Universidade Federal Rural do Semi-Árido
Universidade do Estado do Rio Grande do Norte

Programa de Pós-Graduação em Ciência da Computação

Dissertação de Mestrado

Interface Tangível do Usuário como Dispositivo de Entrada e Saída

Mestrando
Rodrigo Fontes Maia

Orientador
D. Sc. Angélica Félix de Castro
Co-Orientador
D. Sc. Sílvio Roberto Fernandes de Araújo

Mossoró - RN
Fevereiro - 2016

Universidade Federal Rural do Semi-Árido
Universidade do Estado do Rio Grande do Norte

Programa de Pós-Graduação em Ciência da Computação

Dissertação de Mestrado

Interface Tangível do Usuário como Dispositivo de Entrada e Saída

Mestrando
Rodrigo Fontes Maia

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação para a obtenção do título de Mestre em Ciência da Computação
Orientador(a): D. Sc. Angélica Félix de Castro

Mossoró - RN
Fevereiro - 2016

Mi Maia, Rodrigo Fontes.
Interface Tangível do Usuário como Dispositivo de Entrada e Saída / Rodrigo Fontes Maia. - 2016.
105 f. : il.

Orientador: Angélica Félix de Castro.
Coorientador: Sílvio Roberto Fernandes de Araújo.
Dissertação (Mestrado) - Universidade Federal Rural do Semi-árido, Programa de Pós-graduação em Ciência da Computação, 2016.

1. Interface Tangível do Usuário. 2. Dispositivo de Entrada e Saída. 3. Sistemas Embarcados . 4. Biblioteca de Software. I. Castro, Angélica Félix de, orient. II. Araújo, Sílvio Roberto Fernandes de, co-orient. III. Título.

RODRIGO FONTES MAIA

INTERFACE TANGÍVEL DO USUÁRIO COMO DISPOSITIVO DE ENTRADA E SAÍDA.

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação para a obtenção do título de Mestre em Ciência da Computação.

APROVADA EM: 29 / 07 / 2016.

Angélica Félix de Castro.
Profa. Dra. Angélica Félix de Castro
Presidente e Orientadora

Silvio R. Fernandes de Araújo
Prof. Dr. Sílvio Roberto Fernandes de Araújo
Coorientador

Paulo Gabriel Gadelha Queiroz
Prof. Dr. Paulo Gabriel Gadelha Queiroz
Membro Interno - UFERSA

Monica Magalhães Pereira
Profa. Dra. Monica Magalhães Pereira
Membro Externo - UFRN

Agradecimentos

Agradeço demais aos dois professores que me “adotaram” nesse mestrado, meus orientadores Angélica e Sílvio, obrigado pelo voto de confiança, a minha mãe Ibanez, por ser uma das pessoas mais especiais que conheço, e aos meus colegas Alexandro, Álamo, Juliene, Dênis e Felipe, pelo suporte e tardes de trabalhos e conversas, e agradeço também à CAPES, pelo financiamento deste trabalho.

Resumo

O presente trabalho se insere no contexto de TUIs (Tangible User Interfaces), definindo e apresentando um projeto de interface tangível de propósito geral que funciona como dispositivo de entrada e saída; que se trata de uma plataforma com pinos que se movem, mudam de cor e são capazes de captar o toque do usuário. Neste trabalho também apresenta-se uma biblioteca de software com um conjunto de funções que permitem a programadores utilizar o dispositivo nas mais diversas aplicações, de forma simplificada, ou seja, desenvolver os programas a serem executados em conjunto com o dispositivo. Ao longo do projeto vários conceitos são apresentados com o intuito de contextualizar o leitor sobre os conceitos envolvidos na construção do hardware, citando motores, placas de prototipação e microcontroladores. Referências a biblioteca de software e API foram abordadas, abrangendo a área da programação do dispositivo. O que são TUIs e a maneira como se avalia interfaces, de acordo com o IHC (Interação Homem-Computador) foram descritas de forma que se pôde fazer uma avaliação do protótipo. Aplicações foram implementadas para o dispositivo, exemplificando não só as suas funcionalidades como também mostrando o uso da biblioteca de software disponível. Questionários aplicados a usuários, e analisados de acordo com o o modelo TAM (*Technology Acceptance Model*) (DAVIS; BAGOZZI; WARSHAW, 1989), resultaram em 90% dos usuários totalmente satisfeitos com a facilidade de uso, e mais de 80% totalmente satisfeitos com a percepção de utilidade do protótipo, o que mostrou uma boa aceitação do dispositivo.

Palavras-chaves: Interface Tangível do Usuário, Dispositivo de Entrada e Saída, Sistemas Embarcados, Biblioteca de Software.

Sumário

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	*Application Programming Interface* (Interface de programação de aplicativos)
ARM	*Advanced RISC Machine* (Máquina RISC avançada)
DC	*Direct Current* (Corrente contínua)
ES	Entrada e Saída
FPGA	*Field Programmable Gate Array* (Arranjo de Portas Programável em Campo)
GNOME	Acrônimo para *GNU Network Object Model Environment*
GPS	*Global Positioning System* (Sistema de Posicionamento Global)
GTK	*GNOME Tool Kit* (Kit de ferramentas GNOME)
GUI	*Graphical User Interface* (Interface de usuário gráfica)
HDL	*Hardware Description Language* (Linguagem de Descrição de Hardware)
IDE	*Integrated Development Environment* (Ambiente de desenvolvimento integrado)
IHC	Interação Humano-Computador
IoT	*Internet of Things* (Internet das Coisas)
IR	Infra-Red (Infravermelho)
ISA	*Industry Standard Architecture*
IT	*Information Technology*
KDS	*Kinetis Design Studio*
KSDK	*Kinetis Software Development Kit*
LED	*Light Emitting Diode* (Diodo emissor de luz)
MIT	*Massachusetts Institute of Technology* (Instituto de Tecnologia de Massachusetts)

OpenSDA	*Open-Standard Serial and Debug Adapter* (Padrão aberto serial e adaptador de depuração)
SO	Sistema Operacional
PC	*Personal Computer* (Computador pessoal)
RA	Realidade aumentada
RFID	*Radio Frequency Identification* (Identificação por rádio frequência)
RGB	*Red, Green, Blue* (Vermelho, verde, azul)
RISC	*Reduced Instruction Set Computer* (Computador com conjunto reduzido de instruções)
RV	Realidade virtual
TAR	*Tangible Augmented Reality* (Realidade aumentada tangível)
TUI	*Tangible User Interface* (Interface de usuário tangível)
UML	*Unified Modeling Language* (Linguagem de Modelagem Unificada)

1 Introdução

Há algumas décadas, não existia preocupação com a acessibilidade no desenvolvimento de programas, uma vez que esses eram escritos e direcionados para o uso dos programadores, que por sua vez, compreendiam como usar o computador de maneira mais direta, utilizando conhecimentos de programação.

Com a popularização dos computadores, o conceito de interface ganhou maior importância, pois era necessário haver uma intermediação entre a máquina e o usuário comum (OLIVEIRA, 2014). Assim surgiram as interfaces gráficas, que se tornaram primordiais, pois facilitam a vida dos usuários abstraindo funções complexas dos sistemas, que não seriam acessíveis a pessoas sem o devido treinamento.

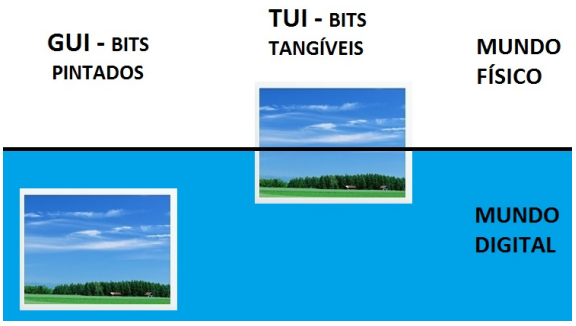
Surgiram então interfaces amigáveis, que exigiam menos esforço, por parte dos usuários, para se aprender a interagir com um computador, e consequentemente facilita o trabalho dos usuários.

Os computadores passaram a fazer parte do ambiente de convivência dos seres humanos, e à medida que evoluem para modelos cada vez menores e mais interligados nas atividades do cotidiano, tornam realidade o conceito de computação ubíqua, como previa Weiser (1999): os computadores se integram fortemente ao meio ambiente, fazendo sua manipulação tão usual, que a naturalidade da interação torna-os quase imperceptíveis, em consequência da psicologia humana.

Seguindo a linha do desenvolvimento de novas interfaces surgiram as TUIS (*Tangible User Interfaces*), que se baseiam na incorporação física da informação digital, e vice-versa, a fim de ir além do paradigma dominante atual de "Bits Pintados" ou GUIs (*Graphical User Interfaces*) (MIT, 2014). Os "bits pintados" ou GUIs são representações de um plano cartesiano em uma tela, onde cada pixel é uma informação de cor em uma coordenada x,y. Nessa forma de representação o usuário não consegue interagir diretamente com a informação, isso deve ser feito por meio de um dispositivo de entrada, cujo processamento resultará em outra representação no plano. Mesmo que seja usada uma tela sensível ao toque, a leitura da entrada (o toque do usuário) não é feita diretamente pela informação, e sim por um mecanismo que interpreta o comando, o processa e atualiza o conteúdo mostrado na tela.

A ideia das TUIs é justamente permitir que o usuário entre em contado direto com a informação, fazendo-o capaz de manipular os dados com as próprias mãos. Na Figura 1, está representada a imersão do mundo digital no mundo físico. A analogia feita aqui é que a informação (a imagem de uma paisagem), antes acessível apenas no mundo digital, agora está exposta no mundo real, podendo ser alterada por este.

Figura 1: Imersão Digital no Mundo Físico



Fonte: Autoria Própria.

As interfaces tangíveis constituem uma abordagem inovadora que propõem a utilização da computação de outras formas além dos computadores pessoais, em benefício das mais diversas áreas. Embutindo elementos computacionais em materiais concretos, cria-se um novo grupo de recursos que une as vantagens da manipulação física à interação e multimídia providas pela tecnologia. Enriquecendo os materiais concretos, os recursos computacionais podem ajudar a estimular e trabalhar diversos sentidos (visão, audição, tato) e inclusive promover uma maior inclusão de portadores de deficiências (FALCAO; GOMES, 2007).

As TUIs podem unir as propriedades de entrada e saída das interfaces, e estão associadas a significados e objetivos específicos, dependendo de seu intuito, abrangendo a ideia de que, o que será manipulado será a própria informação, ao contrário de um periférico como o mouse, que é um dispositivo de entrada sem nenhum significado próprio (FALCAO; GOMES, 2007), e que não representa nenhuma informação. A manipulação da informação digital é feita por meio dos objetos reais que possuem uma representação virtual, permitindo a interação e a imersão do usuário de forma mais significativa com a computação.

1.1 Motivação

Usando tecnologias emergentes da era ubíqua, tais como realidade virtual, realidade aumentada, telas em 3D, telas sensíveis ao toque, internet das coisas, aplicativos educacionais e jogos que capturam movimentos, usuários recorrentes ou casuais podem facilmente e intuitivamente interagir com um computador ou com outros usuários. Essa interação pode ser até mesmo por meio de objetos virtuais ou reais, usando diversos dispositivos, que demandam novas formas de interfaces.

Também é esperado dessas interfaces, que sejam facilmente personalizadas e contextualizadas para usuários ou grupos de interesses específicos. Particularmente, conteúdos multimídias baseados em áudio e vídeo ainda são os principais meios de comunicação para

a difusão de conhecimento e entretenimento. No entanto, há uma procura crescente em usar modelos virtuais 3D nestas aplicações (LEE; SEO; RHEE, 2011), o que implica no desenvolvimento contínuo dessas tecnologias.

Qualquer objeto pode ser usado como entrada para um sistema tangível, e é comum que o formato desses objetos tenha alguma relação com a sua funcionalidade. Os cartões RFID (*Radio Frequency Identification*), por exemplo, podem ser instalados em uma chave e servir para destrancar portas eletronicamente. Câmeras que reconhecem padrões de objetos podem identificar o tema de uma festa, e fazer o aparelho de som tocar a música apropriada.

Em aplicações, nas quais haja interação, o feedback ativo (confirmação de que um usuário realizou alguma ação com sucesso) pode ser considerado fraco em ambientes de realidade aumentada ou virtual, em que apesar de ver e interagir com os objetos, o usuário não é capaz de senti-los. Isso leva a necessidade de uma interface física responsiva, que torne as ações dos usuários mais imersivas.

Em um sistema de simulação de voo, por exemplo, um simples teclado ou joystick de PC (*Personal Computer*), seria capaz de interagir e controlar perfeitamente um avião, no entanto, não haveria praticidade nesse controle. O treinamento de um piloto, por exemplo, terá bem mais resultado se for feito em uma cabine de simulação que represente todas as possibilidades reais de um voo. O mesmo pode ser dito para máquinas que realizam cirurgias à distância, que demandam uma interface específica que facilite o trabalho do médico, colocando-o em uma situação semelhante a uma sala de cirurgia, como mostrado na Figura 2.

Figura 2: Médico Realizando Cirurgia Remota.

Fonte: Galileu (2013).

Em ambientes tangíveis, objetos físicos representam a informação digital e são também utilizados para manipulação dessa informação, de forma que os próprios objetos podem servir como periférico de entrada ou de saída de dados. O usuário modifica a representação de dados, que é convertida em bits, processada e devolvida como uma nova representação mostrada na própria interface seja ela qual for, e não necessariamente em uma tela, por exemplo.

Interfaces tangíveis também se destacam pela sua capacidade de permitir uma significativa colaboração entre seus usuários, em que essa capacidade dependerá do formato e função da interface, mas uma vez que o objetivo seja tirar proveito de trabalhos que exijam colaboração, pode-se conseguir bons resultados.

Vale a pena citar um trabalho desenvolvido pelo Grupo de Tecnologia de Música da Universidade Pompeu Fabra, Barcelona. Lá foi criada a *Reactable* (REACTABLE, 2016) , um instrumento musical eletrônico colaborativo com uma interface tangível, no qual vários usuários simultâneos compartilham o controle, movendo e girando objetos físicos na superfície de uma mesa circular brilhante, como mostrado na Figura 3. Manipulando esses objetos, que representam componentes clássicas de um sintetizador modular, os usuários podem criar tipologias complexas e dinâmicas de som.

Figura 3: Uso da Reactable.

Fonte: Reactable (2016).

Ao transformar objetos do cotidiano em periféricos de entrada/saída para computadores, a utilização de sistemas torna-se intuitiva, dinâmica, natural e sem a necessidade de treinamentos especiais, fazendo a computação aproximar-se da realidade de uma maneira amigável.

Em sua maioria, as TUIs são de aplicação específica, de forma que há uma lacuna de interfaces tangíveis de propósito geral, que possam ser usadas nas mais diversas aplicações, isto torna a área da computação tangível atrativa à pesquisa e exploração de novas possibilidades.

1.2 Objetivo

Neste trabalho apresenta-se a definição e implementação de um dispositivo de entrada e saída de propósito geral, que teve como base em seu desenvolvimento, as características das interfaces tangíveis. O dispositivo em questão é capaz de executar variadas aplicações, que podem ser desenvolvidas usando a API (*Application Programming Interfaces* ou Interface de Programação da Aplicação) do periférico, também apresentada ao longo deste trabalho, e escrita visando facilitar o desenvolvimento de aplicações para o dispositivo, proporcionando aos usuários uma interface em linguagem de programação de alto nível.

O dispositivo constitui-se de uma malha de pinos luminosos, que pode simular um monitor 3D físico, sendo capaz de mostrar imagens variando as alturas e as cores de seus pinos, análogos a pixels, e se for considerado que cada um desses pinos, também é capaz de receber o toque físico de um usuário, pode-se considerar que cada pino seja chamado de "pixel tangível".

Para a validação deste trabalho, três softwares foram desenvolvidos, explorando as capacidades de se usar o dispositivo como monitor tangível. As aplicações mostram representações de gráficos em barras, letras e números binários, usando os pixels tangíveis do dispositivo para formar os caracteres em questão. É possível que o usuário mova os pinos e interaja com as aplicações, podendo formar caracteres válidos a serem identificados pelo dispositivo, no intuito de servir como entrada de dados.

Também é mostrado como usar as funções disponíveis na API do periférico, tomando como exemplo os softwares descritos, que foram testados por usuários, para que uma avaliação do desempenho do projeto fosse feita.

De maneira geral, o principal intuito da pesquisa surgiu pela necessidade de se buscar preencher a lacuna entre o mundo digital e o tangível, criando uma interface de propósito geral, capaz de executar diversas aplicações, em prol da busca de novas maneiras de interação com sistemas computacionais e de apresentação de informações.

O objetivo do dispositivo proposto é exibir qualquer tipo de informação utilizando os pixels tangíveis. Como o dispositivo foi construído usando-se quatro pixels, as aplicações desenvolvidas até o momento se limitam a essa resolução, mesmo assim, pode-se dizer que a interface é de propósito geral ao inferir que, em vez de quatro, sejam inúmeros pixels, e comparar a plataforma a um monitor de PC, que é um dispostivo capaz de representar qualquer informação desenhando-a na tela.

A aplicabilidade do dispositivo como interface tangível, além da capacidade de representar informações, mostra-se quando os pixels podem ser tocados, ou seja, o usuário entra em contato direto com a informação que está sendo mostrada, e não utiliza-se de outro dispositivo para isto. O dispositivo ainda pode ser usado para aplicações direcionadas

a deficientes físicos, principalmente deficientes visuais, que podem identificar caracteres a partir das alturas dos pixels usando somente as mãos, e interagir com a informação ao mesmo tempo.

1.3 Organização do Trabalho

Este trabalho está organizado da seguinte forma: No Capítulo 2, apresentam-se os fundamentos teóricos sobre conceitos necessários a criação de um dispositivo de entrada e saída, além da descrição de algumas caracteristicas sobre microcontroladores e TUIs, que são consideradas no dispositivo em questão; no Capítulo 3, é mostrada uma compilação de textos que apresentam trabalhos semelhantes na área das TUIs e servem de inspiração para este projeto; no Capítulo 4, são descritas as partes de Hardware e Software do dispositivo, bem como a metodologia usada na avaliação destes, feita por usuários; no Capítulo 5, descrevem-se as aplicações implementadas para o dispositivo e mostram-se os resultados dos testes feitos com usuários; e no capítulo 6, apresentam-se as conclucsões do texto e os trabalhos futuros.

2 Referencial Teórico

Neste capítulo são descritos conceitos que serviram de referência para a construção do dispositivo tangível em questão, referenciando interfaces tangíveis; a Interação Humano Computador; metodologia de desenvolvimento de sistemas embarcados e bibliotecas de software. Também é mostrado alguns materiais usados na implementação do hardware, tais como microcontroladores, neste caso, placas de prototipagem que contém um microcontrolador; motores e placas controladoras de motores.

2.1 TUIs (*Tangible User Interfaces*)

Tangível é uma qualidade das coisas que podem ser tocadas. Atualmente, o termo Interface Tangível é usado para conceituar periféricos de entrada, saída ou ambos que são "palpáveis" ou "tocáveis", e também podem possuir um significado associado ao que representa, fazendo com que as interações com sistemas computacionais se tornem mais "realistas", uma vez que considera-se o uso de um objeto "real" e não digital, para manipular informação digital.

Os equipamentos tangíveis se baseiam em três conceitos: (1) a interatividade através do contato físico, contato esse que se reflete em respostas no meio digital; (2) a praticidade em aplicações do dia-a-dia que possam facilitar a vida e que, cada vez mais, ganham espaço na pesquisa e no mercado; e (3) a colaboração, que torna possível realizar cirurgias à longa distância, por exemplo (STERMAN, 2014).

Embora dispositivos que usem da interface padrão GUI (como teclados, mouses e telas) também sejam físicos, a representação física em uma TUI fornece uma importante distinção. A forma de manipulação física de um objeto usado como interface permite um maior controle e manipulação de seus parâmetros, o que afeta diretamente a simulação digital subjacente (ISHII, 2008).

Uma interface tangível, localizada no "mundo real" (mundo físico, existente), geralmente possui uma representação digital, localizada no "mundo digital" (uma rede ou PC), as modificações que ocorrem em um dos lados, mundo real ou mundo digital, estão conectadas e reagem de forma parecida. Esse é o mecanismo que permite o controle de informação digital por meio de interfaces no mundo real.

Em outras palavras, as representações físicas de uma plataforma tangível estão computacionalmente ligadas à informação digital, por exemplo: pode-se ter a informação de um gráfico salva em um arquivo, e é possível usar essa informação para plotar um gráfico em uma mesa 3D com interface tangível, fazendo o gráfico na mesa também tornar-se

uma chave de entrada, e quando tocado, as modificações resultantes desse toque serão transmitidas para a informação guardada no arquivo. Da mesma forma, um arquivo com as informações de um gráfico pode ser criado do zero, manipulando-se apenas a interface tangível, logo, os dados resultantes dessa manipulação irão constituir nova informação digital.

No entanto, algumas TUIs possuem formato especifico e não tem a capacidade de mudar as formas de representações tangíveis durante a interação com o usuário. Nesses casos, os usuários devem usar um conjunto finito predefinido de objetos que servem como TUIs, e alterar apenas a relação espacial entre eles, não a sua forma individual. Exemplo: mudar a posição de uma xícara em uma cozinha pode habilitar as funções existentes de cozinha inteligente que possam haver naquele ambiente, mas não é possível mudar o formato da xícara durante esse processo. Mesmo assim, as TUIs tiram proveito da inerente habilidade humana de compreender e manipular formas físicas, enquanto proporciona poder de processamento proveniente de computadores (ISHII, 2008).

As plataformas tangíveis apresentam interação imediata, reagindo a entrada dos usuários em tempo real. Usando o exemplo já citado do gráfico em 3D, pode-se supor que ele seja capaz de responder uma interação direta do usuário, mudando de cor ao ser tocado ou invertendo seus valores apresentados anteriormente. Isso gera um grau de imersão maior com usuários em comparação a interfaces puramente digitais. Além do mais, há o fato de que as TUIs são projetadas para se parecerem com objetos relacionados com sua função, o que torna seu uso intuitivo, principalmente para usuários que nunca as usaram.

Em geral, TUIs dão forma física para informação digital e computação, facilitando a manipulação direta de bits. O objetivo é capacitar colaboração, aprendizagem, e tomada de decisão através de tecnologia digital aproveitando a capacidade humana de agarrar e manipular objetos físicos e materiais (ISHII, 2008).

A relação interativa entre esses elementos torna a experiência com os ambientes virtuais mais real e convincente para o usuário, que pode fazer uso dos cinco sentidos do ser humano, fato que pode ser considerado inclusive para aplicações voltadas a ajuda de deficientes físicos, além de enriquecer a quantidade de recursos disponíveis para se interagir com um computador.

2.2 Interação Humano Computador

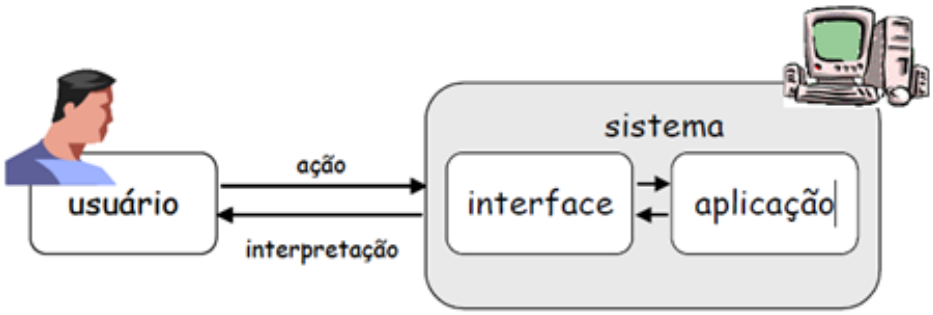
Interação Humano Computador ou IHC é responsável por investigar o "projeto (design), avaliação e implementação de sistemas computacionais interativos para uso humano, juntamente com os fenômenos associados a este uso" (HEWETT et al., 2003). Tendo em vista o conceito generalizado, é natural que IHC seja usado para melhorar o desenvolvimento de sistemas computacionais, assim como suas interfaces, logo se entende

a necessidade do uso de IHC para validar o projeto da TUI que é tratado neste trabalho.

Além de considerar que o conceito de interface tem se desenvolvido na mesma proporção em que ocorre o desenvolvimento da tecnologia, pode-se dizer que ele acompanha também o desenvolvimento da natureza humana. O design atual de sistemas reflete e é influenciado pelo conhecimento cientifico sobre a natureza humana, a interface computacional era entendida apenas como uma maneira do humano comunicar-se com um computador, mas a evolução do conceito de interface levou a inclusão de aspectos cognitivos e emocionais de usuários (ROCHA; BARANAUSKAS, 2003), facilitando a interação humano computador.

Os estudos relativos à avaliação de IHC buscam qualificar um projeto de interface, baseando-se na interpretação do usuário das respostas provenientes de sistemas interativos. Neste processo, o usuário interage com um sistema, fazendo o uso das funcionalidades disponíveis e observando as reações, que podem ser desde um simples "bip", som indicando que houve interação, até operações complexas que alteram o estado do mundo. Uma ilustração das ações que o usuário realiza usando a interface de um sistema, e suas interpretações das respostas transmitidas pelo sistema por meio da interface pode ser visto na Figura 4 (PRATES; BARBOSA, 2003).

Figura 4: O Processo de Interação Humano-Computador.



Fonte: Prates e Barbosa (2003)

Os objetivos do IHC se resumem em desenvolver ou melhorar a utilidade, efetividade e usabilidade de sistemas, auxiliando sua produção. Jakob Nielsen denomina estes objetivos em um conceito mais amplo: aceitabilidade de um sistema, que é a combinação da aceitabilidade social, ou seja, a visão das pessoas sobre o sistema e a aceitabilidade prática, que trata de parâmetros como custo, confiabilidade, compatibilidade com sistemas existentes, entre outros (ROCHA; BARANAUSKAS, 2003).

Ainda segundo Nielsen, "a usabilidade é um atributo de qualidade que avalia quão fácil uma interface é de usar" (REBELO, 2009) e está distribuída a diversos elementos, sendo tradicionalmente associada a alguns como:

Facilidade de aprendizagem: o sistema deve ser fácil de assimilar pelo utilizador,

para que este possa começar a trabalhar rapidamente;

Eficiência: o sistema deve ser eficiente para que o utilizador, depois de o saber usar, possa atingir uma boa produtividade;

Facilidade de memorização: o sistema deve ser facilmente memorizado, para que depois de algum tempo sem o utilizar, o utilizador se recorde como usá-lo;

Segurança: o sistema deve prever erros, evitar que os utilizadores os cometam e, se o cometerem, permitir fácil recuperação ao estado anterior.

Satisfação: o sistema deve ser usado de uma forma agradável, para que os utilizadores fiquem satisfeitos com a sua utilização.

A usabilidade possui medidas objetivas e subjetivas para avaliar a interação dos usuários com um sistema proposto. As medidas objetivas provêm medição de tempo, velocidade ou ocorrência de eventos particulares, tipicamente relacionadas à execução de testes da interface com um usuário. As medidas subjetivas se direcionam mais para as impressões pessoais, que os usuários tiveram baseadas em suas próprias preferências, depois de interagir com a interface avaliada (FILARDI; TRAINA, 2008).

De acordo com Moran, "a interface de usuário deve ser entendida como sendo a parte de um sistema computacional com a qual uma pessoa entra em contato — física, perceptiva ou conceitualmente" (PRATES; BARBOSA, 2003). Para que se faça uma medição desse entendimento, o IHC utiliza-se da usabilidade, podendo então chegar a um grau de satisfação, do usuário, que possa justificar o uso de uma interface.

A necessidade de usar IHC para avaliar uma interface parte da premissa de obtenção do melhor resultado possível de um projeto, principalmente para um domínio especifico de aplicação. Independentemente da área que abrangida, seja ela uma GUI tradicional ou uma TUI, existe uma complexidade que precisa ser descrita, para que o projeto tenha prosseguimento.

Seguindo adiante com o desenvolvimento de uma interface de hardware, há de se considerar também que o sistema seja planejado como um sistema embarcado, para isso é necessário adotar-se uma metodologia de desenvolvimento de sistemas embarcados, que será descrita na seção a seguir.

2.3 Metodologia de Desenvolvimento de Sistemas Embarcados

Usualmente, o projeto de um sistema embarcado inicia-se com a abstração do que será construído, baseado na funcionalidade que se quer alcançar, sem considerar materiais ou recursos (CARRO; WAGNER, 2003).

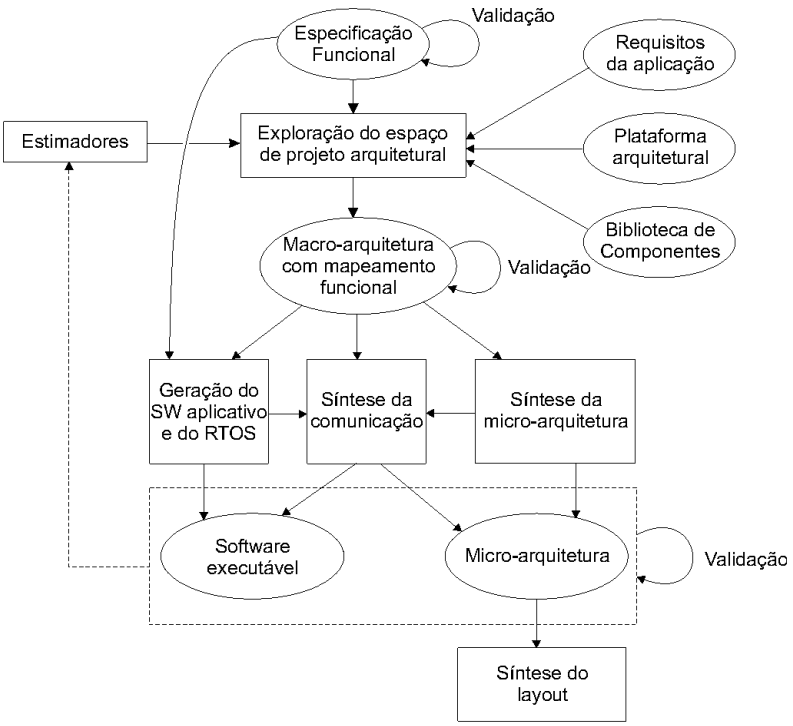
O projeto se inicia com uma especificação funcional, e deve-se atentar para que

a essa seja preferencialmente executável, para fins de validação. Em seguida, deve ser feita uma exploração do espaço de projeto arquitetural, a fim de que se encontre uma arquitetura compatível com a especificação inicial e atenda aos requisitos do projeto em termos de custo, desempenho, consumo de potência, área, etc.

O resultado final dessa etapa é uma macro arquitetura, que já deve mencionar quais tipos de processadores ou microcontroladores serão utilizados, além de outros componentes necessários (memórias, interfaces, blocos dedicados de hardware), e as interconexões feitas por meio de uma infraestrutura de comunicação (um ou mais barramentos). Um mapeamento estabelece o particionamento necessário de funções entre hardware) e software (CARRO; WAGNER, 2003).

A existência de estimadores é fundamental para que uma exploração das melhores opções a serem usadas seja feita, informando com precisão os valores de métricas do projeto, "esta etapa é usualmente simplificada pela escolha prévia de uma plataforma arquitetural conhecida e adequada ao domínio da aplicação" (CARRO; WAGNER, 2003). Considerando também a especificação do sistema, o software do sistema é então gerado, uma vez definidos os componentes de hardware da macro arquitetura, incluindo a infraestrutura de comunicação e os eventuais adaptadores, pode ser feita a síntese do hardware (CARRO; WAGNER, 2003). A Figura 5 mostra uma metodologia completa de projeto de um sistema eletrônico embarcado, embora na prática não existam ambientes comerciais que a implementem completamente.

Figura 5: Metodologia de Projeto.



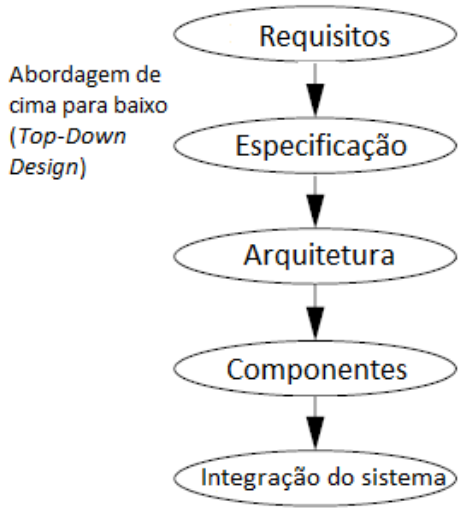
Fonte: Carro e Wagner (2003)

Para modelar um hardware, existe a disponibilidade de linguagens de descrição de hardware (HDL - *Hardware Description Language*), tais como VHDL, Verilog e System C. Também existem linguagens de descrição de arquiteturas como LISA e ArchC. No caso de um dispositivo a ser construído com uma placa de prototipagem, o uso de uma linguagem de descrição não se faz necessário por enquanto, por outro lado, é necessário decidir quais peças, motores, cabos e outros materiais a serem utilizados. A placa de prototipagem geralmente vem com portas USB incluídas, e uma linguagem própria de programação, o que facilita a escrita de aplicações e testes iniciais.

Há de se considerar também a linguagem a ser usada no desenvolvimento de aplicações para o dispositivo que se intenta construir, uma vez que a linguagem de programação do hardware será basicamente a nativa de uma placa escolhida. Para essa escolha, considera-se linguagens que já possuem módulos de softwares prontos a serem utilizados e disponibilizem comunicação com a placa escolhida. Linguagens orientadas a objetos são o melhor caminho pois já vêm com inúmeros recursos prontos para serem usados (CARRO; WAGNER, 2003).

A Figura 6 resume os principais passos no processo de *design* de sistemas embarcados. Nesta visão "de cima para baixo" ou *top-down*, começa-se com os requisitos do sistema. Na próxima etapa: especificação, cria-se uma descrição mais detalhada do que se quer, mas isso indica apenas como o sistema se comportará, e não como ele será construído. Os detalhes internos do sistema começam a tomar forma quando se desenvolve a arquitetura, que proporciona a estrutura do sistema em termos de componentes. Uma vez que se sabe quais componentes necessários, pode-se projetá-los, incluindo tanto módulos de software quanto de hardware, e com base nestes componentes, pode-se, por fim, integrar o sistema completo (WOLF, 2001).

Figura 6: Níveis de Abstração em um Processo de *Design*.



Fonte: Adaptado de Wolf (2001)

2.3.1 Requisitos

Primeiramente, é necessário recolher uma descrição informal de potenciais usuários, que será usada como base para os requisitos, em seguida, refinar os requisitos em uma especificação que contenha informações suficientes para se começar a conceber a arquitetura do sistema. Os requisitos devem incluir requisitos funcionais e não funcionais, pois muitas vezes a descrição funcional, ou seja, a descrição de tarefas que o sistema é capaz de executar, não é suficiente, sendo assim os requisitos não funcionais incluem (WOLF, 2001):

Desempenho: A velocidade do sistema é muitas vezes uma consideração importante para a capacidade de utilização do sistema e para o seu custo final. O desempenho pode ser uma combinação de métricas de desempenho de software, como tempo aproximado para executar uma função em nível de usuário e prazos rígidos pelos quais uma determinada operação deve ser concluída.

Custo: O preço de custo e de compra para o sistema é importante e releva dois componentes principais: custo de produção, que inclui o custo de componentes e montagem e os custos com pessoas, que implementam o sistema. Além da consideração de quanto será vendido ou dos recursos disponíveis antes da implementação, para que ao fim o fator financeiro compense a concepção do sistema.

Tamanho físico e peso: Os aspectos físicos do sistema podem variar dependendo da aplicação. Um dispositivo portátil, por exemplo, tem requisitos rígidos sobre tamanho e peso que podem repercutir através de todo o projeto do sistema.

Consumo de energia: Energia, é claro, é importante para a sistemas que precisem de baterias, A energia pode ser especificada na fase de requisitos em termos de vida útil.

2.3.2 Especificação

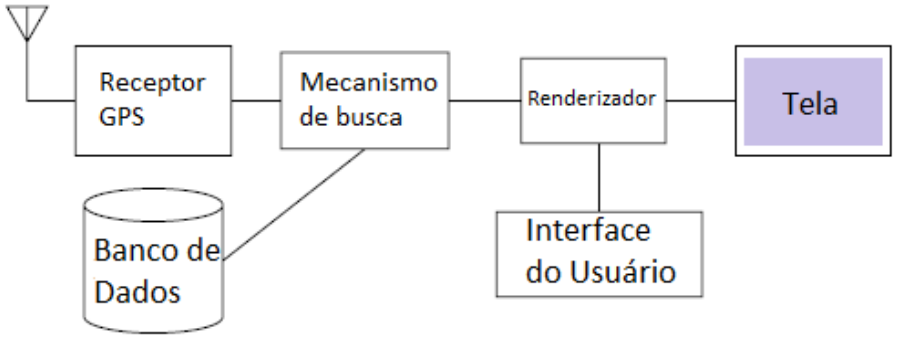
A fase de especificação é mais precisa, serve como o contrato entre o cliente e os arquitetos, deve ser cuidadosamente escrita de modo que reflita com precisão as necessidades do cliente, de uma forma que possa ser claramente seguida durante o projeto. No caso de uma especificação de um trabalho acadêmico, onde não há clientes, os próprios autores assumem metas a serem alcançadas (WOLF, 2001).

Geralmente, designers que não têm uma ideia clara do que precisam construir podem acabar equivocados quanto a decisões no início do desenvolvimento, o que pode provocar problemas em fases posteriores. Isso reforça a importância da especificação e a ideia de que ela deve ser compreensível o suficiente para que alguém possa verificar se é possível atender aos requisitos de sistema e as expectativas globais do cliente (WOLF, 2001).

2.3.3 Arquitetura

A especificação não diz a maneira que o sistema executa suas funções, apenas descreve essas funções. Descrever como o sistema executa as referidas funções, é a finalidade da arquitetura, que trata-se de um plano para a estrutura global do sistema, e é utilizada para projetar os componentes que o compõem (WOLF, 2001). A Figura 7 mostra uma arquitetura exemplo, formado por um diagrama de blocos que mostra as principais operações e dados de um sistema de GPS (*global positioning system* ou sistema de posicionamento global).

Figura 7: Diagrama de Blocos para um GPS.



Fonte: Adaptado de Wolf (2001)

Depois de criada uma arquitetura inicial, mesmo sem considerar detalhes de implementação, deve-se refinar o diagrama de blocos do sistema em dois diagramas de blocos: um para hardware e outro para software. A descrição da arquitetura deve abranger os requisitos funcionais e não funcionais, tais requisitos podem ser estimados através de pesquisa, experiência e modelos simplificados.

2.3.4 Componentes

A descrição da arquitetura mostra quais componentes são necessários para construção do sistema. Na fase de *Designing* de Componentes é atribuída a construção desses componentes, tanto de hardware quanto de software, em conformidade com a arquitetura e a especificação.

Os componentes em geral compreendem o hardware de FPGAs (Field Programmable Gate Array), placas de prototipação e módulos de software. A CPU será um componente padrão em quase todos os casos, como também chips de memória.

Usar módulos de software prontos relacionados ao sistema é uma boa maneira de economizar tempo de implementação, mas alguns componentes específicos têm de ser implementados do início, inclusive projetar placas de circuito impresso se for o caso, ao invés de usar placas padrão de circuitos integrados (WOLF, 2001).

Ao criar módulos de software embarcado, deve-se assegurar que o sistema funcione corretamente em tempo real e que não ocupe mais espaço de memória que o permitido. Deve se ter uma atenção a mais quanto a programação de acessos a memória, uma vez que são uma fonte importante de consumo de energia (WOLF, 2001).

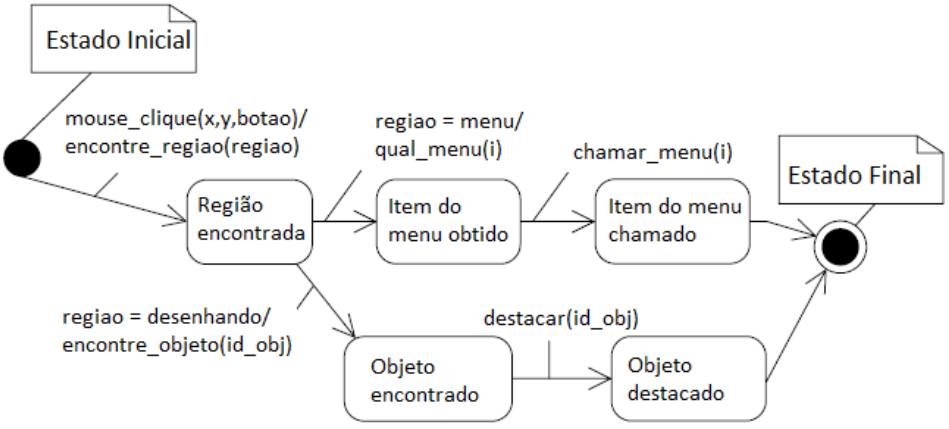
2.3.5 Integração do Sistema

Após a construção dos componentes chega a fase de juntá-los e fazer o sistema funcionar, mas essa fase é marcada principalmente pela detecção de erros. A depuração de módulos de software feita em separado ajuda a corrigir os erros em determinadas situações, por isso a independência dos módulos e divisão das fases deve ter sido em planejada desde o início.

2.3.6 Descrição Comportamental

Muitas vezes, diagramas são uteis para se conceituar tarefas. Uma maneira de especificar o comportamento de um sistema, bem como a sua estrutura, é por meio de uma máquina de estados. A Figura 8 mostra um diagrama UML (Unified Modeling Language ou Linguagem de Modelagem Unificada) que representa uma máquina de estados. A transição entre dois estados é mostrada por setas e a mudança de estado é desencadeada no sistema a partir da ocorrência de eventos, que podem ter sido disparados por usuários ou por um tempo limite atingido (WOLF, 2001).

Figura 8: Exemplo de uma Máquina de Estados.



Fonte: Adaptado de Wolf (2001)

Os estados da máquina de estado representam diferentes operações conceituais. Em alguns casos, ocorre uma transição condicional de estado baseada em dados ou em resultados de alguma computação feita no estado. Em outros casos, faz-se uma transição incondicional ao próximo estado. Tanto transições condicionais quanto incondicionais fazem o uso de eventos de chamada. Dividir uma operação complexa em vários estados

ajuda a documentar os passos necessários a execução de uma tarefa do sistema, o que facilita a estruturação do código do sistema (WOLF, 2001).

Considerando as necessidades do desenvolvimento de um sistema embarcado, destaca-se logo o fato de, como mencionado nessa sessão, ser preciso o uso de um processador ou microcontrolador como componente básico do sistema. Em se tratando de um protótipo, os microcontroladores se tornam uma opção mais viável, devido a sua facilidade de programação, preço e de sua demanda menor por recursos, quando comparado a um microprocessador, a seção a seguir dissertará sobre microcontroladores.

2.4 Microcontroladores

Microcontroladores são computadores embutidos, encontrados em uma grande variedade de aparelhos diferentes, como eletrodomésticos, eletroeletrônicos, aparelhos de comunicação e brinquedos. Eles são responsáveis por gerenciar os dispositivos e manipular a interface de usuário (TANENBAUM; AUSTIN, 2013).

Os microcontroladores possuem flexibilidade para satisfazer aplicações com requisitos muito variados, diversos tipos de E/S (entrada e saída), periféricos de comunicação incorporados, entre outros, usando um número mínimo de componentes adicionais (FERREIRA, 1998).

Sistemas complexos, formados por mais de um subsistema, facilmente apresentam mais de um microcontrolador. Um avião a jato, por exemplo, chega a ter duzentos ou mais deles e dentro de alguns anos, quase tudo que funciona com energia elétrica ou baterias possuirá, pelo menos, um microcontrolador (TANENBAUM; AUSTIN, 2013).

De fato, eles estão presentes em quase tudo o que envolve a eletrônica, pois possuem tamanho reduzido, e consequentemente diminuem o tamanho dos dispositivos onde estão instalados (MARTINS, 2005). Frequentemente usa-se a expressão *embedded applications* ou *embedded Systems* (aplicações embarcadas ou sistemas embarcados em português), para referir as aplicações nas quais o uso de microcontroladores permite atingir soluções muito compactas, com um reduzido número de componentes e requisitos mínimos de espaço/volume (FERREIRA, 1998).

Apesar de serem computadores pequenos, são completos, pois cada microcontrolador tem um processador, memória e capacidade de E/S. A capacidade de entrada e saída inclui: detectar o pressionamento de botões e interruptores de um aparelho, e com isso controlar luzes, monitores, sons e motores. Na maioria dos casos, o software está incorporado no chip na forma de memória apenas de leitura, criada na fabricação do microcontrolador (TANENBAUM; AUSTIN, 2013).

Outro fato é que quase todos os microcontroladores funcionam em tempo real, ou

seja, provém uma resposta imediata ao estímulo que lhe foi dado, o que torna seu uso imprescindível em aplicações com tal exigência. Além do mais, os sistemas embutidos muitas vezes têm limitações físicas relativas a tamanho, peso e consumo de energia, mas os microcontroladores utilizados nesses casos podem ser projetados, tendo essas restrições atendidas (TANENBAUM; AUSTIN, 2013).

Por meio da abordagem dos aspectos teóricos e práticos, é possível desenvolver projetos e implementações de sistemas microcontrolados de pequeno e médio porte, o que possibilita ao desenvolvedor usar criatividade e imaginação para desenvolver novos projetos de hardware e software. Os microcontroladores tornaram-se uma das melhores relações custo/benefício em se tratando de soluções que demandam processamento, baixo custo de hardware e limitação de espaço físico (MARTINS, 2005).

Tratando especificamente de trabalhos acadêmicos, como é o caso deste, os microcontroladores podem ser encontrados em algumas plataformas de prototipagem. Essas plataformas têm se popularizado em pesquisas e no desenvolvimento de diversas aplicações, dentre os motivos, destaca-se: por demandam uma pequena margem de aprendizado, possuindo fácil programação; existe uma grande variedade de *gadgets* para diversos tipos de aplicações; e, por manter o baixo custo em relação a outros dispositivos.

A seguir serão descritas as características de algumas dessas placas disponíveis no mercado como Arduino, Freescale Freedom, e outras, incluindo também placas como a Rapsberry Pi e Galileo, que também possuem um microprocessador, o que gera uma gama de novos recursos para os protótipos construídos com essas placas, caso seja necessário.

2.4.1 Arduino

Em 2005, foi desenvolvida uma ferramenta de prototipagem para propósitos educacionais chamada Arduino, uma placa de circuito embutida de um microcontrolador programável que pode ser usada para a criação de inúmeros projetos eletrônicos, de maneira facilitada. A placa dispõe de pinos de entrada e saída para conectar-se a dispositivos, tais como motores, atuadores e sensores.

A plataforma de modo geral é caracterizada como *hardware open source*, no qual a documentação para elaboração de projetos é disponibilizada para os usuários no site da plataforma. Além disso, também é disponibilizado um ambiente de desenvolvimento chamado IDE para a programação, fornecendo ainda o código fonte deste ambiente. O IDE Arduino pode executar nas plataformas Windows, Linux e Mac. A linguagem de programação utilizada é chamada Wiring, baseada em C++, e bem mais acessível que Assembly (linguagem comumente usada para controladores) (RAMOS et al., 2007).

Ao longo do surgimento da placa Arduino também surgiram as Shields, placas que podem ser conectadas as placas de prototipagem para estender suas capacidades. As

Shields Podem ser construídas manualmente ou podem ser compradas prontas, geralmente com alguma função específica, como controlador de rede sem fio, sensor de temperatura, controlador de motores, receptor de GPS, displays, entre outras.

A flexibilidade das Shields para aplicações é tal, que outras plataformas de prototipagem adotaram o mesmo modelo usado pela Arduino para a construção de suas placas Shields, inclusive apresentando compatibilidade de suas plataformas com as Shields Arduino. As Shields costumam vir com bibliotecas de software e exemplos, facilitando seu uso em um primeiro contato (EVANS; NOBLE; HOCHENBAUM, 2013).

A plataforma Arduino tem se popularizado em uma infinidade de aplicações, possibilitando que pessoas não especialistas possam colocar em prática suas ideias de interação com objetos e ambientes, fazendo uso de recursos da eletrônica e da programação. A popularidade do projeto incentivou diversas empresas a fabricarem o Arduino a custos acessíveis, inclusive no Brasil, como o projeto da placa X-Duino (**??**).

Desde que começaram a ser desenvolvidas, as placas Arduino foram sendo construídas em diversas versões, cada uma delas voltadas a aplicações específicas. As placas também passaram por diversas atualizações e upgrades, e hoje em dia, os modelos mais modernos de algumas versões são:

Arduino Uno:

É a versão mais recente da placa Arduino básica, é baseada no microcontrolador ATmega328P e se conecta ao computador por meio de um cabo USB padrão, contém todo o necessário para ser programada e usada em aplicações, além de poder ser estendida com uma variedade de Shields (placas de expansão que aumentam as capacidades da Arduino).

O nome "Uno" ("um" em italiano) foi escolhido para marcar o lançamento do Software Arduino (IDE) 1.0, ambiente amigável de programação onde programas podem ser escritos usando-se a linguagem Wiring.

Arduino Bluetooth:

A Arduino BT (bluetooth) é uma placa que inicialmente baseou-se no microcontrolador ATmega168, mas agora vem equipada com o ATmega328 e o módulo Bluetooth Bluegiga WT11. A placa suporta comunicação serial sem fios através de Bluetooth (mas não é compatível com fones de ouvido Bluetooth ou outros dispositivos de áudio).

Arduino Serial:

É uma placa básica que utiliza a entrada serial RS-232 como interface para ser programada ou se comunicar com um computador. Essa placa é fácil de montar e foi projetada para usar os componentes mais simples e mais fáceis de serem encontrados, de forma que sua montagem pode ser usada como um exercício de aprendizagem. Atualmente já se encontra na versão 2.0.

Arduino Single-Sided Serial

A placa de face simples já se encontra em sua terceira versão e é menos robusta do que outros modelos, na verdade trata-se de um projeto PCB, que pode ser construído de forma caseira, mas possui compatibilidade com as Shields Arduino.

Arduino Mega:

A Arduino Mega é uma das placas mais robustas dos modelos Arduino, baseada no microcontrolador ATmega1280, tem 54 pinos digitais de entrada/saída (dos quais 14 podem ser usados como saídas PWM), 16 entradas analógicas, 4 UARTs (portas seriais de hardware), um cristal oscilador de 16 MHz, uma conexão USB, entrada para fonte de energia, um cabeçalho ICSP, e um botão de reset. Ela contém todo o necessário para dar suporte ao microcontrolador, basta conectá-la a um computador com um cabo USB para começar a usar.

A placa é compatível com a maioria das Shields projetadas para versões anteriores da Arduino Uno, e pode ser alimentada através da conexão USB ou com uma fonte de alimentação externa, selecionada automaticamente, caso haja conexão simultânea de alimentação.

Arduino Lilypad:

O Arduino Lilypad é projetado para tecidos inteligentes e projetos *wearables* (vestíveis). A placa pode ser costurada em tecidos e montada com fios condutores a fontes de alimentação, sensores e atuadores.

A versão mais atual usa o microcontrolador ATmega328V, e tem um cabeçalho de programação de 6 pinos que é compatível com os cabos USB FTDI e Sparkfun FTDI Basic Breakout. A placa tem suporte para rearme automático, permitindo que esboços de programas sejam carregados sem pressionar o botão de reset na placa.

Arduino MKR1000:

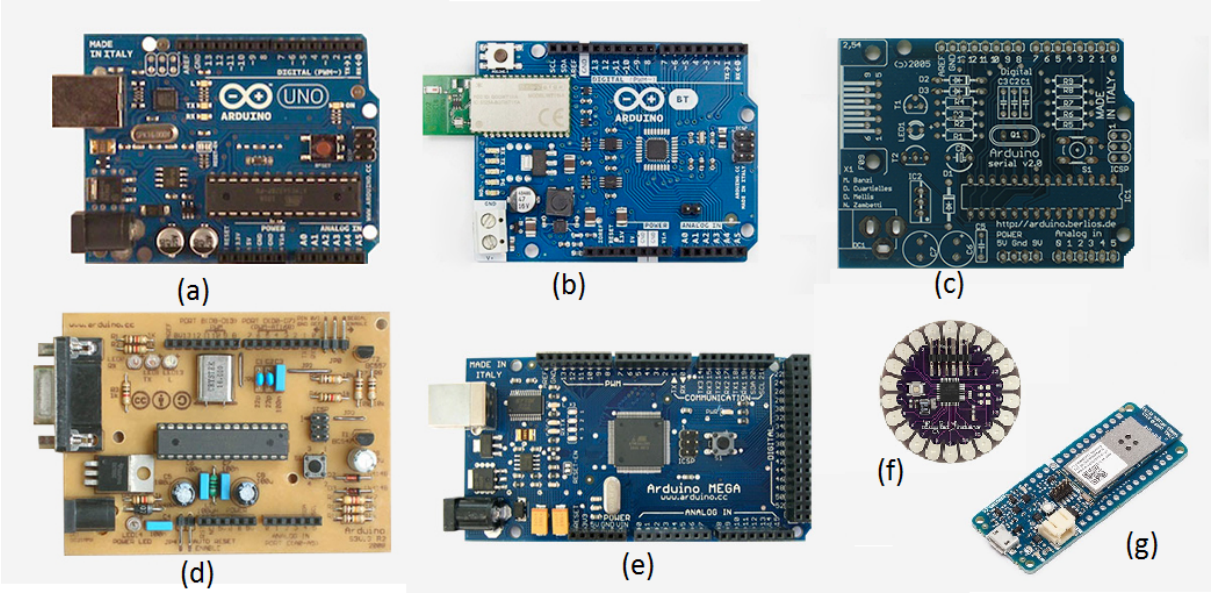
É uma poderosa placa que combina a funcionalidade de uma paca Arduino com uma Shield de Wi-Fi. É a solução ideal para os que desejam desenvolver projetos na área de Internet das coisas, oferecendo uma solução prática e rentável para os fabricantes que buscam acrescentar conectividade Wi-Fi em seus projetos. A placa é baseada no microcontrolador Atmel ATSAMW25 SoC (*System on Chip*), que é parte da família SmartConnect de dispositivos Atmel sem fio, projetado especificamente para projetos e dispositivos da Internet das coisas.

A Arduino MKR1000 possui poder computacional de 32 bit e um rico conjunto de interfaces de E/S, de baixa potência Wi-Fi com um *Cryptochip* para comunicação segura, e a facilidade de uso do Software Arduino (IDE) para o desenvolvimento de código e programação. Todas essas características tornam esta placa a escolha preferida para

os projetos emergentes de internet das coisas, que dependem da energia de baterias e precisam ocupar pouco espaço.

A seguir, a Figura 9 mostra as placas Arduino citadas anteriormente, em sequência tem-se: (a) Arduino Uno, (b) Arduino BT, (c) Arduino Serial, (d) Arduino Single-Sided Serial, (e) Arduino Mega, (f) Arduino Lilypad e (g) Arduino MKR1000.

Figura 9: Exemplos de Placas Arduino.



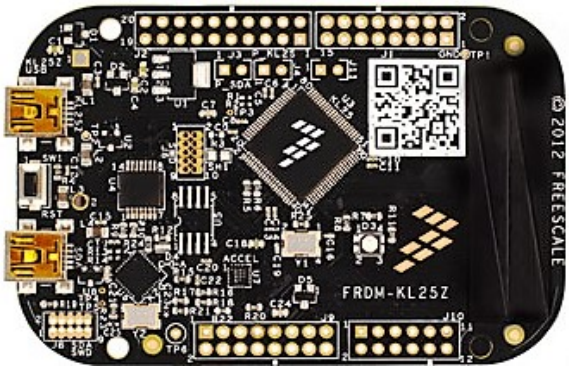
Arduino (2016).

2.4.2 Freedom

Em sua linha de ferramentas de desenvolvimento de IT *(Information Technology)*, a Freescale possui as placas Freedom, que permitem a desenvolvedores terem o primeiro contato com os dispositivos da marca, e em especial os microcontroladores de sua linha de processadores Kinetis (PRADO, 2015) com ARM Cortex M0 + core. São placas ideais para experiências rápidas, validação de conceitos e fins educacionais, como também, para primeiros protótipos de engenharia, em projetos que demandam um microcontrolador ou sensores (LIMA, 2014).

As placas de desenvolvimento Freedom são pequenas e consomem pouca energia, as mais simples chegam a custar 12 Dólares, tendo um bom custo benefício (no que se refere a prototipagem). Oferecem armazenamento em memória flash programável, possuem porta serial virtual e gerência de controle de recursos. Também podem ser usadas com várias placas de expansão de terceiros, inclusive Shields Arduino (PRADO, 2015), pois sua interface é de padrão aberto integrado (OpenSDA) (NXP, 2016) e vem integrada com um acelerômetro digital de três eixos, um *touch slider* embutido, sensor de luz ambiente e um magnetômetro. Um exemplo dessa placa é mostrado na figura 10.

Figura 10: Placa Freescale Freedom.



NXP (2016).

A placa possui ainda uma interface USB de uso geral e uma outra interface USB para comunicação serial e depuração através do padrão OpenSDA da Freescale. Esta mesma interface USB de depuração é usada para alimentar a placa. Para os microcontroladores da linha Kinetis, a Freescale provê gratuitamente e sem limitações o ambiente de desenvolvimento integrado *Kinetis Design Studio* (KDS). O KDS é baseado no Eclipse e utiliza o toolchain do projeto GNU (gcc, binutils, gdb, etc). Suporta oficialmente os sistemas operacionais Windows 7, Windows 8 e GNU/Linux (Ubuntu, Redhat, CentOS). O guia de usuário do KDS pode ser baixado do site da Freescale (PRADO, 2015).

Tem-se ainda, o *Kinetis Software Development Kit* (KSDK), um conjunto de bibliotecas, drivers, pilhas de protocolo e aplicações de exemplo para desenvolver projetos para a linha de microcontroladores Kinetis da Freescale. É gratuito e pode ser baixado diretamente do site da Freescale (NXP, 2016). Possui drivers para os periféricos de todos os microcontroladores da linha Kinetis, pilhas de protocolo TCP/IP e USB, suporte à sistemas de arquivo FAT, suporte aos sistemas operacionais de tempo real MQX, FreeRTOS, uC/OS-II e uC/OS-III, além de vários projetos de exemplo para a maioria das placas Freedom (PRADO, 2015).

2.4.3 Raspberry Pi

A placa Raspberry Pi é um computador completo de baixo custo, possuindo um microprocessador no lugar do microcontrolador, geralmente presentes neste tipo de placa, memória e entrada e saída. Mesmo sendo do tamanho de um cartão de crédito, pode ser conectada a um monitor, teclado e mouse normalmente, e fazer tudo o que se espera de um desktop comum. Dessa forma, é possível acessar a internet, reproduzir vídeos de alta definição, fazer planilhas, processamento de texto e executar jogos, uma vez que a plataforma é capaz de executar sistemas operacionais com ambientes gráficos conhecidos.

É uma plataforma flexível e educacional, vem pré-carregada com interpretadores e compiladores de diferentes linguagens de programação, desde Scratch (SCRATCH, 2016)

para iniciantes, passando por Python (PYTHON, 2016) e abrangendo as mais conhecidas, tais como C, Ruby e Java. Os programas podem ser escritos usando alguma dessas linguagens, e podem ser executados sem a necessidade de um sistema operacional, inclusive um sistema operacional inteiro pode ser escrito do início e executado na plataforma (RASPBERRYPI, 2016).

Além do mais, o Raspberry Pi tem a capacidade de interagir com o mundo exterior, por meio de sensores, atuadores, luzes e motores, podendo ser usado como controlador de vários projetos. Outras plataformas podem ser usadas para prototipagem de sistemas, porém, usando Raspberry Pi, os projetos terão uma capacidade além daqueles desenvolvidos em outras placas de prototipagem, pois o fato de possuir mais memória RAM e a capacidade de ter um SO carregado, amplia os recursos de desenvolvimento da placa.

É possível, por exemplo, se construir um termostato de uma maneira mais simples usando uma placa Arduino, porém, com a Raspberry Pi, o termostato poderia permitir acesso remoto e download do histórico de temperaturas (RASPBERRYPI, 2016). O fato de ser um computador totalmente funcional (um sistema em chip) torna a plataforma interessante para projetos mais robustos, caso haja necessidade de um SO e de memória RAM.

Aplicações que usam somente um microcontrolador geralmente são dedicadas, pois a lógica do desenvolvimento de sistemas embarcados preza economizar recursos como energia e espaço. Porém os sistemas dedicados precisam ser ligados a computadores externos para serem programados, e o fato da Raspberry Pi ser um computador completo em si, elimina essa necessidade, a partir do momento que um SO é carregado na placa. Em uma placa Raspberry Pi pode haver várias aplicações salvas em um cartão SD, que seriam executadas conforme algum controle, isso faria um sistema ter bem mais funcionalidades além dos sistemas embarcados comuns.

Existem vários modelos de placas Raspberry Pi disponíveis no mercado, a maior diferença entre esses modelos é preço e recursos. Há também uma nomenclatura aplicada pela Raspberry Pi Foundation (RASPBERRYPI, 2016) a suas criações: as versões com o nome "Model B" referem-se aos computadores mais capacitados da geração. O sinal "+" representa que o produto é uma revisão da versão original. "Model A" é o nome dado para as versões de entrada, mais simples, mas nem sempre mais baratas. Por fim, o "Zero" aparece apenas no Raspberry Pi mais barato (TECHTUDO, 2016). Os modelos atendem a diferentes perfis, e as versões suportadas oficialmente pela Raspberry Pi Foundation (RASPBERRYPI, 2016) são os seguintes:

Raspberry Pi 1 Model B+:

Lançada em julho de 2014, é a placa mais antiga disponível e se origina da primeira versão da Raspberry Pi. Possui um processador Broadcom de núcleo único e frequência de

700 MHz, além de 512 MB de memória RAM padrão DDR2. Para ligar periféricos a placa tem uma interface GPIO de 40 pinos, além de quatro portas USB 2.0, saída HDMI, slot para cartão de memória do tipo microSD, saída de som P2, porta Ethernet e interface para câmera e tela. O consumo de energia da placa fica entre 0,5 e 1 watt.

Apesar de ser uma revisão dos primeiros modelos da placa, a Raspberry Pi 1 Model B+ pode fazer sentido em aplicações menos ambiciosas e em projetos simples, seu preço fica em torno de R$ 100 se comprada no exterior (TECHTUDO, 2016).

Raspberry Pi 1 Model A+:

Versão mais acessível do Raspberry Pi original, o A+ funciona com o mesmo processador usado nos primeiros Raspberry Pi, mas tem a memória RAM limitada a 256 MB, que precisam ser compartilhados entre CPU e processador gráfico. Lançada em novembro de 2014, possui a porta GPIO de 40 pinos, mas apenas uma entrada USB 2.0. Contendo ainda saída de vídeo HDMI e para som tipo P2, além de slot para cartão de memória no formato microSD.

É a placa Raspberry Pi mais simples disponível no mercado e se torna uma alternativa interessante para quem precisa de um grande volume de placas para uso no ensino, ou para a aplicação em projetos bastante simples, seu preço oscila até os R$ 100 (TECHTUDO, 2016).

Raspberry Pi Zero:

O modelo foi criado para custar 5 dólares e é o computador mais simples já lançado pela fundação, em contrapartida dispõe de um hardware mais poderoso que o modelo A+: tendo um processador single-core de 1 GHz e memória RAM de 1 GB. Possui ainda uma saída microHDMI com capacidade para transmitir vídeo a 1080p e 60 quadros por segundo.

As limitações dessa versão são causadas pelo seu próprio tamanho: há apenas uma porta microUSB, mas o slot para cartão microSD e saída de som foram mantidos (TECHTUDO, 2016).

Raspberry Pi 2 Model B:

Faz parte da segunda geração de placas Raspberry Pi e possui diferenças consideráveis, das quais se destaca o uso de um processador de quatro núcleos com frequência de clock de 900 MHz, além de 1 GB de memória RAM. Oferece um conjunto de portas e interfaces igual ao apresentado na placa antecessora, a Raspberry Pi 1 Model B+, e é completamente compatível com sistemas operacionais mais conhecidos, como Windows, Ubuntu (versão Snappy) e Android.

Versátil e poderosa, a Raspberry Pi 2 Model B é alternativa para quem precisa rodar aplicações mais exigentes, ou busca criar projetos que dependam do suporte de

sistemas operacionais mais amigáveis. Mais fácil de ser encontrada no Brasil, a placa pode custar até R$ 370 (TECHTUDO, 2016).

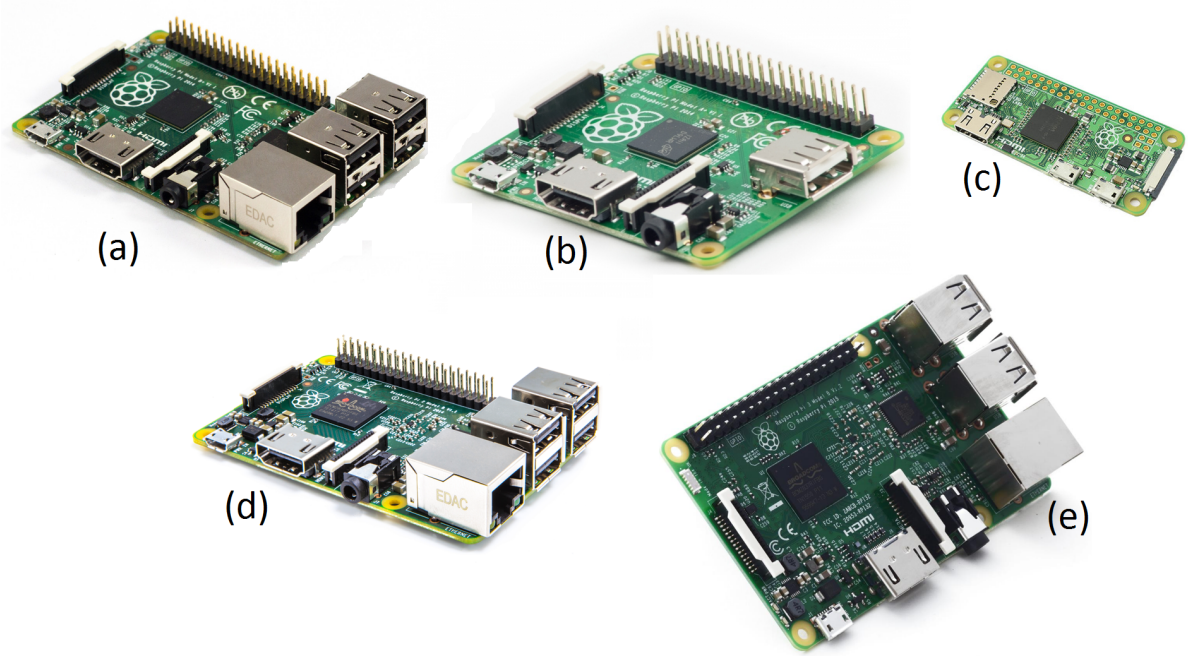
Raspberry Pi 3 Model B:

A terceira geração do modelo B aumenta a performance do processador para 1.2 GHz, mantendo os quatro núcleos, e é capaz de trabalhar com arquitetura de 64 bits. A RAM continua em 1 GB. Inclui interfaces sem fio de diversos tipos, como Bluetooth e Wi-Fi, desobrigando o usuário a ligar periféricos com essas funcionalidades. As demais interfaces do modelo B anterior são mantidas.

É o lançamento mais recente da Raspberry Pi Foundation, ideal para quem está interessado em projetos que dependam de hardware mais capacitado, ou que deseje investir em um modelo que terá um ciclo mais prolongado. No exterior, a placa foi anunciada com preço de £ 30 (libras esterlinas) (TECHTUDO, 2016).

A seguir, a Figura 11 mostra as placas Raspberry Pi citadas anteriormente, em sequência tem-se: (a) Raspberry Pi 1 Model B+, (b) Raspberry Pi 1 Model A+, (c) Raspberry Pi Zero, (d) Raspberry Pi 2 Model B e (e) Raspberry Pi 3 Model B.

Figura 11: Modelos da Plataforma Raspberry Pi.



Fonte: TechTudo (2016).

2.4.4 Beagle

A Fundação Beagle (2016) é uma corporação sem fins lucrativos com sede nos EUA que intenta fornecer educação e promoção da concepção e utilização de software open-source e hardware em computação embarcada. Assim surgiram as placas Beagle, que

assim como as Raspberry Pi, são minúsculos computadores com todos os recursos de um PC tais como processador, memória e suporte a sistemas operacionais.

Usando um cabo USB, a placa pode ser ligada em um PC, e será identificada como um dispositivo de armazenamento, com isso, a distribuição Angstrom (software que vem com a placa) já está em execução, e pode ser acessada via console. Assim como as plataformas mencionada anteriormente, existem diversos modelos de placas Beagle, com diferentes preços e especificações, vejamos alguns deles:

BeagleBoard:

Custando 125 dólares a BeagleBoard possui um processador OMAP3530 720MHz ARM Cortex-A8, com extensões NEON e VFP para a aceleração de clock adicional, hardware gráfico PowerVR, que lhe permite a capacidade de transmitir vídeo de alta resolução como um player de mídia portátil alimentado por USB. A placa se equivale em funcionalidade a um laptop, mesmo com seu tamanho reduzido.

BeagleBoard-xM:

A BeagleBoard-xM vem com o processador AM37x ARM de 1GHz, permitindo que amadores, estudantes e inovadores desenvolvam projetos robustos. Traz novamente o desempenho equivalente a um laptop, possui conectividade direta suportada por um hub de quatro portas on-board padrão 10/100 Ethernet.

BeagleBoard-X15:

BeagleBoard-X15 possui o melhor desempenho entre as placas Beagle, com processador TI AM5728 21.5-GHz ARM Cortex-A15, 2GBde memória RAM DDR3, suporte ao software Cloud9 IDE on Node.js para aplicações na nuvem, duas entradas Gigabit Ethernet além de uma interface eSATA (500mA) e uma saída de vídeo full-size HDMI. Em suma está placa pode ser usada para "qualquer tarefa computacional".

BeagleBone:

Os modelos BeagleBone são mais baratos que os modelos BeagleBoard, tendo seu preço situado na casa dos 90 dólares. O BeagleBone original é praticamente um computador do tamanho de um cartão de crédito, que se conecta à Internet e executa SOs tais como Android 4.0 e Ubuntu. Com a abundância de pinos de E/S e poder de processamento para análise em tempo real fornecido por um processador ARM AM335x 720MHz.

BeagleBone Black:

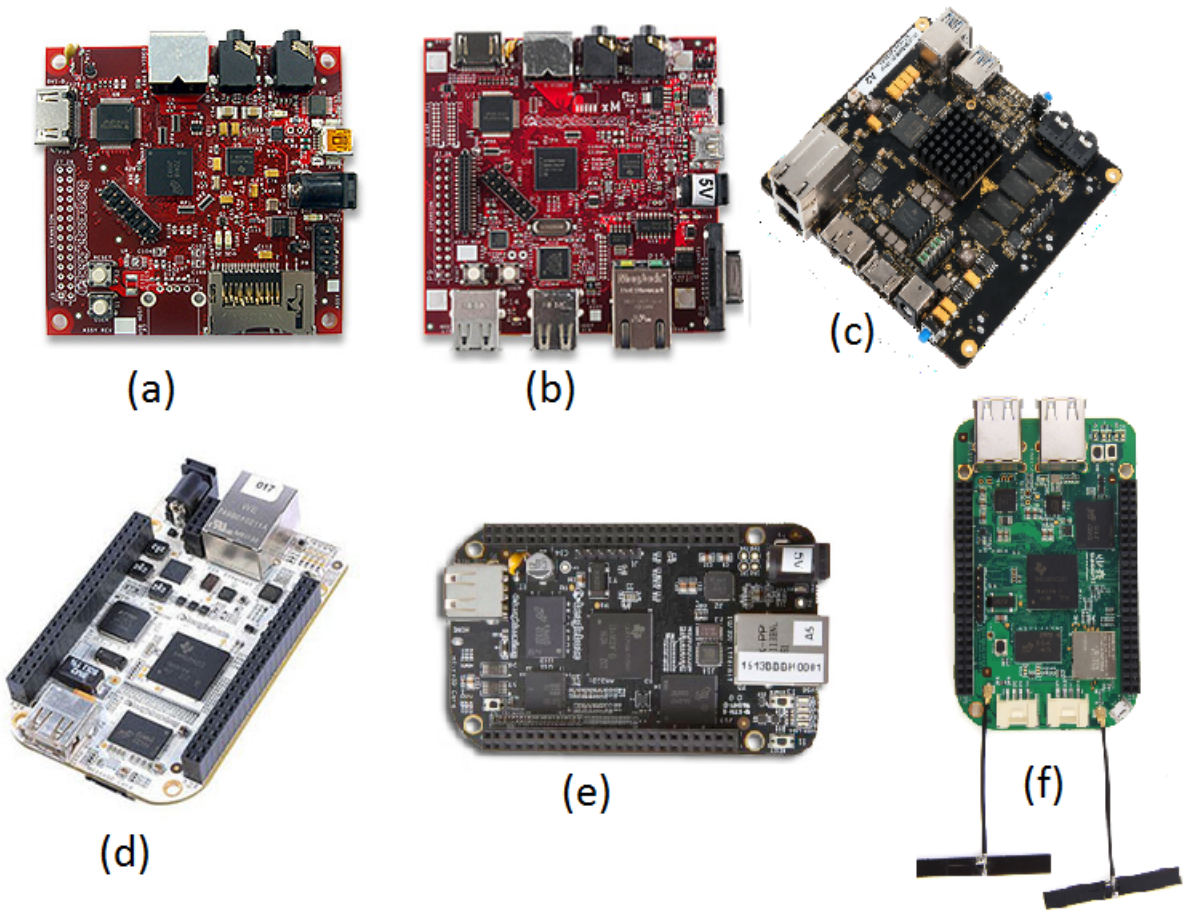
Um pouco mais robusta que a BlackBone original, executa o Boot do Linux em menos de 10 segundos e em 5 minutos já está pronta para o desenvolvimento de aplicações, sendo necessário apenas um cabo USB. Possui suporte a Cloud9 IDE on Node.js w/ BoneScript library para desenvolvimento em nuvem e interface HDMI.

SeeedStudio BeagleBone Green Wireless:

Está placa faz parte de um esforço em conjunto da Beagle (2016) e Seeed Studio (SEED.CC, 2016). É baseada no design de hardware de código aberto do BeagleBone Black e inclui uma interface Bluetooth flexível de alto desempenho e Wi-Fi, além de dois conectores Grove (do Seed Studio), tornando mais fácil para se conectar à grande família de sensores Grove. O on-board HDMI e Ethernet são removidas para dar espaço para essas funções sem fios e conectores Grove.

As placas contam com suporte para ambientes de desenvolvimento familiares como Ubuntu, QNX, Windows Embedded, Android e até mesmo programação Arduino (BEAGLE, 2016). A Beagle também possui uma comunidade bastante ativa. A Figura 12 mostra as placas acima mencionadas: (a) BeagleBoard, (b) BeagleBoard-xM, (c) BeagleBoard-X15, (d) BeagleBone, (e) BeagleBone Black e (f) SeeedStudio BeagleBone Green Wireless.

Figura 12: Modelos de Placa Beagle.



Fonte: Beagle (2016).

Independente do modelo, a placa Beagle vai operar como uma unidade flash que proporciona uma cópia local da documentação e controladores. Essa interface não pode

ser usada para reconfigurar o cartão microSD com uma nova imagem, mas pode ser usada para atualizar os parâmetros de inicialização usando o arquivo uEnv.txt.

O modelo BeagleBone Black é capaz de iniciar o Linux a partir de um eMMC (Multimedia Card) on-board de 2GB ou 4GB. O BeagleBone Black ou BeagleBone Original também podem ser iniciados a partir de um cartão microSD. O BeagleBone Original é fornecido com um cartão pré-configurado microSD de 4GB.

As placas Beagle podem aumentar sua funcionalidade sendo complementadas com placas plug-in chamadas "capes". As "capes" se assemelham as Shields da Arduino e sendo então, produtos da comunidade (BEAGLE, 2016), foram concebidas pelos próprios desenvolvedores usuários das placas Beagle, e embora já exista uma variedade de "capes", a comunidade ainda suporta o registro de novas por seus usuários.

Entre as disponíveis, se encontram "capes" para displays (DVI-D, HDMI, VGA, LCD, etc.), controle de motor, prototipagem e fonte de alimentação (bateria, solar, etc.) - entre outras funcionalidades.

2.4.5 Galileo

A placa Galileo é a primeira das placas Arduino a ser produzida baseada na arquitetura Intel. Projetada especificamente para fabricantes, estudantes, educadores e entusiastas de eletrônicos, para aplicações que demandam baixo consumo e alta performance.

A placa é considerada fácil de usar e foi pensada para iniciantes que queiram fazer projetos rápidos, que podem até ter um nível de complexidade mais alto, é um projeto completamente open-source, com todo o esquema eletrônico e os detalhes da placa disponíveis para download. O primeiro hardware completamente open-source da Intel (LIMA, 2016).

Possui um processador Intel Quark SoC X1000 de 32 bits, single-core, single-thread, e um conjunto de instruções compatível com a arquitetura ISA (Industry Standard Architecture), operando em velocidades de até 400 MHz; possui suporte para uma ampla gama de padrões da indústria, Interfaces de E/S, incluindo um slot para mini-PCI-Express, 100 Mb de porta Ethernet, entrada microSD, porta USB e porta de cliente em tamanho real USB; 256 MB de memória RAM DDR3, 512 kb SRAM incorporado, 8 MB Flash NOR e padrão EEPROM 8 kb na placa, além de suporte para cartão microSD de até 32 GB (INTEL, 2016b).

O Hardware da placa tem compatibilidade com uma vasta gama de Shields (placas de expansão) para Arduino, e também é programável através da IDE da Arduino e das bibliotecas de código livre. É capaz de executar sistemas operacionais como Linux e Windows IoT (Internet of Things) (INTEL, 2016b).

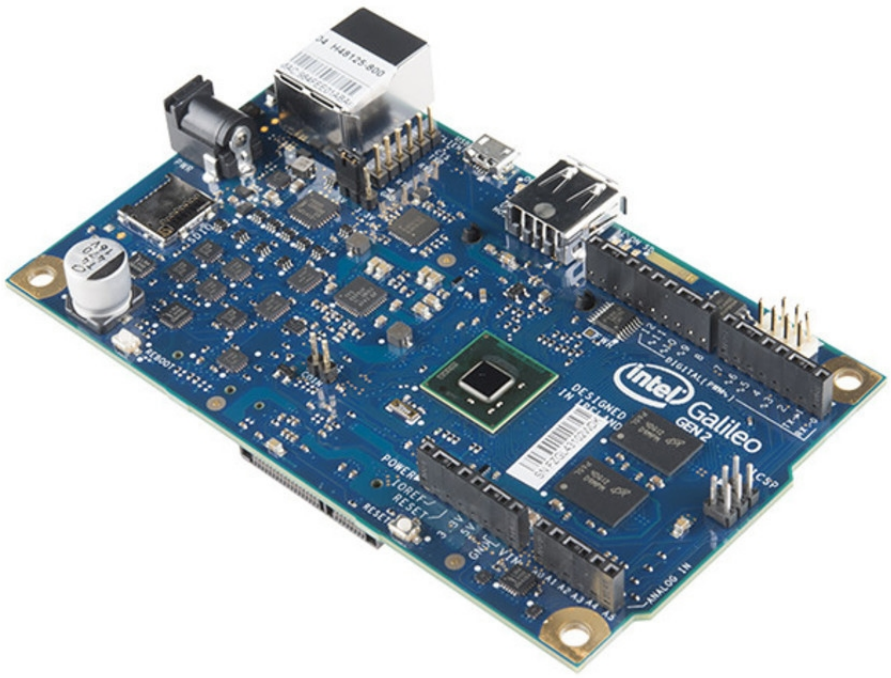
Pode-se dizer, que a placa Intel Galileo possui um Linux minimizado e um simulador de Arduino UNO R3 que funciona a partir de software, com um processamento bem mais alto do que uma placa Arduino. A distribuição linux que vem instalada na placa é o Poky 9.0.2 (Yocto Project 1.4 Reference Distro) e ainda é possível instalar uma distribuição linux no SD card externo e executar o boot para rodar com suporte a diversos módulos externos, que poderiam ser conectados aos barramentos extras, por exemplo (LIMA, 2016).

Para alimentar a placa é necessária uma fonte de 5V e para comunicação com um PC, é necessário um cabo micro USB. Se a placa for conectada a uma shield, ela também consegue fornecer a tensão de 5 V para a shield, o mesmo acontece caso a placa alimente uma shield ou circuito externo em 3V3. Isso faz com que os shields Arduino, que servem no UNO R3, funcionem perfeitamente na Galileo.

A placa possui dois botões, um para resetar o sketch programado (programa em arduino) na placa e o outro para reiniciar inteiramente o SO embarcado. Se a placa possuir um cartão SD, é possível que armazene diferentes sketches (LIMA, 2016).

Atualmente a placa se encontra na sua segunda versão: Intel Galileo Gen 2, mostrada na Figura 13 . Com dimensões um pouco maiores que sua antecessora, a Gen 2 pode ser alimentada com tensão de 7V a 17V sem problemas, além de permitir alimentação via entrada Ethernet, usando um módulo externo; possui controle de PMW de 12 bits, interface padrão UART para debug e padrão GPIO em seus pinos de entrada e saída, o que permite uma maior drenagem ou injeção de corrente nesses pinos.

Figura 13: Intel Galileo.



Fonte: Intel (2016b).

2.4.6 Edison

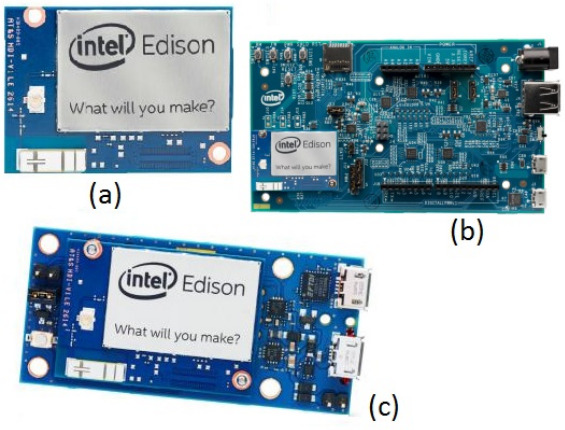
Projetado para fabricantes, empresários e algumas aplicações industriais de Internet das coisas, o Intel Edison é um módulo de computador que oferece facilidade de desenvolvimento para uma série de projetos de prototipagem ou empreendimentos comerciais quando o desempenho é importante.

O módulo possui ambiente de desenvolvimento de software de código aberto, alta performance, CPU dual-core e microcontrolador single-core. Oferece suporte a coleção de dados complexos usando baixa potência;

Wi-Fi integrado, Bluetooth 4.0, memória RAM de 1GB DDR e 4GB de memória flash, que simplifica a configuração e aumenta a escalabilidade (INTEL, 2016a). Este módulo é direcionado para projetos altamente robustos, com grande processamento de dados e necessidade de memória.

Os módulos Intel Edison suportam o desenvolvimento com Arduino, C/C++, seguidos de Node.JS, Python e RTOS. Também suporta o linux Yocto 1.6. Como o módulo vem separado de qualquer componente, ou seja, é praticamente como se fosse um microprocessador (e microcontrolador) isolado, ele pode ser adquirido junto com kits, como o Kit For Arduino que permite a conexão com *shields* Arduino e o kit *Breakout Board*, que apesar de poucas conexões permite prototipagem rápida e desenvolvimento de software. A Figura 14 mostra um módulo Edison (a) e os dois kits citados: Kit for Arduino (b) e *Breakout Board* (c).

Figura 14: Intel Edison.

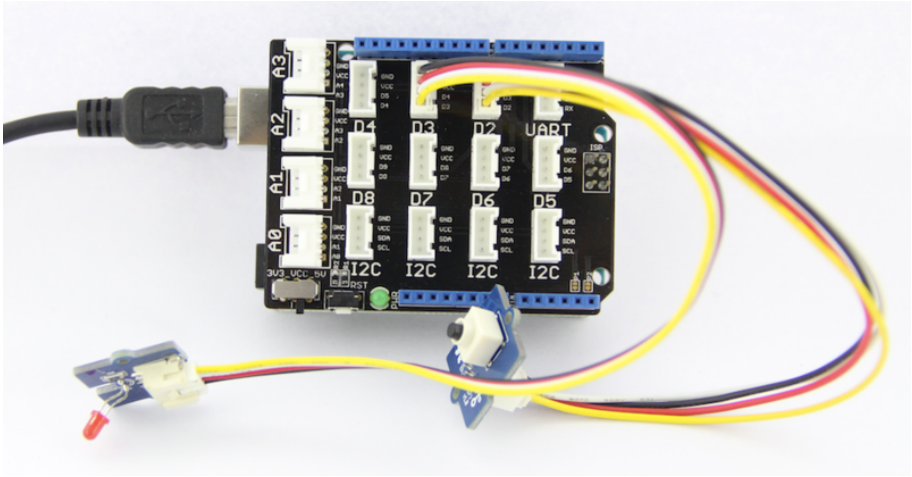


Fonte: Intel (2016a).

Por se tratar de uma arquitetura x86, muitas distribuições Linux podem ser adaptadas, ou até mesmo ser criada com foco nessa plataforma. Além disso, a Intel também destaca que o módulo Edison inclui um framework de conectividade que permite comunicações Inter dispositivos envolvendo comunicação direta e com a nuvem (CURVELLO, 2014).

A Intel também usa placas de expansão para os módulos Edison, assim como tem-se as Shields Arduino e Capes para Beagle. Aqui, as expansões usadas são as "Groves" da seed.cc (2016), sendo inclusive possível, usar-se uma "grove base", que implementa um sistema de conexão baseado em cabos equipados com um conector padrão (ZTOP, 2016), combinando várias placas de expansão de um sistema, como visto na figura 15 .

Figura 15: Grove Base Shield.



Fonte: seed.cc (2016).

2.4.7 Conclusão Sobre os Microcontroladores Apresentados

Microcontroladores tendem a ser mais específicos na realização de determinadas tarefas, gerenciando diretamente os recursos do hardware, sem a necessidade de um processamento extra adicional. As placas de prototipagem apresentam interfaces de comunicação e outros recursos adicionais, que facilitam um primeiro contato com a complexidade de programação de um microcontrolador, permitindo que programas e testes sejam executados facilmente.

O laboratório LAACOSTE (Laboratório de Automação, Arquitetura de Computadores e Sistemas Embarcados) da UFERSA (Universidade Federal Rural do Semi-Árido), onde o projeto deste trabalho foi desenvolvido, possui as placas Arduino e Raspeberry Pi disponíveis para serem usadas. Em um primeiro momento, não houve necessidade de se usar a placa Raspberry Pi, apesar de que ela permitiria usar mais recursos, tais como armazenar as aplicações no próprio dispositivo e não precisar usar um PC para executar os programas, mesmo assim a placa Arduino ainda se sobrepôs pela simplicidade de uso.

Em comparação com as demais, a placa Arduino foi escolhida para o desenvolvimento deste trabalho, devido também a sua maior disponibilidade, suporte da grande comunidade online e abundante existência de Shields. Pode-se perceber que algumas das outras placas citadas são compatíveis com as mesmas Shields da Arduino e inclusive suportam a mesma linguagem de programação (Wire) da Arduino, porém, as Shields foram

primeiramente projetadas para Arduino, o que evita algum conflito de compatibilidade que eventualmente poderia ocorrer, e no que se refere a linguagem, o interessante é conhecer primeiro como os programas se comportariam na plataforma original, para depois se pensar em migrá-lo para outra plataforma, já que todas as plataformas citadas poderiam ser usadas, fazendo-se as devidas modificações.

Outro fator é que placas como a Beagle, por exemplo, possuem um poder computacional muito alto, não sendo necessária em projetos que não necessitem de muito processamento, outros recursos como as variadas interfaces de entrada e saída também não seriam utilizados, logo, a placa Arduino ganha no quesito custo, altamente importante em sistemas embarcados. A seguir, Figura 16 resume algumas especificações das seis plataformas citadas anteriormente.

Figura 16: Especificações das Plataformas.

Plataforma	Arduino	Freedom	Raspberry Pi	Beagle	Galileo	Edison
Variante	Uno	FRDM-KL25Z	Model B	Rev. C4	Gen 2	Intel Edison Compute Module (IoT)
Sistema Operacional	-	-	Linux, RISC OS	Android, Linux, Windows CE, RISC OS	Yocto Project	Yocto Project, Brillo
Ambiente	Arduino IDE, Ecplipse	Codewarrior, Ecplipse	OpenEmbedded, QEMU, Scratchbox, Eclipse	Eclipse, Android ADK, Scratchbox	Arduino IDE	Arduino IDE, Eclipse
Linguagem de Programação	Wiring-based(C++)	C	Python, C, Java, Ruby, Scratch	Python, C	C, C++, Python, Javascript	C/C++, Python, Node.js, HTML5, JavaScript
Arquitetura	8 Bits	32 Bits	32 Bits	32 Bits	32 Bits	32 Bits
Controlador/Processador	ATMEGA328	ARM Cortex M0 + core	BCM2835 (ARM)	TI DM3730 (ARM)	Intel Quark SoC X1000	Dual-Core Intel Atom
Frequência de Operação	16 MHz	700 MHz	48 MHz	720 MHz	400 MHz	500 MHz
Memória RAM	2 Kb	256 Mb	128 Kb	256 Mb	256 Mb	1 GB

Fonte: Arduino (2016); NXP (2016); RaspberryPi (2016); Beagle (2016); Intel (2016a).

2.5 Motores

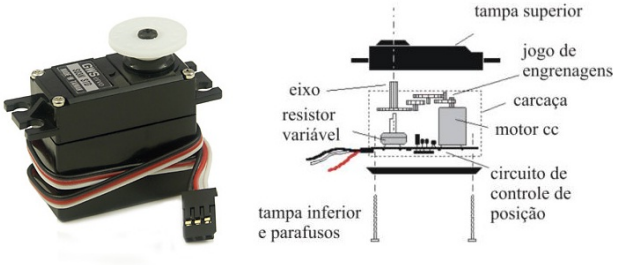
Ao se pensar em construir um dispositivo que seja dotado de movimento próprio, surge a necessidade de se trabalhar com motores. Existem alguns tipos de motores que são comumente usados em prototipação de projetos acadêmicos, e que podem ser facilmente controlados por uma placa de prototipação como as citadas anteriormente. No intuito de escolher qual motor ideal para este trabalho, foi feito um pequeno levantamento com os três motores mais comuns utilizados por projetos iniciais, são eles: o servo motor, o motor de passo e o motor DC.

2.5.1 Servo Motor

Os motores servo são bastante utilizados em projetos de robótica, especialmente nos braços e pernas de robôs. Existem vários modelos, mas no geral são motores pequenos,

precisos e capazes de girar 90°(noventa graus), 180°(cento e oitenta graus) e alguns modelos alcançam os 360°(trezentos e sessenta graus). São capazes de se posicionarem em um ângulo específico, recebendo um sinal de controle, verificando a posição atual do eixo, e a partir daí girar até a posição desejada. Para isso, possuem um sensor, que geralmente é um potenciômetro solidário ao eixo. Possuem ainda um sistema atuador composto de um motor de corrente contínua ligado a engrenagens e um circuito de controle que indica o sentido no qual o motor deve girar (ROBOLIV, 2016). A Figura 17 mostra um servo motor e suas partes.

Figura 17: Servo Motor e Suas Partes.



Fonte: Ricardo (2016).

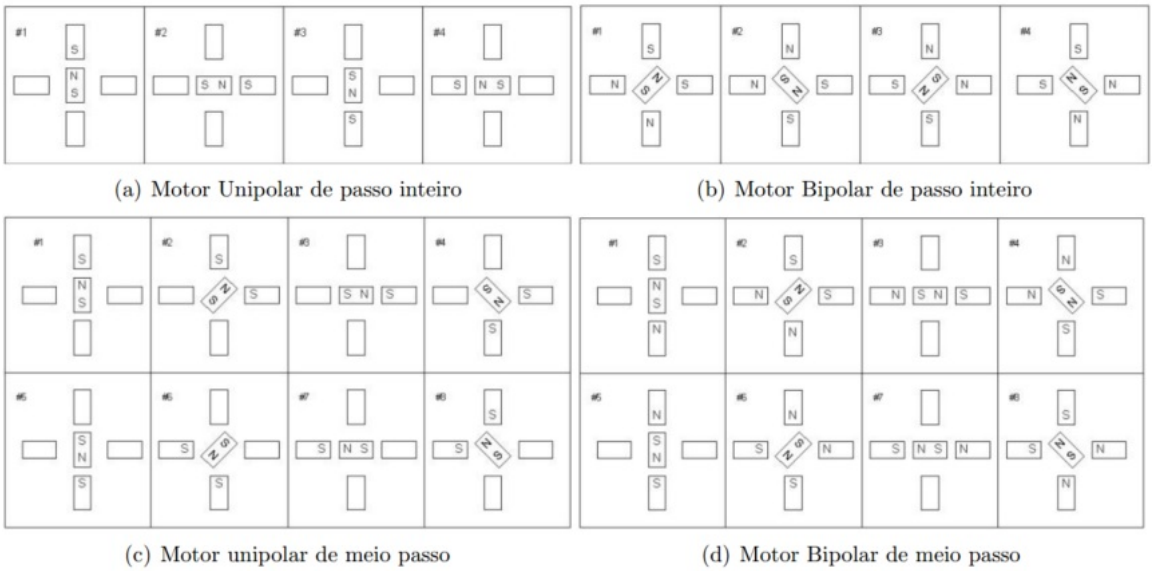
2.5.2 Motor de Passo

Motores de passo ou *steppers motors* são dispositivos que convertem pulsos elétricos em movimentos mecânicos, rotacionando seu eixo em pequenos incrementos angulares, denominados “passos”. São motores precisos, porém lentos, logo são indicados para projetos que demandam alta precisão, mas não grandes velocidades. Sua velocidade de giro do rotor (eixo) varia de acordo com a frequência de pulsos elétricos e o tamanho do ângulo é relacionado ao número de pulsos aplicados. Solenoides (condutores enrolados em forma de espiral) alinhados dois a dois, quando energizados, atraem o rotor para uma direção, o que gera o movimento do motor, determinada pela ordem em que os solenoides são ativados (SANTOS; BRITES, 2008). A Figura 18 mostra como acontece o movimento do rotor em quatro tipos diferentes de motores de passo.

2.5.3 Motor DC

Um motor DC (*direct current* ou corrente contínua) é um dispositivo que converte a energia elétrica em energia mecânica. Apesar da baixa precisão, o motor possui boa velocidade (CARDARELLI et al., 2016) e deve ser alimentado com tensão contínua, que pode provir de pilhas e baterias, para motores pequenos, ou de uma rede alternada após retificação, no caso de motores maiores. Os principais componentes desse tipo de motor são: o estator, um enrolamento alimentado por uma fonte, ou um simples imã permanente, em se tratando de motores pequenos; o rotor ou armadura, que é alimentado por uma fonte

Figura 18: Tipos de motor de passo e movimentos.



(a) Motor Unipolar de passo inteiro

(b) Motor Bipolar de passo inteiro

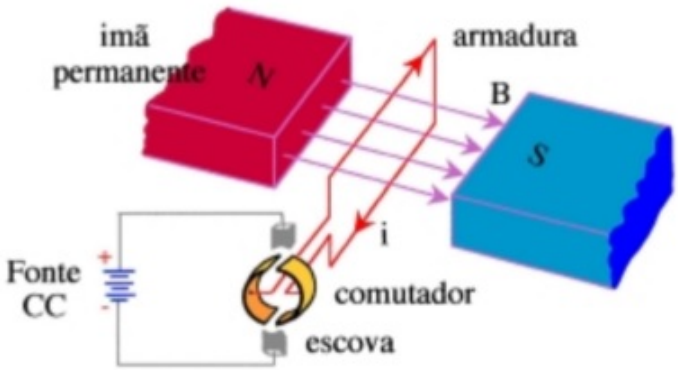
(c) Motor unipolar de meio passo

(d) Motor Bipolar de meio passo

Fonte: Santos e Brites (2008).

de tensão contínua através do comutador e escovas de grafite; e o comutador, dispositivo mecânico no qual estão conectados os terminais da armadura, e cujo papel é inverter sistematicamente o sentido da corrente contínua que circula na armadura (FRANcA, 2001). A Figura 19 mostra a estrutura básica de um motor de corrente contínua elementar com imã permanente no estator.

Figura 19: Estrutura Básica de um Motor de Corrente Contínua



Fonte: França (2001).

2.6 Motor Shield

As placas de prototipagem são perfeitamente capazes de controlar motores, como os citados anteriormente, mas não vários deles ao mesmo tempo, pois estão limitadas ao número de pinos de entrada e saída disponíveis em sua interface, principalmente se

estas entradas estiverem sendo usadas para outras funcionalidades. Para isso utiliza-se das placas de expansão, neste caso, para controlar vários motores.

Uma *motor Shield* é capaz de controlar vários motores simultaneamente de forma independente, e não se limita as entradas de controle da placa a qual está anexada. Além do mais, os motores geralmente precisam de mais energia, necessitando muitas vezes de uma fonte de alimentação externa (ADAFRUIT, 2016).

Como a plataforma de prototipagem escolhida foi a Arduino, é natural que seja escolhida uma Shield Arduino para controlar os motores do projeto, existem várias dessas placas controladoras de motor no mercado, dentre algumas pode-se citar:

Motor Shield L293D Driver Ponte H para Arduino:

Conhecida popularmente como Ponte H, está Shield Arduino é baseada no chip L293D, e trata-se de um circuito integrado de alta tensão, alta corrente e controle de 4 canais, o que significa que pode ser ligada em motores DC de tensão de mais de 36 V sem problemas. Com esta Shield é possível controlar até 4 Motores DC, 2 Servos ou 2 Motores de Passo.

O chip L293D possui internamente duas pontes H e suporta uma corrente de saída de 600mA por canal, ou seja, é possível controlar até dois motores com 600 mA (miliamperes) cada, visto que nesta Shield temos 2 chips (FILIPEFLOP, 2016).

SparkFun Ardumoto - Motor Driver Shield:

Esta é uma motor shield para Arduino que controla dois motores DC ou um motor de passo. Possui uma ponte H L298 que o permite controlar até 2 A (amperes) por canal. Inclui LEDs azuis e amarelos para indicar a direção ativa de um motor, e todas as linhas de drivers são de diodo protegido contra EMF de volta (SPARKFUN, 2016).

A placa é indicada para projetos com arduino tais como controlar carrinhos RC, apesar de controlar apenas dois motores DC ao mesmo tempo, é capaz de ser ligada com 3.3 V apenas, e pode alcançar toques maiores se ligada a uma bateria externa.

Adafruit Motor Shield V2:

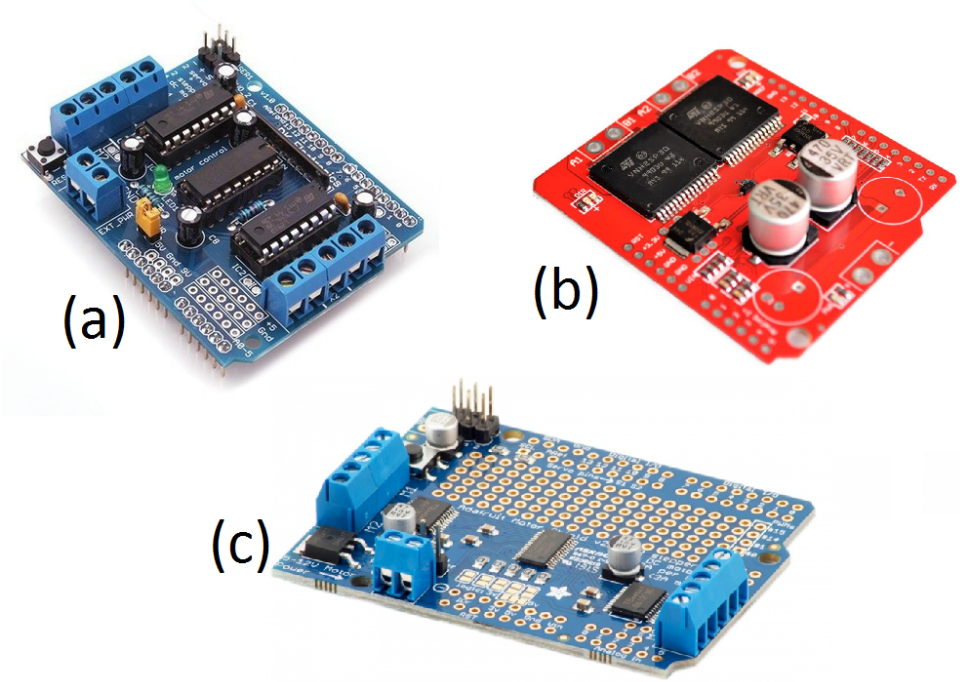
Segunda versão de uma das mais populares placas controladoras de motor: a Adafruit Motor Shield, é capaz de controlar até 4 motores DC ou 2 motores de passo. Usa drivers TB6612 MOSFET com 1.2A por canal e tem menos queda de tensão, o que a deixa capaz de extrair mais torque de uma bateria. Tem-se um chip controlador PWM totalmente dedicado a bordo e foi projetada para ser completamente empilhável, permitindo a sobreposição de shields que aumentaria o número de motores controláveis.

Possui ainda 2 conexões para motores servos de 5V, 4 pontes H: chipset TB6612 com 1.2 A por ponte com proteção por desligamento térmico e pode executar motores nas tensões de 4.5V a 13.5V. A placa também possui uma biblioteca própria com o uso de

funções e instruções (ADAFRUIT, 2016).

A Adafruit Motor Shield foi escolhida para integrar este projeto, tendo em vista que é capaz de controlar quatro motores DC ao mesmo tempo, além do mais possui biblioteca de fácil utilização e é capaz de ser empilhada, o que proporciona ao projeto a possibilidade de ser expandido futuramente, com o uso de mais motores. Na Figura 20 se mostra as motor shields descritas anteriormente: (a) Motor Shield L293D Driver Ponte H para Arduino, (b) SparkFun Ardumoto - Motor Driver Shield e (c) Adafruit Motor Shield.

Figura 20: Motor Shields Arduino.



Fonte: Filipeflop (2016); Sparkfun (2016) e Adafruit (2016).

2.7 Bibliotecas de Software

Assim como existem bibliotecas de software disponibilizadas para placas controladoras de motor, por exemplo, este projeto disponibiliza também uma biblioteca para ser usada em conjunto com o dispositivo proposto, com intuito de simplificar suas funcionalidades a futuros programadores e usuários.

Uma biblioteca é um conjunto de funções pré-escritas por outros programadores, de tal sorte, que não é necessário implementar todas as funções que serão utilizadas em determinado programa, principalmente se forem muito complexas, existe a possibilidade de se usar as já existentes em uma biblioteca, importando a biblioteca para o código em questão (JÁCOME, 2010).

Alguns exemplos de bibliotecas muito conhecidas são a stdlib.h, que possui funções básicas da linguagem C, e a stdio.h, também da linguagem C, e possui funções que tratam de entrada e saída. Na prática, basta digitar o comando que inclui a biblioteca no início do código, assim pode-se utilizar as funções da mesma no decorrer da programação.

Empregar bibliotecas de terceiros em projetos de software é uma prática comum, pois elas encapsulam um grande número de funções úteis, bem testadas e robustas, e melhoram consideravelmente a produtividade dos programadores (SUN; KHOO; ZHANG, 2011).

As aplicações de alto nível direcionadas para microcontroladores, geralmente, requerem a implementação de algum algoritmo de controle. Essas aplicações podem se beneficiar de uma biblioteca de software que contenha tais algoritmos, além de outras funções uteis à aplicação, de maneira que esses códigos estejam prontos para serem usados, sem a necessidade de serem escritos novamente (DOLINAY; DOSTÁLEK; VAŠEK, 2015).

3 Revisão Bibliográfica

A seguir, apresenta-se um levantamento bibliográfico com alguns trabalhos relacionados ao objetivo da pesquisa. Os trabalhos tratam do desenvolvimento de projetos com características tangíveis, focados mais especificamente em mesas e trabalhos que usam a plataforma Arduino como microcontrolador.

3.1 Projetos Tangíveis Usando Mesas

O artigo de Kato et al. (2000) mostra como manipular objetos em uma mesa de realidade aumentada. Ele destaca o fato do móvel ser usado para reuniões de projetos e tomadas de decisão, e apresenta o conceito de TAR (*Tangible Augmented Reality*) ou Realidade Aumentada Tangível. O material usado no trabalho foi uma mesa em que havia a interação de cartas com símbolos, e pessoas equipadas com óculos para enxergarem através de câmeras, com o objetivo de visualizar e interagir com objetos virtuais que a posição das cartas mostrava (Figura 21).

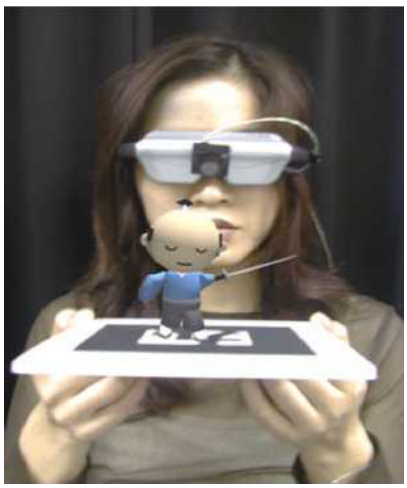
Foi usado um jogo para demonstrar o funcionamento do sistema, cujo objetivo era encontrar cartas com objetos que possuíam alguma relação. Quando o usuário estava equipado com o óculos especial, ele era capaz de ver objetos virtuais no ambiente, esses objetos "surgiam" no momento que a câmera do óculos identificava um símbolo correspondente, o sistema então renderizava o objeto em questão na posição em que a carta se encontrava. Quando duas cartas com objetos relacionados se aproximavam, um 'alienígina' e um 'disco voador', os objetos interagiam, criando uma animação virtual: o alienígina entrava no disco voador e seguia viagem para longe, assim essas cartas eram descartadas e uma nova interação iria ocorrer quando outras cartas fossem jogadas.

Os usuários não realizavam grandes movimentos com o cartão, uma vez que viam os objetos, apesar de que vários deles giravam o cartão para enxergar as figuras em diferentes ângulos. Havia cartas que não possuíam um relacionamento tão obvio, então alguns usuários precisavam da ajuda de outros colaboradores na mesa e os jogadores, muitas vezes colaboravam espontaneamente com estranhos que tinham a carta que precisavam.

A fim de acompanhar a posição do usuário e do objeto, foram feitas modificações no ambiente de RA (realidade aumentada), anexando pontos de rastreamento à superfície da mesa. O rastreamento foi feito considerando os padrões de localização desses pontos, e assim, definido um conjunto de eixos de coordenadas alinhadas com a superfície da mesa. A câmera anexada aos óculos detectava automaticamente a posição do objeto na RA, baseando-se na distância que media entre os pontos e o objeto.

É possível destacar alguns métodos de design de TUIs descritos na publicação, tais como: as *affordances* (capacidade do objeto de se parecer com sua função) dos objetos devem coincidir com as limitações físicas dos requisitos das tarefas; a capacidade de suportar a atividade paralela no qual vários objetos ou elementos de interface estejam sendo manipulados de uma só vez; suporte para técnicas de interação baseadas na física (como o uso de objeto proximidade ou relações espaciais); o fato de que a forma dos objetos deve encorajar e apoiar manipulação espacial e suporte para interação de várias pessoas. Uma das aplicações reais desse projeto seria usá-lo para projetar construções.

Figura 21: Um Objeto Virtual em uma Carta



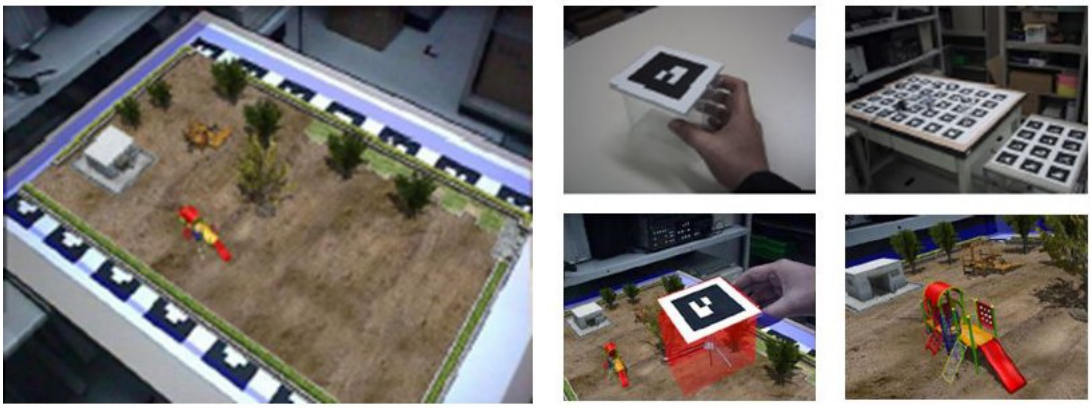
Fonte: KATO et al. (2000)

Em Kato et al. (2003), apresenta-se um projeto semelhante ao descrito anteriormente: uma mesa com a realidade aumentada usada para planejar projetos arquiteturais. Também foram usados princípios parecidos com os que foram apresentados no artigo anterior, tais como usar uma mesa, equipar usuários com óculos que possuem câmeras e cartas com símbolos para que o computador localize a posição da carta e faça o usuário do óculos enxergar um objeto virtual.

A diferença entre os *papers* está no fato de que este projeto se direciona para o planejamento de cidades ou áreas urbanas como parques e praças, mas o grande diferencial é usar uma interface com o formato de copo, que serve de cursor para plotar, pegar, colocar, mover e apagar objetos durante o processo de criação. A Figura 22 mostra algumas etapas do uso da ferramenta.

A forma como o usuário pode manipular objetos virtuais tem que ser intuitiva e fácil. Nesse sistema, a interface é um copo transparente que contém um marcador, e a partir deste é que a câmera presente no óculos consegue localizar a interface e renderizar uma imagem dentro do copo. O copo é usado de cabeça pra baixo e o marcador localiza-se por cima dele, para utilizar esse tipo de marcador foi usada uma biblioteca chamada ARToolKit (KATO; BILLINGHURST, 1999).

Figura 22: Projeto Tangível MagicCup



Fonte: KATO et al. (2003)

A ação de pegar um objeto acontece quando o copo "vazio" abrange um objecto virtual sobre a mesa, o software identifica os marcadores do copo e da mesa e em seguida, quando o copo é retirado, o objeto virtual sai da mesa e permanece no copo. Para colocar de volta, basta posicionar o copo, que agora contém um objeto, em uma localização na mesa e posicionar a direção, ao retirar o copo, o obejto permanece na mesa.

As duas ações anteriores também são usadas para mover um objeto virtual de uma posição para outra, mas se o usuário quiser apenas envolver um objeto e deslizá-lo sobre a mesa para mudar sua posição sem levantar o copo, tamém é possível. A inserção de objetos acontece quando se convenciona qual obejto o marcador do copo representa, e esse marcador pode ser trocado, permitindo a inserçao de vários tipos de objetos. Para excluir um obejto, pega-se o objeto como já descrito e agita-se o copo, assim o objeto some de seu interior.

Toney e Thomas (2006) apresentam um estudo sobre os limites do alcance físico das pessoas em uma mesa que possua características tangíveis. O artigo argumenta que entender este alcance permite que os pesquisadores e *designers* produzam mesas mais utilizáveis para aplicações, abrangendo vários projetos de mesas inteligentes e interfaces de usuário otimamente dimensionadas para um determinado conjunto de usuários.

Várias experiências informais foram conduzidas, a fim de desenvolver a intuição sobre o alcance da mesa e confirmar o entendimento de ferramentas estruturais antropométricas – que mensuram o corpo humano ou uma de suas partes - existentes, indivíduos de pé foram instruídos a manipular objetos tão longe quanto possível, sob várias condições, posicionado peças brancas com a mão esquerda e peças pretas com a mão direita, o que resultou em dois semicírculos com uma grande intersecção e sobras individuais para cada lado, dando uma ideia de alcance possível. Os resultados foram comparados com indivíduos sentados e considerando as devidas proporções, os resultados foram considerados equivalentes. Desenvolvedores desejavam que usuários fossem capazes de produzir mais em um espaço

físico, e aproveitar algum espaço restante para a interação com colaboradores.

No trabalho de Boussemart e Giroux (2007), ocorre a descrição de uma interface tangível que simula uma cozinha inteligente, onde o objetivo é ajudar pessoas com déficit cognitivo. A simulação também é feita usando-se uma mesa, que representa um fogão responsivo ao usuário: dependendo das ações, e cartas que fazem o papel dos objetos da cozinha, há também alguns controles para uso de medidas, tais como quantidade, tempo e temperatura. A ideia é fazer esse tipo de interface direcionado para todos os cômodos de uma casa, e assistir pessoas em suas tarefas do dia-a-dia.

O sistema usa as cartas para simular objetos, esses objetos serão as entradas tangíveis do sistema, no exemplo dado no artigo, se o usuário quiser cozinhar arroz, ele pode falar sua intenção para o sistema, que o indicará a pegar uma panela, no caso da simulação: uma carta que represente uma panela. Em seguida o sistema diz ao usuário para colocar a panela no fogão (a carta em cima da mesa de simulação), e continua informando determinados passos até finalmente ter o arroz cozido.

A ideia principal não é dizer ao usuário tudo o que ele tem que fazer, mas prestar uma assistência. Há um tempo de espera entre as instruções que o sistema dá ao usuário, ou seja, o usuário só terá uma informação se não lembrar do que tem que fazer em seguida, mas se lembrar, poderá realizar a tarefa praticamente sozinho. Além do mais, o sistema pode ser personalizado, o que o torna mais amigável e mais preciso em ajudar necessidades individuais de usuários.

O sistema inForm, de Follmer et al. (2013), apresentou uma mesa com "bits tangíveis" que renderizam formas físicas em tempo real, usando os movimentos de mãos que são lidos por uma câmera. A mesa muda de forma dinâmica ou sob demanda dos estados do programa e o contexto do usuário. As diferentes alturas dos pinos formam as imagens assumidas pela mesa, que também pode mover os pinos para mudar ou segurar objetos. A intenção desse projeto é permitir a telepresença física de usuários, de modo que um usuário possa interagir com objetos sólidos sem está presente onde se encontra o objeto, a Figura 23 ilustra bem essa situação, mostrando um usuário remoto segurando uma lanterna por meio do inForm.

O sistema usa a câmera de profundidade Microsoft Kinect para rastrear as mãos dos usuários e também a superfície de pinos móveis, capturando o formato das mãos do usuário e em seguida medindo a profundidade dos pinos na mesa para movê-los de forma que adquiram formato semelhante a imagem que foi capturada das mãos.

Os formatos que o sistema inForm pode assumir são de duas formas, formatos pré-estabelecidos e formatos dinâmicos, ambos são renderizados fisicamente e provêm mecanismos para interação com usuário, além de permitir aplicações que utilize os dois modos simultaneamente. Dependendo da forma que os pinos do inFORM estejam renderizados,

eles podem reagir as ações do usuário, que pode puxa-los ou empurrá-los.

Se uma aplicação quer formar um botão na malha de pinos, ela levanta um dos pinos que pode ser ativado por um usuário e registrado com entrada binária para o sistema. A malha de pinos também pode mover objetos, rearranjando seus pinos em tempo real e fazendo-os acompanhar um objeto pela malha. A malha de pinos do inForm é capaz de aplicar força mecânica em um objeto e fazê-lo mover-se de vários modos. Isso expande as possibilidades de interação.

O sistema abre uma gama de possibilidades a partir do momento que considera não tão somente renderizar a presença física de um usuário, mas também formar ferramentas que possam ser utilizadas por usuário e que possam mudar de forma dinamicamente. Os autores ainda citam o fato da interface poder ser construída com outros tipos de matérias e abrir oportunidades de interação para a interação humano-computador.

Figura 23: inForm

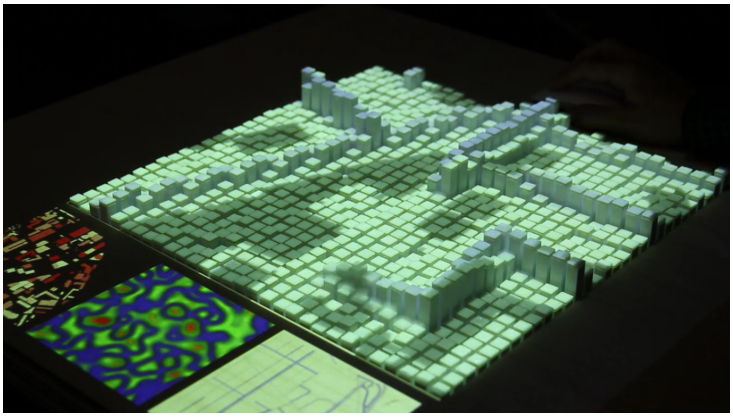
Fonte: MIT (2014)

Vários outros projetos foram desenvolvidos pelo Tangible Media Group do MIT, alguns deles correlacionados diretamente a ideia de "bits tangíveis", seguindo diretamente a linha apresentada pelo inForm.

O Tangible CityScape (Figura 24), por exemplo, é a proposta de um display

físico que representa a arquitetura tangível de uma cidade, visando o auxílio a políticas urbanas, planejamento de construções e uma melhor colaboração de especialistas através da interface tangível. Os usuários podem manipular representações físicas e digitais da cidade e interagir com parceiros remotos (MIT, 2014).

Figura 24: Tangible CityScape



Fonte: MIT (2014)

O Transform (Figura 25) proporciona usar os bits tangíveis para simular o comportamento físico de elementos encontrados na natureza, reagindo com a interação de um usuário, simulando vento, água e areia na natureza, é uma máquina dinâmica impulsionada por fluxo de dados e de energia (ISHII et al., 2015). O Transform foi usado também para simular uma mobília dinâmica, capaz de adaptar seu formato de acordo com o ambiente e a necessidade de seus usuários, moldando compartimentos para guardar objetos e auxiliando usuários em tarefas do dia-a-dia (VINK et al., 2015).

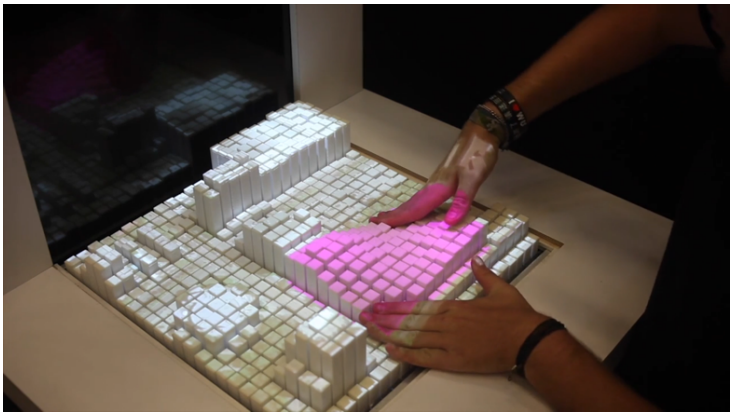
Figura 25: Transform

Fonte: Vink et al. (2015)

Materiable (Figura 26) vai mais além, molda interfaces físicas e simula as propriedades de materiais, se deformando ou aplicando resistências ao toque de um usuário.

Usando um protótipo de pinos como prova de conceito, o sistema pode criar propriedades variáveis computacionalmente de materiais deformáveis que são visualmente e fisicamente perceptíveis. Através de interação direta usuários podem perceber propriedades como flexibilidade, elasticidade e viscosidade dos materiais simulados (NAKAGAKI et al., 2016).

Figura 26: Materiable



Fonte: Nakagaki et al. (2016)

KinéPhone (Figura 27) explora como uma exibição a base de pinos atuados pode servir como uma plataforma sobre a qual é possível se construir instrumentos musicais e controladores. Foram projetados três instrumentos a partir do protótipo, e os pinos usados não serviam apenas como interface de entrada, mas também como uma fonte de som acústica (MIT, 2014).

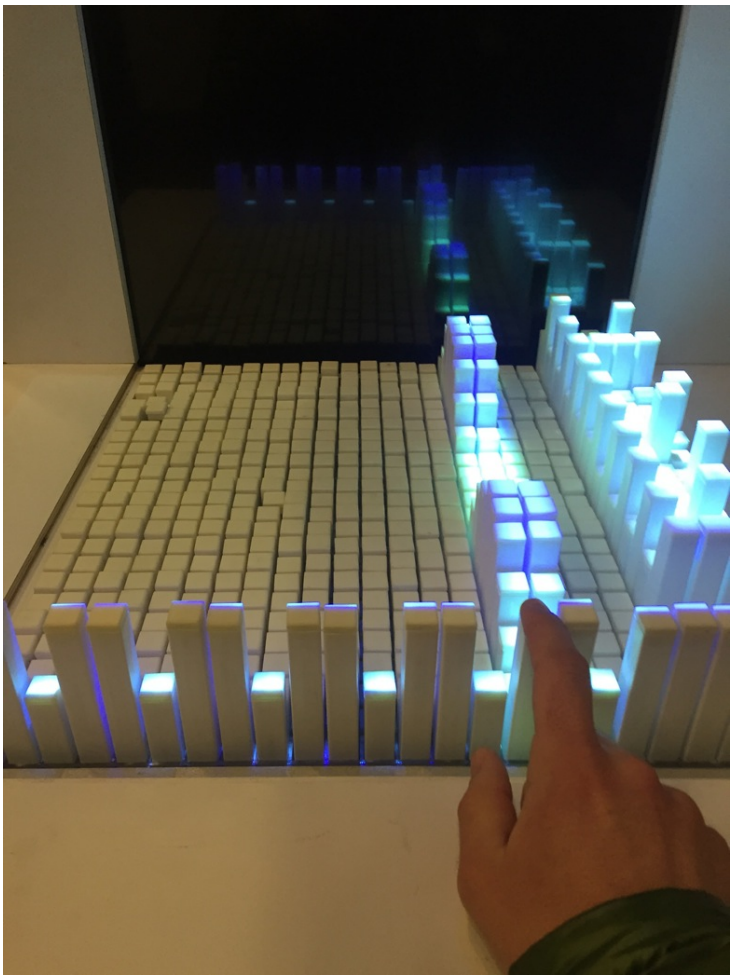
Figura 27: KinéPhone

Fonte: MIT (2014)

SoundForm (Figura 28) cria um novo método para compositores de música eletrônica de interagir com suas composições. Utilizando um display que muda de forma, formas de onda sintetizadas são projetadas em três dimensões em tempo real proporcionando a capacidade de ouvir, visualizar e interagir com o timbre das notas. Dois tipos de composição de música são explorados: geração de tons de oscilador, e desencadeamento de

amostras de áudio pré-gravadas. Por meio da utilização de gestos, o usuário pode tocar diretamente ou modificar as formas de onda sintetizadas (COLTER et al., 2016).

Figura 28: SoundForm

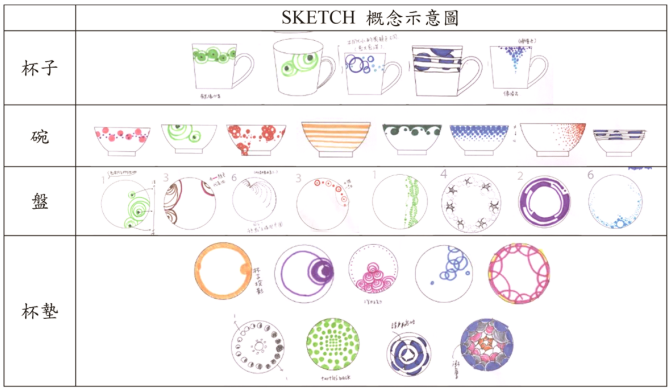


Fonte: Colter et al. (2016)

Wu et al. (2013) propõe usar câmeras que reconheçam os objetos usados no cotidiano como entradas, para controlar uma casa automatizada. A princípio, a câmera ficaria localizada em cima da mesa de jantar, e cada objeto serviria de controle para algum sistema, como de som ou de TV, inclusive tendo a capacidade de valorar os parâmetros dos equipamentos. Ao girar um copo, seria possível aumentar ou diminuir o volume do som, por exemplo, ou mudar o canal de TV. O sistema reconheceria padrões dos objetos, adaptando as escolhas, um exemplo disto seria perceber a presença de velas na mesa e ativar uma *playlist* de músicas românticas no som. Participantes de grupos focais contribuíram para esboçar alguns parâmetros de reconhecimento dos objetos, veja um desses esboços na Figura 29.

O estudo utiliza a tecnologia de reconhecimento de padrões e faz desses os componentes de interface. O design do padrão deve cumprir com requisitos determinados para que seja capaz de ser reconhecido por computadores, por outro lado tem que ser útil ao

Figura 29: Esboço de Padrões para Reconhecimento de Imagens



Fonte: WU et al. (2013)

usuário, assim, a metodologia usada foi a de design participativo, para que os padrões fossem escolhidos.

Por meio de testes, um estudo de caso em uma mesa de jantar com o sistema inteligente foi feito e os resultados foram satisfatórios. Existem ainda algumas limitações a serem superadas tais como a influência da iluminação nos objetos; o fato da identificação de tons de cinza serem mais rápidos que o reconhecimento do colorido; a preferência por objetos não brilhosos; as linhas de desenhos em objetos devem ser finas e o fato de considerar apenas partes de objetos, uma vez que fatores externos ao reconhecimento tais como os próprios usuários e a comida podem cobrir os objetos parcialmente.

Catala et al. (2011) faz o estudo sobre o impacto de plataformas tangíveis na educação, inovação e criatividade. Foram criadas duas plataformas capazes de realizar as mesmas tarefas, como montagens de blocos e estruturas para pequenos projetos, semelhante a brinquedos de montar. A primeira plataforma era totalmente tangível, sem nenhum suporte digital, a segunda (Figura 30) apresentava uma interface ainda tangível, mas dotada de uma tela digital onde os projetos eram montados.

Na plataforma totalmente tangível, era disponibilizado além dos blocos de montar, ferramentas para a junção desses blocos, tais como elásticos, parafusos, ganchos, porcas e parafusos. Era possível combinar vários blocos em articulações complexas, permitindo a construção de qualquer tipo de componentes fixos ou articulados com base na base rigidez dos objetos. Como esta plataforma é completamente tangível e física, fica sujeita às leis reais da física.

A plataforma digital se baseia numa superfície de uma tela sensível ao toque, onde os dedos podem ser usados para posicionar formas geométricas, aumenta-las, diminuí-las e movê-las, há também o uso de pequenos discos tangíveis, separados da tela, esses discos possuem identificadores que ao serem reconhecidos pela superfície da tela, o “transformam” em chaves de entrada, cada disco tem um significado diferente e são usados para construir e modificar o ambiente digital que se forma na tela.

Figura 30: Montagem de uma Tarefa na Plataforma Tangível/Digital

Fonte: CATALA et al. (2011)

Seguindo as considerações de design de experiência e devido a limitações de tempo, o teste com os usuários só foi realizado uma vez, com dois grupos de designação equivalente. Assim, cada grupo só interagiu com uma das plataformas. Gravações de vídeos foram analisados para extrair informações sobre desempenho em soluções de execução, a sua complexidade, padrões de comportamento e grau de colaboração.

Grupos diferentes de estudantes testaram ambas plataformas e foram obtidos alguns resultados: a abordagem digital apresentou menos soluções, no entanto as soluções apresentadas eram mais complexas, e em média, cerca de 47% do tempo foi conduzido em cooperação real quando usada a plataforma digital, a cooperação caiu para 28% no ambiente totalmente tangível.

Roh et al. (2010) mostra a implantação de um arranjo de motores em uma mesa, com a intenção de fornecer respostas com feedback ativo, ou seja, a mesa vibra em diferentes intensidades dependendo dos comandos ou da resposta mostrada. A intenção é fazer com que a interação se torne mais acreditável e o usuário tenha realmente a impressão de que realizou determinada tarefa, uma vez que sente as vibrações nos dedos.

A parte superior da interface é uma tela sensível ao toque, uma placa de acrílico é utilizada sobre a mesa. Sua largura é de 1,2 m, a sua altura é 0.9m, e sua espessura é de 1 centímetro. Um projetor e um IR (Infra-Red ou infravermelho) com uma câmera são posicionados sob o tampo da mesa. O projetor exibe imagens na tela e a câmara reconhece pontas dos dedos na tela.

Motores são usados para gerar feedbacks táteis. O s motores têm pêndulos desequilibrados que podem criar vibrações. A inconsistência entre o centro de massa do pêndulo e o eixo de rotação cria força centrífuga, sendo assim a fonte de geração de vibrações do motor.

Se um usuário toca a superfície da mesa, o evento de toque é gerado a partir de sinais sonoros que são enviados ao sistema, que por sua vez aciona os motores com base em sinais de controle. Foram posicionados seis motores nas laterais da mesa para gerar o feedback com intensidades diferentes em toda superfície, de acordo com a velocidade definida em cada motor.

3.2 Projetos Tangíveis Usando Arduino

O Arduino é uma placa de prototipagem eletrônica de hardware livre que possui um microcontrolador, suporte para controle de entrada/saída analógica e digital (E/S programável) e linguagem de programação baseada em C. Também possui vários acessórios que são aplicados à placa principal, para simular diferentes ambientes, tais como sensores, buzinas e luzes de LED (*Light Emitting Diode*). Sendo assim, é plenamente possível projetar o dispositivo do presente trabalho usando esta plataforma, logo foram pesquisados alguns trabalhos que já usaram o Arduino em seu desenvolvimento.

Jiaqi, Santoso e Gook (2013) descrevem a realidade misturada, formada pelas camadas de mundo real, realidade aumentada, realidade virtual e ambiente virtual, assim, foi desenvolvido um simulador de condução de carros para crianças. O simulador captura os movimentos de uma criança usando o *kinect* (plataforma que captura movimentos através de câmeras), a interface do sistema foi feita em *flash* e usa *TinkerProxy*, um software que permite a aplicação de flash acessar portas seriais em um computador, para enviar os dados em para uma placa Arduino.

Por meio do *Xbee*, um sensor wi-fi compatível com Arduino, os comandos chegam ao arduino e controlam os motores de um carrinho, ou movimenta a câmera do carrinho, no caso uma câmera de *Iphone*. Ainda são usados cartões de realidade aumentada, que uma vez identificados pela câmera do celular, representam obstáculos ou placas de sinalização na tela do usuário, completando assim a simulação com a mistura de realidades. A Figura 31 mostra a visão que se tem na câmera instalada no carrinho.

Depois montado o ambiente físico, com as devidas conexões feitas com o minicarro, o arduino e o Kinect, é necessário a integração com diversos softwares, como principal foi usado o Adobe Flash, plataforma multimídia e software usado para criação de gráficos vetoriais, animação, jogos e aplicações para internet. É preciso garantir que o Kinect SDK já esteja instalado no PC, e em seguida instalar a biblioteca AIRKinect, que combina o Kinect com o flash, além do já citado *TinkerProxy*, responsável pela conexão com a placa Arduino presente no carro. A câmera usada é de um Iphone e utiliza um aplicativo chamado “webcamera”, disponível na loja do aparelho celular.

O resultado é a imagem capturada da câmera do *smartphone* sendo mostrada no monitor de um PC, e os movimentos capturados pelo Kinect serão usados para controlar o

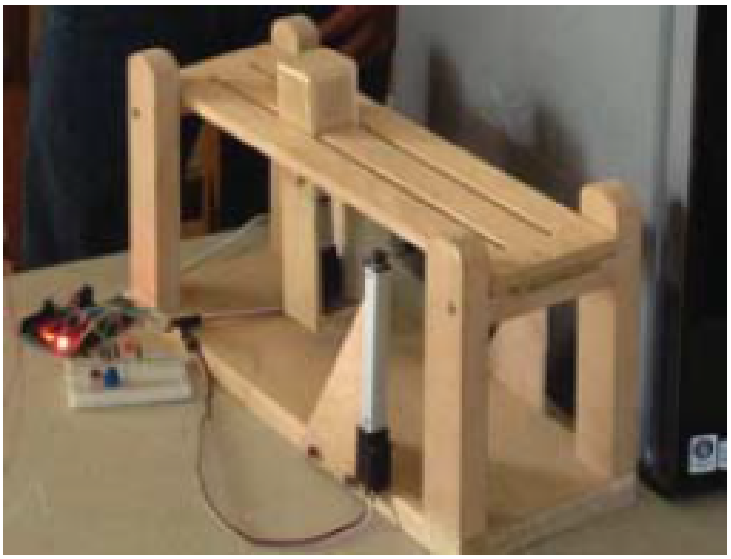
Figura 31: Simulador de Direção e Sinalização

Fonte: JIAQI; SANTOSO; GOOK (2013)

carrinho.

Garcia-Canseco et al. (2013) usam Arduino para montar estruturas conhecidas na física, tal como um plano inclinado (Figura 32) e uma alavanca. O objetivo é ensinar física fazendo o uso de objetos tangíveis, para que os alunos observem os resultados práticos de equações. Além dos sensores e atuadores conectados com o Arduino, os autores implementam as soluções de interface com o suporte de *Processing* e *GTK+toolkit*.

Figura 32: Dispositivo do Plano Inclinado

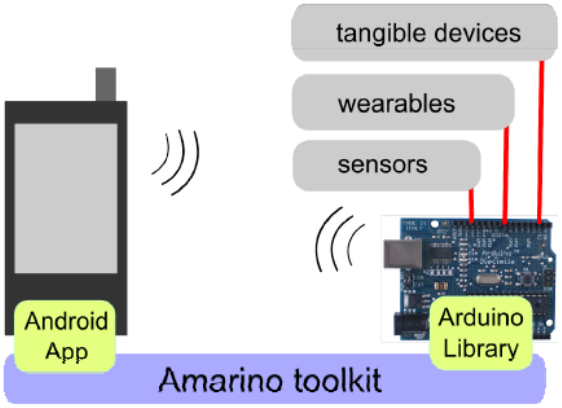


Fonte: GARCIA-CANSECO et al. (2013)

Armarino, um *toolkit* para a prototipagem rápida de computação ubíqua móvel, permite a rápida criação de protótipos ligando o sistema operacional Android com o

microcontrolador Arduino, fazendo com que o dispositivo móvel sirva de interface para controlar a aplicação. Além do Android, o *toolkit* usa uma biblioteca de software para o Arduino e um protocolo de comunicação via *bluetooth*, o que torna o telefone um dispositivo de entrada pronto para o uso, sem que nenhuma programação extra seja feita, permitindo ainda, usar eventos do Android como entrada, tais como acelerômetro do telefone (KAUFMANN; BUECHLEY, 2010). A Figura 33 mostra a arquitetura da aplicação.

Figura 33: Armarino



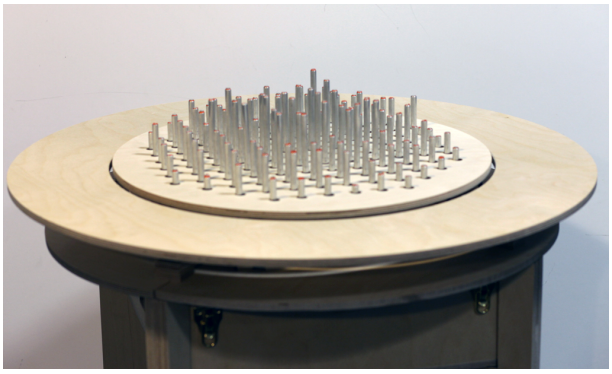
Fonte: KAUFMANN; BUECHLEY (2010)

O projeto "Relief" (LEITHINGER; ISHII, 2010), reúne características similares ao que está sendo proposto neste trabalho. Foi construída uma mesa com pinos que funcionam tanto como entrada quanto como saída (Figura 34), e são capazes de plotarem gráficos referentes a formas de terrenos geográficos, tais como montanhas e depressões. Em seu dispositivo, os autores usaram motores DC ligados em placas *shield*, que expandem a capacidade das placas Arduino, para que mais motores possam ser plugados. Já a aplicação era executada no próprio microcontrolador do Arduino.

O hardware atual interage com aplicações *open source* implementadas no ambiente de desenvolvimento *Processing*, principalmente pela facilidade de uso, portabilidade e o fato de ser livre de custos. Cada placa Arduino executa um programa para detectar a posição dos pinos e controlar o motor ligado para alcançar uma posição desejada. A posição do pino é enviada para uma aplicação em execução em um computador, que mantém o controle da correlação entre todas as posições de pinos e o modelo 3D. O mesmo programa também gera gráficos, que são projetados de volta para a mesa e exibem alguma forma.

Relief é a primeira versão de um *display* de formas com atuadores, e a plataforma de hardware se mostra plenamente escalável, pois possui módulos de acionamento são individuais, funcionando de forma independente, ou seja, novos módulos poderão ser inseridos futuramente.

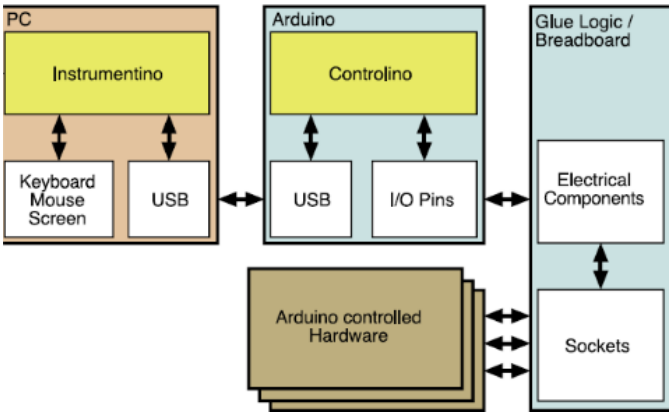
Figura 34: Relief



Fonte: LEITHINGER; ISHII (2010)

Em Koenka, Sáiz e Hauser (2014), é citado que a saída mais comum e direta para controlar um Arduino é a serial, no entanto essa saída é limitada a comandos textuais, e uma abordagem mais sofisticada seria usar uma GUI que fosse executada em um PC e mandasse comandos para o Arduino. O problema é que o Arduino possui um déficit de memória e poder de processamento, e esse método necessita de uma customização no programa que é executado dentro do microcontrolador. Assim, os autores desenvolveram *Controlino*, um programa escravo que fica em execução dentro do microcontrolador do Arduino esperando as entradas provenientes de uma aplicação, construída com a API *Instrumentino*, que provê uma gama de comandos necessários para o desenvolvimento de uma interface de aplicação personalizada. A Figura 35 mostra o fluxo de dados na arquitetura do *Instrumentino*.

Figura 35: Fluxo de Dados De um Sistema usando *Instrumentino*



Fonte: KOENKA; SáIZ; HAUSER (2014)

4 Desenvolvimento do Protótipo

O dispositivo descrito neste trabalho propõe o conceito de “pixel tangível”, inspirado em trabalhos do Tangible Media Group do MIT, que criou a ideia de “bits tangíveis” na conferência CHI de ‘97.

A ideia de “bits tangíveis” levou a trabalhos como o sistema inForm (FOLLMER et al., 2013), citado na seção anterior, permitir a telepresença física, usando a interface tangível como representação física de um usuário remoto; e o projeto Relief (LEITHINGER; ISHII, 2010), também mostrado em trabalhos relacionados e se direciona para a interação com representações de superfícies terrestres.

Enquanto os trabalhos do MIT, citados acima, possuem aplicações específicas relacionadas a “bits tangíveis”, este trabalho apresenta os “pixels tangíveis”, propondo uma mesa composta por pinos móveis, que servem tanto como entrada quanto como saída de dados. Cada pixel tangível é representado por um desses pinos, que podem variar em altura e cor.

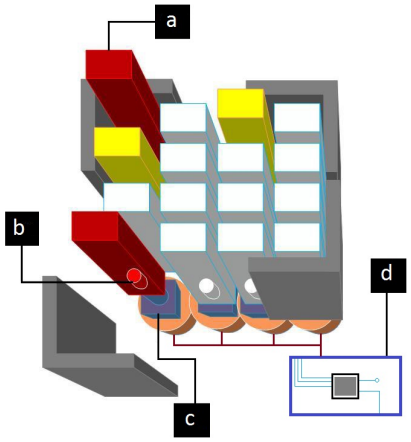
O diferencial do dispositivo em questão é ser de propósito geral, baseando-se no fato de que suas funções de entrada e saída podem ser programadas, ou seja, terá um propósito tal como seja a funcionalidade de uma aplicação. Pode-se fazer analogia a um monitor “touch screen” de computador, que também é um dispositivo de propósito geral, sendo praticamente indispensável para mostrar informações a usuários nos dias de hoje, logo, a interface apresentada também uma nova maneira de representação e interação com dados e imagens.

O trabalho também disponibilizada uma API que auxilia na criação das aplicações. Usuários programadores podem escrever diversos programas para o dispositivo, que é capaz de representar letras, números e formas, e assim, “imprimir” imagens físicas que representam informação, ao mesmo tempo que oferece iteração do usuário, pois a informação representada se torna a interface de entrada.

A característica tangível do dispositivo se apresenta quando a imagem impressa no dispositivo é capaz de ser manipulada, e se torna ela mesma a interface de entrada de dados do usuário, ou seja, o usuário poderá modificar diretamente a informação mostrada a ele. A Figura 36 exemplifica a ideia do dispositivo, mostrando o esboço inicial do projeto.

Cada “pixel tangível” - Figura 36 (a) - representado por um pino, precisa de um motor independente - Figura 36 (c) - para que seja possível o movimento que regula sua altura, enquanto as cores das imagens são formadas por luzes - Figura 36 (b) - dentro dos pinos. Assim, os pixels serão capazes de imprimir o resultado de um programa, desenhando

Figura 36: Esboço do Dispositivo Proposto.

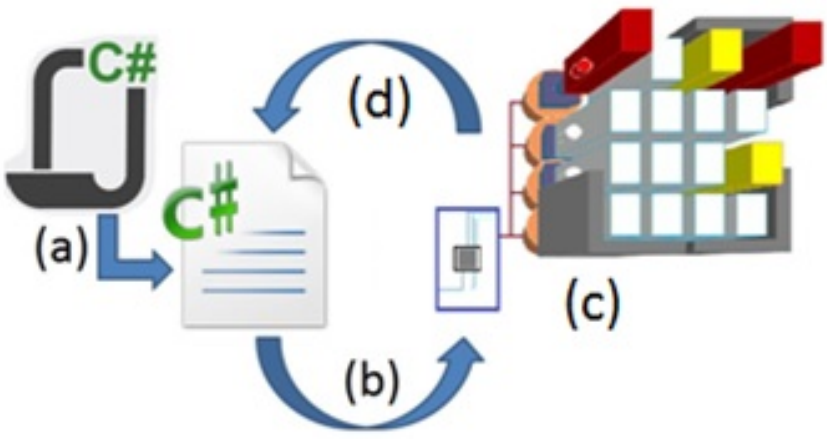


Fonte: Autoria Própria

imagens que variam altura dos pinos e suas cores, executando sua função como dispositivo de saída de dados. Além do mais, há um sensor - Figura 36 (c) - capaz de identificar o toque de um usuário, que por sua vez, confere ao dispositivo a capacidade de entrada de dados. Por fim, é necessário que haja um controlador - Figura 36 (d)- para coordenar toda operação de movimento dos motores e mudanças de cor.

O fluxo de informação que acontece no desenvolvimento/execução de uma aplicação feita por um usuário é mostrado na Figura 37: o programador escreve seu código chamando as funções específicas do dispositivo, presentes em uma biblioteca (a); durante a execução, o aplicativo envia os comandos para o microcontrolador do dispositivo (b), que por sua vez, fará a impressão do resultado nos pixels tangíveis (c); quando o usuário pressiona ou puxa um dos pixels, o valor de entrada é enviado ao programa (d), que pode disparar um evento no dispositivo ou em outras variáveis do programa.

Figura 37: Fluxo de Informações.



Fonte: Autoria Própria

A programação do dispositivo foi feita usando a linguagem natural da placa

Arduino chamada "Wire". Presentes neste código se encontram as funções mais básicas do dispositivo, tais como posicionar um pixel em uma determinada altura e indicar a cor que este irá assumir, existindo também uma função que sempre retorna os valores das alturas dos pixels.

Uma vez que o programa esteja carregado no dispositivo, ele permanecerá de prontidão, sendo escravo de aplicações externas, que serão escritas baseadas em uma biblioteca, desenvolvida em conjunto com o dispositivo, de funções em C#. Nesta biblioteca estão disponíveis funções que modificam um conjunto de pixels tangíveis, como a impressão de caracteres e valores binários, como também funções de manipulação individual desses pixels. Tais funções são usadas como prova de conceito e estão limitadas pela capacidade de representação de informação da pequena quantidade de pixels do protótipo, mas poderá ser estendida facilmente quando a malha de pixels for aumentada.

A razão pela qual a linguagem C# foi escolhida é que algumas funções nativas dessa linguagem permitem o uso facilitado da porta serial, como por exemplo, a componente *"Port"*, melhorando o fluxo de dados entre um PC e o microcontrolador do dispositivo. As funções nativas de C# poderão ser chamadas normalmente nas aplicações de um usuário.

4.1 Metodologia

Como descrito na Seção 2.3, o desenvolvimento do sistema embarcado, o protótipo a qual chegou-se neste trabalho, começa pela análise de requisitos deste sistema, então um conjunto básico de requisitos é escrito:

- O protótipo deve ser capaz de subir e descer seus pinos e fazê-los mudar de cor.
- O protótipo deve receber dados do usuário por meio de seus pinos.
- O protótipo deve responder em tempo real a comandos de aplicações.
- O protótipo deve estar apto a executar aplicações diversas.

Pode-se colocar os requisitos como no esquema da Figura 38:

Em seguida tem-se que especificar o que o sistema faz, baseando-se nos requisitos pode-se dizer que o sistema irá:

- Mostrar imagens tangíveis em seu dispositivo tangível, variando altura e cor de cada um de seus pinos;
- Receber e processar o toque do usuário de acordo com a aplicação;
- Executar aplicações diversas de usuários, a partir de um PC.

Figura 38: Requisitos em Termos de Engenharia.

Nome	Projeto 'Force The Light'
Propósito	Ser usado como dispositivo de entrada e saída tangível
Entradas	Pinos sensíveis ao toque de usuário, entrada serial para carregamento de programas
Saídas	Pinos luminosos que formam figuras, saída serial para informações sobre as posições dos pinos
Funções	Mostrar imagens tangíveis que podem ser manipuladas
Atuação	Pode ser carregado com aplicações que manipulem seus pinos, os pinos se movem e mudam de cor um de cada vez, até formarem uma imagem
Custo de Fabricação	R$ 1200
Consumo de Energia	12 V
Dimensões e Peso	O objeto deve ter dimensões de maneira que seja possível de ser pego com as mãos, não deve pesar mais de 1 kg

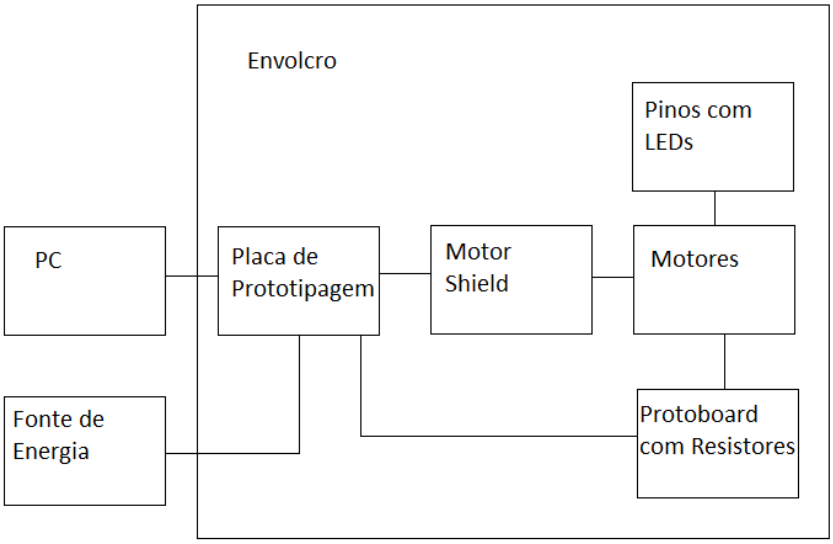
Fonte: Autoria Própria

Algumas dessas etapas podem ser feitas em paralelo ou em ordens diferentes de execução, pode-se, por exemplo, listar componentes para que os requisitos sejam atendidos e o sistema seja construído, antes de esquematizar a arquitetura, essa fase descreve os componentes de hardware e software que serão usados.

- Placa de prototipagem com microcontrolador;
- Placa controladora de motores;
- Cabos;
- Motores;
- Luzes de LED;
- Protoboard;
- Resistores;
- Cabo USB;
- Fonte de 12 V;
- PC;
- Papelão;
- Pinos de acrílico;
- Biblioteca de Software com funções para o dispositivo;
- API para desenvolvimento de programas;
- Código do microcontrolador.

A seguir é feito um plano arquitetural de como os componentes especificados serão montados, um diagrama de blocos pode ser usado para tal, e a Figura 39 mostra como ficou a arquitetura do dispositivo deste trabalho.

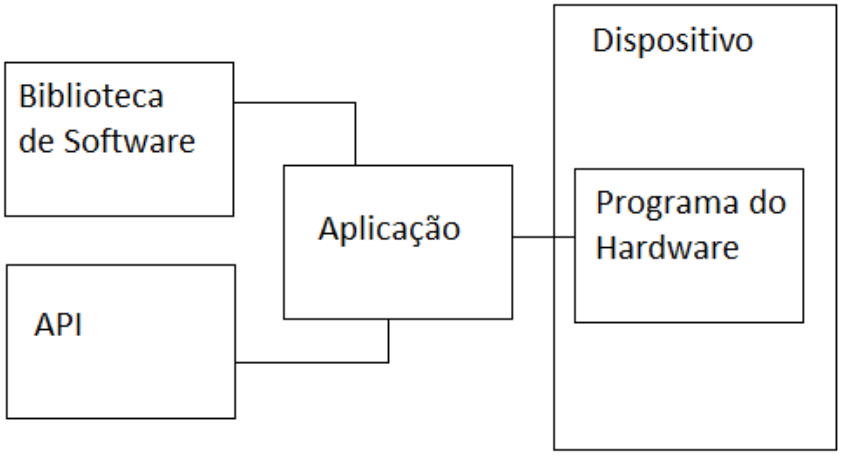
Figura 39: Arquitetura em Diagrama de Blocos.



Fonte: Autoria Própria

Após esta definição inicial, agora será mostrado na Figura 40 um diagrama de blocos contendo a parte de software, poderia ser feito também outro diagrama contendo uma parte de hardware mais especifica, considerando memória, CPU, buffers, etc. Porém, como o PC já possui uma arquitetura própria, o PC inteiro é tomado como um único componente.

Figura 40: Arquitetura de Software em Diagrama de Blocos.



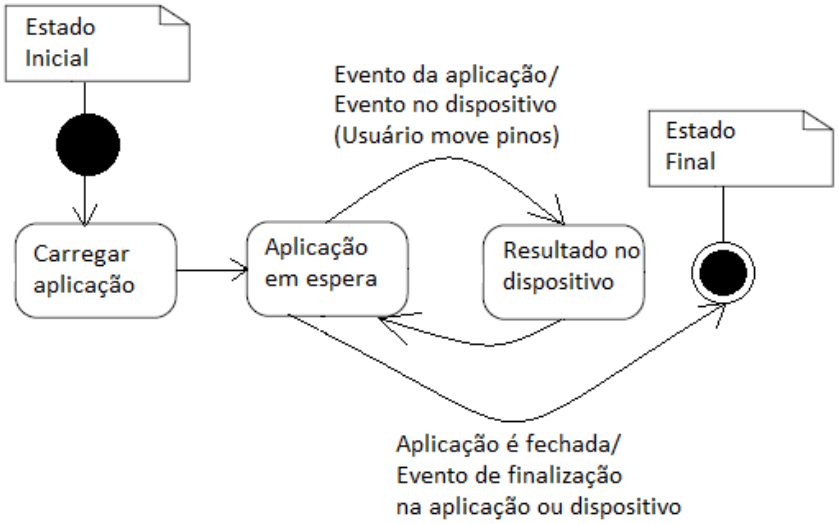
Fonte: Autoria Própria

Por fim, usa-se os componentes e monta-se o protótipo baseado na arquitetura descrita, a biblioteca de software, API e programa do microcontrolador são implementados

e o dispositivo é testado, bugs são descobertos e todas as modificações necessárias para que o dispositivo funcione corretamente são feitas, se foi usado por exemplo, um tipo de fio ou resistor que não funcionou corretamente, estes serão trocados.

Uma vez o sistema montado, pode-se definir uma sequência de passos que marcará a sua utilização, o dispositivo é projetado para executar programas variados, de acordo com as ideias dos programadores, porém todos seguirão um padrão básico, mostrado na Figura 41 que descreve a máquina de estados do dispositivo.

Figura 41: Máquina de Estados em UML.



Fonte: Autoria Própria

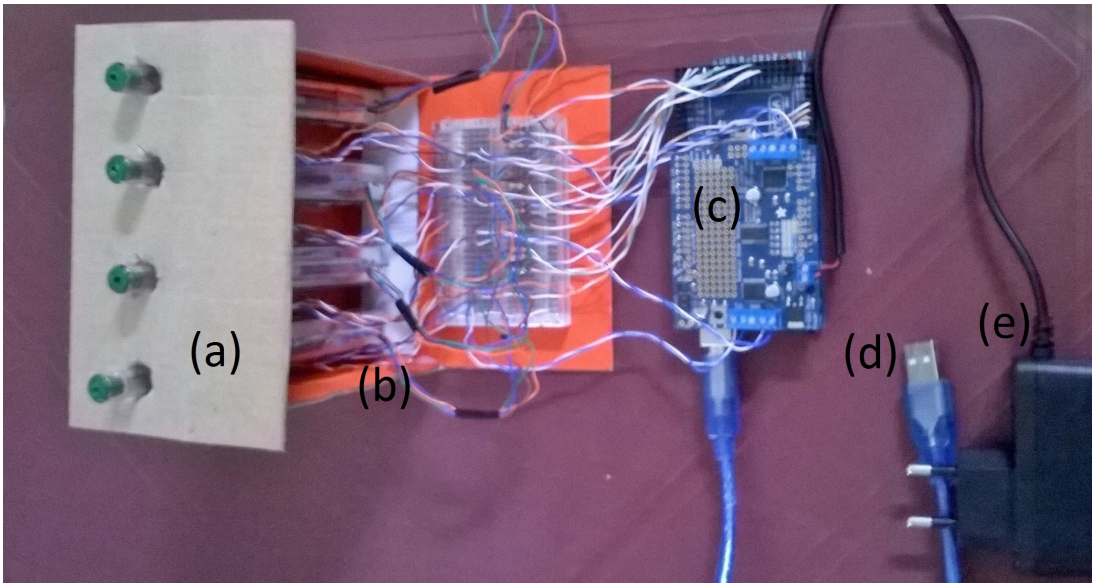
O estado inicial e final são estados especiais que ajudam a organizar o fluxo da máquina de estados. Os estados representam diferentes operações conceituais. Em alguns casos é necessário a ação de um usuário para alcançar o próximo estado, em outros apenas um intervalo de tempo é necessário (WOLF, 2001), sendo que quem irá definir tais regras é o programador, motivo desta máquina de estados se mostrar genérica o suficiente para quaisquer aplicações.

4.2 Hardware

A Figura 42 apresenta o protótipo desenvolvido neste trabalho, constituído de quatro pixels tangíveis (a). Dentro de cada pixel do dispositivo, há uma luz de LED RGB, para que o mesmo possa mudar de cor, e para que haja o movimento, foi usado um potenciômetro deslizante equipado com um motor DC (b), comumente utilizado em mesas de mixagem de áudio. Para controlar os pixels tangíveis, foi usada uma placa de prototipagem Arduino Mega (ARDUINO, 2016), e anexada a esta, uma placa controladora de motor Adafruit (ADAFRUIT, 2016) (c). A comunicação do dispositivo com uma

aplicação se dá através da uma entrada serial USB (d) e para alimentar os motores se faz necessário o uso de uma fonte externa.

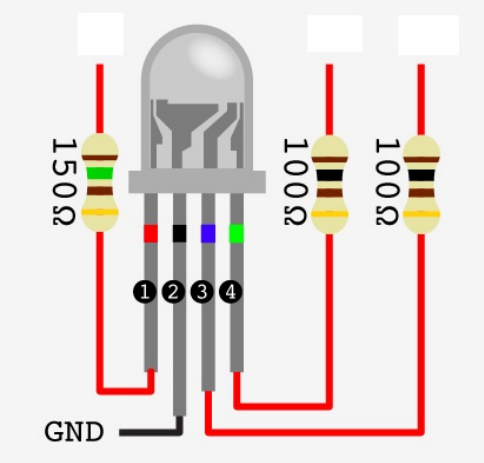
Figura 42: Dispositivo Tangível de Entrada e Saída - Visão de Seus Componentes.



Fonte: Autoria Própria.

Por se tratar de um protótipo, o material usado na construição do dispositivo foi de baixo custo, os pixels são formados por canetas transparentes, com os LED RGB por dentro, cada LED necessita de três correntes com resistores para ter a voltagem correta de cada cor, como mostra a Figura 43, além de um um fio com aterramento, isso baseado na voltagem de uma placa arduino, que possui 5 V em suas saídas de energia.

Figura 43: Resistências Necessárias a um LED RGB.



Fonte: Engineering (2016)

Os resistores e o aterramento ficam instalados em uma *protoboard* que intermedia a conexão dos fios entre o Arduino e os pixels, os fios usados foram retirados de cabos de rede. Os pixels são então acoplados aos motores, tendo uma caixa de papelão como envolcro.

O motor escolhido para compor o projeto foi do modelo ALPS RSA0N11M9A0K (Figura 44) (ALPS, 2016). Trata-se de um motor DC anexado a um cursor deslizante, que se move para frente ou para trás (ou cima e baixo dependendo da posição em que se encontre), por meio de uma roldana e também vem equipado com um potenciômetro, que muda o valor de sua resistência também baseado no movimentno da roldana. A razão para esta escolha veio do fato deste motor já ter sido usado em projetos semelhantes, como o inForm (FOLLMER et al., 2013), pois além de sua estrutura já permitir o movimento que se quer ter dos pinos do dispositivo, o potenciômetro é usado para fazer a leitura da posição em que o pino se encontra.

Figura 44: Motor do Tipo ALPS RSA0N11M9A0K

Fonte: ALPS (2016)

Um potenciometro é uma resistência ligada a dois terminais que pode ser ajustada por intermédio de um cursor. Dependendo da posição do cursor essa resistência pode variar de 0 (zero) até a tensão total da corrente disponível no circuito. A leitura dessa tensão feita por uma placa Arduino é convertida em valores absolutos de 0 até 1023, valores que representam o mínimo e máximo da tensão, respectivamente. Com essa variação de valores ocorrendo quando se movimenta os pinos, é possível ler a posição destes a cada momento e usar como entrada para o dispositivo.

Para ligar um motor do tipo ALPS RSA0N11M9A0K se faz necessário o uso de seis fios: três para o motor DC, incluindo dois com corrente (necessário para haver a mudança de sentido do giro do motor, dependendo de qual dos dois fios está sendo ligado) e um terra, e três para o potenciômetro: corrente, terra e resistência. Mais detalhes e especificações sobre esse motor são encontradas no Anexo 1.

Como a placa Arduino usada possui um número limitado de entradas, ocupa boa parte delas com as conexões feitas usando os LEDs RGB e também não possui voltagem suficiente para controlar vários motores ao mesmo tempo, se faz necessário o uso de uma expansão da mesma, usando-se uma placa controladora de motores. Foi usada então uma

placa Adafruit Motor Shield (ADAFRUIT, 2016), já citada anteriormente na seção 2.6, e ligados a ela quatro motores e uma fonte externa de 12 V. Seria possível usar apenas a voltagem nativa do arduino para mover os motores, porém os movimentos seriam lentos e sem força, assim se faz necessário uma tensão maior capaz de levantar o peso dos pinos em uma velocidade razoável.

Para controlar os pinos ou pixels tangíveis foi usada uma placa de prototipagem Arduino Mega (ARDUINO, 2016), escolhida por ser acessível no mercado, ter programação relativamente fácil e possuir suporte de uma grande comunidade. Baseada no microcontrolador ATMEGA2560, a Arduino executa um programa interno que controla as funções básicas do dispositivo, enviando comandos a placa de motores, lendo os valores dos potênciometros e indicando quais cores dos LEDs serão acesas. No programa interno a placa Arduino possui todas as instruções necessárias para se controlar o dispositivo, mas são as aplicações externas que solicitam as ações que serão efetuadas. A comunicação entre a plataforma e as aplicações, que são executadas em um PC, se dá através de comunicação serial USB, que também proporciona os 5 V necessários ao funcionamento da Arduino, lembrando que a fonte externa de 12 V é apenas para uso dos motores, por meio da placa Adafruit.

4.3 Software

Como visto anteriormente, a programação do dipositivo é feita no microcontrolador, neste caso no ATMEGA2560 da placa arduino. Não é necessário escrever o código do microcontrolador em Assembly ou em outra linguagem de baixo nível, a Arduino possui uma interface própria de programação composta por uma IDE *(Integrated Development Environment)* para desktop e uma linguagem baseada em C, chamada "Wire".

Usando a IDE da Arduino em ambiente Windows, o código do dispositivo foi escrito e instalado no microcontroldor via porta serial USB, de tal maneira que uma vez instalado, execute as funções do dispositivo quando receber um comando de controle, pela mesma porta.

Os comandos de controle foram convencionados em sequências de cinco dígitos, criados baseando-se no que o dispositivo pode fazer. Para compor os comandos, primeiro indica-se qual dos motores realizará alguma ação, depois determina-se a altura e a cor do pixel. Foi definida uma escala para as alturas dos pixels, dividindo-se o valor total que pode ser lido de um potenciômetro por dez, ou seja 1024(de 0 a 1023)/10, assim, fica-se com dez possíveis posições (de 0 a 9) que um pixel pode assumir.

Os LEDs RGB podem formar várias cores, variando a tensão aplicada em mais de um pino do LED (*red, green* ou *blue*) ao mesmo tempo. Por convenção foram utilizadas as três cores básicas (Vermelho, verde e azul) mais algumas resultantes de mais de uma

cor ligada ao mesmo tempo (amarelo, roxo, branco) e sem cor (sem energia). A tabela 1 mostra como definir um código para o programa do microcontrolador, por exemplo: '01R15' resulta em o primeiro pixel ficar vermelho e na altura '5'; '03W19' resulta em o terceiro pixel ficar branco e na altura '9'. O código completo usado na Arduino se encontra no Apêndice 1.

Tabela 1: Códigos dos Comandos de Controle

Motor	**Cor**	**Altura**
01	B2 (apagado)	0
02	R1 (vermelho)	1
03	G1 (verde)	2
04	B1 (azul)	3
(não há mais motores)	P1 (roxo)	4
	Y1 (amarelo)	5
	W1 (branco)	6
		...
		9

Fonte: Autoria Própria.

Presente no código da Arduino também se encontra uma função que retorna, quando requisitada, a posição de cada um dos potenciômetros, enviando estes dados de volta pela mesma porta serial.

Para a programação em alto nível, a linguagem C# foi escolhida, devido à presença de uma componente nativa desta linguagem chamada "Port", que facilmente estabelece uma comunicação serial com o dispositivo e possui funções de leitura e escrita. Assim, pode-se criar aplicações em C# que envie os comandos de controle para o dispositivo, e trate os dados de saída que o dispositivo enviar de volta.

Para que uma "imagem" seja mostrada no dispositivo é necessário que uma aplicação envie uma sequência de quatro comandos, um para cada pixel do protótipo, resultando em uma cadeia com vinte caracteres (cinco para cada código). Foi criada então, uma biblioteca que possui várias representações de símbolos que podem ser mostrados no dispositivo, assim, um programador usa essas funções em uma aplicação de alto nível, sem se preocupar com os códigos que serão enviados ao microcontrolador.

As funções específicas da biblioteca, que aqui será chamada de "Lightforce", são análogas as que se encontram inseridas no programa da placa Arduino, o que faz a função do microcontrolador se equiparar a função de um *driver* de dispositivo, isto é, cria uma interface para o programa em alto nível executar ações em baixo nível de forma indireta. A "Lightforce" possui em sua classe "Lightining" as seguintes rotinas:

Lightining.draw_Pin(int id_pin, int colour, int pos) - Indica o pixel, a cor e a altura que este deve assumir, por exemplo: se os parâmetros forem indicados como Light.draw_Pin(3, 2, 4), significa que o pino 3 ficará verde (1 - vermelho, 2 - verde, etc) e

subirá até a altura 5 (relativa as alturas que o pixel pode assumir). Esta função permite ao programador criar seus próprios resultados, ou sua representação, de maneira dinâmica, no dispositivo tangível. A tabela 2, a seguir, indica os valores e resultados que esses parâmetros podem apresentar.

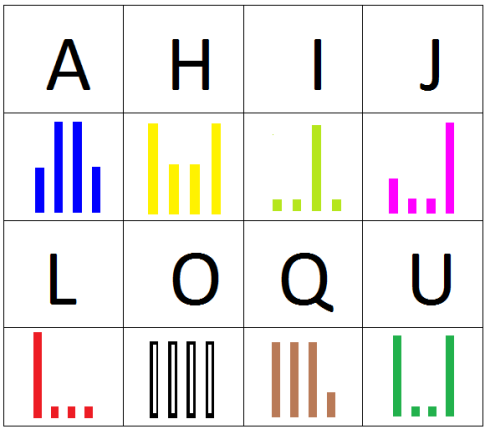
Tabela 2: Parâmetros da função Lightining.draw_Pin()

Id_Pin	**Colour**	**Pos**
'0' - pino 1	'0' - sem cor	'0' - posição 1
'1' - pino 2	'1' - vermelho	'1' - posição 2
'2' - pino 3	'2' - verde	'2' - posição 3
'3' - pino 4	'3' - azul	'3' - posição 4
—	'4' - roxo	'4' - posição 5
—	'5' - amarelo	'5' - posição 6
—	'6' - marrom	'6' - posição 7
—	'7' - branco	'7' - posição 8
—	—	'8' - posição 9
—	—	'9' - posição 10

Fonte: Autoria Própria.

Lightining.print_Symbol(char sym) - Imprime algum símbolo que seja passado como parâmetro. Se for passado o parâmetro 'A' por exemplo, o dispositivo mostrará sua impressão equivalente da letra 'A', e assim o raciocínio se segue com outros caracteres. A Figura 45 mostra alguns exemplos de resultados do dispositivo em questão, no caso a representação de algumas letras, gráficos em barras ou outros símbolos, que o usuário desejar, também podem ser impressos.

Figura 45: Forma Equivalente de Letras dos Pixels tangíveis.

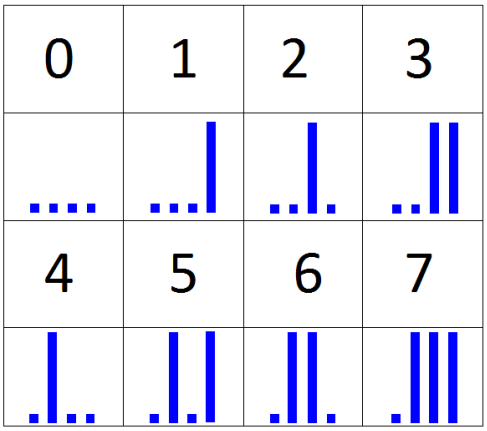


Fonte: Autoria Própria

Lightining.print_Bin(int bin) - Passando um valor inteiro decimal como parâmetro, o dispositivo imprime um valor Binário. Quando o pino está "abaixado" seu valor é '0' e quando está "levantado" seu valor é '1', as diferentes posições dos pinos foram as sequencias de 0s e 1s dos valores binários. A Figura 46 mostra esta forma de representar

valores binários, no caso as representações chegam até o valor 7 (0111), mas como dispõe-se de 4 pinos, o valor, usando esta forma de representação, pode chegar até valor 15 (1111), isto porque outras formas de representação podem ser usadas pelos programadores, mudar a cor de um pino para indicar que o valor seja negativo, ou dobrar o valor usando alturas intermediárias.

Figura 46: Forma Equivalente de Binários dos Pixels tangíveis.



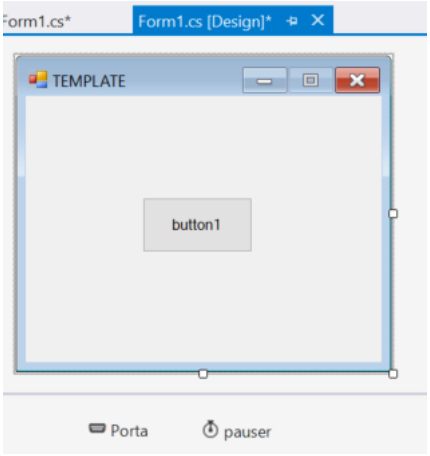
Fonte: Autoria Própria

O código da "Lightforce" se encontra no Apêndice 2, e pode também ser modificado por um programador que deseje inserir suas próprias funções a biblioteca, como outros tipos de representção para letras ou formas geométricas.

Além da biblioteca, foi escrito um template em C#, que inicializa, além de *"Port"*, uma outra componente chamada *"Pauser"*, permitindo assim o uso da função *"Pauser_Tick()"*, que funciona como um relógio, e chama a função de leitura de *"Port"* a cada momento, o que resulta no programa em execução sempre ter atualizados os valores dos potenciômetros. Os programas são escritos modificando-se o código do template, usando as funcões da biblioteca de acordo com o intuito do programador. A Figura 47 mostra o formulário base do template, o código do formulário se encontra no Apêndice 3.

Quando o programador quer mostrar um símbolo no dispositivo ele chama uma função "Lightining.Print_Bin()", por exemplo, em seguida, a função de escrita da classe *"Port"* do C# é chamada, e por sua vez, envia os dados pela porta serial até o dispositivo, que imprime o que foi requisitado. Se o programador desejar usar os valores de entrada que o dispositivo envia de volta, ele pode fazer testes comparando os valores de altura atuais dos pixels com os valores de um momento anterior, identificando assim o movimento que ocorreu nos pixels.

Figura 47: Formulário Base do Template.



Fonte: Autoria Própria

4.4 Avaliação

Teste com usuário é um método fundamental de usabilidade. Gerentes de desenvolvimento estão percebendo que investir neste tipo de teste cria um poderoso incentivo para o término da fase de design. Resultados práticos demonstram que os testes de usabilidade além de acelerarem projetos, produzem uma significativa redução em seus custos (ROCHA; BARANAUSKAS, 2003).

Como se quer uma visão mais global da interface, geralmente se utilizam testes que deem medidas de performance (ROCHA; BARANAUSKAS, 2003). Em uma avaliação que meça a usabilidade de uma interface, normalmente é necessário fornecer pelo menos uma medida para cada um dos seguintes pontos: eficácia, eficiência e satisfação (FILARDI; TRAINA, 2008).

As medidas de eficácia estão relacionadas às medidas de acurácia e completude, onde a acurácia mede a quantidade de erros da interface durante seu funcionamento e a completude mede a quantidade de tarefas realizadas com sucesso; as medidas de eficiência referem-se ao nível de eficácia alcançado com os recursos usados, como esforço mental ou físico, e as medidas de satisfação são as impressões pessoais de usuários, que podem ser medidas, por exemplo, pelo interesse do usuário pela interface ou por seu nível de conforto e desconforto, durante os testes (FILARDI; TRAINA, 2008).

Também pode-se apontar três objetivos principais em uma avaliação: a funcionalidade do sistema, avaliando a eficiência do sistema na execução das tarefas pelo usuário; o efeito da interface junto ao usuário, que avalia o impacto do design junto ao usuário, considerando facilidade de aprendizado do sistema, flexibilidade e robustez; e problemas específicos do sistema, como aspectos de design que ocasionam resultados inesperados e confusos (FILARDI; TRAINA, 2008).

Levando em consideração os fatores necessários para se medir a usabilidade de uma

interface, foi elaborado um questionário para avaliar o dispositivo deste trabalho perante testes com usuários reais. O questionário foi dividido em três partes: eficácia, eficiência e satisfação, e busca englobar também os objetivos citados acima em seus itens de avaliação, que medem as impressões do usuário em uma escala de 1 a 5, onde os valores próximos a 1 indicam impressão negativa e os valores próximos a 5, uma impressão positiva. O questionário usado é encontrado no Apêndice 4.

Dois problemas se vinculam a um teste de usabilidade: a confiabilidade, que entende-se pelo grau de certeza de que o mesmo resultado será obtido se o teste for repetido; e a validade, que é o fato dos resultados de teste refletirem os aspectos de usabilidade que eram buscados. No quesito confiabilidade deve-se estar atento às diferenças individuais entre os usuários, e a regra principal quanto à validade é que a escolha dos usuários para os testes seja a mais representativa possível (ROCHA; BARANAUSKAS, 2003). Isto implica em vários contextos em que uma avaliação de interface pode ser feita, considerando tanto a diferença dos tipos de usuários abordados nos testes ou o ambiente em que os testes são realizados.

Neste trabalho, os usuários foram mestrandos de ciência da computação e têm um forte relacionamento com novas tecnologias. Também é pertinente citar que uma das aplicações a serem testadas tem apelo com a formação dos usuários, por se tratar de um conversor de números binários, assunto elementar na área de tecnologia, por fim, o ambiente de testes foi o próprio laboratório da universidade onde o dispositivo foi desenvolvido.

5 Validação

O principal resultado deste trabalho se mostra na próprio dispositivo tangível construído, além do desenvolvimento de sua API, sendo possível de se mostrar a escrita e execução de programas, de fato, que se utilizam do dispositivo. Algumas aplicações foram contruídas para exemplificar e validar o seu uso.

As imagens que o protótipo é capaz de mostrar se aproximam da ideia de um caractere pintado em uma tela, simulando uma tela primitiva de computador, com resolução de quatro pixels. O formato de letras e números apresentados nas aplicações a seguir foram convencionados a partir do que se pode mostrar com os quatro pinos. Mesmo assim, as funções usadas servem como prova de conceito, e poderão ser facilmente estendidas caso a malha de pixels seja aumentada para uma resolução maior. Como os pinos são disposto em linha, as imagens formadas são de uma visão frontal.

Os pinos ou pixels tangíveis podem assumir diferentes alturas e cores, o que acarreta em um grande número de possibilidades, ou formas impressas que a saída do dispositivo pode assumir. Para que seja possível a escrita de uma aplicação, usa-se o template e a biblioteca do protótipo, chamada aqui de *"LightForce"*, que deve ser referenciada no programa, disponibilizando assim as funções mencionadas anteriormente (**Lightining.draw_Pin()**, **Lightining.Print_Symbol()**, e **Lightining.Print_Bin()**).

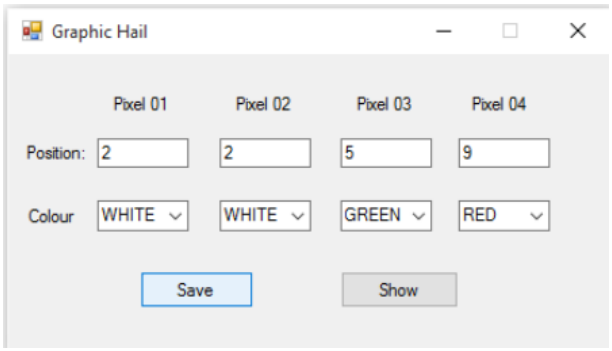
A seguir, são apresentadas três aplicações C# para desktop, desenvolvidas para exemplificar como o periférico funcionou na prática. Também foi feita uma avaliação usando técnicas de IHC, para medir o desempenho e satisfação da utilização do dispositivo, por parte de usuários reais.

5.1 Graphic Hail

O primeiro dos programas chama-se "Graphic Hail", ele recebe como entrada dados pré-definidos pelo usuário, referentes a gráficos em barras que serão mostrados em uma apresentação (variando a altura e as cores dos pixels tangíveis), e uma vez que os dados estejam inseridos, bastará ao usuário clicar no botão "Show" da aplicação, que o dispositivo exibirá os gráficos na ordem que estes foram armazenados.

Como mostra a Figura 48, há slots para que cada pixel receba um valor e uma cor, os valores pode ser de 0 a 9, referente as dez alturas disponíveis para cada pino no dispositivo, e as cores são uma das sete oferecidas (sem cor, vermelho, verde, azul, amarelo, roxo e branco). O usuário preenche o formulário com os dados do primeiro gráfico e clica em salvar.

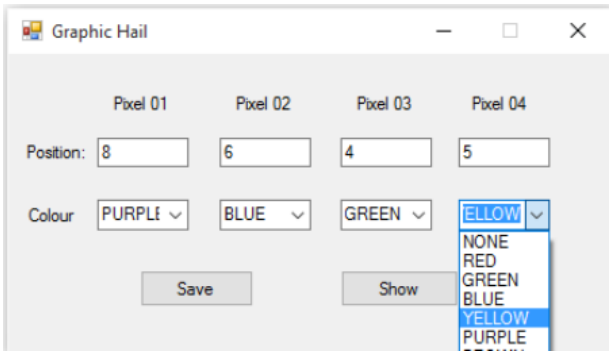
Figura 48: Graphic Hail - Inserindo gráfico 1.



Fonte: Autoria Própria.

Em seguida pode-se inserir novos gráficos quantas vezes achar necessário, alterando os valores da aplicação e clicando em "Save". Neste exemplo é feita uma apresentação de três gráficos, considerando que o primeiro seja referente aos dados da Figura 48, seguido de uma nova inserção mostrada na Figura 49, que também mostra a seleção de cores feita com uma *combobox*.

Figura 49: Graphic Hail - Inserindo gráfico 2.



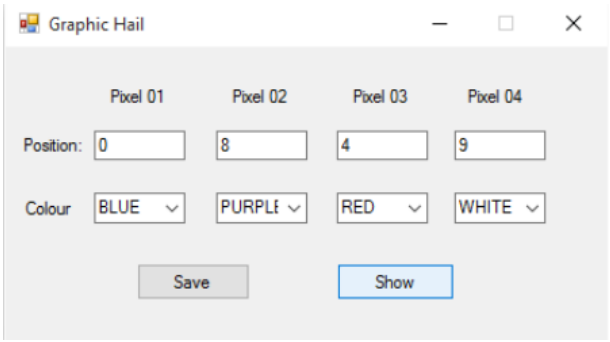
Fonte: Autoria Própria.

Continuando o exemplo, é feita a inserção de dados de mais um gráfico, lembrando que o usuário tem que clicar em "Save" a cada novo gráfico inserido, e quando terminar de armazenar os dados, clicar em "Show" (Figura 50) para começar a apresentação no dispositivo. A Figura 51 já mostra o primeiro gráfico resultado dos primeiros dados que foram inseridos.

Ao clicar novamente em "Show", será mostrado o segundo gráfico (Figura 52) inserido e seguindo o mesmo raciocínio, chega-se então ao terceiro e último gráfico (Figura 53), finalizando esta apresentação exemplo.

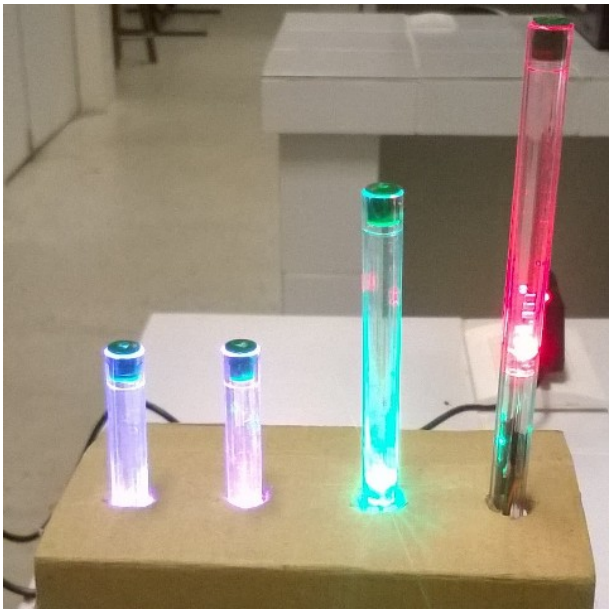
Caso o botão "Show" seja pressionado sem que haja gráficos salvos, a aplicação informará que não há dados, e que o usuário insira algum, e mesmo durante a execução do programa, enquanto alguns gráficos são mostrados no dispositivo, o usuário pode inserir novos gráficos.

Figura 50: Graphic Hail - Inserindo gráfico 3.



Fonte: Autoria Própria.

Figura 51: Graphic Hail - Gráfico 1.



Fonte: Autoria Própria.

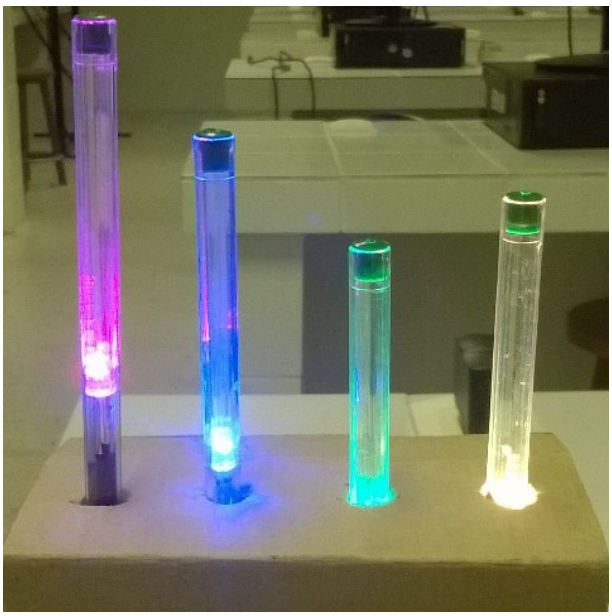
A função da biblioteca *"Lightining"* que predomina neste programa é a **Lightining.draw_Pin()**, esta função recebe parâmetros específicos para cada pino, formando imagens a partir dos dados independentes de cada um, o que possibilita resultados mais dinâmicos, dependendo do programa ou de um usuário.

5.2 Printer

O segundo dos programas chama-se "Printer", a aplicação recebe como entrada uma letra ou palavra via teclado, e mostra o que foi inserido no dispositivo físico.

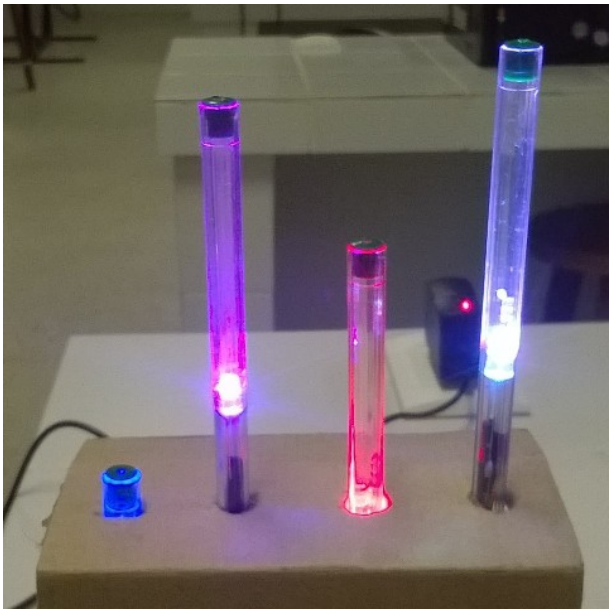
O usuário escreve uma letra ou palavra no formulário e pressiona o botão "imprimir", a letra ou primeira letra da palavra escrita é então mostrada no protótipo. Uma vez que a entrada seja uma palavra e a primeira letra esteja impressa no dispositivo, o usuário pressiona um dos pinos do protótipo, que ao reconhecer o toque, imprime o próximo

Figura 52: Graphic Hail - Gráfico 2.



Fonte: Autoria Própria.

Figura 53: Graphic Hail - Gráfico 3.



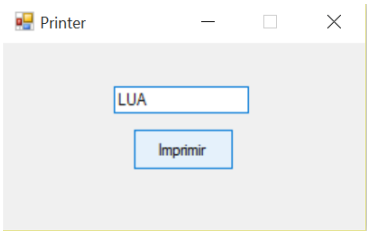
Fonte: Autoria Própria.

caractere da sequencia. A execução segue de maneira semelhante até o fim da palavra.

A seguir, é mostrada a execução passo a passo do programa descrito: na caixa de texto da aplicação, foi escrita a palavra “LUA”, palavra a ter suas letras impressas no dispositivo, uma vez que seja clicado no botão “Imprimir”, também presente no formulário, mostrado na Figura 54.

A Figura 55 mostra o dispositivo exibindo um valor qualquer (a), representando o estado anterior do dispositivo antes do programa começar a ser executado. Uma vez que o

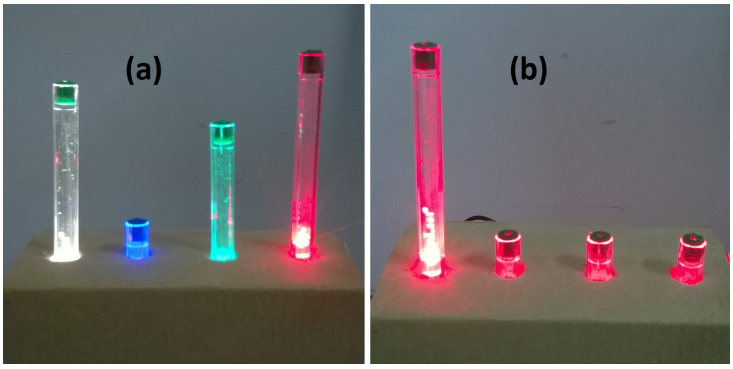
Figura 54: Formulário da aplicação “Printer”.



Fonte: Autoria Própria

botão imprimir seja clicado, usando a entrada “LUA”, o valor referente à letra “L” será “impresso” pela aplicação (b), e espera que o usuário aperte o primeiro pixel para que seja impressa a próxima letra.

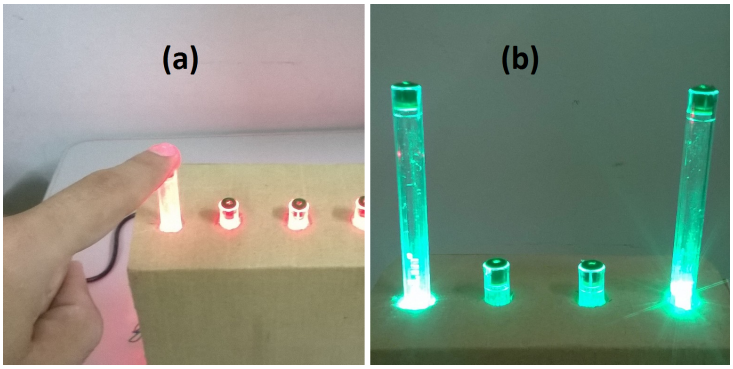
Figura 55: Dispositivo Tangível em Execução - 1.



Fonte: Autoria Própria

A Figura 56 (a) ilustra a ação do pino sendo pressionado, e a Figura 56 (b) a próxima letra impressa, neste caso, a letra “U”.

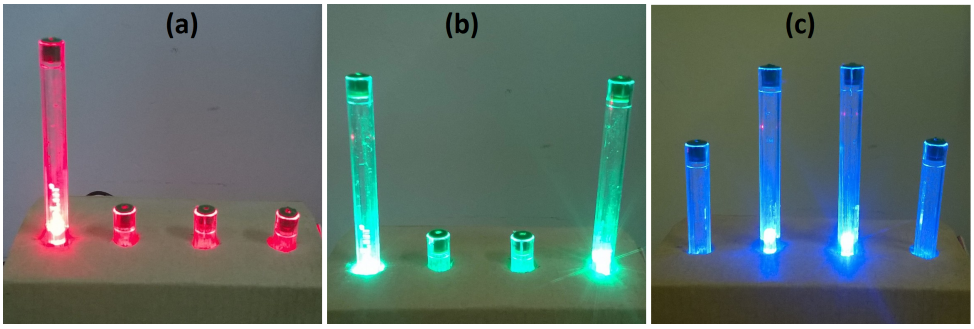
Figura 56: Dispositivo Tangível em Execução - 2.



Fonte: Autoria Própria

Dando continuidade a execução do programa, o primeiro pino será pressionado novamente e a última letra será mostrada, a letra “A”. A Figura 57 mostra em ordem, cada um dos resultados das interações com o dispositivo tangível para o caso apresentado, que postas em sequência, mostram o valor que foi inserido no início, a palavra “LUA”.

Figura 57: Palavra “LUA” Impressa no Dispositivo Tangível.



Fonte: Autoria Própria

No caso deste programa, quando a última letra é impressa, o dispositivo fica na espera de outra entrada simplesmente para sair de execução, isso não é uma regra, vai depender da necessidade de cada aplicação. A intenção em fazer com que o dispositivo espere algum comando do usuário, para que possa avançar ao próximo estado da execução, é justamente mostrar a característica tangível do dispositivo. O usuário interagiu não com a aplicação, mas sim com a própria imagem, o que resultou em uma reação da aplicação. O fato de ser o primeiro pino a ser pressionado também é ilustrativo, pois qualquer pino poderia ter sido escolhido para esta ou outras funções, inclusive poderia ser puxado em vez de pressionado.

A função predominante neste caso é a **Lightining.print_Symbol()**, esta função recebe um caractere do usuário e o “imprime” no dispositivo, baseando-se em uma lista pré-definida com vários caracteres presentes na biblioteca *Lightining*. Aqui também é usada a função **Pauser_Tick()**, nativa do próprio C#, implementada para checar a cada momento se houve movimento de algum pino e assim disparar o evento que mostra a letra seguinte ao perceber alteração nas alturas dos pinos.

5.3 Binary Play

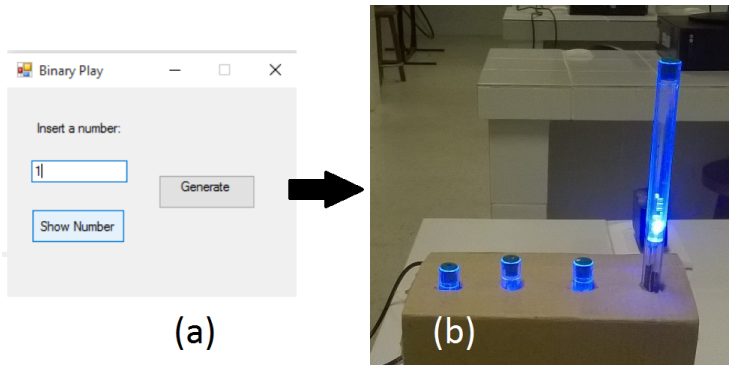
A terceira aplicação feita para o dispositivo é a *Binary Play*, consiste em duas partes: mostrar números binários ao usuário no dispositivo (“imprimindo” diretamente como na aplicação anterior), e pedir que o usuário mostre um número, utilizando da interação manual, no dispositivo, para “desenhar” o valor binário correspondente. Quando o usuário mostra o valor correto no dispositivo, os pinos mudam de cor e aparece a mensagem *“Correct”* na aplicação, indicando que o valor “desenhado” pelo usuário está correto.

Considerando essas duas funcionalidades, a aplicação se assemelha a um jogo de aprendizagem de números binários, onde em um primeiro momento é mostrado como são as representações de números binários no dispositivo (no caso um pino abaixado representa

o valor ‘0’e um pino levantado representa o valor ‘1’) e no segundo momento é feito um teste para saber se o usúario lembra ou aprendeu tais representações.

Para começar a iteração, o usuário digita um número na caixa de texto da aplicação e clica em *“Show Number”*, isso faz com que o dispositivo mostre a sua versão em binário do número digitado, a Figura 58 apresenta um primeiro exemplo, quando foi digitado o número ‘1’ na aplicação (a) e mostrado o resultado no dispositivo (b).

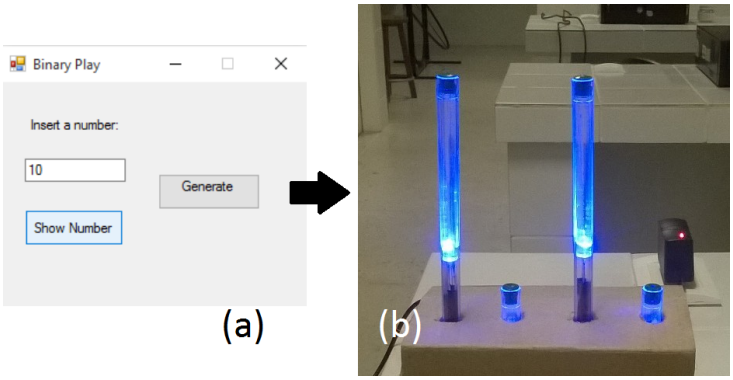
Figura 58: Execução do Binary Play - 1.



Fonte: Autoria Própria

Se utilizando de outro exemplo para a mesma funcionalidade temos para a entrada ‘10’ na Figura 59 (a) o resultado correspondente (b).

Figura 59: Execução do Binary Play - 2.



Fonte: Autoria Própria

Uma vez que o usuário perceba como o dispositivo trata os valores binários, ou seja, a maneira que ele representa os 0s e 1s, é possível que o próprio usuário seja capaz de mostrar esses valores se utilizando dos pixels do dispositivo, assim chega-se na segunda funcionalidade da aplicação.

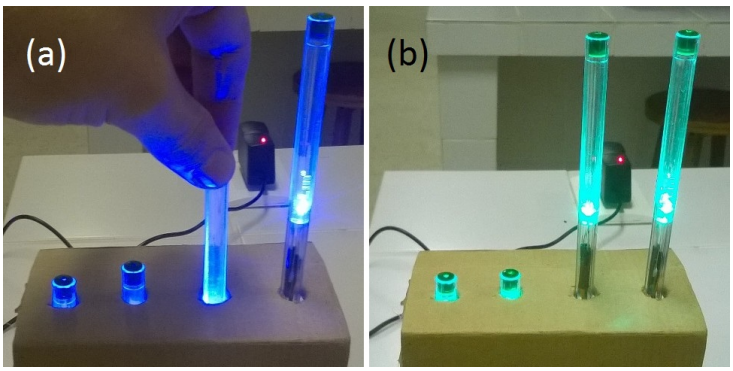
Ao clicar na opção *“Generate”*, um número aleatório de 1 ao 15 será mostrado (pois é o que o dispositivo permite mostrar com quatro pinos, onde as possibilidades em números binários são de 0000 a 1111), e é pedido que o usuário insira esse número no dspositivo, no caso da Figura 60 o número ‘3’ foi requisitado.

Figura 60: Aplicação Binary Play.

Fonte: Autoria Própria

O que o usuário tem a fazer em seguida é formar o número '3' no dispositivo, em binário, ou seja, o número '0011', que será equivalente aos dois primeiros pinos abaixados e os dois últimos pinos levantados. O usuário então começa a mover os pixels - Figura 61 (a) - e uma vez que se tenha o formato correto, os pinos que antes estavam com a cor azul, ficam verdes, indicando que a resposta está correta, como mostrado na Figura 61 (b).

Figura 61: Utilização do Binary Play - 1.

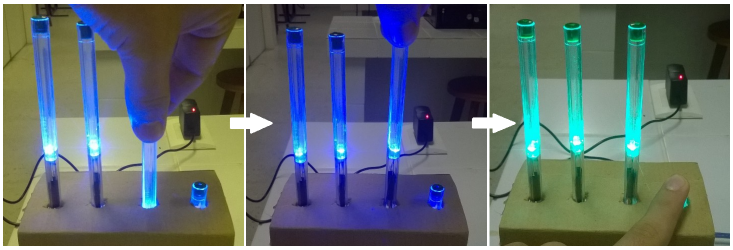


Fonte: Autoria Própria

A seguir, a Figura 62 exemplifica novamente o usuário mostrando uma resposta no dispositivo, neste caso, a aplicação tinha requerido o número '14' (1110 em binário).

A Figura 63 mostra que quando a resposta do usuário está correta, além dos pinos mudarem de cor (no caso ficam verdes), aparece a mensagem "CORRECT!" (a) no formulário da aplicação, que também indica, por parte do programa, que o número inserido está correto. Também pode-se observar que: se durante a execução, o usuário quiser checar novamente o formato de um número, ele pode fazer isto digitando o número

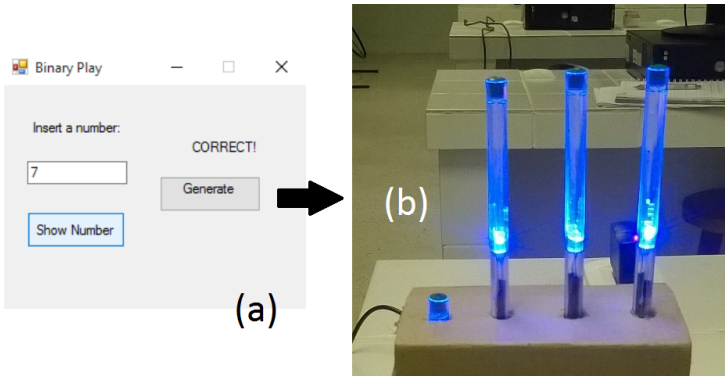
Figura 62: Utilização do Binary Play - 2.



Fonte: Autoria Própria

na caixa de texto e clicando em "Show Number", como já foi feito anteriormente, o que resulta consequentemente no dispositivo exibir o número informado (b).

Figura 63: Utilização do Binary Play - 3.



Fonte: Autoria Própria

Da biblioteca *"Lightining"*, além da função **Lightining.Print_Bin()**, que imprime os valores binários no dispositivo tendo como parâmetros valores inteiros, foi usada uma função especialmente criada para esta aplicação, a **Lightining.Print_BinG()**, que nada mais é do que uma função espelhada na primeira, só que com a cor verde no lugar da azul. A função foi utilizada para mudar a cor dos pixels quando o usuário chega à resposta correta da aplicação, o que demonstra que a biblioteca entregue ao usuário também pode ser incrementada com novas funções, dependendo da necessidade.

5.4 Avaliação

Para análise das respostas obtidas pelo questionário, citado anteriormente, foi utilizado o TAM (*Technology Acceptance Model* ou Modelo de Aceitação de Tecnologia) (DAVIS; BAGOZZI; WARSHAW, 1989), que objetiva identificar se determinada tecnologia é aceita ou rejeitada por usuários, baseando-se na percepção dos mesmos, sobre a utilidade e sobre a facilidade de uso da tecnologia.

No que concerne à percepção de facilidade do uso, os questionamentos sobre: manuseio do dispositivo; tempo de aprendizado; intuitividade no uso e compreensão da

representação dos caracteres pelo dispositivo foram respondidos como totalmente fáceis ou satisfatórios por 90% dos usuários.

No quesito percepção de utilidade, as questões: reação ao toque; resposta e precisão das aplicações; potencial para upgrade e novas aplicações e se o testador usaria o dispositivo no dia-a-dia, as respostas ficaram entre 80% e 90% como totalmente boas ou satisfatórias.

Os resultados foram baseados considerando-se que respostas marcadas como 5 no questionário indicavam uma completa satisfação do usuário, e respostas marcadas como 1 significariam completa insatisfaçao. Ambos os resultados foram positivos, mostrando que o dispositivo é fácil de usar e possui boa utilidade. Houve uma pequena queda na resposta dos usuários quanto a utilidade em relação à facilidade de uso, que segundo comentários dos testadores, justifica-se pelo fato do dispositivo ser um protótipo.

A pesquisa de validação foi feita tendo como testadores 7 estudantes de computação. A Figura 64 mostra os resultados detalhados em porcentagem de cada item, considerando que respostas mais próximas de 100% foram respostas que se aproximarem de 5, na escala de 1 a 5 mencionada anteriormente.

Figura 64: Resultados da Avaliação.

Fonte: Autoria Própria

6 Considerações Finais

A constante evolução humana demanda evolução tecnológica, novas necessidades e busca para facilitar a vida das pessoas sempre provêm espaço para novas ideias. Com esse horizonte, este trabalho mostrou uma pesquisa feita sobre interfaces tangíveis, buscando entender e desenvolver ideias desta área.

Também foi apresentada a proposta de uma TUI que funciona como um dispositivo de entrada e saída de propósito geral. O dispositivo apresenta uma malha de pixels tangíveis e permite executar diversas aplicações.

Junto com a TUI, foi escrita uma biblioteca e um template C#, proporcionando aos usuários uma interface de alto nível para a escrita de programas, abstraindo as funções mais básicas do hardware e facilitando o desenvolvimento de aplicações para o mesmo.

Foram desenvolvidas três aplicações para exemplificar o funcionamento do dispositivo, testando suas características tangíveis, através da impressão de palavras, gráficos e números binários.

Testes foram feitos com usuários a fim de verificar a aceitabilidade do dispositivo construído, e o resultados foram em sua maioria satisfatório.

Os trabalhos futuros incluem: o aumento da malha de pixels tangíveis, possível com o empilhamento de placas controladoras de motor em uma placa Arduino; o desenvolvimento de uma malha ainda maior controlada por várias placas Arduino; o desenvolvimento de novas funções na biblioteca de software, que podem incluir a impressão direta de figuras prontas, strings, figuras geométricas e etc.; desenvolver uma série de aplicações a partir das novas funções e bibliotecas; e submetê-las a diversos tipos de usuário, a fim de realizar medidas do grau de imersão ou o nível de quão amigável é a interface em comparação a uma GUI, para as mesmas aplicações.

Aplicações direcionadas a deficientes visuais também podem ser consideradas como trabalhos futuros, ao passo que a parte tangível do dispositivo entra em contato direto com as mãos dessas pessoas, que são seu principal recurso de comunicação visual. Se a malha estiver grande o suficiente, poderá renderizar um rosto por exemplo, e o usuário pode usar as mãos para enxergar este rosto, isto sem mencionar aplicações mais voltadas ao mundo digital tal como navegar na internet sem ter apenas o audio do computador como guia.

Referências

ADAFRUIT. *Adafruit Industries*. 2016. Disponível em: <https://www.adafruit.com/>. Citado 4 vezes nas páginas 44, 45, 66 e 69.

ALPS. 2016. Disponível em: <http://www.alps.com/prod/info/E/HTML/Potentiometer/SlidePotentiometers/RSN1M/RSA0N11M9A0K.html>. Citado na página 68.

ARDUINO. *Arduino*. 2016. Disponível em: <https://www.arduino.cc/>. Citado 4 vezes nas páginas 30, 41, 66 e 69.

BEAGLE. *beagleboard.org*. 2016. Disponível em: <https://beagleboard.org/>. Citado 4 vezes nas páginas 34, 36, 37 e 41.

BOUSSEMART, B.; GIROUX, S. Tangible user interfaces for cognitive assistance. In: *Advanced Information Networking and Applications Workshops, 2007, AINAW '07. 21st International Conference on*. [S.l.: s.n.], 2007. v. 2, p. 852–857. Citado na página 50.

CARDARELLI, L. et al. Tangima display de imagens tangíveis. *Pontificia Universidade Cata do Rio de Janeiro*, 2016. Citado na página 42.

CARRO, L.; WAGNER, F. R. Sistemas computacionais embarcados. *Jornadas de atualização em informática. Campinas: UNICAMP*, 2003. Citado 3 vezes nas páginas 20, 21 e 22.

CATALA, A. et al. Agoras: Exploring creative learning on tangible user interfaces. In: *Computer Software and Applications Conference (COMPSAC), 2011 IEEE 35th Annual*. [S.l.: s.n.], 2011. p. 326–335. ISSN 0730-3157. Citado na página 55.

COLTER, A. et al. Soundforms: Manipulating sound through touch. In: *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. New York, NY, USA: ACM, 2016. (CHI EA '16), p. 2425–2430. ISBN 978-1-4503-4082-3. Disponível em: <http://doi.acm.org/10.1145/2851581.2892414>. Citado na página 54.

CURVELLO, A. *Intel Edison Lanento do mo IoT da Intel*. 2014. Disponível em: <http://www.embarcados.com.br/intel-edison-lancamento-modulo-iot-da-intel/>. Citado na página 39.

DAVIS, F. D.; BAGOZZI, R. P.; WARSHAW, P. R. User acceptance of computer technology: a comparison of two theoretical models. *Management science*, INFORMS, v. 35, n. 8, p. 982–1003, 1989. Citado 2 vezes nas páginas 3 e 83.

DOLINAY, J.; DOSTÁLEK, P.; VAŠEK, V. Microcontroller software library for process control. *WSEAS Transactions on Systems and Control*, World Scientific and Engineering Academy and Society (WSEAS), 2015. Citado na página 46.

ENGINEERING, E. *How do I calculate what resistors I need for RGB LEDs with 3.7V?* 2016. Disponível em: <http://electronics.stackexchange.com/questions/112773/how-do-i-calculate-what-resistors-i-need-for-rgb-leds-with-3-7v>. Citado na página 67.

EVANS, M.; NOBLE, J.; HOCHENBAUM, J. *Arduino em ação*. [S.l.]: Novatec Editora, 2013. Citado na página 28.

FALCAO, T. P.; GOMES, A. S. Interfaces tangíveis para a educação. *Simpósio Brasileiro de Informática na Educação (SBIE)*, 2007. Citado na página 12.

FERREIRA, J. M. M. *Introdução ao projecto com sistemas digitais e microcontroladores*. [S.l.]: FEUP Edições, 1998. Citado na página 26.

FILARDI, A. L.; TRAINA, A. J. M. Montando questionários para medir a satisfação do usuário: avaliação de interface de um sistema que utiliza técnicas de recuperação de imagens por conteúdo. In: SOCIEDADE BRASILEIRA DE COMPUTAÇÃO. *Proceedings of the VIII Brazilian Symposium on Human Factors in Computing Systems*. [S.l.], 2008. p. 176–185. Citado 2 vezes nas páginas 20 e 73.

FILIPEFLOP. *Motor Shield L293D Driver Ponte H para Arduino*. 2016. Disponível em: <http://www.filipeflop.com/pd-6b643-motor-shield-l293d-driver-ponte-h-para-arduino.html>. Citado 2 vezes nas páginas 44 e 45.

FOLLMER, S. et al. inform: dynamic physical affordances and constraints through shape and object actuation. In: *UIST*. [S.l.: s.n.], 2013. v. 13, p. 417–426. Citado 3 vezes nas páginas 50, 61 e 68.

FRANcA, A. L. M. *Energia Eletrica Para Engenheiros*. [s.n.], 2001. Disponível em: <https://cdn.hackaday.io/files/9127390489568/motor_cc.pdf>. Citado na página 43.

GALILEU, R. *Primeira cirurgia feita apenas por robs m sucedida*. 2013. Disponível em: <http://revistagalileu.globo.com/Revista/Common/0,,EMI181382-17770,00-PRIMEIRA+CIRURGIA+FEITA+APENAS+POR+ROBOS+E+BEM+SUCEDIDA.html>. Citado na página 13.

GARCIA-CANSECO, E. et al. Development of tangible haptic interfaces for use in physics instruction. In: *Haptic Audio Visual Environments and Games (HAVE), 2013 IEEE International Symposium on*. [S.l.: s.n.], 2013. p. 13–15. Citado na página 58.

HEWETT, T. et al. *ACM SIGCHI Curricula for Human Computer Interaction. Chapter 2: Human Computer Interaction*. [S.l.]: ACM, consultado, 2003. Citado na página 18.

INTEL. *Intel Edison Compute Module, Boards, and Kits*. 2016. Disponível em: <http://www.intel.com/content/www/us/en/do-it-yourself/edison.html>. Citado 2 vezes nas páginas 39 e 41.

INTEL. *Intel Galileo Gen 2 Board*. 2016. Disponível em: <http://www.intel.com.br/content/www/br/pt/do-it-yourself/galileo-maker-quark-board.html>. Citado 2 vezes nas páginas 37 e 38.

ISHII, H. The tangible user interface and its evolution. *Communications of the ACM*, ACM, v. 51, n. 6, p. 32–36, 2008. Citado 2 vezes nas páginas 17 e 18.

ISHII, H. et al. Transform: Embodiment of "radical atoms"at milano design week. In: *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*. New York, NY, USA: ACM, 2015. (CHI EA '15), p. 687–694. ISBN 978-1-4503-3146-3. Disponível em: <http://doi.acm.org/10.1145/2702613.2702969>. Citado na página 52.

JÁCOME, J. *O que é biblioteca de programação / library / lib ? O que é API / Application Programming Interface ?* 2010. Disponível em: <https://jarbasjacome.wordpress.com/o-que-e-biblioteca-de-programacao-library-lib-o-que-e-api-application-programming-interface/>. Citado na página 45.

JIAQI, H.; SANTOSO, M.; GOOK, L. B. Immersive driving car simulation for children using natural user interface controller. In: *Ubiquitous Virtual Reality (ISUVR), 2013 International Symposium on*. [S.l.: s.n.], 2013. p. 23–25. Citado na página 57.

KATO, H.; BILLINGHURST, M. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In: IEEE. *Augmented Reality, 1999.(IWAR'99) Proceedings. 2nd IEEE and ACM International Workshop on*. [S.l.], 1999. p. 85–94. Citado na página 48.

KATO, H. et al. Virtual object manipulation on a table-top ar environment, isar. *Presence*, p. 111–119, 2000. Citado na página 47.

KATO, H. et al. Magiccup: a tangible interface for virtual objects manipulation in table-top augmented reality. In: *Augmented Reality Toolkit Workshop, 2003. IEEE International*. [S.l.: s.n.], 2003. p. 75–76. ISSN 0953-5683. Citado na página 48.

KAUFMANN, B.; BUECHLEY, L. Amarino: A toolkit for the rapid prototyping of mobile ubiquitous computing. In: *Proceedings of the 12th International Conference on Human Computer Interaction with Mobile Devices and Services*. New York, NY, USA: ACM, 2010. (MobileHCI '10), p. 291–298. ISBN 978-1-60558-835-3. Disponível em: <http://doi.acm.org/10.1145/1851600.1851652>. Citado na página 59.

KOENKA, I. J.; SáIZ, J.; HAUSER, P. C. Instrumentino: An open-source modular python framework for controlling arduino based experimental instruments. *Computer Physics Communications*, v. 185, n. 10, p. 2724 – 2729, 2014. ISSN 0010-4655. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0010465514002112>. Citado na página 60.

LEE, J. Y.; SEO, D. W.; RHEE, G. W. Tangible authoring of 3d virtual scenes in dynamic augmented reality environment. *Computers in Industry*, v. 62, n. 1, p. 107 – 119, 2011. ISSN 0166-3615. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0166361510001077>. Citado na página 13.

LEITHINGER, D.; ISHII, H. Relief: A scalable actuated shape display. In: *Proceedings of the Fourth International Conference on Tangible, Embedded, and Embodied Interaction*. New York, NY, USA: ACM, 2010. (TEI '10), p. 221–222. ISBN 978-1-60558-841-4. Disponível em: <http://doi.acm.org/10.1145/1709886.1709928>. Citado 2 vezes nas páginas 59 e 61.

LIMA, T. Freescale freedom board - placa de sensores (parte i). *Embarcados*, 2014. Disponível em: <http://www.embarcados.com.br/freescale-freedom-board-sensores-i/>. Citado na página 30.

LIMA, T. *Intel Galileo - Placa Arduino*. 2016. Disponível em: <http://www.embarcados.com.br/intel-galileo/>. Citado 2 vezes nas páginas 37 e 38.

MARTINS, N. A. Sistemas microcontrolados. *São Paulo: Novatec*, 2005. Citado 2 vezes nas páginas 26 e 27.

MIT. *Tangible Media Group. Disponl em: <http://tangible.media.mit.edu/> Acesso em: 13 dez. 2014*. 2014. Citado 4 vezes nas páginas 11, 51, 52 e 53.

NAKAGAKI, K. et al. Materiable: Rendering dynamic material properties in response to direct physical touch with shape changing interfaces. In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, 2016. (CHI '16), p. 2764–2772. ISBN 978-1-4503-3362-7. Disponível em: <http://doi.acm.org/10.1145/2858036.2858104>. Citado na página 53.

NXP. *NXP Semiconductors*. 2016. Disponível em: <http://www.nxp.com/products/software-and-tools/hardware-development-tools/freedom-development-boards:FREDEVPLA?fsrch=1&sr=1&pageNum=1>. Citado 3 vezes nas páginas 30, 31 e 41.

OLIVEIRA, F. de. Interfaces usuário-máquina. Rio Pomba, 51 p. Apostila do curso de ciência da computação. *Instituto Federal de Educação, Ciência e Tecnologia Sudeste de Minas Gerais*, 2014. Citado na página 11.

PRADO, S. *Freedom Board FRDM-KL46Z*. 2015. Disponível em: <https://sergioprado.org/freedom-board-frdm-kl46z/>. Citado 2 vezes nas páginas 30 e 31.

PRATES, R. O.; BARBOSA, S. D. J. Avaliação de interfaces de usuário–conceitos e métodos. In: *Jornada de Atualização em Informática do Congresso da Sociedade Brasileira de Computação, Capítulo*. [S.l.: s.n.], 2003. v. 6. Citado 2 vezes nas páginas 19 e 20.

PYTHON. *Python Brasil*. 2016. Disponível em: <www.python.org.br/>. Citado na página 32.

RAMOS, J. et al. Iniciativa para robótica pedagógica aberta e de baixo custo para inclusão social e digital no brasil. *Simpósio Brasileiro de Automação Inteligente (SBAI)*, 2007. Citado na página 27.

RASPBERRYPI. *Raspberry Pi - Teach, learn and make with Raspberry Pi*. 2016. Disponível em: <https://www.raspberrypi.org/>. Citado 2 vezes nas páginas 32 e 41.

REACTABLE. *Reactable - Music Knowledge Technology*. 2016. Disponível em: <http://reactable.com/>. Citado na página 14.

REBELO, I. In: *Apostila IHC*. [S.l.: s.n.], 2009. Citado na página 19.

RICARDO, M. *Posi e Rota do Servomotores*. 2016. Disponível em: <http://moisesricardo.xpg.uol.com.br/posicao-e-rotacao-do-servomotores.html>. Citado na página 42.

ROBOLIV. *Servo Motor*. 2016. Disponível em: <http://www.roboliv.re/conteudo/servomotor>. Citado na página 42.

ROCHA, H. V. D.; BARANAUSKAS, M. C. C. *Design e avaliação de interfaces humano-computador*. [S.l.]: Unicamp, 2003. Citado 3 vezes nas páginas 19, 73 e 74.

ROH, Y. et al. Haptic u-table: A believable feedback system with a quasi-tangible tabletop interface. In: *Ubiquitous Intelligence Computing and 7th International Conference on Autonomic Trusted Computing (UIC/ATC), 2010 7th International Conference on*. [S.l.: s.n.], 2010. p. 266–271. Citado na página 56.

SANTOS, V. P. de A.; BRITES, F. G. Motor de passo. *Universidade Federal Fluminense*, 2008. Citado 2 vezes nas páginas 42 e 43.

SCRATCH. *Scratch - Imagine, program, share*. 2016. Disponível em: <https://scratch.mit.edu/>. Citado na página 31.

SEED.CC. *Welcome To seed.cc*. 2016. Disponível em: <http://cc.seeed.cc/index.html>. Citado 2 vezes nas páginas 36 e 40.

SPARKFUN. *SparkFun Ardumoto - Motor Driver Shield*. 2016. Disponível em: <https://www.sparkfun.com/products/9815>. Citado 2 vezes nas páginas 44 e 45.

STERMAN, E. *A Revolução Tangível. Disponl em: <http://www.2600hz.com.br/materias_tangible2.html> Acesso em: 13 dez. 2014*. 2014. Citado na página 17.

SUN, C.; KHOO, S.-C.; ZHANG, S. J. Graph-based detection of library api imitations. In: *Software Maintenance (ICSM), 2011 27th IEEE International Conference on*. [S.l.: s.n.], 2011. p. 183–192. ISSN 1063-6773. Citado na página 46.

TANENBAUM, A. S.; AUSTIN, T. *Organização estruturada de computadores*. 6. ed. [S.l.]: PEARSON, 2013. Citado 2 vezes nas páginas 26 e 27.

TECHTUDO. *Raspberry Pi: conheça os modelos e saiba qual o mais indicado para você*. 2016. Disponível em: <http://www.techtudo.com.br/listas/noticia/2016/03/raspberry-pi-conheca-os-modelos-e-saiba-qual-o-mais-indicado-para-voce.html>. Citado 3 vezes nas páginas 32, 33 e 34.

TONEY, A.; THOMAS, B. Considering reach in tangible and table top design. In: *Horizontal Interactive Human-Computer Systems, 2006. TableTop 2006. First IEEE International Workshop on*. [S.l.: s.n.], 2006. p. 2 pp.–. Citado na página 49.

VINK, L. et al. Transform as adaptive and dynamic furniture. In: *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*. New York, NY, USA: ACM, 2015. (CHI EA '15), p. 183–183. ISBN 978-1-4503-3146-3. Disponível em: <http://doi.acm.org/10.1145/2702613.2732494>. Citado na página 52.

WEISER, M. The computer for the 21st century. *SIGMOBILE Mob. Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 3, n. 3, p. 3–11, jul. 1999. ISSN 1559-1662. Disponível em: <http://doi.acm.org/10.1145/329124.329126>. Citado na página 11.

WOLF, W. Computer as components. *Principles of Embedded Computing System Design*, 2001. Citado 6 vezes nas páginas 22, 23, 24, 25, 26 e 66.

WU, F.-G. et al. Use pattern-recognition-based technology to explore a new interactive system on smart dining table. In: *Orange Technologies (ICOT), 2013 International Conference on*. [S.l.: s.n.], 2013. p. 252–255. Citado na página 54.

ZTOP. *Zen e a arte de prototipar com o Intel Edison*. 2016. Disponível em: <http://www.ztop.com.br/intel-edison/>. Citado na página 40.

7 Apêndice 1

```
//Programa Controlador do Dispositivo - carregado em Arduino

//bibliotecas da placa Adafruit Motor Shield
#include <Wire.h>
#include <Adafruit_MotorShield.h>
#include "utility/Adafruit_PWMServoDriver.h"

//Definindo uma váriavel que será entendida como a 'placa Addafruit'
Adafruit_MotorShield AFMS = Adafruit_MotorShield();

//seleciona as portas da placa Adafruit para cada motor: M1, M2, M3 e M4
Adafruit_DCMotor *my1Motor = AFMS.getMotor(1);
Adafruit_DCMotor *my2Motor = AFMS.getMotor(2);
Adafruit_DCMotor *my3Motor = AFMS.getMotor(3);
Adafruit_DCMotor *my4Motor = AFMS.getMotor(4);

//variáveis globais

//variávies de controle, e recebimento de comandos da porta serial
boolean a = false;
String inString = "", s1 = "", s2 = "", s3 = "", s4 = "";
char dig;
int i, val1, val2, val3, val4;
byte B;

//Inicializando as variáveis de posições dos motores
int potvalor1 = 0;
int potvalor2 = 0;
int potvalor3 = 0;
int potvalor4 = 0;

//Pinos da placa Arduino para cada pino (R, G e B) dos LEDs

//led do motor 1
int red1=25;  int green1=26;  int blue1=27;

//led do motor 2
int red2=28;  int green2=29;  int blue2=30;

//led do motor 3
int red3=33;  int green3=35;  int blue3=36;

//leds do motor 1
int red4=42;  int green4=43;  int blue4=44;


// setup() inicializa variáveis da placa arduino e define os modos de entrada ou saída dos pinos

void setup() {
```

```
Serial.begin(9600);  // Define a taxa de dados em bits por segundo para transmissão de dados
via serial

AFMS.begin();  // ligando a placa Adafruit

// Define as velocidades dos motores de 0 a 255
// o valor máximo foi escolhido para que o motor tenha força para levantar o pino
my1Motor->setSpeed(255);
my2Motor->setSpeed(255);
my3Motor->setSpeed(255);
my4Motor->setSpeed(255);

}

// loop() cria um laço que se repete consecutivamente, permitindo que o programa para
//possa ser alterado e responda a comandos
void loop() {

//ler constantemente a porta serial, buscando dados (no caso os comandos de controle que
//vierem de uma aplicação) para que uma ação seja
//realizada; caso o comando 'T' seja solicitado, retorna a leitura das posições dos pixels, que é
//usada como entrada no dispositivo
while(Serial.available() > 0){
   dig = Serial.read();
   if(dig == '\n'){
    break;
   }if(dig=='T'){
    Serial.println(analogRead(A8));
    Serial.println(analogRead(A9));
    Serial.println(analogRead(A10));
    Serial.println(analogRead(A11));
   }else{
    inString += dig; //cria um comando de controle
      }
   delay(1);
 }

if(inString!=NULL){ //se houver um comando de controle

// se o comando for para o MOTOR 1

// '$$' (marcador)

   if(inString[0]=='0' && inString[1]=='1'){

    //ascende uma cor caso seja uma das abaixo
    analogWrite(red1,0); analogWrite(green1,0);  analogWrite(blue1,0);

    if(inString[2]=='R' && inString[3]=='1'){

      analogWrite(red1,255);  //VERMELHO
```

```
    } else if (inString[2]=='G' && inString[3]=='1'){

     analogWrite(green1,255);  //VERDE

    } else if (inString[2]=='B' && inString[3]=='1'){

     analogWrite(blue1,255);  //AZUL

    } else if (inString[2]=='P' && inString[3]=='1'){

     analogWrite(red1,128); analogWrite(blue1,128);  //ROXO

    } else if (inString[2]=='B' && inString[3]=='3'){

     analogWrite(red1,0); analogWrite(green1,0);  analogWrite(blue1,0); //SEM COR

    } else if (inString[2]=='W' && inString[3]=='1'){

     analogWrite(red1,255); analogWrite(green1,255);  analogWrite(blue1,255); //BRANCO

    } else if (inString[2]=='Y' && inString[3]=='1'){

     analogWrite(red1,255); analogWrite(green1,255);   // AMARELO

    } else {

    }

    s1 += inString[4]; //recebe a altura indicada para o motor em String
    val1 = s1.toInt(); //converte para inteiro
    s1 = ""; // apaga a string para que receba outra posteriormente

    potvalor1 = analogRead(A8); // ler em que altura está o motor e baseado nisso o move para
//outra posição

     if(val1==0){ //altura 1
      while(potvalor1>102){
       my1Motor->run(FORWARD);
       potvalor1 = analogRead(A8);
      }

     } if(val1==1){ //altura 2
      while(potvalor1>102){
       my1Motor->run(FORWARD);
       potvalor1 = analogRead(A8);
      } while(potvalor1<204){
        my1Motor->run(BACKWARD);
       potvalor1 = analogRead(A8);
      }
     } if(val1==2){ //altura 3, etc...
      while(potvalor1>204){
       my1Motor->run(FORWARD);
```

```
 potvalor1 = analogRead(A8);
 } while(potvalor1<306){
  my1Motor->run(BACKWARD);
 potvalor1 = analogRead(A8);
 }
} if(val1==3){
 while(potvalor1>306){
 my1Motor->run(FORWARD);
 potvalor1 = analogRead(A8);
 } while(potvalor1<409){
  my1Motor->run(BACKWARD);
 potvalor1 = analogRead(A8);
 }
} if(val1==4){
 while(potvalor1>409){
 my1Motor->run(FORWARD);
 potvalor1 = analogRead(A8);
 } while(potvalor1<511){
  my1Motor->run(BACKWARD);
 potvalor1 = analogRead(A8);
 }
} if(val1==5){
 while(potvalor1>511){
 my1Motor->run(FORWARD);
 potvalor1 = analogRead(A8);
 } while(potvalor1<613){
  my1Motor->run(BACKWARD);
 potvalor1 = analogRead(A8);
 }
} if(val1==6){
 while(potvalor1>613){
 my1Motor->run(FORWARD);
 potvalor1 = analogRead(A8);
 } while(potvalor1<716){
  my1Motor->run(BACKWARD);
 potvalor1 = analogRead(A8);
 }
} if(val1==7){
 while(potvalor1>716){
 my1Motor->run(FORWARD);
 potvalor1 = analogRead(A8);
 } while(potvalor1<818){
  my1Motor->run(BACKWARD);
 potvalor1 = analogRead(A8);
 }
} if(val1==8){
 while(potvalor1>818){
 my1Motor->run(FORWARD);
 potvalor1 = analogRead(A8);
 } while(potvalor1<920){
  my1Motor->run(BACKWARD);
 potvalor1 = analogRead(A8);
```

```
      }
     } If(val1==9){
      while(potvalor1<920){
       my1Motor->run(BACKWARD);
       potvalor1 = analogRead(A8);
      }
     }

    // "solta o motor"
    my1Motor->run(RELEASE);

}

    // o código a partir de '$$' é repetido para cada motor
    // ...
    // ...
    // ...

   // limpa a string que recebe os códigos de controle para receber um código posteirior
   inString = NULL;

  }

}
```

8 Apêndice 2

```
namespace LightForce
{
    public class Lightining
    {

       // função Lightining.draw_Pin() - recebe três inteiros que indicam, qual
       // dos quatro pixels será modificado, e qual cor e altura ele receberá

        static public string draw_Pin(int id_pin, int colour, int pos)
        {
            char[] d = new char[5];

            d[0] = '0'; //Motores 1, 2, 3 e 4
            if (id_pin == 1) d[1] = '1';
            if (id_pin == 2) d[1] = '2';
            if (id_pin == 3) d[1] = '3';
            if (id_pin == 4) d[1] = '4';

            if (colour == 0) //PRETO - APAGADO
            {
                d[2] = 'B'; d[3] = '3';
            }

            if (colour == 1) //VERMELHO
            {
                d[2] = 'R'; d[3] = '1';
            }

            if (colour == 2) //VERDE
            {
                d[2] = 'G'; d[3] = '1';
            }

            if (colour == 3) //AZUL
            {
                d[2] = 'B'; d[3] = '1';
            }

            if (colour == 4) //ROXO
            {
                d[2] = 'P'; d[3] = '1';
            }

            if (colour == 5) //AMARELO
            {
                d[2] = 'Y'; d[3] = '1';
            }

            if (colour == 6) //MARROM
            {
                d[2] = 'B'; d[3] = '2';
            }

            if (colour == 7) //BRANCO
            {
                d[2] = 'W'; d[3] = '1';
            }
            // ALTURAS
            if (pos == 0) d[4] = '0';
            if (pos == 1) d[4] = '1';
            if (pos == 2) d[4] = '2';
            if (pos == 3) d[4] = '3';
```

```
    if (pos == 4) d[4] = '4';
    if (pos == 5) d[4] = '5';
    if (pos == 6) d[4] = '6';
    if (pos == 7) d[4] = '7';
    if (pos == 8) d[4] = '8';
    if (pos == 9) d[4] = '9';

    string s = new string(d);

    return s;
}

// função Lightining.print_Bin() - exibe uma representação em binário no
// dipositivo para números inteiros de 0 a 15

static public string print_Bin(int bin)
{
    bin = bin % 16;

    string s = "01B3002B3003B3004B30";

    if (bin == 0) s = "01B1002B1003B1004B10";

    if (bin == 1) s = "01B1902B1003B1004B10";

    if (bin == 2) s = "01B1002B1903B1004B10";

    if (bin == 3) s = "01B1902B1903B1004B10";

    if (bin == 4) s = "01B1002B1003B1904B10";

    if (bin == 5) s = "01B1902B1003B1904B10";

    if (bin == 6) s = "01B1002B1903B1904B10";

    if (bin == 7) s = "01B1902B1903B1904B10";

    if (bin == 8) s = "01B1002B1003B1004B19";

    if (bin == 9) s = "01B1902B1003B1004B19";

    if (bin == 10) s = "01B1002B1903B1004B19";

    if (bin == 11) s = "01B1902B1903B1004B19";

    if (bin == 12) s = "01B1002B1003B1904B19";

    if (bin == 13) s = "01B1902B1003B1904B19";

    if (bin == 14) s = "01B1002B1903B1904B19";

    if (bin == 15) s = "01B1902B1903B1904B19";


    return s;
}
```

```
        // função Lightining.print_Symbol() - exibe um simbolo no dispositivo
        // de valor correspondente a uma das letras: A, H, I, J, L, O, Q, U.

        static public string print_Symbol(char sym)
        {
            string s = "01B3002B3003B3004B30";

            if (sym == 'A') s = "01R1102R1603R1604R11";

            if (sym == 'H') s = "01G1702G1203G1204G17";

            if (sym == 'I') s = "01B1002B1003B1604B10";

            if (sym == 'J') s = "01P1202P1103P1004P16";

            if (sym == 'L') s = "01R1002R1003R1004R19";

            if (sym == 'O') s = "01B3602B3603B3604B36";

            if (sym == 'Q') s = "01W1602W1603W1604W10";

            if (sym == 'U') s = "01Y1602Y1003Y1004Y16";

            return s;

        }

    }

}
```

9 Apêndice 3

```
// Template C#
// Bibliotecas

using System;                    // contém classes fundamentais e classes base
using System.Threading;          // Usada para chamar a função sleep();
using System.Windows.Forms;      // permite o uso dos formulários e componentes
using LightForce;                // permite o uso das funções criadas

//criação de um formulário C#
namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        // variáveis globais

        string t, l, l1, l2, l3, l4;
        int A, B, C, D;
        char[] letra = new char['0'];
        string[] lr = {"0","0","0","0"};
        char[] comando = new char[5];

        // inicializa formulário e componente 'pauser'
        public Form1()
        {
            InitializeComponent();
            pauser.Enabled = true;
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            try
            {

                // abre a porta serial, função da componente 'port'
                Porta.Open();
            }
            catch
            {
                // se houver erro fecha a porta serial
                MessageBox.Show("Cabo desconectado", "Erro");
                Close();
            }
        }

        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            // quando termina a aplicação fecha a porta serial
            try
            {
                if (Porta.IsOpen)
                {
                    Porta.Close();
                }
            }
            catch
            { }
        }
```

```
// define a ação de um botão do formulário
private void button1_Click_3(object sender, EventArgs e)
{
    try
    {
        // Escreve 'L' no dispositivo chamando a função da biblioteca
        // e a função que envia o dado para o dispositivo
        t = Lightining.print_Symbol('L');
        enviarDado(sender, e);

      // aqui é onde o programador entra com seu código
      // podendo inserir mais botões ou mais funções no programa


    }
    catch
    {
        // se houver erro fecha a porta serial
        MessageBox.Show("Cabo desconectado", "Erro");
        Close();
    }
}


// a string comando recebe o código da aplicação e o envia ao dispositivo em
// partes de 5 dígitos, que correspondem a um pixel, depois disso dorme meio
// segundo, tempo suficiente para os dados não se sobreporem

private void enviarDado(object sender, EventArgs e)
{
    comando[0] = t[0];
    comando[1] = t[1];
    comando[2] = t[2];
    comando[3] = t[3];
    comando[4] = t[4];
    Porta.Write(comando, 0, 5); // função de escrita da componente port
    Thread.Sleep(500);

    comando[0] = t[5];
    comando[1] = t[6];
    comando[2] = t[7];
    comando[3] = t[8];
    comando[4] = t[9];
    Porta.Write(comando, 0, 5);
    Thread.Sleep(500);

    comando[0] = t[10];
    comando[1] = t[11];
    comando[2] = t[12];
    comando[3] = t[13];
    comando[4] = t[14];
    Porta.Write(comando, 0, 5);
    Thread.Sleep(500);

    comando[0] = t[15];
    comando[1] = t[16];
```

```
            comando[2] = t[17];
            comando[3] = t[18];
            comando[4] = t[19];
            Porta.Write(comando, 0, 5);
            Thread.Sleep(500);
        }


       // função Tick, da componente 'pauser'

        private void pauser_Tick(object sender, EventArgs e)
        {

            if (Porta.IsOpen == true)
            {
                Porta.Write("T");  // envia 'T' a Arduino

             l = Porta.ReadExisting(); // Arduino responde com os valores dos
             // potenciômetros

               lr = l.Split('\r');

             // quebra a string resposta em quatro, cada pedaço vai ser referente a
             //posição de um pixel

              if (lr.Length == 5)
                {
                    l1 = lr[0];
                    l2 = lr[1];
                    l3 = lr[2];
                    l4 = lr[3];

                }

             // converte as strings em inteiro para que os valores possam ser usados
             pelo programador na aplicação
                A = Convert.ToInt32(l1);
                B = Convert.ToInt32(l2);
                C = Convert.ToInt32(l3);
                D = Convert.ToInt32(l4);


            }
        }

    }
}
```

10 Apêndice 4

Marque um valor baixo para reação negativa e um valor alto para reação positiva:

Reação geral do sistema:

	1	2	3	4	5	
Frustrante						Satisfatório
Tedioso						Estimulante
Difícil						Fácil
Inadequado						Adequado

Medidas de eficácia:
Manuseio do dispositivo:
Difícil 1( ) 2( ) 3( ) 4( ) 5( ) Fácil

Reação ao toque:
Ruim 1( ) 2( ) 3( ) 4( ) 5( ) Boa

Resposta das aplicações:
Ruim 1( ) 2( ) 3( ) 4( ) 5( ) Boa

Precisão das respostas:
Difícil 1( ) 2( ) 3( ) 4( ) 5( ) Fácil

Medidas de eficiência:
Tempo de aprendizado para utilizar o dispositivo:
Lento 1( ) 2( ) 3( ) 4( ) 5( ) Rápido

Compreensão das letras, números e símbolos representados no dispositivo.
Difícil 1( ) 2( ) 3( ) 4( ) 5( ) Fácil

Intuitividade do dispositivo:
Ruim 1( ) 2( ) 3( ) 4( ) 5( ) Boa

Potencial para Upgrade:
Baixo 1( ) 2( ) 3( ) 4( ) 5( ) Alto

Medidas de satisfação:
Usaria o dispositivo no dia a dia ou no trabalho?
Não 1( ) 2( ) 3( ) 4( ) 5( ) Sim

Acredita no apelo ou proposta do dispositivo?
Não 1( ) 2( ) 3( ) 4( ) 5( ) Sim

Potencial para aplicações:
Baixo 1( ) 2( ) 3( ) 4( ) 5( ) Alto

11 Anexo 1

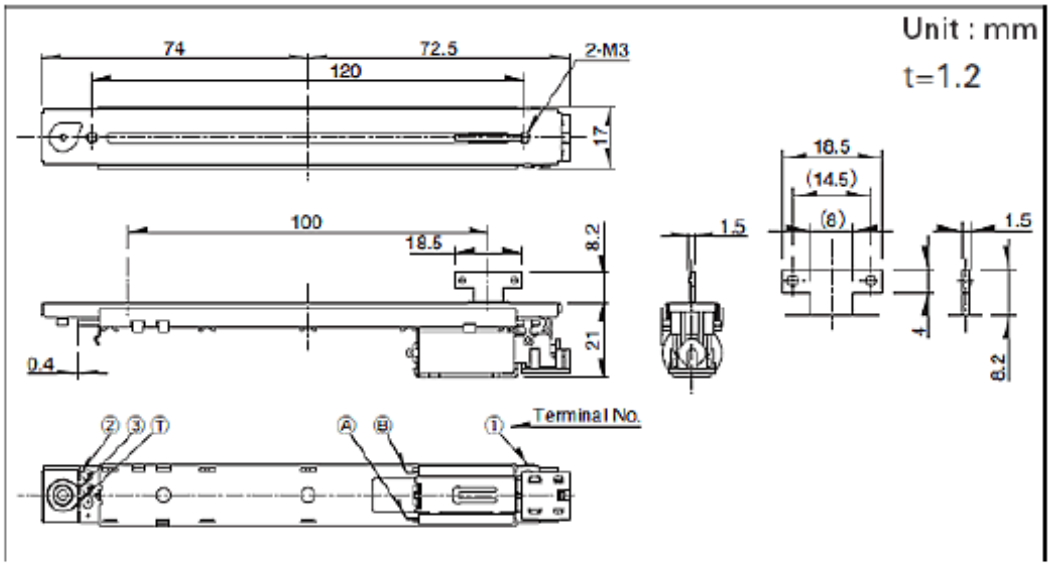
Perfecting the Art of Electronics
ALPS

Motor-driven Master Type (Motor N Fader, Motor K Fader, Motor V Fader)

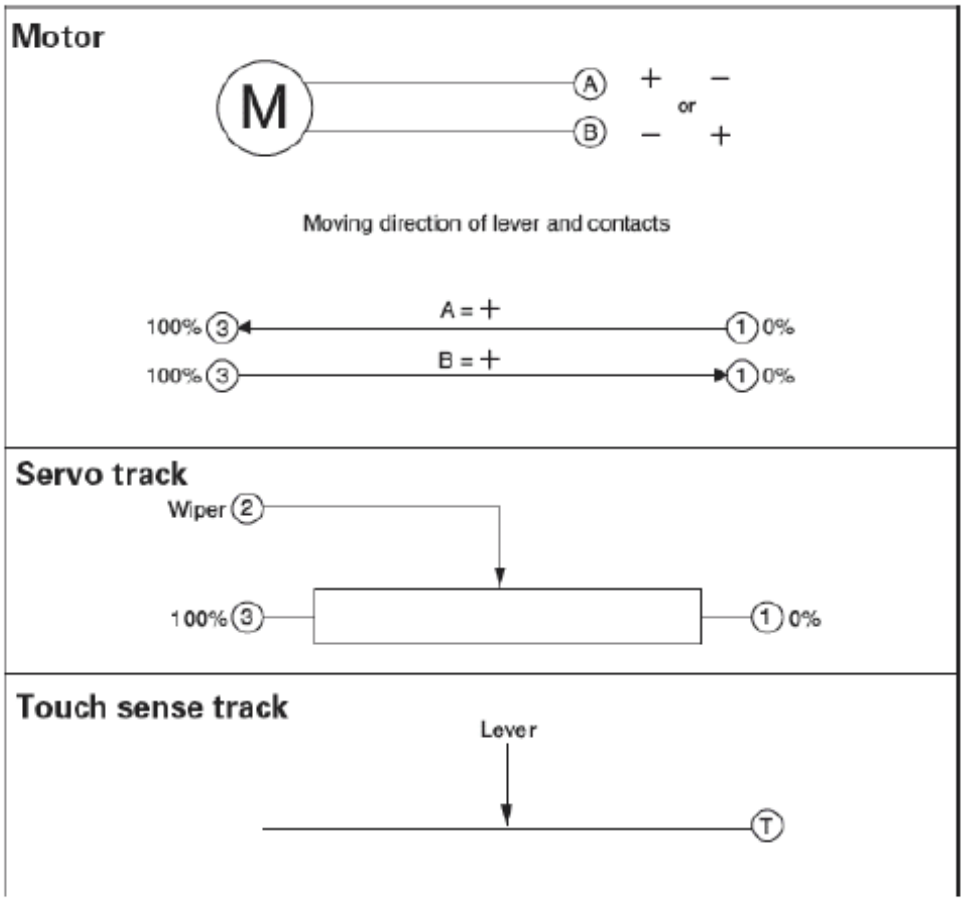
Part number		RSA0N11M9A0K
Number of resistor elements		Single-unit
Direction of lever		Vertical
Travel		100mm
Type		Motor N fader
Lever type		9-T (T-Bar)
Length of lever		8.2mm
Total resistance		10kΩ
Resistance taper		1B
Terminal style		Lead
Touch sense track		With
Operating temperature range		-10° C to +60° C
Electrical performance	Total resistance tolerance	±20%
	Maximum operating voltage	500V AC, 10V DC
	Rated voltage of motor	10V DC
	Maximum current of motor	800mA or less (at 10V DC)
	Rated power	0.5W
	Insulation resistance	100MΩ min. 250V DC
	Voltage proof	250V AC for 1 minute
Mechanical performance	Operating force	0.8±0.5N
	Stopper strength	100N
	Lever push-pull strength	50N
	Lever wobble (Both side)	1.312mm
	Lever deviation (One side)	0.5mm max.

Durability	Operating life	30,000 cycles
Minimum order unit (pcs.)	Japan	100
	Export	200

Dimensions



Circuit Diagram



Packing Specifications

Tray		
Number of packages (pcs.)	1 case / Japan	100
	1 case / export packing	200
Export package measurements (mm)		455 × 578 × 175

Soldering Conditions

Reference for Hand Soldering	
Tip temperature	350°C max.
Soldering time	3s max.
No. of solders	1 time

Notes are common to this series/models.

1. This site catalog shows only outline specifications. When using the products, please obtain formal specifications for supply.
2. Please place purchase orders per minimum order unit (integer).
3. Products other than those listed in above products are also available. Please contact us for details.
4. "L" in the "Lever Wobble" column of the above table indicates the length of lever.