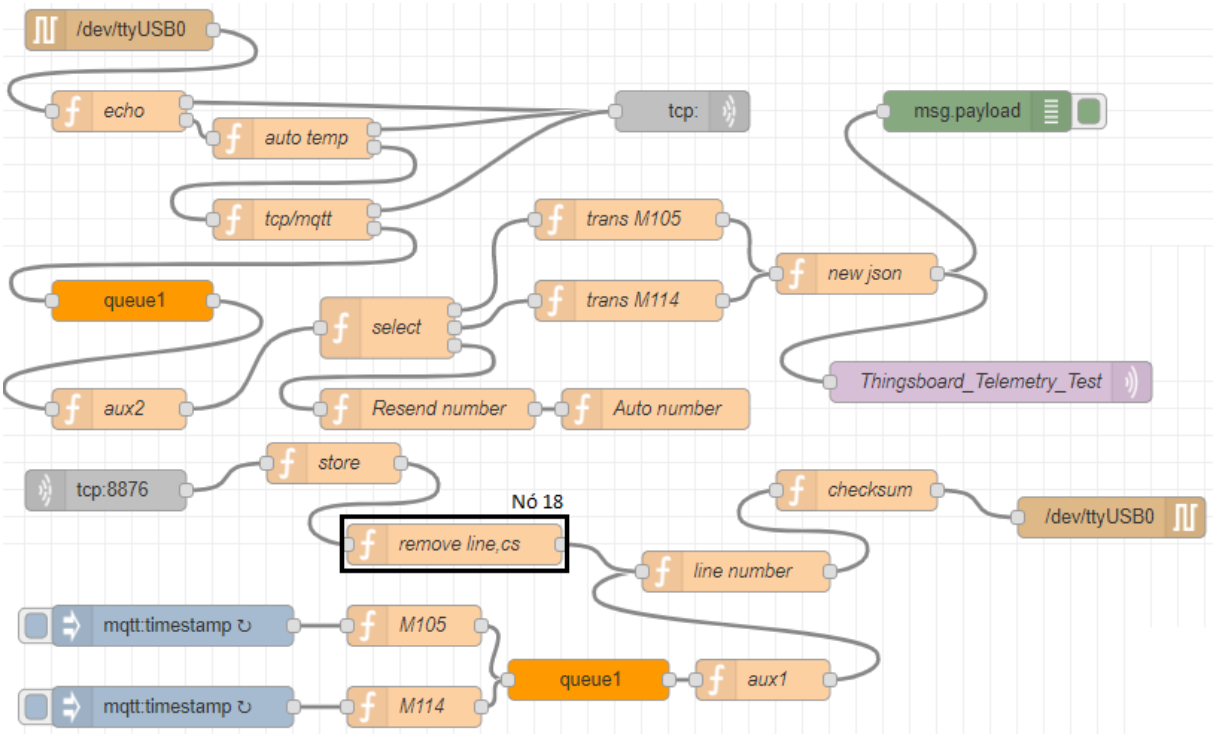


Figura 4.21 - Fluxo com a funcionalidade de substituição de número de linha e *checksum* para comandos do PC *Host*

Com a introdução do nó 18, serão removidos aos comandos provenientes do PC *Host* o número de linha e o *checksum*. Desta forma, terá de se assegurar a adição de um novo número de linha e *checksum* através da conexão do nó 18 com o conjunto de nós 16 (Figura 4.17).

No entanto, as funções representadas pelo conjunto de nós 16 permite a adição do número de linha e *checksum* para apenas uma linha de código G por mensagem. Ou seja, caso sejam enviadas duas ou mais instruções de código G numa só mensagem (possível de acontecer durante uma impressão por PC *Host*), apenas serão adicionados número de linha e *checksum* para a primeira linha da mensagem. Serão, então, realizadas as respetivas alterações nas funções do conjunto de nós 16 (Figura 4.17) que irão colmatar esta falha, possibilitando a adição destes dois parâmetros para todas as linhas de código G existentes numa mensagem.

4.2.6 Implementação de comandos manuais

Por fim, de modo a que seja possibilitada a injeção manual de uma ou mais instruções de código G terá de ser implementado mais um conjunto de funções. Nesse sentido, é obtido o fluxo demonstrado na Figura 4.22.

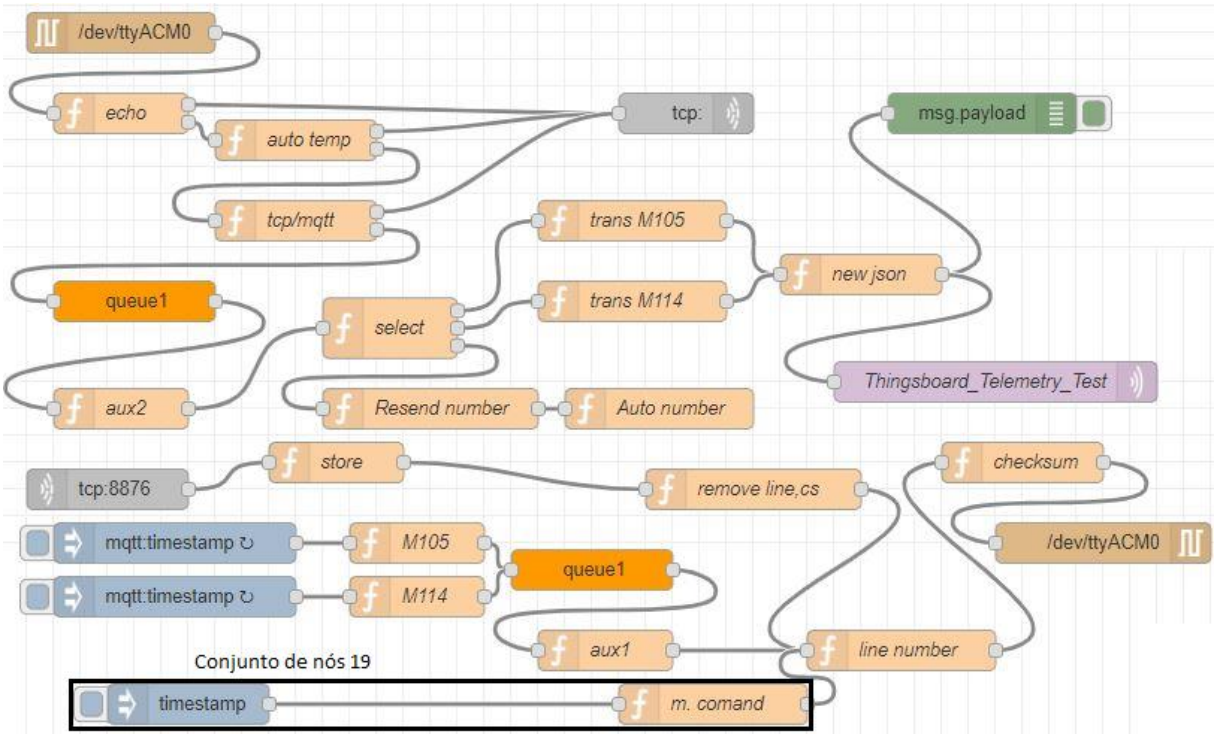


Figura 4.22 - Implementação de comando manual na aplicação IoT

Desta forma, o conjunto de nós 19 permite fazer a injeção de comandos manuais no IDE do Node-Red, sendo os nós adicionados os seguintes:

- Um *input* "inject" que permite a injeção de um *timestamp* no IDE. Estará especificado para injeção manual pelo utilizador;
- Uma "function" que permite alterar o conteúdo do *timestamp* para um código G determinado pelo utilizador.

De seguida, é estabelecida uma ligação ao conjunto de nós 16 (Figura 4.17) para que seja adicionado um número de linha e um *checksum*, de maneira a completar o comando manual. Esta funcionalidade poderá ser bastante útil num futuro desenvolvimento onde na plataforma IoT seria enviado um comando e a interface o converteria no código G necessário (por exemplo, a plataforma envia um comando "Stop printer" e a interface converte no código G M112 (permite a paragem de todos os motores da impressora)).

Assim, estão contabilizados todos os desenvolvimentos necessários que permitem cumprir as funcionalidades enunciadas em 2.4.1 para o *firmware* Marlin. É, então, apresentada na Figura 4.23, a solução adaptada às versões 1.0.0 e 1.0.9 do *firmware* Marlin.

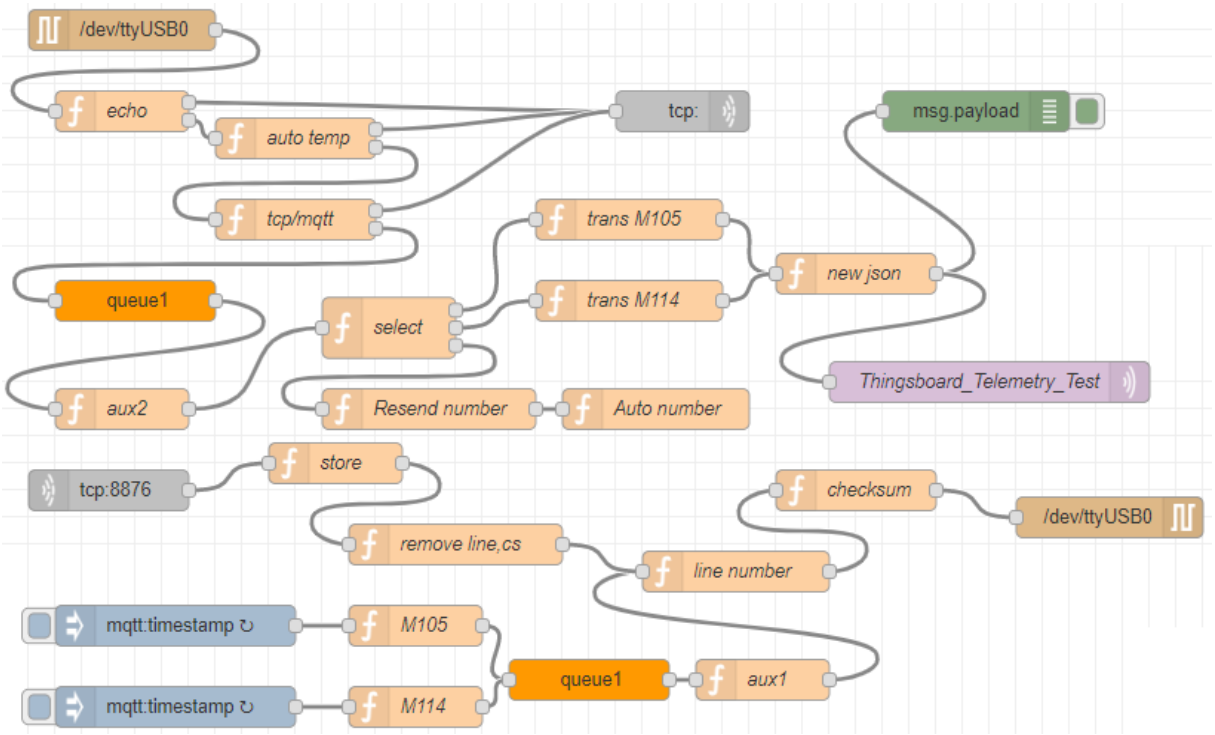


Figura 4.23 - Solução para *firmware* Marlin 1.0.0 e 1.0.9

4.3 Desenvolvimentos para *firmware* Repetier

Serão enunciados, nesta secção, os desenvolvimentos que permitem adaptar a solução desenvolvida na secção 4.2 para o *firmware* Repetier. Será estabelecido como ponto de partida os progressos já efetuados e descritos em 4.2, sendo de seguida devidamente alteradas apenas as funções que são necessárias para obter um funcionamento semelhante à solução desenvolvida para *firmware* Marlin.

Nesse sentido, serão efetuados, inicialmente, testes iniciais que permitem a compreensão das diferenças entre ambos os *firmwares*. De seguida, serão adicionados ou alterados nós que permitem adaptar a solução para o *firmware* Repetier e, por fim, apresentado o resultado final.

A impressora testada que permitiu esta adaptação corresponde a uma Stacker com o Repetier de versão 1.0.2, que tem a particularidade de incorporar 4 extrusores.

4.3.1 Testes iniciais

Ao fluxo do *firmware* Repetier foi, inicialmente, implementada comunicação série de forma idêntica ao implementado em 4.2.1 representado na Figura 4.3. Este programa de teste inicial permite descobrir quais as adaptações necessárias a efetuar para o desenvolvimento da solução para o Repetier.

À semelhança do que ocorre com o *firmware* Marlin, aquando da iniciação da impressora são recebidas mensagens com informação sobre parâmetros da máquina. Esta informação é apresentada na Figura 4.24.

Figura 4.24 - Mensagens iniciais *firmware* Repetier

Estas mensagens iniciais transmitem assim para o Repetier a mesma informação que as mensagens "echo" para o Marlin. O Repetier tem também a particularidade de enviar automaticamente uma mensagem "wait↵" a cada segundo quando não recebe qualquer pedido.

A partir do envio de instruções "M105" e "M114" poderá observar-se as respostas da impressora e comparar com o *firmware* anterior. Essas respostas são mostradas na Figura 4.25.

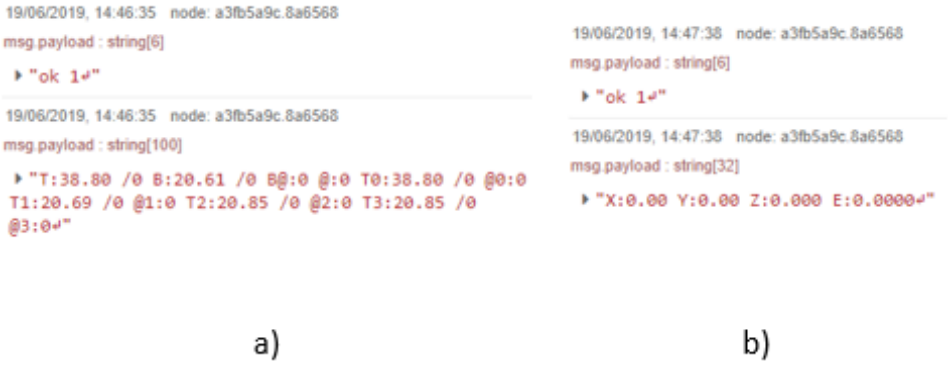


Figura 4.25 - Resposta da impressora Stacker aos comandos: a) M105; b) M114.

Pela análise da Figura 4.25, é possível aferir que existem diferenças em ambas as respostas. Em relação ao comando "M105" a resposta enviada contém a informação da temperatura de quatro extrusores e da base de impressão. No caso do comando "M114" observa-se que a resposta não contém tanta informação como para a resposta ao comando do *firmware* Marlin. Quando conectada a impressora ao PC *Host*, verifica-se também a inexistência de mensagens de temperatura enviadas automaticamente

Desta forma, deverão ser efetuadas algumas alterações a certas funções da solução desenvolvida para *firmware* Marlin, para se poder obter uma aplicação com funcionalidades idênticas para ambos os *firmwares*. Estas alterações serão abordadas na próxima subsecção.

4.3.2 Nós alterados

A partir deste ponto, procedeu-se à implementação dos comandos que fizeram parte da configuração do anterior fluxo do *firmware* Marlin, obtendo-se o seguinte arranjo ilustrado na Figura 4.26.

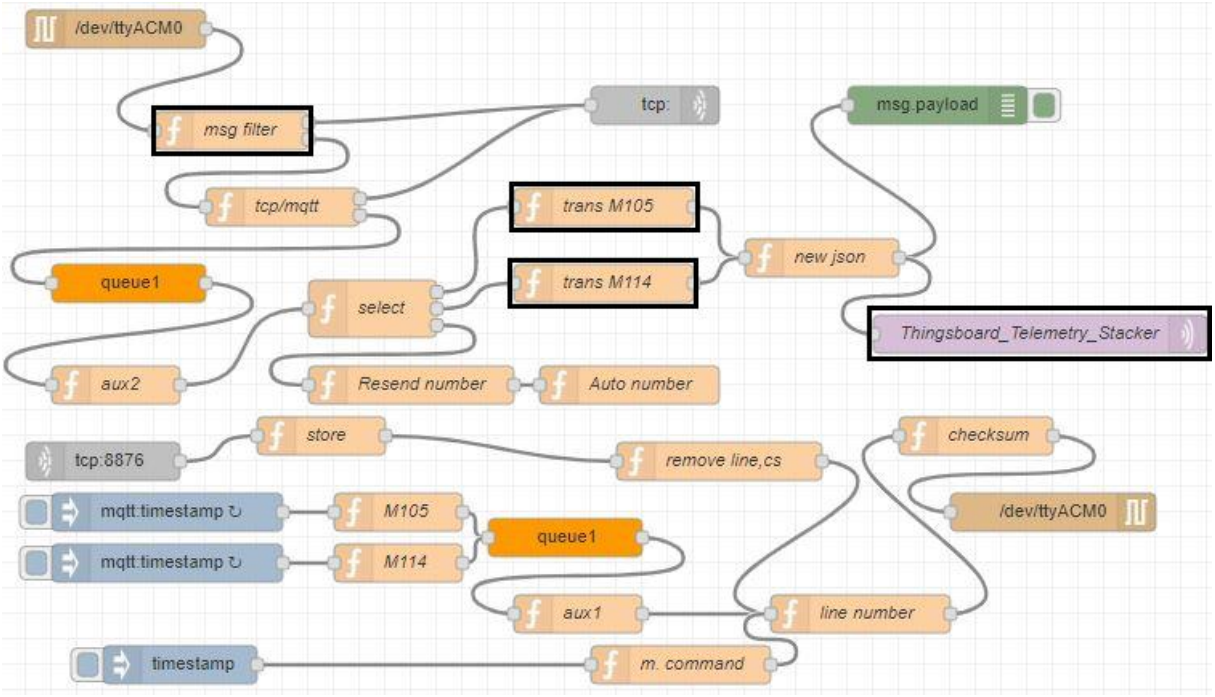


Figura 4.26 - Nós alterados para *firmware* Repetier.

Pela análise da Figura 4.26 é possível aferir que apenas foram efetuadas alterações para este *firmware* num número reduzido de nós, nomeadamente cinco, em que em quatro deles foram realizadas adaptações no seu código ou na sua configuração e um foi removido do fluxo. Portanto, as adaptações que foram realizadas poderão ser explicadas da seguinte forma:

- **"msg filter":** Cumpre o mesmo propósito da função "echo" do conjunto de nós 12 (Figura 4.12), sendo que apenas se alterou a forma como algumas das mensagens são filtradas;
- **"trans M105":** Com o suporte para multi-extrusor da impressora Stacker foi necessário efetuar a adaptação desta função, tanto para esse suporte, como para alguns detalhes na disposição dos dados na *string*. Após efetuadas as alterações foi obtida a função "trans M105" com um código que funciona para ambos os *firmwares* e que funciona corretamente para um ou mais extrusores. O código desta função pode ser observado na Figura 4.27;
- **"trans M114":** Sendo que o conteúdo da resposta ao comando M114 do *firmware* Repetier é ligeiramente diferente em relação ao *firmware* Marlin (Figura 4.10 e Figura 4.25), realizou-se a adaptação do código desta função para obter um funcionamento idêntico ao da solução para Marlin;
- **"ThingsBoard_Telemetry_Stacker":** Nó *output* MQTT para onde são enviados os dados obtidos da impressora Stacker. No lado da plataforma IoT está criado um dispositivo para o qual os dados obtidos desta impressora estão destinados a ser enviados por MQTT. Portanto, da mesma maneira que foi configurada a ligação MQTT para a solução desenvolvida para Marlin (Figura 4.11), também é

configurada a mesma ligação para o Repetier, com os dados que permitem associar o dispositivo na plataforma IoT a serem distintos.

A função “auto temp” da solução para Marlin (conjunto de nós 12, Figura 4.12) foi removida, devido ao facto de neste *firmware* não serem enviadas as mensagens automáticas da temperatura aquando do estabelecimento da conexão da impressora com o PC *Host*.

```
var newmsg
var aux
newmsg=msg.payload;
newmsg=newmsg.substring(0,newmsg.length -2)
newmsg=newmsg.split(" ");
for (var i=0;i<newmsg.length;i++) {
    aux=newmsg[i]
    if (aux.search("/")!=-1) {
        aux=newmsg[i-1].substring(0,newmsg[i-1].search(":"))+"sp:"+aux.substring(aux.search("/")+1)
        newmsg[i]=aux
    }
}
msg.payload=newmsg
return msg;
```

Figura 4.27 - Código função "trans M105" adaptado a multi-esxtrusor.

Analisando o código da Figura 4.27, é possível destacar duas fases diferentes no seu desenvolvimento. Numa primeira fase, procedeu-se à criação do *array*, com cada variável a ocupar uma linha diferente do *array*. Na fase seguinte, a representação das variáveis que mostram os valores das temperaturas de *set point* dos extrusores e da base de impressão é alterada, para uma mais simples interpretação dos valores e, também, para que seja realizada corretamente a conversão do *array* para JSON.

Desta forma, com as alterações promovidas, enviando uma instrução M105 ou M114 poderão as mensagens ser convertidas corretamente no formato JSON. Poderá observar-se o resultado final desta conversão na Figura 4.28.

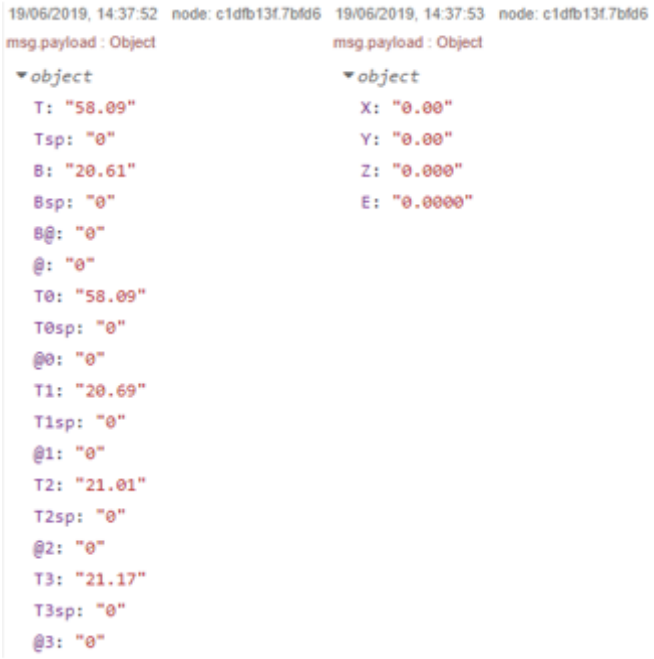


Figura 4.28 - Resposta do *firmware* Repetier convertida para o código G: a) M105; b) M114.

Para as restantes funções do fluxo não foram efetuadas alterações, já que a sua função não é afetada pela mudança de *firmware*.

4.4 Resumo do capítulo

Este capítulo desenvolveu a solução para a proposta sugerida que permite cumprir os objetivos e requisitos delineados anteriormente. Desta forma, a solução foi adaptada ao *firmware* Marlin, versões 1.0.0 e 1.0.9, e Repetier, versão 1.0.2. A interface está preparada para estabelecer comunicação bidirecional entre a impressora 3D e o PC *Host* e também para obter e enviar dados por MQTT para a plataforma IoT. Inicialmente, foi contextualizado o funcionamento da ferramenta Node-Red. De seguida, solução foi desenvolvida para *firmware* Marlin e depois para Repetier, na qual para o segundo foi possível aproveitar grande parte da solução desenvolvida do primeiro e fazer as devidas alterações e adaptações.

5 Testes e resultados

Este capítulo terá como propósito a apresentação dos testes efetuados ao sistema construído nesta dissertação, com o intuito de validar a solução construída para todas as versões de *firmware* mencionadas no capítulo 4. De seguida, serão apresentados os resultados decorrentes desses testes, onde haverá espaço para análise dos mesmos.

De forma a avaliar se o objetivo principal é cumprido, ou seja, se a interface que se pretende implementar permite monitorizar uma impressora 3D, serão efetuados dois testes que avaliam o impacto da aplicação desenvolvida durante um trabalho de impressão: Um teste de impressão via cartão SD (principal método de validação) e um teste de impressão pelo PC *Host*. A possibilidade de envio de comandos manuais, outro dos requisitos propostos, também é testada através do envio de comandos manuais pelo PC *Host* ou por injeção no IDE na interface. Por fim, com o intuito de analisar se a interface consegue monitorizar de forma viável uma impressão, um teste que permite avaliar a taxa máxima de mensagens que poderão ser comunicadas à impressora é efetuado.

5.1 Teste de impressão via cartão SD

Este teste consiste na colocação de um ficheiro *gcode* a imprimir via cartão SD e, por conseguinte, na análise dos resultados dessa impressão. É certamente o método de validação mais importante deste projeto, uma vez que o objetivo principal determinado traduz-se na capacidade de impressão via cartão SD, em simultâneo com solicitações automáticas de código G através da aplicação IoT desenvolvida. Assim, a obtenção de dados de uma impressão terá de ser executada sem que haja qualquer tipo de interferência com o funcionamento normal do equipamento durante a produção de uma peça.

Desta forma, podemos aferir que este teste serviu dois propósitos:

- Determinar a forma correta de desenvolver o código que permite o reencaminhamento correto das mensagens apenas para o caso da plataforma IoT. Na Figura 5.1 e Figura 5.2 apresentam-se dois gráficos que podem comprovar a obtenção de dados na plataforma, na qual um corresponde à variação do posicionamento do extrusor e outro à variação da temperatura da base e do extrusor, respetivamente;
- Validação da solução para todos os *firmwares*, ou seja, verificação se a implementação da interface não afeta os trabalhos de impressão que são executados.

O teste realizado consiste na execução de uma impressão completa de uma peça de nome “3DBenchy”, sendo uma peça teste de simples produção, que permite averiguar a calibração e a existência de irregularidades na impressora. Ao mesmo tempo que esta impressão ocorre deverão estar a ser obtidos dados da temperatura e da posição pela interface

e a serem encaminhados para a plataforma ThingsBoard. Os resultados obtidos permitem aferir que o trabalho de impressão não foi afetado pela implementação da interface, visto não ter sido interrompida a produção e que a peça foi impressa sem defeitos. A Figura 5.1, Figura 5.2 e Figura 5.3 mostram o sucesso deste teste. Portanto, a Figura 5.1 representa um gráfico da variação da posição do extrusor da impressora com o tempo durante a impressão. A Figura 5.2 consiste num gráfico da variação da temperatura. Em ambas estas duas últimas figuras, os dados foram retirados da plataforma ThingsBoard. Na Figura 5.3 apresenta-se uma foto da peça impressa a partir do cartão SD. Ao lado da peça encontra-se uma lapiseira que serve como referência do seu tamanho. A impressão aqui enunciada foi realizada na impressora Monoprice Marlin 1.0.0. Impressões nas restantes impressoras foram também realizadas, sendo que os dados obtidos e a foto da peça impressa nessas impressoras encontram-se em anexo.

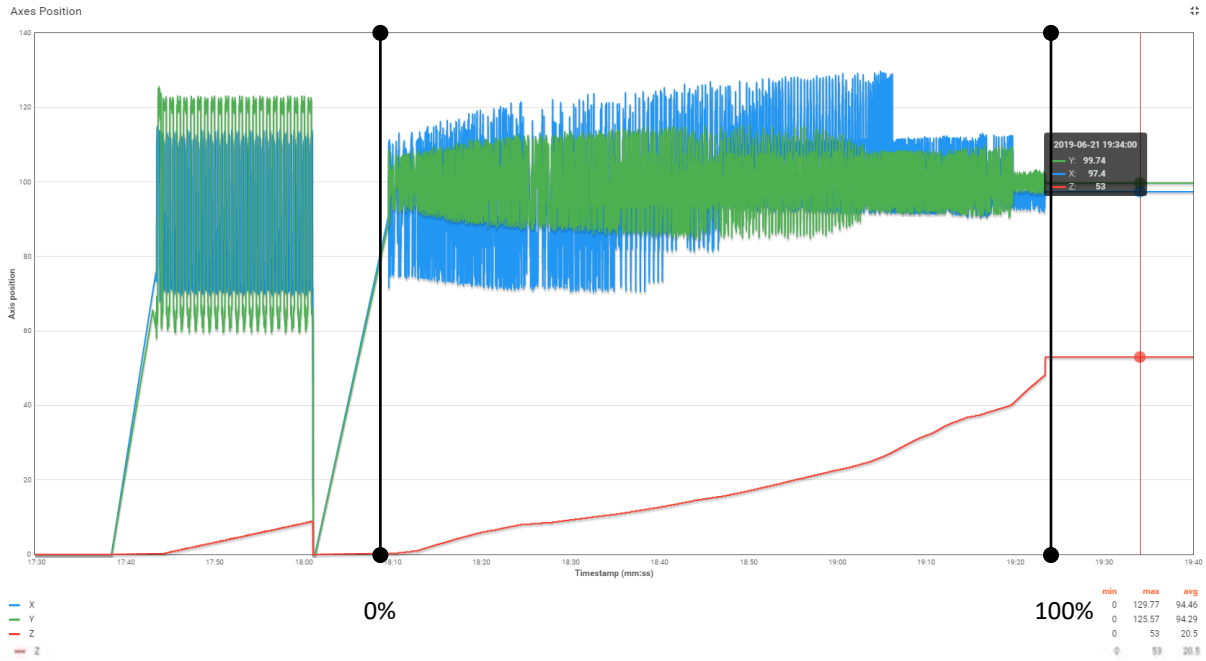


Figura 5.1 - Variação da posição (mm) do extrusor em função do tempo (minutos).

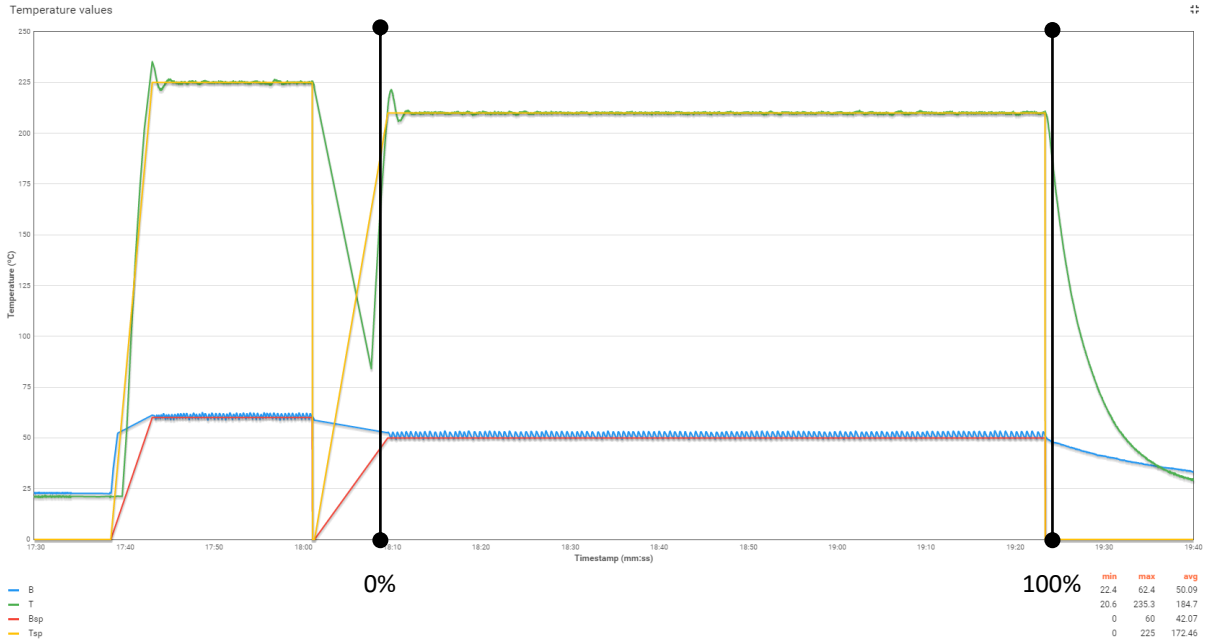


Figura 5.2 - Variação da temperatura (ºC) da base e do extrusor em função do tempo (minutos).

Figura 5.3 - Peça de teste impressa via cartão SD.

Tanto na Figura 5.1, como na Figura 5.2 estão identificados os intervalos de tempo em que se iniciou e terminou a impressão da peça. Poderá ser efetuada uma análise desta impressão partindo da peça impressa e dos gráficos obtidos. Essa análise encontra-se em anexo.

5.2 Teste de envio de comandos na aplicação do PC Host

Outro dos objetivos essenciais do projeto consiste na possibilidade de comunicação, caso seja necessário, do PC *Host* com a impressora. Dessa forma, ao mesmo tempo que o programa envia comandos de monitorização, terá que ser possível utilizar o PC *Host* caso essa utilização se justifique, como por exemplo, alteração de parâmetros (temperatura de *set point* da base ou do extrusor) ou ações de posicionamento do extrusor. E, portanto, foram realizados diversos testes como o envio de comandos bem variados, com a impressora a reagir bem, respondendo à totalidade dos pedidos. Como forma de demonstração dos resultados

(positivos) obtidos, serão apresentadas duas figuras que expressam isso mesmo da seguinte maneira:

- Com a Figura 5.4 tem-se a perspetiva da interface gráfica da aplicação do PC *Host* no momento antes de ser enviada qualquer solicitação para a máquina;
- Na Figura 5.5 apresenta-se a mesma interface gráfica da Figura 5.4 com uma instrução destacada na janela de comandos (parte de baixo da figura), que representa o movimento linear de 50mm no eixo X. Poderá também observar-se na janela de controlo manual (à direita) o valor da coordenada X do extrusor nos 50mm, demonstrando o sucesso no envio do comando pelo PC *Host*.

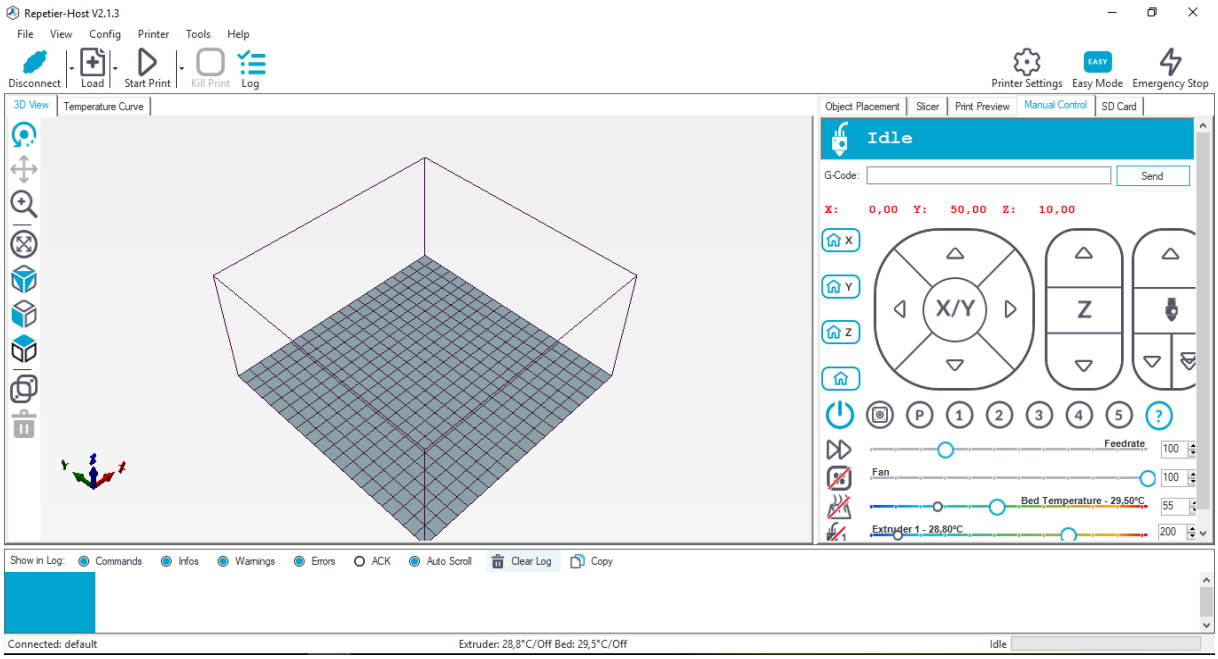


Figura 5.4 - Interface gráfica do PC *Host* antes do envio de uma instrução.

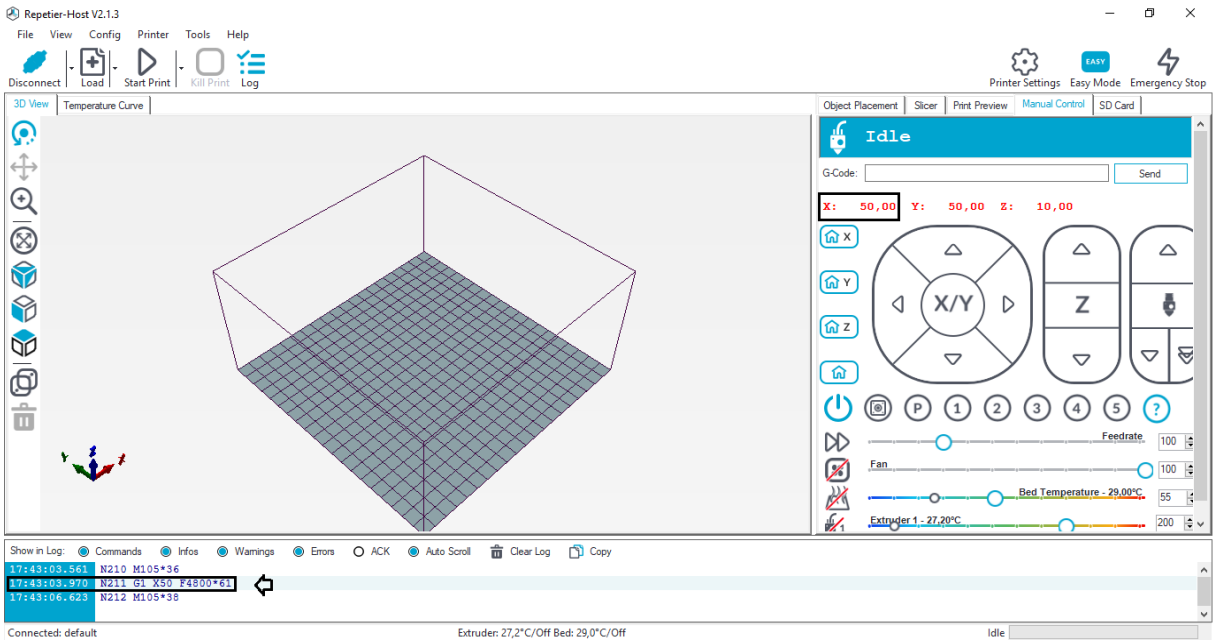


Figura 5.5 - Interface gráfica do PC *Host* após o envio da instrução G1 X50 F4800.

5.3 Teste de taxa de envio de mensagens

Este teste permite avaliar a taxa máxima de envio de mensagens pela interface, que possibilita uma impressora comunicar os dados sem falhas na impressão ou falhas na comunicação. Os resultados variam de impressora para impressora, sendo apresentados na Tabela 5.1.

Tabela 5.1 - Taxa máxima de envio de mensagens

Impressora testada (versão do *firmware)*	**Resultados sem impressão a decorrer (mensagens enviadas por segundo)**	**Resultados com impressão a decorrer (mensagens enviadas por segundo)**
Monoprice (Marlin 1.0.0)	25	10
Monoprice (Marlin 1.0.9)	17	10
Stacker (Repetier 1.0.2)	50	13

Os resultados verificados na Tabela 5.1 mostram uma diferença entre os tempos obtidos com uma impressão a decorrer e sem uma impressão a decorrer. Tal deve-se a que durante uma impressão os recursos computacionais da impressora sejam divididos entre o processamento dos comandos de impressão e os comandos de monitorização enviados pela interface, e na ausência de impressão a capacidade computacional da impressora se centrar unicamente nos comandos de monitorização.

De uma forma generalizada, os valores da Tabela 5.1 permitem monitorizar adequadamente o processo, pois serão possíveis de ser detetadas grande parte das variações dos valores que existem das variáveis a monitorizar durante uma impressão. No entanto, dificilmente poderão ser detetados todos os movimentos do extrusor com as taxas apresentadas, já que a quantidade de movimentos poderá muitas vezes superar a quantidade de mensagens que a impressora envia num curto espaço de tempo.

No caso dos valores apresentados na Tabela 5.1 serem ultrapassados, as impressoras deixarão de responder corretamente aos pedidos efetuados, sendo perdidas uma grande quantidade de mensagens.

5.4 Teste de envio de comandos manuais pelo IDE do Node-Red

Tal como no caso do envio de comandos individuais via PC *Host*, a interface terá que estar projetada para enviar comandos manuais através do IDE do Node-Red. Neste caso, para proceder ao envio de um comando, este terá de ser escrito em código G na função “m. command” (conjunto de nós 19, Figura 4.22) e, após implementado no programa, premir o botão associado ao *input* “inject*”* para proceder ao envio manual do comando pretendido. Na Figura 5.6 é mostrado um exemplo de um envio de um comando manual pelo IDE do Node-Red.

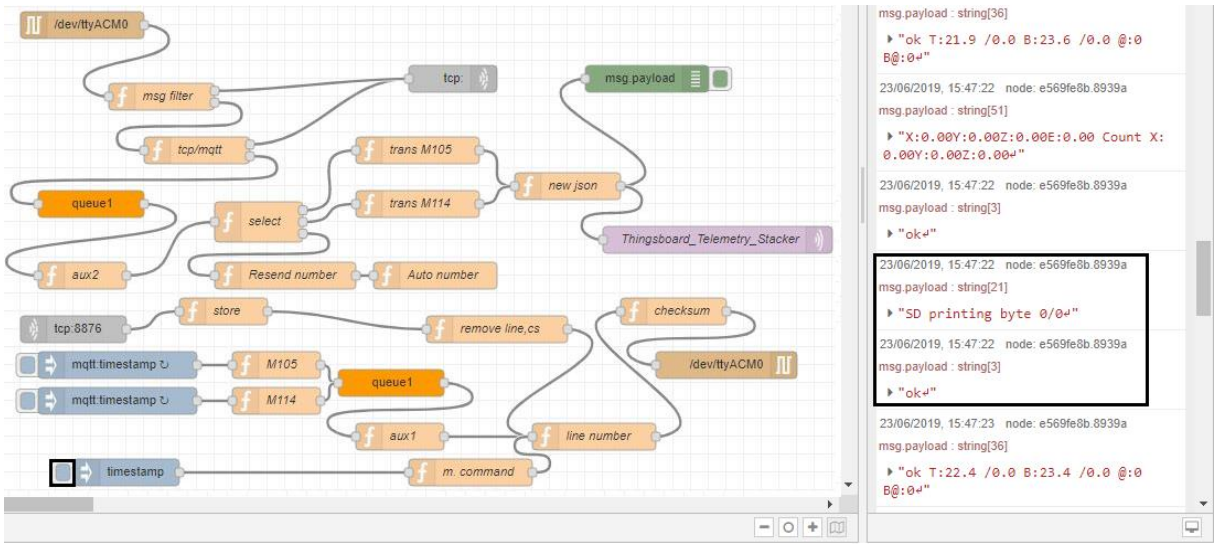


Figura 5.6 - Envio de comando manual pelo IDE do Node-Red.

Neste exemplo apresenta-se a implementação de um comando "M27" que permite a obtenção do estado de uma impressão no cartão SD. Pretende-se, assim, que o envio deste comando não cause transtorno no envio dos restantes comandos. Esse objetivo é conseguido.

Esta metodologia poderá ser adotada para a execução de outro tipo de instruções de código G a serem definidos pelo utilizador, tal como num PC *Host*, mas com a particularidade de o código G que se pretende implementar ser escrito. Esta forma de envio de código G permite usar comandos personalizados ou *macros*, quer manualmente, quer futuramente através da plataforma, facilitando a implementação de funcionalidades (plataforma IoT apenas iria enviar o código G sem N número de linhas ou *checksum* e sem necessidade de desenvolver funções à medida de cada *firmware*).

5.5 Teste de impressão via PC Host

Finalmente, outro dos testes executados consistiu na impressão de uma peça com o envio do código G da produção a ser efetuado pela via do PC *Host*, que controla a impressora.

Este seria o teste que, na teoria, se prognosticava um resultado negativo, devido à grande taxa de mensagens que seriam enviadas ao mesmo tempo (um número elevado de comandos enviados pelo PC *Host*, que contém a informação que permite a produção da peça e também as instruções de monitorização enviadas automaticamente pela interface). Portanto, o que seria expectável acontecer seria que, durante a comunicação, algumas das mensagens fossem perdidas ou a impressora respondesse com o erro no número de linha ou do *checksum*. O PC *Host* não conseguiria repetir o envio das mensagens (isso deve-se à contagem do número de linha na aplicação de PC *Host* utilizada, o Repetier-Host, e na aplicação do Node-Red serem diferentes, sendo que a contagem que a impressora se baseia é da aplicação do Node-Red criada), o que faria com que fossem perdidas linhas de código G cruciais para a impressão correta da peça.

Desta forma, nas tentativas de impressão efetuadas foram observados os seguintes resultados:

- Após o envio da ordem para imprimir, a impressora começa naturalmente a aquecer a base até à temperatura de *set point*. Este processo, ainda que tenha decorrido pouco tempo após a sua iniciação, termina sem que tenha sido atingida a temperatura pretendida e avança para o aquecimento do extrusor que, à

semelhança do ocorrido com a base, também não aquece totalmente. Estas ocorrências, comprometem todo o processo de impressão 3D faz com que seja inevitável cancelar a produção da peça de forma a que a máquina não seja danificada;

- No caso de se conseguir atingir as temperaturas *set point*, tanto para a base, como para o extrusor, devido à perca de algumas mensagens contendo o código G da impressão, a peça fica incompleta em algumas áreas, devido à falta do código que permitiria completar essas mesmas áreas. Desta forma, a peça obtida será sempre defeituosa.

Assim, pode-se aferir como resultado final deste teste a não possibilidade de impressão de uma peça via PC *Host*.

No entanto, a impressão através do PC *Host* poderá ser realizada caso os pedidos de monitorização automáticos gerados pela aplicação desenvolvida suspenderem aquando da impressão de uma peça. Nesse caso particular, mesmo não obtendo dados de monitorização da impressora para a plataforma, poderá tirar-se proveito das utilidades de uma impressão remota, na qual poderá trazer essa grande vantagem de não ser necessário estar presente junto da impressora para imprimir uma peça.

5.6 Resumo do capítulo

Neste capítulo são apresentados os testes que permitiram a validação do sistema criado com a realização de cinco testes distintos. Foram testadas as capacidades de impressão, tanto por cartão SD como por PC *Host*, o envio de comandos manuais da interface e também do PC *Host* e a taxa de envio de mensagens. Os resultados obtidos corresponderam às expectativas com a exceção do teste de impressão via PC *Host* a ser único com um resultado negativo, não sendo este o caso mais importante. Verificou-se também o funcionamento correto da comunicação entre a aplicação IoT desenvolvida e a plataforma IoT, através dos gráficos obtidos dos dados retirados durante uma impressão.

Assim, é possível aferir que a aplicação desenvolvida cumpriu os objetivos e requisitos mínimos estabelecidos previamente, tornando-a válida para monitorizar e comandar uma impressora 3D.

6 Conclusões e trabalhos futuros

Após a descrição dos desenvolvimentos e testes efetuados, serve o presente capítulo para apresentar as conclusões do projeto desenvolvido e perspetivas de trabalhos que se pretendem desenvolver no futuro.

Com base nas conclusões retiradas, deduz-se quais os requisitos estabelecidos que foram cumpridos e o contributo da aplicação criada para a área de impressão 3D. A partir dos trabalhos futuros, são descritas algumas ideias e funcionalidades que poderiam ser futuramente implementadas na solução desenvolvida de forma a melhorá-la.

6.1 Conclusões

A presente dissertação teve como principal objetivo o desenvolvimento de uma interface que permitisse o comando e monitorização de impressoras 3D em ambientes de produção modernos e abertos, característicos da mais recente revolução industrial, a Indústria 4.0. Após uma análise ao problema, foi adotada uma proposta de solução sugerida pela Instituição onde o projeto decorreu, o INEGI. Desta forma, foram estudados os objetivos e os requisitos que o sistema criado deveria cumprir no final da dissertação, com o fim de a solução desenvolvida permitir a monitorização e comando de impressoras 3D num contexto abrangido pela Indústria 4.0. Os objetivos e requisitos delineados foram cumpridos, obtendo-se uma solução viável.

No estudo do problema das impressoras 3D, identificou-se uma restrição na capacidade de comunicação promovida por este tipo de equipamentos. Estes estão limitados à comunicação série com somente um dispositivo externo. Para cumprir os objetivos e os requisitos foi implementada uma interface entre uma impressora 3D e o seu PC de comando, PC *Host*. Foi necessária também a inclusão no sistema de uma plataforma IoT, para onde dados de monitorização obtidos serão enviados.

A implementação da solução proposta para este projeto esteve assente em quatro partes:

1. Desenvolvimento de meios para a obtenção de comunicação simultânea e bidirecional entre uma impressora 3D e o PC *Host*, e unidirecional no sentido impressora 3D para plataforma IoT;
2. Estudo e desenvolvimento das funcionalidades a implementar e os dados relevantes a comunicar entre a impressora 3D e a plataforma IoT para *firmware* Marlin;
3. Adaptação da solução desenvolvida para *firmware* Marlin ao *firmware* Repetier;
4. Realização de testes que validam a solução para ambos os *firmwares*.

A aplicação desenvolvida no *software* Node-Red permite obter dados da posição e temperatura das impressoras 3D, converte-os no formato JSON e encaminha-os para a plataforma IoT. Possibilita também a comunicação com o PC *Host*, sendo que foram desenvolvidas funcionalidades que proporcionam a gestão de mensagens recebidas na interface, para permitir a comunicação em simultâneo da interface e do PC *Host* com a impressora 3D. Adicionalmente, a estrutura, tanto do próprio *software* Node-Red, como da aplicação desenvolvida, permite a fácil adaptação da solução para diferentes plataformas IoT, *firmwares* de impressoras 3D e PC's *Host*.

Após a realização de testes ao sistema criado, verificou-se para ambos os *firmwares* de impressoras 3D (Marlin e Repetier), que a solução foi desenvolvida com sucesso, visto que uma impressão por cartão SD não foi afetada pela recolha de dados de temperatura e posição, sendo, de seguida, essa mesma informação armazenada na plataforma IoT ThinsBoard. O envio de comandos de controlo ou de configuração para a impressora, tanto do seu PC de comando, como da interface, também foi testado, na qual se constatou a sua execução. Outro dos testes realizados permite quantificar a taxa máxima de mensagens possível de comunicar para cada impressora, onde se verificou que a essa taxa máxima, a monitorização de uma impressora poderá realizar-se corretamente. Apesar de tudo, quando procedido à execução de uma impressão através do PC *Host*, ficou percetível que a solução desenvolvida afetava esse trabalho, o que infere o insucesso do teste.

Desta forma, a solução implementada cumpre os principais objetivos estabelecidos para esta dissertação, permitindo monitorizar remotamente impressoras 3D numa plataforma IoT externa e possibilitando a impressão remota através do PC *Host*, onde para este caso teriam de ser suspendidos os comandos enviados pela interface.

6.2 Trabalhos futuros

Tendo em conta o trabalho desenvolvido, os trabalhos futuros poderão centrar-se não só ao nível da continuidade de desenvolvimentos no *software*, mas também promover a criação de funcionalidades ao nível da plataforma IoT.

Assim, ao nível do *software* desenvolvido sugerem-se os seguintes trabalhos:

- Desenvolvimento de funcionalidades que permitam a impressão via PC *Host*. Poderão ser alterados códigos de determinadas funções e adicionadas novas funcionalidades que permitam à aplicação IoT gerir de uma forma mais viável as mensagens trocadas entre impressora 3D e PC *Host*, e permitir a impressão sem interferências pelo PC *Host*;
- Implementação de outros comandos de código G que permitam a obtenção de dados de outras variáveis da impressora. Esses dados poderão ser, por exemplo, o estado de uma impressão que está a decorrer (percentagem de peça já impressa), tempo que falta até terminar uma impressão ou velocidade de movimentação do extrusor;
- Desenvolvimento de funcionalidades que permitam converter os dados provenientes da impressora para outro modelo de disposição para além do JSON;
- Implementação de comunicação através de outro protocolo de comunicação, como por exemplo, OPC-UA que a nível de dispositivos industriais a sua utilização tenderá a ser *standard*;

- Implementação de comunicação bidirecional entre impressora 3D e plataforma IoT. Este tipo de comunicação irá permitir o desenvolvimento de comandos ao nível da plataforma que permitam o envio de mensagens em função de determinado comportamento adotado pela impressora, que estará a ser monitorizada.

No lado da plataforma IoT, trabalhos que complementam a solução criada nesta dissertação também poderão ser propostos. Funcionalidades que permitem uma maior automatização do sistema poderão ser implementadas e são bastante relevantes durante uma monitorização real de uma impressão. Para esse efeito, a primeira tarefa base será o desenvolvimento de comunicação bidirecional entre impressora 3D e plataforma, em detrimento da comunicação unidirecional existente. Daí poderão surgir outros trabalhos a desenvolver na plataforma que permitem acrescentar outras funcionalidades ao sistema já criado.

Desta forma, poderão ser propostos os seguintes trabalhos a desenvolver na plataforma IoT:

- Envio dos comandos a partir da plataforma, concentrando a aplicação IoT desenvolvida unicamente na gestão das mensagens e possível transformação no seu formato de dados;
- Desenvolvimento de funcionalidades que permitem uma paragem de emergência automática de um trabalho de impressão, quando os valores dos dados obtidos não correspondem aos padrões estabelecidos para a correta produção de uma peça;
- Criação de um sistema que permita a gestão de uma unidade de produção por impressão 3D, como já referido em 2.4.2. Neste sistema seriam delineados e enviados remotamente os trabalhos de impressão a serem executados nas diferentes impressoras dessa unidade. Adicionalmente, todas as impressões poderiam ser monitorizadas na plataforma.

Referências

1. Khorram Niaki, M. and F. Nonino (2017), *Additive manufacturing management: a review and future research agenda.* International Journal of Production Research. **55**(5): p. 1419-1439.
2. Pühringe, M.J. (2018), *Digitalization in the manufacturing sector: Current state, impact on performance and determinants of adoption* Tese de Mestrado em Master of Science. Alpen-Adria-Universität Klagenfurt
3. Lee, E.A. (2008). *Cyber physical systems: Design challenges*. in *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*. IEEE.
4. INEGI (2011). *INEGI - Instituto de Ciência e Inovação em Engenharia Mecânica e Engenharia Industrial*. [consultado em: 24/04/2019]; Disponível em: http://www.inegi.up.pt/.
5. Berman, B. (2012), *3-D printing: The new industrial revolution.* Business horizons. **55**(2): p. 155-162.
6. ALL3DP (2019). *3D Printing Technology Guide: All 10 Types of 3D Printing Technology in 2019*. [consultado em: 15/07/2019]; Disponível em: https://all3dp.com/1/types-of-3d-printers-3d-printing-technology/.
7. Horvath, J. and R. Cameron (2018), *Mastering 3D Printing in the Classroom, Library, and Lab*. Apress.
8. Evans, B. (2012), *Practical 3D Printers: The Science and Art of 3D Printing*. Apress.
9. Jones, M. (2019). *3D Printer Firmware – Which to Choose & How to Change It*. [consultado em: 15/04/2019]; Disponível em: https://all3dp.com/2/3d-printer-firmware-which-to-choose-and-how-to-change-it/.
10. Horvath, J. (2014), *Mastering 3D Printing*. Apress.
11. Efremov, S., et al. (2015). *Cloud IoT Platforms: A Solid Foundation for the Future Web or a Temporary Workaround?* Cham: Springer International Publishing.
12. Li, S., E. Freije, and P. Yearling (2017), *Monitoring 3D Printer Performance using Internet of Things (IoT) Application*.
13. Pereira, E.M. (2018), *Plataforma Industrial de Internet das Coisas.* Tese de Mestrado em Mestrado Integrado em Engenharia Eletrotécnica e de Computadores. FEUP
14. Go, J. (2015), *High-Throughput Extrusion-Based Additive Manufacturing.* em Master of Science. Massachusetts Institute of Technology

15. Schoffer, F. (2016). *How expiring patents are ushering in the next generation of 3D printing*. [consultado em: 19/07/2019]; Disponível em: https://techcrunch.com/2016/05/15/how-expiring-patents-are-ushering-in-the-next-generation-of-3d-printing/?guccounter=2.

16. All3DP (2019). *Best 3D Printing Software Tools*. [consultado em: 18/07/2019]; Disponível em: https://all3dp.com/1/best-free-3d-printing-software-3d-printer-program/.

17. Ultimaker© (2019). *Ultimaker Cura*. [consultado em: 22/07/2019]; Disponível em: https://ultimaker.com/software/ultimaker-cura.

18. Simplify3D© (2019). *Simplify3D Software*. [consultado em: 22/07/2019]; Disponível em: https://www.simplify3d.com/.

19. Repetier (2019). *Repetier-Host*. [consultado em; Disponível em: https://www.repetier.com/.

20. OctoPrint® (2019). *OctoPrint*. [consultado em: 22/07/2019]; Disponível em: https://octoprint.org/.

21. Bell, C. (2014), *Maintaining and Troubleshooting Your 3D Printer*. Apress.

22. RepRap (2019). *List of Firmware*. [consultado em: 14/04/2019]; Disponível em: https://reprap.org/wiki/List_of_Firmware.

23. RepRap (2015). *Sprinter*. [consultado em: 22/07/2019]; Disponível em: https://reprap.org/wiki/Sprinter.

24. Marlinfw (2019). *Marlin Firmware*. [consultado em: 22/07/2019]; Disponível em: http://marlinfw.org/.

25. Repetier (2019). *Repetier-Firmware Documentation*. [consultado em: 22/07/2019]; Disponível em: https://www.repetier.com/documentation/repetier-firmware/repetier-firmware-introduction/.

26. RepRap (2017). *Teacup Firmware*. [consultado em: 22/07/2019]; Disponível em: https://reprap.org/wiki/Teacup_Firmware.

27. RepRap (2019). *RepRap Firmware*. [consultado em: 22/07/2019]; Disponível em: https://www.reprap.org/wiki/RepRap_Firmware.

28. RepRap (2019). *G-code*. [consultado em: 02/08/2019]; Disponível em: https://reprap.org/wiki/G-code.

29. ThingsBoard© (2019). *ThingsBoard - Open-source IoT Platform*. [consultado em: 22/07/2019]; Disponível em: https://thingsboard.io/.

30. JSON (2007). *Introducing JSON*. [consultado em: 23/07/2019]; Disponível em: https://www.json.org/.

31. Butcher, M. (2019). *REST Without JSON: The Future of IoT Protocols*. [consultado em: 01/09/2019]; Disponível em: https://dzone.com/articles/json-http-and-the-future-of-iot-protocols.

32. JSON (2006). *JSON: The Fat-Free Alternative to XML*. [consultado em: 01/09/2019]; Disponível em: https://www.json.org/xml.html.

33. Sałuch, M., et al. (2018). *Raspberry PI 3B+ microcomputer as a central control unit in intelligent building automation management systems*. in *MATEC Web of Conferences*. EDP Sciences.

34. Raspberry Pi Foundation (2018). *Raspberry Pi 3 Model B+*. [consultado em: 19/07/2019]; Disponível em: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/.

35. Stanton, C. (2018). *Raspberry Pi 3 Model B Plus (B+) Technical Specifications*. [consultado em: 19/07/2019]; Disponível em: https://www.element14.com/community/docs/DOC-88853/l/raspberry-pi-3-model-b-plus-b-tech-nical-specifications.

36. Ahmad, N. (2018), *Beacon Reader Simulator Software Implementation on Raspberry Pi.*

37. Morrison, J.P. (2010), *Flow-based Programming: A New Approach to Application Development*. J.P. Morrison Enterprises.

38. Bacon, J. and T. Harris (2003), *Operating Systems: Concurrent and Distributed Software Design*. Addison-Wesley.

39. Masek, P., et al. (2016), *Implementation of True IoT Vision: Survey on Enabling Protocols and Hands-On Experience.* International Journal of Distributed Sensor Networks. **12**(4): p. 8160282.

40. Grgić, K., I. Špeh, and I. Heđi (2016). *A web-based IoT solution for monitoring data using MQTT protocol*. in *2016 International Conference on Smart Systems and Technologies (SST)*.

41. Naik, N. (2017). *Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP*. in *2017 IEEE international systems engineering symposium (ISSE)*. IEEE.

42. Soni, D. and A. Makwana (2017). *A survey on MQTT: a protocol of internet of things (IoT)*. in *International Conference On Telecommunication, Power Analysis And Computing Techniques (ICTPACT-2017)*.

43. Cunha, M.J.d., et al. (2016). *Proposal for an IoT architecture in industrial processes*. in *2016 12th IEEE International Conference on Industry Applications (INDUSCON)*.

44. Wagle, S. (2016). *Semantic data extraction over MQTT for IoTcentric wireless sensor networks*. in *2016 International Conference on Internet of Things and Applications (IOTA)*.

45. HiveMQ (2015). *MQTT Topics & Best Practices - MQTT Essentials: Part 5*. [consultado em; Disponível em: https://www.hivemq.com/blog/mqtt-essentials-part-5-mqtt-topics-best-practices/.

46. Bellavista, P. and A. Zanni (2016). *Towards better scalability for IoT-cloud interactions via combined exploitation of MQTT and CoAP*. in *2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*. IEEE.

47. Forouzan, B. (2009), *TCP/IP Protocol Suite*. McGraw-Hill Higher Education.

48. Caldas, H. (2015), *Smart Home - Smart Appliances.* Tese de Mestrado em Mestrado Integrado em Engenharia Eletrotécnica e de Computadores. FEUP

49. Vandendriessche, C. (2017), *PLC virtualization for an agile Industry 4.0 compliant distributed system.* Tese de Mestrado em Master of Science in Information Engineering Technology. Faculty of Engineering and Architecture, Ghent University

50. Giang, N.K., et al. (2015). *Developing iot applications in the fog: A distributed dataflow approach*. in *2015 5th International Conference on the Internet of Things (IOT)*. IEEE.

51. Blackstock, M. and R. Lea (2014). *Toward a distributed data flow platform for the web of things (distributed node-red)*. in *Proceedings of the 5th International Workshop on Web of Things*. ACM.

ANEXO A: Análise de uma peça impressa

Durante a dissertação foi explicado que a obtenção de dados permite a monitorização e a análise do processo de impressão 3D. Sendo assim, a partir de uma peça impressa e o seus respetivos gráficos de posição e temperatura em função do tempo de impressão poderá efetuar-se uma análise básica, mas que elucida de que forma poderá ser avaliado o processo a partir da aplicação IoT desenvolvida.

Na peça utilizada para testar a aplicação IoT é possível destacar quatro etapas de impressão distintas, sendo estas observáveis na Figura A.1.

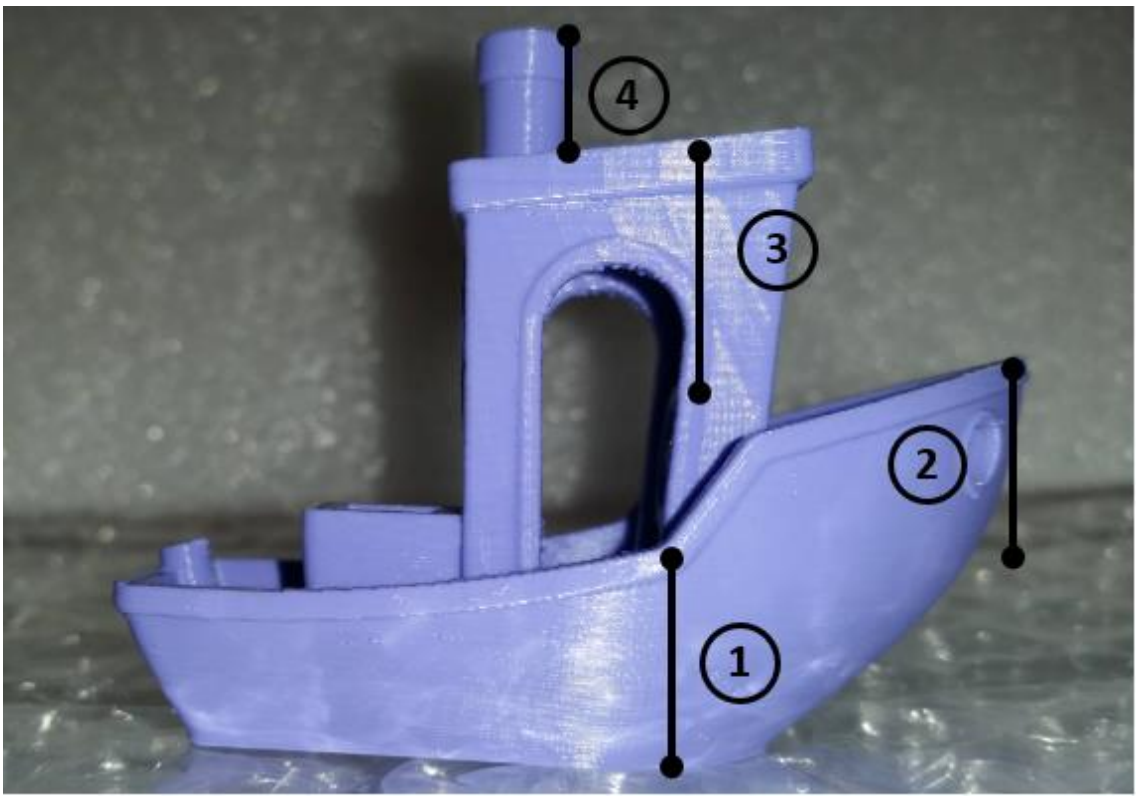


Figura A.1 - Quatro etapas da impressão da peça "3DBenchy"

Tendo por base a divisão da peça pelas quatro partes visíveis na Figura A.1 poderão associar-se estas mesmas aos diferentes intervalos de tempo no gráfico de posição obtido no teste exposto no capítulo 5.1. Esta associação está representada na Figura A.2. A partir desta associação, são conhecidos os intervalos de tempo que corresponde à impressão de cada parte e, daí efetuar uma outra associação observável na Figura A.3 que permite ligar os intervalos de tempo de cada parte no gráfico da temperatura.

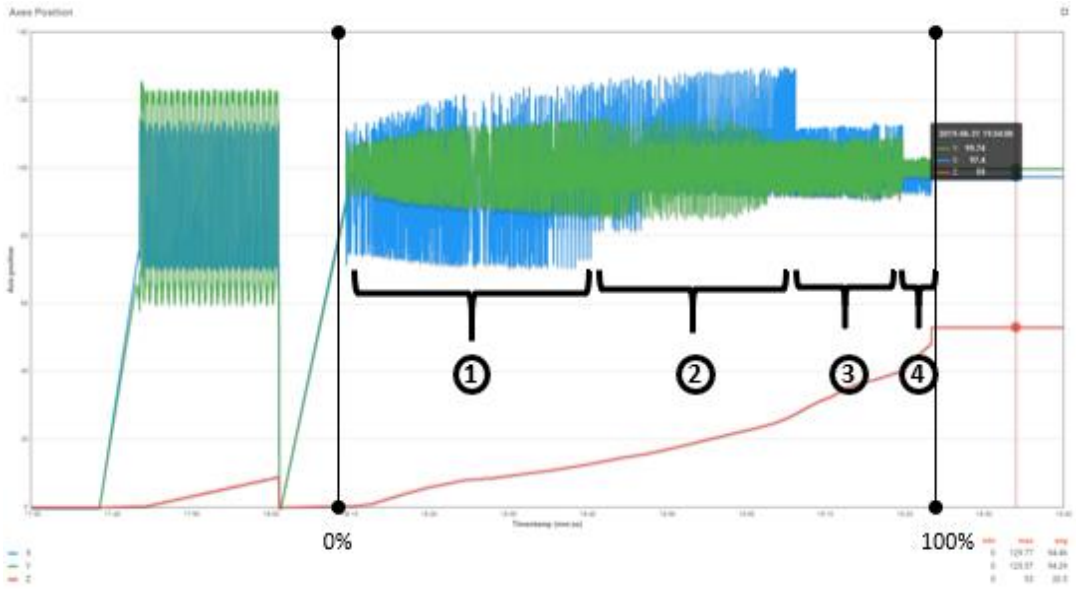


Figura A.2 - Associação das partes divididas do “3DBenchy” a intervalos de tempo no gráfico de posição

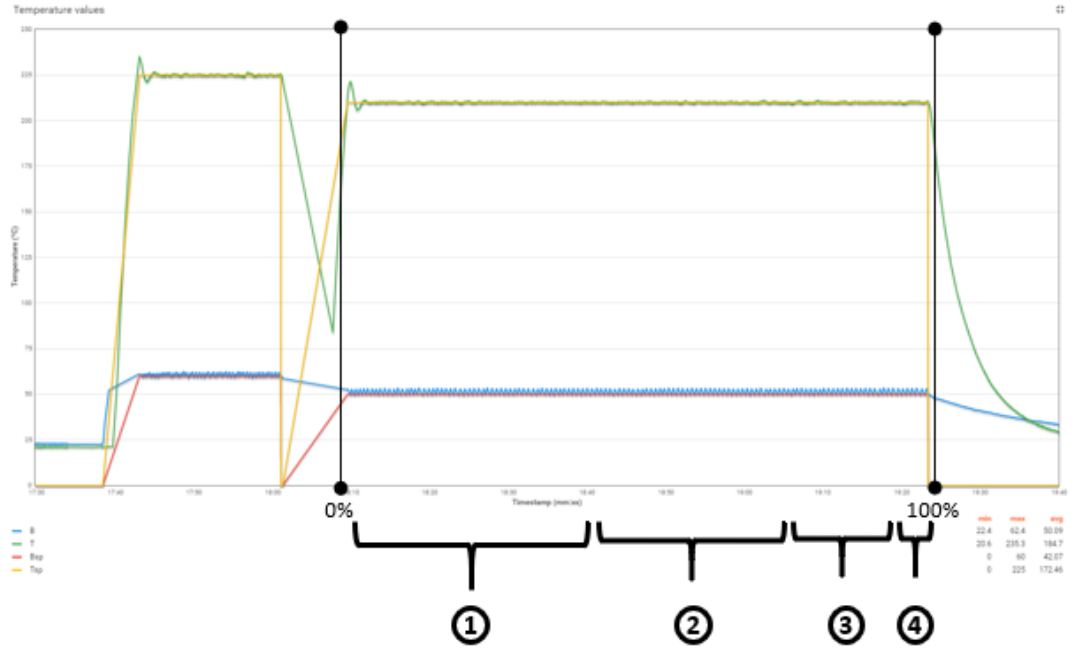


Figura A.3 - Associação das partes divididas do “3DBenchy” a intervalos de tempo no gráfico de temperatura

Com o conhecimento dos intervalos de tempo a que estão associadas a impressão das diferentes partes da peça, poderá realizar-se esta associação também ao gráfico da temperatura.

Numa perspetiva de análise da impressão da peça, estando identificadas estas quatro partes, deverá proceder-se a três passos:

- O primeiro passo a adotar consiste numa verificação de defeitos na forma da peça;
- De seguida, procede-se à identificação da parte da peça onde se encontra o defeito;
- Por fim, com a parte identificada é analisado o intervalo de tempo que está associada essa parte e procurar nos gráficos de posição e de temperatura da peça por um desvio dos dados que habitualmente são obtidos. Especialmente para a temperatura, caso estes

valores não correspondam aos definidos na geração do código G, poderá o material não aderir corretamente à camada anterior e comprometer toda a impressão da peça.

Esta possibilidade de análise poderá futuramente tornar-se mais completa à medida que forem implementadas mais funcionalidades na interface.

ANEXO B: Peças e gráficos de impressão restantes

Como mencionado em 5.1, foram testadas impressões no teste desta secção com os dados a serem igualmente obtidos na plataforma IoT e a confirmar o correto funcionamento da interface para impressões através do cartão SD.

Assim, na Figura B.1 é observável a peça “3DBenchy” impressa pela impressora Stacker com a mesma lapiseira mostrada na peça da Figura 5.3.

Figura B.1 - "3DBenchy" imprimido pela impressora Stacker

Os gráficos que confirmam a obtenção de dados para o *firmware* Repetier estão apresentados na Figura B.2 e Figura B.3. Em relação ao gráfico da Figura B.2, as diferenças em relação ao anterior da Figura 5.1 explicam-se pela diferença de posicionamento da peça na base, aquando da configuração dos parâmetros no *Slicer*. No entanto, a variação dos valores do posicionamento extrusor nos diferentes eixos são idênticos, o que permitiu a impressão do “3DBenchy” no seu todo semelhante ao impresso pela Monoprice Marlin 1.0.0 (Figura 5.3).

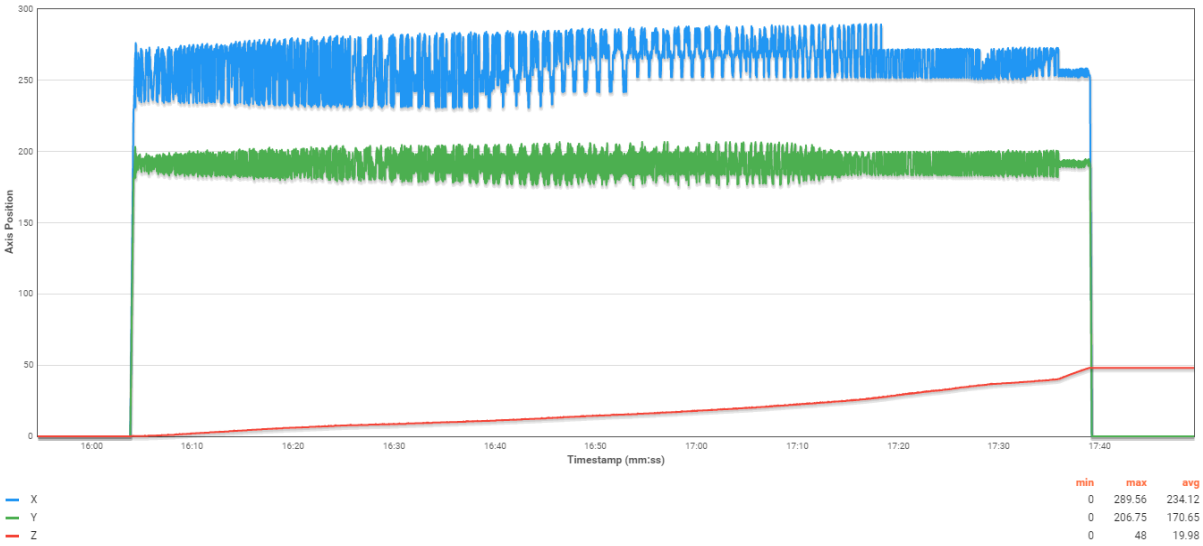


Figura B.2 - Gráfico da variação da posição (mm) em função do tempo em (min) da impressão realizada pela Stacker

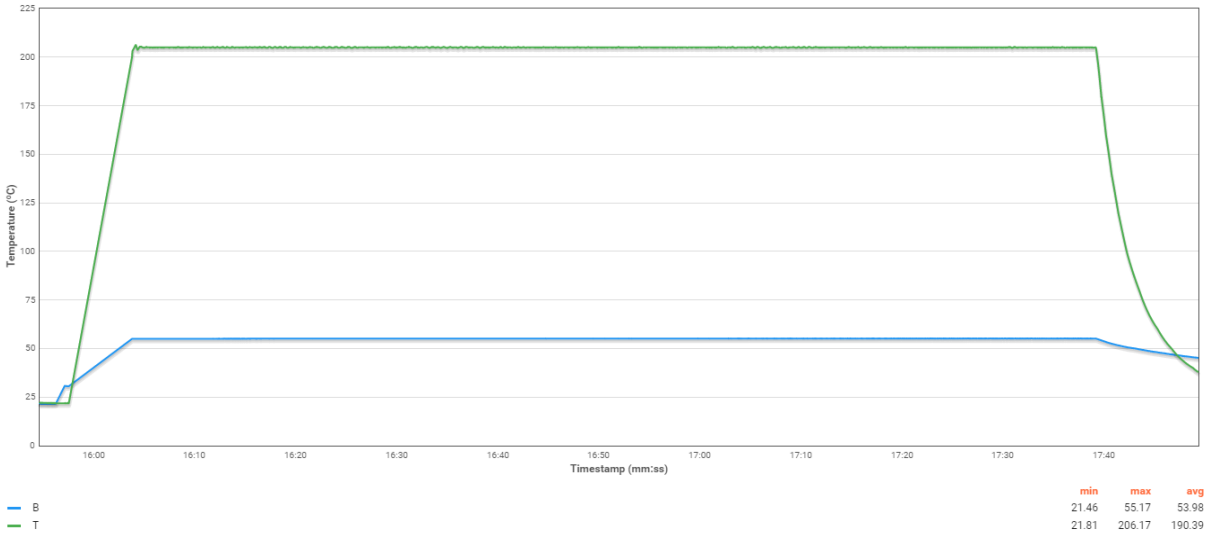


Figura B.3 - Gráfico da variação da temperatura da base e do extrusor (ºC) em função do tempo (min) da impressão realizada pela Stacker

À semelhança das impressoras Monoprice Marlin 1.0.0 e Stacker Repetier 1.0.2, foi também realizada a impressão através da impressora Monoprice Marlin 1.0.9, sendo que nesta não foi obtida peça devido à falta de painel de vidro na base, o que impossibilita esta mesma impressão. No entanto, foi realizada a simulação sem a adição de filamento, tendo sido gerados os mesmos gráficos que confirmam a obtenção dos dados de uma impressão. Estes gráficos são apresentados na Figura B.4 e Figura B.5. Pela sua análise, comparativamente aos gráficos obtidos nas restantes duas impressões, os valores apresentados são idênticos o que permite aferir que caso houvesse possibilidade de imprimir com filamento e placa de vidro, a peça seria produzida corretamente.

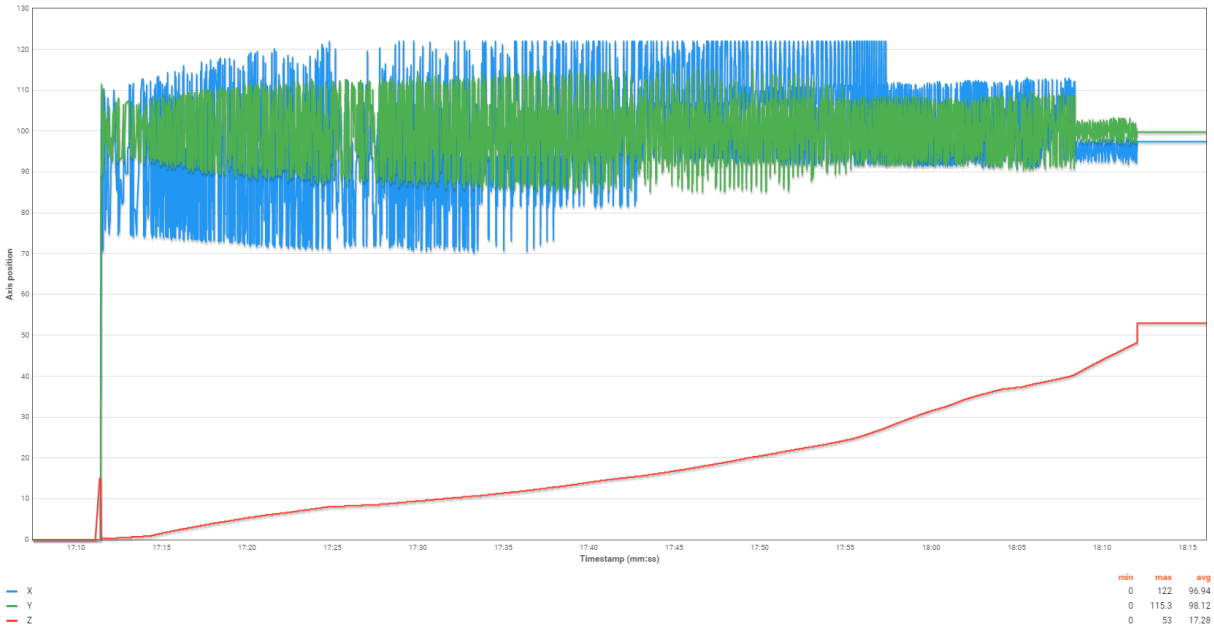


Figura B.4 - Gráfico da variação da posição (mm) em função do tempo em (min) da impressão realizada pela Monoprice Marlin 1.0.9

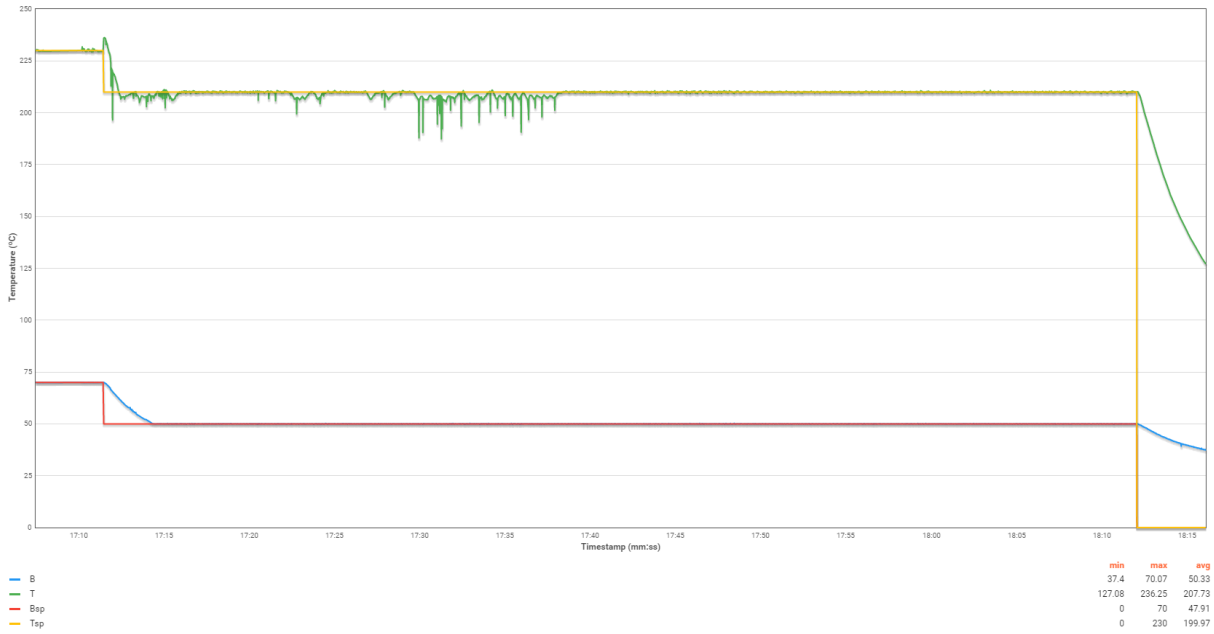


Figura B.5 - Gráfico da variação da temperatura da base e do extrusor (ºC) em função do tempo (min) da impressão realizada pela Monoprice Marlin 1.0.9