

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO



tAMARINO: uma abordagem visual para prototipagem rápida em computação física

Dissertação de Mestrado

Aluno: Ricardo Borges Brasileiro
Orientador: Geber Lisboa Ramalho
Co-orientador: Abel Guilhermino da Silva Filho

rbb2@cin.ufpe.br
glr@cin.ufpe.br
agsf@cin.ufpe.br

RECIFE, Agosto de 2013

para Rudah Brasileiro

Agradecimentos

Agradeço primeiro à Existência, representado na natureza em forma de plantas e paisagens; a minha família, em especial à Aninha, Rudah e Damião, que cuidou com muito amor, e aos meus pais, irmã e parentes mais próximos pela minha formação.

Em relação à pesquisa, tenho agradecimentos especiais para Geber Ramalho, Abel Guilhermino, Ricardo Ruiz, Jarbás Jácome, Filipe Calegário, Jeraman, Giordano Cabral e aos laboratórios MustIC, 3Ecologias e Laboca.

Em relação aos suportes para avaliação dessa pesquisa, gostaria de agradecer aos laboratórios que disponibilizaram os equipamentos e o tempo dos participantes para contribuir com a mesma, em especial aos coordenadores do laboratório CRC Marista Recife, da Escola de Arte e Tecnologia Oi Kabum Salvador e ao grupo de pesquisa EcoArte da Universidade Federal da Bahia.

Além, gostaria de agradecer a Adriano Belisário, Alan Fachini, Alexandre Andrade, Alexandre Braz, Amanda, Bruno Freitas, Cíntia Mendonça, Daniel do Espaço Ciência, Dee Harvey, Desiree Santos, Don Dagen, Eva moolab.net, Flávio Luciano, Frank Junior, Gabriel Menotti, Giuliano Obici, GuimaSan, Henrique Mineiro, Ian Archbell, Isaac Filho, James Tu, Jean Cardoso, João Tragtenberg, José Neto, Karla Brunet, Luca Bezerra, Marco Egito, Moabe Filho, Paulo Lima, Pedro Ângelo, Pedro Barbosa, Ricardo Palmieri, Sofia Galvão, Steve Bull, Thiago Hersan, Victor Souza e todas as pessoas que contribuíram com ideias, sugestões e críticas para o tAMARINO.

Uma atenção especial para os elogios de Maximo Banzi (CEO do Arduino) a respeito do trabalho desenvolvido e pelos comentários de Tom Igoe do *ITP/New York University* e Philip van Allen, desenvolvedor do Netlab Toolkit.

Por fim, agradeço ao meu fone de ouvido que me inspirou em diversas situações durante a pesquisa.

Resumo

As práticas de desenvolvimento em computação física embutem dois desafios: a construção do hardware e o desenvolvimento do software. Engenheiros, técnicos e demais especialistas foram historicamente responsáveis pela missão de resolver estes desafios, o que minimizou a necessidade de ferramentas amigáveis para programação e prototipagem em eletrônica. O conhecimento compartilhado de tecnologias e o movimento *open-source* - para muito além da técnica – impulsionaram o desenvolvimento de produtos fora do *modus operandis* industrial e o surgimento de desenvolvedores incomuns: artistas, designers e *Makers*. Diante disso, *toolkits* como Arduino e Fritzing surgiram para solucionar os desafios. No entanto, muito pode ser feito para acelerar o processo de concepção e prototipagem à esses *Makers*.

tAMARINO é uma proposta de um ambiente visual único e intuitivo para acelerar a prototipagem em computação física. O sistema atua na prototipagem do software e do hardware ao mesmo tempo, o que acelera os primeiros passos. A primeira versão da aplicação foi concebida para microcontroladores Arduino, mas sua arquitetura é extensível a muitas outras placas. A avaliação revelou o sucesso da aplicação em diminuir o tempo necessário para o desenvolvimento de protótipos, reduzindo o *time-to-market*.

Palavras-chave: Computação Física, Prototipagem Rápida, Programação Visual, Internet das coisas, Arduino, Fritzing.

Abstract

Prototype concept and building practices in physical computing embed two challenges: the hardware construction and the software development. Engineers, technicians and specialist were historically assigned the mission to solve those challenges. This maximized the lack of toolkits for easy programming and prototyping. Shared technological knowledge and the open source movement - far beyond onto technics – established all-embracing products and goodies outside industrial *modus operandis* as well as uncommon developers: artists, designers and *Makers*. As consequence, toolkits like Arduino and Fritzing emerged. However, much can be done to bolster this *Makers* class.

tAMARINO proposes an unique and intuitive visual enviroment toolkit to accelerate physical computing prototypes. The evaluation reveal tAMARINO' success to straightforward quick development - even on first-time prototyping – further lowering the time-to-market. This first version is designed for Arduino microcontrolleres, obviously extendable to many other boards.

Key-words: Physical Computing, Rapid Prototyping, Visual Programming, Internet of Things, Arduino, Fritzing.

"...the street finds its own uses for things"
William Gibson, Burning Chrome.

Sumário

Agradecimentos.....	3
Resumo.....	4
Abstract.....	5
1. Introdução.....	12
1.1 Motivações.....	12
1.2 Objetivo.....	14
1.3 Estrutura da dissertação.....	14
2. Contexto.....	16
2.1 A Computação Física.....	16
2.2 Visão Geral da Computação Física.....	18
2.3 Do movimento open-source para os Makers.....	20
3. Desafios.....	25
3.1 Princípios da prototipagem.....	25
3.2 O contexto da prototipagem na eletrônica.....	27
3.3 Os percursos típicos das experiências em computação física.....	33
3.4 Princípios para uma abordagem visual integrada.....	38
4. Trabalhos relacionados.....	41
4.1 Ambientes para prototipagem em computação física.....	41
4.2 Ambientes de prototipagem de software.....	44
4.2.1 Os softwares stand-alone.....	44
4.2.2 Os software de tempo-real.....	48
4.3 Ambientes de prototipagem de componentes eletrônicos.....	53
4.3.1 Os kits de sensores e atuadores fechados.....	53
4.3.2 Design de automação eletrônica com a breadboard.....	58
4.4 Análise dos ambientes relacionados.....	60
5. tAMARINO.....	64
5.1 Aspectos gerais do ambiente.....	64
5.2 Funcionalidades básicas do tAMARINO.....	66
5.2.1 Programação visual.....	68
5.2.2 Conexões Dinâmicas.....	70
5.2.3 Prototipagem Automática.....	72
6. Processos de desenvolvimento.....	75
6.1 Descrição dos processos de design.....	75
6.2 Inspiração.....	78
6.2.1 Experiência Pessoal.....	78
6.2.2 Inspiração em mesas multi-toque e web-frameworks.....	80
6.3 Concepção.....	83
6.3.1 Protótipos preliminares e experimentos iniciais.....	84
6.3.1.1 Ciclos de Prototipação.....	85
6.3.2 Avaliação dos protótipos preliminares.....	87
6.3.2.1 Opiniões sobre a abordagem.....	90
7. Implementação.....	92
7.1 Arquitetura do Sistema.....	92
7.2 Tecnologias utilizadas.....	96
7.2.1 Detalhamento das aplicações utilizadas.....	98

8. Avaliação.....	103
8.1 Prova de conceito.....	103
8.1.1 Perfil dos Participantes.....	104
8.1.2 Sobre os conceitos abordados.....	106
8.1.3 Principais características abordadas.....	107
8.1.4 Contexto de utilização da ferramenta.....	109
8.1.5 Perfis de usuários.....	110
8.1.6 Projetos relacionados.....	112
8.1.7 Impressões da interface visual.....	113
8.1.8 Pontos positivos e negativos.....	114
8.1.9 Sugestões e Críticas.....	117
8.1.9.1 Críticas especializadas.....	118
8.2 Experimentos em laboratórios.....	119
8.2.1 Perfil dos laboratórios e dos participantes.....	120
8.2.2 Resultados dos experimentos.....	121
8.2.3 Opiniões sobre os experimentos e as ferramentas.....	123
9. Conclusão.....	125
9.1 Trabalhos futuros.....	126
10. Referências.....	128

Figuras

Figura 1: Projeto Piano Staircase do grupo The Fun Theory.....	17
Figura 2: Os componentes de um sistema de Computação Física.....	17
Figura 3: O perfil dos makers: dos quartos sujos às invenções criativas.....	20
Figura 4: Uma breadboard por dentro.....	25
Figura 5: Projeto Arduino com um hardware físico e uma IDE para programar as entradas e saídas digitais e analógicas.....	27
Figura 6: Atividades típicas para projetos de computação física.....	30
Figura 7: Os componentes que formam as etapas de programação e prototipagem na Computação Física.....	31
Figura 8: Exemplo de aparato com os ingredientes da Computação Física desenvolvido pelo autor desta investigação.....	32
Figura 9: Projetos relacionados com as etapas de prototipagem de protótipos em computação física.....	37
Figura 10: S4A Scratch com a biblioteca que envia dados para o Arduino.....	41
Figura 11: Na esquerda, a área de programação orientada a blocos gráficos, semelhante ao Scratch e na direita o ambiente de setup dos componentes.....	42
Figura 12: Interface do usuário e ambiente de programação do Splish.....	43
Figura 13: Patch Pduino para enviar e receber dados do Arduino via Firmata.....	45
Figura 14: Widgets do Netlab Toolkit.....	46
Figura 15: Widgets em HTML5 do Netlab Toolkit.....	47
Figura 16: Editor online do Codebender.....	48
Figura 17: Phidget com sensor de pressão.....	50
Figura 18: O ambiente d.tools.....	51
Figura 19: O hardware e o modelo para configurar novos dispositivos.....	52
Figura 20: Principais componentes do Tinkercad um exemplo de conexão de um led.....	52
Figura 21: Exemplo de uso do Makey Makey.....	53
Figura 22: Múltipla visualização no Fritzing.....	55
Figura 23: Modo de edição de um componente e zoom no Fritzing.....	56
Figura 24: Percurso mais indicado para um ambiente de prototipagem em computação física.....	58
Figura 25: tAMARINO e seu ambiente de programação com widgets configuráveis e plugáveis.....	61
Figura 26: tAMARINO e sua área de prototipagem eletrônica integrada.....	62
Figura 27: Na imagem da esquerda, a tela inicial do ambiente e na imagem mais a direita, (1) é a barra de ferramentas, (2) é a área de esboço (ou sketch) e (3) é a área de prototipagem eletrônica (ou make) do tAMARINO.....	63
Figura 28: As três etapas-chave da abordagem proposta para acelerar as experiências através do ambiente tAMARINO.....	63
Figura 29: Representação geral de um módulo no tAMARINO.....	64
Figura 30: Os módulos de entrada do ambiente tAMARINO.....	65
Figura 31: Os módulos de saída do ambiente tAMARINO.....	66
Figura 32: Exemplo de conexão entre um módulo de input com um output no ambiente tAMARINO.....	66
Figura 33: Conexões entre um POT e um SERVO.....	67
Figura 34: Conexões dinâmicas entre mais de um módulo ao mesmo tempo.....	68

Figura 35: Área de prototipagem automática (MAKE) do ambiente tAMARINO.....	69
Figura 36: Linha do tempo da prototipagem automática.....	70
Figura 37: Os processos do Design Centrado no Usuário. [IDEO, 2011].....	73
Figura 38: Vlad Trifa sobre o conceito de Web of Things no Lift'11.....	76
Figura 39: Protótipo de interação entre software e hardware por meio de mesa multi-toque. [Conradi et al., 2010].....	76
Figura 40: Framework Meemoo e uma interface para controlar animação.....	78
Figura 41: Web Audio Playground e seus widgets plugáveis e configuráveis.....	78
Figura 42: Processo iterativo em espiral para o processo de concepção.....	79
Figura 43: Soluções para prototipagem rápida de ideias.....	80
Figura 44: Primeiro protótipo desenvolvido do tAMARINO.....	82
Figura 45: Segundo protótipo do tAMARINO.....	82
Figura 46: Terceiro protótipo do tAMARINO.....	83
Figura 47: O framework para task-based proposto por [Burstien & Linger, 2002].....	85
Figura 48: Avaliação da fase de concepção. Do (1) ao (3) a coleta foi realizada presencialmente. Do (4) ao (6) a coleta foi realizada por meio da Internet com compartilhamento da tela.....	85
Figura 49: Arquitetura para a abordagem da interface tAMARINO.....	89
Figura 50: Máquina de estados do modelo MVC segundo [Krasner et al., 1988].....	89
Figura 51: Arquitetura de aplicações em ambientes integrados para Computação Física.....	90
Figura 52: Arquitetura e níveis de abstração do tAMARINO.....	91
Figura 53: Arquitetura e Tecnologias utilizadas no tAMARINO.....	92
Figura 54: Diagrama de Classes utilizadas no desenvolvimento do protótipo do ambiente tAMARINO.....	96
Figura 55: Descrição de uma conexão com as abstrações do jsPlumb.....	97
Figura 56: Nível dos participantes em relação aos conhecimentos em Computação Física na escala de 0-5, onde 5 significa integração total com o contexto.....	100
Figura 57: Experiência dos participantes no contexto da Computação Física.....	101
Figura 58: Experiência prática dos participantes com a plataforma Arduino.....	101
Figura 59: Faixa etária dos participantes.....	101
Figura 60: Principais princípios abordados pelos usuários em relação ao tAMARINO.....	103
Figura 61: Nível de características mais marcantes pelos participantes.....	104
Figura 62: Principais contextos de utilização levantados na entrevista sobre o tAMARINO.	105
Figura 63: Principais perfis de usuários indicados pelos participantes para utilizar o tAMARINO.....	107
Figura 64: Projetos relacionados com o contexto do tAMARINO.....	108
Figura 65: Impressões marcantes da interface visual do tAMARINO.....	109
Figura 66: Pontos positivos destacados pelos usuários.....	110
Figura 67: Pontos negativos da ferramenta.....	111
Figura 68: Lista de sugestões dos participantes para a ferramenta.....	113
Figura 69: Perfil dos participantes no Lab 01.....	116
Figura 70: Perfil dos participantes no Lab 02.....	117
Figura 71: Perfil dos participantes no Lab 03.....	117

Tabelas

Tabela 1: Número de pesquisas com o termo "Physical Computing" e "Internet of Things" na biblioteca digital ACM.....	20
Tabela 2: Quadro comparativo entre aplicações e princípios norteadores.....	61
Tabela 3: Opiniões sobre a abordagem do projeto a partir da experiência com o protótipo de baixa-fidelidade desenvolvido na etapa de concepção. Dados obtidos através de perguntas e respostas na escala de Likert.....	90

1. Introdução

Está cada vez mais fácil perceber um aumento substancial de projetos criativos em computação física desenvolvidos por pessoas sem conhecimentos técnicos aprofundados [Mellis et al., 2007] [Knörrig et al., 2009]. Estes projetos utilizam entradas e saídas de sensores e atuadores para dialogar com ambientes físicos, especialmente protótipos que interagem com componentes eletrônicos de percepções visuais, sonoras e táteis. Como esses projetos são, na maioria das vezes, modelados por *designers*, artistas e não especialistas técnicos, o contexto de utilização da computação física ganha um novo terreno e uma coleção de moradores que se estabelecem num espaço antes só populado por especialistas, engenheiros e cientistas.

É neste contexto que discutimos as motivações iniciais para esta pesquisa, os principais objetivos e a estrutura da dissertação.

1.1 Motivações

Um dos projetos que impulsionou o aumento na produção de protótipos interativos em computação física foi o Arduino¹, uma plataforma *open-source* que compõe um microcontrolador e um software para escrever códigos, compilar e exportar instruções em portas digitais e analógicas de forma fácil. A plataforma é baseada nas dificuldades encontradas nos primeiros projetos de microcontroladores como a *Mini Board* [Martin, 1998], *Handy Board* [Martin, 2000], *Programmable Brick* [Resnick et al., 1996], *GoGo Board* [Sipitakiat et al., 2002], *AvrMini* [Wilson et al., 2003] e *Programma 2003* [Mellis et al., 2007], que exigia dos usuários um conhecimento técnico aprofundado em eletrônica e computação.

O Arduino tem o seu modelo baseado na documentação dos projetos como vetor para agregar pessoas não-especialistas e disseminar o conteúdo de forma viral, com inspirações na cultura DIY (faça-você-mesmo) [Anderson, 2012]. A partir do desenvolvimento desta plataforma, é possível perceber um engajamento das pessoas para desenvolver seus próprios *gadgets*, seja um dispositivo interativo em instalações artísticas ou um objeto para monitorar

¹ <http://arduino.cc/>

comportamentos urbanos.

Apesar das transformações culturais provocadas pelo Arduino, o que tornaram as atividades de produção de conteúdo e aprendizagem mais ágeis, ainda existem obstáculos nos primeiros passos em computação física, seja para programar sensores e atuadores de baixo-custo ou para montar o circuito eletrônico. A linguagem de programação *Wiring*, nativa na IDE do Arduino, é desenhada para pessoas com familiaridade em linguagens tipo *Processing* e *Java*, o que exige um interesse em se aprofundar nos detalhes técnicos da linguagem para experimentar as interações, transformando a atividade numa barreira para os iniciantes. Além disso, existem desafios na etapa da prototipagem eletrônica, principalmente para se conectar e instalar os componentes físicos para completar o ciclo de prototipação da experiência.

Os desafios da prototipagem impulsionaram os usuários a publicarem as soluções técnicas em diversos *blogs* e *sites*² numa linguagem mais fácil do que os *datasheets*, normalmente utilizadas por um público mais técnico para analisar os componentes e projetar os circuitos. As informações nos *datasheets* não possuem detalhes de conexão entre os componentes de uma forma amigável, como nas placas de prototipagem (*breadboard*). As práticas de *hacking* em tutoriais se tornaram comuns no contextos de usuários iniciantes como uma forma de adaptar as soluções a partir de exemplos genéricos [Hartmann et al., 2008] [Conradi et al., 2010]. Essas práticas exigiram dos usuários uma maior experiência em buscas na Internet e autonomia para adaptar os conteúdos para os projetos pessoais. Para realizar um *Hello World* é necessário atravessar os diversos passos em busca de soluções semelhantes para identificar as conexões entre os componentes, o que atrasa e desestimula as primeiras experiências.

Diversas iniciativas de ambientes visuais surgem como efeito colateral das dificuldades encontradas por usuários em relação à aprendizagem computacional. Projetos como Scratch [Resnick, 2009], Splish [Kato, 2010], Pduino [Stainer, 2009], Modkit [Millner et al., 2011], Netlab Toolkit [Allen et al., 2007] ou Codebender são desenvolvidos com objetivos de encontrar soluções de software para facilitar as interações com usuários iniciantes. Os softwares visuais são focados na etapa da programação dos componentes eletrônicos através de módulos, blocos e ícones programáveis.

2 Make Magazine, Instructables etc.

Para os desafios dos circuitos eletrônicos, os usuários necessitam de um alto nível de experiência em navegação e pesquisa na Internet para dialogarem com as barreiras técnicas dos componentes. Ferramentas como o Fritzing [Knörig et al., 2009] surgem como uma solução de EDA (*Electronic Design Automation*) para desenho e documentação de circuitos eletrônicos produzidos pelos entusiastas e inventores, os *Makers*, como cita Anderson [Anderson, 2012]. Mesmo assim, para usar a automação do Fritzing, é necessário conhecimento prévio dos componentes e do circuito a ser montado [Conradi et al., 2010].

Frente às barreiras da programação e da prototipagem levantadas, surgem desafios para construir abordagens integradas a essas duas camadas com o objetivo de acelerar as primeiras experiências. As pesquisas dessa dissertação apontam que, além da expansão da comunicação com os microcontroladores através de *middlewares* como o Firmata [Stainer, 2009], as soluções em *frameworks* que utilizam as infra-estrutura dos *browsers* e da web para recombinar diferentes soluções, são as práticas contemporâneas mais conectadas com os desejos dos usuários e desenvolvedores [Hartmann et al., 2008].

Esta investigação também é inspirada na motivação pessoal do autor desta dissertação em suas práticas cotidianas em concepção e prototipagem de projetos de interação com aparatos físicos e interativos. As motivações são baseadas na larga experiência do autor em *workshops* de interatividade para um público não-técnico advindo das diversas expressões artísticas.

1.2 Objetivo

Esta pesquisa tem como objetivo desenvolver o tAMARINO como uma solução de ferramenta visual e intuitiva para prototipagem rápida de projetos em computação física para não-especialistas técnicos e iniciantes

1.3 Estrutura da dissertação

Esta dissertação possui dez capítulos. O primeiro capítulo trata sobre as questões introdutórias das perspectivas do trabalho e o segundo trata o contexto da computação física e dos usuários que se agregam nas práticas inventivas. A partir do terceiro capítulo, são apresentados os desafios do primeiro passo para os usuários não-especialistas e são estabelecidos os princípios

norteadores para solucionar os desafios nas prototipagens iniciais. No quarto capítulo são apresentados os trabalhos relacionados com as perspectivas de investigar as lacunas entre a prototipagem digital e física, bem como uma realizar uma análise mais crítica em relação aos propósitos de acelerar as experiências iniciais. No quinto, é apresentada o tAMARINO como uma proposta de sistema que dialoga com uma abordagem mais completa em relação as etapas-chave da prototipagem na computação física. No sexto capítulo estão as descrições dos processos de desenvolvimento. No sétimo capítulo trata-se dos detalhes da implementação, bem como os aspectos da arquitetura e das tecnologias utilizadas. No oitavo capítulo são realizadas avaliações de prova de conceito e avaliações práticas em laboratórios com diversos perfis de usuários. Por fim, no nono capítulo, são apresentadas as conclusões dessa pesquisa e os trabalhos futuros. O capítulo dez expõe as referências desta pesquisa.

2. Contexto

Neste capítulo será apresentado o contexto da computação física a partir da evolução das experiências em realidade virtual e aumentada e também em relação aos transbordamentos das telas para os ambientes reais. Além disso, será apresentada uma visão geral sobre a computação física e suas pesquisas mais emergentes no contexto da interatividade. Por último, serão elucidados os perfis dos usuários que utilizam sensores e atuadores de forma criativa que, de certo modo, se transformou num dos vetores de reconfiguração de uma emergente “revolução industrial”, mais autônoma, conectada e longe dos antigos modelos de fábrica.

2.1 A Computação Física

*When we see, hear and feel real-world objects we are enabled
to train both cognitive and perceptual skills in combination.
Such objects can help us create interfaces that are
easier and more fun to use.*

[Papadimatos, 2005, p6]

O contexto da computação física está inserido numa área de pesquisa que trata das transformações dos ambientes e objetos físicos através do processamento de informações por meio de microcontroladores, sensores e atuadores. O tema está relacionado também a sistemas que permitem a inserção de dispositivos físicos em ambientes controlados e, a partir disso, o usuário pode criar relações reais e virtuais entre os aparatos.

Para Papadimatos [Papadimatos, 2005], os processos de evolução que modelaram o contexto das interações físicas estão baseados nas seguintes fases:

(1) **A extrapolação do virtual** a partir de aplicações em VR (*Virtual Reality*) e AR (*Augmented Reality*), que mixam as realidades em ambientes virtuais e reais com equipamentos imersivos (*displays* e salas especiais). A tese principal é que a manipulação de

artefatos físicos e a atuação em ambientes reais, conectadas com informações digitais, proporciona um maior engajamento por parte do usuário quando comparado com interações mais abstratas [Dourish, 2001];

(2) **A quarta geração das máquinas** onde a computação está inserida nos objetos do dia a dia. Alguns autores chamam de pervasiva ou ubíqua [Grønback & Krogh, 2001] mas essa mesma noção pode ser expandida para o sentido em que a computação física distribui a inteligência por meio de artefatos tangíveis que estão espalhados pelos ambientes;

(3) **A revolução dos processamentos de informações embarcados** surgem como um impulso para as experiências em computação física. As realidades mixadas e aumentadas são interpretadas por *displays* e ambientes imersivos. Para atuar em ambientes e objetos reais, os microcontroladores são as tecnologias que contemplam as necessidades das inteligências embarcadas em objetos físicos no dia a dia, ou como Weiser propõe: os “milhares de computadores” para cada pessoa [Weiser, 2003].

(4) **A transformação do computador** torna-se uma anti-abordagem às aplicações de realidade virtual e aumentada. As GUI (*Graphical User Interfaces*) criaram uma relação tradicional entre as pessoas e as máquinas, e conseqüentemente, as afastaram das percepções físicas. A computação física provoca uma interação que transborda das telas para os ambientes reais.

Essa tendência é percebida em produtos como o do grupo *The Fun Theory*, que transformou uma escada de uma estação de metrô em *Odenplan/Stockholm*, num piano, afim de alterar o comportamento das pessoas e, ao mesmo tempo, economizar energia na estação pelo não uso das escadas rolantes e incentivar a prática de exercícios físicos. Este exemplo contempla as práticas sugeridas na evolução das interações para o contexto do ambiente real e dos microcontroladores embarcados em objetos do dia a dia.



Figura 1: Projeto Piano Staircase do grupo The Fun Theory.

2.2 Visão Geral da Computação Física

A Computação Física se estabelece como um diálogo entre o mundo físico e o mundo virtual através de um processo de transdução³ de uma energia em outra, onde a direção dessas energias são chamadas de *input* ou *output* e os sinais podem ser analógicos ou digitais. Também é possível controlar os eventos que ocorrem a cada tempo para determinar um plano de interação [Igoe & O'Sullivan, 2004].

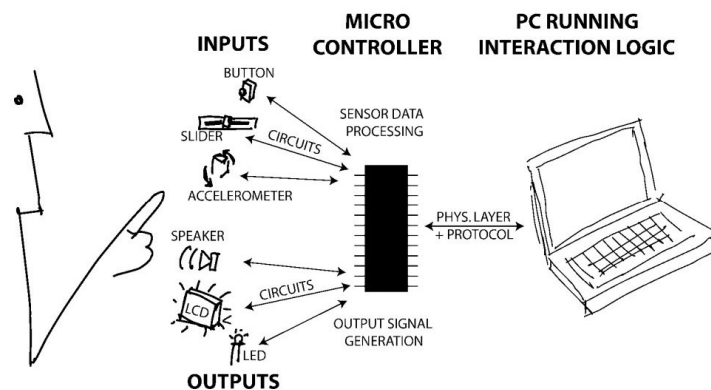


Figura 2: Os componentes de um sistema de Computação Física. [Hartmann, 2005]

3 s. f. Transformação de uma energia numa energia de natureza diferente. (Dicionário Priberam)

Para Greenwold, essas interações se configuram como um componente essencial para fazer das máquinas um objeto útil no dia a dia das pessoas e que sirvam tanto para o trabalho quanto para o lazer [Greenwold, 2003]. Num outro contexto, as interações podem ser entendidas como um processo de **escuta**, **pensamento** e **fala** entre dois ou mais atores e a qualidade dessa interação vai depender também da qualidade em cada um desses processos [Crawford, 2002].

No artigo *Opportunities and Obligations for Physical Computing Systems* [Stankovic et al., 2005], as aplicações em computação física prometem dar suporte à sociedade e prover novos padrões de moradia, segurança, integração e eficiência. Para o autor, essa infra-estrutura modela e aponta para uma revolução técnica, econômica e social onde o foco no desenvolvimento de aplicativos e sistemas *high-level* se torna decisivo para tal.

Na mesma perspectiva, a IEEE *Computer Society* lançou⁴ em seu portal as principais tendências para as pesquisas em tecnologia para os anos de 2013 e 2014, onde, em primeiro lugar, está o tema *Internet of Things*, do qual trata basicamente do contexto da computação física aplicada na inter-conexão de objetos reais do dia a dia embarcados em tecnologias de rede [Guinard & Trifa, 2009].

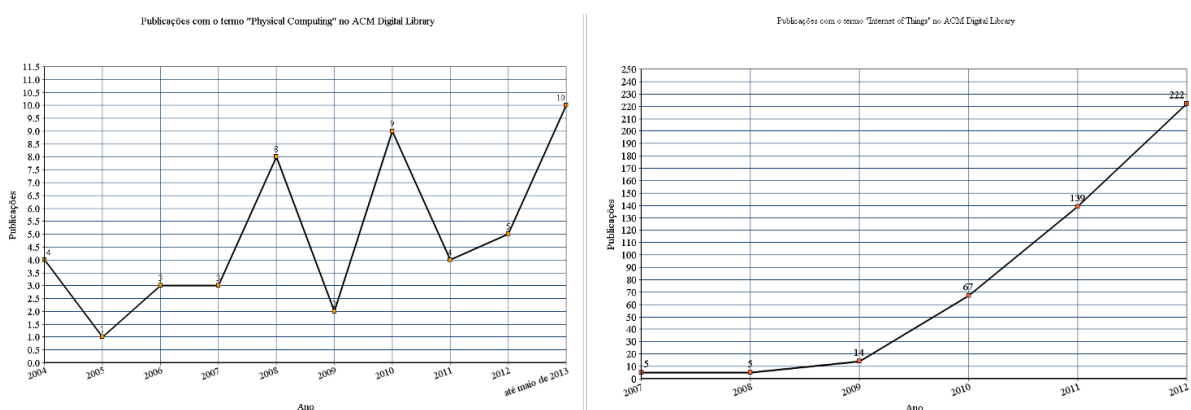


Tabela 1: Número de pesquisas com o termo "Physical Computing" e "Internet of Things" na biblioteca digital ACM.

Do ponto de vista da academia, as pesquisas relacionadas cresceram a partir de 2004. O gráfico à esquerda da Tabela 1 mostra uma pesquisa que realizamos e que indica um

4 <http://www.computer.org/portal/web/membership/13-Top-Trends-for-2013>

equilíbrio no número de pesquisas apresentadas com o termo “Physical Computing”, tendo em 2013 como o ano de maior inserção de artigos e publicações, com um total de 49 artigos até metade do ano. Já as publicações com o termo “Internet of Things”, na mesma biblioteca, foram iniciadas a partir de 2007 e apresenta um crescimento exponencial com quase 500 publicações em aproximadamente 8 anos de pesquisa, o que demonstra um tema bastante discutido e pesquisado.

As relações entre Computação Física e Internet das Coisas são diretamente proporcionais às técnicas e aplicações reais de comunicação através de objetos expandidos no dia a dia. Enquanto a Computação Física lida com as questões inerentes às técnicas de *design* e interação com objetos reais através da eletrônica básica, a Internet das Coisas reflete essas práticas em ações que podem alterar os comportamentos das pessoas e das cidades através da popularização de microcontroladores e dispositivos embarcados em objetos úteis no cotidiano das pessoas, como bolsas, geladeiras, sapatos, semáforos, postes, metrô etc.

2.3 Do movimento *open-source* para os *Makers*

We are all Makers. We are born Makers (just watch a child's fascination with drawing, blocks, Lego, or crafts) and many of us retain that love in our hobbies and passions.

[Anderson, 2012 p13]

A história de sucesso do movimento *open-source* nasce quando Richard Stallman funda a *Free Software Foundation*, em outubro de 1985, como uma organização sem fins lucrativos com o objetivo de produzir um sistema operacional para computadores onde qualquer pessoa pudesse usar, modificar e distribuir livremente [Weber, 2004].

Stallman especificou 4 liberdades⁵ essenciais para o *open-source*:

- Liberdade para executar o programa para qualquer fim;
- Liberdade para estudar como o programa funciona e modificar para suas necessidades;

5 <http://www.gnu.org/philosophy/philosophy.html>

- Liberdade para redistribuir cópias, tanto grátis quanto pagas;
- Liberdade de modificar e melhorar o programa e redistribuir as versões de modificações para o público para que outros tenham os mesmos benefícios a partir das suas alterações.

O *Arduino* é uma plataforma *open-source* para programação e prototipagem de objetos interativos através de sensores, atuadores e componentes eletrônicos. O projeto consiste em: um hardware *open-source*, onde é possível imprimir o circuito da placa sem custo e replicar livremente; e um software também *open-source*, capaz de escrever, compilar e enviar códigos para controlar a placa. [Mellis et al., 2007].

Impulsionados pela força desse movimento, Knörig [2009] argumenta que grupos de pesquisa e estudantes em escolas de arte e *design* ao redor do mundo⁶ expandiram o uso de softwares e hardwares como modelos para inovação, criatividade e colaboração. Essa tendência é inspirada a partir dos movimentos *underground* e das comunidades *DIY (Do-it-Yourself)* que desenvolviam e compartilhavam conteúdos diversos em formatos e suportes diferenciados para conquistar e criar redes de colaboradores [Anderson, 2012].



Figura 3: O perfil dos makers: dos quartos sujos às invenções criativas.

A plataforma *Arduino* dá suporte para uma comunidade de inventores que trabalham com desenvolvimento de protótipos interativos em universidades, laboratórios caseiros e *hackerspaces*. Esses grupos atuam na construção de instrumentos musicais, interfaces visuais

⁶ ITP/NYU, Royal College of London, FH Postdam etc.

interativas, instalações urbanas e no contexto do design de interação aplicado à objetos tangíveis [Mellis et al., 2007].

Anderson [2012] argumenta em seu livro *Maker: The New Industrial Revolution* que a ideia de fábrica mudou radicalmente por conta da democratização da inovação na Internet, em outras palavras, o acesso às tecnologias de baixo-custo, através de um modelo de compartilhamento de ideias e a prototipagem rápida de soluções, fizeram com que projetos mais experimentais desenvolvidos em laboratórios caseiros tomassem dimensões impressionantes, como foi o caso da impressora 3D⁷, que revolucionou o conceito da indústria em seu cerne.

Essa revolução apontada por Anderson [2012] impulsionou uma nova classe de produtos e bens de consumo desenvolvido por entusiastas e não-especialistas em tecnologias com perfil de inventores, os chamados *Makers*. Para Hartmann [2008], esse reposicionamento dos produtos é para atender as necessidades mais emergentes de uma nova classe de consumidores e produtores [Hartmann et al., 2008]

“From cooking to 3d printing, to making just about anything fly, Instructables⁸ became the recipient of countless hours of tinkering, soldering, stitching, frying, and fun, making just about anything.”

Instructable Team

Quando a Internet foi criada, em meados dos anos 60, ela foi pensada como um espaço virtual para compartilhar conhecimentos entre os próprios desenvolvedores: “os bens comuns intelectuais” [Barbrook, 2009]. Esse contexto de “facilidade no acesso de materiais específicos” é fruto desses movimentos, iniciados na academia, para tornar uma série de conteúdos mais acessíveis e com menos proibições. Para Lovell & Buechley [2010], as pessoas estão cada vez mais na Internet com objetivo de adquirirem novos conhecimentos e habilidades. Uma rica e crescente coleção de tutoriais DIY (faça-você-mesmo) postadas em sites tipo Instructables, Youtube, Vimeo e Make Magazine⁹ habilitam estudantes, iniciantes e

7 <http://www.makerbot.com/>

8 <http://www.instructables.com> - Plataforma de compartilhamento de projetos em eletrônica DIY.

9 <http://makezine.com/>

não-especialistas técnicos a aprenderem sobre uma diversidade impressionante de novas ideias, materiais e ferramentas.

Essa facilidade também transforma as diversas camadas de produção e remodela o papel dos laboratórios nas universidades [Nedic et al., 2003] e dos laboratórios caseiros [Oliver et al., 2009]. O baixo-custo do hardware e o fácil acesso a uma documentação em constante desenvolvimento, possibilita a prototipagem de experiências fora dos espaços educacionais e institucionais. Para Oliver et al [2009], ao invés de experiências em laboratórios tradicionais, as metodologias aplicadas na computação física favorecem as garagens e os laboratórios caseiros como ambientes potentes para experimentações de ideias e criação de novos produtos.

No artigo “*Lab Kits Using the Arduino Prototyping Platform*”, cerca de 40% dos estudantes com limitações em conhecimentos em eletrônica e com pouca experiência em programação se sentiram contagiados para experimentar o Arduino em laboratórios caseiros, 30% preferem os laboratórios tradicionais das universidades e os outros 30% não tinham preferência quanto ao espaço ideal para a prática. Essa remodelação dos espaços e das práticas impulsionaram os inventores e produtores focarem no desenvolvimento das ideias sem se preocupar com a técnica ideal para o projeto. A medida em que a Internet possibilita o compartilhamento de conteúdos em rede, mais pessoas com menos conhecimentos técnicos se identificam com os processos e as possibilidades de interação.

A prototipagem rápida, antes predominante apenas na indústria [Yan & Gu, 1996], passou a ser assunto de interesse dos *Makers* no momento em que a *web* se transformou numa potente produtora de serviços de alta performance [Hartmann et al., 2008]. A evolução dos processos de *design* amplia também as necessidades por ferramentas visuais mais expressivas e conectadas com as experiências dos usuários, principalmente para estimular essa nova classe de pessoas com interesses em produzir interações em ambientes visuais com a aceleração das experiências iniciais.

3. Desafios

Neste capítulo vamos aprofundar os desafios para a prototipagem de software e de hardware inerentes ao processo de experimentação na computação física. Na primeira seção serão elucidados os princípios e valores da prototipagem na concepção de um produto. Na segunda parte serão levantadas as metodologias, os passos e as dificuldades para realizar experiências de prototipagem com microcontroladores, sensores e atuadores. Além disso, serão esclarecidos os principais desafios da primeira experiência, desde o entendimento dos ingredientes necessários até a prática com as próprias mãos.

3.1 Princípios da prototipagem

Current designers of information appliances have to possess expert knowledge in a number of specialized areas, such as programming, embedded microcontrollers, and circuit design in order to build usable prototypes.
[Hartmann et al., 2005]

O processo de experimentação na computação física segue os mesmos princípios da prototipagem no contexto do design de produtos, onde as ideias são elaboradas, testadas e experimentadas para facilitar a tomada de decisão [Warfel, 2009]. Um protótipo é capaz de capturar a “intenção” e “simular” múltiplos estágios do design, o que permite um melhor estudo sobre o projeto em diferentes direções e olhares.

Para Warfel [2009], um protótipo é diferente dos outros trabalhos de imaginação porque ele é real, existe independentemente da nossa mente, isso significa que ele pode ser testado em vários cenários e pode ser avaliado em várias hipóteses. O autor estabelece valores para um protótipo:

(1) A prototipação é **generativa**, isto é, um único protótipo é capaz de gerar centenas de novas ideias, algumas brilhantes e outras não. As que não são brilhantes podem ser catalisadas para se transformarem em plausíveis um dia;

- (2) Um protótipo tem o poder de **apresentar, conversar e experimentar**, é um modelo ou simulação de um sistema final, ao contrário de uma documentação específica. Os protótipos vão mais longe porque realmente é possível experimentar, não apenas olhar;
- (3) A prototipagem **reduz os erros** de interpretação por se configurar como uma representação tátil de um sistema a ser desenvolvido;
- (4) A prototipagem **reduz o tempo, esforços e dinheiro**.
- (5) A prototipagem **reduz perdas**, uma vez que um protótipo traz foco e reduz riscos para o produto.
- (6) A prototipagem **habilita valores do mundo real**.

No artigo *Reflective Physical Prototyping through Integrated Design, Test, and Analysis* [Hartmann et al., 2006], a prototipagem é considerada uma atividade pivô que estrutura a inovação, a colaboração e a criatividade no *design*. Os protótipos encorpam os projetos e os habilitam para o teste real, ou seja, é através dos protótipos que os inventores estudam e se aprofundam a respeito dos problemas que estão para resolver.

“Reflective practice, the framing and evaluation of a design challenge by working it through, rather than just thinking it through, points out that physical action and cognition are interconnected.”

[Hartmann et. al, 2006, p1]

Um dos itens mais utilizados na prática de prototipagem em eletrônica para interações físicas com sensores e atuadores é a *breadboard*, em português é chamada de *protoboard* ou placa de ensaio¹⁰ [Knörig et al., 2009]. A placa acelera o processo de investigação porque permite plugar componentes sem necessidade de soldar e habilita vários ciclos de iteração com o protótipo, até que o circuito tenha sua solução contemplada. Para Knörig et al., [2009], a *breadboard* é essencial no aspecto do processo de *design* porque permite mudanças rápidas, a possibilidade da tentativa e erro e a cultura do passo-a-passo. Esses recursos são importantes para o exercício da prototipagem em projetos de computação física principalmente para quem

¹⁰ https://pt.wikipedia.org/wiki/Placa_de_Ensaio

não tem conhecimento técnico em engenharia e eletrônica.

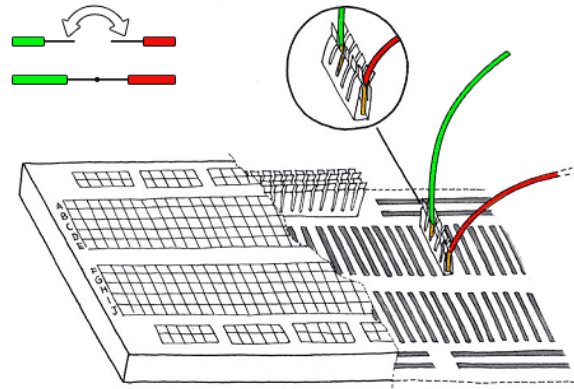


Figura 4: Uma breadboard por dentro

3.2 O contexto da prototipagem na eletrônica

Desde a década de 1980, a maior transformação no universo da computação foi fazer do computador uma máquina fácil de usar, com a perspectiva de prover uma utilização com menor rigor cognitivo e maior foco nos ciclos mentais do usuário [Soloway, 1994]. Mais na frente, Norman [1998] comenta que a emergência dos novos dispositivos embarcados e o desenvolvimento da rede sem fio amplificam o domínio de atuação da computação para além dos laboratórios e escritórios: as práticas tecnológicas passam para outras conjunções da vida cotidiana [Norman, 1998].

O processo evolutivo das GUI (*Graphical User Interface*) voltadas para usuários finais e a confluência entre o *design* de interação e a Computação Física [Ishii, 1997], influenciaram para tornar o desenvolvimento de dispositivos eletrônicos embarcados mais acessíveis para usuários não-especialistas. As GUI para microcontroladores foram historicamente¹¹ compostas por desafios nas etapas da programação e prototipagem física dos circuitos eletrônicos. As primeiras linguagens de programação habilitadas foram as versões nativas do *Assemblers*, *C*, *BASIC* e outras linguagens mais populares, como *Java* e *Logo*. Esses ambientes possuem pontos fortes e fracos quando se trata das facilidades de uso, funcionalidades e *frameworks* para desenvolvimento. As barreiras para programar de acordo com a sintaxe e a exigência de

¹¹ Vide [Barragán, 2004] páginas 24 à 32.

um pensamento computacional desestimularam os *designers* e os artistas com o desejo de desenvolver ideias através de protótipos eletrônicos e interativos [Barragán, 2004].

Apesar de soluções criadas pela indústria para encorajar profissionais de design para a área de interações físicas, como a ferramenta *BASIC Stamps*¹², da *Parallax Inc.*, ainda existem barreiras nos ciclos de iterações que compreende as etapas de desenvolvimento de código para o microcontrolador e da prototipagem do circuito eletrônico.

As ferramentas de prototipagem ajudam a reduzir as dificuldades do trabalho com a eletrônica e a expandir o número de pessoas com desejo de experimentar, mas os custos para adquirir as ferramentas, as proibições de licença de uso e as dificuldades inerentes das linguagens de programação para os dispositivos, tornam os ambientes direcionados apenas para um público especialistas [Mellis et al., 2007]. Para interpretar as barreiras dos usuários iniciantes sem conhecimentos aprofundados em tecnologias e eletrônica, Mellis [2007] classifica os passos e dificuldades para se realizar uma operação básica com microcontroladores:

- Encontrar um microcontrolador em particular;
- Entender as necessidades do circuito;
- Comprar os componentes para realizar a experiência;
- Realizar o *download* dos aplicativos para acessar e programar o microcontrolador;
- Entender os detalhes técnicos da conversa entre o microcontrolador e o computador;
- Instalar diversos *drivers* para configurar a comunicação com o microcontrolador;
- Comprar ou construir um dispositivo externo para programar o microcontrolador, o que requer uma grande quantidade de leitura técnica de *datasheets*;
- Escrever o código e trabalhar com argumentos em linha de comando para compilar e fazer o *upload* do código para a placa.

12 <http://www.parallax.com/?tabid=295>

Com a referência nas dificuldades acima, o projeto Arduino surgiu como uma perspectiva de encorajar *designers* e não-especialistas a criarem protótipos rápidos em computação física com o objetivo de experimentar e desenvolver projetos interativos com componentes físicos de fácil acesso e de baixo custo. O projeto foi desenvolvido dentro do Instituto de *Design* de Interação *Ivrea* no norte da Itália, uma escola com foco em *interfaces* visuais, objetos físicos e produção faça-você-mesmo.

O projeto permite que usuários criem protótipos eletrônicos de um jeito mais fácil, sejam embarcados ou conectados no computador. A plataforma possibilita também a leitura de uma grande quantidade de sensores e atuadores sem a necessidade de relaciona-lo a um kit de hardware específico. A arquitetura do projeto facilita a comunicação do microcontrolador com a IDE (*Integrated Development Environment*) em execução no computador e acelera as experiências em design de interação física com dispositivos produzidos pela própria comunidade.

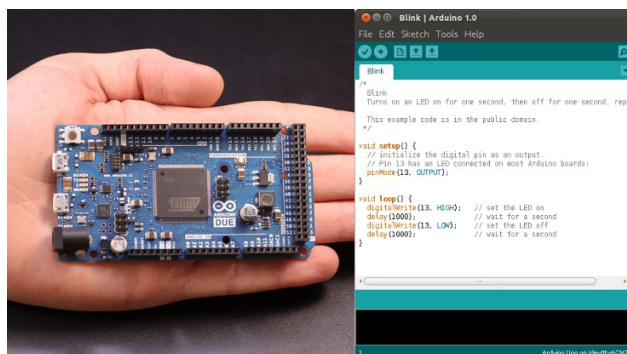


Figura 5: Projeto Arduino com um hardware físico e uma IDE para programar as entradas e saídas digitais e analógicas.

O ambiente de programação do Arduino utiliza a linguagem textual *Wiring*, com vários aspectos baseados nos princípios do Processing¹³, principalmente na estrutura da linguagem, que não necessita conceitos avançados de programação como classes, objetos e métodos para o desenvolvimento de interações físicas por parte dos usuários. A conexão com os aspectos do Processing é estratégica pela grande quantidade de artistas, inventores e pesquisadores em

13 O Processing foi criado como uma ferramenta aplicada para o ensino dos fundamentos de programação visual através da metáfora de sketches (esboços) e ferramenta de prototipagem rápida. [Reas & Fry, 2007]

todo o mundo que utilizam a ferramenta como ambiente de experimentação. Hernando Barragán, o desenvolvedor do *Wiring* tensiona sobre a aproximação das experiências do Processing para o contexto da computação física:

How could I extended this experience {Processing} to hardware programming and prototyping with electronics?

[Barragán, 2004, p 36]

No artigo “*Arduino for Teaching Embedded Systems. Are Computer Scientists and Engineering Educators Missing the Boat?*” o autor, Jamieson [2010], identifica alguns benefícios no uso da plataforma Arduino no contexto educacional:

- Fácil de usar;
- Vários exemplos para controlar periféricos inseridos na IDE;
- Vários projetos *open-source* para seguir como exemplo;
- Trabalha em diferentes sistemas operacionais: *Windows*, *Mac* e *Linux*;
- O hardware é acessível, com possibilidade de construir em casa ou comprar pronto;
- O software é livre;
- Manutenção barata, o microprocessador é fácil de ser trocado;
- Estudantes podem realizar prototipagem mais dinâmicas;
- Pode ser programado em inúmeras linguagens¹⁴.

Numa observação mais detalhada, o projeto Arduino agrega os argumentos de “flexibilidade” e “fácil de usar” como uma ponte para apresentar aos usuários o conceito da prototipagem rápida de projetos de interações físicas através de práticas inspiradas do movimento *open-source* que modelou uma história de sucesso através de uma mudança de atitude nos usuários.

¹⁴ Através de protocolos de comunicação serial entre o computador e o microcontrolador, como por exemplo o projeto Firmata. [vide o capítulo 4]

Esses argumentos implicam na transformação dos perfis dos educadores, artistas e não-especialistas para *designers*, *Makers* e inventores que constroem projetos baseando-se em conteúdos especializados em tutoriais online (*how-to*), produzidos pela própria comunidade com o objetivo de auxiliar e facilitar o acesso aos circuitos eletrônicos e aos códigos-fontes.

Nedic et al [2003] estabelece cinco pilares que formam uma metodologia voltada para as práticas e habilidades nas pesquisas e desenvolvimentos de projetos em computação física:

- (1) Testar os conhecimentos teóricos;
- (2) Trabalhar colaborativamente;
- (3) Interagir com equipamentos;
- (4) Estudar através da tentativa e erro;
- (5) Realizar análise de dados com os aparatos.

Os usuários iniciantes estão dispostos a estudar a linguagem do Arduino com a metodologia do passo-a-passo através das referências disponíveis na Internet e a partir disso, tentar usar os componentes eletrônicos e montar em seus protótipos. Essa iniciativa autodidata é reforçada pelo pouco tempo disponível em *workshops* e cursos específicos para praticar as repetições necessárias para o amadurecimento no contexto da prototipagem com microcontroladores, *breadboard*, fios, sensores e atuadores.

Para Sarik et al [2010], a pouca experiência de estudantes em contato inicial com o *Arduino*, aumentou a preferência por conteúdos prontos com o objetivo de adquirir uma aprendizagem rápida nas ferramentas e técnicas necessárias para completarem seus experimentos. Jamieson [2010] propôs três atividades de prototipagem e os resultados mostraram que os estudantes escolheram o *Arduino* como microcontrolador ao invés de microprocessadores *PIC*¹⁵ e *FPGA*¹⁶. O autor comenta que mais de 90% da classe optou pelo *Arduino* em seus projetos, no entanto, foram notadas limitações no poder computacional do dispositivo em relação à conexões com dispositivos como *Xbox Kinect* e em outras interatividades com o computador.

15 <http://www.microchip.com/pagehandler/en-us/products/picmicrocontrollers>

16 http://en.wikipedia.org/wiki/Field-programmable_gate_array

3.3 Os percursos típicos das experiências em computação física

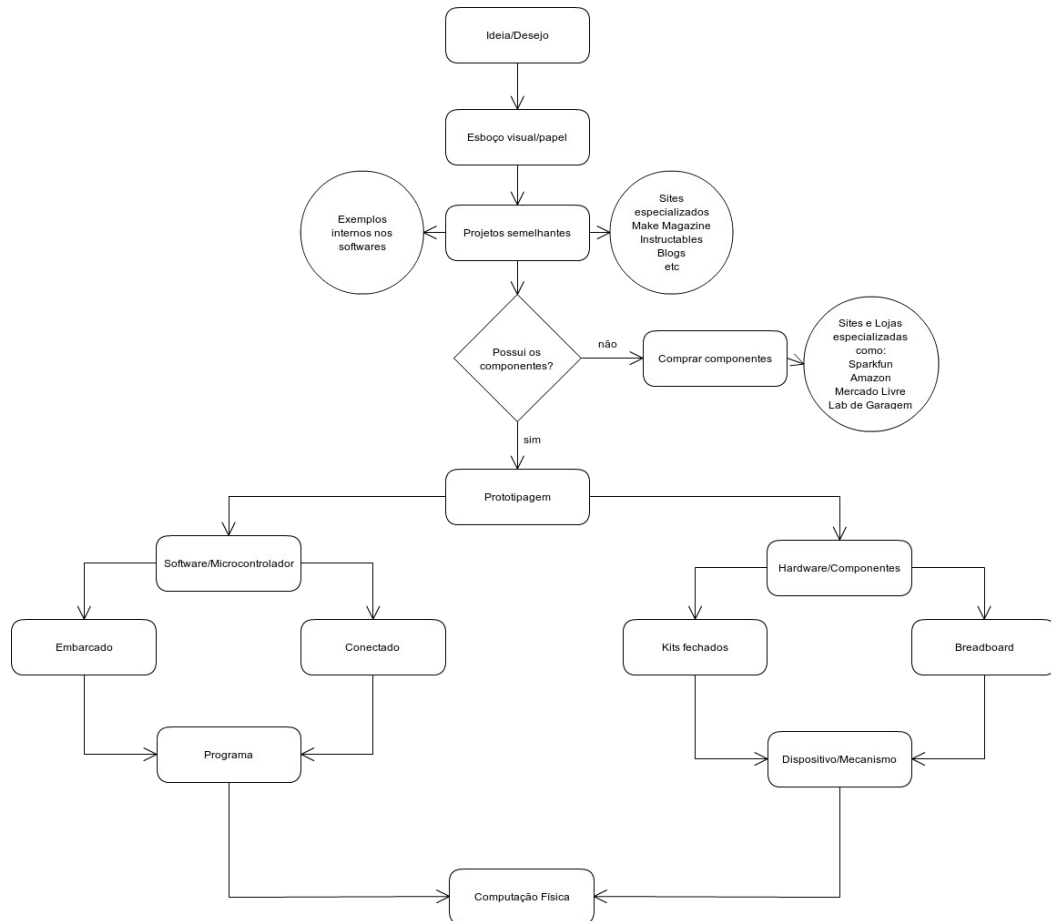


Figura 6: Atividades típicas para projetos de computação física.

A partir do momento em que usuários não-técnicos são contemplados no processo de experimentação no contexto das interações com objetos físicos em ambientes reais, surgem situações típicas para as práticas neste contexto. A Figura 6 representa um fluxograma, desenvolvido durante as investigações desta dissertação, que traça um percurso comum em que usuários percorre para realizarem experiências de protótipos em computação física. O fluxograma inicia a partir de uma ideia ou desejo e caminha, desde as etapas dos primeiros esboços, até a prototipagem do hardware e do software para completar o ciclo de interação.

As primeiras experiências nos fluxos de atividades da computação física elucidam diferentes desafios. O primeiro desafio está na concepção, pesquisa de projetos relacionados e compra

de materiais, o que exige dos usuários práticas em pesquisas na Internet. Um segundo e importante desafio está na etapa da prototipagem, multiplicada em duas instâncias: a programação do software no microcontrolador e a prototipagem dos dispositivos e componentes eletrônicos, combinadas com a construção de um mecanismo. Não existe um consenso em relação como o usuário percorre o fluxo, mas é muito comum iniciar a prototipagem pelo circuito eletrônico para depois interpretar os requisitos necessários para realizar as interações via software.

Para Conradi et al [2010], as primeiras experiências em desenvolvimento de protótipos na computação física presumem duas importantes buscas:

- (1) desenvolvimento do hardware: os sensores e atuadores necessitam ser propriamente conectados ao microcontrolador em um circuito eletrônico. Essa etapa requer um conhecimento básico dos princípios da eletrônica e conhecimento das ferramentas de prototipagem como a *breadboard* e o multímetro.
- (2) desenvolvimento de software: o microcontrolador necessita ser programado para ler e controlar os componentes acoplados nos seus pinos. Essa etapa requer um conhecimento em algoritmos e noções de sintaxe de programação para desenvolver as interações com os protótipos.

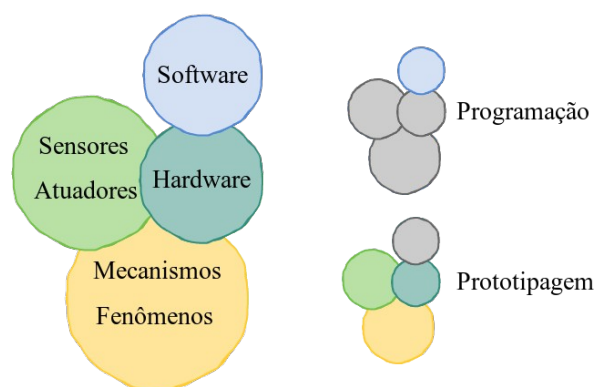


Figura 7: Os componentes que formam as etapas de programação e prototipagem na Computação Física.

De fato, existem desafios concretos para as primeiras experiências quando se trata de usuários novatos. Na camada de software, o “pensamento computacional” é um dos pré-requisitos para dar sentido às relações entre os objetos físicos e os dados digitais. Wing [2006] sintetiza que o poder de abstração expandiu de um assunto puramente técnico e especializado para o domínio do *design* estratégico, principalmente quando se colocado nessa situação de prototipagem para a construção de novos produtos interativos [Wing, 2006].

O artigo *Programming Enviroments for Novices* [Guzdial, 2002] explica que a especialização dos ambientes de programação para não-especialistas tem entre suas premissas a programação como uma habilidade difícil de estudar e aplicar. O pensamento abstrato e os problemas com estruturas de *loop*, condicionais e a compilação do código estão entre os principais fatores para afastar os iniciantes das experiências iniciais. Na visão de Papert [1980], esses problemas são resolvidos com diferentes níveis de acesso às linguagens: nível fácil de iniciar, nível avançado com oportunidades de aumentar a complexidade dos projetos com o tempo e escalabilidade para dar suporte à diferentes tipos de projetos e pessoas com interesses distintos [Papert, 1980].

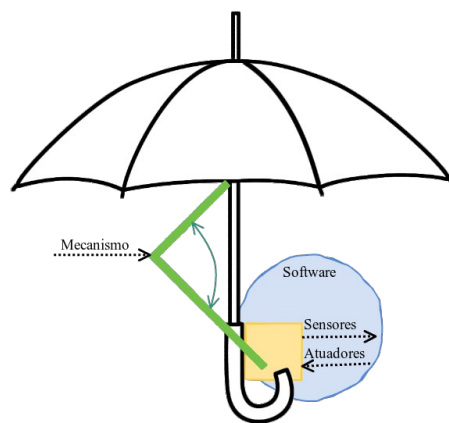


Figura 8: Exemplo de aparato com os ingredientes da Computação Física desenvolvido pelo autor desta investigação.

A Figura 8 representa um dispositivo equipado com sensores que percebem mudanças no ambiente e acionam atuadores que controlam um mecanismo acoplado num guarda-chuva. O

software interpreta os dados dos sensores e decide o ângulo de abertura do servo-motor. Este projeto serve como um exemplo dos ingredientes requeridos para o desenvolvimento de protótipos em computação física.

Em relação à prototipagem do dispositivo físico, os usuários necessitam de habilidades para pesquisar e estudar os *datasheets* dos componentes eletrônicos, como também para montar o circuito eletrônico, seja por meio de kits de sensores e atuadores *plug-and-play* ou placas de prototipagem (*breadboard*) conectadas com o microcontrolador. Por fim, acoplar todo o hardware num mecanismo para completar o ciclo de implementação do projeto. Após essa fase, o usuário está apto a decidir as diferentes interações possíveis com o dispositivo. Para Resnick [2009], a influência do digital requer algumas habilidades para a criação e invenção com componentes eletrônicos de baixo-custo, isso inclui, a pesquisa através dos tutoriais online e as interações em comunidades específicas através de lista de e-mail e fórum [Resnick, 2009].

Os projetos com objetos físicos interativos estão cada vez mais presentes em revistas como a *Make Magazine* e em comunidades como a Instructables, como comenta Sivek [2011]. Se bem observada, esta fase de prototipação de projetos com dispositivos eletrônicos ainda é resolvida nos mesmos aspectos dos movimentos de bricolagem da década de 1970, onde os zines e as revistas com imagens dos esquemáticos eletrônicos serviam como fonte para os entusiastas [Anderson, 2012]. Os tutoriais online servem de suporte para os não-especialistas através de um passo-a-passo no estilo “receita de bolo”, com a montagem de uma solução previamente experimentada. Após o teste e a prova de conceito, os usuários se sentem aptos a realizar as adaptações necessárias para o seu projeto em específico [Lovell & Buechley, 2010].

Nesse contexto, a plataforma Fritzing¹⁷ surge como uma ferramenta para auxiliar os designers e inventores a documentarem melhor os circuitos desenvolvidos e experimentados. O projeto nasce a partir das dificuldades inerentes das especializações nas ferramentas de EDA (*Electronic Design Automation*) e da necessidade por ambientes que dialoguem com as metáforas do mundo real (dos *Makers*). A ferramenta amplia as possibilidade de

¹⁷ <http://fritzing.org/>

documentação de projetos e a fabricação dos circuitos em grande escala e sua impressão através da visualização em PCB (*Printable Circuit Board*). O projeto também levanta questões a respeito da durabilidade e da escalabilidade dos protótipos desenvolvidos [Knörig et al., 2009].

Conradi et al [2010] argumenta que dispositivos mais sofisticados, como Arduino, permite que usuários criem projetos mais complexos, mas para isso, são necessários conhecimentos em eletrônica para montar o protótipo através de uma *breadboard* junto com componentes analógicos e digitais. O autor comenta também que tutoriais online e softwares de representações digitais de circuitos, como o Fritzing, ajudam na aprendizagem de habilidades mas não proporcionam interações semânticas necessárias para acelerar as experiências.

Apesar de soluções para facilitar o acesso aos microcontroladores por meio de plataformas como Arduino e para documentar o projeto, como o Fritzing, estas ferramentas não atendem plenamente as etapas de prototipagem de software e de hardware, o que desestimula as primeiras experiências de usuários iniciantes não-especialistas técnicos. As pesquisas de projetos semelhantes, descrito no fluxograma da Figura 6, desaceleram as experiências iniciais e moldam os projetos a partir de situações pré-experimentadas. Além disso, a IDE do Arduino possui uma linguagem textual, o que exige pré-requisitos técnicos de computação e o ambiente Fritzing só faz sentido para os usuários que já estão inseridos no contexto da eletrônica e sabem quais componentes serão necessários bem como as conexões entre a *breadboard* e o microcontrolador.

Diante disso, pode-se perceber duas situações em relação às barreiras da primeira experiência para interagir com a computação física:

- a dificuldade na instalação, configuração e na curva de aprendizagem do ambiente de programação e da linguagem textual para o microcontrolador;
- a lacuna entre o desenvolvimento do software e a etapa da concepção e montagem do circuito, o que maximiza a responsabilidade dos tutoriais online e dos *datasheets* como o único caminho para modelar o circuito eletrônico de um protótipo.

3.4 Princípios para uma abordagem visual integrada

A redução do ciclo de tempo no desenvolvimento e melhorias num produto tem se transformado em objetivos estratégicos para vários projetos de tecnologias [Cohen et al., 1996]. Mesmo com a explosão da cultura *copy-and-past*¹⁸ e o *frankensteining*¹⁹ de códigos e circuitos, aplicados por boa parte dos desenvolvedores com interesse em criar protótipos rápidos, as abordagens das ferramentas de prototipagem em eletrônica ainda não são suficientes para acelerar as experiências iniciais, melhorar o *time-to-market* e transformá-las em atividades mais expressivas [Hartmann et al., 2008].

A transformação do *browser* em suíte de desenvolvimento de projetos integrados, facilitados por novas arquiteturas como a do Firefox e do Chrome [Nyrhinen et al., 2010], possibilita a construção de softwares mais ágeis para os usuários, ampliando o desenvolvimento de projetos por uma classe de desenvolvedores acostumados com o jeito fácil de plugar serviços nessas aplicações web, como é o caso de projetos como Wordpress²⁰ e Google Drive²¹. Linguagens e bibliotecas como Javascript²², Node.js²³, jQuery²⁴ e diversos projetos hospedados no Github²⁵, podem ser as novas fontes de recursos tecnológicos aplicados para o desenvolvimento de interfaces criativas, esclarece Rogers [2013].

Alguns argumentos a respeito das barreiras técnicas para designers e programadores são visualizadas por Snavaes [2000]. O autor comenta a respeito de duas situações criadas pelo desenvolvimento tecnológico a partir do ponto de vista do usuário com intenção de criar softwares interativos sem nenhum *background* em tecnologia:

- (1) o processo de simplificação e abstração criada pelas ferramentas tornou praticamente impossível a experimentação e prototipação de soluções interativas que eram tecnicamente viáveis;

18 A cultura *copy-and-past* levantada por Hartmann et al [2008] está relacionada com o ato dos desenvolvedores copiarem códigos prontos como uma forma de testar APIs e desenvolver protótipos através de uma engenharia reversa.

19 Está relacionado com a mistura de vários códigos de diferentes projetos para gerar um novo artefato.

20 <http://wordpress.org/>

21 <http://drive.google.com/>

22 <http://www.w3schools.com/js/>

23 <http://nodejs.org/>

24 <http://jquery.com/>

25 <http://github.com/>

(2) a imagem criada pelos usuários a respeito da área do *design* que responde pela interação é fragmentada e incompleta. Essa situação foi modelada tanto pelas ferramentas utilizadas, quanto pelas possíveis soluções aplicadas. Com isso, as soluções que eram tecnologicamente possíveis foram inimagináveis pela maioria dos usuários.

Com objetivo de acelerar as primeiras experiências em computação física, os usuários iniciantes necessitam então de ferramentas visuais expressivas que dialoguem de forma fácil e integrada com a programação dos componentes e a prototipagem do circuito eletrônico. A programação por módulos visuais plugáveis como na web e uma simulação semântica da montagem do circuito eletrônico são tópicos interessantes para a concepção de uma abordagem integrada para prototipagem em computação física.

Os princípios norteadores para o desenvolvimento de um ambiente visual para prototipagem rápida em computação física são estabelecidos em dois conjuntos. O primeiro situa-se em relação à **usabilidade** e o segundo em relação à **abrangência** da ferramenta.

Sobre a **usabilidade**, destaca-se:

- **Integração entre o hardware e o software:** um sistema capaz de dialogar com a prototipagem do software com a prototipagem do hardware de forma integrada;
- **Programação visual:** o ambiente deverá permitir a programação dos componentes de entrada e saída como módulos gráficos, assim como no Pure Data [Puckett, 1997];
- **Feedback em tempo-real:** o ambiente deverá manipular componentes físicos em tempo-real e possibilita processamentos de informações em rede para conectar sensores distantes geograficamente.

Sobre a **abrangência**, destaca-se:

- **Portabilidade:** o ambiente deverá funcionar em qualquer sistema operacional, independente de sua distribuição e configurações. Deve ser propiciado meios para que a portabilidade se estenda para dispositivos móveis como celulares e *tablets* para que as experiências se tornem cada vez mais embarcadas.

4. Trabalhos relacionados

Os argumentos levantados nas seções deste capítulo são orientados a partir do fluxograma apresentado na Figura 6. Foram levantados os ambientes de software que preenchem as etapas descritas no fluxograma para obter uma radiografia e apontar possíveis lacunas não resolvidas por meio de software. A seção 4.1 apresenta uma síntese dos ambientes desenvolvidos para prototipagem em computação física. A seção 4.2 trata dos ambientes aplicados na prototipagem do software. São descritas as ferramentas que atuam em ambientes embarcados e em ambientes em tempo-real. A seção 4.3 apresenta os ambientes aplicados na prototipagem de hardware. São expostos os kits de hardware específico e o Fritzing como solução única para preencher a prototipagem por meio da breadboard. Na última seção, serão realizadas análises nas aplicações de acordo com os princípios norteadores.

4.1 Ambientes para prototipagem em computação física

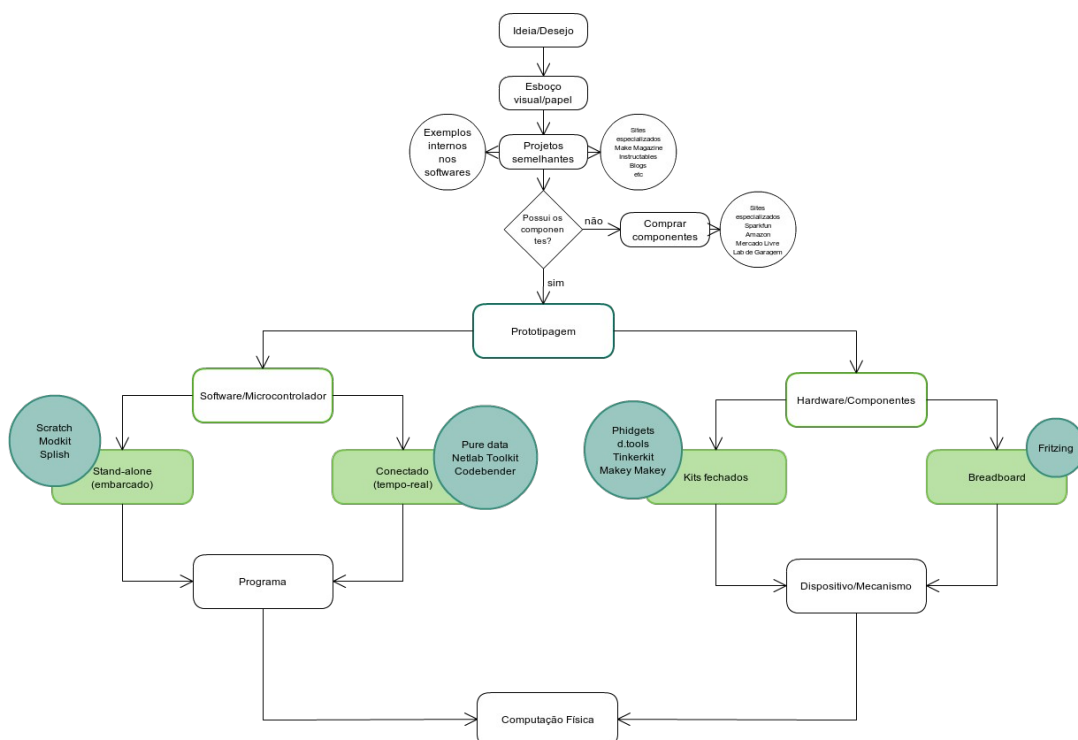


Figura 9: Projetos relacionados com as etapas de prototipagem de protótipos em computação física.

Os softwares na Figura 9, correspondem as etapas de prototipagem de software e de hardware. Os softwares que funcionam no modo embarcado necessitam de ser compilados na linguagem específica do microcontrolador. Já os softwares que funcionam de forma conectada, permitem a prototipagem de software em tempo-real com o microcontrolador por meio de *middlewares*. Os ambientes para prototipagem de hardware são classificados como kits fechados e *breadboard*. Os kits são projetos de hardwares *plug-and-play* com entradas e saídas prontas para uso por meio de ambientes preparados. Já a *breadboard* corresponde ao uso de práticas típicas no universo dos Makers com sensores e atuadores acessíveis.

Os ambientes relacionados utilizam de linguagens visual para possibilitar a programação através de elementos gráficos e expressões visuais com arranjos espaciais de textos e símbolos [Lin & Kuo, 2010] . Para Kaučič & Asič [2011], a programação visual é uma área bem discutida no contexto dos desafios da programação para não-especialistas com diversas aplicações possíveis:

- linguagem específica e ambientes para realizar tarefas que atendam determinados critérios;
- desenvolvimento de games;
- controle de produtos industriais;
- visualização de dados e;
- ferramentas visuais sofisticadas com linguagem *high-level* para atender um propósito geral ou específico.

Myers [1986] cita que alguns sistemas de programação visual obtém sucesso quando demonstram que não-especialistas podem criar programas complexos com pouco tempo de treinamento, principalmente por ter descrições mais fáceis das ações e um nível de abstração mais alto, sem muita ênfase nos problemas de sintaxe, o que torna a experiência mais atrativa para iniciantes [Myers, 1986]. Em contraposição, para Resnick [2009] as primeiras experiências com programação visual aplicadas para não-especialistas apresentam problemas estruturais [Resnick, 2009]. O autor cita que existem fatores importantes para o insucesso das

primeiras iniciativas de ferramentas:

- as primeiras linguagens de programação eram muito difíceis de usar e muitos usuários não conseguiam dominar sua sintaxe;
- as práticas em programação eram muitas vezes introduzidas através de atividades que não eram conectadas com os interesses dos usuários (gerar números primos, desenhar linhas e vetores por exemplo);
- as práticas em programação eram muitas vezes introduzidas em contextos onde ninguém poderia fornecer orientações para corrigir os erros ou incentivar explorações aprofundadas quando as atividades deram certo.

Conradi et al [2010] também percebe contrapontos em relação aos benefícios da simplificação na programação e nos processos de prototipagem:

- a programação visual facilita o desenvolvimento mas em geral são aplicações com possibilidades limitadas. Consequentemente, os usuários não sentem necessidade de investigar novos conhecimentos para produzirem projetos mais complexos.
- os kits fechados, como Lego, incluem uma grande quantidade de componentes prontos que podem ser conectados facilmente. Porém, assim como nas ferramentas de software, os usuários encontram limitações quando tentam usar um sensor ou um atuador específico não incluído no pacote da *toolkit*.

Essa série de problemas impulsionam o desenvolvimento de ferramentas visuais voltadas à acelerar as experiências em programação de sensores e atuadores através de microcontroladores e componentes fáceis de usar. Os projetos de interfaces visuais descritos no fluxograma acima são frutos da necessidade por ambientes mais acessíveis e capazes de dialogar com as tendências criadas pela revolução dos *Makers*, que amplificam as produções de *gadgets* de uma forma cada vez mais rápida e independente da indústria e da fábrica. [Anderson, 2012].

4.2 Ambientes de prototipagem de software

Nesta seção serão apresentados os ambientes de prototipagem de software que permitem o controle do microcontrolador de forma embarcada ou em tempo-real. Essas duas abordagens de programação são descritas nas sub-seções seguintes.

4.2.1 Os softwares *stand-alone*

As ferramentas apresentadas nesta seção são aplicadas para iniciantes em programação que utilizam o Arduino para desenvolver pequenas interações embarcadas. Os ambientes utilizam aspectos da programação visual em blocos e ícones, semelhante a um quebra-cabeça tipo Lego, com o objetivo de facilitar e aproximar dos usuários os conceitos de algoritmos e iterações computacionais.

O **Scratch** é um ambiente visual de programação com histórias, jogos, música e artes, criado pelo *Lifelong Kindergarten Group*²⁶ do *MIT Media Lab*. O software é para a faixa etária de 8 a 16 anos mas também é utilizado por grupos e pessoas de todas as idades. O ambiente suporta a criação de diferentes artefatos como imagens, áudio e textos. As principais características são:

- utiliza a metáfora de blocos onde os usuários constroem *scripts* através de conexões parecidas com um quebra-cabeça. Comandos e dados são representados por blocos de diferentes formatos que se encaixam uns aos outros criando relações entre eles. A interface do usuário é inspirada no *Logo Microworlds*;
- manipulação de outras mídias como imagens, animações, vídeos e áudios;
- compartilhamento de projetos *online* e modo de exportação para vários níveis de utilização;
- integração com o mundo físico inicialmente através da *LEGO/Logo* e *Programmable Bricks* com possibilidade de controle dos comportamentos dos objetos dentro do mesmo ambiente.

26 <http://llk.media.mit.edu/>

A integração com o Arduino foi desenvolvida pelo grupo *Citilab Smalltalk Programming Group* e é chamada de S4A²⁷ (*Scratch for Arduino*). Trata-se de uma biblioteca que fornece suporte a conexão do Arduino para controle de sensores e atuadores através do Scratch. A biblioteca funciona nas versões *Arduino Diecimila*, *Duemilanove* e *Uno*.

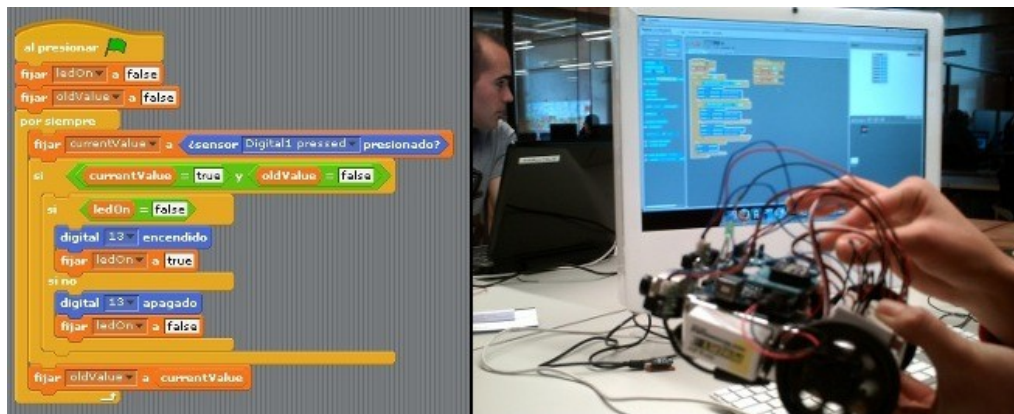


Figura 10: S4A Scratch com a biblioteca que envia dados para o Arduino.

Existem alguns caminhos para desenvolver interação entre o Arduino e o Scratch. É possível criar uma conexão com o microcontrolador por um objeto virtual do Arduino dentro do Scratch. Desta forma, o objeto se que conecta com as portas de entrada e saída do microcontrolador e permite o acesso às portas. Uma outra opção está no envio de dados para atuadores e recebimento de dados dos sensores em tempo-real. Esse modo só funciona se utilizar um *firmware* específico para *PicoBoard*, com instruções pelo site oficial²⁸ do Arduino.

Na perspectiva dos inventores e entusiastas da classe de desenvolvedores emergentes, o **Modkit** é uma ferramenta para iniciantes que combina a programação em blocos, inspirada no Scratch, com o acesso ao microcontrolador Arduino por meio do *browser*. O ambiente habilita pessoas a criarem interfaces físicas com associação direta com as produções ricas em diversidades, conteúdos e expertises das comunidades do Scratch e do Arduino [Millner et al., 2011].

O ambiente é *web-based*, ou seja, ele utiliza o *browser* e a Internet para rodar serviços de interface com o usuário e comunicação com o microcontrolador. O sistema também é

27 <http://seaside.citilab.eu/>

28 <http://arduino.cc/>

orientado à blocos gráficos com paletas de comandos e categorias de acordo com o tipo de função. O ambiente permite que o usuário selecione qual hardware está conectado no computador e oferece comandos específicos para o dispositivo escolhido.

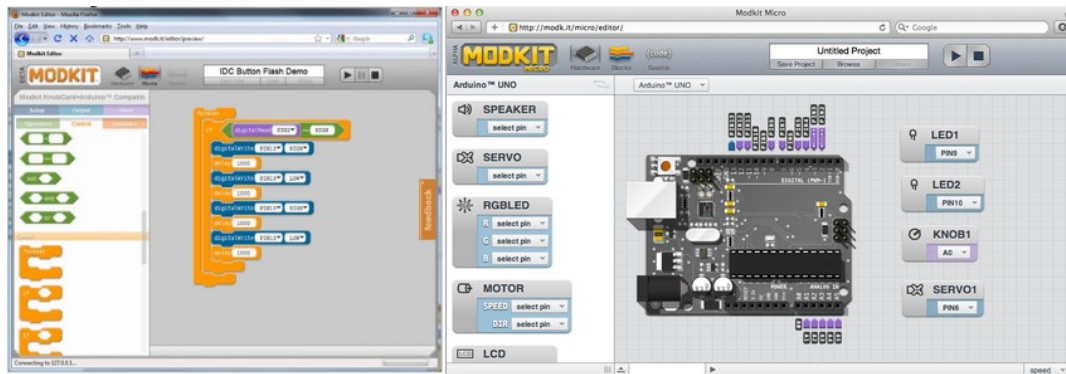


Figura 11: Na esquerda, a área de programação orientada a blocos gráficos, semelhante ao Scratch e na direita o ambiente de setup dos componentes.

A interface apresenta uma imagem de um hardware (Figura 11 à direita) conectado a um computador onde é possível configurar entradas e saídas dos pinos graficamente, uma inovação no contexto de ferramentas para a computação física, segundo Millner et al [2011]. As cores e formas dos blocos sugerem conexões entre os diferentes funções, o que elimina riscos de erro de sintaxe. A versão do Modkit não suporta multi-tarefas (*multi-threads*), por isso, é necessário transferir o programa para a placa através do botão *play* na parte superior da interface. Essa arquitetura não permite o funcionamento em tempo-real entre a interface e os objetos físicos. Os programas podem ser escritos em modo de texto e a ferramenta realiza a transferência do código para o microcontrolador através da instalação de uma biblioteca específica no computador.

De um outro ponto de vista, o **Splish** é um ambiente gráfico baseado em ícones desenvolvido por Kato [2010] e tem o objetivo de acelerar as experiências com o Arduino. O ambiente utiliza a programação visual para auxiliar usuários em cursos de introdução à programação e *workshops* de eletrônica. A ferramenta foi escrita em JavaFX, uma linguagem de *script* aplicada no desenvolvimento de ferramentas gráficas. O sistema comunica com o microcontrolador através da API *Java Communication* implementada para o ambiente *Windows*.

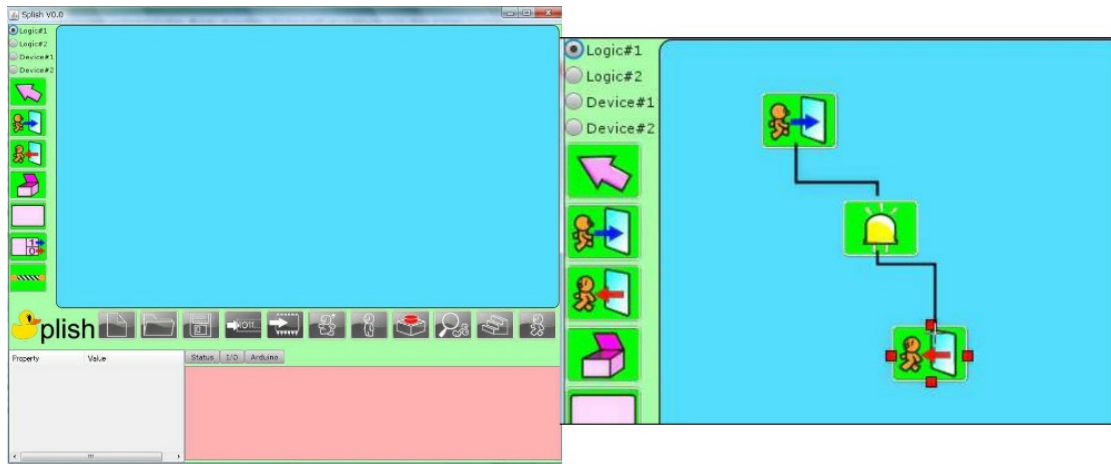


Figura 12: Interface do usuário e ambiente de programação do Splish.

Kato[2010] cita algumas das principais ênfases adotada no ambiente:

- **visualização:** a construção do programa como estruturas de controle (*if...then...else*, *while*) e os componentes de entrada e saída representadas por ícones.
- **funcionalidade:** funcionalidades do Arduino implementada para que os usuários migrem para a linguagem original quando necessário;
- **dispositivo I/O:** não há limite para usar as entradas e saídas do microcontrolador;
- **monitoramento:** o status do microcontrolador é representada no PC quando a placa é conectada;
- **depuração:** os usuários podem debugar os erros do programa interativamente;
- **stand-alone:** os programas continuam em execução mesmo se a placa for desconectada do PC;
- **portabilidade:** o ambiente é independente de um sistema operacional específico, com isso os usuários podem escrever os programas em seus ambientes preferidos;
- **extensibilidade:** o design do programa foi escrito para dar suporte a outros microcontroladores;

A hipótese levantada por Conradi et al[2010] a respeito das limitações estruturais na programação em blocos, como no Scratch, foram solucionadas no ambiente Splish. Neste ambiente, o usuário tem a possibilidade de migrar para a programação na linguagem padrão do Arduino e com o tempo aprender num contexto mais baixo nível da linguagem. Contudo, mesmo com todas as ênfases levantadas, o Splish tensiona para acelerar as experiências apenas dos usuários no contexto de um *workshop* ou um curso específico, onde um tutor ou mediador apresenta as funcionalidades do ambiente e realiza toda a instalação dos requisitos do sistema para os iniciantes, o que limita a abrangência da ferramenta.

4.2.2 Os software de tempo-real

As ferramentas apresentadas nesta seção funcionam de forma conectada em tempo-real com o microcontrolador, o que permite um *feedback* mais rápido e real quando colocado em análise com softwares que compilam o código no microcontrolador. Em geral são ferramentas orientadas à fluxogramas e necessitam de um *middleware* para interagir em tempo-real com o microcontrolador.

Para Myers [1995], as GUI (*Graphical User Interfaces*) aplicadas no desenvolvimento de softwares em tempo-real através de módulos (ou *widgets*) simplificam o papel do desenvolvedor. Estas interfaces abstraem e envelopam os padrões de *design* e os controles de entrada e saída das aplicações. Os sistemas encapsulam os detalhes da implementação e expõem as funcionalidades da API através de uma programação relativamente simples onde usuários interconectam objetos para que funções específicas de cada módulo interaja entre eles [Myers, 1995].

Os Sistemas Orientados a Fluxogramas (SOF) podem ser vistos como linguagens onde o usuários podem desenvolver aplicações através de um fluxo de dados e execução modelados em ambientes visuais. As ferramentas visuais aplicadas no contexto da computação musical tem suas interfaces inspiradas em *patchchords*, comuns nos sintetizadores analógicos. Os objetos de controle e interação são conectados via *inlets* e *outlets*, o que habilita a troca de dados entre funções em tempo-real [Puckette, 1997].

O **Pure Data** é um ambiente de programação em tempo-real orientado à fluxograma (SOF)

aplicado para áudio, vídeo e gráficos desenvolvido por Puckette [1997]. A ferramenta tem como principal característica a grande quantidade de aplicações multimídias desenvolvidas por conta da sua natureza modular, flexível e dinâmica, como cita Jácome [2007]. A essência *open-source* também agrega uma comunidade muito ativa no desenvolvimento de novos objetos para aplicações interativas em diversas mídias. Entre os objetos, o Pduino²⁹, gerencia os microcontroladores Arduino através do *middleware* Firmata [Stainer, 2009].

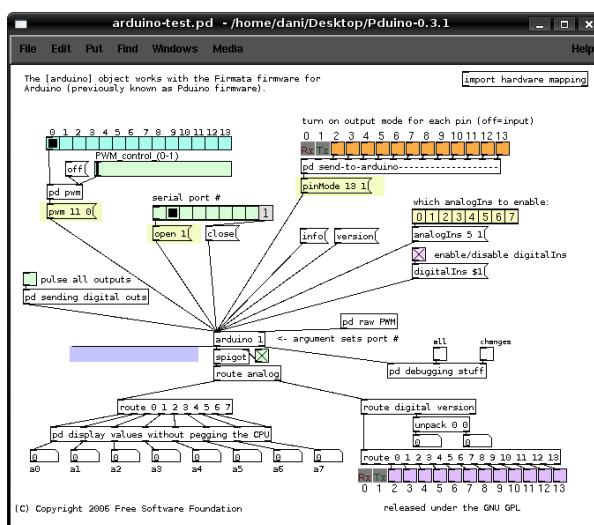


Figura 13: Patch Pduino para enviar e receber dados do Arduino via Firmata.

O Firmata é um protocolo genérico para comunicação com microcontroladores e tem o objetivo de transformar as atividades com sensores e atuadores mais acessíveis por programas e ambientes visuais. O design do *firmware* permite que qualquer ambiente de programação suporte-o, permitindo uma fácil implementação tanto para o microcontrolador quanto para o software. Existem módulos do Firmata para Pure Data, *OpenFrameworks*³⁰, *Max/MSP*³¹, *Processing*³², *Node.js*³³, *Ruby on Rails*³⁴ e *frameworks* em *Javascript* como *Jhonny-Five*³⁵ e *Breakout.js*³⁶, o que, de certa forma, amplifica a comunicação do Arduino para outros

29 <http://at.or.at/hans/pd/objects.html>

30 <http://www.openframeworks.cc/>

31 <http://cycling74.com/products/max/>

32 <http://processing.org>

33 <http://nodejs.org/>

34 <http://rubyonrails.org/>

35 <https://github.com/rwldrn/johnny-five>

36 <http://breakoutjs.com/>

conceitos e abordagens em desenvolvimento de software, o que confirma a escalabilidade do *middleware* proposto por Stainer [2009].

O projeto **Netlab Toolkit** (NLTK) desenvolvido por Allen [2007] junto com o *New Ecology of Things Lab* apresenta um sistema visual de interações físicas entre microcontroladores e mídias tipo vídeos, textos, gráficos e áudios, direcionados para a interação com objetos tangíveis. A interface do NLTK possui um kit de componentes *drag-and-drop* inspirados nos populares *widgets*. Este formato tem o objetivo de facilitar a criação de projetos que combinam sensores, atuadores e mídias digitais de forma rápida e sem necessidade de realizar nenhuma programação específica. Para Greenberg & Fitchett [2001], os *widgets* trazem para o programador um grande repertório de componentes que podem ser utilizados para esboçar interações. Isto resulta numa maior concentração nas funcionalidades dos módulos ao invés de uma preocupação com as operações mais baixo-nível da programação.

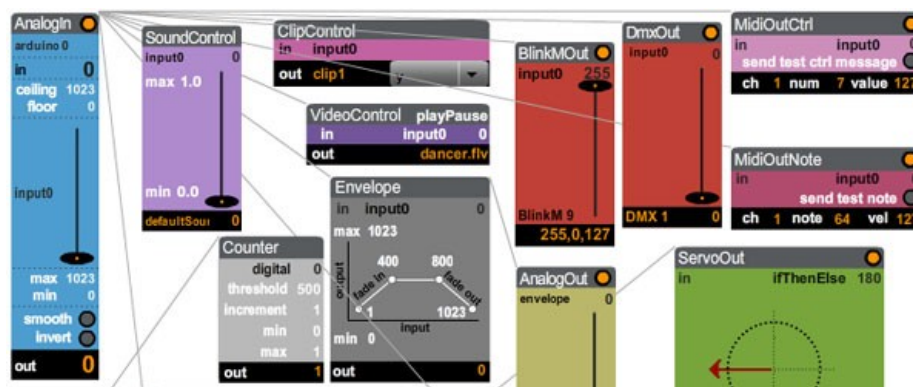


Figura 14: Widgets do Netlab Toolkit.

O sistema é composto por duas partes:

- (1) *Widgets*³⁷: são módulos manipuláveis que realizam diferentes funções no sistema. Normalmente recebem dados, realizam algumas operações e transferem valores para outros módulos. Com isso, é possível receber dados de uma entrada analógica e aplicar os valores para controlar a posição de uma imagem na tela, por exemplo.
- (2) Hub: é um servidor de aplicação que coordena a comunicação entre os *widgets* e os

³⁷ <http://www.netlabtoolkit.org/reference/widgets/>

dispositivos externos. O sistema comunica com diversos dispositivos como Arduino, Xbee e softwares como OSC (*Open Sound Control*). Os *widgets* se comunicam com o *Hub* para receber dados de *input* ou enviar dados de *output*. Também é possível comunicar através do protocolo *Socket*, o que facilita aplicações com o *Processing*, por exemplo.

Com um olhar nas adaptações para o suporte em *browsers*, a ferramenta Netlab Toolkit, apresentada na seção 4.2, criou uma alternativa ao ambiente padrão através de uma API em Javascript e HTML5 para controlar os módulos pela Internet. Para usar o NLTK junto com o Arduino na versão web, é necessário fornecer o IP do dispositivo. Caso o hardware esteja na mesma máquina, basta colocar o endereço *localhost*.



Figura 15: Widgets em HTML5 do Netlab Toolkit.

Com perspectiva de embarcar todo o serviço para a Internet, o **Codebender** surge como uma aplicação web e, ao mesmo tempo, como plataforma de colaboração para usuários de Arduino. A ferramenta tem o objetivo de eliminar as barreiras iniciais de configuração do microcontrolador. Nesta perspectiva, a plataforma é baseada na tríade:

- **código:** possui um editor *online*, um compilador em *cloud computing* e um sistema de correção de erro para facilitar a codificação;
- **upload:** possui sistema de envio de códigos para o dispositivo através da *USB* ou para um dispositivo conectado na Internet através de um *Shield Ethernet*. Esta função possibilita

diferentes formatos de colaboração e instalações interativas;

- **colaboração:** possui um sistema de compartilhamento e busca avançada de projetos. É possível clonar códigos existentes na plataforma e colaborar em outros projetos.

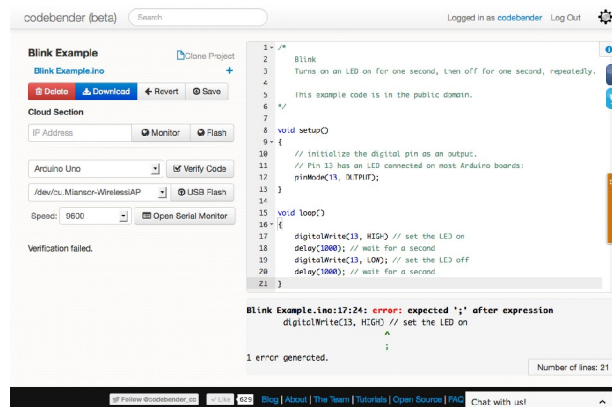


Figura 16: Editor online do Codebender

O usuário pode programar o microcontrolador pelo *browser* sem necessidade de procurar, instalar e gerenciar bibliotecas e dispositivos no computador. A plataforma possui um *plugin* que resolve todos os problemas de instalação e configuração em poucos minutos. Também é possível programar em um dispositivo remoto por *Ethernet Shield* através de um *Cloud Flash*. O sistema utiliza APIs e bibliotecas *open-source* como: HTML5³⁸; CSS3³⁹; ACE⁴⁰; Bootstrap⁴¹; Symfony⁴²; MySQL⁴³; MongoDB⁴⁴; PHP Fog⁴⁵; AWS⁴⁶; TFTP bootloader⁴⁷; Clang⁴⁸; e Github⁴⁹.

38 http://www.w3schools.com/html/html5_intro.asp

39 <http://www.w3schools.com/css3/default.asp>

40 <http://ace.ajax.org/>

41 <http://twitter.github.io/bootstrap/>

42 <http://symfony.com/>

43 <http://www.mysql.com/>

44 <http://www.mongodb.org/>

45 <http://www.appfog.com/>

46 <http://aws.amazon.com>

47 <https://github.com/arduino/TFTP-Bootloader>

48 <http://clang.llvm.org/>

49 <https://github.com/codebendercc/codebender.cc>

4.3 Ambientes de prototipagem de componentes eletrônicos

Nesta seção serão apresentados os ambientes de prototipagem de componentes eletrônicos para montagem e documentação dos protótipos físicos desenvolvidos. A etapa de prototipagem eletrônica aponta para dois caminhos: o primeiro para os kits de sensores e atuadores fechados e o segundo para a breadboard (placa de prototipagem) junto com sensores e atuadores acessíveis no mercado.

4.3.1 Os kits de sensores e atuadores fechados

As maiores dificuldades apontadas por Barragán [2004] e Mellis [2007] foram as barreiras inerentes à programação de microcontroladores e as dificuldades nos primeiros passos para um iniciante realizar experiências em computação física. Os kits de hardware, no qual compõem dispositivos fechados de entrada e saída, pode acelerar as experiências dos usuários por não necessitar conhecimentos para montar o circuito eletrônico e desenvolver algoritmos. Por outro lado, este modelo também pode criar um enviesamento em cima das intenções propostas pelo dispositivo físico.

Para Greenberg & Fitchett [2001], os kits de hardwares programáveis devem ser:

- simples o suficiente para que desenvolvedores se concentrem nos objetivos mais gerais do projeto ao invés de se concentrar em instâncias mais baixo nível e de implementação;
- fácil o suficiente para os iniciantes em programação e possibilidade de extensão.

O **Phidgets** é um conjunto de interface física onde dispositivos de entrada e saída são inspirados nos conceitos de *widgets*, comumente aplicados nos ambientes gráficos. A interface fornece componentes físicos programáveis para serem plugados entre eles. Além, existe um dispositivo que serve como hub físicos, um protocolo de comunicação e uma API para programação no PC.

Greenberg & Fitchett[2001] levantam algumas funcionalidades do Phidgets:

- **gerenciador de conexões** para controlar quando um componente é conectado e desconectado da placa. O ambiente monitora e comunica através da API quais os dispositivos estão plugados em tempo-real;
- **identificador** para fazer o link entre o software e o componente físico. O dispositivo é capaz de identificar quais dispositivos estão plugados;
- **modo de simulação** para testar os programas mesmo se o usuário não possuir um componente em específico. É possível incluir uma API para representar graficamente os componentes e simular o seu funcionamento.

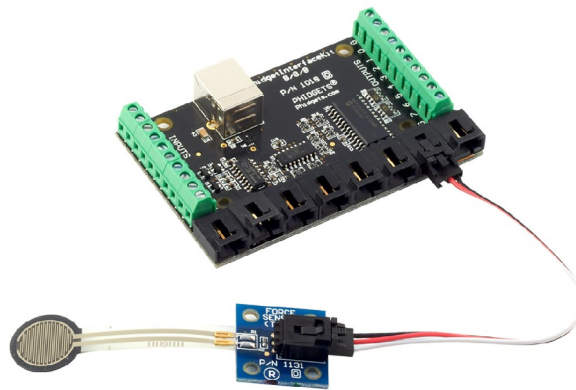


Figura 17: Phidget com sensor de pressão.

Inspirado nos princípios do Phidgets, o ambiente **d-tools** é um sistema integrado de hardware e software que habilitam desenvolvedores para a prototipagem rápida de interfaces físicas. O sistema prover suporte e foco na concepção ao invés de um foco apenas na implementação técnica do projeto. Através da plataforma, designers desenvolvem e controlam visualmente entradas físicas, como botões e *sliders*; sensores, como acelerômetros e saídas, como *leds* e telas LCD.

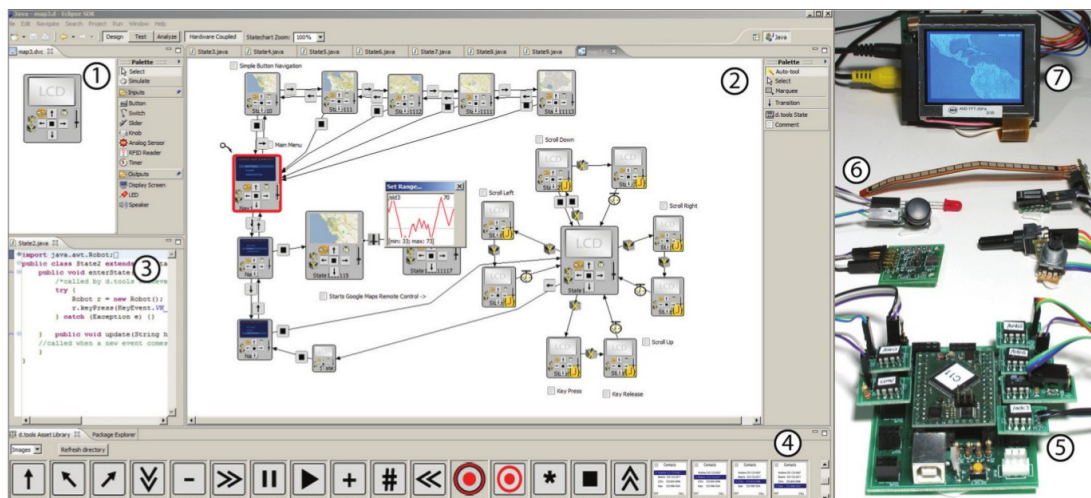


Figura 18: O ambiente d.tools: (1) gerenciador de dispositivo ; (2) editor statechart ; (3) editor textual ; (4) navegador. Na direita (5) a interface de hardware ; (6) os conectores compatíveis com a entrada do dispositivo e (7) uma tela LCD compatível com o sistema. (Hartmann, 2006)

O ambiente é inspirado no contexto da linguagem visual *Statecharts*, introduzido por Harel [1987] e é modelado em Java como um *plugin* do Eclipse IDE. O usuário é capaz de organizar componentes físicos de entrada e saída e mídias como áudio representados graficamente na tela através da metáfora *Plug-and-Draw*. O d.tools detecta dinamicamente a presença de componentes e habilita o software para configurar os dispositivos [Hartmann, 2006].

A plataforma prove um sistema *plug-and-play*, onde cada objeto é conectado e identificado num mesmo dispositivo. O sistema inclui um *display* que pode ser conectado em placas gráficas através da saída de vídeo de um PC. O software utiliza o protocolo I2C⁵⁰, um barramento serial desenvolvido pela *Philips*, feito para conectar periféricos de baixa velocidade em dispositivos embarcados. A comunicação do hardware com o PC é feita através do protocolo OSC (*Open Sound Control*).

50 <http://pt.wikipedia.org/wiki/I%C2%B2C>

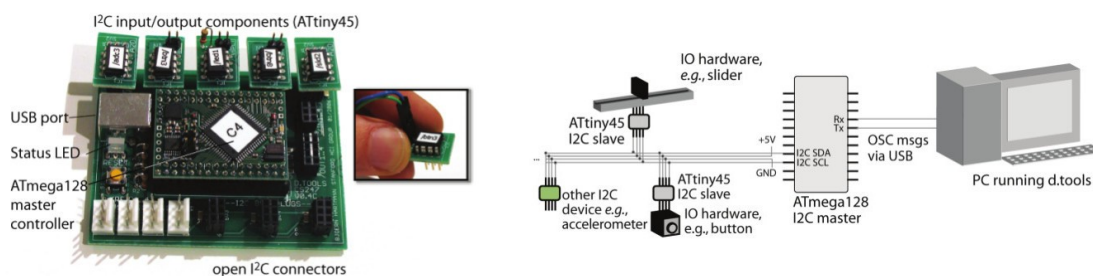


Figura 19: O hardware e o modelo para configurar novos dispositivos.

O ambiente também conta com a possibilidade de construir extensões para incluir novos dispositivos na plataforma através de uma suíte de configuração entre a interface e o hardware. O sistema possui uma biblioteca, escrita em Java, que serve como modelo para ser estendida para outros dispositivos. É possível realizar a comunicação com o Arduino e outros microcontroladores a partir do protocolo OSC.

Com uma proposta direcionada para problemas reais com a prototipagem no contexto dos *Maker*, o **Tinkerkit**⁵¹ surge como um conjunto de sensores e atuadores plugáveis e um *shield* que se conecta com o microcontrolador Arduino. O kit facilita e torna mais rápida a prototipação do circuito eletrônico para usuários que não tem interesse em soldar componentes ou usar a *breadboard* nos experimentos com o Arduino.



Figura 20: Principais componentes do Tinkerkit um exemplo de conexão de um led.

O projeto contém uma biblioteca para a IDE do Arduino onde diversos exemplos para os componentes do kit estão aptos a serem utilizados apenas com o *upload* do programa para a

51 <http://www.tinkerkit.com/>

placa e a conexão física no *shield*. Existem diversos módulos, como sensores, atuadores e hubs. Os sensores são acelerômetros, botões, giroscópios, *joystick*, sensor de luz, potenciômetros e outros. Os atuadores mais interessantes são os motores servo, os *relays* e um conjunto de *leds*. Entre os hubs, os mais interessantes são os o próprio *shield* para conectar os componentes, um LCD e um receptor de sinal digital DMX.

Com a orientação focada em interações rápida, o **Makey Makey** possibilita a construção de interfaces tangíveis que habilitam pessoas sem conhecimento em eletrônica transformarem qualquer objeto do dia a dia em interfaces interativas. O ambiente é compatível com qualquer software e não necessita programar e nem montar o circuito eletrônico. O sistema se aplica para um grande numero de usuários com desejos de emergirem ideias e dispositivos de uma forma mais livre e menos burocrática.

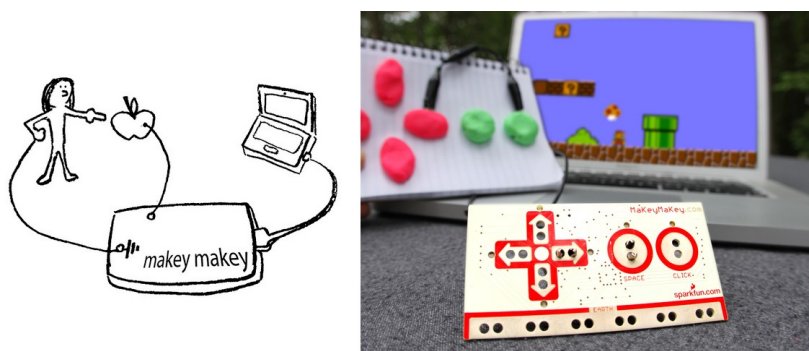


Figura 21: Exemplo de uso do Makey Makey.

No exemplo mais à esquerda da Figura 21, o usuário conecta o seu próprio corpo na porta GND (*Ground*) do dispositivo e, com uma maçã plugada, o usuário é capaz de enviar sinais do teclado para o computador. Quando o usuário toca na maçã, o sinal do caractere conectado à maçã é enviado para o computador. Como o dispositivo é *plug-and-play*, ele não necessita de nenhuma configuração no computador.

O dispositivo é um HID (*Human Interface Device*) que permite enviar eventos do teclado e do mouse para o computador sem necessidade de instalação de nenhum *driver* ou programa. O usuário conecta objetos e materiais naturais no dispositivo com o objetivo de criar interfaces capazes de controlar qualquer software em execução no computador. O sistema habilita todas

as interações possíveis por mouse e teclado.

O *Beginner's Mind Collective*⁵² apresenta as principais características do dispositivo:

- **início rápido para iniciantes:** é possível construir uma interface com entradas físicas para controlar uma saída digital em poucos minutos;
- **funciona em qualquer software:** é compatível com qualquer programa que recebe entradas de mouse e teclado do computador;
- **interfaces naturais:** permite a criação de interfaces com materiais naturais como plantas, frutas, água, óleo e o corpo humano;
- **não necessita de programação:** o sistema pode ser usado para criar interfaces sem escrever nenhum código;
- **não necessita de solda:** o usuário não precisa montar circuitos complexos nem realizar nenhuma prototipagem eletrônica na *breadboard*.

Para Conradi et al [2010], apesar desses kits ter uma grande quantidade de componentes prontos para o uso, usuários encontram limitações quando tentam ampliar os projetos com um sensor ou um atuador não incluso no pacote do kit. Essas barreiras podem ser incorporadas em projetos como Phidgets, d.tools e Tinkerkit, onde sistemas são modelados para serem trabalhados em kits de hardwares específicos para a plataforma. O dispositivo Makey Makey é interessante apenas para interações com poucas possibilidades de condicionais, o que não permite o desenvolvimento de qualquer interface.

4.3.2 Design de automação eletrônica com a breadboard

Nos aspectos visuais da prototipação do circuito eletrônico, a ferramenta **Fritzing** fornece suporte para *designers* e artistas na documentação de seus projetos em eletrônica. Para Knörig et al [2009], na medida em que a *breadboard* se torna um item importante para esboçar o trabalho, a mesma não facilita a produção de cópias em grande escala. Ao invés de forçar um esquemático técnico, como na maioria das ferramentas, o Fritzing permite que usuários

52 Jay Silver, Eric Rosembaum, MIT Media Lab.

documentem seus projetos através de uma metáfora visual que imita o contexto utilizados pelos *Makers* [Knôrig et al., 2009].

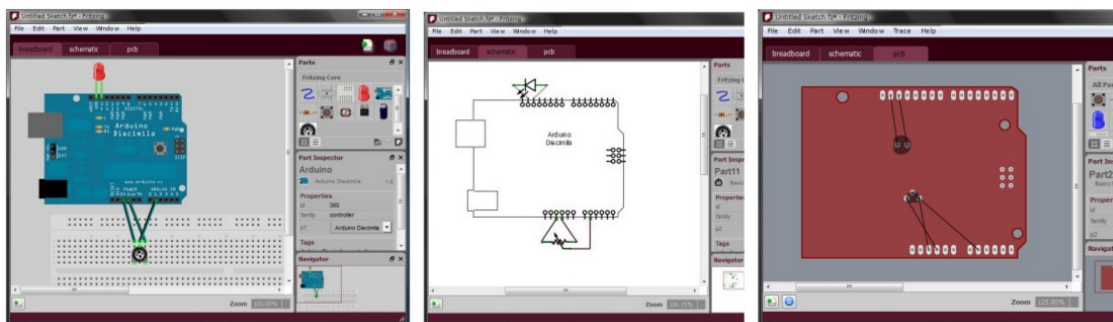


Figura 22: Múltipla visualização no Fritzing.

De acordo com a Figura 22, o ambiente oferece três alternativas de visualização:

- (1) visualização da *breadboard*, que imita o mundo real da prototipagem eletrônica;
- (2) visualização clássica de esquemático eletrônico para engenheiros e especialistas;
- (3) visualização PCB (*Printable Circuit Board*) para realizar cópias do projeto em grande escala.

O sistema possui uma base de dados com ilustrações, em perspectiva, de uma grande quantidade de componentes eletrônicos comuns em projetos de computação física. Os componentes podem ser deslocados para a área de esboço através do mouse e as conexões entre os objetos podem ser realizadas no mesmo ambiente. O modo PCB serve para *designers* com desejo de expandir o projeto para placas de circuito impresso. O sistema faz a conversão da *breadboard* para o circuito impresso, onde o usuário organiza apenas o posicionamento dos componentes e das rotas para ter um protótipo pronto para ser impresso em grande escala.

O Fritzing utiliza o conceito de *Open Interfaces* para garantir uma flexibilidade no ambiente de trabalho. A ferramenta funciona com tecnologias *open XML* como padrão de arquivos nativos do sistema. Também é possível importar arquivos *footprints* e exportar os circuitos para as ferramentas de EADs mais populares, como o Eagle⁵³. O ambiente também oferece

⁵³ <http://www.cadsoftusa.com/>

uma modo de edição dos componentes onde é possível alterar as propriedades e criar novas peças para o sistema. O editor possibilita a definição de imagens (vetores gráficos), conectores e metadados. Como o sistema utiliza arquivos gráficos em SVG (vetoriais), é possível interagir com um zoom infinito para identificar melhor as partes de um componente.

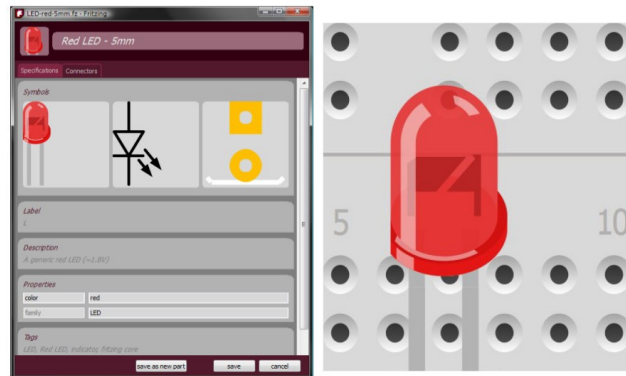


Figura 23: Modo de edição de um componente e zoom no Fritzing.

O Fritzing possui um site⁵⁴ para organizar uma base de dados com as produções individuais e coletivas dos usuários. A documentação *online* contém tutoriais e exemplos com informações sobre componentes eletrônicos e circuitos. Esta iniciativa demonstra a importância da organização dos conteúdos de forma que novos usuários possam encontrar, mais facilmente, exemplos para serem adaptados. O software é completamente *open-source* e foi desenvolvido sobre o *framework* Qt⁵⁵.

4.4 Análise dos ambientes relacionados

A partir do levantamento do estado da arte, torna-se necessária uma análise nos projetos relacionados com o objetivo de identificar as lacunas entre os percursos estabelecidos para efetivar uma prototipagem de software e de hardware mais completa e direcionada para o perfil de usuários não-técnicos. O quadro a seguir apresenta uma radiografia entre os princípios norteadores com os softwares de prototipagem em computação física.

⁵⁴ <http://fritzing.org/projects/>

⁵⁵ <http://qt-project.org/>

Aplicativos	Princípios			
	Integração entre Software e Hardware	Programação Visual	Feedback em tempo-real	Portabilidade
Scratch	Não	Sim	Não	Não
Modkit	Não	Sim	Não	Não
Splish	Não	Sim	Não	Não
Pure data	Não	Sim	Sim	Não
Netlab Toolkit	Não	Sim	Sim	Sim
Codebender	Não	Não	Sim	Sim

Tabela 2: Quadro comparativo entre aplicações e princípios norteadores.

A primeira lacuna observada está na integração entre software e hardware. Não existe nenhuma ferramenta que aborde, de forma semântica e integrada, as etapas de prototipagem em um único ambiente. Essa divisão entre a prototipagem do software e a do hardware, como visto em capítulos anteriores, desestimula o usuário iniciante por exigir diversas etapas prévias para a concepção e montagem do circuito eletrônico, bem como as habilidades para a pesquisa e desenvolvimento do software.

A programação visual é um item comum nos softwares que pretendem atender um perfil de usuário iniciante. Apesar de ampla maioria ser orientado a linguagem visual, os ambientes que trabalham com programação em tempo-real optam pelo paradigma de fluxograma como uma solução já testada e aprovada por uma série de usuários, artistas e Makers. Este sintoma indica que há uma preferência por ambientes que funcionam em tempo-real com o microcontrolador. Ora por facilitar as primeiras experiências, ora por ampliar as possibilidades de interação com outras fontes e mídias.

A portabilidade é um princípio norteador que permite uma abrangência do software em diferentes situações, principalmente para acelerar as barreiras iniciais de instalações e configurações de software, o que elimina velhos problemas de instalação de *drivers* e incompatibilidade com sistemas operacionais. Além disso, a portabilidade permite inserir

diferentes formas de interação por meio de celulares e *tablets*, o que pode sugerir novas formas de interatividades embarcadas em dispositivos uteis no cotidiano. Dos ambientes analisados, apenas os que funcionam sobre plataformas de *browsers* e na web, são os que atendem ao requisito proposto.

Esta análise amplia diferentes percepções sobre a investigação:

- as lacunas entre as etapas de prototipagem pode ser determinante para acelerar as experiências iniciais e reconfigurar os padrões de desenvolvimento de projetos em computação física;
- a programação visual por meio de fluxogramas tem relações diretas com o *feedback* em tempo-real, o que pode aproximar as práticas da computação física das interações bem sucedidas em softwares interativos para música, imagem e processamento gráfico;
- a portabilidade amplifica o papel do microcontrolador para um controle mais híbrido no sentido de permitir conexões de diferentes dispositivos independentes de configurações avançadas e distâncias geográficas.

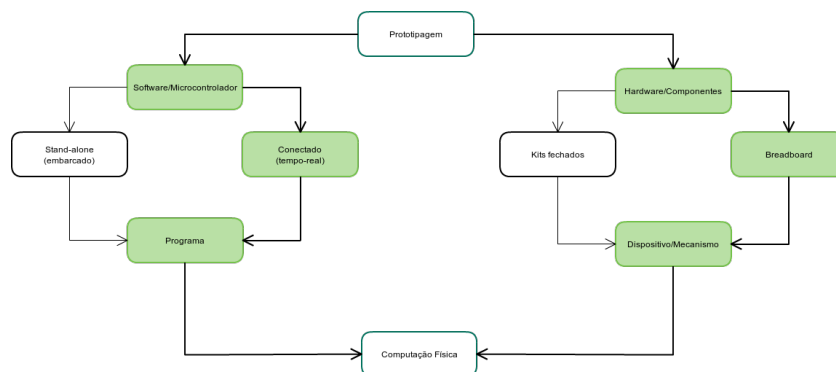


Figura 24: Percurso mais indicado para um ambiente de prototipagem em computação física.

Diante disso, há, claramente, espaço para o desenvolvimento de uma ferramenta que integram os princípios norteadores num só ambiente para acelerar e transformar as experiências de computação física em práticas de interação atrativas tanto para usuários iniciantes quanto para usuários em processo contínuo de experimentação nas artes por meio da computação.

5. tAMARINO

Ancorados nos princípios norteadores desta pesquisa, principalmente nas lacunas entre ferramentas de prototipagem de software e de hardware, o tAMARINO surge como uma abordagem para acelerar as práticas em computação física por meio de um software visual e intuitivo que atua em tempo-real com o microcontrolador.

Neste capítulo vamos apresentar os aspectos gerais do ambiente desenvolvido, bem como suas funcionalidades básicas para atender os princípios norteadores apresentados.

5.1 Aspectos gerais do ambiente

O tAMARINO é um protótipo de ambiente visual para prototipagem rápida de projetos em computação física através de microcontroladores. Trata-se de uma ferramenta que possibilita a experimentação rápida de sensores e atuadores por usuários sem *background* técnico, com o objetivo de acelerar as práticas e diminuir as barreiras do primeiro contato com técnicas de interação física. A proposta de utilização da interface modela um foco maior nas ideias do que nos desafios tecnológicos iniciais.

O ambiente apresenta uma interface visual funcional em navegadores avançados como o Google Chrome e Mozilla Firefox. O sistema roda em cima de APIs *open-source* capazes de dialogar com o microcontrolador em tempo-real e habilitar o acesso e manipulação das portas digitais e analógicas através de módulos gráficos. A ferramenta propõe uma abordagem onde a programação dos componentes de *input* e *output* está diretamente conectada com a montagem do protótipo eletrônico. Esta funcionalidade é representada por dois modos, um de esboço e outro de visualização. O ambiente é capaz de identificar os componentes utilizados e desenhar o circuito eletrônico necessário para o projeto funcionar.

A interface visual tem o objetivo reduzir o tempo da primeira experiência e acelerar a prototipagem na medida em que facilita o caminho no qual o usuário precisa percorrer para montar o circuito e experimentar a interação. Ao invés de se aprofundar em buscas de tutoriais específicos para realizar as etapas de programação e montagem dos sensores e atuadores, o usuário tem disponível um sistema que não precisa realizar codificação textual para

configurar um componente e, ao mesmo tempo, não precisa procurar por soluções e adaptações de outros circuitos.

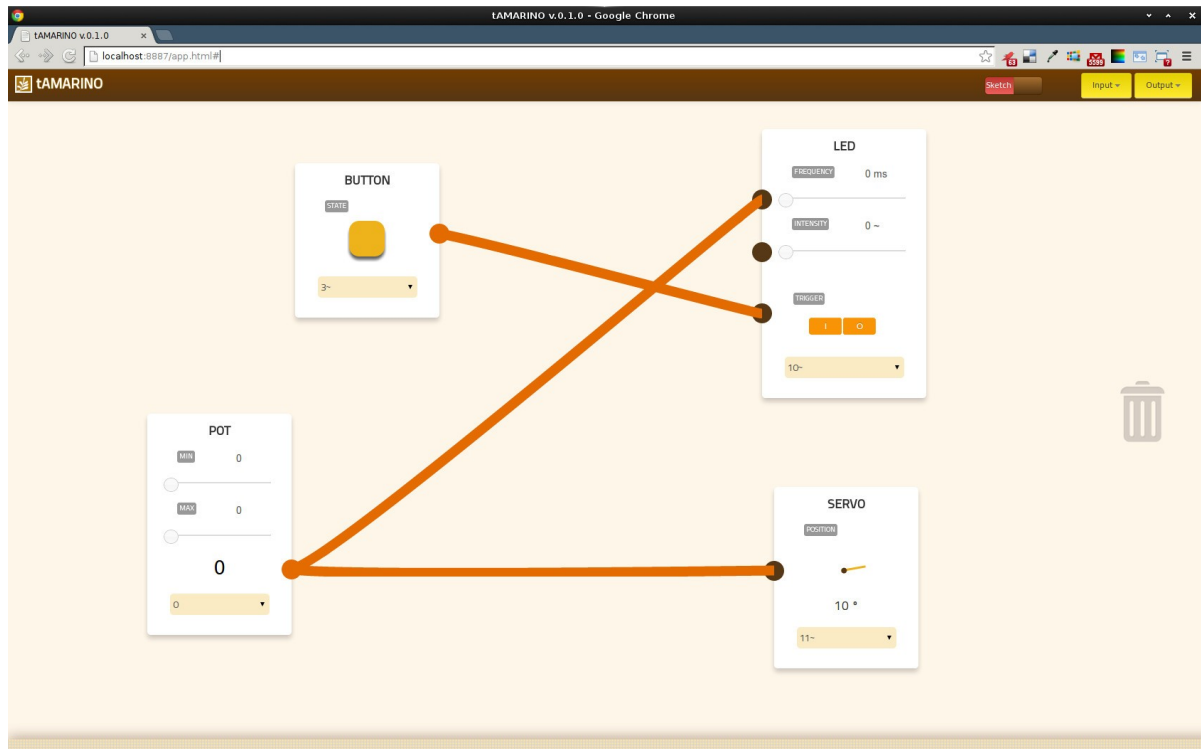


Figura 25: *tAMARINO* e seu ambiente de programação com *widgets* configuráveis e plugáveis.

Através de um sistema modular desenhado com *widgets* plugáveis, o usuário manipula diversos componentes de entrada e saída ao mesmo tempo e altera os parâmetros dentro do próprio módulo. A Figura 25 exemplifica um esboço capaz de realizar várias operações em tempo-real com sensores e atuadores. Na imagem, um potenciômetro controla, ao mesmo tempo, o método de frequência de um led e o ângulo de um servo-motor. Mais acima, um botão controla o *trigger* do mesmo led.

A integração com a prototipagem eletrônica utiliza uma chave (*Sketch-Make*) na barra de ferramentas localizada no topo da interface. A qualquer momento, o usuário pode alterar o modo e visualizar a simulação da montagem dos componentes eletrônicos. O programa monta automaticamente cada componente numa *breadboard* virtual e indica as conexões necessárias para que o projeto funcione tanto digital quanto fisicamente.

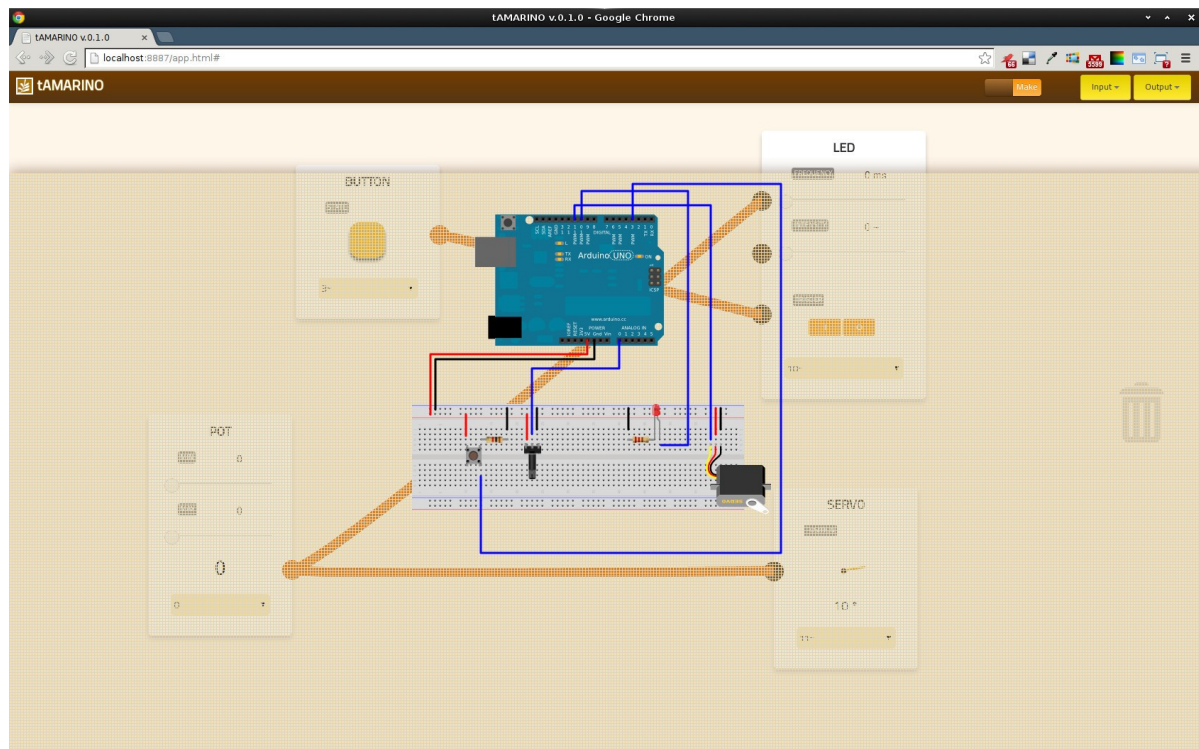


Figura 26: tAMARINO e sua área de prototipagem eletrônica integrada.

5.2 Funcionalidades básicas do tAMARINO

O tAMARINO é composto por três funcionalidades básicas: **programação visual**, representado por módulos de entrada e saída; **conexões dinâmicas**, capazes de criar interações entre módulos e métodos com **feedback em tempo-real** e; **prototipagem automática**, responsável pela simulação das conexões entre os componentes e a placa de prototipagem conectada no microcontrolador.

A tela inicial convida o usuário a selecionar o microcontrolador utilizado. Após esta seleção, o usuário entra na tela principal do sistema que contém:

- uma barra de ferramentas na parte superior com o menu para acessar os módulos e as funcionalidades básicas;
- uma área para esboço (*Sketch*) onde são configurados os módulos de entrada e saída, bem como as conexões entre eles e;

- uma área em “segundo plano” para gerar a prototipagem visual automática (*Make*) que conectam os componentes utilizados no *Sketch*.

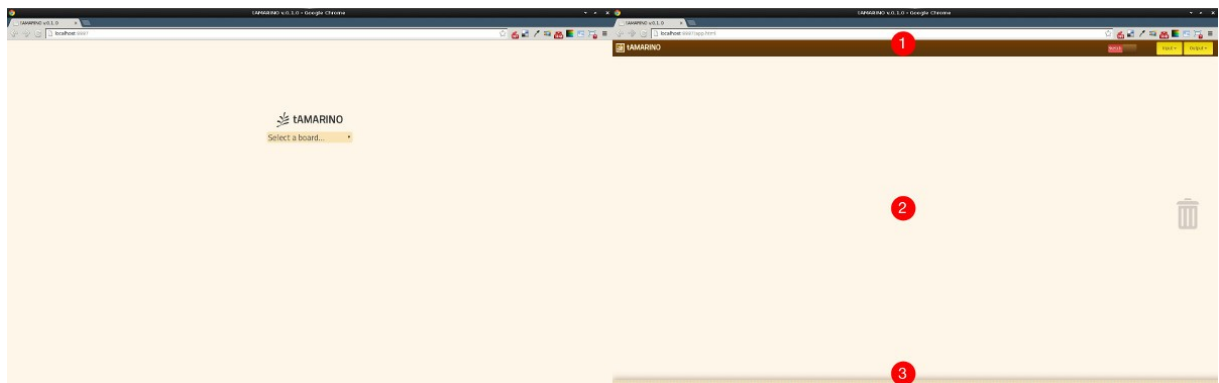


Figura 27: Na imagem da esquerda, a tela inicial do ambiente e na imagem mais a direita, (1) é a barra de ferramentas, (2) é a área de esboço (ou sketch) e (3) é a área de prototipagem eletrônica (ou make) do tAMARINO.

A partir do momento que o usuário entra no sistema, um *tour* é acionado pelo ambiente com o objetivo de apresentar as principais funcionalidades e ao mesmo tempo demonstrar um caminho “básico” para o usuário percorrer dentro da interface. Este comportamento tem o objetivo de guiar o usuário para experimentar a abordagem que integra o esboço do software e o protótipo do hardware em tempo-real.

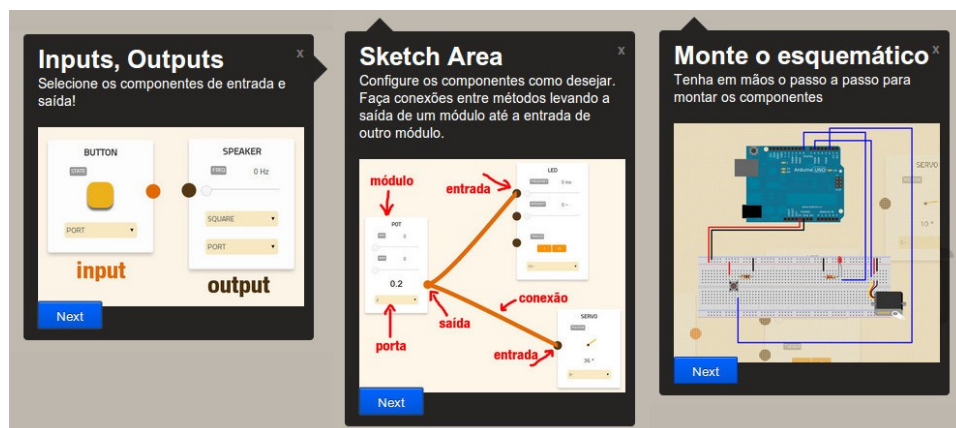


Figura 28: As três etapas-chave da abordagem proposta para acelerar as experiências através do ambiente tAMARINO.

5.2.1 Programação visual

O ambiente visual funciona por meio de módulos, que são representações abstratas dos sensores e atuadores físicos e seus conjuntos de métodos e propriedades. O modelo básico de um módulo no tAMARINO é composto por <<nome do componente>>, <<métodos>> e <<porta>>. Os métodos são específicos de cada sensor e atuador, como frequência, ângulo e escala. Já a porta configura a entrada e a saída dos dados do microcontrolador.

Os módulos são capazes de comunicarem entre si a partir das conexões entre *inlets* e *outlets*. Essa funcionalidade permite que o usuário represente digitalmente a conexão que deseja realizar fisicamente. A conexão entre módulos estabelecem trocas de dados em tempo-real entre os componentes físicos e expandem as possibilidades de interações entre sensores e atuadores.

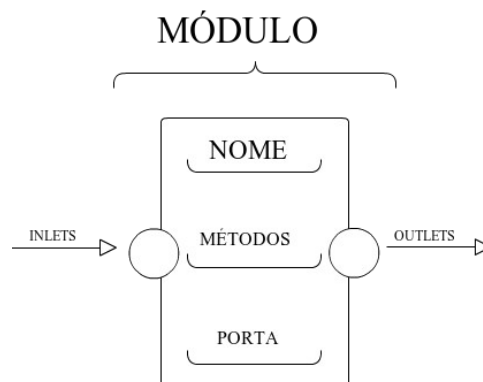


Figura 29: Representação geral de um módulo no tAMARINO

Os métodos disponíveis em cada módulo são representados por operações específicas de cada componente. O módulo desenvolvido para controlar um Led, por exemplo, possui três métodos possíveis: <<frequência>>, <<intensidade>> e <<trigger>>. A interação com os métodos são realizadas através do mouse. O *feedback* se dá em tempo-real tanto visual quanto fisicamente. Os métodos foram desenvolvidos a partir da análise dos códigos fontes em Tutoriais⁵⁶ oficiais do Arduino.

⁵⁶ <http://arduino.cc/en/Tutorial/HomePage>

Os módulos de entrada representam os sensores físicos como botões, potenciômetros e foto-resistores. Em cada módulo, o usuário pode editar facilmente parâmetros dos métodos e definir em qual porta o componente será conectado. Após as configurações, o sistema inicia a leitura da porta e aplica os parâmetros definidos pelos usuários no microcontrolador em tempo-real através de comunicação serial.

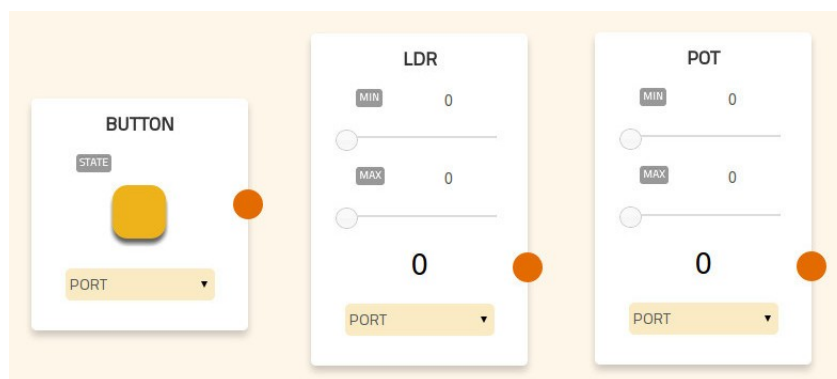


Figura 30: Os módulos de entrada do ambiente tAMARINO.

No módulo de botão (*BUTTON*), representado na Figura 30, o usuário visualiza o estado <<pressionado>> <<não-pressionado>> através de uma animação gráfica. No módulo de potenciômetro (*POT*), o usuário define o escopo entre valores mínimo e máximo do sensor. O módulo de foto-resistor (*LDR*) possui os mesmos princípios do potenciômetro. Em geral, a leitura das portas analógicas podem ser aplicadas para vários tipos de sensores, necessitando apenas uma organização no formato de saída dos dados. Entre os métodos e a porta, a interface mostram os valores atualizados em tempo-real com o componente físico conectado.

Já os módulos de saída representam atuadores físicos de emissão de luz (*Led*), movimentos mecânicos através de motores (*Servo*) e atuadores sonoros (*Speakers*). Do mesmo modo que os sensores, os atuadores também possuem métodos que podem ser configurados facilmente através do mouse e aplicados no microcontrolador em tempo-real.



Figura 31: Os módulos de saída do ambiente tAMARINO.

No módulo de Led, os métodos são: frequência (*Frequency*) aplicado em mile segundo (ms); intensidade (*Intensity*) de luz do sensor, no qual envia valores entre 0 e 255 para uma porta digital PWM~ (*Pulse-width Modulation*) e; disparo (*Trigger*), capaz de ligar e desligar o componente em tempo-real. O módulo de *Servo* possui um método de posição (*Position*) do motor, definido através de um ângulo entre 0° e 180°. O módulo de *speaker* possui um método de frequência onde o usuário configura entre 0 e 33 Hz a velocidade do som. Além disso, o módulo possui quatro opções de saídas, representados por *Square*, *Sin*, *Saw*, *Triangule* e *Impulse*.

5.2.2 Conexões Dinâmicas

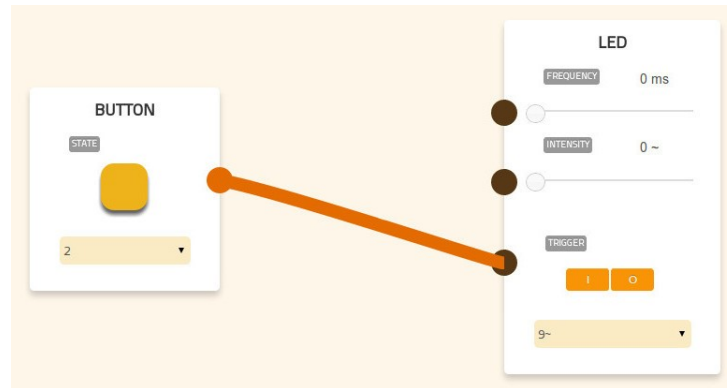


Figura 32: Exemplo de conexão entre um módulo de input com um output no ambiente tAMARINO.

A interface oferece a possibilidade de conectar dinamicamente módulos de entrada e saída através de *inlets* e *outlets*. As conexões são realizadas quando o usuário conecta o *outlet* de um componente de entrada num *inlet* de um componente de saída. A figura 32 representa a conexão entre o estado de um botão conectado no método de *trigger* de um led. A representação da conexão se dá através de uma linha laranja que simboliza a conexão entre os módulos. As conexões são estabelecidas a partir do momento em que um módulo é conectado ao outro e a interação com os dados é realizada digitalmente através do mouse e fisicamente através da *breadboard*. Os *inlets* não possuem limites de conexões e os *outlets* só podem estabelecer uma conexão por vez.

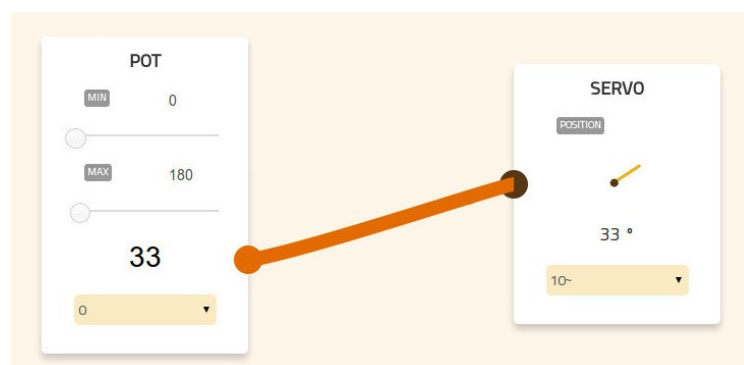


Figura 33: Conexões entre um POT e um SERVO.

O exemplo da Figura 33 representa a conexão entre um potenciômetro e um servo em tempo-real onde, os valores do sensor estão configurados para atuarem entre 0 e 180 graus. O exemplo representa a transferência de valores do módulo de entrada para o método que define o ângulo do módulo de saída, o que posiciona o atuador em 33°.

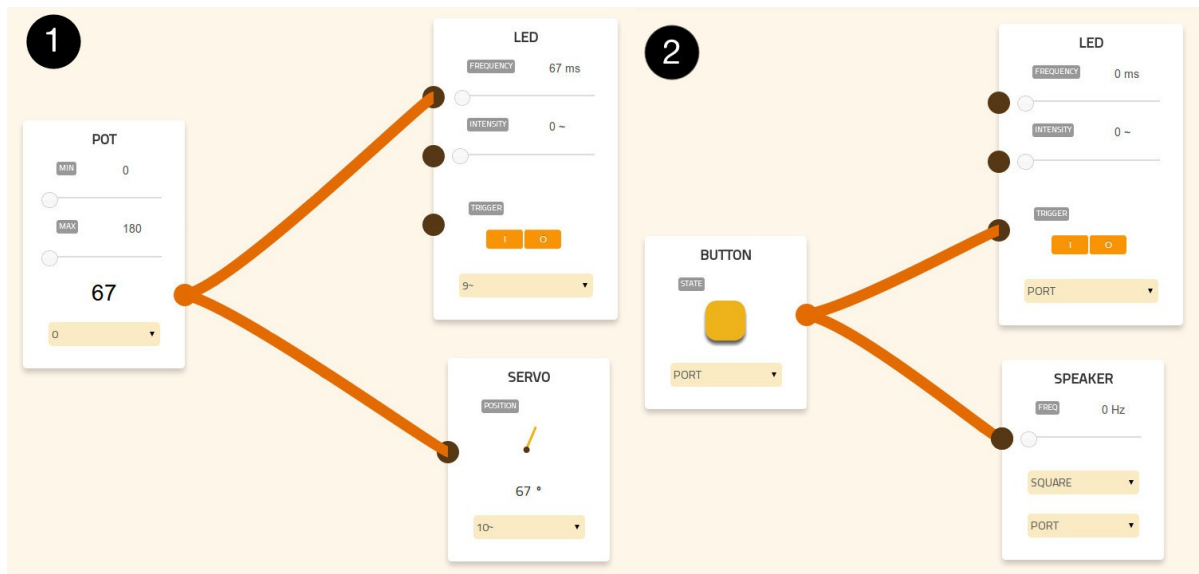


Figura 34: Conexões dinâmicas entre mais de um módulo ao mesmo tempo.

O sistema também possui uma funcionalidade para realizar conexões dinâmicas com mais de um módulo ao mesmo tempo. Na figura 34, (1) representa a conexão entre um módulo de entrada com dois módulos em tempo-real. Em (2), o usuário conecta um módulo digital de entrada para controlar diferentes módulos de saída. No exemplo, um botão acende um led e envia valores aleatórios para um *speaker*.

5.2.3 Prototipagem Automática

A interface do tAMARINO possui uma funcionalidade que identifica componentes de entrada e saída utilizados no modo de edição e simula o circuito eletrônico necessário para que a experiência física e digital se concretize num mesmo ambiente. O modo de visualização apresenta uma *breadboard*, um microcontrolador e componentes eletrônicos do banco de gráficos da plataforma Fritzing. O sistema identifica as portas configuradas nos módulos e realiza simulações das conexões necessárias entre a placa de prototipagem e o

microcontrolador. A conexão entre sensores, atuadores e microcontrolador são as mesmas representações do mundo real onde, usuários conectam componentes através de fios sem a necessidade de solda e ficam livres para prototipar.

O modo de edição e visualização possibilita que usuários naveguem entre os dois modos por meio de uma chave localizada na parte superior da interface. O sistema realiza uma animação de como o usuário deve montar o circuito para que o protótipo físico responda. Caso o usuário altere uma das configurações de um componente de entrada ou saída, a visualização do circuito eletrônico também é atualizada ao mesmo tempo, o que indica onde o usuário deve reconfigurar o circuito.

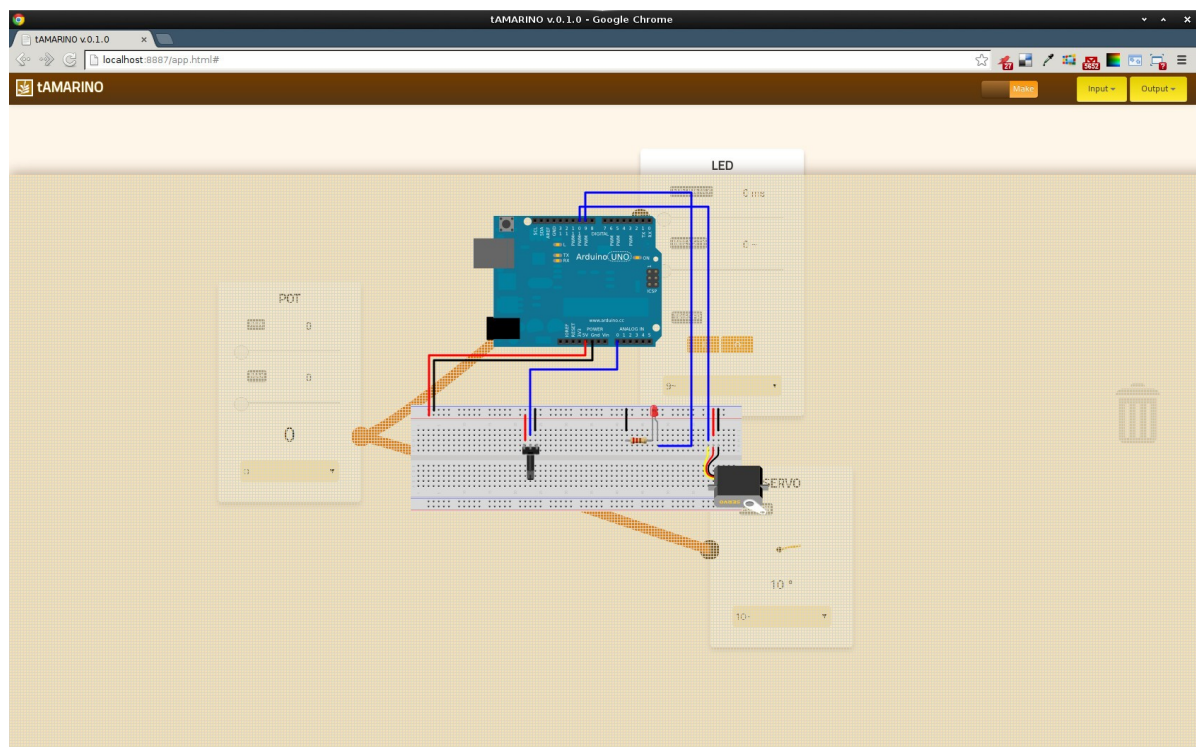


Figura 35: Área de prototipagem automática (MAKE) do ambiente tAMARINO.

A Figura 35 representa a conexão entre o módulo de potenciômetro, um led e um servo. O sistema reconhece quais portas foram configuradas e simula as conexões do circuito eletrônico necessárias para o usuário testar o esboço concebido. Os fios são apresentados através de uma animação que mostra a rota entre a porta do microcontrolador e os pinos da *breadboard*. A partir do momento em que cada componente é chamado no modo de edição, o

sistema desenha em segundo plano, o circuito eletrônico, semelhante a um projeto montado no Fritzing. Na mudança do modo de edição para o de visualização, o sistema reconhece os componentes utilizados e ativa uma animação dos fios se encaixando nas portas e na placa de prototipagem.



Figura 36: Linha do tempo da prototipagem automática.

De acordo com a Figura 36, em (1) o esboço apresenta apenas as conexões entre o microcontrolador e a *breadboard*. Neste modo, a prototipagem simboliza que o usuário não inseriu nenhum componente no modo de edição. Já em (2), o usuário insere o módulo de potenciômetro e o ambiente responde com as configurações necessárias para o sensor ser montado. Em (3), o usuário adiciona um led e na etapa (4) o usuário chama o módulo de servo e o sistema adiciona imagens dos sensores e atuadores numa posição para ser imitada pelo usuário.

O modo de visualização é uma das funcionalidades principais nesse processo de avaliação para uma abordagem integrada de ambientes rápidos com objetivos de acelerar experiências de iniciantes e não-especialistas técnicos. O aspecto que integram o desenvolvimento e a prototipagem de sensores e atuadores tem o objetivo de avaliar se tornam os processos de concepção e construção de protótipos mais intuitivos e rápidos.

6. Processos de desenvolvimento

Neste capítulo são elucidados os processos de design utilizados durante a concepção e desenvolvimento das versões preliminares do tAMARINO. A seção 6.1 contempla as descrições dos processos de design e os aspectos gerais. Na seção 6.2 são descritas as inspirações iniciais para concepção do projeto, onde há uma forte influência nas experiências pessoais do autor. Por último, na seção 6.3, são processadas as ideias levantadas nas fases anteriores, bem como organizadas e sintetizadas em protótipos de baixa-fidelidade e avaliadas por meio de entrevistas com usuários potenciais.

6.1 Descrição dos processos de design

Os processos de design utilizados na concepção e desenvolvimento do tAMARINO foram baseados no *design thinking* [Brown & Wyatt, 2010] aplicado à inovação e no HCD (*Human-Centered Design*) [IDEO, 2011], que possui intenções centradas nas experiências dos usuários. Os processos foram divididos em inspirações, concepção e implementação.

O processo de *design thinking* é esclarecido por Brown & Wyatt [2010] como um conjunto de módulos (*spaces*) ao invés de uma sequência ordenada de passos, o que remete à processos mais tradicionais de modelos em cascata. No *design thinking* existem três blocos livres em mente:

- (1) **inspiração:** está relacionado a um problema ou a uma oportunidade que motiva uma pesquisa por soluções;
- (2) **concepção:** está relacionado a um processo de geração, desenvolvimento e testes de ideias levantadas no processo de inspiração;
- (3) **implementação:** é o caminho que leva o projeto ao estágio de testes reais com as pessoas.

Para Brown & Wyatt [2010], este processo depende da nossa capacidade de ser intuitivo e reconhecer padrões para o desenvolvimento de ideias carregadas de signos emocionais e

funcionais. O processo também depende da habilidade de se expressar nos meios de comunicação em formatos diferentes. Esses requisitos são recompensados por soluções de alto-impacto que borbulham de baixo pra cima ao invés de soluções impostas de cima para baixo [Brown & Wyatt, 2010].

O artigo *Engineering Design Thinking, Teaching, and Learning* proposto por Dym et al [2005], descreve algumas abordagens que caracterizam o *design thinking*. Para os autores, essas características destacam as capacidades de:

- tolerar ambiguidades que aparecem ao ver o projeto como um processo de investigação ou de um ciclo de pensamentos convergentes e divergentes;
- manter a visão de um quadro geral do design, integrando o *thinking* e a prática reais no mesmo sistema;
- gerenciar as incertezas;
- pensar e comunicar em diversas linguagens do design.

O *Human-Centered Design* (HCD) está relacionado a um processo e também a um kit de ferramentas que tem objetivo de gerar soluções criativas para o mundo. Essas soluções incluem produtos, serviços, ambientes, organizações e interatividades. O processo é centrado no usuário porque ele inicia pelas pessoas que desejam uma nova solução [IDEO, 2011].

Um processo centrado no usuário enxerga o mundo através de uma lente onde são examinadas necessidades, desejos e comportamentos das pessoas envolvidas no processo de concepção de um projeto. A lente com os desejos são colocadas junto com questões de praticabilidade e visibilidade para gerar uma avaliação de acordo com as pessoas mas com um olhar para as viabilidades técnicas, organizacionais e financeiras [IDEO, 2011].

O processo é dividido em três fases: ouvir, criar e implementar. Essas etapas alternam entre o abstrato, na identificação de temas e oportunidades e no concreto, com soluções e protótipos reais. As fases são detalhadas abaixo:

- **ouvir:** fase de inspirações e motivações do projeto onde são coletadas histórias e

desejos das pessoas;

- **criar:** análise e tradução dos desejos dos usuários transformados em estruturas, oportunidades, soluções e protótipos. Essa fase circula entre o concreto e o abstrato na identificação dos temas e na criação de soluções;
- **implementar:** desenvolvimento das soluções através de uma prototipagem rápida de acordo com os desejos levantados.

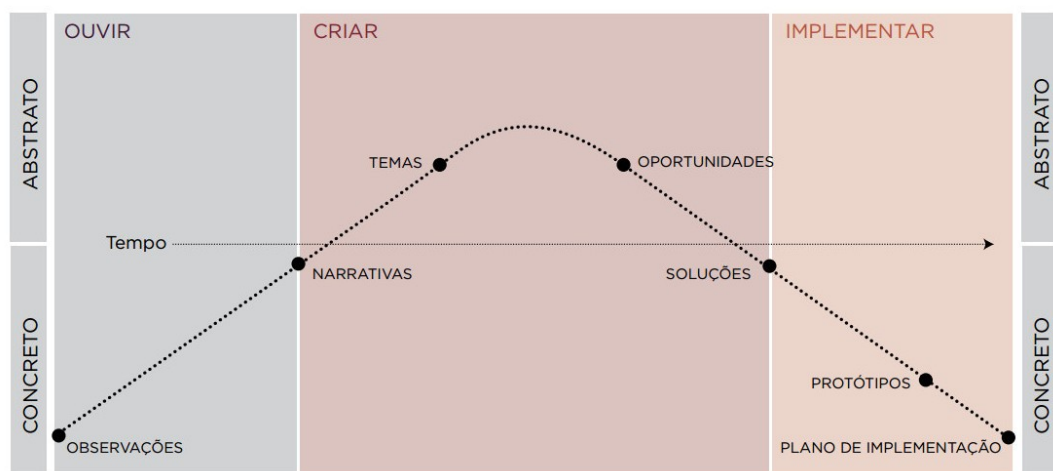


Figura 37: Os processos do Design Centrado no Usuário. [IDEO, 2011]

A partir disso, os processos de design aplicados para o ambiente tAMARINO foram modelados com as seguintes etapas:

- **inspiração:** a partir de experiência pessoal do autor desta dissertação e experiências interativas em softwares modulares;
- **concepção:** por meio de geração de protótipos de baixa-fidelidade para realizar testes e coletas de dados iniciais por meio de avaliação rápida com usuários;
- **implementação:** refinamento das ideias e desenvolvimento de versões de protótipos funcionais.

6.2 Inspiração

As inspirações para o projeto foram levantadas a partir da experiência pessoal do autor desta dissertação em diferentes perspectivas e papéis dentro do contexto da computação física. Além disso, alguns *frameworks* de interação modular na web também foram consideradas inspirações para esta investigação.

6.2.1 Experiência Pessoal

O autor dessa dissertação acompanha assiduamente o contexto das ferramentas *open-source* aplicadas nas interações com áudio, vídeo, gráficos e objetos físicos. Durante a sua vivência, o autor circulou entre o contexto de usuários iniciantes a especialistas. O autor participou como convidado em diversos *workshops* de arte e tecnologia, encontrando usuários com interesses na tríade Arduino, Processing e Pure Data.

Durante *workshops* em festivais como FILE⁵⁷, Píksel⁵⁸, LabSurLab⁵⁹, Interactivos⁶⁰ e em laboratórios do grupo Laboca⁶¹, o autor desta dissertação percebeu que as ferramentas Processing e Pure Data atraíam usuários pelos *feedbacks* visuais e sonoros rápidos e também pelas possibilidades de interações entre os softwares de forma relativamente fácil. Os usuários com interesse em construção de interfaces físicas e conexões entre sensores e atuadores no mundo real, eram atraídos por ferramentas como Arduino. Para o autor, a plataforma Arduino estimulava grupos para continuar a pesquisa mesmo depois do *workshop*.

As diferentes buscas por soluções técnicas para realizar a montagem dos componentes eletrônicos através da *breadboard* (ou protoboard) refletiram num maior **desgaste** nas experiências iniciais dos usuários. Para o autor desta dissertação, enquanto as ferramentas visuais tinham um *feedback* rápido e expressivo por conta do acesso a exemplos de uso disponíveis no próprio ambiente, as interatividades físicas travavam nos pré-requisitos técnicos em eletrônica para avançar nas etapas de prototipação.

57 Festival Internacional de Linguagem Eletrônica <http://file.org.br/?lang=pt>

58 <http://www.piksel.no/>

59 <https://labsurlab.org/>

60 <http://medialab-prado.es/interactivos>

61 <http://olaboca.wordpress.com/>

A partir da utilização do *middleware* Firmata nos *workshops*, as possibilidades de programação com Arduino expandiram para outras abordagens de ambientes visuais. As ferramentas mais utilizadas foram o Pure Data ou Processing, principalmente pelos resultados em tempo-real, expressividade da visualização e possibilidade de conexão com outras mídias como áudio, vídeo e processamento gráfico. A medida em que emergiram novos conceitos de ferramentas de programação visual para interações físicas, aumentaram as barreiras do primeiro contato, principalmente pelas dificuldades nas configurações iniciais da comunicação entre o Arduino e outros ambientes de programação. Além disso, eram exigidos pré-requisitos para o pensamento computacional, o que afastou o foco na concepção de ideias em detrimento dos desafios técnicos impostos.

A necessidade de conhecimentos técnicos em eletrônica impulsionaram usuários a realizarem buscas por tutoriais *online* para auxiliar a montagem do circuito. Para o autor desta dissertação, havia uma inquietação para atravessar pela primeira experiência de um *Hello World* com sensores e atuadores. Por outro lado, os serviços e aplicações com suporte da infra-estrutura dos *browsers* emergiram como soluções rápidas e intuitivas para prover aplicativos úteis no dia a dia, como é o caso do Google Drive, WordPress, Tumblr, Cacao, Trello etc.

Ao mesmo tempo que surgiram novos aplicativos na Internet, foram notadas iniciativas para utilizar as mesmas infra-estruturas para prover o desenvolvimento de interfaces para interagir com microcontroladores. Um exemplo disso são os projetos *node-serialport*, *Jhonny-Five* e *Breakout.js*, ambos sobre a linguagem Javascript. O artigo⁶² publicado na plataforma destinada a estudos sobre Internet das coisas, a Web of Things⁶³, toca num contexto interessante para esses aspectos:

Now how many of you have already made a Web page? Or setup a blog? Get my point? Everyone is a potential developer for the Web of Things. And you will be able to access and integrate real-time data from all kinds of sensors, simply by pasting some HTMLcode on your web page.

[Trifa, 2011]

⁶² <http://www.webofthings.org/2011/02/04/lift11-talk-transcript/>

⁶³ Dirigido por Dominique Guinard e Vlad Trifa durante o doutorado na ETH Zurich.

Trifa [2011] comenta a respeito da necessidade de expandir o conceito das linguagens da Internet para além das possibilidades dos hipertextos, das imagens e dos vídeos na tela. A sua intenção provoca para a conexão entre as características das plataformas Arduino e Processing como caminhos para novas funcionalidades em rede. Em outras palavras, Trifa [2011] encontra uma forma para impulsionar o acesso de não-especialista técnicos no contexto de desenvolvimento de projetos embarcados em objetos físicos no dia a dia.

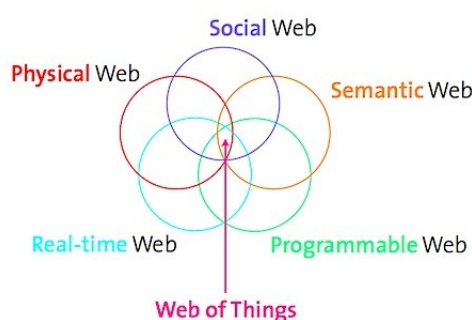


Figura 38: Vlad Trifa sobre o conceito de Web of Things no Lift'11.

6.2.2 Inspiração em mesas multi-toque e *web-frameworks*

O artigo *From Digital to Physical: Learning Physical Computing on Interactive Surfaces* [Conradi et al., 2010] levanta a hipótese de utilizar os benefícios de mesas interativas como guia para iniciantes em prototipagem eletrônica. Através das telas, usuários aproveitam objetos físicos aumentados com dicas a respeito dos componentes (*datasheets*, visualização do circuito eletrônico atual etc) e exploram as potências da computação física.

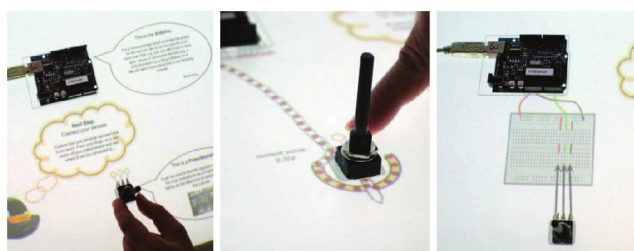


Figura 39: Protótipo de interação entre software e hardware por meio de mesa multi-toque. [Conradi et al., 2010]

Ao passo em que as orientações dos tutoriais auxiliam usuários a entender os princípios, as conexões virtuais servem para a construção e a configuração física real. As telas interativas oferecem vários benefícios, entre elas:

- **o suporte para multi-usuários:** o que convida os usuários para o trabalho coletivo na exploração e discussão dos modelos mentais dos princípios da eletrônica;
- **ambientes de trabalhos mais ricos:** onde o usuário pode expandir a visualização do componente físico através de informações digitais;
- **transição do digital pro físico:** usuários iniciantes avançam os estudos gradualmente no ambiente como um guia de transição do digital para o físico.

Os projetos com telas interativas refletem vantagens a respeito da experiência no aprendizado colaborativo. Estas práticas beneficiam grupos de trabalho a aprenderem mais sobre as práticas de interação. Outra vantagem está na mediação através de componentes físicos aumentados digitalmente para facilita o processo de aprendizagem e auxiliar na transição do digital para o físico.

O tAMARINO também é inspirado em alguns protótipos de aplicações *web* destinadas ao design interativo de mídias em tempo-real. Dentre as aplicações, destaca-se o Meemoo⁶⁴, um *framework* modular desenvolvido durante a pesquisa de mestrado em arte e novas mídias de Forrest Oliphant dentro da Escola de Artes, Design e Arquitetura na Universidade Aalto. O autor argumenta que pessoas podem criar e modificar ferramentas dentro de um *browser* apenas através de módulos gráficos. Estes foram criados para conectarem-se, muito focados na construção de artefatos interativos.

64 <http://meemoo.org/>

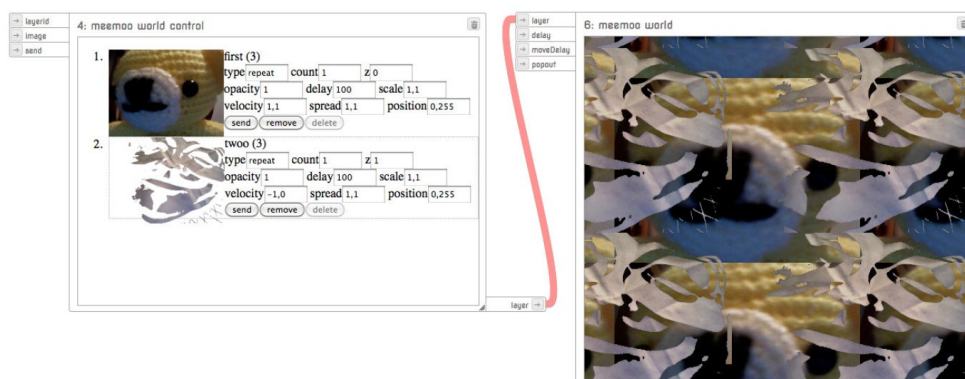


Figura 40: Framework Meemoo e uma interface para controlar animação.

Um outro projeto que inspira parte da interface dos módulos do tAMARINO é o projeto Web Audio Playground⁶⁵, hospedado no Github por Chris Wilson⁶⁶. Trata-se de um *framework* onde usuários constroem interfaces musicais através de conexões entre módulos como osciladores, entrada de áudio e efeitos sonoros.

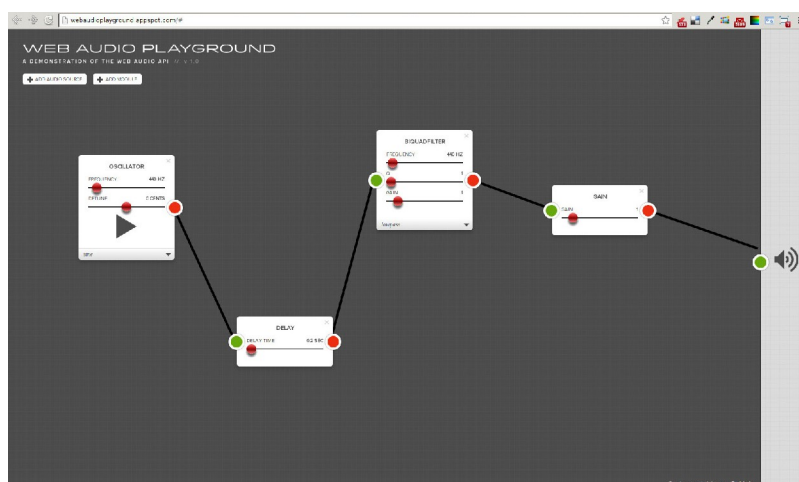


Figura 41: Web Audio Playground e seus widgets plugáveis e configuráveis.

65 <http://webaudioplayground.appspot.com/#>

66 <https://github.com/cwilso/WebAudio>

6.3 Concepção

A fase de concepção é o segundo passo no método aplicado na construção do tAMARINO. Para Brown & Waytt [2010], esta é a fase de síntese do que foi absorvido na etapa de inspiração e tem o objetivo de transformar as ideias em soluções e oportunidades. Os métodos de concepção utilizados nessa fase foram recombinaados a partir o processo *Creative Thinking Spiral* proposto por Resnick [2007]:

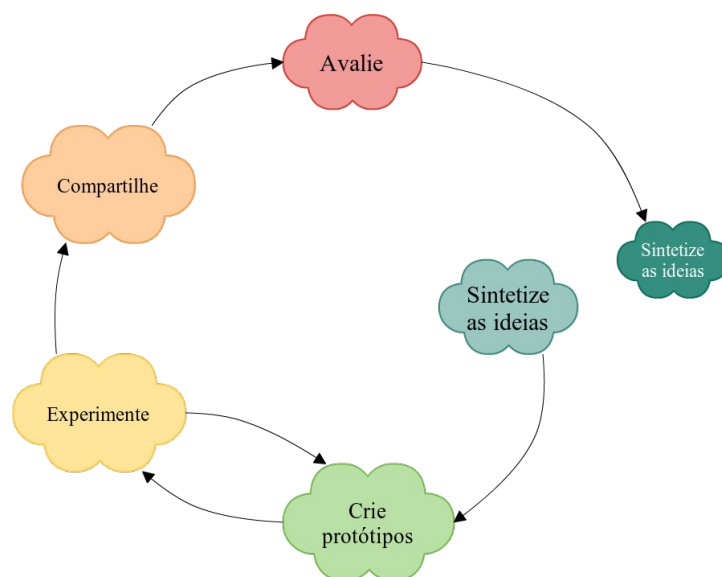


Figura 42: Processo iterativo em espiral para o processo de concepção.

Através desse processo em espiral, o ciclo de concepção aplicado na construção da abordagem é executado com no mínimo duas iterações para haver um melhor refinamento das ideias. As etapas-chaves são descritas abaixo:

- **sintetize as ideias:** fase para organizar os dados coletados e sintetizar em ideias-chaves;
- **crie protótipos:** fase para desenvolvimento de protótipos não-funcionais;
- **experimente:** fase para executar o protótipo de baixa-fidelidade com ciclos de refinamento;

- **compartilhe:** fase para testar o protótipo com usuários em potenciais;
- **avaliar:** fase para coleta de informações dos usuários e avaliação do protótipo para, por ventura, uma nova fase de síntese de ideias.

6.3.1 Protótipos preliminares e experimentos iniciais

A partir de uma análise nos trabalhos relacionados e no quadro comparativo entre as aplicações e os princípios norteadores, descritos na Tabela 2, a fase de prototipação ganha corpo e estabelece a transposição das ideias do campo abstrato para o concreto. Esta fase é a interseção entre a criação e a implementação de uma nova solução.

Os protótipos de baixa-fidelidade foram desenvolvidos com aplicações web para *mockup* e ambientes de prototipagem rápida de sites com o intuito de realizar as simulações necessárias para o usuário interpretar as interações propostas pelo ambiente. Os softwares utilizados nessa fase foram o Balsamiq⁶⁷ para o *wireframing* e o HotGlue.Me⁶⁸ como ferramenta para montagem das interações entre as imagens.



Figura 43: Soluções para prototipagem rápida de ideias.

A proposta de interação foi estabelecida para que o usuário desenvolva uma experiência com sensores e atuadores de acordo com os princípios de integração entre hardware/software, modularidade e expressividade. Com base nessa proposta, foram pensados em dois exercícios que são vistos como um *Hello World* no universo da computação física:

- (1) programar um LED (*output*) e aplicar o método *Blink* (frequência) com um valor fixo;

⁶⁷ <http://www.balsamiq.com/>

⁶⁸ <http://hotglue.me/>

(2) controlar a velocidade do *Blink* do LED com os valores de um Potenciômetro em tempo-real.

Depois de estabelecer os exercícios a serem praticados, inicia-se a fase de prototipação das telas de interação através do método Wizard of Oz⁶⁹. Foram rodados ciclos de prototipagem até chegar numa versão com possibilidade de ser experimentado e compartilhado por usuários em potencial.

Para facilitar o entendimento dos ciclos de prototipação e experiência, foram criadas tabelas com informações a respeito dos protótipos e das experiências obtidas durante o ciclo de experimento e refino das ideias foram geradas. As tabelas foram organizadas como no exemplo abaixo:

Imagem do <i>screenshot</i> do protótipo	
Protótipo	Descrição do protótipo.
Experimento	Descrição dos experimentos realizados pelo autor desta investigação, com objetivo de refinar o primeiro protótipo de baixa-fidelidade.

6.3.1.1 Ciclos de Prototipação

Foram realizados três ciclos de prototipação para sintetizar as ideias sugeridas e revelar as características de uma abordagem integrada hardware/software para ambientes visuais em computação física. Em cada ciclo, experimentos verificaram se os princípios estabelecidos estavam sendo abarcados pelo protótipo. A seguir, as tabelas descrevem os ciclos de prototipação e experimentos:

⁶⁹ Veja o artigo “Wizard of Oz studies: why and how” escrito por Nils Dahlbäck et al., 1993

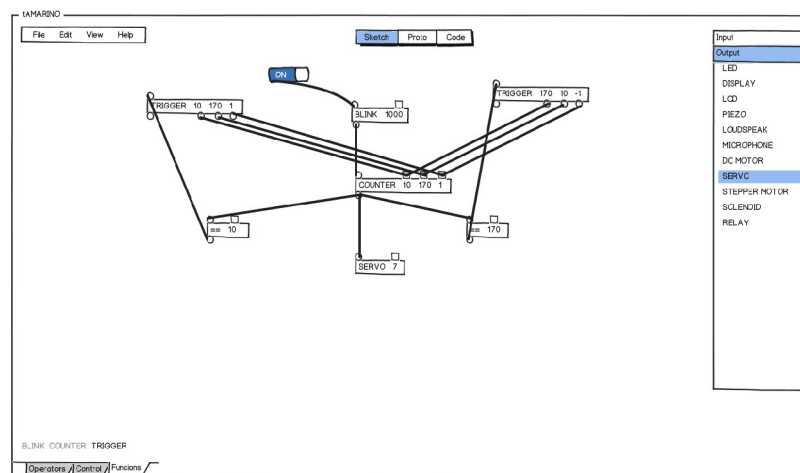


Figura 44: Primeiro protótipo desenvolvido do tAMARINO.

Protótipo 1

Foi inspirado em objetos e paradigma de programação *dataflow* baseado no ambiente Pure Data.

Experimento 1

A primeira impressão do protótipo provocou os frequentes problemas de desorganização dos ambientes orientados a fluxogramas onde existem objetos muito específicos.

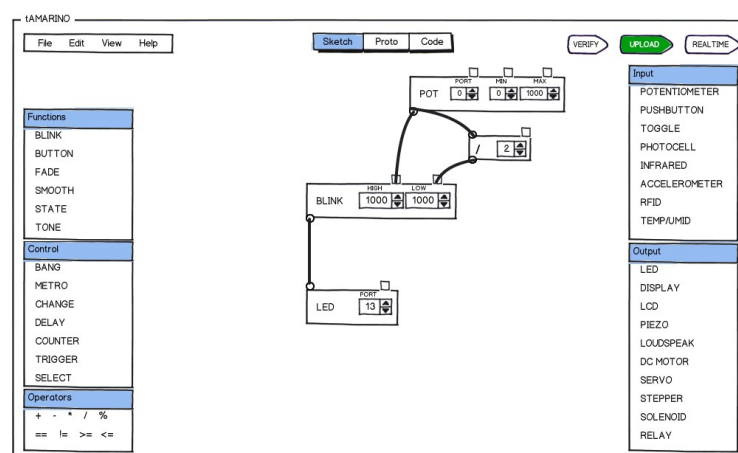


Figura 45: Segundo protótipo do tAMARINO.

Protótipo 2	A partir do segundo protótipo, os conceitos de modularidade foram introduzidos. Os componentes eram compostos por módulos e suas respectivas propriedades como partes desses módulos.
Experimento 2	Mesmo com a evolução, ainda persistiam as barreiras encontradas no primeiro protótipo: desorganização e dificuldade para interpretação visual do <i>sketch</i> e do menu.

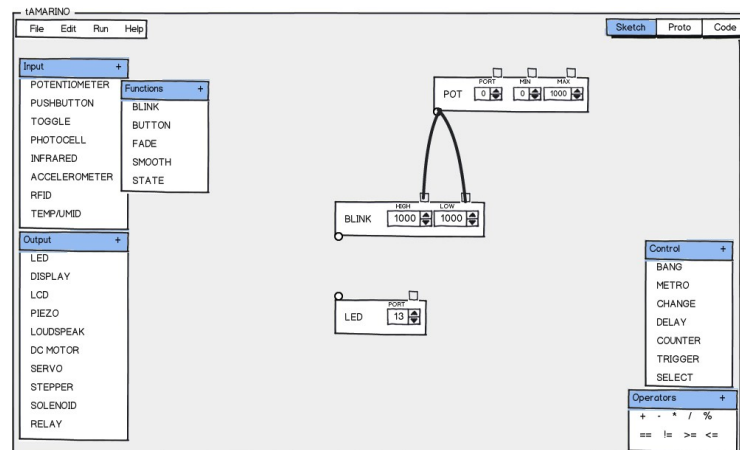


Figura 46: Terceiro protótipo do tAMARINO

Protótipo 3	O terceiro protótipo foi desenvolvido com o objetivo de melhorar a interface e corrigir o posicionamento do menu. A abordagem de sub-menu que define os métodos de cada componente foi introduzido no Protótipo 3.
Experimento 3	As experiências mostraram que as abordagens iniciais estavam inseridas e prontas para a fase de compartilhamento com grupos de usuários em potencial.

6.3.2 Avaliação dos protótipos preliminares

A fase de compartilhamento e avaliação se estabelece como a última parte do ciclo iterativo.

Foram escolhidos três diferentes perfis de usuários:

- leigos sem nenhum contato com o contexto;
- usuários não-especialistas com conhecimentos básicos;
- especialistas, programadores e designers de interação.

Os usuários foram convidados para avaliar a experiência através de uso do protótipo online descrito na seção 6.3.1. O procedimento utilizado para a avaliação foram os experimentos baseados em tarefas (*task-based*), de acordo com os argumentos de Burstein & Linger [2002].

O artigo *A task-based framework for supporting knowledge work practices* descreve que o foco do experimento está na geração e coleta de conhecimentos através de experiências associadas a execução de tarefas. Os autores propõe um *framework* para gerenciamento de conhecimentos e descreve os elementos principais desta abordagem:

- o foco na tarefa;
- um conjunto de práticas;
- uma memória organizacional;
- os resultados da tarefa;
- o apoio no processo de conhecimento.

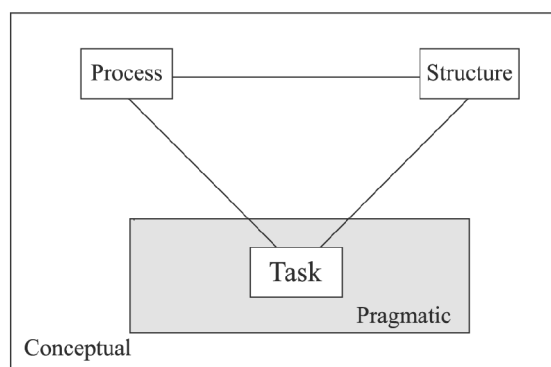


Figura 47: O framework para task-based proposto por [Burstein & Linger, 2002]

Para uma coleta de dados abrangente, foi aplicado o método *Think-aloud* que possui a capacidade de capturar as primeiras percepções e os pensamentos semânticos dos usuários no exato instante das ações executadas nas tarefas [Somerén et al., 1994]. Para Someren et al [1994], o método consiste em solicitar às pessoas “pensarem alto” durante a resolução de um problema e depois realizar uma análise dos discursos verbais capturados.

Após a execução das tarefas e da captura das impressões iniciais, foram coletados dados quantitativos e qualitativos. A coleta foi registrada através de questionário online com perguntas baseadas na escala Likert⁷⁰ e em entrevistas semi-estruturadas para captura qualitativa baseada no conhecimento dos participantes.

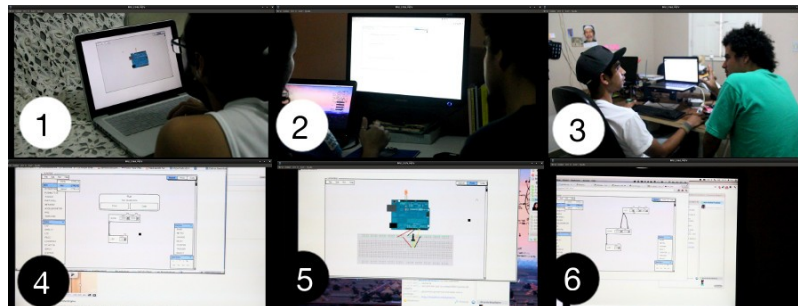


Figura 48: Avaliação da fase de concepção. Do (1) ao (3) a coleta foi realizada presencialmente. Do (4) ao (6) a coleta foi realizada por meio da Internet com compartilhamento da tela.

⁷⁰ Publicado por Rensis Likert no artigo *A technique for the measurement of attitudes* em 1932.

6.3.2.1 Opiniões sobre a abordagem

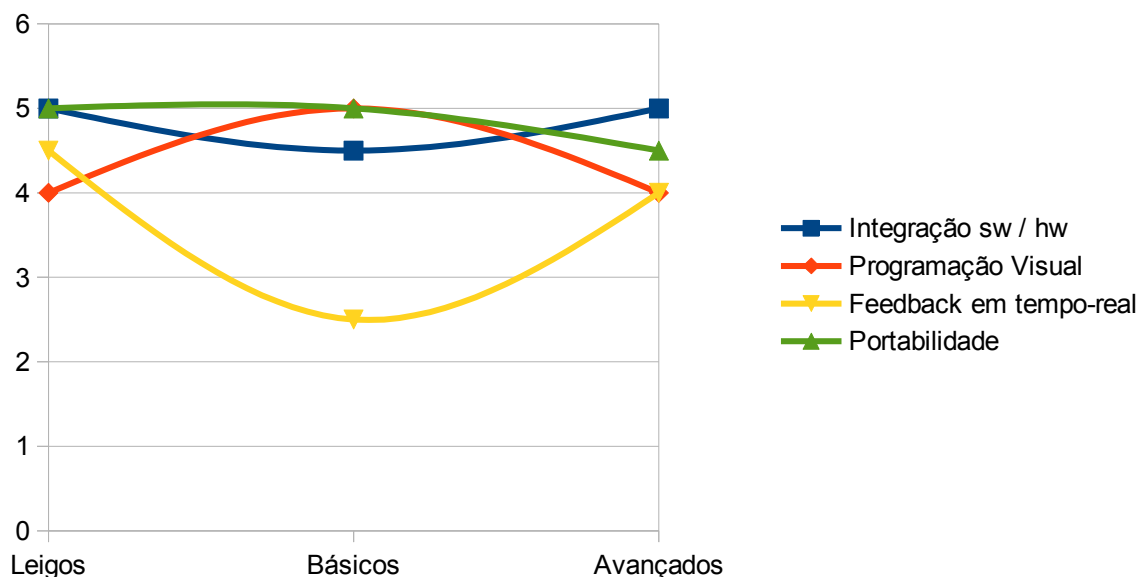


Tabela 3: Opiniões sobre a abordagem do projeto a partir da experiência com o protótipo de baixa-fidelidade desenvolvido na etapa de concepção. Dados obtidos através de perguntas e respostas na escala de Likert.

Num contexto geral da análise do discurso das experiências com os protótipos de baixa-fidelidade, os três grupos (leigos, não-especialistas e designers) avaliaram como positiva as abordagens relacionadas com os princípios norteadores. No geral, todos os princípios são de interesse dos usuários leigos e avançados. No caso dos usuários com experiências básicas, a questão da expressividade com feedback em tempo-real não soou como importante.

Sobre a integração entre hardware e software, um dos participantes sem experiência em Arduino comenta que, para ele, a representação do desenho físico “*facilita entender como funcionam as coisas*” e ajuda porque parece “*um professor online*”. Outro usuário leigo comenta que o ambiente é “*relativamente fácil*” de trabalhar e que “*o fato de aparecer o Arduino, facilita e muito e dá vontade de fazer*”(sic).

Um dos participantes com conhecimentos básicos em computação física comenta que a visualização da prototipagem eletrônica é “*essencial*”, ajuda no “*desenvolvimento de projetos*

de maneira autônoma, autodidata” e *“ajuda na documentação”*. Outro usuário com o mesmo perfil, avalia como *“essencial para compreender o que está sendo feito e para a aprendizagem”*. Um dos participantes sugere uma *“lista de componentes”* para o diagrama eletrônico.

Em relação à programação visual, houve um consenso sobre a facilidade para quem não tem experiência em programação. Um dos usuários comenta que a ferramenta *“lembra muito o PD”*, o mesmo usuário avalia o uso dos módulos gráficos como uma abordagem *“poderosíssima”*. Ainda sobre a programação visual, um dos leigos afirma que a maior dificuldade está *“na hora de programar, de fazer as funções”*. Para o outro participante, a visualização do código fonte *“não fez diferença”*. O mesmo usuário argumenta que *“facilita o fato de ter as funções aqui de lado pra gente ir clicando”*(sic), o que confirma a indiferença pelo código textual. Os usuários avançados sugerem a opção de *“implementar novas funções e poder usar elas como objetos”* e *“ter acesso a implementações de shields ou funções mais avançadas”*. Em contrapartida, outro usuário lembra de que *“você tem que entender um pouco a linguagem”* para evoluir as experiências.

A respeito da portabilidade, os usuários avaliam que *“se torna um atrativo”* quando se utiliza a web. Outro usuário aponta que o ambiente hospedado na Internet pode prover uma opção onde *“você cria objetos e compartilha com outros usuários”*. Não houve comentários a respeito da configuração do Arduino e das possibilidades de configurações rápidas do microcontrolador.

Numa avaliação geral, são tecidos importantes comentários: *“achei muito interessante, acho que pode auxiliar pessoas que tem interesse em utilizar programação e eletrônica em projetos mas não tem conhecimento técnico.”* Outro usuário confirma que *“auxilia no desenvolvimento ágil de aplicações, sendo muito útil para a criação de protótipos e testes de novas ideias.”*

7. Implementação

Este capítulo tem como objetivo apresentar a arquitetura adotada no desenvolvimento do tAMARINO, bem como as tecnologias utilizadas para a modelagem do protótipo utilizado para testes reais. As avaliações coletadas durante a fase da concepção foram importantes para delinear as estratégias de implementação do protótipo funcional. Na seção 7.1 será apresentada a arquitetura geral adotada pelo sistema. Já na seção 7.2, serão apresentadas as tecnologias utilizadas numa visão geral das camadas do sistema. Por fim, a seção 7.2.1 detalha as aplicações utilizadas para o desenvolvimento do protótipo utilizado para testes reais.

7.1 Arquitetura do Sistema

A abordagem utilizada nesta investigação foi modelada em cima de uma arquitetura onde os componentes de hardware e de software conversam em diferentes camadas através de servidores e *scripts* locais ou em rede. O modelo facilita o papel dos designers e desenvolvedores na implementação de novas funcionalidades e seguem as orientações para o desenvolvimento de ambientes interativos proposto por Hartmann et al [2010].

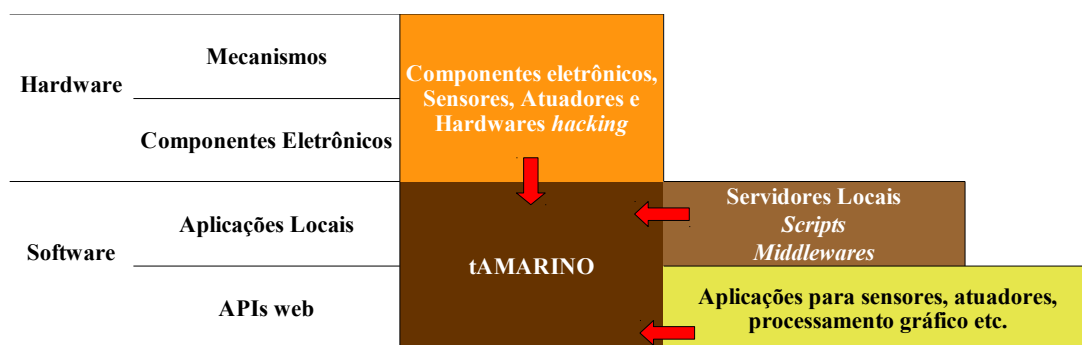


Figura 49: Arquitetura para a abordagem da interface tAMARINO.

A arquitetura é definida pelos blocos de hardware e software. Os hardwares compõem mecanismos e componentes eletrônicos que são representados por sensores, atuadores e dispositivos. Os softwares são compostos por aplicações locais ou em redes e são representadas por servidores, *scripts* e *middlewares*. A interface com o usuário conversa

através de APIs que conectam os componentes e realizam processamentos baseados em eventos.

A respeito das camadas que integram as funcionalidades do ambiente, a arquitetura adota o modelo de desenvolvimento baseado em MVC (*Model-View-Controller*). Essa estratégia de representar, mostrar e controlar a informação possui dois objetivos centrais segundo Krasner et al [1988]:

- (1) criar um ambiente especial de componentes para fornecer suporte aos processos de desenvolvimento softwares;
- (2) prover um sistema de componentes que seja possível criar aplicações gráficas facilmente.

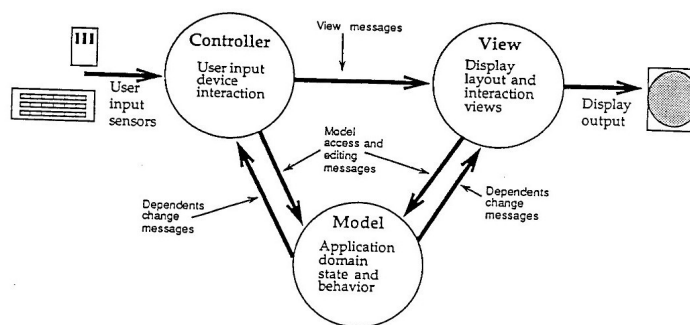


Figura 50: Máquina de estados do modelo MVC segundo [Krasner et al., 1988]

A arquitetura é modelada a partir de duas camadas principais: *back-end* e *front-end*. As aplicações em *back-end* são responsáveis pela: concepção dos módulos e métodos de entrada e saída; comunicação com o microcontrolador; conexão entre os módulos e; funcionalidades de prototipagem eletrônica automatizada. Já as aplicações em *front-end* representam a visualização da interface de interação com o usuário, onde APIs são responsáveis pelo gerenciamento do ambiente gráfico. O sistema nesta abordagem é orientado à eventos com algoritmos assíncronos, o que deixa a plataforma dinâmica e robusta.

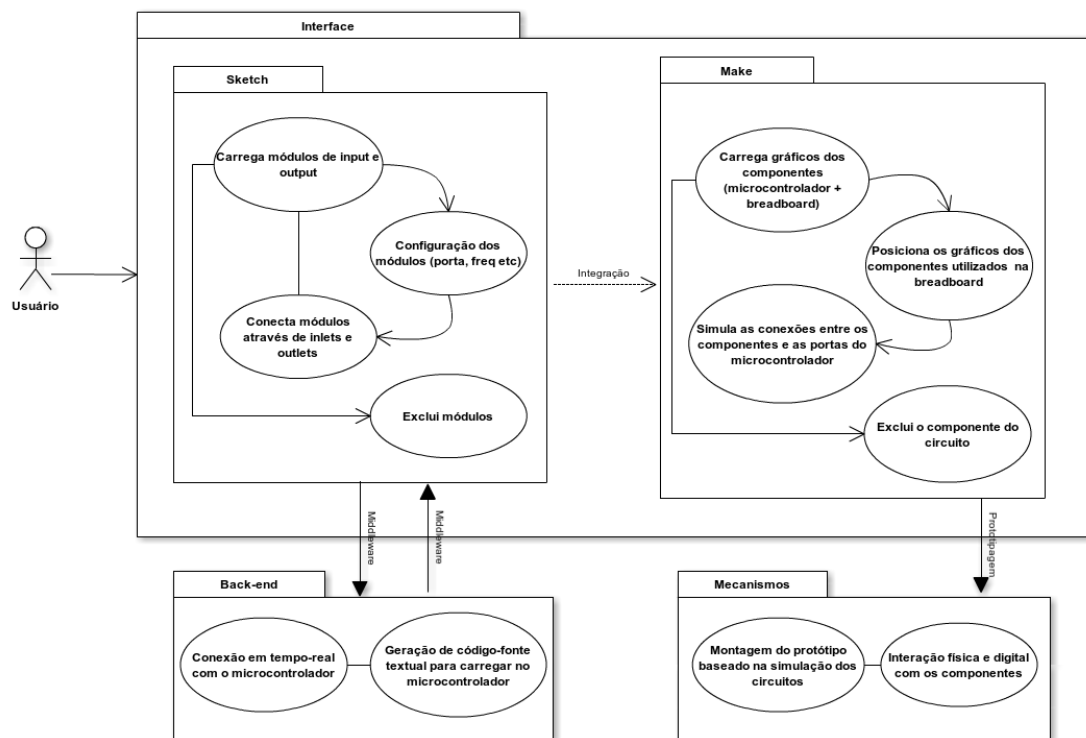


Figura 51: Arquitetura de aplicações em ambientes integrados para Computação Física.

O ambiente trabalha em dois modos: um modo de edição (*Sketch*) dos componentes e um modo de visualização (*Make*) do circuito. No primeiro modo o usuário pode: carregar módulos de sensores e atuadores, configurar portas e métodos e conectar módulos através de *inlets* e *outlets*. O segundo modo é responsável pela visualização do circuito eletrônico com rotas e arranjos definidos pelas configurações da etapa de edição. O ambiente simula as conexões necessárias entre o microcontrolador e os componentes físicos através de uma placa de ensaio (*breadboard*) e monta um circuito específico para o projeto desenhado. Esta etapa funciona em segundo-plano, ou seja, as alterações no modo de edição refletem no modo de visualização.

Os serviços em *back-end* realizam interações necessárias para enviar e receber dados dos microcontroladores através de protocolos e *middlewares*, tornando a experiência de interação com sensores e atuadores com *feedback* mais rápido e com controle mais fácil entre as camadas física e digital. Uma outra opção sugere a configuração para gerar o código-fonte do

projeto de acordo com as combinações montadas visualmente. O sistema formata a linguagem de acordo com o microcontrolador utilizado e libera para *upload* o projeto. A arquitetura adotada prioriza uma experiência em computação física com a mesma fluidez de ferramentas em tempo-real e orientadas à fluxogramas.

Neste modelo, o usuário monta o circuito eletrônico fisicamente e interage com o protótipo dentro da própria aplicação visual via mouse e teclado. Desta forma, o usuário pode experimentar diferentes conexões entre componentes de entrada e saída sem a necessidade de realizar nenhuma programação textual e configurações avançadas. Tanto os *inputs* são percebidos e representados nos módulos de sensores quanto os *outputs* do usuário acionam atuadores em tempo-real. Esta dinâmica acelera os processos iniciais de experimentação no contexto da computação física.

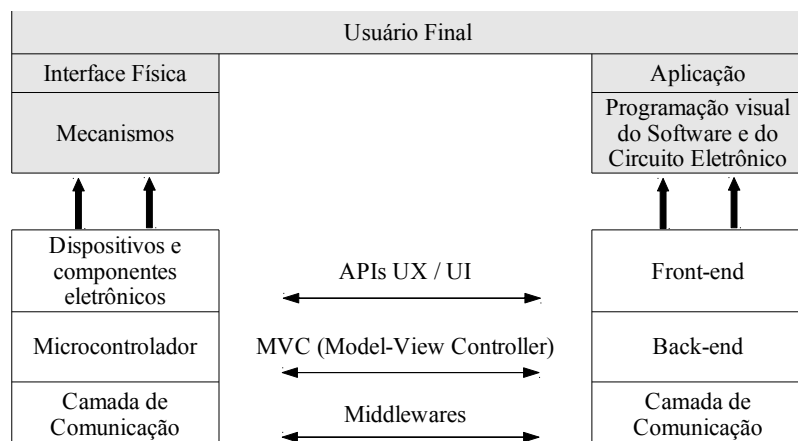


Figura 52: Arquitetura e níveis de abstração do tAMARINO.

O tAMARINO está na parte mais alta do nível de abstração, representado na Figura 52 pela “Aplicação”. O modelo é formado por duas camadas: uma que responde pela parte física, com os mecanismos, dispositivos e microcontroladores e; outra que interage diretamente com dispositivos físicos através de *middlewares*, representado pela “Camada de Comunicação”. As conversas entre o microcontrolador e o servidor *back-end* é realizada através do modelo MVC, no qual representa abstratamente entradas, saídas e operações. No *front-end*, onde são desenhados os módulos e as interações visuais com o usuário, a arquitetura é composta por APIs que dialogam com GUIs e com serviços em *back-end*.

7.2 Tecnologias utilizadas

A implementação reúne os conteúdos coletados nas fases anteriores e estabelece os requisitos mínimos para o desenvolvimento de uma solução que possa ser aplicada no mundo real. A respeito da arquitetura, o ambiente está no topo do nível de abstração em relação a IDE padrão do Arduino. O sistema é dividido em quatro camadas, onde os componentes físicos estão no nível mais baixo junto com os microcontroladores e os mecanismos. Acima, um *middleware* gerencia a comunicação entre as operações de hardware e os serviços de software. A parte de visualização é de responsabilidade da camada de aplicação, onde as linguagens de marcação controlam as interações dos usuários e enviam comandos para os servidores em *back-end*.

Aplicação (<i>front-end</i>)	tAMARINO			
	Bootstrap			
	HTML5/CSS3	jCanvas	jQuery	jsPlumb
Serviço (<i>Back-end</i>)	Breakout			
Middleware	Firmata			
Hardware	Arduino / microcontrolador			
	Sensores e Atuadores Físicos			

Figura 53: Arquitetura e Tecnologias utilizadas no tAMARINO.

Em relação a esse *framework*, seus princípios para o desenvolvimento de interfaces com APIs apoiam-se em quatro premissas:

- (1) ter um mecanismo em mente;
- (2) possuir sensores, atuadores e microcontroladores;
- (3) utilizar *scripts* e programas pré-compilados;
- (4) utilizar serviços de infraestrutura da *web* para realizar buscas e mapear APIs;

As APIs favorecem o desenvolvimento de inúmeros projetos e serviços agregados em

plataformas construídas a partir do *mash-up* de componentes. Por meio de uma pesquisa realizada nesta investigação, foi verificado um aumento no número de APIs e *Mashups*. O site *Programmable Web*⁷¹ contabiliza um total de 9281 APIs e 7084 *Mashups* catalogados até Junho de 2013. Já em Junho de 2008, Hartmann [2008] destaca o desenvolvimento de 3109 APIs e 775 *Mashups*. Isto significa um aumento de mais de 198% no desenvolvimento de APIs e mais de 814% em projetos recombinaados em apenas cinco anos.

O tAMARINO utiliza APIs e *Mashups* por dois motivos:

- (1) o aumento substancial de APIs para controlar microcontroladores através de linguagens como HTML5, CSS3 e *plugins* em Javascript ampliam as possibilidades de desenvolvimento de novos experimentos de protótipos rápidos em computação física;
- (2) a abordagem de APIs e *Mashups* agregam usuários em interação física a uma rede criativa de colaboradores fundamentados no movimento *open-source* e *Maker*, ampliando as possibilidades do protótipo.

Com um olhar para as facilidades no desenvolvimento através de APIs, o desenvolvedor do projeto *node-serialport*, uma aplicação em Node.js⁷² para controlar portas seriais, argumenta que o desenvolvimento e controle de sensores e atuadores deveriam ser fáceis e ágeis como esse código em Javascript:

```
$("#livingroom").bind("motion", function() {  
  $(this).find("lights").brightness("75%").dimAfter("120s");  
});
```

Esse argumento é fundamentado também por Zaefferer & Onken [2010], através de uma apresentação no JsConf (*JavaScript Conference*) chamada *Robotic Javascript*. Os pesquisadores defendem que tecnologias de interfaces de usuários em HTML5, junto com serviços assíncronos tipo Node.js e *sockets* que suportam servidores web, estão a serviço do desenvolvimento de novas soluções em controle de sensores e atuadores pelo *browser* [Zaefferer & Onken, 2010].

71 <http://www.programmableweb.com/>

72 Plataforma para desenvolvimento de aplicações web de forma rápida e robusta.

7.2.1 Detalhamento das aplicações utilizadas

O Breakout⁷³ é um *framework* de prototipagem que explora as interseções entre a web o mundo físico através da conexão com a plataforma Arduino e o protocolo Firmata, habilita o acesso às entradas e saídas do microcontrolador por *Javascript*. O *framework* se torna acessível quando o usuário tem conhecimentos básicos na linguagem de script e nos padrões de desenvolvimento web.

Desenvolvido por Jeff Hoefs⁷⁴, o Breakout é baseado no kit de ferramentas Funnel⁷⁵, que contém bibliotecas de software e hardware para realizar esboços em computação física conectando sensores e atuadores em diversas linguagens de programação (ActionScript3, Processing, Ruby etc). O Breakout foi desenvolvido como uma plataforma simples para *designers* e especialistas modelarem interfaces funcionais no contexto da computação física através do *browser*.

As classes do Breakout(BO) utilizadas para modelar a parte do *back-end* do tAMARINO foram:

- **BO.IOBoard:** negocia a comunicação entre a aplicação e o microcontrolador. A classe representa as entradas digitais e analógicas e as saídas do dispositivo que dão suporte para o protocolo I2C e enviam *strings* entre o IOBoard e a aplicação em *Javascript*.
- **BO.IO.BoardEvent:** representa o objeto de evento para ser disparado através do BO.IOBoard.
- **BO.Pin:** representa os pinos analógicos e digitais do dispositivo físico. Esse objeto é utilizado em diversas partes do *framework* e é a base para o desenvolvimento dos módulos e métodos da interface.
- **BO.PinEvent:** representa um evento disparado pelo objeto BO.Pin. O evento <<CHANGE>> é chamado toda vez que houver alterações nas portas digitais e analógicas.

⁷³ <http://breakoutjs.com/>

⁷⁴ <http://www.jeffhoefs.com/>

⁷⁵ <http://funnel.cc/>

- **BO.io.LED:** cria uma interface com um LED. Este objeto provem métodos rápidos para aplicar efeitos como frequência e intensidade.
- **BO.io.Potentiometer:** cria uma interface com um sensor analógico de entrada que pode ser um potenciômetro ou qualquer outro sensor numa porta analógica.
- **BO.io.Servo:** cria uma interface com um servo motor. Este objeto aplica um ângulo entre 0 e 180° no atuador.
- **BO.filtersScaler:** escalona os valores de entrada de um sensor para um escopo de valores mínimo e máximo especificado pelo usuário.
- **BO.generators.Oscillator:** cria *waveforms* que podem ser conectadas em pinos de saída. Este objeto é utilizado para gerar frequências de onda quadrada, linear, triangular e pulsos para Leds e Speakers.

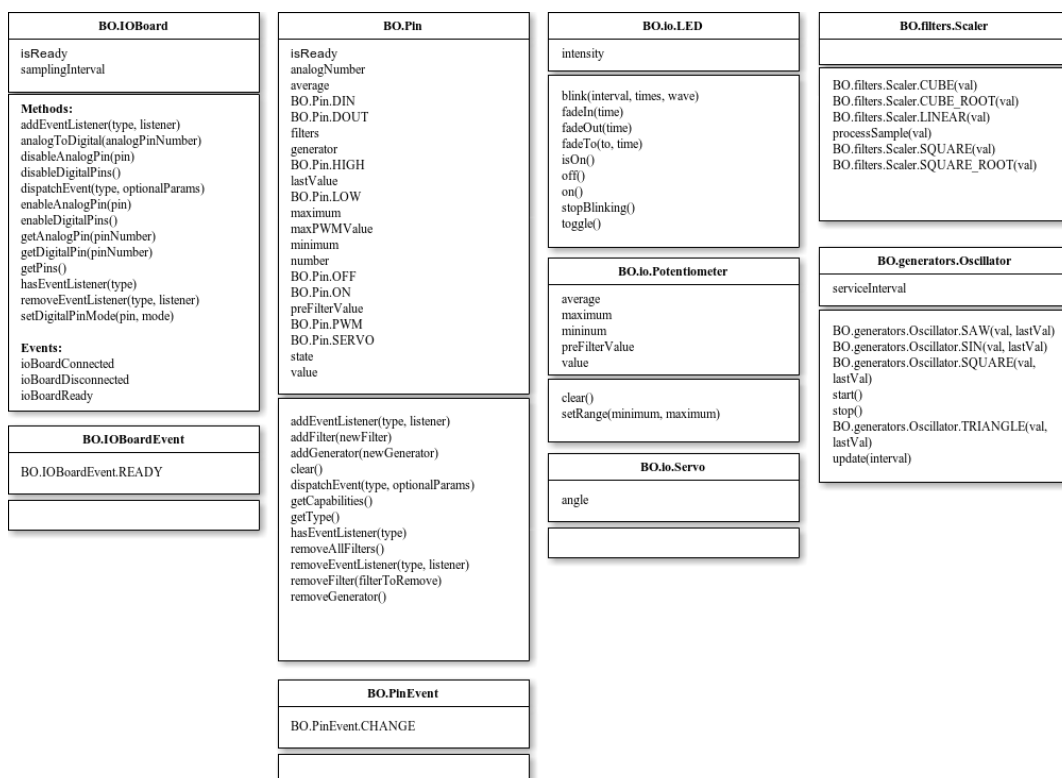


Figura 54: Diagrama de Classes utilizadas no desenvolvimento do protótipo do ambiente tAMARINO.

A modelagem das conexões entre módulos e métodos dos componentes de entrada e saída do sistema foi desenvolvida com a biblioteca jsPlumb⁷⁶. Esta API permite desenvolver conexões entre elementos visuais em plataformas web. O sistema utiliza SVG(*Scalable Vector Graphics*) em navegadores modernos e VML(*Vector Markup Language*) em navegadores antigos.

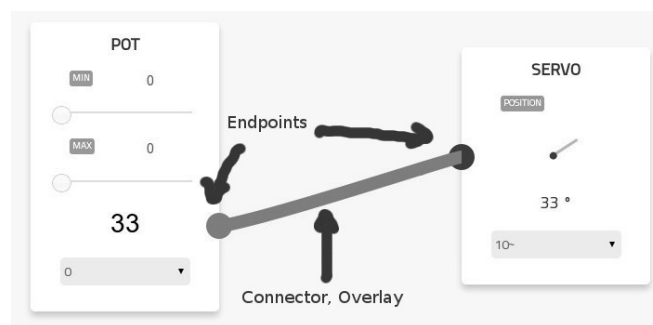


Figura 55: Descrição de uma conexão com as abstrações do jsPlumb.

O **jsPlumb** possui uma série de funcionalidades específicas para realizar conexões entre elementos. O núcleo do plugin é o objeto `<<Connection>>` que possui quatro conceitos:

- **anchor:** uma localização relativa ao elemento de origem em que um *Endpoint* pode existir. Pode ser referenciado por um nome ou um *array* com vários parâmetros;
- **endpoint:** uma representação visual de um ponto final de uma conexão. O desenvolvedor pode criar pontos-finais e adiciona-los aos elementos e habilitar funções como *drag-and-drop*;
- **connector:** uma representação visual da linha que conecta dois elementos. O jsPlumb possui quatro tipos de linhas: (1) bezier, (2) linha reta, (3) flowchart, (4) máquina de estado. O tipo escolhido para o tAMARINO foi o (1);
- **overlay:** um componente de interface de usuário para decorar o conector.

Uma conexão é feita com dois *endpoints*, um *connector* e zero ou mais *overlays*, que servem

⁷⁶ <http://jsplumbtoolkit.com/jquery/demo.html>

para juntar os elementos gráficos. O jsPlumb controla a criação e a configuração dessas conexões internamente, através de funções *bind* que possuem parâmetros para gerenciar os eventos chamados pelos usuários.

A simulação gráfica do circuito eletrônico no tAMARINO foi desenvolvida através da API **jCanvas**. Esta aplicação habilita o desenvolvimento de ferramentas gráficas através dos métodos de canvas, incorporados nas atualizações do HTML5. As principais características são:

- API simples para desenho de formas básicas e complexas;
- disponível em *Desktop* e *Mobile*;
- leveza, com menos de 25.5kb;
- código-fonte aberto.

Os componentes eletrônicos, a placa de prototipagem e o microcontrolador são modelados através de funções específicas para desenhar camadas, o **drawLayers()**. Dentro dessa função, são utilizados métodos tanto para chamar as imagens dos componentes (baseadas na biblioteca do Fritzing) quanto para desenhar os vetores que realizam as conexões.

A camada de interface com o usuário foi desenvolvida através da biblioteca jQuery⁷⁷. Com este *framework* é possível trabalhar dinamicamente na manipulação, controle, animação e operações com o DOM (Document Object Model). A API habilita funcionalidades de HTML em conjunto com CSS e Javascript em diversos *browsers*. Segundo a comunidade do jQuery, sua versatilidade e extensibilidade alteraram a forma como as interfaces para a Internet são construídas.

⁷⁷ <http://jquery.com/>

8. Avaliação

A metodologia aplicada na avaliação foi dividida em duas estratégias e compreendem uma prova de conceito para avaliações sobre os aspectos implementados e uma avaliação prática no sistema para validar a abordagem em diferentes perfis de usuários. Em resumo, as avaliações foram divididas em:

- (1) **prova de conceito** através de um vídeo que demonstra as possibilidades de interação com sensores e atuadores no tAMARINO. O objetivo é realizar uma coleta de opiniões a respeito dos conceitos, características, contexto de utilização, perfil de usuários, projetos relacionados, impressões sobre a interface, pontos positivos, negativos e sugestões. Usuários em potencial em diferentes níveis de experiência foram convidados a participar da avaliação;
- (2) **experimentos em laboratório** para avaliar os aspectos de três abordagens de sistemas para computação física com usuários iniciantes, não-especialistas e com experiência. O objetivo nesta etapa é obter dados a respeito da velocidade no ato da prototipagem de experimentos básicos e avaliações sobre o primeiro contato com os contextos desenvolvidos.

8.1 Prova de conceito

Esta avaliação tem o objetivo de apresentar as abordagens propostas para um grupo mais abrangente de usuários, incluindo pesquisadores, especialistas, designers e *makers*. A estratégia desta avaliação está na coleta dados qualitativos através de questionários online com perguntas abertas e fechadas. Os participantes avaliaram as funcionalidades a partir de um vídeo inserido no portal Vimeo⁷⁸ com link distribuído em listas de discussão de computação física, grupos de pesquisas, *hacklabs*, festivais e comunidades de tecnologia e arte nacionais e internacionais.

O vídeo produzido, com cerca de 12 minutos de interações no ambiente, tem o objetivo de apresentar as funcionalidades e as possibilidades do *framework*. A estrutura do vídeo é composta por duas telas, uma com o *screenshot* do ambiente e outra com um recorte dos componentes em manipulação, para apresentar o ciclo de prototipação digital até a

⁷⁸ <https://vimeo.com/65594452>

prototipagem do circuito na *breadboard*. O vídeo mostra o *feedback* digital e físico dos componentes nas duas telas para que os participantes avaliem a interface.

Depois de apresentar os componentes de entrada e saída e realizar interações básicas, o vídeo apresenta as possibilidades de conexões dinâmicas entre os módulos e métodos. São apresentados exemplos de conexão entre o módulo *button* com um módulo de Led, um Potenciômetro conectado a um servo motor e outras conexões possíveis. Após a visualização do vídeo, os usuários foram convidados a escrever suas impressões num **questionário online** bilíngue (PT/EN) com perguntas abertas e fechadas.

8.1.1 Perfil dos Participantes

Em relação ao perfil, dos 45 participantes, 69% são brasileiros residentes e não-residentes no país e 31% são estrangeiros. Sobre a relação dos participantes com a computação física, mais da metade se classificaram na escala de uma **relação máxima** com o contexto. Cerca de 19% se declararam no meio da escala.

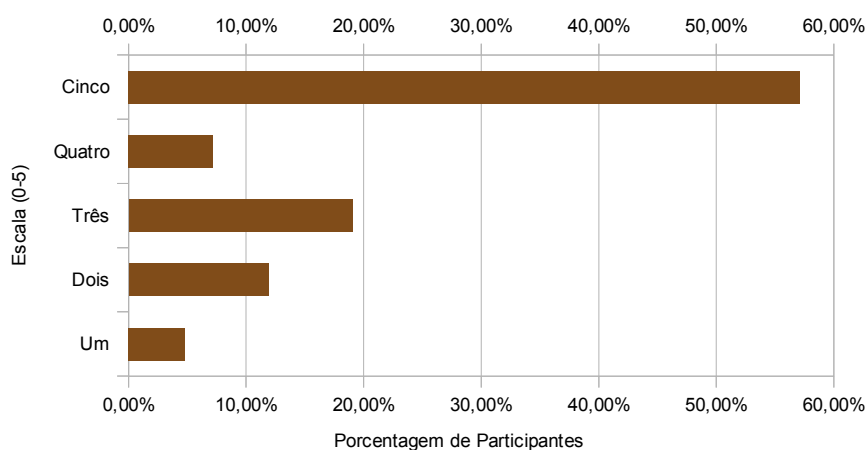


Figura 56: Nível dos participantes em relação aos conhecimentos em Computação Física na escala de 0-5, onde 5 significa integração total com o contexto.

Em relação ao nível de prática com microcontroladores, mais da metade declararam que **já realizaram experimentações avançadas** com bibliotecas específicas e conexões com outros

softwares via protocolo de comunicação. Cerca de 32% dos participantes já utilizaram o microcontrolador Arduino em funções básicas e apenas 5% declararam que nunca utilizaram.

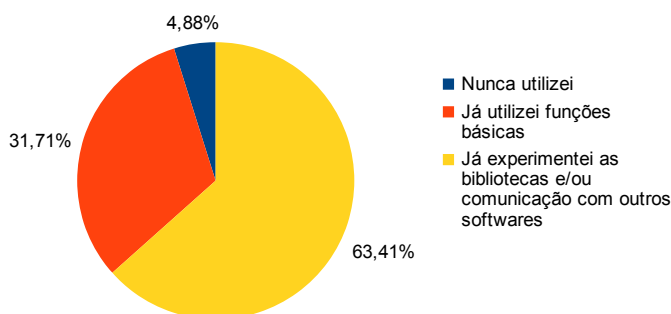


Figura 57: Experiência dos participantes no contexto da Computação Física.

Sobre as experiências práticas dos usuários, os dados mostram que mais de 80% dos participantes declararam que já desenvolveram ou participaram de algum laboratório para criar projetos interativos. Em relação à faixa etária, **2,38%** dos participantes possuem **menos de 18 anos**, **19,05%** entre **18 e 24 anos** e **21,43%** entre **25 e 30 anos**. O maior grupo de usuários está na faixa entre **31-40 anos**, com **33,33%** do total. Os usuários entre **41 e 50 anos** correspondem a **16,67%** e numa parcela **4,76%** estão os usuários entre **51 e 60 anos**. Apenas um participante possui mais de 60 anos de idade.

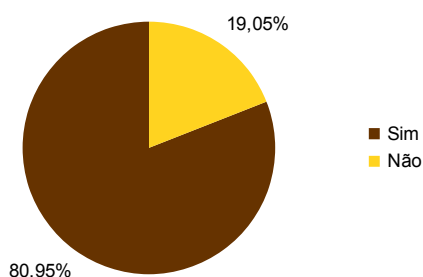


Figura 58: Experiência prática dos participantes com a plataforma Arduino.

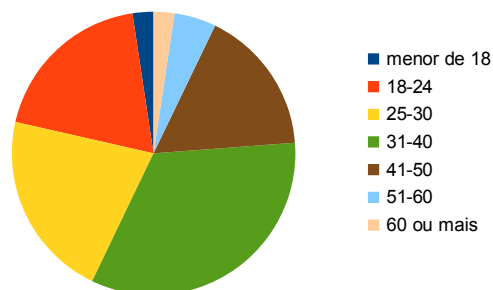


Figura 59: Faixa etária dos participantes.

A segunda parte do questionário online foi composta por **perguntas abertas** com o objetivo de coletar informações amplas sem um direcionamento específico. A intenção foi de identificar temas, conceitos e sugestões não pensadas previamente.

As perguntas foram compostas por:

- O que achou do conceito do aplicativo?
- Quais as características principais que você destaca na ferramenta?
- Você utilizaria esta ferramenta? Em quais contextos?
- Que perfil de usuários você acha que utilizaria a ferramenta?
- Você poderia citar alguns projetos que se relacionam com a ideia do tAMARINO?
- Quais as suas impressões sobre a interface da ferramenta?
- Para você, quais os pontos positivo da ferramenta?
- E os negativos?
- Quais as suas sugestões para aprimoramento da ferramenta?

8.1.2 Sobre os conceitos abordados

Em relação aos conceitos, a maioria dos participantes avaliaram a **integração** entre a programação do software e o esquemático eletrônico como ponto principal da ferramenta. Um dos participantes comenta a respeito: *“presents another way for newbies to understand the interaction between a sensor or switch and the outcome through a circuit.”* Na mesma orientação, outro participante comenta que :*“o foco é a prototipação rápida, sem preocupação com detalhes de implementação”*. Para um dos participantes com perfil avançado a ferramenta :*“causa um impacto pois parece ser muito fácil de controlar atuadores e sensores com esse software”*.

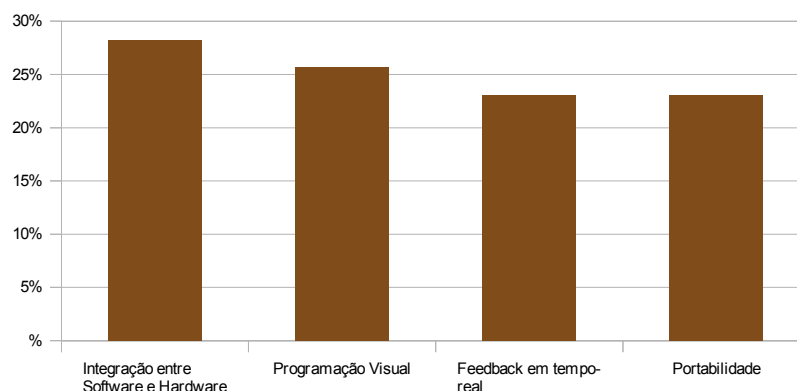


Figura 60: Principais princípios abordados pelos usuários em relação ao tAMARINO.

Dentre outros pontos positivos, destaca-se a relação entre funcionalidades por meio de programação visual, bem como o *feedback* em tempo-real e a portabilidade, com a proposta de utilizar o *browser* como *suíte*. Um dos participantes comenta a respeito da velocidade afirmando que o ambiente “*é muito interessante, principalmente por causa da rapidez das respostas e da interatividade*”. Outro participante argumenta que: “*o feedback visual é (...) didático e agradável, com um bom design*”.

A respeito da facilidade no uso provocada pela programação visual e o feedback em tempo-real, um dos usuários sintetiza: “*a possibilidade de poder conectar atuadores de forma simples e rápida numa interface web é incrível e facilita muito (...) os trabalhos*”. Vale também salientar sua utilidade nos ambientes de aprendizagem: “*very useful for my students who learn Physical Interface Design in my class*”. Alguns participantes elogiaram também o design da interface: “*bastante agradável e chamativo*”, “*bem elaborado e fácil de utilizar*” e argumentaram sobre a didática e a intuitividade: “*gostei do conceito de ligar caixinhas*”.

8.1.3 Principais características abordadas

A integração entre a visualização da programação do hardware ao mesmo tempo que a do software foi uma das característica mais marcante da ferramenta, como cita um dos

comentários: *“high level components, low level schematic”*. Salienta-se entre a análise dos participantes, a integração entre a rapidez e a facilidade para a construção de protótipos. Outro ponto observado diz respeito à forma automatizada da visualização da prototipagem eletrônica: *“mostrando o circuito e todas as ligações necessárias pra ligar aqueles determinados componentes”*(sic). Além disso, o feedback em tempo-real se traduz como uma forma *“didática”* para os usuários. Outro usuário aponta para o *“Auto wiring guide when the bread board is dragged onto the screen.”* como um outro ponto interessante da ferramenta.

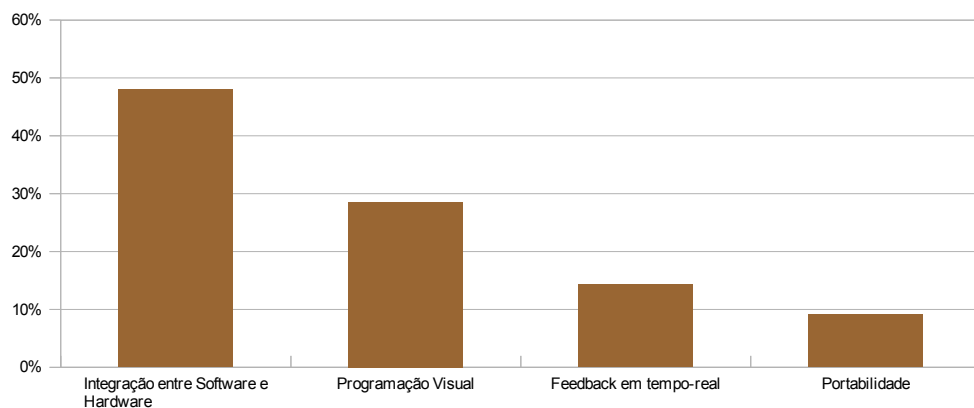


Figura 61: Nível de características mais marcantes pelos participantes

A programação visual também foi uma das características mais apontadas pelos usuários, com argumentos do tipo: *“forma de interligar os componentes é bem interessante e fácil de usar”*. Um dos participantes explica a característica do *drag-and-drop* como: *“cada componente é uma caixinha com suas propriedades”* e *“poder ligar componentes diferentes”*. Outros usuários argumentaram que o fato de ser modular e em tempo-real, apresenta *“simplicidade”* e *“clareza”* na programação e propõe uma *“interatividade instantânea”* com os componentes de entrada e saída.

Ainda sobre a programação visual, um dos usuários lembra que ela está relacionada com a possibilidade do usuário interagir sem *“precisar escrever código”*. Para alguns participantes, além do *“fato de ser mais pratico de programar”*, a *“facilidade de juntar várias funções”* e o *“feedback visual”* está diretamente conectado com a proposta do ambiente.

Em relação à portabilidade, um argumento interessante foi em relação ao funcionamento através do *browser*: *“faz com que ele seja plugável em várias arquiteturas e embedado em qualquer projeto web”*(sic). O usuário qualifica que o tAMARINO pode ser :*“uma ferramenta extremamente viral e flexível”*. Outro usuário argumenta também que a arquitetura :*“can make authoring components easier”*.

Os participantes salientaram que a ferramenta é *“intuitiva”*, possui *“layout moderno”* e *“boa tonalidade”*. A ferramenta também foi classificada como *“didática”* por sua interface *“simples e amigável”*, além de *“ensinar”* como se monta o esquemático. Em relação à praticidade, os participantes argumentam que, além do fato de ser *“fácil”* e *“rápida”* para a montagem de protótipos iniciais, é rápida nos *feedbacks* visuais para o usuário.

8.1.4 Contexto de utilização da ferramenta

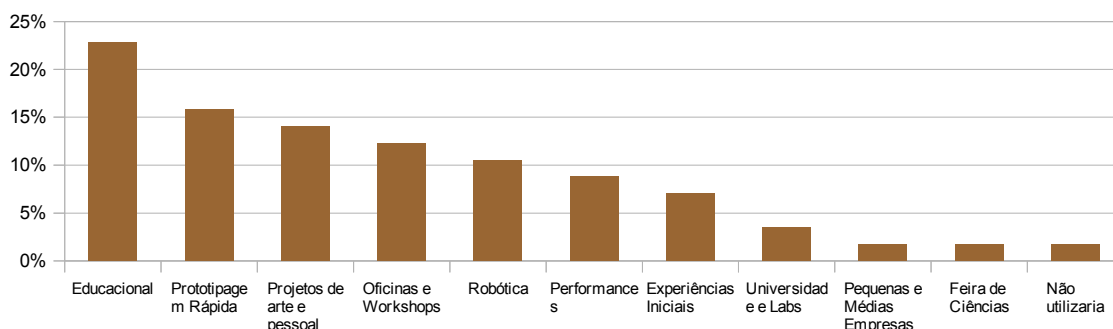


Figura 62: Principais contextos de utilização levantados na entrevista sobre o tAMARINO.

De acordo com a análise da Figura 62, o contexto educacional foi o mais comentado pelos participantes como um excelente contexto para uso do tAMARINO. Quando perguntado sobre o uso da ferramenta com usuários iniciantes, foi respondido que *“yes I would, probably with beginner classes and to allow non-specialists (like designers) to make early prototypes of an interactive project”*.

Em contraposição, um dos usuários comenta que não utilizaria no contexto educacional, pois defende o uso dos tutoriais como uma ajuda valiosa para as práticas iniciais. No geral, foi observado como ótimo o uso da ferramenta no ambiente educacional pois possibilita

“aprender o básico de Arduino” e “fazer projeto bem rápido e não gastar tempo com código”. Para outro usuário, a palavra-chave que descreve o *framework* também é educacional: *“pode ser uma ferramenta que chame o indivíduo a ***perder*** receios e se aproximar de novos engenhos físicos computacionais”* (sic).

Em relação aos usos em projetos de arte e performance, um dos participantes acha *“que a ferramenta facilitaria bastante o trabalho de artistas menos envolvidos com computação”.* Outros usuários comentam sobre o uso em contextos *“experimentais de arte e música”* e também *“to build musical instruments”.* Outros tópicos abordados foram o *“desenvolvimento de protótipos de pesquisa para projetos pessoais”*, *“aplicações criativas”* e *“aplicações não-comerciais”.*

Outros contextos de uso foram os *workshops* e as oficinas de introdução a computação física. Para um dos usuários, existem dificuldade para o primeiro contato do participante com a programação e a montagem do circuito: *“normalmente existe uma dificuldade dos alunos no começo de fazerem os circuitos funcionarem e isso gera frustrações”.* Para ele, *“O tAMARINO é o tipo de ferramenta que pode eliminar essas frustrações”.*

Dentre outros possíveis usos da ferramenta, estão o desenvolvimento de robôs em oficinas e “feiras de ciências” bem como na criação de interações educacionais em robótica e física. Um usuário visualiza a ferramenta *“como um aliado na demonstração física de fenômenos com resistência, luz e temperatura”.*

Alguns usuários comentam que utilizariam apenas *“for fun in my spare time”.*

8.1.5 Perfis de usuários

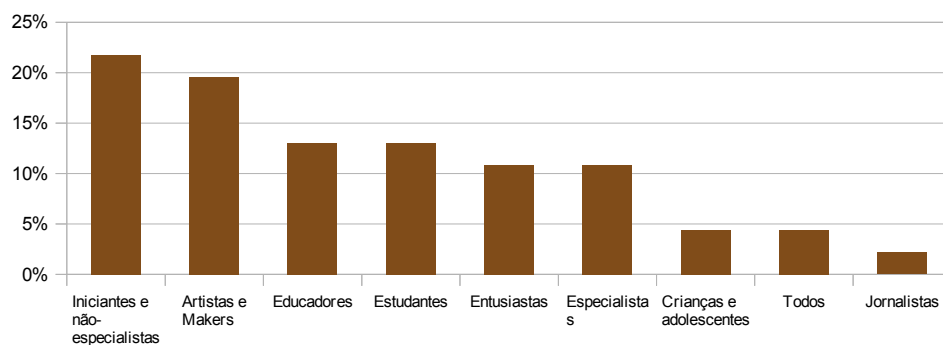


Figura 63: Principais perfis de usuários indicados pelos participantes para utilizar o tAMARINO.

Mais da metade dos participantes elegeram os usuários iniciantes, não-especialistas, artistas, *makers* e educadores como os principais perfis de usuários para a ferramenta. Um usuário argumenta que o projeto é para “*pessoas que não tem conhecimento de eletrônica mas querem implementar algum projeto*”. Já outro usuário contra-argumenta que a ferramenta é voltada apenas para quem já possui um “*conhecimento básico em eletrônica*”.

Durante as análises do discurso, foi percebida uma dualidade em relação ao uso da ferramenta para leigos e especialistas como no comentário de alguns participantes: “*Leigos, para começar a utilizar o Arduino*” e “*Especialistas, para prototipar de forma mais rápida que a convencional*”. Outro usuário sobrepõe essa dualidade, afirmando que a ferramenta pode funcionar “*de iniciantes à desenvolvedores*”.

Alguns participantes generalizaram o contexto dos usuários abrangendo diversos segmentos como “*People who read Make magazine*”, “*Teachers*”, “*Students of Pcomp*”, “*Artists*”, “*Parents with kids*”. Alguns comentários foram em relação o uso da ferramenta para “*crianças e adolescentes*”, “*usuários com pouca ou nenhuma experiência com programação*” e também usuários com perfis fora-da-caixa, como “*Jornalistas*”.

8.1.6 Projetos relacionados

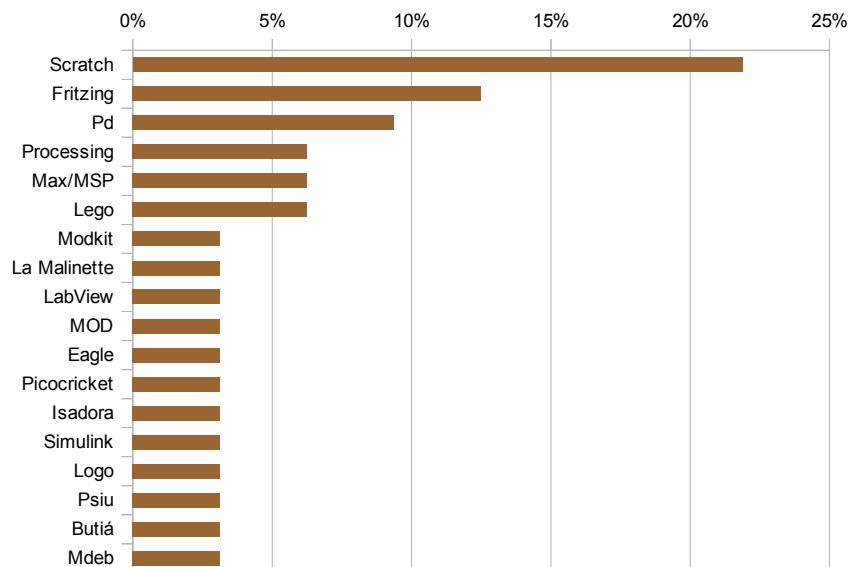


Figura 64: Projetos relacionados com o contexto do tAMARINO.

A Figura 64 indica as ferramentas citadas pelos participantes que possuem algum tipo de semelhança com o tAMARINO. A interface mais lembrada foi o Scratch, talvez pela facilidade no uso para iniciantes. Em segundo lugar, o Fritzting, pela funcionalidade proposta de montagem do circuito eletrônico e na utilização da biblioteca de imagens vetoriais dentro do sistema automático.

O Pure Data (Pd) foi lembrado em um dos comentários : “*por se tratar de um ambiente de programação que se utiliza de blocos e da conexão entre eles para criar programas complexos*”. Os softwares da Lego e o Processing foram também lembrados pela expressividade, interatividade e facilidade para o contexto dos não-especialistas. Um usuário comenta que essas ferramentas “*pretendem facilitar a tarefa de programar para designers*”.

Uma observação importante está na diferença entre o Scratch e o projeto Modkit. Apesar dos dois projetos utilizarem os mesmos paradigmas de orientação à blocos, o Modkit integrou melhor a comunicação entre as features do Scratch com o Arduino, permitindo que usuários

programem pelo navegador de forma mais rápida do que a instalação e compilação do Scratch. O resultado acima reflete mais para o Scratch por ser mais difundido, porém, o Modkit é muito mais prático.

Um projeto relacionado que também possui uma abordagem semelhante é o francês La Malinette⁷⁹, que incorpora “*une mallette pédagogique open-source pour faciliter la programmation interactive*” que mistura diversas características do Pure-data, Processing e Arduino num ambiente que incorpora *patches* e bibliotecas semelhantes ao Pure-data. A abordagem da La Malinette contempla não só o contexto da Computação Física, mas abrangem outras áreas do design de interação. A crítica da ferramenta está na especialização dos módulos que provoca emaranhamento e dificuldades na interpretação e no foco para especialistas.

8.1.7 Impressões da interface visual

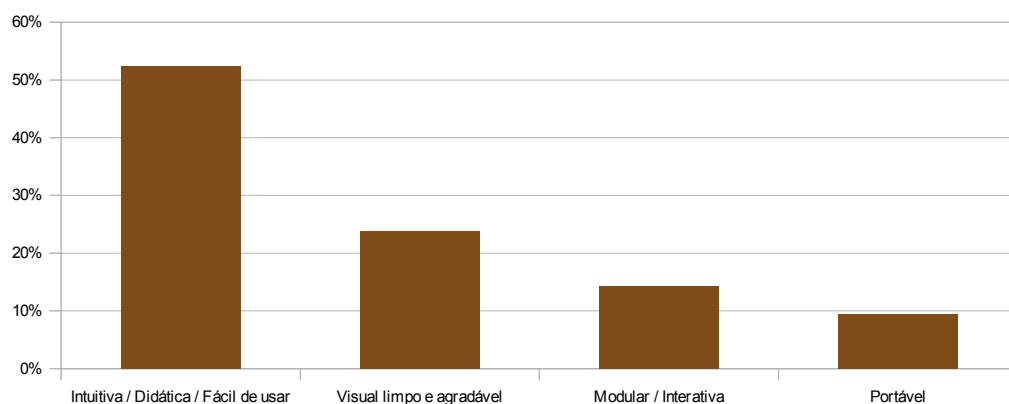


Figura 65: Impressões marcantes da interface visual do tAMARINO.

Mais da metade dos participantes elegeram a interface como intuitiva, didática e fácil de usar. Um dos participantes faz um comentário que contempla um sentimento coletivo: “*O design é legal, gostei da paleta de cores utilizada. A interação é muito boa e fácil de entender e começar a testar sem se preocupar muito com funcionamento*”(sic). Em contrapartida: “*se tivesse que mudar algo, seria apenas as linhas de ligação entre componentes, que parecem*

79 <http://reso-nance.org/malinette/#/accueil>

um pouco grosseiras se comparadas com o resto dos itens” (sic).

Outro usuário comenta a respeito da interface modular: *“vc desenhar atraves de drag-and-drop, eu gostei muito, pq vc fica bem livre pra organizar os atuadores (representados por aquelas caixas) na tela”(sic).* Sobre as conexões, o usuário comenta que *“as ligações foram bem intuitivas, com pontos do lado esquerdo representando entradas, e pontos do lado direito representando saídas”*. A modularidade é vista como uma das principais vantagem da interface porque evita *“uma tela com milhões de botões, parâmetros e opções pro usuário”*.

Alguns usuários comentam que o *“drag-and-drop facilita por se assemelhar ao contexto real”* e é prático porque *“tem tudo em uma parte só”*. Em relação à portabilidade, a utilização do browser como o ambiente para funcionar o projeto soa como um atrativo e também a *“interação por mouse”*. No geral, os usuários compartilham do comentário: *“looks very easy to use and intuitive”*.

Foi percebida uma crítica em relação ao sistema de organização dos componentes na breadboard virtual, principalmente quando se utiliza vários componentes. Um participante argumenta que *“when the breadboard is dragged onto the screen, the input and output objects should reorder to make space for the breadboard and make the display of wiring flows tidy. This is specially important in larger projects toward the end of the video with many objects.”*

8.1.8 Pontos positivos e negativos

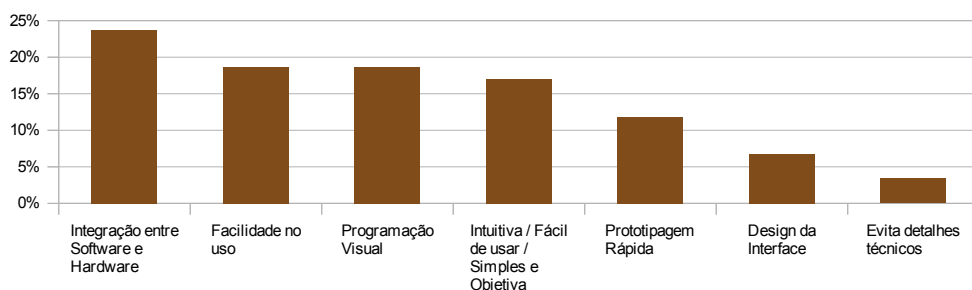


Figura 66: Pontos positivos destacados pelos usuários.

De acordo com a avaliação dos participantes, a integração, a facilidade no uso da interface e a programação visual são os principais pontos positivos da ferramenta. Um dos usuários sintetiza isso no argumento sobre a facilidade e expressividade na prototipação: *“Fast prototyping without the need for programming. Very good for planning a prototype”*. Outro usuário completa que o ambiente: *“acelera o processo de prototipação e facilita a construção de artefatos eletrônicos para pessoas que tem pouco conhecimento sobre o assunto”*.

Um dos pontos interessantes em uma das respostas foi a relação entre o foco nos objetivos do projeto ao invés dos detalhes técnicos. Sobre a facilidade no uso para iniciantes, um dos participantes argumenta que a ferramenta é *“fácil de começar a usar, não precisa saber usar o arduino, nem programar, perfeito para newbas e pessoas que não querem ser especialistas”* (sic). Um outro ponto positivo levantado foi em relação a integração entre a prototipagem e a programação, o que oferece, segundo um dos participantes, *“múltiplos pontos de vista do mesmo projeto”*. Outro contexto levantado foi sobre o *“feedback instantâneo com o hardware”* e o *“drag-and-drop”* integrado com os componentes físicos.

Vários participantes comentaram a respeito dos aspectos da interface de interação com o usuário, os argumentos são sobre a simplicidade na interação com o visual “limpo” e “agradável” e a respeito da “rapidez” e “agilidade” da ferramenta. Também foi observada uma relação entre o design da interface com a facilidade no uso: *“intuitivo, fácil manuseio, ótimo layout, fácil para quem não sabe programar”*.

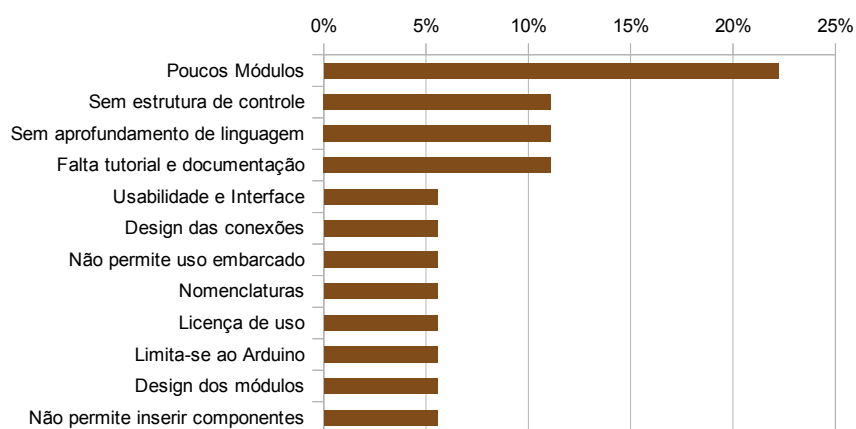


Figura 67: Pontos negativos da ferramenta.

O fato da versão do protótipo utilizado nos testes e na produção do vídeo abarcar somente seis componentes de entrada e saída, de certa forma causou uma impressão de poucos módulos no ambiente. Porém, a estrutura da arquitetura do tAMARINO permite que os módulos sejam facilmente estendidos. Os usuários também sentiram falta de uma estrutura de controle que pudesse fornecer objetos intermediários entre os módulos de sensores e atuadores para realizar um controle mais avançado. A falta de objetos de controle para um dos usuários: *“limita o potencial da ferramenta”*.

Em relação ao aprofundamento da linguagem, alguns usuários comentaram que não fica claro no vídeo se o ambiente possui uma interface baixo nível para que o usuário possa entender o funcionamento dos módulos através de linguagens intermediárias com permissão para edição. Esse assunto deixa claro a vontade dos usuários experientes construírem novos módulos e utilizar a ferramenta para projetos mais complexos.

Muitos participantes questionaram também se a ferramenta tem o código-fonte aberto e atentaram para a falta de uma documentação e um tutorial técnico sobre a ferramenta. Outro comentário foi em relação ao ambiente funcionar apenas no modo conectado, o que inviabiliza do usuário desconectar o microcontrolador e utilizar o código embarcado. Essa demanda ressalta a necessidade de uma máquina para fazer o *upload* do projeto para a placa.

A nomenclatura foi questionada pelo fato do software em inglês, o que para alguns participantes é uma dificuldade. Outra questão foi em relação ao uso do ambiente pelo *browser* depender de uma conexão estável, o que deixa claro que há uma preocupação com lugares que não possuem uma boa banda para o tráfego. É importante definir bem os conceitos e as diferenças entre ser baseado na web e funcionar no browser: o ambiente tAMARINO funciona em ambos os casos, todos os serviços podem rodar tanto numa instalação online quanto local – uma vez carregado pelo browser, ele não necessita mais da conexão.

8.1.9 Sugestões e Críticas

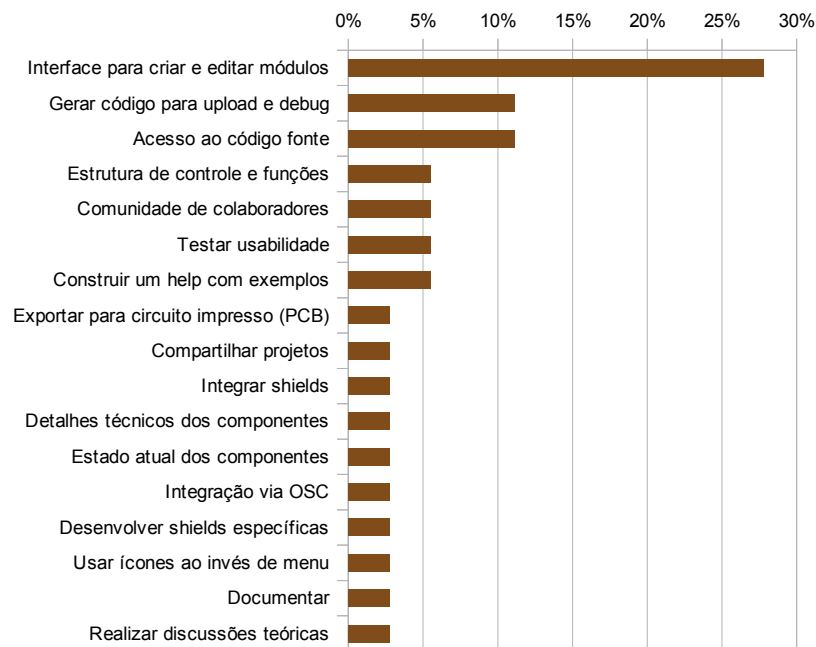


Figura 68: Lista de sugestões dos participantes para a ferramenta.

A necessidade dos usuários em aprofundar o contexto do protótipo para realizar outras interações mais específicas são refletidas na sugestão unânime da necessidade de uma interface para construir e editar módulos de entrada e saída no tAMARINO. Essa demanda acena para um interesse de usuários experientes em expandir as bibliotecas do sistema para dar suporte às diferentes demandas de sensores, atuadores e *shields* cada vez mais presentes no dia a dia do universo *Maker*.

Esse mesmo sentimento pode ser refletido na necessidade de se criar uma comunidade de colaboradores do sistema para que novos módulos sejam desenvolvidos coletivamente. Os participantes também salientam a possibilidade de compartilhar projetos e construir um *help* com exemplos de uso na própria ferramenta, o que reforça a necessidade de se criar uma comunidade de entusiastas do ambiente. As mesmas preocupações do ambiente *Fritzing* também foram pautadas como na possibilidade de exportar o circuito para impressão em

placas industriais.

Uma outra funcionalidade foi a de geração do código do projeto para realizar *upload* para o microcontrolador e trabalhar em ambientes embarcados. Diante disso, existe de fato uma necessidade de viabilizar o desenvolvimento de um sistema que gere o código e faça *upload* sem que o usuário perca a interatividade proposta na programação visual do ambiente em tempo-real. Outro contexto interessante foi a sugestão de criar uma interface OSC para que o tAMARINO comunique com outros softwares interativos como Pure Data, Processing e Openframeworks, tornando a ferramenta num ambiente de controle de componentes eletrônicos e da comunicação com outros contextos de aplicações interativas.

Foram levantados alguns detalhes aplicados no contexto da eletrônica, como o desenvolvimento de um módulo de monitoramento dos componentes para saber o estado atual de cada sensor e atuador conectado, e um relatório completo dos componentes utilizados na prototipagem eletrônica. Um participante sugere “*um modo virtual de trabalho*” para possibilitar o “*planejamento de projetos de computação física inteiramente no computador, ajudando a otimizar os recursos*”.

Do ponto de vista da pesquisa acadêmica, foi sugerido mais testes de usabilidade com a ferramenta para coleta de dados e aprimoramento do ambiente. Também foram sugeridos a troca do contexto de menu de textos para ícones. Além disso, a documentação de todo o processo da pesquisa e a realização de discussões teóricas a respeito de assuntos como computação física, Internet das coisas e design de interação física são sugestões de alguns participantes para a evolução das abordagens propostas no tAMARINO.

8.1.9.1 Críticas especializadas

Durante a divulgação e convite para a prova de conceito, o autor desta dissertação enviou um e-mail para a lista de discussão oficial do projeto Arduino com o objetivo de convidar usuários especialistas para validarem as abordagens desenvolvidas. Entre as respostas, diferentes críticas e elogios foram tecidos por pesquisadores e personalidades da computação física, entre eles destacam-se:

Massimo Banzi, o CEO do Arduino:

Hello Ricardo

I'm the CEO of Arduino

That's a nice demo. would you like to collaborate with us to refine your prototype?

m

--

*Massimo Banzi
CEO Arduino SA*

Tom Igoe, autor do livro Physical Computing e Professor do NYU/ITP⁸⁰:

It looks a lot like Philip Van Allen's NETLab Toolkit. You may want to talk to him about it.

Philip Van Allen's, desenvolvedor do NLTK e Professor do Art Center College of Design também comentou sobre o tAMARINO, principalmente sobre a portabilidade do projeto em navegadores web:

Hi Ricardo,

As Tom mentioned, I created the NETLab Toolkit. You've done some nice work. If you want to correspond offline of this list, please email me privately. By the way, we're about to embark on a complete redesign of the NETLab Toolkit (to update and better work in the browser and mobile), and it would be interesting to learn what you've found in your process.

Best,

.phil

80 New York University - <http://itp.nyu.edu/itp/>

8.2 Experimentos em laboratórios

Os experimentos em laboratório foram organizados para avaliar o contato dos usuários com diferentes contextos de ambientes para a computação física, entre eles Arduino IDE, Modkit e tAMARINO. O objetivo do experimento é desenvolver um *Hello World* com sensores, atuadores e uma placa de prototipagem para compor uma situação real. O exercício proposto inclui o controle individual de cada componente e a realização de interações dinâmicas entre eles.

8.2.1 Perfil dos laboratórios e dos participantes

Os experimentos foram aplicados em **três laboratórios** com perfil distintos: o **Lab 01** é formado por estudantes de hardware, robótica e manutenção de computadores com faixa etária entre **18 e 25 anos** e com **pouca experiência** em aplicações de Computação Física. No **Lab 02**, os estudantes possuem foco em *design*, mas na maioria sem experiências em aplicações interativas. Já no **Lab 03**, o perfil é formado por estudantes universitários de **arte e tecnologia** com diferentes níveis de experiência em aplicações.

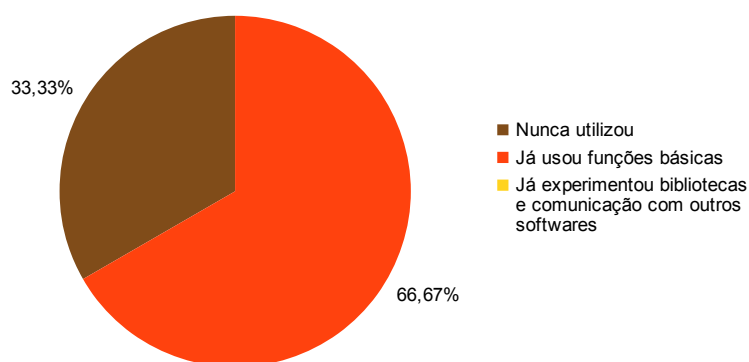


Figura 69: Perfil dos participantes no Lab 01.

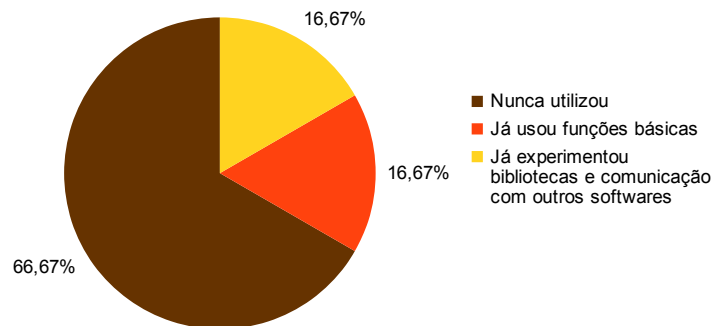


Figura 70: Perfil dos participantes no Lab 02.

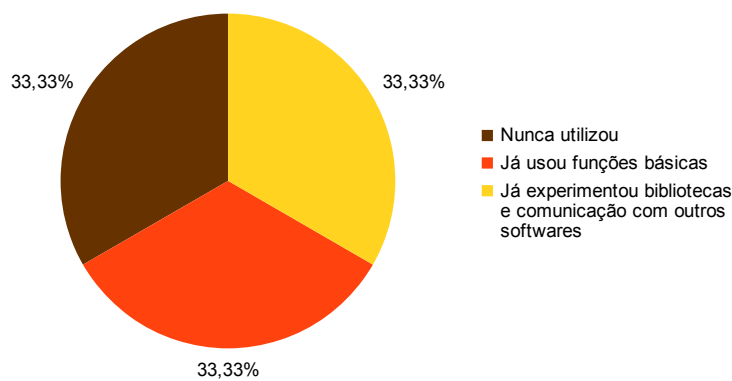


Figura 71: Perfil dos participantes no Lab 03.

Em cada laboratório, **três grupos** com diferentes níveis de conhecimento mantem um equilíbrio entre participantes sem conhecimento técnico com iniciados em alguma pesquisa. Cada grupo tem o objetivo de aplicar os experimentos solicitados num dos contextos da programação, seja textual, em blocos ou em módulos. A proposta de dividir os experimentos em grupos serve para obter informações fieis com a proposta de acelerar a prototipagem.

A estratégia principal dos experimentos é avaliar o **tempo** em que o usuário leva para desenvolver um experimento básico nas três abordagens propostas. Os grupos são habilitados a pesquisar as soluções de código e de circuito eletrônico em tutoriais na Internet e a avaliação não considera as questões de instalação do software e do hardware.

8.2.2 Resultados dos experimentos

Nos experimentos realizados no **Lab 01** e **02**, os grupos que utilizaram o **tAMARINO** foram os primeiros a terminar o exercício proposto com sensores e atuadores. Os tempos foram aproximadamente **20 minutos** no primeiro laboratório e **30 minutos** no segundo. No **Lab 03**, o grupo que utilizou o **tAMARINO** finalizou em aproximadamente **45 minutos**.

Os grupos que utilizaram o **Modkit**, terminaram o experimento em aproximadamente **50 minutos** no **Lab 01**. Já no **Lab 02**, o grupo finalizou o experimento em **37 minutos**. Por último, o grupo no **Lab 03** precisou de **47 minutos** para finalizar o experimento.

Em relação aos experimentos com a **IDE** padrão do Arduino, o grupo do **Lab 01** não conseguiu finalizar os exercícios durante o período de **60 minutos**. Já no **Lab 02**, o experimento foi concluído em **34 minutos** e os mais rápidos foram os grupos do **Lab 03**, que precisaram de **31 minutos** para finalizarem seus experimentos.

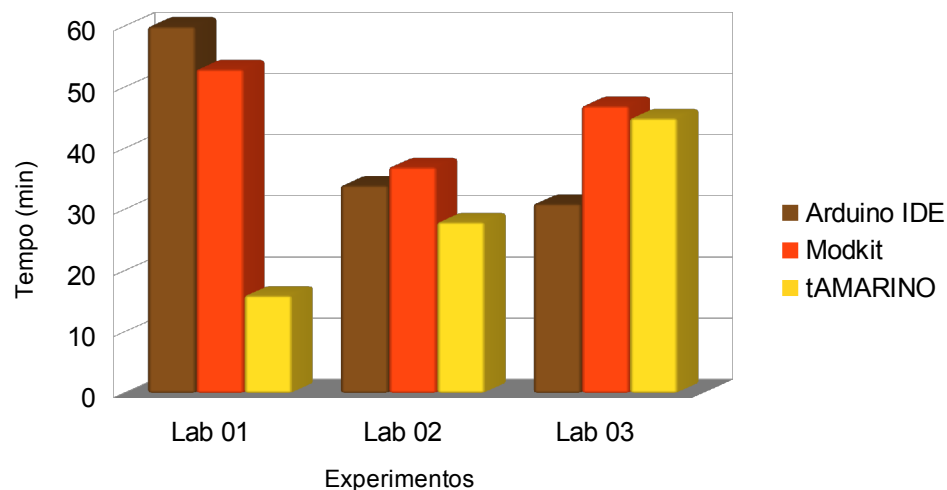


Figura 72: Relações entre o tempo para realizar as experiências com Computação Física em diferentes abordagens.

A partir da análise dos gráficos da Figura 72, em dois terços dos experimentos o ambiente **tAMARINO** foi o **mais rápido**, tendo uma velocidade superior a 150% em relação ao grupo que desenvolveu o experimento na plataforma **Modkit** no **Lab 01**. O grupo que utilizou o

Arduino IDE **não conseguiu finalizar** o experimento por problemas de sintaxe do código e nas conexões dos componentes na *breadboard*, o que confirma a dificuldade na realização dos primeiros experimentos num ambiente em modo texto.

A análise do segundo laboratório é semelhante ao primeiro, onde o grupo que utilizou o tAMARINO terminou mais rápido os experimentos. O grupo que utilizou a IDE do Arduino ficou em segundo e por último o grupo do Modkit. A velocidade superior do Arduino IDE em relação ao Modkit foi pela experiência em Processing de um dos participantes do grupo, o que facilitou o contato inicial por conta da semelhança na linguagem e na interface. Além do conhecimento em outras linguagens, o grupo encontrou tutoriais que auxiliaram na montagem e na cópia do código-fonte. No Modkit, os usuários montaram o circuito através de exemplos na Internet mas tiveram dificuldades para conectar os blocos e realizar a programação graficamente.

O perfil do terceiro laboratório destoa dos outros por terem participantes de diferentes níveis de conhecimento em interatividade. Em cada grupo, existia um líder com conhecimentos avançados em programação com Arduino. O restante era formado por iniciantes no assunto mas com conhecimento das possibilidades. O perfil do usuário experiente influenciou no resultado do exercício. Os usuários experientes descobriram formas ágeis de encontrar soluções em tutoriais e resolver de forma mais rápida através da IDE do Arduino.

Numa análise geral, a abordagem do ambiente tAMARINO mostra-se mais rápida para usuários iniciantes e não-especialistas técnicos em relação ao contexto de blocos do Modkit e da programação textual do Arduino IDE. A medida em que os usuários se tornam mais experientes na programação do microcontrolador, a programação textual não se mostra um problema, porém a conexão com a Internet é fundamental para a busca de soluções em tutoriais.

8.2.3 Opiniões sobre os experimentos e as ferramentas

Em relação às opiniões a respeito dos ambientes, os usuários que utilizaram o tAMARINO comentaram ser “*bastante fácil de utilizar*”, “*bem fácil de entender porquê já vem com pequeno manual*” e “*bem interativo*”. Outros participantes apontaram como positivo os

aspectos visuais como “*a parte gráfica dos cabos se movimentando*”(sic) e, a respeito do acesso aos iniciantes, como de “*fácil compreensão e estimulante para praticar*”(sic).

Em relação ao Modkit, alguns participantes não acharam o ambiente de fácil uso mas avaliaram que “*evita você escrever*” citando a parte da programação em blocos. Os participantes tiveram dificuldades em realizar a configuração dos componentes e utilizar os blocos nas iterações necessárias para controlar os sensores e atuadores. Já para outros participantes, as maiores dificuldades está na busca de exemplos de projetos na Internet que sirvam como guia no aprendizado.

Os participantes que utilizaram a IDE do Arduino avaliaram a ferramenta pouco intuitiva e não acharam o ambiente fácil de usar para quem não compreende lógica de programação. Os participantes avaliaram como positiva a possibilidade de usar exemplos prontos, o que na opinião de um dos participantes facilita o processo: “*interessante pois existem exemplos que ajudam muito*”. Em contrapartida, outro usuário comenta que é necessário “*ter conhecimento básico de lógica de programação*” para utilizar e alterar os códigos. A semelhança entre as abordagens da linguagem *Java-like* da IDE do Arduino foi bem aceita para os participantes que já tinham este conhecimento.

Como ponto negativo geral, os participantes apontaram para a falta de tutoriais em português com exemplos de códigos e circuito. Um dos participantes confirma os problemas da linguagem: “*por ser em inglês, atrapalhou um pouco*”. Outro usuário comenta que, para ele, o ponto negativo também foram os tutoriais em inglês mas contra-argumenta a respeito das imagens nos tutoriais: “*mas com imagens que ajudam muito*”.

9. Conclusão

Diante do que foi discutido e analisado nesta dissertação, a conclusão desta pesquisa é de que existem possibilidades de modificar as experiências nas ferramentas de prototipagem em computação física para experiências mais atrativas, rápidas e expressivas. A principal contribuição desta pesquisa foi o desenvolvimento do tAMARINO como uma solução original e única de ambiente para prototipagem de software e hardware em tempo-real por meio de programação visual em ambientes portáteis. Uma contribuição secundária foi o desenvolvimento do fluxograma com as atividades típicas da computação física. Não foi encontrada na literatura nenhuma abordagem neste aspecto.

A proposta de integrabilidade desenvolvida no tAMARINO apresenta um aumento no nível de abstração nas camadas de software e de hardware ao mesmo tempo, onde as peculiaridades da programação e das conexões dos circuitos eletrônicos são integradas num único sistema para diminuir o *time-to-market*. Além disso, o ambiente facilita o contato inicial de usuários acostumados com ambientes *high-level* e serviços com usabilidade acima da média, como as aplicações e serviços na Internet.

Os processos de design aplicados no desenvolvimento do tAMARINO foram compostos por diversos ciclos de prototipação e experimentação. Os dados elucidaram a necessidade por ferramentas visuais para além dos aspectos puramente técnicos da computação física, capazes de provocar outras possibilidades de interações com GUIs que liberam os usuários para pensarem nas ideias ao invés dos problemas técnicos.

A escolha por uma arquitetura baseada em *browser* ultrapassam as questões de portabilidade. As aplicações para interatividades físicas na web, chamada de “Internet das coisas”, promete se transformar na próxima revolução dos sistemas embarcados com tecnologias de rede em diversos contextos da vida cotidiana [Trifa, 2009]. As ferramentas que suportam as infra-estruturas das redes são estratégicas nesse ponto, por prover novas interações e funcionalidades nesses aspectos. O tAMARINO se insere nesse contexto como uma ferramenta para possibilitar uma experiência de esboço de interações com componentes físicos através de serviços e protocolos que se integram facilmente em rede.

A respeito das experiências com o tAMARINO, pode-se observar diversos fatores nas opiniões dos participantes. No geral, a ferramenta obteve sucesso em todas as abordagens propostas e especificamente para o público não-técnico. No caso de um público com mais conhecimentos técnicos, o ambiente precisa responder aos quesitos levantados nas sugestões para garantir outros perfis de usuários.

9.1 Trabalhos futuros

Como trabalhos futuros, além das contribuições sugeridas de criar uma interface para desenvolver e editar módulos, geração de código-fonte para upload, a ferramenta precisa estar apta a realizar operações condicionais: loops, objetos matemáticos, conexões com outras mídias (áudio, vídeo, gráficos), tratamento dos circuitos para outras fontes, detalhamento dos componentes, comunidades de colaboradores, entre outros aspectos sugeridos.

As investigações e resultados deste trabalho abrem caminhos para realizar novas pesquisas a respeito do conceito de “invenção” de dispositivos interativos. A proposta é trazer questões para o “domínio da técnica” de tal modo que o usuário desenvolva experiências aplicadas para solucionar problemas reais em diversas áreas do conhecimento, possibilitando que, ao invés de focar apenas em robôs que resolvam por completo os problemas da vida dos humanos, desenvolver possíveis interações entre os próprios humanos, mediadas por dispositivos embarcados em dispositivos e objetos do cotidiano. Este foco implica na expansão do conceito da computação física para além das conexões entre sensores e atuadores puramente físico, mas através da conexão de tudo o que compõe os sistemas interativos e sensíveis, com possibilidades de comunicação por protocolos de rede de qualquer tipo de dado, seja para processamento gráfico, sonoro ou para controlar outros dispositivos espalhados pelo globo.

Do ponto de vista técnico, a criação de módulos de comunicação via OSC (*Open Sound Control*) e MIDI possibilita a conexão de usuários de diferentes áreas do conhecimento. Além disso, pretende-se trabalhar também com dispositivos que se comunicam via RFID (*Radio-Frequency Identification*), para que possam prover novas funcionalidades na perspectiva da Internet das coisas.

Na parte da prototipagem eletrônica, as opiniões e sugestões apontam para o desenvolvimento

de uma API para incorporar serviços desenvolvidos no ambiente Fritzing para a área de prototipagem do tAMARINO. Esta solução resolveria principalmente as questões de detalhamento dos componentes utilizados, exportação para outras fontes como PCB, e rearranjo do circuito de forma intuitiva, como nos softwares EDAs.

Por fim, o tAMARINO busca agregar uma massa de colaboradores dispostos a desenvolver novas funcionalidades, a medida em que a própria comunidade e a indústria dos *Makers* sugerem novos *shields* e componentes a cada instante.

10. Referências

ALLEN, P, STERLING, B.; BURDICK, A.: The New Ecology of Things.

[http://www.sketching10.com/presentations/philvanallen_sketching10.pdf]

ANDERSON, C.: Makers: The New Industrial Revolution. London: Random House Business Books Anderson, C., 2012.

BARBROOK, R.: Futuros imaginários: das máquinas pensantes à aldeia global. São Paulo. Editora Peirópolis, 2009.

BARRAGÁN, H.: Wiring: Prototyping physical interaction design. Interaction Design Institute, Ivrea, Italy, 2004.

[http://people.interactionivrea.org/h.barragan/thesis/thesis_low_res.pdf] Acessado em maio de 2013.

BEGINNER'S M.C.; SHAW, D.: Makey Makey: improvising tangible and nature-based user interfaces. TEI '12 Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction Pages 367-370. 2012

BREWER, E.A.: Lessons from Giant-Scale Services. IEEE Internet Computing 5(4): IEEE Educational Activities Department. pp. 46-55, 2001.

BROWN, T.; & WYATT, J.: Design Thinking for Social Innovation. Stanford Social Innovation Review, Winter, 30-35. 2010.

BUECHLEY, L.; EISENBERG, M.; CATCHEN, J.; CROCKETT, A.: The LilyPad Arduino: Using Computational Textiles to Investigate Engagement, Aesthetics, and Diversity in Computer Science Education. Craft Technology Group, Department of Computer Science University of Colorado at Boulder Boulder, CO 80309 USA, 2008.

BURSTEIN, F.; LINGER, H.: A task-based framework for supporting knowledge work practices. - Third European Conference on Knowledge Management. 2002

COHEN, M.A.; J. ELIASHBERG, T.H. Ho: New Product Development: The Performance

and Time-to-Market Tradeoff. *Management Sci.* 42(2) 173-186. 1996

CONRADI, B.; HOMMER, M.; KOWALSKI, R.: From Digital to Physical: Learning Physical Computing on Interactive Surfaces. *ITS '10 ACM International Conference on Interactive Tabletops and Surfaces* Pages 249-250. 2010.

COOPER, A; REIMANN, R; CRONIN, D.: *About Face 3: The Essentials of Interaction Design*. 2007

CRAWFORD, C.: *The Art of Interactive Design: A Euphonious and Illuminating Guide to Building Successful Software*, June 2002.

DYM, C.; AGOGINO, A.; ERIS, O; FREY, D.; LEIFER, L.: *Engineering design thinking, teaching, and learning*. American Society for Engineering Education. 2005

NYRHINEN, F.; MIKKONEN, T.: Web Browser as a Uniform Application Platform: How Far Are We? In *Proceedings of the 35th Euromicro Conference on Software Engineering and Advanced Applications (SEAA'2009, Patras, Greece, August 27-29, 2009)*, IEEE Computer Society, pp.578-584.

GALBRAITH, M.: *Embedded Systems for Computational Garment Design*. Thesis (S.M.) Massachusetts Institute of Technology, School of Architecture and Planning, Program in Media Arts and Sciences, 2003. [<http://dspace.mit.edu/handle/1721.1/61136>] Acessado em maio de 2013.

GREENBERG, S.; FITCHETT, C.: *Phidgets: Easy development of physical interfaces through physical widgets*. *UIST 2001 Symposium on User Interface Software and Technology*, November 11-14, Orlando, Florida.

GREENWOLD, S.: *Spatial Computing*, Master's thesis, MIT. 2003. [<http://pubs.media.mit.edu/pubs/papers/SpatialComputing.pdf>] acessado em Maio de 2013

GUINARD, D.; TRIFA, V.: "Towards the Web of Things: Web Mashups for Embedded Devices," in *2nd Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009)*, Madrid, Spain, Abril 2009.

GUZDIAL, M.: Programming Environments for Novices. MIT Press. College of Computing, Georgia Institute of Technology, 2002.

HAREL, D.: Statecharts: A visual formalism for complex systems. Science of Computer Programming 8. 231-274. North-Holland. 1987.

HARTMANN, B.; DOORLEY, S.; KLEMMER, S.: Hacking, Mashing, Gluing: Understanding Opportunistic Design. Pervasive Computing, IEEE (Volume:7 , Issue: 3). 2008.

HARTMANN, B.; KLEMMER, S.; BERNSTEIN, M.; ABDULLA, L.; BURR, B.; ROBINSON, A.; GEE, J.: Reflective Physical Prototyping through Integrated Design, Test, and Analysis. UIST '06 Proceedings of the 19th annual ACM symposium on User interface software and technology Pages 299-308. 2006.

HARTMANN, B.; KLEMMER, S.R.; BERNSTEIN, M.; MEHTA, N.: d.tools: Visually Prototyping Physical UIs through Statecharts. Extended Abstracts of UIST. 2005.

IDEO.: Human-Centered Design Toolkit: An Open-Source Toolkit to Inspire New Solutions in the Developing World. 2011.

IGOE, T; O'SULLIVAN, D.: Physical Computing: Sensing and Controlling the Physical World with Computers. Premier Press. ISBN 1-59200-346-X. 2004.

JÁCOME, J.: Sistemas Interativos de Tempo Real para Processamento Audiovisual Integrado. Dissertação de Mestrado. Centro de Informática, UFPE. 2007.

JAMIESON, P.: Arduino for Teaching Embedded Systems. Are Computer Scientists and Engineering Educators Missing the Boat? Miami University, Oxford, OH, 45056, 2010.

Johnson, S: Interface Culture: How New Technology Transforms the Way We Create and Communicate. New York, NY: Harper Collins Publishers Inc., 1997. 264 pp. ISBN 0-06-251482-2.

KATO, Y.: Splish: A Visual Programming Environment for Arduino to Accelerate Physical

Computing Experiences. Eighth International Conference on Creating, Connecting and Collaborating through Computing. 2010.

KAUČIČ, B.; ASIČ, T.: Improving Introductory Programming with Scratch? MIPRO 2011, May 23-27, 2011.

KNÖRIG, A.; WETTACH, R.; COHEN, J.: Fritzing – A tool for advancing electronic prototyping for designers. Proceedings of the 3rd International Conference on Tangible and Embedded Interaction Pages 351-358. 2009.

KRASNER, G.E.; POPE, S.T.: A description of the model-view-controller user interface paradigm in the smalltalk-80 system. Journal of Object Oriented Programming, 1(3):26–49, 1988.

LIKERT, R.: A technique for the measurement of attitudes. Archives of psychology. 1932.

LIN, H.T; KUO, T.H.: Teaching programming technique with edutainment robot construction, in 2nd International Conference on Education Technology and Computer (ICETC), Shanghai, China, pp. 226-229, 2010.

LOVELL, E.; BUECHLEY, L.: An E-Sewing Tutorial for DIY Learning. IDC '10 Proceedings of the 9th International Conference on Interaction Design and Children Pages 230-233. 2010.

MARTIN, F.G.: The Mini Board Technical Reference. 1998 - sunsite.univie.ac.at.

MARTIN, F.G.: The Handy Board Technical Reference. Handy Board Documentation, 2000

MELLIS, D.; BANZI, M.; CUARTIELLES, M.; IGOE, T.: Arduino: An open electronics prototyping platform. alt. chi section of the CHI conference in San Jose. 2007.

MILLNER, A.; BAAFI, E.: Modkit: blending and extending approachable platforms for creating computer programs and interactive objects. Proceedings of the 10th International Conference on Interaction Design and Children. Pages 250-253. 2011.

MYERS, B.A.: State of the art in user interface software tools. In Baecker, R., Grudin, J. Buxton, W. and Greenberg, S. Reading in Human Computer Interaction: Towards the Year

2000. Morgan Kaufmann, 1995.

MYERS, BA.: Visual Programming, Programming by Example, and Program Visualization: A Taxonomy, in ACM Conference on Human Factors in Computing Systems (SIGCHI'86), Boston, MA, USA, pp. 59-66, 1986.

NEDIC, Z.; MACHOTKA, J.; NAFALSKI, A.: Remote Laboratories versus Virtual and Real Laboratories. Frontiers in Education. FIE 2003 33rd Annual(Volume:1) 2003.

NORMAN, D.: The Invisible computer : why good products can fail, the personal computer is so complex, and information appliances are the solution. Cambridge, MA: MIT Press, xii, 302 p. ISBN 0262640414 (broch.). 1998.

OLIPHANT, F.: Meemoo: Hackable Web App Framework. Master of Arts, New Media. Aalto University. School of Arts, Design and Architecture. 2012.

OLIVER, J.P.; Haim, F.: Lab at Home: Hardware Kits for a Digital Design Lab. Education, IEEE Transactions on (Volume:52 , Issue: 1). 2009.

PAPADIMATOS, P.: Physical computing: using everyday objects as communication tools. Masters thesis, UCL (University College London). 2005.
http://discovery.ucl.ac.uk/2036/1/physical_computing_Thesis_MediumRes1.pdf] acessado em Maio de 2013.

PAPERT, S.: Mindstorms: Children, Computers, and Powerful Ideas. Basic Books, New York, 1980.

PUCKETTE, M.S.: Pure-data: another integrated computer music enviromental. 1997.

REAS, C.; FRY, B.: Processing, a Programming Handbook for Visual Designers and Artists. Cambridge: MIT Press, 2007. [<http://processing.org/learning/books/>] Acessado em maio de 2013.

RESNICK, M. ; MARTIN, F. ; SARGENT, R. ; SILVERMAN, B.:Programmable bricks: Toys to think with. IBM Systems Journal (Volume:35 , Issue: 3.4) pp 443 – 452. 1996.

RESNICK, M.; MALONEY, J.; MOROY-HERNÁNDEZ, A.: Scratch: programming for all. Communications of the ACM, 52(11), pp 60-67. 2009.

RESNICK, M.: Sowing the Seeds for a More Creative Society. Learning and Leading with Technology. s.l.: International Society for Technology in Education. 2007
[<http://web.media.mit.edu/~mres/papers/Learning-Leading.pdf>] Acessado em maio de 2013.

ROGERS, M.: The GitHub Revolution: We're All in Open Source Now (wired.com). Wired Magazine, March 7, 2013.

SARIK, J.; KYMISSIS, I.: Lab kits using the arduino prototyping platform, in Frontiers in Education Conference (FIE/IEEE) pp. T3C-1 –T3C-5. 2010.

SIPITAKIAT, A. ; BLIKSTEIN, P. ; CAVALLO, D.P.: The GoGo Board: Moving towards highly available computational tools in learning environments. Massachusetts Institute of Technology. 2002.

SIVEK, S.: We Need a Showing of All Hands: Technological Utopianism in MAKE Magazine. Linfield College. Journal of Communication Inquiry, 2011.

SNAVAES, D.: Understanding Interactivity: Steps to a Phenomenology of Human-computer Interaction. Phd Thesis, Trondheim, 2000.[<http://www.idi.ntnu.no/~dags/interactivity.pdf>]
Acessado em maio de 2013.

SOLDATOS, J.; DIMAKIS, N.; STAMATIS, K.; POLYMENAKOS, L.: A breadboard architecture for pervasive context-aware services in smart spaces: middleware components and prototype applications. Personal and Ubiquitous Computing. Volume 11, Issue 3, pp 193-212. March 2007.

SOLOWAY, E.; GUZDIAL, M.; HAY, KE.: Learner-centered design: the challenge for HCI in the 21st century. Magazine interactions. Volume 1 Issue 2, pp 36-48 Abril 1994.

SOMEREN, M.; BARNARD, Y.; SANDBERG, J.: The think aloud method: A practical guide to modelling cognitive processes. 1994.

- STAINER, C.: Firmata: Towards making microcontrollers act like extensions of the computer. New Interfaces for Musical Expression. 2009. [https://lagunak.gisa-elkartea.org/attachments/download/74/firmata_porque.pdf] Acessado em maio de 2013.
- STALLMAN, R.: The GNU Manifesto. Free Software Foundation, Inc. 1985. [https://facwiki.cs.byu.edu/OSSResearch/images/7/70/Stallman_TheGNUManifesto.pdf] Acessado em maio de 2013.
- STANKOVIC, J.; VIRGINIA, U.; CHARLOTTESVILLE, V. ; INSUP, L. ; MOK, A.; RAJKUMAR, R.: Opportunities and Obligations for Physical Computing Systems. P 23 - 31 2005.
- WARFEL, T.: Prototyping: A Practitioner's Guide. Ro senfeld Media. ISBN: 1933820217 | 197 pages. 2009.
- WEBER, S.: The success of open-source. Cambridge Univ Press. 2004.
- WILSON, S. ; VERPLANK, B. ; GUREVICH, M. ; STANG, P.: Microcontrollers in Music HCI Instruction. New Interfaces for Musical. 2003.
- WING, J: Computational Thinking. Communications of the ACM, pp. 33-35. March 2006. [http://www-cgi.cs.cmu.edu/afs/cs/usr/wing/www/CT_at_CMU.pdf] Acessado em maio de 2013.
- YAN, X.; GU, P.: A review of rapid prototyping technologies and systems. Computer-Aided Design, Elsevier, 1996.
- ZAEFFERER, J.; ONKEN, N.: Robotic Javascript. Jsconf 2010. <http://www.slideshare.net/nonken/robotic-javascript> Acessado em Junho de 2013
- ZHANG, D.Q; ZHANG, K.: On the Design of A Generic Visual Programming Environment, in IEEE Symposium on Visual Languages, Halifax, New Scotia, Canada, pp. 88-89, 1998.