

**UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELETROTÉCNICA
ENGENHARIA INDUSTRIAL ELÉTRICA ÊNFASE AUTOMAÇÃO
ENGENHARIA DE CONTROLE E AUTOMAÇÃO**

**LUCAS EDUARDO DE FREITAS
PEDRO HENRIQUE GAIO PACHECO**

**SISTEMA DE MEDIÇÃO E FATURAMENTO DE FILAMENTO PARA
IMPRESSORAS 3D TIPO FDM**

TRABALHO DE CONCLUSÃO DE CURSO

**CURITIBA
2015**

**LUCAS EDUARDO DE FREITAS
PEDRO HENRIQUE GAIO PACHECO**

**SISTEMA DE MEDIÇÃO E FATURAMENTO DE FILAMENTO PARA
IMPRESSORAS 3D TIPO FDM**

Trabalho de Conclusão de Curso apresentada à disciplina de Trabalho de Conclusão de Curso como requisito parcial à obtenção do título de Bacharel em Engenharia pelo Departamento Acadêmico de Eletrotécnica (DAELT), da Universidade Tecnológica Federal do Paraná (UTFPR).

Orientador: Prof. Dr. Amauri Amorin Assef
Co-orientador: Prof. Me. David Kretschek

**CURITIBA
2015**

Lucas Eduardo de Freitas
Pedro Henrique Gaio Pacheco

Sistema de medição e faturamento de filamento para impressoras 3D tipo FDM

Este Trabalho de Conclusão de Curso de Graduação foi julgado e aprovado como requisito parcial para a obtenção do Título de Engenheiro de Controle e Automação, do curso de Engenharia de Controle e Automação do Departamento Acadêmico de Eletrotécnica (DAELT) da Universidade Tecnológica Federal do Paraná (UTFPR).

Curitiba, 25 de novembro de 2015.

Prof. Paulo Sérgio Walenia, Esp.
Coordenador de Curso
Engenharia de Controle e Automação

Prof. Amauri Amorin Assef, Dr.
Responsável pelos Trabalhos de Conclusão de Curso
de Engenharia de Controle e Automação do DAELT

ORIENTAÇÃO

Prof. Amauri Amorin Assef, Dr.
Universidade Tecnológica Federal do Paraná
Orientador

Prof. David Kretscheck
Universidade Tecnológica Federal do Paraná
Co-Orientador

BANCA EXAMINADORA

Prof. Amauri Amorin Assef, Dr.
Universidade Tecnológica Federal do Paraná

Prof. Mariana Antonia Aguiar Furucho, M.Sc.
Universidade Tecnológica Federal do Paraná

Prof. Guilherme Luiz Moritz, Dr.
Universidade Tecnológica Federal do Paraná

AGRADECIMENTOS

Gostaríamos de agradecer a Deus pela criação do universo e de todas as coisas belas do mundo e abundantes da natureza.

Aos nossos amados pais, Aurea e Luiz e Lúdia e Pedro, por nos oferecerem a vida e nos acompanharem desde os primeiros passos com dedicação e cumplicidade únicas, bem como pelas oportunidades oferecidas e a força necessária para chegarmos até aqui.

Às nossas famílias, que de toda forma contribuíram para a conclusão deste trabalho e, em especial, aos nossos irmãos, companheiros de todas as horas. Agradecemos imensamente pela enorme paciência e força.

À Cindi Sayumi Shinohara por todo o apoio oferecido a mim, Lucas, abrindo mão de certos momentos de nosso relacionamento em virtude da realização do objetivo e sucesso deste trabalho. Muitas vezes passamos mais tempo com o companheiro de equipe do que com nossas namoradas, porém toda abdicção sempre foi recebida com enorme carinho e mais incentivo para seguirmos.

À Luana Rutz Passos de Souza por sempre acreditar em mim, Pedro, dando-me apoio e motivação para seguir nesta caminhada. Obrigado por sempre se fazer presente mesmo nos momentos mais difíceis, com grande amor e compreensão. Entramos juntos no ensino superior e completar esta fase estando ao seu lado é de imenso orgulho.

Aos nossos amigos e colegas que também participaram desta jornada e nos mostraram o valor e significado de uma amizade sincera.

Ao nosso professor e orientador Amauri Amorin Assef pela grande dedicação e orientação enriquecedora neste estudo.

Ao nosso professor, co-orientador e amigo David Kretschek pelas oportunidades oferecidas, dúvidas esclarecidas e apoio. Sem o GIP3D este trabalho não seria possível.

Aos professores componentes da banca examinadora, Mariana Antonia Aguiar Furucho e Guilherme Luiz Moritz, pelas importantes observações apresentadas desde o início deste trabalho.

À Universidade Tecnológica Federal do Paraná e seu corpo docente por nos proporcionarem ensino público, gratuito, de qualidade e formação reconhecida.

Ao serviço Temboo pelo suporte oferecido.

RESUMO

DE FREITAS, Lucas Eduardo; PACHECO, Pedro Henrique Gaio. Sistema de medição de filamento para impressora 3D tipo FDM. 2015. 81 f. Trabalho de conclusão de curso (Engenharia Industrial Elétrica, ênfase em Automação e Engenharia de Controle e Automação) – Universidade Tecnológica Federal do Paraná. Curitiba, 2015.

O presente trabalho de conclusão de curso busca delimitar a quantidade de insumo gasto durante o processo de impressão 3D em máquinas de baixo custo, possibilitando a precificação do mesmo a fim de criar uma nova alternativa para os atuais processos de análise de material gasto. Nova opção esta que se dedica a superar falhas pontuais existentes tanto no método da pesagem quanto no método de análise do *gcode*. O sistema tem como interface com o usuário uma aplicação para iOS desenvolvida exclusivamente para este trabalho, que conta também com um servidor na nuvem para armazenamento dos dados capturados pelo *encoder* e Arduino, sendo estes os principais *hardwares* estudados. Dessa forma, criou-se um processo preciso, exato e robusto para uma cobrança justa e real das impressões realizadas para a comunidade da UTFPR.

Palavras chave: Faturamento. Impressora 3D. Arduino. iOS. Encoder. UTFPR. GIP3D.

ABSTRACT

DE FREITAS, Lucas Eduardo; PACHECO, Pedro Henrique Gaio. Measurement and Billing Filament System for FDM 3D Printers. 2015. 81 f. Trabalho de conclusão de curso (Engenharia Industrial Elétrica, ênfase em Automação e Engenharia de Controle e Automação) – Universidade Tecnológica Federal do Paraná. Curitiba, 2015.

This final work of graduation looks for delimit the amount of spent material during the 3D printing process on low-cost machines, enabling the pricing of it in order to create a new alternative to the current process of spent material analysis. This new alternative is dedicated to overcoming existing punctual failures both in the weighing method as in gcode analysis method. The system has, as user interface, an application for iOS developed exclusively for this work, that also includes a cloud based server to store data collected by the encoder and Arduino, which are the main hardware studied. In this way, we have developed a precise, accurate and robust process for a fair and real charging of prints made for the UTFPR community.

Keywords: Billing. 3D Printing. Arduino. iOS. Encoder. UTFPR. GIP3D.

LISTA DE FIGURAS

Figura 1 – Representação simplificada do sistema FDM.	18
Figura 2 – Impressora 3D RepRap Prusa i3.....	20
Figura 3 – Impressora 3D RepRap Prusa Mendel.....	20
Figura 4 – (a) Impressora 3D presente no GIP3D e (b) rolo de filamento ABS.	21
Figura 5 – Ambiente no qual o GIP3D está situado.....	21
Figura 6 – Arduino Mega 2560.	24
Figura 7 – <i>Shield</i> Ethernet.....	24
Figura 8 – Detecção fotoelétrica.....	26
Figura 9 – Representação da geração de pulsos no encoder de quadratura.....	26
Figura 10 – Encoder absoluto 3 <i>bits</i>	27
Figura 11 – Arquitetura do iOS.	28
Figura 12 - <i>Encoder</i> Sparkfun modelo COM-09117.....	33
Figura 13 – Encoder OMRON modelo E6R2-CWZ3E.....	33
Figura 14 – Tela de componentes para instalação.	37
Figura 15 – Visão inicial da IDE do Arduino.	38
Figura 16 – Adição de uma biblioteca para o Arduino.	39
Figura 17 – Representação do circuito de teste com o <i>encoder</i> mecânico.....	40
Figura 18 – Suporte e mecânica do <i>encoder</i>	41
Figura 19 – Detalhe da fixação central e da cobertura de cola.	42
Figura 20 – Tela inicial Parse.com.....	45
Figura 21 – Tela de cadastro Parse.com.	46
Figura 22 – Tela de criação de classes Parse.com.	46
Figura 23 – Classes criadas no Parse.com.	47
Figura 24 – Tabela da classe LoggedUser completa no Parse.com.....	47
Figura 25 – Tabela da classe Print completa no Parse.com.....	48
Figura 26 – Tela inicial Temboo.com.	49
Figura 27 – Tela de configuração Choreo <i>CreateObject</i> Tembo.com.	50
Figura 28 – Xcode disponível para <i>download</i> na AppStore.....	52
Figura 29 – Nova Single View Application no Xcode.....	52
Figura 30 – Telas criadas no StoryBoard dentro do Xcode.	53
Figura 31 – Tela de <i>login</i> App iOS.....	55
Figura 32 – Tela de acompanhamento impressão atual App iOS.....	56

Figura 33 – Tela com histórico de impressão da impressora App iOS.....	57
Figura 34 – Tela de usuário App iOS.	57
Figura 35 – Resultado obtido pelo autor do modelo.	59
Figura 36 – Resultados obtidos pela equipe em duas provas distintas.	59
Figura 37 – Xícara grande.....	61
Figura 38 – Xícara pequena, à direita da xícara grande à esquerda.....	61
Figura 39 – Chaveiro '#1 DAD'.....	62
Figura 40 – Alien 1 à esquerda e Alien 2 à direita.	63
Figura 41 – Tiranossauro Rex.	63
Figura 42 - Diagrama funcional do sistema.....	73

LISTA DE TABELAS

Tabela 1 – Características do servidor na nuvem Parse.com.....	36
Tabela 2 – Funções básicas utilizadas biblioteca <i>Rotaryencoder.h</i>	40
Tabela 3 – Resultados experimentais da impressão de peças diversas.....	64
Tabela 4 – Testes de repetição.....	64
Tabela 5 – Componentes e custos aproximados.....	65

LISTA DE ABREVIATURAS E SIGLAS

ABS	Acrilonitrila Butadieno Estireno
CAD	<i>Computer Aided Design</i>
CPU	<i>Central Process Unit</i>
DADIN	Departamento Acadêmico de Desenho Industrial
DAELT	Departamento Acadêmico de Eletrotécnica
DAFIS	Departamento Acadêmico de Física
E/S	Entrada/Saída
FDM	<i>Fused Deposition Modeling</i>
GIP3D	Grupo de Impressão 3D
GPS	<i>Global Positioning System</i>
HTTP	<i>Hypertext Transfer Protocol</i>
IDE	<i>Integrated Development Environment</i>
LCD	<i>Liquid Crystal Display</i>
LED	<i>Light Emiting Diode</i>
LOM	<i>Laminated Object Manufacturing</i>
OS	<i>Operational System</i>
OSI	<i>Open Source Initiative</i>
PC	<i>Personal Computer</i>
PET	Politereftalato de etileno
PLA	Ácido Polilático
PPSF	Polifenilsulfona
RA	Registro Acadêmico
RP	<i>Rapid Prototyping</i>
RPM	Rotações por Minuto
SD	<i>Secure Digital</i>
SLA	Estereolitografia
SLS	<i>Selective Laser Sintering</i>

SML	Extensão para formato de arquivo eletrônico '.sml'
STL	Extensão para formato de arquivo eletrônico '.stl'
TCC	Trabalho de Conclusão de Curso
TDP	<i>Three Dimensional Printing</i>
USB	<i>Universal Serial Bus</i>
UTFPR	Universidade Tecnológica Federal do Paraná

SUMÁRIO

1 INTRODUÇÃO	12
1.1 MOTIVAÇÃO	13
1.2 OBJETIVOS	13
1.2.1 Objetivo Geral.....	13
1.2.2 Objetivos Específicos.....	14
1.3 PROBLEMA	14
1.4 ESTRUTURA DO TRABALHO	15
2 FUNDAMENTAÇÃO TEÓRICA	16
2.1 PROTOTIPAGEM RÁPIDA	16
2.2 FDM	17
2.3 GIP3D	19
2.4 HARDWARE	22
2.4.1 Microcontrolador	22
2.4.2 <i>Encoder</i>	25
2.5 SOFTWARE	27
2.5.1 iOS.....	27
2.5.1.1 Arquitetura do iOS	28
2.5.2 <i>Web Server</i>	29
2.5.3 <i>Cloud Computing</i>	29
2.5.4 Requisições HTTP	29
3 MATERIAIS E MÉTODOS	31
3.1 PROCEDIMENTOS METODOLÓGICOS	31
3.2 MATERIAS	32
3.2.1 Microcontrolador	32
3.2.2 <i>Encoder</i>	32
3.2.3 Aplicativo para iOS	34
3.2.4 Linguagem de programação	34
3.2.5 Características e funcionalidades da aplicação iOS	35
3.2.6 <i>Web Server</i>	35
3.2.7 Temboo	36
4 DESENVOLVIMENTO DO SISTEMA	37
4.1 PROGRAMA PARA MEDIÇÃO DE FILAMENTO	37
4.2 CONFIGURAÇÃO DO <i>WEB SERVER</i>	45
4.2.1 Temboo	48
4.3 CRIAÇÃO DO APLICATIVO	51
5 RESULTADOS	55
6 CONCLUSÕES	66
REFERÊNCIAS	68
APÊNDICE A – DIAGRAMA FUNCIONAL DO SISTEMA	73
APÊNDICE B – <i>FIRMWARE</i> DO ARDUINO	74

1 INTRODUÇÃO

As impressoras são uma realidade presente na maioria das residências e escritórios. Contudo, a impressão em três dimensões (3D) é um novo tipo que vem se popularizando, a partir da qual se torna possível a materialização física de uma peça modelada em computador, normalmente por aplicações CAD (*Computer Aided Design*).

Em um passado recente, os processos de impressão tridimensional voltavam-se apenas para as chamadas prototipagens rápidas, ou seja, criação de projetos ou peças protótipo, as quais careciam de qualidade. Atualmente, torna-se possível a obtenção de produtos finais, os quais possuem bom acabamento e qualidade, permitindo sua utilização nas mais diversas etapas da fabricação (CARVALHO & VOLPATO, 2007).

Entretanto, as máquinas comerciais de impressão 3D ainda são muito onerosas e de difícil acesso, especialmente no Brasil, ou seja, essa não é uma tecnologia largamente difundida, o que impede sua popularização.

Visando solucionar os empecilhos supracitados, surgem as RepRaps, máquinas de impressão tridimensional de baixo custo, as quais podem ser montadas em casa ou compradas prontas através da Internet, explicitando a tendência atual de simplificação e democratização deste processo. Essas máquinas têm como princípio básico de funcionamento a tecnologia FDM (*Fused Deposition Modeling*), na qual derrete-se um filamento, geralmente de ABS (*Acrylonitrile Butadiene Styrene*) ou PLA (ácido polilático), e realiza-se a extrusão do material fundido de forma controlada e sequencial, camada por camada até se obter a forma final desejada (CARVALHO & VOLPATO, 2007; CAMPOS, 2011).

Partindo da iniciativa de alunos e professores da UTFPR, criou-se o Grupo de Impressão 3D (GIP3D), o qual disponibiliza para toda a comunidade universitária impressoras tridimensionais de baixo custo para uso gratuito e desburocratizado, cobrando-se unicamente o material utilizado durante as impressões. Essa cobrança se deve ao elevado preço dos filamentos de ABS ou PLA, podendo o quilo dos polímeros chegar a R\$ 120,00¹ (cento e vinte reais).

¹ Preço filamento de ABS natural, 2015. Disponível em: <<http://www.filamentos3dbrasil.com.br/abs/filamentos-abs-3mm/1kg/filamento-abs-3mm-para-impressora3d-natural-1-kg/>>. Acesso em: 09 nov. 2015.

Nesse trabalho de conclusão de curso (TCC), foi desenvolvido um sistema mecânico acoplado ao topo da impressora 3D utilizando um *encoder*, a fim de detectar o movimento do filamento consumido. Com isso, um microcontrolador converte os pulsos recebidos do sensor em milímetros que, por sua vez, é convertido em valor monetário e valor de peso. Sendo assim, o controlador comunica-se via Ethernet com o *Web Server*, responsável por disponibilizar todos os dados através de um aplicativo de interface gráfica em sistema operacional iOS. Nesse utilitário, o usuário tem acesso ao seu histórico de impressões, bem como acompanhamento *online* da atividade corrente em relação ao seu custo e medida.

1.1 MOTIVAÇÃO

Sendo a manufatura aditiva uma tecnologia ainda de difícil acesso, foi observada a necessidade da criação de um sistema automático de medição e faturamento do filamento para impressoras do tipo FDM. Tal tarefa tem o objetivo de tornar acessível esse tipo de inovação para a comunidade da UTFPR e evitar burocracias com a transferência de arquivos, tempo de espera, controle da impressão e movimentações financeiras.

O estudo feito durante a realização deste TCC contribuiu de forma significativa no aprofundamento dos conhecimentos sobre o microcontrolador Arduino, enfatizando a programação do *firmware* com as seguintes capacidades: interpretação de sinais vindos de um dispositivo *encoder* (utilizado para detecção do material gasto), configuração de *Web Server* e envio de dados via Ethernet. Além disso, foi desenvolvido um aplicativo *mobile* para iOS.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Criar um dispositivo capaz de mensurar e apresentar ao usuário do sistema a quantidade de filamento utilizada em um processo de impressão 3D por FDM que

servirá de ferramenta para o sistema de faturamento dos protótipos desenvolvidos pelo GIP3D da UTFPR.

1.2.2 Objetivos Específicos

- Estudar a técnica FDM e o funcionamento da impressora 3D RepRap Prusa i3 (disponível no GIP3D);
- Estudar o funcionamento e especificação do *encoder* para medição do material;
- Desenvolver o *firmware* do Arduino para leitura do *encoder* do sistema mecânico e troca de informações com o *Web Server*;
- Pesquisar e configurar um *Web Server* para a aplicação;
- Configurar o serviço para requisições HTTP;
- Programar o *software* do aplicativo iOS;
- Implementar os sistemas junto à impressora;
- Testar e analisar a construção e funcionamento do sistema;
- Estudar a viabilidade econômica do sistema implementado.

1.3 PROBLEMA

Atualmente, existem duas formas de medir e precificar o material utilizado durante a impressão: estimativa via *software* ou pesagem posterior à impressão. Os dois métodos se mostram ineficientes, pois não se considera o material já introduzido no cabeçote de extrusão, o qual é expurgado quando se inicia o processo de aquecimento e, ainda havendo qualquer problema durante o processo, não há contabilização do material inutilizado quando a peça não é finalizada com sucesso. Tanto o *software* quanto a pesagem consideram apenas o material utilizado para a perfeita confecção da peça, não levando em conta processos iniciais ou possíveis erros e desvios durante a impressão.

Partindo dessa premissa, faz-se necessário um método de medição com maior precisão. Buscou-se através desse estudo, a criação de um sistema para

mensurar todo o material utilizado no processo de impressão tridimensional, bem como os custos totais do mesmo. Esse sistema utiliza um *encoder* situado no topo da impressora 3D, o qual detecta todos os movimentos naquele, reconhecendo a quantidade de material gasto. Desse modo, o sucesso da impressão ou a qualidade da mesma tornam-se aspectos subsidiários, tendo em vista que o material fundido não poderá ser reaproveitado, devendo ser igualmente cobrado.

1.4 ESTRUTURA DO TRABALHO

Este trabalho é composto de seis capítulos. O primeiro é destinado à introdução; incluindo-se a motivação deste estudo, objetivos geral e específicos, além de uma visão sobre o problema. O capítulo dois explica detalhadamente através de pesquisa bibliográfica os principais conceitos que envolvem a prototipagem, baseando-se em manufatura aditiva, impressoras 3D, *encoders*, materiais utilizados e conhecimentos sobre sistemas destinados ao gerenciamento geral das tarefas. Os materiais empregados são descritos no capítulo três, cabendo o detalhamento da forma de uso e desenvolvimento destes ao longo da atividade. No quarto capítulo, está descrita a validação do projeto através da utilização da impressora 3D e fabricação de pequenos modelos tridimensionais com intuito de certificar a precisão do sistema. Para avaliação das peças são utilizadas figuras geométricas espaciais, com dimensões bem definidas, disponibilizadas para *download* em diversos endereços eletrônicos. Estas formas são utilizadas em fases de calibração da impressora por terem aspecto simples, sendo facilmente identificado seu tamanho, formato, peso e material consumido. No capítulo cinco, podem ser verificados os resultados obtidos. Por fim, no sexto capítulo, estão enunciadas as principais conclusões provenientes do desenvolvimento deste estudo.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão abordados os pontos relevantes para o desenvolvimento e boa compreensão do trabalho. A divisão do capítulo se apresenta de acordo com o conteúdo aprofundado e o conhecimento necessário para a implementação do sistema.

2.1 PROTOTIPAGEM RÁPIDA

O termo Prototipagem Rápida (*Rapid Prototyping* – RP – em inglês) surgiu próximo ao final dos anos 80 e é usado para definir a forma mais moderna para obtenção de modelos físicos gerados a partir de um sólido virtual em sistema CAD (ROMEIRO FILHO & NAVEIRO, 2011).

Segundo Cooper (2005) o conceito de fabricação em camadas se trata de uma manufatura aditiva capaz de representar aos engenheiros e *designers* suas ideias em três dimensões, ou seja, de forma palpável. Além disso, este processo pode ser elaborado de forma relativamente rápida e barata produzindo modelos funcionais de protótipos.

Campos (2011) compara a prototipagem rápida às formas convencionais de usinagem, em que as peças fabricadas por esta podem levar dias, enquanto na primeira forma podem ser confeccionadas em algumas horas, com maior facilidade para correção de erros e, de certa forma, também antecipando o lançamento de produtos ao mercado. Entretanto, o autor ressalva que esta rapidez é relativa, pois está atrelada à complexidade da forma e da tecnologia aplicada em sua fabricação.

Basicamente, Romeiro Filho & Naveiro (2011) explicam que, para a confecção dos protótipos, parte-se de um *software* de modelagem e um desenho tridimensional. Esta geometria é então dividida em seções transversais – como camadas – por outro programa dedicado a esta tarefa. A máquina de prototipagem deposita o material partindo das camadas gerenciadas pelo *software*, dando origem à peça verdadeira. O objeto é originado pelo empilhamento das camadas, permitindo que formas sejam criadas com alta liberdade em sua geometria (CAMPOS, 2011).

Segundo Carvalho & Volpato (2007) a prototipagem rápida é o agrupamento da manufatura tradicional com as tecnologias mais atuais. Assim, a soma das técnicas serve de base para os principais sistemas de RP existentes hoje.

As diversas técnicas possuem características próprias e são adequadas de acordo com o projeto do produto desejado. As soluções mais comuns e difundidas são:

Estereolitografia (SLA - *Stereolithography*): Baseada na solidificação de camada por camada de uma resina líquida e curada a *laser*;

Fabricação de objeto laminado (LOM – *Laminated Object Manufacturing*): Trabalha com a sobreposição e corte de camadas de um tipo de papel especial;

Modelagem por deposição de material fundido (FDM – *Fused Deposition Modeling*): Técnica em que a peça é processada a partir da extrusão de um filamento plástico;

Impressão tridimensional (TDP – *Three Dimensional Printing*): Camadas de pó cerâmico recebem um jato de aglomerante, dando origem a uma peça cerâmica;

Sinterização a *laser* seletivo (SLS – *Selective Laser Sintering*): Um *laser* de baixa potência pode sinterizar diversos tipos de materiais em pó, sejam estes metálicos, cerâmicos, policarbonatos ou náilon (ROMEIRO FILHO & NAVEIRO, 2011).

Existem ainda outras técnicas que poderiam ser citadas, porém não é o foco do trabalho. O modelo FDM será detalhado na secção seguir, pois é o processo a ser utilizado e que está relacionado ao tema deste TCC.

2.2 FDM

A técnica de manufatura por deposição de material fundido, ou FDM, segundo Liou (2008), tem origem no Arizona a partir de pesquisas da *Advanced Ceramics Research*, com significativo avanço por conta da empresa Stratasys, Inc.

Este processo é baseado na extrusão de um material rígido, na forma de fio, que derrete para tomar a forma do sólido desejado. Os materiais fundidos são termoplásticos de engenharia – ABS, Policarbonato, PPSF (Polifenilsulfona) e outros – em forma de filamento que, quando aquecidos, são extrudados em finas camadas que constituirão o modelo real com resistência suficiente para possíveis testes.

Para chegar à confecção final do produto, as impressoras de FDM se baseiam em alguns modelos de arquivos digitais. Um arquivo em formato '.STL' define o caminho do cabeçote extrusor pelas camadas em que o material será depositado. Este formato é obtido da conversão de um arquivo '.SML' gerado a partir de um objeto virtual tridimensional em CAD (LIOU, 2008).

Todd Grimm (2004) explica como funciona o processo de construção por deposição de material. O filamento de material é derretido para uma forma semi-fundida ao passar pelo cabeçote aquecedor para ser extrudado através do bico, conforme pode ser visto na Figura 1. Cada etapa da extrusão percorre um caminho com espessura definida pelo diâmetro da ponta do bico e pela velocidade da cabeça extrusora.

Uma característica do FDM é a confecção de um modelo de construção contínua, pois passa de uma camada para outra sem, ou quase sem, apresentar interrupção. Liou (2008) complementa que o processo inicia com a entrada do filamento direcionado por polias, que o forçam para dentro do cabeçote de aquecimento para ser fundido. A parte ainda sólida do filamento é responsável por empurrar a parte derretida para fora do bico extrusor.

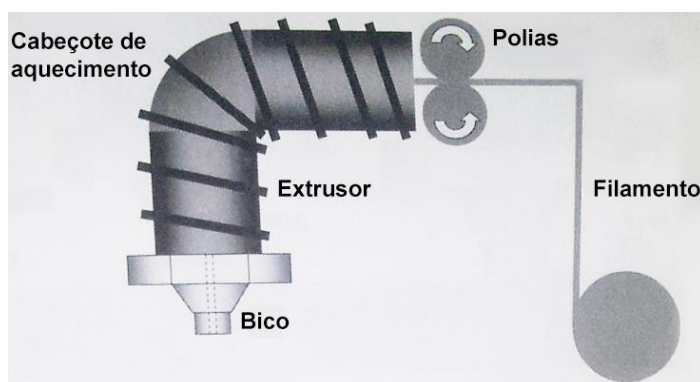


Figura 1 – Representação simplificada do sistema FDM.
Fonte: Adaptado de Liou (2008).

Os princípios do FDM são baseados na química de superfície, energia térmica e tecnologias de manufatura por camadas. As características de flexibilidade e resistência do material, a precisão de posicionamento do bico, seu diâmetro e sua taxa de fluxo volumétrico, a largura das linhas e velocidade de deposição, a temperatura do invólucro e a geometria das peças são aspectos fundamentais para o desempenho satisfatório do processo (CHUA *et al.*, 2010).

O FDM pode ser utilizado em diversas aplicações, sendo algumas delas citadas por Chua *et al.* (2010). Os modelos elaborados podem passar por trabalhos de acabamento e pintura para serem utilizados para apresentação ou conceitos.

Devido à precisão da construção e relativa qualidade no acabamento final, as peças resultantes desta técnica podem ser utilizadas em análises de *design* ou testes funcionais e servem ainda de matriz para confecção de modelos ferramentais para outros processos, como fundição, modelagem ou *vacuum forming*, processo que, segundo Klein (2009), consiste em uma chapa plástica que é aquecida e então aplicada sobre um molde de cavidade. Este contato sela a chapa ao molde. O ar contido na cavidade é evacuado e a pressão atmosférica força a chapa contra os contornos da cavidade.

2.3 GIP3D

Dentro do grupo GIP3D da UTFPR, optou-se por trabalhar somente com impressoras de baixo custo, conhecidas na Internet como RepRaps. Seus projetos são desenvolvidos em comunidade e de fácil modificação.

Existem hoje dois principais modelos em operação atendendo a comunidade acadêmica: RepRap Prusa I3 e RepRap Prusa Mendel. Ambos os modelos são controlados via o sistema Arduino Mega enquanto seus *softwares* supervisórios operam em computadores pessoais, usando princípios muito parecidos de funcionamento e deposição de material, sendo que apenas mudanças na estrutura e montagem são notadas. Na Figura 2 é mostrada uma RepRap Prusa i3 montada, enquanto na Figura 3 uma RepRap Prusa Mendel.

Para o presente trabalho, os esforços foram focados no modelo RepRap Prusa I3, por se tratar de um modelo mais novo e simplificado. Este também foi o modelo que realizou o maior número de impressões ao longo do tempo de operação do GIP3D e, portanto, vem a ser aquele com o qual se tem maior experiência no uso.

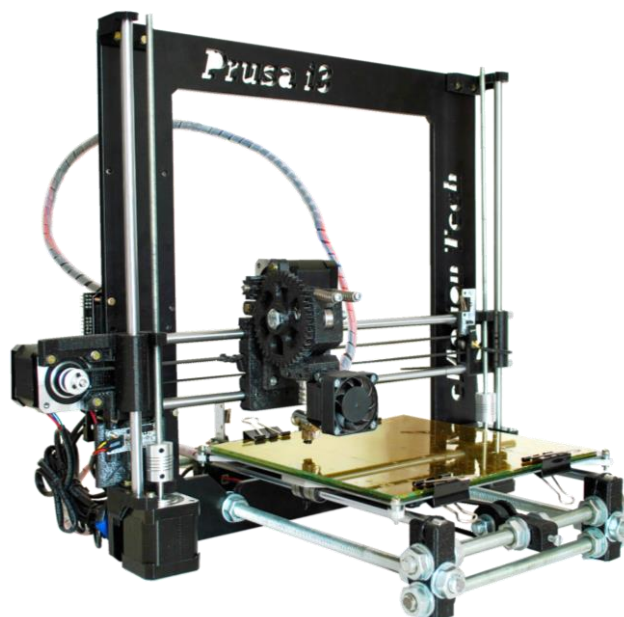


Figura 2 – Impressora 3D RepRap Prusa i3.
Fonte: RepRap (2015).

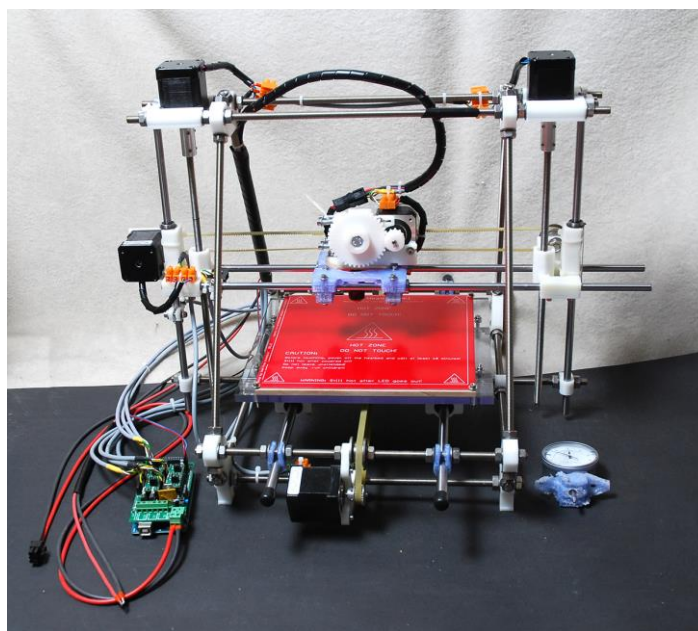
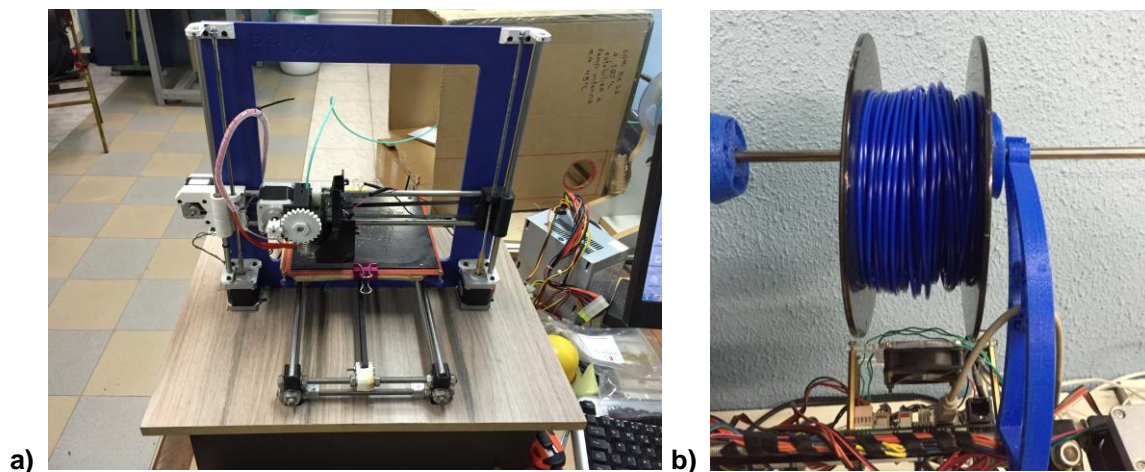


Figura 3 – Impressora 3D RepRap Prusa Mendel.
Fonte: ImageFriend (2015).

Na Figura 4(a) é exibida a impressora existente no GIP3D, enquanto na Figura 4 (b) é mostrado o rolo de insumo ABS para impressão. Para melhor contextualização, na Figura 5 é apresentada a atual sala de trabalho do GIP3D.



**Figura 4 – (a) Impressora 3D presente no GIP3D e (b) rolo de filamento ABS.
Fonte: Autoria própria.**



**Figura 5 – Ambiente no qual o GIP3D está situado.
Fonte: Autoria própria.**

Este é um ambiente novo para o GIP3D, já que sua área foi expandida em relação à sala anterior, que ficava no mesmo local, porém com metade do tamanho e a entrada era feita somente a partir do Departamento Acadêmico de Desenho Industrial (DADIN). Atualmente há uma nova entrada abaixo do bloco G. Os voluntários estão organizando as bancadas e impressoras aos poucos, a fim de torná-lo um ambiente cada vez melhor e mais adequado para todos.

2.4 HARDWARE

2.4.1 Microcontrolador

Os microcontroladores são componentes que apresentam basicamente os principais blocos da arquitetura de computadores, ou seja, CPU (*Central Process Unit*), memórias e interfaces de E/S (entrada/saída). A característica mais importante é a flexibilidade de uso para diversos fins a partir de modificações em *software* e complemento via componentes de *hardware* simples, permitindo soluções compactas, de poucos componentes e mínimo volume (FERREIRA, 1998).

Para o desenvolvimento do sistema proposto foi pensando inicialmente na utilização de um microcontrolador do tipo PIC, da fabricante Microchip Technology Inc, para execução das tarefas de leitura do *encoder* e a troca de informações com um computador através de uma das portas USB (*Universal Serial Bus*). Porém, posteriormente, com adição das funções para o servidor e para o aplicativo e ainda a necessidade de conexão à Internet poder-se-ia até mesmo eliminar a ligação ao computador através do uso de uma plataforma microcontrolada como o Arduino, conhecido também pela grande versatilidade e disponibilidade para aquisição.

Uma plataforma de Arduino pode ser utilizada como base para pequenas unidades computacionais, oferecendo capacidade para sensoriamento e controle, evitando-se o uso de um PC (*Personal Computer*) completo, com gabinete, monitor, teclado, *mouse*, etc., o que possibilita o desenvolvimento de sistemas compactos. A interatividade entre objetos pode ser realizada através de chaves, sensores e outros geradores de sinal conectados às suas entradas e controlando luzes, motores e atuadores através das portas de saídas (ARDUINO LLC, 2015).

O Arduino baseia-se na modelagem de código aberto (*open-source*, em inglês), ou seja, um *software* pode ser desenvolvido, utilizado, alterado e compartilhado livremente por todas as pessoas que assim desejem. A criação pode ser realizada por muitas pessoas, mas a distribuição é regida sob licenças de acordo com a definição de código aberto controlada pela Open Source Initiative (OSI), grupo global sem fins lucrativos dedicado a promover e monitorar *softwares* de código aberto, bem como seu desenvolvimento e seus envolvidos (OPEN SOURCE INITIATIVE, 2015).

Por se tratar de um projeto de código aberto, as pessoas tornam-se livres para a criação de clones e variantes do Arduino. Assim, a partir da placa oficial, surgiram no mercado outros nomes, tais como Freeduino, Seeeduino, Boarduino, Sanguino, Roboduino e muitas variações de “uinos”, que são totalmente compatíveis com a IDE (*Integrated Development Environment*), módulos adicionais – *shields* – e com tudo o que possa ser conectado ou utilizado com a plataforma Arduino oficial (McROBERTS, 2010).

As placas Arduino, segundo McRoberts (2010), são compostas basicamente por um microprocessador Atmel AVR, um cristal oscilador, um regulador de 5 Volts e uma porta USB para conexão ao computador para a troca de dados. Para programá-lo é utilizado o *software* – também livre – Arduino IDE com escrita em linguagem C. A partir desta IDE, escreve-se o programa desejado e então o transfere para o *hardware*, que faz a leitura e passa então à fase de execução, interagindo com o que houver conectado a ele.

No ano de 2010, o primeiro modelo foi lançado oficialmente sob o nome Arduino UNO, durante a feira *Maker Faire NYC*. As versões anteriores eram consideradas *alpha* para testes. A novidade era o resultado de uma evolução curta, mas muito ativa. Outra plataforma lançada na mesma feira, foi o Arduino Mega2560 (BANZI, 2010), que na verdade é a evolução do Mega, apresentado em 2009 (MELLIS, 2009).

A vantagem desta linha – além da quantidade de E/S – é que o Arduino Mega, tinha processador ATmega1280 com 128 kB de memória FLASH para o programa e 8 kB de memória RAM. Entretanto o Mega2560 (Figura 6) foi lançado oficialmente com um processador ATmega2560 com memória de 256 kB, mantendo as demais características similares e garantindo a compatibilidade com *shields* já existentes.

Mesmo diante da versatilidade das placas, em alguns casos pode ser necessário ampliar o *hardware* a fim de se obter mais da plataforma. Opções para isto são os *shields*, que são módulos adicionais contendo diferentes modelagens para dispositivos, com capacidade para funções extras e que são plugados sobre a placa base do Arduino (WHEAT, 2011).

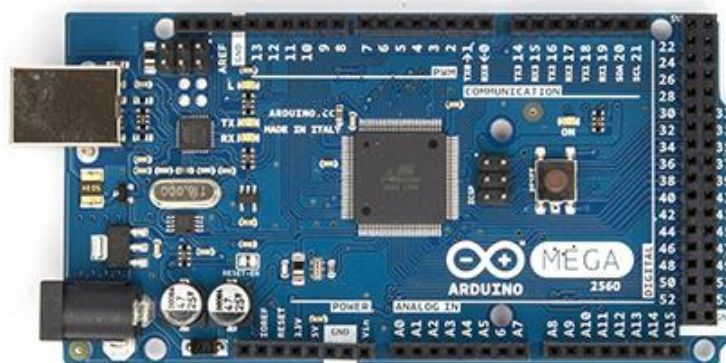


Figura 6 – Arduino Mega 2560.
Fonte: Arduino LLC (2015).

Os *shields* podem ser criados de forma customizada em placas de circuito impresso ou outras, podendo até mesmo serem muito maiores do que a placa do Arduino, e não são necessariamente obrigatórios, pois em muitos casos somente a placa base já fornece capacidade total para os recursos utilizados (WHEAT, 2011).

Existem também as expansões em formatos prontos, fabricados pela marca oficial ou pelos muitos clones, que podem executar inúmeras funções desde uma extensão para as E/S até a adição de tela LCD (*Liquid Crystal Display*) ou ainda permitir a conexão com uma rede Ethernet, por exemplo (McROBERTS, 2010).

Conforme descrito anteriormente, os *shields* adicionam características e capacidade adicionais a uma placa padrão do Arduino. Existe no mercado um *shield*, mostrado na Figura 7, que possibilita conexão Ethernet, usando como base a placa do Arduino Mega 2560.



Figura 7 – Shield Ethernet.
Fonte: Arduino LLC (2015).

O *shield* Ethernet, através da sua porta RJ45 conecta o Arduino à Internet. Além desta capacidade ele também possibilita a leitura de cartão micro-SD. Contudo, não é possível com o mesmo realizar requisições HTTP (*Hypertext Transfer Protocol*) (ARDUINO LLC, 2015).

2.4.2 Encoder

Os *encoders* são equipamentos eletromecânicos que convertem movimentos rotativos ou deslocamentos lineares em pulsos digitais elétricos (S&E, 2014). Para o caso de modelos rotativos a detecção de rotação do eixo pode ser realizada de maneira mecânica, óptica ou magnética, estando estas, respectivamente, em ordem de onerosidade.

Os modelos magnéticos são mais dedicados a aplicações de alta velocidade, portanto sua vida útil é mensurada em milhares de horas para uma velocidade de rotação fixa verificada em rotações por minuto (RPMs) (JONES *et al.*, 2014).

Os modelos ópticos são voltados a aplicações mais refinadas, com velocidades superiores ao mecânico. Por não apresentar contato físico entre as partes, possui grande leveza de movimento do eixo, podendo ser usado para ajustes mínimos em controles analógicos. Entretanto, é muito utilizado como sensores de rotação de motores, chamados então de tacômetro óptico.

É possível ser encontrado com centenas ou milhares de passos/volta, porém seu valor torna-se mais alto também pela maior complexidade de montagem quando comparado ao modelo mecânico (SMITH, 2005).

A conversão dos pulsos digitais elétricos é feita através da detecção fotoelétrica, representada na Figura 8, em que uma série de pulsos é produzida devido à passagem da luz em um disco opaco que contém várias aberturas transparentes igualmente espaçadas. O receptor detecta a luz e a falta dela enviada pelo emissor gerando ondas quadradas, em pulsos digitais (0 e 1).

Dessa forma, existem duas modelagens de *encoders*: incrementais e absolutos (HEISS, 2012).

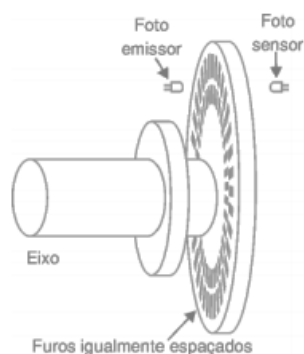


Figura 8 – Detecção fotoelétrica.
Fonte: Heiss (2012).

Os *encoders* incrementais simples são compostos por um único canal de aberturas e detectam a posição linear ou angular de um objeto móvel através da contagem de pulsos elétricos (UNO, 2012). Os *encoders* incrementais de quadratura são compostos por dois canais defasados por uma distância correspondente a metade da largura de uma abertura (BOLTON, 2010).

A Figura 9 pode ajudar a entender como funciona um *encoder* de quadratura, também chamado gerador rotativo de pulso (MAZUROV, 2010). A detecção das duas formas de onda permite definir o sentido de rotação, por exemplo, anti-horário na Figura 9(a), com sinal 00-01-(...), e sentido horário na Figura 9(b), através do sinal 00-10-(...), e assim por diante ao longo do tempo (JONES *et al.*, 2014).

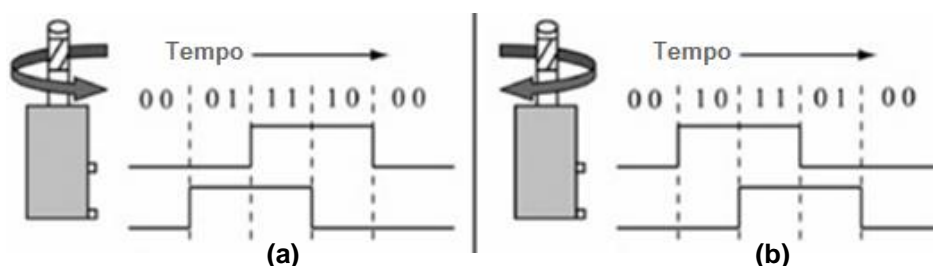


Figura 9 – Representação da geração de pulsos no encoder de quadratura.
Fonte: Jones et al. (2014).

Segundo Heiss (2012), na maioria dos *encoders* incrementais pode ser encontrado um terceiro canal, denominado ponto zero ou ponto absoluto. Este canal é composto por apenas uma abertura e determina a origem do *encoder*. Assim, em caso de queda de tensão, o *encoder* é capaz de retornar à posição zero.

Os *encoders* absolutos possuem sensores fotoelétricos e canais que fornecem saídas em formato de número binário com uma determinada quantidade de dígitos. Na Figura 10 é representado um *encoder* absoluto de 3 *bits* (BOLTON, 2010). Cada posição é representada por um número binário específico, o qual possibilita a detecção da posição angular de um objeto (BOLTON, 2010).

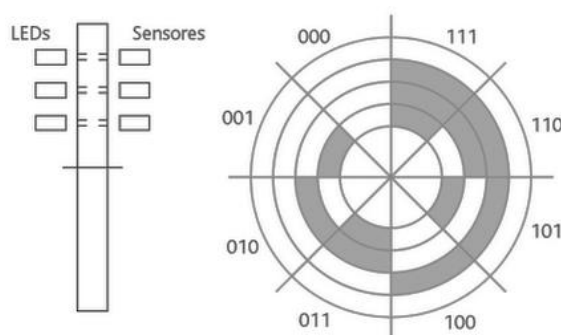


Figura 10 – Encoder absoluto 3 *bits*.
Fonte: Bolton (2010).

2.5 SOFTWARE

2.5.1 iOS

O iOS (do inglês *Operational System* - OS) é o sistema operacional de todos os aparelhos móveis da fabricante estadunidense Apple. Inicialmente o sistema foi criado para atender as necessidades somente do iPhone, mas com sua popularização, atingiu outros dispositivos móveis como iPod, iPad e até mesmo Apple TV (MILANI, 2014).

Este sistema teve desenvolvimento próprio pela Apple e funciona com êxito somente nos *hardwares* móveis produzidos por ela mesma. Apesar de seus dispositivos terem capacidade reduzida, em relação a um computador pessoal da atualidade, o sistema operacional é muito robusto e suporta vários periféricos integrados, como por exemplo, câmera de alta resolução, giroscópio, acelerômetro e GPS (*Global Positioning System*) (MILANI, 2014).

Por se tratar de um OS focado em mobilidade, existem tratativas diferentes daqueles padrões. Ele roda sobre dispositivos que não tem nativamente nem *mouse*

nem teclado. Para tanto, a tela multi toque serve como elemento de *input* básico, podendo, quando necessário, ser utilizado um teclado virtual pelo usuário.

Outra característica bastante peculiar e interessante diz respeito às aplicações, somente uma podendo estar ativa por vez, não havendo possibilidade de duas ou mais rodarem e interagirem entre si ao mesmo tempo (MILANI, 2014).

2.5.1.1 Arquitetura do iOS

O iOS foi concebido em uma arquitetura que o divide em quatro camadas, como observado na Figura 11:



Figura 11 – Arquitetura do iOS.
Fonte: Adaptado de MILANI (2014).

A camada de mais alto nível, aquela que o desenvolvedor interage, é a *Cocoa Touch*. Esta é uma API (*Application Programming Interface*) que dá liberdade para o programador controlar tarefas de comunicação com arquivos, multitoque, interface gráfica, entre outros.

Já a segunda camada, chamada de *Media*, fica responsável por gerenciar áudio, animações, vídeo, focado nas criações gráficas. A penúltima camada é a *Core Services*, e disponibiliza para o programador recursos de grande importância, como a manipulação de arquivos e acesso ao SQLite.

Por fim, a *Core OS* é a camada mais importante, também chamada de núcleo do sistema, sendo, por esta razão, uma camada de difícil acesso. Ela é responsável pelas partes mais críticas, como certificados e gerenciamento de energia e, portanto, deve controlar a comunicação com o ambiente externo de forma mais segura (MILANI, 2014).

2.5.2 *Web Server*

Servidores *Web* são partes essenciais para a estrutura de informação tanto de *sites* quanto de aplicativos. Em curtos intervalos de tempo, eles têm capacidade de modificar aquilo que o usuário acessa através de base de dados e rotinas programadas, trazendo dinamicidade para esses ambientes e, dessa forma, tornando-se mais produtivo e responsivo (SILVA, 2007).

2.5.3 *Cloud Computing*

O termo *Cloud Computing*, ou nuvem, vem a ser utilizado na atualidade para descrever uma enorme rede de servidores, tanto virtuais quanto físicos. Pode ser entendida como a evolução da virtualização, na qual virtualiza-se o próprio *data center* (TAURION, 2009).

Dentre as vantagens do uso da nuvem, destaca-se o fato de uma ilusão de recursos infinitos, extinguindo a necessidade de antecipar valores e ampliações de *hardware*, além de ter um comportamento elástico provocando um uso de recursos por demanda, variando seu volume para se adequar ao momento (TAURION, 2009).

Grandes corporações da Internet, como Google, Amazon e Yahoo, já fazem uso dessa tecnologia e mantêm enormes parques computacionais para manter suas atividades. Este tipo de serviço foi denominado SaaS (*Software as a Service*), que tem como exemplo o editor de texto *online* Google Docs, com grande número de usuários.

Contudo, a denominação também é usada para aplicações de servidor na nuvem, nas quais o usuário não tem a necessidade de configurar o *hardware* envolvido. Deste modo, a tarefa do programador é facilitada através do contrato do servidor na nuvem, podendo focar esforços e investimentos em seu produto (TAURION, 2009).

2.5.4 Requisições HTTP

HTTP é um protocolo de transferência de hipertexto que define os padrões de como a *Web* funciona, sendo o princípio de funcionamento da mesma. Foi

concebido tendo como base requisições e respostas entre cliente e servidor. Através de um cabeçalho cliente faz uma requisição ao servidor que, por sua vez, retorna uma resposta contendo um recurso ou outro cabeçalho (VIEIRA, 2007).

Toda requisição criada deve seguir um método para que esta saiba de qual forma ela deve atuar. Os métodos mais comuns são: GET – existe basicamente para receber informações do servidor; POST – tem como função a criação de recursos no servidor; PUT – atua atualizando algum recurso no servidor; DELETE – de fácil entendimento, deleta um recurso no servidor (VIEIRA, 2007).

3 MATERIAIS E MÉTODOS

3.1 PROCEDIMENTOS METODOLÓGICOS

Como orientação para a equipe durante a execução deste trabalho, foram propostas algumas etapas básicas, seguidas como forma de garantia para o atendimento satisfatório dos objetivos traçados.

1ª etapa: Análise bibliográfica – Nesta etapa inicial foram pesquisados em livros, artigos e páginas eletrônicas os conteúdos relevantes ao tema desta proposta, tais como manufatura aditiva, prototipagem rápida, sensoriamento e *encoders*, sistemas microcontrolados e plataformas baseadas em sistemas *Web*. O conhecimento prévio destas áreas foi de fundamental importância para o decorrer do projeto, bem como para conhecimento técnico dos integrantes.

2ª etapa: Análise de *hardware* e *software* – Nesta divisão do trabalho, foi destinado esforço para conhecimento do *hardware* e a montagem do dispositivo de medição. Juntamente, foi estudada a integração dos sistemas, incluindo a montagem e programação de *software*. Foi utilizado um microcontrolador da família Arduino modelo Mega 2560 em conjunto com o *shield* Ethernet, devido às suas especificações de comunicação.

3ª etapa: Testes de funcionamento – Os testes foram realizados nesta etapa de forma isolada. Após o desenvolvimento do *software* e *hardware* cada sistema foi avaliado quanto ao seu funcionamento. Os resultados foram descritos de forma a ser possível replicação dos mesmos.

4ª etapa: Implementação do sistema – A quarta etapa foi dedicada especialmente à integração das partes realizadas pelo trabalho com a impressora já existente. Foi ainda dedicado tempo para correção de problemas que surgiram nesta etapa, de modo a garantir o perfeito funcionamento de todo o sistema.

5ª etapa: Análises e comprovações finais – A última etapa deste desenvolvimento foi dedicada aos testes do conjunto finalizado. As observações e anotações realizadas nas etapas anteriores já tinham sido solucionadas de modo a possibilitar o funcionamento adequado do sistema. A partir da plena função dos componentes, foi possível rever as análises anteriores e extrair os dados para

certificar a viabilidade do sistema final. Foi comparado ainda o método de medição do filamento com o *encoder* com o método da pesagem com algumas amostras arbitradas.

3.2 MATERIAS

A fim de completar a realização prática deste sistema, a equipe precisou também de alguns materiais físicos e lógicos, além de todo o conhecimento adquirido. A partir deste ponto, foi feito um paralelo entre as partes abordadas no capítulo de fundamentação teórica e a aplicação final desejada.

3.2.1 Microcontrolador

Para o sistema proposto, optou-se pela utilização de um modelo Arduino Mega2560 em função do número de entradas e saídas digitais, capacidade de processamento e pela compatibilidade com o *shield* Ethernet, que possibilita a conexão com rede de Internet via Ethernet.

O principal atrativo pela adoção deste conjunto de componentes é que ambos não possuem preço elevado, sendo assim de fácil acesso. Outro ponto decisivo na montagem diz respeito ao fato da equipe já possuir um Arduino Mega 2560, cedido pelo GIP3D, poupando-se então recursos financeiros nesta fase.

3.2.2 *Encoder*

Dentre as opções de *encoders* de fácil acesso existentes no mercado, muitas podem não ser tão adequadas para o sistema de medição das impressoras 3D. Isto se deve a características construtivas, tamanho apropriado, ou até capacidades ideais para o objeto em estudo, sem esquecer os fatores financeiros que possam tornar a proposta viável ou não.

Os *encoders* mecânicos rotativos de quadratura são populares para plataformas como o Arduino e podem ser tratados como um par de interruptores,

devido ao sinal gerado. Porém, tais componentes dificilmente são encontrados com mais de 32 posições por volta e são dedicados a baixas velocidades (SMITH, 2005). Um exemplo deste tipo de *encoder* é o COM-09117, fabricado pela Sparkfun, apresentado na Figura 12. Este exemplar possui como diferencial a disponibilidade de um botão – tipo *push-button* – juntamente ao eixo, ou seja, ele tem 5 pinos, sendo 3 para o *encoder* e mais 2 para o botão (SPARKFUN, 2008).

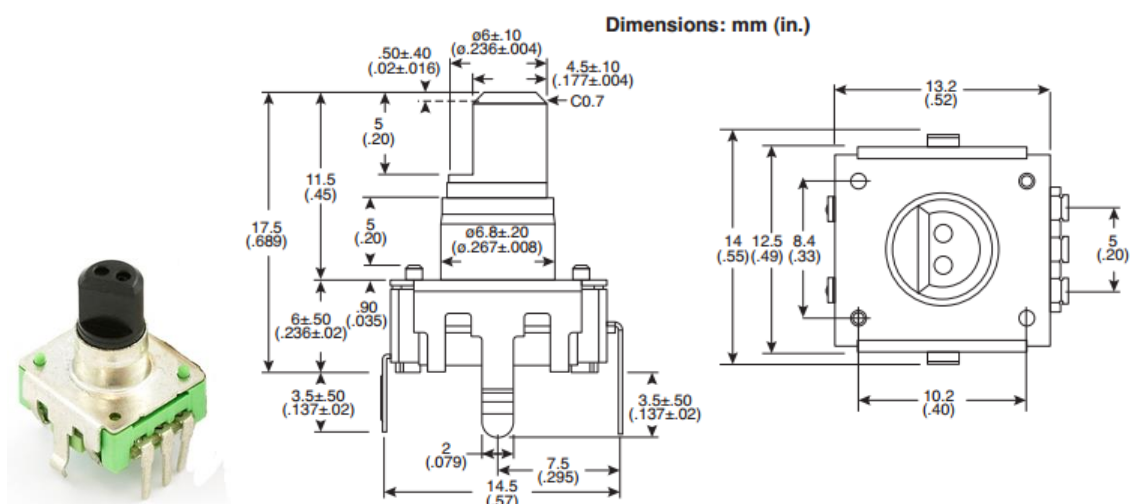


Figura 12 - Encoder Sparkfun modelo COM-09117.
Fonte: Adaptado de Laboratório de Garagem (2012) e Sparkfun (2008).

A fabricante Omron possui em sua linha o modelo E6B2-CWZ3E (Figura 13), que representa um *encoder* rotativo de quadratura com dimensões relativamente compactas para a impressora 3D.

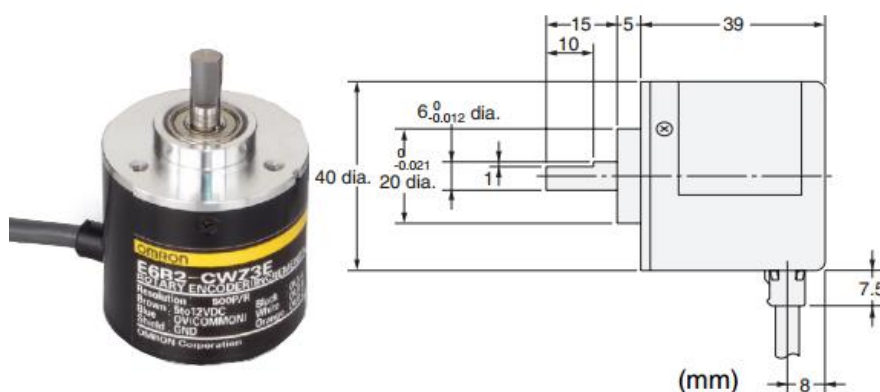


Figura 13 – Encoder OMRON modelo E6B2-CWZ3E.
Fonte: Adaptado de OMRON Co. (2008).

A especificação deste modelo atende 1000 pulsos por rotação, trabalhando com tensão de entrada entre 5 e 12V, capacidade máxima de rotação de 6000 RPM

e possui canais de saída à nível de tensão (OMRON Co, 2008). A partir destas configurações, é possível encontrar no mercado modelos similares de diversos outros fabricantes.

As diferenças de características entre os dois modelos apresentados anteriormente são grandes. Assim, durante fases de testes e implementação foram verificadas as duas opções para finalmente chegar à definição de qual seria o mais adequado. De um lado tem-se um modelo mecânico de construção simples, menor custo e baixa resolução; de outro, tem-se o modelo óptico de quadratura com construção muito mais complexa, alta resolução, mas de maior valor para aquisição.

3.2.3 Aplicativo para iOS

O sistema de medição e faturamento necessita uma interface com usuário. Por esta razão se optou pela utilização de um aplicativo *mobile*, dada a popularização dos *smartphones* no Brasil e outros *gadgets* conectados à Internet.

Através de estudo realizado pela IDC Brasil, constatou-se crescimento de 55% nas vendas de *smartphones* no país em 2014 em relação ao ano anterior, chegando a 55,4 milhões de vendas realizadas no período (IDC, 2015). Desta forma justifica-se a escolha desse modo de interface com os utilizadores.

Sendo o segundo sistema operacional *mobile* mais utilizado, com uma fatia de 14,8% do mercado global (IDGNow, 2015), o iOS – desenvolvido pela Apple – foi escolhido na realização deste projeto devido ao amplo conhecimento dos autores no mesmo.

3.2.4 Linguagem de programação

Quando, em 2008 (APPLE, 2008), a Apple abriu sua loja de aplicativos para desenvolvedores, os *softwares* eram escritos unicamente em Objective-C, linguagem de programação orientada a objeto considerada robusta para a época. Contudo, na conferência de programadores da Apple de 2014 (IBTIMES, 2014), a empresa liberou a opção de desenvolvimento em Swift, linguagem também orientada a objeto,

porém, mais nova, com escrita mais rápida, mais segura e com ganho de desempenho até 2,6 vezes maior em relação ao Objective-C (SWIFT, 2015).

Elencadas as vantagens, optou-se por criar a interface com usuário do sistema de medição e faturamento de filamentos através de uma aplicação para iOS escrita em Swift 2.0.

3.2.5 Características e funcionalidades da aplicação iOS

Com o intuito de facilitar a interação com o usuário, foi desenvolvida uma plataforma simples, fluída e intuitiva. As telas foram desenvolvidas para apresentar somente os dados e informações relevantes, sendo estes: *status* da impressora com atualização periódica, quantidade de filamento gasto em milímetros, peso estimado da peça, tempo de impressão decorrido e histórico pessoal.

Cada usuário tem um *login* no sistema, por meio de um cadastro previamente criado pelos voluntários do GIP3D, o qual possui vínculo com seu registro acadêmico (RA) da UTFPR.

3.2.6 Web Server

Considerando a ideia principal de criar um sistema robusto e de baixo custo, escolheu-se para esse trabalho utilizar como servidor principal de dados *online* o serviço na nuvem do *site* parse.com. Esta é uma aplicação muito utilizada no mundo dos aplicativos. Grandes empresas como Samsung, McDonalds, Dubsplash e Carrefour (<https://www.parse.com/customers>) já disponibilizaram aplicações as quais utilizavam como servidor o SaaS do Parse.com.

O serviço é gratuito suportando grande fluxo de tráfego, muito além daquele esperado para o sistema proposto, conforme apresentado na Tabela 1. Neste trabalho, espera-se que o tempo entre cada requisição seja próximo de 10 segundos.

Tabela 1 – Características do servidor na nuvem Parse.com.

Requisições por segundo	Capacidade Armazenamento	Capacidade Base de Dados	Transferência Máxima de Arquivos
30	20 GB	20 GB	2 TB

Fonte: Adaptado de <http://www.parse.com/plans> (2015).

3.2.7 Temboo

Tendo em vista que o *Shield* Ethernet não tem a capacidade nativa de realizar requisições HTTP, encontrou-se uma forma de transpassar esta barreira com um serviço chamado Temboo (<http://www.temboo.com>), que foi criado para facilitar a comunicação entre sistemas embarcados com serviços providos na Internet. Com este, o *Shield* transmite informações para seu serviço de forma facilitada, que por sua vez realiza todas as requisições HTTP com o *Web Server*. Servindo assim apenas como um intermediário entre o Arduino e o Parse.com.

Existe um limite de 250 requisições por mês no plano gratuito, contudo, após contato com o suporte, ficou garantido 10.000 requisições gratuitas por mês para o GIP3D de forma vitalícia. Considerando-se que uma impressão média usa 100 requisições, podem-se realizar 100 impressões por mês. Atendendo-se assim, as necessidades do grupo até mesmo com folga.

4 DESENVOLVIMENTO DO SISTEMA

Para melhor ilustração, foi elaborado um diagrama de representação das partes do sistema proposto bem como o direcionamento de informações entre elas, conforme é apresentado no Apêndice A – Diagrama funcional do sistema.

4.1 PROGRAMA PARA MEDIÇÃO DE FILAMENTO

Para criação do programa de medição, o primeiro passo deve ser o *download* da IDE do Arduino no *site* oficial dos desenvolvedores. A instalação do programa é bem simplificada, iniciando-se pelo aceite dos termos e condições de uso e pela escolha de quais componentes devem ser instalados, podendo optar pela adição de atalhos, conforme mostrado na Figura 14.

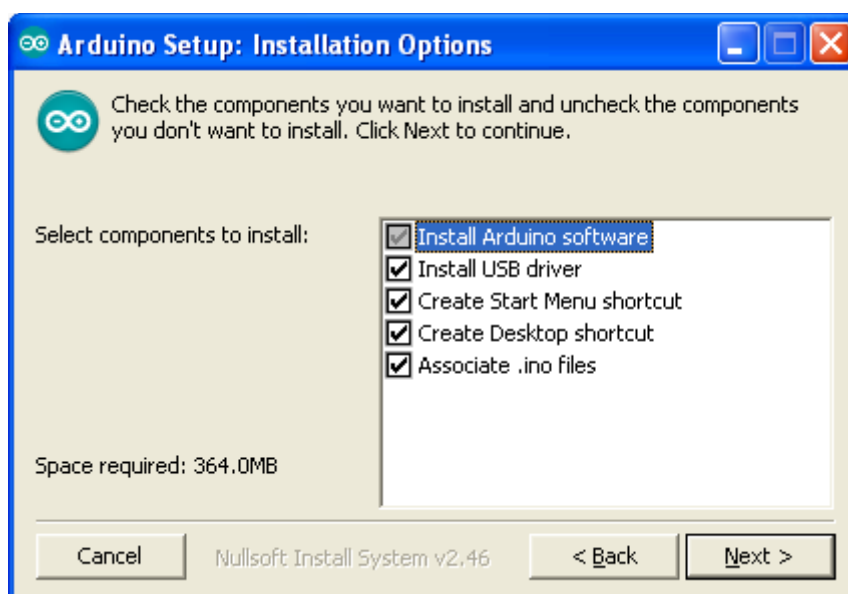


Figura 14 – Tela de componentes para instalação.
Fonte: Autoria própria.

Após esta seleção, o passo seguinte é a escolha da pasta de destino para os arquivos. O programa requer 364 MB disponíveis e Sistema Operacional Windows para a instalação, que leva pouco mais de um minuto para ser finalizada.

Ao executar o programa IDE do Arduino, a primeira visão é como apresentado na Figura 15. É importante conhecer os botões de atalho que ficam na barra superior, sendo dois deles ícones circulares e quatro ícones quadrados. Respectivamente, da esquerda para direita, as funções são:

- Verificar: Compila e verifica a sintaxe do programa e possíveis erros;
- Carregar: Compila e envia o programa ao Arduino para iniciar a execução;
- Novo: Inicia um novo programa com tela limpa;
- Abrir: Carrega um programa já salvo no computador;
- Salvar: Salva a rotina escrita;
- Monitor Serial: Exibição de dados recebidos do Arduino através da Serial.

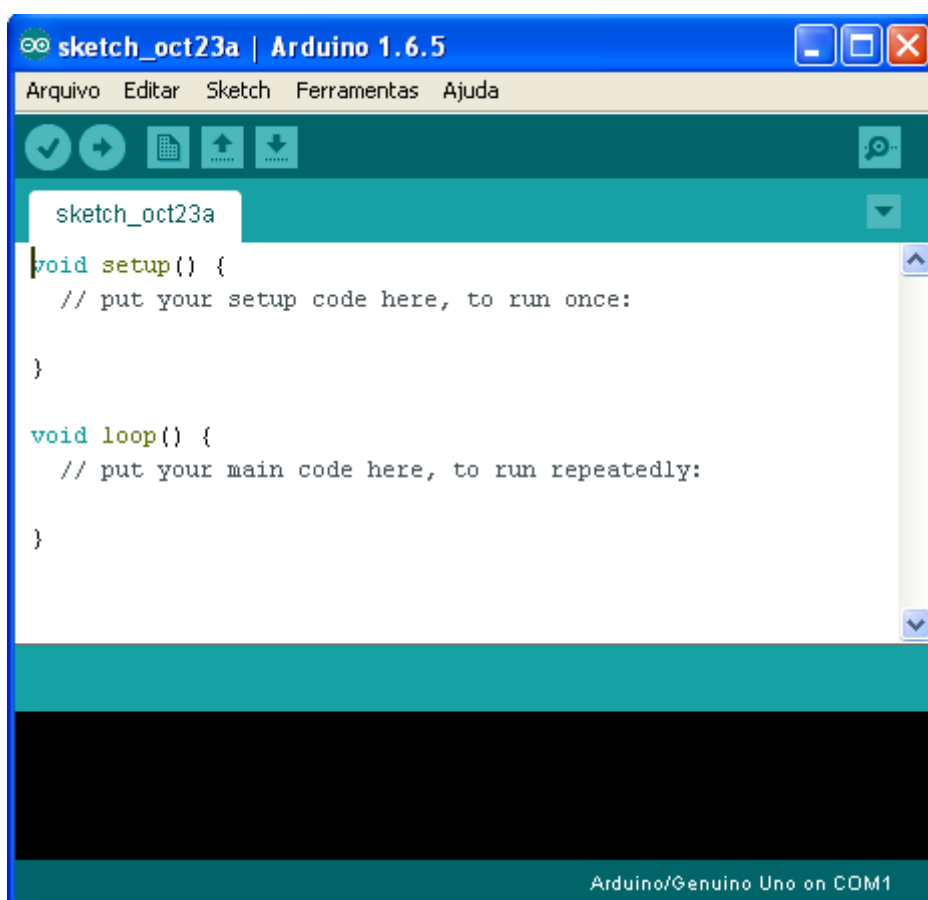


Figura 15 – Visão inicial da IDE do Arduino.
Fonte: Autoria própria.

Antes de iniciar a escrita do programa, é importante indicar o modelo de Arduino que está sendo utilizado. Para tanto, basta acessar na barra superior o *menu Ferramentas*, em seguida **Placa**. No sistema em questão, foi utilizada então a placa **Arduino/Genuino Mega or Mega 2560**. Adicionalmente – para alguns casos – deve-se ainda indicar o processador utilizado acessando novamente **Ferramentas**, mas desta vez no submenu **Processador**, no qual foi escolhida a opção

ATMega2560 (Mega2560). Ainda, deve-se indicar qual porta de comunicação USB está conectado ao Arduino naquele momento. O caminho é simples, basta acessar **Ferramentas**, submenu **Porta** e escolher a porta USB que está conectada ao Arduino – este número varia conforme configurações de *hardware* do computador.

O início da programação do sistema partiu de um rápido teste para certificar que o *encoder* realmente funcionaria. Para esta tarefa foi seguido um tutorial contido no endereço eletrônico Arduino e Cia (2015) que disponibiliza ainda a biblioteca *RotaryEncoder.h* para *download* a ser utilizada pelas linhas de comando. Esta biblioteca deve ser movida para a pasta **Libraries**, dentro da pasta no local de instalação da IDE do Arduino. Para incluir esta biblioteca na IDE, deve-se ir até a barra inicial *menu Sketch*, em seguida **Include Library** e selecionar a opção **Add .ZIP Library**, como apresentado na Figura 16.

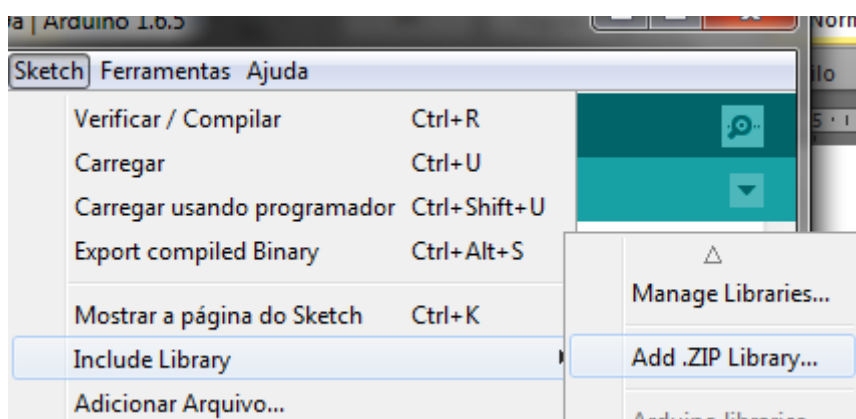


Figura 16 – Adição de uma biblioteca para o Arduino.
Fonte: Autoria própria.

O *encoder* utilizado nesta etapa foi um modelo rotacional mecânico, modelo Keyes KY-040, com resolução de 20 pulsos por volta, encontrado em módulos prontos específicos para utilização com plataformas como o Arduino.

O esquema de montagem foi seguido conforme apresentado na Figura 17. Para ligação do componente, os pinos analógicos **A2** e **A3** foram ligados nos pinos de medição do módulo *encoder* (ARDUINO E CIA, 2015).

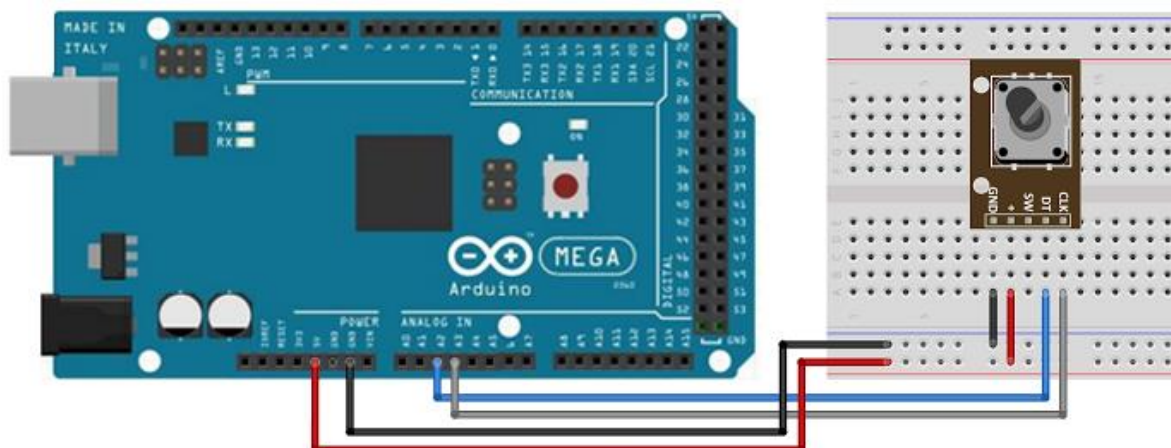


Figura 17 – Representação do circuito de teste com o *encoder* mecânico.
Fonte: Arduino e Cia (2015).

Através da biblioteca *RotaryEncoder.h* e utilizando as suas funções básicas listadas na Tabela 2, conseguiu-se detectar a rotação tanto no sentido horário quanto anti-horário.

Tabela 2 – Funções básicas utilizadas biblioteca *Rotaryencoder.h*.

Função	Descrição
<code>RotaryEncoder encoder(X, Y)</code>	Configura portas de leitura do <i>encoder</i>
<code>encoder.tick()</code>	Inicia período de amostragem do <i>encoder</i>
<code>encoder.getPosition()</code>	Recebe nova posição do <i>encoder</i>
<code>encoder.setPosition(0)</code>	Configura posição do <i>encoder</i> como zero

Fonte: Autoria própria.

Estes modelos poderiam operar em conjunto com a impressora devido às características mais compactas e simples, porém, por serem mecânicos, no momento que está marcando uma posição entre dois pulsos consecutivos, ele não consegue identificar qual posição está e passa a enviar dados errôneos para o Arduino. Este fenômeno se chama *bouncing* e pode ser minimizado com a utilização de capacitores, mas neste caso nunca será eliminado. Este efeito é um impeditivo para a boa precisão do sistema, logo, deve-se recorrer ao uso de *encoder* óptico de quadratura.

Para o *encoder* óptico, pode-se utilizar a mesma biblioteca anteriormente implementada, basta plugar os fios nas mesmas posições equivalentes do *encoder* mecânico. Notou-se que este tem uma precisão muito grande, especialmente para o projeto em questão no qual o modelo E6B2-CWZ3E possui 1000 pulsos por revolução. Definiu-se assim este sensor para a implementação final do sistema.

Com o *encoder* já definido, foi preciso elaborar uma forma de aplicá-lo juntamente com a impressora. Isto foi feito a partir da montagem de uma mecânica para passagem do filamento de forma que não interferisse no seu funcionamento. A partir de uma ideia similar à utilizada para empurrar o filamento para dentro da extrusora, foi criado o mecanismo com o *encoder* agindo de forma passiva no sistema ao invés de um motor que realmente force o material.

Para girar adequadamente, foi colocada no eixo do *encoder* uma coroa reutilizada de uma sucata de impressora comum de papel. A engrenagem possui 38 dentes e aproximadamente 1,15 cm de raio. Este dado é utilizado posteriormente na parametrização inicial do sistema. Nesta coroa foi feita uma cobertura com cola quente para garantir aderência do material do filamento. Ainda, foi adicionado um mecanismo contraposto por mola a esta coroa que garante que o filamento fique sempre no local girando o *encoder*, conforme a necessidade de material no bico extrusor.

Todo este conjunto visualizado na Figura 18 foi confeccionado em placas de PET (Politereftalato de etileno) de 2 mm de espessura e fixado na impressora 3D com placas de madeira na moldura metálica existente na Prusa I3. A opção pela montagem fixa neste local permite que o filamento seja fornecido à extrusora sem que todo o conjunto faça peso na parte móvel.

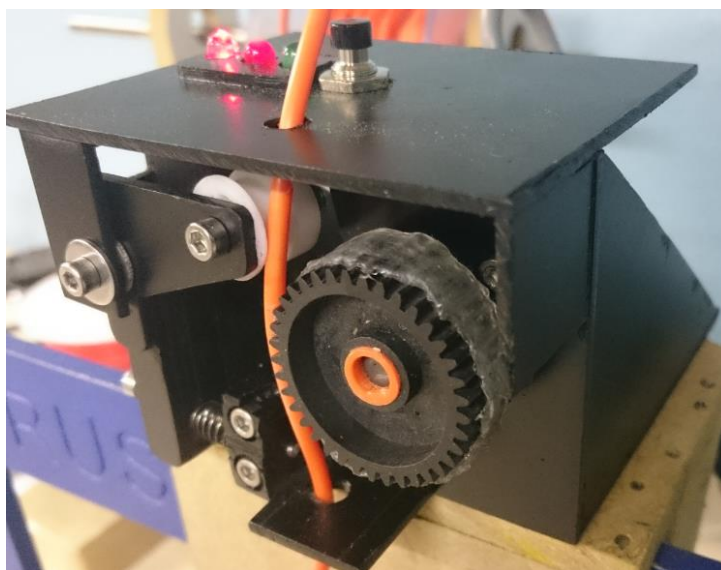


Figura 18 – Suporte e mecânica do *encoder*.
Fonte: Autoria própria.

Adicionalmente, com ajuda dos voluntários do GIP3D, foi criada uma peça na própria impressora 3D que se encaixasse perfeitamente entre o eixo e a roda dentada a fim de evitar folgas e giros falsos do *encoder*. Esta peça pode ser melhor visualizada na Figura 19, assim como a cobertura de cola quente.

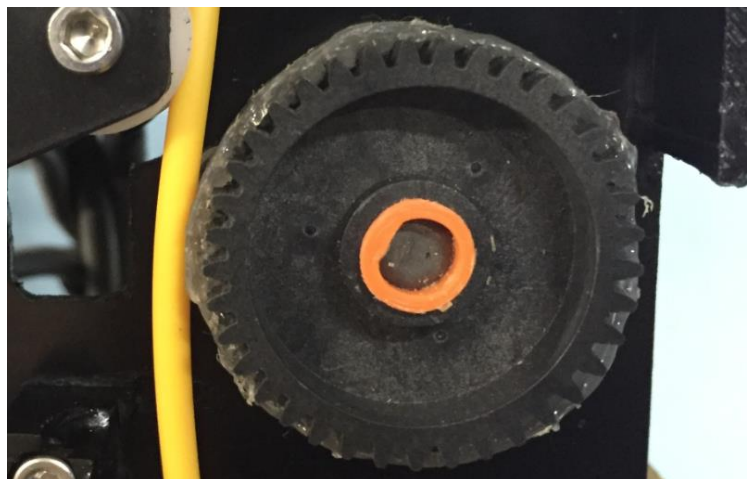


Figura 19 – Detalhe da fixação central e da cobertura de cola.
Fonte: Autoria própria.

A fim de facilitar a interatividade do usuário diretamente com o sistema, foram adicionados 3 LEDs: sendo um deles de cor verde, outro de cor vermelha e ainda um RG (verde e vermelho). Seu modo de operação é bastante simples.

O primeiro LED, RG (verde e vermelho), indica o status da comunicação DHCP. Inicialmente este fica com as duas cores acesas, aguardando a conexão com o servidor e o retorno por um usuário logado. Quando um usuário está logado através do aplicativo, este ficará somente verde. Caso não haja usuário ou o sistema encontre algum problema na conexão, este LED ficará somente vermelho.

O segundo LED, vermelho, é um sinalizador de falhas. Além de indicar o erro da ausência de um usuário, também sinaliza erros na operação ou situações indesejadas de funcionamento, causando a parada do sistema.

O terceiro LED, verde, indica que o sistema está pronto para funcionar e acompanhar uma impressão, ou seja, após o *login* do usuário este se acende para sinalizar que a impressão pode ser iniciada através do *software* de supervisão da impressora 3D.

Adicionalmente foi acrescentado um botão do tipo *push-button* que é basicamente um espelho do botão *RESET* existente nas placas Arduino. Este botão tem a função de apontar para o servidor que uma nova impressão começou, ou seja,

sempre que um novo usuário logar no sistema deve ser pressionado. Assim se dá início à sequência de diagnósticos pelos LEDs.

Para uso do *encoder* junto à programação, deve-se converter seus pulsos de entrada em valores de peso, medida e monetário para ser útil e interpretável para o GIP3D. Todos os cálculos foram implementados no *firmware* do Arduino, seguindo as seguintes equações:

- 1) Parâmetros básicos:
 - a) Taxa de atualização – A cada quantos reais deve-se contar 1 ciclo para enviar ao servidor;
 - b) Bitola – Diâmetro em centímetros do filamento da impressora 3D;
 - c) Valor kg – Preço de 1 kg de filamento;
 - d) PPR Original – Número de pulsos por revolução do *encoder*;
 - e) Divisão PPR – Número que se pretende dividir os pulsos por revolução do *encoder* a fim de diminuir a precisão do mesmo, diminuindo-se assim o número de requisições com o servidor;
 - f) Material – Valor inteiro para escolha do material do filamento: 1 para ABS, 2 para PLA e 3 para material genérico;
 - g) Densidade ABS – Densidade em $\frac{g}{cm^3}$ do ABS;
 - h) Densidade PLA – Densidade em $\frac{g}{cm^3}$ do PLA;
 - i) Densidade Outro – Densidade em $\frac{g}{cm^3}$ de material genérico;
 - j) Raio Acoplado – Raio em centímetros da engrenagem conectada ao eixo do *encoder*.
- 2) Cálculo de quantos gramas de material equivalem a determinado valor em reais configurado na Taxa de Atualização:

$$massa = \frac{(1000 * Taxa\ de\ Atualização)}{Valor\ Kg} \quad (1)$$

A Fórmula 1 é apenas uma regra de três básica relacionando peso com valor monetário.

- 3) Cálculo de quantos centímetros de filamento equivalem a determinado valor em reais configurado na Taxa de Atualização:

$$medida = \frac{massa}{Densidade * \pi * \left(\frac{Bitola}{2}\right)^2} \quad (2)$$

Chegou-se na Fórmula 2 aproximando uma pequena parte do filamento como sendo um cilindro maciço. Para tanto, obtém-se **medida** como sendo a altura deste cilindro. Esta fórmula é o resultado da manipulação das etapas abaixo. Dessa forma, o Volume do cilindro maciço é igual a:

$$V = \pi * \left(\frac{Bitola}{2}\right)^2 * medida \quad (3)$$

Sabe-se que Volume também é igual a:

$$V = \frac{massa}{Densidade} \quad (4)$$

Igualando as Fórmulas 3 e 4, tem-se:

$$\frac{massa}{densidade} = \pi * \left(\frac{Bitola}{2}\right)^2 * medida \quad (5)$$

Reordenando a Fórmula 5, chega-se à Fórmula 2.

- 4) Na sequência foi preciso obter quantos centímetros lineares têm cada pulso do *encoder*. Para tanto, obteve-se os Pulsos Por Revolução (PPR) a serem considerados na conta, conforme os parâmetros básicos configurados anteriormente.

$$PPR = \frac{Pulsos\ Por\ Revolução\ Original}{Divisão\ PPR}$$

$$relação = \frac{2 * \pi * Raio\ Acochado}{PPR} \quad (6)$$

Esta **relação** obtém-se dividindo o perímetro da engrenagem acoplada ao eixo do *encoder* pelo número de pulsos por revolução resultante após a divisão de pulsos para diminuir a resolução do mesmo.

- 5) Cálculo de quantos pulsos do *encoder* equivalem a determinado valor em reais configurado na Taxa de Atualização, sendo chamado de **ref**. Basta dividir o resultado da Fórmula 2 pela Fórmula 6:

$$ref = \frac{medida}{relação} \quad (7)$$

Após todas as conversões calculadas, deve-se enviar os dados para o servidor, que será explicado na seção seguinte.

4.2 CONFIGURAÇÃO DO *WEB SERVER*

Conforme explicado no capítulo 3 do presente trabalho, foi utilizado o serviço de servidor e base de dados do *site* Parse.com. Para iniciar foi necessária a criação de uma conta através do botão *Sign up for free* no canto superior direito, conforme a Figura 20.

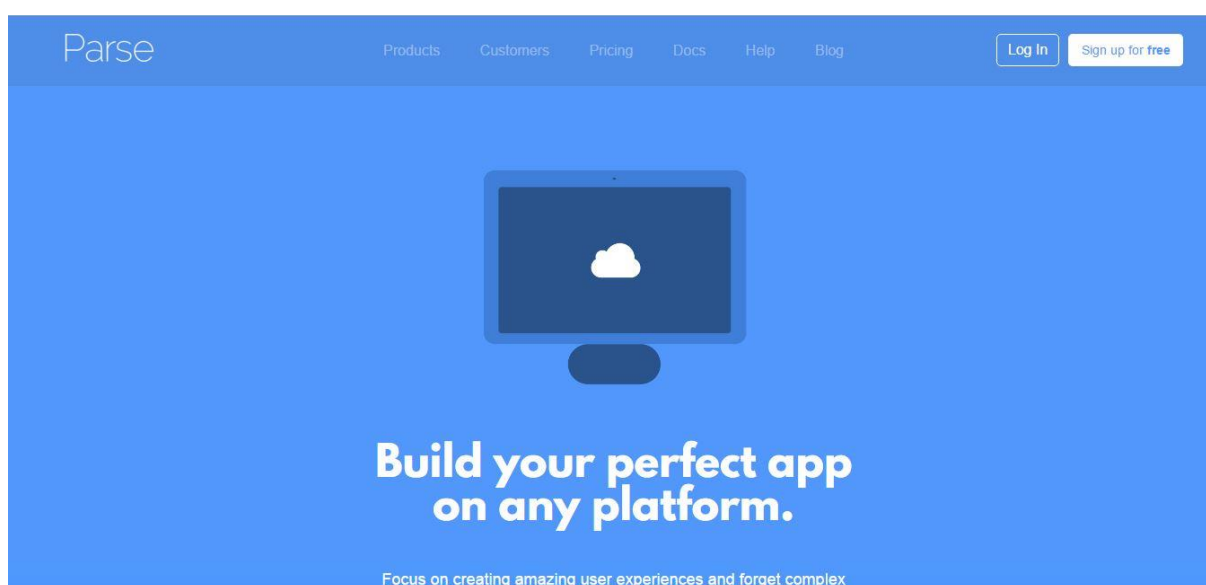


Figura 20 – Tela inicial Parse.com.
Fonte: Autoria própria.

Estando na tela de cadastro, todos os campos devem ser preenchidos. Nesta etapa é preciso dar nome ao projeto dentro do Parse. Neste caso, chamou-se de “TCC2”, conforme a Figura 21.

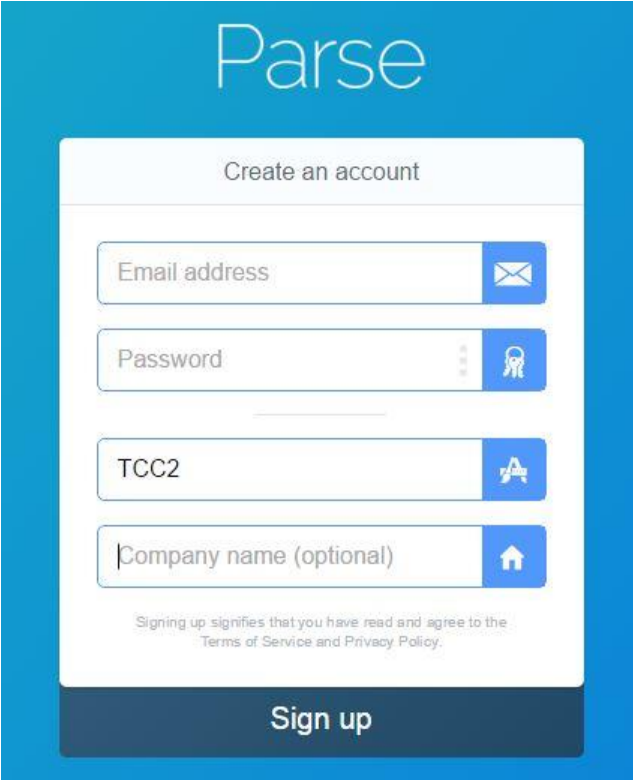


Figura 21 – Tela de cadastro Parse.com.
Fonte: Autoria própria.

Com a conta criada, o passo seguinte foi a geração de classes através do botão *Add Class*, conforme a Figura 22.

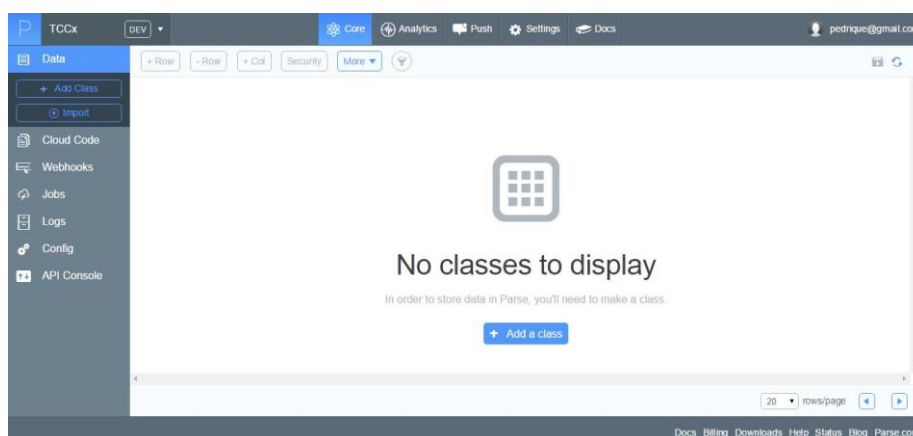
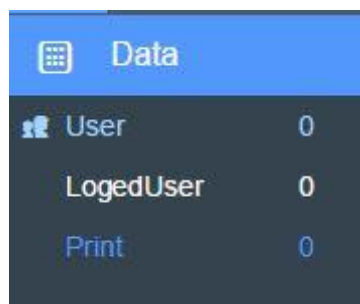


Figura 22 – Tela de criação de classes Parse.com.
Fonte: Autoria própria.

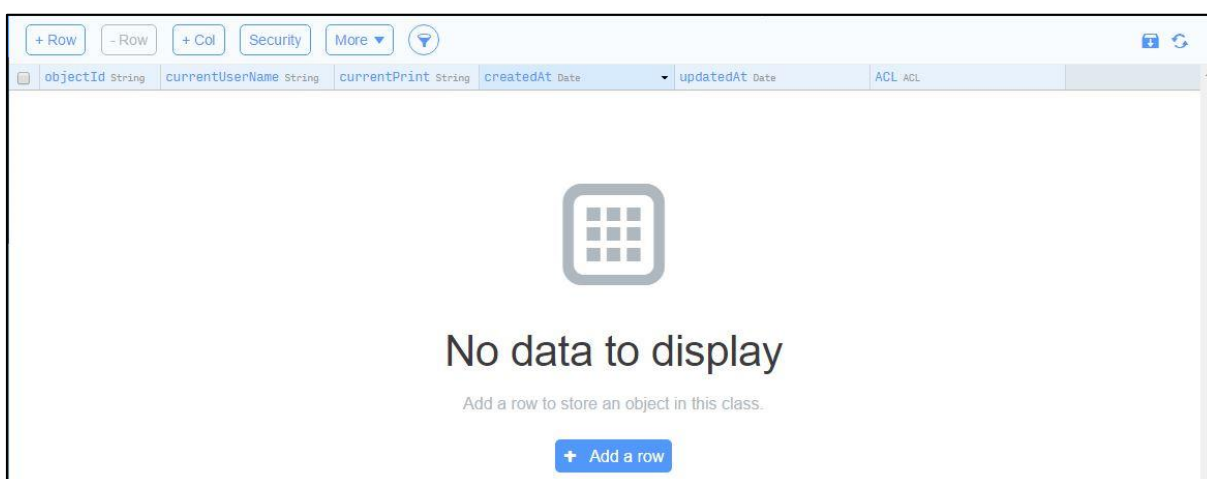
Para a correta divisão do servidor, foram iniciadas três classes. A primeira **User**, do tipo *User*, depois **LogedUser**, do tipo *Custom* e por fim, **Print**, do tipo *Custom*, resultando na tela da Figura 23.



Data	
User	0
LogedUser	0
Print	0

Figura 23 – Classes criadas no Parse.com.
Fonte: Autoria própria.

Com as classes criadas, foi necessário configurar as tabelas. A primeira classe **User**, por ser do tipo *User*, não necessitou nenhuma modificação. Nela tem-se a tabela de todos os usuários cadastrados e suas respectivas senhas. Na classe **LogedUser** foram criadas duas colunas: primeiro *currentUserName*, do tipo *String*, que armazena o usuário atual conectado ao sistema, depois *currentPrint*, também do tipo *String*, que tem a função de definir o código de impressão atual. Isto garante que somente um usuário e uma impressão estejam conectados ao sistema por vez. Resultando-se assim, na tabela de dados apresentada na Figura 24.



objectId	currentUserName	currentPrint	createdAt	updatedAt	ACL
No data to display					

Figura 24 – Tabela da classe LogedUser completa no Parse.com.
Fonte: Autoria própria.

Indo para a classe **Print**, cinco colunas devem ser adicionadas:

- *user*, do tipo *String* para vincular um usuário a impressão;

- *weight*, do tipo *Number* para armazenar o peso atual da peça;
- *length*, do tipo *Number* para salvar o comprimento atual de filamento consumido;
- *cost*, do tipo *Number* para armazenar o custo atual da peça;
- *paid*, do tipo *Boolean* para existir o controle se a peça foi ou não paga.

Com a presente classe, tem-se o controle de todas as impressões realizadas pelo GIP3D. Com todas as colunas criadas, o resultado é o seguinte exibido na Figura 25.

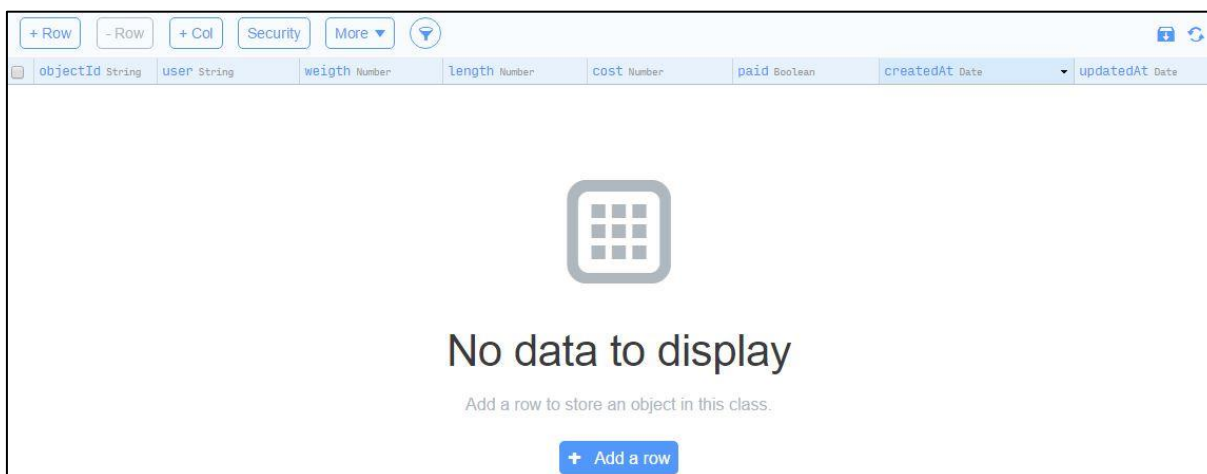


Figura 25 – Tabela da classe Print completa no Parse.com.
Fonte: Autoria própria.

Última etapa a ser seguida no Parse.com é salvar as chaves exclusivas do projeto para habilitar as requisições HTTP. Para isso, selecionou-se no *menu* superior a opção **Settings** e depois no *menu* lateral esquerdo **Keys**. Por fim, foram anotadas para uso futuro as chaves **ApplicationID** e **REST API Key**.

Com todas as tabelas criadas no Parse, a próxima etapa foi configurar o Temboo que faz a ponte entre o Arduino e o Parse.com.

4.2.1 Temboo

O serviço Temboo gera de forma fácil, rápida e eficiente os algoritmos que foram implementados no *firmware* do Arduino para todas as requisições HTTP com o

Parse.com. Para tal, na página principal de seu *site*, já se tem a opção de cadastrar nova conta, sendo que todos os campos são obrigatórios, como visto na Figura 26.

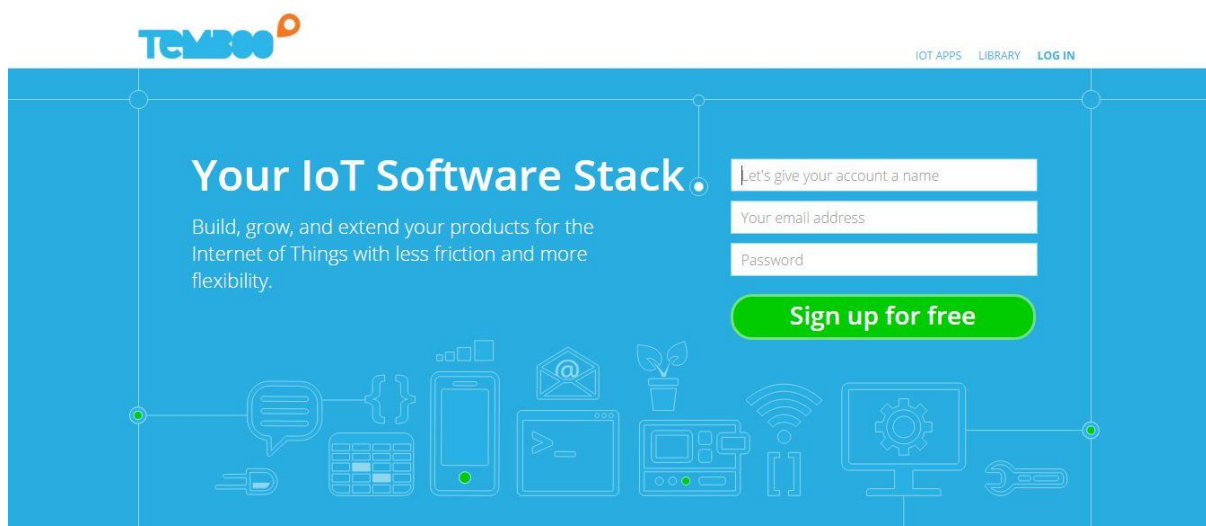


Figura 26 – Tela inicial Temboo.com.
Fonte: Autoria própria.

As requisições HTTP criadas pelo Temboo chamam-se Choreos. No presente TCC, faz-se necessária a criação de três Choreos, cada um tendo relação com uma escrita ou leitura do Arduino no Parse.com. Estando conectado ao Temboo com a conta recém-criada, no *menu* lateral direito selecionou-se a opção **Parse**, logo em seguida **Objects**. O primeiro Choreo está relacionado à criação de uma nova impressão na classe **Print** no Parse.com. Sendo assim, selecionou-se a opção **CreateObject**. A configuração é intuitiva e foi realizada conforme a Figura 27. Logo em seguida, deve-se clicar no botão **Run**.

Arduino shieldEthernet

Want to stream sensor data?

Parse . Objects . **CreateObject** ☆

Creates a new object on Parse.

+ Is this Choreo triggered by a sensor event?

INPUT Select Profile Save

ApplicationID
The Application ID provided by Parse.
APPLICATION ID SALVA ANTERIORMENTE

RESTAPIKey
The REST API Key provided by Parse.
RESTAPIKey SALVA ANTERIORMENTE

ClassName
The class name for the object being created.
Print

ObjectContents
A JSON string containing the object contents.
MANTER ESTE CAMPO VAZIO

This is not valid JSON

Run

Figura 27 – Tela de configuração Choreo *CreateObject* Tembo.com.
Fonte: Autoria própria.

Os códigos foram gerados automaticamente e foram implementados no arquivo principal do *firmware* do Arduino, conforme o campo gerado **CODE**. Foi necessário ainda criar um novo arquivo chamado *TembooAccount.h*, adicionado ao projeto do *firmware* com os dados gerados no campo **HEADER FILE**. Com isso, foi fechada a primeira requisição, que serve para criar uma nova impressão no sistema. O mesmo processo foi repetido para as opções de objetos do Parse

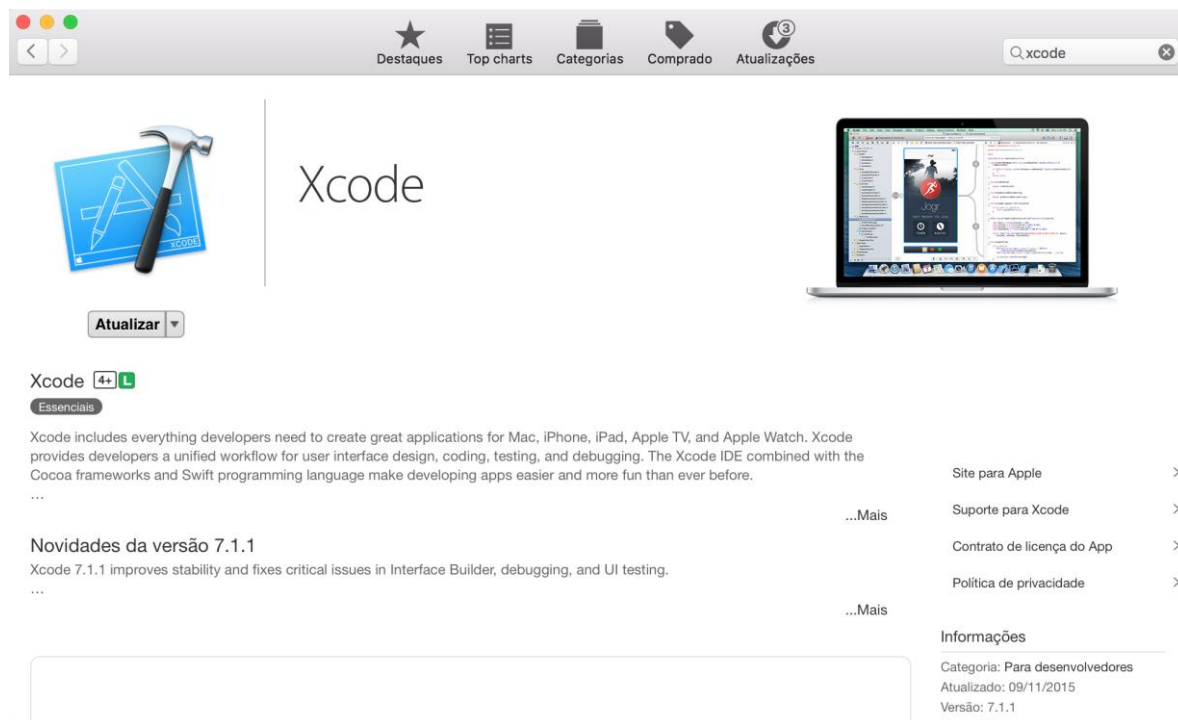
RetrievePropertyValue para receber o usuário atual conectado na classe **LoggedUser** coluna *user*, e **UpdateObject** para atualizar os campos *Weight*, *Length* e *Cost* da impressão atual na classe **Print**. Sendo assim todas as requisições HTTP foram criadas e implementadas no Arduino, fechando-se o ciclo de medição e armazenamento dos dados das impressões. Como estas serão usadas inúmeras vezes, foram salvas no Temboo e estão acessíveis através de *profiles* do próprio *site*. O *firmware* completo do Arduino pode ser encontrado no Apêndice B do presente trabalho.

Apesar de todas as funcionalidades terem sido implementadas, os dados devem estar acessíveis de alguma forma para os usuários finais. Com esta premissa, criou-se uma aplicação iOS para esta tarefa.

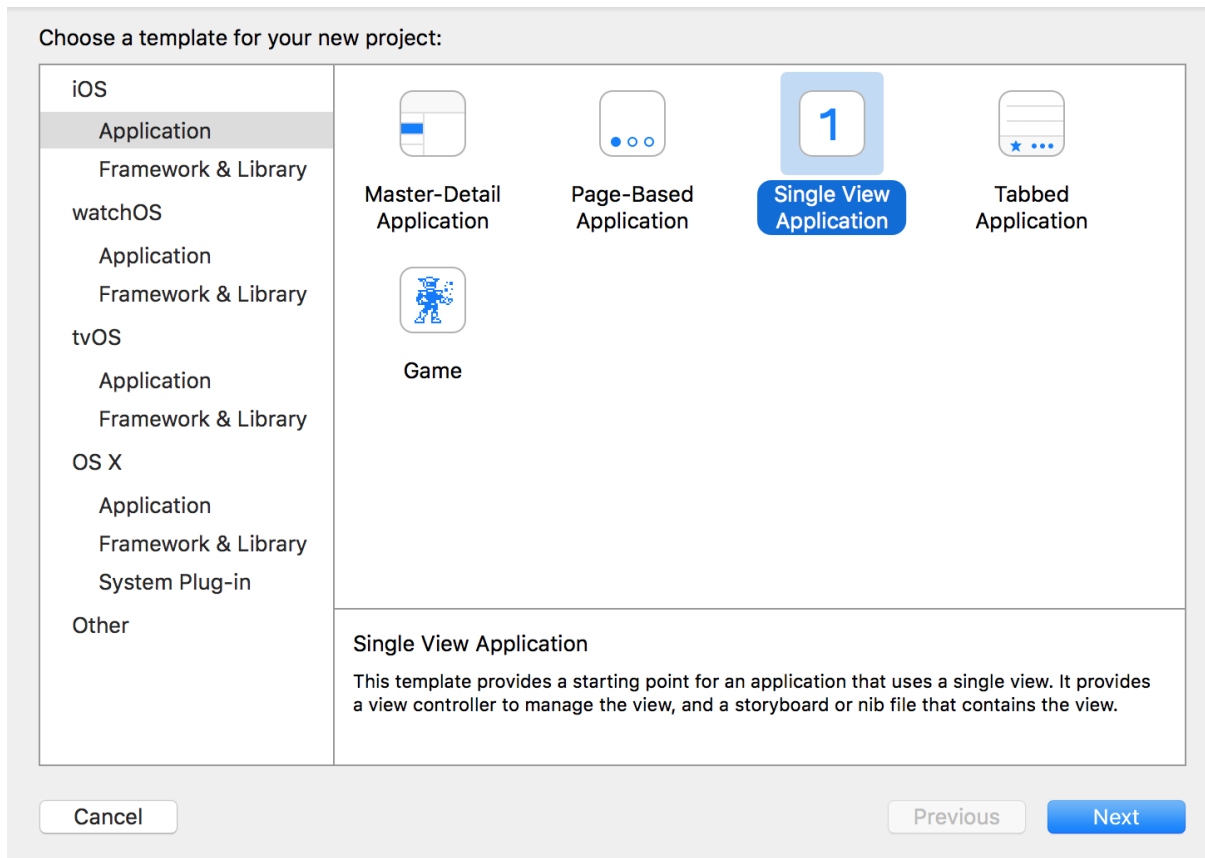
4.3 CRIAÇÃO DO APLICATIVO

Todos os aplicativos para iOS devem, necessariamente, ser desenvolvidos em computadores rodando o sistema operacional OSX da companhia norte-americana Apple. O ambiente de desenvolvimento próprio é o Xcode, disponível para *download* na App Store, conforme a Figura 28. A instalação é extremamente simples, sem nenhuma etapa complementar.

Este projeto foi criado como um *Single View Application*, no qual o aplicativo se molda para uma interface comum, conforme a Figura 29. Não se fez necessária nenhuma outra configuração especial, somente foi escolhido o nome do projeto.



**Figura 28 – Xcode disponível para *download* na AppStore.
Fonte: Autoria própria.**



**Figura 29 – Nova Single View Application no Xcode.
Fonte: Autoria própria.**

Os aplicativos se baseiam em elementos básicos de interface e, neste caso, foram utilizados os seguintes itens:

- Rótulo – Elemento para exibição de texto;
- Botão – Elemento que executa alguma ação quando selecionado;
- Campo de Texto – Elemento de *input* de dados;
- UIImage – Elemento de imagem;
- *Tab Bar* – Elemento para inclusão de telas na forma de abas.

Para facilitar o desenvolvimento de interface, o Xcode conta com arquivos especiais chamados *StoryBoards*. Nestes os elementos – como botões e rótulos – podem ser arrastados e posicionados dentro das telas de forma visual. No presente TCC, foram desenvolvidas cinco telas conectadas via *Tab Bar*, conforme a Figura 30, que apresenta o *Storyboard* da aplicação.

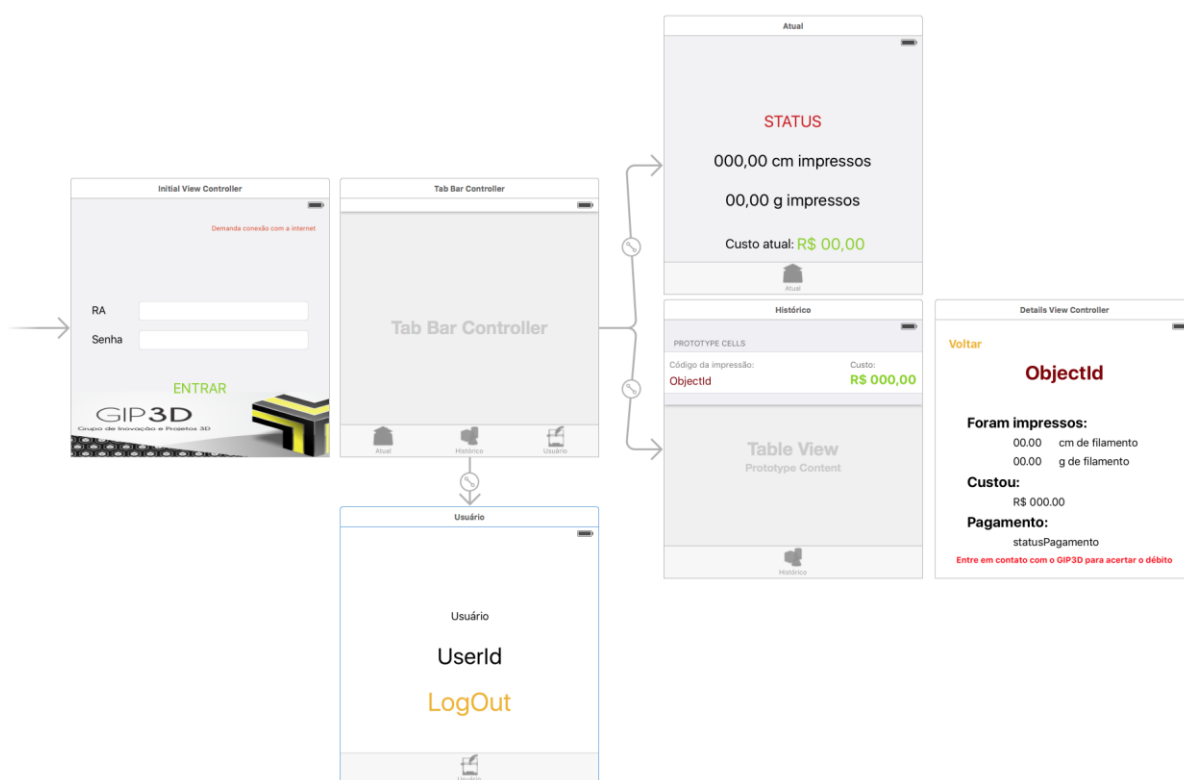


Figura 30 – Telas criadas no StoryBoard dentro do Xcode.
Fonte: Autoria própria.

Pode-se observar uma tela inicial, marcada por uma ponta de seta na Figura 30, na qual é feito o *login* dos usuários no sistema. Formada por elementos simples como rótulos, campos de texto e botão, aparece ao usuário somente no início da

aplicação. Existe também um rótulo para indicar se já existe alguém conectado impedindo uma conexão simultânea.

Após o *login*, o usuário acessa a tela de Impressão Atual. Nesta, rótulos apresentam ao aluno qual o *status* e custo de impressão do trabalho corrente.

Já na tela de histórico, uma tabela em forma de lista exhibe via rótulos os códigos de impressão e seus respectivos custos vinculados à conta do aluno. Os rótulos de preço são destacados de vermelho, indicando quando os débitos referentes à impressão ainda não foram quitados, ou de verde, quando a mesma foi paga. Caso algum item da tabela seja selecionado, uma tela com informações mais completas sobre aquela impressão é exibida.

Por fim, foi criada outra tela com rótulos, para exibir o registro acadêmico do aluno conectado ao sistema, e um botão que permite a saída e fechamento da conexão para liberar o sistema para um novo usuário realizar impressões.

5 RESULTADOS

São apresentadas entre as Figuras 31 e 34 as telas do aplicativo descrito no capítulo anterior. Na Figura 31 é ilustrada a tela inicial do aplicativo.

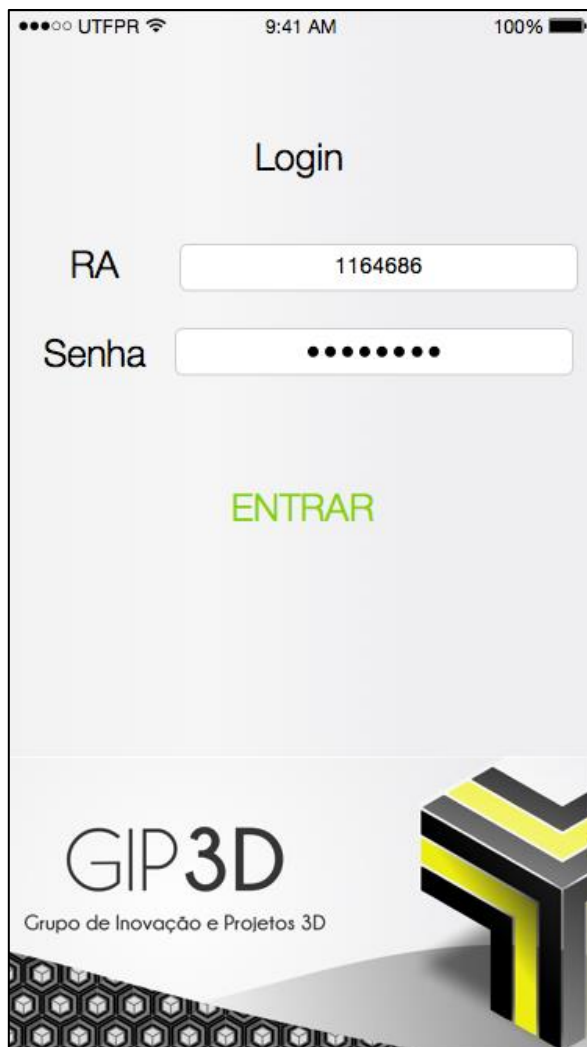


Figura 31 – Tela de *login* App iOS.
Fonte: Autoria própria.

Na sequência, na Figura 32 é exibida a tela de acompanhamento da impressão atual. Nela, pode-se conferir o *status* da impressão, assim como a quantidade de material gasto, o tempo transcorrido, custo atual.



Figura 32 – Tela de acompanhamento impressão atual App iOS.
Fonte: Autoria própria.

Observa-se na Figura 33 o histórico de todas as impressões realizadas pela conta conectada ao sistema, sendo listadas pelos códigos dos objetos que deram origem à impressão no *Web Server*. Nesta tela, tem-se ainda o custo de cada peça, estando verde quando a peça foi paga e vermelho quando existe débito pendente.

Por fim, na Figura 34, uma tela de usuário onde se confere o RA conectado atualmente e tem-se a opção de fazer *logout* para sair do sistema.

O aplicativo permite somente uma conexão de usuário por vez para se ter um melhor controle de quem está imprimindo no momento.

Código da impressão:	Custo:
cyl3afg469	R\$ 0.02
hUwdSLXG30	R\$ 0.06
maV38SjoH6	R\$ 0.30
xVKnauu0aw	R\$ 0.07
HCSEEvlcPo	R\$ 0.00
IzrRdebcH0	R\$ 2.50
4tawj9XTI1	R\$ 1.00
PmrN8vloTT	R\$ 0.00
ApXSaZM2Cx	R\$ 5.00

Figura 33 – Tela com histórico de impressão da impressora App iOS.
Fonte: Autoria própria.



Figura 34 – Tela de usuário App iOS.
Fonte: Autoria própria.

Os testes finais puderam ser iniciados em conjunto com a impressora após a integração de todas as partes do sistema. Todas as avaliações foram feitas em uma única impressora presente no GIP3D, modelo Prusa I3, conforme apresentado anteriormente no item 2.3.

Para certificação dos testes foram elaboradas oito peças de tamanhos diversos que possibilitassem variedade dos dados coletados. O método da pesagem, utilizado atualmente, era então contraposto com os dados obtidos pelo sistema de medição.

Os primeiros testes – ainda na fase de verificação da precisão do sistema – apontavam bons resultados quando se passava somente tamanhos conhecidos do filamento pelo *encoder*, sem fazer parte da impressora. Neste início havia mínimos erros, como por exemplo, conseguido para um filamento configurado para 56 cm de filamento por ciclo em que a diferença de pulsos tanto para cima e para baixo era constante e muito próxima de zero.

Devido ao *encoder* possuir uma enorme resolução, em que a quantidade de pulsos precisou ser dividida para se adequar ao conjunto, dificilmente seria possível obter total acurácia, porém de qualquer forma este não era o principal objetivo. Um sistema muito refinado exigiria mais de todo o conjunto, o que não é necessário para o uso do GIP3D. Assim, neste primeiro experimento isolado já era possível identificar que no funcionamento total do conjunto este erro seria pouco maior, precisando ser avaliado novamente. O importante desta etapa era garantir que o *encoder* em seu ambiente de trabalho e toda a programação estivessem de acordo com suas funções. Foi possível então garantir que o sistema funcionava, a lógica estava correta, a parte mecânica de seu funcionamento estava adequada e que o sucesso da ideia inicial seria promissor.

As primeiras impressões foram duas unidades de cubos de calibração, sendo estas com medidas definidas e quantidade de material já conhecida. O modelo foi conseguido através da página *online* Thingiverse.com, sob o nome “*Hollow Calibration Cube*”, sendo um cubo de 20 mm de arestas em cada uma de suas faces, muito utilizado para calibração do percurso linear dos eixos de impressão (THINGIVERSE, 2014). A Figura 35 representa o resultado obtido pelo autor do modelo 3D disponibilizado através da página, onde é possível verificar que os 20 mm foram de fato obtidos.

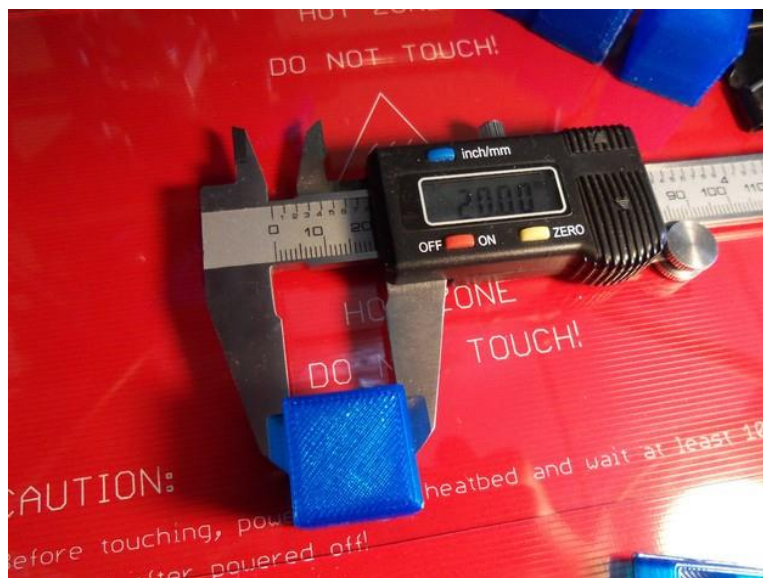


Figura 35 – Resultado obtido pelo autor do modelo.
Fonte: Thingiverse (2014).

Os modelos obtidos nos testes realizados dentro do GIP3D estão mostrados lado a lado na Figura 36. É importante lembrar que cada impressora possui uma calibração diferente e estão sujeitas aos mais diversos tipos de variações, questões às quais os integrantes da equipe não possuem acesso. De toda forma, indiferentemente das dimensões finais da peça, o que foi levado em conta é a quantidade de filamento gasto e a comparação do peso real da peça com o peso obtido com a utilização do sistema de medição.



Figura 36 – Resultados obtidos pela equipe em duas provas distintas.
Fonte: Autoria própria.

O primeiro cubo foi confeccionado adotando-se para o sistema uma taxa de atualização de 0,005, ou seja, a obtenção dos dados do *encoder* ocorre a cada meio

centavo de Real. Assim, obteve-se ao final um total de 62 pulsos do *encoder*, resultando através do sistema em um peso esperado de 2,48 g para cada peça. Fazendo a contraprova em uma balança de precisão existente no DAFIS (Departamento Acadêmico de Física), o cubo marcou o peso de 2,41g. Isto representa uma diferença de 2,9% para o sistema.

Levando em consideração o peso esperado pelo sistema, esta peça custaria R\$0,31. Comparando-se ao método de pesagem utilizado atualmente, o custo seria de R\$0,30 de acordo com o peso medido na balança.

Aqui é importante lembrar-se de que para o usuário o sistema atualiza a cada R\$0,50. Isto significa que tanto para o valor medido quanto para o valor pesado, ambos seriam arredondados para cima e esta peça ao final deveria ser cobrada em R\$0,50.

O processo para o segundo cubo foi feito de forma um pouco diferente. Desta vez a taxa de atualização foi parametrizada para 0,01, ou seja, o dobro da anterior. O resultado foi um total de 28 ciclos, pouco abaixo da metade da primeira peça, afinal a taxa de atualização era diferente. Essa tarefa gerou uma peça de 2,24 g, segundo o sistema. Fazendo a prova real, novamente na balança de precisão, foi pesado 2,15 g desta peça, o que significou uma diferença de 4,19%.

Novamente fazendo as conversões de peso para valores monetários, obteve-se um custo de R\$0,28 fornecido pelo novo sistema contra R\$0,27 através do atual método da pesagem. Conforme procedimento, o valor passado ao usuário seria arredondado para cima, confirmando o custo de R\$0,50 para ambos os cubos.

Na sequência das provas, foram executadas peças mais variadas, buscando experiências com formatos e tamanho diferentes. Para início desta segunda fase, a primeira peça escolhida foi uma xícara. Esta peça poderia ser um pouco mais complicada devido ao tamanho, bastante superior aos cubos anteriores. Para isto, foi parametrizada uma taxa de atualização de 0,005. Ao final foi obtido um total de 517 ciclos que foram medidos pelo sistema para 20,68 g ao custo de R\$2,59. Na prova da balança esta peça rendeu 27,01 g, que custariam então R\$3,38, ou seja, um erro de 23,44%. Em contrapartida, ao realizar o arredondamento o custo ao usuário seria de R\$3,00 pelo sistema e R\$3,50.

A xícara grande, apresentada na Figura 37, foi a peça que apresentou maior diferença, porém quando se trata do valor final repassado ao usuário esta diferença ainda é pequena. A diferença de R\$0,50 devido nesta peça é compensada em

outras impressões em que o valor arredondado fica superior ao valor real da peça. Uma analogia simples é quando, por exemplo, em uma gráfica, pede-se a impressão de uma página com poucas linhas de texto escrito comparado a uma página com diversas figuras e imagens. O valor cobrado será por página, ficando assim as duas ao mesmo preço.



Figura 37 – Xícara grande.
Fonte: Autorial própria.

A peça seguinte foi também uma xícara, mas esta menor, como pode ser visto na comparação da Figura 38. Os resultados desta tarefa foram conseguidos a partir de uma taxa de atualização de 0,01, que gerou um total de 47 ciclos ao final do processo. Para esta xícara o sistema mediu 3,76 g contra 4,42 g pesados pela balança, o que significaria respectivamente custos de R\$0,47 e R\$0,55. Para o usuário, o custo seria arredondado em R\$0,50 pelo sistema, porém o arredondamento a partir do peso saltaria o custo para R\$1,00.



Figura 38 – Xícara pequena, à direita da xícara grande à esquerda.
Fonte: Autorial própria.

Neste caso, os custos ficaram muito próximos ao ponto de mudança do preço, assim uma diferença de 14,93% resultou numa diferença entre os custos para o usuário.

A quinta peça avaliada (Figura 39) foi também baixada através do site Thingiverse, sob o nome “#1 Dad Keychain” (THINGIVERSE, 2015), sendo uma peça mais simples se comparada às anteriores.



Figura 39 – Chaveiro '#1 DAD'.
Fonte: Autoria própria.

O processo de impressão desta vez contabilizou 12 pulsos com uma taxa de atualização de 0,01, sendo medido pelo sistema um esperado de 0,96 g contra 1,17 g na balança, o que representa uma diferença de 17,9%. Em custos, o sistema calculou R\$0,12, enquanto pela balança sairia R\$0,15. De forma arredondada ao usuário, ambos custariam igualmente R\$0,50.

Na continuação das provas, foram testadas novamente duas peças iguais. Também disponibilizadas pelo endereço Thingiverse (2011), sob o nome “*Little Green Men (flying saucer pilots)*”, optou-se pelo menor modelo. Estes dois extraterrestres sentados não tiveram um bom resultado da impressão por serem muito pequenos e o material depositado não resfriou corretamente ao longo do processo, perdendo assim os pormenores de suas formas, como pode ser observado na Figura 40. Contudo, independentemente da qualidade visual, a medição do material ocorreu sem maiores problemas até o final.

Ambos foram acompanhados pelo sistema de medição com taxa de atualização de 0,01, gerando 15 ciclos para o Alien 1 e 10 ciclos para o Alien 2. Assim, as peças ficaram, respectivamente, com 1,2 g e 0,8 g de acordo com o

sistema e foram pesadas com 1,29 g e 0,92 g, o que representa uma diferença de 6,98% para o primeiro e 13,04% para o segundo.



**Figura 40 – Alien 1 à esquerda e Alien 2 à direita.
Fonte: Autoria própria.**

Em valores de custo, a medição através do sistema para o Alien 1 seria de R\$0,15 e de R\$0,10 para o Alien 2. Os custos por peso correspondem a R\$0,16 e R\$0,12, respectivamente, quando colocados na balança. A partir disto, o preço final ao usuário seria arredondado para cima, ficando ambos em R\$0,50

Outro teste foi realizado, desta vez com o “*Robber Rex*”, um tiranossauro Rex também disponível na página do Thingverse (2015), apresentado na Figura 41.



**Figura 41 – Tiranossauro Rex.
Fonte: Autoria própria.**

A modelagem foi realizada com 70 pulsos a uma taxa de atualização de 0,01. Assim, com uma diferença de 13,04% para menos, o sistema mediu 5,6 g contra a balança com 6,4 g. Estes valores representam então que pelo sistema esta peça custaria R\$0,70, mas por peso sairia por R\$0,81. Novamente pelo ajuste de custos aos usuários, de ambas as formas a peça custaria R\$1,00.

A Tabela 3 apresenta de forma resumida e complementar os resultados descritos anteriormente.

Tabela 3 – Resultados experimentais da impressão de peças diversas.

Peça	Taxa de Atualização	Ciclos	Peso por ciclo (g)	Peso estimado pelo sistema (g)	Peso Real (g)	Erro	Valor medido pelo sistema	Valor real	Arredondamento do valor de cobrança	Arredondamento do valor por peso
Cubo 1	0,005	62	0,04	2,48	2,41	2,90%	R\$ 0,31	R\$ 0,30	R\$ 0,50	R\$ 0,50
Cubo 2	0,01	28	0,08	2,24	2,15	4,19%	R\$ 0,28	R\$ 0,27	R\$ 0,50	R\$ 0,50
Xícara Grande	0,005	517	0,04	20,68	27,01	23,44%	R\$ 2,59	R\$ 3,38	R\$ 3,00	R\$ 3,50
Xícara Pequena	0,01	47	0,08	3,76	4,42	14,93%	R\$ 0,47	R\$ 0,55	R\$ 0,50	R\$ 1,00
#1 Dad	0,01	12	0,08	0,96	1,17	17,95%	R\$ 0,12	R\$ 0,15	R\$ 0,50	R\$ 0,50
Alien 1	0,01	15	0,08	1,2	1,29	6,98%	R\$ 0,15	R\$ 0,16	R\$ 0,50	R\$ 0,50
Alien 2	0,01	10	0,08	0,8	0,92	13,04%	R\$ 0,10	R\$ 0,12	R\$ 0,50	R\$ 0,50
Rex	0,01	70	0,08	5,6	6,44	13,04%	R\$ 0,70	R\$ 0,81	R\$ 1,00	R\$ 1,00

Fonte: Autoria própria

Para aferição do erro gerado durante a utilização do sistema proposto, foram confeccionadas repetidamente 10 peças iguais do pequeno cubo de 20x20mm.

Os dados obtidos são apresentados na Tabela 4.

Tabela 4 – Testes de repetição

Sequência	Peso estimado pelo sistema (g)	Peso Real (g)	Diferença (g)	Erro
01	2,00	2,1	0,1	4,76%
02	2,00	2,1	0,1	4,76%
03	2,08	2,2	0,1	5,45%
04	2,00	2,2	0,2	9,09%
05	2,08	2,2	0,1	5,45%
06	2,16	2,2	0,0	1,82%
07	1,92	2,1	0,2	8,57%
08	2,08	2,2	0,1	5,45%
09	2,08	2,2	0,1	5,45%
10	2,00	2,1	0,1	4,76%

Fonte: Autoria própria

Como pode ser notado, os erros ficaram dentro de uma faixa de 1,8 a 9% garantindo assim uma repetibilidade do sistema. Basicamente não houve diferenças significativas entre as peças, porém devido ao seu baixo peso, o erro distribuiu-se entre este intervalo, representando um erro médio de 5,56%.

As diferenças de peso entre a estimativa do sistema e o peso real dos cubos não foram maiores que 0,2 g – valor pouco significativo para o estudo.

Ao final da implementação, todos os itens foram doados à UTFPR, mais especificamente ao GIP3D visto que todo o sistema foi criado para suprir uma necessidade do grupo. A Tabela 5 apresenta os componentes adquiridos e necessários para a elaboração do conjunto, bem como os custos envolvidos.

Tabela 5 – Componentes e custos aproximados

Item	Descrição	Valor unitário
01	Arduino Mega2560	R\$ 150,00
02	Shield Ethernet	R\$ 130,00
03	Encoder KY-040	R\$ 20,00
04	Encoder 1000 pulsos	R\$ 100,00
05	Componentes mecânicos	R\$ 25,00
06	Componentes elétricos/eletrônicos	R\$ 10,00
Total		R\$ 435,00

Fonte: Autoria própria

6 CONCLUSÕES

Pode-se considerar que o principal objetivo deste trabalho, o desenvolvimento de um sistema capaz de mensurar e apresentar ao usuário a quantidade de filamento utilizada em um processo de impressão 3D por FDM para o sistema de faturamento dos protótipos desenvolvidos pelo GIP3D da UTFPR, foi atingido.

Após a realização dos trabalhos juntos à impressora, chegou-se a principal conclusão de que embora o filamento seja caro para a compra por quilo, peças pequenas não apresentaram alto custo relativo para confecção. Ainda que estas sejam peças com parâmetros construtivos simples, seu peso final é pouco se comparado ao todo da bobina de filamento que é comumente vendida com 1 kg de material, ou seja, com uma bobina destas é possível confeccionar muitas peças. Ainda que às vezes ocorram problemas durante a impressão e o processo precise ser reiniciado o maior prejuízo talvez esteja somente no tempo gasto na necessidade de um recomeço. Diante disto, pode-se concluir também que pequenos desperdícios de filamento ocorridos no início do processo de extrusão não representam tamanho desperdício financeiro.

O mais importante ao final deste trabalho é poder ofertar ao GIP3D um sistema para controle das ações da impressora que serão armazenados no banco de dados automaticamente, eliminando que um voluntário precise intervir na ação. Para estes somente será preciso acompanhar o processo e tomar conta da impressora, pois o sistema de medição foi elaborado na tentativa de não intervir na impressão. As ações humanas serão necessárias apenas para iniciar o processo com o toque de um botão e para a conferência após o termino.

Vale ressaltar também que juntamente ao sistema para monitoramento por parte dos voluntários, tem-se ainda o aplicativo para sistema iOS que permite oferecer aos usuários a transparência do processo através do acompanhamento em tempo real da atividade. Para isto não é necessário estar próximo da impressora, pois graças à ligação com a Internet as informações podem ser obtidas de qualquer lugar que se tenha conexão disponível.

Considera-se que os objetivos propostos foram atingidos, pois ao longo da realização deste trabalho muitas mudanças ocorreram se comparada à proposta inicial. A cada mudança surgiam novas informações que puderam ser absorvidas

pelos integrantes através desta constante evolução. O sistema foi elaborado conjuntamente ao servidor e ao aplicativo móvel gerando uma grande integração de conteúdos.

Ainda que o sistema tenha apresentado funcionamento satisfatório, algumas melhorias são sugeridas como forma de trabalhos futuros:

- Desenvolvimento do aplicativo para sistemas Android e Windows Phone e uma aplicação *Web* que possibilite acesso aos mais diversos tipos usuários;
- Aperfeiçoamento do suporte para o *encoder* com peças confeccionadas na própria impressora 3D;
- Simplificação da mecânica do sistema a fim reduzir seu custo ao máximo e facilitar qualquer tipo de manutenção ou reparo;
- Manipulação da programação para funcionamento em uma única unidade controladora, como por exemplo, a Raspberry Pi, que seria capaz de gerenciar tanto as atividades da impressora como o sistema de medição.

REFERÊNCIAS

APPLE. Apple Introduces the New iPhone 3G. 2015. Disponível em: <<http://www.apple.com/pr/library/2008/06/09Apple-Introduces-the-New-iPhone-3G.html>>. Acesso em: 23 jun 2015.

ARDUINO E CIA. Como Utilizar um Encoder Rotativo com Arduino, 2015. Disponível em: <<http://www.arduinoecia.com.br/2015/08/como-usar-encoder-rotativo-ky-040-arduino.html>>. Acesso em: 12 nov 2015.

ARDUINO LLC. Arduino Ethernet Shield 2015. Disponível em: <<https://www.arduino.cc/en/Main/ArduinoEthernetShield>>. Acesso em: 09 nov 2015.

_____. Arduino Fórum 2015. Disponível em: <<http://forum.arduino.cc/index.php?topic=177178.0>>. Acesso em: 09 nov 2015.

_____. Arduino Mega 2560. 2015. Disponível em: <<http://www.arduino.cc/en/Main/ArduinoBoardMega2560>>. Acesso em: 13 jun 2015.

_____. Compare board specs. 2015. Disponível em <<http://www.arduino.cc/en/Products.Compare>>. Acesso em: 20 jun 2015.

_____. What is Arduino? 2015. Disponível em: <<http://www.arduino.cc/en/Guide/Introduction>>. Acesso em: 13 jun 2015.

BANZI, Massimo. Dinner is Ready. Blog Arduino, set 2010. Disponível em <<http://blog.arduino.cc/2010/09/24/dinner-is-ready/>>. Acesso em: 20 jun 2015.

_____. Updating About Arduino Yún And Arduino Robot. Blog Arduino, ago 2013. Disponível em <<https://blog.arduino.cc/2013/08/21/updating-about-arduino-yun-and-arduino-robot/>>. Acesso em 26: jun 2015.

BOLTON, William. Mecatrônica: uma abordagem multidisciplinar. 4. ed. São Paulo: Bookman, 2010.

CAMPOS, Luiz Emanuel S. M. Prototipagem rápida: Definições, conceitos e prática. 1. ed. Buenos Aires: Delearte Emcampos, 2011. ISBN 978-987-27142-4-6.

CARVALHO, Jonas de; VOLPATO, Neri. Prototipagem rápida como processo de fabricação. In: AHRENS, Carlos Henrique et al. Prototipagem rápida: Tecnologias e aplicações. Editor: Neri Volpato. São Paulo: Edgard Blücher, 2007. ISBN 85-212-0388-8.

CHUA, Chee Kai; LEONG, Kah Fai; LIM, Chu Sing. Rapid Prototyping: Principles and Applications. 3. ed. Singapura: World Scientific Publishing, 2010. ISBN 978-981-277-897-0.

COOPER, Kenneth G. Rapid prototyping technology: Selection and Application. Nova York: Marcel Dekker Inc, 2005. ISBN 0-8247-0261-1.

FERREIRA, José Manuel Martins. Introdução ao Projecto com Sistemas Digitais e Microcontroladores. Porto: FEUP Edições, 1998. ISBN 972-752-032-4.

GRIMM, Todd. User's guide to rapid prototyping. Dearborn: Society of Manufacturing Engineers. 2004. ISBN 0-87263-697-6.

HEISS, Augusto. Encoders: Saiba como funcionam os sensores mais usados na automação industrial. Revista Mecatrônica Atual, São Paulo, ano 11, n.56, mai-jun/2012. Disponível em: <<http://www.mecatronicaatual.com.br/files/file/MA56web.pdf>>. Acesso em: 25 nov 2014.

IBTIMES. Apple WWDC 2014: Swift Announced, New Programming Language May Herald New Era In Coding. Disponível em: <<http://www.ibtimes.com/apple-wwdc-2014-swift-announced-new-programming-language-may-herald-new-era-coding-1593747>>. Acesso em: 28 jun 2015.

IDC. Estudo da IDC Brasil aponta que, em 2014, brasileiros compraram cerca de 104 smartphones por minuto, 2015. Disponível em: <<http://br.idclatin.com/releases/news.aspx?id=1801>>. Acesso em: 23 jun 2015.

IDGNow. Android aumentou domínio no mercado de smartphones em 2014, diz IDC. IDG News Service, 25 de fevereiro de 2015. Disponível em: <<http://idgnow.com.br/mobilidade/2015/02/25/android-aumentou-dominio-no-mercado-de-smartphones-em-2014-diz-idc/>>. Acesso em: 12 out 2015.

IMAGEFRIEND. Galleries for Prusa Mendel. Disponível em <<http://imagefriend.com/prusa-mendel.shtm>>. Acesso em: 06 jun 2015.

JONES, Bryan A.; REESE, Robert; BRUCE, J. W. Microcontrollers: From Assembly Language to C Using the PIC24 Family. 2. ed. Boston: Cengage, 2014. 608p. ISBN 9781305076556.

KLEIN, Peter W., Fundamentals of Plastics Thermoforming. Synthesis Lectures On Material Engineering. Ohio: Morgan & Claypool Publishers, 2009. ISBN 9781598298857.

LABORATÓRIO DE GARAGEM. Tutorial: como utilizar encoder rotativo com Arduino. São Paulo: Lab de Garagem, 2012. Disponível em <<http://labdegaragem.com/profiles/blogs/tutorial-como-utilizar-encoder-rotativo-com-arduino>>. Acesso em: 08 jul 2015.

LIOU, Frank W. Rapid prototyping and engineering applications: a toolbox for prototype development. Boca Raton: CRC Press, 2008. ISBN 978-0-8493-3409-2.

MAZUROV, Oleg. Reading rotary encoder on Arduino. Circuits@Home. Abr, 2010. Disponível em <<https://www.circuitsathome.com/mcu/programming/reading-rotary-encoder-on-arduino>>. Acesso em: 08 jul 2015.

McROBERTS, Michael. Beginning Arduino. 2. ed. Nova York: Springer Science+Business Media, 2010. ISBN 978-1-4302-3240-7.

MELLIS, David. Arduino Mega: Bigger, More Powerful, Still Blue. Blog Arduino. Ivrea, mar 2009. Disponível em <<https://blog.arduino.cc/2009/03/26/arduino-mega-bigger-more-powerful-still-blue/>>. Acesso em: 20 jun 2015.

MILANI, André. Programando para iPhone e iPad. São Paulo: Novatec, 2014. ISBN 978-85-7522-394-9.

OMRON Corporation. E6B2-C – Incremental 40-mm-dia. Rotary Encoder, 2008. Disponível em <http://www.mouser.com/ds/2/307/e6b2-c_dsheets_csm491-217673.pdf>. Acesso em: 12 out 2015.

OPEN SOURCE INITIATIVE. Welcome to The Open Source Initiative. California, 2015. Disponível em <<http://opensource.org>>. Acesso em: 14 jun 2015.

REPRAP. Prusa i3 Rework Introduction. Disponível em <http://reprap.org/wiki/Prusa_i3_Rework_Introduction>. Acesso em: 06 de jul 2015.

ROMEIRO FILHO, Eduardo; NAVEIRO, Ricardo Manfredo. Uso de modelos e protótipos no projeto de productos. In: FERREIRA, Cristiano Vasconcelos et al. Projeto do produto. Coord.: Eduardo Romeiro Filho. Rio de Janeiro: Elsevier: ABEPRO, 2011. ISBN 978-85-352-5191-3.

SPARKFUN. Folha de dados do *encoder* COM-09117, 2008. Disponível em: <<https://www.sparkfun.com/>>. Acesso em: 23 jun 2015.

S&E Instrumentos de Testes e Medição Ltda. Manual Encoders: Geradores de Impulsos. São Paulo, 2014. 2p. Disponível em <<http://www.seinstrumentos.com.br/pdf/cat-encoders.pdf>>. Acesso em: 26 nov 2014.

SILVA, Steve. Web Server Administration. Boston: Cengage, 2007. ISBN 978-1-4239-0323-9.

SMITH, Jack R. Programming the PIC Microcontroller with MBasic. Burlington: Elsevier, 2005. ISBN 0-7506-7946-8.

SWIFT, Apple. Swift. A new language that lets everyone build amazing apps. Disponível em: < <http://www.apple.com/swift/>>. Acesso em: 23 jun 2015.

TAURION, Cezar. Cloud Computing : computação em nuvem: transformando o mundo da tecnologia da informação . Rio de Janeiro: Brasport, 2009. ISBN 978-85-7552-423-8.

THINGIVERSE. #1 Dad keychin yaneirag17, Nov 2015. Disponível em <<http://www.thingiverse.com/thing:1123166>>. Acesso em: 12 nov 2015.

_____. Hollow Calibration Cube by 3D-ME, Mar 2014. Disponível em <<http://www.thingiverse.com/thing:271736>>. Acesso em: 12 nov 2015.

_____. Little Green Men (flying saucer pilots) by gpvillamil, Set 2011. Disponível em <<http://www.thingiverse.com/thing:11810>>. Acesso em: 12 nov 2015.

_____. Robber Rex by zheng3, Jan 2015. Disponível em <<http://www.thingiverse.com/thing:613445>>. Acesso em: 12 nov 2015.

UNO Robótica Educacional. Manual Encoder de Quadratura. Rio Grande do Sul, 2012. 20 p. Disponível em <<http://www.unorobotica.com.br/docs/encoder.pdf>>. Acesso em: 25 nov 2014.

VIEIRA, Nando. Entendendo um pouco mais sobre o protocolo HTTP. São Paulo, 2007. Disponível em <<https://nandovieira.com.br/entendendo-um-pouco-mais-sobre-o-protocolo-http>>. Acesso em: 12 nov 2015.

WHEAT, Dale. Arduino Internals. Nova York: Springer Science+Business Media, 2011. ISBN 978-1-4302-3882-9.

APÊNDICE A – DIAGRAMA FUNCIONAL DO SISTEMA

Na Figura 42 é apresentado o diagrama funcional do sistema formado por: Suporte do *encoder*, kit com Arduino Mega2560 e *shield* Ethernet, sistema Temboo, servidor Parse.com e o aplicativo para iOS.



Figura 42 - Diagrama funcional do sistema.
Fonte: Autoria própria.

APÊNDICE B – FIRMWARE DO ARDUINO

Arquivo principal, encoder.ino:

```

/* UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELETROTÉCNICA
ENGENHARIA INDUSTRIAL ELÉTRICA ÊNFASE EM AUTOMAÇÃO / ENGENHARIA DE CONTROLE E AUTOMAÇÃO
TRABALHO DE CONCLUSÃO DE CURSO 2
AUTORES: LUCAS EDUARDO DE FREITAS / PEDRO HENRIQUE GAIO PACHECO

CONFIGURE PARÂMENTRO EM Configs.h
*/

//Inclusão de todas as dependências
#include <RotaryEncoder.h> //Biblioteca para gerenciamento do encoder
#include <SPI.h> //Biblioteca adicionada automaticamente pelo Temboo
#include <Dhcp.h> //Biblioteca adicionada automaticamente pelo Temboo
#include <Dns.h> //Biblioteca adicionada automaticamente pelo Temboo
#include <Ethernet.h> //Biblioteca adicionada automaticamente pelo Temboo para funcionamento do Shield Ethernet
#include <EthernetClient.h> //Biblioteca adicionada automaticamente pelo Temboo para funcionamento do Shield Ethernet
#include <Temboo.h> //Biblioteca adicionada automaticamente pelo Temboo
#include "TembooAccount.h" //Arquivo contendo as configurações de conta do Temboo
#include "TembooProfiles.h" //Arquivo contendo as configurações de profiles do Temboo
#include "ParseAccount.h" //Arquivo contendo as configurações de conta do Parse.com
#include "Config.h" //Arquivo de configurações

byte ethernetMACAddress[] = ETHERNET_SHIELD_MAC; //Definindo o MAC address do Shield Ethernet
EthernetClient client; //Definindo "client" como chamada do EthernetClient (vindo de Ethernet.h)

RotaryEncoder encoder(A1, A2); //Pinos de ligacao do encoder

//Define variaveis básicas
int newPos = 0; //Posição atual do encoder
int ciclos = 0; //Ciclos de atualização
String currentUser; //Usuário atual
String currentPrint; //Código de impressão atual
int PPR = PPROriginal / divisaoPPR; //Pulsos por revolução que será usado nos cálculos
float relacao; //Quantos centímetros por pulso de filamento
int ref; //Referencia de pulsos para atualização do servidor, a cada 'ref' pulsos ele atualiza o servidor que equivale aos reais
da taxa de atualização
float massa; //Calcula quantos gramas equivalem a taxaAtualiza centavos
float densidadeAtual; //densidade que será usada nos cálculos
float medida; //Quantos centímetro de filamento equivalem a taxaAtualiza centavos

void setup() //Configurações iniciais
{
  //Configuração dos pinos
  pinMode(13, OUTPUT); //LED vermelho indicador de qualquer erro;
  pinMode(5, OUTPUT); //LED RGB vermelho indicador servidor erro de conexao; PARA STATUS CONECTANDO pinos 5 E 6
HIGH
  pinMode(6, OUTPUT); //LED RGB verde indicador servidor conectado!;
  pinMode(7, OUTPUT); //LED verde indicador impressao pode iniciar.

  //Inicia serial com 9600 de bit rate
  Serial.begin(9600);

  //Aguarda 4 segundos e aguarda comunicação com a serial
  delay(4000);
  while(!Serial); //comentar esta linha para uso stand-by

```

```

Serial.print("Sistema ira atualizar a cada "); Serial.print(taxaAtualiza); Serial.println(" reais"); //Escrita na serial

//Calcula quantos gramas equivalem a X centavos de atualização
massa = (1000 * taxaAtualiza) / valorKg;
Serial.print("Equivale a "); Serial.print(massa); Serial.println(" g de material"); //Escrita na serial

//Checa qual o material e configura a densidade correta
switch (material) {
  case 1:
    densidadeAtual = densidadeABS; //Escolhe densidade do ABS
    break;
  case 2:
    densidadeAtual = densidadePLA; //Escolhe densidade do PLA
    break;
  case 3:
    densidadeAtual = densidadeOUTRO; //Escolhe densidade de outro material genérico
    break;
  default:
    Serial.println("ATENCAO: MATERIAL NAO ENCONTRADO, TODOS OS CALCULOS ESTARAO ERRADOS!!! Arrume e reinicie 1-
ABS 2-PLA 3-OUTRO"); //Mensagem de erro escrita na serial
    digitalWrite(13, HIGH); //Liga LED de erro
    while(1); //Trava o programa aqui e aguarda reinício do sistema
    break;
}

//Calcula quantos centímetros de filamento equivalem a X centavos de atualização
float raio2 = pow((bitola/2),2);
medida = massa / (densidadeAtual * pi * raio2);
Serial.print("Equivale a "); Serial.print(medida); Serial.println(" cm do filamento"); //Escrita na serial

//Calcula relação de quantos cm tem em cada pulso
relacao = (2 * pi * raioAcoplado)/PPR;

//Calcula quantos pulsos equivalem a X centavos de atualização
ref = medida / relacao;

Serial.print("Equivale a "); Serial.print(ref); Serial.println(" pulsos do encoder"); //Escrita na serial
Serial.print("Considerando uma relacao de "); Serial.print(relacao); Serial.println(" cm para cada pulso do encoder");
//Escrita na serial
Serial.println("\n\n"); //Escrita na serial

// Inicia conexao com servidor
Serial.print("DHCP:"); //Escrita na serial
digitalWrite(5, HIGH); digitalWrite(6, HIGH); //Liga LED RGB amarelo indicando que está tentando conectar com o servidor

if (Ethernet.begin(ethernetMACAddress) == 0) { //Se não conseguir conectar via Ethernet com o MacAddress
  digitalWrite(5, HIGH); digitalWrite(6, LOW); digitalWrite(13, HIGH); //Liga LED RGB vermelho indicando erro na conexão e
desliga verde
  Serial.println("FAIL"); //Escrita na serial
  while(true); //Trava o programa aqui e aguarda reinício do sistema
}
digitalWrite(6, HIGH); digitalWrite(5, LOW); //Liga LED RGB verde indicando servidor OK e desliga vermelho
Serial.println("OK"); //Escrita na serial
delay(5000); //Delay de 5 segundos

Serial.println("Setup server complete.\n"); //Escrita na serial

receiveCurrentUser(); //Chama função do Temboo que recebe o usuário atual

if (currentUser.length() != 10){ //Caso o usuário tenha menos de 10 dígitos
  Serial.println("Nenhum user logado ou user invalido. Favor fazer login no aplicativo movel e REINICIE o sistema");
//Escrita na serial mensagem de erro

```

```

    digitalWrite(13, HIGH);//Liga LED indicando erro
    while(1); //Trava o programa neste ponto e espera reinício do sistema
}
Serial.print("User:"); Serial.println(currentUser); //Escrita na serial

createNewPrint(); //Função do Temboo que cria um novo código de impressão (nova linha no banco de dados)
Serial.print("PrintObjectID:"); Serial.println(currentPrint); //Escrita na serial

updateCurrentObjectID(); //Atualiza no servidor o código de impressão atual

digitalWrite(7, HIGH);//Liga LED indicando que pode começar a impressão
Serial.println("\n Impressao pronta para comecar!"); //Escrita na serial
}

void loop() //Looping infinito
{

//Le as informacoes do encoder
static int pos = 0;
static int posVirt = 0;

encoder.tick(); //Inicia período de amostragem do encoder
int newPos = encoder.getPosition(); //Recebe nova posição do encoder

//verifica os pulsos para desconsiderar os intermediarios (diminui a resolução do encoder)
if (newPos != pos + divisaoPPR && newPos >= pos){
    goto fim;
}
if (newPos != pos - divisaoPPR && newPos <= pos) {
    goto fim;
}

//Se a posicao foi alterada, novos cálculos são feitos a seguir
if (pos != newPos) { //se posição atual é diferente da última posição
    if (newPos == ref * divisaoPPR) { //Se fechou um novo ciclo (posição atual está na posição que é igual a posição de novo
ciclo de atualização)
        ciclos++; //Incrementa número de ciclos
        pos = 0; //reinicia posição zero
        newPos = 0; //reinicia posição atual
        encoder.setPosition(0); //Configura a posição atual do encoder como zero

//ENVIA PARA SERVIDOR DADOS
float w = ciclos*massa; //cálculo atual de peso
float l = ciclos*medida; //cálculo atual de medida
float c = ciclos*taxaAtualiza; //cálculo atual de preço
updatePrint(w,l,c); //Função do Temboo que atualiza os dados da impressão atual

    } else { // se não for posição de um novo ciclo
        pos = newPos; //armazena última posição
        posVirt = (pos / divisaoPPR); //armazena posição virtual
    }
}
//Serial.println(pos);
Serial.print("Pulsos: ");Serial.print(posVirt);Serial.print(" -- ");Serial.print("Ciclos: ");Serial.println(ciclos); //Escrita na serial
fim; //Label de escape
}

}

void createNewPrint(){ //Função do Temboo para criar nova linha de impressão
    TembooChoreo CreateObjectChoreo(client);

// Inicia cliente Temboo

```

```

CreateObjectChoreo.begin();

// Configura credenciais de conta Temboo
CreateObjectChoreo.setAccountName(TEMBOO_ACCOUNT);
CreateObjectChoreo.setAppKeyName(TEMBOO_APP_KEY_NAME);
CreateObjectChoreo.setAppKey(TEMBOO_APP_KEY);

// Configura profile Temboo que será usado
CreateObjectChoreo.setProfile(TEMBOO_CREATE_NEW_PRINT_PROFILE);

//Configura os inputs da requisição
String ObjectContentsValue = "{\"paid\":false,\"cost\":0,\"length\":0,\"weight\":0,\"user\":\":";
ObjectContentsValue += currentUser;
ObjectContentsValue += "\"}";
Serial.println(ObjectContentsValue);
CreateObjectChoreo.addInput("ObjectContents", ObjectContentsValue);

// Identifica a requisição que irá rodar
CreateObjectChoreo.setChoreo("/Library/Parse/Objects/CreateObject");

// Roda a requisição
CreateObjectChoreo.run();

while(CreateObjectChoreo.available()) { //quando resultados estiverem disponíveis armazena resultado
  String completeString;
  completeString = String(CreateObjectChoreo.readString()); //String completa de resposta do servidor
  currentPrint = completeString.substring(78,88); //Cria substring pegando somente a parte que nos importa da string
  completa de resposta do servidor (código da impressão atual)
}
CreateObjectChoreo.close(); //Termina requisição
}

void receiveCurrentUser(){ //Função do Temboo para receber o usuário atual
  TembooChoreo RetrievePropertyValueChoreo(client);

  // Inicia cliente Temboo
  RetrievePropertyValueChoreo.begin();

  // Configura credenciais de conta Temboo
  RetrievePropertyValueChoreo.setAccountName(TEMBOO_ACCOUNT);
  RetrievePropertyValueChoreo.setAppKeyName(TEMBOO_APP_KEY_NAME);
  RetrievePropertyValueChoreo.setAppKey(TEMBOO_APP_KEY);

  // Escolhe o profile para executar a requisição
  RetrievePropertyValueChoreo.setProfile(TEMBOO_RECEIVE_CURRENT_USER_PROFILE);

  // Identifica a requisição que irá rodar
  RetrievePropertyValueChoreo.setChoreo("/Library/Parse/Objects/RetrievePropertyValue");

  // Roda a requisição
  RetrievePropertyValueChoreo.run();

  while(RetrievePropertyValueChoreo.available()) { //quando resultados estiverem disponíveis armazena resultado
    String completeString;
    completeString = RetrievePropertyValueChoreo.readString(); //Armazena string completa de resposta
    currentUser = completeString.substring(23,33); //Sub string somente da parte importante da resposta completa (usuário
    atual logado no sistema)
  }
  RetrievePropertyValueChoreo.close(); //Fecha a requisição
}
}

```

```
void updatePrint(float currentWeigth, float currentLength, float currentCost){ //Função do Temboo para atualizar
parâmetros da impressão atual
```

```
    TembooChoreo UpdateObjectChoreo(client);
```

```
    // Inicia cliente Temboo
```

```
    UpdateObjectChoreo.begin();
```

```
    // Configura credenciais de usuário Temboo
```

```
    UpdateObjectChoreo.setAccountName(TEMBOO_ACCOUNT);
```

```
    UpdateObjectChoreo.setAppKeyName(TEMBOO_APP_KEY_NAME);
```

```
    UpdateObjectChoreo.setAppKey(TEMBOO_APP_KEY);
```

```
    // Configura parâmetros da requisição
```

```
    UpdateObjectChoreo.addInput("RESTAPIKey",PARSE_RESTAPI_KEY);
```

```
    UpdateObjectChoreo.addInput("ObjectID", currentPrint);
```

```
    UpdateObjectChoreo.addInput("ApplicationID", PARSE_APPLICATION_ID);
```

```
    String ClassNameValue = "Print";
```

```
    UpdateObjectChoreo.addInput("ClassName", ClassNameValue);
```

```
    String ObjectContentsValue = "{\n \"weigth\":";
```

```
    ObjectContentsValue += currentWeigth;
```

```
    ObjectContentsValue += ",\n \"length\":";
```

```
    ObjectContentsValue += currentLength;
```

```
    ObjectContentsValue += ",\n \"cost\":";
```

```
    ObjectContentsValue += currentCost;
```

```
    ObjectContentsValue += "\n}";
```

```
    UpdateObjectChoreo.addInput("ObjectContents", ObjectContentsValue);
```

```
    // Identifica requisição para rodar
```

```
    UpdateObjectChoreo.setChoreo("/Library/Parse/Objects/UpdateObject");
```

```
    // Roda a requisição
```

```
    UpdateObjectChoreo.run();
```

```
    while(UpdateObjectChoreo.available()) { //Quando resultados estiverem disponíveis, armazena resposta
```

```
        String c = UpdateObjectChoreo.readString(); // Armazena resposta completa da requisição
```

```
        Serial.println("Enviou"); //Escreve na serial
```

```
    }
```

```
    UpdateObjectChoreo.close(); //Fecha a requisição
```

```
}
```

```
void updateCurrentObjectID(){ //Função Temboo para atualizar o código de impressão atual
```

```
    TembooChoreo UpdateObjectChoreo(client);
```

```
    // Inicia cliente Temboo
```

```
    UpdateObjectChoreo.begin();
```

```
    // Configura credenciais de conta Temboo
```

```
    UpdateObjectChoreo.setAccountName(TEMBOO_ACCOUNT);
```

```
    UpdateObjectChoreo.setAppKeyName(TEMBOO_APP_KEY_NAME);
```

```
    UpdateObjectChoreo.setAppKey(TEMBOO_APP_KEY);
```

```
    // Configura inputs da requisição
```

```
    UpdateObjectChoreo.addInput("RESTAPIKey",PARSE_RESTAPI_KEY);
```

```
    UpdateObjectChoreo.addInput("ObjectID", "3hSETXea7m");
```

```
    UpdateObjectChoreo.addInput("ApplicationID", PARSE_APPLICATION_ID);
```

```
    String ClassNameValue = "LoggedUser";
```

```
    UpdateObjectChoreo.addInput("ClassName", ClassNameValue);
```

```
    String ObjectContentsValue = "{\n \"currentPrint\":\":";
```

```
    ObjectContentsValue += currentPrint;
```

```
    ObjectContentsValue += "\n}";
```

```
    UpdateObjectChoreo.addInput("ObjectContents", ObjectContentsValue);
```

```
    // Identifica requisição para rodar
```

```

UpdateObjectChoreo.setChoreo("/Library/Parse/Objects/UpdateObject");

// Roda a requisição
UpdateObjectChoreo.run();

while(UpdateObjectChoreo.available()) { //Quando resultados estiverem disponíveis armazena resposta
  String c = UpdateObjectChoreo.readString(); //Armazena resposta completa
  //Serial.println("Enviou");
}
UpdateObjectChoreo.close(); //Fecha requisição
}

```

Arquivo de configurações, config.h:

```

/*
CONFIGURAÇÕES INICIAIS
*/

float taxaAtualiza = 0.5; //Taxa em REAIS que deve enviar novos dados ao servidor DEFAULT: 0.5(cinquenta centavos de real)
float bitola = 0.28; //Bitola do flamento em CENTÍMETROS (aferir com paquímetro)
float valorKg = 120.00; //Valor em REAIS do Kg do material usado DEFAULT 120,00
int PPRoriginal = 1000; //PULSOS POR REVOLUÇÃO original do encoder atual.
int divisaoPPR = 100; //Quantas VEZES deseja dividir os Pulsos Por Revolução do encoder
int material = 1; // 1 - ABS, 2 - PLA, 3 - OUTRO
float densidadeABS = 1.05; //Densidade do ABS em g/cm^3
float densidadePLA = 1.25; //Densidade do PLA em g/cm^3
float densidadeOUTRO = 1.5; //Densidade material GENÉRICO em em g/cm^3
float raioAcoplado = 1.55; //Raio em CENTÍMETROS do circulo acoplado ao eixo do encoder que estará em contato com o
filamento.
const float pi = 3.14159265359;

```

Arquivo da conta Parse.com, ParseAccount.h:

```

/*
CONFIGURAÇÕES DE CONTA PARSE.COM
*/

#define PARSE_RESTAPI_KEY "XXXXXXXXXXXXXXXXXXXX" // define chave REST API Parse
#define PARSE_APPLICATION_ID "XXXXXXXXXXXXXXXXXXXX" // define chave APPLICATION ID Parse

```

Arquivo da conta Temboo.com e MAC address, TembooAccount.h:

```

/*
CONFIGURAÇÕES DE CONTA TEMBOO
*/

#define TEMBOO_ACCOUNT "pedrique" // define nome da conta Temboo
#define TEMBOO_APP_KEY_NAME "myFirstApp" // define nome da aplicação Temboo
#define TEMBOO_APP_KEY "XXXXXXXXXXXXXXXXXXXX" // define chave APP Temboo

#define ETHERNET_SHIELD_MAC {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED} //Define MAC Address do Shield Ethernet

```


Arquivo da profiles Temboo.com, TembooProfiles.h:

```
/*  
  CONFIGURAÇÕES DE PROFILES TEMBOO  
*/  
  
#define TEMBOO_CREATE_NEW_PRINT_PROFILE "newPrint" //define o profile do Temboo para a função createNewPrint  
#define TEMBOO_RECEIVE_CURRENT_USER_PROFILE "TCC2currentUser" //define o profile do Temboo para a função  
receiveCurrentUser
```